Methodologies and Tools for the Evaluation of Transport Protocols in the Context of High-Speed Networks

Romaric Guillier

N° d'ordre: 537
N° attribué par la bibliothèque: _ENSL537

THÈSE

en vue d'obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE LYON - ÉCOLE NORMALE SUPÉRIEURE DE LYON
Spécialité : Informatique

LABORATOIRE DE L'INFORMATIQUE DU PARALLÉLISME
École Doctorale de Mathématiques et Informatique Fondamentale

présentée et soutenue publiquement le 29 Octobre 2009 par

Monsieur **Romaric GUILLIER**

Titre :

Méthodologies et outils pour l'évaluation des protocoles de transport dans le contexte des réseaux très haut débit.

Directeur de thèse :

Madame Pascale	VICAT-BLANC PRIMET	

Après avis de :

Monsieur Michel	DIAZ	
Monsieur Jean-Jacques	PANSIOT	

Devant la commission d'examen formée de :

Monsieur Michel	DIAZ	Membre/Rapporteur
Monsieur Serge	FDIDA	Membre
Monsieur Jean-Jacques	PANSIOT	Membre/Rapporteur
Madame Pascale	VICAT-BLANC PRIMET	Membre
Monsieur Thierry	PRIOL	Membre

Methodologies and Tools for the Evaluation of Transport Protocols in the Context of High-Speed Networks

Romaric Guillier

21 janvier 2010

Table des matières

Acknowledgments **xi**

Introduction **xvii**

1 Transport Protocols **1**
1.1 TCP functions . . . 2
1.1.1 Reliability . . . 2
1.1.2 Flow control . . . 4
1.1.3 Congestion Control . . . 4
1.1.4 TCP Properties . . . 6
1.2 TCP Limits . . . 7
1.2.1 Applications limits . . . 7
1.2.2 Technological limits . . . 7
1.3 Solutions . . . 9
1.3.1 Other layer modifications . . . 10
1.3.2 Transport Protocols other than TCP . . . 12
1.3.3 Congestion Control modifications . . . 14

2 Transport Protocol Evaluation Principles **25**
2.1 General principles of performance evaluation . . . 25
2.2 Points of view . . . 26
2.3 Metrics . . . 28
2.3.1 Categories of metrics . . . 28
2.3.2 Metrics and Points of View . . . 31
2.4 Network models . . . 32
2.4.1 Topology . . . 33
2.4.2 Network parameters . . . 34
2.4.3 Network Model and Point of Views . . . 35
2.5 Workload models . . . 36
2.5.1 Workload parameters . . . 37
2.5.2 Workload Model and Points of View . . . 38

3 Analysis of Transport Protocol Evaluation Methods **41**
3.1 Analytical Modeling . . . 42
3.1.1 Closed-form Modeling . . . 42
3.1.2 Stochastic Modeling . . . 44
3.1.3 Fluid Modeling . . . 45

3.1.4 Synthesis . . . 46
3.2 Simulation . . . 46
3.2.1 Packet-level Simulation . . . 47
3.2.2 Flow-level Simulation . . . 49
3.2.3 Synthesis . . . 50
3.3 Emulation . . . 50
3.3.1 Software Emulation . . . 51
3.3.2 Hardware Emulation . . . 52
3.3.3 Synthesis . . . 53
3.4 Experimentation . . . 54
3.4.1 Experimentation in Controlled Environments . . . 54
3.4.2 Experimentation in Uncontrolled Environments . . . 56
3.4.3 Synthesis . . . 58

4 Large-scale Experiments : motivations and tools 61
4.1 Motivation . . . 61
4.2 Transport Layer Evaluation . . . 63
4.2.1 Evolution of network infrastructure . . . 63
4.2.2 Evolution of Traffic Demand . . . 64
4.2.3 Scenario . . . 65
4.2.4 Results . . . 66
4.2.5 Synthesis . . . 67
4.3 Tool . . . 68
4.3.1 Architecture . . . 69
4.3.2 Related work . . . 73
4.4 NS vs. NXE . . . 75
4.4.1 Scenario . . . 75
4.4.2 Experiment Description Language . . . 76
4.4.3 Resulting Output . . . 76
4.4.4 Influence of the queue size . . . 77
4.4.5 Synthesis . . . 79
4.5 Protocol-Designer-Oriented Test Suite For the Evaluation of Transport Protocols 80
4.5.1 Network models . . . 82
4.5.2 Scenarios and Metrics . . . 82
4.5.3 Synthesis . . . 83

5 Benchmarking Transport Layer solutions 85
5.1 Benchmark and Test-suite Design . . . 86
5.1.1 Definition and goals . . . 86
5.1.2 Guidelines for transport protocol benchmarking . . . 86
5.2 User-Oriented Test-suite . . . 87
5.2.1 Metrics . . . 87
5.2.2 Network model . . . 88
5.2.3 Test suite design . . . 88
5.2.4 Synthesis . . . 91
5.3 Bulk data transfers experiments . . . 91
5.3.1 BU test results . . . 92
5.3.2 Transfer predictability . . . 92
5.3.3 Influence of numerous parallel streams . . . 96

6 Exploration of End-to-End Path Characteristics **101**
6.1 State of the art . . . 102
6.1.1 Networking Diagnostic Tools . . . 102
6.1.2 Autotuning . . . 103
6.1.3 Synthesis . . . 104
6.2 Proposal . . . 104
6.2.1 Workflow . . . 106
6.2.2 Detailed presentation of each stage . . . 106
6.3 Hardware Config Detection . . . 107
6.3.1 Bottlenecks . . . 107
6.3.2 Formula . . . 107
6.4 Software Config Detection . . . 110
6.4.1 Bottlenecks . . . 110
6.4.2 Formula . . . 111
6.5 Network Characteristics Detection . . . 112
6.5.1 Enhancements . . . 112
6.6 Case study . . . 113

Conclusion and Perspectives **115**

Table des figures

1.1 "Three-way handshake" used to establish a TCP connection between two end-hosts. . . . 3
1.2 Evolution of the size of the congestion window during the Slow Start and the Congestion Avoidance phases. . . . 5
1.3 Congestion window size evolution for two independent TCP connections, on a path with a fixed RTT of t (resp. 2t) . . . 6
1.4 The existing hardware bottlenecks encountered in an end-host hardware . . . 8
1.5 Communication through a network stack . . . 10
1.6 Tree of the different alternatives to scale up in performance in high-speed networks 10
1.7 Chronology of Transport Protocol developments . . . 11
1.8 Congestion Window size evolution pattern for the presented Slow Start modifications . . . 14
1.9 Tree of the different solutions of the Congestion Avoidance modifications . . . 16
1.10 Congestion window evolution pattern for presented Loss-based TCP variants . . 16

2.1 Interaction of the workload model, network model and metrics in an evaluation process . . . 26
2.2 Type of actors and their relation with the Internet Reference Model . . . 28
2.3 Concentric network borders . . . 33
2.4 Typical topologies for network models . . . 34
2.5 Network parameters . . . 35
2.6 Three different visions for the topology of a end-to-end path . . . 37

3.1 Comparison of the response functions for TCP variants . . . 44
3.2 Grid'5000 backbone network . . . 55

4.1 Impact of the multiplexing factor on the completion time . . . 64
4.2 Timeline for the Congestion Collapse in Grid'5000 experiment . . . 65
4.3 Experiment topology . . . 66
4.4 Goodput of the reference heavy flow between the Sophia and Rennes sites (18 ms RTT) during the execution of the Congestion Collapse in Grid'5000 scenario . . 67
4.5 Workflow of NXE . . . 70
4.6 Screenshot of the NXE GUI, resource selection part . . . 71
4.7 Description of a simple scenario to compare the NS-2 and NXE approaches . . 75
4.8 Goodput measurement for two flows simulated using NS-2 . . . 78
4.9 Goodput measurement for two flows obtained through a real experiment with NXE . . . 78

5.1 Dumbbell topology used for the experiments 87
5.2 Impact of the TCP behavior on the MPI traffic 90
5.3 Bulk data transfer test results between Rennes and Toulouse clusters 92
5.4 Scenario for the completion time predictability experiment 92
5.5 Impact of congestion level on the mean completion time for some TCP variants 93
5.6 Completion time distribution for several TCP variants, 19.8 ms RTT 94
5.7 Evolution of the completion time coefficient of variation as a function of the congestion level for several TCP variants 95
5.8 Impact of reverse traffic level on mean completion time for CUBIC, 19.8 ms RTT 96
5.9 Scenario description for the evaluation of multiple parallel streams experiment . 97
5.10 Comparison between the analytical model and experimental measurements for the multiple parallel streams experiment 98

6.1 End host chain 105
6.2 Workflow of PathNIF to detect hardware, software and network bottlenecks . . 105
6.3 The existing hardware bottlenecks encountered in an end-host hardware 107
6.4 Simple model for transmitting/receiving packets in a GNU/Linux stack 111

Liste des tableaux

1.I The different layers of the Internet Reference Model and their function 1
1.II The most important functions of TCP 2
1.III AIMD "constants" used by some TCP variants for LFN networks 18
1.IV Summary of known TCP limits and existing solutions 22
1.V Summary of main ideas used by solutions to TCP defiency problems and their limit 22
1.VI Summary of the Congestion Avoidance technique used by the presented Transport Protocol variants 24

2.I Steps for a systematic approach to performance evaluation 26
2.II Examples of metrics and their formulation 31
2.III Three different visions for metrics 32
2.IV Parameters defining the network model 36
2.V Factors defining the workload model 38

3.I Parameters of the response function for some TCP variants, $R = \frac{MSS}{RTT} * \frac{C}{p^d}$ . . . 43
3.II Usefulness of a Evaluation Method for a given type of actor 58
3.III Advantages and limits of different evaluation methods 60

4.I Evolution of the capacities of local area, access, and core networks 63
4.II Impact of the choice of the α parameter on the mean and the variance of a Pareto distribution 65
4.III Relative evolution of the goodput during the successive time periods of the Congestion Collapse in Grid'5000 scenario 67
4.IV Comparison of the different characteristics of existing tools to perform large-scale experiments 75
4.V Decomposition of the execution time of the scenario with NS-2 and NXE 77
4.VI Existing methodologies to analyse the behaviour of transport protocols 81
4.VIINetwork models used in the TMRG test suite 82

5.I Average goodput matrix for a single not-tuned TCP flow between two 1 Gbps nodes on different Grid'5000 sites 89
5.II Average goodput matrix for a single tuned TCP flow between two 1 Gbps nodes on different Grid'5000 sites 89
5.III Results of the evaluation of the multiple parallel streams experiment 97
5.IV Impact of the competition of 10 nodes, each emitting 10 streams on a node using an inferior number of streams 98

6.I Input Parameters 106
6.II Retrieved Network Configuration Parameters 112
6.III Case study results 113

« Peeple of zee wurl, relax ! »

— Fierce Invalids Home From Hot Climates, *Tom Robbins*

Remerciements

- Pascale pour m'avoir permis de travailler sur cette thèse.
- Vincent, Martin, Franck et Andrzej du Drakkar qui m'ont donné envie de faire de la recherche.
- les gens du bureau 316 Centre (Sébastien, Ludovic, Dinil, Marcelo, Patrick) et tous les autres doctorants/chercheurs de l'équipe RESO et du LIP.
- Aurélien et Matthieu pour les coups de main pour faire fonctionner mes expériences sur Grid'5000.
- les copinous de la Mitose Mentale pour avoir supporter mes devinettes et mes photos de tourtes pendant des années.
- les copains de l'ENSIMAG.
- Laure et tous ceux qui ont des mots pour moi.

Résumé

Mots-clefs

Protocoles de Transport, Contrôle de congestion, Évaluation de Performance, Benchmark de protocoles, Expérimentation large échelle, Transfert déterministe, Réseaux très haut-débit.

Méthodologies d'évaluation de performance des protocoles de transport

Au cours des vingt dernières années, c'est une évolution très rapide non seulement en termes de nombre d'éléments inter-connectés mais surtout en termes de capacités des liens qui a pu être constatée au sein des réseaux de communications. A la création du réseau ARPANET en 1969, il y a plus de quarante ans, quatre ordinateurs, tous situés aux États-Unis étaient connectés avec des liens avec un débit de 50 kilobits par seconde. En 1986, à la création du réseau des universités américaines, le NSFNET, le nombre de machines dépassait déjà plusieurs milliers mais le débit des liens de coeur était encore de 56 kilobits par seconde. A partir de ce moment, la croissance à la fois en termes de nombre d'éléments inter-connectés et de capacités des liens a été exponentielle, grâce au succès de nouvelles applications populaires comme le WWW ou les avancées technologiques comme SDH, SONET ou DWDM dans le domaine des réseaux de fibres optiques et ATM et les protocoles DSL qui en dérivent. L'Internet actuel interconnecte des milliards d'utilisateurs (un demi-milliard de machines fin 2006) répartis partout dans le monde, pouvant utiliser des liens de coeur dont les débits cumulés sont supérieurs à plusieurs Térabits par seconde et dont les liens d'accès peuvent atteindre jusqu'à 10 Gigabits par seconde.

Une autre évolution technologique fondamentale est que, aujourd'hui, certains liens d'accès ont des débits du même ordre de grandeur que les débits des liens de coeur auxquels ils sont connectés. Il est aujourd'hui courant de trouver des fermes de serveurs où les noeuds sont interconnectés avec des liens à 1 Gbps ou 10 Gbps et qui ont une connexion vers l'extérieur avec un débit similaire. Cette modification est importante car désormais quelques connexions suffisent à provoquer une saturation des liens de coeur. Cette situation risque d'empirer au fur et à mesure du remplacement des technologies DSL par la Fibre-Jusqu'à-La-Maison (FTTH). Des connexions à 5 Mbps agrégées sur des liens OC48 (2.5 Gbps) vont alors être remplacées par des connexions à 100 Mbps, voire 1 Gbps, agrégées sur des liens à 10 Gbps, voire moins suivant la technologie optique utilisée sur le réseau d'accès. Il est fort probable que dans de tels environnements des niveaux de congestion élevés ne soient pas rares, engendrant des perturbations pour de nombreuses applications.

Un grand nombre d'applications prometteuses ont émergé et pourraient bénéficier de capacités de communication accrues, comme par exemple la Vidéo-à-la-Demande (VoD) qui permet aux utilisateurs de regarder des films en haute définition chez eux quand ils le désirent ou les

expériences scientifiques générant des quantités massives de données (comme le LHC) qui doivent être acheminées dans des centres de calcul répartis tout autour du globe. L'interaction d'utilisateurs avec des objets distants via un réseau de communication ou Téléprésence commence également à prendre un rôle prépondérant dans de nombreux domaines allant du médical au militaire en passant par l'exploration spatiale. Mais ces applications sont encore construites au dessus de logiciels ou de protocoles à succès comme TCP et IP, conçus il y a longtemps pour un contexte de réseaux bas-débit. Le conception était dirigé par un but principal : fournir une interconnexion robuste entre des réseaux existants. Plongés maintenant dans un contexte de réseaux très haut-débit, ces protocoles extrêmement robustes et capables de passer à l'échelle sont toujours en activité mais commencent à montrer leurs limites en termes de performance, sécurité, fonctionnalités et flexibilité.

L'exemple le plus typique est TCP au niveau de la couche de transport. TCP a été conçu comme un protocole "à tout faire" mais à mesure que la capacité des liens augmente dans les réseaux fixes, que les caractéristiques intrinsèques changent pour les réseaux sans fils et que les besoins des applications divergent, TCP n'est plus capable de permettre une utilisation performante pour tous des capacités nominales des liens du réseau.

De nombreuses solutions ont été proposées, mais les concepteurs de protocoles d'aujourd'hui ont un large désavantage par rapport à ceux d'il y a quarante ans. Faire fonctionner ensemble quatre machines identiques n'est pas exactement pareil que concevoir un logiciel de communication ou protocole susceptible d'être utilisé entre des milliards de machines hétérogènes. Apporter du changement dans un système d'une telle ampleur est problématique : il n'est pas possible d'envisager une migration massive du jour au lendemain surtout si des équipements physiques sont concernés. Le changement ne peut être que graduel et via des mises à jour logicielles dans les hôtes d'extrémité.

La peur d'un effondrement de l'Internet dû à la congestion comme à la fin des années 80 resurgit. Causés par une surabondance de trafic réseau sur des liens saturés, ces troubles ont mis en avant la nécessité d'introduire un mécanisme de contrôle de congestion dans les protocoles de transport. Si les nouvelles solutions introduites se comportent de façon égoïste quand elles partagent une ressource commune (un lien), alors cette peur est fondée. Il est donc nécessaire de fournir des moyens de vérifier si une solution donnée possède les propriétés adéquates avant de l'introduire dans un système partagé à très large échelle.

Notre première contribution a donc consisté à définir des scénarios reproductibles pour définir une méthodologie "standard" d'évaluation des protocoles de transport ciblés pour les réseaux très haut débit [Andrew et al., 2008]. Une analyse des différentes méthodologies d'évaluation existantes et de leurs avantages respectifs a été réalisée au préalable. Cette démarche a débouché sur la proposition d'un banc d'essai "orienté utilisateur" à destination des transferts de masse [Guillier & Vicat-Blanc Primet, 2009a].

Outil de réalisation d'expériences réseaux à grande échelle sur réseaux très haut débit

Une fois que l'on dispose d'une méthodologie d'évaluation, il est alors possible de l'instancier dans différents environnements (réels ou simulés) pour confronter les protocoles à différentes conditions réseaux (à la fois structurelles et temporelles).

Du point de vue de l'utilisateur final, l'expérimentation réelle est le meilleur moyen de déterminer si un protocole de transport a les caractéristiques souhaitées dans les conditions de fonctionnement réseau de l'utilisateur. Mais ce type d'expérimentations est difficile et coûteux (en temps) à mettre en place. Il est donc nécessaire de disposer d'un outil permettant de diriger facilement ce genre d'expériences. J'ai conçu et développé un contrôleur d'expérience large échelle d'évaluation de protocoles réseaux pour répondre à ces attentes. Il fournit un cadre simple pour configurer les hôtes d'extrémité, pour contrôler les différentes interactions

entre les différentes machines en utilisant un calendrier défini par l'utilisateur et en analysant les résultats expérimentaux obtenus.

Cet outil [Guillier & Vicat-Blanc Primet, 2008b, Guillier & Vicat-Blanc Primet, 2009a] a été utilisé pour un grand nombre d'expériences [Soudan et al., 2007a, Guillier et al., 2007a, Guillier et al., 2007b, Guillier & Vicat-Blanc Primet, 2008a, Guillier et al., 2009] sur la plateforme expérimentale française Grid'5000. Il est également utilisé conjointement avec l'infrastructure de mesure à très haut débit MetroFlux [Loiseau et al., 2009a] pour activer et synchroniser automatiquement de multiples sources de trafic afin de réaliser des expériences contrôlées d'analyse de trafic Internet ou Grille.

Outil d'évaluation des capacités intrinsèques de communication entre deux hôtes d'extrémité

La dernière conséquence de l'existence d'une vaste hétérogénéité de matériels et de logiciels déployés dans le monde entier est qu'il est maintenant difficile de déterminer une configuration par défaut qui serait optimale pour tous. Il est également difficile de conclure sur la viabilité d'une solution alternative.

Enfin, partant du constat que la recherche de performance n'a de sens que si les hôtes d'extrémité impliqués sont correctement configurés, nous proposons un outil qui analyse toute la chaîne matérielle, logicielle et réseau existant sur un chemin entre deux hôtes d'extrémité afin de déterminer les goulots d'étranglements et de fournir les procédures permettant de les résoudre [Guillier & Vicat-Blanc Primet, 2009c].

Introduction

Keywords

High-speed Transport Protocols, Performance Evaluation, End-to-End, Protocol Benchmark, Congestion Control, Large-Scale Experimentation, Deterministic Transfer, High-Speed Networks.

Context

Over the last twenty years, there has been a large increase both in terms of number of interconnected hosts and in terms of link capacities. At the beginning of the ARPANET [Froehlich & Kent, 1991] in 1969, more than forty years ago, there were four nodes, all located in the USA and connected with a link speed of 50 kilobits per second. In 1986, when the NSFNET, the network of the US universities, was created, the number of hosts was over a few thousands, but the backbone links' speeds were still of 56 kilobits per second. From that moment on, the growth in number of interconnected hosts and in link capacity has been exponential, thanks to the success of popular applications like the WWW [Gillies & Cailliau, 2000, Berners-Lee, 1999] and technological advances like SDH, SONET or DWDM in the fiber optics networks domain or like ATM and the DSL technologies derived from it. The current-day Internet interconnects billions of users (half a billion hosts in late 2006) all around the world, has a cumulated core links speed over several Terabits per second and access links up to 10 Gbps [Zakon, 2006].

Another fundamental technological change is also that nowadays, access links start having speeds in the same order of magnitude as the backbone link's speed. For instance, it is common to find data centers where the nodes are interconnected with 1 Gbps or 10 Gbps links and have a connection to the outside with a similar speed. This change is important as only a few connections would be necessary to cause saturation in the backbone. This situation is likely to spread as DSL technologies are gradually replaced by Fiber-To-The-Home (FTTH). 5 Mbps connections aggregated on OC48 (2.5 Gbps) links will then be replaced by 100 Mbps to 1 Gbps links aggregated over 10 Gbps links or less depending on the actual optical technology used in the access network. In such environments, high congestion levels might not be rare and might cause disruption to high-end applications.

Lots of promising applications have emerged and would benefit from this increase in capacity like Video On Demand (VoD) that allows end-users to host their own high-definition movie entertainment whenever they want or scientific experiments that gather huge volumes of data (*e.g.*, LHC) that need to be transported to computational facilities scattered all around the globe. The interaction of users with remote objects through the use of communication networks, also known as Telepresence, starts to have a paramount role in a variety of domains

ranging from medical or military applications to space exploration. But these applications are still built on top of the very successful softwares or protocols like TCP and IP that were designed a long time ago for a context of low-speed networks. Their design [Clark, 1988] was dictated by one goal : providing a robust and fair interconnection between existing networks. Now put in a high-speed networks context, these extremely robust and scalable protocols are still able to deliver but they are starting to show their limits in terms of performance, security, functionalities and flexibility.

The most typical example is the quasi "one-fits-all" TCP protocol at the transport layer. This protocol was designed to provide a connection-oriented "virtual circuit" service for applications like remote login or file transfer. These two applications have two nearly conflicting sets of requirements : the first one needs a low delay and has low requirements for bandwidth while the second one mostly requires high bandwidth. Both require reliable transmissions. TCP was designed as a "one-size-fits-all" protocol but as the capacity of the links increased in the wired networks, as the intrinsic characteristics changed for the wireless networks and as the application requirements diverged, TCP is no longer able to scale up in performance.

Multiple solutions have been proposed over time, but today's protocol designers have a large disadvantage over those forty years ago : making four similar computers interact is not exactly the same thing as designing a communicating software or protocol fit to be used by billions of vastly heterogeneous computers [Floyd & Paxson, 2001]. Introducing change in a system of such a scale is problematic : it makes it very improbable to make a complete overnight migration of every element in the network, especially if it requires an upgrade of all physical equipments. Thus the change can only be gradual and as software updates in the end-hosts at the very edge of the networks, as it would be easier than changing core hardware elements [Floyd & Fall, 1999, Saltzer et al., 1984].

The fear to see congestion collapse events in the Internet as in the late 80s is coming back. Caused by too much traffic on crowded links leading to poor performance for everybody over long periods of time, it had advocated for the introduction of a congestion control mechanism in the transport protocol. If the new solutions are behaving selfishly when sharing the common resources (the links), this fear is grounded [Sherwood et al., 2005]. It is then necessary to provide ways to assert if a given solution has the appropriate properties before allowing it to roam freely over a large-scale shared system.

The last consequence of the vast heterogeneity of the hardware and software deployed all around the world is that it is now difficult to provide optimal default configuration parameters for everyone [Mathis et al., 2003] and difficult to conclude on the viability of any alternative solution.

Objectives

The goal of this thesis is twofold :

1. contribute to methodologies for the performance evaluation of transport protocols in the context of large-scale high-speed networks, especially by providing a common ground (and in a spirit of cooperation with the TMRG work group) for the comparison of alternative transport protocols. Indeed, over the last decade a gogolplex of TCP variants have been proposed but little has been made to compare them in a square and sound way. This can only lead to confusion for the users (protocol designers do not have the same objectives as end-users or network providers), especially as different variants start to be easily available (selection by changing a single parameter) in current operating systems (Windows Vista, GNU/Linux).
2. The second goal, taking end-user needs as a first concern, is to put forward tools to help analyze the behavior of given transport solutions in a real technological and usage context. What better way to identify it than by trying them out in the exact same

environment where they are planned to be deployed ? This also points out the necessity for having a tool to locate the existing bottlenecks interfering with them and if possible to solve them, to achieve the best possible performance.

Contributions

- A synthesis of transport solutions dedicated to high-speed networks and an analysis of how they have been evaluated.
- A tool [Guillier & Vicat-Blanc Primet, 2009b] designed to automate the activation of very large sets of customized traffic sources [Loiseau et al., 2009a] and to pilot large-scale experiments in high-speed networks [Guillier & Vicat-Blanc Primet, 2008b].
- Contributions to the international effort for a standardized test-suite for the evaluation of transport protocols that can be used both in simulation and in real experimentation [Andrew et al., 2008].
- A vast body of experiments using French research grid Grid'5000 to explore the behavior of high speed transport protocols in real networks. This allowed to show interesting results such as the impact of congestion on file-transfer completion time [Guillier et al., 2007b], the need to properly tune end-hosts in high-speed environments and how we solve the wizard gap in Grid'5000 nodes [Guillier et al., 2006] or check via real experiments the impact of parallel streams on the efficiency of a transfer [Guillier et al., 2007a].
- The proposal of a end-user-oriented transport protocol benchmark dedicated to bulk data transfers [Guillier & Vicat-Blanc Primet, 2009a].
- A tool to help users detect the bottlenecks existing between two end-hosts [Guillier & Vicat-Blanc Primet, 2009c, Guillier & Vicat-Blanc Primet, 2009d].

Structure

This dissertation is organized as follows. Chapter 1 introduces the role of the transport layer in any networking stack and focuses on the example of TCP, the most widely deployed transport protocol (80% to 95% of the Internet traffic). It shows that a large number of solutions (more than thirty) have been proposed to try to overcome some of its performance downfalls, especially in the context of high-speed networks, but that vast amount of solutions complicate the choice of the one most suited to a particular need for a particular user.

Chapter 2 presents the general principles behind the process of performance evaluation of a transport protocol. It then identifies three different entities interested by the performance evaluation : the network provider, the protocol designer and the end-user. It then analyzes how the metrics (instantiation of what question the user is trying to answer to), the topologies (instantiation of a E2E path over a networking infrastructure) and the scenarios (instantiation of the network's workload) used relate to these three different entities.

Chapter 3 discusses the various proposals to model, simulate, emulate, or experimentally evaluate the performance of transport protocols, showing their respective advantages and limits. It is then possible to identify which technique is more adapted to a specific type of actor or study. For instance, experimentation in uncontrolled environments is one of the only evaluation methods that is practical for end-users.

Chapter 4 motivates the need for end-users to perform their experiments at a large scale, due to current changes in the Internet in terms of applications and in terms of capacity. Examples of large-scale experiments are provided. A methodology and tool developed to conduct the execution of such large-scale experiments in a real networking environment is presented. It provides the workflow to configure end-hosts involved, to pilot their different interactions according to a user-defined timeline and to analyze the produced experimental results. A comparison to NS-2's approach is made to highlight the particular advantages of such

a method to evaluate the performance of transport protocols. Finally, we present the current international effort to produce a standardized test-suite dedicated for protocol designers to evaluate transport protocols.

Chapter 5 presents the general principles of benchmarking and designing a test-suite. It is then illustrate through our proposal of a test-suite dedicated to end-users to allow the comparison of high-speed transport protocols against typical applications. The specific example of bulk data transfers is detailed to show the impact of different workload parameters : the congestion level, the reverse traffic level and the number of parallel streams.

Finally, Chapter 6 briefly presents a method (currently under patenting process) and tool designed to clear the brushwood on a end-to-end path. It finely analyzes the communication chain between two end-hosts so as to detect the existing bottlenecks and to provide procedures to solve them.

— One —

Transport Protocols

Introduction

The International Standardization Office (ISO) started to work in 1977[1] on a framework model [ISO/IEC, 1996] to define a standard on how to interconnect open and heterogeneous systems and allow the transfer of information between them. The main issue to solve being the management of a very high complexity. The proposed layered model permits to insulate the different functions (*e.g.*, binary transmission, addressing, routing, etc) needed to provide a complete network stack and to limit the interactions between each layer. Indeed, a layer at the N^{th} level should only require to work with the $(N-1)^{th}$ and the $(N+1)^{th}$ levels. Data transmission between successive layers is performed by encapsulation. Each layer adds a new level of header containing the control information needed and processed by the current layer.

The OSI Model has inspired the Internet Reference Model [Postel, 1981a, Braden, 1989] that defines the four layers that interact to enable the communication of computers over a network, based on the engineering model of the TCP/IP protocols. Table 1.I summarizes the different layers of the Internet Reference Model and their respective functions.

These models are frameworks to help the engineering of protocols but it can also be seen as an optimization decomposition [Chiang et al., 2007]. There are a few cases where the need for a cross-layering approach was deemed necessary, for instance when information available at lower layers (*e.g.*, MAC layer in Wireless networks) could be used to optimize the behavior of the upper layers (*e.g.*, Transport Protocol layer in Wireless networks). Multiplying the number of layers allow to have more structured components and to share more common parts, but it increases at the same time the overhead of the communication, the complexity of the

[1] Final version in 1996

Type	Layer	Function	Examples
Host layers	Application	Process-to-process communication	FTP, HTTP, SMTP, SSL Sockets, SIP
	Transport	Host-to-host communication	TCP, UDP, DCCP, SCTP
Media layers	Internet	Routing and addressing	IP, ICMP
	Link	Local link transport	{10Mbps, Fast}Ethernet {1Gb, 10Gb}Ethernet ATM, SONET, PPP

Tab. 1.I – The different layers of the Internet Reference Model and their function

Name	Function
Reliability	ensure that data makes it in its entirety, without corruption, to the other side
Flow control	avoid overrunning the destination
Congestion Control	avoid sending over the available capacity of the network

TAB. 1.II – The most important functions of TCP

network stack and the difficulty to optimize each layer of the full stack [Bush & Meyer, 2002]. It is difficult or unrealistic to map one model to the other as, for instance, TCP also handles sessions and thus would overlap on OSI's Session Layer. But these models highlight the importance of the end-to-end transport protocol layer that handles the communication between end-hosts [Iren et al., 1999].

The transport layer is a critical layer at the interface between the user and the network which has a end-to-end vision of the network . It is where important functions like flow control (avoid overrunning the destination), reliable transfer (ensuring that data makes it in its entirety, without corruption, to the other side) and congestion control (avoid sending over the available capacity of the network) are performed. This cohabitation of functions related to the user (reliability) and functions related to the network (congestion control) is a potential point of issue as problems are likely to accumulate in a very changing technological context. These functions are illustrated in the following section through the example of TCP, the most popular transport protocol, responsible for 80 to 95 % in bytes of the total Internet traffic [John et al., 2008, Fomenkov et al., 2003, Pang et al., 2005, Pang et al., 2004].

The rest of this chapter is organized as follows : Section 1.1 presents the vital functions of Reliability, Flow Control and Congestion Control that are performed by TCP in today's networks. Section 1.2 highlights some of the known limits of TCP that are caused by changes both in terms of applications and in terms of technological evolution. The following Section 1.3 will then present a range of solutions that were introduced to alleviate these limits.

1.1 TCP FUNCTIONS

This section describes the vital functions that are performed by TCP as a transport protocol to allow a reliable, error-free, and rate-controlled connection between two end-hosts. Table 1.II provides a summary of the most important functions of TCP.

1.1.1 RELIABILITY

The messages (or packets) sent by TCP [Postel, 1981b] are made of two parts : a "payload" part that contains the data the end-user wants to transfer and a "control" part (or header) that contains the information needed to properly transport the data (*e.g.*, source and destination ports, sequence number, acknowledgment number, advertised window, checksum, options).

The reliability function is designed to ensure that packets sent from one end-host reaches the other side with guaranteed data integrity and without any loss of information. It is necessary as TCP is usually used over IP whose datagram's transmission is not reliable.

Figure 1.1 presents the mechanism that allows to synchronize two end-hosts when opening a new TCP connection. The end-host that initiates the connection is called the client, the other one is the server. The client and the server will exchange a number (32 bits integer from the TCP header), called sequence number, which will be used to count the number of bytes that were sent by the client. It is used to guarantee data delivery to the destination.

This mechanism is called "three-way handshake" and is set in three stages :

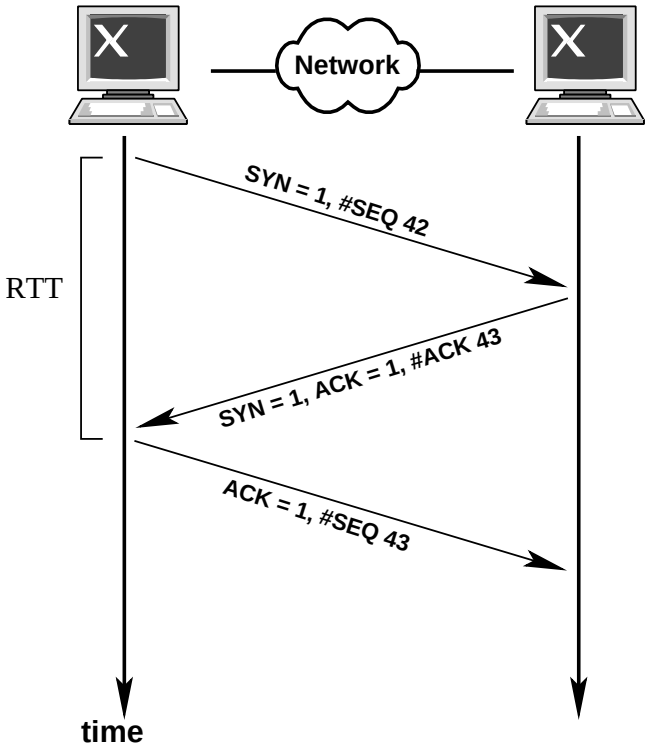


FIG. 1.1 – "Three-way handshake" used to establish a TCP connection between two end-hosts.

1. The client sends a synchronization packet (flag SYN set to 1). The sequence number is set to a random value x.
2. The server receives this packet, records the x sequence number from the client and answers with an acknowledgement and synchronization packet (SYN-ACK). It contains the next sequence number expected by the server $x + 1$. The server also starts a session by sending its own initial sequence number y.
3. The client answers with a packet containing its next sequence number $(x + 1)$ and acknowledges the server sequence number by sending back $(y + 1)$

After this step, it is then possible to send data in both directions full duplex. To have a guaranteed delivery, the receiver is expected to send a positive acknowledgment (ACK) for each byte sent. To reduce the number of ACKs to be sent, they are cumulative : they validate the whole byte sequence sent till their acknowledgment number. The checksum in the packet's headers allows to check the integrity of data sent. If there is a mismatch, the packet is discarded by the receiver and it will wait for a correct retransmission to proceed.

If a byte sequence is lost during the transmission due to a packet integrity error detected through the checksum or due to a networking equipment dropping it during a congestion event (it typically occurs in a switch when the sum of all the input ports' bandwidths exceeds the capacity of the output ports' bandwidth and buffering capacity during a time slot), it is detected on the receiver's side when a discontinuous byte sequence arrives. The receiver then resends the last ACK packet previously sent. When the sender receives several of these duplicated acknowledgments (dupACKs), it retransmits the lost packets.

A timer is also used to detect the lost packets. If the corresponding ACK for a byte sequence is missing for a time longer than RTO, a value computed from the history of the Return-Trip Time (RTT) of a packet between the two end-hosts, the corresponding packet is retransmitted.

All these actions (checksum and positive acknowledgment handling) provide a reliable data delivery for all the applications using TCP as their transport protocol.

1.1.2 Flow control

The flow-control function is designed to make sure that the sender does not send more than what the receiver is capable of handling. This kind of problem can occur when the receiving application is not fast enough to drain the TCP's receive buffer in time. It can be due to hardware limitations (*e.g.*, a very CPU-intensive computation on every piece of data received) or just to the application not performing its *read()*s often enough (*e.g.*, waiting for user input).

In TCP, flow control is performed through the field "advertised window" in the packet's TCP header. The receiver sets it to the maximum amount of un-acknowledged data the sender can send at any given time. The initial value chosen is usually based on the amount of buffering available on the receiver's side.

1.1.3 Congestion Control

As early as October 1986, Internet users noticed some persisting performance anomalies during which the applications' bandwidth was dramatically reduced. This phenomena was caused by a number of users that was too large for the network's capacity (causing congestion) and a badly controlled scheme for the retransmissions of the packets lost due to this congestion.

John Nagle [Nagle, 1984] foretold this kind of anomalies in 1984 and named it "congestion collapse". One example is the case of the link between the Lawrence Berkeley Laboratory (LBL) and the Berkeley California University only separated by a distance of 400 meters whose real bandwidth plummeted from 32 Kbps to 40 bps.

To solve this problem, Van Jacobson introduced in 1988 [Jacobson, 1988] in the TCP protocol a combination of distributed algorithms to handle the network congestion, creating the TCP Tahoe version. It was then specified by the IETF as a standard [Allman et al., 1999]. The main idea was to use a robust mechanism able to efficiently and quickly detect congestion signals. As, by design, an IP network does not generate any explicit congestion signal, the detection of lost packets started to be interpreted on the sender's side as a sign of saturation of network equipments on the end-to-end path of the TCP connection. These losses are detected on the basis of a timeout (RTO) computed from the constantly evaluated RTT or the reception of duplicate ACKs.

To avoid the fast saturation of a path, every new TCP connection gradually increases its sending rate following the Slow Start algorithm.

1.1.3.1 Slow Start

During this phase, the sender aims at finding out the maximum number of packets it can send before generating a loss in the network. A variable, called *cwnd*, corresponds to the current maximum amount of packets (or congestion window) that can be sent at given moment. *cwnd* is raised every time an ACK is received.

$$\textbf{ACK}: \quad cwnd \leftarrow cwnd + 1 \tag{1.1}$$

Despite its name, this phase is rather aggressive as the number of packets sent is at most doubled every RTTs. The left side of Figure 1.2 illustrates the evolution of the size of the congestion window during the Slow Start phase. This exponential growth is however limited by a threshold (*ssthreshold*), after which the congestion window's evolution is ruled by the congestion avoidance algorithm.

If a packet is lost before going over this threshold, the congestion window size and the ssthreshold are set to half the current congestion window size and a phase of Congestion Avoidance starts. A loss too early in the Slow Start phase can harshly handicap a flow as it will begin the next phase with a congestion window size further away from the maximum capacity of the network and it also can have long-term consequences, as the threshold is also smaller.

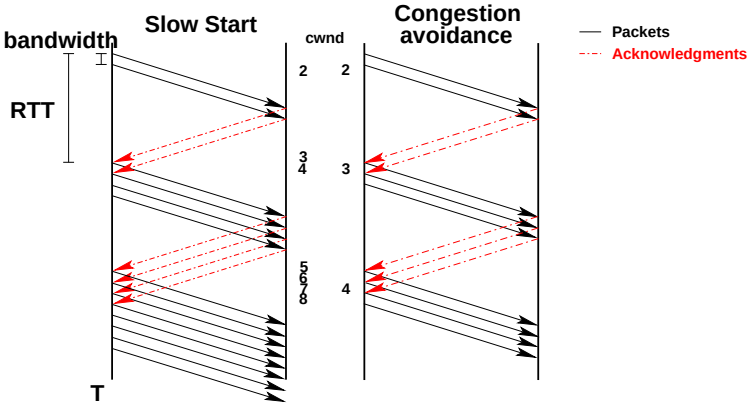


FIG. 1.2 – Evolution of the size of the congestion window during the Slow Start and the Congestion Avoidance phases.

1.1.3.2 Congestion Avoidance

After having "discovered" the available capacity of the path, the TCP connection enters the Congestion Avoidance phase and slowly increases the size of the congestion window. The goal is to send as much as possible and to quickly and conservatively decrease when a packet loss is detected.

$$\textbf{ACK}: \quad cwnd \leftarrow cwnd + \frac{\alpha}{cwnd} \tag{1.2}$$

$$\textbf{Loss}: \quad cwnd \leftarrow cwnd - \beta * cwnd \tag{1.3}$$

In Van Jacobson's proposal, the constant value chosen for the increase α is 1 and for the decrease β is $\frac{1}{2}$. This scheme is often called AIMD standing for Additive Increase and Multiplicative Decrease. If we neglect some of the later implementations' subtleties, like the delayed ACKs [Nagle, 1984] and the appropriate byte counting [Allman, 1998, Allman, 2003] (ABC, changing the congestion window size in terms of the number of bytes acknowledged instead of the number of packets), the congestion window size increases by 1 every RTT as shown on the right side of Figure 1.2. In the advent of packet losses on a timeout (RTO), it goes back to a Slow Start phase.

1.1.3.3 Improvements

To avoid false timeout detections, the value of the RTO needs to be significantly larger than the RTT (typically more than 4 times the RTT). It is then possible to wait for a very long time to be able to retransmit lost packets. To solve this problem, a Fast Retransmit algorithm [Stevens, 1997] is used : if 3 duplicated ACKs are received, it is considered that the following packet has been lost (and not subjected to some reordering on the path to the receiver) and it is immediately retransmitted without waiting for a timeout. It is often used combined with the mechanism of Fast Recovery that only reduces the congestion window to the ssthreshold when a loss is detected through multiple duplicated ACKs. TCP Reno is the first TCP version that included all these modifications.

Figure 1.3 illustrates the temporal evolution of the congestion window size. W is the maximum size that the congestion window can reach on the path. One can see the typical "sawtooth" that characterize its behavior. If the RTT is doubled, it takes twice as much time to reach the same congestion window size.

Over the years, a lot of improvements have been made to improve congestion control in some specific cases, such as SACK [Mathis et al., 1996] for the detection of multiple losses during a RTT period. But the AIMD algorithm has been for 20 years, the basic principle of

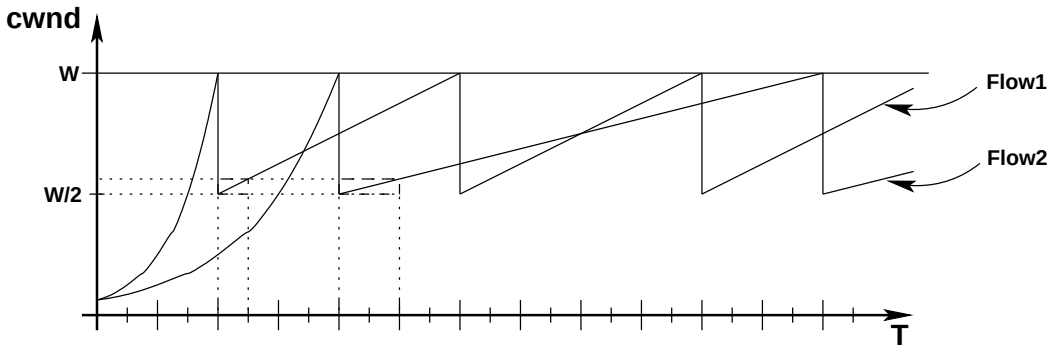


FIG. 1.3 – Congestion window size evolution for two independent TCP connections, on a path with a fixed RTT of t (resp. 2t)

all TCP implementations. Currently the reference version of TCP is called TCP NewReno, which improves TCP Reno's Fast Recovery algorithm. It is referred to as the standard TCP throughout the rest of this dissertation.

1.1.4 TCP PROPERTIES

TCP has shown to have a number of note-worthy properties that derive from the functions it implements :

1. **Scalability in number of users :** Due to the fact that from the start some TCP implementations were open-source, it became quickly wide-spread and is now included in most Operating Systems including GNU/Linux, Microsoft Windows or MacOSX. As a result, billions of end-systems are able to use TCP nowadays, even on their mobile phones. It makes it all the more difficult to quickly push changes to everybody (but it would be still easier than pushing a hardware change, like introducing a new type of technology overnight, in the core of the Internet).
2. **Stability and convergence :** In 1989, Jain proved [Chiu & Jain, 1989] the convergence of the AIMD algorithm. It is one of the most wanted properties for a congestion control mechanism, especially if the goal is to avoid congestion collapses. However, convergence time is another matter.
3. **Fair statistical sharing of the bandwidth :** If different flows are sharing a bottleneck and are experiencing the same RTT, it has been shown that each of them will statistically get an equal share of the bandwidth [Fredj et al., 2001]. It is a good property as it means that every flow will be treated fairly in these conditions.
4. **Robustness :** TCP is robust to various conditions of packet losses and RTT variations. It is designed to provide a best-effort service, just as the IP layer it is implemented on : if it can be delivered, it will be delivered (but with no time guarantee).
5. **RTT-clocked :** The granularity at which TCP's congestion control is able to react is the RTT. It is typically a few tens or hundreds of milliseconds in the Internet, mostly determined by the physical distance that needs to be crossed and bounded by the speed of light. The only explicit feedback TCP gets are the ACK packets and they can not come sooner than one RTT after the packet has been sent. It contributes to prevent congestion by limiting the amount of data an end-host can send during a RTT-period.

TCP possesses a wide range of useful and interesting properties, which accounts for its success throughout the years as a universal transport protocol. However, as the technology evolved and as new applications emerged, TCP started to show some limits in several use cases.

1.2 TCP Limits

In this section, some of the known limits of TCP are presented. These limits come mainly from two sources : applications and the tremendous evolution of technologies. The first case corresponds to new emerging applications whose use cases are slightly off TCP capabilities. The second case corresponds to limits that appeared through technical advances.

1.2.1 Applications limits

1.2.1.1 Multimedia applications

"Being robust" is one of the important principles behind the conception of TCP but this objective can be an hindrance to some time-sensitive applications. Indeed, the multimedia applications that have time-constraints (*e.g.*, VoIP or any real-time streaming applications) can be considerably handicapped by TCP. The reliable delivery feature of TCP can be more a bother than a convenience as there is no way to known precisely when the lost packet will actually be retransmitted and received at the destination. If we assume that this packet is immediately retransmitted after an RTO event, a delay of more than 200 ms will be added and it can be a too long delay for the application. These kind of applications (*e.g.*, VoD) usually require a constant bit rate (CBR) throughput rather than the widely fluctuating throughput achieved by TCP according to the congestion encountered.

Most of VoIP applications use UDP for their transport protocol, which is not reliable. Any lost packets would not be retransmitted and the data corresponding to the packet would simply be skipped over by the application[2]. Most web streaming applications (*e.g.*, YouTube) use TCP as their transport protocols but rely on buffering to cope with congestion instead. Specific transport protocols such as RTP/RTCP [Schulzrinne et al., 2003] (Real Time Protocol/Real Time Control Protocol) or SSTP [Wei et al., 2005] (Simple Streaming Transport Protocol) are designed to solve the problem of multimedia streaming. They are not discussed in this dissertation, which is more focused on the problem of high-speed networks.

1.2.1.2 Web applications

For applications that requires the successive transfers of small files, TCP is not necessarily the ideal transport protocol. Indeed, TCP's congestion control first step is designed to probe the available bandwidth through the Slow Start phase. To reach a congestion window size of W (typically the equivalent of BDP packets or 2 * BDP packets) at the end of the Slow Start phase, requires at least $log_2 W * RTT$ seconds. This phase can be extremely long if the RTT is large. It is especially important for small TCP flows that often do not have enough packets to send to even reach a large congestion window size. In this case, these flows will only be able to send at a small rate. About $2^{log_2 W+1}$ packets need to be sent to reach a congestion size of W.

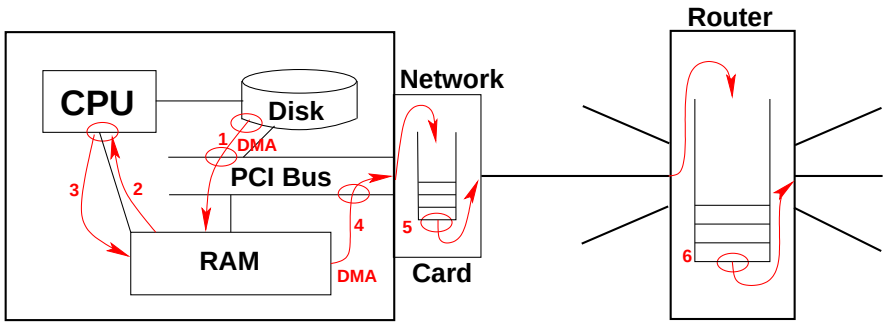
One typical example is the WWW where web pages usually consists in a collection of several small file objects (text or images). As their size rarely exceed a few kilobytes, it might be possible that they never exit the Slow Start phase.

1.2.2 Technological limits

1.2.2.1 End-hosts Hardware

Most hardware limits are external to the algorithms used by TCP but they can have a very strong impact on it by hampering its normal operating way. Over the path that TCP packets must travel through, they might encounter networking equipments that do not behave well. One of the typical example is the case of the middle boxes. Some of these intermediary network equipments were configured to drop TCP packets which were using the "window scale" option, a value negotiated at the opening of a TCP connexion to overcome the 64 KB size limitation

[2] resulting in a blank or a bleep depending how robust to missing data the audio decoder used is



1) Data copied from the disk to the RAM via the PCI bus
2) Data moved to the CPU (splitting into packets)
3) Packet copied from the RAM to the transmit queue
4) Packet copied to the NIC via the PCI bus
5) Packet transfered on the network after waiting in a queue in the NIC
6) Packet handled in a router after waiting in a queue

FIG. 1.4 – The existing hardware bottlenecks encountered in an end-host hardware

of the congestion window's size. This drastically limits the maximum rate achievable by TCP while using a path going through such equipments [Medina et al., 2005].

Due to the different evolution speed of the various hardware parts that make up a end-system, it is possible that some elements do not provide a sufficient bandwidth for the end-system to optimally perform. One example is when the first generation of 1 Gbps network interface card (NIC) were introduced. The CPU speeds or PCI bandwidths of the time were not enough to achieve gigabit speeds in most end-systems. A few years later, after the introduction of new standards and a new generation of hardware, the problem is pushed back to the next order of magnitude : 10 Gbps network interfaces.

Figure 1.4 presents the possible different bottlenecks encountered while trying to transfer a file read from a hard drive disk to a remote location. The failure of one of these points is likely to severely drop the performance of the whole system.

1.2.2.2 Wireless Networks

TCP congestion control was designed to be conservative with dealing with congestion. This accounts for the coefficient used to reduce the sending rate when losses are detected. Unfortunately when TCP uses a wireless medium instead of the wired medium for which it was initially designed for, congestion is not the only reason to account for losses : packets can also be lost through corruption due to bits error in the packet, detected by an invalid checksum and dropped on the receiver's side. Packet corruption can also occur in wired environments but the rate of such events is much lower. Both kinds of losses – congestion and corruption – cannot be discriminated against by TCP, which will have the same reaction : dividing by two the congestion window size.

It might not be the best way to cope with this unfortunate property of wireless networks and using a recovery mechanism at the MAC layer (L2) [Heusse et al., 2006, Heusse et al., 2005] or efficient redundant network coding might prove more efficient. This problem is all the more worsened when energy limitations increase the cost of sending packets or when used over long distances.

1.2.2.3 Long Fat Networks

"Long Fat Networks" or "LFN" [Jacobson & Braden, 1988] is the term that names networks characterized by a large bandwidth capacity (*e.g.*, optical networks) and/or a large end-to-end delay. They are called LFN whenever their bandwidth-delay product or BDP is larger than

10^5 bits (about 12 kB). Nowadays a typical DSL connexion of 10 Mbps retrieving a document over a 100 ms RTT transatlantic path (BDP of about 120 kB) would fall into this category.

The main problem comes from the algorithm used within the Congestion Avoidance phase. The AIMD algorithm is designed to be very conservative and only increases linearly the number of packets that can be sent in a time period in the network. It usually results in a very slow convergence to the maximum available bandwidth. Indeed, to be able to send at 1 Gbps on a 100 ms RTT path using 1500-bytes packets, a congestion window size of about $W_{1Gbps} = 8333$ packets must be reached. As the congestion window size can grow at most by one every RTT in the Congestion Avoidance phase, it will take about 4617 RTT-periods, more than 460 s, to go from $\frac{W_{1Gbps}}{2}$ to W_{1Gbps} (*i.e.*, to recover from a loss) and this recovering time is linear with the BDP.

Padhye [Padhye et al., 1998] introduced in 1998 an analytical model of TCP, expressing a response function (the achievable mean throughput) of TCP as a function of the loss rate (p), of the RTT, of the RTO and of the TCP maximum segment size (MSS). This equation put forward the impact of the RTT on the performance of TCP : to keep a constant throughput, a RTT multiplied by ten must be compensated for by a loss rate divided by one hundred. To achieve a continuous 10 Gbps transfer over a 100 ms RTT path using 1500 B packets, it would mean losing at most one packet every 5 billion packets, which is a difficult guarantee to keep, even for current-time optical networks.

$$R = \frac{MSS}{RTT * \sqrt{\frac{2p}{3}} + RTO * \min(1, 3\sqrt{3p/8}) * p(1 + 32p^2)} \tag{1.4}$$

The second problem comes from the RTT-clocked nature of TCP. If flows sharing a bottleneck have roughly the same RTT, every one of them gets a fair share of the capacity of the bottleneck (statistically speaking [Fredj et al., 2001]), it may not be the case when considering instantaneous throughputs). The problem arises when flows with different RTTs are sharing the bottleneck, which is very likely to occur in a network like the Internet as the traffic is most like a mix to different destinations. In this case, flows with smaller RTTs will get their ACK feedbacks much more quickly than the ones with larger flows. The flows with small RTTs end up increasing their share of the bandwidth faster than the flows with large RTTs as seen in Figure 1.3, resulting in an unfair sharing of the bottleneck among the flows. This problem is known as RTT-unfairness [Floyd, 1991].

Since these limits have been observed [Vicat-Blanc Primet & Martin-Flatin, 2005] [Vicat-Blanc Primet et al., 2005b], several solutions have been proposed. Some have already been implemented as extensions of TCP, for instance SACK [Mathis et al., 1996] that limits the cost of losing several packets in the same RTT window. Some are solving the fore-mentioned problems by introducing new congestion control schemes.

1.3 Solutions

In this section, we present a review of solutions that have been considered to replace the legacy TCP in order to alleviate the limits in the LFN presented in the previous section. As new proposals are made every year, it is very difficult to be completely exhaustive, but this work sorts them out and cites the most prevalent ones, focusing on solutions that allow a total reliability of the transfer. Other types of classifications can be found in [Ryu et al., 2003, Welzl, 2005, Vicat-Blanc Primet et al., 2005a, Wang et al., 2009].

We identify three solution classes to scale up in performance in high-speed networks :

- Applying a modification at an upper or lower layer level than the transport protocol.
- Using a different transport protocol that would not have the same deficiencies as TCP.
- Modifying directly TCP through the change of the algorithms used by the Congestion Control mechanism.

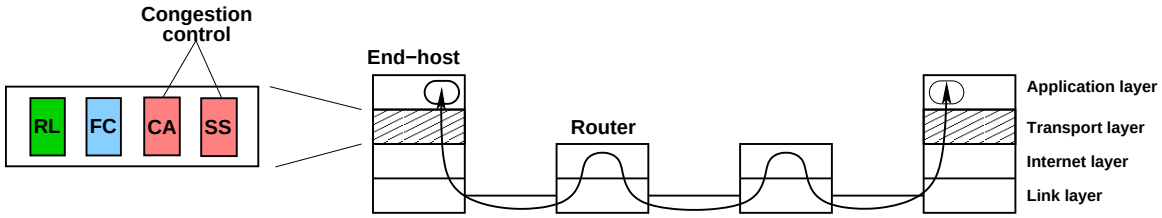


FIG. 1.5 – Communication through a network stack

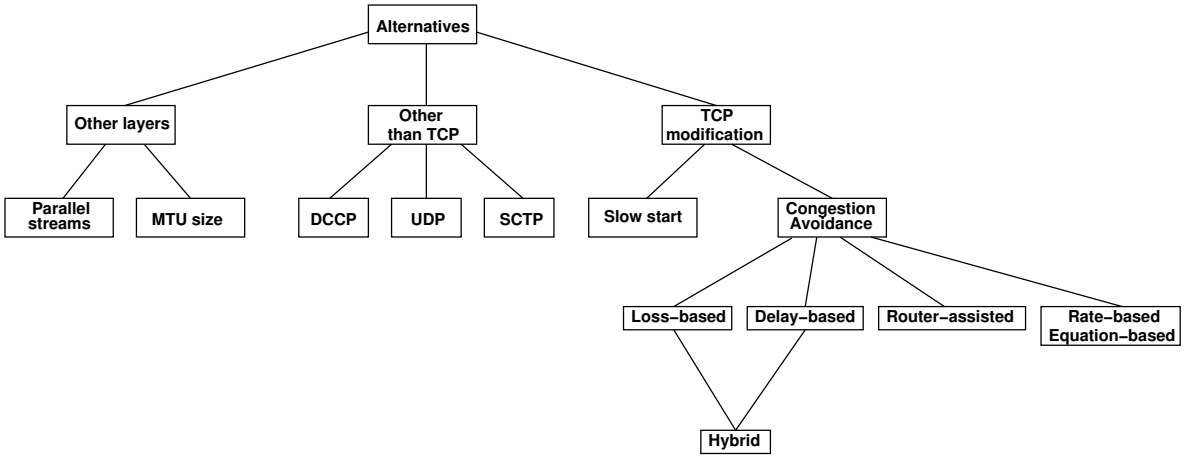


FIG. 1.6 – Tree of the different alternatives to scale up in performance in high-speed networks

Their main goal is to workaround the long RTT feeback loop of TCP and to allow to send, in average, a larger quantity of data during a RTT. Figure 1.5 shows the differents places in the network stack where it is possible to solve the limits of TCP and Figure 1.6 presents the tree of the different alternatives presented in this dissertation. Figure 1.7 presents the chronological apparition of the different solutions that are presented in this document. The elements in red correspond to important events in terms of development of the network's organization.

1.3.1 Other layer modifications

1.3.1.1 MTU size increase

An alternative to achieve a higher rate is to use a larger packet size (MTU) at the data link layer. In an Ethernet network, the size of the MTU is set in adequacy with the size of the Ethernet frame. It allows to reduce the cost of the protocol headers, which increases the efficient bandwidth of the network (going from an overhead of 52/1500= 3.5 % per packet to 52/8192 = 0.6 %). It also has a beneficial effect on the hardware limitations as it reduces the number of packets that need to be handled by the operating system for the same amount of data. They are usually referred to as "Jumbo Frames" [Dykstra, 1999]. The limit of this solution is that it requires all the networking equipments along the path to handle larger MTU size.

1.3.1.2 Parallel streams

As TCP allows the bandwidth of a bottleneck to share equally among flows, it has been envisioned to multiplex a flow over several TCP connections to increase its share of the bandwidth. It has been theoretically [Altman et al., 2006] shown that this technique is able to improve the performance of TCP connections.

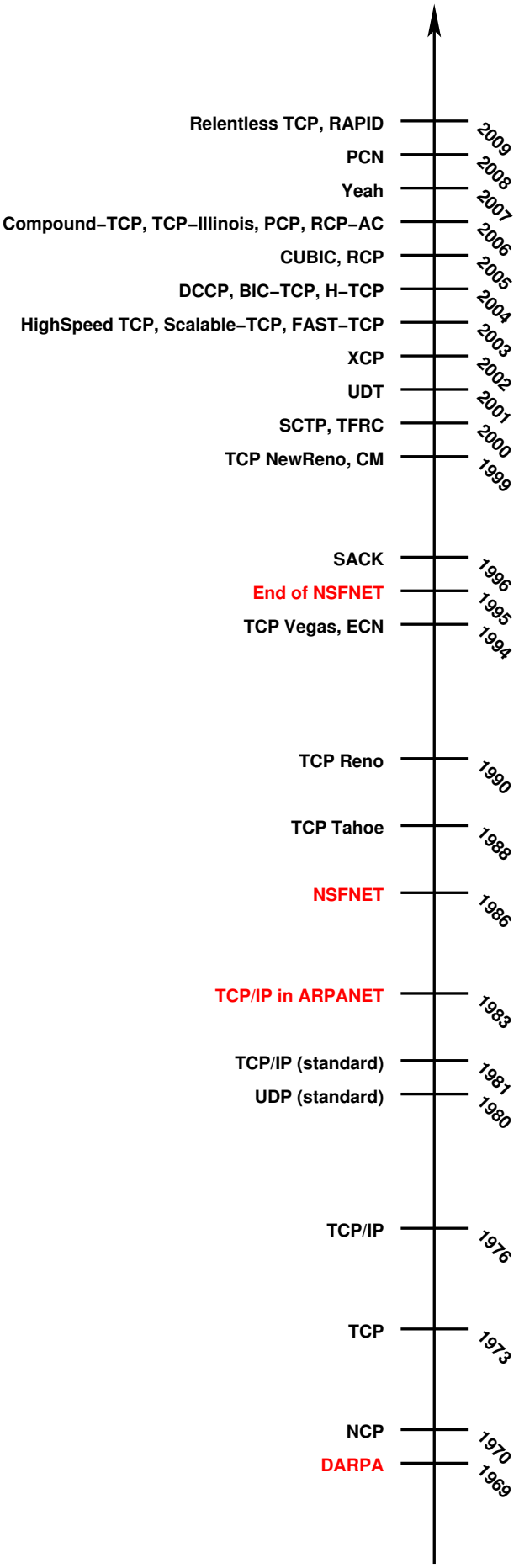


FIG. 1.7 – Chronology of Transport Protocol developments

MulTCP [Crowcroft & Oechslin, 1998] is a TCP that emulates a virtual collection of several TCP connections. During the Congestion Avoidance phase, it aims at increasing the congestion window the same way N TCP connections would do. It is then possible to get as much throughput as N TCP connections would. To regulate the choice of N (and prevent a congestion collapse), it is suggested to introduce a billing mechanism.

PSockets [Sivakumar et al., 2000] is an application-level library that aims at enhancing the current UNIX Socket API. The goal is to provide a way to transparently do network stripping, *i.e.*, to partition data and send it over several streams, with little modifications to existing applications. It allows choosing the number of parallel streams to use for a transfer and it removes the need for the end-user to tweak several transport protocol parameters to achieve good performance in high-speed networks.

ParallelTCP [Hacker et al., 2004] also promotes the use of parallel TCP streams to improve the performance of applications. It introduces in the Congestion Avoidance phase a fractional factor corresponding to a virtual RTT that is shared by the multiple streams composing the TCP connection. When this factor increases or decreases, it influences the growth rate of the congestion window, and so its way of competing with other flows to get bandwidth.

GridFTP [Allcock W., 2003] is an extension of the FTP protocol designed for grid applications that can use parallel streams to accelerate file transfers. The file is cut into several chunks and the transfer can be distributed over several flows, targeting either a single computer or several. The end-user needs to indicate the number of flows he wants to use.

BitTorrent [Bram Cohen, 2008] is a popular file-sharing protocol (Peer-to-Peer). Files are cut into chunks and each chunk can be exchanged concurrently by several users. In effect, a file is retrieved from multiple sources at the same time. A "tit-for-tat" mechanism (users get an higher priority from users they upload chunks to) is implemented to ensure that every user is participating fairly.

The usual drawback with this kind of approach is that it requires a modification of the end-user application (even if it only means compiling it against a specific library) or the use of a specific application. It has also been shown that synchronization [Zhang et al., 2005], local overheads and buffer occupancy can downgrade this performance. It is also open the discussion on the fairness and friendliness principles of the Internet as the sharing of the bandwidth is performed at the flow level : a single end-host can possibly get more than several others if it uses a large number of non-cooperative parallel streams.

1.3.2 Transport Protocols other than TCP

It has also been envisioned to completely change TCP by plugging another transport protocol to be used by the application. This allows to have a transport protocol with different functions and properties than TCP and possibly solving its limits.

1.3.2.1 UDP-based

The User Datagram Protocol (UDP) [Postel, 1980] is a very simple transport protocol that differs from TCP by the fact that it is not a reliable protocol. Indeed, it does not establish a connection between the communicating end-hosts, there is no guarantee that the datagrams sent will reach the destination application in the right order and it is not able to retransmit lost packets as there are not any feedbacks or acknowledgments returned by the destination. The only integrity check is performed using a checksum on the destination host, which will drop the datagram on a mismatch. UDP is traditionally used by streaming applications (*e.g.*, VoIP, VoD) whose performance greatly depend on the inter-packets jitter, and which would

be hampered by the retransmissions performed by TCP on lost packets. UDP is also used by applicative protocols like DNS or DHCP that do not require the exchange of a lot of packets to operate or for which the cost of opening a reliable connection would still be greater than the cost of retransmitting the whole request.

The approaches based on UDP for other applications like file transfers are required to be reliable, so a reliability mechanism is usually added at the application level. It is also up to the application level to control the sending rate to avoid congesting the network heavily and dramatically deteriorating the performance of all the other users. In adequate conditions, like in a private network with few packet losses, UDP approaches can be more efficient than TCP [Casey et al., 2007]. UDT [Gu & Grossman, 2007] is an example of such approaches. It creates a reliable connection using TCP to open a control channel between the end-hosts and a UDP channel to transfer the data. It uses an explicit mechanism of notification by positive and negative acknowledgments to handle the reliability and the packets' retransmission. UDT was also designed to be modular for the congestion control algorithm it is using at the application level. It is then able to simulate the behavior of any congestion control scheme to adapt to the context of the other flows in the networks.

The common limit of these kinds of approaches is the memory and CPU overhead of the software implementation of reliability at the application level. Some network providers are also known to put filtering rules in their network equipments to drop and/or to limit the bandwidth of UDP streams [Medina et al., 2005].

1.3.2.2 DCCP main functions, features and properties

The Datagram Congestion Control Protocol (DCCP) [Kohler et al., 2006] is designed to be some sort of middle-point between TCP and UDP as it aims to remove the constraint of implementing the congestion control function at the application layer but still provide an unreliable message-transfer service suited for streaming applications. The end-hosts using DCCP manage their connection in a similar way to TCP. The receiver sends back an acknowledgment for every packet received. If the datagram is dropped on the receiver's end, an ACK can be sent anyway with a field indicating the reason for the drop (*e.g.*, bit error, buffer overflow) which allows the sender to discriminate against the different kinds of losses. The sender can then identify the packets that were probably lost (or whose ACKs got lost), but it does not need to retransmit the last datagrams. The accounting of the sent data is performed using a packet number rather than a byte number like in TCP.

It is also designed to take notice of the ECN [Floyd, 1994] markings if the network equipments used on its end-to-end path have this capability. Currently two specific congestion control schemes have been defined : a "TCP-like" one [Floyd & Kohler, 2006] which reacts like TCP when a loss is detected or an equation-based congestion-control scheme that is similar to TFRC [Floyd et al., 2006]. The congestion control scheme to use is negotiated at the beginning of the connection and new schemes can be added. However this transport protocol is still not widely used due to the experimental status of the implementation of this protocol in most major operating systems.

1.3.2.3 SCTP main functions, features and properties

The Stream Control Transmission Protocol (SCTP) [Stewart, 2000] is a transport protocol first designed for the public switched telephone network. It provides a reliable transport service with SACK-like feedbacks to handle retransmissions. A congestion control scheme similar to TCP is used. The main difference with TCP is that it is natively able to handle multi-homing (the capacity to use several paths connected to different physical network interfaces seamlessly) and multi-streams (the capacity to sent several data flows multiplexed over the same connection). Implementations exist in several major operating systems but it is not widely used. SCTP is mostly used by VoIP applications and telephone signaling systems (OpenSS7).

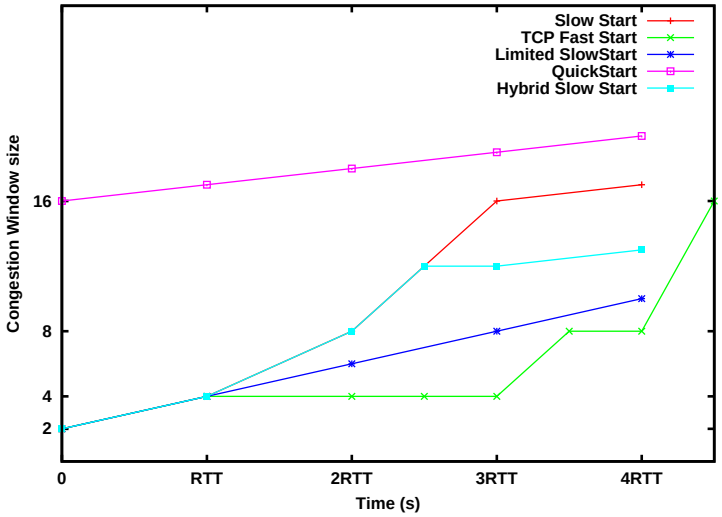


FIG. 1.8 – Congestion Window size evolution pattern for the presented Slow Start modifications

Several transport protocols other than TCP have been introduced but they are not currently used by a large number of applications, limiting their impact.

1.3.3 CONGESTION CONTROL MODIFICATIONS

The last way to change the behavior of TCP is to change the way it acquires and releases bandwidth, namely by changing the congestion control algorithms. It is typically done by proposals that aim at cranking up the performance of a single TCP connection on networks with large capacities and long distance connections.

1.3.3.1 SLOW START MODIFICATIONS

One of the first ways to change the behavior of the TCP's congestion control is by changing the Slow Start algorithm. To solve this problem, several solutions have been presented. Some schemes introduce another algorithm that has a faster increase rate than the original Slow Start algorithm. Figure 1.8 provides an idea of the evolution pattern of the Congestion Window size for the presented Slow Start modifications for ssthreshold of 16 packets.

TCP Fast Start [Padmanabhan & Katz, 1998] is designed to improve the transfer of web pages. Web pages usually consist of a collection of small objects, a few KB in size and a new TCP connection (and a new Slow Start) might be needed to retrieve each of them. Due to their small size, it is very likely that each independent TCP connection will not be able to exit the Slow Start phase. The idea here is to keep in memory the TCP variables involved in the Slow Start over several connections (namely the congestion window size and the ssthreshold). In order to avoid congesting the network, a lower priority is given to the packets using old state variables. A timeout is used to reset the variables after a long idle time (*e.g.*, five minutes).

Limited SlowStart [Floyd, 2004] takes as a starting point that brutally doubling the congestion window size every RTT is not a good idea. In the worse-case scenario, the congestion window size will reach exactly the number of packets that can be sent through the bottleneck. At the next RTT, the sender will transmit twice the capacity of the link. No matter the amount

of buffering available along the path, it can only lead to massive losses. The Limited Slow Start scheme introduces a $max\_ssthresh$ parameter. After the congestion window size has grown over $max\_sstresh$, it slows down the increase by a fraction of the congestion window. In effect, the congestion window grows up by $\frac{max\_sstresh}{2}$ every RTT.

Quickstart [Sarolahti et al., 2006] was proposed to completely short-circuit the Slow Start algorithm and use indications provided by the encountered networking equipment along the path. In this algorithm, a request is sent with the SYN packet as an IP option with the value the sender wants to start emitting at. The routers along the path would then change this option by putting a new value if necessary, *e.g.*, if their output capacity or available output bandwidth is lower than the value asked for. The sender will receive the minimum value it is entitled to start sending at with its first ACK.

Hybrid Slow Start [Ha & Rhee, 2008] is based on a packet-train technique using the successive packet bursts generated by the Slow Start phase to estimate the available bandwidth of the bottleneck. If an increase of the RTT is detected, it is interpreted as a pre-congestion signal and the flow goes to the Congestion Avoidance phase. This algorithm is designed to be less aggressive than the usual Slow Start by causing less packet drops as new-coming flows would not cause drops to existing flows.

This kind of modification is a good way to improve the performance of small TCP flows, but the slow start only corresponds to a small part of the total transfer time of a large transfer. Consider a 1 GB file transfer over a 1 Gbps network with a 100 ms RTT (the BDP of the path is then about 12 MB). During the Slow Start phase, about $2W_{max}$ packets will be sent. It will only account for a little more than 2 % of the total transfer (but in duration, it will account for about 13 %), which is negligible. Thus, modifying the Congestion Avoidance part of the TCP's congestion control is also necessary to improve the performance for large transfers.

1.3.3.2 Congestion Avoidance modifications

This section presents some of the different proposals modifying the Congestion Avoidance algorithm to improve the performance of TCP on LFNs. They are usually called "TCP variants". Most of them follow the guidelines stated in the Congestion Control Principles RFC [Floyd, 2000], *i.e.*, preventing congestion collapse, being fair to best-effort traffic (also called "being TCP-friendly"), and optimizing performance regarding throughput, delay, and loss. These guidelines are provided to avoid some sort of transport protocol arms-race in which everyone would crank up some parameters to improve the performance of their own flows till everything collapses. The TCP variants presented here are organized according to the general technique they use.

[Yang & Reddy, 1995] already provides a classification of the different types of congestion control algorithms but it takes a control theory point of view. Figure 1.9 presents the tree of different solutions presented in this document.

(1) Loss-based

These variants, like TCP, rely on the detection of losses to infer that there is congestion in the network. From a bird's eye view, they mostly rely on changing the *alpha* and *beta* parameters of the AIMD. Table 1.III summarizes the values used by the different TCP variants. Figure 1.10 provides an idea of the evolution of the Congestion Window size for the Loss-based TCP variants presented in this section. The graphs were made through simulations in NS-2, with a single 1 Gbps flow over a 1 Gbps link with a 100 ms RTT. It highlights the long time necessary for TCP Reno to achieve high throughputs in such conditions (in the order of magnitude of 500 s) while the other TCP variants have reaction times in the order of

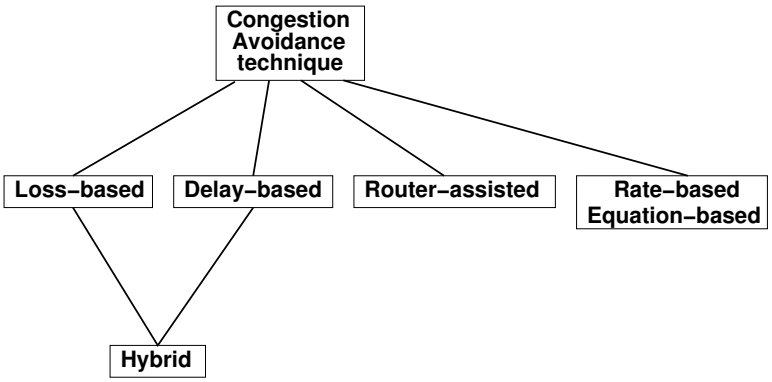
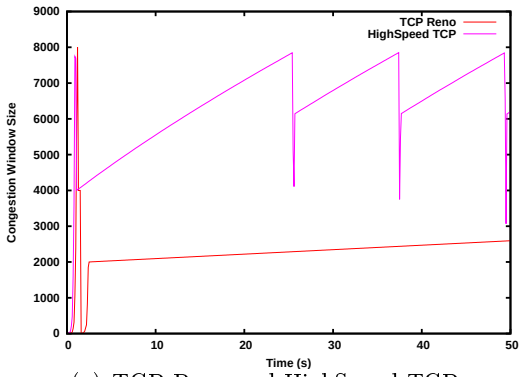
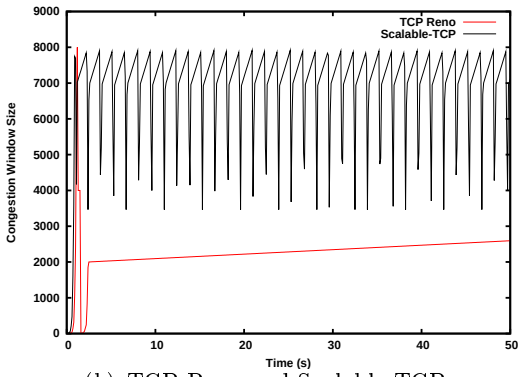


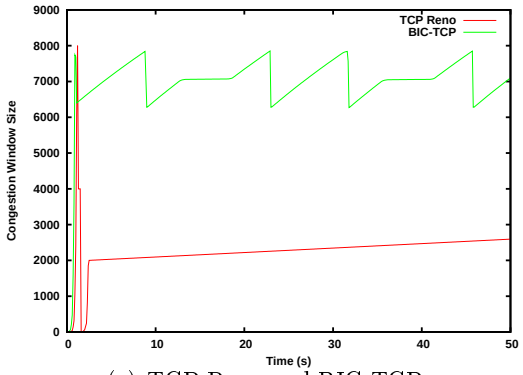
FIG. 1.9 – Tree of the different solutions of the Congestion Avoidance modifications



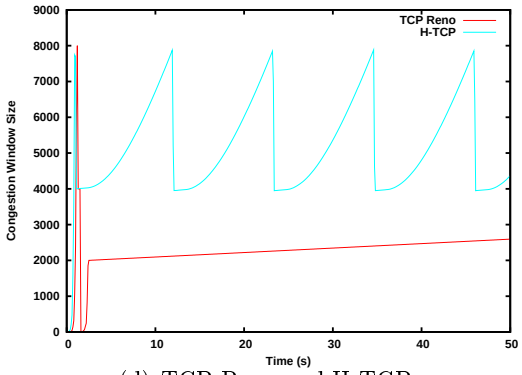
(a) TCP Reno and HighSpeed TCP



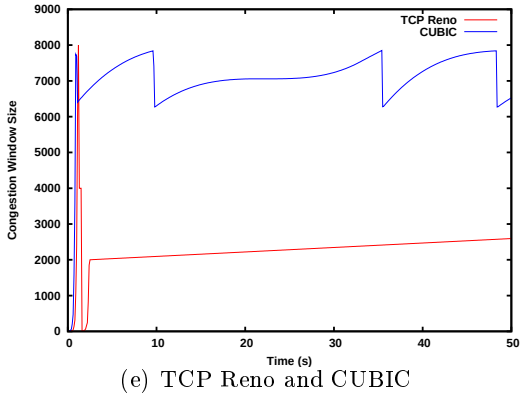
(b) TCP Reno and Scalable-TCP



(c) TCP Reno and BIC-TCP



(d) TCP Reno and H-TCP



(e) TCP Reno and CUBIC

FIG. 1.10 – Congestion window evolution pattern for presented Loss-based TCP variants

magnitude of the second or the tens of seconds. It also possible to notice some of the features of the different TCP variants, such as the famous cubic shape of CUBIC.

HighSpeed TCP [Floyd, 2003] was introduced in 2003 by Sally Floyd. HighSpeed TCP's goal is to solve the problem of the recovering time between two consecutive losses that increases linearly as the capacity of the link increases. It modifies the constant values α and β to replace them by respectively an increasing function and a decreasing function of the congestion window size. This way, HighSpeed TCP is able to be more aggressive in LFNs. To remain fair to TCP, the congestion window needs to go over the threshold of 38 packets for this new mechanism to kick in.

Scalable-TCP [Kelly, 2003] was introduced in 2003 by Tom Kelly for networks with a very large BDP. The idea is to increase by a constant value the congestion window, regardless of its current value, and to use a smaller value to decrease it ($\frac{1}{8}$ instead of $\frac{1}{2}$). Using this scheme, Scalable-TCP is able (for a constant RTT) to guarantee a constant recovery time between losses, regardless of the bandwidth of the link. It starts working this way as soon as a threshold of 16 packets is reached.

BIC-TCP [Xu et al., 2004] was introduced by Injong Rhee in 2004. As its full name (Binary Increase Congestion Control) hints at, it adds a binary search of the optimal working point to the Congestion Avoidance phase. Two values are used : W_{min} and W_{max}, corresponding to respectively the last value of the congestion window for which no loss occurred and the last value for which a loss occurred. If the congestion window size manages to grow over W_{max} a phase of fast increase is used to find a more appropriate value.

H-TCP [Shorten & Leith, 2004] was introduced in 2004 by Doug Leith to solve some of the problems caused by HighSpeed TCP and BIC-TCP : the more their congestion window increases, the more aggressive they are, which is a disadvantage for new incoming flows. The solution introduced defines a function that depends on the last loss event to modify the increase rate of the congestion window size. To decrease the congestion window size when a loss occurs, the measured RTT is used to moderate the β parameter. In a network where the queues are completely saturated (and so the measured RTT will increase), the effect is to decrease all the more the congestion window to go back to a more stable situation.

CUBIC [Rhee & Xu, 2005] was introduced in 2005[3] by Injong Rhee and used some of the ideas from BIC-TCP, especially in the function used to increase the size of the congestion window (which this time is a cubic function). The epoch of the last loss is used to regularly update, uncorrelated with the actual RTT value, the congestion window size. Strictly speaking, CUBIC is not using an AIMD-like algorithm. Another difference with BIC-TCP is that CUBIC has a "TCP-mode" in which it tries to be fair with TCP flows when low BDP conditions are detected. CUBIC was set to be the default version of TCP used in the GNU/Linux kernel since the 2.6.19 version.

The main problem with this kind of TCP variants is that they are often plagued with problems of convergence speed and of fairness to other protocols as they tend to be very aggressive in the way they acquire a share of bandwidth.

[3] An updated version of CUBIC was introduced by the authors in 2008 on their web site. The main difference lies in the existence of an additionnal fast convergence mode during which the congestion window is only decreased by one tenth during a loss event. This version of CUBIC is not considered in the rest of this dissertation

TCP variant	α	β
TCP Reno	1	$\frac{1}{2}$
BIC-TCP	1 or $bin.search$	$\frac{1}{8}$
CUBIC	$cub(cwnd, history)$	$\frac{1}{5}$
HighSpeed TCP	$inc(cwnd)$	$decr(cwnd)$
H-TCP	$f(last_{loss})$	$1 - \frac{RTT_{min}}{RTT_{max}}$
Scalable-TCP	$0.01 * cwnd$	$\frac{1}{8}$

TAB. 1.III – AIMD "constants" used by some TCP variants for LFN networks

(2) DELAY-BASED

These variants relies on the evolution of the delay existing between the end-hosts to infer if there is congestion along the path. Indeed, the network equipment often contains buffers that are used to keep extra packets when there is contention for the same output port – depending on the actual router/switch architecture used – instead of just dropping them. These buffers are of variable size, the engineering rule-of-thumb being to set them to a value close to the BDP for a 100 ms RTT. It is usually done so for the high-end routers that are used in the core of the Internet but recent works have shown that under some specific traffic conditions (high level of multiplexing), lower values can be used [Appenzeller et al., 2004, Enachescu et al., 2006, Prasad et al., , Beheshti et al., 2008].

Using such a buffer might result in an increase of packet-transit delay. This delay increase is then considered to be a congestion signal and the variants act before causing actual losses in the network. This is completely different from the loss-based TCP variants as loss-based TCP variants need to have dropped packets (by generating congestion) to notice congestion.

TCP-Vegas [Brakmo et al., 1994] While the idea was already used in 1989 by Jain [Jain, 1989], one of the variants that popularized such an approach is called TCP-Vegas. It assumes that the expected throughput is the congestion window size divided by the minimum RTT measured. It then compares to the actual throughput. The difference between the two is caused by the packets that spent time in some queue in a networking equipment. Two parameters α and β are used to linearly increase and decrease the congestion window size according to this difference. A different Slow Start phase is used by only doubling the congestion window size every other RTT. It also uses a different retransmission algorithm with respect to the duplicate acknowledgments : if a dupACK is not received in the RTT-period in which the ACK for a packet was expected to arrive, then the packet is retransmitted right away.

FAST-TCP [Wei et al., 2006] is another example. FAST-TCP flows try to maintain a constant number of packets in the queues of the network. It is measured through the evolution of the RTT with respect to the minimum RTT that has been observed during the connection on the path. In the case where the number of queued packets seems to have increased (resp. decreased), the sending rate is decreased (resp. increased). The further away from the goal, the more important the variation of the sending rate will be in order to improve the convergence rate.

The main problem with this kind of approach is that it is necessary to have precise delay measurements : in a 1 Gbps network a 1500 B packet is sent in 12 μs, the measure needs to have a similar granularity to perceive the evolution of the queueing delay. It is usually difficult

to do so with software timestamps but hardware dedicated boxes can do that, which is the probable solution used by Fastsoft, the company commercially exploiting FAST-TCP.

Temporal spacing Other examples of delay-based TCP variants are variants that tries to take advantage of the temporal spacing between packets to deduce the current available bandwidth of the network after their dispersion through a bottleneck link. One of such schemes was introduced in 1999 by Allman [Allman & Paxson, 1999] to describe a receiver-side TCP algorithm that could send indication to the sender about the right rate to send at through the advertised window. The idea is to use the cumulative acknowledgment property to make the sender transmit small bursts of packets back-to-back, use their dispersion to compute the available bandwidth and then provide a feedback to the sender. In [Anderson et al., 2006], PCP targets networks with low load and medium-sized flows and uses short regular probes to detect the rate at which the transport protocol should send. The same kind of idea is used by RAPID [Konda & Kaur, 2009] that uses a packet train technique to measure the available bandwidth and uses this information to dramatically reduce the time needed to reach a congestion window size corresponding to the probed available bandwidth.

These techniques usually yield good results on a theoretical (and in simulations) point of view but they have shown accuracy limits when tried in real world situations. It is mainly due to current limitations in terms of OS's scheduler granularity and resolution of timestamps.

(3) Hybrid

This category regroups the TCP variants that are using both previous congestion detection methods. The idea is that having two sources of information to detect congestion events will be more efficient than using just one. Usually one is used as a primary congestion detection method (*e.g.*, detection by loss) whose effect is moderated by the secondary congestion method (*e.g.*, detection by queueing delay).

TCP-Illinois [Liu et al., 2006] is a TCP variant whose idea stems for the shape of the evolution of congestion window of a TCP. As it has a triangular shape during the Congestion Avoidance phase, one half of the "space" is lost. The idea then is to select the appropriate functions as AIMD parameters α and β to minimize the amount of lost "space", *i.e.*, to have a concave function. These functions evolve according to the evolution of the average queuing delay. If the queueing delay indicates that congestion might occur, then a small α and a large β should be used and the contrary if it seems that there are no risks to have congestion.

Compound-TCP [Tan et al., 2006] uses two counters to keep track of the number of packets it can send. The first one is similar the congestion window of TCP and evolves the same way, the second is a delay-based window and evolves in the opposite direction of the queueing delay (increases if the queueing delay is low and decreases if the queueing delay increases). The goal of this TCP variant is to maintain the sum of both windows constant with a value close to the BDP of the used path. When its estimation of the number of queued packets increases, it tries to compensate it so as to empty the queue. Compound is different from TCP-Illinois in the sense that it uses both delay and loss as a primary congestion detection method. This variant was supposed to be used as the default congestion control for Microsoft Vista, however fears of instability led it to only be a optional experimental feature.

Yeah [Baiocchi et al., 2007] defines 2 different operating modes it uses during the Congestion Avoidance phase : a slow one and a fast one. During the "Slow" mode, it acts as TCP. In

the "Fast" mode, it tries to be aggressive in its way to increase the congestion window size[4]. The selection of the operating mode is selected on the basis of the number of packets queued in the network and the amount of congestion caused by the flow. If too many packets are queued, it then tries to perform a "decongestion", but limiting its sending rate to flush the queues.

The main limit of these techniques is that they require a fine tuning of their internal parameters to achieve their goal of combining the advantages of the loss-based and the delay-based techniques. Invalid parameters in a given context can lead to the instability of the protocol.

(4) Router-assisted

The usual way routers handle congestion is not particularly smart : they drop all packets that exceed their capacity and buffers (the drop tail mechanism). This is not the best way to provide a feedback to a sender, but it allowed the routers to do just one simple thing : (store and) forward packets quickly. Some authors argue that it is no longer enough and that new functions have to be added in routers. Over the years, new schemes [Ryu et al., 2003, Labrador & Banerjee, 1999] were introduced to modify the behavior of the network queues so as to provide a much earlier congestion warning. They are usually called "Active Queueing Management" schemes. One example is RED [Floyd & Jacobson, 1993] and its multiple variants which introduce a mechanism to randomly drop packets before experiencing real congestion. Another example is FairQueuing [Nagle, 1987] that uses a round-robin-like algorithm between the end-hosts to make sure each of them get a fair share of the bandwidth. This category regroups the TCP variants that rely on information provided by the networking equipments to adapt their sending rate.

TCP-ECN [Floyd, 1994] starts from the idea that the networking equipments are the most suited to know when a congestion event might occur. But instead of just dropping some packets, they can also forewarn the end-users that they are close to its capacity limit. It is performed by a random marking of TCP packets by using the ECN bit in the TCP headers. It usually works by setting two thresholds on the networking equipment's queues : a low and a high one. Above the high one, packets are dropped ; below the low one, nothing happens and between the two thresholds, some packets are marked. A TCP variant that would abide by these marked packets would then reduce its sending rate to avoid actual packet drops and the cost of retransmissions. Recent proposal PCN [Menth et al., 2008] is an extension of this principle.

XCP [Katabi et al., 2002] can be seen as a sophistication of ECN. XCP modifies the headers to have the sender fill three new fields : one containing its current congestion window size, one containing its current estimation of the RTT and the last one containing a router-provided feedback. Initially the feedback contains the number of extra packets that the sender would like to send to reach the rate wanted by the application. The routers can then provide information by reducing or increasing the value in this feedback field depending on their available bandwidth. The receiver sends this header back with the corresponding ACKs and the sender is then able to adjust its sending rate to match the feedback. Modifications of the routers are needed to modify the fields in each packet and keep flow stats to ensure fairness and efficiency between the flows.

RCP [Dukkipati et al., 2005] is designed to improve the completion time of file transfers. To avoid keeping a state for each flow like XCP, the routers' feedback contains a rate that corres-

[4] using the same scheme as Scalable-TCP but it can be easily changed as this TCP variant is modular by design

ponds to the equal sharing of the bandwidth among the flows crossing it. This value is estimated once per RTT in average using an estimate of the number of flows currently crossing the router. The feedback the sender finally receives is the minimum of bandwidth allocated along the way. RCP tries to emulate processor sharing. Its evolution RCP-AC [Dukkipati et al., 2006] adds a feature to allow new-coming flows to quickly reach a high sending-rate, as well as a control mechanism to moderate the buffer occupancy if there is congestion in the router, but it reinstates the need to do per-packet treatment like XCP.

Relentless TCP [Mathis, 2009] is a TCP variant that requires the existence of networking equipment that allocates a given capacity to each flow (such as FairQueueing). The idea is to keep the congestion window close to the estimation of the good value of its share of throughput. It is done by reducing the congestion window size by exactly the same number of packets as the number of packets dropped or marked by the networking equipment. As this TCP does not back off as much as the legacy TCP when packets are lost, it is not TCP-friendly.

The limit of these techniques is that they break the end-to-end principle by needing additional (potentially costly) treatment in the networking equipments and that they often fail to deliver when some routers along the path do not have the capability to handle the special feature required by the transport protocol. It might also be difficult to scale up in speed if transport protocols requires per-packet treatments because, to reach full-wire speed, a 1 Gbps router output port has to process a 1500 B packet every 12 μs. Treatment time should be in the same order of magnitude to avoid adding extra delays.

(5) Rate-based/Equation-based

Instead of relying on a window size to send a certain number of packets, it is also possible to make a TCP-like transport protocol send at a fixed rate like it is done with UDP, for example. This kind of approach started when closed-formula of the TCP rate as a function of the loss rate and the RTT were established (more details on this subject are provided in Chapter 3).

TFRC (TCP-Friendly Rate Control) [Floyd et al., 2000] takes its root in the equation of a steady-state rate of a single TCP flow that was introduced in 1998 by Padhye [Padhye et al., 1998]. It expresses the throughput as a function of some of the TCP connection parameters such as the MTU, the loss rate and the RTT. The idea here is to obtain a good estimate of the loss rate and of the RTT, to feed the formula and find the appropriate rate to send at. Several mechanisms to smooth out the oscillations of both the loss rate and the RTT are included in the protocol. An extra flow-control algorithm is added on the receiver's side to prevent it from more than doubling its rate during two consecutive RTTs in the Slow Start phase. This feature allows TFRC to be friendly to TCP as it would at most be as aggressive as a newcoming TCP flow.

CM (Congestion Manager) [Balakrishnan et al., 1999] can be seen as a direct evolution of this kind of approach as it envisions the use of a single entity inside each host responsible to maintain the congestion state of all the host's flows whatever their transport protocol is. It would then be able to set their respective sending rates through a rate-shaping scheme. It also allows benefit from the knowledge that several connections are sharing the same path so as to limit contention between them.

The limit of this class of solution is mainly that there are based on mathematical models that are defined under restrictive conditions that may not apply for the context under which the protocol is actually used. It is then necessary to add safeguards to the protocol. For instance, a lot of effort has been made to make sure that the estimates of the loss rate in TRFC are robust to avoid instabilities.

Limit type	Limit	Solution
Application	Multimedia	Transport protocols other than TCP
	Web	Slow Start modifications,
		Other layer modifications
Technology	End-hosts hardware	Wait for next generation hardware,
		Other layer modifications
	Wireless	Transport protocols other than TCP
	LFN	Other layer modifications,
		Congestion Control modifications

TAB. 1.IV – Summary of known TCP limits and existing solutions

Solution Class	Solution	Main idea	Limit
Other Layer mod.	MTU size	Reducing protocol overhead	Availability in hardware
	Parallel streams	Flow multiplexing	Application modification
Other than TCP mod.	UDP	"Removing" congestion control	Memory/CPU overhead
	DCCP	Unreliable message-transfer	Limited implementation
	SCTP	Multi-homing, multi-streams	Not widely used
Congestion Control mod.	Slow Start	Faster increase rate	Little impact for large flows
	Loss-based	Faster increase rate	Unfairness
	Delay-based	Prior congestion detection	Inaccuracy
	Hybrid	Association loss/delay-based	Instability
	Router-assisted	Equipment feedbacks	Availability in hardware
	Equation-based	Mathematical models	Limited models

TAB. 1.V – Summary of main ideas used by solutions to TCP defiency problems and their limit

Conclusion

This chapter has shown that plenty of solutions (about thirty are presented in this document, but it is only the tip of the iceberg) have been introduced to solve some of the limits of the legacy TCP, especially to bypass the problem of scalability in performance in the LFNs. Table 1.IV summarizes the various approaches that can be used to solve a given problem. Table 1.V lists the main ideas and limits that have been used in order to overcome TCP defiency problems. It also highlights some of the limits of each of these approaches. Table 1.VI summarizes the Congestion Avoidance technique that is used by each transport protocol variant presented in this chapter. The trend seems to be towards solutions that are using multiple information sources or that are collaborating with network equipments in order to achieve a more efficient use of the network.

With so many alternatives proposed, some even coming in different versions with slight modifications of the parameters used and new introduced every year, one can wonder how the networking community can ensure that these solutions will preserve the five properties of TCP – scalability, stability, fair-sharing, robustness and RTT-clocked – if deployed in the Internet or in an independent network. How can a network provider ensure that a newly-prevalent solution would not completely disrupt a network that it had provisioned assuming a classical main TCP usage ? How can an end-user identify the right solution that will answer to the data movement constraints in his specific case ?

All these questions put forward the need for evaluation methods to discriminate between all these solutions and quantify if they are adapted to a given usage in a given context.

Transport Protocol	**Year**	**Loss-based**	**Delay-based**	**Router-assisted**	**Rate-based Equation-based**
UDP	1980				X
TCP	1988	X			
TCP-Vegas	1994		X		
TCP-ECN	1994			X	
CM	1999				X
TFRC	2000				X
SCTP	2000	X			
UDT	2001				X
XCP	2002			X	
FAST-TCP	2003		X		
Highspeed TCP	2003	X			
Scalable-TCP	2003	X			
BIC-TCP	2004	X			
H-TCP	2004	X			
DCCP	2004	X			X
CUBIC	2005	X			
RCP	2005			X	
RCP-AC	2006			X	
PCP	2006		X		
TCP-Illinois	2006	X	X		
Compound-TCP	2006	X	X		
Yeah	2007	X	X		
PCN	2008			X	
RAPID	2009		X		
Relentless TCP	2009			X	

TAB. 1.VI – Summary of the Congestion Avoidance technique used by the presented Transport Protocol variants

— Two —

Transport Protocol Evaluation Principles

Introduction

The goal of this chapter is to present the principles behind the process of evaluating a transport protocol and how different points of view can modify the focus of the evaluation. We identify three different kind of actors : the network provider, the protocol designer and the end user.

Each of them have different objectives and have different questions for which they seek to get answers to through the evaluation of transport protocols. After defining the system where the transport protocol interacts (the network model) and defining what happens within it (the workload model), the answers will be provided through the use of metrics adapted to the particular question of the user.

In this chapter, we will first introduce the general principles of performance evaluation in Section 2.1. The three different points of view corresponding to the three different kind of actors are described in Section 2.2. Section 2.3, 2.4 and 2.5 then detail the metrics, the network models, and the workload models that are used in the process of evaluating a transport protocol. And more importantly how they relate to the three different kind of actors.

2.1 General principles of performance evaluation

In his book "The Art of Computer Systems Performance Analysis" [Jain, 1992], Jain introduces in 1992 a series of 10 steps that defines the necessary stages that should take place during the process of performance evaluation. The goal is to provide a systematic approach to avoid common mistakes in performance evaluation such as not "defining properly the goals of the study" or "applying a non-representative workload to the system". These mistakes often lead to false results or results that are too biased to be of use.

Making explicit and answering the simple questions "for whom ?", "what ?", "why ?" and "how ?" are the keys of performing performance evaluation. Table 2.I lists these ten necessary steps.

In the present document, we consider the performance evaluation of a communication process. The system is then the network (or a subset of the network) in which the communication process takes place. The metrics, the network model, and the workload model used in the process of evaluating a transport protocol can be defined as follows :

Metrics : It is a way to quantitatively characterize the properties that need to be studied. It expresses the goals of a given actor or a given property that is expected from a transport protocol.

Network Model : It is a description of the network infrastructure used for the communication process. It can be characterized by the spatial organization of the equipments

1) State goals and define system boundaries
2) List system services
3) Select performance metrics
4) List system and workload parameters
5) Select factors and their values
6) Select evaluation techniques
7) Select workload
8) Design experiments
9) Analyze and interpret data
10) Present results

TAB. 2.I – Steps for a systematic approach to performance evaluation

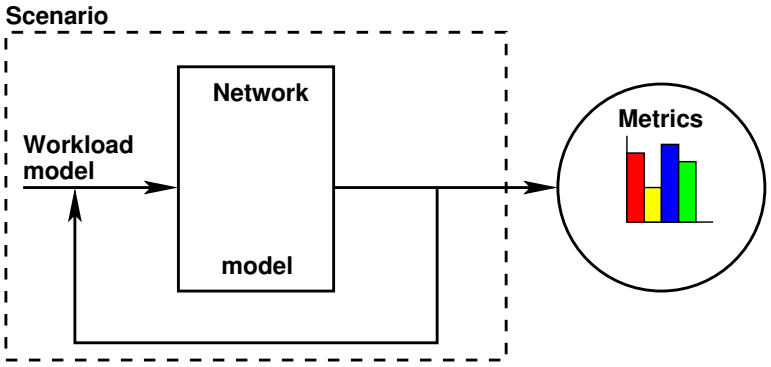


FIG. 2.1 – Interaction of the workload model, network model and metrics in an evaluation process

composing the network (a topology, often represented by a non-oriented graph) and their physical capabilities (*e.g.*, capacities, RTT). For the purpose of evaluating a transport protocol, the network model is the End-to-End path that is used for the communication process.

Workload Model : It is a description of the dynamic usage of a network infrastructure. It can be characterized by distribution laws, *e.g.*, to describe the ON and OFF times of the emitting sources or by type of applications. It can also be divided into two categories : the one relevant to the user (*e.g.*, the traffic generated by the user and forwarded throughout the End-to-End path) and the cross-traffic (*e.g.*, the background traffic generated by other users that partially crosses the End-to-End path).

Figure 2.1 shows how the network model, the workload model and the metrics relate to each other in an evaluation process.The combination of the network model and the workload model creates a scenario that describes the behavior of the system. The book "High Performance TCP/IP Networking" [Hassan & Jain, 2004] provides examples of performance evaluations of TCP. These different aspects and how they get instantiated in the specific case of a type of user is detailed further in the next sections.

2.2 POINTS OF VIEW

Different actors have different uses of the network. Each actor has different expectations from the network, which leads to different needs during the performance evaluation process. Each of these different expectations correspond to a different point of view or actor. We identify

three kinds of actors : the end-user, the protocol designer and the network provider. Each of them have an active part in a communication process over a network and each of them has a different goal.

Network provider : it is an entity that maintains and operates the physical means used for the communication (*e.g.*, optic fibber) between different communicating entities over a long distance. The goal of the network provider [Shenker, 1995] is to ensure that its network infrastructure is fit to accommodate a given number of customers and provide them a good enough Quality of Service to keep them satisfied. He might also consider load balancing and security issues – *e.g.*, detection of flash crowds, detection of Distributed Denial Of Service attacks (DDoS)) –, so as to avoid a catastrophic collapse of a part (or the totality) of his network infrastructure. The network provider focuses on the network cloud.

Protocol designer : it is an entity that produces algorithms to be used for the purpose of communication between two end-hosts over a network. In the context of this dissertation, it designates people proposing new schemes as seen in Chapter 1, Section 1.3 to solve the performance deficiencies of TCP. The goal of a protocol designer is to ensure that the protocols he designs are suited to be used in a shared environment, according to the principles described in the Congestion Control Principles [Floyd, 2000], *i.e.*, preventing congestion collapse, being fair to best-effort traffic (also called "being TCP-friendly"), and optimizing performance regarding throughput, delay, and loss. Under specific circonstances like optimizing the performance of an application in a particular environment (*e.g.*, private network, lossy network), he might develop protocols that forsake some of these principles. The protocol designer's point of interest will be the Transport layer.

End-user : it is an entity that wants to communicate over a network between a local resource (*e.g.*, end-user computer) and a distant resource (*e.g.*, data server, web server) using a given application (*e.g.*, bulk data transfer, web browsing). The goal of the end-user is to get the best performance out of the application he plans to use. In this respect, the end-user's behavior might be selfish and he might choose solutions that will apparently yield the best performance for him. This can lead to the problem known as the "Tragedy of the Commons" [Hardin, 1968] where it is shown that a shared resource can be destroyed if users are driven by their own selfish interest. It might not occur if a responsible common administration is set or if the users collaborate, which is typically what a fair transport protocol like TCP wants to achieve. The end-user mainly focuses on the Application layer.

Figure 2.2 presents the different types of actors and which part of the Internet Reference Model their interest lies in. As the protocol designers work at the level of the transport layer, at the interface between the end-users and the network providers, they may have a conciliatory role.

This section shows that the three different kinds of actors have different goals that are sometimes conflicting. The most notable example is what might oppose a end-user and a protocol designer. Typically, a selfish end-user might have no consideration for fairness if it means getting faster transfer times. It is not uncommon for application developpers to get flamed by the community of protocol designers and the IETF for providing an easy access to potential congestion-collapse-causing transport protocols to a wide range of end-users. In recent years, it happened several times. Two such examples are when the Linux kernel developers switched the default transport protocol to BIC-TCP and when BitTorrent developpers (a popular Peer-to-Peer application) wanted to change the transport protocol used to an UDP-based protocol. Prior performance evaluation of the impact of these modifications could have helped calm the discussion down.

The following sections will now present the metrics, the network model, and the workload model that are generally used when performing a transport protocol evaluation and how they

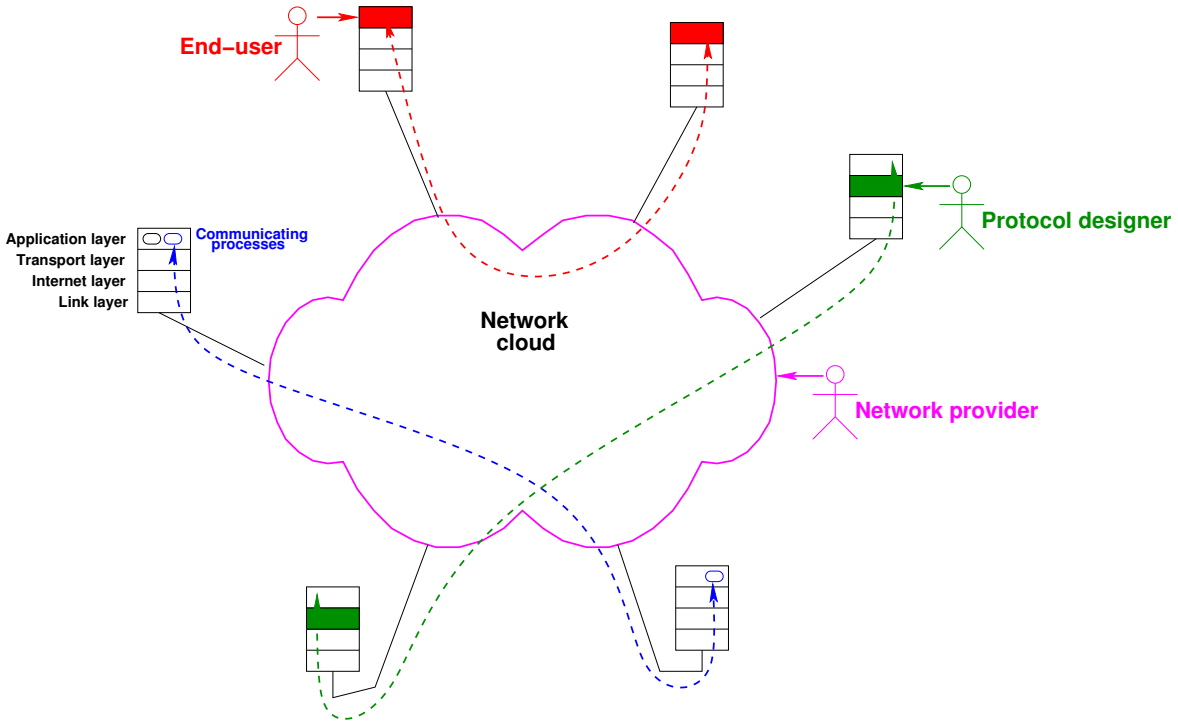


FIG. 2.2 – Type of actors and their relation with the Internet Reference Model

instanciate for each type of actors.

2.3 METRICS

The Transport Modeling Research Group (TMRG) of the Internet Research Task Force (IRTF) has been chartered to produce a series of documents on Models for the Evaluation of Transport Protocols. The documents include a survey of models used in simulations, analyses, and experiments for the evaluation of transport protocols. The goal of their work is to improve the methodologies for evaluating transport protocols.

2.3.1 CATEGORIES OF METRICS

In [Floyd, 2008], metrics commonly used when evaluationg transport protocols are divided into nine different categories. It is then shown how each kind of metrics can be instantiated differently depending on the kind of actors involved.

Throughput : this kind of metrics characterizes the capacity of the transport protocol to use the available capacity of the network. A file transfer will then have a completion time T_i (the amount of time necessary to complete the transfer), a instantaneous goodput $g_i(t)$ (the amount of useful data transfered averaged over a short period of time) and an instaneous throughput $x_i(t)$ (the total amount of data transfered averaged over a short period of time). These metrics can be aggregated over the N competing flows :

$$G(t) = \sum_{i=1}^{N} g_i(t) \tag{2.1}$$

$$X(t) = \sum_{i=1}^{N} x_i(t) \tag{2.2}$$

The efficiency is defined as the ratio of the useful traffic over the total traffic

$$E(t) = \frac{G(t)}{X(t)} \tag{2.3}$$

Delay : it corresponds to the metrics describing the evolution of the time required by packets to go end-to-end in the network. It can be represented by the RTT or by the evolution of the queuing delay **q**.

Loss : it represents the amount of packet losses (and/or its consequences) faced by the transport protocols due to the current networking conditions. It can be expressed through the packet loss rate **p** or the number of timeouts **t** that are experienced by the flow.

Reactivity : this category of metrics [Floyd et al., 2000, Yang et al., 2001, Tsao et al., 2007] expresses the way the transport protocol is able to react to sudden changes in traffic conditions, *e.g.*, flash crowds. Responsiveness **R** measures how fast the transport protocol decrease its sending rate when there is a step increase of network congestion. Aggressiveness **A** measures how fast the transport protocol increase its sending rate to improve network utilization when there is a step increase of available bandwidth. They are usually expressed as the time needed, for instance, to divide the rate by two in the case of the responsiveness.

$$\mathbf{R} = min_u\{u | x_i(t + u * RTT) \leq \frac{x_i(t)}{2}\} \tag{2.4}$$

$$\mathbf{A} = max\{x_i(t + RTT) - x_i(t)\} \tag{2.5}$$

Stability : it regroups the metrics responsible for evaluating the stability of a transport protocol through the study of the delay jitter or the oscillations of the throughput around its optimal functioning point. It is typically the variance :

$$\sigma_i = \sqrt{\frac{1}{T_i}\sum_{t=0}^{T_i}(g_i(t) - \overline{g_i})^2} \tag{2.6}$$

(for the goodput of a flow). Smoothness **S** measures the small sending rate variations over time for a particular flow in a stationary environment.

$$\mathbf{S} = max|x_i(t) - x_i(t + \delta_t)| \tag{2.7}$$

The goodput coefficient of variation :

$$CoV_i = \frac{\sigma_i}{\overline{g_i}} \tag{2.8}$$

allow to characterize the oscillations of the goodput around its mean value.

Fairness : this kind of metrics [Mo & Walrand, 2000, Denda et al., 2000] characterizes the capacity of flows from a given transport protocol to share the available capacity of the network among themselves (intra-fairness) or with flows using a different transport protocol (inter-fairness). One example is the Jain index [Jain et al., 1984] :

$$\mathbf{J} = \frac{(\sum_{i=1}^{N}\overline{g_i})^2}{N(\sum_{i=1}^{N}\overline{g_i}^2)} \tag{2.9}$$

that indicates how the allocation of goodput is made between the different flows. If all flows get an equal share, J is equal to 1. In the worse case-scenario (one flow get all the bandwidth), the "fairness" is equal to $\frac{1}{N}$. Other types of fairness metrics include

Max-Min Fairness, Proportionnal Fairness or α-fairness [Mo & Walrand, 2000]. Some recent works [Briscoe, 2007] developped the idea that a concept of fairness based on flow-rate is fundamentally broken as it can be easily exploited through the use of multiple parallel streams or using long-lived TCP sessions (these are techniques used in most current-day Peer-to-Peer applications). Other measures of fairness need to be used to solve these issues, like cost fairness (making the users "pay" for the congestion they are generating [Briscoe, 2009], for instance).

Convergence : it represents the metrics expressing how the transport protocol is able to reach the optimal functionning point in given conditions. It can be expressed, for instance, as the time necessary for the congestion window size to reach a given threshold (*e.g.*, 80 %) of the BDP.

$$\mathbf{s_{cwnd_i}} = min_u\{u|cwnd_i(t + u * RTT) \leq .8 * BDP\} \tag{2.10}$$

Robustness : it characterizes the capacity of the transport protocol to resist dramatic traffic conditions (*e.g.*, huge packet loss rate, huge delay in the network). It is different from the Reactivity category in the sense that the former is the study of a transient state.

Deployability : it presents the level of difficulty involved when deploying the given transport protocol in an existing environment, *e.g.*, a network infrastructure. It might be difficult to express these kind of metrics numerically, but it is always possible to consider the number of code lines that needs to be changed or the cost of upgrading an equipment.

N corresponds to the number of flows that are competing for the same resource. It is usual to add a qualifier to fully define the metrics : it might be more interesting to get the average goodput rather than the maximum instantaneous goodput in a study of bulk-data transfers. Likewise, the maximum instantaneous throughput might be more useful than the average throughput to understand interactions between flows.

In [Low et al., 2005], Low *et al.* use only throughput, queuing delay and loss as metrics, but they are categorized into three groups : equilibrium, stability and responsiveness according to the kind of qualifier used. Equilibrium metrics characterize the long term evolution of the flow and are expressed as aggregated and averaged metrics like the aggregate mean throughput

$$\overline{X} = \sum_{i=1}^{N} \overline{x_i} \tag{2.11}$$

Stability metrics characterized the oscillations of the system variables and are based on the coefficient of varition of metrics :

$$\overline{CoV_X} = \frac{\sqrt{\frac{1}{T-1}\sum_{i=1}^{T}(x_i(t) - \overline{x_i})^2}}{\overline{x_i}} \tag{2.12}$$

Responsiveness metrics characterize how fast the transport protocol can adapt to a change and is expressed as the time needeed by the protocol to reach a range of 10 % around the equilibrium value :

$$\overline{R_X} = max_t\{t : |\frac{\sum_i x_i(t) - \sum_i \overline{x_i}}{\sum_i \overline{x_i}}| > 0.1\} \tag{2.13}$$

Some of these properties are fundamentally conflicting : it might be easy to provide a fast converging solution (*i.e.*, sending directly at the maximum possible rate) but it is likely that it would be at the expense of fairness. The real difficulty of designing a new transport protocol, especially in the context of the LFNs, is to find the right trade-off between all these properties, that is likely to solve one given problem.

Metric type	Metric name	Formula
Throughput	Aggregate goodput	$G(t) = \sum_{i=1}^{N} g_i(t)$
	Aggregate througput	$X(t) = \sum_{i=1}^{N} x_i(t)$
	Efficiency	$E(t) = \frac{G(t)}{X(t)}$
Reactivity	Responsiveness	$\mathbf{R} = min_u\{u \vert x_i(t + u * RTT) \leq \frac{x_i(t)}{2}$
	Average responsiveness	$\overline{R_X} = max_t\{t : \vert\frac{\sum_i x_i(t) - \sum_i \overline{x_i}}{\sum_i \overline{x_i}}\vert > 0.1\}$
	Aggressiveness	$\mathbf{A} = max\{x_i(t + RTT) - x_i(t)\}$
Stability	Goodput variance	$\sigma_i = \sqrt{\frac{1}{T_i}\sum_{t=0}^{T_i}(g_i(t) - \overline{g_i})^2}$
	Throughput smoothness	$\mathbf{S} = max\vert x_i(t) - x_i(t + \delta_t)\vert$
	Goodput CoV	$CoV_i = \frac{\sigma_i}{\overline{g_i}}$
Fairness	Jain Index	$\mathbf{J} = \frac{(\sum_{i=1}^{N} \overline{g_i})^2}{N(\sum_{i=1}^{N} \overline{g_i}^2)}$
Convergence	Convergence speed	$\mathbf{s_{cwnd_i}} = min_u\{u \vert cwnd_i(t + u * RTT) \leq .8 * BDP\}$

TAB. 2.II – Examples of metrics and their formulation

2.3.2 METRICS AND POINTS OF VIEW

2.3.2.1 NETWORK PROVIDER

Given his goal of properly managing a network infrastructure, it is necessary for the network provider to have access to metrics that allow analyzing the usage of the network infrastructure.

For example, a network provider will be interested to know that its main backbone link has a 95 % average utilization, showing that the existing network infrastructure is efficiently used. But if at the same time, it shows that there is a 10 % average loss rate, the network provider might consider that it is bad for the Quality of Service of the customers and might start planning to increase the capacity of its links or further investigate about possible technical anomalies in the networking equipments.

2.3.2.2 PROTOCOL DESIGNER

The metrics used by the protocol designer often reflect the goals of designing a transport protocol following the Congestion Control Principles. These metrics mainly focus on the study of the interactions between multiple flows, *i.e.*, fairness, stability, or speed of convergence. For example, the protocol designer might be satisfied to know through the use of the Jain index [Jain et al., 1984] that all flows are getting an equal share of the bandwidth. It might not be the case if it appears that in some conditions (a path with a long RTT for instance), the transport protocol will take an infinite amount of time to reach an equilibrium point.

2.3.2.3 END-USER

Depending on the application used by the end-user, different sets of metrics can be of interest but all of them share the same principle : "the faster, the better". If the end-user wants to perform a bulk-data transfer, then throughput metrics like the completion time can be used as it will represent the time he will have to wait to get his file. For delay-sensitive applications, delay-related metrics (for instance the RTT for a multiplayer game) or stability metrics (for instance delay jitter for VoIP) are more suited.

Table 2.III illustrates how the nine metrics types can be used by each type of actor to evaluate the characteristics of a given transport protocol. Each metric expresses a required property of the transport protocol according to a given user-goal.

Metrics of	End User	Protocol Designer	Network Provider
Throughput	Goodput **G**, Completion time **T**	Cong. window **cwnd** Throughput **X** Efficiency **E**	Throughput **X**, Link utilization **U**,
Delay	RTT	Queueing delay **q**	Queueing delay **q**
Loss	Timeouts events **t**	Packet loss rate **p**	Packet loss rate **p**
Reactivity	Aggressiveness **A**	Responsiveness **R** Aggressiveness **A**	Aggressiveness **A**
Stability	Variance σ	Variance σ Smoothness **S**	Coeff. of Variation **CoV**
Fairness	Delta-fair convergence δ_f	Jain Index **J**	Max-min, Proportional, α-fairness
Convergence	N/A	Convergence Speed **s**	N/A
Robustness	Retransmission **r**	Performability p_S**[1]**	Kolmogorov-Smirnov statistic θ_w**[2]**
Deployability	Application modification	Code complexity	Hardware upgrades complexity

[1] : [Meyer, 1980]

[2] : [England et al., 2005]

TAB. 2.III – Three different visions for metrics

2.4 NETWORK MODELS

The network model corresponds to the representation of the environment in which the evaluation process takes place. It is usually composed of two parts : a topology materializing the spatial organization of the entities involved in the communication process and the physical characteristics of the network itself.

Figure 2.3 presents a simplistic model of the concentric organization of the network. Four levels are identified :

Local Area Network : Covering a small physical area, this corresponds to a network that interconnects several computers from the same administrative entity (*e.g.*, a university, a company). It is connected to the access network thanks to an Internet Service Provider (ISP).

Access Network (WAN) : This is the intermediary network that links a customer (*e.g.*, end-user, company) to the backhaul network.

Backhaul Network (MAN) : This network collects the traffic from the access networks.

Core Network : The heart of the network, it aggregates a large number of backhaul networks and allows the transit of traffic from one point to another in the world.

This model can be easily mapped, for instance, on the images generated[1] by the CAIDA project, that studies practical and theoretical aspects of the Internet's infrastructure, behavior, usage, and evolution[2] [claffy et al., 2009]. Depending on how the studied topology overlaps with these different levels, the network may have different properties, especially in the range of the parameters encountered.

[1] see http://www.caida.org/research/topology/as_core_network/pics/ascore-simple.2008_big.png
[2] see the CAIDA project page http://www.caida.org/home/about/

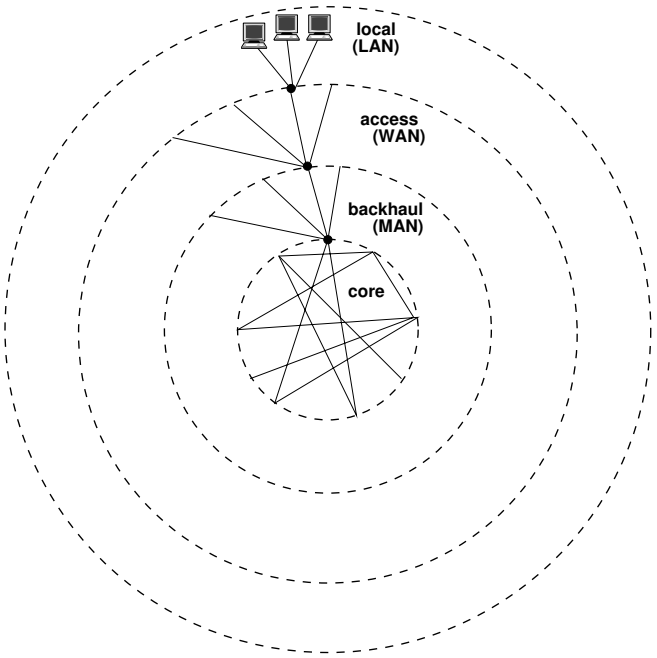


FIG. 2.3 – Concentric network borders

For instance, if a topology is fully enclosed in a Local Area Network, the RTT is likely to be smaller than 1 ms. In a topology crossing over through a Core Network, the RTT might well be over 300 ms. It is also important to note that there is no obligation for a link to be symmetric. It can occur due to routing choices at the IP level or due to the intrinsic characteristics of a protocol or of a physical medium. The typical example is ADSL (the 'A' stands for asymmetric) where uplink capacity is lower than downlink capacity. This choice was made by providers that estimated that users would mostly perform downloads rather than uploads, so they allocated a larger part of the modulation spectrum to the downlink.

The bottleneck of the network model is likely to be one of the points where the traffic gets aggregated at the boundary between two levels of the network. Currently, the bottlenecks are probably located at the entry of the access network as core networks are generally over-provisionned.

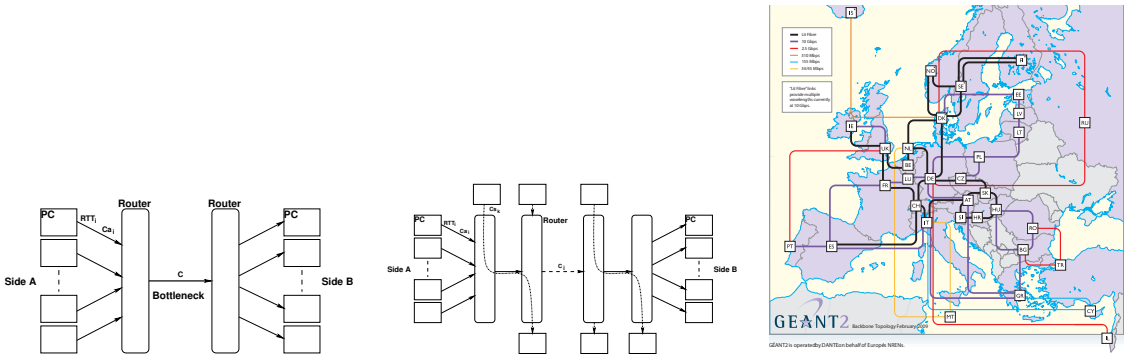
2.4.1 TOPOLOGY

Describing formally a topology is not an easy task [Poole et al., 2008], even though its representation can be fairly straight-forward. The problem gets even more complicated as soon as the scale of the network increase. Over the years, several proposals have been made to provide a topology description language [van der Ham et al., 2006, Koslovski et al., 2008]. The OGF-NML Workgroup is currently trying to combine these various efforts to propose a standardized network description : Network Mark-up Language (NML)[3].

Figure 2.4 presents typical topologies that can be used for evaluating transport protocols. In [Ha et al., 2006], Rhee states that the dumbbell topology (see Figure 2.4(a) is a good topology to perform experiments. The first reason is that it is impossible to reproduce the complexity of a topology of the scale of the Internet. The second is that we can reproduce what is happening in a router of the core of the Internet by choosing appropriate background traffic loads.

In the context of computational grids and data centers, a dumbbell is also a good local topology. The aggregation and congestion of the flows is very likely to occur in the switches

[3]See the OGF-NML Workgroup's webpage https://forge.gridforum.org/sf/projects/nml-wg



(a) Dumbbell topology (b) Parking Lot topology (c) Géant2 topology

FIG. 2.4 – Typical topologies for network models

or routers at the border of one given cluster/site. The parking-lot topology in Figure 2.4(b) can be seen as an extension of the dumbbell topology. In this case, the flows are required to cross a succession of bottlenecks that are shared by cross-traffic.

Figure 2.4(c) presents the topology used by the GÉANT2 european project which is the high-bandwidth, academic Internet serving Europe's research and education community. Connecting over 30 million researchers with a multi-domain topology spanning 34 European countries and links to a number of other world regions, GÉANT2 is at the heart of global research networking. However, due to the large size and complexity of current-day network topologies, it might not be possible to emulate or to recreate a satisfying topology of the same scale [Floyd & Paxson, 2001].

2.4.2 NETWORK PARAMETERS

Table 2.IV summarizes the parameters that are used to define a network model and provides some typical values that can be found in the Internet. Figure 2.5 identifies their position in the topology.

RTT : the Round Trip Time ; it is the time required for a packet to transit between two end-hosts. It is usually composed of three parts : the propagation delay, the transmission delay and the queueing delay. The propagation delay is static[4] and can not be reduced as it depends on the physical route taken by the packet (*i.e.*, bounded by the speed of light). It can take values of up to 300 ms when land-lines are used and up to 800 ms when satellite routes are used.

The transmission delay is the time needed to push the packet into the network and depends on the bit-rate of the link. For instance, for a 1 Gbps link and 1500 B packets, it takes $12\mu s$ to transmit them. The queueing delay can be highly dynamic and corresponds to the delay introduced when packets are queued. The RTT can be provided as a decomposition of the individual one-way delay of every individual links or directly as the RTT of a given path.

C : the capacity of the bottleneck link ; it corresponds to the maximal value above which the aggregated bandwidth of all flows crossing this link cannot go. It can be either a physical limit (*e.g.*, due to the kind of physical medium used) or a limit enforced by some network equipment (*e.g.*, QoS enforcement policies). Depending on the location of the bottleneck, typical values can be anywhere between 56 kbps (the bottleneck is the user's modem) to 40 Gbps (the bottleneck is the core network).

[4] provided there is no routing change

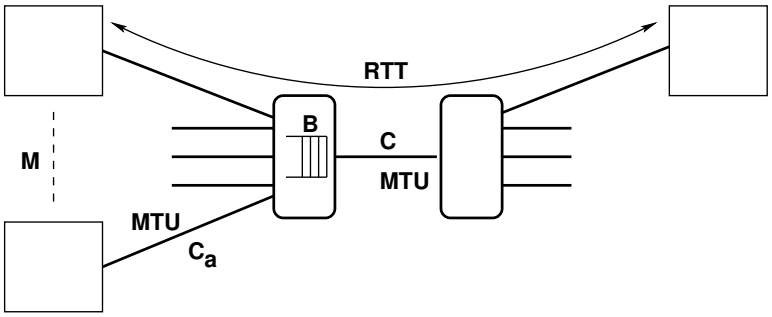


FIG. 2.5 – Network parameters

B : the buffering capacity of the bottleneck ; it corresponds to the number of packets that can be temporary stored when the bottleneck capacity is exceeded. It is necessary to have some buffering capacity in the intermediary equipments so as to provision for the bursty behavior of transport protocols. Typical values are usually expressed as a fraction of the BDP of the path. For instance, high-end Cisco routers can have enough buffering capacity for a path with 100 ms RTT. In low-end switches, it can be only a few tens of packets.

K : the aggregation level (or K factor) ; it is defined [Crowcroft et al., 2003, Srikant, 2007] as the ratio between the uplink/downlink's capacity C and the access link's nominal capacity C_a : $K = \frac{C}{C_a}$, typically at a network border (see Figure 2.3). In DSL context and more generally in the Internet, it is common to have K ranging over 1000 as users have up to 25 Mbps access uplinks and up to 10 Gbps as the Internet providers bottleneck. A lower K factor means that a few flows can congest the link. It has been shown to be an important factor to be taken into consideration when sizing buffer routers. Indeed, if K is large (more than 200), small buffers will not affect much the average flow completion time [Lakshmikantha et al., 2008].

M : the multiplexing factor is the number of contributing sources (computers or links) that are sending traffic through a given bottleneck. For instance, TCP is known to behave better when the multiplexing factor is high, as the statistical bandwidth sharing works better [Bonald et al., 2002]. In the core of the Internet, it can reach large values (greater than 100). In data centers, it can be significantly smaller (lower than 100).

MTU : the maximum transmission unit is a parameter that describes the maximum size of a packet crossing a given link. If a packet is bigger than the MTU, it will be either cut into smaller fragments (fragmentation) or more likely dropped by the networking equipment. For the rest of this dissertation, it is assumed that all links have the same MTU or that a path MTU discovery [Mogul & Deering, 1990] mechanism is available to prevent fragmentation. By default, in Ethernet networks, the MTU is 1500 B.

It is also important to note that the kind of hardware used can be of importance. For instance, a switch produced by Cisco will not be strictly equivalent to a switch produced by Alcatel because there are probably not using the same architecture, hardware and firmware. For the same reason, the hardware of end-hosts can have an impact on the performance of a transport protocol. It is why sanity checks (calibration tests and careful tuning) should be executed before any test involving real equipments.

2.4.3 Network Model and Point of Views

In this section, we present the main differences that exist between the three kinds of users in the way they relate to the studied network model when they are considering a path used for the

Parameter	Description	Typical range in the Internet
RTT	Round Trip Time	0 to 300 ms (land-line) 0 to 800 ms (satellite)
C	Bottleneck capacity	56 kbps to 40 Gbps
B	Buffer size	20 % BDP to 100 % BDP
$K{=}\frac{C}{C_a}$	Aggregation level	1 to 10000
M	Multiplexing factor	1 to 1000
MTU	Maximum transmission unit	1500 B to 9000 B

TAB. 2.IV – Parameters defining the network model

communication of two end-hosts. For each case, the cross-section of the network (Figure 2.6) that is of interest for them is detailled.

2.4.3.1 Network Provider

The network provider is only interested in the cross-section of the links composing the network he operates. The focus might be on one link or set of links in a specific part of his network or on the networking equipments bridging them together. It is also possible that he might consider the points where traffic is entering or leaving his control area (*e.g.*, peering points) as the traffic's balance there might have an impact on the economic relationship he has with other network providers.

2.4.3.2 Protocol Designer

The protocol designer (Figure 2.6(b)) is more interested, as seen in the Metrics' section, in studying the interactions between different flows. For this reason, the protocol designer's focus for the network model's topology will be centered on the different points where the transport protocol's flows are aggregated with other traffic.

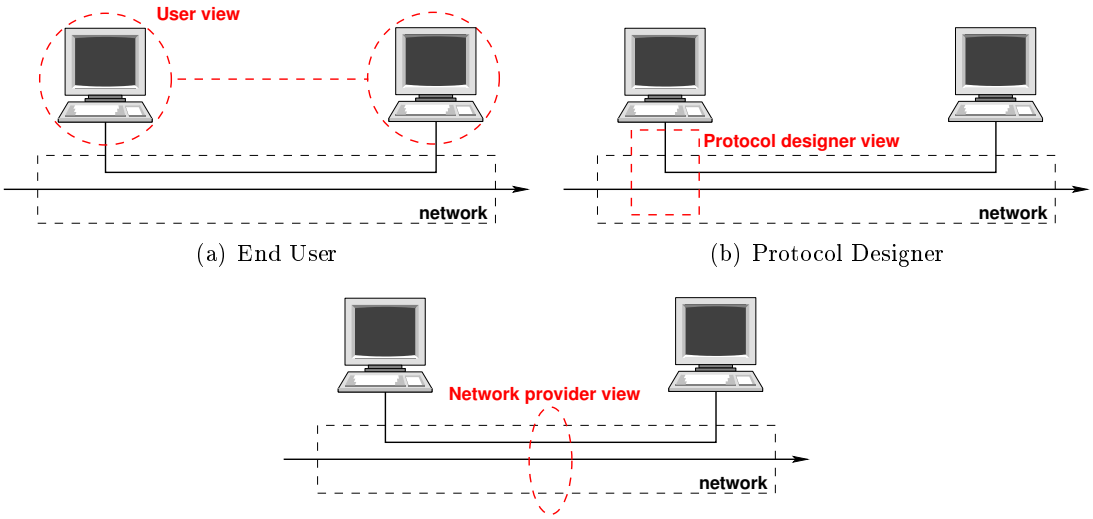
It can be at different levels of the hierarchical organization of the network : the point of aggregation could be a switch in a Local Area Network, a router at the access border or a router in the core of the network. At each of these different levels, the transport protocol's flows will be facing different kind of traffic conditions.

2.4.3.3 End-User

Figure 2.6(a) presents the vision of the network model for the end-user. The end-user has no insight on the actual topology that is used at a physical level. From his point of view, only the (virtual) end-to-end path is of interest.

2.5 Workload models

The workload model in a networking environment corresponds to the traffic that transits through the network model. It can be generated by the object of the evaluation process or by other users (in this case, it is usually called adverse or background traffic). Usually two kind of workloads are considered : real workloads corresponding to traffic generated by real applications (if such applications can be used at will) and synthetic workloads that replicate the behavior of real applications by using distribution laws representative of its characteristics. [Floyd & Kohler, 2008] presents some typical scenarios that have been used to evaluate transport protocols.



(a) End User (b) Protocol Designer

(c) Network Provider

FIG. 2.6 – Three different visions for the topology of a end-to-end path

2.5.1 WORKLOAD PARAMETERS

It is quite complex to define representative workloads of a given universe of real workloads in the case of transport protocols. Exploiting the results and analyses of flow and traffic measurement is a way to solve this issue. The commercial Internet traffic is generally captured and analyzed by network providers who rarely make this data available. Several studies on academic networks or on the Internet have been conducted by researchers to characterize and model the mix of applications, packet and flow arrival processes as well as packet and flow size distributions[Willinger et al., 1998, Feldmann et al., 1998, Willinger et al., 1997, Fomenkov et al., 2003].

In terms of applications mix, these works highlight that on any Internet link there is always a mix of flows from a variety of applications, carried by various transport protocols, especially TCP and UDP. There is no such thing as a "typical" WAN application "mix" : the dominant WAN applications vary greatly from site to site. For traffic analysis, researchers generally adopt the popular mice/elephants dichotomy, referring respectively to short traffic transfers and elephants to long transfers.

Applications in the Internet generally fall into one of two classes : elastic applications, *e.g.*, web or e-mail, that can tolerate throughput and delay variations ; or real-time applications, that are delay-sensitive (*e.g.*, voice over IP) or throughput-sensitive (*e.g.*, video-on-demand). Recent measurements from an Internet backbone link carrying TCP traffic towards different ADSL areas [Azzouna & Guillemin, 2003] show that the commercial traffic now includes a significant part generated by peer-to-peer applications.

In very-high-speed contexts like in data center-like and grid environments, there are few traffic measurements and characterisation studies [Loiseau et al., 2009]. Thus, assumptions and inferences from the Internet's case have to be proposed. From our own analysis of grid application requirements, we assume that in grids, bulk data transfers for high performance distributed applications constitutes the main part of traffic volume. Distributed and parallel applications also transfer very short messages for inter-process communication with MPI communication libraries, for example. The real distribution and the inter-arrival processes of these elephant and mice are currently not characterised. Most applications use reliable TCP-based transport protocols in this context.

The main parameters used to describe the workload model are as follows :

Parameter	Description	Typical Range in the Internet
C_g	Congestion level	0 to 2.0
R	Reverse traffic level	0 to 2.0
B_g	Background traffic	0 to 0.8
N_s	Parallel streams	0 to 10

TAB. 2.V – Factors defining the workload model

C_g : the congestion level; it is the amount of traffic generated by the user's application or a traffic generator reproducing a given workload on the forward path of a network. It can be expressed as the amount of traffic generated by the sources divided by the capacity of the bottleneck. For instance, a congestion level of 1.0 would be mean that the traffic generated is sufficient to saturate the bottleneck.

B_g : the background traffic level; it is the amount of traffic that is generated by sources outside of the studied object (*e.g.*, other users transmitting over the same network infrastructure). It can be seen as the network's background noise. It usually ranges from 0 to 0.8 of the bottleneck capacity depending on the average load of the link.

R : the reverse traffic level; it is the amount of traffic transiting on the reverse path. It can be defined as the ratio between the reverse-path nodes' nominal capacity and the bottleneck capacity. The reverse traffic is an important parameter as it has been shown [Zhang et al., 1991, Mascolo & Vacirca, 2006] that a congested reverse path will have an impact on the performance of TCP. Indeed, if ACKs are lost or delayed, unnecessary retransmissions might occur.

N_s : the number of parallel streams that a studied flow on a source node is using to transfer its data.

If the application is known then the number of participating sources is enough to define the level of occupancy of the network. If the application is unknown, then the distribution laws for the ON/OFF times of the packets or of the flows are needed to describe the level of traffic in the network. In this case, a definition of the congestion level would be :

$$C_g = \sum_{sources} \frac{C_a}{C} * \frac{\mu_{ON}}{\mu_{OFF} + \mu_{ON}} \tag{2.14}$$

Table 2.V summarizes the parameters that compose a workload model and provides examples of the typical values they can have.

2.5.2 Workload Model and Points of View

In this section, we present the main differences that exist between the three kinds of users in the way they relate to the workload model.

2.5.2.1 Network Provider

What is of interest to the network provider, according to his goal of surveying the usage of his network infrastructure, are the aggregated characteristics of the traffic at a coarse-grained level.

To reproduce the characteristics of the traffic mix entering inside his network, traffic matrices are used. They correspond to matrices that describe the traffic going between two points of a given network using several metrics [Medina et al., 2002a]. Throughput, delay and loss rate are considered to be enough to define such a matrix [Medina et al., 2002b, Chang et al., 2005]. Such matrices are not usually made available by network providers.

Monitoring tools provided in routers like Sflow or Netflow are designed to retrieve metrics like throughput at the IP level. As CPU resources are limited in core-network routers, the measures are sampled and/or averaged over a fixed period of time (a few seconds to several minutes) or a fixed number of packets sent through a given interface.

2.5.2.2 Protocol Designer

The goal of the protocol designer is to study the interactions of a given transport protocol with a range of traffic types and levels. It is thus necessary for him to use a fine-grained description of the traffic through the laws of distribution of the ON/OFF times of the flows. Depending on the actual context of the study, for instance if the transport protocol is meant to be used with a specific application, it might be better to use the application directly. Another solution consists in capturing packet traces corresponding to the application so as to sure that the evaluation experiment is reproducible.

2.5.2.3 End-User

For the end-user, the workload model is mainly characterized by the application he plans to use. All the rest of the traffic generated by other users are considered to be adverse or background traffic. Due to their lack of control over the end-to-end path they are using, end-users typically do not have any information about the characteristics of the adverse traffic other than its interference with the user's application.

Conclusion

This chapter has provided the definition of evaluation process of a transport protocol and shown how the different points of view can have an impact on the system considered for the evaluation. Indeed, the three kind of actors described in this chapter don't focus on the same point of the network models and not on the same kind of workloads. They also are interested in retrieving different types of information through the metrics considered. All is due to the initial point of view that is taken by each of these actors and that is driven by the questions they want to answer related to the performance of a high-speed transport protocol.

The next chapter will detail the different evaluation methods that can be used and extract their avantages and limits. It will then allow to find which type of evaluation method is suited to help a given type of actor answer their questions.

— Three —

Analysis of Transport Protocol Evaluation Methods

Introduction

Chapter 1 has demonstrated that, currently, it is possible to find a large number of alternatives to TCP that aim at solving TCP's limits in the LFNs. Chapter 2 has shown that different kinds of actors exist and that each one of them has different goals and different visions of the environment in which they plan to use a given transport protocol. Over the years, a large number of techniques have been proposed to study the behavior of TCP and its subsequent variants.

It is thus essential to know how a given transport protocol will behave in a given environment. The motivation for that is twofold : the Internet is a resource shared by billions of users and it has become a major tool for a lot of economic actors. A congestion collapse is a long period of time during which the traffic is brought down to a slow crawl as a result of a large congestion event. If one were to happen due to these new TCP alternatives, could have a crippling impact on the economy as a whole.

The second motivation is that since all these TCP variants have been proposed to improve the performance of TCP over LFNs, it is necessary to prove that they are able to do so and that they are able to retain the other characteristics of TCP.

This chapter presents the different methods that have been introduced and, through one specific example, show what this particular method teaches us about transport protocols. Four different methods are generally used to evaluate transport protocols :

Analytical Modeling : Section 3.1 presents the main mathematical techniques that have been used to model transport protocols so as to analyze their behavior in a given context. These technics are based on queuing theory, control theory or optimization theory. In particular, we present closed-form modeling, stochastic modeling, and fluid modeling.

Simulation : Section 3.2 focuses on techniques that are used to reproduce the behavior of a transport protocol fully enclosed inside a computer program. Usually, packet-level simulation and flow-level simulation are considered.

Emulation : Section 3.3 shows the main techniques that are used to replicate the characteristics of networking elements (switches, links) to provide a testing environment in which the infrastructure parameters of the network model are configurable. It is usually done through software or hardware emulation.

Experimentation : Finally, Section 3.4 presents two different methods to perform experimental evaluations in real environments : those performed in controlled environments and those performed in uncontrolled environments.

It is important to note that usually these methods can be used independently but they are often used in a succession of steps during an evaluation process. For instance, it is possible to

start by defining a model that will then be checked against simulations or experimentations. The order in which these steps are taken is user-dependant. A network provider or a end-user is likely to go from a stage where he notices a strange behavior pattern in his logs, then tries to reproduce the problem experimentally, and finally analyzes the conditions and parameters leading to the strange behavior using simulations. On the other hand, the protocol designer might probably spend some time modeling its protocol if possible to avoid performing simulations on parameters that are not relevant to his study.

3.1 Analytical Modeling

This section presents the main mathematical techniques that have been used throughout the years to model TCP. These methods give formulae expressing the behavior of TCP, typically the average throughput, under specific workload and network models. The network model in such studies is usually defined as a network of queues. In some cases, much like the underlying queueing theory that is used by most of these works, it requires numerical solving as soon as the distribution laws used for the losses or the flow characteristics are more complicated than exponential laws.

The three methods considered in the literature are :

Closed-form Modeling : it consists, using mathematical analysis, in deriving the exact formulae for the evolution of a given metric for the transport protocol.

Stochastic Modeling : Probability laws are used to characterize the behavior of the transport protocol.

Fluid Modeling : Partial differential equations are used to represent the evolution of the transport protocol's internal parameters at flow level.

A detailed survey of these analytical methods can be found in [Olsén, 2003].

3.1.1 Closed-form Modeling

Closed-form modeling aims at deriving exact formulae. Usually, it relies on modeling the interactions of the transport protocol with other users and with the network at packet level. It is also possible to derive exact formulae using stochastic and fluid modeling. Mathis [Mathis et al., 1997] introduced in 1997 an analytical model of TCP, expressing the achievable mean throughput of a single TCP flow as a function of the loss rate (p), the RTT, and TCP's maximum segment size (MSS) :

$$R = \frac{MSS}{RTT * \sqrt{\frac{2p}{3}}} \tag{3.1}$$

This model, also known as the p-square-root model, is similar to Padhye's model presented in Chapter 1 with more constrained hypothesis. Both models are equivalent in the case where the rate loss (p) is low (in fact, this formula can be seen as the first-order development, with Padhye's being the development to the second and third order). In this study, the TCP model is restricted to the Congestion Avoidance part of a TCP Congestion Control algorithm, similar to the AIMD.

The authors also make additional assumptions about the behavior of the TCP connection. It is assumed that the flow is in a steady state with enough packets to transmit. The RTT is considered to be constant, which is the case when enough bandwidth is available and when the adverse traffic is not even to cause queue build-ups. The packets are lost due to a lossy link. In the simplest model, the random packet loss p is approximated by a period loss every $\frac{1}{p}$ packets. With these assumptions, the evolution of the congestion window follows a periodic sawtooth pattern. It is then easy to compute the number of packets that are transfered during a cycle and deduce the mean throughput (Equation 3.1) for such a flow. The authors also

TCP Variant	C	d
TCP Reno	1.22	0.5
BIC-TCP	15.5	0.5
HighSpeed TCP	0.12	0.835
H-TCP	0.12	0.835
Scalable-TCP	0.08	1.0

TAB. 3.I – Parameters of the response function for some TCP variants, $R = \frac{MSS}{RTT} * \frac{C}{p^d}$

studied the case where the system was subjected to a constant loss rate in [Ott et al., 1996] and derived a similar expression as Equation 3.1 using a fluid model approach. This model is at the root of TFRC, a rate-based TCP variant designed to be as fair as TCP.

This model was then validated against simulations using the NS-1 simulators and against experimentations over the Internet. For low values of loss rate, the measures seems to fit with the model. Since then, this kind of technique has been extended to include more aspects of TCP's congestion control, especially the Slow Start phase and its impact on short flows or to analyze new TCP variants. But it was often performed using stochastic or fluid modeling and does not usually lead to as a simple expression as Equation 3.1.

3.1.1.1 TCP VARIANTS STUDY

Equation 3.1 presented previously is rewritten to a more general case in Equation 3.2. C, the proportionality coefficient, is a constant that depends on the implementation of the congestion control algorithm (*e.g.*, the AIMD constants, see Table 1.III), on the handling of ACKs on the receiver's side (*e.g.*, delayed ACKs), and the distribution of loss events in the network (*i.e.*, random losses or only due to congestion). d expresses the cost of the transport protocol's response to a loss event. For TCP, when there are no delayed ACKs and losses due to congestion, C equals $\sqrt{\frac{3}{2}}$ and d equals $\frac{1}{2}$ (which is the case for Equation 3.1).

$$R = \frac{MSS}{RTT} * \frac{C}{p^d} \tag{3.2}$$

It is possible to compute the closed-formula for most TCP variants [Xu, 2007], even though a stochastic modeling like the one performed by Padhye is generally used. This approach and the response function is used by protocol designers [Rhee & Xu, 2005, Tan et al., 2006] to enforce a fair behavior towards TCP in low BDP conditions. Indeed the constants used for BIC-TCP and CUBIC were chosen so as to have a lower or equal throughput compared to TCP in the regions with a high loss rate. Compound-TCP's authors selected its parameters so that this particular transport protocol would be as efficient –minus some considerations about integer computation– as HighSpeed TCP in networks with a low loss rate, *i.e.*, in a network without congestion. In [Blanc et al., 2009], the authors show that Compound-TCP's response function is also a function of θ, the parameter used during the constant phase of Compound-TCP's congestion window evolution. Thus, Compound does not fit into the model of Equation 3.2.

Table 3.I summarizes the values of C and d used to model some of the most prominent TCP variants and Figure 3.1 presents the log-scale representation of each of them. CUBIC-TCP has been left out of this summary as it only fits in the model while in the "TCP-mode". For BIC-TCP, the model is only valid if $p < 10^{-4}$.

3.1.1.2 ADVANTAGES

The main advantage of this technique is that it provides a very simple formula to estimate the throughput of a TCP connection with only some of the parameters characterizing the

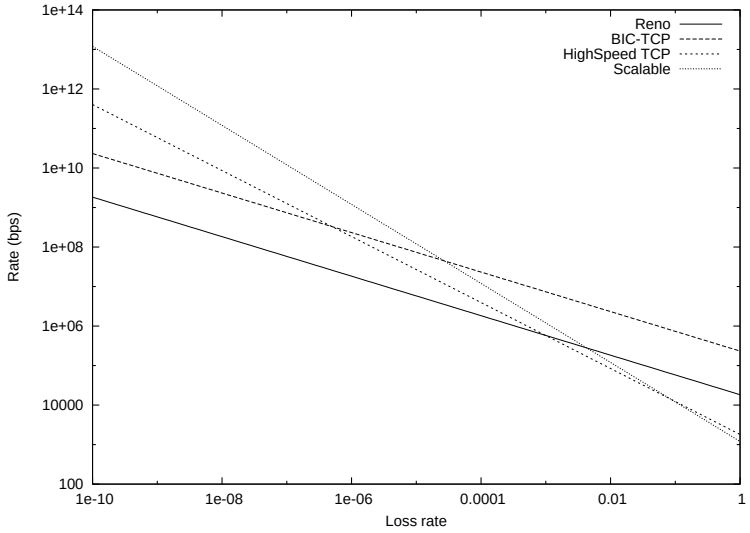


FIG. 3.1 – Comparison of the response functions for TCP variants

system. It then provides a theoretical bound that can be directly checked against the results achieved in a real network. For instance, if a TCP flow using 1500 B packets over a 100 ms RTT path experiences a 5 % loss rate, then the user should not expect to reach a throughput higher than 640 kbps. As this is simple, it is easy for end-users to apply it to their test cases if they are able to retrieve the values of the different parameters used. It is also useful for protocol designers as it provides a simple ground to compare with other TCP variants and to adjust the inner parameters of the transport protocol.

3.1.1.3 LIMITS

However the model makes a number of assumptions on the behavior of TCP that are not necessarily realistic. For instance, the periodic loss model does not correspond to anything that would normally occur in a real network (except in the case of a faulty network interface that resets itself every N seconds). Later works like Padhye's provided similar closed-form formulae for more realistic versions of TCP by taking into consideration aspects like Slow Start. But the formulae quickly get impractical. It might also be difficult to apply this technique for transport protocols that do not have such a regular pattern, *e.g.*, CUBIC, whose congestion window size is increased in real-time, or any TCP variant using delay as a loss detection method.

Finally, such formulae only yield a very aggregated result. The average throughput is a metric usable to study the behavior of transport protocols but it is only one metric out of nine possible kinds and it does not allow studying the interactions between flows (even though the interactions might be limited as such studies usually consider a single flow interacting with a loss model).

3.1.2 STOCHASTIC MODELING

3.1.2.1 TEST CASE

In [Altman et al., 2005], a stochastic model of a single long-lived TCP flow is presented. Its main feature is its capacity to integrate any distribution of inter-loss times. The losses are assumed to be generated by a stationary ergodic random process. The considered model aims at defining the evolution of the transmission rate of a TCP flow. It is also possible to derive higher moments of the transmission rate, including the variance. This model allows getting an expression similar to Equation 3.1 if the random losses are deterministic. It is stressed by the authors that extra care should be taken when defining the loss process. A wrong assumption on the real loss process can lead to under- or over-estimation of the throughput.

Additional conditions are also studied, like the impact of flow control or timeouts on the transmission rate. The model is then validated using real experiments over different Internet paths. Several loss distribution are considered including deterministic, Poisson, general i.i.d (independant and identically distributed), and general correlated losses. It is shown that except in the case of the long-distance path, the loss distribution chosen led to an over-estimation of the throughput and that it is difficult to find the perfect match, even after some correction of the model to take into account the differences between fluid and discrete models.

3.1.2.2 Advantages

The stochastic modeling is a good choice when the interactions of each component of the network follow random laws. It is usually the case when human interactions (*e.g.*, in terms of OFF times) need to be taken into consideration.

In [Bonald et al., 2003], this technique is used to help network providers dimension their access network to accommodate a number of users (*e.g.*, a ADSL provider). Simple formulae are derived to express relations between the bottleneck capacity, the access link capacity, the number of users, the average offered traffic and the average goodput they can achieve. These formulae are valid as long as the number of users sharing the link (*i.e.*, the multiplexing factor M) is large.

Two strategies of dimensioning are presented to accommodate the two possible operating regimes of the network : a transparent regime which corresponds to the case when the flows can all be transmitted at the access link's rate and a saturated regime when the bottleneck is saturated. It is also shown that the results are independent from the exact characteristics of the traffic (*e.g.*, file size distribution, waiting time) and depend only on the offered traffic rate.

3.1.2.3 Limits

The usual limit for this kind of technique is that it might be difficult to find distribution laws that precisely capture the behavior of a particular component of the network. These approximations can lead to underestimating the evolution of some parameters like the average packet delay or the maximum queue size [Paxson & Floyd, 1995]. These kinds of models also often yield results that are only valid for large sets of users, which can be a problem for limited-scaled network models.

3.1.3 Fluid Modeling

Fluid modeling made its appearance in the late 90's [Kelly et al., 1998] and takes its name from an analogy with fluid mechanics. The links are considered to be pipes that are filled by flows. The bottlenecks are represented as dams with reservoirs to hold the excess rate of flow. Like its fluid analog, it assumes that the flows are incompressible and continuous. It is then possible to describe the temporal evolution of the system using partial differential equations.

3.1.3.1 Test case

In [Low, 2003], a fluid model is used to study the interactions between the TCP congestion control and active queue management schemes. After defining the general problem (maximizing the aggregate utility function of the flows under a constraint of bandwidth capacity), it is then instantiated for different types of transport protocols and AQMs. It is shown that a combination of TCP and RED solves the general utility problem. The same model is used to compare TCP Vegas and TCP. It is shown that depending on the network parameters and the AQM scheme chosen, the bandwidth allocation between Vegas sources and TCP sources will differ. For instance, when RED is used, Vegas sources are able to get individual rates greater than Reno sources.

In [Tang et al., 2005], the same model is used to extend the study to the comparison of the intra-protocol and inter-protocol fairness of FAST-TCP and of TCP. It shows that it is possible to change the bandwidth allocation between FAST and TCP sources that are sharing

the same network by varying one of the parameters of FAST. As such, it could be used to make sure that in a network shared by these two protocols, each flow could obtain an appropriate share of the bandwidth. The other result is that mixing FAST and TCP sources does not modify the intra-fairness of the flows using one protocol. This is important as it shows that delay-based and loss-based protocol can coexist with minimal interferences in the network.

3.1.3.2 Advantages

Fluid modeling is particularly suited for the analysis of the stability of transport protocols, as it is able to capture the evolution of the time-series of the important parameters like the queuing-delay or the throughput. This technique is also generally used for the study of delay-based transport protocols. As Fluid Modeling is a flow-level modeling, flow interactions can be easily studied. It was used as a background to validate the behavior of FAST-TCP, especially its stability [Wang et al., 2005].

3.1.3.3 Limits

In [Tang et al., 2008], the authors use a fluid model to link macroscopic properties (like characteristics at a flow-level,*e.g.*, throughput) to microscopic factors (like events that occur at a packet-level,*e.g.*, packet bursts). It starts from an equation describing the evolution of the queueing delay of a FIFO queue at the bottleneck and an equation that captures the RTT-clocking property of the TCP flows to define a model of the congestion window's evolution.

It is then used over two applications. In a loss-based transport protocol, it is shown that the microscopic factor of packet burstiness will have an impact on the steady state throughput achieved. In effect, a paced transport protocol will get a better throughput than an unpaced transport protocol, such as TCP. The model is also applied to the case of FAST-TCP, a delay-based transport protocol. It is shown that instabilities of the congestion window size can occur when flows with different RTTs coexist in certain conditions. This example shows that focusing on the flow-level only is susceptible to generate imprecision in the results as it is likely that factors at packet-level will have an impact. In [Altman et al., 2005], it is also noted that it is not possible for a real system to be perfectly modeled using a fluid model as some discrete limits, like the impossibility of having very small packets may interfere. It is still possible to correct these aspects in the model.

3.1.4 Synthesis

This class of evaluation methods are very powerful. Indeed, it is possible to obtain exact formula or exact numerical solutions to follow the evolution of some of the internal parameters of the transport protocol or of metrics of interest, like the mean throughput.

All these methods have a common limit : restrictions need to be put on the parameters or the number of entities considered to avoid combinatorial explosion. Even if a given user is only looking for a numerical solution, it might take some time to compute if the fluid model used implies solving a system with millions of variables. Such size restrictions might be a problem if the goal is to model the realistic behavior of the transport protocol. It is also possible that the models used to represent the entities involved are not representative of real equipments and are not able to capture all the aspects of the behavior of real transport protocol.

It might also be difficult to solve a specific problem if the probability laws involved are not Poisson or exponential laws, but it is always possible to use numerical solvers to get approximate solutions.

3.2 Simulation

In this section, we present the different methods that have been used to reproduce the behavior of a transport protocol inside a computer program : a simulator. Such a program aims at reproducing a networking environment, *e.g.*, the network model, and the transport

protocol mechanisms by means of mathematical models or, more typically in this case, by implementing the algorithms defining the transport protocol (*e.g.*, Slow Start, Congestion Avoidance). The user then needs to define the simulation setup, *i.e.*, the scenario, and provide the set of parameters and their initial values.

For the study of transport protocols, it can usually be done at two different levels : at packet-level or at flow-level. These two kinds of techniques are focusing on a different network granularity and have different purposes for the kind of users that is targeted by each technique.

3.2.1 Packet-level Simulation

The goal of the packet-level simulation is to reproduce the treatments that are performed on a packet by the networking equipments crossed along the path so as to reproduce as closely as possible what would happen to a packet in a real network. Each entity involved in the simulation is described as an algorithm or as a mathematical model. Simplifications are made to reduce the amount of operations necessary to execute the simulation while trying to keep as close as possible to the reality. For instance, packets can be cut down to the headers and routers using a DropTail can be represented as simple FIFO queues. With such a packet-level approach, it is possible to study the transport protocol at the finest grain possible. However, it is still possible to aggregate the results at a coarser granularity if it is relevant, like at a RTT-level or flow-level.

The typical example of packet-level simulator is NS-2, a discrete event simulator[1] that has been developed since the beginning of the 90's by the LBL and the ICSI. At first meant to study TCP, it has since been extended to add a variety of transport protocols and different physical layers. As a result, NS-2 has been used by hundreds of researchers since its creation. NS-2 is discussed further in Chapter 4, Section 4.4. Another example of packet-level simulator is OMNeT++ [Pongor, 1993][2] which features a large collection of modules and frameworks, including networking stacks allowing the study of transport protocols in a variety of situations.

3.2.1.1 Test case

In the paper defining the CUBIC TCP variant [Rhee & Xu, 2005], Rhee performs some simulations in NS-2 to compare it to other TCP variants. The simulation setup consists of a dumbbell with varying capacities and RTTs. An unspecified background traffic of about 10 % of the total capacity is applied to the network to limit the impact of the phase effect. 3 types of simulations are performed to assess the TCP-friendliness in short RTT-networks, the TCP-friendliness in long RTT-networks, and the stability of TCP variants in LFNs.

For the first two experiments, the bottleneck capacity varies between 20 Mbps and 1 Gbps. The purpose was to verify that their assumption of choosing the parameters used by the CUBIC transport protocol, so that it would be TCP-friendly in low BDP conditions, was correct. At the same time, it was possible to highlight some fairness issues with Scalable-TCP. The last experiment studies the impact of a varying queue size for the stability of the TCP variants over LFNs using the Coefficient of Variation. It shows that the stability improves with the size of the queue, that CUBIC has the best stability, and that H-TCP appears to have short-term stability issues.

In [Mascolo & Vacirca, 2006], Mascolo *et al.* study the impact of adverse traffic on TCP variants in simulations. A dumbbell topology is used with a 250 Mbps bottleneck and two different RTTs. It shows that both H-TCP and Scalable-TCP have a higher retransmission probability that the other TCP variants. It also highlights the necessity of considering reverse traffic as, for instance, if FAST TCP is able to achieve fair sharing in different RTT conditions, it is not able to fully use the available bandwidth when there is reverse traffic. Also all the TCP variants experienced a large number of timeouts and retransmissions compared to those

[1] See the project page for NS-2 http://nsnam.isi.edu/nsnam/index.php/Main_Page
[2] See the project page for OMNet++ http://www.omnetpp.org/

of TCP. It shows that the aggressiveness of these new TCP variants should be taken into consideration.

3.2.1.2 Advantages

The main advantage of this technique is that it only takes a few moments to setup a topology and to define a scenario (assuming that no specific modules for NS-2 or OMNet++ need to be written). The underlying network models used are usually simple as they are mainly interconnections queues and delay lines. As such a network is easy to imagine, it simplifies all the more the interpretation of the results.

The fact that the simulation is performed at packet-level gives a fine-grained control over the network. For instance, it is possible to study the evolution of every variable of the transport protocol when packets are sent and ACKs are received. Finally, it is also possible to modify every parameter of the system independently, without any restriction of range, which makes simulation a perfect tool for protocol designers.

3.2.1.3 Limits

The first limit is that this kind of approach can be very resource-consuming. Depending on the scale of the simulation the user wants to perform, the amount of CPU and memory necessary to execute it grow exponentially with the scale and capacity of the network. In [Riley & Ammar, 2002] Riley *et al.* computes that 4 days worth of computation could be necessary to simulate a second's worth of events for a network of the scale of the Internet. As it is necessary to keep track of every packet in flight in the network and perform computation each time a packet reaches an interconnection point in the topology, PU and memory are critical resources. If the user also wants to keep track of all the events that took place in the network, instead of focusing on a few parameters, it can also require a lot of disk space.

Another limit comes from the fact that packet-level simulators are deterministic by default. Randomness has to be explicitly added by the user, for instance by providing random OFF times at the beginning of the simulation to every source. Otherwise, the result is entirely defined by the initial conditions provided. Being deterministic is not a bad thing. It is particularly reassuring to know that two successive runs of the simulator performed with the same initial parameters will yield the same consistent result, and that another user working at the other side of the globe will also be able to get the same results if he is provided with the initial parameters. The trouble occurs if users do not pay attention to the initial conditions. The typical example is the "phase effect" [Floyd & Jacobson, 1991] problem in NS-2. If several sources are started at the same time, all sources will send their packets and receive their ACKs at the same time if there is no congestion in the network. All sources will then increase their sending rate at the same time. If the router at the bottleneck uses a deterministic drop policy (DropTail for instance), several sources might each experiment a simultaneous drop. In effect, these sources are synchronized and are likely to stay so, which will have an effect in the final result of the simulation. This kind of behavior is less likely to occur during a real experiment as random jitter will be generated by the OS and the other processes present on a computer.

For larger topologies involving thousands of nodes or networks with high-speed links, the number of packets that needs to be handled even over short periods of time can be dramatically huge in simulations at the packet-level. For instance, a single 1 Gbps source can have a packet rate of 83333 packets per second. If there are 100 such sources, then more than 8 million packets per second need to be handled. The time required to perform such simulations is then likely to increase to such large proportions that this technique becomes impractical to use, *e.g.*, a full day worth of simulation for 100 s experiment. In [Wei & Cao, 2006], it has been shown that the simulation time in NS-2 is exponential with the bandwidth. Using a greater level of aggregation, *e.g.*, the flow level, might help to reproduce large-scale LFNs, but as shown in the next section, we will not be able to study the same questions. Other solutions based on

parallelization of the packet-level simulator have been proposed [Riley et al., 1999] to decrease the execution time.

3.2.2 Flow-level Simulation

Flow-level simulation aims at taking into consideration the drawbacks of packet-level simulations when the size of the network model is too large. Deriving from the equations obtained through fluid modeling, flow-level simulation envisions to use a higher level of aggregation than packet-level to reduce the duration of experiments. FSIM [Sakumoto et al., 2007] or NetScale [Baccelli & Hong, 2003][3] are recent examples of such flow-level simulator.

3.2.2.1 Test case

In [Baccelli & Hong, 2002], Baccelli defines a model, called the AIMD model, defining the joint evolution of multiple long-lived TCP flows sharing a common tail drop bottleneck using a fluid model. A synchronization rate, corresponding to the probability of a TCP flow to detect a congestion epoch, is used to express the mean throughput and statistical properties (like variance or fairness) of each TCP flow. This model allows deriving a similar formula to [Padhye et al., 1998, Mathis et al., 1997] as a function of the packet loss rate and of the synchronization rate. The model is then used to simulate more than 100 TCP flows over long periods of time. A fractal analysis is then performed to show that the model presents the same multi-scale properties as real TCP traffic.

The AIMD model is extended in [Baccelli & Hong, 2003] to take into consideration several other factors both in terms of workload model (by extending the model to ON/OFF sources) and in terms of network models (by considering heterogeneous flows over arbitrary network topologies). This more general model called multi-AIMD is then used to study aggregated traffic over a 4-level tree involving millions of flows. It shows that the multiple scale properties of the traffic is insensitive to the level of aggregation considered. Other applications of the flow-level simulator, including analysis of instantaneous bandwidth sharing, and an extension of the model to a more aggregated network-level can be found in [Baccelli & Hong, 2005].

3.2.2.2 Advantages

The main advantage of this method is that it is possible to compute results much faster than when using a packet-level simulator, especially if the scale of the network considered is well above thousands of nodes. For instance, in [Baccelli & Hong, 2003], the considered flow simulator has a cost that is linear with the number of congestion epochs and with the number of TCP flows.

In a comparison with NS-2, it yields a huge speedup (more than 500 times as fast) for the LFN case. As it only yields results aggregated at the coarser grain of the flow or at the level of a bottleneck, it is mostly suited for users that only need very aggregated metrics at some points of the network, namely network providers.

In [Ingham et al., 2009], the authors present a test-case on how to validate the capacity of a Service Delivery Platform (*e.g.*, an infrastructure used to deliver content to customers) using simulations. Such a technique can help networking providers find out the impact of new types of equipments (*e.g.*, a new queueing policy in the routers, a new inter-connexion point in the network) before their actual deployment in the network. It is especially useful if this equipment is costly or if it requires precise tuning.

3.2.2.3 Limits

The main limit of this technique is the fact that it has a limited accuracy compared to the packet-level simulation approach due to the fact that the computation is based on aggregated metrics. Results are only provided at a very aggregated level, which may not be useful for studying fine-grained aspects like the evolution of the queueing delay in a router.

[3] See the company's web page http://www.n2nsoft.com/

Constraints like "the sum of the throughput of each flow can not exceed the capacity of the link" ($\sum_{i=1}^{N} x_i(t) \leq C$) are used to model the interactions of the flows in the bottleneck. But it does not indicate how the actual bandwidth allocation (in terms of fairness) is performed by the transport protocol. The most used allocation model is Max-Min fairness (a bandwidth allocation in which it is not possible to increase the allocation of one flow without decreasing the allocation of another), but Chiu [Chiu, 2000] has shown that TCP does not implement Max-Min fairness. If the wrong assumptions are made in the flow simulator about the bandwidth allocation, then the results may be wrong.

It is also necessary to have a fluid model of every equipment in the network with a sufficient precision so as to replicate a satisfying behavior of the transport protocol in such an environment. But in some cases, it might prove difficult, for instance, to reproduce the exact behavior of a queue that dynamically adapts its size according to parameters like the average queue size of all the other queues of the router.

It is also important to note that, by construction, the packet emission of the TCP sources in a fluid model will be smoothed out over the considered period of time. It is different from the typical behavior in discrete models where the packets are transmitted as bursts at the beginning of each RTT period. In that respect, TCP sources in a fluid simulation are similar to real TCP sources using a pacing technique.

3.2.3 Synthesis

Simulation techniques are very useful for protocol designers as there is virtually no limit over the kind of simulations one can design. It is also possible to change one parameter at a time without affecting the other ones. It is not always the case for other evaluation techniques. If it is needed to modify the size of a queue in a router, it might mean in a real network to buy a new router, which will modify several parameters at the same time.

A common critic against simulators is that the models are always over-simplified (but less than the ones used in analytical modeling) and, as such, the results they yield are unrealistic. It is then difficult to foretell if they can be extended to the real world : a transport protocol might well be able to give perfect fairness and optimal throughput in simulations, it might still be useless in a real situation because the assumptions made over the timestamps' accuracy can not be met on current-day hardware equipments, for instance. It is also very easy to get confused over which parameter are relevant to the study and which are not. Simulations might then not be very useful for end-users as it might be difficult for them to recreate the characteristics of the End-to-End path they want to use.

[Liu et al., 2001] performs a comparison of fluid and packet-level approaches for the execution of simulations over different networking scenarios. It highlights the limits of the fluid approach that can affected by the "ripple effect", when a single change of rate in a flow will cause a cascade of changes of rate in other flows in the system. To reduce this problem, it is possible to aggregate flows even more. It is also noted that hybrid simulation models aiming at combining the accuracy of packet-level simulators and the quickness of flow-level simulators exist [Yan & Gong, 1998].

3.3 Emulation

Some parameters such as the RTT or the queue size of the routers are usually only available in a limited range. It is also very difficult or impossible to modify the firmware of these networking equipments so as to test the effect of a specific queueing policy on a transport protocol for instance. The user might also not have the rights to modify the routers' configurations, even though recent work from McKeown *et al.* [Gude et al., 2008][4] tries to improve network architecture so that users, *e.g.*, researchers, would be able to configure specific treatments for flows

[4] See the project page of the OpenFlow Controller http://noxrepo.org/wp/

crossing switches compatible with the OpenFlow abstraction [McKeown et al., 2008][5]. But it will be a long time before such equipments are widely available, even in campus networks.

It might then be necessary to replicate the behavior of the switches by other means to allow users to explore the behavior of a transport protocol (and its implementation in a real system) in a variety of networking conditions. In this section, we present techniques that are used to duplicate the properties of some elements composing a network (*e.g.*, switches or routers, links) through the use of specific components. Users who want to study the behavior of transport protocols that requires the assistance of network equipments is likely to go through this stage. Indeed, it might be difficult to find a real environment in which the requested assistance is available, especially if the mechanism is new and cannot be derived from the capabilities of existing routers.

3.3.1 Software Emulation

To do so, it is possible to use special software that will reproduce the behavior of a specific networking equipment, network link or network cloud. For instance, it is possible to have an equivalent to a router [Kohler et al., 2000][6] in the GNU/Linux kernel. Three widely used link-emulation tools are Dummynet [Rizzo, 1997], NISTNet [Carson & Santay, 2003] and NetEm [Hemminger, 2005], a built-in network-emulation facility of the Linux kernel's Traffic Control (TC) subsystem. With them, it is possible to increase the latency or to limit the bandwidth. It is also possible to completely emulate a network path or cloud [Agarwal et al., 2005, Sanaga et al., 2009] to create a testing environment with adjustable networking configuration. For instance, the Emulab testbed [White et al., 2002] uses Dummynet in its FreeBSD nodes and TC/Netem in its Linux nodes to allow users to define topologies relevant for their experiments.

3.3.1.1 Test case

In [Leith et al., 2007], Leith *et al.* present a comparative testing of several TCP variants using software emulation. Some of the grounds to justify this approach are that it is necessary to evaluate transport protocols in a broad range of networking conditions to really qualify their behavior. A range of bandwidths, RTTs, router buffer sizes, and mix of connection sizes should be used. It is also noted that only considering one metric is not enough (*e.g.*, throughput) and several metrics should be evaluated at the same time including fairness, responsiveness, *etc.*

The network model used consists of a dumbbell topology with four end-hosts using 1 Gbps Ethernet cards. A Dummynet router is inserted in the middle of the topology to have configurable capacities, queue-sizes, and RTTs. It is then possible to compare the behavior of a given transport protocol for different networking conditions. For instance, Scalable-TCP is shown to have an intra-protocol fairness problem (ratio of the average throughput of two flows experiencing the same RTT) due to a poor speed of convergence. In this study, we can also see that this result depends on the bottleneck capacity. For low bottleneck capacity values (10 Mbps), Scalable-TCP has a fairness that is not so far from other TCP variants most of the time. While at higher bottleneck capacity values (250 Mbps), it has by far the worse fairness of all other TCP variants.

Other scenarios are explored to study other aspects of the behavior of these TCP variants, among which fairness with different RTTs, TCP friendliness, convergence time, and impact of Web traffic. The software emulator also allows to set an arbitrary random loss rate in the network. This feature is used to check the mathematical models of the response function as shown in Section 3.1.1.1 against measurements in simulated conditions. It is shown that for TCP, Scalable-TCP and HighSpeed-TCP, the measurements fit well in the response function model. However there are two slight differences : firstly as the capacity is limited, the maximum throughput achievable is also limited when the loss rate is below a given value. Secondly, as

[5] See the project page of the OpenFlow Switch Consortium http://openflowswitch.org
[6] See the project page of the Click Modular Routerhttp://read.cs.ucla.edu/click/

HighSpeed-TCP and Scalable-TCP both have a threshold before which they act as TCP, they both display a discrepancy. This is normal, as this threshold is not taken into consideration by the models used for the response function.

As a conclusion, the authors stress the fact that a very simple network model and scenario (two competing flows over a dumbbell topology) is enough to "eliminate" some TCP variants (namely Scalable-TCP and FAST-TCP) because of their unfairness, if it possible to test them over a large range of networking conditions. Such a thing is easily done through software emulation.

In [Ha et al., 2007], the same technique is used to evaluate several TCP variants under five different types of background traffic conditions. It is shown that background traffic can have an huge impact on these transport protocols. For instance, FAST-TCP is shown to have stability issues in networks with long delays and small buffers. H-TCP also demonstrates a lack of stability when facing background traffic. However the authors were not able to reproduce the results of [Leith et al., 2007] concerning the intra-protocol fairness of Scalable TCP (even similar results of Scalable-TCP were reported in [Rhee & Xu, 2005]).

3.3.1.2 Advantages

The main advantage of this technique is that it is possible to reproduce a great variety of network conditions using a single computer as a bridge in a limited testbed. These tools are also usually easy to setup. This technique is very useful for protocol designers as they can browse through a wide range of conditions to evaluate their creation. It can be seen as a complement to simulation as it introduces aspects of the real world (the implementation of the transport protocol in the real hardware) with still a lot of control over the network and workload models. The other advantage of emulation over simulation is that the time required to perform an emulated experiment is the same as the elapsed real-time (modulo the time to set up the experiment)

3.3.1.3 Limits

In [Nussbaum & Richard, 2009], the three link emulators, Dummynet, NISTNet, and TC/NetEm, are compared. It is shown that their latency emulation displays some sawtooth patterns that might introduce a bias in the experiments. This problem can be reduced if high resolution timers are used in the Linux kernel. The second problem is that their bandwidth-limitation mechanism stops being accurate for bandwidth greater than 500 Mbps (using a 1 GBps NIC), which may be a drawback when emulating LFNs with such a scheme. It is also possible that choosing a too-high emulated bandwidth might lead to traffic burstiness and modifying the characteristics of the traffic.

Generally, the software emulation techniques are considered to be very costly in terms of computer resources, depending on the kind of treatment that needs to be done. The software overhead they introduce makes it very difficult to use them in the context of high-speed networks. Most transport protocol studies that use this approach limit the bandwidth to around 500 Mbps to avoid interferences due to the software [Ha et al., 2006, Li et al., 2006]. Finally, as these techniques are limited by the granularity of the scheduler's clock used, there is a limit on the precision that can be achieved by software emulation (even though new features like high-resolution timers can help improving that).

3.3.2 Hardware Emulation

It has also been envisioned to rely on dedicated hardware such as FPGA devices to perform the emulation of some components of the network [Guillier et al., 2007a]. It has become increasingly necessary for the following reason : the speed of current networks is growing faster than the treatment speed possible in software. Indeed to keep up with a 1 Gbps wire speed, it might be necessary to be able to handle a maximum packet rate of 83333 packets per second

(a packet of 1500 B should be treated and sent every 12 μs) and this kind of speed is difficult to achieve in software with current-day computers.

One typical example is the AIST GtrcNet boxes [Kodama et al., 2004] that are FPGA-based devices providing a number of parameterizable functions including traffic monitoring at a microsecond resolution, traffic shaping, and WAN emulation at 1 Gbps or 10 Gbps wire speed. Private companies like PacketStorm, Apposite Technologies, or Spirent sell similar solutions. It might also be possible to add in this category NICs which have built-in network processors [Venkatachalam et al., 2003]. They have been used to emulate network links in [Joglekar, 2004]. Such networking cards are available in the Emulab test-bed for this purpose but their use is not wide-spread.

3.3.2.1 Test case

The GtrcNet boxes have been used in a number of studies where precise evaluation was needed. It was the case in [Takano et al., 2005, Takano et al., 2006] where Kudoh *et al.* presented a mechanism in GNU/Linux kernels to precisely pace packets in Ethernet networks. To prove that this software was efficient, it was necessary to have a device able to perform measurements at the packet level. As the network used was a 1 Gbps, a precision of a few μs is necessary. This mechanism of pacing is then used in conjunction with FAST-TCP to show that it could help achieve better performance by "removing" the burstiness of some phases of the transport protocol, like the Slow Start.

A GtrcNet-1 was also used in [Guillier et al., 2007a] for the evaluation of TCP variants over a range of RTTs in a 10 Gbps network. It allowed to identify three regions of RTT : low [0-20ms], medium [20-50ms], and high [>50ms] for which the TCP variants have a different behavior in terms of fairness and in terms of average goodput. It would have been very difficult to quickly find a set of networks that would have allowed us to do the same kind of experiments with the same set of end-hosts.

3.3.2.2 Advantages

The main advantage of this technique is that it will allow the user to have a fine-grained control over the parameters that are modified by emulation as hardware emulators are usually more precise than their software counterparts. It is possible, for example, to increase the RTT using the GtrcNet-10 by increment of a microsecond in the range of 0 to 800 ms. Also hardware emulators are easy to setup as it usually consists in just adding a box in the network.

3.3.2.3 Limits

The main problem with this kind of solution is that it is costly : hardware emulators are not mass-produced equipments like routers or switches. They also require high-end hardware components to be able to operate in the context of high-speed networks, which adds to the cost.

Their use might be somewhat limited for some users as hardware emulators usually comes with a limited set of functionalities. To develop new functionalities for this kind of hardware emulator might require time and expertise as their architectures are somewhat exotic (FPGA) and/or their programming SDK is difficult to handle (Network processors). It might also be necessary to assess (at least once) if such boxes do not interfere with the packets' inter-arrival distribution law when they are internally buffering packets, for instance.

3.3.3 Synthesis

Network emulation is usually a good solution to have access to a larger range of networking conditions (infrastructure parameters) without the need to deploy a new network infrastructure and it is generally easy to setup. Problems of bias can appear if not enough care is put into the calibration of the system.

As emulation might downgrade the performance of an existing networking infrastructure – *e.g.*, by limiting the bandwidth, or by increasing the RTT, or by increasing the loss rate– this is not a choice solution for end-users. This technique is more suited for users that want to implement "what-if ?" scenarios. This is typically the kind of questions a protocol designer asks himself : "What if this transport protocol crosses a lossy link ?", "What if the flows do not experience a similar RTT ?" , *etc.*The network provider might also be concerned if he wants to test failure cases before putting a network in production. In this case, he will probably use specialized hardware boxes like Spirent traffic generators.

Emulation can be seen as a complement of simulation, as it allows deploying the real implementation of the transport protocol or of the application in a sand-boxed environment in which the user can modify the network model as he likes or needs.

3.4 Experimentation

Finally, the last kind of technique used to evaluate the performance of transport protocols is experimentation. It implies deploying the object of study in a real networking environment and observing how it fares against the existing real workload or against a generated workload. It usually comes at a latter point of the development of a transport protocol, as it is necessary to have the protocol implemented, including all the hardware modifications in the networking equipments if applicable.

Basically, there are two ways of performing experimentations, depending on the kind of environment it is taking place in :

Experimentation in Controlled Environment : it corresponds to experimentation in environments where users have a perfect knowledge of the traffic that transits over the networking infrastructure. In some cases, users might also have measures of control over the topology. It is typically the case of ad-hoc testbeds that are created by researchers/companies and used by a small community, like Grid'5000 [Bolze et al., 2006].

Experimentation in Uncontrolled Environment : It corresponds to experimentation in environments where users has little-to-no knowledge of the traffic other than the one they are producing. It is typically the case of large networking environments that are shared with a large number of users, like the Internet or any production networks.

3.4.1 Experimentation in Controlled Environments

Basically, two computers and a switch are enough to create a controlled environments. Depending on the kind of study users want to perform and especially the size of the network they want, it can become more difficult. But over the years, a number of research testbeds have been set up (sometimes only for a temporary event [Cottrell et al., 2005]) to study distributed applications and transport protocols at a larger scale. One example is WAN-In-Lab [Lee et al., 2007], a project from CalTech University that aims at reproducing the characteristics of a Wide Area Network inside a server room (hence the name). It features 2400 kilometers of optic fibbers used to create topologies of 2.5 Gbps links with RTTs of up to 130 ms.

Another example is Grid'5000 [Bolze et al., 2006]. Grid'5000 is a 5000-CPU nation-wide grid infrastructure dedicated to network and grid computing research. 17 French laboratories are involved and 9 sites geographically distributed host one or more cluster(s) of about 500 cores each (Figure 3.2). The sites are interconnected by a dedicated optical network provided by RENATER, the French National Research and Education Network. It is composed of private 10 Gbps Ethernet links connected to a DWDM core with dedicated 10 Gbps lambdas, with a bottleneck of 1 Gbps in Bordeaux. Two international interconnections are also available : a 10 Gbps one with DAS3 (Netherlands' computer grid) and a 1 Gbps one with Naregi (Japan's center for grid research development).

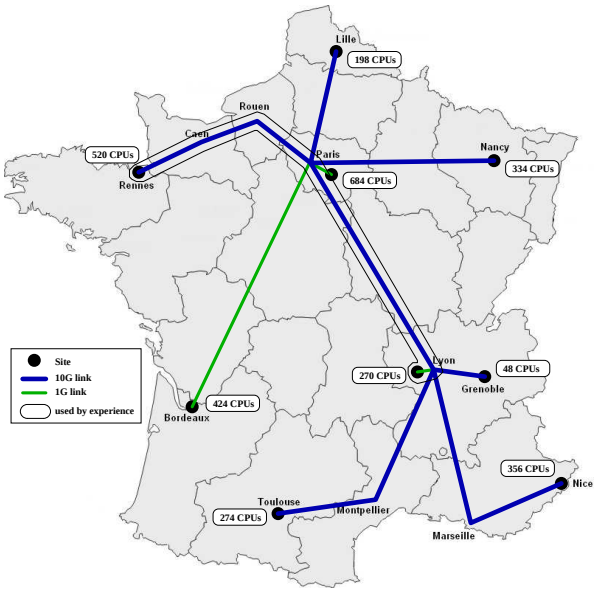


FIG. 3.2 – Grid'5000 backbone network

It is a research tool, featuring deep control, reconfiguration, and monitoring capabilities to complement network simulators and emulators. It allows users to reserve the same set of dedicated nodes across successive experiments, and to have full control of these nodes to run their own experimental condition injector and measurement software. This can be achieved thanks to two tools : OAR and Kadeploy. OAR is a reservation tool which offers advanced features (CPU/Core/Switch reservation). Kadeploy is an environment deployment system that allows users to have their own customized environment automatically deployed on a large number of nodes. For example, kernel modules for rate limitation, congestion control variants, or QoS measurement can be added to the native operating system. As a private testbed dedicated to research, Grid'5000 made it easy to implement distributed software on a large scale like the one that will be presented in the next chapter.

3.4.1.1 TEST CASE

WAN-In-Lab[7] was initially developed for the purpose of testing FAST-TCP. It has also been used to evaluate other transport protocols including MaxNet [Suchara et al., 2005], a TCP variant that uses a multiple-bit explicit congestion notification scheme, and is currently one of the reference testbed for defining a set of baseline tests for the evaluation of transport protocols (see Chapter 5, Section 4.5).

In [Lee et al., 2007], the WAN-In-Lab testbed is used to verify the fact that multiple equilibria may arise when transport protocols using different congestion detection methods (*e.g.*, loss or delay) share a multi-bottlenecks topology. It confirmed some results that were observed using software emulation [Tang et al., 2005], simulation [Tang et al., 2006] and mathematical modeling [Low, 2003], even though it was necessary to change the value of some of the parameters (for instance the queue drop rate) to the limitations of the experimental testbed.

In [Leith et al., 2008], WAN-In-Lab along with a testbed using software emulation is used to compare the behavior of two hybrid TCP variants : TCP-Illinois and Compound-TCP. Experiments to study the impact of reverse path congestion and of cross-traffic on convergence time, RTT fairness, and TCP friendliness were performed. Following these real experiments, some deficiencies were found in the implementation of both protocols. For instance, TCP-Illinois was found to over-estimate the maximum RTT on the path. Local processing delays

[7] See WAN-In-Lab's project page http://wil.cs.caltech.edu/

can become preponderant in the RTT computation if TCP buffers are set to a large value. It is due to the fact that TCP timestamps are handled at the TCP level rather than when the NIC receives the packet. This behavior leads to unfairness towards TCP. Compound-TCP also has issues due to ambiguities in the specification of the transport protocol. Instabilities were also noted when the throughput was close to 1 Gbps due to CPU overloading.

3.4.1.2 Advantages

The real advantage of this method is that it is possible to use the transport protocols as they are implemented inside real networking stacks with their optimizations and tweaks that are usually not considered in simulations or in mathematical modeling. For instance, until recently, float numbers were not supported in the GNU/Linux kernel. Transport protocols that required precise computation for some of their variables may be affected by this, as rounding can have undesirable effects, such as creating instabilities. Some of these effects can be detected through modeling, but there are some aspects that can not be predicted (or thought about) before the transport protocol is actually implemented.

Real networking equipments are also used. They are complex systems that can not be reduced to the simple queue they are usually modeled as. They also implement arbitration algorithms that can have a impact on the transport protocol [Soudan et al., 2007a]. These elements are difficult to take into consideration in mathematical modeling and in simulations.

Finally, the fact that the environment is controlled is a great advantage. The characteristics of the network model and of the end-hosts are fully known. It allows knowing the kind of traffic that has been injected inside the network. The user has access to the complete set of measures at every end-host (and every networking equipment if necessary) : it is much easier to analyze the results.

These advantages make it a good way for protocol designers to study the behavior of a transport protocol in an as-real-as-possible environment.

3.4.1.3 Limits

The main problem with this technique is that deploying a real networking infrastructure is very expensive, even for a limited topology. It is common to see nation-wide (or continent-wide) projects, that associate research entities to be able to get a platform with interesting properties. Then, once the infrastructure is set, the parameters linked to the infrastructure are set too. Depending on the scale of the testbed, parameters like the RTT might have a very restricted range, which might be a problem in itself to test a transport protocol in a variety of conditions.

The second point is that the background traffic used might not be representative of real "live" traffic, as it will always be difficult to reproduce the exact mix that exists in the Internet, for instance. There is also the fact that there will always be a limit in terms of the number of machines available in such an environment.

Finally, the usual problem of software bugs in the implementation of the networking stack might cause interferences and yield results that can not be compared with those obtained with another technique. It might also be necessary to carefully tune the end-hosts so as to remove the risk of misinterpreting results due to bad configuration.

3.4.2 Experimentation in Uncontrolled Environments

Contrary to the experiments performed in controlled environments, experimentation in uncontrolled environments usually takes place in shared networks, such as the Internet, over which it is difficult to know the nature of the adverse traffic. Some testbeds, like PlanetLab [Peterson et al., 2003] or OneLab, have sites distributed all around the world that are connected through public or academic networks that are shared by a large number of users.

3.4.2.1 Test case

In [Cottrell et al., 2005], Cottrell *et al.* study the behavior of a great number of TCP variants and UDT over the academic and research networks existing between Caltech, the University of Florida and the CERN with a 1 Gbps bottleneck. The scenario used in this experiment consists of an end-host that is using a variable number of parallel streams to study how they interact. BIC-TCP over a long-RTT (160 ms) path is shown to be the protocol that yields the best results in terms of mean throughput, stability and intra-protocol fairness. This experiment also allows verifying that TCP and HighSpeed-TCP have some difficulties converging when the amount of available bandwidth increases.

Kumazoe *et al.* [Kumazoe et al., 2005, Kumazoe et al., 2006, Kumazoe et al., 2007] produced a series of evaluation of TCP variants over the JGNII network, an open 10 Gbps Ethernet network between the USA and Japan. The aim of these studies was to test how these different TCP variants react to RTT-unfairness and how they can coexist with TCP in such an environment. Some of the results are difficult to interpret because some variants present instabilities over time. These studies also highlight the need to take into consideration the problem of tuning properly the end-hosts as in some of the experiments, they were not capable of reaching 10 Gbps speeds, due to some hardware limitations. They also show that it might be dangerous to introduce TCP variants in a shared network as they can have an important impact on delay-dependant traffic and on traffic using legacy TCP.

3.4.2.2 Advantages

Just like experiments in a controlled environment, the main advantage comes from using real equipments and real networking stacks to evaluate the transport protocol. In addition to that, the transport protocol will interact with real traffic coming from sources that share a part of the End-to-End path with the flow of the user. Real traffic is usually very difficult to reproduce in other types of experiments, even when using traffic generators based on real captured traffic traces.

This is typically the kind of method an end-user would want to use, as he probably will not have access to a specific testbed to perform experiments or he will not be able to insulate his traffic from that created by other users. It is also consistent with his goal of evaluating a given transport protocol on a given End-to-End path.

3.4.2.3 Limits

The first limit of this method is that it is impossible to know the exact nature of the traffic that has been in competition with the studied transport protocol. The protocol could get bad performance at a given moment due to a transient congestion event. It might then not be possible to dissociate what problems come from the protocol, what problems come from the network parameters, and what problems come from the adverse workload. It makes it mandatory to repeat the experiment over a number of tries, at different times of the day (*i.e.*, be sure not to make all experiments at the peak times of the traffic's diurnal pattern), so as to have a good statistical sampling of the results.

In such an environment, users will probably not have access to sites to perform measurements other than the end-hosts they control. If some other sites are available, it is likely that the user will only have access to coarse-grained information (SMNP-like techniques using aggregate measures over a minute) from which it will be difficult to extract meaningful results.

The other problem is that such an environment can contain black boxes, over which users have no control (for instance if they are outside the Local Area Network of the users) and which they will have problems detecting (for instance, if it is a misbehaving L2 equipment). These black boxes can still have a huge impact on the performance of the transport protocol if for instance, they randomly drop packets.

Evaluation Method	Protocol Designer	Network Provider	End-User
Closed-form modeling	X	X	X
Stochastic modeling	X		
Fluid modeling	X	X	
Packet-level simulation	X		
Flow-level simulation	X	X	
Software emulation	X	X	
Hardware emulation	X	X	
Experimentation in controlled environments	X		
Experimentation in uncontrolled environments	X	X	X

TAB. 3.II – Usefulness of a Evaluation Method for a given type of actor

3.4.3 SYNTHESIS

This section has shown that experimentations matter mostly for protocol designers and end-users. The network provider might also want to perform experimentations, but it will probably be more for performing routine checks or sanity checks on its network infrastructure rather than evaluating transport protocols.

Experimentation can be seen as the final step in a chain of several transport protocol evaluation methods as it implies having a real implementation of the transport protocol and a real network on which to deploy it. In the end, it is impossible to do without it as some aspects linked to the behavior of full real systems, like limits due to the kernel's scheduler granularity or misbehaving middle-boxes, can not be taken into consideration in the other types of evaluation methods.

CONCLUSION

In this chapter we have seen that different approaches : analytical modeling, simulation, emulation, and experiments are available to evaluate the performance of transport protocols. Each of these approaches has its own advantages and limits that make it appropriate for a given study by a given actor. Table 3.III summarizes the main advantages and limits of each of the evaluation methods presented in this chapter. Chapter 4, Section 4.4 presents a more detailed comparison between a simulation approach and a real experimentation approach. It tries to answer the question "how is it possible to compare results that are obtained through different evaluation methods ?".

Table 3.II summarizes the usefulness of a given evaluation method for a given type of actor. It shows that protocol designers can use all these methods to achieve their goal. Indeed, they will probably use them in a succession in order to check that the proposed transport protocol works and has all the desired properties. Network providers also have tools to ensure that their network infrastructure is properly provisioned or that a newly introduced transport protocol will not wreak havoc. The only "neglected" type of actors is the end-user that has mostly one available method (if it is not possible to find a closed-form formula that fits) : experimentation in uncontrolled environments.

The next chapters will then focus more on this category of users by providing automatic tools necessary for the execution of networking experiments in uncontroled environments. It is especially necessary as the growing trends in application development points towards vastly distributed applications.

	Analytical Modeling	**Simulation**	**Software Emulation**	**Hardware Emulation**	**Experimentation**
Examples	N/A	NS-2, OMNeT++	Dummynet, NISTNet	AIST-GtrcNet	WanInLab, Grid'5000, PlanetLab
Advantages	Simple models Fine grained control Closed-form formulae	Simple models Parameter decoupling Fine-grained control	Easy to setup Coarse-grained control	Easy to setup Fine-grained control	Real equipment Real behavior
Limits	Limited models Calculability	CPU intensive Memory intensive Disk intensive Phase effect Limited models	CPU intensive Memory intensive Software overhead Precision limitation	Cost Limited parameters Black boxes	Cost Limited range Limited topologies Black boxes Bugs

TAB. 3.III – Advantages and limits of different evaluation methods

— Four —

Large-scale Experiments : motivations and tools

Introduction

In the last chapter, we have shown that real experiments are almost the only easily available way for end-users to evaluate the behavior of a transport protocol.

In this chapter, we will motivate the need for end-users, especially a subtype of them called the network-application designers, to perform these experiments at a large-scale in Section 4.1. The principal reasons are linked to current changes in the Internet in terms of applications and in terms of capacity. These changes open new perspectives to application designers but at the same time put more constraints on the kind of tests they need to perform before releasing their application.

Examples of large-scale experiments will be provided and analyzed to highlight the need for a tool to automate these large-scale experiments in Section 4.2. The design of this tool, called NXE, will then be presented in Section 4.3. Finally, a comparison of the simulation approach and of the real experimentation approach is made in Section 4.4 through the example of NS-2 and NXE, so as to try to answer to the question "how is it possible to compare results that are obtained through different evaluation methods ?". Finally, Section 4.5 presents the current international effort to produce a standardized test-suite dedicated for protocol designers to evaluate transport protocols.

4.1 Motivation

Generally, transport protocol experiments are done by using a dumbbell topology and aim at evaluationg the behavior of a particular transport protocol between a small number of end-points. But they are more and more situations in which exploring the behavior and the interaction of a large number of end-points is necessary : for instance to study the impact of a large proportion of users deciding to use a different transport protocol than TCP in a large-scale environment such as the Internet or to study the behavior and performance of applications designed to be executed among a large set of end-points, such as distributed Peer-to-Peer applications or grid applications.

Simulation can be used for studies involving both small and large numbers of end-points, but for the study of the above situations, it has show to be unsuitable, insufficient or impractical. Carefully designed large-scale experiments may help in completing these and would give a better insight of potential trends and dangers. The next paragraphs motivate these aspects through the evolution of the Internet in terms of capacity and usage and through the special needs of a subtype of end-users : the network-application designers.

Evolution of the Internet Today, the capacity and the usage of the Internet is fundamentally changing. In the forthcoming year, millions homes will have access to the Internet with fibber lines. Network companies will offer speeds on fibber of up to 1 Gbps. Consequently ultra-high-speed applications will be enabled for low-cost, such as high-definition teleconferencing, telemedicine and advanced telecommuting for people working from home.

The recent years have seen the apparition of new widely-distributed applications, like Peer-to-Peer applications, Grid applications [Foster & Kesselman, 1999, Foster et al., 2001], or Cloud applications [Buyya et al., 2008, Weiss, 2007] that have dramatically changed to way data is exchanged through a network.

The data is no longer going exclusively from the servers to the clients. This is changing the traffic ratio between the forward-path and the reverse-path. It also starts to put into question some of the choices made by network providers as for instance the asymmetry of uplink and downlink capacities in the ADSL networks can be considered as an hindrance by customers. It might push for the development of symmetric networks in the next FTTH generation.

Sources transmitting over the Internet are divided into two categories : collaborative and non-collaborative sources (or greedy sources). The first kind corresponds to TCP flows or flows using TCP-friendly transport protocols. The other is composed of flows that does not respond to congestion signals/packet losses (UDP sources) or by sources aggressively grabbing bandwidth like applications that are using multiple parallel streams. Currently, it has been measured that TCP flows corresponds to 95 % of the current Internet traffic. If performance with collaborative sources decreases, there is a risk that users will switch to "better" but selfish solutions (*i.e.*, greedy ones) and cause further problems. Recent work [Bonald et al., 2009] has shown that in networks with low access rates, the absence of congestion control is sustainable provided proper AQM schemes are enabled in routers and efficient erasure coding is available in the nodes. But the current trend in terms of access' speed does not point in this direction.

As the transport layer is responsible for the stability of the current Internet, it is then important to study and better understand what impact changes to this layer will have and which of them are sustainable. To study this, simulation may be limited as it is not possible to model precisely the Internet both in terms of network models (topology constantly evolving and widely heterogeneous) and in terms of workload models (mix of applications constantly changing) [Floyd & Paxson, 2001]. Large-scale experiments are then needed to study the potential dangers of changes in the transport layer.

A specialized kind of end-users The end-user' kind of actors (seen in Chapter 2, Section 2.2) can be divised into more specialised types of actors. One of them is the network-application designer. It corresponds to actors who are developing applications (for other end-users) based on the Internet transport layer. A typical example of such an application could be BitTorrent[1]. This subtype of actors aims at optimizing a given communicating distributed software.

As the transport layer is highly evolving (as seen in Chapter 1 and the thirty different transport protocol solutions presented therein) and offering lot of improvements for high speed networks, it would be very interesting for network-application developers to experiment their algorithms and applications with different transport variants or tunings.

However, it is often very difficult for them to set up such experiments. Real experiments give the flexibility to explore a wide range of rates and configurations, and the real interaction of the protocol with end-points, network equipments, and real applications. But real experiments are difficult to deploy and analyze [Newton, 1675]. Users are faced with problems related to the hardware (*e.g.*, faulty components in the servers), the software managing the environment (*e.g.*, lack of usable output for the middleware handling the environment) or just improper configuration of the networking stack in the kernel.

[1] See official web-site http://www.bittorrent.com/

Year	1965	1970	1975	1980	1985	1990	1995	2000	2005	2010
Home	<56Kbps	3Mbps	3Mbps	10Mbps	10Mbps	10Mbps	100Mbps	1Gbps	10Gbps	100Gbps
Access				56Kbps		2Mbps		100Mbps	1Gbps	10Gbps
Core	1200bps	56Kbps		1.544Mbps	44.736Mbps	155Mbps	622Mbps	2.5Gbps	10Gbps	40Gbps
K	?	?	?	30	800	80	30	25	10	4

TAB. 4.I – Evolution of the capacities of local area, access, and core networks

The following section will present an evaluation example of the transport protocol that highlights the need to perform real experiments at a large scale : Section 4.2 presents the scenario and the results of a congestion collapse experiment in the Grid'5000 testbed.

4.2 Evaluating the Transport Layer at a Large Scale : the Congestion Collapse experiment

Traditional congestion control approach may present a strong barrier to the large-scale deployment of the exciting and very useful envisioned applications that exploit the emerging huge access capacities. How can we study this question ?

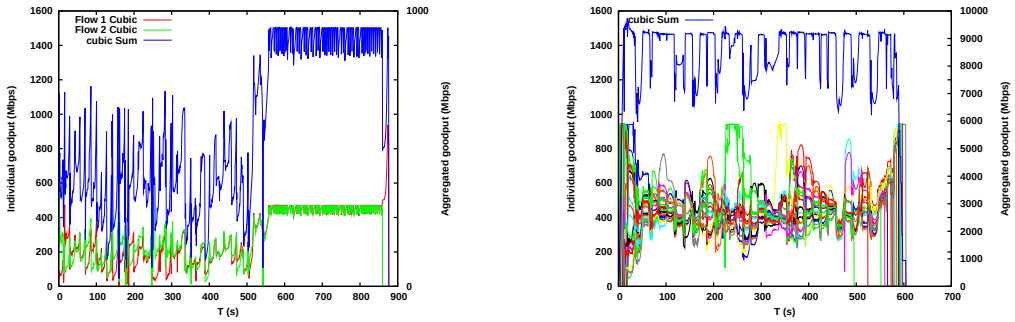
First, we will analyze the changes that are currently occuring both in the network infrastructure and in the traffic demand. Then, we will introduce a scenario designed to show the impact of a huge congestion event on the transport layer and finally present the results of a execution of this scenario at a large scale in the Grid'5000 testbed.

4.2.1 Evolution of network infrastructure

The initial TCP specification was designed with a robustness principle in mind : "be conservative in what you do, be liberal in what you accept from others". Throughout the years, several improvements were made to cope with emerging problems, like congestion [Jacobson, 1988], but the solutions provided were all designed to follow the same robustness principle, providing a "one-size-fits-all" solution. With the evolution of network infrastructures, it is important to determine if it will be possible to keep up with this kind of solutions, especially when it means Internet providers have to over-provision their core links [Paxson, 1994] in order to keep up with the rapid evolution of traffic in the access links. We identify 5 parameters which can contribute to this change.

Aggregation level (K factor) The aggregation level (or K factor) is defined as the ratio between the uplink/downlink's capacity C and the access link's nominal capacity C_a, typically at a network border. In DSL context and more generally in the Internet, it is common to have K ranging over 1000 as users have up to 25 Mbps access uplinks and up to 10 Gbps as the Internet providers bottleneck. In the grid context or the FTTH context, it is more likely to have a smaller value of K, close to 1 or 10. For instance, if Gigabit-capable Passive Optical Network is the technology used for FTTH, users may have up to 100 Mbps access uplinks and a bottleneck size of 1.2 Gbps. A lower K factor means that a few flows can congest the link.

In [Crowcroft et al., 2003], Crowcroft *et al.* note that the parameter has been subjected to cyclic fluctuations and that the current trend indicates a decreasing K. The conclusions are that any QoS scheme is prone to be deployable too early (or too late) and thus not be able to fulfil the needs of ISPs. Simpler solutions like admission control at the edges of the network might be more effective and easier to deploy. However there are hints [Simeonidou, 2008] that the speed of optical access links is increasing faster than that of core links. There is currently



(a) 2x Cubic 1Gbps flows, 1Gbps bottleneck (b) 20x Cubic 1Gbps flows, 10Gbps bottleneck

FIG. 4.1 – Impact of the multiplexing factor on the completion time

no indication of a new technology that might reverse the trend and start a new cycle. Table 4.I summarizes the evolution of the K factor over the last forty years.

M, Multiplexing factor In Figure 4.1 and in [Guillier et al., 2007b], we show that the multiplexing factor, defined as the number M of contributing sources, has a significant impact on the completion time. In the figure, it amounts to 30 % more time needed to complete all transfers. Moreover, we can see that we were able to maintain a more stable aggregate goodput and to have a lower completion time standard deviation by increasing M in the case where more flows are used to generate the same level of congestion. The explanation being that a larger number of flows will be able to interleave and share statistically better the bandwidth. It might not be true if some mechanism generate synchronization between the different sources.

Heterogeneity of access speeds With the advent of FTTH, the level of heterogeneity of access speeds in the Internet will increase. There will be speeds corresponding to 6 orders of magnitude ranging from the modem's 56Kbps to the FTTH's 1 Gbps.

Heterogeneity of RTT There is already a wide range of existing RTTs in the networks, starting from less than one millisecond in a local area network to a little less than a second for network using stationary satellites as relay. TCP has shown over the years that flows with smaller RTTs gain an unfair share of the capacity over flows with larger RTTs.

4.2.2 Evolution of Traffic Demand

File size distribution With the advent of new demanding applications like HDTV, users will need to transfer more data to get access to enhanced content. After the study by Paxson, Taqqu *et al.* on the distribution laws of the ON times of Internet sources [Willinger et al., 1998, Willinger & Paxson, 1998], a Pareto distribution has generally been used to model file size distribution of Internet traffic. These new applications may cause an increase of the mean of this distribution, going from the kilobytes of a web-page to the gigabytes of a multimedia content. They might also change the α parameter that represents the "weight" of the tail of a Pareto distribution. Heavy-tail distributions are obtained when $1 < \alpha < 2$. Table 4.II illustrates the impact of choice of the α parameter on the shape of the Pareto distribution.

The distribution of flows' inter-arrival times (the OFF times) is not likely to change as it is highly dependant on the user's actions and we can assume they will keep following a Poisson distribution. However, with the advent of applications like AJAX, that change the content update scheme from the pull model to the push model, it might not be completely true in the

α	**Mean** μ	**Variance** σ
$\alpha \leq 1$	∞	∞
$1 < \alpha < 2$	$< \infty$	∞
$\alpha \geq 2$	$< \infty$	$< \infty$

TAB. 4.II – Impact of the choice of the α parameter on the mean and the variance of a Pareto distribution

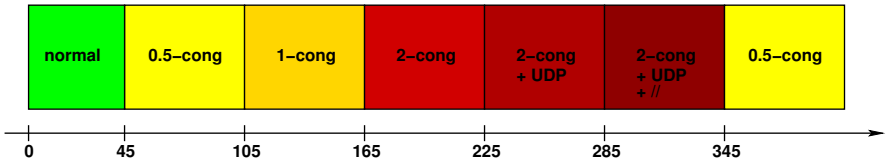


FIG. 4.2 – Timeline for the Congestion Collapse in Grid'5000 experiment

future. Automated retrieval methods like RSS or retrieval methods based on timers might also change the flow's inter-arrival times' distribution.

Quality of Service The flows that coexist in the Internet have usually two possible different constraints : temporal and throughput. In the first case, the application (VoIP, video streaming) requires to receive each successive packets in good order and with reasonable jitter. In the second case, the application only wants to get its data as soon as possible to allow the user to act. TCP was not developed to provide a time-sensitive transport protocol. A shift in the mix might have consequences, especially if another transport protocol like UDP is used by the application.

All these technological and application changes are bound to have an impact on the performance of a network and of the flows crossing it. It is then necessary to use the real network and the real application in its context in order to so as to qualify and potentially quantify the risks and their factors.

4.2.3 SCENARIO

We have designed a specific scenario to study the impact of a huge congestion level on specific transport applications. Up to 100 sources performing simultaneous bulk-data transfers were used to generate this amount of congestion level. This scenario has been run in real-time during the workshop"The Future of TCP : Train-wreck or Evolution ?"[2] that took place at Stanford University in April 2008 [Guillier & Vicat-Blanc Primet, 2008a].

Without any special mechanism to script the organization of these transfers, the low aggregation level that could be found in emerging networks quickly lead to a huge congestion level, disrupting the bulk data transfers and deeply impacting all interactive traffic. The metric considered here is the transfer's completion time, as it is relevant to what users want : receiving their files in a reasonable time. It can also be used by the network-application designers that will be interested in providing applications yielding the minimum transfer completion time, through the choice of the adequate transport solution.

The timeline of the experiment is shown in Figure 4.2. The idea is to increase the congestion level (defined as the ratio between the sum of the capacity of the sources and the capacity of the bottleneck) in the network over time and observe the behavior of the reference flows. It also simulates what could be the typical behavior of users when they start experiencing

[2]See the workshop web page http://yuba.stanford.edu/trainwreck/

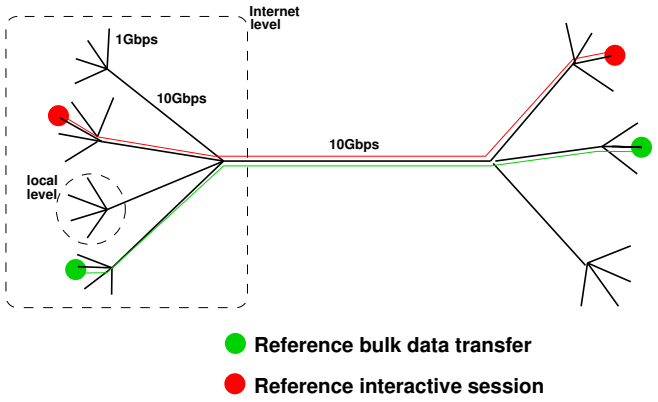


FIG. 4.3 – Experiment topology

large congestion events : changing their protocol transport to UDP and then move to multiple parallel streams as a last ditch effort to get themselves a bigger part of the bandwidth. All heavy flow sources are using iperf as a transfer application and TCP as a transport layer. A reference heavy-flow source is used to perform a continuous bulk-data transfer, highlighting the impact of the current congestion level on the transfer.

The seven time periods are organized as follows :

- $\mathbf{T} \in [0 - 45s]$ **:** Only the reference flow is started. It allows getting a baseline for the completion time of the heavy transfer.
- $\mathbf{T}$ $in[45 - 105s]$ **:** The congestion level rises to 0.5.
- $\mathbf{T} \in [105 - 165s]$ **:** The congestion level rises to 1.0.
- $\mathbf{T} \in [165 - 225s]$ **:** The congestion level rises to 2.0.
- $\mathbf{T} \in [225 - 285s]$ **:** The congestion level stays at 2.0 but 20 % of the sources are now using UDP.
- $\mathbf{T} \in [285 - 345s]$ **:** The congestion level stays at 2.0 but 20 % of the sources are using UDP and 20 % are using TCP with 10 parallel streams.
- $\mathbf{T} \in [345 - End]$ **:** The congestion level recedes to 0.5.

For the instanciation in the Grid'5000, we used a hierarchical dumbbell, with 10 Gbps bottlenecks as seen in Figure 4.3. The backbone corresponds to the 10 Gbps Paris-Lyon link. 100 independent sources distributed in the southern sites (Lyon, Toulouse and Sophia) and the northern sites (Rennes, Lille and Nancy) generated bulk data transfers. Other sources created interactive sessions in both directions. All sources used 1 Gbps Ethernet network cards, which means that the aggregation level is 10 for the local level and 1 for the Internet level.

4.2.4 RESULTS

Figure 4.4 and Table 4.III presents the results for the reference heavy flow. There are three distinct phases : low congestion ($[0 - 165\ s]$), heavy congestion ($[165 - 345\ s]$), and recovery ($[345 - End]$). When the congestion is low (congestion level close to 1), we can see that the flow is not heavily affected. The only notable event during the $[0 - 165\ s]$ time-period is that the flow experienced a sharp drop in goodput at the other flows' arrival time that was caused by the other flows' Slow Start.

During the heavy congestion period, the flow experienced a sharp goodput lost when the congestion started, with a 70 % difference with the first time period. It is less than the goodput it would get if the capacity was shared equally among all flows. The mean goodput achieved during the $[165 - 225\ s]$ and $[225 - 285\ s]$ periods is similar. As soon as the number of flows increase in the last period, there is another large drop in goodput.

In the last period when the congestion level is decreased to a more reasonable level, it still

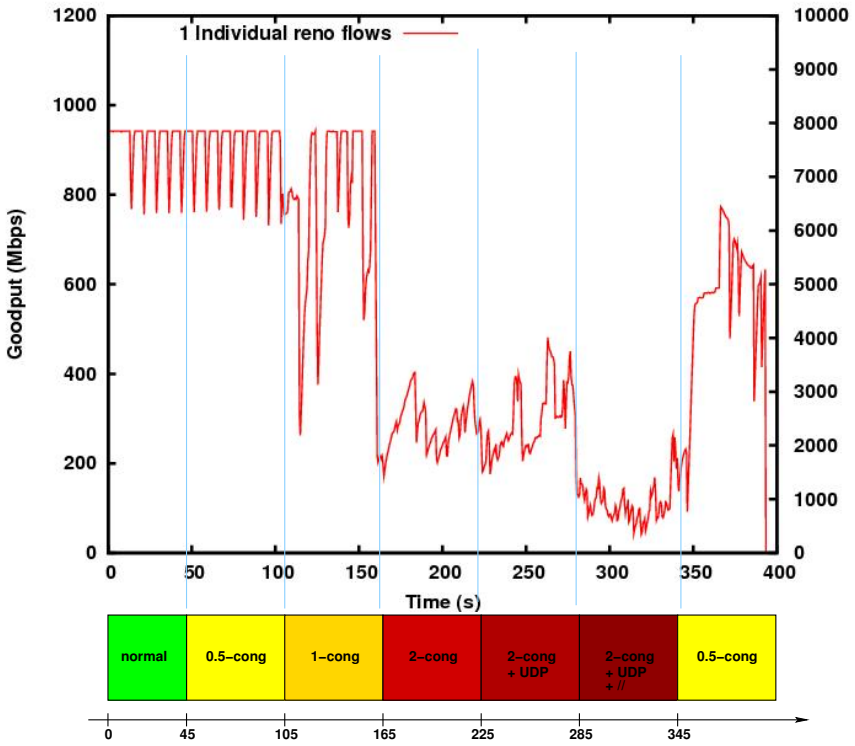


FIG. 4.4 – Goodput of the reference heavy flow between the Sophia and Rennes sites (18 ms RTT) during the execution of the Congestion Collapse in Grid'5000 scenario

takes a long time for the flow to recover and go back to a high goodput value.

Another heavy flow performing a regular 3000 MB transfer was monitored in real-time. During the non-congested period, it was able to complete the transfer in 28 seconds. As soon as the heavy-congestion started, its completion time dramatically increased and was multiplied by more than ten. In such a case, it is very likely that the end-user will get impatient and cancel his transfer [Yang & Veciana, 2001].

4.2.5 SYNTHESIS

This use-case has shown the impact of a wide range of congestion levels and traffic types on TCP in a large-scale experimental testbed. It has shown that TCP is not fit to "survive" in an environment that is dominated by selfish users. Indeed, by using techniques like multiple

Time period (s)	Mean goodput (Mbps)	Relative difference with first time period
0-45	940	0 %
45-105	940	0 %
105-165	800	-15 %
165-225	270	-70 %
225-285	270	-70 %
285-345	100	-90 %
345-End	600	-35 %

TAB. 4.III – Relative evolution of the goodput during the successive time periods of the Congestion Collapse in Grid'5000 scenario

parallel streams or unresponsive UDP flows, such users dramatically decrease the share of bandwidth of "fair" users whose transport solutions are using TCP (a decrease of 90 % of the throughput was noted when a fair allocation would have decrease by at most 50 % the share of the bandwidth). It should also be noted that such huge congestion levels can have a long-lasting effect as the reference flow considered was not able to go back to its "normal" throughput state in 50 seconds.

To make sure that such a situation doesn't happen (for real) in networks like the Internet, several solutions can be envisioned. One would be to implement bandwith-sharing mechanisms (or similar mechanisms allocating quotas to users/end-hosts rather than their individual flows) instead of the implicit "fair-sharing" of TCP that has shown its limits in this example. Another solution would be to provide the end-users with a TCP variant that is both efficient and fair, so as to remove the motivations for selfish users to switch to other techniques. But this require a methodology such as the one we will present in Section 4.5 to make sure that such a new TCP variant has all the required properties.

Such experiments are difficult to implement because they require to activate and control simultaneously hundreds of real high-speed end-hosts in a nation-wide real testbed. It also required to have access to real-time measurements to be able to follow the evolution of the metrics (here the goodput) during the execution of the scenario. Similar simulations could have been done, using NS-2 for instance, but would have required days to complete. This highlights the interest of this evaluation method and stresses the need for automatic tools to perform such studies. The next section will then present our tool, called NXE, designed to automate large-scale experiments in real environments.

4.3 Tool

The automation of the construction and execution of large-scale experiments, allowing to instantiate a range of topologies and scenarios to evaluate the performance of a transport solution thanks to the varied metrics presented in the previous chapter, appear to be complex and time consuming. A rigorous framework defining stages during the experiments corresponding to specific functions such as setting up every individual end-hosts, synchronizing the execution between all involved entities, or collecting the results at the end of the experiment, would be highly helpful. Indeed, doing everything manually is tedious or impossible due to the size of the system. It also increases the risk of introducing bias into the results, for instance, if some sources are started before others and gain an unfair advantage.

The first solution that comes to mind is to develop ad-hoc and specialized scripts to perform a defined scenario. But it can be considered as a waste of resources if everything has to be thrown away and started afresh every time a new kind of scenario needs to be performed. It is then much better to use a framework that will take care of all the "administrative" operations, leaving only the logic of the scenario to be added.

But the biggest issue to tackle is linked to the dimension of the parametric space (as seen in Tables 2.IV and 2.V). Even when carefully limiting the range of selected parameters to "interesting" values, it is possible to end up very quickly with hundreds of hours worth of planned experiments to run and analyze. It is also necessary to remove as much as possible the "human factor" from the evaluation process so as to ensure that successive experiments are reproductible. These points advocate for the use of tools to automate these tasks.

The rest of this section is organized as follows : Section 4.3.1 will detail the architecture and concept designs that were chosen for NXE, our tool to automate large-scale networking experiments and Section 4.3.2 presents the state of the art of existing tools designed to perform large-scale experiments.

4.3.1 Architecture

This section aims at presenting the concepts of a tool we have designed to automate large-scale experiment in real environments.

4.3.1.1 Definitions

A networking experiment is described as a distributed application that involves a set of synchronized and controlled processes that implements a given experiment scenario in real environments.

An experiment scenario skeleton defines a succession of dates at which an event occurs. The events corresponds to the starting point of an action (*e.g.*, the start of a new bulk data transfer, of a new web session) combined with the parameters relevant to this action (*e.g.*, distribution law of file sizes, inter waits) between a set of end-hosts, whose size depends on the kind of application modelled (*e.g.*, 2 for data transfers, many for parallel applications). The network experiment involves a set of end-hosts that are organized according to a topology.

An abstract topology defines sites (*e.g.*, aggregation of end-hosts, as in a cluster), aggregation points between them (*e.g.*, switches or routers) and networking links. The "abstract" term refers to the fact that an instantiation of the nodes over this topology is needed at run-time as in real testbeds, and resources allocation mechanisms might be used to accommodate multiple users.

A network experiment workflow is the sequence of various operations that are done at each stage of the execution of a protocol evaluation scenario. Figure 4.5 shows a typical experiment workflow that can be used during an evaluation process. It is composed of a number of tasks which are connected in the form of a directed graph. These tasks can be broken up into elementary operations to explicit what is done precisely at each stage. The tasks can be designed so that there is as little interaction as possible between successive tasks.

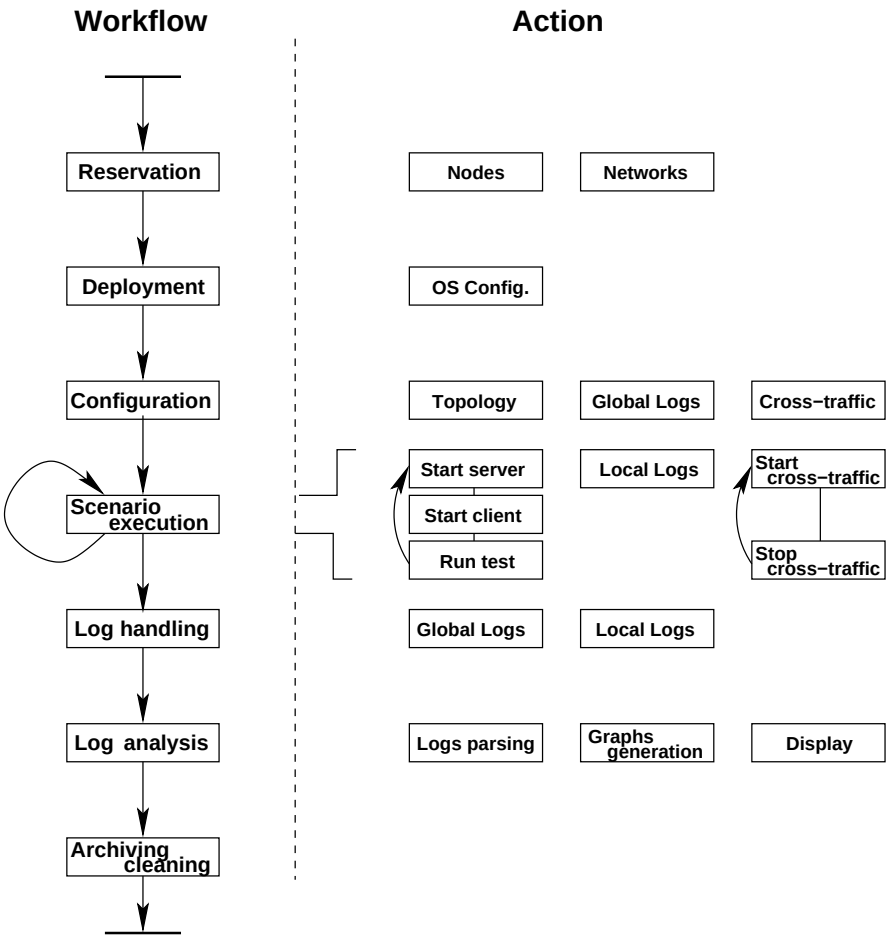
4.3.1.2 NXE : a tool to automate large-scale networking experiments

Network eXperiment Engine (NXE) [Guillier & Vicat-Blanc Primet, 2009b] is an application that scripts the execution of a schedule based on a centralized relative timer that is generated from the input scenario description. It assumes that the granularity of the scenario's execution steps is coarse and of the same order of magnitude as a second. The timers used are much more fine-grained (in the order of a few milliseconds), so the tool could be enhanced to be more precise, but currently it does not seem relevant to the general user-context. For scalability purposes, it launches a separate thread for every node involved in the scenario, and issues the commands at the appropriate time via an SSH remote command execution. Only one SSH connection is opened per node and it is kept during the whole life of the experiment. The commands are executed via different SSH channels over this single SSH connexion.

The scenarios are described through XML files that provide a simple and hierarchical description of the topology, the general configuration, and the interactions between the end-hosts. The XML input files are divided into three parts :

Topology : a simple abstract topology description, providing an easy way to describe the resources and how to exploit them. External scripts are required to indicate the way the nodes' reservation and deployment will be performed. This way, it is easy to adapt to local nodes management policy.

Configuration : a collection of flags that affect the global behavior of the application, such as forcing the synchronization of scripts prior to running an experiment or changing the SSH authentification information.



Reservation : at this stage, the available resource allocator services are contacted to get the resources needed by the experiment, *e.g.*, computing nodes or network links.
Deployment : this configuration phase can be either a reboot of the nodes under an adequate kernel image, or just the setting of the OS internal variables (*e.g.*, TCP buffer size) to the appropriate value.
Configuration : at this stage, the available hardware (*e.g.*, hardware latency emulator, routers) is contacted to alter the topology, or to activate the gathering of statistical information (*e.g.*, aggregate throughput) according to the needs of the experiment.
Scenario execution : here the actual execution of the scenario is started. The scenario can be run multiple times in a row to ensure that the results are consistent.
Log handling : the logs generated by the nodes and the global logging facility are gathered at a single point for analysis.
Log analysis : the logs are parsed, and metrics are computed from them to generate graphs that can be easily interpreted by the user.
Archiving and cleaning : resources are reset and released.

FIG. 4.5 – Workflow of NXE

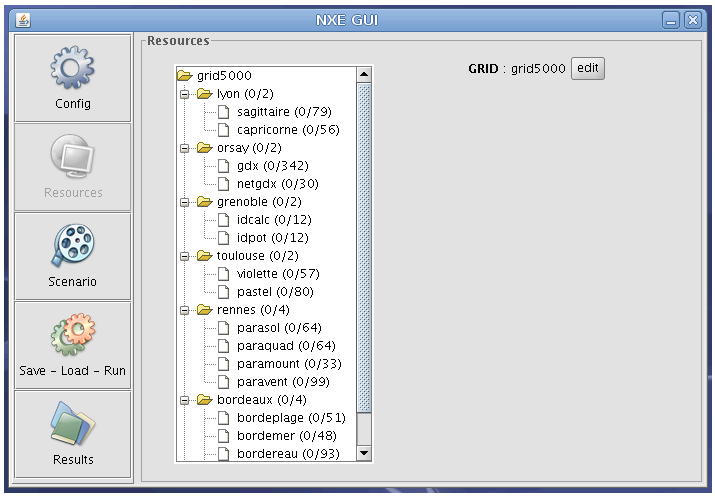


FIG. 4.6 – Screenshot of the NXE GUI, resource selection part

Scenario : each node (or set of nodes) involved in the scenario is given a role (server, client, *etc.*) and a list of execution steps that are to be run during the experiment, *e.g.*, a set of dates according to a centralized relative timer and a set of scripts that are to be executed on the target node(s) at the appropriate time.

To ease the writing process of the experience's description that can be cumbersome, especially if a large number of nodes or a lot of individual actions must be set up, a graphical user interface has been developed, as seen in Figure 4.6. It allows a much simpler selection of the available resources and a simpler organization of events occurring during the experiment.

Example Algorithm 1 describes the pseudo-definition of two separate sets of nodes on two different clusters of the Grid'5000 testbed. It includes the necessary information to perform the reservation and the deployment of the nodes. Through aliases, it is possible to define multiple sets of nodes on the same cluster/site. The information provided on the connection are used during the graph generation part of the execution of the scenario and not to perform routing operations.

Algorithm 2 presents a set of the configuration parameters that can be used to select special functions. For instance it is possible to execute the scenario with a different username or to make sure that the SSH keys or the scripts needed during the execution are synchronised on the nodes.

Algorithm 3 presents an example of pseudo-definition of scenario that can be easily translated into the XML grammar used by NXE. The use of aliases allows to write scenarios that are independant of the location of its execution. All the nodes from $ClusterA$ act as servers. The experiment is set to last only 100 s. A first part of the nodes from $ClusterB$ are performing TCP transfers. The transfer sizes and interarrivals are taken from random distributions[3] and at most 10 of such transfers will be performed (depending if the timeout expires first). A second set of nodes from $ClusterB$ are started later on to perform UDP transfers at a fixed rate. It is possible to perform different actions during different time periods on given sets of nodes.

Implementation The actual implementation is made in Python and uses the python Expat XML library, the paramiko SSH2 library and the Gnuplot python bindings. In addition, *iperf* and *D-ITG* programs are used to simulate the workloads, but it would be very easy to replace

[3] it is also possible to provide an array of values

Algorithm 1 Example of a pseudo-definition of the abstract topology definition for two sets of nodes in two clusters

define set of nodes **from** *Sagittaire*
  $alias \leftarrow ClusterA$
  $frontend \leftarrow lyon.grid5000.fr$
  $reservation\_tool \leftarrow OAR$
  $deployment\_tool \leftarrow Kadeploy$
  $number \leftarrow 12$
  $duration \leftarrow 3h$
  $bandwidth \leftarrow 1000Mbps$

define set of nodes **from** *Violette*
  $alias \leftarrow ClusterB$
  $frontend \leftarrow toulouse.grid5000.fr$
  $reservation\_tool \leftarrow OAR$
  $deployment\_tool \leftarrow Kadeploy$
  $number \leftarrow 12$
  $duration \leftarrow 3h$
  $bandwidth \leftarrow 1000Mbps$

define connection **between** *ClusterA* **and** *ClusterB*
  $bandwidth \leftarrow 10000Mbps$
  $latency \leftarrow 12ms$

Algorithm 2 Example of a pseudo-definition of configuration parameters

define global config
  $login \leftarrow root$
  $ssh\_key \leftarrow id\_rsa.pub$
  $sync\_keys \leftarrow$ **True**
  $archive\_path \leftarrow$ /archives/
  $script\_path \leftarrow$ /bin/
  $sync\_scripts \leftarrow$ **True**

Algorithm 3 Example of a pseudo-definition of a scenario involving two sets of nodes

```
for all ClusterA nodes do
    role ← server
    at time = 0s do start_server for 100s
end for

for ClusterB nodes 1 to 5 do
    role ← client
    target ← node from {ClusterA nodes 1 to 5 }
    TCP_variant ← Reno
    transfer_size ← pareto(10M,1000M)
    interarrival ← exponential(1s)
    occurences ← 10
    at time = 10s do start_client_TCP for 90s
end for

for ClusterB nodes 6 to 12 do
    role ← client
    target ← node from {ClusterA nodes 6 to 12 }
    rate ← 600Mbps
    at time = 50s do start_client_UDP for 50s
end for
```

them. Bash scripts are used to wrap the calls to these softwares on the end-hosts that are used in the scenarios. More information about the project or more details on the software can be found on this web page : http://www.ens-lyon.fr/lip/reso/software/NXE/index.html . Version 1.0 of NXE has been released under the CECILL software licence.

NXE has notably been used for a number of our experiments on the behavior of transport protocols under the strain of heavy congestion as seen in Section 4.2. Chapter 5 will also present other evaluation examples performed thanks to the help of NXE.

NXE is also used with the Metroflux system[Loiseau et al., 2009a, Loiseau et al., 2009b] as a way to activate a large number of supervised ON/OFF sources over a large distributed system. Metroflux is a fully operational metrology platform built GtrcNet FPGA-based device technology to capture the traffic at packet level on a very-high-speed link and to analyze it at the packet-, flow- or any higher aggregation-levels, providing researchers and network operators with a very flexible and accurate packet-level traffic analysis toolkit configured for 1 Gbps and 10 Gbps links. In [Loiseau et al., 2009a] it was used more specifically to finely analyze how traffic distributions and transport protocols interact which each other in high-speed networks.

4.3.2 Related work

In this section, we present some existing tools that have been proposed before and during this research work to perform large-scale experiments in real testbeds and compare their respective characteristics.

ZENTURIO [Prodan & Fahringer, 2002] is an experiment management tool for cluster and grid architectures, most notably those using GLOBUS [Foster, 2006] as an infrastructure. It relies on the ZEN directive-based language to manually describe the experiment through annotations put in the application files. It provides three Grid services : Registry Service, Experiment Executor and Experiment Generator that are used together to compile, execute, control and monitor experiments on-line.

DART [Chun, 2004] is a testing tool developed for Emulab [White et al., 2002], a network testbed that allows researchers to perform emulated experiments in which it is possible to specify arbitrary network topologies. It features a system to automatically test distributed applications using a set of predefined tests. Extra modules allowing to generate faults (*e.g.*, nodes failures, network failures) and introducing performance anomalies (*e.g.*, overloaded CPU, memory) were developed to provide a set of various conditions for the application's execution. It strongly relies on Emulab's components to work.

Plush [Albrecht et al., 2006] is an application management tool developed for PlanetLab [Peterson et al., 2003], an overlay network testbed. Three kinds of applications often used in PlanetLab are defined : short-lived applications, long-lived Internet services and Grid-style parallel applications. From these categories, specific choices are made in terms of resource reservation to suit the needs of the application. It then can be deployed across the selected nodes and a daemon is responsible for monitoring failures. A call-back system is available to implement application-specific recovery in case of failure. This tool is not directly designed for running experiments and has no support for handling post-treatments (like analyzing logs) when the application has finished its work. The useful call-back system means that the application needs to be specifically developed for this tool.

Emulab Workbench [Eide et al., 2007] is a tool introduced in 2007 to define experiments in the EMulab testbed. It aims at replacing the web interface that could be used to define experiments (both static and dynamic parts using a language similar to the one used by NS) but it was found to be lacking some properties : dependence of experiment definition and instantiation, no way to describe the relation between experiments, and limited help in managing the experiments including data-handling and testbed setup. It introduces the definition of an experiment template that contain all the necessary tools to execute an experiment (*e.g.*, source code, input files). It is also possible to define parameters that can be used to instantiate the template over the testbed resource. It is also able to record experimental data if the necessary tools are deployed in the testbed. A GUI is provided to ease the creation of templates. Several use cases are provided to show what their tool can do to help developing a system, analyzing the performance of an application or monitoring an application. The authors point some limits especially with the storage system for which it is not sustainable to keep forever all the recorded data of the experiments and plan to store only data relevant to the user. Limits were also found on the way errors are handled by the tool.

Expo [Videau & Richard, 2008] is an experiment engine for dedicated testbeds, like Grid'5000, based on a client/server architecture. A case study was performed beforehand in [Videau et al., 2007] to define the necessary properties of such an experiment engine. It relies on a specific language to describe the experiments. Through the language, it is possible to describe parallel task execution and logical aggregation of resources. Some of these commands are specific to tools used in the Grid'5000, like kadeploy and OAR. Parallel launcher Taktuk is used to improve the deployment time on the nodes.

Table 4.IV provides a comparison of the different characteristics of existing tools designed to perform large-scale experiments. All these alternatives rely on a particular middleware or software controlling scheme developed for the specific testbed. In our user-oriented scope of transport protocols' evaluation in real environments, it is also better if the end-hosts are not shared among several users and if the network is not emulated.

New tools like Netset [Arora et al., 2009] or SPLAY [Leonini et al., 2009] have been proposed since the development of NXE. They push forward the same idea : a tool to automatically perform networking experiments at a large-scale in a real context is necessary.

Name	year	Real Resources	Experiment description language	Application modification	System independent
ZENTURIO	2001	Yes	ZEN	Required	No (GLOBUS)
DART	2004	No	XML	Not required	No (Emulab)
Plush	2006	No	XML	Required	No (PlanetLab)
Emulab Workbench	2007	No	Web interface	Not required	No (Emulab)
Expo	2008	Yes	Ruby	Not required	No (Grid'5000)

TAB. 4.IV – Comparison of the different characteristics of existing tools to perform large-scale experiments

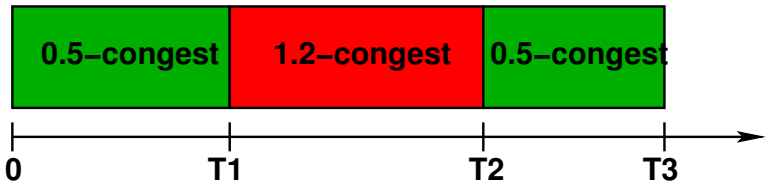


FIG. 4.7 – Description of a simple scenario to compare the NS-2 and NXE approaches

The next section will perform a comparaison between the simulation and the experimentation in a controlled environment approaches using our tool to automate as a testcase example.

4.4 Methodologies comparison : NS vs. NXE

In this section, we explore the differences between the NXE and NS-2 approaches. A few years ago, the main problem for comparing these two softwares would have been to provide similar TCP stacks for both. TCP-Linux [Wei & Cao, 2006], a NS-2 module designed to interface with a GNU/Linux TCP congestion control module, has been developed. It is now possible to have the same congestion control algorithms, removing part of the risk of coding errors. Now the main problem lies in the difficulty to provide a similar configuration for both stacks. A typical example would be the delayed ACKs [Allman, 1998]. Depending on the implementation of the TCP stack on the receiver's end and the way the receiver sends acknowledgements back, the resulting behavior of the congestion window's evolution will not be the same if it is done every other ACKs (the default behavior in most GNU/Linux TCP stack) or if an ACK is sent for every byte received.

4.4.1 Scenario

Figure 4.7 presents the timeline of the simple scenario used to compare the NS-2 and NXE approaches. Each phase is as follows :

- **[0-T_1] :** 5 nodes start emitting.
- **[T_1-T_2] :** More nodes are introduced into the network, raising the congestion level to 1.2.
- **[T_2-T_3] :** The congestion level goes back to 0.5 by stopping the emission of the newest flows.

T_1,T_2 and T_3 are chosen so as to theoretically allow at least two full cycles of the AIMD algorithm (*i.e.*, increasing the congestion window to W_{max} twice). In this scenario, each time period is roughly equal to 82 s. The topology used is a classical dumbbell with end-hosts using 1 Gbps network interfaces. The bottleneck is the 10 Gbps output port of a switch. To allow a

feasible RTT in the testbed environment used by NXE, a RTT of 20 ms is chosen. Packet size is set to the Ethernet frame size. The flows consist of unilateral bulk-data transfers, continuously emitting during their period of existence.

4.4.2 Experiment Description Language

This section provides a comparison of the experiment description language of each performance evaluation tool.

A NS-2 user defines the whole life of an experiment, from the definition of the entities involved and their organization in a topology to the definition of the analysis of the results, passing by the execution of the scenario with the Tcl programming language. It provides an object-oriented grammar to describe the end-hosts capabilities and the networking stack (*e.g.*, the application and the transport protocol). It is a scripting language and as such it is interpreted at execution time. However, it uses byte-code to increase the speed of execution. The defined topology provides paths as a series of L3 queues that are crossed by the packets to reach their destination.

A NXE user describes the available resources (the end-hosts) and the way they interact during the experiment with the XML language. XML allows a hierarchical description of the experiment. As NXE only provides a framework for the execution of a scenario, the code actually executed by the nodes needs to be provided by the user, even though sample Bash scripts are available. The topology and routing are set by the environment in which NXE is used. However it is possible to use emulation means like with Gtrc-Net boxes to alter the network configuration.

One must also note that both languages provide facilities to set up experiments involving hundreds of nodes or a large number of networking events. NS-2's description language being a true programming language provides a greater flexibility in the sense that it is possible to use specific statements to avoid defining a particular treatment for every entity involved in the experiment, *e.g.*, loops and complex data structures. The XML parser of NXE allows defining sets of nodes for which a given step of action should be performed. In NXE, the logic of the experiment is put inside the scripts that are used on the end-hosts rather than in the framework that controls the execution of the experiment. It means that NXE does not interfere with the parameters (of which it has no knowledge) that are given to the end-hosts. If the scenario requires specific parameters for each end-host, it is necessary to define a separate step description for every node, which can be lengthy and error-prone.

There is also the problem of the default value used in both softwares. Through NXE, users will probably get the parameter values used in a real environment but they might not be completely adapted to their use-case. A typical example here is the size of the TCP buffers. In GNU/Linux kernels, it usually defaults to 64 kB, which is not enough to accommodate a flow over a LFN. Through NS-2, the default values might not reflect a realistic use-case. A typical example being the packet size which defaults to 1000 B in NS-2 : that does not correspond to a packet size [Thompson et al., 1997] usually encountered in the Internet. In both case, thought should be put in the matter of defining the experiment and how all the parameters involved might have an effect on the results.

4.4.3 Resulting Output

In NS-2, if the necessary time needed to write and correct the input file is neglected, the setting up of the experiment and of all the entities involved in it, is instantaneous. The execution time is more problematic, as it is a function of the number of events that need to be processed (roughly speaking, the total number of packets sent multiplied by the number of entities crossed). For example, in the 5-minute scenario studied here, due to the high-speed nature of the network with a 10 Gbps bottleneck (and thus a large amount of packets sent), the simulation took about 40 minutes to be executed once on a desktop computer (and about 150 MB of memory was used). It is almost one order of magnitude longer (9 times) than the

Tool	Experiment setup (s)	Experiment execution (s)	Experiment post-processing (s)	Total (s)
NS-2	1	2360	5	2366
NXE	460 (12)	260	10	730 (282)

TAB. 4.V – Decomposition of the execution time of the scenario with NS-2 and NXE

intended experiment duration. Solutions have been proposed to decrease the execution time by scaling down the speed of every entity [Pan et al., 2003], but not all the mechanisms involved in the simulation are linear (typically the slow start phase and the congestion avoidance phase of a lot of TCP variants). This might lead to an inaccurate reconstruction of the results when going back to the right scale. At the end of the experiment, the user will be able to plot the evolution of factors at the packet level, even though it is clearly not practicable for the number of packets involved in a high-speed transfer.

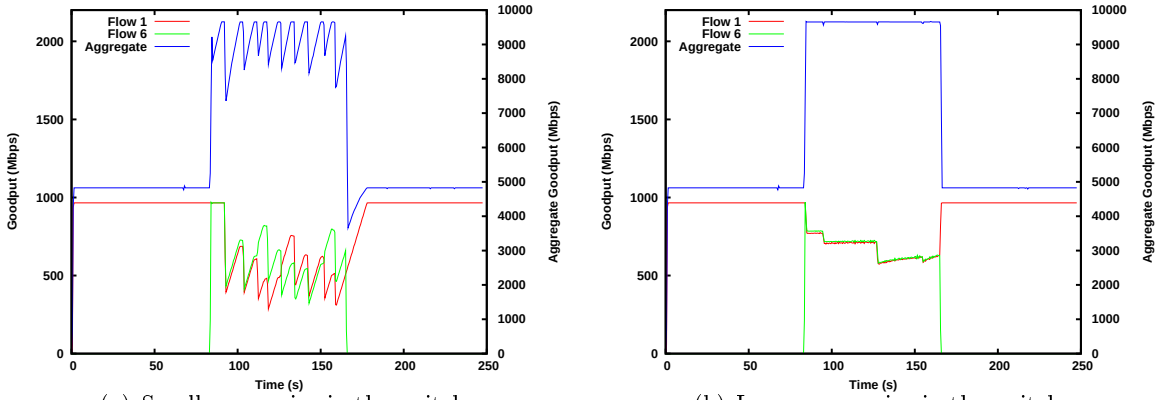
For NXE, the set up phase can take a long time. First it is necessary to reserve the desired number of resources and possibly wait for them to become available. But the largest part of the set up phase is generally due to the deployment of a kernel image on each node. It is done so as to make sure that every end-hosts taking part in the scenario are properly configured. In our example, the deployment through Kadeploy in Grid'5000 takes an average of 400 to 500 seconds depending on the hardware available on each Grid'5000 site and independent of the number of nodes used. To alleviate this cost, it is possible to use the same reservation set for several experiments through NXE, moving the required set up time from about 460 s to 12 s. The second biggest part of the set up time is due to the opening of the SSH connections and the synchronization of the code to be executed on the node, and is linear with the number of nodes. Once the set up phase is completed, the execution phase starts and lasts only for the duration of the experiment. As a comparison, it might be possible to run 7 to 8 full experiments according to our scenario while waiting for the single NS-2 execution to complete. In the end of the experiment, NXE retrieves the logs generated by the code executed on the nodes and it is then possible to plot the results (*e.g.*, goodput, congestion window size if the Web100 patch is used) though usually at a coarser grain (of the same order of magnitude as a second or a millisecond) than at the packet level. Additional equipments need to be installed, as shown in [Loiseau et al., 2009a], if a more fine-grained approach is needed, since traditional packet-acquisition techniques like *tcpdump* do not scale well in high-speed environments.

Table 4.V provides a more precise decomposition of the time spent in every stage of the experiment.

4.4.4 INFLUENCE OF THE QUEUE SIZE

In this section, we conclude our comparison of the functionalities and characteristics of the NS-2 and NXE approaches by comparing the results of the simple scenario presented above. To present the results, two flows that took part in the experiment are singled out : one that lasts during the whole experiment and another one that only exists during the congestion period $[T_1 - T_2]$. For both tools, the end-hosts' buffer size was set over the BDP value of the path (2 MB), so as to make sure the bottleneck is in the network, not in the end-hosts. In both cases, a dumbbell topology was used with a central-link RTT of 18 ms. This value was chosen because it is the RTT existing between Grid'5000's Sophia and Rennes sites. The major difference between the two network models is that in Grid'5000, the value of the queue size is fixed and unknown. The aggregate goodput corresponds to the sum of all the flows' individual goodput at a give time.

Figures 4.8 and 4.9 present the results of the experiments using the NS-2 and NXE ap-



(a) Small queue size in the switch (b) Large queue size in the switch

FIG. 4.8 – Goodput measurement for two flows simulated using NS-2

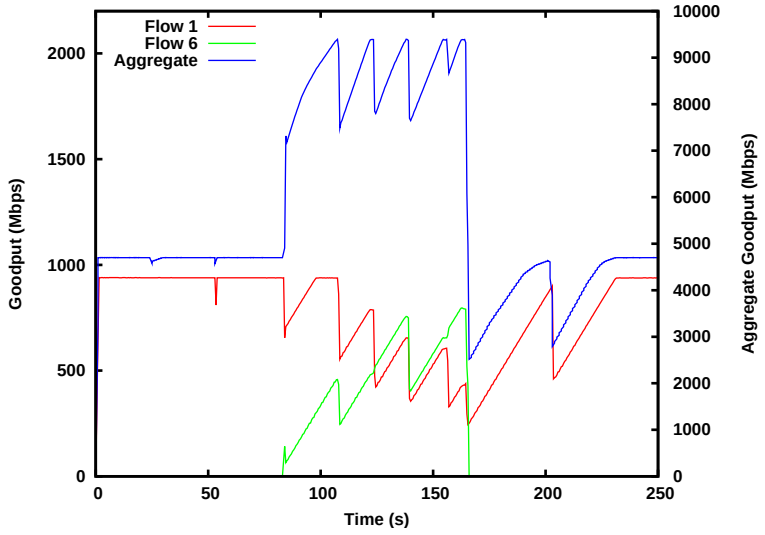


FIG. 4.9 – Goodput measurement for two flows obtained through a real experiment with NXE

proaches. In the case where a large queue size is used in the switch in the simulation (see Figure 4.8(b)), we can see that the aggregate goodput is a straight line for the whole duration of the experiment. Outside of the periods of congestion ([0-T_1] and [T_2-T_3]), the first flow is able to keep a constant goodput at 940 Mbps. During the congestioned period ([T_1-T_2], the two displayed flows get an equal share of bandwidth and are almost constant. It is mainly due to the fact that having such a large queue in the network can dramatically increase the RTT in the network and thus the feedback loop of TCP increases too. The saw-tooth of TCP are replaced by a slow sub-linear (logarithmic) increase.

Figures 4.8(a) and 4.9 show a different picture. The aggregate goodput in both cases is stable during the first part of the experiments. In the [T_1-T_2] period, the aggregate goodput presents a cyclic behavior with a succession of spikes at a regular interval. The frequency is however different, probably due to the fact that the queue size in both cases are different. Finally around T_2, it takes some time to get back to a more stable value. The flows singled-out in both graphs are clearly displaying the TCP's saw-tooth behavior. Due to the long duration of the NS-2 experiments, we did not try to find the queue size in NS-2 that would yield the closer match to the result with NXE but still we managed to get a similar behavior (or a different behavior depending on the point of view) by changing the queue size in the switches.

4.4.5 Synthesis

During an experimentation over a real network like PlanetLab, except for some particular cases like the WAN-In-Lab testbed, a fixed infrastructure is used and the user has limited means to change some of its characteristics like the RTT or the queue size. It limits the range of the parameters that can be studied in a real testbed. There are also configuration and hardware problems that can easily cast doubts over the reproductability or the truthfulness of the results.

During a simulation, the user is not limited by the infrastructure and can set up his system using the parameter's value of its choice. Of course some cases do not correspond to real situations as networking equipment vendors set the queue size of their switches to arbitrary values. For instance Cisco uses a queue size of one BDP for a RTT of 100 ms in its high-end routers. But trying different parameter values though simulations allows checking if a given protocol would perform better in a non-real/non-existing environment and proves that vendors should provide such an environment. By modelling the network at a L3 level, NS-2 does not taken into consideration some factors. The most prevalent one being that routers and switches are complex pieces of equipment that can not be simply modelled by queues. They often use complex algorithms when there is contention at an output port and we have shown that these arbitration algorithms have a deep impact on the upper layers as they can cause long periods of starvation [Soudan et al., 2007a]. These periods can last up to 300 ms and during them, the packets from a specific flow may delayed or discarded. It is then very difficult to extrapolate the complete behavior of a given transport protocol just by doing simulations. Simulations help in getting insights of the behavior of a particular algorithm like the Congestion Control but not the whole protocol. Each method can help to shed a different light on the behavior of a given transport protocol, with both their strong and weak points. They are complementary.

One interesting question would be to know if it is possible to verify some NS-2 results with NXE and vice-versa. Some part of the scenario are rather straightforward : describing the timeline of the experiment in one or the other description language is not too difficult if it is possible to find a similar application and/or to have a complete description of the distribution laws of the traffic used in the scenario and to extract all the software parameters that have an influence on the behavior of the protocol.

At the same time, other part will be trickier like the definition of the topology. Without a perfect knowledge of the real topology and of all the networking equipments it contains and their characteristics (especially the amount of buffering and the kind of algorithms used), it

might not be possible to construct a NS-2 topology that is close enough to the reality so as to get comparable results. In the testcase shown in the previous section, we were able to find the values for which it was possible to obtain a similar behavior in both cases. It might be more difficult to do so with a much more complicated topology for which there is no knowledge of every networking equipments on the path. Going from a NS-2 topology to a NXE topology might also be a hurdle if it is not possible to map directly the simulated topology over the real topology. It might then be necessary to use additional emulation means, like the Gtrc-Net boxes to modify the necessary characteristics (*e.g.*, RTT, loss rate) to have a closer match between the two topologies, hoping that they would not introduce a bias in the behavior of the transport protocol studied.

The need to define a similar topology and the need to know the complete software configuration limit a possible direct synergy between the two tools but it might still be possible to verify, perhaps not on a quantitative level but at least on a qualitative level, that the behavior of a given transport protocol is similar both in simulations and in real experiments. Such an information would put much more weight for the validation of a transport protocol and reduce the fears associated to its deployment in the wild.

In [Lucio et al., 2003], a comparison is made between the results obtained by two packet-level simulators (OPNET and NS2) and a hardware testbed. A similar conclusion is found as it highlights the importance of fine-tuning parameters and understanding the underlying assumptions of the models used if the user want to have results that matches with a good accuracy. If the user is only seeking a coarse approximate or a general behavior of the transport protocol, it might not be necessary to finely model every component of the network.

4.5 Protocol-Designer-Oriented Test Suite For the Evaluation of Transport Protocols

To conclude this chapter, we now present our contribution to the international effort for a standardized test-suite for the evaluation of transport protocols that can be used both in simulation and in real experimentation.

Over the years, researchers, especially protocol designers, have used their own sets of scenarios. But as it can be seen in Table 4.VI, there is no consensus on the kind of metrics, network model and workload models that should be used. Protocol designers might also decide to present a specific situation where their protocol "looks good" and overlook the situations where problemes might occur (like unstability or slow convergence speed). It is then very difficult to fairly compare each of these transport protocol with each other.

In [Andrew et al., 2008], we identify seven kinds of scenarios and the associated metrics that are the baseline for evaluating transport protocols. The main idea is to provide a common evaluation suite that would allow protocol designers to check the performance of their proposal against well-defined standard test cases. Such a test suite would allow the networking community to simplify the comparison of several transport protocols by providing a common ground for everyone, instead of relying on each protocol designer to provide the series of tests in which his own creation "looks good". Currently two partial implementations are available, one in the real testbed Wan-In-Lab [Lee et al., 2007] and the other one as a set of NS-2 scripts provided by NEC Labs [Wang et al., 2007].

However this test-suite should be considered as an "entry-level" exam for the transport protocol and does not aim at being exhaustive. After passing these tests, it might still be necessary to perform additionnal tests, especially at a larger scale in a real environment to prove the effectiveness of the transport protocol.

We contributed to this test-suite. More specifically, we provided inputs for the **Transients** scenario.

Methodology	Wang **[1]**	Mascolo **[2]**	Rhee **[3]**	Leith **[4]**	Kumazoe **[5]**
Type	Simulation	Simulation	Sw. emul.	Sw. emul.	Real
Topology	Dumbbell, Parking Lot, 4 Domain Network	Dumbbell	Dumbbell	Dumbbell	Dumbbell
Number of sources	n/a	6	4	2	2
Rate max (Mbps)	n/a	250	400	250	10000
RTT range (ms)	n/a	40,80,160	16,64,162, 324	16,22,42, 82, 162	18,180
Traffic model	FTP, Web, Voice Video streaming	FTP, Web	FTP, Web	FTP, Web	FTP
Metrics	X, q, σ_{RTT}, p, J, R, δ_f, robustness	cwnd, t	p, J, δ_f	U, G, cwnd, p, J	X

[1] : [Wang et al., 2007]
[2] : [Mascolo & Vacirca, 2006]
[3] : [Ha et al., 2006]
[4] : [Li et al., 2006]
[5] : [Kumazoe et al., 2007]

TAB. 4.VI – Existing methodologies to analyse the behaviour of transport protocols

Name	Edge links	Central link	Path RTT
Data Center	$>$ 1 Gbps	$>$ 1 Gbps	$<$ 1 ms
Access Link	100 Mbps	100 Mbps	4-200 ms
Trans-Oceanic Link	1 Gbps	1 Gbps	150-300 ms (delay on central link)
Geostationary Satellite	4 Mbps UL/40 Mbps DL	40 Mbps	600-800 ms (delay on central link)
Wireless Link	11 Mbps/54 Mbps	100 Mbps	4-200 ms
Dial-up Link	64 kbps	100 Mbps	14-210 ms

TAB. 4.VII – Network models used in the TMRG test suite

4.5.1 NETWORK MODELS

The topology used in every test is based on a dumbbell topology composed of three pairs of nodes that will inject the useful traffic. On this topology, six different network models are instanciated, so as to be representative of different current-day networking environments : from the high-speed low-delay networks that are commonly found in data centers to the asymmetric low-speed high-delay links from networks composed with geostationary satellites. A protocol designer can select a subset of them depending on the particular usecases he is targeting. Table 4.VII summarises the characteristics of the proposed network models. For all models except the data center, a mix of RTTs is used by setting a different one-way propagation delay for every edge link. Additional parameters like the buffer sizes used in the routers are specified in the scenarios.

In the case of the instanciation of these network models in real networks, it might be necessary to combine experimentation and emulation (software or hardware) to have a similar mix of RTTs. It is also possible to use a larger number of nodes (for instance to generate the background traffic to have a more realistic large-scale system), as long as the properties of the traffic injected presented in the next section are kept.

4.5.2 SCENARIOS AND METRICS

This section presents the seven types of scenarios that have been identified and presents their main characteristics. The background traffic is generated using the Tmix traffic generator from real traces. Some of the scenarios are very similar (for instance convergence times and inter-RTT fairness) but the fact that they are using a different network model (*e.g.*, same RTT or not), different workload model (*e.g.*, some background traffic or not) and different metrics gives a new focus on a particular property of the transport protocol.

Basic : the goal of this scenario is to test the behavior of the transport protocol within the six proposed network models. Three levels of congestion are set as a background traffic : uncongested, mild congestion and moderate congestion[4]. The reverse path is set to a 10 % load. The performance is evaluated through throughput, delay and loss metrics on the central and edge links. Stability metrics are also used for the individual flows.

Delay/Throughput tradeoff as function of queue size : the goal of this scenario is to evaluate the behaviour of the transport protocols when the routers use a specific AQM scheme. Different buffer sizes, set as a fraction of the BDP of a 100 ms RTT flow, are used. The metrics of interest are the throughput and the drop rate as a function of the queueing delay.

Convergence times : the goal of this scenario is to study how two long-lived flows from a given transport protocol, one in equilibrium and one arriving in the network, will be able

[4]The actual values are still to be defined according to the latest version of the draft RFC [Andrew et al., 2009]

to share the bandwidth. A 10 % bidirectional background traffic is used. The metric used is the time necessary for the new-coming flow to transfer a given number of packets.

Transients : the goal of this scenario is to study the impact of a sudden change in congestion level. Three cases are considered : brutal decrease (75 % to 0 % of bottleneck capacity), brutal increase (0 % to 75 % of bottleneck capacity) and step increase (+2.5 % every second). The transient traffic is generated using UDP. The metrics of interest are the time required for a flow using the tested transport protocol to increase back to 80 % of its maximum value and the number of packets dropped by the background traffic.

Impact on standard TCP traffic : the goal of this scenario is to quantify the tradeoff between the gain of throughput of a tested transport protocol and the loss of throughput of standard TCP flows. Two sets of identical flows are defined, one for TCP and the other for the tested transport protocol. The aggregated throughput of each set is compared to the case where both sets use TCP.

Intra-protocol and inter-RTT fairness : the goal of this scenario is to evaluate the way the studied transport protocol shares a bottleneck for flows with the same RTT or for flows with different RTTs. Two long-lived flows are started at random times. The ratio between the average throughput of the two flows is compared.

Multiple bottlenecks : The goal of this test is to test the behavior of flows of a given transport protocol when they encounter multiple bottlenecks. For this purpose, a parking-lot topology with three bottlenecks (see Figure 2.4(b)) is used instead of the dumbbell topology. A 10 % bidirectional background traffic is used. The throughput of flows going through multiple bottlenecks is compared to flows that are only going through one bottleneck.

The number of useful flows is mostly low (two in most scenarios), as the main goal is to provide a comparison ground for protocol designers. It is more easy to analyze the behavior of a given transport protocol when two flows interact than when thousands of flows are involved. The last scenario (**Multiple bottlenecks**) provides a more realistic experiment, through the use of multiple bottlenecks and a larger number of active flows.

4.5.3 Synthesis

This section has presented a test-suite oriented for the protocol designers with the goal of providing a common ground for the comparison of transport protocols. Six types of networks (seven with the multi-bottleneck topology) are used to confront the transport protocol to a large panel of network conditions. Seven kind of scenarios (and their associated metrics and workload models) are provided to identify if the tested transport protocol has the required properties of convergence, fairness, *etc.*

This already corresponds to a large set of experiments. One of the design concepts proposed by Sally Floyd for the test-suite, was to have a set of scenarios that could be executed in atmost two days in NS-2. This highlights the need for automatic tools to execute and analyze such a large quantity of experiments, especially in real networks.

Conclusion

We have presented the need for large-scale experiments due to inherent changes both in the infrastructure and in the composition of the application mix of the Internet. These large-scale experiments require helper-software to execute them : automating the tedious tasks, ensuring the reproductibility of the scenarios used and providing a workflow for experience deployment in real environments. Our proposal, NXE, is designed to do so.

Our other contributions in this chapter were on a methodological level. We have presented a scenario that highlights the risk of a congestion collapse if users start to massively migrate

to unresponsive transport solutions, like UDP or based on multiple parallel streams (these solutions are frequently used in most P2P applications) in a network where the multiplexing factor and the aggregation level are low. It also introduced the joint effort of the TMRG workgroup of the IRTF to provide a base set of tests to protocols designers, in order to have a well-defined common ground for comparing transport protocols. Both these contributions would benefit from automated tools to perform experiments as they either involve large sets of end-hosts to control or large range of parameters to study.

Most of the techniques presented in Chapter 3 are mostly adapted for the use of the protocol designers and of the network providers. End-users can only rely on real experiments that are by essence difficult to analyze, even if they can manage to execute their experiments with helper-software such as NXE. End-users, including the network-application designer subtype, need tools (especially if they automated and ready-to-use) to assess the impact of a given transport protocol on their applications. The next chapters will now focus on the end-users by providing benchmarking transport layer solutions, specially suited for the end-users.

— Five —

Benchmarking Transport Layer solutions

Introduction

In [Low et al., 2005], Low *et al.* highlight the fact that among all the new Transport Protocols variants that have been introduced to solve the performance problem of TCP over LFNs, there is no clear winner, no solution that could TCP's replacement as the "one-size-fits-all" transport protocol. It makes it all the more necessary to test each of them in the same conditions to assess if they have the required properties (stability, fairness, *etc.*). Table 4.VI has shown that even protocol designers didn't have, until recently, a common set of scenarios against which they could compare their respective creation.

It is then necessary to define standardized benchmarks or test-suites that will allow to compare easily the large quantity of transport protocols available with each other. Such a test-suite designed for protocol-designers has already be presented in Chapter 4, Section 4.5. This same chapter has also highlighted the difficulties of performing large scale experiments in real networks and proposed a fully-automated tool to execute these experiments.

Other types of actors, especially the end-users, also require comprehensive tools for them to study the characteristics of a given transport protocol in their current networking environment. Traditionnally, when end-users want to perform experiments at a large scale, their focus point will be on the application they want to use and on defining a network model that will have the wanted properties (especially in terms of scale). But in doing so, they can neglect other factors, such as the impact of the Transport Protocol layer on the performance.

Indeed, the "wizard gap" [Mathis et al., 2003] is a serious problem that can account for up to 90 % of the performance defiency (as we will show in Section 5.2.3.1). It designates the difference between the performance that is "theoretically"' possible on today's high-speed networks, and the performance that most users actually perceive. The idea is that today, the "theoretical" performance can only be (approximately) obtained by "wizards" with superior knowledge and skills concerning system tuning.

End-users are seldom in the category of "wizards" and will require the assistance of a more advanced user or of automatic tools to properly configure the path prior to its use (more details about this particular point will provided in Chapter 6)

This chapter presents the general principles of benchmarking and designing a test-suite in Section 5.1. Section 5.2 will then present our proposal for a user-oriented test-suite and provide some initial results in the Grid'5000 testbed. The case of bulk data transfers is further detailed in Section 5.3 through the analysis of the predictability of transfer time of different transport variants and of the impact of using multiple parallel streams on the performance and fairness.

5.1 Benchmark and Test-suite Design

Many alternatives to TCP have been proposed to overcome the limitations of TCP in the context of very-high-speed wide area networks. The goal of this section is to illustrate the transport protocol principles and the different points of view via the presentation of the design of two series of scenarios or tests. We take the perspective of end-users and their difficulty to compare different transport protocol solutions quantitatively.

5.1.1 Definition and goals

A benchmark is a program, or a set of programs, that calculates the relative performance of a machine or an architecture in the case of a hardware benchmark, or of another program in the case of a software benchmark. Each benchmark may either focus on quantity (execution speed, amount of data computed, *etc.*) or on quality (robustness, security, *etc.*). Benchmark can be executed at two different levels [Ehliar & Liu, 2004] :

Microbenchmarks : these benchmarks are dedicated to testing the performance of one particular component or function at a low level, like the rendering capacities of a video card or the number of instructions that can be executed by a CPU.

Application level : these classes of benchmarks aim at representing typical applications or workloads of a platform that needs evaluating. Among the existing High Performance Computing (HPC) benchmarks, the NAS Parallel Benchmark (NPB) [Faraj & Yuan, 2002] is well known. This set of programs represents the typical applications classes executed on clusters. The NPB uses six classes to represent the size of the problem, using different sizes of input data arrays. They have different kinds of communication schemes representing typical parallel applications.

5.1.2 Guidelines for transport protocol benchmarking

In the case of transport protocols, we first have to identity representative applications (or scenarios) that can be used to capture the needs of important classes of applications (or scenarios). Applications that are part of a benchmark should be :

Easy to use : no tuning or modification of application should be necessary to use the benchmark.

Representative : the benchmark applications should be representative of user applications.

Portable : the benchmark is usable on a large variety of machines or environments.

Reproducibility : running the same experiment several times will yield similar results every time.

Well-defined : it is necessary for the benchmark to be well-defined, to have a real support for design and development.

The service offered by the system to be evaluated is the transport of flows from a source node to a corresponding sink node in the context of high performance distributed applications.

A flow in the Internet is a loosely defined object representing a stream of packets having some criteria in common (IP addresses, port numbers, etc.). Most end users are interested in the transfer of a particular message, file or document. The document might be a Web page, an in-line object, a data file or an MP3 track. The defining feature is that the flow is manifested by a more or less continuous stream of packets using a considered network link or path. The flow may consist of several overlapping TCP connections pertaining to the same document or by one period of activity of a sporadically used long-term TCP connection. The object to transfer is characterised by its starting time and its size in bits.

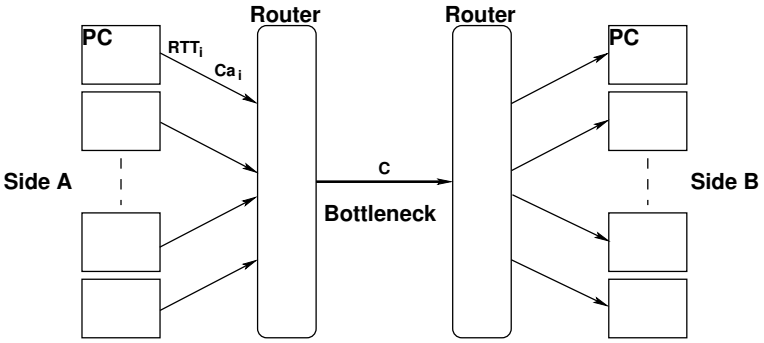


FIG. 5.1 – Dumbbell topology used for the experiments

5.2 User-Oriented Transport Protocol Test-suite

Table 3.II has shown that only a few evaluation methods are suitable for the needs of the end-user. The one that is the most accessible to the end-user, experimentation in uncontrolled environments, is a difficult path to follow as it requires not only to setup these experiments (Chapter 4 has presented a tool that allows to set-up easily experiments in real networks) but also to define scenarios that are suitable for the study of an application using a transport protocol.

But just as there was no consensus on the kind of scenario that should be used by protocol designers to assess the basic properties of their transport protocol until recently (see Chapter 4.3.1.2, Section 4.5), they are no ready-made test-suite for end-users.

In this section, we present metrics and scenarios, oriented towards end-users. Representative applications like bulk-data transfer ou MPI application are used to capture and compare the behavior of different transport protocols in the context of the end-user's networking environment.

Unless mentioned otherwise, the topology used in the following examples is as seen in Figure 5.1 : a dumbbell with a single bottleneck, usually located in the output port of a L2 switch. The nodes on each side are paired together to form an independent client-server couple. In the testbed used, C_a is instantiated to 1 Gbps and C equals 10 Gbps. All RTTs are uniform. All these exemples are based on experiments performed in a controlled environment, the Grid'5000 testbed.

5.2.1 Metrics

To analyze the data acquired during the tests, several metrics are used to synthetically characterize the behavior of the transport system. The IRTF TMRG group [Floyd, 2008] has identified several metrics for protocol evaluation : fairness, throughput, delay, packet loss rate, variance of goodput, utilization, efficiency, and transfer time. Among these, four of them are particularly suited to summarize the important properties from an end-user perspective : the transfer completion time of a given workload, the predictability of the transfer time, the fairness, and the efficiency.

By evaluating the fairness, the end-user will know if a transport protocol or service (a transport service is an application built on top of a transport protocol whose main purpose is to transfer data between two end-points of the network, for instance FTP) has a selfish or a cooperative behavior. Most end-users do not have any incentive to use unfair transport protocols in a shared network because such solutions are likely to degrade eventually the performance for everyone. In a private network, end-users can choose whatever solutions he wants (even unfair ones) because they can only hurt themselves. But they still need to know which solutions are unfair as a forewarning to any potential problem.

Predictions of large transfer throughput can be used in path selection for overlay and multi-homed networks, dynamic server selection and peer-to-peer parallel downloads. The predictability is measured by the standard deviation in completion time. Finally, the efficiency allows the user to know if the protocol used is able to fully take advantage of the networking resources. It is defined as the ratio between the goodput achieved by the transfers and the bottleneck capacity of the network.

Formally, the four metrics we selected for the test-suite are defined as follows :

- **Mean completion time :** $\overline{T} = \frac{1}{N}\sum_{i=1}^{N_f} T_i$
- **Standard deviation of completion time** $\sigma_{T_i} = \sqrt{\frac{1}{N}\sum_{n=1}^{N_f}(T_i - \overline{T})^2}$
- **Fairness :** $J = \frac{(\sum_{i=1}^{N}\overline{T_i})^2}{N(\sum_{i=1}^{N}\overline{T_i}^2)}$
- **Efficiency :** $E(t) = \frac{G(t)}{C}$

where T_i is the completion time of the i^{th} file transfer, N the number of sources participating on the forward path and G the aggregate goodput.

5.2.2 Network model

This test-suite is end-user-oriented and focuses on the execution of real experiments to compare the behavior of different transport protocols. As such, it assumes that the end-user would have a limited knowledge and impact on the topology and on the infrastructure parameters of the network model. If the test involves sets of nodes from data-centers or from grids, it can be assumed that the topology used is a dumbbell and that each node taken from a set of nodes has the same characteristics. For more general cases, it may not be possible to know the exact topology and some of the network parameters, such as the bottleneck capacity.

The hardware of the tested end-nodes needs to be taken into account as well. Some of our previous experiments [Guillier et al., 2007a] were seriously disturbed by a fault in the Base Board Management controller firmware of IBM e-server325 nodes. Such problems should be identified using a calibration application, such as the calibration test proposed within this test-suite before launching additional experiments on a larger scale.

5.2.3 Test suite design

The test-suite consists in a series of tests that aim at capturing the characteristics of major applications. These different applications that are defined by the mix of useful traffic applied to the network. The adverse traffic can be either the current level of background traffic in the network or generated by the user to match a given level. Preliminary results are provided in the case of the MPI application and Bulk data application in the Grid'5000 testbed.

5.2.3.1 Calibration test

The calibration test enables end-users to check if all the configuration parameters from the *txqueuelen* (the size of the queue used by GNU/Linux kernels to transmit packets to the network interface) to the TCP buffer sizes are properly set, to get the best performance out of their hardware infrastructure.

The calibration test consists in measuring the average goodput achieved by a given transport solution over a fixed period of time and comparing it to a theoretical TCP that is able to send at full speed all the time, without slow-start, losses, or retransmissions. By counting the overhead of the TCP headers (including the timestamp option), the IP headers, and the Ethernet overhead (including the headers and the Inter Frame Gap), this TCP should operate at 941.5 Mbps on a 1 Gbps Ethernet link.

Table 5.I and Table 5.II, taken from [Guillier et al., 2006], highlight the need to perform this calibration test. In Table 5.I, the average goodput (averaged over 300 s of continuous transfer) matrix for a single not-tuned TCP flow between two 1 Gbps nodes on different Grid'5000 sites is shown. The experiment was performed using the default system image of the considered

		Source								
		Bordeaux	Grenoble	Lille	Lyon	Nancy	Orsay	Rennes	Sophia	Toulouse
Destination	Bordeaux		58.1	61.8	55.9	81.2	111	76.3	68.9	181
	Grenoble	32.3		34.0	151	39.8	33.7	34.3	52.6	48.4
	Lille	53.3	70.0		53.6	112	199	55.0	44.3	33.9
	Lyon	61.5	230	71.2		97.6	106	49.8	100	72.0
	Nancy	48.0	162	78.5	52.4		777	54.7	43.3	32.0
	Orsay	67.8	54.1	150	58.8	936		68.7	36.2	50.8
	Rennes	64.2	33.6	46.6	41.4	45.5	56.5		27.4	26.3
	Sophia	47.0	46.1	29.5	67.4	28.9	22.3	25.1		34.0
	Toulouse	166	47.6	29.8	65.7	29.7	44.3	26.3	36.1	

TAB. 5.I – Average goodput matrix for a single not-tuned TCP flow between two 1 Gbps nodes on different Grid'5000 sites

		Source								
		Bordeaux	Grenoble	Lille	Lyon	Nancy	Orsay	Rennes	Sophia	Toulouse
Destination	Bordeaux		771	725	862	911	884	852	875	685
	Grenoble	900		701	925	812	893	787	911	647
	Lille	738	838		120	922	848	916	598	579
	Lyon	425	912	786		904	740	864	926	730
	Nancy	725	851	742	865		854	938	931	622
	Orsay	799	866	777	869	936		849	878	523
	Rennes	912	831	787	859	914	912		839	651
	Sophia	901	839	653	543	611	900	321		694
	Toulouse	928	859	784	882	933	923	939	909	

TAB. 5.II – Average goodput matrix for a single tuned TCP flow between two 1 Gbps nodes on different Grid'5000 sites

sites. Some of them such as Orsay and Nancy had a different initial configuration (*i.e.*, TCP buffers set a higher value than the GNU/Linux default maximum value of 256 KB) which explains their better result. For most of the sites' pairs, the average goodput is well below 100 Mbps. It corresponds to a performance defiency of more than 90 %.

In Table 5.II, the same test was performed after setting the TCP buffer size to a more appropriate value. It set to 4 MB, a value larger than the BDP of each possible path (as the maximum RTT in Grid'5000 is a little bit more than 20 ms). In this case, we can observe a clear improvement of the performance. In most of the cases, the average goodput is above 800 Mbps, which corresponds to a performance defiency of "only" 15 %.

This calibration test also allowed to anticipate some problems. For instance, it was shown that Grid'5000 Sophia site had some performance issues in their LAN due to some choice made for the topology.

This test should be run systematically to calibrate the performance achievable on both sides of the network, with the potential limitations generated by the MTU sizes, buffer sizes, loss rates *etc.*Typical signatures are proposed to identify any configuration limitations. The appropriate tuning has to be done before starting the other tests of the suite.

5.2.3.2 WEB APPLICATION TEST

This test aims at replicating the behavior of a Web application. The nodes transfer a succession of random-sized files. The file sizes can be exponentially distributed or heavy-tailed [Barford & Crovella, 1998] depending if the user wants to consider objects like web pages

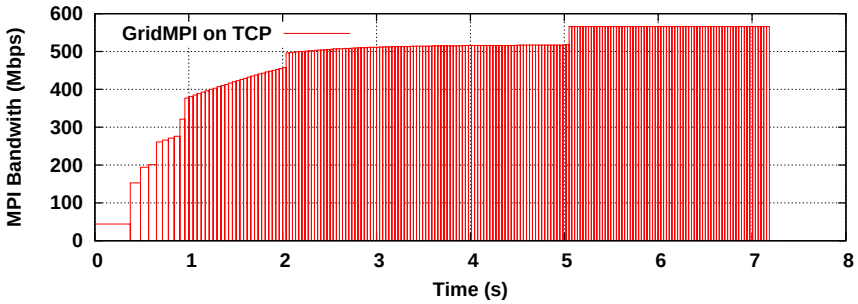


FIG. 5.2 – Impact of the TCP behavior on the MPI traffic

that are typically small or larger objects like music files or PDF documents. Inter-arrivals are following a Poisson law. To match the asymmetry of the web traffic (most of which goes in the server-client direction), the ratio between forward-path and reverse-path traffic is set to 0.8. This test helps observing the behavior of the transport protocol against a succession of small transfers.

5.2.3.3 Peer-to-Peer application test

This test aims at emulating the behavior of a Peer-to-Peer application through transfers set up between random nodes in the network. In this sense, the file sizes used are heavy-tailed and the packet sizes has a large proportion of the maximum packet size. The ratio between forward-path and reverse-path is 0.5, which corresponds to typical P2P behavior (sending as much as you receive). If asymmetric networks are used, like ADSL, it is possible to modify the ratio to better capture the asymmetry of the network (typically the uplink speed/downlink speed ratio is between $\frac{1}{20}$ and $\frac{1}{3}$). This test corresponds to the study of the transport protocol agaisnt large bidirectionnal transfers.

5.2.3.4 MPI application test

MPI is the standard communication library used to write parallel applications. GridMPI[1] implementation has been designed to optimise long-distance communications in MPI applications. One of the most critical part of these applications is the dense exchange of small messages between processes composing the parallel application.

In the test-suite, the MPI application (PA) is implemented as a MPI ping-pong of 200 messages of 1 MB between two nodes on two different sites. This application explore the dynamics and the slow-start effect of the evaluated protocol.

Figure 5.2 represents the goodput for each of the 200 messages as function of time for an execution of the MPI application test between two sites separated by 11.6 ms RTT. On this figure, we can see the impact of TCP behavior (slowstart and congestion avoidance) on the MPI implementation.The slowstart and the congestion avoidance occur on each node. The transfer of 200 messages takes 7.2 s. Due to the impact of Slow Start and congestion avoidance mechanisms on GridMPI, the maximum bandwidth is only reached after 5 s. Without these mechanisms, the total transmission time would have been less than 6 s.

5.2.3.5 Bulk data transfer test

This test aims at replicating large data movements (several Terabytes) from sets of end points to other sets for long durations (for hours or days). The system to be evaluated is a transport service offered by a transport protocol and executed on a complex network infrastructure. The user running this test-suite wants to compare the performance and the behavior of several transport services for transferring bulk data on this infrastructure.

[1] Doc. and code available at http://www.gridmpi.org/gridmpi-1-1/

The traffic injected in the system is a composition of the analyzed traffic (the forward and reverse traffic part which is evaluated). The adverse traffic (the one which is supposed to perturb the protocol : forward and reverse background traffic) can be injected artificially, if it is not already present in the network. The bulk data transfer application (BU) consists of simultaneous unidirectional large file transfers (typically 30 GB).

5.2.4 Synthesis

In this section, we have presented our proposal for an end-user-oriented test-suite. With the help of metrics meaningful to end-users (such as the transfer time) and with the definition of four typical applications (Web, Peer-to-Peer, MPI and Bulk data), it provides an easy-to-use tool for end-users to compare the performance of different transport solutions in various system and workload conditions.

We also highlighted the importance of calibration and fine-tuning of the environments in which the experiments take place. If not, the results will be extremely bad, which can only lead to the frustration of the end-users ("we paid for 1 Gbps and we are only getting 50 Mbps"). These aspects will be detailed further in the next chapter.

The next sections will further detail the rationale for the bulk data transfer test parameter choice. We explore some questions of interest for the end-user related to the workload model used such as the impact of congestion on the forward path, or of the congestion on the reverse path on the predictability of a transfer performed with a given transport protocol. Other aspects like the impact of multiple parallel streams on the performance and the fairness are also considered.

5.3 Bulk data transfers experiments

Bulk data transfer is a concern for a vast number of end-users, especially researchers in physics or astronomy that need to collect for analysis purpose the results of experiments or measurements generating large amounts of data. One of the typical examples is CERN's LHC[2] that is foreseen to generate petabytes (10^{12} Bytes) of data every year. Such huge amount of data can not be analysed locally and will require the use of data grids (like the LCG[3], high performance and large scale distributed computational and storage systems, to perform these tasks. But to be successful, the movement of these sets of data has demanding performance requirements such as reliable and predictable delivery [Foster et al., 2004, Vicat-Blanc Primet, 2003, Ferrari et al., 2004].

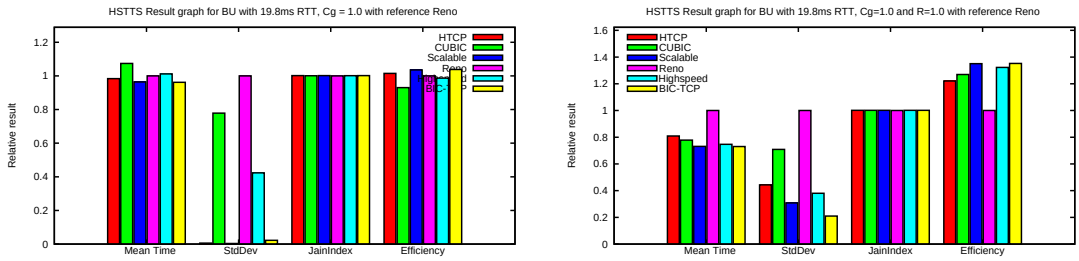
This was also one of the main goals of the French-government-funded IGTMD project[4] that aimed at designing, developping and validating mechanisms that make the interoperability of heterogeneous grids a reality. The key idea was to fully exploit the specificity of LCG applications and their real infrastructures to analyse and experiment new communication and replication models, alternative transport protocols emerging within the international scientific community.

The next sections will further detail the bulk data transfer test of the test-suite by exploring some questions of interest for the end-user related to the workload model used : what is the impact of congestion (on the forward and on the reverse path) on the predictability of a transfer performed with a given transport protocol ? What will be the impact on performance (and on the performance of others through the study of fairness) if multiple parallel streams are used to perform a transfer ?

[2]See the LHC project web-page http://public.web.cern.ch/public/fr/LHC/

[3]See the LCG project web-page http://lcg.web.cern.ch/LCG/

[4]See the IGTMD project web-page http://www.ens-lyon.fr/LIP/RESO/Projects/IGTMD/ProjetIGTMD.html



(a) $C_g = 1.0$ (b) $C_g = 1.0$ and $R = 1.0$

FIG. 5.3 – Bulk data transfer test results between Rennes and Toulouse clusters

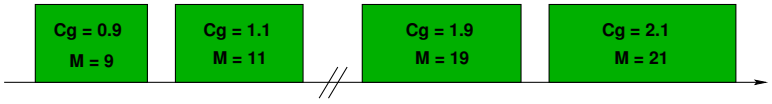


FIG. 5.4 – Scenario for the completion time predictability experiment

5.3.1 BU TEST RESULTS

Figures 5.3(a) and 5.3(b) present the results of two separate executions of the BU test between the Rennes and Toulouse clusters, normalized by the results achieved with Reno. The difference between the two runs is that in the second case, a congesting reverse traffic has been injected into the network ($C_g = 1.0$ and $R = 1.0$). The reverse traffic is generated using constant bit rate UDP flows.

As far as the fairness in concerned, the differences between the protocols in both experiments are very slight. Reno yields the best fairness among all the TCP variants, even though the difference is less than 0.1 %. In the absence of reverse-traffic, Scalable presents both better mean completion time (4 % faster than Reno) and standard deviation (99 % smaller than Reno) than the other protocols, while maintaining a Jain fairness index (0.1 % more than Reno) close to Reno's.

In the experiment where there is a congesting reverse-traffic (Figure 5.3(b)), the differences with Reno are much bigger, even though all the other TCP variants manage to have very similar mean completion times. Here, Scalable, HighSpeed, and BIC-TCP are all about 25 % faster than Reno. BIC-TCP exhibits the smallest standard deviation (80 % smaller than Reno). BIC is thus a good candidate for massive data transfers in a network where there is congesting traffic in both directions.

5.3.2 IMPACT OF CONGESTION ON THE PREDICTABILITY OF TRANSFER TIME

One of the most important metrics from the end-user's point of view is the one that indicates the time needed to transfer a given volume of data, for instance a video file [Dukkipati & McKeown, 2006]. The predictability of this completion time is also very important as it will allow users to efficiently schedule their transfers. In this section, we present an experimental study [Guillier et al., 2007b] of the impact of different workload parameters (*e.g.*, the congestion level and the reverse traffic level) on the completion time of a fixed-size transfer.

5.3.2.1 SCENARIO

Figure 5.4 presents the scenario that is used in this section. Three parameters are modified between successive runs of this experiment : the congestion level C_g and the multiplexing

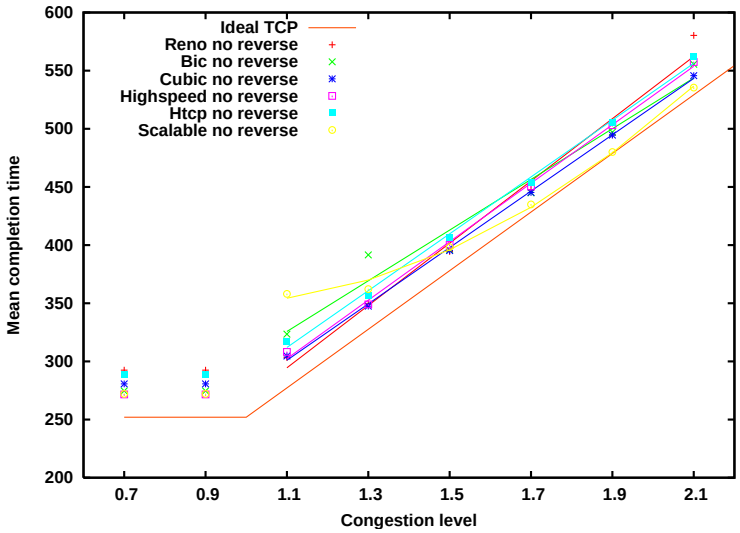


FIG. 5.5 – Impact of congestion level on the mean completion time for some TCP variants

factor M, modified through the number of nodes N_f that are taking part in the experiment on the forward path (from side A to side B in Figure 5.1) and the reverse traffic level R modified through the number of nodes N_b that are taking part in the experiment on the reverse path (from side B to side A in Figure 5.1).

The workload consists of identical fixed-size memory-to-memory 3000 MB transfers performed using *iperf* to simulate the transfer of a large database or of the content of a DVD. A single TCP variant is used during an experimental run. The RTT of the path is 19.8 ms.

5.3.2.2 RESULTS

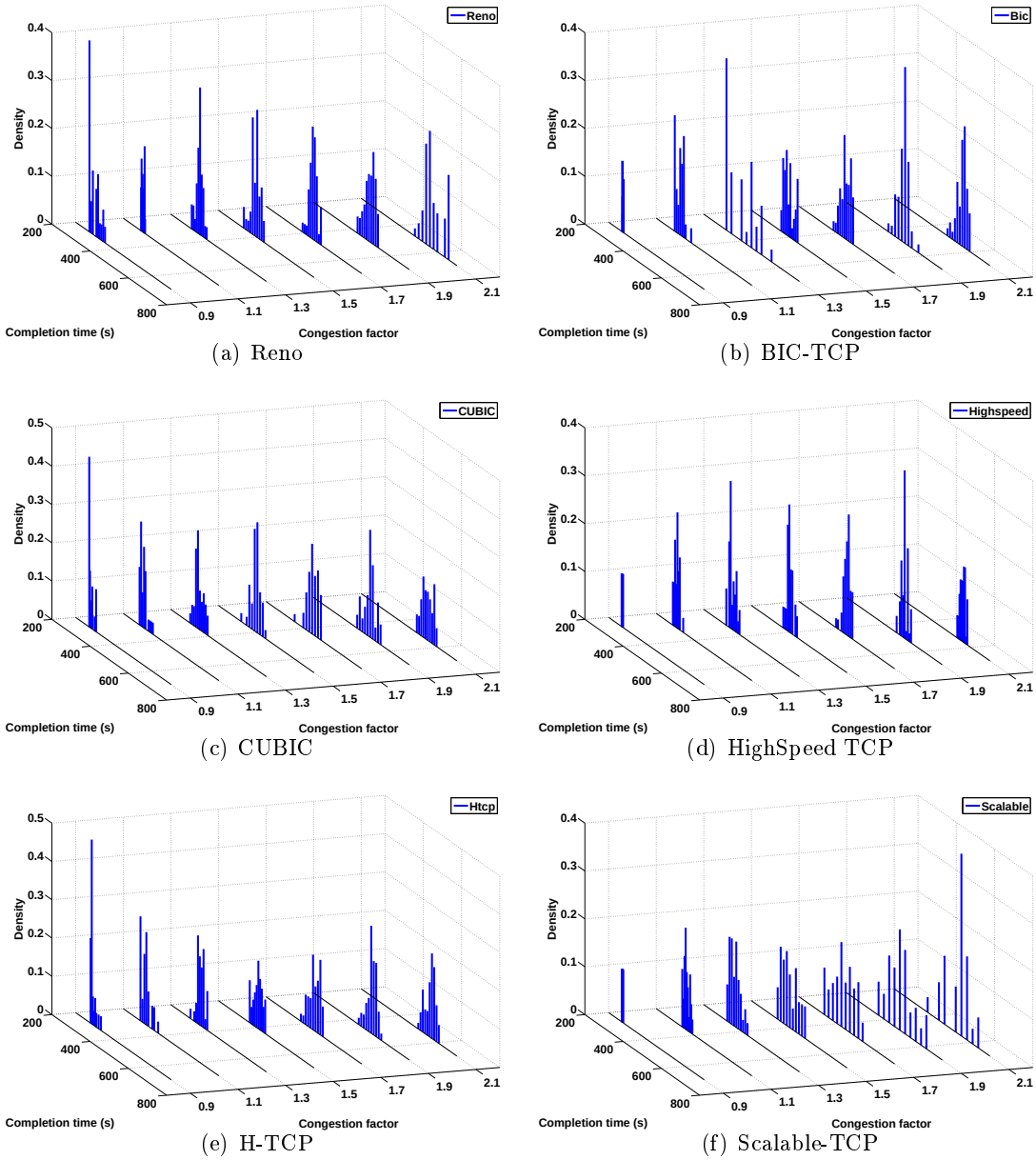
In this section, we present the results of the completion time predictability experiment by comparing the performance of different TCP variants when facing different levels of congestion.

(1) IMPACT OF THE CONGESTION LEVEL

Figure 5.5 presents the impact of the congestion level on mean completion time for several TCP variants : Reno, BIC-TCP, CUBIC, HighSpeed TCP, H-TCP and Scalable-TCP. The ideal TCP represented on the same figure corresponds to a TCP able to send continuously over 1 Gbps links, without slow start phase, without losses or retransmissions, and with equal sharing of the bottleneck link. All protocols, except Scalable, behave similarly : a previous work [Guillier et al., 2007a] has shown that for the RTT used in our experiments here, most TCP variants tend to have similar performance. When there is no reverse traffic, the TCP variants present a linear behavior with respect with an increase of the number of simultaneous transfers and the increase of congestion level it generates.

Figure 5.6 presents the completion time distribution of all the TCP variants. Scalable is somewhat remarkable in that it often displays the shortest and the longest completion time for a given N_f. Even though both distributions are roughly Gaussian-shaped, Scalable is more spread out (294 s *vs.* 114 s for the 2.1 congestion level case) than CUBIC for instance. It makes Scalable a poor choice if we need to wait for all transfers to complete. But if we can start computation on a limited dataset (like a DNA sequencing), we might be able to increase the usage of the computation nodes. It might not be the case in other applications like astronomy interferometry that will need full transfer of all images before the start of a computation phase. It seems that HighSpeed TCP is the best choice if we are interested in good predictability, as it has the lowest dispersion of all protocols for high congestion levels.

Figure 5.7 presents the evolution of the completion time CoV for all the TCP variants



(a) Reno (b) BIC-TCP

(c) CUBIC (d) HighSpeed TCP

(e) H-TCP (f) Scalable-TCP

FIG. 5.6 – Completion time distribution for several TCP variants, 19.8 ms RTT

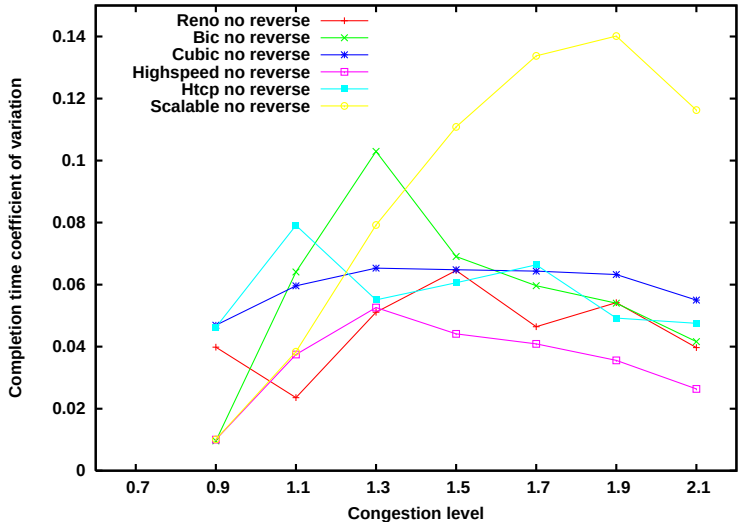


FIG. 5.7 – Evolution of the completion time coefficient of variation as a function of the congestion level for several TCP variants

tested. Here we can see that they all display the same kind of tendency as they all seem to follow a parabola as the congestion level increases. The apex of the parabola seems to depend on the protocol. This behavior might indicate that there is a congestion level/multiplexing level region in which we should not be so as to minimize the variability of our completion time (and thus increase the predictability).

We can also note that the CoV of most TCP variant stays below 6 %. This means that if we are not able to control the way transfers are started to ensure that we are well under the congestion level, we would have to consider at least a 6 % margin on an estimated completion time to be sure not to fail the deadline in the case when there is no reverse traffic. If we assume that the distribution of the completion time is indeed Gaussian, using such a margin would provide a 68 % confidence interval for the completion of our transfers. If we want a more precise (say 95 % confidence interval), we would need to push the margin up to 12 %. But adding such a big margin is not the best solution, especially if the transfers have very strict time windows and if we want to be efficient.

(2) INFLUENCE OF THE REVERSE TRAFFIC LEVEL

The reverse traffic in this experiment consists of large 30 GB file transfers, as on the forward path. The reverse transfers are started after the forward transfers with the same interval of 1 s to prevent interactions during the slow start phase. Here we are studying the impact of the reverse traffic on the completion time for transfers on the forward path.

For instance, for CUBIC (Figure 5.8), reverse traffic level for values lower than 1.0 has a limited effect on the mean completion time (about 2.5 %). The fluctuations observed for a 0.9 reverse traffic level are mainly due to the fact that we are close to the congestion point of the system and thus to a very unstable point. When the reverse traffic is congesting, we observe that the difference with the case without reverse traffic is much more important (about 10 %).

In this case it seems that for most TCP variants, having reverse traffic might be a good thing as we can observe lower CoV than in the case where there is no reverse traffic, that is to say less variability. It may not be enough to determine which conditions are optimal to achieve the best performance possible in terms of completion time as the CoV is inversely proportional to the mean completion time.

Most protocols, except BIC, have a CoV that remains below 6 % under most reverse traffic conditions. So again with a reasonable margin, all protocols might be able to finish within

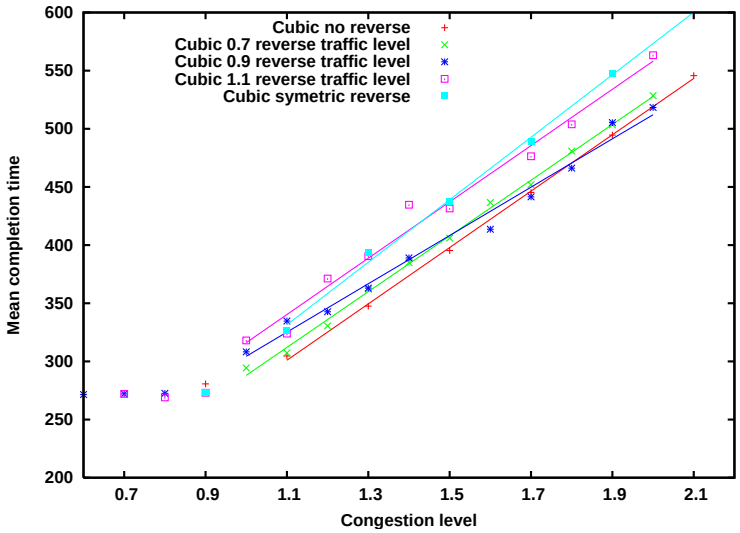


FIG. 5.8 – Impact of reverse traffic level on mean completion time for CUBIC, 19.8 ms RTT

their deadline. A margin of about 15 % should be added to the estimated mean completion time if there might be congesting traffic on the reverse path. It might be a major drawback as in this case, resource usage will probably not be optimal. The results for the other TCP variants can be found in the research report [Guillier et al., 2007c].

5.3.2.3 SYNTHESIS

The section has examined the impact of a range of factors on the transfer delay predictability in classical bandwidth sharing approach proposed by high-speed TCP variants. It has shown that the congestion level has a linear impact on the average goodput achieved by simultaneous bulk-data transfers. Transfer time efficiency and predictability depend on the chosen protocol. The factor that has the most impact is the reverse traffic level. This factor is usually not taken into consideration in analytical modeling but can still degrade the average goodput by more than 10 %.

Depending on the final application targeted and its requirements, the user can use this scenario to choose the TCP variant which is adapted to its need of a fast or predictable transfer. However, this study highlights the need for a flow scheduling service [Soudan et al., 2007b] available in the network to regulate the congestion in the forward and reverse path to ensure a good transfer time predictability to high-end applications. With such a mechanism, it would be possible to indicate to VoD users if it would be better to start a video transfer earlier to avoid congestion periods. Or as shown in [Laoutaris et al., 2009], take advantage of diurnal patterns of traffic to schedule massive transfers (such as the databases required by a scientific application).

5.3.3 INFLUENCE OF NUMEROUS PARALLEL STREAMS

As seen in Chapter 1 Section 1.3, using multiple parallel streams is a possible way to improve the performance of TCP without changing the transport protocol. This technique has also been used to overcome some configuration problems like insufficient TCP buffering capacity in the end-hosts. This experiment explores the impact of numerous parallel streams on the global throughput of the system. These results can be found in [Guillier et al., 2007a].

5.3.3.1 SCENARIO

Figure 5.9 presents the sequence of experiments used for the evaluation of the impact of multiple parallel streams over the average goodput of flows. The idea is to use a fixed

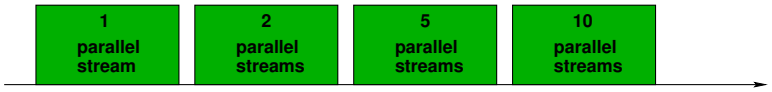


FIG. 5.9 – Scenario description for the evaluation of multiple parallel streams experiment

Number of streams by node	**1**	**2**	**5**	**10**
Total number of streams	11	22	55	110
Mean total goodput (Mbps)	8353.66	8793.92	8987.49	9207.78
Stream mean goodput (Mbps)	761.70	399.83	163.53	83.71
Goodput gain	/	4.9%	7.3%	9.8%
Jain Index per stream	0.9993	0.9979	0.9960	0.9973
Jain Index per transfer	0.9993	0.9994	0.9998	0.9998

TAB. 5.III – Results of the evaluation of the multiple parallel streams experiment

number of independent sources (eleven) transmitting continuously for 600 s for each run of the experiment. The only changing parameter is the number of parallel streams that each source will use in a given run. BIC-TCP is used as the transport protocol.

5.3.3.2 RESULTS

The results for this experiment are summarized in Table 5.III. As expected, large number of parallel streams manage to obtain more bandwidth than a single stream. This confirms the convergence to an asymptotic value of throughput deficiency as in [Altman et al., 2006]. Here, the deficiency is about 700 Mbps (i.e. 7 %). As each pair of nodes is using the same number of parallel streams, no fairness problem appears, since both the inter-flows Jain index and the inter-nodes Jain index are above 0.99.

Figure 5.10 is a comparison we tried to make between our measures and Altman's model. Assuming that we can apply the formula introduced in Altman's work to BIC-TCP :

$$\overline{x}(N) = C(1 - \frac{1}{1 + \frac{1+\beta}{1-\beta}N})$$

– β is the multiplicative decrease factor of an AIMD-like protocol and C the bottleneck capacity – we can see that both converge to an asymptotic value. The difference between the two graphs is about 8 %. It is likely due to the fact that in our experiments, the goodput (and not the throughput) was considered.

Another experiment was also performed to study the impact of the competition of ten nodes emitting ten parallel streams on a node emitting an inferior number of streams. In the case when a single flow is used, the singled-out node is not able to use more than one fifth of its own 1 Gbps link. The other streams put too much pressure on the bottleneck and prevent the lone stream from taking a bigger share of the bandwidth. Here, there is clearly a per node unfairness toward the node emitting only one flow. As we increase the number of streams on the singled-out node, the problem of fairness disappears. These results are summarized in Table 5.IV.

This shows that, even though using parallel streams helps maximizing the usage of a link, it might have a huge impact if the link is shared by other regular flows. A fairer and more efficient solution would require the use of an additional tool such as scheduling and/or access control to avoid mixing greedy parallel streams and regular flows.

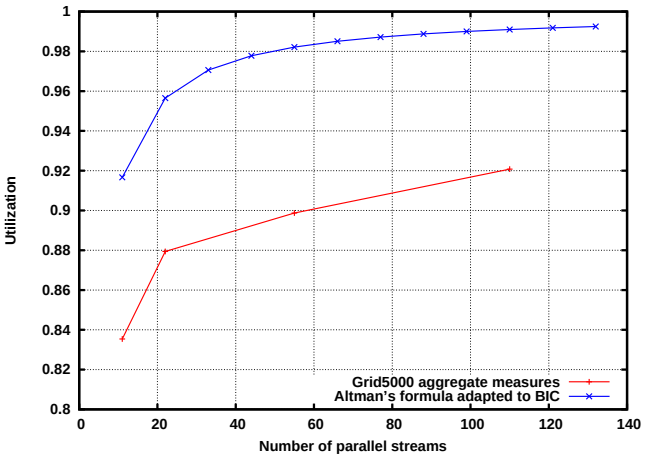


FIG. 5.10 – Comparison between the analytical model and experimental measurements for the multiple parallel streams experiment

Number of streams on the singled-out node **Total number of streams**	**1** 101	**2** 102	**5** 105	**10** 110
Mean total goodput (Mbps)	8953.12	8970.11	8973.83	8984.18
Stream mean goodput (Mbps)	88.6	87.9	85.46	81.67
Jain Index per transfer	0.806764	0.926327	0.962753	0.999949

TAB. 5.IV – Impact of the competition of 10 nodes, each emitting 10 streams on a node using an inferior number of streams

5.3.3.3 Synthesis

This scenario shows the positive impact of using multiple parallel streams on the average flow goodput as it was possible to increase the maximum aggregated goodput by 10 %. A comparison has been made with the analytical model proposed by Altman [Altman et al., 2006] and shows a good match with the results from the experimentation.

Finally, a study of the fairness has shown that multiple parallel streams should be used with caution in shared networks as the impact on the other traffic (if it is not using parallel streams too) is far from negligible (up to 90 % with a per-node fair would have). It is explained by the fact that TCP (and most of TCP variants in a DropTail network) share the bandwidth at the bottleneck on a per-flow basis, rather than on a per-node basis. If users abuse this property, then it can lead to severe degradation of performance for everyone else, just as seen in the Congestion Collapse experiment of Chapter 4, Section 4.2.

Conclusion

This chapter has presented our proposal for an end-user-oriented test-suite to define the metrics, meaningful and easy-to-interpret for the end-user, and workload models, defined through the use of representative applications : Web, Peer-to-Peer, MPI and Bulk data transfer applications. This test-suite will allow end-users to easily compare different transport protocols variants, so as to help them make a choice of the one (or the mix of transport protocols) that will be more suited for their need.

The case of Bulk data transfer is further illustrated through two examples that aimed at showing the impact of the congestion on the predictability of transfers and the consequences of using multiple streams to perform transfers. Such examples allow end-users to discover the properties of transport solutions and to avoid making mistakes by, for instance, choosing the very-fast unpredictable transport protocol for their tighly time-constrained application.

This chapter only presents a small set of the all experimental measurements (that amounts to several hundreds of hours, mostly on the Grid'5000 testbed) I have made during this PhD due to space limits, further details and other results of interest can be found in the published articles and research reports whose list is located at the end of this dissertation in the bibliography of my publications.

This chapter has also highlighted the need for calibration and tuning of the networking environment the end-user plan to use. A bad configuration can only lead to bad performance. End-users usually don't have the necessary knowledge to tune their configuration themselves and require the help of more advanced users or of automated tools to solve their configuration problems. This is the purpose of the next chapter.

— Six —

Exploration of End-to-End Path Characteristics

Introduction

In the last few years, the components related to the network (link capacity, network interfaces' bandwidth capacity, *etc.*) increased by at least three orders of magnitude. But in the meantime, the other components (like the PCI bus or the hard drives) were not all able to scale with similar speeds. It has led to the creation of potential bottlenecks in the interconnection of these hardware components. The same increase of speed also created a greater strain on the software parts of the operating system. At Gbps wire speed, the OS is required to handle packets every few nanoseconds to keep up with the rate. And finally, we have seen in Chapter 1 that the AIMD (Additive Increase, Multiplicative Decrease) algorithm that is used in TCP, is widely known to have deficiencies in very high-speed environments, due to a slow feedback loop [Floyd, 2003] when the RTT is important. The performance problem is all the more exacerbated when the user performs large transfers as it assumes the whole system is able to process a large stream of packets for a long period of time.

In the end, such an user will only experience one symptom : a throughput lower than the advertised capacity of its network interface and with little more than nothing to find out what the real problem was. It requires a lot of experience to identify where the actual bottleneck is and even more experience to finely tune the element that is causing the throughput loss. This problem is widely known as "the wizard gap"[Mathis et al., 2003]. The calibration test presented in Chapter 5, Section 5.2.3.1 has highlighted the dramatic consequences of this problem : a single parameter like the TCP buffer size was held accountable for a 90 % performance deficiency.

As not every end-user is an expert, there is a need to have an adequate estimation of the end-to-end performance of a networking path before launching a network-costly application so as to avoid missing deadlines and to prevent unnecessary network resources consumption and reservation. Having a tool able to detect the existing bottlenecks at the hardware, software and network levels may help to get such an estimation. The objective of this chapter is to present the problem of end-to-end path characterisation and detection of the potential bottlenecks in the end-to-end communication chain so as to provide useful insights of what the corrections could be.

The other sections are organised as follows : Section 6.1 presents similar approaches to identify and solve potential bottlenecks. Section 6.2 describes our general approach for the detection, the analysis and the resolution of hardware, software and network bottlenecks existing between two end-hosts. Sections 6.3, 6.4, and 6.5 further detail each of these steps.

6.1 State of the art

6.1.1 Networking Diagnostic Tools

Some propositions of fully integrated tools have been made to allow end-users to solve their networking problems. They mainly focus on solving the "last-mile" problem, as it is assumed that most problems related to end-users occur in the very last portion of the network.

6.1.1.1 Mping

Mping [Mathis, 1994] is a tool from the Pittsburgh Supercomputing Centre by Matthew Mathis to directly measure the max queue size, the loss rate and the throughput of a given path. It is based on a mechanism that tries to maintain a given number of packets (ICMP or UDP) in flight between two hosts. By slowly increasing the number of packets in flight, it is then possible to detect the queue size by locating where a queue starts to build up and when packets starts to be discarded.

Unfortunately the technique used is no longer working with modern-day networking stacks. ICMP packets if sent in large quantities are very likely to get discarded in the routers on the path. Moreover to reduce the effectiveness of port scans, Linux kernel limits the rate of the ICMP error messages sent to 80 per 4 seconds with a penalty of 1/4 second if exceeded. When sending UDP packets, mping behaves like a UDP port scanner (as it relies on receiving ICMP UNREACH error packets to estimate the number of packets received at the hosts), so the results obtained drastically underestimate the true capacity of the network.

6.1.1.2 PathDiag

PathDiag is a project [Mathis et al., 2008][1] from the Pittsburgh Supercomputing Centre to diagnose possible performance anomalies in the last mile of the network. The idea behind this tool is the use of a parametric model scaling [Mathis et al., 1997] to estimate the performance of a long-distance link on a short portion of the path. It uses a version of the classical formula expressing TCP throughput as a function of the RTT, loss rate and segment size : $R = \frac{MSS}{RTT}\frac{c}{p^d}$ where $c = 1.22$ and $d = 0.5$ for TCP Reno. The reason being that a small problem that could pass unnoticed in a local test could prove to be a major one once, incorporated in a long-distance link.

The full architecture used is described in a white paper [Mathis, 2003a]. The use of software latency emulation is discussed but it is rejected as being difficult to enforce on end-hosts and fairly inaccurate.

The software is implemented as a client-server application written in Python with the requirement that the server must be running on a GNU/Linux box with a kernel using the Web100 patch. The server can be contacted via a command line tool or a Java applet.

The client needs to provide the estimated RTT to the real target and the throughput that the user wants to achieve using the link. Once the connection has been established, the server will start a series of transfers with the client to gather data about the state of the link. They are used to find the capacity of the link, the maximum queue size on the short path and the loss statistics that will be used to deduce the estimated performance of the link. As soon as this phase is completed, a report is generated to help users identify problems linked to TCP configuration, data rate, loss rate, or Ethernet duplex mismatch. If anomalies are detected, pointers to a Frequently Asked Questions page is provided to indicate the possible causes of the problem and ways to solve it.

There are some limitations, as the target server can not be more than a few milliseconds away from the client, *i.e.*, in the path before the border edge router, in order to have an accurate estimation of the link's performance. Therefore it is extremely difficult to use this

[1] See the project page : http://www.psc.edu/networking/projects/pathdiag/

tool to test end-to-end problems even with networks with only a limited national-RTT range (about 20 ms for high-speed networks).

The main problem with this method is that the server needs to be adequately placed along the network path as only a small portion of the link is tested. If the problems are not due to this part of the link, another server is required in another location. It may be difficult to install such software inside an ISP domain. As the formula used to estimate the performance is clearly dependant on the TCP congestion control method, it needs to be adjusted every time the performance of a particular TCP variant needs to be assessed. It mostly helps solving the problems on the downloading side.

6.1.1.3 Network Diagnostic Tool

Network Diagnostic Tool [NDT, 2007] (NDT) is a project[2] from the Internet2 End-To-End Performance Initiative group (E2EPI). It is designed to allow users to run diagnostic tests right from their desktop computers and help them to solve some common problems.

This software was developed around a client/server architecture. The server relies on the Web100 patch [Mathis et al., 2003] of a GNU/Linux kernel to perform measurements of TCP statistics. The client is a web-based java applet. The packet pair technique is used to determine the size of the bottleneck link.

The test is performed in several phases. It will establish if middle boxes or a firewall disturb the transfer before performing a bidirectional bandwidth test. It will use the data acquired (information negotiated during the opening of the TCP transfer, throughput, loss rate), to establish a "network signature" and deduce the empirical maximum throughput that is achievable. As the bandwidth test is set to last for a fixed length of 10 seconds, it might not be enough to experience a drop over long latencies or it might be representative of the size of transfer a user would want to perform. Also it does not provide configuration hints other that duplex mismatch. It is assumed that the user would know what to do with the results of the tests or forward them to people who do.

6.1.1.4 Gridftp-netlogger

Gridftp-netlogger[3] is a module that can be used simultaneously with the gridftp application to identify the bottlenecks during a transfer. It is done through the instrumentation of the read() and write() system calls to measure the average throughput of the moving data from the disk to the memory and moving data from the memory to the network. Then it performs a comparison of the throughputs collected to indicate which of the disk or of the network is the most likely culprit for the bottleneck. It is based on the Netlogger library [Gunter et al., 2000] developed by the LBNL.

6.1.2 Autotuning

Several proposals were proposed to automatically improve the performance of TCP over high-speed networks in the last few years. These proposals mainly focus on the TCP buffer size which is responsible for a large number of the network performance problems. It can amount to up to 80 % drop in performance as seen in our research report on the performance evaluation of the Grid'5000 links [Guillier et al., 2006]. Other parameters (like the number of packets that can be stored temporarily in a queue before being treated) might also need to be updated to achieve better performance. But as they usually affect all connections, the modification is left to the user/administrator.

6.1.2.1 Net100 WAD autotuning daemon

WAD [Dunigan et al., 2002], standing for Work Around Daemon, is a software that is able to monitor through a Web100 patched kernel the newly opened TCP connections, tune a

[2]See the project page : http://e2epi.internet2.edu/ndt/
[3]see project website : http://www.cedps.net/index.php/Gridftp-netlogger

few parameters like the buffer's size and the algorithm used during the slow start phase, or even override the AIMD algorithm used. It can also provide a virtual MSS to improve the performance of the application. It can use additional software like *pipechar* to evaluate the capacity of the bottleneck link to provide a more suitable value for the buffer size.

The main problem of this mechanism is that it requires a kernel patched with the Web100 software, which is not available by default in standard GNU/Linux kernels and thus requires a little bit of work from the users.

6.1.2.2 SOBAS

In [Prasad et al., 2004], Dovrolis presents a scheme to allow the auto-sizing of socket buffers for high-performance data transfers, called SOBAS. The adjustment of the ideal TCP buffer's size is entirely done on an application level. The goal is to limit the size of the send window so as to avoid buffer overflows and self-induced losses if the link is not congested. The authors were able to obtain up to 80 % improvement over cases where the maximum buffer size available was used.

It does not require a previous knowledge of the network path as it acquires the data it needs to operate while performing the transfer (RTT, available bandwidth). The main drawback is that it would be necessary to recompile applications with the SOBAS library in order to benefit from it.

6.1.2.3 Linux TCP auto tuning

This mechanism has been present in the GNU/Linux stack since version 2.4 (released around 2001). It is designed to adjust the sending window/receive window to the actual capacity of the link (in conjunction with the TCP window's scaling option).

As the memory is potentially shared by several applications, the allocated value may change depending on the simultaneous needs of the other applications.

It is enabled by default but it is disabled if the user/application uses the setsockopt() operation to set a fixed value to the TCP buffer's size (assuming that the user/application chooses a better suited value). If the user has enough memory available, then only the receiver window size (and buffer) may be a software limitation.

There is also a "save metrics" feature that can keep, for a certain amount of time, information about a previously used path to immediately set the TCP parameters. But it can be an inconvenience as congestion on previous flows could have dramatically reduced some parameters (like the slow start threshold), and thus impact usage of the following connections.

Similar mechanisms are used by default in Microsoft Windows, starting with Vista. The mechanisms used by Mathis' Automatic buffer tuning [Semke et al., 1998] and Dynamic right-sizing [Fisk & Feng, 2001]) are based on the same ideas.

6.1.3 Synthesis

Several projects exist to solve the problem of the adequate configuration of end-hosts to achieve the best performance possible for users. But none of them is taking the whole chain, from the hardware to the network, into consideration to find and solve the existing bottlenecks.

6.2 Proposal

This section describes PathNIF [Guillier & Vicat-Blanc Primet, 2009d] [Guillier & Vicat-Blanc Primet, 2009c], our method and tool to detect, analyse, and provide solving hints for hardware, software, and network bottlenecks. Figure 6.1 shows a big picture of the entities that are involved in the process of a transfer. The black dots represents the interconnection points of our system.

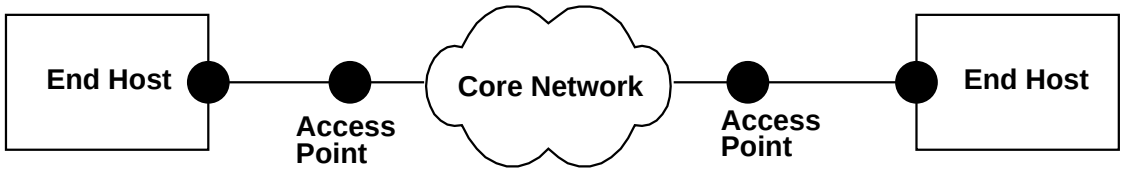


FIG. 6.1 – End host chain

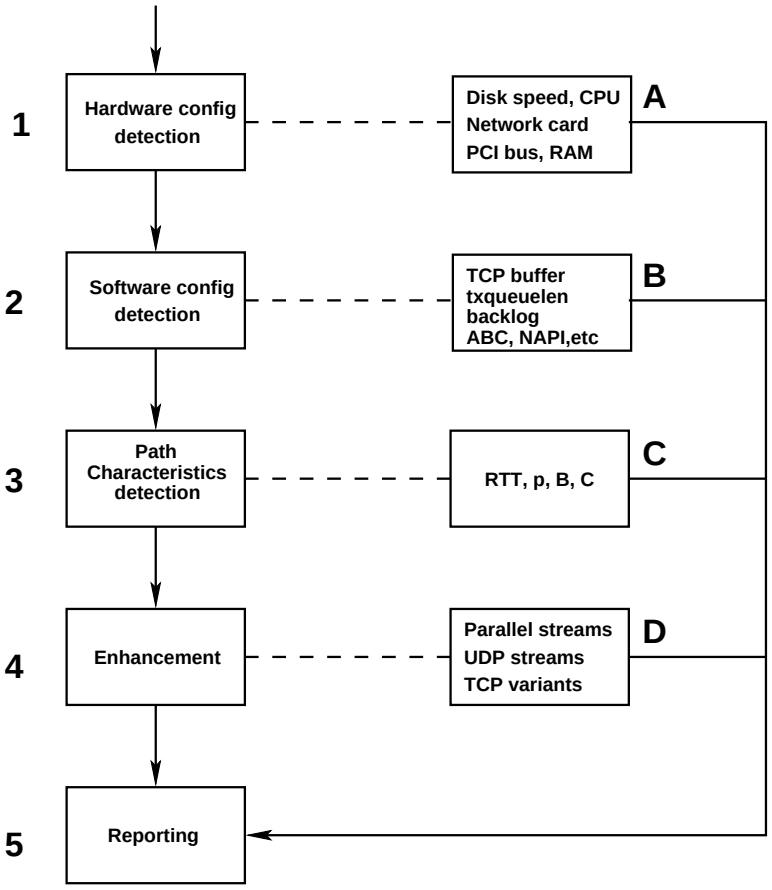


FIG. 6.2 – Workflow of PathNIF to detect hardware, software and network bottlenecks

Name	Description	Type
src	Source of the transfer	mandatory
dst	Destination of the transfer	mandatory
V	Order of magnitude of the volume to transfer	mandatory
RTT	RTT	optional
T	Target speed	optional

TAB. 6.I – Input Parameters

6.2.1 WORKFLOW

Figure 6.2 presents the general workflow that is used in our bottleneck detection system. The different stages are described as follows :

1. **[Hardware Config detection]** This stage consists in checking some important part of the local hardware configuration to assess the theoretical limits of the hardware used, specifically the known bottlenecks – CPU, PCI bus, Network Interface Card, and RAM.
2. **[Software Config detection]** In this stage, the tool will retrieve some of the vital parameters of the local software configuration (such as the buffer's size or the txqueuelen) and it will be used to check if the current software configuration is consistent with the network configuration. If not, hints about a more optimal configuration will be provided.
3. **[Network Config detection]** This stage consists in evaluating a certain number of network infrastructure information parameters, such as the RTT, the loss rate or the bottleneck capacity that likely causes performance deficiency. The information collected here is used in later stages to determine the best networking configuration possible. Since only estimates can be obtained, users could also provide values as it is likely that they have the correct information already. If the estimates obtained (or the values provided) are not correct, then it is possible that the following stages yield incorrect advice or results.
4. **[Enhancements]** In this stage, the tool will evaluate several known enhancements to improve the performance achieved. Depending on what is available on the local resource, parallel streams, UDP streams or TCP variants may be used. It will be up to users to modify/configure their applications to take advantage of the best solution.
5. **[End]** The final stage of the execution. No action is performed on this stage except for displaying the report of the complete execution. It is possible to reach this stage from several points during the execution if it is detected from the local conditions that it is impossible to improve anything. The rest of the execution is then aborted.

It is important to note that it is possible to stay at a given stage for a long time as it may be necessary to test several solutions (especially during stage 4) before concluding. It is also easy to extend our tool by adding new variables that we may want to monitor at a given stage.

The next sections will describe precisely what is done at each stage.

6.2.2 DETAILED PRESENTATION OF EACH STAGE

In this section, it is assumed that a GNU/Linux kernel is used. It might be possible to port the tool for another OS, especially BSD and Unix-like, which have similar utilities to configure the TCP stack.

6.2.2.1 INPUTS

Table 6.I summarises the required input parameters of the program.

src corresponds to the hostname from which we want to issue the transfer. If this parameter is missing, it can be assumed that the source of the transfer is the default network interface of

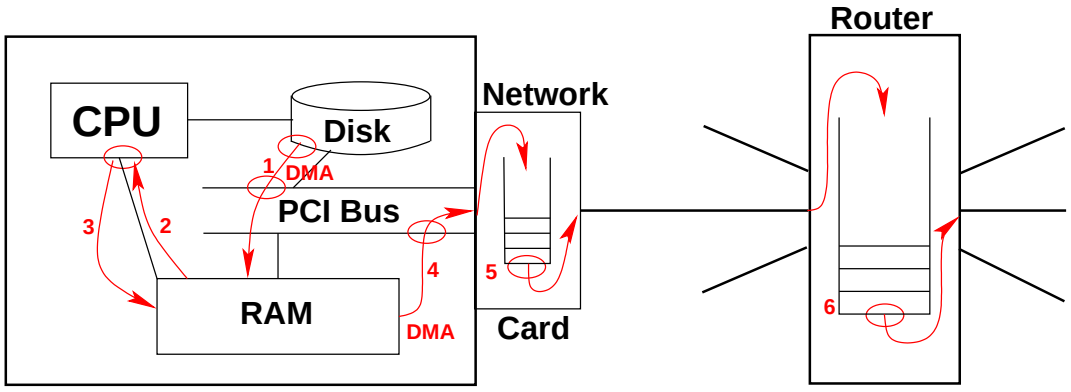


FIG. 6.3 – The existing hardware bottlenecks encountered in an end-host hardware

1) Data copied from the disk to the RAM through the PCI bus
2) Data moved to the CPU (splitting into packets)
3) Packet copied from the RAM to the transmit queue
4) Packet copied to the NIC via the PCI bus
5) Packet transfered on the network after waiting in a queue in the NIC
6) Packet handled in a router after waiting in a queue

the computer. *dst* corresponds to the name of the destination of the transfer. It is necessary to have access to both sides of the transfer as both sides can have bottlenecks.

V, the order of magnitude is a very important parameter because it will define the duration of the several tests that will be executed during the execution of stages 3 and 4. Typically, if the duration of the transfer's duration is of the same order of magnitude as the Slow Start phase, then the amount of optimisation that can be done is limited.

Some parameters like RTT and T are optional and can be provided by an advanced user who already has some ideas about the characteristics of the networking path that will be used for the transfer and just wants the generation of the appropriate configuration file.

6.3 Hardware Config Detection

6.3.1 Bottlenecks

Figure 6.3 presents the possible bottlenecks at the different steps necessary to transmit disk-to-disk data over a network.

Symmetrical operations are performed on the receiver side to bring the packets from the network to the hard drive. Potentially there are several places where there can be bottlenecks in the hardware : disk input/output, PCI bus, CPU, RAM, and buffer space in the routers.

The hardware config detection uses a collection of existing tools easily available on GNU/Linux kernels to determine the maximum bandwidth that is achievable by a given hardware configuration. If the user does not have root access, then the amount of information that we will be able to gather is limited. However, the result of the execution of this part of the program (for instance by somebody who has root access on the target computer) can be saved into a text format. This data can be used by a non-root user later on the perform the rest of the execution.

6.3.2 Formula

This section presents our formula to evaluate the maximum bandwidth that can be achieved with a given hardware configuration. The basic idea is that r_{max}, the maximal achievable rate,

is the minimum of the achievable bandwidth via each of the possible bottlenecks that have been identified in the previous section.

$$r_{maxhardware} = min(B_{disk}, B_{RAM}, B_{CPU}, B_{NIC}, B_{PCI}) \tag{6.1}$$

If one part of the formula can not be retrieved/computed, the user is warned and it is discarded from the final computation. Some parameters like the current load of the CPU or of the PCI bus are not taken into consideration in the global formula. As they are highly dynamic in nature, it is very difficult to integrate them except by having some knowledge of the average load on these elements. But only having an average is clearly not enough, because if there is not a lot of margin between the required bandwidth and what is possible with the available hardware, any perturbation caused by any other processe can have a huge impact on the performance of the transfer process. In the experiments presented in [Chase et al., 2001], it is shown to cause a decrease in performance of one half.

6.3.2.1 Disk Part

B_{disk} is only considered in the case of a disk-to-disk transfer. Here we are taking into consideration two possible bottlenecks : the speed of the hard-disks themselves and the speed of the controller onto which they are attached.

$$B_{disk} = \sum_{controller} min(speed(controller), \sum_{HD \in controller} speed(HD)) \tag{6.2}$$

The speed of the disks is detected using the benchmarking option (buffered read) of the *hdparm* program. Another benchmarking tool could easily be used to provide a more accurate value for the disk speeds. It is important to note that we are currently considering that the write speed and the read speed of the disk are the same. The controller speed is identified through the mode (PIO, DMA, UDMA, SATA) in which the disk is operating.

6.3.2.2 RAM Part

$$B_{RAM} = Data_{width} * Frequency / N_{access} \tag{6.3}$$

N_{access} is the number of access to the RAM needed to perform the transfer of one chunk of data. Here it is equal to 4. The other values are retrieved via the DMI tables.

6.3.2.3 CPU Part

Several parameters needs to be taken into consideration to establish the "bandwidth" that we are able to get out of the CPU. It is also possible that the CPU is no longer such a major bottleneck as in the last few years it has increased much more rapidly than the other components of a computer such as memory or the interconnect bus.

First, the classical rule of thumb "you need 1 HZ to read or write one bit of a TCP/IP" [Foong et al., 2003] is applied so the raw bandwidth is defined as the frequency of the CPU multiplied by the number of CPUs and cores available. This is not completely true as it is shown in the same article [Foong et al., 2003] that it does not scale too well with high frequency as other bottlenecks start to appear, but these other bottlenecks are taken care of in our model. Here it is assumed that we have a uniform distribution of the tasks performed over the physical resource. It is not very realistic because most OSes are using scheduling algorithms that strongly take into account the affinity of one task on a given CPU (*i.e.*, avoid if possible moving data that is already in the cache of one CPU rather than moving it back and forth between two CPUs with every change of context for the task) but it is representative of what we could get out of the system (the selection of the appropriate policy is left to the user, even though we provide a few hints in Section 6.5.1). The CPUs and their cores are placed on an equal footing because usually a CPU with many cores is operating at a lower frequency than a

single-cored CPU. It might also be necessary to take into account the fact that in multi-cored CPUs, the L1 and L2 cache can be shared. The limitations of the Northbridge are also not taken into consideration.

The second step is to compute several coefficients that will reflect how efficiently the CPU is used, they are the following :

NAPI The new Networking API used in the GNU/Linux kernel, it is designed to prevent livelocks (state in which all the CPU is consumed by the kernel to handle interrupts and context changes rather than doing anything useful) by allowing the card's drivers to poll at regular intervals the NICs rather than wait for interrupts. According to [Murta & Jonack, 2006], the CPU is never overloaded with this policy and it reaches at most 70 % of CPU usage. So this policy allows to save 30 % of the CPU bandwidth. Unfortunately a control test is missing in their experiments(*i.e.*, case where no special policy is applied) to model this parameter more precisely. It is very likely that there is another bottleneck in their system (RAM or PCIbus) as they are not able to reach gigabit-speed and the interrupt parameters were not tuned.

NIC Interrupt coalescing It is nearly the same thing as above, as it is designed to reduce the number of interrupts that are send by the NIC to the localhost. It is performed by waiting for several packets to arrive (or a timer to expire) before sending the interrupt. This way, the kernel is interrupted for a large number of packets.

NIC computing offload It corresponds to a variety of techniques to put a part of the TCP stack into the NIC to remove some of the computation that is made by the CPU. It is generally used to do the checksums and the fragmentation of packets. In [Tomonori & Masanori, 2003], the authors evaluate the impact of zero-copy, checksum offloading and TCP segmentation offloading on CPU utilisation for the iSCSI protocol (an equivalent of the SCSI protocol to be used over LANs). Combined together, these policies result in a 5 % to 10 % gain in CPU utilisation, depending on whether the operations performed are read or write.

NIC MTU size One of the typical methods to achieve a better performance when the CPU is overloaded is to use larger MTUs [Mathis, 2003b]. It is based on the MTU size that is advertised by the NIC to the kernel (usually negotiated with the switch/router to which it is connected). We use a simple logarithmic modelisation based on the experiments presented in [Chase et al., 2001] to show the impact of the MTU on the CPU. The idea is that we are able to get a 20 % improvement when using a 8 KB frame over the 1.5 KB frame case for the same CPU. The logarithm model was chosen because of the shape of the results. It is corroborated in [Foong et al., 2003] where the authors measured that smaller MTUs were far more CPU-costly than larger ones. In another article [Wadge, 2001], larger MTUs allowed a 38 % increase in performance.

$$MTUcoeff = 1 - \frac{0.2}{ln(8000) - ln(1500)} * (ln(8000) - ln(mtu)) \tag{6.4}$$

DMA This technique is used to bypass the CPU when there is a need to perform read and write operations to a hard drive. Currently we assume that DMA improves the performance by 25 % based on the analysis of Clark in [Clark et al., 1989], about 50 % of the time is spent doing copies.

$$B_{CPU} = Frequency * N_{CPU} * N_{core} * MTUcoeff * Othercoeff \tag{6.5}$$

6.3.2.4 PCI Part

Here there are two cases depending on the kind of PCI bus that is available : Standard PCI (PCI and PCI-X), or PCI-Express (PCI-e). The main difference between the two categories

is that the PCI-e bus is designed to be full-duplex. Also a PCI-e port has a given number of independent lanes of fixed speed assigned rather than having to share a single line with the devices installed on it. We assume that there is no contention with other devices.

$$B_{PCI} = Data_{width} * Frequency * PCIcoeff | N_{lanes} * speed(lane) * PCIEcoeff \quad (6.6)$$

As we consider a continuous flow rather than just one packet, it means that at one time or another, we will have simultaneous read/writes. As Standard PCI buses are not full-duplex, PCIcoeff is equal to $\frac{1}{2}$. PCI-Express is full-duplex but 2 bits out of 10 are used for signalling purpose, PCIEcoeff is equal to $\frac{8}{10}$.

6.3.2.5 NIC Part

For this part of the formula, only a static aspect of the card is considered, that is to say the speed that has been negotiated with the next hop, *e.g.*, a switch. The MTUcoeff is a coefficient to take into consideration the overhead of the transport, network and data link protocol over the payload transmitted. It is assumed that TCP is used as the transport protocol, IP as the network protocol and Ethernet as the data link protocol.

$$B_{NIC} = speed(NIC) * MTUcoeff \quad (6.7)$$

$$MTUcoeff = \frac{MTU - 52}{MTU + 14 + 4 + 12 + 8} \quad (6.8)$$

6.4 Software Config Detection

6.4.1 Bottlenecks

In this section, we analyze the software part of the configuration of a given computer. It is done to find out if any part of the configuration of the kernel is currently a bottleneck of the system or to determine the maximum throughput that is achievable given the current configuration. As most of these factors are configurable, it is possible to provide hints to the user to indicate which configuration could help improve the potential performance of the whole chain. Three different kinds of parameters are identified :

- NIC driver configuration
- Kernel configuration
- TCP configuration

These categories separate the different elements that are involved in the emission/reception of packets, separating the transport protocol from the rest (*i.e.*, we could change the transport protocol independently from the rest). It is also necessary to separate the elements that require a recompilation (kernel or driver module) from the elements that just need to be changed through a variable (*e.g.*, a key in the procfs to configure the GNU/Linux kernel).

It is important to note that no specific user application (and the way it performs its network communications) is taken into consideration. If the application is not able to continuously fill the transmit buffer or does not empty the receiver buffer on a regular basis, the actual performance will be lower.

The dashed boxes in the figure 6.4 present the different software elements that are involved in moving data from/to an application to/from the network. Each arrow indicates a treatment. It can be a full copy (using DMA or not) or just a copy of the description headers of the data (inside the kernel space). The cost of the copy is neglected in this model as it is taken into consideration in the part devoted to the hardware 6.3. This model was first introduced in [Rio et al., 2004] where the authors describe the implementation of the TCP stack in the GNU/Linux kernel. It was later used in [Wu et al., 2006] to estimate the average response time of the system, but not the bottleneck value.

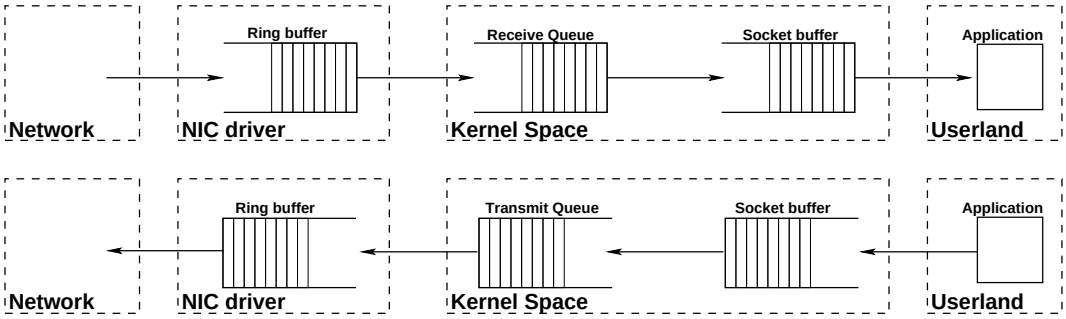


FIG. 6.4 – Simple model for transmitting/receiving packets in a GNU/Linux stack

6.4.2 FORMULA

This section presents the formula that is used to compute the theorical limits of the software bandwidth and establishes a software configuration that is likely to provide a better performance for the user (usually by fixing the target bandwidth and deducing the parameter value)

$$r_{maxsoftware} = min(B_{NIC}, B_{queue}, B_{application}) \tag{6.9}$$

In the following sections, MTU corresponds to a normalisation coefficient to take into account the fact that the retrieved values from the kernel are provided in packets rather than in bytes. Typically, it is equal to $1500 * \frac{8}{1000000}$ to convert in Mbps if the MTU used is the Ethernet frame size.

6.4.2.1 NIC BANDWIDTH

We define the software bandwidth limitation of the NIC as the amount of data that can transit through the ring buffer of the card. It is limited by the number of interrupts that can be handled over a period of time.

$$B_{NIC} = Buffer_{ring} * interrupt_{rate} * MTU \tag{6.10}$$

The existence of a *queuedisc* policy to limit the amount of available bandwidth is also taken into consideration. It corresponds to a queuing policy like Token Bucket that is used to shape the outgoing traffic. Nevertheless this feature is rather limited due to the ambiguous grammar used by the *tc* program and the difficulty to perform a match between a given destination and a tc filter.

6.4.2.2 QUEUE BANDWIDTH

The queue bandwidth is the bandwidth achieveable when the kernel is copying packets from the backlog queue to the TCP buffer (or respectively from the TCP buffer to the transmission queue if we are on the sender's side). HZ is the minimum granularity of the scheduler of the Linux kernel. It allows to have a rough estimate of how many tasks (as copying data from the kernel queue to the application buffer) can be executed in a second. It is only an upper bound as the time required to execute these various functions may vary.

$$B_{queue} = backlog * HZ * MTU || txqueuelen * HZ * MTU \tag{6.11}$$

6.4.2.3 APPLICATION BANDWIDTH

This part of the formula corresponds to the maximum dataspeed at which the application will be able to read or write inside the TCP buffer during the transfer. It is determined by the

Name	Description	Retrieval method
MTU	MTU size	ifconfig, ethtool
RTT		ping
C	Bottleneck capacity	pathload
A	Available bandwith	pathload
p	Loss rate	ping, web100, netstat
B	Max buffering space	
T	Completion time	iperf

TAB. 6.II – Retrieved Network Configuration Parameters

amount of data that pushed through the network. This value is determined by the bandwidth-delay product.

$$B_{application} = Buffer_{TCP} * RTT \tag{6.12}$$

In the implementation, we are considering the case when the Linux autotuning mechanism is used and when the user is explicitly setting a value for the TCP buffer.

6.5 Network Characteristics Detection

It is then assumed that PathNIF is the only application that is transfering through the local networking interface during this stage. If otherwise, it would be difficult to estimate the actual capacity of the network. It is also assumed that the workload on the networking path is representative. Transient bursts from other users can make PathNIF report worse results than the actual capacity of the bottleneck link.

In this stage, we run a simple transfer of V bytes using iperf. It is used as a baseline for the comparison of other transport protocols in stage 4. Additional software, such as *pathload*, is used to estimate the other parameters. Depending on the software available on the *src* or the *dst* node, it might be possible to use different methods to retrieve the same parameter. Typically if the web100 patch is not available, a less precise estimation of the loss rate can be retrieved through netstat during the test. This estimation is not as accurate as it is impossible to break the counters provided into something as fine-grained as a flow.

6.5.1 Enhancements

This section describes the various enhancements that can be used to improve the performance of the transfer. The basic idea is to reproduce the simple transfer done at stage 3 but using a different transport protocol.

6.5.1.1 Multiple parallel streams

In this test, we evaluate the number of parallel streams that would yield the best performance. The test consists in sending the same volume V but split over several connections for a variable number N of streams.

6.5.1.2 TCP variants

After checking the available TCP variants on the system, their response function (*e.g.*, a closed-formed model of the mean throughput achieved as a function of several network characteristics like the loss rate or the RTT) is used to determine the one which is likely to perform better in the conditions detected during the network stage. It is then possible to change the TCP variant used by the system and perform the test to assert if it is indeed the

Path	Interface bandwidth (Mbps)	Mem to Mem (Mbps)	Disk to Disk (Mbps)
Desktop-Desktop	100	94.1	92.5
Capricorne-Parasol	1000	941	492
Scorpion-Scorpion	10000	6400	499

TAB. 6.III – Case study results

best option. If it is not possible to change the TCP variant, we only provide the necessary configuration line that can be used to change it.

6.6 CASE STUDY

In this section, we put PathNIF to the test by using our software on some sample computer systems. Table 6.III summarises the results obtained.

For the first test case, PathNIF has been used on two desktop computers. They are computers with Pentium 4 CPUs. The bottleneck is known to be a 100 Mbps switch. After the hardware phase, we noticed that all results were well above 100 Mbps. It is then confirmed at the network phase that we are able to achieve the maximum capacity available through the network.

The second test case corresponds to two nodes on different sites in the Grid'5000 experimental testbed [Bolze et al., 2006]. One is a HP ProLiant DL145G2, the other one is an IBM eServer 325. The two nodes are separated by a dedicated link with a 12 ms RTT that has a 1 Gbps bottleneck. After the hardware stage, it appears that every component is capable of handling a 1 Gbps Ethernet speed except for the hard drive disk that is limited to 450 Mbps in the IBM server. It is corroborated at the network stage as we achieve a 941 Mbps throughput in M2M transfer and only 492 Mbps in D2D transfer.

The last test case is using two HP Proliant 385G2 servers that are part of Grid'5000's Lyon site. The configuration of this cluster has been chosen so that it should be capable of doing 10 Gbps transfers. During the hardware analysis, it was found that indeed the system should be 10 Gbps-capable (M2M bottleneck limit of 9150 Mbps due to the CPU and D2D bottleneck limit of 470 Mbps) but during the network stage with a local computer (RTT 0.1 ms), it appeared that it was impossible to achieve more than 6400 Mbps with *iperf* (*pathload* indicated 7989 Mbps), even after a careful tuning thanks to the software bottleneck analysis. Extra analysis has shown that *iperf* completely saturates one of the two CPUs while the other sits idle. Two independent *iperf* processes were able to achieve a 9150 Mbps throughput which is what was predicted by the hardware model. It shows the limits of our assumptions about the repartition of tasks on CPUs/cores and the impact of the kernel scheduler on performance. The disk-to-disk network result is consistent with what was found at the hardware detection stage.

CONCLUSION

In this chapter, we have presented PathNIF, our tool designed to detect, analyze and solve hardware, software, and network bottlenecks existing between two end-hosts. Preliminary validation tests show that this kind of tool is useful for users to detect configuration problems.

As future works, we plan to improve the hardware formula to take into account the impact of the kernel scheduler on the CPU usage, depending on the memory architecture used (NUMA or SMP), we also want to improve the estimation of the maximum interrupt rate of the system and finally integrate other ways to evaluate the network's capacity by using less disruptive

means. We also plan to extend this work as a service usable, for example, by a bandwidth-reservation system to avoid overprovisionning for hosts that are not able to send over a given rate.

Conclusion and Perspectives

Conclusion

The transport layer is a vital part of the networking stack, and a transport protocol, TCP, has been developed since the beginning of the '80s to assume this role according to some design principles (the robustness principle, the end-to-end principle, *etc.*) and some implicit assumptions (minimal intelligence in the network, transparency of the network, fair-sharing, no pricing, *etc.*). Such assumptions may no longer be true in the future as the infrastructure of the Internet changes. Some authors of the NSF's project "Clean State Design for the Internet"[4] even advocate for the complete removal of the transport layer to reinvent the Internet's infrastructure. But some design principles such as the end-to-end principle are likely to remain, because they are the only practical way to deploy changes in a massive system such as the Internet.

This dissertation, entitled "Methodologies and Tools for the Evaluation of Transport Protocols in the Context of High-Speed Networks", has exposed the following points :

- Providing methodological material allowing the study of end-to-end performance is all the more important as it would still be valid despite all the possible changes of the Internet.
- TCP has been very successful (80 % to 95 % of the total Internet traffic), but as the capacities of the links increased, performance issues have been discovered. To solve these performance issues, a large variety of solutions have been proposed. This abundance of solutions highlighted another problem : how can the different types of actors (network provider, protocol designer, and end-user) concerned by the impact of the transport layer make sure that a given solution will have the desired effect ?
- After presenting the different questions that each group of actors tries to answer through the evaluation of transport protocols, we analyzed the different methods that have been introduced to study these transport solutions and have shown the adequacy (or inadequacy) of a particular method to the goal of given type of actors. If protocol designers and network providers do have quite a lot of methods suitable for their study of transport protocols, it is not so for end-users, for whom only real experimentation in uncontrolled environments is deemed useful.
- The evolution of Internet, both in terms of applications and of capacities, makes it necessary to perform real experiments at a large scale when the network-application designers, a subtype of end-users, want to evaluate the performance of their applications against a given transport solution. The study of the new characteristics (or the evolution of the parameters) of the Internet are very important, because transport solutions will be deeply affected by the conditions in which they are operating. For instance, it is shown

[4] See the project page http://cleanslate.stanford.edu/

that in a network with low multiplexing and low aggregation level, congestion collapses can occur if a large number of users decide to switch to unresponsive transport solutions (like UDP or multiple parallel streams, typical solutions used by P2P applications) to selfishly grab more bandwidth. To study these aspects, a fully-automated tool designed to execute large-scale experiments, like our proposal NXE, is extremely valuable. A more detailed comparison of simulation and real experimentation, through the examples of NS-2 and NXE, is made to highlight the fact that these tools are complementary and allow the study of different aspects of the transport layer.
- Other methodological contributions were made through our participation to the international effort of the TMRG workgroup of the IRTF to define a standardized test-suite for the evaluation of transport protocols that can be used both in simulation and in real experimentation.
- With a tool such as NXE, it is easy to perform experiments at a large scale in real networks. We then presented an end-user-oriented test-suite. It defines meaningful and easy-to-interpret metrics for the end-user and representative applications : Web, Peer-to-Peer, MPI, and Bulk-data transfer applications. An additional calibration application is proposed to make sure that the end-hosts are properly configured. The case of Bulk-data transfers is further detailed through experiments in the Grid'5000 testbed. These examples have shown the impact of congestion on the transfer time's predictability and the impact of the use of multiple parallel streams on fairness.
- Finally, we took into consideration the need for adequately configured systems to ensure that end-users will get a good performance out of their end-hosts (and avoid the frustrations of the "wizard-gap"). Our tool, PathNIF, is designed to finely analyze the existing bottlenecks in the hardware, the software, and the network in an end-to-end path and compute the maximal achievable bandwidth on the path. This analysis can then be used to help end-users to properly configure their end-hosts.

Perspectives

- NXE is a very useful tool to perform experiments at a large scale. It would be interesting to make it available to a larger community of users. For instance, integrating NXE as an automatic experiment execution service to testbeds such as Grid'5000 could be beneficial for a wide number of users. This way, they would be able to focus on the analysis of the results, instead of losing time writing and debugging management code for their experiments.
- The results presented in Chapter 6 are satisfying but it is still necessary to validate our approach on a larger number of systems, especially on 10 Gbps systems that are the current "performance frontier" in terms of high-speed transfers. Our tool, PathNIF, could also benefit from enhancements in the hardware model by taking into consideration finer aspects in the computation of some bottleneck bandwidths. For instance, the computer memory architecture used in current multi-processor or multi-core computers (NUMA or SMP) can have an impact on the global performance of the system but is not currently taken into consideration in our model. It will also be interesting to integrate additional enhancement proposals to our tool and develop a tool to automatically perform the configuration operations of a communication path on behalf of the end-user.
- Current bandwidth-reservation proposals usually rely on the end-users' input ("I would like to have a 1 Gbps link between A and B, please") to perform their allocation. As end-users might not have a clear idea of the performance achievable (if they are not using PathNIF or a similar tool), it is likely that the reserved path will be over-provisioned for the capacity of the users' end-hosts. It would then be interesting to develop a service around PathNIF to provide the information of the users' maximal achievable bandwidth

(after helping them improve it) to bandwidth-reservation systems. It could allow a more efficient use of the available allocatable capacities and make sure that end-users won't have to pay for bandwidth they can not exploit.

- Peer-to-Peer applications are increasingly popular applications for the purpose of transferring large data sets. Over the last few years, these applications have caused a lot of fears. If end-users were mostly concerned about the enforcement (or absence thereof) of copyright laws, protocol designers and network providers were worried about risks of congestion collapse and the huge amount of bandwidth these applications consume (a 2007 study by Ipoque [5] estimated the P2P traffic to be between 50 % and 90 % of the global Internet traffic). Still, there is no clear insight yet on the exact impact such applications will have on networks with low multiplexing and aggregation levels. For instance, it might be interesting to verify that the conclusions of the congestion collapse experiment we presented in this dissertation are true (or not) when using a P2P application such as Bittorrent to perform the transfers.
- Finally, the virtual infrastructures are starting to take a preponderant role through the cloud services that are currently deployed in the Internet. This virtualization can occur either in the physical nodes or in the network, and, by design, hides the true physical properties of each of these elements. It would be interesting to extend the work of this PhD to this domain : what are the constraints of these new infrastructures ? What is their impact on the performance of the transport protocol ? And more importantly, how can we make sure that the end-user will get the best performance out of it (for the price they are paying) ?

[5] See the company web page http://www.ipoque.com/

Publications

[Andrew et al., 2008] Andrew, L., Marcondes, C., Floyd, S., Dunn, L., Guillier, R., Gang, W., Eggert, L., Ha, S., & Rhee, I. (2008). Towards a Common TCP Evaluation Suite. In *PFLDnet 2008*.

[Guillier et al., 2007a] Guillier, R., Hablot, L., Kodama, Y., Kudoh, T., Okazaki, F., Takano, R., Vicat-Blanc Primet, P., & Soudan, S. (2007a). A study of large flow interactions in high-speed shared networks with Grid5000 and GtrcNET-10 instruments. In *PFLDnet 2007*.

[Guillier et al., 2006] Guillier, R., Hablot, L., Vicat-Blanc Primet, P., & Soudan, S. (2006). *Evaluation des liens 10 GbE de Grid'5000*. Research Report 6047, INRIA.

[Guillier et al., 2007b] Guillier, R., Soudan, S., & Vicat-Blanc Primet, P. (2007b). TCP variants and transfer time predictability in very high speed networks. In *Infocom 2007 High Speed Networks Workshop*.

[Guillier et al., 2007c] Guillier, R., Soudan, S., & Vicat-Blanc Primet, P. (2007c). *TCP Variants and Transfer Time Predictability in Very High Speed Networks*. Research Report 6256, INRIA. Also available as LIP Research Report RR2007-37.

[Guillier et al., 2009] Guillier, R., Soudan, S., & Vicat-Blanc Primet, P. (2009). *UDT and TCP without Congestion Control for Profile Pursuit*. Research Report 6874, INRIA. Also available as LIP Research Report RR2009-10.

[Guillier & Vicat-Blanc Primet, 2008a] Guillier, R. & Vicat-Blanc Primet, P. (2008a). Congestion Collapse in Grid5000. demo, "The Future of TCP : Train-wreck or Evolution ?", Stanford Congestion Collapse workshop.

[Guillier & Vicat-Blanc Primet, 2008b] Guillier, R. & Vicat-Blanc Primet, P. (2008b). Methodologies and Tools for Exploring Transport Protocols in the Context of High-Speed Networks. In *IEEE TCSC Doctoral Symposium*.

[Guillier & Vicat-Blanc Primet, 2009a] Guillier, R. & Vicat-Blanc Primet, P. (2009a). A User-Oriented Test Suite for Transport Protocols Comparison in DataGrid Context. In *ICOIN 2009*.

[Guillier & Vicat-Blanc Primet, 2009b] Guillier, R. & Vicat-Blanc Primet, P. (2009b). NXE. Software, APPcode : IDDN.FR.001.030005.000.S.P.2009.000.10800.

[Guillier & Vicat-Blanc Primet, 2009c] Guillier, R. & Vicat-Blanc Primet, P. (2009c). PATHNIF. #09/05285.

[Guillier & Vicat-Blanc Primet, 2009d] Guillier, R. & Vicat-Blanc Primet, P. (2009d). PathNIF. Software, APPcode : IDDN.FR.001.260002.000.S.P.2009.000.10800.

[Heusse et al., 2006] Heusse, M., Rousseau, F., Duda, A., & Guillier, R. (2006). PROCEDE D'ACCES POUR RESEAU LOCAL SANS FILS A ACCES ALEATOIRE. Patent #FR2875983 (A1).

[Heusse et al., 2005] Heusse, M., Rousseau, F., Guillier, R., & Duda, A. (2005). Idle Sense : An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs. In *ACM SIGCOMM 2005*.

[Loiseau et al., 2009a] Loiseau, P., Gonçalves, P., Guillier, R., Imbert, M., Kodama, Y., & Vicat-Blanc Primet, P. (2009a). Metroflux : A high performance system for analyzing flow at very fine-grain. In *TridentCom*.

[Loiseau et al., 2009b] Loiseau, P., Guillier, R., Goga, O., Imbert, M., Goncalves, P., & Vicat-Blanc Primet, P. (2009b). Automated Traffic Measurements and Analysis in Grid'5000. demo, SIGMETRICS/Performance 2009, Recipient of Best Student Demo Award.

[Soudan et al., 2007a] Soudan, S., Guillier, R., Hablot, L., Kodama, Y., Kudoh, T., Okazaki, F., Takano, R., & Vicat-Blanc Primet, P. (2007a). Investigation of Ethernet switches behavior in presence of contending flows at very high-speed. In *PFLDnet 2007*.

[Soudan et al., 2007b] Soudan, S., Guillier, R., & Vicat-Blanc Primet, P. (2007b). End-host based mechanisms for implementing Flow Scheduling in GridNetworks. In *GridNets 2007*.

Introduction Bibliography

[Berners-Lee, 1999] Berners-Lee, T. (1999). *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web*. HarperOne.

[Clark, 1988] Clark, D. (1988). The Design Philosophy of the DARPA Internet Protocols. In *SIGCOMM Computer Communication Review*.

[Floyd & Fall, 1999] Floyd, S. & Fall, K. (1999). Promoting the Use of End-to-End Congestion Control in the Internet. In *IEEE/ACM Transactions on Networking*.

[Floyd & Paxson, 2001] Floyd, S. & Paxson, V. (2001). Difficulties in Simulating the Internet. In *IEEE/ACM Transactions on Networking*.

[Froehlich & Kent, 1991] Froehlich, F. & Kent, A. (1991). ARPANET, the Defense Data Network, and Internet. In *Encyclopedia of Communications*, volume 1.

[Gillies & Cailliau, 2000] Gillies, J. & Cailliau, R. (2000). *How the Web Was Born - The Story of the World Wide Web*. Oxford University Press.

[Mathis et al., 2003] Mathis, M., Heffner, J., & Reddy, R. (2003). Web100 : extended TCP instrumentation for research, education and diagnosis. volume 33 (pp. 69–79).

[Saltzer et al., 1984] Saltzer, J., Reed, D., & Clark, D. (1984). End-To-End Arguments in System Design. In *ACM Transactions on Computer Systems*.

[Sherwood et al., 2005] Sherwood, R., Bhattacharjee, B., & Braud, R. (2005). Misbehaving TCP Receivers Can Cause Internet-Wide Congestion Collapse. In *ACM CCS*.

[Zakon, 2006] Zakon, R. (2006). Hobbes' Internet Timeline v8.2. Web page.

Transport Protocols Bibliography

[Allcock W., 2003] Allcock W. (2003). GridFTP : Protocol extension to FTP for the Grid. In *Grid Forum Document 20*.

[Allman, 1998] Allman, M. (1998). On the generation and use of tcp acknowledgments. *SIGCOMM Comput. Commun. Rev.*, 28(5), 4–21.

[Allman, 2003] Allman, M. (2003). RFC 3465 : TCP Congestion Control with Appropriate Byte Counting (ABC). RFC 3465.

[Allman & Paxson, 1999] Allman, M. & Paxson, V. (1999). On estimating end-to-end network path properties. In *SIGCOMM '99 : Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (pp. 263–274). : ACM.

[Allman et al., 1999] Allman, M., Paxson, V., & Stevens, W. (1999). RFC 2581 : TCP Congestion Control. RFC 2581.

[Altman et al., 2006] Altman, E., Barman, D., Tuffin, B., & Vojnovic, M. (2006). Parallel TCP Sockets : Simple Model, Throughput and Validation. In *Proceedings of the IEEE INFOCOM*.

[Anderson et al., 2006] Anderson, T., Collins, A., Krishnamurthy, A., & Zahorjan, J. (2006). PCP : Efficient Endpoint Congestion Control. In *NSDI*.

[Appenzeller et al., 2004] Appenzeller, G., Keslassy, I., & McKeown, N. (2004). Sizing router buffers. *SIGCOMM Comput. Commun. Rev.*, 34(4).

[Baiocchi et al., 2007] Baiocchi, A., Castellani, A., & Vacirca, F. (2007). YeAH-TCP : Yet Another Highspeed TCP. In *PFLDNet*.

[Balakrishnan et al., 1999] Balakrishnan, H., Rahul, H., & Seshnan, S. (1999). An integrated congestion management architecture for Internet hosts. In *SIGCOMM*.

[Beheshti et al., 2008] Beheshti, N., Ganjali, Y., Ghobadi, M., McKeown, N., & Salmon, G. (2008). Experimental study of router buffer sizing. In *IMC '08 : Proceedings of the 8th ACM SIGCOMM conference on Internet measurement* : ACM.

[Braden, 1989] Braden, R. (1989). RFC 1122 : Requirements for Internet Hosts – Communication Layers. RFC 1122.

[Brakmo et al., 1994] Brakmo, L. S., O'Malley, S. W., & Peterson, L. L. (1994). TCP vegas : New techniques for congestion detection and avoidance. In *SIGCOMM* (pp. 24–35).

[Bram Cohen, 2008] Bram Cohen (2008). The BitTorrent Protocol Specification version 11031. Standard.

[Bush & Meyer, 2002] Bush, R. & Meyer, D. (2002). RFC 3439 : Some Internet Architectural Guidelines and Philosophy. RFC 3439.

[Casey et al., 2007] Casey, S., Hughes-Jones, R., Kershaw, S., Spencer, R., & Strong, M. (2007). Real Time Data Transfer for Very Long Baseline Interferometry. In *IEEE Real Time Conference*.

[Chiang et al., 2007] Chiang, M., Low, S. H., Calderbank, A. R., & Doyle, J. C. (2007). Layering as Optimization Decomposition : A Mathematical Theory of Network Architectures. In *Proceedings of the IEEE*.

[Chiu & Jain, 1989] Chiu, D. & Jain, R. (1989). "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1), 1–14.

[Crowcroft & Oechslin, 1998] Crowcroft, J. & Oechslin, P. (1998). Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP. In *ACM CCR*.

[Dukkipati et al., 2005] Dukkipati, N., Kobayashi, M., Zhang-Shen, R., & McKeown, N. (2005). Processor Sharing Flows in the Internet. In *International Workshop on Quality of Service (IWQoS)*.

[Dukkipati et al., 2006] Dukkipati, N., McKeown, N., & Fraser, A. G. (2006). RCP-AC : Congestion Control to make flows complete quickly in any environment. In *IEEE Infocom High-Speed Networking Workshop*.

[Dykstra, 1999] Dykstra, P. (1999). Gigabit ethernet jumbo frames and why you should care. http ://sd.wareonearth.com/ phil/jumbo.html.

[Enachescu et al., 2006] Enachescu, M., Ganjali, Y., Goel, A., McKeown, N., & Roughgarden, T. (2006). Routers with very small buffers. In *IEEE Infocom*.

[Floyd, 1991] Floyd, S. (1991). Connections with multiple congested gateways in packet-switched networks part1 : One-way traffic. In *ACM CCR*, volume 21.

[Floyd, 1994] Floyd, S. (1994). TCP and Explicit Congestion Notification. In *ACM Computer Communication Review*, volume 24 (pp. 10–23).

[Floyd, 2000] Floyd, S. (2000). RFC 2914 : Congestion Control Principles. RFC 2914.

[Floyd, 2003] Floyd, S. (2003). RFC 3649 : HighSpeed TCP for Large Congestion Windows. RFC 3649.

[Floyd, 2004] Floyd, S. (2004). RFC 3742 : Limited Slow-Start for TCP with Large Congestion Windows. RFC 3742.

[Floyd et al., 2000] Floyd, S., Handley, M., Padhye, J., & Widmer, J. (2000). Equation-based congestion control for unicast applications. In *SIGCOMM*.

[Floyd & Jacobson, 1993] Floyd, S. & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4).

[Floyd & Kohler, 2006] Floyd, S. & Kohler, E. (2006). RFC 4341 : Profile for Datagram Congestion Control Protocol (DCCP), Congestion Control ID 2 : TCP-like Congestion Control. RFC 4341.

[Floyd et al., 2006] Floyd, S., Kohler, E., & Padhye, J. (2006). RFC 4342 : Profile for Datagram Congestion Control Protocol (DCCP), Congestion Control ID 3 : TCP-Friendly Rate Control (TFRC). RFC 4342.

[Fomenkov et al., 2003] Fomenkov, M., Keys, K., Moore, D., & Claffy, K. (2003). Longitudinal study of internet traffic from 1998-2003. In *Winter International Symposium on Information and Communication Technologies (WISICT)*.

[Fredj et al., 2001] Fredj, S. B., Bonald, T., Proutiere, A., Regnie, G., & Roberts, J. (2001). Statistical bandwidth sharing : A study of congestion at flow level. In *SIGCOMM* : ACM.

[Gu & Grossman, 2007] Gu, Y. & Grossman, R. L. (2007). UDT : UDP-based data transfer for high-speed wide area networks. *Comput. Networks*, 51(7), 1777–1799.

[Ha & Rhee, 2008] Ha, S. & Rhee, I. (2008). Hybrid Slow Start for High-Bandwidth and Long-Distance Networks. In *PFLDnet*.

[Hacker et al., 2004] Hacker, T., Noble, B., & Athey, B. (2004). Improving Throughput and Maintening Fairness using Parallel TCP. In *INFOCOM*.

[Iren et al., 1999] Iren, S., Amer, P., & Conrad, P. (1999). The Transport Layer : Tutorial and Survey. In *ACM Computing Surveys*, volume 31.

[ISO/IEC, 1996] ISO/IEC (1996). X.200 : Information technology - Open Systems Interconnection - Basic Reference Model : The Basic Model. International Standard ISO/IEC 7498-1.

[Jacobson, 1988] Jacobson, V. (1988). Congestion avoidance and control. In *SIGCOMM'88*.

[Jacobson & Braden, 1988] Jacobson, V. & Braden, R. (1988). RFC 1072 : TCP Extensions for Long-Delay Paths. RFC 1072.

[Jain, 1989] Jain, R. (1989). A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks. In *ACM CCR*.

[John et al., 2008] John, W., Tafvelin, S., & Olovsson, T. (2008). Trends and Differences in Connection-Behavior within Classes of Internet Backbone Traffic. In *PAM*.

[Katabi et al., 2002] Katabi, D., Handley, M., & Rohrs, C. (2002). Congestion control for high bandwidth-delay product networks. In *ACM Sigcomm*.

[Kelly, 2003] Kelly, T. (2003). Scalable TCP : Improving Performance in Highspeed Wide Area Networks. In *Computer Communication Review*, volume 32.

[Kohler et al., 2006] Kohler, E., Handley, M., & Floyd, S. (2006). RFC 4340 : Datagram Congestion Control Protocol (DCCP) . RFC 4340.

[Konda & Kaur, 2009] Konda, V. & Kaur, J. (2009). RAPID : Shrinking the Congestion-control Timescale. In *INFOCOM*.

[Labrador & Banerjee, 1999] Labrador, M. & Banerjee, S. (1999). Packet dropping policies for atm and ip networks. *IEEE Communications Surveys and Tutorials*, 2(3).

[Liu et al., 2006] Liu, S., Basar, T., & Srikant, R. (2006). TCP-Illinois : A Loss and Delay-Based Congestion Control Algorithm for High-Speed Networks. In *VALUETOOL*.

[Mathis, 2009] Mathis, M. (2009). Relentless TCP. In *PFLDNet*.

[Mathis et al., 1996] Mathis, M., Mahdavi, J., Floyd, S., & Romanov, A. (1996). RFC 2018 : TCP Selective Acknowledgment Options. RFC 2018.

[Medina et al., 2005] Medina, A., Allman, M., & Floyd, S. (2005). Measuring the evolution of transport protocols in the internet. *SIGCOMM Comput. Commun. Rev.*, 35(2), 37–52.

[Menth et al., 2008] Menth, M., Lehrieder, F., Briscoe, B., Eardley, P., Moncaster, T., Babiarz, J., Chan, K.-H., Charny, A., Karagiannis, G., & Zhang, X. J. (2008). PCN-Based Admission Control and Flow Termination.

[Nagle, 1984] Nagle, J. (1984). Congestion Control in IP/TCP Internetworks. In *SIGCOMM*, volume 14 : ACM.

[Nagle, 1987] Nagle, J. (1987). On Packet Switches with Infinite Storage. *IEEE Transactions on Communications*, 35(4).

[Padhye et al., 1998] Padhye, J., Firoiu, V., Towsley, D., & Kurose, J. (1998). Modeling tcp throughput : A simple model and its empirical validation. In *ACM SIGCOMM '98*.

[Padmanabhan & Katz, 1998] Padmanabhan, V. & Katz, R. (1998). Tcp fast start : a technique for speeding up web transfers. In *GLOBECOM*.

[Pang et al., 2005] Pang, R., Allman, M., Bennett, M., Lee, J., Paxson, V., & Tierney, B. (2005). A First Look at Modern Enterprise Traffic. In *Internet Measurement Conference*.

[Pang et al., 2004] Pang, R., Yegneswaran, V., Barford, P., Paxson, V., & Peterson, L. (2004). Characteristics of Internet Background Radiation. In *Internet Measurement Conference*.

[Postel, 1980] Postel, J. (1980). RFC 768 : User Datagram Protocol. RFC 768.

[Postel, 1981a] Postel, J. (1981a). RFC 791 : Internet Protocol. RFC 791.

[Postel, 1981b] Postel, J. (1981b). RFC 793 : Transmission Control Protocol. RFC 793.

[Prasad et al.,] Prasad, R., Dovrolis, C., & Thottan, M. Router Buffer Sizing Revisited : The Role of the Output/Input Capacity Ratio. In *CoNEXT'07* : ACM.

[Rhee & Xu, 2005] Rhee, I. & Xu, L. (2005). CUBIC : A New TCP-Friendly High-Speed TCP Variants. In *PFLDnet*.

[Ryu et al., 2003] Ryu, S., Rump, C., & Qiao, C. (2003). Advances in Internet Congestion Control. *IEEE Communications Surveys and Tutorials*, 5(1).

[Sarolahti et al., 2006] Sarolahti, P., Allman, M., & Floyd, S. (2006). Determining an appriorate sending rate over an underutilized network path. *Elsevier Computer Networks (COMNET) Journal, Special issue on "Hot topics in transport protocols for very fast and very long distance networks"*.

[Schulzrinne et al., 2003] Schulzrinne, H., Casner, S., Frederick, R., & Jacobson, V. (2003). RTP : A Transport Protocol for Real-Time Applications. RFC 3550.

[Shorten & Leith, 2004] Shorten, R. & Leith, D. (2004). H-TCP : TCP for high-speed and long-distance networks. In *PFLDnet'04* Argonne, Illinois USA.

[Sivakumar et al., 2000] Sivakumar, H., Bailey, S., & Grossman, R. (2000). PSockets : The Case for Application-level Network Stripping for Data Intensivve Applications using High Speed Wide Area Networks. In *SuperComputing*.

[Stevens, 1997] Stevens, W. (1997). RFC 2001 : TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001.

[Stewart, 2000] Stewart, R. (2000). RFC 4960 : Stream Control Transmission Protocol. RFC 4960.

[Tan et al., 2006] Tan, K., Song, J., Zhang, Q., & Sridharan, M. (2006). A Compound TCP Approach for High-speed and Long Distance Networks. In *IEEE INFOCOM*.

[Vicat-Blanc Primet et al., 2005a] Vicat-Blanc Primet, P., He, E., Welzl, M., & al. (2005a). *Survey of Protocols other than TCP - GFD 57*. Technical report, Global Grid Forum. GFD 57.

[Vicat-Blanc Primet et al., 2005b] Vicat-Blanc Primet, P., Hughes-Jones, R., & Jin, C. (2005b). *Proceedings of the 3rd International Workshop on Protocols for Very Long Distance networks*.

[Vicat-Blanc Primet & Martin-Flatin, 2005] Vicat-Blanc Primet, P. & Martin-Flatin, J.-P. (2005). Special issue high performance networking and services in grids : the datatag project. *International Journal of Future Generation Computer System, FGCS*, 21(Issue 4).

[Wang et al., 2009] Wang, R., Taleb, T., Jamalipour, A., & Sun, B. (2009). Protocols for Reliable Data Transport in Space Internet. *IEEE Communications Surveys and Tutorials*, 11(2).

[Wei et al., 2006] Wei, D. X., Jin, C., Low, S. H., & Hegde, S. (2006). FAST TCP : motivation, architecture, algorithms, performance. In *IEEE/ACM Transactions on Networking*.

[Wei et al., 2005] Wei, Y., Yuan, C., & Maosheng, R. (2005). A simple streaming media transport protocols based on IPv6 QoS mechanism. In *ICCNMC*, volume 3619 (pp. 951–960).

[Welzl, 2005] Welzl, M. (2005). *Network Congestion Control*. Wiley.

[Xu et al., 2004] Xu, L., Harfoush, K., & Rhee, I. (2004). Binary increase congestion control for fast long-distance networks. In *INFOCOM*.

[Yang & Reddy, 1995] Yang, C.-Q. & Reddy, A. (1995). A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. In *IEEE Network*.

[Zhang et al., 2005] Zhang, Z., Hasegawa, G., & Murata, M. (2005). Is parallel TCP really effective for fast long-distance networks ?

Evaluation Principles Bibliography

[Andrew et al., 2009] Andrew, L., Floyd, S., & Wang, G. (2009). Common TCP Evaluation Suite. Draft, http ://tools.ietf.org/html/draft-irtf-tmrg-tests-01.

[Azzouna & Guillemin, 2003] Azzouna, N. & Guillemin, F. (2003). Analysis of ADSL traffic on an IP backbone link. In *IEEE GLOBECOM*, volume 7 (pp. 3742–3746).

[Bonald et al., 2002] Bonald, T., Oueslati-Boulahia, S., & Roberts, J. (2002). Flow-Aware Admission Control for a Commercially Viable Internet. In *EURESCOM*.

[Briscoe, 2007] Briscoe, B. (2007). Flow Rate Fairness : Dismantling a Religion. *ACM SIGCOMM Computer Communication Review*, 37(2).

[Briscoe, 2009] Briscoe, B. (2009). *Re-feedback : Freedom with Accountability for Causing Congestion in a Connectionless Internetwork*. PhD thesis, UC London.

[Chang et al., 2005] Chang, H., Jamin, S., Mao, Z., & Willinger, W. (2005). An Empirical Approach to Modeling Inter-AS Traffic Matrices. In *ACM Internet Measurement Conference*.

[claffy et al., 2009] claffy, k., Hyun, Y., Keys, K., Fomenkov, M., & Krioukov, D. (2009). Internet Mapping : from Art to Science. In *IEEE DHS Catch Conference*.

[Crowcroft et al., 2003] Crowcroft, J., Hand, S., Mortier, R., Roscoe, T., & Warfield, A. (2003). QoS's Downfall : At the bottom, or not at all. In *ACM SIGCOMM Workshops*.

[Denda et al., 2000] Denda, R., Iv, P. I., & D-Mannheim (2000). The fairness challenge in computer networks. In *QofIS*.

[Ehliar & Liu, 2004] Ehliar, A. & Liu, D. (2004). Benchmarking network processors. In *Swedish System-on-Chip Conference (SSoCC)*.

[England et al., 2005] England, D., Weissman, J., & Sadagopan, J. (2005). A new Metric for Robustness with Application to Job Scheduling. In *High Performance Distributed Computing*.

[Faraj & Yuan, 2002] Faraj, A. & Yuan, X. (2002). Communication Characteristics in the NAS Parallel Benchmarks. In *IASTED PDCS* (pp. 724–729).

[Feldmann et al., 1998] Feldmann, A., Gilbert, A. C., & Willinger, W. (1998). Data networks as cascades : investigating the multifractal nature of internet wan traffic. *SIGCOMM Comput. Commun. Rev.*, 28(4), 42–55.

[Floyd, 2000] Floyd, S. (2000). RFC 2914 : Congestion Control Principles. RFC 2914.

[Floyd, 2008] Floyd, S. (2008). RFC 5166 : Metrics for the Evaluation of Congestion Control Mechanisms. RFC 5166.

[Floyd et al., 2000] Floyd, S., Handley, M., & Padhye, J. (2000). A Comparison of Equation-Based and AIMD Congestion Control. In *Workshop on Modeling of Flow and Congestion Control Mechanisms*.

[Floyd & Kohler, 2008] Floyd, S. & Kohler, E. (2008). Tools for the evaluation of simulation and testbed scenarios. In *http ://www.icir.org/tmrg/draft-irtf-tmrg-tools-05.txt*.

[Floyd & Paxson, 2001] Floyd, S. & Paxson, V. (2001). Difficulties in Simulating the Internet. In *IEEE/ACM Transactions on Networking*.

[Fomenkov et al., 2003] Fomenkov, M., Keys, K., Moore, D., & Claffy, K. (2003). Longitudinal study of internet traffic from 1998-2003. In *Winter International Symposium on Information and Communication Technologies (WISICT)*.

[Ha et al., 2006] Ha, S., Le, L., Rhee, I., & Xu, L. (2006). A step toward realistic performance evaluation of high-speed tcp variants. *Elsevier Computer Networks (COMNET) Journal, Special issue on "Hot topics in transport protocols for very fast and very long distance networks"*.

[Hardin, 1968] Hardin, G. (1968). The Tragedy of the Commons. *Science*, 162.

[Hassan & Jain, 2004] Hassan, M. & Jain, R. (2004). *High Performance TCP/IP Networking : Concepts, Issues, and Solutions*. Pearson Prentice Hall.

[Jain, 1992] Jain, R. (1992). *The Art of Computer Systems Performance Analysis*. Wiley Professional Computing.

[Jain et al., 1984] Jain, R., Chiu, D., & Hawe, W. (1984). *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems*. Technical Report DEC-TR-301, Digital Equipment Corporation.

[Koslovski et al., 2008] Koslovski, G., Vicat-Blanc Primet, P., & Charão, A. S. (2008). VXDL : Virtual Resources and Interconnection Networks Description Language. In *GridNets 2008*.

[Lakshmikantha et al., 2008] Lakshmikantha, A., Srikant, R., & Beck, C. (2008). Impact of file arrivals and departures on buffer sizing in core routers. In *IEEE INFOCOM*.

[Lee et al., 2007] Lee, G., Andrew, L., Tang, A., & Low, S. H. (2007). WAN-in-Lab : Motivation, Deployment and Experiments. In *PFLDnet*.

[Loiseau et al., 2009] Loiseau, P., Gonçalves, P., Dewaele, G., Borgnat, P., Abry, P., & Vicat-Blanc Primet, P. (2009). Investigating self-similarity and heavy-tailed distributions on a large scale experimental facility. In *IEEE/ACM Transactions on Networking*.

[Mascolo & Vacirca, 2006] Mascolo, S. & Vacirca, F. (2006). The effect of reverse traffic on the performance of new tcp congestion control algorithms for gigabit networks. In *PFLDnet'06*.

[Medina et al., 2002a] Medina, A., Fraleigh, C., Taft, N., Bhattacharyya, S., & Diot, C. (2002a). A Taxonomy of IP Traffic Matrices. In *SPIE ITCOM : Scalability and Traffic Control in IP Networks II*.

[Medina et al., 2002b] Medina, A., Taft, N., Salamatian, K., Bhattacharyya, S., & Diot, C. (2002b). Traffic matrix estimation : Existing techniques and new directions. In *ACM SIGCOMM*.

[Meyer, 1980] Meyer, J. (1980). On Evaluating the Performability of Degradable Computing Systems. In *IEEE Transactions on Computers*, volume 29.

[Mo & Walrand, 2000] Mo, J. & Walrand, J. (2000). Fair End-to-End Window-Based Congestion Control. In *IEEE/ACM Transactions on Networking*, volume 8.

[Mogul & Deering, 1990] Mogul, J. & Deering, S. (1990). RFC 1191 :Path MTU Discovery. RFC 1191.

[Poole et al., 2008] Poole, E. S., Chetty, M., Grinter, R. E., & Edwards, W. K. (2008). More than meets the eye : transforming the user experience of home network management. In *DIS '08 : Proceedings of the 7th ACM conference on Designing interactive systems* : ACM.

[Shenker, 1995] Shenker, S. (1995). Fundamental Design Issues for the Future Internet. In *IEEE Journal on Selected Areas in Telecommunications*, volume 13.

[Srikant, 2007] Srikant, R. (2007). Plenary Session : The Role of Mathematical Modelling in the Design of Congestion Control Algorithms for High-Speed Networks. In *PFLDNet*.

[Tsao et al., 2007] Tsao, S.-C., Lai, Y.-C., & Lin, Y.-D. (2007). Taxonomy and Evaluation of TCP-Friendly Congestion-Control Schemes on Fairness, Aggressiveness, and Responsiveness. In *IEEE Network*, volume 21.

[van der Ham et al., 2006] van der Ham, J., Dijkstra, F., Travostino, F., Andree, H., & de Laat, C. (2006). Using rdf to describe networks. *Future Generation Computer Systems, Feature topic iGrid 2005*.

[Wang et al., 2007] Wang, G., Xia, Y., & Harrison, D. (2007). An NS2 TCP Evaluation Tool Suite.

[Wei et al., 2005] Wei, D., Cao, P., & Low, S. (2005). *Time for a TCP benchmark Suite ?* Technical report, Caltech.

[Willinger et al., 1998] Willinger, W., Paxson, V., & Taqqu, M. (1998). *A Practical Guide to Heavy Tails : Statistical Techniques and Applications*, chapter Self-similarity and Heavy Tails : Structural Modeling of Network Traffic. Birkhauser.

[Willinger et al., 1997] Willinger, W., Taqqu, M. S., Sherman, R., & Wilson, D. V. (1997). Self-similarity through high-variability : statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Trans. Netw.*, 5(1), 71–86.

[Yang et al., 2001] Yang, Y. R., Kim, M. S., & Lam, S. S. (2001). Transient behaviors of tcp-friendly congestion control protocols. In *IEEE INFOCOM*.

[Zhang et al., 1991] Zhang, L., Shenker, S., & Clark, D. D. (1991). Observations on the dynamics of a congestion control algorithm : The effects of two-way traffic. In *ACM Computer Communication Review* (pp. 133–147).

Evaluation Methods Bibliography

[Agarwal et al., 2005] Agarwal, S., Summers, J., & Barford, P. (2005). Scalable Network Path Emulation. In *IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*.

[Altman et al., 2005] Altman, E., Avrachenkov, K., & Barakat, C. (2005). A Stochastic Model of TCP/IP with Stationary Random Losses. In *IEEE/ACM Transactions on Networking*, volume 13.

[Baccelli & Hong, 2002] Baccelli, F. & Hong, D. (2002). AIMD, Fairness and Fractal Scaling of TCP Traffic. In *IEEE Infocom*.

[Baccelli & Hong, 2003] Baccelli, F. & Hong, D. (2003). Flow level simulation of large ip networks. In *IEEE INFOCOM*, volume 3.

[Baccelli & Hong, 2005] Baccelli, F. & Hong, D. (2005). Interaction of tcp flows as billiards. *IEEE/ACM Transactions on Networking*, 13(4).

[Blanc et al., 2009] Blanc, A., Avrachenkov, K., Collange, D., & Neglia, G. (2009). Compound tcp with random losses. In *Networking*.

[Bolze et al., 2006] Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P. V.-B., Quetier, B., Richard, O., Talbi, E.-G., & Touche, I. (2006). Grid'5000 : a large scale and highly reconfigurable experimental Grid testbed. *Int. J. of High Performance Computing Applications*, 20(4).

[Bonald et al., 2003] Bonald, T., Olivier, P., & Roberts, J. (2003). Dimensioning high speed IP access networks. In *International TELETRAFFIC CONGRESS*.

[Carson & Santay, 2003] Carson, M. & Santay, D. (2003). NIST Net – A Linux-based Network Emulation Tool. In *ACM CCR*, volume 3.

[Chiu, 2000] Chiu, D. (2000). Some Observations on Fairness of Bandwidth Sharing. In *IEEE ISCC*.

[Cottrell et al., 2005] Cottrell, R. L., Ansari, S., Khandpur, P., Gupta, R., Hughes-Jones, R., Chen, M., McIntosh, L., & Leers, F. (2005). Characterization and Evaluation of TCP and UDP-based Transport on Real Networks. In *PFLDNet*.

[Floyd & Jacobson, 1991] Floyd, S. & Jacobson, V. (1991). On Traffic Phase Effects in Packets-Switched Gateways. In *CCR*, volume 21.

[Gude et al., 2008] Gude, N., Koponen, T., Petit, J., Pfaff, B., Casado, M., McKeown, N., & Shenker, S. (2008). Towards an Operating System for Networks. In *SIGCOMM CCR*.

[Ha et al., 2006] Ha, S., Le, L., Rhee, I., & Xu, L. (2006). A Step toward Realistic Performance Evaluation of High-Speed TCP Variants. *Elsevier Computer Networks (COMNET) Journal, Special issue on "Hot topics in transport protocols for very fast and very long distance networks"*.

[Ha et al., 2007] Ha, S., Le, L., Rhee, I., & Xu, L. (2007). Impact of background traffic on performance of high-speed tcp variant protocols. *Computer Networks*, 51(7), 1748 – 1762.

[Hemminger, 2005] Hemminger, S. (2005). Network Emulation with NetEm. In *linux.conf.au*.

[Ingham et al., 2009] Ingham, T., Rajhans, S., Sinha, D. K., Sastry, K., & Kumar, S. (2009). Design Validation of Service Delivery Platform Using Modeling and Simulation. In *IEEE Communnications Magazine*.

[Joglekar, 2004] Joglekar, A. A. (2004). High Capacity Network Link Emulation Using Network Processors. Master's thesis, University of Utah.

[Kelly et al., 1998] Kelly, F., Maulloo, A., & Tan, D. (1998). Rate control in communication networks : shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, volume 49.

[Kodama et al., 2004] Kodama, Y., Kudoh, T., Takano, R., Sato, H., Tatebe, O., & Sekiguchi, S. (2004). Gnet-1 : Gigabit ethernet network testbed. In *Proc. of 2004 IEEE International Conference on Cluster Computing (Cluster2004)* (pp. 185 – 192).

[Kohler et al., 2000] Kohler, E., Morris, R., Chen, B., Jannott, J., & Kaashoek, M. F. (2000). The Click modular router. In *ACM Transactions on Computer Systems*.

[Kumazoe et al., 2005] Kumazoe, K., Kouyama, K., Hori, Y., Tsuru, M., & Oie, Y. (2005). Transport Protocols for fast long-distance networks : Evaluation of their penetration and robustness on JGNII. In *PFLDNet*.

[Kumazoe et al., 2006] Kumazoe, K., Kouyama, K., Hori, Y., Tsuru, M., & Oie, Y. (2006). Can high-speed transport protocols be deployed on the Internet ? : Evaluation through experiments on JGNII. In *PFLDNet*.

[Kumazoe et al., 2007] Kumazoe, K., Tsuru, M., & Oie, Y. (2007). Performance of high-speed transport protocols coexisting on a long distance 10-Gbps testbed network. In *GridNets*.

[Lee et al., 2007] Lee, G., Andrew, L., Tang, A., & Low, S. H. (2007). WAN-in-Lab : Motivation, Deployment and Experiments. In *PFLDnet*.

[Leith et al., 2008] Leith, D., Andrew, L., Quetchenbach, T., Shorten, R., & Lavi, K. (2008). Experimental Evaluation of Delay/Loss-based TCP Congestion Control Algorithms. In *PFLDNet*.

[Leith et al., 2007] Leith, D., Li, Y.-T., & Shorten, R. (2007). Experimental Evaluation of TCP Protocols for High-Speed Networks. In *IEEE/ACM Transactions on Networking*.

[Li et al., 2006] Li, Y.-T., Leith, D., & Shorten, R. N. (2006). Experimental Evaluation of TCP Protocols for High-Speed Networks. In *Transactions on Networking*.

[Liu et al., 2001] Liu, B., Figueiredo, D., Guo, Y., Kurose, J., & Towsley, D. (2001). A Study of Networks Simulation Efficiency : Fluid Simulation vs. Packet-level Simulation. In *IEEE INFOCOM*.

[Low, 2003] Low, S. (2003). A duality model of TCP and queue management algorithms. In *IEEE/ACM Transactions on Networking*.

[Mascolo & Vacirca, 2006] Mascolo, S. & Vacirca, F. (2006). The effect of reverse traffic on the performance of new TCP congestion control algorithm. In *PFLDnet'06*.

[Mathis et al., 1997] Mathis, M., Semke, J., Mahdavi, J., & Ott, T. (1997). The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. In *Computer Communication Review*, volume 27 : ACM SIGCOMM.

[McKeown et al., 2008] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., & Turner, J. (2008). Openflow : enabling innovation in campus networks. In *SIGCOMM CCR*.

[Nussbaum & Richard, 2009] Nussbaum, L. & Richard, O. (2009). A Comparative Study of Network Link Emulators. In *Communications and Networking Simulation Symposium*.

[Olsén, 2003] Olsén, J. (2003). *Stochastic modeling and simulation of the TCP protocol*. PhD thesis, Uppsala University.

[Ott et al., 1996] Ott, T., Kemperman, J., & Mathis, M. (1996). The stationary behavior of ideal TCP congestion avoidance.

[Padhye et al., 1998] Padhye, J., Firoiu, V., Towsley, D., & Kurose, J. (1998). Modeling tcp throughput : A simple model and its empirical validation. In *ACM SIGCOMM '98*.

[Paxson & Floyd, 1995] Paxson, V. & Floyd, S. (1995). Wide area traffic : the failure of poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3).

[Peterson et al., 2003] Peterson, L., Anderson, T., Culler, D., & Roscoe, T. (2003). A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1).

[Pongor, 1993] Pongor, G. (1993). OMNeT : Objective Modular Network Testbed. In *International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*.

[Rhee & Xu, 2005] Rhee, I. & Xu, L. (2005). CUBIC : A New TCP-Friendly High-Speed TCP Variants. In *PFLDnet*.

[Riley & Ammar, 2002] Riley, G. & Ammar, M. (2002). Simulating Large Networks – How Big is Big Enough ? In *Conference on Grand Challenges for Modeling and Simulation*.

[Riley et al., 1999] Riley, G., Fujimoto, R., & Ammar, M. (1999). A Generic Framework for Parallelization of Network Simulations. In *MASCOTS*.

[Rizzo, 1997] Rizzo, L. (1997). Dummynet : A Simple Approach to the Evaluation of Network Protocols. In *ACM CCR*, number 27.

[Sakumoto et al., 2007] Sakumoto, Y., Asai, R., Ohsaki, H., & Imase, M. (2007). Design and Implementation of Flow-Level Simulator for Performance Evaluation of Large Scale Networks. In *MASCOTS*.

[Sanaga et al., 2009] Sanaga, P., Duerig, J., Ricci, R., & Lepreau, J. (2009). Modeling and Emulation of Internet Path. In *NSDI*.

[Suchara et al., 2005] Suchara, M., Witt, R., & Wydrowski, B. (2005). TCP MaxNet - implementation and experiments on the WAN in Lab. In *IEEE International Conference on Networks*.

[Takano et al., 2006] Takano, R., Kodama, Y., Kudoh, T., Matsuda, M., & Okazaki, F. (2006). Realtime Burstiness Measurement. In *PFLDNet*.

[Takano et al., 2005] Takano, R., Kudoh, T., Kodama, Y., Matsuda, M., Tezuka, H., & Ishikawa, Y. (2005). Design and Evaluation of Precise Software Pacing Mechanisms for Fast Long-Distance Networks. In *PFLDNet*.

[Tan et al., 2006] Tan, K., Song, J., Zhang, Q., & Sridharan, M. (2006). A Compound TCP Approach for High-speed and Long Distance Networks. In *IEEE INFOCOM*.

[Tang et al., 2008] Tang, A., Andrew, L., Jacobsson, K., Johansson, K., Low, S., & Hjalmarsson, H. (2008). Window flow control : Macroscopic properties from microscopic factors. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*.

[Tang et al., 2005] Tang, A., Wei, D., Hedge, S., & Low, S. (2005). Equilibrium and fairness of networks shared by TCP Reno and Vegas/FAST. In *Telecommunications Systems*.

[Tang et al., 2006] Tang, A., Wei, D., Low, S., & Chiang, M. (2006). Heterogeneous congestion control : Efficiency, fairness and design. In *IEEE International Conference on Network Protocols*.

[Venkatachalam et al., 2003] Venkatachalam, M., Chandra, P., & Yavatkar, R. (2003). A highly flexible, distributed multiprocessor architecture for network processing. *Computer Networks*, 41.

[Wang et al., 2005] Wang, J., Wei, D., & Low, S. (2005). Modelling and Stability of FAST TCP. In *IEEE INFOCOM*.

[Wei & Cao, 2006] Wei, D. X. & Cao, P. (2006). NS-2 TCP-Linux : an NS-2 TCP implementation with congestion control algorithms from Linux. In *WNS2 '06 : Proceeding from the 2006 workshop on NS-2 : the IP network simulator* : ACM Press.

[White et al., 2002] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., & Joglekar, A. (2002). An integrated experimental environment for distributed systems and networks. In *OSDI02*.

[Xu, 2007] Xu, L. (2007). Extending equation-based congestion control to high-speed and long-distance networks. *Comput. Networks*, 51(7), 1847–1859.

[Yan & Gong, 1998] Yan, A. & Gong, W. (1998). Time-driven fluid simulation for high-speed networks with flow-based routing. In *Appliend Telecommunications Symposium*.

Experiments Automation Bibliography

[Albrecht et al., 2006] Albrecht, J., Tuttle, C., Snoeren, A. C., & Vahdat, A. (2006). Planetlab application management using plush. *SIGOPS Oper. Syst. Rev.*, 40(1).

[Allman, 1998] Allman, M. (1998). On the generation and use of TCP acknowledgments. *SIGCOMM Comput. Commun. Rev.*, 28(5).

[Arora et al., 2009] Arora, P., Wang, Y., & Rhee, I. (2009). Netset : Automating Network Performance Evaluation . In *PFLDnet*.

[Bonald et al., 2009] Bonald, T., Feuillet, M., & Proutiere, A. (2009). Is the "Law of the Jungle" Sustainable for the Internet. In *Infocom*.

[Buyya et al., 2008] Buyya, R., Yeo, C. S., & Venugopal, S. (2008). Market-Oriented Cloud Computing : Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In *HPCC*. Keynotes.

[Chun, 2004] Chun, B. N. (2004). DART : Distributed Automated Regression Testing for Large-Scale Network Applications. In *International Conference on Principles of Distributed Systems*.

[Crowcroft et al., 2003] Crowcroft, J., Hand, S., Mortier, R., Roscoe, T., & Warfield, A. (2003). QoS's Downfall : At the bottom, or not at all. In *ACM SIGCOMM Workshops*.

[Eide et al., 2007] Eide, E., Stoller, L., & Lepreau, J. (2007). An Experimentation Workbench for Replayable Networkin Research. In *NSDI*.

[Floyd & Paxson, 2001] Floyd, S. & Paxson, V. (2001). Difficulties in Simulating the Internet. In *IEEE/ACM Transactions on Networking*.

[Foster, 2006] Foster, I. (2006). Globus Toolkit Version 4 : Software for Service-Oriented Systems. In *IFIP International Conference on Network and Parallel Computing*.

[Foster & Kesselman, 1999] Foster, I. & Kesselman, C. (1999). *The Grid : Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann.

[Foster et al., 2001] Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid : Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3).

[Jacobson, 1988] Jacobson, V. (1988). Congestion avoidance and control. In *SIGCOMM'88*.

[Leonini et al., 2009] Leonini, L., Riviere, E., & Felber, P. (2009). SPLAY : Distributed Systems Evaluation Made Simple (or how to turn ideas into live systems in a breeze). In *NSDI*.

[Lucio et al., 2003] Lucio, G. F., Paredes-Farrera, M., Jammeh, E., Fleury, M., & Reed, M. J. (2003). OPNET Modeler and NS-2 : Comparing the Accuracy of Network Simulators for Packet-Level Analysis Using a Network Testbed. In *WSEAS Transactions on Computers*, volume 2.

[Newton, 1675] Newton, I. (1675). Considerations on the former reply ; together with further Directions, how to make the Experiments controverted aright. In *Philosophical Transactions of the Royal Society*.

[Pan et al., 2003] Pan, R., Prabhakar, B., Psounis, K., & Wischik, D. (2003). SHRiNK : A Method for Scaleable Performance Prediction and Efficient Network Simulation. In *INFOCOM*.

[Paxson, 1994] Paxson, V. (1994). Growth Trends in Wide-Area TCP Connections. In *IEEE Network*, volume 4.

[Peterson et al., 2003] Peterson, L., Anderson, T., Culler, D., & Roscoe, T. (2003). A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1).

[Prodan & Fahringer, 2002] Prodan, R. & Fahringer, T. (2002). ZENTURIO : An Experiment Management System for Cluster and Grid Computing. In *IEEE Cluster*.

[Simeonidou, 2008] Simeonidou, D. (2008). Photonics, the Optical layer and Transport Protocol issues. In *Plenary Session PFLDNet*.

[Thompson et al., 1997] Thompson, K., Miller, G., & Wilder, R. (1997). Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network*.

[Videau & Richard, 2008] Videau, B. & Richard, O. (2008). Expo : un moteur de conduite d'expériences pour plate-formes dédiées. In *CFSE*.

[Videau et al., 2007] Videau, B., Touati, C., & Richard, O. (2007). Toward an Experiment Engine for Lightweight Grids. In *MetroGrid*.

[Wei & Cao, 2006] Wei, D. X. & Cao, P. (2006). NS-2 TCP-Linux : an NS-2 TCP implementation with congestion control algorithms from Linux. In *WNS2 '06 : Proceeding from the 2006 workshop on ns-2 : the IP network simulator* (pp.9). New York, NY, USA : ACM Press.

[Weiss, 2007] Weiss, A. (2007). Computing in the Clouds. In *netWorker*, number 4.

[White et al., 2002] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., & Joglekar, A. (2002). An integrated experimental environment for distributed systems and networks. In *OSDI02*.

[Willinger & Paxson, 1998] Willinger, W. & Paxson, V. (1998). Where Mathematics meets the Internet. *Notices of the American Mathematical Society*, 45(8).

[Willinger et al., 1998] Willinger, W., Paxson, V., & Taqqu, M. (1998). *A Practical Guide to Heavy Tails : Statistical Techniques and Applications*, chapter Self-similarity and Heavy Tails : Structural Modeling of Network Traffic. Birkhauser.

[Yang & Veciana, 2001] Yang, S. & Veciana, G. D. (2001). Bandwidth sharing : The role of user impatience. In *Proc. IEEE GLOBECOM*.

Evaluation in Practice Bibliography

[Altman et al., 2006] Altman, E., Barman, D., Tuffin, B., & Vojnovic, M. (2006). Parallel tcp sockets : Simple model, throughput and validation. In *Proceedings of the IEEE INFOCOM, 2006*.

[Barford & Crovella, 1998] Barford, P. & Crovella, M. (1998). Generating Representative Web Workloads for Network and Server Performance Evaluation. In *ACM SIGMETRICS*.

[Dukkipati & McKeown, 2006] Dukkipati, N. & McKeown, N. (2006). Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.*, 36(1).

[Ferrari et al., 2004] Ferrari, T., Travostino, F., & al. (2004). Grid network services. In work in progress (Ed.), *https ://forge.gridforum.org/projects/ghpn-rg/document/draft-ggf-ghpn-netservices-1/en/1*.

[Foster et al., 2004] Foster, I., Fidler, M., Roy, A., Sander, V., & Winkler, L. (2004). End-to-end quality of service for high-end applications. *Computer Communications*, 27(14), 1375–1388.

[Ha et al., 2006] Ha, S., Le, L., Rhee, I., & Xu, L. (2006). A Step toward Realistic Performance Evaluation of High-Speed TCP Variants. *Elsevier Computer Networks (COMNET) Journal, Special issue on "Hot topics in transport protocols for very fast and very long distance networks"*.

[Kumazoe et al., 2007] Kumazoe, K., Tsuru, M., & Oie, Y. (2007). Performance of high-speed transport protocols coexisting on a long distance 10-Gbps testbed network. In *GridNets*.

[Laoutaris et al., 2009] Laoutaris, N., Smaragdakis, G., Rodriguez, P., & Sundaram, R. (2009). Delay Tolerant Bulk Data Transfers on the Internet. In *ACM Sigmetrics*.

[Li et al., 2006] Li, Y.-T., Leith, D., & Shorten, R. N. (2006). Experimental Evaluation of TCP Protocols for High-Speed Networks. In *Transactions on Networking*.

[Low et al., 2005] Low, S., Wei, D., & Cao, P. (2005). *Time for a TCP benchmark Suite ?* Technical report, Caltech.

[Mascolo & Vacirca, 2006] Mascolo, S. & Vacirca, F. (2006). The effect of reverse traffic on the performance of new TCP congestion control algorithm. In *PFLDnet'06*.

[Vicat-Blanc Primet, 2003] Vicat-Blanc Primet, P. (2003). High performance grid networking in the datagrid project. *special issue Future Generation Computer Systems*, 19, 199–208.

[Wang et al., 2007] Wang, G., Xia, Y., & Harrison, D. (2007). An NS2 TCP Evaluation Tool Suite.

Path Characterisation Bibliography

[Bolze et al., 2006] Bolze, R., Cappello, F., Caron, E., Daydé , M., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Vicat-Blanc Primet, P., Quetier, B., Richard, O., Talbi, E.-G., & Irena, T. (2006). Grid'5000 : a large scale and highly reconfigurable experimental Grid testbed. *International Journal of High Performance Computing Applications*, 20(4), 481–494.

[Chase et al., 2001] Chase, J., Gallatin, A., & Yocum, K. (2001). End system optimizations for high-speed tcp. *IEEE Communications Magazine*, 39, 68–74.

[Clark et al., 1989] Clark, D. D., Jacobson, V., Romkey, J., & Salwen, H. (1989). An Analysis of TCP Processing Overhead. In *IEEE Communications Magazine*.

[Dunigan et al., 2002] Dunigan, T., Mathis, M., & Tierney, B. (2002). A tcp tuning daemon. In *SuperComputing*.

[Fisk & Feng, 2001] Fisk, M. & Feng, W. (2001). Dynamic Right-Sizing : TCP Flow-Control Adaptation. In *Super Computing : High-Performance Networking and Computing Conference*.

[Floyd, 2003] Floyd, S. (2003). RFC 3649 : HighSpeed TCP for Large Congestion Windows. RFC 3649. experimental.

[Foong et al., 2003] Foong, A. P., Huff, T. R., Hum, H. H., Patwardhan, J. P., & Regnier, G. J. (2003). TCP Performance Re-visited. In *IEEE International Symposium on Performance Analysis of Systemes and Software*.

[Gunter et al., 2000] Gunter, D., Tierney, B., Crowley, B., Holding, M., & Lee, J. (2000). NetLogger : A Toolkit for Distributed System Performance Analysis. In *Mascots Conference* : IEEE.

[Mathis, 1994] Mathis, M. (1994). Windowed Ping : An IP Layer Performance Diagnostic tool. In *INET*.

[Mathis, 2003a] Mathis, M. (2003a). Effective Diagnostic Strategies for Wide Area Networks. NSF proposal ANI-0334061.

[Mathis, 2003b] Mathis, M. (2003b). Pushing up the internet mtu. presentation.

[Mathis et al., 2008] Mathis, M., Heffner, J., O'Neil, P., & Siemsen, P. (2008). Pathdiag : Automated TCP diagnosis. In *Passive and Active Measurement Conference*.

[Mathis et al., 2003] Mathis, M., Heffner, J., & Reddy, R. (2003). Web100 : extended TCP instrumentation for research, education and diagnosis. volume 33.

[Mathis et al., 1997] Mathis, M., Semke, J., & Mahdavi, J. (1997). The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. In *Computer Communication Review*, volume 27 : ACM SIGCOMM.

[Murta & Jonack, 2006] Murta, C. D. & Jonack, M. A. (2006). Evaluating livelock control mechanisms in a gigabit network. In *ICCCN*.

[NDT, 2007] NDT (2007). Network Diagnostic Tool (NDT) : An Internet2 Cookbook. http ://e2epi.internet2.edu/ndt/.

[Prasad et al., 2004] Prasad, R., Jain, M., & Dovrolis, C. (2004). Socket buffer auto-sizing for high-performance data transfers. *Journal of Grid Computing*, 1(4), 361–376.

[Rio et al., 2004] Rio, M., Goutelle, M., Kelly, T., Hughes-Jones, R., Martin-Flatin, J., & Li, Y. (2004). *A Map of the Networking Code in Linux Kernel 2.4.20.* Technical Report DataTAG-2004-1, FP5/IST DataTAG Project.

[Semke et al., 1998] Semke, J., Mahdavi, J., & Mathis, M. (1998). Automatic TCP Buffer Tuning. In *Computer Communications Review*, volume 28 : ACM SIGCOMM.

[Tomonori & Masanori, 2003] Tomonori, F. & Masanori, O. (2003). Performance of optimized software implementation of the iSCSI protocol. In *International workshop on Storage network architecture and parallel I/Os.*

[Wadge, 2001] Wadge, W. (2001). Achieving Gigabit Performance on Programmable Ethernet Network Interface Cards.

[Wu et al., 2006] Wu, W., Crawford, M., & Bowden, M. (2006). The Performance Analysis of Linux Networking - Packet Receiving. In *International Journal of Computer Communications* : Elsevier.