Automated network packet traces analysis methods for fault recognition and TCP flavor identification

Ziad Tlaiss

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
MINES-TÉLÉCOM ATLANTIQUE BRETAGNE
PAYS DE LA LOIRE – IMT ATLANTIQUE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Télécommunications*

Par
Ziad TLAISS

Automated network packet traces analysis methods for fault recognition and TCP flavor identification

Thèse présentée et soutenue à IMT Atlantique, Brest, le 21/12/2023
Unité de recherche : Lab-STICC
Thèse N° : 2023IMTA0384

Rapporteurs avant soutenance :

Chadi Barakat	Directeur de recherche, INRIA Université Côte d'Azur
André Luc Beylot	Professeur, Toulouse INP/ENSEEIHT

Composition du Jury :

Président :	André Luc Beylot	Professeur, Toulouse INP/ENSEEIHT
Examinateurs :	Chadi Barakat	Directeur de recherche, INRIA Université Côte d'Azur
	Mohamed-Aymen Chalouf	Maître de conférences HDR, Université de Rennes 1
Dir. de thèse :	Sandrine Vaton	Professeure, IMT Atlantique
Co-dir. de thèse :	Isabelle Hamchaoui	Ingénieure HDR, Orange Innovation

Invité(s) :

Isabel Amigo	Maître de conférences, IMT Atlantique, Co-encadrante
Alexandre Ferrieux	Ingénieur de recherche, Orange Innovation, Co-encadrant

Acknowledgement

My PhD journey started on the brink of the Covid-19 pandemic, a period that presented numerous challenges and uncertainties. Yet, here I am at the finish line! I would like to express my deepest gratitude to the exceptional individuals who supported and guided me throughout this transformative journey, enabling me to successfully obtain my doctorate despite the many unforeseen obstacles that arose.

First and foremost, I wish to express my deepest gratitude to my PhD supervisors, Sandrine Vaton, Isabelle Hamchaoui, Isabel Amigo, and Alexandre Ferrieux. Their continuous support, guidance, and encouragement have been invaluable throughout my entire journey. From the initial stages of my research to the final submission of my thesis, their consistent presence and insightful feedback have been crucial in shaping my academic development. The daily calls and bi-weekly meetings we held during the lockdowns were not only essential in keeping my research on track but also provided substantial encouragement. I am deeply thankful for their immeasurable contributions to my growth and success.

In addition to my supervisors, I owe a great deal of gratitude to my exceptional team members, whose support has been a constant source of motivation. Our informal chats and discussions whether conducted via screens during lockdowns or in person whenever circumstances allowed, provided a lifeline during the most challenging times.

I would also like to thank the members of the jury for taking the time to read and evaluate the work done in this thesis.

Lastly, I want to express my deepest gratitude to my family and friends for believing in my abilities and offering their unwavering support. Your encouragement has played an integral role in my accomplishments.

Résumé en Francais

L'Internet occupe une place incontestablement cruciale dans la vie quotidienne de la population mondiale. Son importance transcende les frontières et les cultures, devenant un pilier essentiel de la connectivité et de la communication. Cependant, cette ubiquité de l'Internet engendre une demande sans cesse croissante pour une transmission de données à haut débit et une communication en temps réel. Cette demande massive crée ainsi de nouveaux défis majeurs pour les opérateurs de réseau qui s'efforcent de fournir la meilleure qualité d'expérience à leurs utilisateurs. Parmi ces défis, la gestion de la congestion, les retards dans la transmission des données, les pertes de paquets et la préservation de la qualité de service figurent en bonne place. Ces enjeux complexes requièrent des solutions innovantes pour maintenir et améliorer l'expérience des utilisateurs tout en faisant face à une demande en perpétuelle augmentation.

Afin d'améliorer l'expérience des utilisateurs, des algorithmes de contrôle de congestion ont été développés et intégrés dans de nombreux protocoles de transport, tels que TCP et QUIC. Les algorithmes de contrôle de congestion jouent un rôle fondamental dans la régulation des flux d'Internet, revêtant une importance capitale dans le maintien de la stabilité et de l'efficacité de ce vaste réseau mondial. Leur fonction essentielle est de surveiller en temps réel les conditions du réseau, garantissant ainsi que la quantité de données transmises ne dépasse pas la capacité de traitement disponible. En d'autres termes, ils agissent comme des gardiens du trafic sur Internet, veillant à ce que les données circulent de manière fluide, sans engorgement ni saturation des ressources. L'importance de ces algorithmes réside dans leur capacité à prévenir la congestion excessive, qui pourrait entraîner des retards dans la transmission des données, des pertes de paquets et une dégradation de la qualité de service. Grâce à leur intervention proactive, ils veillent à ce que les performances du réseau restent optimales, garantissant ainsi une expérience utilisateur fluide et de qualité. De plus, les algorithmes de contrôle de congestion s'adaptent aux conditions changeantes du réseau, ajustant dynamiquement le débit de transmission des paquets en fonction de la demande et de la capacité disponible. Cette adaptabilité est cruciale pour faire face aux fluctuations naturelles du trafic Internet, qu'il s'agisse de pics de demande ou de variations imprévisibles. En somme, les algorithmes de contrôle de con-

gestion assurent la viabilité et la continuité d'Internet en régulant avec précision les flux de données, contribuant ainsi à maintenir un réseau mondial fiable, rapide et accessible à tous. Ils demeurent un pilier fondamental de l'infrastructure numérique moderne.

Ces dernières années, le domaine du troubleshooting de réseau a attiré une attention significative de la part des chercheurs en raison de la complexité et de l'importance de cette tâche. Il désigne le processus visant à identifier, diagnostiquer et résoudre les problèmes qui surviennent au sein d'un réseau informatique. Il revêt une importance cruciale pour les opérateurs de réseau, car il permet de maintenir le bon fonctionnement du réseau, d'assurer une expérience utilisateur optimale et de minimiser les interruptions de service. La détection des anomalies dans les réseaux nécessite généralement d'effectuer des captures de paquets du trafic et de les analyser. La capture de paquets dans les réseaux informatiques est une pratique essentielle pour le dépannage et revêt une importance capitale pour les opérateurs afin de détecter et comprendre les différents types de dégradation au sein de leurs réseaux. Cette technique consiste à intercepter et enregistrer les paquets de données qui circulent à travers le réseau, permettant ainsi une analyse détaillée de leur contenu et de leur comportement. L'importance de la capture de paquets réside dans sa capacité à fournir une vue granulaire du trafic réseau en temps réel. Elle permet aux opérateurs de surveiller étroitement les échanges de données entre les appareils. Cette visibilité approfondie est particulièrement importante pour détecter et caractériser les différentes formes de dégradation au sein du réseau. De plus, la capture de paquets peut servir de précieuse source d'informations pour la planification et l'optimisation du réseau. En analysant les données capturées sur une période plus longue, les opérateurs peuvent identifier les tendances et les modèles de trafic, ce qui peut les aider à dimensionner correctement leur infrastructure, à améliorer l'efficacité de la gestion du réseau, et à anticiper les besoins futurs.

L'observation des schémas d'émission des paquets éclaire le type de dégradation subi par une connexion. Plus précisément, les paquets sont généralement envoyés selon un schéma spécifique connu sous le nom de "schéma d'envoi de paquets". Dans le cas du trafic de transport fiable où le contrôle de congestion est effectué, comme le trafic TCP ou QUIC, ces schémas sont des décisions prises par l'algorithme de contrôle de congestion, en fonction de sa propre perception des conditions du réseau. En particulier, les experts en troubleshooting se basent sur ces schémas d'envoi des paquets lors du transfert de données (transmission, retransmission, réception des accusés de réception, etc.) pour identifier la cause de la dégradation. L'algorithme de contrôle de congestion intégré aux

piles TCP/QUIC vise à atteindre, de manière équitable, le débit le plus élevé tolérable en toute sécurité par le réseau. Sous son contrôle, le trafic émis diminue dès qu'il détecte des signes de congestion, c'est-à-dire des paquets perdus ou de latence excessive. Ce comportement peut être observé avec une capture des paquets du flux, mais également via la séquence de transitions des états de l'algorithme de contrôle de congestion. Les transitions d'état sont généralement déclenchées par des événements de dégradation tels que la détection de signaux de congestion. La série de transitions de la machine à état fournit des éléments cruciaux pour les experts de troubleshooting.

CUBIC est un algorithme de contrôle de congestion bien connu pour TCP, basé sur la perte de paquets pour décider de l'évolution temporelle de la fenêtre de congestion. Un algorithme de contrôle de congestion comme CUBIC réduit le débit d'envoi en cas de pertes de paquets, ce qui entraîne une baisse de débit pour les utilisateurs. Afin d'offrir une bonne qualité d'expérience à leurs clients, dans le cas de CUBIC, les opérateurs de réseau doivent investir massivement pour améliorer leurs infrastructures afin de minimiser la perte de paquets. En revanche, BBR est un algorithme de contrôle de congestion introduit par Google en 2016, conçu pour utiliser les ressources réseau disponibles de manière plus efficace que CUBIC. BBR optimise le débit d'envoi des paquets en fonction des conditions estimées du réseau comme la bande passante moyenne mesurée et le temps de propagation aller-retour minimal d'un paquet.

L'arrivée de BBR offre aux opérateurs la possibilité d'adapter leurs infrastructures réseau. En effet, BBR offre une meilleure utilisation des ressources, des performances améliorées et une latence réduite, tout en réduisant massivement le besoin en capacité de buffer. Pour les opérateurs de réseau, il est important de comprendre quels sont les algorithmes de contrôle de congestion utilisés sur leur réseau. Cela fournit des informations sur les performances du réseau et le comportement des appareils, aide à identifier les goulots d'étranglement et à optimiser le réseau et permet de prendre des décisions sur la conception, la configuration et la gestion du réseau. En effet, différents algorithmes de contrôle de congestion ont des comportements et des stratégies variables pour gérer la congestion du réseau, et en identifiant la variante de l'algorithme de contrôle de congestion utilisée, les opérateurs de réseau peuvent obtenir des informations sur la manière dont le trafic est contrôlé et optimisé. De plus, lorsque des problèmes réseau apparaissent, la connaissance de l'algorithme de contrôle de congestion utilisé peut aider à diagnostiquer et résoudre les problèmes, car les experts de troubleshooting analysent les paramètres des algorithmes de contrôle de congestion pour identifier les causes potentielles de dégradation

des performances.

La première contribution de cette thèse réside dans l'extraction de caractéristiques permettant d'identifier la cause fondamentale d'une anomalie. Nous avons commencé notre travail en analysant plus de 500 traces de paquets TCP montrant des connexions médiocres que nous avons identifiées comme des anomalies. Nous avons sélectionné ces captures en comparant le débit obtenu lors du téléchargement de fichiers depuis notre serveur en utilisant TCP avec le débit atteint avec UDP. Contrairement à TCP, UDP ne nécessite pas d'établir une connexion avant de transférer des données. UDP envoie des paquets de données sans vérifier si le destinataire les a reçus. Cela signifie qu'UDP offre une communication potentiellement plus rapide, mais au détriment de la fiabilité et de l'impact sur le reste du trafic. Les téléchargements TCP réalisés étaient principalement effectués en utilisant les algorithmes de contrôle de congestion BBR et CUBIC, qui représentent ensemble la majorité du traffic TCP aujourdhui.

Pour capturer ces traces de paquets, des outils similaires à Wireshark ou Tcpdump ont été utilisé. Ces outils capturent les en-têtes des paquets de la couche de transport ainsi que leurs horodatages d'arrivée. Ainsi, nous obtenons une série temporelle à partir d'une capture bidirectionnelle de paquets, qui représente la séquence chronologique des paquets réseau capturés et enregistrés, accompagnée du temps à laquelle chaque paquet a été capturé. Afin d'obtenir un ensemble de données complet, nos sondes ont été déployées de manière stratégique dans une douzaine de filiales d'Orange. En collectant des traces de paquets provenant de différents pays, notre objectif était d'incorporer une diversité de délai de propagation, et de prendre en compte les différences régionales dans les conditions du réseau.

Après la collecte des captures, nous avons utilisé le comportement des algorithmes de contrôle de congestion et les changements entre leurs états pour analyser les captures de paquets. Étudier le comportement des algorithmes de contrôle de congestion, ainsi que leurs états et les transitions entre ces états, revêt une importance cruciale dans la détection des causes fondamentales d'une anomalie dans une capture de paquets. En effet, ces informations fournissent des indices précieux sur la manière dont le réseau gère le trafic et sur la façon dont les données sont transmises entre les appareils. Les variations dans les états des algorithmes de contrôle de congestion peuvent révéler des schémas spécifiques liés à la dégradation du réseau. Par exemple, une transition fréquente vers un état de congestion peut indiquer un problème persistant de surcharge. Comprendre ces variations permet d'identifier la nature exacte des problèmes rencontrés.

En analysant manuellement ces captures de paquets, nous avons classé quatre causes fréquemment observées de dégradation du réseau : les problèmes de transmission, les problèmes de congestion, les problèmes de gigue et les limitations d'application. L'objectif de ce travail n'était pas d'identifier toutes les causes de dégradation, mais de reconnaître celles qui se répétaient le plus fréquemment dans notre réseau. Les problèmes de congestion étaient parmi les plus courants, représentant plus de 50% des cas traités. Les problèmes de transmission et les problèmes de gigue suivaient de près, couvrant ensemble environ 45% des cas. Les problèmes liés aux limitations d'application n'ont été remarqués que dans certains cas spécifiques (5% des cas) où l'algorithme de contrôle de congestion utilisé était trop agressif, ou lorsque la machine réceptrice était ancienne et/ou avec une capacité de buffer limitée.

Il convient de souligner que ce processus d'analyse était particulièrement complexe, demandant un investissement de temps considérable pour examiner attentivement ces captures. Par exemple, l'analyse de chaque capture en elle-même nécessite entre 10 et 30 minutes, selon le contexte, afin de déterminer la cause fondamentale de la dégradation. En outre, bien que la sélection des traces de paquets défectueuses soit une tâche relativement plus simple en comparaison avec l'analyse, il est important de noter que toutes les captures utilisées ont été choisies et téléchargées manuellement. Ci-dessous, nous présenterons les caractéristiques que nous avons identifiées comme représentatives de chaque type de dégradation:

— Les problèmes de transmission se manifestent généralement par des pertes de paquets individuelles et isolées, où des paquets spécifiques sont éliminés sans qu'il n'y ait de signe de congestion associé. Ces pertes sont principalement observées sur le réseau d'accès radio (RAN) et peuvent être attribuées à divers facteurs tels que la présence de radars, les conditions météorologiques défavorables ou encore des interférences dans l'environnement du RAN. Malgré leur caractère isolé, ces pertes de transmission peuvent avoir des conséquences néfastes, notamment si elles se répètent fréquemment. Si ces pertes isolées surviennent au début d'une connexion, elles peuvent provoquer une sortie précoce de l'état Slow-Start de l'algorithme de contrôle de congestion. Cette situation découle de l'interprétation erronée de ces pertes par l'algorithme de contrôle de congestion. En effet, dans ce scénario, l'algorithme considère ces pertes comme un signe de congestion, ce qui entraîne une sous-estimation de la capacité du goulot d'étranglement du réseau. En conséquence, la bande passante disponible n'est pas utilisée à son plein potentiel, ce qui affecte

les performances globales de la connexion.

— La congestion réseau peut survenir lorsque le débit du trafic à écouler dépasse la capacité du réseau. Ce phénomène peut affecter à la fois les réseaux sans fil et filaires. Contrairement aux pertes de transmission, les problèmes de congestion sont associés à des rafales de pertes de paquets. La présence d'une rafale de pertes de paquets, accompagnée d'une augmentation significative du RTT (Round-Trip Time), constitue une preuve claire de la congestion réseau. La rafale de pertes correspond aux paquets surnuméraires arrivés immédiatement après un remplissage du buffer. L'augmentation subséquente du RTT est directement liée au temps de traversée de ce buffer.

— La gigue est un problème réseau qui affecte généralement les utilisateurs sans fil. La gigue peut considérablement altérer les performances en amenant l'algorithme de contrôle de congestion à une sortie précoce de l'état Slow-Start, interprétant à tort la gigue comme un signe de congestion. Pour déterminer si la gigue est la cause sous-jacente des performances médiocres, il est nécessaire d'identifier d'abord le moment de sortie de l'état Slow-Start. Cela est réalisé en analysant le comportement du l'algorithme de contrôle de congestion et en identifiant le point où il quitte la phase Slow-Start. Au cours de cette analyse, nous recherchons spécifiquement des indications de perte de paquets ou d'augmentation du RTT. Si aucune perte de paquets n'est détectée mais qu'une soudaine variance élevée du RTT est observée, cela suggère que l'algorithme de contrôle de congestion est sorti de l'état Slow-Start en raison de la gigue.

— En plus des problèmes de réseau, la performance d'un transfert TCP peut également être affectée par la capacité des extrémités à envoyer ou recevoir des données à un débit suffisant. Dans ce contexte, l'attention se porte sur le cas d'un récepteur lent, où l'extrémité réceptrice, généralement un appareil client, devient le facteur limitant du transfert de données. Alors que les serveurs sont généralement conçus pour gérer le trafic normal sans être submergés, les appareils clients, souvent alimentés par batterie et ayant des ressources limitées, peuvent devenir le goulot d'étranglement dans la communication. C'est pourquoi nous examinons spécifiquement le scénario où l'appareil client agit en tant que récepteur lent. La situation de "récepteur lent" se caractérise par la taille de la fenêtre de réception de l'utilisateur imposant un plafond à la fenêtre de congestion.

Lors de notre analyse des captures de paquets, nous avons observé l'importance cru-

ciale du premier état de l'algorithme de contrôle de la congestion. Cette phase, connue sous le nom de Slow-Start, est vitale dans le processus de transfert de données. En effet, dans l'état Slow-Start, la source estime la capacité du chemin en augmentant de manière exponentielle son débit d'émission jusqu'à la réception d'un signal de congestion, puis elle quitte l'état Slow-Start pour entrer dans une nouvelle phase avec une croissance de taux beaucoup plus faible. Si l'état Slow-Start surestime le goulot d'étranglement, la source dépassera la capacité du goulot d'étranglement et subira ainsi de multiples pertes de paquets, dont la récupération est laborieuse. Au contraire, s'il sous-estime la capacité du goulot d'étranglement et déclenche une sortie précoce de l'état Slow-Start, la source n'utilisera pas pleinement la bande passante disponible et pourra connaître un faible débit. C'est généralement le cas en présence de pertes de transmission ou d'une gigue excessive liée à l'accès radio mobile, même en sous-charge. Dans les deux cas, la limitation "artificielle" du débit entraînera une mauvaise expérience client, sans aucun bénéfice en matière de congestion, puisque celle-ci est inexistante.

La durée de l'état Slow-Start est donc un indicateur clé pour le diagnostic des défauts. Cependant, la détection précise de cette phase était auparavant un processus manuel et chronophage, effectué par des experts en dépannage. Par conséquent, la deuxième contribution significative de cette thèse réside dans le développement d'une méthode automatisée pour détecter l'instant de sortie de l'état Slow-Start. L'importance capitale de cette méthode est qu'elle permet de gagner un temps précieux dans l'analyse des problèmes de réseau. En automatisant la détection de l'état Slow-Start, nous avons pu non seulement accélérer le processus d'analyse, mais aussi garantir une précision et une fiabilité accrues dans l'identification de cette phase cruciale. Cette avancée contribue ainsi significativement à l'amélioration des méthodes de dépannage, mettant en lumière l'importance de cet état et son impact direct sur la performance du réseau.

En effet, cette identification a été rendue possible grâce à une nouvelle représentation graphique: les octets en vol par rapport au numéro de séquence. Pour détecter l'état Slow-Start, nous avons découvert une relation entre la valeur des octets en vol et la valeur du numéro de séquence qui est uniquement valide pendant cette phase. Nous avons pu calculer la pente, dans cette représentation, de la séparatrice entre la région Slow-Start et le reste, ce qui nous a permis d'identifier la fin de cet état en ne considérant que le dernier paquet au-dessus de cette séparatrice. Pour évaluer notre méthode, nous l'avons testée sur les algorithmes de contrôle de congestion CUBIC et BBR. Pour détecter la sortie de l'état Slow-Start dans ces algorithmes, nous avons réalisé une série d'expériences sur un

serveur auquel accèdent plusieurs sondes actives via l'Internet public. Pour générer les logs des algorithmes de contrôle de congestion sur le serveur, nous avons instrumenté les piles Cubic et BBR du noyau Linux, puis effectué des mesures actives en réalisant à partir de nos sondes de nombreux téléchargements avec ces deux algorithmes. Ensuite, nous avons extrait les instants réels de sortie de l'état Slow-Start à partir de ces logs, lesquels ont été utilisés comme oracle pour notre évaluation. Pour cette évaluation, nous avons utilisé 219 captures de paquets de CUBIC et 241 de BBR. Pour évaluer la gravité de l'écart entre la prédiction de notre méthode et le temps obtenu à partir des logs, nous avons rapporté la différence au RTT initial de chaque capture. Avec CUBIC, nous avons remarqué que cette erreur est inférieure à 1 RTT dans plus de 95% des cas. Il s'agit en effet de la meilleure précision que l'on puisse espérer, car la granularité temporelle typique des décisions de l'algorithme de contrôle de congestion est précisément 1 RTT. Avec BBR, nous avons observé que 90% des cas se situent au-dessous de 2 RTT.

La troisième contribution de cette thèse réside dans l'identification de l'algorithme de contrôle de congestion BBR. En effet, l'objectif principal de notre approche était de détecter si un contrôle de l'envoi des paquets ('pacing') est utilisé dans une connexion TCP. Dans un TCP avec pacing et contrairement à un TCP ordinaire, au lieu d'envoyer de nouveaux paquets immédiatement après la réception d'un accusé de réception, les paquets sont espacés uniformément dans le réseau sur tout le temps de RTT. Dans ce cas, les données ne sont pas envoyées en rafales, ce qui entraîne un trafic TCP moins irrégulier. Or, il se trouve que la fraction "avec pacing" du trafic TCP mondial est directement déterminée par les choix d'algorithmes de contrôle de congestion dans les serveurs: BBR, algorithme avancé s'appuyant sur des mesures lissées par RTT, utilise le pacing; CUBIC, algorithme plus ancien et plus simple à décisions "instantanées", ne l'utilise pas. Des études récentes ont montré l'impact du pacing sur l'amélioration des performances TCP, en particulier dans le cas des routeurs qui possèdent des buffers de petite capacité . Par conséquent, la tendance naturelle est à l'augmentation de la proportion de TCP "avec pacing".

Pour les opérateurs de réseau, comprendre si le TCP utilisé sur leur réseau utilise le pacing ou non est important. Par exemple, détecter l'utilisation du pacing au sein des réseaux des opérateurs les aide à mesurer l'irrégularité du trafic TCP résultant. Cette irrégularité pourrait avoir un impact sur la dimension des buffers réseau. Les opérateurs doivent alors ajuster les tailles de buffer et les stratégies de gestion réseau en conséquence. De plus, un trafic irrégulier peut entraîner des pics soudains d'utilisation du réseau, provo-

quant potentiellement de la congestion. En identifiant un trafic irrégulier, les opérateurs peuvent mettre en place des mesures pour lisser le trafic, prévenant ainsi les problèmes liés à la congestion.

Pour différencier le trafic "sans pacing" (CUBIC) du trafic "avec pacing" (BBR), nous avons utilisé la durée de l'inter-paquet, qui représente le temps écoulé entre l'arrivée de deux paquets consécutifs. Nous avons remarqué que les valeurs de l'inter-paquet sont les plus différentes pendant l'état Slow-Start. En effet, La phase Slow Start de BBR et CUBIC présente une augmentation exponentielle similaire du nombre des paquets, mais elles diffèrent dans le mécanisme d'émission détaillé. Contrairement à CUBIC, BBR n'attend pas les accusés de réception des paquets avant d'envoyer de nouveaux paquets, il se base plutôt sur des estimations des ressources disponibles. Cela se traduit par un motif d'émission "lisse", sans les pauses exhibées par CUBIC. Pour avoir une idée des propriétés statistiques de la distribution de l'inter-paquet, nous avons travaillé avec les fonctions de densité de probabilité pour des trafics BBR par rapport à des trafics CUBIC. Nous avons remarqué que la distribution de CUBIC présente une caractéristique différente. Elle est généralement un mélange de deux composantes, des inter-paquets courts pour les paquets à l'intérieur des rafales et quelques inter-paquets longs correspondant à la période de silence entre deux rafales, ce qui signifie qu'il est plus probable d'observer des inter-paquets plus longs. Contrairement à la distribution des inter-paquets de CUBIC, une distribution typique des inter-paquets de BBR présente une caractéristique distincte où les pluparts des inter-paquets ont principalement de petites valeurs. L'occurrence de grands inter-paquets est alors relativement rare dans BBR. La tâche de discrimination entre les distributions BBR et CUBIC peut donc consister à reconnaître les distributions monomodales dans le cas de BBR par rapport aux distributions bimodales dans le cas de CUBIC.

Dans le cas de CUBIC, nous avons remarqué que la composante à inter-paquet longs était écrasée par la composante à inter-paquet courts. Cela est dû au nombre d'événements significativement plus élevé à l'intérieur d'une rafale par rapport au nombre relativement faible de pauses entre les rafales. Cependant, pour discriminer entre un trafic CUBIC et un trafic BBR, il est nécessaire de résoudre ce déséquilibre en donnant plus de poids pour les inter-paquets longs. Nous avons alors pondéré les valeurs de l'inter-paquet par leur propre valeur, ce qui nous a permis d'obtenir une nouvelle fonction de densité rééquilibrée. Ensuite, pour mieux saisir les caractéristiques de la distribution, nous sommes passés des fonctions de densité de probabilité aux fonctions de répartition cumulative. Nous

avons appliqué notre approche sur un ensemble de données dédié à l'entraînement. Notre objectif était de détecter le point qui sépare le mieux les courbes de fonctions de répartition cumulative du trafic CUBIC et BBR. Nous avons choisi ce point en sélectionnant celui qui minimise le taux d'erreur total (non-détections + identifications incorrectes). Dans cette tâche, nous avons utilisé 221 captures de paquets CUBIC et BBR pour l'entraînement, et le taux d'erreur total minimum atteint sur l'ensemble de données d'entraînement est de 2,6 %. Après avoir déterminé le point de décision optimal atteignant ce taux sur l'ensemble de données d'entraînement, nous l'avons testé avec un nouvel ensemble de données composé de 583 captures de paquets BBR et CUBIC dédié à l'évaluation. Sur ce jeu de données d'avaluation, notre modèle a été capable d'identifier les variantes TCP avec un taux d'erreur global de seulement 4,1 %.

Table of Contents

List of figures

Chapter 1

Introduction

1.1 Motivation

The Internet is an essential part of our lives. The ever-growing demand for high-speed data transmission and real-time communication creates new challenges for network operators. Operators face many critical problems in modern computer networks, like congestion and transmission problems. The problems can cause network delays, packet loss, and degraded Quality of Service (QoS), leading to a bad Quality of Experience (QoE) for operators' clients. In order to improve users' experience, Congestion Control Algorithms (CCAs) have been developed and integrated into many transport protocols, such as the Transmission Control Protocol (TCP) and QUIC. CCAs aim to regulate the transmission rate of packets over the network by monitoring network conditions to prevent excessive data transmission that can cause further congestion and thus ensure optimal network performance.

In recent years, the field of network troubleshooting has attracted significant attention from researchers due to the complexity and importance of the task. Network troubleshooting is the process of detecting, identifying, and resolving network issues. Detecting anomalies in networks usually requires packet level traffic capturing and analyzing. Indeed, the observation of emission patterns sheds some light on the kind of degradation experienced by a connection. More specifically, packets are typically sent in a specific pattern known as the "packet send pattern". In the case of reliable transport traffic where congestion control is performed, such as TCP and QUIC traffic, these patterns are the fruit of decisions made by the CCA, according to its own perception of network conditions. In particular, troubleshooting experts rely on packet send patterns during data transfer (transmission, retransmission, received acknowledgments, etc.) to identify the root cause of degradation. For example, in Radio Access Networks (RAN) an individual and isolated packet retransmission indicating a packet loss is typically identified as a transmission problem.

CCAs regulate the flow of data within a network and optimize data transmission for

efficiency and effectiveness. They do this by continuously estimating available network resources (available bandwidth, latency, packet loss, etc.) and adjusting the data transmission rate accordingly. CUBIC is a loss-based well-known CCA for TCP, that uses a cubic function to model the temporal evolution of the congestion window (cwnd) size after a congestion event. A CCA like CUBIC reduces the sending rate in case of high levels of packet losses, leading to a drop in throughput for users. To provide good QoE to their clients using CUBIC traffic, network operators need to invest heavily in order to provide sufficient bandwidth and buffering space, so as to minimize packet loss. Bottleneck Bandwidth and Round-trip propagation time (BBR) is a CCA introduced by Google in 2016, designed to use available network resources more efficiently than CUBIC. BBR optimizes the sending rate of the packets based on estimated network conditions, with a focus on the Bandwidth-Delay Product (BDP), calculated as the measured average bandwidth multiplied by the minimum Round Trip Time (RTT). The arrival of BBR, with its novel approach both to congestion detection and sending rate control, presents an opportunity for network operators to reduce investment in network infrastructure. Indeed BBR offers better resource utilization, improved performance, and lower latency, while massively reducing the need for buffering space. For network operators, understanding the CCAs in use on their network is crucial. It provides information on network performance and device behavior, helps identify bottlenecks and optimize the network, provides insight into network usage patterns, and enables informed decisions on network design, configuration, and management. Indeed, as different CCAs have varying behaviors and strategies for managing network congestion, by identifying the CCA variant in use, network operators can gain insights into how traffic is being controlled and optimized. Additionally, when network issues appear, knowing the CCA being employed can assist in troubleshooting and diagnosing problems, as troubleshooting experts analyze the CCAs' parameters to identify potential causes of performance degradation.

To enhance clients' QoE and improve their QoS, operators use end-to-end diagnosing techniques to troubleshoot networks by analyzing the performance of a communication path between two endpoints. This aims to pinpoint the root cause of problems affecting the data transfer for the end user. For decades, most operators' end-to-end diagnosis methods have been based on the observation of transport protocol packet headers collected from exhaustive packet-level traces captured on network midpoints and endpoints using active and passive probes. This approach remains the cornerstone of troubleshooting for many operators. However, it is jeopardized by the explosive growth of QUIC where

the encryption not only covers packet payload but also packet headers, making them unreadable for a midpoint observer. It is true that active probes, by generating their own traffic, are not impeded by encryption and can detect QoS degradation. However, contrary to passive probes, their representativeness can be questioned for two reasons: first, test traffic is not real client traffic. Second, active probes typically see only a subset of the network. Operators can counter these problems through massive probe deployment, but as current troubleshooting is mainly based on human diagnosis, automation is certainly a key element for dealing with the data deluge collected via such a dense fleet of active probes. There is therefore an urgent need to find new automated approaches and tools to accelerate the troubleshooting process.

The main objective of this thesis is to provide automated techniques that permit network operators to easily detect the degradation root cause and to identify the dominant CCA on their networks. The behavior of a CCA is represented by a finite state machine (FSM) that changes depending on the type of CCA in use. We mainly focus in our work on the state transition of the CCA. By studying these states, we can reveal important information about the network conditions. In addition, due to the particularity of some states in each CCA variant, we can identify the type of CCA in use. In a nutshell, we interpreted the importance of investing in the type of used CCA to offer a better quality of experience to clients by adapting network infrastructures to better suit the dominant CCA.

By analyzing multiple packet traces and extracting important packet patterns, we were able to classify four of the most typical and repetitive degradation root causes relying on the CCAs' behavior and state transition. Thanks to this exhaustive and deep analysis, we manually classified each of these degradations depending on certain characteristics that we extracted and shared with the research community with a short paper.

By studying the finite state machine of the CCAs, we were interested in automatically detecting the Slow-Start (SS) state of a CCA due to its importance for troubleshooting experts in detecting the degradation root cause and identifying the CCA in use. While state transitions can provide valuable insights into potential root causes, the SS exit time holds particular significance in understanding the behavior of the network. In the SS state, the source aims to estimate the capacity of the path by progressively increasing its transmission rate through an exponential growth mechanism known as binary search. This process continues until a congestion signal is detected, indicating that the network is approaching its capacity limit. At this point, the source exits the SS state and enters

a new phase characterized by a significantly slower rate of growth. Based on this deep analysis of the SS state, we were able to develop a tool to automatically detect the exit from the SS state. We also shared our approach and put an online test link to try our method with the research community with a full paper.

After detecting the SS state, we were able to differentiate between a paced TCP traffic and a non-paced TCP traffic. In TCP, pacing involves evenly spacing data transmission over an RTT to avoid burstiness. As use cases, we worked with two CCAs: BBR and CUBIC, BBR is a CCA that paces, and CUBIC is a CCA that does not. Currently, CUBIC and BBR are estimated to represent together more than 73% of TCP traffic. Due to the different methods of estimating the cwnd and detecting congestion, significant differences in throughput can be observed between CUBIC and BBR on the same network. These differences are strongly correlated with certain network conditions that give one of them an advantage over the other. As network operators are interested in monitoring the growth and percentage of paced TCP traffic to optimize their network infrastructure, we set out to detect BBR traffic and distinguish it from CUBIC CCA. We also shared this work with the research community with a full paper.

1.2 Contributions of the thesis

Apart from the introduction and the conclusion, this thesis is composed of two main parts, the first part contains three chapters, and the second part contains two chapters.

Part I focuses on different aspects of network troubleshooting. It provides a comprehensive overview of different aspects of network troubleshooting, including the methods and processes involved, the importance of congestion control algorithms, and the use of packet traces analysis for identifying root causes of network anomalies.

— Chapter 2 emphasizes the importance of network monitoring and network performance measurements for troubleshooting. It provides an overview of different methods and tools that can be used for this purpose.

— Chapter 3 provides an overview of UDP, TCP, and QUIC protocols and their differences. It then discusses the importance of CCAs for improving network performance and the role of CCAs in troubleshooting network issues. Furthermore, the chapter explores the different types of data that can be extracted from network traffic.

— Chapter 4 goes through the use of packet traces analysis as a tool for identifying the root cause of network anomalies. Packet traces are a detailed record of the network

traffic, and analyzing these traces can help identify specific patterns in packets or flows that lead to detecting the root cause of issues in the network. The chapter explores various techniques and tools for analyzing packet traces to isolate and resolve network issues. The contributions of this chapter have been published in [46]:

> Ziad Tlaiss. Anomaly root cause diagnosis from active and passive measurement analysis, *33rd International Teletraffic Congress (ITC-33), PhD workshop*, 2021

Part II focuses on our proposed methods to automate and accelerate the network troubleshooting process. It provides insights into different approaches for monitoring and detecting network performance issues. We mainly focus on analyzing time series and proposing new packet trace visualization techniques and identifying congestion control algorithms.

— Chapter 5 discusses the importance of automating the detection of the SS state for network troubleshooting together with the challenges associated. As the SS is the first state of a CCA in which the algorithm estimates the bandwidth capacity, troubleshooting experts are interested in this particular state to investigate the available resources in order to examine if they are optimally used or not. This chapter proposes a new approach to detect the SS state accurately and automatically. To automatically recognize the SS phase, we utilize a new representation based on finding a relationship between the bytes-in-flights and the sequence number of a packet that is only valid during the SS state. The main benefit of this representation lies in its predictable shape and slope during the SS state phase. The chapter also evaluates the proposed method using traffic generated by the two most prominent CCAs nowadays, CUBIC and BBR. Moreover, the chapter discusses the benefits of the proposed method for network troubleshooting and performance monitoring. The contributions of this chapter have been published in [49] and [48]:

> Ziad Tlaiss, Isabelle Hamchaoui, Isabel Amigo, Alexandre Ferrieux, Sandrine Vaton. Troubleshooting Enhancement with Automated Slow-Start Detection, *26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023
>
> Ziad Tlaiss, Alexandre Ferrieux, Isabel Amigo, Isabelle Hamchaoui, Sandrine Vaton. Automated Slow-Start Detection for Anomaly Root Cause analysis and BBR Identification, *Annals of Telecommunications*, 2023

— Chapter 6 explains how the distribution of the inter-packet arrival times during the SS state can be used to distinguish between CUBIC and BBR algorithms. We present an approach that characterizes the distribution of packet inter-arrival times and use it for differentiating between CUBIC and BBR TCP traffic. We also evaluate the proposed method with real traffic. The contributions of this chapter have been published in [47]:

> Ziad Tlaiss, Alexandre Ferrieux, Isabel Amigo, Isabelle Hamchaoui, Sandrine Vaton. Automated Identification of BBR Traffic based on Packet Inter-Arrival Times Analysis, *35th International Teletraffic Congress ITC (ITC-35)*, 2023

Finally, we summarise the thesis and offer perspectives for future research in Chapter 7.

Part I

Troubleshooting and Congestion Control Algorithms

Chapter 2

Troubleshooting: Methods and Processes

2.1 Introduction

The computer network is a critical component of many organizations, providing a platform for communication, data sharing, and collaboration [40]. Maintaining a computer network is essential for most organizations. Hence, providing good QoE remains one of the most crucial competitive advantages for an Internet Service Provider (ISP) as it directly impacts its brand image. However, room for improvement remains consistent for network operators as many networks are still impeded by crippling issues bitterly reported by customers and commented on by media which affect operators' reputations. Improving customers' experience is a necessary yet delicate task. It relies on continuous and pervasive monitoring of the network and as soon as a degradation is detected, a quick identification and fixing of the source of the problem is demanded [16]. In their quest towards healthy networks, operators usually run continuous and massive observations, a process named network monitoring [42]. Once a degradation is detected, they launch an ad-hoc troubleshooting process so as to identify and locate the fault [50]. These two processes (monitoring versus troubleshooting), however intricate and somehow overlapping, are different in coverage (whole network versus affected entities), timing (a priori versus a posteriori), and intent. While both processes aim to improve network performance, their approach and goals differ.

In this chapter, we explore the crucial concepts of network monitoring and troubleshooting, highlighting their differences and examining the various monitoring and troubleshooting methods available. Furthermore, we focus on network performance measurement and packet trace analysis, which play a critical role in network monitoring and troubleshooting. We explore the use of active and passive measurement techniques and how they can be utilized to gather network data. The chapter examines the advantages

and disadvantages of each method and highlights how they can be applied in troubleshooting.

2.2 Monitoring vs Troubleshooting

2.2.1 Network monitoring

Network monitoring involves the continuous collection of network data, such as traffic volume and bandwidth utilization in order to gain insight into the overall performance of the network. Monitoring tools provide network operators with real-time data and metrics that can detect anomalies to determine if a network is running optimally and to avoid problems before their appearance. There are several different types of network monitoring tools available, each with its own set of features and capabilities [21]. Some tools are designed to monitor network traffic and extract important metrics (e.g. bottlenecks, congestion, delays, etc.), while others focus on monitoring network devices such as switches, routers, and servers.

2.2.2 Network troubleshooting

On the other hand, network troubleshooting is the process of identifying, diagnosing, and resolving problems on the network after their appearance. The troubleshooting process is initiated in response to an anomaly or degradation that has been detected by monitoring tools or reported by users. It involves the use of diagnostic tools and techniques to identify the root cause of the problem (e.g. a misconfigured device, a security breach, etc.). The goal of network troubleshooting is to restore network functionality as quickly as possible and minimize the impact on users by detecting and resolving the source of the problem. Effective network troubleshooting requires an understanding of network protocols, devices, and configurations, as well as experience with troubleshooting tools and techniques. The troubleshooting process can be time-consuming, requiring significant expertise and training to diagnose and resolve issues.

2.3 Network monitoring methods

Network monitoring relies on a system that looks for slow or failing components, which can affect the speed of traffic within the network. Upon detection of these problematic

elements, the system promptly dispatches notifications to network administrators, who are responsible for investigating the root cause of the decreased traffic speed. Network administrators resolve these notifications through comprehensive analysis to identify the underlying issues and devise appropriate solutions.

2.3.1 Counters polling

Counter polling is a technique used to collect data from network devices. It relies on interrogating the network equipment in the lower layers (layers 2 and 3) such as routers and switches to measure their performance using Simple Network Management Protocol (SNMP) which is a protocol used to manage and monitor network devices [19]. Counter polling technique relies on SNMP counters on a network device to obtain essential information such as interface traffic and the percentage of lost packets through a router. The polling is typically done at regular intervals, to provide a continuous stream of data that can be used to identify trends and anomalies over time. Once the data has been collected it can be analyzed to identify issues or potential problem areas on the network.

SNMP proves beneficial due to its widespread support across devices within an IP network. The data accessible through SNMP on a device is organized within an abstract data structure known as a Management Information Base (MIB). A Network Measurement Station (NMS) periodically initiates requests or polls for the relevant SNMP MIB data from routers or other devices. Most routers and switches have standard MIBs that include a cyclic counter, providing information about the quantity of transmitted and received bytes on each interface. Therefore, by utilizing an SNMP poller that consistently captures these counters, we can obtain fundamental traffic statistics for the entire network with minimal additional infrastructure requirements.

SNMP counters do not provide the number of packets per interval; instead, they only offer a cumulative count. To compute packets per interval accurately, precise polling times need to be established. SNMP data is subject to several known limitations. Firstly, data can be lost during transmission as SNMP employs an unreliable transport protocol. Additionally, if the NMS experiences a crash or reboot, data may not be captured effectively. Furthermore, inaccuracies can arise due to suboptimal implementations of SNMP agents. Counter wrapping can occur when a counter reaches its maximum representable value, causing it to reset to zero and resume counting from there. This behavior is defined by SNMP standards. Counter resets can also occur when a router undergoes a reboot. Additionally, controlling the timing of SNMP polls can be challenging, leading to potential

discrepancies in data accuracy.

Although counter polling is highly efficient in detecting major network device problems like slow response times, packet drops, and connectivity issues, however, it is not efficient in QoE improvement as it can't track specific clients to detect their issues.

2.3.2 Traffic sampling

Traffic sampling is a technique used to gather and analyze data from a subset of network traffic [14]. It allows sampling traffic flows based on particular input interfaces and various fields in the packet header. Analyzing the totality of network traffic is often resource-intensive, thus, capturing a smaller sample of the network traffic is then considered. The sampled traffic can then be analyzed for patterns or anomalies that may indicate issues or problems on the network. Sampling can be done using a variety of techniques. For example, network operators can analyze traffic over a specific time interval or select traffic based on a specific protocol or traffic source. Once a traffic sample is collected, it can be analyzed to identify issues.

NetFlow is a network protocol system developed by Cisco, it is used to sample network traffic to provide insight into network performance [15]. It monitors, collects, and records all traffic as it passes into or out of a network interface. When using NetFlow, a network administrator can get IP flow information. In other words, when a packet gets to a router or a switch, Netflow will collect information, such as IP source, IP destination, source port, destination port, Bytes, etc... This information is then sent to a Netflow collector to be analyzed and get important insight on your network, e.g. how much traffic is flowing through the network, the top protocols being used, where the traffic is going, etc... In order to not be overwhelmed by the amount of the collected data, network devices often employ packet sampling techniques for generating NetFlow statistics. However, the use of low sampling rates, one in every one thousand packets, significantly compromises network visibility. This limitation can hinder the identification of performance issues by network administrators. Moreover, by the very nature of its sampling process, Netflow does not deal with loss nor latency, which are the root of many troubles and so it is more oriented to check and analyze the network interface and it is not efficient to detect specific network problems.

2.4 Network troubleshooting method for an improved QoE

After detecting a degradation in the network performance, operators launch an ad-hoc troubleshooting process so as to identify and locate the fault. To achieve this, end-to-end network performance measurements are still the operators' best choice to identify and localize the root cause of the degradation. In this section, we focus on the network performance measurements and the difference between active and passive network performance measurements.

2.4.1 End-to-End network performance measurements

End-to-end network performance measurements refer to the evaluation and analysis of network performance from the source to the destination of data transmission. It involves measuring various aspects of network performance, such as latency, throughput, packet loss, jitter, and bandwidth utilization [37]. These metrics have a direct impact on the QoE spotted by clients. By collecting and analyzing these metrics, network administrators can pinpoint the root cause of problems and take the necessary steps to resolve them. Network performance measurements have several advantages over other troubleshooting techniques when it comes to ensuring good QoE for users as they provide detailed information about the performance of the network for end users, hence operators can track network performance changes over time for a specific user which helps to accurately detect the root cause of degradation.

End-to-end network performance measurements involve the transmission of test packets or traffic between specific source and destination points. Typically, there are two main types of network performance measurements: active and passive measurements [36], each with its unique characteristics and benefits. In the next section, we will introduce the concept of active and passive network measurements and discuss the differences and advantages of each approach.

2.4.2 Active and passive network performance measurements

Active measurements

Active measurements are a type of network measurement that involves injecting test traffic using probes over a network and watching its behavior as it travels through the network to extract and evaluate important network performance measures. Active measurements can be performed using a variety of tools and techniques. For example, traceroute and ping are among the most widely recognized active monitoring tools commonly employed in IP networks to conduct connectivity tests, trace the path of network traffic, and isolate network faults [56].

In our work, we focus on metrics of network performance that we derive from packet traces. Packet traces, also known as packet captures, refer to the collection of data that is captured when network packets are intercepted and recorded as they flow through a network. Packet traces contain information about individual packets, including their headers, payload, size, source and destination IP addresses, and timestamps. Packet traces are often collected using specialized software or hardware tools such as Wireshark or tcpdump and can be saved in various formats for later analysis [12], [9]. Although packet captures are not normally known as active measurement techniques, however, they can also be used for this purpose. Indeed, by using probes to generate traffic, the generated traffic could be captured at the probe side (client side), or in cases where probes are reaching a server that belongs to the operator the capture point could be at the other end-point: the server side.

Figure 2.1 illustrates an example of how an active packet trace can be captured in a network. In the figure, an end-to-end type measurement is being executed between a probe and a server, both belonging to the operator. Here, we execute a download of a file from a server using a probe/robot that plays the role of a customer. We then capture this traffic from the user or server side and obtain a packet trace. We then extract important measures from the packet trace to investigate the performance of the network. For instance, from a packet trace, we can calculate the latency and throughput of the connection by extracting information from the packet headers like the sequence number of a packet. The primary disadvantage of active measurements is that it is invasive, as it introduces additional traffic to the network and consumes network resources. Additionally, with active measurement, we have a representativity issue as test traffic is not real clients' traffic.

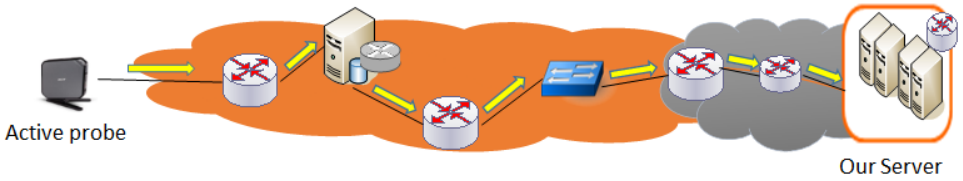

Figure 2.1 – An example of active measurement

Passive measurements

Passive measurements use passive probes to gather packet trace data from real clients/users. These probes can be special purpose devices that capture network traffic, or they can be built into other devices such as routers, switches, or end node hosts. To capture the biggest possible number of users usually passive probes are deployed in the center of the network. The gathered data give a global view or a partial view of a network's performance depending on where the probes are deployed. Since with passive troubleshooting we capture and analyze packet traces from actual users' data, network administrators are informed of issues that directly impact end-users.

Figure 2.2 shows an example of how passive measurements can be performed in a network. In the example, several routers and switches are collecting network traffic to a database. Passive measurements pose a significant challenge due to the vast amount of data they collect. Moreover, the equipment required for capturing and analyzing these data can be costly, as it needs to handle and store all the gathered information. High-performance hardware is necessary, especially for main memory speed. Thus, to ensure adequate measurement accuracy, passive measurements must minimize both the number of measuring devices and the amount of data collected [10]. This can be achieved through several techniques, such as discarding irrelevant data from network packets. For example, by saving only the packet headers and removing the payload, the amount of stored data can be reduced significantly.

2.5 Conclusion

This chapter has highlighted the difference between network monitoring and troubleshooting and the importance of network troubleshooting for operators to ensure a good QoE for their users. It has been shown that there are different monitoring and troubleshooting methods and approaches. The advantages and disadvantages of these

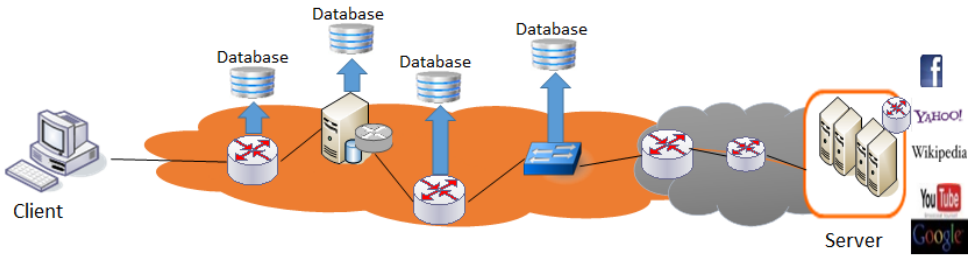

Figure 2.2 – An example of passive measurement

methods have been presented and compared. In our work, we focus on network performance measurements gathered from the analysis of packet captures obtained using active and passive probes. By analyzing packet traces, we can extract important metrics and determine the root cause of network degradation. The following chapter will focus on transport protocols and CCAs types and states. We will show the importance of analyzing the behavior and mechanism of the CCA during a connection to investigate a packet trace. We will also present the most important metrics and measures that we extract from a packet trace.

Chapter 3

Congestion Control Algorithm Importance for Troubleshooting

3.1 Introduction

Transport layer protocols, such as TCP, UDP, and QUIC, play a vital role in ensuring reliable data transfer across computer networks. These protocols are responsible for establishing and maintaining end-to-end connections, as well as for handling congestion control mechanisms. Congestion control is a crucial aspect of network troubleshooting to maintain network health, as it helps to avoid network congestion and maintain good QoS for end-users. This chapter will provide an overview of different transport layer protocols and their congestion control algorithms. Furthermore, we will discuss the different states of the congestion control algorithm and their importance for network troubleshooting. Finally, we will explore the various data extraction techniques that can be applied to packet captures to analyze and troubleshoot network congestion issues.

3.2 TCP, UDP, and QUIC

3.2.1 Transmission Control Protocol

TCP is a protocol used to establish and maintain reliable communication between two devices on a network [18]. TCP operates at the transport layer of the TCP/IP protocol stack and is responsible for providing error checking, flow control, and retransmitting lost data [20]. TCP uses a three-way handshake process to establish a connection as shown in Figure 3.1. This involves sending and receiving packets with synchronize (SYN), synchronize-acknowledge (SYN-ACK), and acknowledge (ACK) flags, respectively [1].

Once a connection starts, TCP divides the data into segments and numbers each segment and sends them individually to the receiver. The receiver acknowledges the

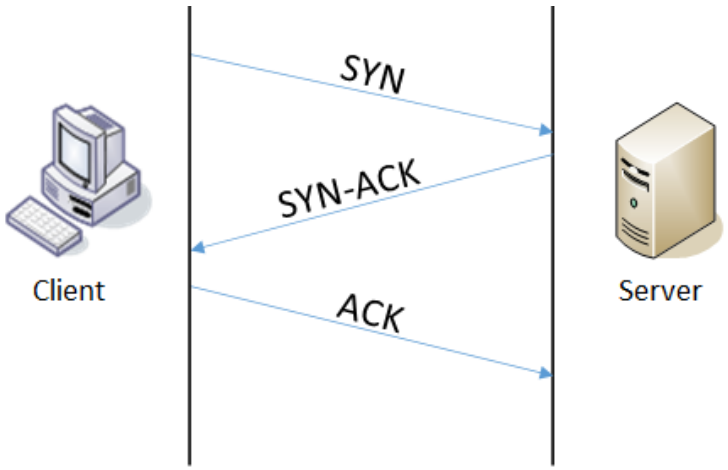

Figure 3.1 – TCP Connection Establishment - Illustration demonstrating the three-way handshake process for establishing a synchronized and reliable connection.

receipt of each segment, and if a segment is not acknowledged within a certain amount of time, TCP retransmits it. This acknowledgment mechanism helps to ensure that all data is received reliably and in the correct order. TCP also provides flow control by adjusting the rate at which data is sent based on the receiver's ability to process it. This helps to prevent data loss or congestion on the network [6]. Additionally, TCP provides error checking by using a checksum to verify the integrity of each segment [17].

TCP is widely used in applications that require reliable and ordered delivery of data (e.g. web browsing, file transfer, email, remote access, etc.). Because of the reliability of TCP as a connection-oriented protocol, it can be slower and less efficient than connectionless protocols like UDP. As the foundation for many applications and services, several studies found that TCP represents 85% of total Internet traffic [33], [38]. AS TCP traffic dominates the Internet traffic, operators are interested in troubleshooting it. Any issues or errors in TCP traffic can lead to degraded performance and ultimately a poor QoE for users. Analyzing active and passive TCP packet traces is still the operators' best choice for anomaly root cause identification. Usually troubleshooting experts extract important metrics from packet traces, more precisely from captured packet headers, and then they analyze these metrics to detect the degradation type. From packet headers operators have access to packet sequence numbers and acknowledgments. By interpreting the evolution of the sequence number troubleshooting experts can detect when losses occurred. This is possible by detecting the absence of the acknowledgment of some packets. Another example of the use of packet headers is by calculating the time between sending a packet

and receiving its acknowledgment. This allows operators to track latency evolution during the connection. More information about the extracted metrics is given in Section 3.5.

3.2.2 User Datagram Protocol

User Datagram Protocol (UDP) is a protocol used in computer networking for transmitting data packets between devices [54]. UDP is a connectionless and unreliable protocol that does not establish a dedicated communication channel between devices before transmitting data. It is designed for applications that prioritize speed and efficiency over reliability and error checking. Unlike other transport protocols, UDP does not provide data flow control or error recovery and does not guarantee that packets will be delivered in the correct order. Instead, it simply sends packets to their destination using the link's maximum capacity and assumes that they will arrive intact. If a packet is lost or damaged in transit, there is no mechanism for retransmission or recovery.

Because of its simplicity, UDP is often used for applications such as online streaming and VoIP (Voice over Internet Protocol), where speed and low latency are critical. However, in the case of use of applications that require reliable data transfer TCP is a better choice.

3.2.3 QUIC

QUIC is a relatively new protocol introduced and developed by Google in 2015. It provides a secure, reliable, and low-latency connection over the Internet [25]. QUIC operates at the transport layer of the TCP/IP protocol stack. But unlike TCP, it is based on UDP, which makes it faster and more efficient than TCP [35] [29]. Furthermore, QUIC is implemented on the application layer in user space instead of the lower level that touches kernel space. As a result, QUIC can be developed and modified in a fast iteration, making it easy to evolve without the need to update the operating system kernel on the client and server site. Figure 3.2 illustrates the differences between QUIC and TCP stacks in terms of their layered architecture.

QUIC is designed to address some of the limitations of TCP, such as the high latency caused by the three-way handshake required to establish a connection and the reliance on retransmission to recover from packet loss [45].

The traditional TCP connection establishment uses a 3-way handshake that takes 1.5 RTTs to finish. If TLS is also used, it takes 3 RTTs to establish a secure connection.

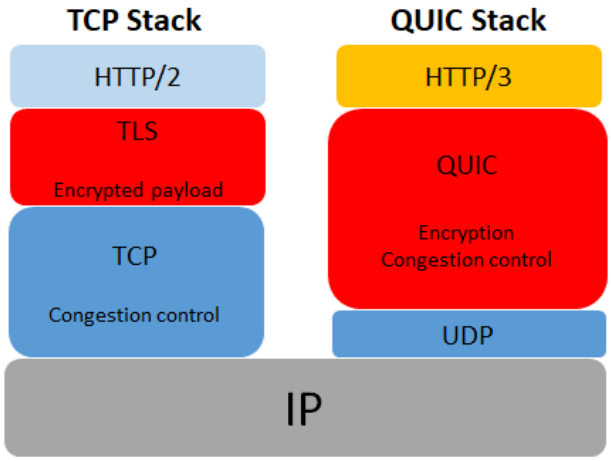

Figure 3.2 – Comparison of the protocol stack between TCP and QUIC, highlighting the differences in the transport and application layers.

However, QUIC allows a faster handshake. It uses a handshake process similar to TCP, but with fewer round trips, and it incorporates mechanisms for error correction and packet loss recovery, similar to those used in TCP.

0-RTT is the name of the feature in the QUIC protocol that allows for establishing a connection and sending encrypted data in the very first round trip. With 0-RTT, a client that has previously established a connection with a server can include encrypted data in the initial request. The server, upon receiving the 0-RTT packet, can decrypt the data and start processing the request immediately. 0-RTT enables faster connection establishment and reduced latency for subsequent requests. Another important feature of QUIC is that it provides end-to-end security [32]. QUIC applies encryption at the transport layer, instead of above it. The entire UDP payload is authenticated, preventing any transparent modification by intermediaries, and almost all transport information is encrypted. All data sent over QUIC is encrypted by default, and there is no option for clear text communication. This helps to protect against attacks, making QUIC a good choice for applications that require high levels of security.

3.2.4 QUIC impact on network troubleshooting

QUIC is supported by several major web browsers such as Chrome and Firefox. It is used by many popular websites and services like Google, Facebook, and YouTube. Its popularity is expected to continue to grow after its standardization in 2022 as more web

developers and network administrators become aware of its benefits.

The deep encryption of QUIC covers not only packet payload, as in the case of TCP, but also packet headers which makes them unreadable for a midpoint observer as shown in Figure 3.3. Due to this encryption, QUIC invalidates many troubleshooting methods that rely on analyzing packet traces. More precisely, with QUIC passive measurements are no longer possible because of its encryption mechanisms, ensuring that packet headers remain secure and private during transmission and thus operators have no longer access to packet sequence numbers or their acknowledgments. Although QUIC has an option that allows monitoring latencies using the spin bit feature [52], it will likely not be implemented by client software (e.g. browsers, mobile applications) in the future [25]. With the spin bit, both endpoints, the client and the server, maintain a spin value, 0 or 1, for each QUIC connection, and they set the spin bit on packets it sends for that connection to the appropriate value. Both sides then send out packets with that spin bit set to the same value for as long as one round trip lasts and then it toggles the value. The effect is then a pulse of ones and zeroes in that bitfield that observers can monitor.

The spin bit feature is optional, that is to say, a QUIC implementation should allow for its deactivation, either globally or on a per-connection basis. Additionally, even if the spin bit is enabled, QUIC should deactivate it for at least one out of every sixteen connections, randomly chosen, to prevent individuals who wish to remain discreet and therefore disable the spin bit from being easily distinguishable from others. This is a classic principle in privacy protection: one should not stand out. The argument made by those who are not in favor of the addition of the spin bit to the protocol is that the exposure of any information beyond the IP header and UDP header is not necessary and not safe [34]. QUIC observability via passive probes will thus probably be lacking for a long time.

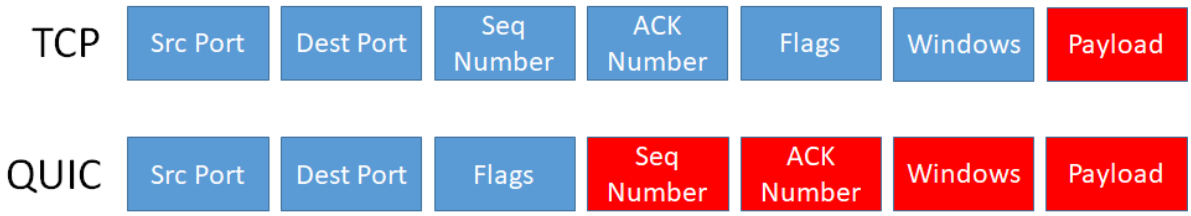

Figure 3.3 – Comparison of packet header encryption in QUIC and TCP, in blue the not encrypted part and in red the encrypted part

It is true that active probes, by generating their traffic, are not impeded by encryption

and can detect QoS degradation. Indeed, the operator that generates this test traffic holds the key to the encryption together with the other end-points. However, contrary to passive probes, the representativity of active probes can be questioned as test traffic is not real client traffic. This point can be balanced through a massive probe deployment. But as current troubleshooting is mainly based on human diagnosis, automation is certainly a key element for dealing with the data deluge collected via such a dense fleet of active probes. Beyond this scalability issue, the whole troubleshooting process should be revisited in light of active measurement specificity. Indeed, even if active end-to-end measurements easily report QoS issues, they give no hints on their location as they indicate the situation and performance of the whole network.

In this context, observing the source behavior appears as a promising strategy. Source emission patterns (send rate, cwnd size, etc..)) derive from decisions of the CCA embedded in TCP and QUIC stacks, and reflect network conditions rather accurately. The CCA is in charge of regulating the source emission to obtain a good throughput without flooding the network. It calms down as soon as it detects early signs of congestion, such as packet loss or delay. Tracking its behavior thus reveals crucial hints for fault qualification. More information on the CCA is given in the next section.

3.3 Congestion Control Algorithm

Amongst the CCAs, the most commonly encountered ones are CUBIC and BBR (Bottleneck Bandwidth and Round-trip propagation time) [22], [8]. They alone contribute to the vast majority of today's traffic as they represent together more than 73% of TCP traffic [2]. In particular, BBR traffic probably represents more than half of all Internet traffic. Other CCAs can be observed but in a significantly lower proportion. However, the CCA landscape remains diverse as many flavors coexist for a given CCA, depending on the underlying implementation (e.g. Linux version on the server side). Furthermore, QUIC primarily uses BBR CCA, even though it started by using CUBIC as its CCA in 2012 but today CUBIC is not directly associated with QUIC. In this section, we introduce the different types of CCAs together with the most known states. We also compare and contrast the similarities and differences between CUBIC and BBR as the two most dominant algorithms among the used CCAs. We are particularly interested in comparing CUBIC and BBR as one of our objectives is to detect the type of CCA in use in our network, whether it is CUBIC or BBR. The importance of CCA identification is presented

in Section 3.4.2.

3.3.1 CCA types

CCAs control the number of packets that can be transmitted over a network depending on the network conditions. They do this by controlling the data transmission rate so that it does not exceed the so-called cwnd size [6]. The cwnd is a TCP parameter that represents the amount of data that a sender can transmit to a receiver before receiving an acknowledgment for the previous packet transmission. The cwnd size is continuously determined by the CCA and adjusted based on network conditions. Depending on the type of CCA, the cwnd size is modified to achieve the best performance while taking into consideration factors such as packet losses, delay, and Packet Delay Variation (PDV), among others. Different CCAs identify and respond to congestion in unique ways. By understanding the CCAs and their responses, operators can use this information to troubleshoot their network or adapt their network infrastructures in order to offer better QoS and QoE to their clients. Thus, this insight is valuable for implementing targeted solutions.

CCAs can be classified into three different types: loss-based, delay-based, and rate-based.

Loss-based CCA:

Loss-based CCAs use packet losses to determine the congestion window size and buffer size. Buffer size determines the amount of data that can be stored by routers and clients before it is processed and transmitted to its intended destination. We note that usually, the network equipment buffer is larger than the client buffer. When packet loss is detected, the algorithm reduces the sending rate of packets to alleviate congestion and prevent further packet loss. These algorithms typically rely on the receipt of ACK packets to determine if packets have been successfully delivered and adjust their sending rates accordingly. A loss-based CCA tries to use as much bandwidth as it can in order to adjust its bit rate to network conditions, via the Additive Increase/Multiplicative Decrease (AIMD) mechanism. The algorithm slowly increases the congestion window when the network conditions are good and strongly decreases its send rate, and thus the cwnd size, when packet loss is detected by a multiplicative decrease factor.

These loss-based CCAs are susceptible to packet loss for reasons other than an over-

loaded link, for example, a broken or unreliable link. This can result in poor performance due to retransmissions and a decreased cwnd which can reduce the effective throughput. Although loss-based algorithms are the simplest to implement, yet, they are limited in their accuracy and functionality. Indeed, with a loss-based algorithm, any packet loss is treated as a sign of congestion, even if this loss is isolated or happens occasionally. The most well-known loss-based CCA is CUBIC [22].

Delay-based CCA:

Delay-based CCAs use variations of latency to determine the cwnd size and buffer size. These algorithms use more advanced measurements to examine and monitor network conditions. They detect that a link is overloaded by measuring the delay in which acknowledgments arrive, also known as the Round-Trip-Time (RTT). Delay-based algorithms can respond to congestion more quickly, even before the first packet is lost, which can improve the performance of some applications. Furthermore, instead of reducing the congestion window by a fixed factor, such as 0.3 for CUBIC, these algorithms gradually reduce the congestion window in proportion to the increase in the RTT value.

VEGAS is the most well-known CCA that uses delay as a key metric [55]. It uses packet delay instead of packet loss to determine the optimal packet sending rate. Unlike CUBIC, which only detects congestion after packet drops occur, VEGAS detects congestion at an earlier stage by monitoring the increase in the RTT values. As a result, VEGAS is able to detect network congestion before packet losses occur.

Rate-based CCA:

Recently, new rate-based CCAs have been introduced. A rate-based algorithm does not adhere to the AIMD mechanism but is designed to improve Internet performance by efficiently using available network resources. These algorithms send data at a rate that is independent of current packet losses. The most well-known rate-based CCA is BBR [8].

BBR works by observing how fast a network is already delivering traffic and the current latency. It then uses that data as input to control packet pacing in a way that can improve performance. In contrast, BBR evaluates congestion by estimating both the RTT and the available bandwidth to determine the optimal sending rate. It operates by maintaining a model of the network path's bottleneck bandwidth and RTT. It adjusts the sending rate based on this model, striving to fill the network pipe without causing buffer bloat or

excessive queuing delays. This mechanism provides higher and more stable throughput, while also improving latency.

The BBR algorithm provides several advantages over other CCAs. CCAs such as CUBIC and Vegas attempt to balance packet transmission with how they detect or respond to congestion or packet loss. This can result in latency and throughput oscillation because they more aggressively back off after a loss and they recover more cautiously in response to packet loss detection. BBR responds more agilely to changing conditions but continues to aggressively attempt to transmit as much data as possible even when there are transient issues.

3.3.2 CCA states

CCA implementations are required to detect network condition changes. There are many CCA implementations, each one improving the older one to be more reactive and more precise. In this section, we discuss the states of the two most popular CCAs, BBR and CUBIC. We explain how these algorithms work, their different states, and the impact of these states on network performance. Understanding the states of these algorithms is crucial for network troubleshooting and optimizing network performance. Indeed, troubleshooting experts interpret the transition between the states and the time the CCA stays in each state to detect the degradation type. Therefore, a deep understanding of the CCAs and their states is essential to troubleshoot networks using packet trace analysis techniques.

CUBIC

CUBIC is a loss-based congestion control algorithm that is designed to efficiently utilize bandwidth and improve the performance of data transfer. It quickly responds to network congestion as a single packet loss provokes a decrease of 30% on the cwnd size [22]. The behavior of the algorithm is represented by a FSM with three states: Slow-Start (SS), Congestion Avoidance (CA), and Fast Retransmit & Fast Recovery (FRR). Figure 3.4 shows the FSM of the CUBIC CCA. We note that we are only interested in the SS and the CA states in our work and thus the FRR state and the transition from SS and CA to this state are not considered.

– **Slow-Start:**

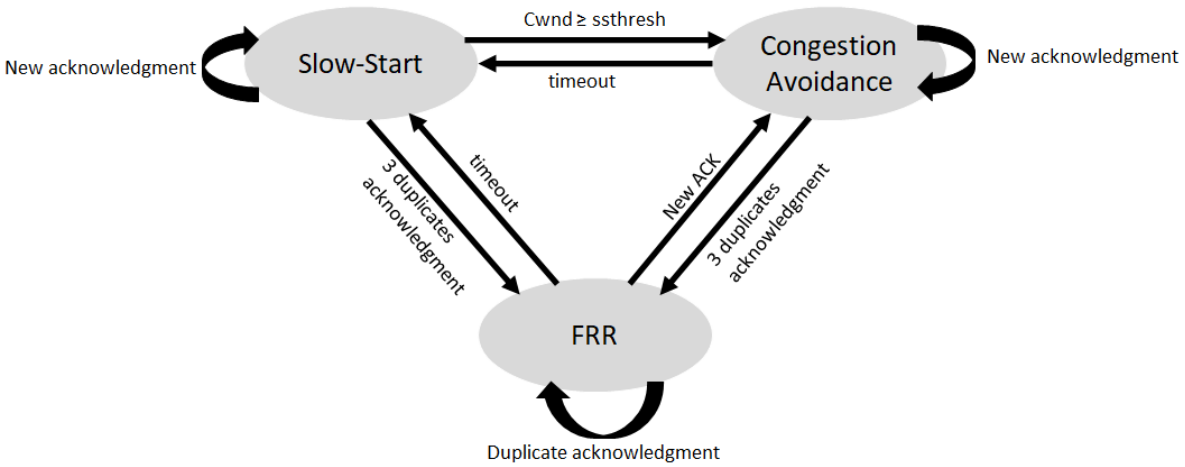

Figure 3.4 – Finite State Machine of CUBIC.

During the SS phase, CUBIC increases its packet sending rate exponentially to quickly reach the bottleneck capacity. The amount of data transmitted between the sender and receiver is controlled by the value of the cwnd and the Receiver WINdow (RWIN) value [6]. The first packets are sent according to the Initial Congestion Window (ICW) and the cwnd size is doubled after every RTT upon reception of acknowledgments for the sent packets [28], for this specific reason we talk about an exponential growth during the SS phase.

CUBIC requires confirmation of received packets through acknowledgments before proceeding to send another burst of packets. In particular, this results in an ON/OFF traffic emission pattern. Finally, upon a congestion signal reception, such as packet loss or increased latency, as indicated by Hystart [5], or when the threshold (ssthresh) is reached, CUBIC switches from the SS phase to the CA state [44]. The ssthresh is a value set to limit slow start, it determines the deactivation of slow start. The cwnd is incremented until it reaches the ssthresh value and then exits the SS state.

– **Congestion Avoidance:**

CUBIC is known as an AIMD-type CCA. To avoid congestion on the network the exponential increase of cwnd must be halted. CA handles this by lowering the cwnd evolution, a linear increase to only 1 packet for every RTT allows CUBIC to control and lower the growth of the cwnd. The rate of growth changes depends on whether CUBIC is in the convex or concave region of the cubic function that models CUBIC

cwnd [24]. This process continues until congestion occurs and is detected either through Retransmit Time Out (RTO) or duplicate acknowledgments [41], [6]. If the RTO occurs, CA will then set ssthresh to half the current cwnd and after this reset cwnd to one and initiate a SS. If CUBIC detects a duplicate acknowledgment, it will force CCA to invoke the Fast retransmit and Fast recovery algorithms [6]. During SS or CA state, if a loss is detected, CUBIC decreases its cwnd value by 30%.

BBR

BBR was introduced by Google in 2016, it optimizes the sending rate of the packets based on estimated network conditions, with a focus on the Bandwidth-Delay Product (BDP) that it calculates by using and estimating the bottleneck bandwidth (btlbw) and latency instead of losses to determine the cwnd size. It uses the maximum available bandwidth and the minimum RTT to build a model of the network capacity in order to determine the packet sending rate [26]. Figure 3.5 shows the FSM of BBR CCA, with four states: STARTUP, Drain, Probe_BW, and Probe_RTT. We note that on our work we are only interested in the STARTUP and Drain states.

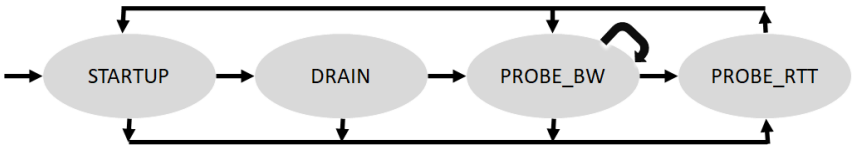

Figure 3.5 – Finite State Machine of BBR.

– **STARTUP:**

During the STARTUP phase, BBR performs a binary search and increases the cwnd size exponentially by doubling the number of transmitted packets within 1 RTT. The goal is to quickly reach the bottleneck bandwidth capacity. The STARTUP phase of BBR and the SS phase of CUBIC have a similar exponential increase in cwnd size, but they differ in the sending rate mechanism. Unlike CUBIC, BBR does not wait for packet acknowledgments before sending new packets, relying instead on estimates of available resources. This results in an emission pattern where there are mainly no interruptions. Finally, once BBR determines that the pipe is full by estimating the maximum bandwidth capacity, more particularly when the bandwidth growth does not exceed 25% after 3 RTT, it exits the SS phase and enters the Drain state.

Even if CUBIC and BBR differ in major ways, they share a few common mechanisms, particularly at the beginning of the connection life. Should it be called "Slow Start" or whatever equivalent (e.g. Start-UP, Hystart or Hystart++, see [5]), the CCA behavior in the first state is the same: an exponential rate growth until reaching the bottleneck capacity. An example of this exponential growth in the cwnd value is shown in Figure 3.6 and Figure 3.7 for a CUBIC and BBR capture. For simplicity's sake, in the remainder of this thesis, we call both of these states, for either CUBIC or BBR, the SS state.

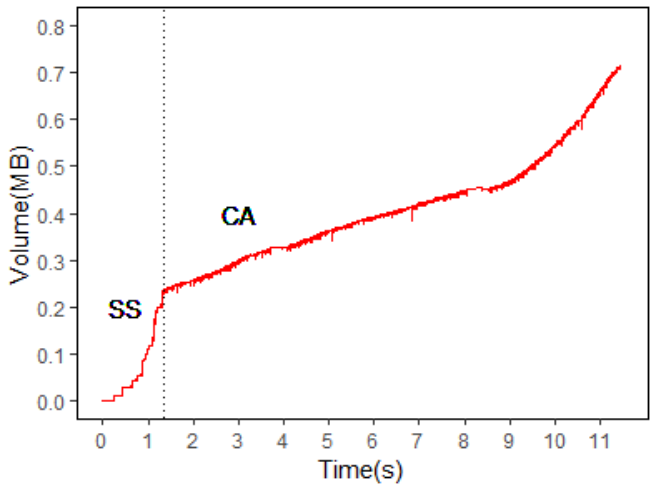

Figure 3.6 – Congestion window over time evolution for a CUBIC capture showing SS & CA states

– **Drain:**

The goal of this phase is to drain packets that have accumulated during the SS phase by reducing its Bytes In Flight (BIF) value to 1 BDP ($BDP = maximumBandwith \times minimumRTT$). The BIF value represents the number of bytes sent but not yet acknowledged, the BIF value is usually strictly bounded to the cwnd size value. The cause of packet accumulation is the aggressive SS that continues to exponentially increase the cwnd even with multiple packet losses. When BBR estimates that the buffer queue is fully drained, BBR enters the ProbeBW phase. BBR can temporally enter the Drain state during the SS state, a phase that can be called STARTUP-Drain. This state is entered when a loss occurs during the SS state. BBR reduces its sending rate to drain any packets that may be in the network and then re-enters the STARTUP state. In order to comprehensively understand the divergent behaviors

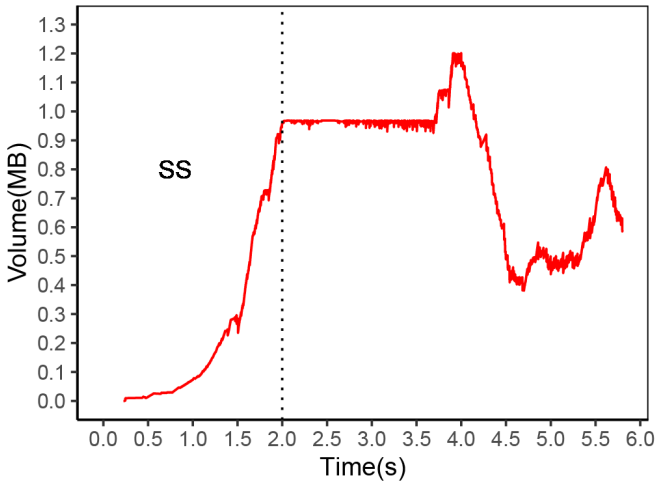

Figure 3.7 – Congestion window over time evolution for a BBR capture showing SS state and the drain state

of CUBIC and BBR Congestion Control Algorithms, a brief comparative analysis of their distinct states is presented in Table 3.1.

State	CUBIC	BBR
SS	Congestion window is doubled every RTT until congestion is detected.	Exponential increase duringg 1 RTT in congestion window.
	SS threshold determines when to exit SS phase.	Exit determined when the bandwidth growth does not exceed 25% after 3 RTT.
CA/Drain	Additive increase: window increased by 1 per RTT. Multiplicative decrease during congestion events.	Bytes in flight value $\leq$ 1 BDP.

Table 3.1 – Comparison of CUBIC and BBR Congestion Control Algorithm States during SS and CA/Drain

3.4 Congestion control algorithms importance for network troubleshooting

CCAs can provide important insights into the cause of the degradation, to pinpoint the origin of the problem (e.g. congestion, transmission, etc.). On the other hand, identifying

the used CCA among operators' networks can help them to better understand network behavior and optimize the system for improved QoE. In this section, we will explore the importance of CCAs for network troubleshooting, focusing on the detection of degradation root causes and the importance of identifying the type of CCA for network operators.

3.4.1 CCAs impact on degradation root cause identification

The CCA role is to prevent congestion collapse while taking into account fairness and improving connection performance [20]. Roughly, the CCA embedded in TCP or QUIC stacks aims at reaching, in a fair manner, the highest throughput safely tolerable by the network. Under its control, the emitted traffic falls back as soon as it detects signs of congestion, that is, lost packets or growing delays. This behavior can be observed with an exhaustive capture of the flow's packets, but also, via the sequence of transitions of the CCA FSM. State transitions are typically triggered by degradation events such as the detection of congestion signals. The FSM transitions series contains factually rich semantic information providing crucial elements for troubleshooting.

3.4.2 CCAs identification impact on operators

CCAs aim to regulate the rate of data transmission over a network by monitoring network conditions to prevent excessive data transmission and causing further congestion. A loss-based algorithm like CUBIC reduces the sending rate in case of high levels of packet losses, leading to a drop in throughput for users. To provide good QoE and QoS and support applications and services running on data centers for clients using CUBIC traffic, network operators need to invest heavily in their Data Center Networks (DCN) in order to provide sufficient bandwidth, speed and minimize packet loss rates.

The arrival of BBR, which aims to enhance network connection performance and efficiency, presents an opportunity for network operators to reduce investment in network infrastructure. BBR offers better resource utilization, improved performance and lower latency, reducing the need for additional resources to support the same number of users or applications.

For network operators, understanding the CCAs in use on their network is crucial. It provides information on network performance and device behavior, helps identifying bottlenecks and optimizing the network, provides insight into network usage patterns, and enables informed decisions on network design, configuration and management. Hence,

network operators are interested in automatically identifying the type of CCA in use on their network, particularly if it is BBR or not, in order to optimize their network infrastructure accordingly.

3.5 Data: extraction, processing and analysis

Packet captures are a valuable source of information for network troubleshooting and performance analysis. They allow us to observe network traffic in real-time and to capture the details of individual packets including their headers and payloads. However, the captured data file is often too large and complex to be analyzed directly, it is then necessary to extract specific metrics from packet headers to gain insight into specific issues to identify and diagnose network problems such as packet loss, delay, and throughput. In this section, we will discuss the different types of data that can be extracted from packet captures and the steps involved in processing these data to extract meaningful metrics to be analyzed in order to troubleshoot networks.

3.5.1 Data extraction

In this step, we use active and passive probes to carry out performance measures. These probes have a similar functionality to Wireshark [12] and TCP dump [9]. These tools capture transport layer packet headers together with their arrival times and so we obtain a time series from a bidirectional packet capture that refers to the chronological sequence of network packets that have been captured and recorded along with the time at which each packet was captured. An example of a time series from a packet capture is shown in Figure 3.8.

Captured data will be treated in real time or stored for further investigation. The most important metrics that we collect directly from packet headers are :

- Sequence number (SEQ): identifies the first byte in a segment [53]. For a better match-up with the acknowledgments, we denote SEQ as the last byte of the transmitted segment plus one, i.e. $SEQ = sequence\,number + length$. By redefining the SEQ number it is easier to analyze the SEQ values with the acknowledgment values.
- Acknowledgment (ACK): it informs the source about the sequence number of the next expected segment.

Time (sec)	SEQ (byte)
47.740081	38679095
47.740272	38680543
47.740291	38681991
47.740314	38683439
47.740322	38684887
47.740533	38686335
47.740551	38687783
47.765699	38689231
47.765717	38690679
47.765737	38692127
47.765745	38693575
47.810710	38695023
47.810728	38696471
47.810748	38697919
47.810756	38699367
47.810978	38700815

Time (sec)	ACK (byte)
47.739973	38616831
47.740098	38619727
47.740118	38622623
47.765639	38628415
47.810644	38644343
47.811003	38650135
47.833856	38655927
47.872981	38664615
47.873527	38673303
47.949871	38684887
47.949995	38687783
47.975502	38693575
48.020602	38709503
48.020855	38715295
48.043443	38721087
48.082784	38732671

Time (sec)	BIF (byte)
47.740081	62264
47.740272	57920
47.740291	59368
47.740314	60816
47.740322	62264
47.740533	63712
47.740551	65160
47.765699	60816
47.765717	62264
47.765737	63712
47.765745	65160
47.810710	50680
47.810728	52128
47.810748	53576
47.810756	55024
47.810978	56472

Figure 3.8 – A bidirectional packet trace showing the captured metrics with their arrival times

– Receiver window (RWIN): it identifies the number of bytes that the receiver can accept. It is sent to the source by the receiver in every ACK.

3.5.2 Data processing

This step consists of using algorithms that will manipulate the data collected in order to present them in a more efficient way which will facilitate the analysis of the data. Different manipulations like aggregating the packet headers and timestamps will not only allow us to present the data in a much simpler form but it will even allow us to calculate some important indicators. The most significant ones are:

– Bytes in flight (BIF): indicates the amount of data bytes sent by the source but not yet acknowledged. BIF is not included in packet headers but can be deduced from SEQ and ACK values by deducing the last received ACK value from the last emitted SEQ value as shown in Figure 3.8 and Figure 3.9. As can be noticed in Figure 3.8, at time 47.740081 sec the BIF was calculated using the SEQ value at the same time and the ACK value that was captured immediately before the SEQ arrival time. In this case, the BIF value is equal to $38679095 - 38616831 = 62264$ bytes. We note that the BIF value is strictly bounded by the cwnd size at any given time.

– Round-Trip-Time (RTT): represents the delay between a packet emission and the

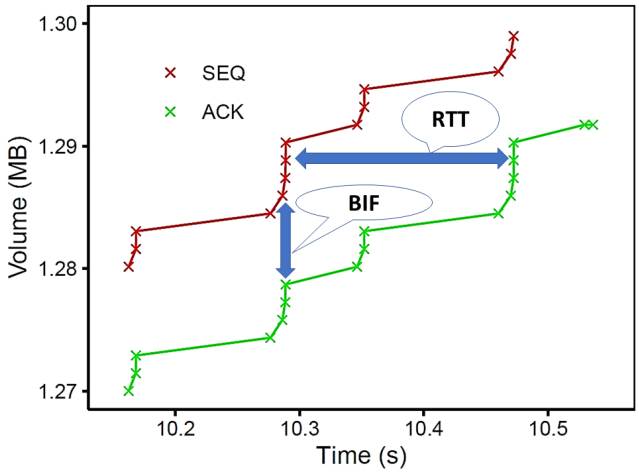

Figure 3.9 – BIF and RTT calculation method

reception of the corresponding acknowledgment. We can calculate the RTT value using the SEQ and corresponding ACK arrival times as shown in Figure 3.9. As can be noticed in Figure 3.8, if we focus on the SEQ and ACK tables, more precisely let us track the SEQ value of 38684887 bytes at time 47.740322 sec. The ACK of this packet is at time 47.949871 sec, thus, the RTT value is here equal to $47.949871 - 47.740322 = 0.209$ sec.

– Packet inter-arrival times (INTRPKT): it represents the elapsed time between the arrival of two consecutive packets. For example, in Figure 3.8, if we focus on the first two consecutive SEQ values: 38679095 bytes and 38680543 bytes, in this case, the INTRPKT is equal to $47.740272 - 47.740081 = 0.000191$ sec.

3.5.3 Data analysis

Data analysis is the process of examining and interpreting data in order to extract meaningful insights and draw conclusions. This step typically consists in visually analyzing the temporal evolution of the above-mentioned metrics, e.g. with a tool such as tcptrace [30]. For each trace, a human expert should visually detect the state transitions and QoS degradations affecting the connection to identify patterns, trends, and anomalies that may indicate issues with network performance or reliability. Data analysis helps network administrators pinpoint the root cause of the problem and take the necessary steps to resolve it. This process demands a deep knowledge of the CCA mechanism by troubleshooting experts to quickly detect a degradation root cause. Examples of the analysis

of packet traces in order to extract the root cause of degradation are given in Chapter 4. On the other hand, a well-trained eye can sometimes manually detect the type of CCA in use by analyzing different captured metrics. This is possible by tracking the evolution of the BIF over time and tracking the changes in the states and the behavior of the CCA in each state.

Recently, network operators have shown increasing interest in automating the troubleshooting process. Indeed, as networks are becoming more complex with a growing number of devices, services, and users, manual troubleshooting is time-consuming, and less accurate due to human error and it can lead to long downtime for users. Automation should allow operators to quickly identify the root cause of a problem and take appropriate actions to resolve it. It should also enable them to detect and address potential issues before they impact the network performance. With automation, operators can increase the efficiency of their troubleshooting process in order to improve their QoS and the QoE for their users.

3.6 Conclusion

In this chapter, we explored the different transport protocols available and we highlighted the importance of CCAs for network troubleshooting. We focused on the two popular CCAs CUBIC and BBR and we discussed their various states and roles. By extracting important data from packet captures, we can identify the root cause of network degradation and determine the type of CCA in use, in this aspect we presented the most important metrics that we extract from packet captures.

In the next chapter, we will introduce our packet capture analysis techniques together with our interpretation of the CCA mechanism in order to detect the root cause of the degradation.

Chapter 4

Packet Traces Analysis for Anomaly Root Cause Identification

The contributions of this chapter have been published in the conference ITC [46]:

— Ziad Tlaiss. Anomaly root cause diagnosis from active and passive measurement analysis, *33th International Teletraffic Congress (ITC-33), PhD workshop*, 2021

4.1 Introduction

This chapter serves as an introduction to our methods and strategies for detecting the underlying causes of anomalies that impact clients' QoE. Our analysis involved the processing of faulty packet captures to identify the most recurring root causes of anomalies and determine the essential characteristics required for effective diagnostics. The primary objective was to identify common characteristics shared among the analyzed packet traces.

We have identified four frequently encountered root causes of anomalies, namely transmission problems, congestion problems, application limitations, and latency/jitter issues. A transmission problem refers to a flaw that occurs during the process of sending data from one device to another over a network. These errors can happen due to various reasons and can occur at different layers of the network stack. A congestion problem occurs when there is a traffic overload in a network. Application Limited problem happens when the performance of a network connection is limited not by the network itself but by the processing capabilities or inefficiencies within the application running on the client device. The jitter problem refers to the variability in the delay of received packets in a network. It is the difference in the time it takes for packets to reach their destination.

To provide insights into our analysis, we present the results by showcasing the curve

patterns of the metrics described in Chapter 3 that we observed for each of these anomaly types. Additionally, we present the curve pattern representing a normal/good connection. It is important to note that our packet traces analysis heavily relies on understanding the behavior of the CCA. As a result, a deep knowledge of CCA functionalities is essential for accurate analysis and interpretation of the results (refer to Section 3.3 for more details). By examining the curve patterns of relevant metrics, we aim to establish a framework for diagnosing and troubleshooting anomalies that affect clients' QoE. This knowledge will enable network administrators and engineers to quickly identify and address the root causes of anomalies, leading to improved network performance and enhanced user experiences. The following sections will explore in detail our methodology, providing a comprehensive understanding of the analysis process and the insights gained from our study. This study may be an important building block in the future for automating the detection of the root cause of degradation and for the automation of the troubleshooting process.

4.2 Packet traces capturing and visualisation

4.2.1 Packet traces capturing techniques

To ensure the representativeness of our packet traces, we collected them using active probes that conducted numerous downloads from our servers. The servers belong to Orange and are based in France and they are used by Orange engineers to troubleshoot the network. The conducted downloads were predominantly carried out using: BBR, CUBIC, and UDP. By utilizing these different CCAs, we aimed to capture a diverse range of network scenarios and behaviors. In order to achieve a comprehensive dataset, our probes were strategically deployed across 12 Orange affiliates, as shown in Figure 4.1. The green dot represents a probe in use and the red dot represents a probe that was put in sleep mode or shut down. By capturing packet traces from various countries, we aimed to incorporate geographical diversity and account for regional differences in network conditions. Each country represented in our dataset exhibited varying congestion levels, reflecting different network usage patterns during peak and off-peak hours.

Capturing packet traces under different congestion levels allowed us to observe the behavior of the network and the corresponding performance metrics during periods of high and low traffic. This comprehensive approach enabled us to gather a rich and diverse

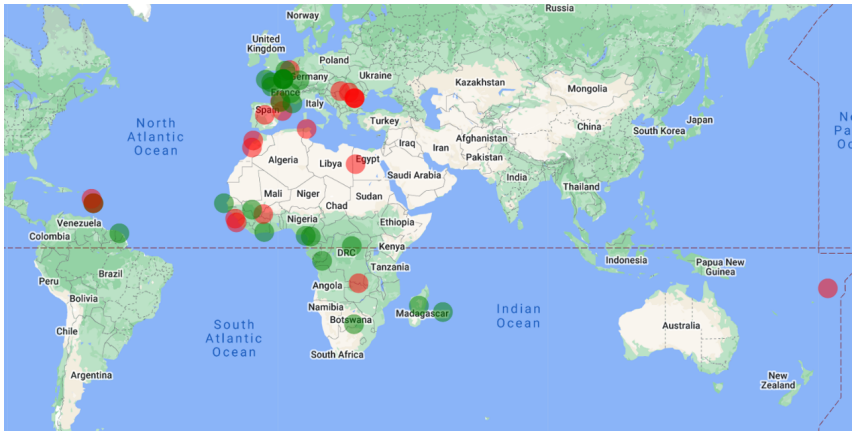

Figure 4.1 – Active probes installed in different countries are represented with green and red dots

set of data that covers a wide range of network scenarios and congestion conditions. The extensive coverage of Orange affiliates and the inclusion of diverse congestion levels in our dataset enhance the reliability and generalizability of our findings in terms of detecting the most repeated degradation root causes among networks. This ensures that our analysis and conclusions are applicable to a broad range of network environments.

Our study primarily focused on the analysis of TCP packet traces exhibiting poor connections. To identify TCP packet traces with bad connections, we employed a comparative approach by comparing the throughput obtained during file downloads using TCP BBR or TCP CUBIC with the throughput achieved using a UDP connection. UDP, being a connectionless protocol that prioritizes speed and efficiency over reliability, provides a reference point for comparison. In an ideal condition, the throughput of TCP and UDP connections should be similar. Therefore, any significant difference in throughput between the TCP and UDP connections is considered as indication of an anomaly within the TCP packet trace. This approach allows us to detect instances where the TCP connection deviates from the expected behavior, potentially pointing to underlying issues affecting the connection quality. Upon collecting the necessary packet traces, we conducted a detailed analysis of each trace. Manual classification was employed to identify and categorize the anomalies' root causes that we had previously identified. By examining various performance metrics, including throughput, we were able to gain insights into the nature and characteristics of the anomalies present in the TCP packet traces. This methodology enabled us to effectively identify and classify different types of anomalies, providing a comprehensive understanding of the root causes impacting TCP connections'

quality.

4.2.2 Packet traces visualization and analysis techniques

In this section, we dive into the analysis technique that we follow in each capture. To analyze the captured packet traces, our initial step involved graphically visualizing the time series of the various metrics mentioned in Section 3.5: SEQ, ACK, BIF, etc... We visualize each metric by plotting its values over time (in seconds). This visual representation allowed us to gain insights into the behavior and patterns exhibited by these metrics throughout the duration of each packet trace. As we rely on the CCA to detect and diagnose network degradation types, we focused on identifying specific abnormalities in the curves of the metrics. Our particular interest lies in investigating the following aspects within each packet trace: the SS state phase, packet losses, latency, and client buffer. We focus on these aspects because they provide a comprehensive view of how data flows through the network, allowing for efficient troubleshooting. For example, Having many packet losses or/and high latency directly affects the end-user QoE.

The SS state stage holds great significance in network troubleshooting. While the primary purpose of the CCA exiting the SS state is to prevent bandwidth saturation, it can occur due to various circumstances. For example, jitter can also prompt the CCA to exit this state. Understanding the underlying reasons behind the SS state exit is crucial for accurate diagnosis and effective troubleshooting of network issues. By carefully analyzing the metrics and identifying anomalies or irregularities in the curves, we can gain deeper insights into the behavior of the network and the factors influencing its performance. This detailed examination allows us to pinpoint potential causes of degradation, such as congestion or excessive latency, which can have a significant impact on the QoS experienced by end users.

Slow-Start state importance during a connection

During the analysis of a packet trace for any type of CCA, the SS exit time emerges as a crucial element for troubleshooting experts. While state transitions can provide valuable insights into potential root causes, the SS exit time holds particular significance in understanding the behavior of the network. In the SS state, the source of the packet trace aims to estimate the capacity of the path by progressively increasing its transmission rate through an exponential growth mechanism known as binary search. This process

generally continues until a congestion signal is detected, indicating that the network is approaching its capacity limit. At this point, the source exits the SS state and enters a new phase characterized by a significantly slower rate of growth.

Should the SS overestimate the bottleneck, then the source will exceed the bottleneck capacity and thus experience multiple packet losses, from which recovery is painful. This situation can be particularly problematic as the source will need to recover from these losses, resulting in degraded performance and potentially impacting the user's QoE. However, it is worth noting that bottleneck overestimation is relatively uncommon in practice. On the other hand, if the SS underestimates the bottleneck capacity and triggers an early exit from the SS state, the source will underutilize the available bandwidth. This can result in poor throughput and minimal network performance. Underestimation of the bottleneck capacity is a much more common cause of low performance in network scenarios.

An early SS exit occurs when the SS mistakenly detects a congestion signal, leading to an unnecessary reduction in the transmission rate. This can happen due to various factors, including transmission losses or excessive jitter, particularly in Radio Access Networks (RAN), even when they are underloaded. In such cases, limiting the transmission rate based on a false congestion signal only serves to restrict the throughput, resulting in a disappointing customer experience without any real benefit in terms of reducing nonexistent congestion. Understanding the dynamics of the SS and its potential shortcomings is crucial for troubleshooting network performance issues. By accurately detecting and responding to congestion signals, network protocols and algorithms can optimize the utilization of available resources and deliver an improved QoE to users.

Packet losses

The identification of packet losses in a packet trace is a crucial step in diagnosing network anomalies. One effective method for detecting packet losses is by examining the sequence number (SEQ) values of the transmitted packets. A decrease in the SEQ value indicates packet retransmission, implying that the original packet was lost or not successfully delivered. In Figure 4.2, we can observe an example of such a scenario. At around 2.27 seconds, there is a packet transmission indicated by the initial SEQ value. However, at 2.45 seconds, there is a noticeable drop in the SEQ value, indicating a retransmission of the packet. This retransmission suggests that the initial transmission was lost or did not reach its intended destination. The presence of packet losses in a packet trace can

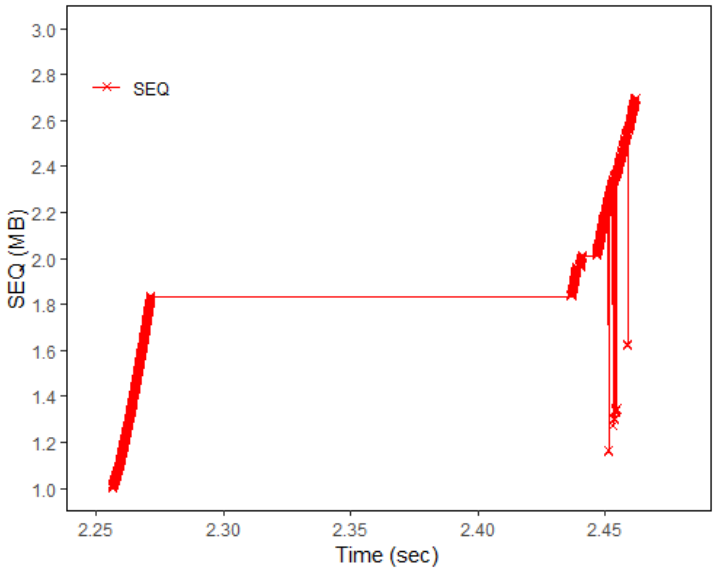

Figure 4.2 – A use case of packet retransmission indicating packet losses

provide valuable insights into the underlying network conditions and potential anomalies. By accurately detecting and analyzing packet losses, we can further investigate the root causes of these anomalies and take appropriate actions to improve network performance and enhance the user experience. After detecting packet losses, we usually try to understand the origin of these losses and investigate their occurrences.

Latency

Measuring and analyzing latency is a crucial aspect of network analysis, and it is typically done by examining the RTT values. The RTT value represents the time taken by a packet to travel from the source to the destination and back again. It is worth noting that the RTT curve often exhibits noise due to the inherent variability in RTT values. However, significant changes or spikes in the RTT value are particularly important as they can have a direct impact on the decisions made by the CCA.

For instance, in the case of the CUBIC algorithm, excessive latency is interpreted as a sign of congestion. When CUBIC detects high latency, it responds by reducing the sending rate. If the CCA is in the SS state during this period, the excessive latency can trigger an early exit from the SS phase. Excessive latency is detected when an important increase in its value is noticed. In other words, the algorithm looks for spikes in RTT, which suggests that the bottleneck buffer is filling up [5]. Therefore, monitoring

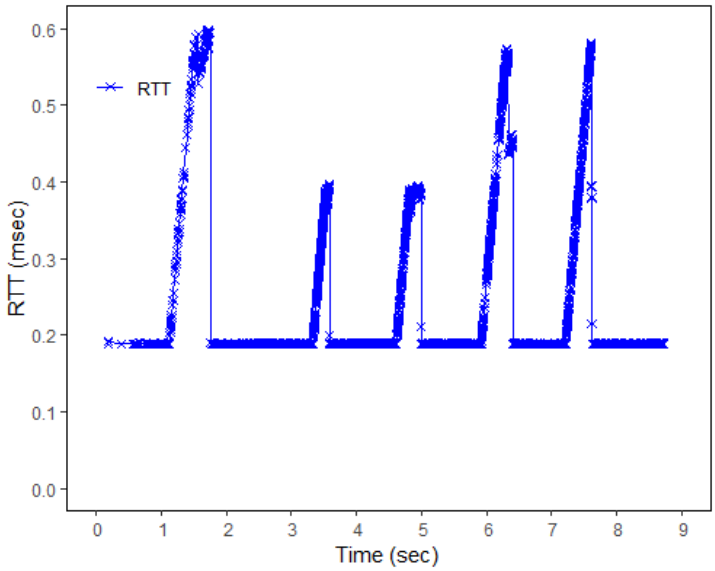

Figure 4.3 – The evolution of the RTT over time

and analyzing latency is crucial for identifying and distinguishing between different network anomaly root causes. In particular, latency variation or latency stability can help differentiate between congestion problems, transmission problems, and packet delay variation problems. More information about the use of latency to differentiate between these problems is given in Section 4.3.

Figure 4.3 provides an example of an RTT curve where noticeable peaks can be observed, indicating significant changes in latency. These peaks highlight instances where latency has experienced substantial fluctuations or variations. By carefully analyzing the RTT curve and identifying such latency changes, we can gain valuable insights into the nature and potential causes of network anomalies. In the case of Figure 4.3, we can conclude that these peaks are due to congestion events during the connection. Indeed, these spikes indicate the filling of the bottleneck and thus the saturation of the network.

Client buffer

Client buffer is another important element we look at when analyzing a packet trace. The client buffer, represented by the RWIN value, plays a significant role in determining the flow of data between the server and the client. The server continuously gets informed of the value of the available RWIN (it is communicated using the window size value field of the TCP header) from the client to ensure efficient data transmission. In our analysis, we

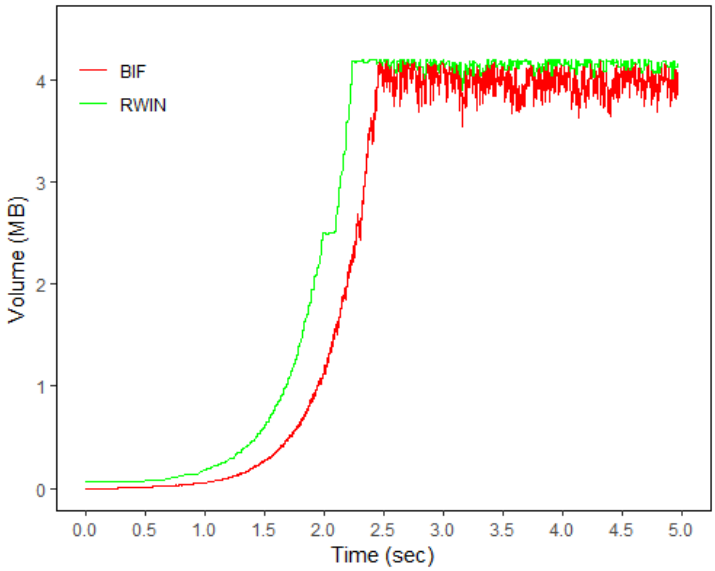

Figure 4.4 – The evolution of the RWIN and BIF over time

compare the RWIN value to the BIF value to identify any potential relationship between them. By comparing the RWIN and BIF values, we can determine if the RWIN is influencing the decrease in the BIF value observed in the packet trace. Figure 4.4 provides an example of a scenario where the RWIN value is causing a limitation in the server's ability to send packets with a higher BIF value. This limitation can be visualized in the RWIN curve, preventing the BIF value from increasing. This behavior suggests that the client's buffer is not allowing the server to transmit data at a higher rate, potentially leading to performance degradation. Identifying such cases of RWIN limitation is crucial for understanding the factors contributing to network degradation. By analyzing the relationship between the RWIN and BIF values, we can gain insights into the performance limitations imposed by the client buffer. This analysis helps in differentiating between degradation that appeared because of the operator's infrastructures or caused by the limitations in the clients' equipment.

4.3 Root cause identification

In this section, we outline our methods for identifying the root causes of anomalies in packet traces based on the behavior of the CCA. Our analysis focuses on four prevalent and frequently observed network incidents. We describe our approach to analyzing and

manually classifying each packet trace into one of these degradation types. To begin, we leverage the behavior of the CCA as a key indicator in identifying network anomalies. Our goal is to identify the common patterns and characteristics associated with each degradation type. Through a meticulous analysis of more than 500 faulty packet captures, we have selected and studied the most repeated anomalies and recognized the important characteristics required for accurate diagnosis. We have identified four major root causes of anomalies: transmission problems, congestion problems, application limitations, and latency/jitter problems.

We should note that this analysis process was heavy as it took a substantial amount of time to analyze these captures. For instance, the analysis process itself for each capture requires between 10 to 30 min (sometimes in some rare cases more time is needed), depending on the case, in order to extract the root cause of degradation. Additionally, the faulty packet traces were also manually selected and downloaded to be analyzed.

4.3.1 Normal/Ideal Case

Before delving into the curve patterns associated with specific anomaly root causes, it is important to establish a reference point by examining a normal case where the throughput of a TCP connection closely matches that of a UDP connection. In this scenario, both TCP variants can effectively utilize the full capacity of the bottleneck.

A key characteristic of a normal connection is the consistent BIF value that remains close to the value obtained when the CCA exits the SS state. This indicates an accurate estimation of the bottleneck capacity during the SS phase. The stability of the BIF value throughout the connection reveals the optimal utilization of the available bandwidth. In addition to the stable BIF value, a good connection is characterized by minimal packet losses and low latency. Packet losses should be exceptional, indicating reliable data transmission, while latency should be kept at a minimum, ensuring efficient communication.

Figure 4.5 provides an example of a normal connection, illustrating the aforementioned characteristics. The BIF curve remains relatively steady after the SS exit, indicating a precise estimation of the bottleneck capacity. Furthermore, the packet loss rate is minimal, and the latency remains consistently low during the connection. By establishing this normal case as a benchmark, we can better understand the deviations and patterns associated with different anomaly root causes.

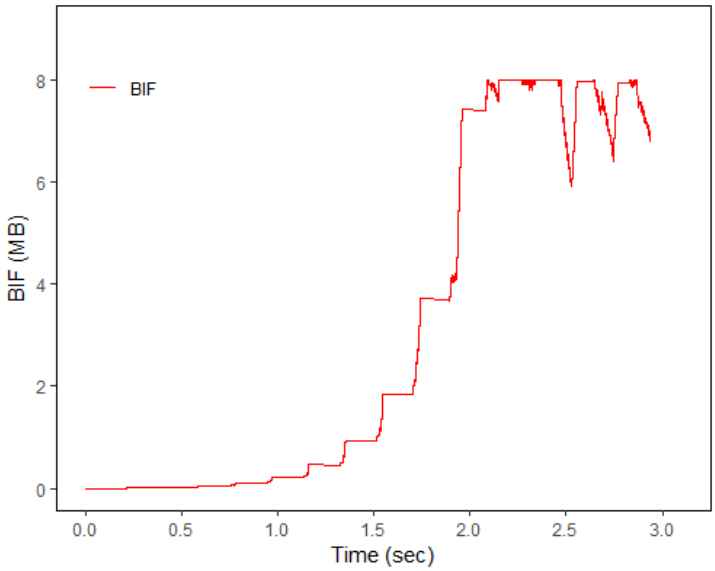

Figure 4.5 – BIF evolution over time for a Case of a packet capture with good QoE

4.3.2 Transmission Loss

Transmission losses refer to the occurrence of packet drops in a network without any associated congestion. These losses predominantly affect RAN and can be attributed to factors such as radars, weather conditions or interference within the RAN environment. By analyzing the SEQ of packets and their corresponding ACK, we can identify instances of packet loss. Transmission loss is characterized by individual and isolated packet losses where specific packets are dropped without a congestion signal. In Figure 4.6, we observe two instances of individual packets retransmission occurring at 13.61s and 13.82s, indicating the occurrence of two packet losses. These losses are evident from the decrease in SEQ values due to packet retransmission.

While transmission losses manifest as isolated occurrences, they can have a harmful impact, particularly if repeated frequently. If these isolated losses take place at the beginning of a connection, they can cause the CCA to prematurely exit the SS state. This happens due to the misinterpretation of these losses by the CCA. Indeed, in this case, it deals with the losses as a sign of congestion, leading to an underestimation of the network bottleneck. As a result, the available bandwidth is not fully utilized and the overall performance of the connection is damaged.

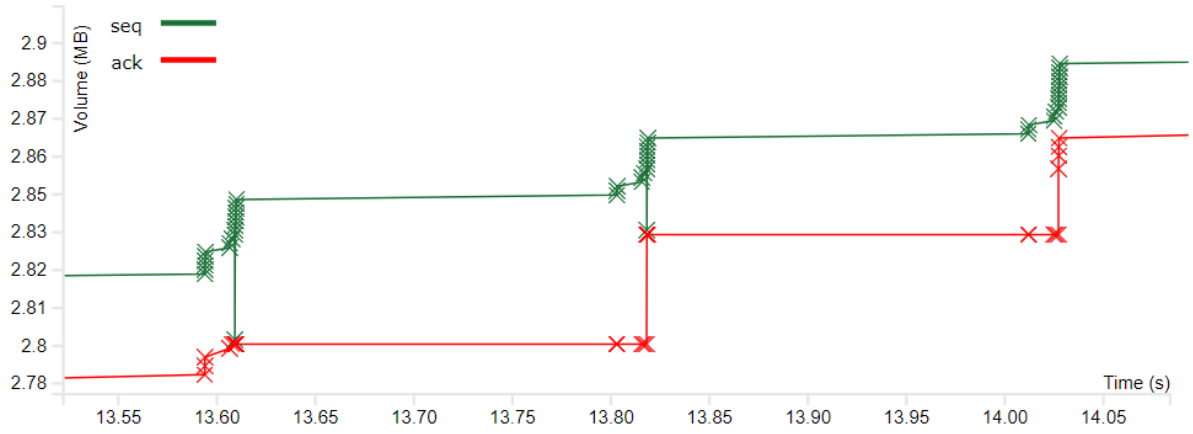

Figure 4.6 – Transmission loss detection using SEQ and ACK evolution during time

4.3.3 Congestion Loss

Network congestion can occur when the traffic throughput being transmitted exceeds the capacity of the network. This phenomenon can affect both wireless and wireline networks and is primarily influenced by factors such as link capacity and buffer sizes. Unlike transmission losses, which occur without congestion, detecting congestion involves identifying a burst of packet losses.

In Figure 4.7, we can observe a burst of packet losses occurring between 1.88s and 1.94s. This burst is indicative of congestion in the network.

These lost packets are accompanied by a notable increase in the RTT, as depicted in Figure 4.8. At 1.90s, the RTT value starts to rise from 200ms and reaches 625ms by 2.30s. The presence of a burst of packet losses, coupled with a significant increase in RTT, provides clear evidence of network congestion. The burst of losses means that the network is unable to handle the incoming traffic at the current rate, leading to packet drops. The subsequent rise in RTT is a consequence of increased queuing delays caused by congestion in the network. Detecting and understanding congestion is crucial for network troubleshooting, as it allows for the identification of capacity-related issues and helps in implementing appropriate congestion control mechanisms. By analyzing the patterns of packet losses and the associated changes in RTT, network experts can effectively diagnose congestion problems and take necessary measures to mitigate their impact on network performance.

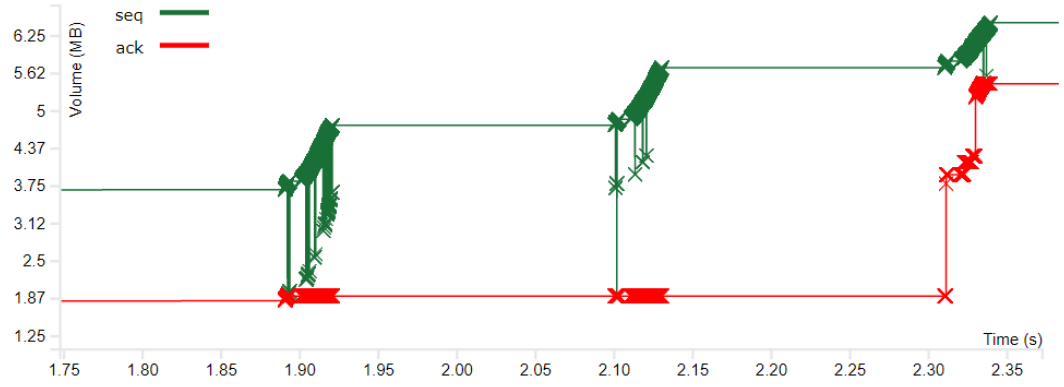

Figure 4.7 – Congestion detection using SEQ and ACK evolution during time

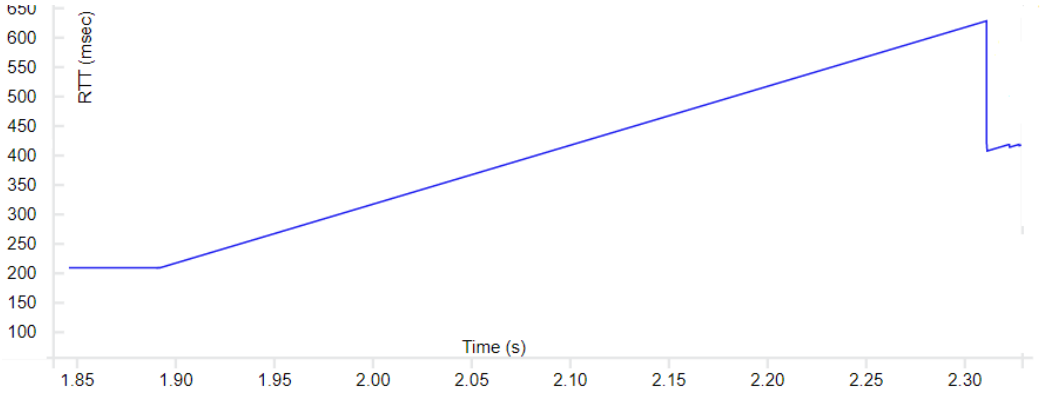

Figure 4.8 – RTT evolution during time

4.3.4 Jitter

Packet delay variation, commonly known as jitter, is a network issue that typically impacts wireless users. Jitter can significantly degrade performance by causing the CCA to prematurely exit the SS state phase, mistakenly interpreting jitter as a sign of congestion. To determine if jitter is the underlying cause of poor performance, we must first identify the SS exit time to determine the specific cause of this exit. This is achieved by analyzing the behavior of the CCA and identifying the point at which it transitions out of the SS phase (end of the exponential growth). Once the SS exit time is determined, we can proceed to correlate it with other relevant time series data, such as SEQ, ACK, and RTT. During this analysis, we specifically look for indications of packet loss or an increase in RTT. If no packet loss is detected, but a sudden high variance in RTT is observed, it suggests that the CCA exited SS due to jitter. This is because the jitter introduces

significant variations in the packet arrival times, leading the CCA to incorrectly perceive it as congestion and triggering an early exit from SS.

Figure 4.9 illustrates an example where the CCA exits SS at 50000 bytes (around 0.7s), while the actual bottleneck capacity, derived from the end of the transmission, is around 1.6×10^6 bytes, as shown in Figure 4.10. This underestimation of the bottleneck capacity results in a poor connection throughput.

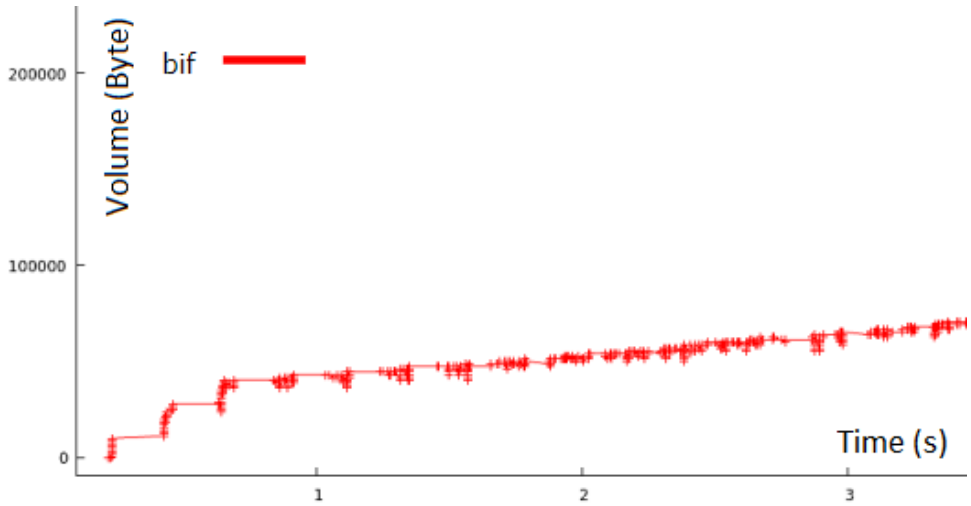

Figure 4.9 – BIF evolution during time focusing on the beginning of the connection

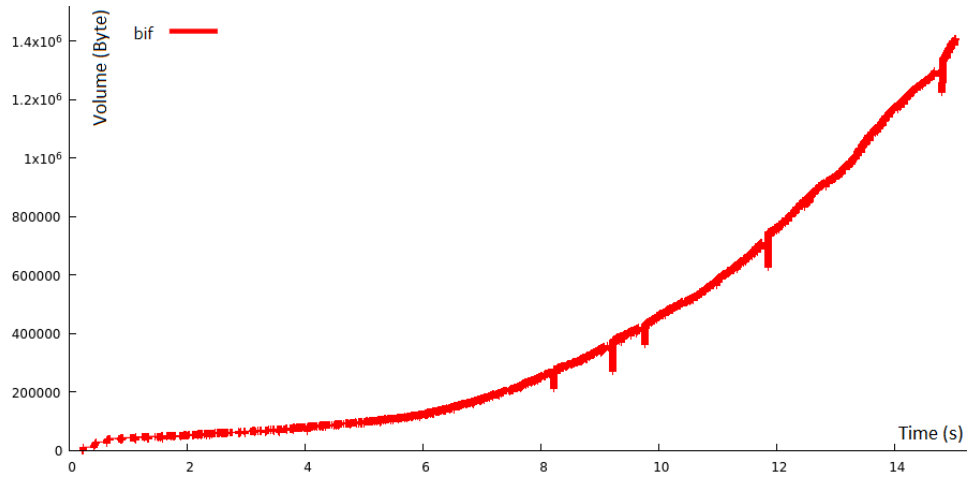

Figure 4.10 – BIF evolution over time for the same capture as in Figure 4.9 focusing on the whole connection

4.3.5 Application limited

In addition to network issues, the performance of a TCP transfer can also be impacted by the endpoints' ability to send or receive data at a sufficient rate. In this context, the focus is on the case of a "slow receiver", where the receiving endpoint, typically a client device, becomes the limiting factor in the data transfer. While servers are typically designed to handle normal traffic without being overwhelmed, client devices, often battery-powered with limited resources, can become the bottleneck in communication. Therefore, we specifically examine the scenario where the client device acts as a slow receiver.

The "slow receiver" situation is characterized by the RWIN value imposing a limitation on the BIF value, as can be noticed in Figure 4.11. The graph shows the exponential growth of the BIF during the Slow Start phase, reaching the RWIN's limitation at 1.4s. This indicates that the receiver's buffer has reached its maximum capacity and cannot grow beyond that size. In consequence, the sending rate from the source cannot increase further without overwhelming the receiver's capabilities.

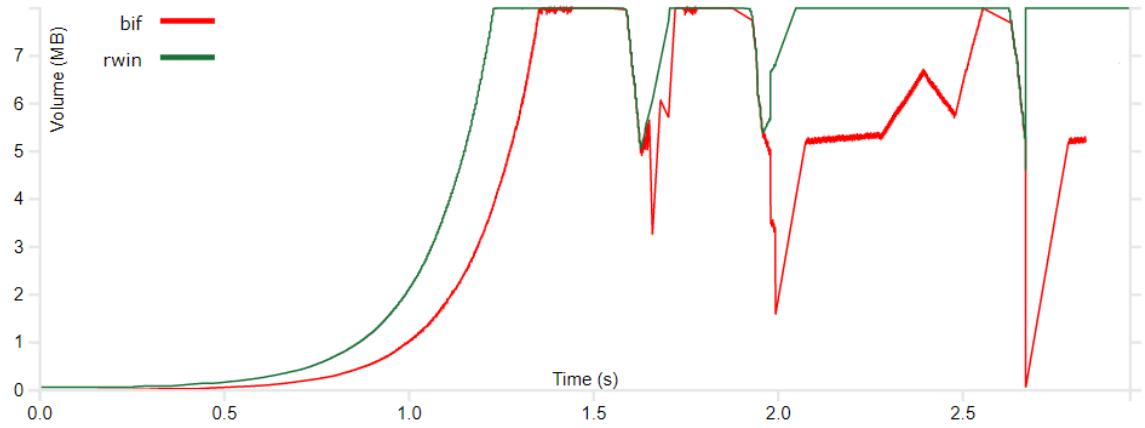

Figure 4.11 – Application limited detection using RWIN & BIF evolution

4.4 Conclusion

The rapid growth of Internet traffic has presented new challenges in detecting and resolving network degradation issues. Identifying the root cause of degradation is a complex and time-consuming task, especially with the increasing scale and complexity of modern networks. However, in order to minimize the impact on customer experience, it is essential to detect and address these issues promptly.

Root Cause of Degradation	**Main Characteristics**
Transmission problem	Isolated losses with no remarkable increase in the RTT value
Congestion problem	Burst of losses with a significant increase in the RTT value
Jitter	Exiting the SS with no loss or any increase in the RTT value + BIF value at the end of the connection much greater than its value at exiting the SS state
Application limited	BIF value blocked by the RWIN

Table 4.1 – Root Causes of Degradations and Their Main Characteristics

Our study focused on manually analyzing packet traces to identify and classify the root causes of anomalies affecting clients' QoE in network connections. Our analysis was based on the behavior of the CCA. By analyzing packet captures with bad performance, we identified common patterns and important characteristics necessary for diagnostic purposes. A summary of each root cause of degradation with its main characteristics is shown in Table 4.1. From the manual analysis, we recognize the importance of the detection of the SS state as a key element for network troubleshooting in identifying the anomaly root cause. In the next chapter, we focus on automatically detecting the SS state in order to accelerate the troubleshooting process and rapidly detect the degradation root cause.

Part II

Troubleshooting enhancement and BBR identification with automated Slow-Start State Detection

Chapter 5

New Packet Traces Visualization approach for Slow-Start automated detection

The contributions of this chapter have been published in the conference ICIN [49] and the journal annals of telecommunications [48]:

— Ziad Tlaiss, Isabelle Hamchaoui, Isabel Amigo, Alexandre Ferrieux, Sandrine Vaton. Troubleshooting Enhancement with Automated Slow-Start Detection, *26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2023

— Ziad Tlaiss, Alexandre Ferrieux, Isabel Amigo, Isabelle Hamchaoui, Sandrine Vaton. Automated Slow-Start Detection for Anomaly Root Cause Analysis and BBR Identification, *Annals of Telecommunications*, 2023

5.1 Introduction

When analyzing a packet trace, the SS exit time is a key element for troubleshooting experts. While many state transitions suggest specific root causes, SS exit time has a particular significance. Indeed, in SS, the source estimates the value of the path's capacity by exponentially increasing its rate until a congestion signal is received, then it exits the SS state to enter a new phase with a much lower growth of rate.

Should the SS overestimate the bottleneck, then the source will exceed the bottleneck capacity and thus experience multiple packet losses, from which recovery is painful. On

the contrary, if it underestimates the bottleneck capacity and triggers an early SS exit, the source will underuse the available bandwidth and possibly experience poor throughput. This is typically the case in the presence of transmission loss, or excessive jitter related to radio mobile access - even underloaded. In both cases, limiting the rate will lead to a bad customer experience, without any benefit regarding a non-existent congestion. The SS state duration is thus a key indicator for the diagnosis of faults; this indicator is estimated empirically by human experts today, which is time-consuming and a cumbersome task with large error risks. As current troubleshooting is mainly based on human diagnosis, automation is certainly a key element for dealing with the data deluge collected via such a dense fleet of active and passive packet captures.

In this Chapter, we design a method and develop a tool to automatically detect the SS exit time on collected traces. For this purpose, we introduce a novel representation, that is the BIF versus SEQ representation that replace the BIF over time representation. We use this representation in order to identify the last packet in the SS state of the CCA, by using a relation between the sequence number values and the bytes-in-flight values that is true only during the SS phase. Due to the similarity of QUIC and TCP CCAs, this method applies to both. However, as it requires accessing transport headers information, it can be applied to QUIC traffic only with active measurements (implying we can decrypt the traffic) while it could be used on TCP traffic with both active and passive measurements. We also introduce how our SS detection method could be used as a powerful tool to easily discriminate between network typical fault types.

5.2 Visual CCA states identification

As explained in section 3.4, the FSM state series, and particularly the SS exit time gives crucial insight into degradation root causes. To get hold of these state series, the first idea that comes to mind in order to identify the CCA states is direct introspection in the sender stack. Unfortunately, this introspection requires cooperation from the sender's server, which is rather impractical, as many servers belong to third-party Internet content providers, often reluctant to open their infrastructures. As a consequence, sticking to measurements from active and passive probes is still the operators' best choice to build these state series.

In this context, troubleshooting experts are used to perform visual analysis of the BIF against time to detect the end of the exponential growth, namely the SS exit time.

This method is highly time-consuming and inaccurate. We can see in Figure 5.1 an exponential increase of the BIF against time until $t = 1.3$ sec, a telltale sign of the SS phase. Root cause analysis is then completed thanks to the SEQ against time graph (Figure 5.2) showing bursts of packet retransmission at this very same time, a typical effect of congestion loss [46]. The SS exit is then a legitimate reaction of the CCA to reaching the actual bottleneck.

In this context, experts often rely on visual analysis of the BIF over time to identify the FSM state transition, particularly the SS exit time, which marks the end of exponential growth. Figure 5.1 illustrates the typical pattern of an exponential increase in the BIF over time until $t = 1.3$ sec, indicating the occurrence of the SS phase. This exponential growth is a characteristic behavior during the initial phase of data transmission. To further investigate and detect the end of the SS state, experts also examine the SEQ over time graph, as shown in Figure 5.2. This graph reveals how packet bursts doubled after each RTT until about $t = 1.25$ sec, at this time packet number stopped doubling and we can notice packet retransmissions occurred at around the same time as the SS exit time identified in the BIF vs. time curve. This observation suggests that the SS exit is a legitimate response of the CCA to reaching the actual bottleneck in the network.

While visual analysis of BIF and SEQ graphs over time has been a traditional approach for detecting the SS exit time, it is important to note that it can be time-consuming and subject to interpretation biases (human error). To overcome these limitations, researchers are exploring automated methods and algorithms that can provide faster and more accurate detection of the SS exit time and other relevant metrics [59], [57], [2]. These automated approaches aim to improve the efficiency and reliability of the SS state detection, allowing for more precise identification of degradation root causes in communication systems, identifying and inferring the CCA variant in the network, and helping study fairness between different CCA variants.

5.3 State of the art on CCA states identification

In this section, we will discuss the relevant works that focus on the detection of the SS state and troubleshooting tools. In consequence, we identify two areas that are directly connected to our work: identification of the CCA and network troubleshooting tools.

Hagos et al. [23] use machine learning approaches to recognize loss-based TCP CCAs and infer the congestion window within passively collected traffic at mid-point. Although

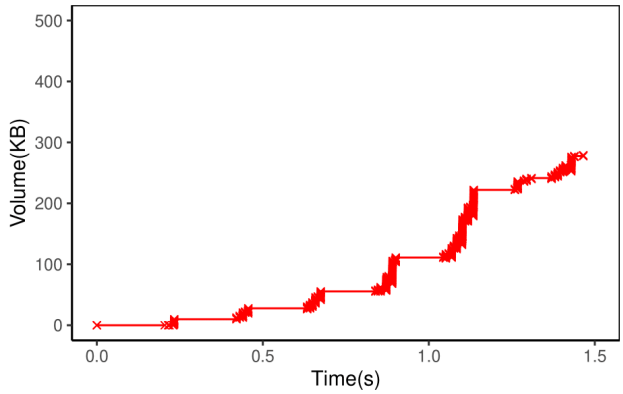

Figure 5.1 – BIF against time during SS state for a CUBIC packet capture

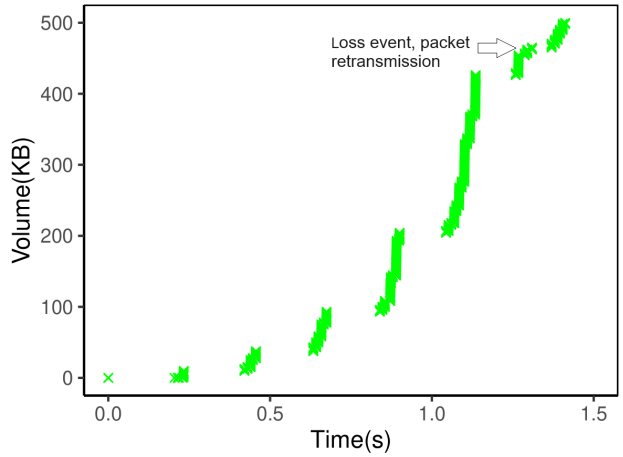

Figure 5.2 – SEQ against time during SS state for the same capture in Figure 5.1

estimating the cwnd can be useful for network operators to troubleshoot their network, it does not cover no-loss based CCAs such as BBR. Our work differs from theirs as our method focuses on the application of CCA SS state detection in order to detect network root causes of anomalies and identify the type of CCA in use. Furthermore, our method could be applied to all types of CCA, loss-based or not.

Padhye et al. [39] developed the TCP Behavior Inference Tool (TBIT), which performs active measurements to infer various TCP behaviors such as the initial window and congestion window (cwnd) of a remote Web server. TBIT can also detect which of the following CCAs is running on a Web server: Reno, New Reno, Reno Plus, or Tahoe.

Yang et al. [57] proposed an active CCA identification approach that uses a random forest algorithm to classify the CCA variants of a Web server. The classification is based on two features: the multiplicative decrease parameter applied when a loss is detected during the SS state and the window growth function driving the congestion avoidance state. The authors were able to identify several famous CCAs, such as NewReno, BIC, VEGAS, and CUBIC.

Jaiswal et al. [27] introduce a passive measurement methodology to infer the cwnd and round-trip-time. They build a replica of the CCA state for each TCP connection at midpoint. This replica updates its estimate of the cwnd based on the observed acknowledgments that could change the CCA state. They use those estimates to recognize three of the TCP flavors: Reno, NewReno, and Tahoe. Even if [27] are interested in initial cwnd, SS state, and congestion avoidance states to identify the CCA types, they do not try to accurately locate state transitions.

Kato et al. [51] use unidirectional packet traces to characterize TCP CCAs. They define a new metric that is seen as being proportional to cwnd size and apply curve fitting to recognize the CCA. In the continuity of their work Kato et al. [31] identify TCP CCAs using a sequence number vs packet arrival time representation.

Zhang et al. [59] analyze TCP passive packet captures and investigate CCA mechanisms to understand the origins of limitations in the transmission rates of flows by grouping packets into flights (they consider each packet burst as a flight) using a round-trip-time estimator.

Mishra et al. [2] developed Gordon, an active tool that measures the congestion window size and identifies TCP CCA variants among websites. Gordon measures the cwnd and then analyzes the reaction of the TCP variants to packet losses to classify them. In particular, depending on the decrease factor after a loss or/and the increase factor during the congestion avoidance state, the TCP variant is identified. To do this they do not rely on common active measurements, but manipulate the client to force the server to react against several scenarios, generating a considerable amount of traffic. For rate-based CCAs such as BBR, which does not change its cwnd after a loss, the no-loss reaction is used to determine the CCA as BBR or unknown.

Apart from works related to identifying CCA, some researchers proposed some automated troubleshooting tools that use packet captures to diagnose the networks and detect some degradations. Guo et al. [13] developed pingmesh, a tool for large-scale data center network latency measurement and analysis to track network latency issues. Zhu et

al. [58] proposed Everflow, a packet-level tracing and analysis tool. While [13] and [58] are 2 troubleshooting solutions, their scope is limited to a specific set of equipment-level performance metrics; this makes sense from a "repairman"'s point of view, to whom exonerating a specific router from guilt is critical, but is not sufficient to address an end-to-end scenario, where the offending connection spans continents and (possibly non-cooperative) actors. In our work, we aim to get a broader view of the issue at hand, by providing a cause-agnostic observable, the SS exit time, as input to further investigations.

In summary, [59], [57] and [2] show interest in the detection of the SS state. However, [59]'s method tracks the SS state in the first flights based on explicit segmentation, which does not work consistently in real life, e.g. when ACKs are not "bursty". On the other hand, [57] and [2] only take losses into consideration in the detection of the SS state.

On the other hand, aside [2], none of the previously mentioned works addresses the identification of BBR on its own by analyzing a packet capture, nor do they compare CUBIC and BBR traffic. Our method differs from these works in that it relies only on the analysis of bidirectional packet traces obtained from classical TCP downloads to identify BBR traffic. Unlike [39] and [2], we don't need to generate multiple traffic and control the cwnd by holding the acknowledgments which can be laborious, as our method consists of analyzing a simple packet trace so one single download/packet capture is sufficient. We also differ from [57] as we do not use a heavyweight machine learning algorithm: indeed, our parameter space is extremely small (two variables), thanks to a novel, well-motivated feature extraction step that taps into the core design differences between the CCAs. And similarly to the aforementioned works, we do not need to process our data online, as our objective is to allow operators to quantify the amount of each CCA present on their networks.

5.4 Challenges towards automation

Network troubleshooting experts often rely on manual detection of CCA states, particularly the SS state, to identify network anomalies and troubleshoot performance issues. However, as network traffic continues to increase in volume and complexity, a manual detection process becomes increasingly difficult and time-consuming. As a result, there is a growing need for automated methods that can accurately detect and identify CCA states, particularly SS, in a timely and efficient manner. In this section, we will explore the challenges associated with automating the detection of SS state in CCA.

5.4.1 Noise

Network traffic can be affected by some intrinsic factors such as network topology and routing, which can cause variations in packet delay, loss, and jitter, which significantly impact network behavior. These factors introduce variations in packet delay, loss, and jitter, making it challenging to develop accurate models and predictions of network performance.

While manual analysis by human experts can often handle the noise and irregularities present in the data, automating such procedures can be difficult, for example, some filtering techniques may be needed. Automated methods need to account for and adapt to the complexities introduced by the aforementioned factors. For instance, in the context of our study, the exponential growth patterns depicted in Figures 5.1 and 5.2 might require smoothing techniques to properly identify and recognize the growth trends. This smoothing process ensures that the underlying exponential behavior is more distinguishable from the noise present in the data.

5.4.2 Non-stationarity

Network traffic is constantly changing over time, and the characteristics of network traffic can vary significantly between different periods. For example, a network may experience increased traffic during certain times of day or during peak usage periods, which can affect the performance of the network and the behavior of the CCA. On the other hand, the performance of a CCA can also be affected by the application running on top of it. Different applications have different traffic patterns and requirements, and the CCA must be able to adapt to these different conditions. This can make it difficult to automate the detection of the SS state, as the behavior of the algorithm may vary depending on the application being used. Additionally, network conditions like packet losses deteriorate the visibility and raise the non-stationarity. An example of this non-stationarity can also be observed in a single packet capture as can be noticed in Figure 5.3 and Figure 5.4. Moreover, in a broader sense, the graphs representing network behavior may need to be segmented into homogeneous regimes or segments, as in the case of Figure 5.3, before any pattern recognition or analysis can be applied.

Figure 5.3 shows the evolution of BIF over time, while Figure 5.4 represents the arrival times of the captured packets. The latter focuses on the so-called "on/off pattern" of packet arrival times which reflects the basic congestion window mechanism, waiting for

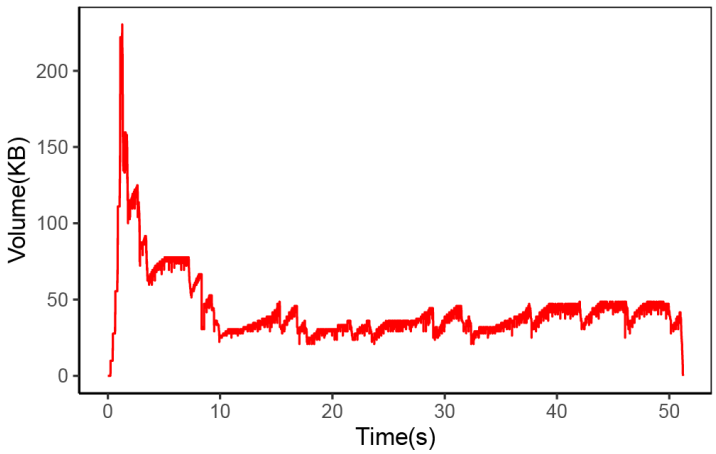

Figure 5.3 – BIF against time showing the non stationarity and noise on BIF values

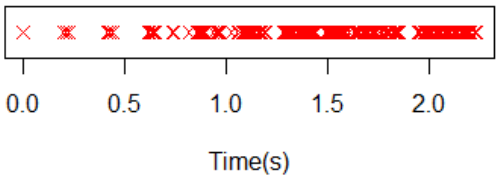

Figure 5.4 – Packets arrival times - blurring of on/off pattern over time

ACKs before sending a new burst of packets. However, while this on/off pattern can be detected at the beginning of the connection (from 0 sec to 0.8 sec), it blurs over time, due to TCP's (intentional) tendency towards "ACK clocking" [6]. In fact, TCP employs a self-clocking scheme that times the sending of packets. The data packets are sent in a burst when the returning acknowledgment packets are received. This self-clocking scheme (also known as ack-clocking) is deemed a key factor in the burstiness of TCP traffic. After reaching the bottleneck capacity, the packet starts to fill the buffer of the slowest link as shown in Figure 5.5. Consequently, the transmitted packets are spaced out, causing the corresponding ACKs to also be distributed more evenly. As a result, the new packets sent subsequently exhibit reduced burstiness.

In conclusion, because of all of the above-mentioned complexity, the automation of the detection of the SS state and of the troubleshooting process in general is not trivial.

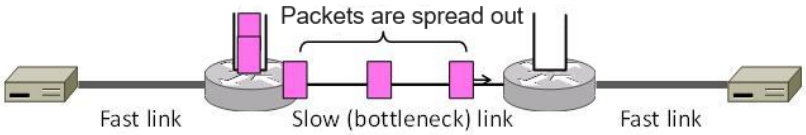

Figure 5.5 – ACK clocking impact on bursts

5.4.3 Exponential regression approach

All previously mentioned challenges invalidate different methods that we have considered and tried when working on the automated detection of the SS state. The most promising approach yielding better initial results was "exponential regression", i.e. fitting the BIF-against-time with an exponential. The objective is to detect the exponential part in the SS state that starts from the beginning of a connection until exiting the SS state.

Exponential growth is a phenomenon that occurs when a quantity increases at a constant percentage rate over time. This growth pattern is commonly observed in a wide range of scientific fields. When analyzing data that exhibit exponential growth, it is often useful to employ an exponential regression analysis, which can provide insights into the underlying growth rate and help make predictions about future trends. Exponential regression is a statistical method that involves fitting a curve to a set of data points using an exponential function. The exponential function takes the form $y = ae^{(bx)}$, where y represents the dependent variable, x represents the independent variable, a is the initial value of y, and b is the growth rate. The goal of exponential regression is to determine the values of a and b that best fit the data as shown in Figure 5.6.

To perform exponential regression, we plotted the data on a logarithmic scale, which helped us to linearize the relationship between the variables. We used a multiple exponential model, which involves fitting multiple exponential curves to the BIF data and selecting the one that minimizes the error between the curve and the data. We select the best curve by calculating the sum of the squared differences between the data and each curve, and we choose the curve that yields the smallest value as it is shown in Figure 5.6. The use of multiple exponential curves allows us to choose the curve that best fits the data, and the process of minimizing the error helps to ensure that the chosen curve provides an accurate representation of the growth pattern.

Although the exponential regression approach was promising in the majority of the cases, however, it turns out that in the case of very early SS exits (within 2 or 3 RTT), the

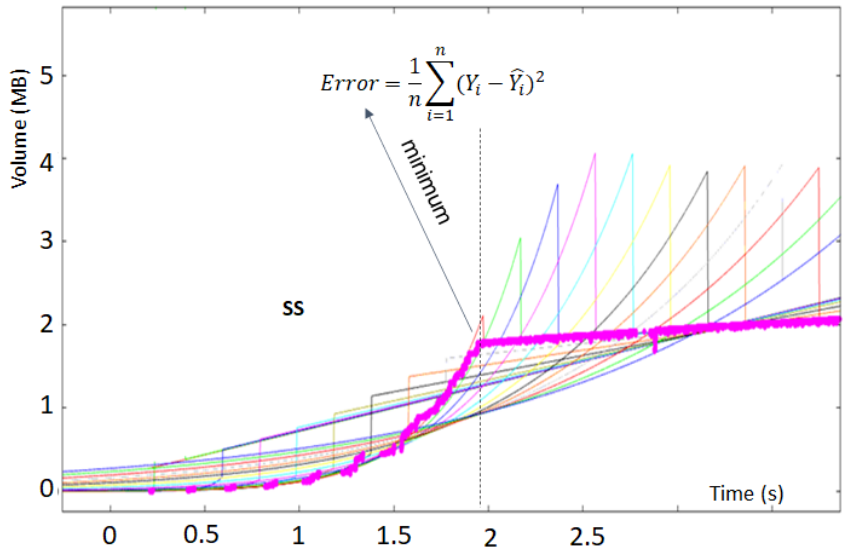

Figure 5.6 – Optimizing BIF Curve Fitting: Selecting the Best Exponential Fit

exponential part is dwarfed by the subsequent evolution, making it impossible to detect the exponential part reliably. This is unfortunate, as our objective is to use SS state detection to troubleshoot networks, where the most frequent cases of bad performance are correlated with a premature exit from SS. As it turns out, this fundamental problem is resolved using the new representation that we introduce in the next section.

5.5 Automated Slow-Start detection method

As the use of CCAs becomes increasingly prevalent in modern networking protocols, it is essential to have accurate methods for detecting when a network flow transitions between different states, specifically exiting the SS state. In this section, we introduce a novel approach for automatically detecting the exit from the SS state using a representation based on BIF versus SEQ. We show how this representation can be used to accurately identify the last packet in the SS state of a CCA by leveraging the unique relationship between SEQ values and BIF values that is only true during the SS phase.

5.5.1 Bytes-in-Flight vs sequence number: a timeless representation

In light of our troubleshooting experience, it turns out that the main hurdle to automation lies in the on/off patterns of the source emissions. These patterns introduce significant variation in the inter-packet spacing, making it difficult to identify and distinguish different phases of the CCA. To overcome this challenge in a natural way without any loss of information we decided to switch to a timeless representation. To this effect, we chose to represent BIF as a function of SEQ as shown in Figure 5.7. In essence, we replace the time axis with the sequence number progression, this naturally wipes out all burst, silence, or RTT variation effects that are commonly observed in time-domain representations. The resulting representation allows us to focus on the essential correlations between significant indicators of CCA dynamics, enabling more accurate and efficient detection of the exit from SS and other key events in the congestion control process. This approach has the potential to significantly improve and open the way for more effective automation of network troubleshooting and optimization.

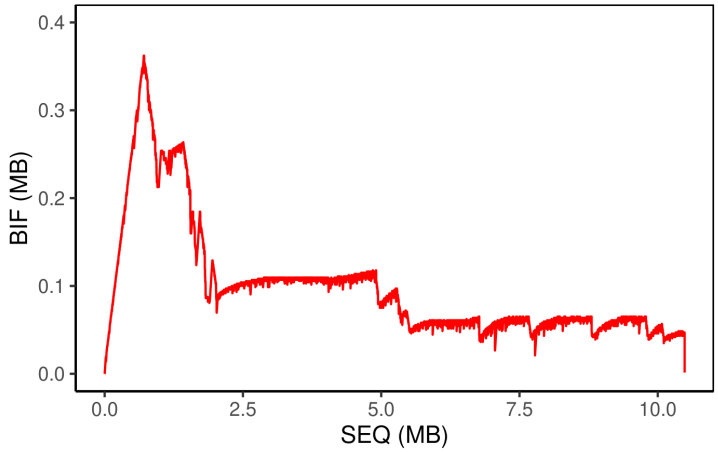

Figure 5.7 – BIF against SEQ: new representation to detect the SS state exit time

5.5.2 BIF vs SEQ characterization during Slow-Start

A few basic properties of the BIF vs SEQ representation can easily be derived analytically. To begin with, the shape of the graph is readily predictable during two phases:

(a) During burst emissions: in the absence of any acknowledgment, during this phase,

each sent packet increments both SEQ and BIF by an equal value, which is the segment's length.

(b) During the reception of burst acknowledgments: assuming all packets previously sent were received, the BIF quickly drops back to zero.

As a result of these two phases, every round-trip time, the graph is expected to display a triangular shape made of a slope 1 due to phase (a), followed by the vertical drop described in (b), as depicted in Figure 5.8.

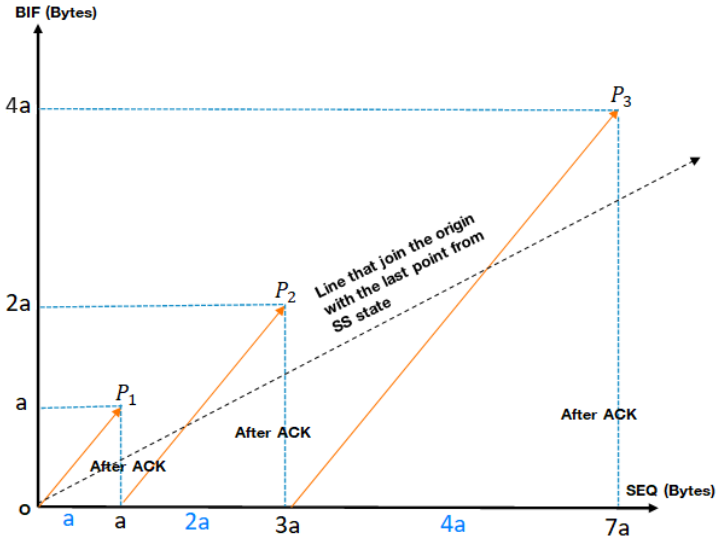

Figure 5.8 – Theoretical representation of the BIF vs SEQ evolution

Furthermore, in an ideal SS state, the vertical extent of this triangular shape, which represents the cwnd, is expected to double every RTT. Thus, the graph should display a *fractal* series of triangles, each one being twice the size of the one before. The position of the highest SEQ point and highest BIF point in the graph, after n round-trip-times, is thus expected to be:

$$SEQ = \sum_{i=0}^{n-1} a \times 2^i = a \times (2^n - 1) \tag{5.1}$$

$$BIF = a \times 2^{n-1} \tag{5.2}$$

The slope of the line from the origin to this point is thus

$$BIF/SEQ = \frac{2^{n-1}}{2^n - 1} \tag{5.3}$$

Hence its limit is

$$\lim_{n \to +\infty} BIF/SEQ = 1/2 \tag{5.4}$$

It can further be seen that this asymptote $y = x/2$ is in fact "approached from above", as the top of each triangle satisfies: $BIF/SEQ = \frac{2^{n-1}}{2^n - 1} > 1/2$

However, as soon as the SS state is exited, the exponential growth of the BIF stops, and no further point can stand above the $y = x/2$ line.
This yields a very simple and practical criterion: the SS exit occurs immediately after the last point satisfying: $BIF \geq \frac{SEQ}{2}$.
It should be stressed that the power of this method lies in its simplicity: no regression or filtering is needed, and a simple linear inequality suffices, once we are in the appropriate representation space.

5.5.3 Slow-Start exit time detection

While the critical state transition event is well characterized by the above criterion, some attention is due to properly interpret the earlier features of the representation. During the SS phase, as mentioned before, local slopes are typically 1, with a series of abrupt drops. As a result, the graph keeps crossing the asymptote, thus, a local decision is not appropriate, as it would readily generate false positives. Fortunately, the global criterion of the last point above the asymptote is more robust. This is fundamentally linked to the fact that after exiting SS, the CCA essentially takes very careful steps to refrain from going too fast, and by definition will never "catch up" to the exponential regime. The asymptote is never to be crossed again. Figure 5.9 is an example of our slope 1/2 method application. We can see the BIF vs SEQ curve slightly exceeding the $y = \frac{1}{2}x$ line until it abruptly drifts below, marking the instant when the CCA has exited the SS state.

5.6 Method evaluation

In this section we assess the accuracy of our method by comparing the SS exit time we obtain, against a "groundtruth" which we define as the CCA state transition time recorded in the server stack logs.

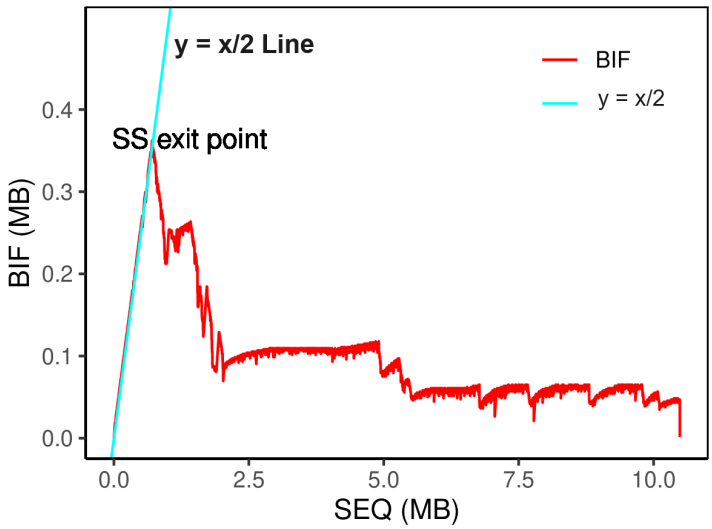

Figure 5.9 – Automatically detecting SS state exit time with slope 1/2 method

5.6.1 Testbed

To evaluate the effectiveness of our proposed method for detecting the exit from SS in the Cubic and BBR CCA, we conducted a series of experiments on a server that is accessed by multiple active probes through the public Internet as presented in Figure 5.10. To generate CCA logs, we instrumented the Cubic and BBR stacks on the server and then performed active measurements by executing many downloads using both CCAs from our probes. We then extracted the groundtruth SS exit times from these logs, which were used as the basis for our evaluation.

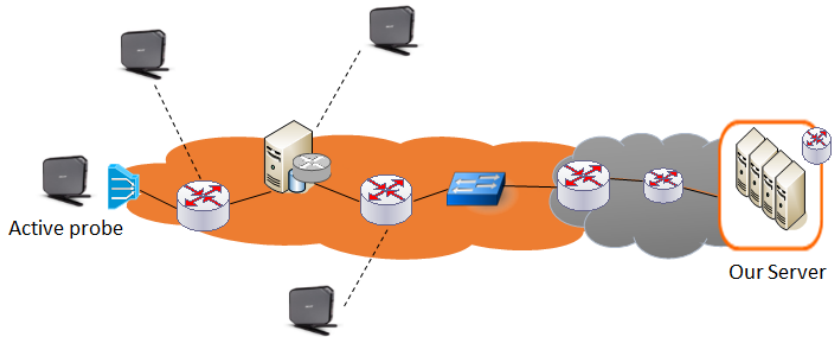

Figure 5.10 – Testbed illustration

5.6.2 Characterization of packet captures

In order to ensure that our experimental results are representative of real-world network conditions, we deployed our probes across four different Orange affiliates located in different countries. These countries were chosen to provide a diverse range of average RTT and packet loss levels. Specifically, the average RTT ranged from 20ms to 200ms, reflecting the typical range of values observed in different regions of the world. Additionally, we performed our downloads in each of these countries at different times of day, including both peak and off-peak hours, in order to capture variations in network congestion levels.

This approach allowed us to obtain a comprehensive view of the performance and behavior of the CUBIC and BBR CCAs under a range of realistic network conditions. By conducting our experiments across multiple countries and time periods, we were able to gather a rich dataset that reflects the diversity and complexity of real-world network environments. This is particularly important in the context of congestion control, where the performance of different CCAs can be highly dependent on the specific network conditions and traffic patterns.

5.6.3 Groundtruth extraction and error calculation

To evaluate the effectiveness of our proposed method for detecting the exit from SS in the Cubic and BBR CCAs, we needed to establish a groundtruth for the SS exit times. To extract this groundtruth from the resulting CCA logs, we identified the packets that corresponded to the SS phase from these logs by investigating the state transition communicated by the CCA and we extracted the arrival time of the last packet that signaled the end of this phase as the groundtruth SS exit times.

Investigating the CCA logs for CUBIC and extracting the groundtruth SS exit times was highly accurate. However, we encountered a significant challenge in the evaluation process once we focused on extracting BBR groundtruth due to the nature of the Linux BBR implementation that we used. In particular, we found that the SS phase, which is typically identified through a binary search period, was very difficult to detect, especially in the presence of massive packet loss. This is because the SS phase is not a single state, but rather a region in parameter space that is difficult to identify precisely, e.g. Startup-Drain phase (see Section 3.3.2). As a result, we had to carefully analyze the instrumented kernel's CCA logs (an example of the logs is shown in Figure 5.11) and use additional post-processing to identify the SS exit times accurately.

```
...
0.153420                kprobe_bbr_main: (bbr_main+0x0/0xa4e
[tcp_bbr]) state=0 full=0 since=0 rttcnt=8 rate=106199836 spo=0x28c8
dpo=0x5000 seq=2290200
...
0.170296                kprobe_bbr_main: (bbr_main+0x0/0xa4e
[tcp_bbr]) state=0 full=0 since=0 rttcnt=9 rate=176839582 spo=0x28c8
dpo=0x5000 seq=3267352
...
0.273222                kprobe_bbr_main: (bbr_main+0x0/0xa4e
[tcp_bbr]) state=0 full=0 since=2 rttcnt=11 rate=177514600 spo=0x28c8
dpo=0x5000 seq=3814168
0.273293                kprobe_bbr_main: (bbr_main+0x0/0xa4e
[tcp_bbr]) state=2 full=1 since=3 rttcnt=12 rate=61493555 spo=0x28c8
dpo=0x5000 seq=3816580
...
```

Figure 5.11 – BBR logs illustration

Detecting the groundtruth SS exit times for BBR CCA also presented another challenge due to the nature of the algorithm itself [7]. Specifically, we observed that BBR waits for three round-trip times (RTT) before announcing that it has exited the SS state. To account for this delay, we subtracted three times the value of RTT that we obtained during the connection's initial handshake from the SS exit time obtained through BBR CCA logs. However, we found that this calculation introduced inaccuracies in the groundtruth SS exit time, especially in cases where the RTT values increased during the connection.

To calculate the error between our method and the groundtruth, we measured the difference between the SS exit time obtained through our method and the SS exit time obtained from the groundtruth data. To better visualize the results, we normalized the time difference by the RTT value for each capture. Thus, the error is computed as:

$$error = \frac{t_{SS_{OurMethod}} - t_{SS_{GroundTruth}}}{RTT}$$

The resulting error is then represented in Figures 5.12 and 5.13, in RTT units. In the next section, we analyze the two curves and discuss our results.

5.6.4 Method evaluation with CUBIC traffic

In this section, we analyze the performance of our method for predicting the exit from SS state for the CUBIC stack. Figure 5.12 represents the distribution of error in

prediction for the CUBIC stack on 219 downloads. The figure shows that the error in prediction is less than 1 RTT in more than 95% of cases, which is a remarkable accuracy considering that the typical time granularity of CCA decisions is precisely the RTT. We note that the remaining 5% of cases exhibit higher errors, which can be attributed to some fluctuations in RTT values. The high accuracy of our method demonstrates its effectiveness in automating the detection of the SS exit time, which can save considerable time and effort in network management and troubleshooting.

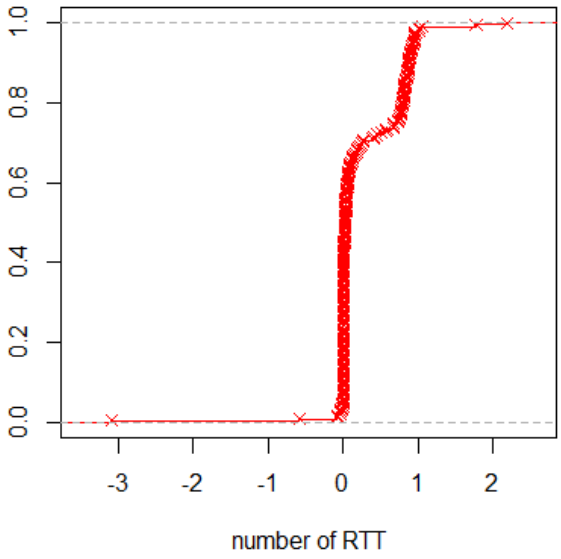

Figure 5.12 – Cumulative distribution function of the difference in RTT units between the SS exit times from the slope 1/2 method and the one logged by the CUBIC server

5.6.5 Method evaluation with BBR traffic

We also analyzed the prediction error distribution for the BBR TCP stack, as shown in Figure 5.13 for 241 downloads. We observed that 90% of the cases are bounded between -2 and 2 RTT, indicating a larger deviation in prediction error compared to the CUBIC stack. However, we believe that this deviation is not entirely due to mispredictions of our slope 1/2 method, but rather due to the difficulty in obtaining an accurate groundtruth in the case of BBR. As we mentioned earlier, BBR algorithm waits for 3 RTT before announcing the exit from the SS state. Moreover, the BBR implementation we used does not have a single state for SS, but rather a region in the parameter space. These factors make it challenging to accurately identify the groundtruth SS exit times.

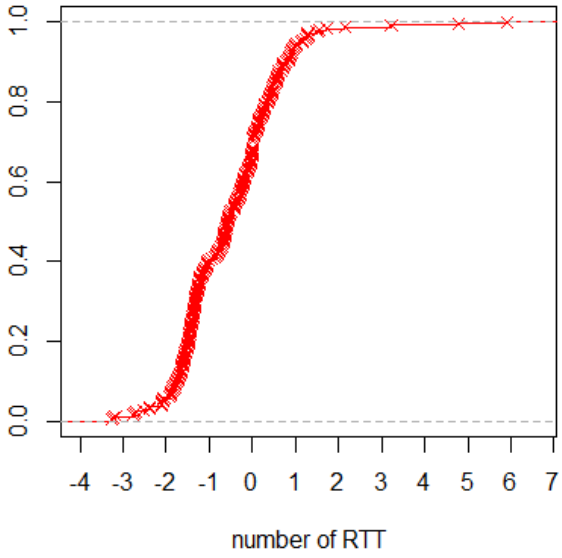

Figure 5.13 – Cumulative distribution function of the difference in RTT units between the SS exit times from the slope 1/2 method and the one logged by the BBR server

5.7 Method application to network fault identification

This representation has revealed distinct patterns that serve as indicators of various types of network degradation. These patterns can be readily identified using straightforward criteria, enabling efficient classification of network faults. The ability to recognize and understand these patterns is crucial in diagnosing and resolving issues that affect network performance. Several common faults have been observed, the graphical representations corresponding to these faults are described below. We mainly focus on three cases that were already mentioned in Chapter 4: normal QoS, QoS degradation due to loss (transmission and congestion), and QoS degradation due to jitter.

5.7.1 Case 1: Normal QoS

In this ideal scenario, the binary search procedure conducted during the SS phase effectively identifies the path bottleneck within the network. As a result, the BIF curve as a function of SEQ remains constant, as illustrated in Figure 5.14, after exiting the SS state. This consistent and unchanging $BIF = f(SEQ)$ curve means a state of good and

stable QoS within the network.

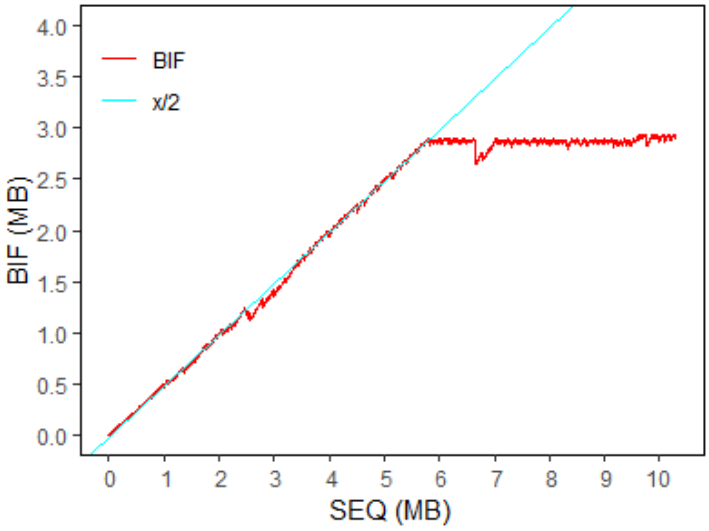

Figure 5.14 – Constant BIF after SS exit: good QoS

The constant $BIF = f(SEQ)$ curve indicates that the CCA successfully maintains a balance between the rate at which packets are transmitted and the available network capacity. This equilibrium is essential for ensuring that network resources are optimally utilized and that there is no excessive buildup of data in transit. In such an ideal case, the network is efficiently handling the traffic load without experiencing congestion or performance degradation. This stable QoS is indicative of a well-provisioned network infrastructure and effective congestion control mechanisms in place, it demonstrates the network's robustness and its capacity to deliver consistent and reliable service to users. Such an outcome is desirable for ensuring smooth and uninterrupted data transmission, minimizing delays, and maximizing the overall efficiency of the network.

5.7.2 Case 2: QoS degradation due to loss

Figure 5.15 showcases a distinct pattern characterized by a sudden decrease in the BIF value following the identified SS exit time. Subsequently, the BIF value remains significantly lower than its peak value during the SS phase. This pattern is commonly associated with a degradation in QoS attributed to packet loss due to the CCA mechanism that decreases the BIF value in case of losses. These losses could be isolated in the case of transmission problems, or in bursts in the case of congestion problems.

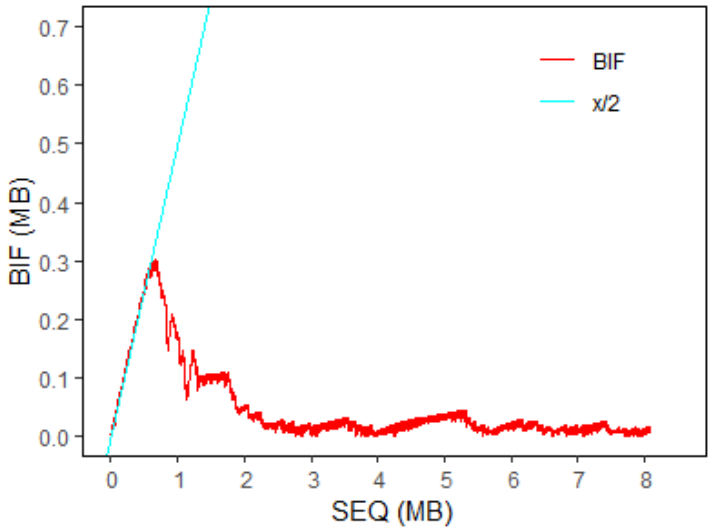

Figure 5.15 – BIF plunge after SS exit: loss

The observed drop in the BIF value indicates a reduction in the amount of data in transit within the network. Such a decrease can be caused by various factors, including cross-traffic competition and transmission errors. Cross-traffic competition refers to the presence of data traffic generated by other sources sharing the network infrastructure, this can lead to congestion and subsequent packet loss due to the high demand, resulting in the observed pattern in the BIF curve. In addition, transmission errors can contribute to the loss of packets within the network. These errors may appear due to issues such as signal interference, network disruptions, or faulty transmission equipment. The loss of packets can adversely affect the overall QoS by disrupting the timely and reliable delivery of data, forcing the CCA to exit the SS before reaching the full true bottleneck capacity. To differentiate between transmission and congestion problems, we should focus on the losses, and thus, on the decrease in the BIF value as after each loss the CCA reduces its BIF value. In the case of a congestion problem, the decrease should happen successively and so the BIF value keeps decreasing as multiple packet losses are noticed in congestion problems. On the other hand, in the case of transmission problems, the losses are isolated. Therefore the reduction in the BIF value should happen only for one time before the BIF value reincreases.

By recognizing this pattern, network administrators and engineers can identify potential sources of congestion or transmission errors and take appropriate measures to mitigate the impact on QoS. This may involve adjusting network configurations, opti-

mizing resource allocation, or implementing error correction mechanisms to enhance the reliability and performance of the network.

5.7.3 Case 3: QoS degradation due to jitter

In addition to packet loss, another common underlying cause for early SS exit is packet delay variation, also known as jitter [5]. Jitter refers to the variability in the time it takes for packets to traverse the network, and it can occur even before the bottleneck capacity has been reached. Figure 5.16 demonstrates this particular scenario.

In contrast to the first case that represents a case without degradation (5.7.1 where the BIF stays stable after the SS exit time, the Figure 5.16 case shows a continuous growth in the BIF, but at a much slower rate. This increase indicates that the CCA did not reach the true bottleneck capacity during the SS state, therefore the BIF value continues to increase after. This distinct difference between the two patterns allows for easy discrimination between them. This behavior is frequently observed in mobile access networks that are impacted by large jitter values. Jitter can occur due to various factors, including network congestion, variations in transmission speeds, or fluctuations in the wireless signal strength. These variations in packet delay can lead to early SS exit, as the CCA perceives the increased delay as a sign of network congestion, prompting it to prematurely exit the SS phase.

Identifying and distinguishing these different patterns is crucial for effectively diagnosing and troubleshooting network performance issues. By recognizing the characteristic behaviors associated with packet loss and jitter, network administrators can gain insights into the root causes of QoS degradation in mobile access networks. This understanding can guide the implementation of appropriate mitigation strategies to minimize the impact of jitter and improve the overall performance and reliability of the network.

5.8 Conclusion

As many network operators use the SS state duration as a key indicator for the diagnosis of faults, it is crucial to automate its extraction to save human experts time. In this work, we have presented a method to automatically detect the exit from the Slow-Start state, enabled by an innovative timeless representation of the observed packet series. We deployed our method in active and passive probes in four countries with varied access

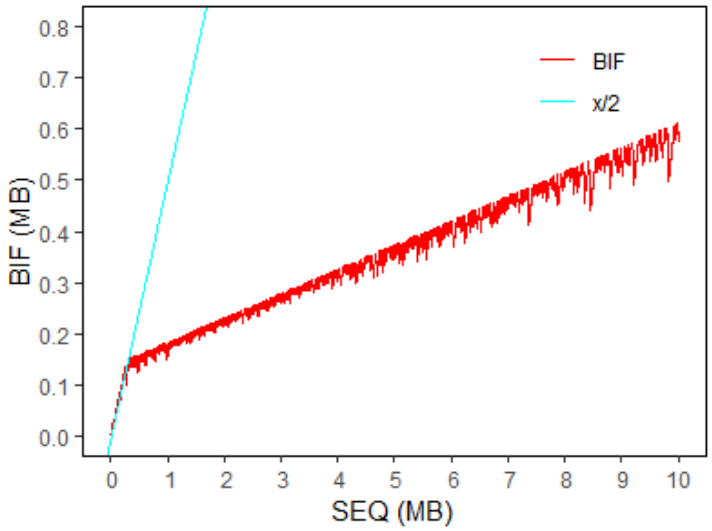

Figure 5.16 – Slow BIF growth after SS exit: jitter issue

networks and traffic conditions and tested it with both CUBIC and BBR. This evaluation shows the method to be accurate enough for the purpose, i.e. very often within 1 RTT of the oracle.

In the next chapter, we explore how this automated SS detection tool is a powerful tool that allows us to identify TCP BBR traffic among other CCA variants.

Chapter 6

BBR Identification

The contributions of this chapter have been published in the conference ITC [47]:

— Ziad Tlaiss, Alexandre Ferrieux, Isabel Amigo, Isabelle Hamchaoui, Sandrine Vaton. Automated Identification of BBR Traffic based on Packet Inter-Arrival Times Analysis, *35th International Teletraffic Congress ITC (ITC-35)*, 2023

6.1 Introduction

The Internet is a complex and constantly evolving system, and CCAs play a crucial role in ensuring its functioning by managing network performance. These algorithms regulate the flow of data within a network and optimize data transmission for efficiency and effectiveness. They do this by continuously estimating available network resources and adjusting the data transmission rate accordingly. For network operators, identifying the CCAs being used on their network is essential to gain valuable insights into network performance and device behavior. This information can help them gain a better understanding of how the network is being utilized and which algorithms are most effective in different scenarios. With a clear understanding of the CCAs in use, they can make decisions about network design, configuration, and management.

A loss-based algorithm like CUBIC reduces the sending rate in case of high levels of packet losses, leading to a drop in throughput for users. In order to ensure a high-quality user experience for clients utilizing CUBIC traffic, network operators are required to make substantial investments. These investments are crucial for provisioning bandwidth and buffering capacity, thereby minimizing the occurrence of packet loss and enhancing overall service quality. The arrival of BBR, with its novel approach both to congestion detection and sending rate control, presents an opportunity for network operators to reduce invest-

ment in network infrastructure. Indeed BBR offers better resource utilization, improved performance, and lower latency, while massively reducing the need for buffering space.

Network operators seek automated methods to identify the specific CCA employed within their network, specifically discerning whether it is BBR or another variant. This information is vital for optimizing their network infrastructure effectively. In this chapter, we present a method for automatically identifying BBR traffic on the Internet. Our method relies on analyzing packet inter-arrival times, specifically comparing the distribution of packet inter-arrival times during the Slow-Start state of a BBR capture with those of a CUBIC capture. In our method, we focus on detecting the silence period after packet bursts that are present in almost all non-BBR congestion control algorithms. This method is characterized by a very simple frontend signal processing that exploits the algorithms' core principles, allowing for a tiny parameter space dimension (two), which is sufficient for robust discrimination.

6.2 Motivation of the method

One of the main objectives of our approach is to detect whether or not pacing is used in a TCP connection. In a paced TCP, instead of sending new packets immediately after receiving an acknowledgment, the packets are held back for a certain duration, which results in less bursty TCP traffic [3]. In other words, TCP pacing is used to evenly space data sent into the network over an entire round-trip-time; in this case, data is not sent in a burst. To measure the fraction of TCP traffic that is paced, we illustrate this using BBR and CUBIC CCAs. BBR, with its novel approach both to congestion detection and sending rate control, is a TCP CCA that paces. On the other hand, CUBIC, a loss-based algorithm that reduces its sending rate in case of high levels of packet loss, is a TCP CCA that does not pace. Recent studies show the impact of pacing on improving TCP performance [11], especially in the case of shallow buffers (i.e. small buffer size in routers).

For network operators, understanding whether the TCP in use on their network is paced or not is crucial. For instance, detecting whether pacing is employed within operators' networks helps them measure the burstiness of the resulting TCP traffic. This burstiness might have an impact on how network buffers are dimensioned; operators need to adjust buffer sizes and network management strategies accordingly. For example, the guidelines outlined in [4] might no longer be applicable. Additionally, bursty traffic can lead to sudden spikes in network utilization, potentially causing congestion. By identi-

fying bursty traffic, operators can implement measures to smooth out traffic patterns, thereby preempting congestion-related issues.

By design, of the two end-points of a connection, only the source has explicit knowledge of the running CCA flavor (CUBIC or BBR). As a consequence, in this paper, we work with traces from controlled sources (i.e. our lab servers) to calibrate our recognition algorithm. Extension of this method to real life traffic from third-party servers will be considered in section 6.4.

From the mere observation of traces, it can be seen that at the beginning of the connection, BBR and CUBIC emission patterns are particularly dissimilar (Figure 6.1): CUBIC is typically bursty whereas BBR exhibits a smooth emission pattern. This results from a fundamental difference between these two CCAs: BBR is rate-based while CUBIC is window-based. In other words, CUBIC sends its full window in a burst and waits for acknowledgements, while BBR paces its emissions according to a rate-based policy.

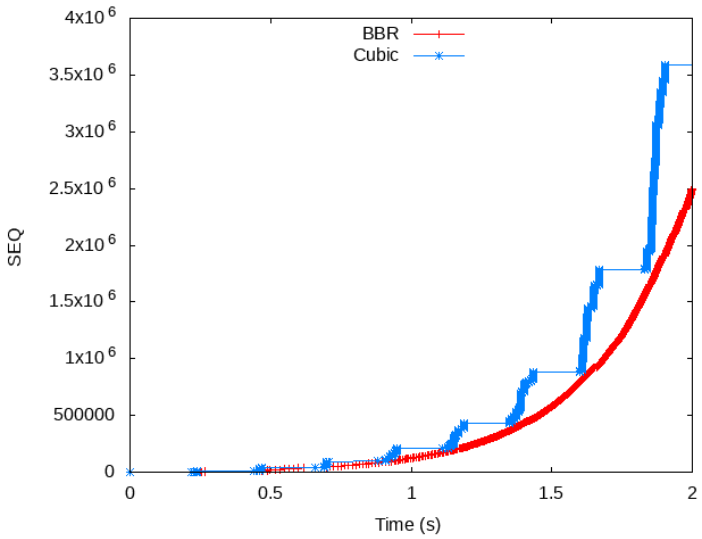

Figure 6.1 – SEQ against time graph of real CUBIC and BBR connections during SS

However, after SS exit, CUBIC also tends to behave smoothly, as it is subject to the so-called "ACK clocking" phenomenon [6], due to bottleneck-induced pacing, as shown in Figure 6.2. This precisely happens when reaching the bottleneck capacity, typically on SS exit.

In a nutshell, the best period for discrimination appears to be during the SS. As a consequence, we isolate the SS period of each connection and characterize the emission patterns - more precisely, the burstiness - in this period only. To assess it, our approach

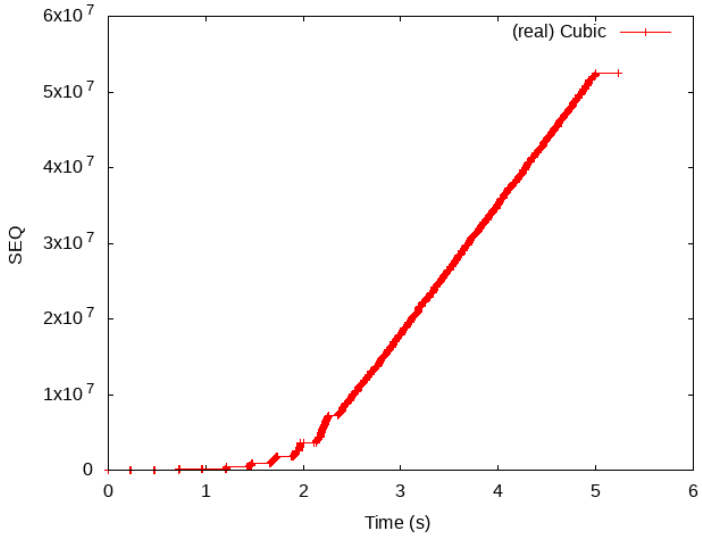

Figure 6.2 – SEQ against time for a CUBIC capture

relies on analyzing the distribution of packet inter-arrival times (INTRPKT), derived from the packet capture.

6.3 Analyzing packet inter-arrival times: modelling and inference

BBR and CUBIC are two popular congestion control algorithms representing together the main part of Internet volume [2]. As their behavior is particularly contrasted in the presence of loss or other degradation, it is crucial to differentiate them when investigations are conducted. Indeed, erroneous conclusions can easily be drawn in case of misidentification of the CCA flavor. For example, analyzing the behavior and reaction of a CCA after a loss highly depends on the type of CCA in use on the network. Despite the abundant literature on their respective strengths and weaknesses, little has been proposed on solid discrimination tools. In the present section, we propose an identification method based on their different behavior during the SS state.

6.3.1 Characterization of CUBIC and BBR during SS state

In order to analyze and understand the statistical properties of the INTRPKT distribution, we present the histograms in Figure 6.3 for a BBR connection (shown in red)

compared to a CUBIC connection (shown in blue). These histograms plots reveal distinct differences in the shapes of the distributions between the two algorithms.

The BBR distribution appears to have a narrower shape, indicating that the packet arrival times follow a more tightly clustered pattern. This is because BBR tends to maintain a more consistent and controlled pacing of packet transmission, resulting in a less varied distribution of INTRPKT. On the other hand, the CUBIC distribution displays a different characteristic. It is typically a mixture of two components: short INTRPKTs for the packets inside the bursts and a few long INTRPKTs corresponding to the period of silence between 2 bursts, meaning that it has a greater likelihood of observing larger INTRPKTs. This nature is because CUBIC can experience longer delays or variations in packet arrival times, which may result in a less consistent transmission rate.

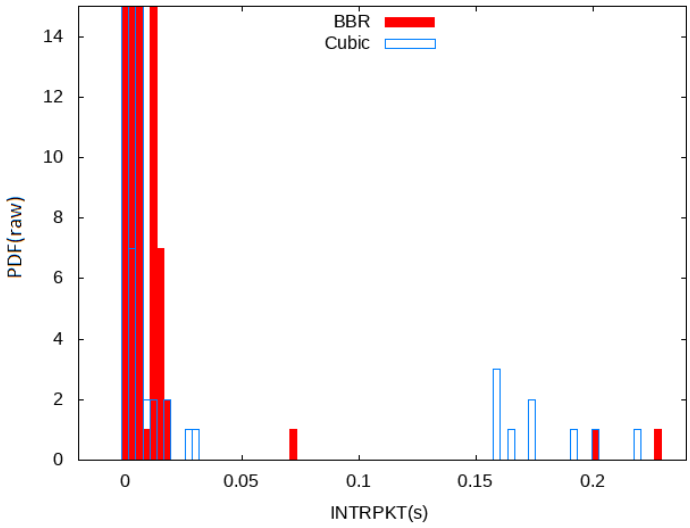

Figure 6.3 – The PDF of INTRPKT for a CUBIC and BBR capture

CUBIC INTRPKT distribution

The main peak of a CUBIC inter-arrival distribution corresponds to the short INTRPKTs within bursts, while the tail of the distribution corresponds to long INTRPKTs resulting from periods of silence between two bursts as shown in Figure 6.4. Let's consider N the average number of packets in a burst. In this context, we can determine the probability of an inter-packet arrival time belonging to a burst or a silence between two bursts. The probability for an INTRPKT to belong to a burst (respectively to silence between two bursts) is $\frac{N}{N+1}$ (respectively $\frac{1}{N+1}$). We can then express the pdf of the INTRPKT as

Figure 6.4 – Packet arrivals in a CUBIC connection

a mixture distribution f, as shown in this equation:

$$f(x) = \frac{1}{N+1}g(x) + \frac{N}{N+1}h(x) \tag{6.1}$$

Here, $g(x)$ is the pdf of the OFF periods that correspond to large INTRPKTs, while $h(x)$ is the pdf of packet interarrival times within bursts.

BBR INTRPKT distribution

In contrast to the CUBIC inter-arrival distribution, a typical BBR INTRPKT distribution shows a distinct characteristic with a peak primarily observed on small INTRPKT values. The occurrence of large INTRPKT is relatively rare in BBR due to its smooth emission pattern. The discrimination task between BBR and CUBIC distributions can therefore be simplified to recognize the difference between single-peak distributions (BBR) and two-component mixture distributions (CUBIC).

Expanding small contributions

In the CUBIC case, it can be observed from Equation 6.1 that the probability of occurrence of a large INTRPKT component is significantly smaller in comparison to the probability of occurrence of a short-INTRPKT component. This is primarily due to the large number of events within a burst, which outweighs the relatively smaller number of pauses between bursts. As our objective is to discriminate between single-peak and two-component mixture distributions, it is essential to address this inherent imbalance by magnifying the minority contribution, specifically the long-INTRPKT component. Since the two components' mixture behavior of the CUBIC distribution is of particular interest, enhancing the representation of the long-INTRPKT component becomes crucial.

Note that, as with any distribution, the CUBIC pdf (Eq. 6.1) can be approximated using an empirical distribution based on the measured INTRPKT values:

$$\hat{f}(x) = \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{x_i}(x) \tag{6.2}$$

We note that T is the number of INTRPKTs and $\mathbb{1}_{x_i}(x)$ is the indicator function that is equal to 1 when $x = x_i$ and to 0 otherwise.

To magnify the minority contribution of the long-INTRPKT component in the CUBIC distribution, we thus weight each observed INTRPKT value x_i by:

$$w_i = \frac{x_i}{\sum_{j=1}^{T} x_j}$$

This ensures that the two-component nature of the distribution receives adequate attention and significance during the discrimination process. So we now consider the rebalanced empirical distribution $\hat{f}_{bal}$:

$$\hat{f}_{bal}(x) = \sum_{i=1}^{T} w_i \mathbb{1}_{x_i}(x) = \frac{1}{\sum_{j=1}^{T} x_j} \sum_{i} x_i \mathbb{1}_{x_i}(x) \tag{6.3}$$

Notably, if the timescale is renormalized to $[0, 1]$ for the observation period (SS state), then $\sum_{j=1}^{T} x_j = 1$ and

$$\hat{f}_{bal}(x) = \sum_{i} x_i \mathbb{1}_{x_i}(x)$$

It is interesting to notice that the resulting empirical distribution $\hat{f}_{bal}$ closely approximates the distribution of INTRPKTs that would be observed if sampling was uniform over time as shown in Figure 6.7 rather than uniform over packets as shown in Figure 6.8. Figure 6.5 and Figure 6.6 represent real-life examples of the two sampling techniques. This observation highlights the impact of the sampling method on the resulting distribution. Choosing each sampling instant uniformly over the observation period corresponds to a Poisson sampling.

The PASTA (Poisson Arrivals See Time Averages) property is a fundamental property of Poisson processes, a result of this property states that the probability of a sampling instant falling during a particular event is equal to the ratio of the average duration of that event to the average total duration of both events combined. In the context of INTRPKT, this implies that the probability of a sampling instant occurring during a burst or a silence period (ON and OFF period as shown in Figure 6.9) can be determined based on the average durations of these periods. Specifically, let T_{ON} denote the average duration of a burst, and T_{OFF} represent the average duration of a silence period. Thus,

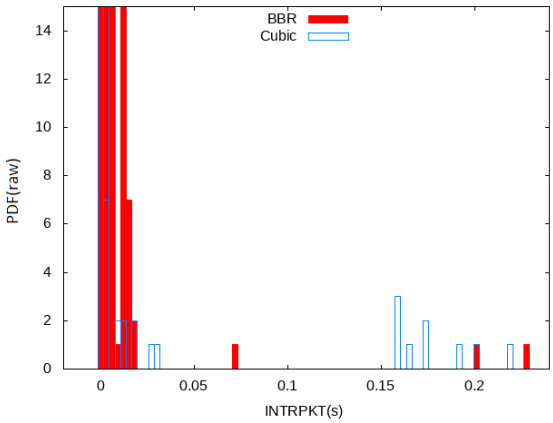

Figure 6.5 – The PDF of INTRPKT for a CUBIC and BBR capture using the uniform over packets sampling

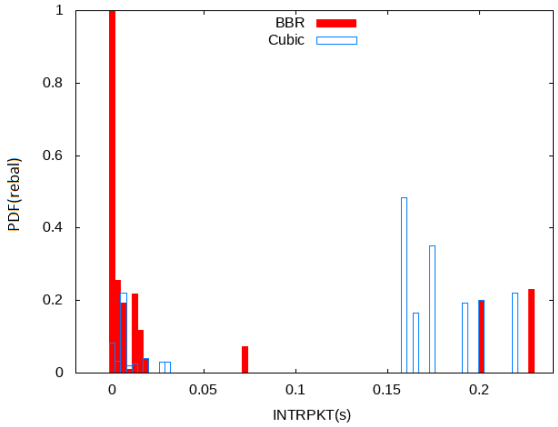

Figure 6.6 – The PDF of INTRPKT for a CUBIC and BBR capture using the uniform over time sampling

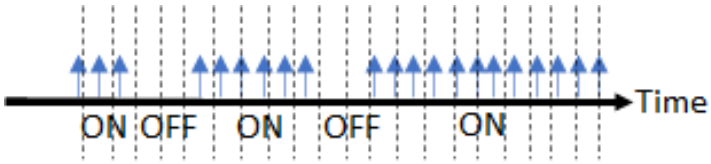

Figure 6.7 – Uniform sampling over time.

the probability of a sampling instant falling during a burst is given by:

$$\frac{T_{ON}}{T_{ON} + T_{OFF}}$$

Similarly, the probability of a sampling instant occurring during a silence period is given by:

$$\frac{T_{OFF}}{T_{ON} + T_{OFF}}$$

These probabilities reflect the relative durations of bursts and silences. The longer the average duration of a burst compared to the average duration of silence, the higher the probability of a sampling instant falling within a burst, and vice versa. Therefore $\hat{f}_{bal}(x)$ is an empirical approximation of the following mixture distribution:

$$\phi(x) = \frac{T_{OFF}}{T_{ON} + T_{OFF}} g(x) + \frac{T_{ON}}{T_{ON} + T_{OFF}} h(x) \tag{6.4}$$

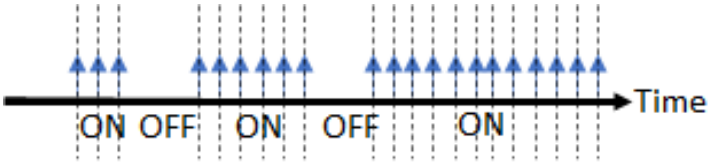

Figure 6.8 – Uniform sampling over packets.

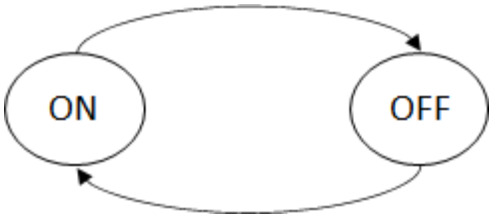

Figure 6.9 – Illustration of the two events ON and OFF

It can be observed that the weight $\frac{T_{OFF}}{T_{ON}+T_{OFF}}$ of $g(x)$ in Eq. 6.4 is not negligible (contrary to the weight $\frac{1}{N+1}$ in Eq. 6.1). Replacing a uniform sampling strategy across packets with a uniform sampling strategy over time gives more weight to the inter-packets which correspond to silences between two bursts. This is an interesting property because it makes it possible to better differentiate the distribution of INTRPKT between CUBIC and BBR.

6.3.2 Separating CUBIC connection from a BBR connection using CDFs

To better capture distribution features, we switch from PDFs (Probability Density Functions) to CDFs (Cumulative Distribution Functions). Figures 6.10 and 6.11 respectively show the raw and re-balanced CDFs of INTRPKTs during the SS state of CUBIC and BBR. We see that in the rebalanced case (which mimics a uniform sampling over time), the two distributions are much better separated than in the raw case (corresponding to the original uniform sampling over packets).

Let us assume that our objective is to determine whether a connection is carrying BBR traffic or not. To accomplish this, we can frame the problem as a hypothesis test between two options: $H_0 : CUBIC$ and $H_1 : BBR$. During the SS state, we measure INTRPKT values $x_1, x_2, \ldots, x_T$, and propose a method that makes a decision based on these values.

Since CUBIC tends to have more long INTRPKTs, we decide that the connection is

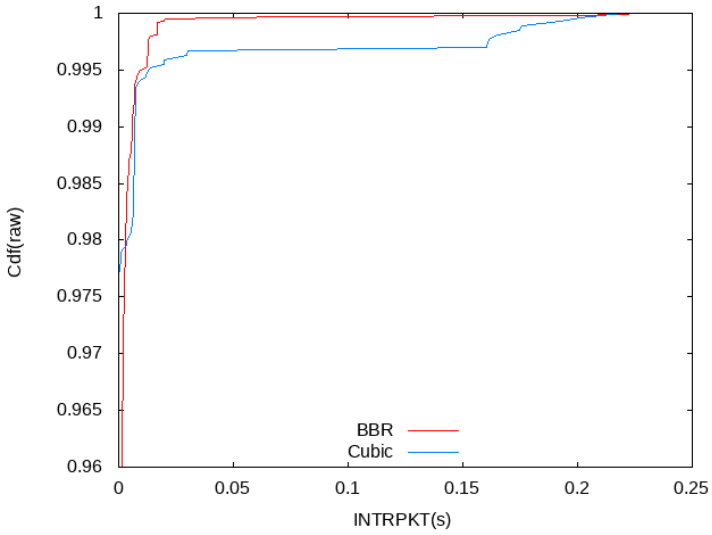

Figure 6.10 – CDF of INTRPKT for a CUBIC and BBR capture using the raw distribution (uniform sampling over packets)

using CUBIC if the proportion of values $(x_i)_{i=1,T}$ smaller than a threshold x is smaller than y. Conversely, if the proportion of values $(x_i)_{i=1,T}$ smaller than x is greater than y, then we conclude that the connection is using BBR.

In other words, we are seeking a point (x,y) that separates the red and blue curves in Figure 6.11. If the curve $(x, \hat{f}_{bal}(x))$ (Eq. 6.3) is below this point, we decide CUBIC; if it is above the point, we decide BBR. Since there are two types of risk in a hypothesis test between two options, false alarm (detecting BBR but the capture was with CUBIC) and non-detection (detecting CUBIC but the capture was with BBR), we propose to fix $y = \theta(x)$ for a particular value of x, so that the two risks have equal probability. This roughly corresponds to positioning the point (x, y) in the middle, between the green and red curves in Figure 6.11 (assuming equal numbers of CUBIC and BBR connections). We then search for the best value of x that minimizes the total (false-alarm + non-detection) probability of error, which is the probability of misclassifying a connection. Put simply, let us consider we fix $x = 0.01$, and we then calculate y that separates the best the red and blue curves and we fix it. We then took $x = 0.02$ (0.01 step) and we calculated y that best separates the two curves. We keep repeating this procedure and we finally select the (x, y) point that separates the two curves with the minimum error rate. In the case where multiple points are selected with equal error rates, we selected the central point to ensure a wider margin.

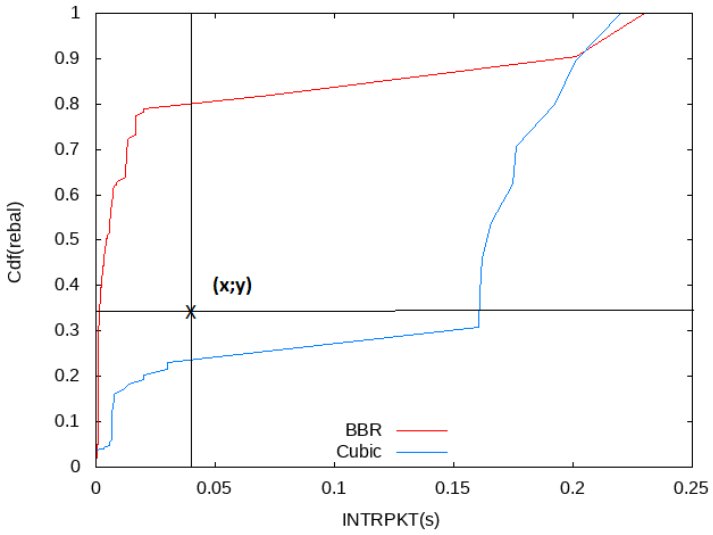

Figure 6.11 – CDF of INTRPKT for the same captures in Figure 6.10 using the re-balanced distribution (uniform sampling over time). A typical decision point $(x; y)$ is shown.

6.4 Model evaluation

In this section, we present the evaluation of our method and model for classifying connection captures. To assess the performance and effectiveness of our approach, we utilized two datasets: a training dataset consisting of 221 packet captures and a testing dataset consisting of 583 packet captures. The training dataset was used to train our model in order to select a decision point that separates a CUBIC connection from a BBR connection, while the testing dataset was employed to evaluate its classification performance.

During the training phase, we selected a single decision point based on the analysis of the training dataset. This decision point serves as a reference for the classification process and guides the model in distinguishing between BBR and CUBIC captures. We measured the classification performance by comparing the predicted classifications against the ground truth labels assigned to each connection capture.

6.4.1 Packet traces characterization

The packet traces were captured on one of our servers based in Paris, which was accessed by multiple active probes via the public Internet. The measurements were performed by conducting several downloads from our server with BBR and CUBIC CCA

algorithms. Our server is based in Europe and our probes are on another continent.

For the sake of representativity, we positioned our probes across different countries. In addition, our downloads were carried out during peak and off-peak hours, seeking different levels of congestion. Table 6.1 shows the variations in the exit time of the SS state across the different downloads (packet captures), showing different resource conditions for the different TCP connections. Indeed, as the CCA stays in the SS state as long as it estimates that the bottleneck capacity is not yet reached, the variation of the exit time of the SS state reveals variations in the available network resources. The average RTT of our captures varies between 100ms and 400ms, while the majority fall between 200ms and 300ms as shown in Table 6.2. For the considered probe destinations and by taking into account the distance between the probes and the server, an RTT value less than 250 ms usually indicates a good QoE, on the other hand, an RTT value higher than 250 ms normally means a congested network.

SSET (sec)	<0.4	[0.4;1[	[1;1.6[	[1.6;2.2[	[2.2;2.8[	>2.8
Donwloads	1%	14%	22%	30%	32%	1%

Table 6.1 – Distribution of exit time of the SS state (SSET) for the considered dataset. Variation on the exit time of the SS state can be an indicator of variation of the available network resources.

RTT value (ms)	<200	[200;250[	[250;300[	>300
Downloads	1%	53%	45%	1%

Table 6.2 – Distribution of RTT values for the considered dataset. For the considered destinations less than 250 ms corresponds to good connections and higher than 250 ms to a congested network.

6.4.2 Choice of the decision point using a training dataset

In order to identify the optimal decision point, we trained on a dataset of 221 packet captures, consisting of 133 BBR captures and 88 CUBIC captures. By applying the equal-error-rate minimization described above to the CDF curves of these packet captures, depicted in Figure 6.12, we determined that the decision point should be set at ($x = 0.14, y = 0.503$): in other words, the optimal decision criterion amounts to comparing the median of the INTRPKT distribution with $0.14 * RTT$.

Using this decision point, the minimum total error rate achieved on the training dataset is 2.6%.

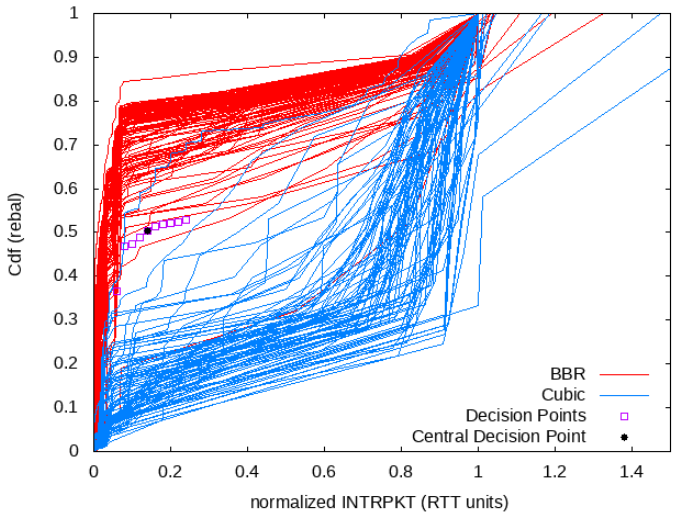

Figure 6.12 – Choice of the decision point on the training set

6.4.3 Testing the decision point to identify BBR CCA

To assess our approach, we gathered a total of 583 packet captures, consisting of 389 BBR and 194 CUBIC captures. The CDF curves of all 583 captures are presented in Figure 6.13. By using the decision point we obtained on the training dataset, the model is able to identify TCP variants with an overall error rate of only 4.1%. None of the 194 CUBIC captures is misclassified, and 16 out of the 389 BBR captures are mistakenly classified as CUBIC. This slight bias points to the necessity of a larger and more diverse training set, slated for future work.

6.5 Conclusion

In this chapter, we presented a method to automatically differentiate between BBR and CUBIC traffic. Our method focuses on the sending rate of each CCA during the slow-start phase, specifically by analyzing the inter-arrival times of packets. We used the empirical CDF of inter-arrival times, under a pertinent resampling allowing us to give a significant weight to underrepresented events (long inter-arrival times, corresponding to off periods). These CDFs are then compared to a decision point calculated during the

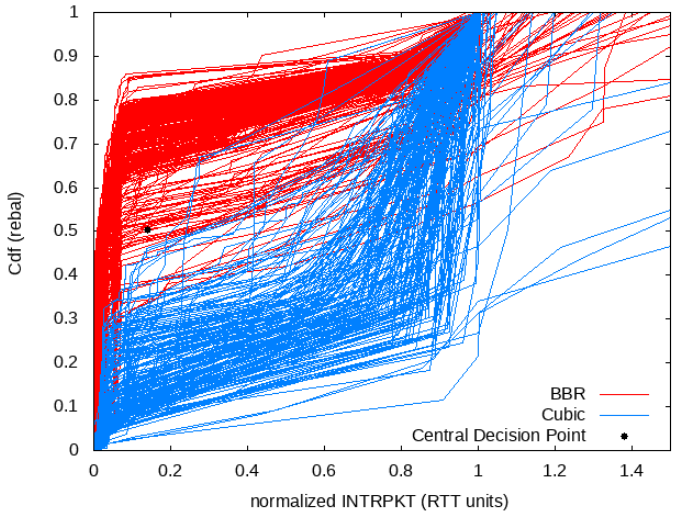

Figure 6.13 – Model evaluation on an independent test set

training phase so as to minimize the total equal error rate. Our method was trained on 221 packet traces, including 133 BBR and 88 CUBIC captures, and evaluated on 583 traces, including 389 BBR and 194 CUBIC captures collected over a 4-month period under various network conditions. Our method achieved a 4.1% total error rate with no false negatives among the 389 BBR captures.

Our "BBR vs. CUBIC" classifier for TCP connections is both extremely cheap in training and runtime computational resources (as the model comprises only two dimensionless scalars and feature extraction is trivial), and quite promisingly accurate as per our preliminary evaluation. Despite our confidence in the approach, we recognize that there are various obstacles that could jeopardize it if newer versions of CUBIC and BBR were to be widely adopted by the industry. The first that comes to mind is BBRv2 [43], the most recent version of BBR, which incorporates an improved loss detection mechanism to better react to changes in network conditions. However, this improvement does not impact our method since the pacing rate during the SS state remains unchanged. Of more concern could be newer versions of CUBIC resorting to some level of pacing, which might blur the rather clear contrast with BBR during the SS phase. However, at the time of writing only a small part of providers turn to this option, partly due to its absence in the default stack tuning of popular operating systems. Should this state of affairs evolve, one might consider addressing this 3-class task with two decision points, with somewhat lowered accuracy.

Chapter 7

Conclusion

The work presented in this thesis focuses on the automation of network troubleshooting using performance measurements extracted from packet captures. Our first objective was to propose new approaches and tools that allow operators' network troubleshooting experts to quickly identify the root cause of degradation. Networks generate vast amounts of data from various sources, and analyzing these data manually is nearly impossible, making automation essential for effective troubleshooting. Automation can significantly reduce operational costs by decreasing the time and manpower required for troubleshooting. It allows experts to focus on more complex issues that require human intervention. We started our work by analyzing over 500 TCP packet traces exhibiting poor connections and identified them as anomalies (Chapter 4). We selected these captures by comparing the throughput obtained during file download from our server using TCP with the throughput achieved using UDP. By manually analyzing these packet captures, we classified four frequently observed network degradation root causes, which are transmission loss, congestion, jitter, and application-limited behavior. The focus of this work was not on identifying all of the degradation root causes but on recognizing the most frequent ones that occurred across our networks. Congestion problems were among the most salient ones, weighing more than 50% of the total. Transmission problems and jitter problems come next covering together about 45% of the cases. Application-limited behavior was only observed in some specific cases (5% of the cases) where the used congestion control algorithm was too aggressive (TCP-BBR) or where the used probe was old and/or with a small buffer capacity.

As a consequence of the manual analysis, by identifying common patterns and characteristics, we recognized the importance of the Slow-Start state in the congestion control algorithm for network packet trace analysis as a key element for identifying the root cause of degradation. To accelerate the troubleshooting process, we proposed a new tool that permits troubleshooting experts to automatically detect the Slow-Start state (Chapter 5). This detection is based on a new representation of the sequence number versus the bytes-

in-flights. In the context of our research, we adopt this particular representation with the purpose of discerning the last packet within the Slow-Start state of the Congestion Control Algorithm (CCA). This determination relies on a distinctive relationship between the sequence number values and the bytes-in-flight values, which holds true exclusively during the SS phase. Notably, this methodology holds applicability to both QUIC and TCP Congestion Control Algorithms, benefiting from their structural similarities. Nonetheless, by requiring the bytes-in-flight, its utilization for QUIC traffic necessitates active measurements, implying the ability to decrypt the traffic, while its application to TCP traffic remains feasible in both active and passive measurement contexts. In order to assess the efficacy of our suggested approach for detecting the exit from the Slow-Start phase, we tested it with Cubic and BBR Congestion Control Algorithms as they represent the two most used congestion control algorithms. With our method, in the case of CUBIC, we were able to predict the exit from Slow-Start in 95% of the cases with a prediction error of less than 1 RTT. In the case of BBR and due to the complexity in accurately detecting the true exit time from the Slow-Start state, in 90% of the cases the prediction error was below 2 RTT. The Slow-Start exit time tool proves to be helpful in quickly identifying three of the degradation root causes mentioned above, namely, congestion problems, transmission problems, and jitter issues. Thus we implemented our tool across Orange probes and it is used today by Orange troubleshooting experts to accelerate the process of troubleshooting in order to quickly identify the repeated degradations, which allows them to focus on more complicated degradation problems. We believe that by using our Slow-Start exit time detection tool, scientists could refine the criteria so as to identify more classes and integrate the method into an automated classifier. On another aspect, detecting the Slow-Start state exit time could be a powerful tool to investigate the fair share of a connection between multiple TCP variants, and it can be used to identify TCP flavors.

After isolating the Slow-Start state in a TCP connection, we focused on developing a method for detecting the use of pacing in TCP connections (Chapter 6). Paced TCP is a methodology where packets are transmitted in a more evenly distributed flow of data, minimizing bursty behavior in TCP traffic. This controlled approach ensures that data transmission is spaced evenly across the entire round-trip time, preventing sudden bursts of data. Due to the impact of pacing, particularly in scenarios with shallow buffers, emphasizing its potential to enhance TCP performance, network operators are highly interested in detecting its presence in their TCP networks. This knowledge enables oper-

ators to measure the burstiness of TCP traffic, a factor critical in determining appropriate buffer sizes and network management strategies. Traditional buffer dimensioning guidelines might need revision in light of this pacing-induced shift in traffic behavior. Moreover, bursty traffic patterns can lead to abrupt spikes in network utilization, potentially causing congestion. Identifying such traffic allows operators to proactively implement measures to smooth out these patterns, mitigating congestion-related challenges and ensuring a more stable network environment. As TCP flavors continue to diversify, understanding and managing paced TCP traffic will remain crucial for efficient network operation and performance optimization. In order to detect paced TCP traffic, we employed BBR and CUBIC Congestion Control Algorithms as case studies to demonstrate the concept of paced TCP. BBR embodies a pacing approach, in contrast, CUBIC does not utilize pacing. Our analysis has highlighted a distinct contrast in emission patterns between CUBIC and BBR Congestion Control Algorithms. This dissimilarity is rooted in the core mechanisms of the two Congestion Control Algorithms. CUBIC sends its entire window of packets at once and then pauses to await acknowledgments. This behavior results in observable patterns of packet bursts followed by periods of silence. This burst-like behavior is intrinsic to CUBIC, as it ensures the successful reception of sent packets before initiating the transmission of the next burst. In contrast, BBR adopts a rate-based strategy, pacing its emissions accordingly. Consequently, the distinct burst/silence pattern observed in CUBIC traffic is absent in BBR. To decide whether the TCP flavor in use is CUBIC or BBR, we focused on the analysis of packet inter-arrival times. We noticed that the occurrence of large packet inter-arrival times is relatively rare in BBR, therefore, to discriminate between BBR and CUBIC distributions we aimed to recognize the monomodal distributions in the case of BBR vs. the bimodal distributions in the case of CUBIC. In view of the significantly large number of events within a burst compared to the relatively small number of pauses between bursts in CUBIC, we noticed that the large-inter-arrival component is dwarfed by the short-inter-arrival component. Thus, in order to effectively discriminate between mono- and bimodal distributions, it is necessary to address this imbalance by amplifying the minority contribution, specifically the large-inter-arrival component. After giving more weight to this component, we found a way to compute a decision point that effectively separates the CUBIC from the BBR curves by minimizing the total probability of error, which encompasses both false alarm and non-detection probabilities. we then evaluated our method for classifying connection captures, using two datasets: one of 221 packet captures for training, and the other of

583 packet captures for testing. The minimum total error rate attained on the training dataset amounts to 2.6% (With this decision point, 1 over 88 CUBIC captures would be mis-identified as BBR, and 2 over 133 BBR captures would be mis-identified as CUBIC). We then tested the decision point obtained with the testing dataset to the captures gathered for evaluation. Our model was capable of identifying the TCP variants with an overall error rate of only 4.1%. None of the 194 CUBIC captures are misclassified, while 16 out of the 389 BBR captures are mistakenly classified as CUBIC.

This thesis has three main contributions: the $BIF = f(SEQ)$ representation, the slope-1/2 method for SS detection, and the weighted-interpacket CDF view to detect pacing. While clearly deserving refinements, we hope they can, together or in isolation, serve as building blocks - or inspiration - for a family of more ambitious, fully automated troubleshooting tools.

Perspectives and recommendations for further exploration

In the pursuit of advancing the applicability and effectiveness of our proposed network troubleshooting tools, in future research, we aim to make our Slow-Start detection tool even better by checking how well it works in different types of networks. For instance, we want to make sure the tool can handle encrypted data while still being effective. We plan to expand our Slow-Start detection mechanism to include the passive measurement of QUIC traffic. This enhancement will specifically focus on accurately estimating the bytes-in-flight values. As QUIC traffic continues to grow and significantly influence network traffic patterns, a strategic expansion would be to ensure that our diagnostic capabilities will evolve alongside. Envision a future where QUIC's ubiquity demands a specialized approach for its unique challenges. This approach will enable us to precisely identify and address performance issues unique to this protocol.

In the future, we plan to develop an extension towards a fully automated tool capable of autonomously detecting the root causes of network degradation. This tool will be invaluable for quick fault detection and is designed to operate independently from operators. It aims to enhance the diagnostic process by exploiting correlations between diverse data extracted through active and/or passive probes. By analyzing these correlations, the tool will systematically identify and locate faulty segments within the network, facilitating fast resolution. While network operators will find this tool useful for efficient

issue resolution, its primary focus goes beyond operational contexts. Researchers will be able to leverage this automated tool as a valuable resource for in-depth studies on network behaviors, gaining insights into the complexity of network degradation causes. This prospective development promises not only to advance operational efficiency but also to provide a transformative tool for researchers exploring the details of network behavior. It marks a step towards autonomous network troubleshooting, signaling significant progress in our ability to understand and manage network performance issues.

In the pursuit of future developments in network traffic identification, extending the identification of BBR traffic and classifying between BBR and CUBIC traffic opens avenues for broader applicability. The next steps involve expanding this capability to cover a wide spectrum of congestion control algorithms beyond BBR and CUBIC, thus creating a classification framework. A particularly intriguing prospect is the real-world testing of BBR traffic identification (*"in the wild"*) using passive probes, allowing for a more holistic assessment of its efficacy across diverse network environments. Furthermore, combining the BBR identification method with the Slow-Start detection method presents a compelling avenue for studying the fairness between different congestion control algorithms. This integrated approach not only enhances the sophistication of our diagnostic tools but also facilitates in-depth investigations into the interactions of various congestion control algorithms under real-world network conditions.

Moving forward, an essential area for future research involves extending the methodology used to identify BBR traffic and distinguish between paced and bursty TCP traffic beyond its current application during the Slow Start phase. By expanding the scope to encompass all phases of BBR traffic, we open up a promising avenue for enhanced flexibility in traffic classification. Accurately capturing the distinctive characteristics of paced and bursty TCP throughout its entire lifecycle will allow us to refine our diagnostic capabilities and improve the classification between different types of TCP traffic. Such an extension will also foster a deeper understanding of the behaviors of both BBR and CUBIC congestion control algorithms. This evolution in our identification method sets the stage for developing a more robust diagnostic tool, capable of accurately discerning between paced and bursty TCP traffic across various network conditions and phases.

REFERENCES

[1] Donald E. Eastlake 3rd, Rajesh Saluja, and Dave Katz. *Three-Way Handshake for IS-IS Point-to-Point Adjacencies*. RFC 5303. Oct. 2008. DOI: 10.17487/RFC5303. URL: https://www.rfc-editor.org/info/rfc5303.

[2] A. Jain; S. Pande; R. Joshi A. Mishra X. Sun and B. Leong. « The great internet TCP congestion control census ». In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* (2019).

[3] A. Aggarwal, S. Savage, and T. Anderson. « Understanding the performance of TCP pacing ». In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. Vol. 3. 2000, 1157–1165 vol.3. DOI: 10.1109/INFCOM.2000.832483.

[4] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. « Sizing Router Buffers ». In: 34.4 (2004), pp. 281–292. ISSN: 0146-4833. DOI: 10.1145/1030194.1015499. URL: https://doi.org/10.1145/1030194.1015499.

[5] Praveen Balasubramanian, Yi Huang, and Matt Olson. *HyStart++ Modified Slow Start for TCP*. Internet-Draft draft-ietf-tcpm-hystartplusplus-13. Work in Progress. Internet Engineering Task Force, Jan. 2023. 9 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-tcpm-hystartplusplus/13/.

[6] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. *TCP Congestion Control*. RFC 5681. Sept. 2009. DOI: 10.17487/RFC5681. URL: https://www.rfc-editor.org/info/rfc5681.

[7] Neal Cardwell and mubashirq. *google bbr*. https://github.com/google/bbr/blob/v2alpha/net/ipv4/tcp_bbr2.c. Source code repository. 2022.

[8] Neal Cardwell et al. *BBR Congestion Control*. Internet-Draft draft-cardwell-iccrg-bbr-congestion-control-02. Work in Progress. Internet Engineering Task Force, Mar. 2022. 66 pp. URL: https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/02/.

[9] Eoghan Casey. *Handbook of Digital Forensics and Investigation*. 2010. URL: https://www.sciencedirect.com/topics/computer-science/tcpdump.

[10] Claude Chaudet et al. « Optimal Positioning of Active and Passive Monitoring Devices ». In: *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*. CoNEXT '05. Toulouse, France: Association for Computing Machinery, 2005, pp. 71–82. ISBN: 159593197X. DOI: 10.1145/1095921.1095932. URL: https://doi.org/10.1145/1095921.1095932.

[11] Santosh Chavan. « Should paced TCP Reno replace CUBIC in Linux? » In: *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*. 2016, pp. 1–8. DOI: 10.1109/COMSNETS.2016.7439950.

[12] Jason Smith Chris Sanders. *Applied Network Security Monitoring*. 2014. URL: https://www.sciencedirect.com/topics/computer-science/wireshark.

[13] Varugis Kurien Chuanxiong Guo; Lihua Yuan; Dong Xiang; Yingnong Dang; Ray Huang; Dave Maltz; Zhaoyi Liu; Vin Wang; Bin Pang; Hua Chen; Zhi-Wei Lin. « Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis ». In: *ACM SIGCOM* (2015).

[14] Kimberly C. Claffy, George C. Polyzos, and Hans-Werner Braun. « Application of sampling methodologies to network traffic characterization ». In: *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. 1993.

[15] B. Claise. « Cisco Systems NetFlow Services Export Version 9 ». In: 2004.

[16] Benoît Claise and Alan Clark. *Guidelines for Considering New Performance Metric Development*. RFC 6390. Oct. 2011. DOI: 10.17487/RFC6390. URL: https://www.rfc-editor.org/info/rfc6390.

[17] *Computing the Internet checksum*. RFC 1071. Sept. 1988. DOI: 10.17487/RFC1071. URL: https://www.rfc-editor.org/info/rfc1071.

[18] Wesley Eddy. *Transmission Control Protocol (TCP)*. RFC 9293. Aug. 2022. DOI: 10.17487/RFC9293. URL: https://www.rfc-editor.org/info/rfc9293.

[19] Mark Fedor et al. *Simple Network Management Protocol (SNMP)*. RFC 1157. May 1990. DOI: 10.17487/RFC1157. URL: https://www.rfc-editor.org/info/rfc1157.

[20] S. Floyd. « Congestion Control Principles ». In: *RFC 2914* (Sept. 2000).

[21] Ibrahim Ghafir et al. « A Survey on Network Security Monitoring Systems ». In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. 2016, pp. 77–82. DOI: 10.1109/W-FiCloud.2016.30.

[22] Sangtae Ha, Injong Rhee, and Lisong Xu. « CUBIC: A New TCP-Friendly High-Speed TCP Variant ». In: *SIGOPS Oper. Syst. Rev.* 42.5 (July 2008), pp. 64–74. ISSN: 0163-5980. DOI: 10.1145/1400097.1400105. URL: https://doi.org/10.1145/1400097.1400105.

[23] Desta Haileselassie Hagos et al. « General TCP State Inference Model From Passive Measurements Using Machine Learning Techniques ». In: *IEEE Access* 6 (2018), pp. 28372–28387. DOI: 10.1109/ACCESS.2018.2833107.

[24] Rhee I. et al. « CUBIC for Fast Long-Distance Networks ». In: *rfc 8312* (Feb. 2018). URL: https://www.rfc-editor.org/info/rfc8312.

[25] Jana Iyengar and Martin Thomson. *QUIC A UDP-Based Multiplexed and Secure Transport*. RFC 9000. May 2021. DOI: 10.17487/RFC9000. URL: https://www.rfc-editor.org/info/rfc9000.

[26] NEAL CARDWELL; YUCHUNG CHENG; C. STEPHEN GUNN; SOHEIL HASSAS YEGANEH; VAN JACOBSON. « BBR Congestion-Based Congestion Control ». In: *Communications of the ACM* (2016).

[27] Sharad Jaiswal et al. « Inferring TCP connection characteristics through passive measurements ». In: Apr. 2004, 1582–1592 vol.3. ISBN: 0-7803-8355-9. DOI: 10.1109/INFCOM.2004.1354571.

[28] Ike Kunze Jan Rüth and Oliver Hohlfeld. « TCP's Initial Window—Deployment in the Wild and Its Impact on Performance ». In: *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT* (June 2019).

[29] Arash Molavi Kakhki et al. « Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols ». In: *Proceedings of the 2017 Internet Measurement Conference*. IMC '17. London, United Kingdom: Association for Computing Machinery, 2017, pp. 290–303. ISBN: 9781450351188. DOI: 10.1145/3131365.3131368. URL: https://doi.org/10.1145/3131365.3131368.

[30] Kary. *Understanding the tcptrace Time-Sequence Graph in Wireshark*. URL: https://packetbomb.com/understanding-the-tcptrace-time-sequence-graph-in-wireshark/.

[31] Toshihiko Kato et al. « Identification of TCP Congestion Control Algorithms from Unidirectional Packet Traces ». In: Nov. 2018, pp. 22–27. DOI: 10.1145/3291842.3291922.

[32] Mirja Kühlewind and Brian Trammell. *Manageability of the QUIC Transport Protocol*. RFC 9312. Sept. 2022. DOI: 10.17487/RFC9312. URL: https://www.rfc-editor.org/info/rfc9312.

[33] DongJin Lee, Brian E. Carpenter, and Nevil Brownlee. « Observations of UDP to TCP Ratio and Port Numbers ». In: *2010 Fifth International Conference on Internet Monitoring and Protection*. 2010, pp. 99–104. DOI: 10.1109/ICIMP.2010.20.

[34] Beatrice Martini and Niels ten Oever. *QUIC Human Rights Review*. Internet-Draft draft-martini-hrpc-quichr-00. Work in Progress. Internet Engineering Task Force, Oct. 2018. 24 pp. URL: https://datatracker.ietf.org/doc/draft-martini-hrpc-quichr/00/.

[35] Péter Megyesi, Zsolt Krämer, and Sándor Molnár. « How quick is QUIC? » In: *2016 IEEE International Conference on Communications (ICC)*. 2016, pp. 1–6. DOI: 10.1109/ICC.2016.7510788.

[36] Venkat Mohan, Y. R. Janardhan Reddy, and K. Kalpana. « Active and Passive Network Measurements : A Survey ». In: 2011.

[37] David Murray and Terry Koziniec. « The state of enterprise network traffic in 2012 ». In: *2012 18th Asia-Pacific Conference on Communications (APCC)*. 2012, pp. 179–184. DOI: 10.1109/APCC.2012.6388126.

[38] Richard Nelson, Daniel Lawson, and Perry Lorier. « Analysis of long duration traces ». In: *Computer Communication Review* 35 (Jan. 2004), pp. 45–52. DOI: 10.1145/1052812.1052824.

[39] Jitendra Padhye and Sally Floyd. « On inferring TCP behavior ». In: *ACM SIGCOMM Computer Communication Review* (July 2001). DOI: 10.1145/383059.383083.

[40] Larry Press. « The Role of Computer Networks in Development ». In: *Commun. ACM* 39.2 (Feb. 1996), pp. 23–30. ISSN: 0001-0782. DOI: 10.1145/230798.230800. URL: https://doi.org/10.1145/230798.230800.

[41] Matt Sargent et al. *Computing TCP's Retransmission Timer*. RFC 6298. June 2011. DOI: 10.17487/RFC6298. URL: https://www.rfc-editor.org/info/rfc6298.

[42] Lee Sihyung, Kyriaki Levanti, and Hyong Kim. « Network Monitoring: Present and Future ». In: *Computer Networks* 65 (June 2014). DOI: 10.1016/j.comnet.2014.03.007.

[43] Yeong-Jun Song et al. « Understanding of BBRv2: Evaluation and Comparison With BBRv1 Congestion Control Algorithm ». In: *IEEE Access* PP (Feb. 2021), pp. 1–1. DOI: 10.1109/ACCESS.2021.3061696.

[44] W. Stevens. « TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms ». In: *RFC 2001* (1997).

[45] Erik Sy et al. « A QUIC Look at Web Tracking ». In: *Proceedings on Privacy Enhancing Technologies* 2019 (July 2019), pp. 255–266. DOI: 10.2478/popets-2019-0046.

[46] Ziad Tlaiss. « Anomaly root cause diagnosis from active and passive measurement analysis ». In: *2021 33th International Teletraffic Congress (ITC-33)*. 2021, pp. 1–3.

[47] Ziad Tlaiss et al. « Automated Identification of BBR Traffic based on Packet Inter-Arrival Times Analysis ». In: *The INTERNATIONAL TELETRAFFIC CONGRESS ITC 35th*. 2023.

[48] Ziad Tlaiss et al. « Automated Slow-Start Detection for Anomaly Root Cause analysis and BBR Identification ». In: *Annals of Telecommunications*. 2023.

[49] Ziad Tlaiss et al. « Troubleshooting Enhancement with Automated Slow-Start Detection ». In: *2023 26th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. 2023, pp. 129–136. DOI: 10.1109/ICIN56760.2023.10073485.

[50] Van TONG et al. « Network troubleshooting: Survey, Taxonomy and Challenges ». In: *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*. 2018, pp. 165–170. DOI: 10.1109/SaCoNeT.2018.8585610.

[51] Ryo Yamamoto Toshihiko Kato Leelianou Yongxialee and Satoshi Ohzahata. « A Study on How to Characterize TCP Congestion Control Algorithms from Unidirectional Packet Traces ». In: *ICIMP 2016 The Eleventh International Conference on Internet Monitoring and Protection* (May 2016).

[52] Brian Trammell and Mirja Kühlewind. *The QUIC Latency Spin Bit*. Internet-Draft draft-ietf-quic-spin-exp-01. Work in Progress. Internet Engineering Task Force, Oct. 2018. 8 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-quic-spin-exp/01/.

[53] *Transmission Control Protocol*. RFC 793. Sept. 1981. DOI: 10.17487/RFC0793. URL: https://www.rfc-editor.org/info/rfc793.

[54] *User Datagram Protocol*. RFC 768. Aug. 1980. DOI: 10.17487/RFC0768. URL: https://www.rfc-editor.org/info/rfc768.

[55] Michael Welzl and David Ros. *A Survey of Lower-than-Best-Effort Transport Protocols*. RFC 6297. June 2011. DOI: 10.17487/RFC6297. URL: https://www.rfc-editor.org/info/rfc6297.

[56] Kenneth D. White. *Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations*. RFC 2925. Sept. 2000. DOI: 10.17487/RFC2925. URL: https://www.rfc-editor.org/info/rfc2925.

[57] Peng Yang et al. « TCP Congestion Avoidance Algorithm Identification ». In: *2011 31st International Conference on Distributed Computing Systems*. 2011, pp. 310–321. DOI: 10.1109/ICDCS.2011.27.

[58] Ming Zhang; Ben Y. Zhao; Haitao Zheng Yibo Zhu; Nanxi Kang; Jiaxin Cao; Albert Greenberg; Guohan Lu; Ratul Mahajan; Dave Maltz Lihua Yuan. « Packet-Level Telemetry in Large Datacenter Networks ». In: *ACM SIGCOM* (2015).

[59] Yin Zhang et al. « On the Characteristics and Origins of Internet Flow Rates ». In: *ACM SIGCOMM Computer Communication Review* 32 (Aug. 2002). DOI: 10.1145/964725.633055.

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Titre : Méthodes automatisées d'analyse des traces de paquets réseau pour la reconnaissance des anomalies et l'identification des variantes TCP

Mot clés : trace de paquet, variants TCP, mesures active, algorithmes de contrôle de congestion

Résumé : Ces dernières années, le domaine du dépannage réseau a suscité un intérêt particulier de la part des chercheurs en raison de la complexité et de l'importance de cette tâche. Le travail présenté dans cette thèse se concentre sur l'automatisation du dépannage réseau à l'aide de mesures de performance extraites des captures de paquets. La première contribution de cette thèse réside dans l'extraction de caractéristiques permettant d'identifier la cause fondamentale d'une anomalie en analysant des traces de paquets TCP montrant des connexions médiocres. Nous avons classé quatre causes de dégradation fréquemment observées : les problèmes de transmission, les problèmes de congestion, les problèmes de gigue et les limitations d'application. La deuxième contribution de cette thèse réside dans le développement d'une méthode automatisée pour détecter l'instant de sortie de l'état Slow-Start. L'importance de cette méthode réside dans le gain de temps précieux dans l'analyse des problèmes réseau, étant donné que l'état Slow-Start est un indicateur clé pour le diagnostic des défauts. La troisième contribution de cette thèse consiste en l'identification de l'algorithme de contrôle de congestion BBR. L'objectif principal est de détecter si un contrôle de l'envoi des paquets ('pacing') est utilisé dans une connexion TCP. Cette méthode repose sur la modélisation de la distribution de la durée de l'inter-paquet pendant l'état Slow-Start. L'objectif est de reconnaître les distributions monomodales de l'inter-paquet dans le cas de BBR par rapport aux distributions à deux composantes mélangées dans le cas de CUBIC.

Title: Automated network packet traces analysis methods for fault recognition and TCP flavor identification

Keywords: packet trace, TCP variants, active measurements, Congestion Control Algorithms

Abstract: In recent years, the field of network troubleshooting has garnered significant attention from researchers due to the complexity and importance of this task. The work presented in this thesis focuses on automating network troubleshooting using performance metrics extracted from packet captures. The first contribution of this thesis lies in extracting features to identify the root cause of an anomaly by analyzing TCP packet traces with bad performance. We have categorized four frequently observed causes of degradation: transmission problems, congestion problems, jitter problems, and application-limited problems. The second contribution of this thesis involves developing an automated method to detect the moment of exiting the Slow-Start state. The significance of this method lies in saving valuable time in the analysis of network degradation, as the Slow-Start state serves as a key indicator for fault diagnosis. The third contribution of this thesis revolves around identifying the BBR congestion control algorithm. The primary goal of our approach is to detect whether packet pacing is employed in a TCP connection. This method relies on modeling the distribution of inter-packet duration during the Slow-Start state. The objective is to distinguish unimodal distributions of inter-packet intervals in the case of BBR compared to mixed two-component distributions in the case of CUBIC.