Mécanismes de supervision distribuée pour les réseaux de communication dynamiques.

Denis Carvin

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National des Sciences Appliquées de Toulouse (INSA de Toulouse)

Présentée et soutenue par :
Denis Carvin

le mercredi 15 juillet 2015

Titre :

Mécanismes de supervision distribués pour la gestion des réseaux dynamiques

---

École doctorale et discipline ou spécialité :
ED MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes (UPR 8001)

Directeur/trice(s) de Thèse :

Philippe Owezarski et Pascal Berthou

Jury :
Christophe Chassot, Professeur des universités, président du jury
Véronique Vèque, Professeure des universités, rapporteur
Fabrice Valois, Professeur des universités, rapporteur
Isabelle Guérin-Lassous, Professeure des universités, membre du jury
Philippe Owezarski, Professeur des universités, membre du jury
Pascal Berthou, Professeur des universités, membre du jury

I am searching for abstract ways of expressing reality, abstract forms that will enlighten my own mystery.

King Eric

Résumé

Avec l'arrivée massive des technologies sans fil, le nombre de terminaux mobiles n'a cessé de croître, pour des usages et des ressources de communication diversifiés. En intégrant les objets du quotidien, nos réseaux de communications sont devenus dynamiques aussi bien en termes de ressources que de topologie physique, offrant accès à des informations de plus en plus riches. La tâche de gestion s'est ainsi complexifiée et requiert des temps de réponse de plus en plus courts difficilement réalisables par un administrateur humain. Il devient indispensable de mettre en œuvre des capacités de gestion autonomes pour les nouveaux réseaux.

Dans tous les cas, la gestion d'un système implique une étape essentielle : sa mesure et sa supervision. Peu importe sa nature, c'est cette étape de prise d'information qui permet sa caractérisation, son analyse et son contrôle. Le domaine des réseaux n'échappe pas à cette règle et les objets qui le composent auront besoin d'acquérir des informations sur leur environnement pour mieux s'y adapter. Dans cette thèse, nous nous intéressons au partage efficace de ces informations de mesures à des fins d'auto-analyse et d'évaluation distribuée de la performance.

Après avoir formalisé le problème de la mesure distribuée, nous nous consacrons dans un premier temps à l'organisation des échanges de mesures dans les graphes dynamiques. Nous proposons une nouvelle heuristique pour le consensus de la moyenne qui converge plus rapidement que celles de l'état de l'art. Dans un second temps, nous considérons des topologies plus stables pouvant utiliser des flux TCP comme moyen d'échange. Nous proposons un mécanisme d'ordonnancement de ces flux qui conserve le même comportement face à la congestion, tout en réduisant leur latence moyenne. Enfin, nous nous intéressons à l'information de mesure échangée. Nous montrons comment les nœuds peuvent superviser diverses métriques telles que la performance d'un système en se basant sur l'utilité de ses agents, et proposons une méthode pour qu'ils puissent analyser l'évolution de cette performance.

Abstract

With the massive rise of wireless technologies, the number of mobile stations is constantly growing. Both their uses and their communication resources are diversified. By integrating our daily life objects, our communication networks become dynamic in terms of physical topology but also in term of resources. Furthermore, they give access to a richer information. As a result, the management task has become complex and requires shorter response time that a human administrator can not respect. It becomes necessary to develop an autonomic management behavior in next generation networks. In any manner, managing a system requires essential steps which are : its measurement and its supervision. Whatever the nature of a system, this stage of information gathering, allows its characterization and its control. The field of networks is not the exception to the rule and objects that compose them will need to acquire information on their environment for a better adaptation. In this thesis, we focus on the efficient sharing of this information, for self-analysis and distributed performance evaluation purposes. After having formalized the problem of the distributed measurement, we address in a first part the fusion and the diffusion of measures in dynamic graphs. We develop a new heuristic for the average consensus problem offering a better contraction rate than the ones of the state of the art. In a second part, we consider more stable topologies where TCP is used to convey measures. We offer a scheduling mechanism for TCP flows that guaranty the same impact on the network congestion, while reducing the average latency. Finally, we show how nodes can supervise various metrics such as the system performance based on their utilities and suggest a method to allow them to analyze the evolution of this performance.

Table des matières

Chapitre 1

Introduction générale

1.1 Motivations

L'idée générale de l'Internet des Objets est de faire bénéficier l'homme d'une intelligence ambiante instanciée par les objets. Dans ce paradigme, les objets ont pour fonction d'assister l'homme dans son quotidien afin de rendre son environnement plus agréable à vivre. Ce concept est assez vaste pour couvrir divers thèmes de recherche à part entière. Il est donc étudié par différentes communautés de chercheurs, depuis la physique des matériaux pour la mise au point de sources énergétiques performantes jusqu'à l'étude sociale des nouveaux modes de vie en passant par les infrastructures de communications. Ces défis a priori d'horizons différents ont néanmoins tous un point commun : l'Information.

L'Information est le nerf de cette guerre technologique. Depuis sa production, jusqu'à son altération et son oubli, il faudra savoir gérer son stockage, sa diffusion, sa protection, son exploitation, mais aussi son impact sociétal. Dans ce monde où hommes, objets physiques et objets virtuels devront apprendre à cohabiter, la communication apparait donc comme un point central. Cette dernière permet de faciliter les interactions et ainsi d'améliorer la productivité. Pour ce faire, la communication devra être optimale du point de vue d'un contexte sans cesse en évolution. Autrement dit, les systèmes de communications devront être intelligents.

À ce jour, nous ne sommes qu'aux prémices des systèmes de communications intelligents. La littérature relative à l'Internet des Objets nous offre une myriade de cas d'utilisation très segmentés derrière lesquels se cachent de nouvelles contraintes et complexités techniques. Pour chacun de ces cas, une solution est spécifiquement développée, offrant une intelligence pour un domaine d'application précis. Ces solutions sont le

résultat d'un compromis entre des aspects de performances (débits, latence, consommation énergétique), de prix et de flexibilité (reconfiguration et capacité d'adaptation). Depuis quelques années, pour des raisons essentiellement économiques, la reconfigurabilité a pris une importance considérable dans le déploiement des systèmes de communication. Ce dernier aspect donne la possibilité à un système de modifier son comportement, autrement dit de se reprogrammer face à un changement de situation. Ce principe est applicable au niveau d'un terminal sans fil (cas de la radio logicielle ou SDR pour Software Defined Radio) comme à l'échelle d'un réseau (cas traité par les réseaux programmables ou SDN pour Software Defined Network).

En fait, la première étape d'adaptation au changement est de prendre connaissance de celui-ci. Pour atteindre un certain niveau de qualité, les systèmes informatiques complexes nécessitent d'avoir des outils de mesure et de supervision. Ces outils sont cruciaux pour mieux comprendre le comportement global d'un système. En effet, en identifiant dans un premier temps les mécanismes internes d'un système, il devient alors possible dans un second temps de l'améliorer en le modifiant. Avec l'avènement des communications sans fil, un degré de complexité supplémentaire a été introduit dans nos systèmes. Entre l'hétérogénéité des technologies et le dynamisme des topologies, de nouvelles métriques sont à prendre en compte et de nouveaux schémas de communication sont à considérer.

1.2 Positionnement et contributions

L'idée conductrice de cette thèse est d'introduire un nouveau cas d'utilisation pour l'Internet des Objets. Ce cas d'utilisation transverse est celui de la mesure distribuée des réseaux. Son principe repose sur l'utilisation et l'exploitation de l'information fournie par un maximum de sources. Ceci implique la participation des objets du quotidien dont les ressources peuvent être très différentes et les informations très variées. Face à cette complexité, la démarche suivie a été de décomposer notre problème en différentes étapes qui sont : l'acquisition d'une mesure ou d'un contexte, sa représentation, son partage, et son analyse. Nous apportons des contributions au niveau des trois dernières étapes.

Au niveau du partage efficace d'une information, nous étudions différents algorithmes de diffusion au sein d'un graphe afin de choisir le plus approprié à la topologie du réseau. En ce qui concerne les réseaux sans fil dynamiques, nous proposons un moyen de construire de nouvelles heuristiques de consensus, offrant des rapidités de diffusion supérieures à celles existantes. En ce qui concerne les topologies statiques, nous proposons un mécanisme d'ordonnancement de flux TCP qui conserve le même comportement face à la congestion, mais nous permet de donner la priorité à certains flux, par exemple

les flux critiques de mesures et de gestion ou encore les petits flux (ce qui réduit la latence au niveau applicatif).

Au niveau des informations acheminées et de leur représentation, le contexte est formalisé sous la forme d'un d'ensemble d'observations. Une observation est l'association d'un temps, d'un agent observateur et d'un fait. Cette formalisation permet notamment d'intégrer la satisfaction des observateurs et d'évaluer la performance d'un système. Il est alors possible de mener une recherche d'information pour comprendre l'évolution de cette performance. Afin de pouvoir mener cette recherche de manière distribuée au sein des éléments d'un système, une méthode d'analyse est développée. Cette méthode se base sur la construction de séries temporelles en agrégeant les observations par propriété et exploite la distance entre séries temporelles afin de permettre aux éléments du système de déterminer les faits paraissant les plus liés à la performance.

1.3 Structure du manuscrit

La suite de ce manuscrit s'articule autour de quatre chapitres principaux, un dernier chapitre fait office de conclusion et d'ouverture. Cette section donne un aperçu du contenu de chacun des chapitres afin de guider le lecteur vers l'information dont il a besoin.

Chapitre 2 - Contexte et problématique Ce chapitre traite des travaux existants relatifs à cette thèse. Il propose un tour d'horizon des différentes technologies de communication qui peuvent être rencontrées dans l'Internet des objets puis s'attarde sur les différentes solutions de mesures distribuées existantes. Il aborde ainsi les solutions adaptées aux réseaux de capteurs, mais aussi ceux prévus pour des infrastructures plus larges. Ce chapitre termine par la proposition d'une pile protocolaire pour la mesure distribuée. Les chapitres suivants se concentreront sur certains protocoles de cette pile protocolaire.

Chapitre 3 - Agrégation et diffusion des mesures dans un réseau Dans ce chapitre nous montrons comment découpler la représentation d'une information de sa diffusion. Puis en raisonnant sur des espaces vectoriels, nous montrons comment aborder le problème de la propagation d'une information comme un produit matriciel. En nous appuyant sur la théorie spectrale des graphes et sur celle du consensus de la moyenne, nous proposons de nouveaux schémas de communication pour les réseaux dont la topologie est fortement dynamique. Nous terminons par une étude des facteurs altérant la diffusion d'une information.

Chapitre 4 - Non-intrusivité et ordonnancement des mesures sur un lien Dans ce chapitre nous proposons un mécanisme permettant d'ordonnancer des flux de mesures qui utilisent le protocole de transport TCP de sorte à maitriser l'intrusivité des celles-ci et tout en donnant la priorité aux plus importantes. Nous donnons les conditions nécessaires et suffisantes pour qu'un découplage de l'ordonnancement et du contrôle de congestion tel que celui de TCP puisse se faire tout en garantissant de rester neutre vis-à-vis du réseau. Nous proposons une implémentation d'un tel protocole en espace utilisateur et montrons également que l'application d'un ordonnancement basé sur la taille des flux peut considérablement minimiser la latence au niveau applicatif (c.-à-d. dans notre cas au niveau des mesures)

Chapitre 5 - Évaluation et analyse distribuée de la performance Dans ce chapitre nous définissons un modèle de représentation du contexte et de la performance basé sur les systèmes multi-agents. Nous proposons un algorithme distribué exploitant les résultats des chapitres précédents, pour estimer une densité de probabilité à temps variable. Nous illustrons au travers de simulations comment la performance d'un réseau ainsi que la notion de contexte peuvent être mesurées en utilisant un tel algorithme. Enfin, nous présentons un algorithme d'extraction d'information afin de rechercher les facteurs les plus liés à l'évolution de la performance.

Ce manuscrit se termine par un dernier chapitre de conclusion dans laquelle nous rappelons le contexte des travaux de cette thèse, ses contributions ainsi que les perspectives et voies de recherches pour de futurs travaux.

Chapitre 2

Contexte et problématique

2.1 Perspective générale

2.1.1 Un bref historique d'Internet

Les réseaux de communications tels que nous les connaissons aujourd'hui doivent bien évidemment leurs existences au progrès de la physique des semi-conducteurs et des théories de l'informatique. L'invention du premier transistor bipolaire en 1947 fut probablement le point de départ de l'essor des télécommunications et de l'informatique moderne. Par la suite on observe une évolution conjointe de la microélectronique, de l'informatique et des télécommunications. En 1956, Bell propose le premier ordinateur à transistors (700 transistors et 10 000 diodes). Un an plus tard, le premier langage universel de programmation fait son apparition : le Fortran. Deux ans plus tard, l'assemblage de plusieurs transistors sur le même morceau de silicium donne naissance au premier circuit intégré de Texas Instrument. Cette même année, Bell invente le premier modem numérique pour la téléphonie. Très vite l'idée d'interconnecter les équipements a émergé. Leonard Kleinrock propose sa première théorie sur la commutation de paquet en 1961. Mais c'est seulement sept ans plus tard que la première conférence sur ARPANET [1] aura lieu. Si durant ce dernier siècle l'évolution des technologies a été initiée par le milieu académique elle a surtout été tirée et promue en grande partie par le monde industriel.

Depuis la publication en 1981 de la RFC 791 par l'IETF spécifiant le protocole Internet, le nombre de nœuds présents sur le réseau a explosé et la valeur économique de ce dernier est difficilement estimable. Il était difficile à l'époque d'imaginer l'ampleur qu'allait prendre ce réseau ni même l'arborescence des technologies qui allait en découler. De ce fait, de nombreux problèmes de sécurité, fiabilité, d'hétérogénéité et de passage à l'échelle se sont posés à postériori, lorsque le nombre d'utilisateurs était suffisant pour rendre le domaine des réseaux vulnérables. Bien heureusement, les organismes de standardisations ont su donner l'une des forces des systèmes actuels de communications, qui est celle d'une conception en *couche* et en *plan*. Aussi bien le modèle OSI que le modèle TCP/IP présentent cet avantage d'être modulaire, laissant la possibilité d'améliorer et de composer les différents services offerts par les protocoles de chaque couche. La supervision d'un tel réseau étant devenue cruciale des méthodes ont très vite été développées. L'essor des radiocommunications depuis la fin des années 90 a encore accéléré la croissance du réseau. Ces technologies radio équipent des terminaux qui pour la plupart sont situés aux extrémités du réseau. Ils partagent un même médium, ont des capacités physiques différentes (mesures plus riches, ressources moindres) et sont mobiles. Une partie

1. Advanced Research Projects Agency Network

de la topologie du réseau se voit dynamique et la supervision de certaines parties du réseau se trouve alors changée.

2.1.2 Une estimation des années à venir

On envisage à présent de connecter capteurs, actionneurs et objets pour former une *intelligence ambiante*. On attribue souvent la paternité de ce terme à Mark Weiser en référence à ces travaux de 1991 chez *Xerox* [1]. Une vision plus militaire, davantage axée sur les capteurs que sur l'interface avec l'humain est introduite au milieu des années 1990 par le projet *Smartdust* de l'Université de Berkeley [2]. Quelques années plus tard naîtront les fameuses *Berkeley Mote* [3] et d'autres technologies non communicantes, mais plus réduites et autonomes en énergie [4]. Le terme Internet des Objets est introduit à la fin des années 1990 par Kevin Ashton (MIT) qui envisage plutôt l'identification radio des objets et des personnes. Ces trois concepts sont à l'origine de nombreux travaux, et ont déclenché une euphorie chez les académiques. Les termes anglophones *Internet of Things*, *ambiant intelligence*, *ubiquitous computing* et *wireless sensors networks* sont devenus les nouveaux mots clefs justifiants de nombreux projets de recherche [5–7]. L'Internet des Objets est un terme suffisamment générique et vendeur pour que tout le monde puisse y contribuer sans que personne ne se comprenne. Il y a globalement trois grandes visions de l'Internet des Objets, l'une sera orientée réseaux, la seconde se voudra centrée sur l'objet lui-même et la dernière est centrée sur l'information et sa sémantique. Cette thèse considère la vision de [8] ou l'Internet des Objets est un réseau mondial d'objets interconnectés, adressés et basés sur des protocoles de communication standard. Parmi ces objets, certains auront une consommation énergétique et un prix bien supérieurs à ce que peut représenter un module de communication classique (voiture, gros électroménager). Pour ce type d'objet, l'optimisation matérielle du module de communication n'est pas forcément justifiée. En revanche pour une grande majorité des objets inertes, les ressources de communications seront limitées. Les usages les plus souvent évoqués couvrent très généralement les cas suivants.

Transports et logistique : Gestion des stocks, lutte anti-contrefaçons, billetterie, conduite assistée et cartes augmentées.

Militaire : Détection d'intrusion, surveillance des champs de bataille, équipements d'infanterie.

Santé et Télé-médecine : Identification des patients, auscultation à distance, capture et régulation des paramètres vitaux.

Environnement intelligents : Performance énergétique, confort et sécurité des bâtiments, études des écosystèmes et des paramètres environnementaux.

Vie personnelle et sociétale : Réseaux sociaux augmentés, vêtements intelligents, étude des biens et de leurs usages, réduction des pertes et vols.

Si d'un point de vue académique le domaine a un avenir prometteur, les industriels ne sont pas moins optimistes. Ces derniers s'accordent à dire que la taille de notre futur Internet est amenée à croître exponentiellement, Ericsson et Cisco annoncent un nombre de 50 milliards de terminaux en 2020 (contre 25 milliards en 2015) [9, 10]. D'un point de vue économique un gain de productivité de 21% par l'utilisation de l'Internet des Objets (soit près de 14 400 milliards de dollars entre 2013 et 2022) est prévu par ce dernier industriel [11]. De son côté, le cabinet de conseil McKinsey & Company dans un rapport de mai 2013 juge l'Internet des Objets comme la technologie la plus génératrice de valeur économique (devant le Cloud, le stockage d'énergie, les véhicules autonomes, et l'exploration avancée des ressources pétrolières et gazières) [12]. L'engouement industriel ne se ressent pour l'instant pas dans le quotidien des usagers lambda, car une majorité des projets sont adaptés aux besoins des entreprises. Pour une connaissance plus approfondie sur le sujet, les chercheurs et industriels pourront orienter leur recherche en étudiant [13], tandis que les curieux auront une approche visionnaire de l'Internet des Objets en lisant [14, 15] et une idée des projets entrepris en visitant [16].

D'une manière générale, l'Internet des objets n'en est qu'à ses débuts et les implémentations concrètes sont rarement mises au point sans l'appui des industriels. De nombreuses enquêtes ont déjà été menées sur le sujet et ont fait ressortir les problèmes suivants : harmonisation des standards, intégration de la mobilité, nommage des objets, adaptation des protocoles de communication, caractérisation du trafic et support de la qualité de service, supervision de l'infrastructure, sécurité de l'information (authenticité, confidentialité, droit à l'oubli) et conception de composants électroniques adéquats. Dans cette thèse, nous nous intéressons au volet supervision de l'infrastructure. Face aux changements à venir, nous cherchons à savoir comment adapter les méthodes de mesure et de supervision de nos réseaux.

La première section de ce chapitre propose un tour d'horizon des différentes méthodes existantes pour la mesure et la supervision des réseaux de communication. En section 2.3 nous explorons les technologies sans fil envisagées pour l'Internet des objets ainsi que leurs particularités. Nous terminons ce chapitre par une troisième section dans laquelle nous proposons une approche en couche pour la supervision distribuée. Dans les chapitres suivants, nous détaillerons certains aspects précis de ces couches.

2.2 Mesure distribuée et supervision des réseaux

Lorsque l'on souhaite comprendre le fonctionnement d'un système réel, sa mesure est sans doute son étape la plus importante. Peu importe sa nature (physique, biologique ou économique...), c'est cette étape qui permet d'acquérir les informations nécessaires à son étude. Toute caractérisation de phénomènes, élaboration de modèles ou validation d'un fonctionnement ne peut être faite que de manière ultérieure à cette phase. Le domaine de l'informatique n'échappe pas à cette règle, il en est de même pour les réseaux. Dans le cas des réseaux, mesurer et superviser sont deux tâches sensiblement identiques. Dans le cas de la mesure, on ne connait pas vraiment les mécanismes qui régissent le système et l'on cherche à en connaitre un peu plus. Une action immédiate n'est pas forcément prévue. En ce sens il est envisageable de procéder sous la forme de campagnes qui peuvent être exploitées par la suite. Dans le cas de la supervision, la structure du système est généralement connue et ses états définis. La tâche est de récupérer l'état du système à des fins de prise de décision. Cette prise de décision est du ressort de la gestion du réseau. Dans cette thèse nous utiliserons les termes mesure, supervision et gestion respectivement pour leurs pendants anglo-saxons *measure*, *monitoring* et *management*. Nous interchangerons parfois mesure et supervision dans le sens ou nous pouvons considérer la supervision comme la mesure de l'état courant du système. L'opération de mesure et de supervision d'un réseau est une tâche complexe, cette section donne un aperçu des outils, architectures, plateformes et techniques d'analyse existantes.

2.2.1 Métrique, méthodologies et outils standards

En terme de mesure et supervision, la particularité d'Internet est d'être un système distribué vaste qui appartient à différentes entités administratives. On pourrait d'un premier abord le comparer à celui des transports routiers, pour lequel on chercherait à connaitre dans chaque ville l'état des routes qui les séparent. Néanmoins, ce qui différencie fondamentalement ces deux systèmes c'est que dans le cas d'Internet l'information de mesure passe par le système lui-même. L'opération de mesure ou la simple communication de son résultat peut être intrusive. D'autre part l'échelle de temps des phénomènes considérés peut nécessiter une synchronisation précise des horloges.

La notion de métrique

On appelle communément une métrique un attribut qui est mesuré sur un système qui permet de mieux le décrire, de mieux le caractériser. Certaines métriques réseau sont clairement définies par le groupe de travail IPPM (IP Performance Metrics) de l'IETF.

Elles sont qualifiées de métriques de performance, car elles sont directement en lien avec le bon fonctionnement du réseau. Le groupe IPPM définit notamment les métriques suivantes : La connectivité (RFC2678), le délai (RFC2679), les pertes (RFC2680), le temps d'aller-retour (RFC2681) les schémas de pertes (RFC3357) la variation du délai ou gigue (RFC3393), des métriques pour les flux périodiques (RFC3432). Ces métriques concernent donc la couche réseau. Il est important de noter que l'on peut considérer des métriques à différents niveaux du modèle OSI. Au niveau de la couche liaison de donnée, des métriques similaires peuvent être utilisées, dans le cas des communications sans fil. Ces métriques peuvent être enrichies par des paramètres physiques tels que le RSSI, le SNR ou encore l'EVM [17]. Au niveau applicatif comme dans le cas d'une architecture de type SaaS (pour Software as a Service), des métriques telles que le temps d'accès au "premier octet" ou temps de réponse, le temps de chargement d'une page ou encore le temps de transaction sont utilisées [18]. Enfin il devient de plus en plus courant d'essayer de faire participer l'utilisateur dans l'évaluation de la performance. On parle alors de qualité d'expérience (QoE). Cette métrique est difficile à mesurer sur de vrais réseaux et des études tentent de faire le lien entre le ressenti d'un utilisateur et des métriques plus objectives telles que présentées précédemment [19]. Les métriques sont souvent liées à la manière dont elles ont été acquises : *passivement*, *activement*, ou de manière hybride.

Mesure passive et mesure active

La mesure passive fait référence à la simple observation du système. Au niveau réseau, certaines méthodes consistent en la collection d'information sur les paquets traversant un ou plusieurs points d'observation (par exemple pour mesurer la latence). Cette mesure peut être considérée comme non intrusive à partir du moment ou les points d'observation ne communiquent pas (le résultat de la campagne de mesure est exploité plus tard). Si ce n'est pas le cas, la communication entre les points d'observations peut venir perturber la mesure. Lorsqu'il y a plusieurs observateurs, la synchronisation de ces derniers est d'autant plus importante que les délais sont brefs. Pour donner un ordre de grandeur, mesurer le délai entre deux machines respectivement à Toulouse et Londres désynchronisées de 20ms est tout aussi précis que de regarder l'heure à l'aéroport de Blagnac, prendre l'avion puis à l'arrivée, se baser sur l'heure de Big Ben pour calculer la durée de son trajet. Par opposition, dans une méthode de mesure active, des données sont explicitement générées pour servir de base à la mesure. La mesure active est donc intrusive de par son principe de fonctionnement. Les données générées sont conçues de sorte à contenir des informations utiles à la mesure (comme un numéro de séquence, un estampillage temporel, etc.). La génération volontaire de trafic peut par exemple servir à tester la bande passante disponible. L'analogie dans les réseaux de transport reviendrait

à remplir les avions d'une compagnie d'employés pour tester leurs limites de fonctionnement. Bien évidemment cette méthode mobilise une partie de la flotte pendant le temps de l'essai aux dépens de réels usagers, elle ne peut donc être utilisée que ponctuellement.

Quelques outils de mesure et de supervision

Dans la panoplie d'outils disponibles pour la mesure et le monitoring, on peut différencier le matériel du logiciel. L'usage d'un matériel spécialisé peut être nécessaire lorsque l'on cherche à faire de la mesure précise, tandis que l'usage d'un matériel standard couplé à une suite logicielle peut être suffisant pour un monitoring qui se contente de vérifier ponctuellement l'état du réseau. Au niveau du matériel de mesure, les cartes les plus connues sont issues du projet DAG de l'université d'Auckland [20]. Plus récemment, l'université de Cambridge propose des cartes NetFPGA dont la conception est libre [21]. Ces types de cartes offrent une synchronisation fine et permettent un traitement dit *offload* allégeant la charge processeur. En ce qui concerne les outils logiciels, on citera *iperf*, comme outil de mesure active de la bande passante et *traceroute* pour mesurer la connectivité d'un réseau. Le nombre de solutions de supervision réseau (aussi bien commerciales que libre) étant conséquent, nous n'en dresserons pas la liste, nous nous contenterons de citer la solution qui nous semble la plus populaire : *ntopng* et renverrons le lecteur vers [22] pour une liste plus complète. Bon nombre de ces outils sont basés sur la librairie `libpcap`, notamment pour les analyseurs de paquets et utilise le protocole SNMP pour la collecte de métriques telles que les compteurs de paquets au niveau des interfaces réseaux.

2.2.2 Plateforme et architectures

Afin de mieux comprendre et de mieux caractériser le comportement de nos réseaux, différents types de plateformes ont été montées. Nous les avons arbitrairement classées en deux catégories : celles rattachées à une infrastructure de production, et celles créées spécifiquement pour le test.

Plateformes en environnement de production

On appelle plateforme en environnement de production, un réseau ayant de réels usagers et pour lequel des mesures ont été faites. Ces environnements sont rares et précieux, les résultats de mesure font en général office de référence pour de nombreux travaux de recherche. En effet la mise en place de tests de performances et de mesures précises sans perturber les utilisateurs est comme nous l'avons dit précédemment une tâche complexe.

Au-delà des aspects techniques, d'autres aspects légaux peuvent aussi entrer en ligne de compte. Au niveau des plateformes de mesure les plus connues, on retrouve celle du laboratoire CAIDA [23]de San Diego qui met à disposition des traces relevées uniquement par des techniques passives. Au niveau des infrastructures usant de techniques à la fois actives et passives, on retrouvera les plateformes du réseau Internet2 [24] aux États-Unis et celles de RIPE[25] au niveau européen. Relativement sur de gros liens réseau, le volume d'information que ce type de plateformes voient transiter est conséquent, par exemple, sur la journée du 2 mars 2015, près de 1,5 petaoctet a transité sur le réseau I2 [26]. Les informations fournies par ce genre de réseaux sont généralement offertes à une granularité de l'ordre de la minute. Il est aussi parfois possible d'accéder à des traces complètes, mais sur des périodes temporelles d'une heure par exemple ou encore agrégées par flux. En terme de plateforme à grande échelle, RIPE-Atlas [27] et SamKnows [28] sont deux solutions permettant à des utilisateurs lambda de participer aux campagnes, agrandissant de manière considérable l'étendue du système de mesure. Alors que le premier nommé permet aux utilisateurs de réaliser leurs propres campagnes de mesures, le second s'ouvre aux terminaux mobiles par le biais d'applications, et entrevoit à terme de s'appliquer à l'Internet des objets. Si ces campagnes s'adressent à des réseaux de grandes échelles, elles ne permettent pas de mesurer les phénomènes liés à l'accès au médium et, dans le cas du sans-fil à des métriques de plus bas niveau. À ce niveau-là, c'est plutôt du côté des plateformes expérimentales qu'il faut se tourner.

Plateforme expérimentale

On appelle plateforme expérimentale un réseau qui a été spécifiquement conçu pour effectuer des tests et des expérimentations. Le but de ces plateformes est de pouvoir maitriser l'environnement de mesure pour pouvoir comparer et valider différents protocoles. Les plateformes pour les réseaux sans fil ont fleuri ces dernières années. L'une des raisons est notamment de pallier le manque de réalisme des simulateurs pour le domaine. Ces plateformes sont lourdes à mettre en place [29]. Une fois de plus les coûts sont mutualisés entre les institutions [30], ce qui rend parfois difficile la maitrise des paramètres environnementaux, ces plateformes n'étant souvent accessibles qu'à distance. Il est possible de trouver des plateformes expérimentales de nature très différentes en termes de nombre de nœuds, de standards de communications, d'environnements physiques et de configuration. Parmi ce genre de plateformes, il existe Emulab [31], ORBIT[32], CREW project, ASSERT[33]. Il est possible de différencier ces plateformes en fonction du niveau de contrôle dont elles disposent sur le médium physique. Contrôler le médium de communication est coûteux et réduit de fait la taille des plateformes et les rend spécifiques

[34, 35]. Enfin, sur un thème un peu en marge des aspects techniques de la communication, le MIT Media-Lab a mis en place des expérimentations bien plus proches de ce que l'on attend de l'Internet des objets, à savoir celle du *Reality-Mining* [36], précurseur de l'ère bien en vogue du Big-Data. Leurs études sont plus centrées sur l'humain et sur son mode de vie au travers des traces relevées sur leurs terminaux mobiles. Nous pensons que c'est vers ce genre de plateforme qu'il faut tendre. Dans le cas des réseaux filaires, il a été possible d'analyser le type de trafic en fonction des heures de la journée pour pouvoir s'y adapter [37]. Dans le cas des réseaux sans fil, c'est à une échelle plus fine qu'il faudra observer les habitudes sociales pour optimiser les communications.

2.2.3 Techniques d'analyse et de supervision

L'acquisition des données n'a un intérêt que si celles-ci sont analysées et exploitées. Les données peuvent servir à mieux comprendre le trafic, ce qui peut s'avérer crucial pour un opérateur. Ces derniers cherchent notamment à (1) inférer l'état des réseaux voisins et celui du trafic, (2) se protéger des utilisateurs mal intentionnés, et (3) comprendre les phénomènes qui régissent les communications et impactent leurs performances.

Inférence des réseaux et tomographie

La mesure directe d'un phénomène est parfois coûteuse, ou impossible si celui-ci apparait sur l'infrastructure d'une autre entité administrative. Une approche est alors de considérer un modèle "boite noire" pour lequel il est possible d'observer les entrées et les sorties (et dans certains cas maitriser les entrées) et d'essayer d'en deviner la boite blanche. Appliqué au réseau, cela peut se matérialiser par l'estimation de l'état des liens en se basant sur les mesures de bout en bout ([38]) ou encore par l'estimation des besoins de bout en bout basée sur l'occupation des liens ([38]). En fonction du contexte, on cherchera à retrouver le taux de pertes sur les liens, leurs délais, la topologie sous-jacente, ou encore la matrice de trafic. Vardi est le premier à avoir introduit le terme tomographie en s'intéressant à ce dernier type de problème dans [39]. Par la suite différentes solutions ont été développées [40, 41] pour estimer la matrice de trafic d'un réseau en se basant sur la matrice de routage et le comptage de paquets au niveau de chaque lien. Outre ces problèmes liés à la charge du réseau, d'autres études ciblées s'intéressent à des aspects plus administratifs tels que la détection de filtres [42] ou l'adoption du protocole IPv6[43].

Classification du trafic, détection d'anomalie et sécurité

En terme de sécurité et de sureté des réseaux, l'analyse du trafic est une voie très largement exploitée. En ce qui concerne la détection d'attaques, le trafic peut-être confronté à un ensemble de signatures afin de s'assurer qu'il n'est pas malicieux, ou anormal. Afin de se passer de l'étape fastidieuse de génération de signature, Casas et coll. ont montré dans [44] l'efficacité des techniques de partitionnement (ou *clustering*) pour la construction de nouvelles règles de filtrages sans connaissance a priori. L'usage du data-mining pour la sécurité et la sureté cherche en général à réduire le taux de faux positifs des détections d'intrusion et d'anomalie sur le réseau (panne, surcharge, etc.). Si les auteurs de [45] mentionnent que le data-mining est un outil précieux, ils précisent que l'analyse humaine, tout particulièrement dans le choix des données reste d'une qualité supérieure. Enfin des campagnes de mesures et d'analyse plus proactives ont pour but d'étudier l'exploitation de certaines failles et vulnérabilités. C'est le cas de l'approche passive des pots de miel [46] visant à attirer les utilisateurs malintentionnés, et des approches plus actives telles que celles de [47] et [48] qui n'hésitent pas à corrompre certains systèmes pour en mesurer les limites.

Compréhension des mécanismes régissant le trafic

La compréhension des règles qui régissent les performances de la communication est la motivation principale des mesures. Par exemple, la source de *reset* TCP anormaux a été mise en évidence dans [49]. Avec la popularité des réseaux sans fil, le géant AT&T s'est penché sur l'usage du data-mining pour l'analyse de sa propre infrastructure sans fil [50], tandis que dans [19], les auteurs expliquent la relation qu'il existe entre le temps de réponse d'un serveur, le temps d'aller-retour et la satisfaction d'un ensemble d'utilisateurs mobiles.

De par l'histoire d'Internet, une grande partie des infrastructures de mesures sont focalisées sur les technologies filaires avec une granularité bien souvent de l'ordre du flot. Les lecteurs intéressés par le sujet peuvent consulter les références suivantes : [51–57]. D'un autre coté, les réseaux sans-fil constituent une fraction croissante des terminaux et font de plus en plus l'objet d'analyses et d'études. La mesure et la compréhension de ces infrastructures, mais aussi des topologies qui leur sont associées ne doivent pas être laissées en marge. La section suivante s'intéresse donc à ces technologies qui sont candidates pour former l'Internet des objets à venir, et avec lesquelles il faudra composer lors des campagnes de mesures.

2.3 Internet des Objets et radiocommunications

L'Internet des Objets ne sera pas un réseau exclusivement composé de lien de radiocommunications. Néanmoins ces types de liens connecteront une grande partie des segments terminaux. Les radiocommunications permettent la mobilité des nœuds ce qui présente de nombreux avantages applicatifs. Cependant elles impliquent des complications techniques en termes d'accès au médium, de fiabilité, mais aussi d'autonomie énergétique. Cette section traite de ces spécificités du sans-fil et de ses standards afin de mettre en exergue leur hétérogénéité et leur sensibilité aux paramètres environnementaux.

2.3.1 Architecture des liaisons de communication sans fil

Le partage d'un médium hertzien

Le médium de communication hertzien est complexe. La couche physique (PHY) du modèle OSI y extrait un canal de communication qui doit être partagé par plusieurs stations. Il existe pour cela trois grandes approches de partage qui sont les multiplexages fréquentiels (FDMA), temporels (TDMA) et par répartition de code (CDMA). Il est possible de les combiner entre elles, mais aussi de faire de la réutilisation spatiale. Le multiplexage fréquentiel attribue une bande de fréquence à chaque élément de la liaison. L'accès CDMA attribue un code orthogonal aux stations. Ce code permet d'étaler le spectre de la communication en utilisant une technique de saut de fréquence (FHSS) ou bien par séquence directe (DSSS). L'accès TDMA consiste à diviser temporellement le canal et attribuer un temps de communication à chaque station. Ces méthodes d'accès sont généralement sans contention, elles nécessitent un coordinateur pour gérer l'attribution des ressources et s'adaptent bien aux réseaux mobiles, pour lesquels il existe une infrastructure où les utilisateurs avait historiquement tous le même besoin de communication (appel vocal). Enfin il est important de remarquer que si le médium est partagé, il est naturellement propice à de la communication en mode diffusion, ce que nous exploiterons par la suite.

L'importance de la couche d'accès au médium (MAC)

Dans le cas des réseaux locaux et personnels, l'accès TDMA est prédominant, mais une souplesse d'accès est souvent permise avec la notion de contention. Celle-ci permet à deux stations d'accéder au médium sans le réserver. Le problème associé est celui de l'accès concurrent pour lequel, si deux stations émettent leur trame en même temps, celles-ci sont perdues. Ce problème traité pour les réseaux filaires avec le protocole

CSMA/CD, se complexifie dans les réseaux sans fil pour deux raisons : (1) détecter une collision causée par une station voisine est limité par le phénomène d'éblouissement radio et (2) la collision peut être causée au niveau du récepteur par une station cachée de l'émetteur. Dans ce cas présent, une collision est coûteuse en temps, et en énergie ce qui réduit considérablement les performances de la liaison. Ainsi divers mécanismes de coordination se sont développés, notamment l'envoi de trames courtes de demande d'autorisation d'envoi (Request To Send / RTS) et de confirmation (Clear to Send / CTS). Il existe aussi des couches d'accès au médium (MAC) hybrides, pour lesquelles un coordinateur définit des périodes d'accès avec contention et des périodes de temps allouées dynamiquement. Ces couches permettent notamment d'offrir différentes qualités de service en fonction des applications visées. Une fois de plus la gestion de l'accès au médium fait face à un compromis. Entre le besoin d'un coordinateur, la flexibilité de la gestion des ressources et la complexité d'implémentation, de nombreuses propositions ont été faites, notamment pour les réseaux de capteurs, alliant techniques d'accès sans mécanismes (Alhoa) ou un peu plus évolués (slotted-Aloha) et prenant parfois en compte la mise en veille des stations.

Spécificité d'un lien radio

Une fois le canal obtenu, le risque d'interférences est réduit, cependant le service de communication offert par la couche PHY est loin d'être parfaitement caractérisé. Dans le cas des communications filaires, le modèle d'un lien de communication est bien maitrisé, le débit binaire est connu, fixe, tout comme le taux d'erreurs binaire. Dans le cas du sans-fil le modèle d'un lien peut vite s'avérer complexe. La qualité du lien de communication va dépendre de la conception des émetteurs-récepteurs, de la qualité de leurs composants, de paramètres tels que leur fréquence de fonctionnement, mais également du milieu environnant (zone de Fresnel et multitrajets, vitesse relative des stations, etc.)

Paramètres de communications et contraintes matérielles

Le compromis principal des couches MAC et PHY met en jeu le débit, le délai, la zone de couverture, l'énergie, et le taux d'erreur trame. Les possibilités d'actions au niveau de la couche PHY sont entre autres : le choix d'un codage canal, d'une modulation, d'un rythme symbole, d'un code correcteur, d'une bande de fréquence, mais aussi d'autres paramètres physiques comme les puissances d'émissions (ou d'amplification en réception) et le choix d'un diagramme de rayonnement d'antenne. Les possibilités d'actions au niveau MAC sont principalement la méthode d'accès choisie, et dans le cas d'un accès avec contention, la politique d'acquittement et de retransmission. La configuration d'un

équipement va donc impacter la qualité de sa communication, mais également celle de ceux qui partagent son médium. Ainsi, il est important pour un terminal de connaitre l'occupation spectrale de la zone géographique dans laquelle il se trouve et ses capacités d'adaptation sont des atouts pour l'optimisation des communications sans fil. Toutefois, celles-ci ne sont pas sans contreparties, l'adaptabilité a un coût. Il existe trois grands types de technologies pour la conception d'un émetteur-récepteur. Les circuits dédiés ou ASIC (Application Specific Integrated Circuit), les circuits logiques programmables tels que les FPGA (Field Programmable Gate Array) et les architectures classiques avec processeur. Ces technologies sont combinées et assemblées sur circuits imprimés pour obtenir un système complet. Chacune d'entre elles offre un compromis entre adaptabilité, performance, taille et coût de production. Des solutions de plus en plus hybrides apparaissent comme des systèmes sur puces (SoC) programmables, des FPGA intégrant des microprocesseurs en dur, mais d'une manière générale, plus l'on se rapproche de la couche physique, plus il est difficile d'être flexible.

La réalisation d'un lien sans fil nécessite la mise au point d'un grand nombre de détails techniques qu'il est nécessaire de coordonner. De nombreux paramètres doivent être choisis et définis. Les standards servent entre autres à harmoniser ces paramètres.

2.3.2 Les acteurs de l'Internet des Objets et leurs standards

La coordination des standards est l'une des problématiques de l'Internet des Objets. Outre les réseaux de capteurs et la radio-identification, il existe d'autres applications nécessitant des besoins en qualité de service différents (réseaux véhiculaires, réseaux multimédias d'une habitation, réseaux d'accès métropolitains). Afin d'optimiser leurs efforts, chercheurs et industriels se réunissent par centre d'intérêt au sein d'organisations pour favoriser le développement d'une solution apte à couvrir les besoins d'un domaine précis tout en la faisant cohabiter avec les technologies voisines. Pour un même domaine, il est possible de voir émerger des technologies concurrentes, bien souvent l'une d'elles devient le standard de fait et les autres tombent en désuétude.

Organisations meneuses et leurs implications

Parmi les organisations génératrices de standards, certaines sont reconnues formellement par les gouvernements (IUT, ETSI . . .) et ont le mot final sur ce qui doit se faire en terme de radiocommunication. Elles s'appuient sur le travail amont de groupes plus réactifs et spécialisés tels que ceux du comité ISO/IEC ou encore de l'ECMA, l'IETF et l'IEEE. Ce sont ces entités-là qui tentent d'harmoniser les discours afin de faire converger les standards, elles travaillent en symbiose avec les alliances d'industriels qui valident et

promeuvent différentes solutions techniques. On retrouve facilement trois grands types de standards qui s'organisent autour de la radio-identification, des réseaux à faibles ressources énergétiques et des réseaux à haut débit de communication.

Les standards pour la radio-identification (RFID)

Le principe de la radio-identification est de fournir un identifiant et une mémoire à un objet au sein d'un composant électronique appelé tag. Un tag se contente de répondre à un lecteur sur l'initiative de ce dernier qui peut l'interroger pour l'identifier, lire et selon modifier sa mémoire. Le lecteur fournit l'énergie aux tags afin que ceux-ci puissent répondre. La communication peut être de type point (lecteur) à multipoints (tags) pour une portée de quelques mètres. C'est le mode d'opération initial utilisé pour la logistique et la gestion des biens (code barres électronique, antivol, librairie, blanchisserie industrielle, etc.), on parlera de RFID. La communication peut aussi être de type point à point pour un échange de données à quelques centimètres, ce que l'on désignera par NFC (de l'anglais Near Field Communication). Ces technologies sont standardisées sur plusieurs niveaux qui sont : l'aspect matériel des tags, les couches basses de communication, le format de données utilisé et son traitement à grande échelle, mais aussi la gestion de certains équipements et l'intégration au réseau Internet. Les standards RFID et NFC diffèrent sur plusieurs points et sont rarement compatibles. Néanmoins la technologie NFC bénéficie historiquement des travaux de standardisation sur la RFID.

Les standards pour les réseaux sans fil à faibles ressources

Certaines applications ont des besoins de communication très réduits, mais nécessitent une bonne durée de vie pour être rentables. Une optimisation énergétique est alors faite à différents niveaux du modèle OSI : au niveau physique dans le choix de la modulation, au niveau de la méthode d'accès au médium, mais aussi au niveau des protocoles de routage pour lesquels les métriques se soucient du bilan énergétique. Les industriels proposent souvent des solutions techniques répondant à un besoin précis, mais non compatibles entre elles. On peut citer les technologies Zigbee, WirelessHart, ou encore Bluetooth, Ant+ et Dash7 qui proposent toutes une pile protocolaire complète, mais différente au niveau de la couche réseau.

Les systèmes Zigbee et WirelessHart sont basés sur le standard IEEE 802.15.4 et définissent principalement les couches réseaux et applicatives pour les réseaux de capteurs et d'actionneurs. Les technologies Bluetooth et Ant+ sont plutôt dédiées aux réseaux personnels et définissent aussi les couches basses. Le standard Bluetooth de 2010 spécifie un

mode *Low Energy* adapté à ces besoins. L'accès complet aux standards pour ces technologies nécessite souvent une contrepartie financière. Il y a néanmoins la recherche d'un consensus vers le réseau IPv6. L'IETF avec 6lowPan et ROLL ainsi que l'IPSO Alliance tentent d'intégrer ces technologies au sein du protocole IPv6. Le groupe 6lowPan a défini les mécanismes d'encapsulation et de compression d'entêtes IPv6 pour l'utilisation du standard IEEE 802.15.4 tandis que le groupe ROLL traite des problématiques de routage. De son côté, l'alliance Zigbee propose dans un standard de 2013 (Zigbee IP) une solution sans passerelle intégrant les standards de l'IETF.

Les standards pour les réseaux sans fil à forts besoins de trafic

La problématique de l'énergie touche tous les types de réseaux, cependant certaines applications sont plus préoccupées par d'autres critères de qualité de service (débit, délai, gigue, couverture). C'est le cas des réseaux mobiles, véhiculaires, locaux, ou encore multimédias.

L'alliance WiMedia est parvenue à standardiser le domaine des réseaux personnels à haut débit là où l'IEEE 802.15.3a s'est essoufflée. Elle développe un standard pour des communications à 3,1 GHz et 10,6 GHz pour des débits allant de 53,3 à 1024 Mb/s. Sa synchronisation fine offre également des propriétés d'évaluation de distance entre les équipements. Ce sont les standards IEEE 802.11 qui structurent le domaine des réseaux locaux sans fil, l'alliance WiFi se base essentiellement sur ces normes. En mode infrastructure, une station se connecte de point d'accès en point d'accès (ou BSS pour Basic Service Set) au sein d'un même ESS (Extended Service Set). Un mode pair-à-pair existe également. Un grand nombre d'amendements a été apporté afin de couvrir : une augmentation du débit (n), la gestion de la qualité de service (e), ou encore les réseaux véhiculaires (p). Le standard actuel IEEE 802.11-2012 regroupe une partie de ces amendements. Il possède déjà cinq amendements dont le plus connu est celui de 2013 (ac) qui porte son débit maximum à 1300Mb/s pour une portée de 70 à 250 mètres selon l'environnement (intérieur ou espace libre).

Enfin, la bataille du haut débit pour les réseaux mobiles et métropolitains 4G-LTE semble avoir été gagnée par les standards du groupe 3GPP au profit de ceux du forum Wimax. La communauté réfléchit déjà à la 5G dont l'enjeu principale est la densification massive du réseau. À ce titre, les ondes millimétriques sont envisagées, mais à 60GHz, la réalisation du moyen de transmission constitue un verrou technologique. S'il est difficile d'être exhaustif sur le domaine, ce tour d'horizon rapide montre combien les technologies radio sont diverses, avec des schémas de communication différents des réseaux filaires. On pourra également garder un œil sur des technologies complètement alternatives comme

des techniques de modulation à bande étroite [58], le Lifi [59], ou encore la modulation du champ électromagnétique ambiant [60].

2.4 Un modèle de supervision pour l'Internet des objets

Comme nous l'avons constaté dans les sections précédentes, la mesure et la supervision des réseaux sont nécessaires à leur bonne évolution et à leur bon fonctionnement. Ce sont des procédés qui d'une part permettent d'adapter les standards aux nouveaux besoins de communication, mais c'est également une phase primordiale en terme d'autonomie des systèmes. Nous avons aussi vu que les communications sans fil représentent une part importante des technologies de communication. Ce type d'accès à l'information change la donne dans le domaine de la supervision pour plusieurs raisons :

- Les nœuds sont mobiles et la topologie du graphe de communication peut évoluer fortement au cours du temps.
- Les besoins applicatifs se diversifient, au même titre que les ressources disponibles pour chaque nœud.
- L'accès à de nouvelles métriques environnementales et sociales est devenu possible.
- L'accès à de nouvelles métriques physiques sur le médium est devenu nécessaire.
- Les piles protocolaires se complexifient et se spécialisent.
- Le trafic au niveau des liens d'accès dépend de phénomènes très locaux.

Afin de prendre en considération ces changements, nous proposons d'intégrer le maximum de technologies dans l'infrastructure de supervision de nos futurs systèmes de communication. Pour ce faire nous devrons composer avec les différentes capacités et les attentes de chacun des équipements depuis la plus petite puce RFID jusqu'au serveur de calcul. L'architecture devra donc être modulaire et intégrer les objets du quotidien.

2.4.1 Intégrer l'objet lambda : lorsque le contexte devient une métrique

L'interconnexion des objets du quotidien est souvent laissée pour compte, car le fait d'être connecté ne leur donne pas une valeur ajoutée immédiate. C'est effectivement le cas si l'on considère qu'un objet communique uniquement pour améliorer ses propres fonctionnalités. Néanmoins, un objet peut communiquer pour assister ses voisins, par exemple en signalant simplement sa présence et en partageant les mesures dont il a la connaissance. En enrichissant les métriques, il est possible de migrer peu à peu d'un *simple* délai, vers une métrique plus complexe telle qu'un *contexte* de communication.

Premièrement, échanger sur le contexte permet de mutualiser certaines ressources de calcul. Par exemple un émetteur radio n'aurait pas besoin d'inférer les bandes de fréquences libres si son plus proche voisin possède déjà l'information. Deuxièmement, en permettant aux objets de mesurer leur contexte, nous leur permettons de mieux répondre à nos besoins : lorsque nous nous promenons dans la rue, nous sommes en permanence informés par de la signalisation, des panneaux publicitaires, des post-it, etc. Nos terminaux mobiles pourraient faire de même en balayant par exemple des étiquettes RFID. Enfin la motivation sans doute la plus actuelle, est de pouvoir analyser le fonctionnement de notre société en fouillant une masse de données issue de ces mesures. Les objets qui nous entourent étant la plupart du temps le reflet de nos vies, les observer peut alors s'avérer être pertinent.

Avant d'atteindre ce stade, nous pourrons sans doute mieux comprendre le fonctionnement de nos systèmes en terme de performance. Dans le cas des réseaux, le trafic généré par un nœud dépend de son contexte d'utilisation. Surveiller le changement de contexte pour chaque nœud peut permettre à un opérateur de mieux prédire le trafic à venir et de mieux comprendre le comportement de son infrastructure.

Une telle démarche passe par une bonne formalisation du contexte et de ses métriques, un partage efficace de ces informations, et enfin des outils d'analyse permettant de faire le lien entre la performance (notion que nous définirons) et les mesures effectuées.

2.4.2 Une architecture modulaire adaptable à des ressources hétérogènes

De par l'étude que nous avons menée jusqu'à présent, nous avons dégagé plusieurs grandes taches autour de la mesure :

1. l'acquisition
2. la synchronisation des horloges
3. la représentation et le stockage
4. la diffusion et le partage efficace
5. l'analyse

En fonction des besoins de mesures et des ressources disponibles, les fonctionnalités précédemment décrites pourront avoir des implémentations très différentes. Les paramètres impactant grandement les choix de solutions sont :

a. la granularité de l'information
b. la précision temporelle des mesures
c. la topologie du réseau
d. les destinataires de l'information

e. la capacité d'analyse et de stockage des nœuds

Par exemple, le cas d'une campagne de mesure sur un réseau fixe, dont les résultats seront exploités a posteriori par une seule machine diffère totalement d'une opération de supervision de réseau mobile sans fil pour laquelle chacun des nœuds doit être informé en temps réel de l'état du réseau. Pour pouvoir faire face à des situations aussi différentes tout en gardant un bon niveau de compatibilité, nous nous appuierons sur ce qui a fait la réussite des modèles de communication actuels : une conception modulaire.

Les fonctionnalités énoncées peuvent être séparées en différents services. L'implémentation d'un service peut être spécialisée pour répondre à un besoin particulier. En associant plusieurs services, nous serons à même de construire la solution de mesure ou de supervision adaptée à la situation. L'architecture que nous proposons est structurée comme dans la figure 2.1.

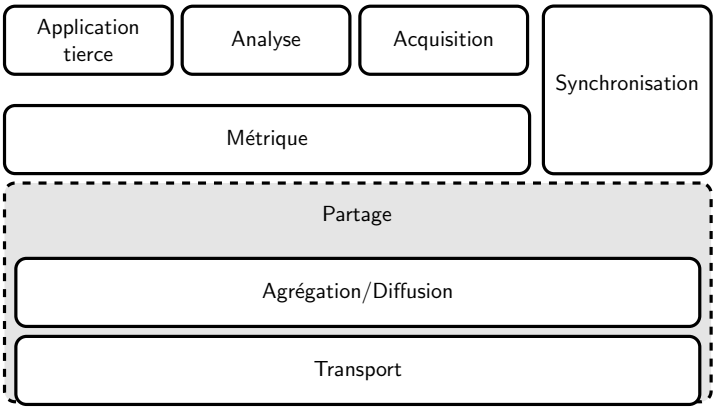


FIGURE 2.1: Modules pour la mesure distribuée

Le module de *Métrique* peut se voir comme un bus pour les autres modules. Il permet de stocker l'information brute acquise par le module *Acquisition*, l'information traitée par le module *Analyse* et l'information agrégée par le module de *Partage*. Le temps étant une métrique particulière dans les systèmes dynamiques, nous considérons son partage dans un module *Synchronisation* séparé. En fonction de la capacité d'un nœud, celui-ci pourra n'implémenter qu'une partie de ces modules. Par exemple pour un simple élément relais, seul le module de partage est nécessaire. Si cet élément est sensible, il pourra en plus supporter un module d'acquisition et de synchronisation. Le ou les éléments ayant le plus de ressources seront capables d'analyser les données afin de générer de nouvelles métriques qui seront à leur tour stockées et propagées. Dans cette thèse nous nous sommes intéressés aux modules *Partage*, *Métrique* et *Analyse*. Nous supposerons que nous sommes capables d'acquérir des mesures datées, sur l'environnement du réseau. Ces mesures auront pu être acquises aussi bien de manière passive (capture de trafic et valeurs de capteurs) qu'actives (tests de connectivité, etc.).

Nous avons décomposé le module de partage en 2 sous-couches appelées respectivement *agrégation/diffusion* et *transport*. La couche *agrégation/diffusion* sera chargée de fusionner les informations reçues du réseau, tandis que la couche *transport* sera responsable de la politique de communication avec les nœuds voisins. La sous-couche *transport* sera adaptée à la topologie et au type du réseau sous-jacent alors que la sous-couche *agrégation/diffusion*, sera plutôt adaptée aux métriques considérées.

Dans le chapitre 3 nous détaillerons la sous-couche diffusion. En modélisant la diffusion d'une information par un produit matriciel, nous montrerons comment la découpler de la sous-couche de transport. Nous présenterons différents types d'implémentation et les situations pour lesquelles elles sont le mieux adaptées. Nous traiterons le cas des topologies fortement mobiles pour lesquelles nous proposerons de nouveaux algorithmes. Enfin nous ferons une étude des facteurs impactant cette couche de diffusion.

Dans le chapitre 4 nous traiterons de la sous-couche transport. Plus particulièrement, nous détaillerons l'ordonnancement des flux de mesures transmis via le protocole TCP. Par le découplage des mécanismes de congestion et d'ordonnancement des flux de mesures, nous montrerons qu'il est possible de rendre prioritaires certains flux tout en conservant le même débit. En nous basant sur la taille des flux pour établir un ordre de priorité, nous montrerons qu'il est possible à débit égal, de réduire le temps de transfert moyen des flux.

Dans le chapitre 5 nous donnerons des exemples de modules *Métrique* et *Analyse*. Nous proposerons un modèle basé sur les systèmes multi-agents définissant une métrique de performance et un modèle de contexte pour les systèmes distribués. La métrique de performance prendra en compte la fonction utilité de chaque nœud du réseau, tandis que le contexte sera décrit par la trace des événements du système. Nous montrerons comment tirer parti des algorithmes développés dans les chapitres précédents pour suivre de manière distribuée l'évolution de la performance au cours du temps pour la confronter aux événements qui ont été observés.

Chapitre 3

Agrégation et diffusion des mesures dans un réseau

3.1 Motivations

Dans le chapitre précédent, nous avons proposé une architecture modulaire permettant de décomposer un système de mesure distribué en différentes couches. Ce chapitre se concentre sur la couche de *partage*. Au sein d'un réseau, son rôle est de fournir à un sous-ensemble de nœuds ou agents des informations mesurées sur leur système. Ces informations représentant l'état du réseau peuvent s'avérer riches et complexes. En effet, les communications sans fil ont rendu complexe l'étude de nos réseaux. Les terminaux sont diversifiés, mobiles et plus nombreux. Les informations contextuelles dont ils disposent sont riches : géographiques, météorologique, mais aussi liées au profil et aux données physiologiques des utilisateurs. Pour autant, les mesures portées à la connaissance de chacun des agents devront être un compromis entre fréquence d'échantillonnage, précision et intrusivité.

Afin de mieux appréhender ces caractéristiques dans le cas de la mesure distribuée, faisons un parallèle avec un système d'acquisition plus classique. Dans le cas d'un capteur, la fréquence d'échantillonnage impose la finesse temporelle des phénomènes mesurés, sa résolution définit le changement minimal perceptible, et son intrusivité caractérise le changement d'état qu'il aura impliqué sur le système. Dans le cas de la mesure distribuée, le temps d'acquisition sera le temps nécessaire aux agents pour avoir sensiblement la même information. La résolution sera liée au niveau d'agrégation appliqué aux mesures. L'intrusivité correspondra à la quantité de ressources qui aura été consommée. Dans ce chapitre nous ne traiterons pas l'intrusivité des mesures actives (qui injectent volontairement du trafic à des fins de mesures). Nous traiterons en revanche de l'intrusivité des mesures passives qui, dans le cas *distribué*, injectent *involontairement* du trafic pour la coordination des nœuds. Un système de mesure distribué performant repose donc sur la distribution efficace de l'information. Il existe pour cela, différentes techniques issues des domaines de l'algorithmique ou encore du contrôle distribué et qui ont été notamment développées pour le monitoring réseau ou la fusion de données dans les réseaux de capteurs.

Dans la prochaine section de ce chapitre, nous étudierons ces diverses techniques, puis en section 3.3 nous montrerons comment les réunir sous un même modèle algébrique. Ceci nous permettra de les décomposer en différents services qui sont l'agrégation, la diffusion et le transport. À la vue des besoins exprimés autour des réseaux sans fil, nous nous pencherons en section 3.4 sur le cas des réseaux aux topologies dynamiques pour lesquels nous proposerons des algorithmes permettant, à ressources égales, la propagation plus rapide d'une moyenne de valeur. Nous conduirons en section 3.5 une étude de sensibilité, détaillant les facteurs impactant une telle diffusion d'information.

3.2 Travaux similaires

Le problème de la mesure distribuée dans lequel un ensemble de mesures sont agrégées et transmises à un sous-ensemble de nœuds est globalement motivé par deux types de travaux. Les premiers du genre, issus de la supervision des réseaux, cherchent à vérifier l'état d'un ensemble de machines afin de s'assurer de leur bon fonctionnement. Les seconds sont issus du domaine des réseaux de capteurs sans fil qui communiquent des mesures physiques afin de caractériser un environnement. Si les hypothèses de travail sont souvent différentes, dans les deux cas, le problème est similaire. Certains nœuds du réseau sont à la recherche d'un *consensus* et souhaitent converger vers un résumé de leurs états. À ces fins, l'état de l'art propose différentes solutions algorithmiques, mais aussi des études plus formelles issues de la théorie du contrôle. Cette section vise à couvrir ces différents aspects .

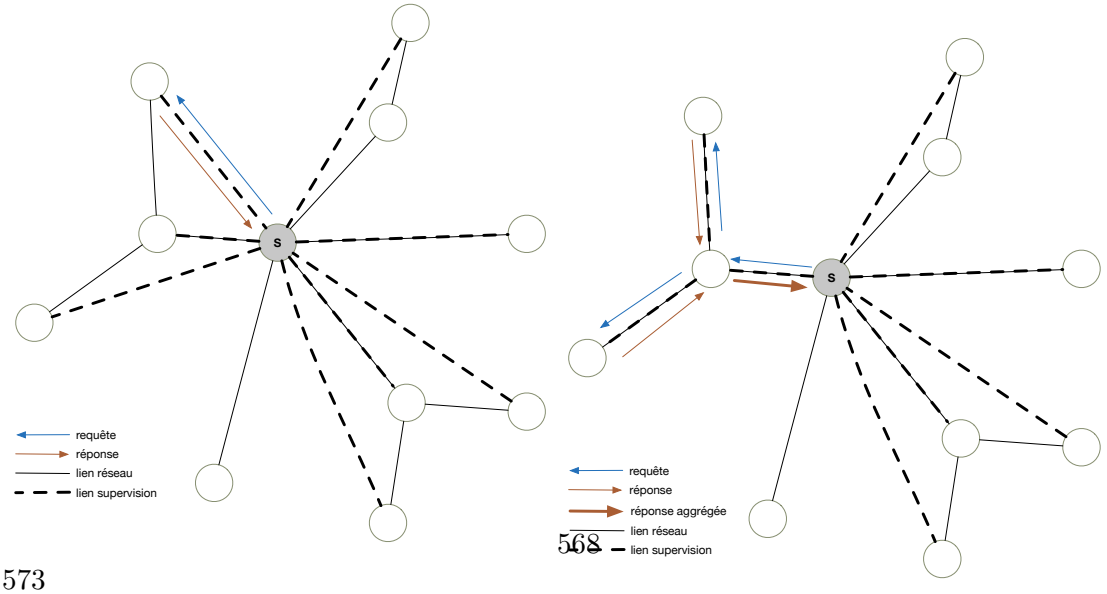
3.2.1 Arbre de communication et supervision

Une partie des solutions existantes traitent de cas où la topologie du réseau est relativement statique. Ils s'adaptent aussi bien à des réseaux filaires que des réseaux sans-fil dans la mesure où la topologie évolue peu au cours du temps (nœuds fixes et liens *fiables*). Le principe repose sur l'organisation du réseau en un arbre afin d'optimiser le transfert de l'information. Il existe différents schémas de communication pour couvrir divers types de besoins. Un point différenciant va être le nombre de superviseurs.

Cas d'un superviseur unique

Le but est ici d'obtenir au sein d'un seul nœud ou superviseur, une information représentative de la totalité du réseau. Une solution classique est de suivre une approche de type attente active ou *polling*. De manière périodique, le superviseur va interroger les nœuds afin de calculer l'état du réseau. Comme illustré dans la figure 3.1, la topologie de supervision (ou réseau *overlay*) est celle d'une étoile. L'avantage d'une telle méthode est sa trivialité d'implémentation. Elle a néanmoins plusieurs inconvénients qui sont sa capacité de passage à l'échelle et son intrusivité (ou consommation de ressource). En effet, interroger périodiquement depuis un seul nœud l'ensemble du réseau peut s'avérer coûteux en terme de communication, voir même inutile si l'état des nœuds n'a pas changé. Pour pallier à ce phénomène, plusieurs solutions ont été proposées. Certaines proposent la mise en place d'une période d'échantillonnage variable [61], d'autres laissent une partie de la communication à l'initiative du nœud [62]. Ces deux approches offrent un compromis entre précision et intrusivité. Une alternative est d'effectuer des opérations de

traitement au plus proche de la source afin d'une part de simplifier la tâche du superviseur et d'autre part d'économiser les ressources de communication. La topologie de communication se voit changée et ne se limite pas à celle d'une étoile, mais à celle d'un arbre dont la racine est le superviseur, comme illustré figure 3.1.



568

573

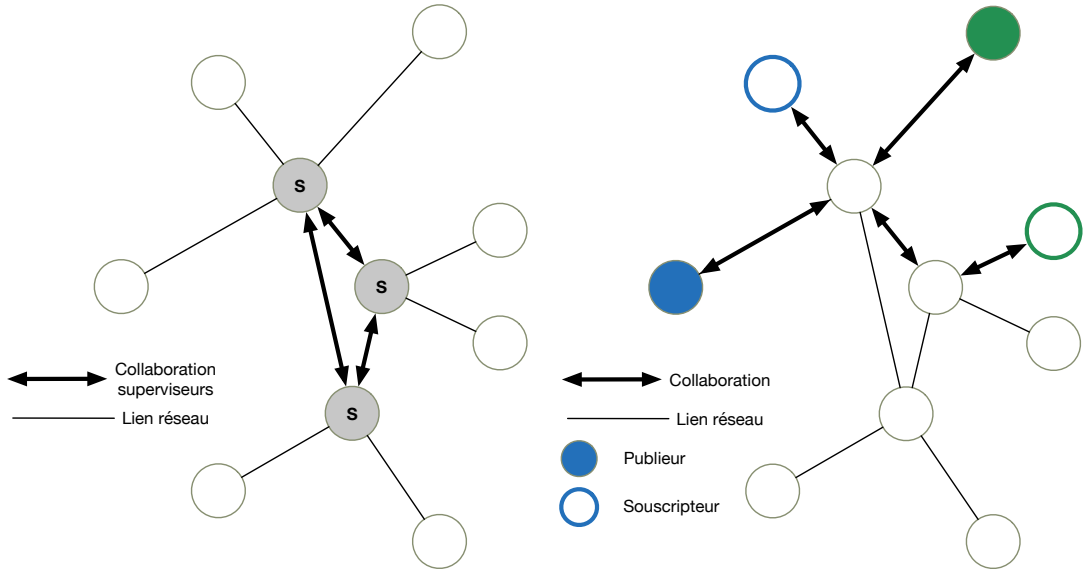
(A) Sans agrégation (B) Avec agrégation

FIGURE 3.1: Monitoring d'un superviseur en mode *polling*

Dans [63] les auteurs proposent un protocole construisant un arbre de recouvrement dans lequel les feuilles de l'arbre remontent de manière asynchrone des valeurs de compteurs issues de leur MIB. Le long de l'arbre les informations sont agrégées par des fonctions de types *somme*, *moyenne*, *minimum* et *maximum*. Les mêmes auteurs proposent dans [64] d'étendre la supervision à l'histogramme des variables d'état mesurées. D'autres travaux proches étudient la surveillance d'un seuil (cas dans lequel la somme de certaines métriques au sein d'un réseau ne doit pas dépasser une borne donnée) c'est le cas de [65] (et ses références). Le type de communication est ici multipoint à point aussi appelé *convergecast*. La racine dispose ainsi d'une vue globale du réseau. L'optimalité d'une telle communication est étudiée pour un réseau de capteurs sans fil dans [66]. Ce type de trafic est couramment rencontré dans le domaine des réseaux de capteurs, c'est le cas des protocoles ANMP, DRAMA ou encore RAIC tel qu'illustré dans [67]. Dans ce schéma, l'information est transmise vers la racine de l'arbre, les feuilles ne disposent pas d'information particulière et les liens sont supposés relativement stables (une maintenance de l'arbre sur défaillance ou changement de topologie est souvent prévue).

Cas de plusieurs superviseurs

Dans le cas de plusieurs superviseurs, le problème se complexifie en termes d'organisation et d'occupation des ressources. On distingue dans un premier temps le cas où les superviseurs sont en fait des collaborateurs qui se partagent la tâche de supervision (voir figure3.2a). Cette technique se retrouve dans les protocoles tels que DAMON, TD et MMAN. Il existe ensuite le cas où les superviseurs ne sont pas nécessairement collaborateurs, mais plutôt souscripteurs des nœuds de mesures. On fait face à un schéma de communication de type publication/souscription (ou pub/sub tel que dans la figure 3.2b). Alors qu'un nœud est capable de fournir des données, un ensemble de nœuds est intéressé par ces données. On retrouve dans cette catégorie de supervision des protocoles plus issus du domaine des réseaux de capteurs tels que SPIN[68] dans lequel un nœud communique dans un premier temps des métadonnées sur ce dont il dispose, laissant ses pairs l'interroger. Une autre façon de procéder est de déclarer, coté souscripteur son centre d'intérêt comme dans le protocole Directed-Diffusion [69], ou d'effectuer des requêtes complexes d'information tel que le fait ACQUIRE [70]. Dans ces trois protocoles, de la mise en cache au sein du réseau pour agréger les données est prévue.



(A) Collaboration entre superviseurs (B) Un superviseur est souscripteur

FIGURE 3.2: Monitoring avec plusieurs superviseurs

Les métriques considérées jusqu'à présent sont des valeurs de compteurs agrégées sous forme de somme, moyenne, maximum ou encore de variance [71]. Les algorithmes présentés sont adaptés à une topologie pour laquelle la constitution d'un arbre de communication est possible. Il existe le cas échéant d'autres solutions.

3.2.2 Algorithmes asynchrones et modèles épidémiques

Lorsque la constitution d'un arbre n'est pas possible ou est jugée trop coûteuse, il est possible d'organiser le monitoring du réseau sans aucune hiérarchie, notamment en utilisant un modèle épidémique (ou protocole *gossip*). Ce type de protocole tient son nom de la façon dont une information ou un virus a tendance à se propager au sein d'un réseau de personnes. Lorsqu'un nœud du réseau possède une information, il la communique de manière spontanée à un (ou plusieurs) de ses pairs. Au bout d'un certain temps, l'information a atteint tous les nœuds ou bien est devenue obsolète. Comme illustré dans la figure 3.3. Le comportement de ce type de modèle est étroitement lié à ceux des chaines de Markov et des marches aléatoires. De nombreux travaux utilisent ces protocoles [72–76]. Ils rentrent néanmoins tous dans le cadre algorithmique proposé par [77] et [78] illustrés dans les Algorithmes 1, 2 et 3 que nous avons volontairement simplifiés.

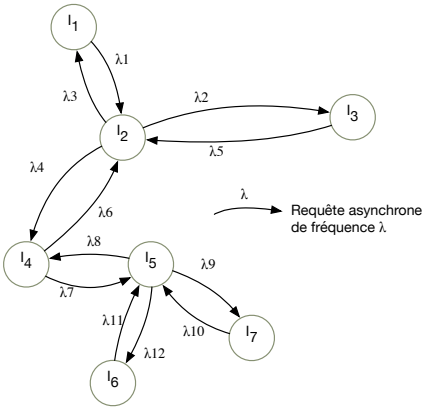


FIGURE 3.3: Une information se propage de nœud en nœud de manière probabiliste, l'état stationnaire dépendant de l'état initial et des fréquences des requêtes entre voisins.

Algorithme 1 boucle_principale()

```
loop
  inactivite(T)
  envoi_demande(selection_pair(), vue_locale)
end loop
```

Algorithme 2 sur_demande(dem, est_sym)

```
vue_locale ← agrégation(vue_locale, dem)
if est_sym then
  envoi_reponse(pair_distant, vue_locale)
end if
```

Algorithme 3 sur_reponse(rep)

```
vue_locale ← agrégation(vue_locale, rep)
```

Dans la version originale, l'opération d'*agrégation* prend en considération l'ancienneté des données afin de maintenir une vue partielle du réseau qui soit à la fois récente et représentative. Il existe une version symétrique et une version asymétrique. Dans la version asymétrique, seul le destinataire procède à l'opération d'agrégation. Dans le cas symétrique, l'initiateur de la requête effectue également l'opération. Lors de l'opération d'agrégation, l'intensité du mélange est définie par 3 paramètres conditionnant le contenu d'un message. Dans ce type de protocole, l'émetteur de la requête choisit son destinataire. Dans le cas d'un réseau où le graphe de communication est complet, le destinataire de la requête est choisi parmi tous les nœuds du réseau. Lorsque ce n'est pas le cas, par exemple pour les réseaux sans fil, il est moins coûteux pour un nœud de communiquer uniquement avec ses premiers voisins. Ainsi, plusieurs phénomènes se sont révélés difficiles à caractériser comme notamment l'impact de la distribution du degré des nœuds dans le graphe sur la convergence ou encore celui de pertes de messages [79].

Si les protocoles épidémiques sont adaptés aux réseaux dont la topologie est mobile, dans la version initiale, il est nécessaire pour un nœud de connaître ses voisins. Ceci peut poser problème si le graphe s'avère très dynamique. Il est toutefois possible de diffuser largement une requête puis dans le cas symétrique d'acquitter la réponse d'un de ses voisins pour confirmer la transaction. Alternativement, il est possible de suivre une approche synchrone, cette voie a été majoritairement explorée par les algorithmes de consensus issus de la théorie du contrôle distribué, que nous détaillons à présent.

3.2.3 Algorithmes synchrones et consensus

Le terme de consensus distribué est souvent associé au travail de Fischer, Lynch et Paterson [80], mais le problème du consensus a également une forte communauté du côté de la théorie du contrôle [81]. D'une manière générale, un consensus fait référence au fait d'atteindre un accord entre plusieurs agents qui partagent un même protocole. Cet accord est fonction de l'état des différents nœuds du réseau ou tout simplement de l'information dont ils disposent. On parle alors de f-consensus. Dans le cas du consensus de la moyenne, les agents doivent s'accorder sur la moyenne de leurs valeurs. Le cadre théorique pour étudier le problème du consensus dans un réseau dynamique d'agents a été introduit par Olfati-Saber et Murray dans [82]. Il est depuis appliqué dans de nombreux domaines comme la synchronisation d'oscillateurs [83], le partage de charge [84], la fusion de données de capteurs [85], ou encore le contrôle de formation d'une flotte de robot[86]. Dans le cadre de cette thèse, nous considérons la supervision de réseau. La théorie du consensus peut s'appliquer sous différentes hypothèses sur la nature des échanges entre les agents qui peut être continue ou discrète, ou bien sur le dynamisme de la topologie (nœuds fixes ou mobiles). Dans notre cas, les échanges entre agents seront

matérialisés par des paquets et seront naturellement discrets. Nous détaillons dans ce qui suit le cas du consensus de la moyenne.

Topologies et états statiques

Le problème du consensus de la moyenne en temps discret pour des topologies et des états statiques a été étudié en profondeur dans [87] et [88]. Comme précisé dans [86], les deux travaux réécrivent le moyennage par itération linéaire comme un produit matriciel. À chaque étape de l'algorithme, les nœuds mettent à jour leur valeur en utilisant le schéma $x_i(t+1) = \sum_{j=1}^{n} w_{ij} x_j(t)$ où w_{ij} est la pondération donnée par le nœud i au nœud j. On note l'état initial par s, $x(0) = s$. Afin de respecter la topologie du réseau, les valeurs de pondération sont contraintes par le graphe et s'il n'y a pas d'arête entre deux nœuds i et j alors nous avons $w_{ij} = 0$. Ainsi en notation matricielle on aura une matrice de pondération $W = (w_{ij})$. Nous obtenons alors $x(t+1) = Wx(t)$. L'objectif est de concevoir une politique de pondération telle que $\lim_{t\to\infty} x(t) = \bar{s}$, où $\bar{s}$ est la moyenne de l'état initial s ce qui est équivalent à l'équation matricielle $\lim_{t\to\infty} W^t = \frac{\mathbf{1}\mathbf{1}^T}{n}$, où $\mathbf{1}$ est le vecteur dont les entrées valent 1. Il a été montré dans [87] que la convergence est obtenue si et seulement si les trois conditions suivantes sont respectées :

(C.1) $W\mathbf{1} = \mathbf{1}$

(C.2) $\mathbf{1}^T W = \mathbf{1}^T$

(C.3) $\rho(W - \frac{\mathbf{1}\mathbf{1}^T}{n}) < 1$, avec $\rho(.)$ le rayon spectral de la matrice.

Les conditions C.1 et C.2 garantissent respectivement que la moyenne soit conservée au cours du temps et que celle-ci soit un point fixe de l'application linéaire associé à W. On peut aussi remarquer que la condition C.3 n'est pas respectée pour les graphes ayant plus d'une seule composante connexe, puisque le rang de la matrice de pondération est nécessairement plus grand que 1.

Topologie dynamique et états statiques

La condition de convergence dans le cas dynamique devient $\lim_{t\to\infty} \prod_{k=0}^{t} W(k) = \frac{\mathbf{1}\mathbf{1}^T}{n}$. De ce fait, trouver une condition équivalente ou nécessaire est loin d'être trivial. Néanmoins des conditions suffisantes sont établies. La convergence d'un produit infini de matrice est étudiée dans [89]. Une propriété intéressante pour une matrice est d'être paracontractante (H.0) pour la norme euclidienne, ce qui signifie que $W.x \neq x \implies ||Wx||_2 < ||x||_2$ pour une matrice quelconque W. En d'autres termes x est préservé seulement dans les directions des vecteurs propres de W. Cette propriété, associée à celle d'être stochastique double peut garantir une convergence (à l'inifini) vers la moyenne sous l'hypothèse

(H.1) que la suite des topologies possède une sous-suite de topologies connexes dans le temps [1], et que cette sous-suite soit présente une infinité de fois.

Topologie dynamique et états dynamiques

Le cas des états dynamiques (aussi appelé référence ou signal dynamique) est traité dans [90]. C'est en fait une relaxation de l'hypothèse du temps continu qui est traitée par [91]. Les auteurs donnent une relation entre la fréquence d'itération et la différence des états. Ils proposent un algorithme linéaire d'ordre n, mais nous ne considérerons que le premier ordre (FODAC pour First Order Dynamic Average Consensus). Une itération de l'algorithme FODAC peut se réécrire de la manière suivante :

$$x_i(t+1) = \delta s_i(t) + \sum_{j=1}^{n} w_{ij}(t) x_j(t), \tag{3.1}$$

où $\delta s_i(t) = s_i(t) - s_i(t-1)$. Les auteurs de [90] dérivent une borne supérieure sur l'erreur à l'état stationnaire qui est valide sous les hypothèses suivantes :

(H.2) La matrice de pondération $W(t)$ est stochastique double,

(H.3) La différence de dynamique du premier ordre des états est bornée,

(H.4) Le réseau est périodiquement connecté dans le temps.

L'hypothèse (H.3) signifie formellement :

$$\exists B \quad | \quad \forall t \quad \max_i(\delta s_i(t)) - \min_i(\delta s_i(t)) < B.$$

En pratique ceci est toujours vérifié (lorsque s_i est une métrique mesurée sur le réseau). En ce qui concerne l'hypothèse (H.4), une succession de graphes $\mathcal{G}_1$, $\mathcal{G}_2$, ..., $\mathcal{G}_n$ avec le même ensemble N de nœuds, est périodiquement connectée dans le temps si $\exists P \in \mathbb{N}^* \quad | \quad \forall t \quad \mathcal{G}_u = (N, \cup_{i=t}^{t+P} \mathcal{E}_i)$ est connecté. Les hypothèses faites sur la dynamique de la topologie (H1 et H4) sont très proches. On peut aussi remarquer que (A.2) exclut les matrices de permutations qui ne respectent pas la paracontraction (H.0).

Le calcul de la moyenne par itération linéaire a aussi été étudié sous d'autres aspects plus pratiques. Par exemple dans [92], l'effet de la troncature des valeurs est pris en compte. Alors que [93] traite la présence d'adversité dans le réseau. Enfin [94] propose une réelle implémentation.

1. Une succession $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n$ définie sur le même ensemble $\mathcal{N}$ de nœuds est connexe dans le temps si leur union $G_u = (N, \cup_{i=1}^{n} E_i)$ est fortement connexe

3.3 Modélisation algébrique de la couche de partage

Dans la section précédente, nous avons vu différentes techniques pour résoudre un problème de supervision dans lequel un ensemble de nœuds doit obtenir un résumé des informations mesurées sur le réseau. Ces techniques sont adaptées à différents cas de figure selon si le réseau est dynamique ou statique, avec ou sans infrastructure, ou encore respectant ou non une certaine hiérarchie. Dans cette section, nous proposons un cadre algébrique inspiré de celui du consensus permettant de formaliser et de comparer ces diverses techniques de supervision et de mesure distribuées.

3.3.1 Le modèle de communication

Que ce soit dans le cas d'une organisation en arbre, d'un algorithme épidémique asynchrone ou d'un algorithme de consensus synchrone, il est possible de décrire la situation par le même modèle algébrique. Nous considérons un graphe $G = (\mathcal{N}, \mathcal{E})$ pour un ensemble $\mathcal{N} = [1, n]$ de n nœuds (ou agents) et un ensemble $\mathcal{E} \in \mathcal{N}^2$ d'arêtes. Une arête matérialise la possibilité pour le nœud j de recevoir une information de mesure du nœud i. Il est important de noter que le graphe de communication dont nous parlons est le graphe de communication au niveau des agents de mesure. Ce graphe peut reposer sur un réseau de communication déjà existant et dans ce cas être considéré comme un sur-réseau (ou réseau overlay). Dans le cas d'une topologie dynamique, le graphe de communication évolue au cours du temps (avec la mobilité des agents par exemple). Certains agents pouvant communiquer à un instant donné peuvent potentiellement ne plus se joindre à l'instant suivant, alors que de nouveaux liens de communication peuvent apparaitre. Nous considérerons que l'ensemble des liens peut changer au cours du temps alors que l'ensemble des nœuds reste inchangé, ainsi, $\mathcal{G}(t) = (\mathcal{N}, \mathcal{E}(t))$ désigne le graphe du réseau à l'instant t. Toutefois, si $\mathcal{N}$ est constant, nous pouvons toujours modéliser l'arrivée d'un nœud en ajoutant des arêtes à un nœud isolé, et le départ d'un nœud en enlevant ses arêtes incidentes. Sauf mentionné explicitement, les graphes que nous considérons sont des graphes simples (sans boucle ni arêtes multiples).

Nous désignons par $x_i \in \mathcal{I}$ la mesure du nœud i et $x \in I^n$ le vecteur de mesure dans le graphe. Dans le cas d'une mesure évoluant dans le temps, $x_i(t) \in \mathcal{I}$ désigne l'échantillon de i au temps $t \in \mathbb{R}$. Nous modélisons une opération de mesure distribuée par une fonction $f : I^n \to I^n$ qui associe au vecteur x de mesure un vecteur image : $y = f(x)$. Suite à l'opération de mesure, chaque nœud i possède une image y_i du réseau. Cette image dépend des mesures de chaque nœud et de la fonction f définie par un administrateur.

Nous traitons le cas où I est un espace vectoriel et l'application f est linéaire. Dans ce cas, f possède une représentation matricielle $A \in M_n(\mathbb{R})$, et $\forall i$, y_i peut s'écrire sous la forme $y_i = \sum_{j=1}^{n} a_{ij}x_j$, ou a_{ij} constitue une entrée de la matrice A. Nous illustrons ce problème sur la figure 3.4. Nous modéliserons donc les différents échanges d'information de manière algébrique et nous aurons une équivalence entre la fonction f, la matrice A et le protocole distribué.

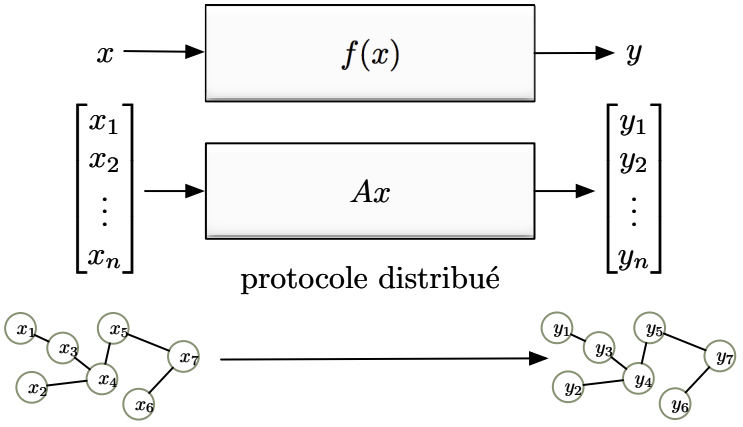


FIGURE 3.4: Modèle algébrique du partage de l'information

Lorsque le graphe G de communication est complet, l'opération est immédiate, chaque nœud i peut demander à ses premiers voisins l'ensemble des valeurs dont il a besoin afin de calculer y_i. Lorsque le graphe G n'est pas complet, la condition sur A pour que l'opération soit immédiate est équivalente à celle évoquée pour un consensus synchrone :

$$\forall i, j \notin \mathcal{E}, a_{ij} = 0$$

En effet si les nœuds i et j ne sont pas voisins, le nœud i ne pourra pas accéder à l'information du nœud j dont il a besoin. Nous sommes donc contraints de factoriser l'opérateur en s étapes, $A = W_s \times W_{s-1} \times W_{s-2} \ldots$, pour lesquelles, les nœuds nécessitent seulement les informations de leurs voisins. De cette manière chacune de ces étapes respecte la contrainte du graphe.

3.3.2 Précision d'une mesure

Une suite de s étapes $W_1, W_2 \ldots W_s$ pourra être jugée *satisfaisante* si la mesure a atteint une précision ϵ telle que :

$$||A - \prod_{k=1}^{s} W_{s-k}|| \leq \epsilon,$$

où $||.||$ est une norme induite et $\epsilon \in \mathbb{R}^+$ sont choisis en fonction du contexte. En effet, pour un x donné, une telle opération de mesure $W^s = \prod_{k=1}^{s} W_{s-k}$ donne une valeur y' et l'erreur introduite est alors :

$$||y - y'|| = ||Ax - W^s x|| = ||A - W^s x|| \leq ||A - W^s||.||x||,$$

La valeur ϵ correspond donc à l'erreur maximale possible pour un vecteur de mesure unitaire. Choisir une norme, revient à choisir ce que l'on cherche à minimiser (ou garantir). Par exemple :

— Minimiser l'erreur maximale revient à choisir la norme infinie : $||W||_\infty = \max_i \sum_{j=1}^{n} |w_{ij}|$.

— Minimiser l'erreur absolue moyenne revient à choisir La norme L_1 $||W||_\infty = \max_j \sum_{i=1}^{n} |w_{ij}|$.

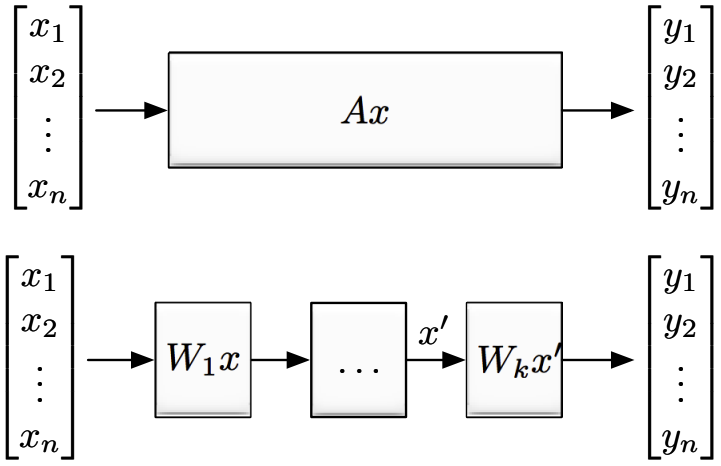


FIGURE 3.5: Factorisation de la matrice A pour satisfaire les contraintes du graphe de communication

3.3.3 Coût d'une mesure

Chaque transfert d'information consomme de la ressource. Ainsi, pour chaque coefficient $w_{ij} \neq 0$ matérialisant un échange entre le nœud i et le nœud j, nous pouvons associer un coût c_{ij} dépendant du lien de communication utilisé. Pour une série $W_1, W_2, \ldots, W_s$, nous aurons donc une série $C_1, C_2, \ldots, C_s$ de matrices de coût. Pour un ϵ donné, le coût total C_ϵ est une norme de la somme des matrices de coût :

$$C_\epsilon = ||\sum_{k=1}^{s} C_k||.$$

La norme $||.||$ pourra une fois de plus être choisie en fonction du contexte. Parmi les normes s'appliquant à notre problème, on retrouve :

— La norme de Frobenius : $||C||_F = \sum_{i=1}^{n} \sum_{j=1}^{n} |c_{ij}|^2$, modélisant le coût total.
— La norme max : $||C||_{\max} = \max_{i,j}\{|c_{ij}|\}$, modélisant le maximum des coûts par lien.
— La norme L_1 : $||C||_1 = \max_j \sum_{i=1}^{n} |c_{ij}|$, modélisant le coût maximum par nœud en terme de transmission.
— La norme L_1 : $||C||_1 = \max_j \sum_{i=1}^{n} |c_{ij}|$, modélisant le coût maximum par nœud en terme de réception (ou d'opération arithmétique).

Dans notre modèle, nous ne tenons pas compte du cout du calcul réalisé au sein d'un nœud, cette valeur est directement liée aux nombres de voisins avec lequel un nœud va échanger ainsi que de la complexité de l'information échangée.

3.3.4 Latence d'une mesure

Chaque transfert d'information nécessite un certain temps. Ainsi, pour chaque coefficient $w_{ij} \neq 0$ matérialisant un échange entre le nœud i et le nœud j, nous associons aussi une latence τ_{ij}, dépendant du lien de communication entre les nœuds. Pour une série $W_1, W_2, \ldots, W_s$, nous aurons donc une série $\mathsf{T_1}, \mathsf{T_2}, \ldots, \mathsf{T_s}$ de matrices de latence. Le temps t_ϵ nécessaire à la réalisation des k opérations est plus complexe à évaluer. En effet, pour une étape k donnée, un nœud devra attendre le résultat de l'étape $k-1$ pour chacun des voisins qu'il doit considérer. Analyser la latence de la mesure revient à calculer un chemin critique dans l'ensemble des tâches de calcul qu'entreprendront les nœuds. Le temps t_ϵ est donné par le produit des matrices de latence au sens de l'algèbre max-plus :

$$t_{epsilon} = || \bigotimes_{s=1}^{k} L_s \; ||_{max}.$$

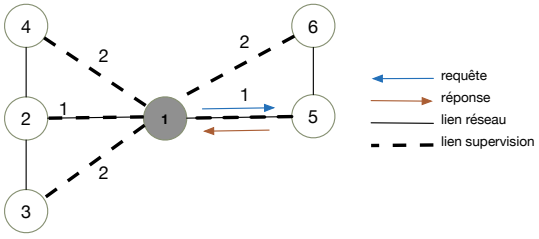
En algèbre max-plus, l'addition se note $\oplus$ et $a \oplus b = \max(a, b)$, tandis que la multiplication se note $\otimes$ et $a \otimes b = a + b$, les éléments neutres de l'addition et de la multiplication sont respectivement $-\infty$ et 0. Par construction, le produit des matrices de latence donne, pour chaque nœud, le temps nécessaire pour qu'il puisse commencer sa dernière opération. Dans ce cas, nous prenons la norme max qui correspond au maximum de ces temps. Dans notre modèle, nous ne tenons pas compte du temps de calcul, comme pour le coût, cette valeur est liée au nombre de voisins impliqués dans le calcul ainsi que de la complexité de l'information considérée.

Le cœur du problème est, pour un ϵ donné, de minimiser temps et coûts. Dans ce qui suit, nous illustrons notre modèle en analysant différents cas d'étude de la littérature.

3.4 Cas d'études

3.4.1 Topologie complète et superviseur unique

Dans un premier temps nous considérons le mode interrogatoire comme illustré 3.6. Le nœud 1 joue le rôle de superviseur et vient périodiquement interroger l'ensemble du réseau. Le réseau est connexe et il est possible d'établir un sur-réseau de partage dans lequel le graphe de communication $\mathcal{G}$ est une étoile dont le superviseur est l'élément central. Dans ce cas de supervision, la matrice A associée à la fonction de partage $y = f(x)$ est illustrée ci-dessous, figure 3.6.



$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{16} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

$$c_{\cdot 1} = \tau_{\cdot 1} = \begin{bmatrix} 0 \\ 2 \\ 4 \\ 4 \\ 2 \\ 4 \end{bmatrix}.$$

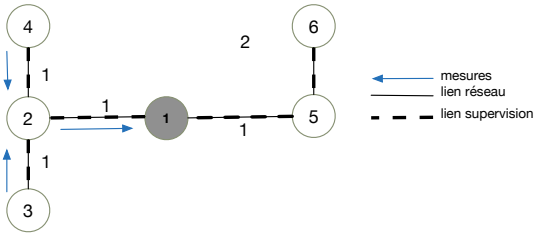
FIGURE 3.6: L'étoile de supervision et sa matrice A

Il n'est pas nécessaire de factoriser cette matrice, car le sur-réseau a été conçu de sorte à rendre le superviseur voisin de chaque nœud. Pour construire la matrice des coûts, nous fixons un coût par message identique sur tous les liens du réseau. Nous ferons de même pour la latence, ce qui revient à se baser sur le nombre de sauts. Les coefficients des matrices C et T sont ici égaux. Ils sont nuls sauf pour la colonne 1, qui vaut $c_{\cdot 1} = \tau_{\cdot 1}$. Ces valeurs correspondent au nombre de sauts nécessaires au superviseur pour atteindre chacun des nœuds multiplié par 2 afin de prendre en compte la requête et la réponse. Sous ces hypothèses minimiser le coût total revient à choisir le superviseur le plus *central* dans le réseau. D'un point de vue de la théorie des graphes, le superviseur doit avoir une *centralité de proximité* minimale. Cette métrique est définie pour un nœud par la somme de ses plus courts chemins vers chaque nœud du réseau. Bien entendu, cette stratégie est loin d'être optimale, elle reste néanmoins la plus simple à mettre en place, car les nœuds ne nécessitent qu'une très faible intelligence et c'est une chose qu'il ne faut pas négliger en pratique. L'approche peut être améliorée si l'on ajoute de l'intelligence aux nœuds. Les résultats obtenus pour $\epsilon = 0$ sont :

$\|\|C_\epsilon\|\|_F$	$\|\|C_\epsilon\|\|_{\max}$	$\|\|C_\epsilon\|\|_1$	t_ϵ
56	4	16	4

3.4.2 Topologie en arbre et traitement dans le réseau

Reprenons à présent le même réseau et la même fonction de partage, mais en donnant à certains nœuds la possibilité d'agréger les mesures réseau. Le graphe de communication $\mathcal{G}$ est maintenant un arbre représenté dans la figure 3.7. Comme la fonction f ne satisfait



$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{16} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 \dots & 0 \end{bmatrix}$$

FIGURE 3.7: L'arbre de supervision et sa matrice A

pas la contrainte du graphe de communication [2], il est nécessaire de factoriser la matrice A en plusieurs matrices, dans notre cas, on peut utiliser W_1, W_2 tels que définis ci-dessous pour obtenir $A = W_2 W_1$ où

$$W_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ et } W_1 = \begin{bmatrix} a_{11} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{12} & a_{13} & a_{14} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & da_{15} & a_{16} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

Dans ce cas, les matrices W_1 et W_2 sont compatibles avec le graphe de communication. La somme des matrices de coût C_ϵ correspond maintenant à la matrice d'adjacence du graphe tandis que la latence correspond au chemin le plus long partant du superviseur. Les résultats obtenus pour cette méthode sont bien supérieurs :

$\|\|C_\epsilon\|\|_F$	$\|\|C_\epsilon\|\|_{\max}$	$\|\|C_\epsilon\|\|_1$	t_ϵ
5	1	2	2

Afin de mieux appréhender la notion de latence, la figure 3.8 illustre les dépendances entre chaque nœud et les étapes de calcul. Comme on peut le constater, le calcul du chemin critique ou encore de la latence dans le calcul distribué peut s'exprimer à l'aide d'un produit matriciel au sens de l'algèbre max-plus. Dans ce cas précis, le premier nœud ne peut commencer son calcul que s'il dispose des valeurs précédentes soit au temps $\max(l_{32} + l_{21}, l_{42} + l_{21}, l_{56} + l_{61}) = l_{32} \otimes l_{21} \oplus l_{42} \otimes l_{21} \oplus l_{56} \otimes l_{61}$, ce qui correspond bien au maximum de la première colonne après multiplication des matrices T_1 et T_2.

2. Par exemple les nœuds 1 et 3 ne sont pas voisins mais $a_{31} = 1 \neq 0$

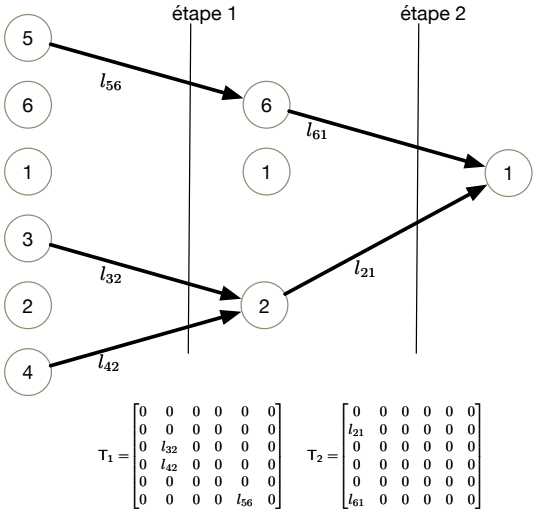


FIGURE 3.8: Latence de mesure et chemin critique

Ce modèle ainsi que celui de l'étoile sont possibles, car la topologie du réseau est connue et fixe. Il est donc possible de construire un sur-graphe complet afin de se libérer de toute contrainte de factorisation sur le graphe. Il est aussi possible de calculer un arbre de couverture minimum en fonction de la topologie du réseau, néanmoins le calcul d'un tel arbre a lui aussi un coût et nécessite un certain temps. Si l'on se réfère aux travaux de Gallager[95], il n'existe pas d'algorithme distribué résolvant ce problème en moins de $|\mathcal{E}|$ messages échangés. Avec un algorithme classique, on peut s'attendre à un nombre maximum de $5n.log(n) + 2|\mathcal{E}|$ messages ; en terme de complexité temporelle, seul le nombre de nœuds importe, et l'on obtient une complexité en $\mathcal{O}(n.log(n))$. Un tel calcul n'est nécessaire qu'une seule fois lorsque la topologie est fixe. Il est donc rentable pour des réseaux à infrastructure, mais dans le cas de réseaux mobiles, le maintien de la topologie est coûteux. En outre, le calcul d'un sur-réseau ou celui de la factorisation tel que présenté ci-avant nécessite un coût et un temps supplémentaires. Le temps que ces algorithmes aient convergé, la topologie peut avoir changé, il n'est donc pas toujours possible de les appliquer. Ceci nous incite à analyser les deux autres modèles présentés, à savoir le modèle épidémique et celui du consensus.

3.4.3 Topologie mobile et superviseurs multiples

Lorsque la topologie est dynamique, le problème initial devient très complexe. En effet, entre deux étapes, le graphe du réseau peut changer, remettant en cause tout le travail d'optimisation réalisé en amont. Il devient impossible de factoriser la matrice A si les contraintes correspondant au graphe ne sont pas connues à l'avance. Dans ce qui suit, nous envisagerons une topologie dont la dynamique est raisonnable et dans laquelle la

durée de vie d'un lien est suffisamment longue pour que deux nœuds puissent effectuer quelques échanges bidirectionnels.

Complexité du problème Pour rappel, nous cherchons à calculer de manière efficace un vecteur $y = Ax$ en prenant en compte les contraintes du réseau. Ne sachant pas à l'avance ces contraintes, le problème peut être résolu si l'on parvient à trouver un ensemble $\mathcal{W}$ de matrices respectant les critères suivants :

— Les coefficients d'une matrice de $\mathcal{W}$ sont calculables par les nœuds sur la base de leurs échanges avec leurs premiers voisins.

— Si W_s est une suite dans $\mathcal{W}$ alors $\lim_{k\to\infty} \prod_{s=1}^{k} W_s = A$.

Bien entendu, une solution générale à ce problème est quasiment infaisable. En revanche il est possible de traiter le cas du monitoring complètement décentralisé où tous les nœuds sont superviseurs et doivent converger vers la même valeur, c'est une situation de consensus. Dans cette situation $y = \alpha.\mathbf{1}$, ce qui signifie que $y_1 = y_2 = \ldots$.

$$\begin{aligned} y_1 &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ y_2 &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ \vdots &= \vdots \end{aligned}$$

Il est clair que si l'on souhaite avoir, $y_1 = y_2 = \ldots$, quelle que soit la valeur de x, il faudra avoir $a_{1i} = a_{2i} = \ldots$ pour toute valeur de i. Ainsi les lignes de A sont égales.

Le consensus de la moyenne permet d'obtenir $y_i = \frac{1}{n}\sum_{j=1}^{n} x_j, \forall i$. Par changement de variables, il est possible pour chaque nœud j de poser $z_j = a_{1j}x_j$. En réalisant un consensus de la moyenne sur z, on obtient alors $y'_i = \frac{1}{n}\sum_{j=1}^{n} z_j = \frac{1}{n}\sum_{j=1}^{n} a_{1j}x_j, \forall i$. Cela signifie que sous réserve de connaître le nombre n de nœuds dans le réseau, un algorithme de consensus de la moyenne est capable de résoudre tous les cas de monitoring où les superviseurs doivent acquérir la même information (c.-à-d. $\sum_{j=1}^{n} a_{1j}x_j$). Comme nous l'avons vu dans la section précédente, il existe deux grandes façons de procéder à un consensus. La première est de suivre un algorithme asynchrone de type *gossip*, la seconde est de suivre une approche synchrone correspondant à une moyenne par itérations. Nous pouvons une fois de plus modéliser ces deux approches par un produit matriciel convergent vers la matrice A.

Protocole *gossip* asynchrone et consensus linéaire synchrone Le principe d'un protocole de type *gossip* est de contacter un pair afin de mettre en commun des informations. La représentation algébrique d'un tel échange entre deux nœuds où i contacte

j est :

$$\begin{aligned} x_j &:= \alpha x_i + (1-\alpha)x_j \\ x_i &:= \beta x_j + (1-\beta)x_i \\ \alpha,\beta &\in [0,1]^2 \end{aligned}$$

Dans le cas d'un protocole asymétrique seul le pair contacté met à jour sa valeur, cela correspond au cas où $\beta = 0$. Un tel échange correspondrait à une étape dont la matrice $W_{\alpha\beta}$ serait telle que ci-dessous :

$$W_{\alpha\beta} = \begin{bmatrix} ((1-\alpha) & \alpha & \dots & 0 \\ \beta & (1-\beta) & \dots & 0 \\ \vdots & \vdots & 1 & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Prenant ceci en considération, tout algorithme de type *gossip* peut s'écrire comme un produit matriciel, où chaque matrice correspond à la finalisation d'un échange entre 2 nœuds. Bien entendu, il est possible de représenter simplement sur une seule matrice plusieurs échanges, tant que ceux-ci concernent des nœuds différents. Nous illustrons ceci figure 3.9. Pour une suite d'échanges concernant des couples exclusifs de nœuds, le produit des matrices W_2W_1 de chaque échange est similaire à une matrice diagonale par bloc. En ce qui concerne la convergence d'un protocole *gossip*, il est clair qu'une matrice

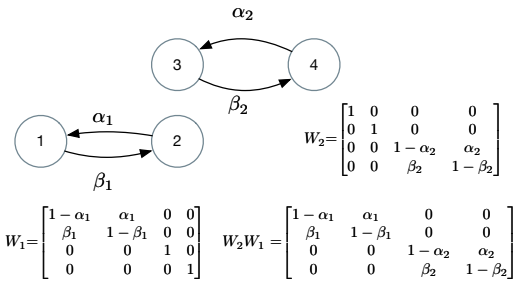


FIGURE 3.9: Représentation matricielle d'un protocole gossip

d'échange est paracontractante et que $\mathbf{1}$ en est une valeur propre à droite ($W_{\alpha\beta}\mathbf{1} = \mathbf{1}$), ce qui garantit la stabilité du protocole. En revanche si $\alpha \neq \beta$, $\mathbf{1}$ n'est pas valeur propre à gauche de W_{asym}, ce qui signifie que la somme des entrées de x n'est pas nécessairement conservée lors d'un échange. Ainsi, la somme $x_i + x_j$ (et de ce fait, la moyenne du vecteur x) peut être modifiée lors d'un échange. C'est pour cette raison qu'il est difficile de caractériser la valeur vers laquelle converge un protocole *gossip* en fonction du graphe, de l'ordre des échanges et de leur fréquence pour chaque nœud. Dans le cas d'un consensus linéaire synchrone, un nœud met à jour sa valeur en utilisant une

moyenne pondérée des valeurs de ses voisins. Les nœuds vont simultanément prendre en compte l'ensemble de leurs voisins, comme nous l'illustrons figure 3.10. Toute la difficulté est de définir une politique de pondération permettant de converger rapidement vers la moyenne. Comme on peut le constater, un protocole *gossip* est très proche d'un protocole

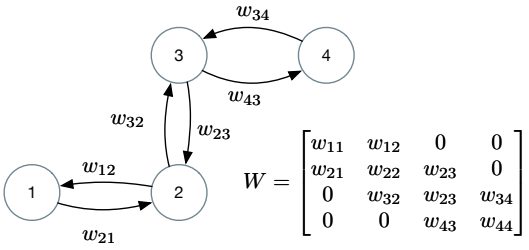


FIGURE 3.10: Représentation matricielle d'un protocole gossip

de consensus linéaire synchrone. Au final, la différence majeure provient du fait qu'un nœud en mode *gossip* ne considère qu'un seul voisin à la fois. Chaque étape de calcul W_s contient un seul échange. Le pouvoir de contraction d'une étape est donc très réduit. Néanmoins son coût est faible et le temps séparant deux étapes de calcul peut être quasi nul si celles-ci concernent des nœuds différents (les échanges peuvent être effectués en parallèle). Dans les deux cas, la convergence est asymptotique (cf figure 3.11) ce qui rend le temps d'acquisition de la mesure bien plus lent que lorsqu'un arbre peut être construit.

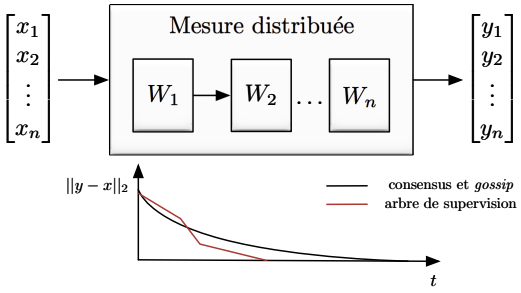


FIGURE 3.11: Système d'acquisition distribuée

Dans cette section, nous avons vu que le problème de la mesure et de la supervision distribuée pouvait être abordé de façon algébrique. De plus, être capable de résoudre le problème du consensus de la moyenne, peut couvrir le cas où tous les nœuds sont superviseurs et doivent effectuer une opération de mesure linéaire. Nous nous concentrons donc sur la réalisation d'une telle opération.

3.5 Décomposition du problème en différents services

Précédemment, nous avons vu que pour un réseau fixe il est préférable de suivre une organisation en arbre et que lorsque la construction d'un arbre était trop coûteuse, une approche de type *gossip* ou consensus devait être privilégiée. Il sera donc nécessaire de composer différentes stratégies en fonction des propriétés du réseau sous-jacent. Une architecture de mesure distribuée devra donc être modulaire.

Afin de pouvoir participer à une opération (ou campagne) de mesure, les agents doivent pouvoir être capables de réaliser trois grandes tâches qui consistent à :

- Transmettre leur mesure à d'autres agents
- Décomposer f (et dans notre cas $\frac{\mathbf{1}\mathbf{1}^T}{n}$) en un produit de matrice
- Effectuer le calcul d'une combinaison linéaire de mesures.

En terme de conception orientée objet, nous avons réparti ces tâches au sein de deux interfaces de services s'articulant autour d'une classe de donnée appelée `Mesure`. Nous nommerons les interfaces de services respectivement `Transport`, et `Distribution`. Chacune de ces interfaces peut posséder différentes stratégies d'implémentations afin de s'adapter au mieux à la technologie de communication et au réseau sous-jacent.

3.5.1 La classe de `Mesure`

La classe de donnée `Mesure` peut être instanciée par une mesure locale ou par une mesure agrégée. Cette classe est la représentation informatique de ce que nous avons défini comme l'ensemble $\mathcal{I}$. Afin de respecter notre modèle algébrique, la classe `Mesure` doit respecter l'interface suivante :

```
Mesure Mesure(void *repr_binaire);  // constructeur
Mesure repr_binaire();
Mesure addition_interne(Mesure m);
Mesure multiplication_scalaire(float scalaire);
NodeId source();
```

Une `Mesure` est composée d'un agent, qui en est la source. Elle doit posséder une opération interne d'addition et une opération externe de multiplication par un scalaire. En fonction du langage il peut-être avantageux de surcharger les opérateurs classiques d'addition, de multiplication et d'affectation, afin de rendre transparent toute opération sur une `Mesure`. Afin de pouvoir la distribuer et la transmettre au sein du réseau, il sera aussi nécessaire d'avoir une représentation binaire d'une `Mesure`. La manière dont sont implémentées ces méthodes pourra être adaptée en fonction de l'information portée, et des ressources disponibles. D'un point de vue implémentation, elle peut être spécialisée

en sous-classes. Par exemple, il est différent d'additionner deux réels ou bien deux vecteurs. De même en fonction de la bande passante disponible, différentes représentations binaires pourront être adoptées (avec ou sans compression par exemple). Nous verrons dans un prochain chapitre comment représenter le cas général d'un espace vectoriel.

3.5.2 Service de `Distribution`

Le service de distribution permet de définir la politique de diffusion des mesures, autrement dit, d'organiser et d'effectuer le calcul distribué de la valeur agrégée. Les fonctionnalités minimales que doit pouvoir offrir un tel service sont :

```
int souscrire(int campagne_id);
int quitter(int campagne_id);
int diffuser(int campagne_id, Mesure locale);
Mesure  aggregat(int campagne_id);
int ajouter_traitement(int campagne_id, void (*traitement) (Mesure));
```

Un agent peut souscrire à une campagne de mesure. En pratique le numéro de campagne peut directement prendre la forme d'un numéro de port (tel qu'`UDP` ou `TCP`). Un agent peut diffuser sa `Mesure`, et la mettre à jour puis la quitter le consensus ou la campagne de mesure. En contrepartie, il peut accéder à la valeur agrégée en cours de calcul. Il sera également utile de pouvoir enregistrer une fonction de rappel (ou callback) sur la mise à jour de la valeur agrégée. Différentes stratégies d'implémentation peuvent être implémentées ici, celle d'un algorithme *gossip*, celle d'un consensus synchrone, ou encore celle d'une approche en arbre. L'implémentation de type *gossip* est alors celle décrite précédemment. En ce qui concerne l'organisation en arbre, comme nous suivons une approche orientée service, ceci revient au cas où le service n'est pas local au nœud, mais est implémenté sur un nœud voisin, et accédé par l'intermédiaire d'un appel distant de procédure. La stratégie locale est alors celle d'un simple proxy.

3.5.3 Service de `Transport`

Le service de transport est chargé de transmettre une information à un ou plusieurs autres agents. En terme d'implémentation informatique, ce service se traduit par une interface implémentant les fonctionnalités suivantes :

```
int transmettre(IdSet destinataires, int campagne_id, void* info_locale, int priority);
int recevoir(int campagne_id, Id *source, void *info_distante);
int ajouter_traitement(void (*traitement) (Mesure));
```

Un service de transport doit pouvoir transmettre une information à un agent ou un ensemble d'agents. C'est le service de `Distribution` qui choisira le type de transport

dont il a besoin. Dans le cas d'un réseau IP la couche transport pourra être `TCP` ou `SCTP` pour une communication fiable et à destination unique, ou `UDP` pour plusieurs destinations. Notons également que pour les réseaux à très faible ressources, aucune couche de routage n'est nécessaire, une simple communication avec les voisins de premiers degrés est suffisante comme dans le cas d'une liaison 802.15.4. D'autres paramètres comme la gestion des priorités pourront être considérés. Cette différentiation de service sera notamment utile lorsque plusieurs campagnes de mesures seront menées en parallèles ou bien en concurrence avec les données utilisateurs. Nous verrons dans le prochain chapitre comment prendre en compte ces priorités dans le cas de `TCP`.

La figure 3.12 ci-dessous illustre différentes stratégies possibles pour une mesure distribuée.

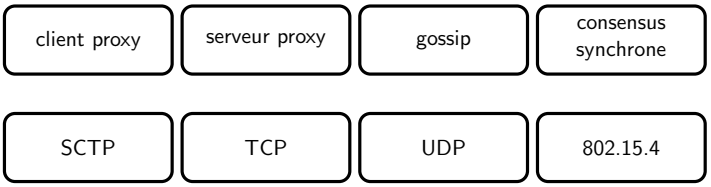


FIGURE 3.12: Couches distribution et transport pour la mesure distribuée

Dans la section suivante, nous considérons la mesure distribuée par un consensus synchrone et proposons une politique de pondération offrant, à coût et latence égaux, des performances supérieures à celles de l'état de l'art.

3.6 Une nouvelle heuristique pour le consensus de la moyenne

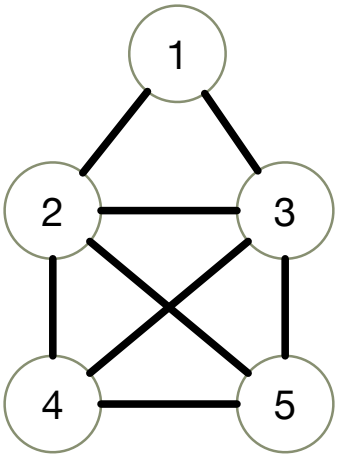
Comme établi précédemment, être capable de résoudre le consensus de la moyenne permet de traiter le cas de la supervision distribuée pour laquelle tous les nœuds superviseurs doivent mesurer la même valeur. Dans ce cas, l'opération de mesure est $\frac{\mathbf{1}\mathbf{1}^T}{n}$ et l'on doit définir une suite de matrice $W(k)$ telle que $\lim_{t\to\infty} \prod_{k=0}^{t} W(k) = \frac{\mathbf{1}\mathbf{1}^T}{n}$. Lorsque le graphe est statique, $W(k) = W$ est constant.

Pour ce problème, le taux de convergence (aussi appelé pouvoir de contraction) a été défini dans [88] par

$$r_{asym}(W) = \sup_{x(0)\neq\bar{x}} \lim_{t\to\infty} \left(\frac{||x(t) - \bar{x}||_2}{||x(0) - \bar{x}||_2} \right)^{\frac{1}{t}}.$$

On lui associe souvent le temps de convergence $\tau_{asym} = \frac{1}{\log(1/r_{asym})}$ à des fins de métrique de comparaison. Un résultat important est que le taux de convergence peut être directement dérivé des propriétés spectrales de la matrice W et $r_{asym}(W) = \rho(W - \frac{\mathbf{1}\mathbf{1}^T}{n})$, où ρ désigne le rayon spectral. Ainsi trouver la pondération optimale peut être vu comme un problème de minimisation. Ce dernier problème n'est pas nécessairement convexe, sauf si W est symétrique. Pour cette raison l'hypothèse d'un graphe G non dirigé est souvent faite. Afin d'avoir un résultat approché, il existe des heuristiques de pondérations. On retiendra trois heuristiques symétriques principales qui sont celle de la meilleure pondération constante (BCW pour Best Constant Weight), celle du degré maximum (MDW pour Maximum Degree Weight) et celle de Metropolis (MW pour Metropolis Weight). Toutes ces heuristiques définissent une matrice positive et stochastique double, ce qui nous permet de faire un lien étroit avec les chaines de Markov. Nous les détaillons dans la table ci-dessous. Toutes ces heuristiques sont dites Laplaciennes et nous en expliquons la raison dans ce qui suit.

Le Laplacien d'un graphe est une matrice le caractérisant. Pour un graphe sans boucle $G = (\mathcal{N}, \mathcal{E})$, la matrice d'adjacence est définie par $A = (a_{ij})$ où $a_{ij} = 1$ si $(i,j) \in \mathcal{E}$ et 0 sinon. Le voisinage V_i d'un nœud i est $V_i = \{j \in \mathcal{N} | (i,j) \in \mathcal{E}\}$. Son degré d_i est $d_i = \sum_{j \in V_i} a_{ij}$. Le Laplacien de G est défini par $L = D - A$ avec $D = I(d_1, d_2, ...d_n)^T$ la diagonale des degrés. Ces notations sont illustrées par le graphe simple de la maison sur la figure 3.13. Dans ce graphe toutes les pondérations valent 1 et $d_i = |V_i|$.



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

FIGURE 3.13: Le graphe de la maison, sa matrice d'adjacence et son Laplacien

Si l'on observe en table 3.1, les coefficients d'une heuristique donnée h sont tels que $w_{ii} = 1 - \sum_{j \in V_i} w_{ij}$. De cette manière, la matrice W_h peut s'écrire $I - L_h$ ou L_h est le Laplacien du graphe pondéré par l'heuristique h.

Heuristique	w_{ij}	Hypothèse sur la connaissance d'un nœud
BCW	$\frac{2}{\lambda_1(L)+\lambda_{n-1}(L)}$	Topologie connue
MDW	$\frac{1}{\lambda_1(d_{max})}$	Degré maximum connu
MW	$\frac{1}{1+\max(d_i,d_j)}$	Voisins de rang 1 connus

TABLE 3.1: Heuristiques Laplaciennes connues

Il a été montré dans [87] que la meilleure pondération constante dépend du spectre de L, qui est le Laplacien du graphe sans pondération. Plus précisément la meilleure pondération constante est donnée par la plus grande valeur propre λ_1 ainsi que sa deuxième plus petite λ_{n-1} de L. Plus récemment, [96] a proposé une heuristique semblant plus performante que les autres. Nous la noterons NW (pour Neighbor Weight). Le calcul des pondérations dépend des voisins de second degré et s'effectue en plusieurs étapes :

(1) $w_{ij} := 1 - \frac{|V_i \cap V_j|}{1+2.\min(d_i,d_j)-|V_i \cap V_j|}$
(2) Si $\sum_{j \in V_i} w_{ij} > 1$ alors $w_{ij} := w_{ij} / \sum_{j \in V_i} w_{ij}$
(3) $w_{ij} := \min(w_{ij}, w_{ji})$.

Ces politiques de pondération peuvent se voir comme une façon de définir une probabilité de transition d'une chaine de Markov puisque la matrice de pondération W est non-négative et stochastique double. Enfin un moyen d'accélérer la convergence est d'effectuer deux itérations en parallèle comme dans [88]. Dans cette méthode, la matrice de pondération change au cours du temps alors que la topologie doit rester statique.

Lorsque le graphe est dynamique, le défi est de permettre aux nœuds de définir rapidement une matrice de pondération $W(k)$, respectant le graphe de communication et permettant de converger vers la moyenne. Les heuristiques les plus intéressantes sont dans ce cas MW et NW. En effet elles offrent de bonnes performances et nécessitent uniquement la connaissance des voisins de premier (et respectivement second degré). En effet la politique MDW nécessite un premier consensus pour déterminer le degré maximum dans le réseau. Ceci nécessite au moins d étapes si d est le diamètre du graphe, et nécessite d'être mis à jour si ce dernier est dynamique. Le calcul du poids constant optimal sous cette dernière hypothèse est encore plus complexe puisqu'il est équivalent au calcul du spectre du Laplacien du graphe. Ces dernières heuristiques serviront néanmoins de référence.

3.6.1 Heuristiques symétriques et le paradoxe de l'étoile

Le défi lors de la construction d'une nouvelle heuristique est de lui donner un pouvoir de contraction meilleur que ses concurrentes, tout en se basant sur des informations acquises localement. Comparer deux heuristiques n'est pas trivial. Une heuristique peut avoir un

meilleur pouvoir de contraction qu'une autre sur un graphe donné sans pour autant la dominer (il existe un graphe pour lequel l'autre heuristique a un meilleur pouvoir de contraction). Intuitivement, on peut dire qu'une heuristique est adaptée à certains types de graphes. Comme nous l'avons vu, il existe un lien étroit entre le spectre d'un graphe et le poids constant optimal (BCW). Pour les mêmes raisons, une heuristique ne tenant pas compte de ce spectre lui sera plus ou moins adaptée.

Trouver une bonne heuristique revient à trouver un moyen pratique et simple de calculer le poids $w_{ij} (= w_{ji})$ que deux nœuds voisins i et j s'attribuent respectivement. Pour cette raison nous focalisons notre intérêt sur l'heuristique MW qui définit sa pondération sur la base de son voisinage local seulement. En dépit de bonnes performances globales, on peut remarquer que l'heuristique MW se comporte de manière contre-intuitive vis-à-vis des nœuds dont le degré est bien supérieur à ceux de leurs voisins. En effet, plus le degré d'un nœud est élevé, plus celui-ci a accès à de l'information, et moins celui-ci reçoit d'importance de la part de ses voisins (voir table 3.1). Pour illustrer ce phénomène particulier, nous utilisons le cas extrême qui est la topologie en étoile. La figure 3.14 illustre la pondération MW pour un tel graphe de 5 nœuds.

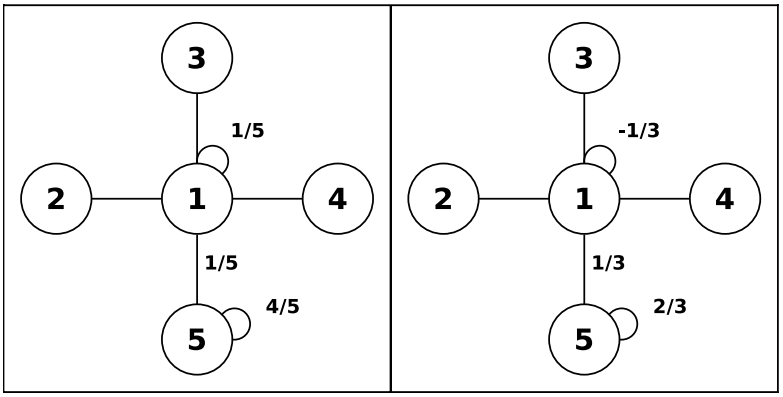


FIGURE 3.14: Pondération *Métropolis* (gauche) et optimale symétrique (droite) pour un réseau en étoile de 5 nœuds

Intuitivement, une topologie en étoile devrait mener à une convergence rapide puisque son diamètre est seulement de 2. En réalité, on peut dire que le nœud central souffre de sa notoriété. En effet, chaque nœud donne au nœud central une importance égale à celle qu'il reçoit de lui. Cette importance est alors faible puisque le nœud central a un degré élevé, et doit donner de l'importance à tous ses voisins. Autrement dit, alors que le nœud central a accès à la totalité de l'information en une étape, personne ne lui accorde de crédit. Ceci est dû au fait que les poids sont symétriques. Néanmoins la symétrie est une propriété nous garantissant que la moyenne est conservée à chaque itération. Afin

de réduire ce phénomène de *notoriété*, nous étudions dans ce qui suit la pondération symétrique optimale pour le réseau en étoile et établissons la proposition suivante.

Proposition 3.1. *La pondération symétrique optimale pour un réseau en étoile de n nœuds est donnée par l'heuristique du meilleur poids constant (BCW) où* $\alpha = \frac{2}{n+1}$.

Démonstration. Soit G un graphe en étoile de n nœuds où le nœud 1 est le nœud central. Étant donné la symétrie de la topologie, le poids w_{1i} donné par le nœud central pour chaque nœud terminal doit être égal, et nous posons $w_{1i} = \alpha \quad \forall i$. Comme nous souhaitons avoir des poids symétriques, $w_{1i} = w_{i1}$. Si nous voulons conserver la moyenne nous devons avoir $\forall i \quad \sum_{j=1}^{n} w_{ij} = 1$. Compte tenu des contraintes du graphe, les poids non nuls restant à déterminer se trouvent sur la diagonale. Par conséquent $w_{ii} = 1 - \alpha d_i \quad \forall i$, qui est par définition une heuristique du poids constant. La meilleure heuristique symétrique étant alors obtenue pour la valeur optimale de α qui est donnée par $\alpha^* = \frac{2}{\lambda_1(L)+\lambda_{n-1}(L)}$ où $\lambda_1(L)$ et $\lambda_{n-1}(L)$ sont respectivement la plus grande et la deuxième plus petite valeur propre du Laplacien du graphe G (selon [87]). Le calcul des valeurs propres est donné ci-dessous. Le laplacien L est tel que :

$$L = \begin{bmatrix} n-1 & -1 & \dots & -1 \\ -1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & \dots & \dots & 1 \end{bmatrix}$$

et par conséquent

$$L - \lambda I = \begin{bmatrix} n-1-\lambda & -1 & \dots & -1 \\ -1 & 1-\lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & \dots & \dots & 1-\lambda \end{bmatrix}$$

Les valeurs propres de L sont les solutions de l'équation $det(L - \lambda Id) = 0$. Bien heureusement, la structure de L est avantageuse et nous pouvons exploiter la formule suivante :

$$\det \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \det(D).\det(A - BD^{-1}C)$$

ce qui dans notre cas en choisissant D de taille $n-1$ donne :

$$\begin{aligned}\det(L-\lambda I) &= (1-\lambda)^{n-1}.\det(n-1-\lambda-\begin{bmatrix}-1 & \dots & -1\end{bmatrix} I_{n-1}\frac{1}{1-\lambda}\begin{bmatrix}-1 & \dots & -1\end{bmatrix}^T \\ &= (1-\lambda)^{n-1}.\det(n-1-\lambda-\frac{n-1}{1-\lambda}) \\ &= (1-\lambda)^{n-2}\lambda(\lambda-n)\end{aligned}$$

Les valeurs propres de L sont donc 0, 1, et n de multiplicité $n-2$, 1 et 1 respectivement. On a donc $\alpha* = \frac{2}{n+1}$ □

Heuristique optimale de l'étoile

Notre heuristique utilise le résultat précédent. Nous dérivons la pondération optimale symétrique de l'étoile à un graphe quelconque. Cette adaptation se fait on réécrivant la valeur de α comme une fonction des degrés des nœuds incidents. On peut remarquer que dans le cas de la topologie en étoile, $d_1 = n-1$ et $d_i = 1 \quad \forall i \neq 1$, dans ce cas là nous pouvons écrire la pondération symétrique optimale comme suit :

$$w_{ij} = \begin{cases} \frac{2}{1+d_i+d_j} & \forall j \in V(i) \\ 1-\sum_{j\neq i} w_{ij} & \text{si } i=j \\ 0 & \text{sinon} \end{cases}$$

Cette formulation peut-être utilisée comme heuristique sur d'autres graphes dans le but de réduire le phénomène précédemment évoqué. Par la suite nous noterons cette politique SOS (pour star optimal symmetric).

Considérons à présent les propriétés de la matrice de pondération issue de la politique SOS. Par construction, la somme par ligne d'une telle matrice vaut 1, puisque la matrice est symétrique, nous avons **1** qui est valeur propre à gauche comme à droite de la matrice. De plus nous avons la propriété suivante :

Proposition 3.2. *Les coefficients d'une matrice issue de la politique SOS a ses coefficients dans l'intervalle semi-ouvert* $(-1, 1]$.

Démonstration. Dans un premier temps, nous traitons le cas de $i \neq j$. Si $i \notin V(j)$ alors $w_{ij} = 0$, sinon la politique donne $w_{ij} = \frac{2}{1+d_i+d_j} \leq \frac{2}{3}$ puisque d_i et d_j valent au moins 1 (cas d'un couple isolé). Lorsque $i = j$, nous commençons par borner les coefficients w_{ij} pour $j \in V(i)$, puis nous bornons leur somme pour un $i \neq j$ et en déduisons des bornes sur w_{ii}. La borne inférieure correspond au nœud pour lequel tous les voisins ont accès à

tous les nœuds du graphe. La borne supérieure est le cas du nœud central d'une étoile.

$$\begin{aligned}
\frac{2}{d_i+n} \leq w_{ij} &\leq \frac{2}{d_i+2} \\
\frac{2d_i}{d_i+n} \leq \sum_{j\neq i} w_{ij} &\leq \frac{2d_i}{di+2} \\
1-\frac{2d_i}{di+2} \leq 1-\sum_{j\neq i} w_{ij} &\leq 1-\frac{2d_i}{d_i+n} \\
\frac{2-d_i}{2+d_i} \leq w_{ii} &\leq \frac{n-d_i}{n+di} \\
-1 < w_{ii} &\leq 1
\end{aligned}$$

□

Il résulte que le poids donné à un voisin est toujours positif, mais inférieur à 1, ce qui exclut toute matrice de permutation. De plus, on peut remarquer que ces poids sont toujours supérieurs ou égaux à ceux qu'aurait attribué la politique MW. Il est facile de voir que $\forall j \in V(i) \quad \frac{1}{1+\max(d_i,d_j)} \leq \frac{2}{1+d_i+d_j}$. Il découle de ceci que des coefficients négatifs peuvent se situer sur la diagonale, ce qui n'était pas le cas pour les autres heuristiques issues des solutions de mélange de chaines de Markov. Il a déjà été remarqué dans [87] que les solutions optimales présentaient souvent des coefficients négatifs (et pas seulement sur la diagonale), nous pouvons quelque part considérer l'heuristique SOS comme une chaine de Markov avec des probabilités *négatives* de rester dans un même état.

Remarque 3.3. Une pondération négative dans le cas d'une moyenne peut sembler contre-intuitive. En fait, lorsqu'un nœud s'attribue un poids négatif, il modère simplement la considération qu'il a reçue de ses voisins. Dans le cas de la politique SOS, la pondération qu'un nœud s'attribue peut s'interpréter comme une mesure de sa *notoriété* locale.

En effet, la pondération attribuée localement dépend de la position du degré du nœud dans la distribution des degrés au sein du voisinage. Ceci peut s'illustrer au travers de la propriété suivante :

Proposition 3.4. *Si un nœud domine (resp. est dominé par) son voisinage en terme de degré, alors sa pondération locale donnée par SOS est négative (resp. positive).*

Démonstration. Nous montrons la première implication, la seconde étant similaire. Si le nœud i domine strictement son voisinage en terme de degré, alors $\forall j \in V(i) \quad d_j \leq d_i - 1$. Ainsi, nous avons $w_{ij} \geq 1/d_i$. En sommant sur $V(i)$ nous obtenons $\sum_{j\neq i} w_{ij} \geq 1$, ce qui implique $w_{ii} \leq 0$. □

Dans la prochaine sous-section, nous comparons les performances des différentes heuristiques avec celle que nous proposons dans cette thèse, l'heuristique SOS, pour différentes configurations réseau.

3.6.2 Comparaison des pouvoirs de contraction

Nous évaluons la politique SOS pour des graphes fixes ayant une seule composante connexe. En effet, être connexe est une condition nécessaire pour qu'un algorithme de consensus converge vers la moyenne sur un réseau fixe. La métrique que nous utilisons pour comparer la performance de deux heuristiques est le temps de convergence. Le temps de convergence d'un algorithme de moyennage est $\tau_{asym} = \frac{1}{\log(1/r_{asym})}$, où le pouvoir de contraction r_{asym} est donné par

$$r_{asym}(W) = \sup_{x(0)\neq\bar{x}} \lim_{t\to\infty} (\frac{||x(t)-\bar{x}||_2}{||x(0)-\bar{x}||_2})^{\frac{1}{t}}.$$

dans le cas du consensus distribué avec des itérations linéaires. Le pouvoir de contraction est caractérisé par les propriétés spectrales de la matrice de pondération W et nous avons $r_{asym}(W) = \rho(W - \frac{\mathbf{1}\mathbf{1}^T}{n})$. Comme W et $\frac{\mathbf{1}\mathbf{1}^T}{n}$ sont toute deux des matrices symétriques, leur différence est également symétrique et les valeurs singulières coïncident avec les valeurs propres. De ce fait, le rayon spectral est équivalent à la norme Euclidienne. Notre comparaison est alors la suivante. Pour un graphe donné, nous calculons la matrice de pondération W_h pour une heuristique h. Puis nous calculons le pouvoir de contraction $r_{asym}(W_h) = ||W_h - \frac{\mathbf{1}\mathbf{1}^T}{n}||_2$, duquel nous dérivons le temps de convergence $\tau_{asym}(W_h)$. Nous considérons cinq politiques qui sont MDW, MW, NW, BCW et SOS. Enfin, nous calculons le ratio entre le temps de convergence de la politique SOS et celle des autres. Notons que la politique MDW nécessite en fait un premier consensus pour déterminer le degré maximum, alors que la politique BCW, nécessite de calculer les valeurs propres du Laplacien du graphe, ce qui n'est pas faisable dans le cas dynamique, enfin la politique NW nécessite des échanges supplémentaires puisqu'il faut tenir compte des voisins de degré 2.

Les performances d'un consensus sont étroitement liées à la topologie du réseau ainsi qu'au nombre de nœuds. Par exemple pour un nombre donné de nœuds, une topologie en ligne ne pourra jamais battre une topologie maillée. De même, pour une topologie donnée, la convergence est d'autant plus rapide que le nombre de nœuds est faible. Nous avons donc considéré plusieurs topologies et plusieurs échelles pour comparer les heuristiques.

Topologies déterministes

Les topologies déterministes que nous avons testées sont les suivantes : ligne, cycle, étoile, grille, et barbell. Nous les avons représentées sur la figure 3.15. Nous avons calculé, pour

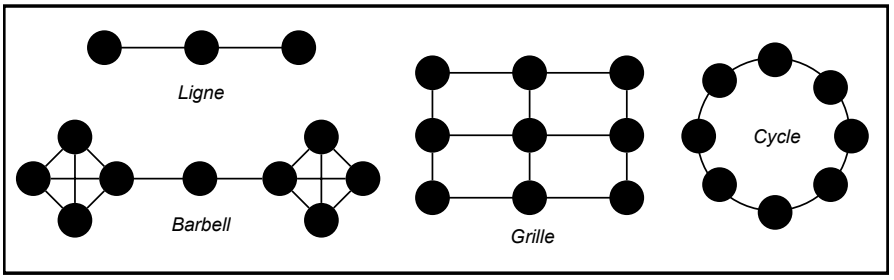
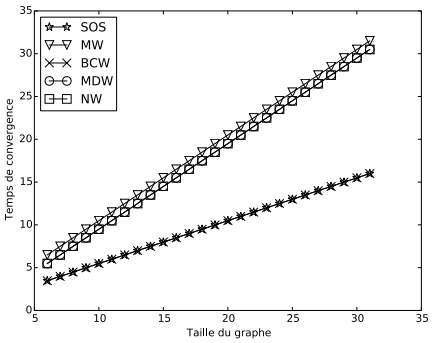


FIGURE 3.15: Topologies déterministes testées

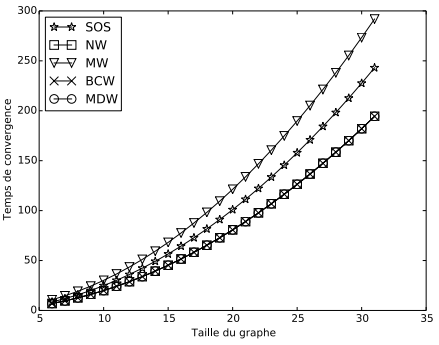
une taille croissante, le temps de convergence des cinq heuristiques pour chaque topologie. Pour les topologies ligne et cycle la taille n correspond au nombre de nœuds, pour la topologie étoile elle correspond au nombre de nœuds périphériques. En ce qui concerne la topologie grille, n représente la taille d'un côté en nombre de nœuds. Enfin pour le graphe barbell, n est le nombre de nœuds par graphe complet ainsi que le la longueur du chemin qui les sépare. La figure 3.16 illustre les résultats obtenus. Comme attendu, l'heuristique SOS a la convergence la plus rapide (comme BCW) pour la topologie en étoile puisque c'est l'heuristique symétrique optimale. Globalement, l'heuristique SOS converge plus rapidement que la politique MW pour toutes les topologies testées (mis à part pour une ligne de 2 nœud ou MW donne une pondération optimale). En ce qui concerne les autres heuristiques, BCW et MDW donnent une pondération identique sur chaque lien. Pour les topologies ayant des degrés réguliers (cycle, ligne et grille), ces pondérations sont meilleures. En revanche dès qu'il existe une grande disparité dans les degrés (barbell et étoile), la politique SOS s'avère être meilleure. Il est important de remarquer que nous n'avons pas considéré les heuristiques NW et MDW pour le graphe de type cycle, car celles-ci ne convergent pas pour des tailles paires de réseau. En effet les matrices de pondérations correspondantes ne sont pas paracontractantes (voir annexe B).

Topologies probabilistes

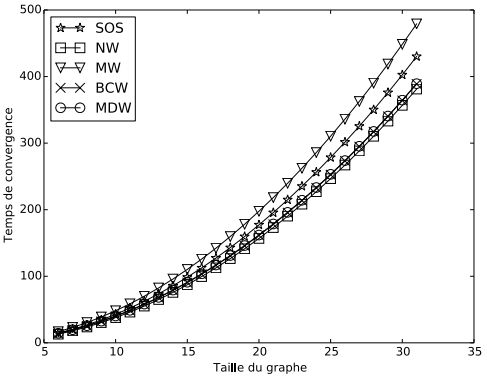
Nous analysons à présent les performances obtenues pour des conditions plus réalistes. Nous avons comparé les 5 heuristiques pour 4 types de graphes aléatoires. Il existe différents modèles de graphe probabilistes parmi lesquels nous avons retenu le modèle de Erdos-Renyi, le modèle Euclidien, le modèle de Barabasi-Albert et le modèle de Watts–Strogatz. Chacun d'entre eux est lié à un algorithme pour construire le graphe.



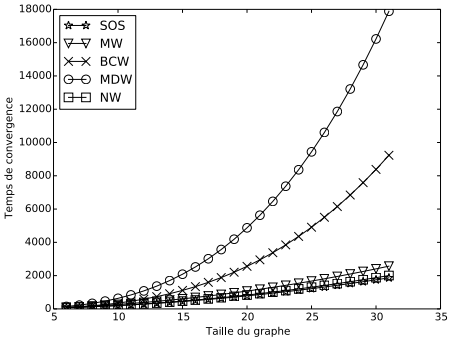
(A) Topologie en étoile



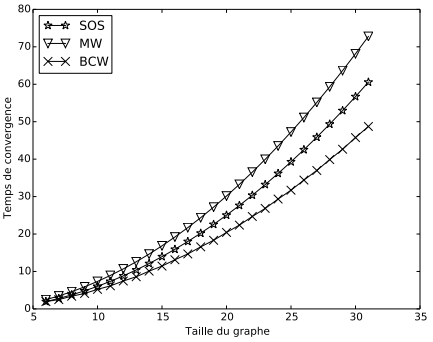
(B) Topologie en ligne



(C) Topologie en grille



(D) Topologie barbell



(E) Topologie cyclique

FIGURE 3.16: Temps de convergence pour différentes topologies déterministes

Un paramètre important qui différencie ces modèles est la distribution des degrés au sein des graphes qu'ils génèrent.

Le modèle Erdos-Renyi(E-R) : Pour un ensemble de n nœuds, chaque couple de nœuds à une probabilité p d'être connecté par une arête. Ces probabilités sont indépendantes. Afin d'avoir un graphe connecté, nous avons fixé la probabilité telle que $p = \frac{2\log(n)}{n}$. Ces graphes ont une distribution des degrés qui est binomiale.

Le modèle Euclidien : Dans ce modèle, les nœuds sont distribués de manière homogène dans un espace euclidien. Une arête existe entre deux nœuds si leur distance euclidienne est inférieure à un rayon r. Nous avons réparti les nœuds sur un carré unitaire. Ce type de modèle est souvent utilisé pour décrire des réseaux sans-fil où r représente la portée de communication. Dans le but d'avoir un graphe connecté, nous avons fixé un seuil $r = \sqrt{\frac{\log(n)}{n}}$. Ces graphes ont une distribution des degrés qui est celle de Poisson.

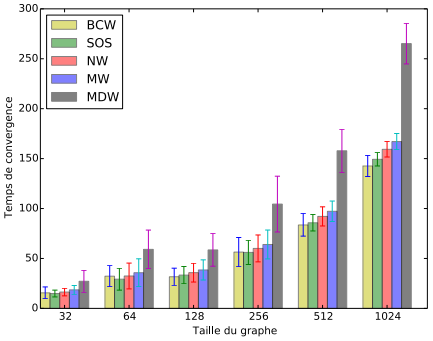
Le modèle de Barabasi-Albert (B-A) : Les nœuds sont ajoutés au graphe de manière itérative avec un nombre m d'arêtes. La probabilité pour un nœud présent $i \in P$ d'être choisi comme voisin par un nouveau nœud est donné par $p_i = \frac{d_i}{\sum_{j \in P} d_j}$. Ce modèle est connu pour être réaliste et pour décrire le phénomène d'attachement préférentiel observé dans les réseaux. Nous avons fixé $m = 1$ et $m = 2$. Ces graphes sont aussi appelés sans-échelle, car leur distribution des degrés suit une loi exponentielle.

Le modèle de Watts–Strogatz (W-S) : Le principe est de partir d'un graphe de n nœuds organisé en treillis de dimension k[3]. Puis chaque lien (i, j) est considéré avec une probabilité p pour être changé en un lien (i, k) tout en conservant des liens uniques entre deux nœuds. Nous avons fixé $k = 2.log(n)$ et $p = 0.3$. Ces graphes capturent les propriétés de grappes et de concentrations observables dans de réels réseaux. Ils ont une distribution des degrés qui est celle de Poisson.

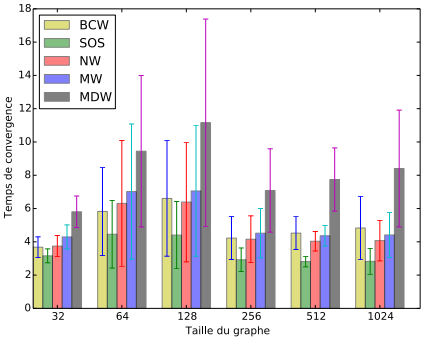
Nous avons calculé le temps de convergence des cinq heuristiques pour chaque modèle et pour différentes échelles. Pour une échelle et un modèle donné, nous avons généré 100 graphes. La figure 3.16 illustre les résultats obtenus en spécifiant le temps de convergence moyen et sa déviation standard pour chaque heuristique.

Nos résultats montrent que l'heuristique SOS offre globalement, un meilleur pouvoir de contraction que les heuristiques NW, MW et MDW et reste comparable à l'heuristique BCW (qui nécessite un calcul des valeurs propres du graphe). Sur l'ensemble des 3000 graphes générés (100 essais pour 6 tailles et 5 modèles), nous avons compté le nombre de cas pour lesquels l'heuristique SOS était meilleure que ses concurrentes (Nombre de cas favorables). Le résultat par heuristique est donné table 3.2, tandis que nous donnons

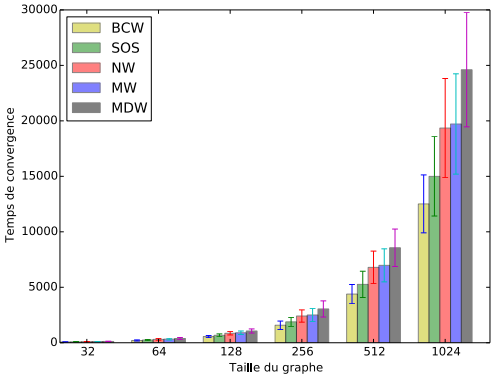
3. la topologie en grille déjà présentée est un treillis de dimension 2



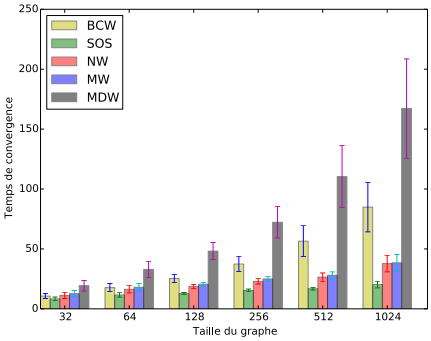
(A) Modèle Euclidien



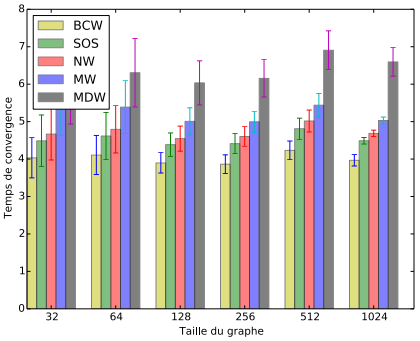
(B) Modèle Erdos-Renyi



(C) Modèle Barabasi-Albert (m=1)



(D) Modèle Barabasi-Albert (m=2)



(E) Modèle Watts-Strogatz

FIGURE 3.17: Temps de convergence pour différentes topologies probabilistes

le détail pour l'heuristique BCW table 3.3. L'heuristique BCW est plus performante pour les modèles Watts-Strogatz, Barabasi-Albert (pour m=1) et Euclidien mais moins performante pour les modèles Erdos-Renyi et Barabasi-Albert (pour m=2).

Le gain relatif obtenu sur une heuristique h en terme de temps de convergence :

$$\left(1 - \frac{\tau_{asym}(W_{SOS})}{\tau_{asym}(W_h)}\right),$$

dépend principalement du modèle de graphe. Pour les cas que nous avons testés, nous

n'avons pas relevé de lien entre ce gain et la taille du graphe. Nous donnons le gain moyen [4] pour chaque heuristique en table 3.2.

Heuristique	**MW**	**MDW**	**NW**	**BCW**
% de cas favorable	100	100	99.7	52.2
gain moyen	0.25	0.48	0.18	0.09

TABLE 3.2: Comparaison avec l'heuristique SOS, nombre de cas favorables et gain moyen

% de cas favorable	**32**	**64**	**128**	**256**	**512**	**1024**
Euclidien	42	46	37	35	25	19
Erdos-Renyi	99	100	100	100	100	100
Barabasi-Albert (m=1)	13	24	33	28	28	30
Barabasi-Albert (m=2)	94	100	100	100	100	100
Watts-Strogatz	12	1	0	0	1	0

TABLE 3.3: Score de l'heuristique SOS face à celle du BCW

Nos résultats de simulations ont montré que la politique SOS est compétitive pour différents types des graphes aléatoires réalistes. Dans la quasi-totalité des cas que nous avons testés, la politique SOS montre un pouvoir de contraction supérieur à celui des politiques MW, NW et MDW. Dans le cas d'une topologie dynamique, à chaque itération une matrice SOS offrirait des propriétés supérieures à celles de ses concurrentes, menant plus rapidement vers un vecteur de consensus pour des conditions identiques de connectivité dans le temps.

3.7 Importance des conditions initiales

Dans la section précédente, nous avons étudié le pouvoir de contraction de différentes heuristiques, ou encore son temps de convergence. Ces critères sont importants, car ils sont indépendants des mesures du réseau. En effet, ils sont directement issus de la matrice de pondération qui dépend du graphe considéré et de la politique induite par l'heuristique. Pour rappel le pouvoir de contraction était donné par :

$$r_{asym}(W) = \sup_{x(0)\neq\bar{x}} \lim_{t\to\infty} \left(\frac{||x(t) - \bar{x}||_2}{||x(0) - \bar{x}||_2} \right)^{\frac{1}{t}}.$$

Le pouvoir de contraction correspond au pire cas de convergence que l'on pourrait rencontrer. Un argument en faveur de ce type de critère est de pouvoir facilement comparer

4. Un gain moyen de α, signifie que en moyenne le temps de convergence de SOS est $(\alpha).100\%$, plus court que celui de l'heuristique considérée

deux heuristiques. Une seconde raison est que l'on souhaite souvent concevoir une bonne heuristique indépendamment des valeurs échangées.

Cependant, en fonction de ces valeurs de mesures, le temps nécessaire pour atteindre un consensus peut varier. Lorsque l'on considère la mesure distribuée et la supervision dans les réseaux, il est important de comprendre les propriétés d'une métrique qui ont un impact sur sa propagation. Dans cette section, nous nous intéressons à ces propriétés.

3.7.1 La déviation standard des mesures

La première propriété intuitive à laquelle on peut penser est la déviation standard de la mesure dans le réseau. La déviation standard pour une mesure $x \in \mathcal{I}I^n$ est définie par :

$$\begin{aligned} std(x) &= (\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{\frac{1}{2}} \\ &= \frac{1}{n^2}||x - \bar{x}.\mathbf{1}||_2 \end{aligned}$$

Comme on peut le constater, la déviation standard d'un vecteur de mesure est une distance entre ce vecteur et le vecteur de la moyenne, c'est-à-dire une distance entre l'état actuel et l'état souhaité. La première estimation qu'un nœud peut avoir de la moyenne des mesures est sa propre valeur. La déviation standard initiale, capture le degré de désaccord des nœuds au sujet cette valeur. Toute l'opération de mesure peut se voir comme une série de négociations qui sera d'autant plus longue que les nœuds sont initialement en désaccord.

À des fins d'illustration, la figure ci-dessous représente un consensus au sein d'un graphe de 100 nœuds généré suivant le modèle de Barabasi-Albert (m=2), pour différentes valeurs de déviation standard. L'ensemble des valeurs d'une mesure est $\mathcal{I} = \{0, \alpha\}$. Nous avons généré un vecteur initial $x_{ref} \in \{0,1\}^n$ pour lequel chaque valeur est choisie indépendamment avec les probabilités $p, 1-p$. Par la suite nous avons choisi différentes valeurs de x de la sorte : $x_\alpha = \alpha x_{ref}$. Nous avons choisi $p = \frac{1}{2}$. Par construction, $std(x_{ref}) = \frac{1}{2}$ et $std(x_\alpha) = \frac{\alpha}{2}$.

3.7.2 Assortativité des mesures et biais d'échantillonnage

Dans l'exemple précédent, nous avons pris soin d'utiliser un vecteur x_{ref} comme référence pour l'ensemble de nos essais. La raison pour laquelle nous avons procédé ainsi était de nous garantir d'une part que la distribution des valeurs ait la même structure (2 valeurs possibles dont les proportions p et $1-p$ sont respectés, mais également afin de conserver

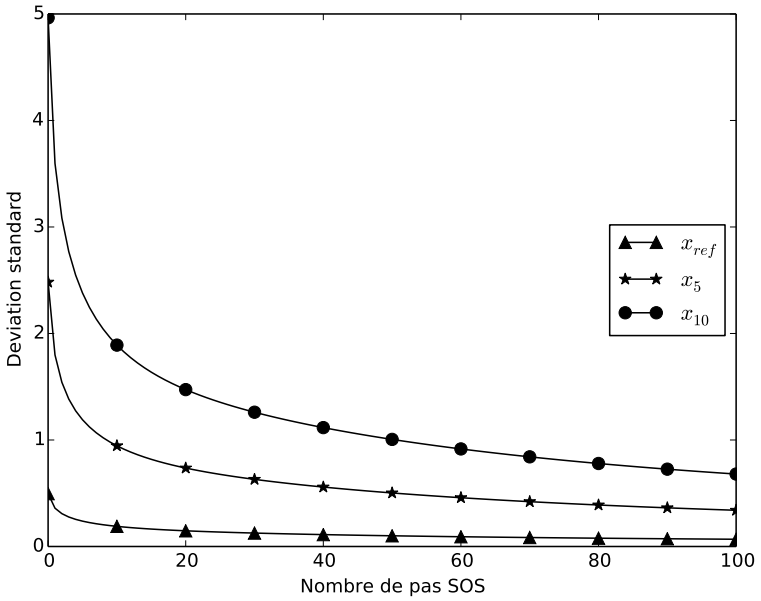


FIGURE 3.18: Consensus avec l'heuristique SOS pour différentes valeurs initiales de déviation standard

la même répartition des valeurs au sein du graphe. Nous montrons à présent que cette répartition a son importance dans le temps de convergence.

Une première intuition

En effet, afin de mieux en cerner la raison, imaginons un instant que chaque nœud représente une personne et la valeur mesurée son âge. Imaginons à présent que le graphe de communication soit celui d'un réseau d'amis, et que le but de l'algorithme de consensus soit de calculer l'âge moyen. Si la première estimation d'une personne correspond à son propre âge, la seconde sera une moyenne pondérée de l'âge de ses amis. Si par chance cette personne possède un panel d'amis représentatif de la population totale, cette première estimation sera déjà de qualité. Si en revanche cette personne ne possède que des amis de son âge, et que cet âge n'est pas l'âge moyen, la seconde estimation sera aussi mauvaise que la première.

La situation que nous avons décrite souligne l'importance de la répartition des valeurs de mesure dans le graphe. Cette répartition conditionnera le biais d'échantillonnage d'un nœud lors de son estimation. Une métrique pouvant caractériser cette répartition est l'*assortativité* des valeurs dans le graphe. L'*assortativité* correspond au coefficient de Pearson des valeurs des nœuds reliées par un lien. Pour un coefficient de 1, ces valeurs sont parfaitement corrélées et le graphe est dit assortatif pour la mesure, une valeur de -1 indique une corrélation négative et le graphe est dit dis-assortatif pour la mesure. Le

coefficient d'assortativité d'un vecteur de mesure x pour un graphe G, est donnée par :

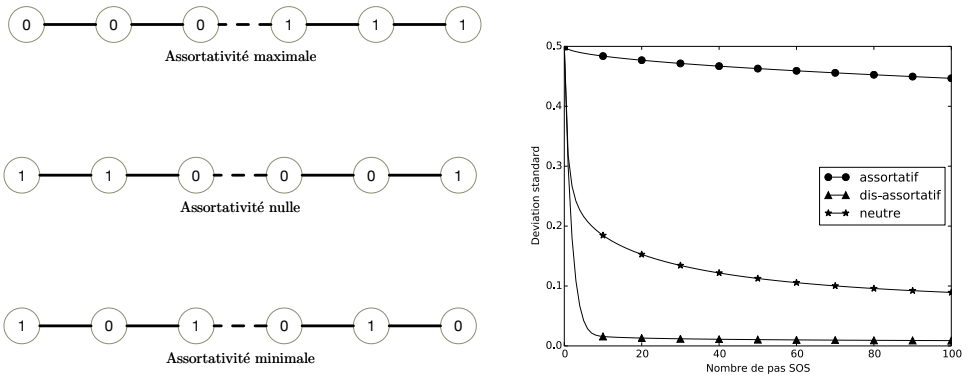
$$r_{G,x} = \frac{\sum_{uv} uv(e_{uv} - a_u.a_v)}{\sigma_a^2}$$

où e_{uv} est la proportion de lien reliant deux nœuds dont les valeurs u et v sont issues de x, a_u est la proportion des liens attachés a la valeur u et σ_a^2 est la déviation standard de la distribution des a_u.

Pour illustrer notre propos, nous prenons le cas d'une topologie en ligne de 100 nœuds dont les valeurs de mesures sont dans $\{0, 1\}$. Nous prenons une distribution telle qu'une moitié des nœuds possède la valeur 0 et une autre moitié la valeur 1. Puis nous considérons plusieurs types de répartitions :

— assortativité maximale : si $i > 50$ alors $x_i = 1$ sinon x_i=0
— dis-assortativité maximale : si $x_i = i \bmod 2$
— assortativité nulle : x_i est le résultat d'un essai de Bernouilli.

Les résultats obtenus pour un consensus avec l'heuristique SOS sont illustrés sur la figure 3.19.



(A) États initiaux considérés (B) Consensus avec l'heuristique SOS

FIGURE 3.19: Illustration de l'importance de l'assortativité

Il est clair que la plage de valeurs d'assortativité d'un vecteur de mesure dépend à la fois de la distribution de ce vecteur et de la topologie du réseau. Ces deux paramètres conditionnent toutes les répartitions possibles des valeurs dans le graphe. Dans notre simple exemple de la ligne, l'assortativité maximale est obtenue en créant 2 camps alors que l'assortativité minimale s'obtient en intercalant toujours 2 valeurs différentes. Les valeurs maximales et minimales vont donc dépendre de la déviation standard (nombre de 0 ou de 1). S'il n'y a pas autant de 0 que de 1, il ne sera pas possible d'atteindre une valeur de $r = -1$. S'il y a au moins un 1 et un 0, il ne sera pas possible d'obtenir la valeur $r = 1$.

Mise en évidence expérimentale

Afin de mettre en évidence l'impact de l'assortativité sur la convergence du consensus nous avons considéré la relation entre

(i.) Le rapport des déviations standards entre deux étapes de consensus $\frac{std(W_h.x}{std(x)}$ pour une heuristique quelconque

(i.) L'assortativité initiale du vecteur de consensus $r_{G,x}$.

Étant donné que la déviation standard nous indique la distance restant à parcourir avant d'atteindre le consensus, le premier élément peut être vu comme le pourcentage de cette distance qui a été parcourue en une étape. Le deuxième élément caractérise l'homogénéité de la répartition des valeurs dans le graphe.

Pour réaliser une telle expérience, nous avons besoin de contrôler le niveau d'assortativité des valeurs dans un graphe afin de voir l'impact sur une étape de consensus. Pour ce faire nous avons uniquement considéré les valeurs 0 et 1 et nous les avons réparties en utilisant l'algorithme 4 de marche aléatoire que nous avons conçu spécifiquement pour le besoin de l'expérience.

Le principe de cet algorithme est de parcourir aléatoirement le graphe en distribuant une valeur à chaque nœud rencontré, cette valeur dépendant de la valeur précédemment donnée. Nous pouvons parcourir le graphe simultanément par plusieurs marches. En changeant la dépendance des valeurs successives (p) et le nombre de marches simultanées (nb_{source}), il nous est possible d'avoir un impact sur la déviation standard des nœuds ainsi que sur l'assortatitivité. Pour nos expérimentations, nous avons utilisé une seule marche, et nous avons réglé le paramètre $p > 0.5$ pour une assortativité positive et $p < 0.5$ pour une assortativité négative.

Nous avons effectué des tests sur toutes les topologies probabilistes présentées précédemment (Erdos-Renyi, Euclidienne, Barabasi-Albert (m=1), Barabasi-Albert (m=2), et Watts-Strogatz) pour une taille fixe de 100 nœuds. Nous avons généré 100 graphes par modèle pour chaque valeur p allant de 0.1 à 0.9 avec un pas de 0.1. Les résultats obtenus montrent une relation claire entre la réduction de la déviation standard et l'assortativité des valeurs indépendamment de l'heuristique utilisée. Chaque topologie possède sa propre relation. La figure 3.20 présente pour chaque heuristique le lien entre la réduction de la déviation standard et l'assortativité des valeurs initiales en fonction de la topologie. On constate que la plage d'assortativité est plus réduite pour les graphes de types Euclidien, Watts-Strogatz, et Erdos-Renyi, tandis qu'une plus grande plage de valeur est possible pour les graphes de types Barabasi-Albert. Si les heuristiques BCW, NW, MDW et MW semblent se comporter linéairement, ce n'est pas le cas de l'heuristique SOS. Les premières heuristiques tendent à avantager les cas dis-assortatifs et désavantager les

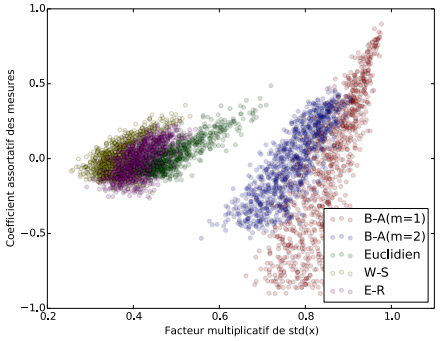
Algorithme 4 repartir_valeurs(G, p, $nb\_sources$, ϵ)

```
marked ← 0
next ← 0
{Initialisations indépendantes des états}
for i ∈ [1, n] do
  x_i ←pick(0,1)
end for
{Choix des sources ou point de départ}
sources ← pick([1, n], nb_sources)
for i ∈ sources do
  x_i ← 0
  mark(i)
  marked+=1
end for
{Tant que l'on n'a pas traité suffisamment de nœuds}
while 1 − marked/n > ε do
  for s ∈ sources do
    targets ←{t ∈ V(s)| is_not_marked(t)}
    {Se déplacer dans un état voisin non marqué et fixer sa valeur si possible}
    if targets ≠ ∅ then
      next ← t_1
      mark(next)
      marked+=1 {La valeur de p influe sur l'assortativité finale}
      if rand_uniform(0,1) < p then
        x_next ← x_s
      else
        x_{t_1} ← 1 − x_s
      end if{Sinon se déplacer dans un état voisin}
    else
      next ←pick(V(s), 1)
    end if
    replace(sources,s, next)
  end for
end while
```

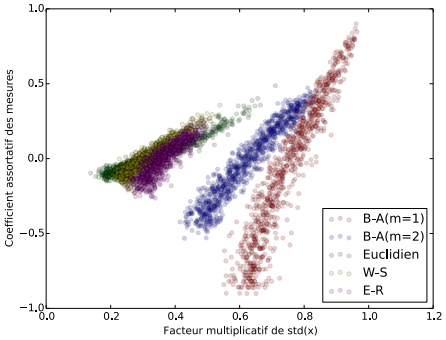
cas assortatifs comme attendu. L'heuristique SOS, offre une meilleure réduction que ses concurrentes pour les cas assortatifs mais n'avantage pas plus les cas neutres que les cas dis-assortatifs, pour lesquels ses concurrentes semblent plus performantes.

Améliorer la convergence par une matrice de permutation

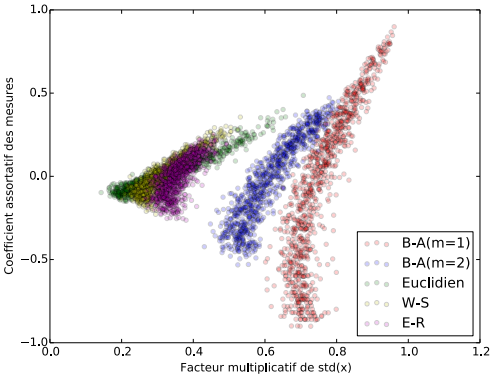
Dans le cas d'une mesure sur un réseau de communication, il y a de fortes chances pour que les mesures des différents nœuds soient assortatives. En effet, on peut se douter que la mesure d'un délai vers une même destination soit proche pour deux nœuds voisins. Comme nous l'avons vu, la répartition assortative des mesures au sein du graphe peut



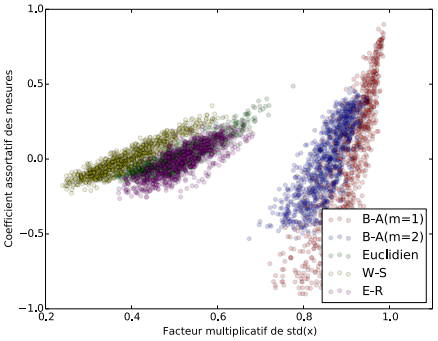
(A) Heuristique BCW



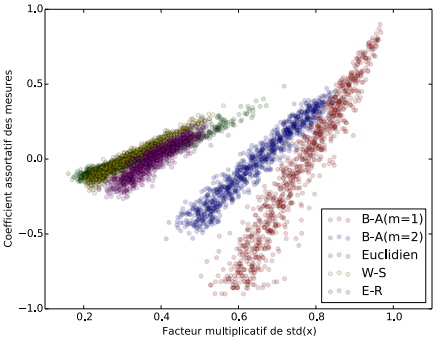
(B) Heuristique NW



(C) Heuristique SOS)



(D) Heuristique MDW



(E) Heuristique MW

FIGURE 3.20: Relation entre la réduction de la déviation standard et l'assortativité des mesures pour différentes heuristiques

avoir un impact négatif non négligeable sur le temps de convergence. Ceci est un facteur à prendre en compte lors de l'établissement du graphe de communication entre les nœuds de mesures. Il sera préférable de faire communiquer des nœuds dis-assortatifs, comme cela est souvent fait dans les protocoles de type gossip [5]. Aussi, nous avons remarqué que dans certains cas, il était possible de précéder le consensus par une étape de permutations des valeurs de mesures. Ceci est faisable en utilisant différentes matrices de permutation. Une telle approche a certes un coût supplémentaire, puisque durant

5. Un nœud accepte un échange avec le voisin ayant la valeur la plus éloignée de la sienne

cette phase de permutation la déviation standard reste identique, mais qui peut s'avérer rentable lorsque l'assortitivité est très élevée. Néanmoins, ceci est plus souvent le cas pour les réseaux de type ligne ou grille et plus rare dans le cas de graphes probabilistes.

3.8 Conclusion et résumé des contributions

Dans ce chapitre nous avons proposé un cadre algébrique permettant de formaliser et de comparer diverses techniques de supervision et de mesure distribuées. Nous avons modélisé une opération de mesure distribuée par une fonction $f : I^n \to I^n$ qui pour chaque nœud i du réseau fournit une image $f_i(x)$ dépendant de la totalité du vecteur de mesure. Nous nous sommes consacrés aux cas où f était linéaire, qui peut être traité comme un produit de matrices pour lesquelles chaque coefficient représente un échange sur le réseau. Le coût et la latence d'une mesure ont également été définis en fonction de ce produit de matrice. Par la suite nous avons montré que la réalisation de ce produit reposait sur deux services séparés qui sont le `Transport` d'une information sur un lien et la `Distribution` de celle-ci. Ces services pourront être composés en fonction de la technologie réseau sous-jacente.

Face à l'évolution de nos réseaux de plus en plus dynamiques, nous nous sommes concentrés sur les graphes dynamiques. Pour ces derniers nous avons suivi une organisation de type consensus et nous avons proposé l'heuristique SOS offrant des performances supérieures à celles de l'état de l'art. Par la suite nous avons montré quelles étaient les propriétés importantes d'une mesure qui impactaient sa propagation. Nous avons notamment relevé que l'assortativité d'une mesure dans le graphe était déterminante. Ce paramètre est conditionné par la distribution de la mesure et la topologie du graphe. Enfin pour ne pas subir l'assortativité inhérente aux réseaux de communication, nous avons proposé d'entamer un consensus par un mélange des valeurs pour réduire cette assortativité et accélérer le consensus.

Nous avons donc dans ce chapitre réduit la latence et le coût d'une mesure essentiellement pour les réseaux dynamiques en travaillant au niveau du service de `Distribution`. En ce qui concerne les réseaux statiques, nous avons montré que cette couche pouvait être optimisée, notamment en se basant sur le calcul d'un arbre de recouvrement minimum. Il reste néanmoins possible d'améliorer la couche dite de `Transport`, qui dans le cas d'un réseau fixe est bien souvent le protocole TCP.

Ainsi nous nous consacrerons dans le prochain chapitre à réduire la latence d'un lien pour un coût donné, lorsque celui-ci utilise la couche TCP.

Chapitre 4

Non-intrusivité et ordonnancement des mesures sur un lien

4.1 Motivations

Dans le chapitre précédent, nous avons étudié le problème de la mesure distribuée et nous avons montré que ce problème peut être modélisé par un produit matriciel. La latence d'une mesure est liée d'une part au nombre d'étapes de calculs (où nombre de matrices dans le produit), et d'autre part au temps nécessaire à chaque échange pour un calcul. Nous avons ensuite décomposé ce problème en deux services différents : `Distribution` et `Transport`, organisés autour d'une donnée de `Mesure`. Le premier service est en charge de l'organisation du calcul distribué tandis que le second est responsable du transfert de l'information.

Jusqu'à présent, nous nous sommes intéressés au service `Distribution` et nous avons proposé une façon de réduire la latence pour les réseaux dynamiques en utilisant une nouvelle heuristique pour réduire le nombre d'étapes de calcul.

Dans ce qui suit, nous nous focalisons sur le service de `Transport`, chargé de transférer une information d'un agent de mesure à un autre agent de mesure. Lors d'un transfert, le service de `Transport` accède à une ressource bien souvent partagée pouvant aller d'un simple lien de communication sans fil, à une route passant par plusieurs routeurs d'Internet. L'acquisition d'une partie de cette ressource est dictée par un algorithme distribué (contrôle d'accès au médium, contrôle de congestion, etc.). Nous cherchons ici à optimiser l'utilisation d'une ressource acquise par un tel algorithme et cette ressource est liée à un couple d'agents de mesure.

Lorsque plusieurs opérations de mesure distribuée sont menées en parallèle, plusieurs informations doivent transiter sur un même lien de communication. Certaines mesures peuvent être plus importantes que d'autres, ou bien nécessiter de faire transiter davantage d'information. Afin de gérer au mieux ces transferts, nous avions évoqué dans le chapitre précédent (3.5.3) la possibilité d'effectuer de la différentiation au niveau de la couche `Transport`. Cette notion de priorité est notamment utile pour :

- Minimiser la latence moyenne des transferts en ordonnançant ceux-ci par taille
- Minimiser la latence des mesures les plus importantes en les favorisant
- Contrer l'impact négatif de l'assortativité d'une mesure en la favorisant.

Nous considérons le cas dans lequel une mesure est échangée via une connexion TCP, tel que cela serait fait en pratique sur un réseau IP. Afin de pouvoir réduire la latence moyenne d'un transfert et de pouvoir gérer différents niveaux de priorités, nous étudions la possibilité d'ordonnancer des flux TCP de manière non intrusive et transparente pour le réseau. Au-delà de l'ordonnancement des mesures, ceci nous permettrait de :

— Contrôler la part des ressources allouées aux données utilisateur, à la mesure et au contrôle du réseau.
— Fournir différents niveaux de qualité de services pour différents besoins applicatifs.
— Réduire l'intrusivité de la mesure active en la classant comme *less than best effort*[1]

Pour les flux d'un couple *origine-destination* (ou *route*) matérialisant deux agents de mesures, l'objectif est donc d'ordonnancer efficacement leurs segments TCP sans modifier les ressources réseau utilisées par ces flux, ce qui, en retour, garantit une certaine transparence pour les autres participants et applications du réseau. Appliquer un ordonnancement par *route* est un moyen d'éviter toute influence néfaste sur une autre *route* du réseau. Bien entendu, la réalisation d'un tel ordonnancement ne se limite pas nécessairement aux flux de mesures mais reste applicable sur l'ensemble des flux transitant entre deux nœuds. Nous montrons que pour réaliser une telle gestion interne sans restrictions sur le comportement applicatif, une condition nécessaire et suffisante est que les mémoires tampons (ou *buffers*) en émission et en réception soient infinies pour TCP.

Nous montrons ensuite que si le canal de communication est sans perte ni déséquencement, le temps de transfert moyen des mesures (ou flux TCP) peut être minimisé par un ordonnanceur qui prend en compte des informations de contexte sur ces mesures comme leurs tailles. Nous proposons alors un nouvel ordonnanceur, `FAIR`, qui garantit que le temps de transfert de chaque mesure pour un couple *origine-destination* soit réduit. Nous développons `SCHED_TCP`, une implémentation de cette solution dans le but d'évaluer les potentielles améliorations dans le cas d'un réseau réel pour lequel les conditions théoriques de faisabilité ne sont pas réunies. Nos expériences montrent que les gains de performance de `SCHED_TCP` peuvent être considérables. Une telle approche tient le passage à l'échelle et pourrait être déployée dans le cadre d'un réseau *overlay* de supervision mais également de manière progressive sur un réseau comme Internet afin d'améliorer la performance des flux pour chaque couple *origine-destination*.

Après avoir donné en section 4.2 un bref aperçu des travaux relatifs, le reste du chapitre est organisé de la sorte : en section 4.3 nous développons *gtcp*, un algorithme de contrôle de congestion générique qui permet un arrangement non intrusif des mesures ou du contenu applicatif au sein des flux partageant une même route. Nous établissons également qu'il est nécessaire et suffisant pour implémenter un tel ordonnancement arbitraire d'avoir des *buffers* d'émission et de réception infinis.

Dans un second temps, nous appliquons en section 4.4 les résultats classiques issus de la théorie de l'ordonnancement à notre contexte, et nous montrons qu'en l'absence de

1. Dans le cas d'une mesure active, un trafic est volontairement injecté afin de mesurer un délai ou une bande passante disponible, ce trafic peut consommer de la ressource utile pour un utilisateur.

perte et de dé-séquencement, certaines politiques basées sur le nombre de données à transmettre ou bien déjà transmises minimisent le nombre de mesures en cours (flux TCP actifs). Nous construisons également `FAIR`, un ordonnanceur qui (sous l'hypothèse qu'il est possible de savoir quelle mesure aurait été transférée en premier avec TCP), garantit que le temps de transfert pour chaque mesure soit réduit.

Enfin nous présentons `SCHED_TCP` en section 4.5, une implémentation de *gtcp* en couche session qui le rend plus facilement déployable par de nouveaux utilisateurs. Nos expérimentations montrent qu'une telle solution améliore la performance du service perçue par la couche applicative, tout en offrant une flexibilité d'usage.

Il est important de souligner que tous les points positifs énoncés précédemment ne dépendent pas de la version de TCP utilisée et que *gtcp* comme `SCHED_TCP` peuvent aussi bien fonctionner avec New Reno [97], CUBIC [98] ou Compound [99].

4.2 Travaux similaires

Ces vingt dernières années ont été le théâtre d'un effort considérable pour comprendre et optimiser la performance des algorithmes de contrôle de congestion sur Internet. Plusieurs variantes de TCP ont été proposées et leurs interactions avec la dynamique des *buffers* ont été l'objet d'études poussées. Sans pour autant être exhaustif, la communauté scientifique a cherché des moyens d'améliorer la performance de TCP en modifiant plusieurs aspects de l'augmentation additive-réduction multiplicative (ou *AIMD* de l'anglais *Additive Increase-Multiplicative Decrease*) de TCP ou en modifiant la façon dont les paquets sont gérés dans les *buffers*. Un important axe de recherche, promu par les défenseurs de l'architecture diffserv, a été notamment d'étudier comment améliorer la qualité d'expérience en implémentant une différentiation de service et la notion de priorité dans les routeurs, voir [100]. Plusieurs travaux ont également montré que l'implémentation d'une politique d'ordonnancement au sein des routeurs peut améliorer la performance, particulièrement dans les cas de surcharge ou pour des régimes proches de la saturation, c'est le cas dans [101] et [102]. Néanmoins, un argument contraire est présenté dans [103], où les auteurs montrent que la performance dans un réseau dans lequel chaque nœud implémente un ordonnancement basé sur la taille des paquets peut s'avérer être extrêmement faible.

Il existe deux principaux axes de recherches liés à notre proposition, l'un traite de la modification et des améliorations de la pile TCP, et l'autre de l'ordonnancement appliqué aux réseaux. Nous couvrons brièvement les principaux résultats de ces deux domaines.

Le nombre de modifications de TCP qui ont été proposées depuis sa création en 1974 est indénombrable. Parmi les plus récentes et significatives on citera : la modification de la taille initiale de la fenêtre de congestion [104], une accélération au départ [105], une gestion plus efficace de la perte de paquets avec un protocole de notification de congestion (ECN pour Early Congestion Notification) [106, 107], l'acquittement sélectif (SACK) [108], le protocole RCP [109], ou encore l'amélioration de l'estimation de la bande passante tel que proposée avec TCP Westwood [110]. Il y a eu des douzaines de nouvelles versions proposées, la version courante sous Linux étant TCP CUBIC [98], alors que celle sous le système d'exploitation Windows est COMPOUND [99]. Ces versions de TCP modifient seulement l'algorithme de contrôle de congestion, mais n'ajoutent pas de nouvelles fonctionnalités. Une approche plus récente d'aborder les protocoles de transport est de séparer les services, en plusieurs sous-couches qui peuvent être partagées entre les applications ou entre les connexions [111]. Cette approche a été suivie par MPTCP [112] qui permet d'améliorer la bande passante en utilisant toutes les interfaces réseau disponibles et leurs chemins associés entre deux hôtes.

L'architecture diffserv a été décrite dans [100]. Dans cette approche, chaque paquet est classé en un nombre limité de classes de trafic, et chaque routeur sur le réseau est configuré pour différencier le trafic en se basant sur ces classes. On en trouve un exemple dans [113] où le trafic interactif se voit prioritaire sur le reste. Dans le but de préserver la qualité de service, d'autres pistes ont été évoquées comme le contrôle d'admission [114] ou la gestion active de file d'attente (AQM pour Active Management Queue). Au-dessus de la couche de transport, des travaux tels que [115] ont été entrepris pour ordonnancer le trafic applicatif dans les fermes de serveurs. Une autre approche, telle que suivie par [116], [117] et [101] aura été d'étudier l'impact d'un ordonnancement basé sur la taille des flux pour un serveur. Ces derniers travaux ont montré que le fait de donner la préférence aux flux les plus courts est bénéfique. En ce qui concerne les résultats les plus connus, on retiendra que parmi les politiques connaissant la taille des données à transmettre, celle qui donne la priorité au flux ayant la taille à transmettre la plus courte (`SRPT` pour Shortest Remaining Processing Time) minimise le nombre de flux actifs dans un système [118]. Parmi les politiques ignorant la taille des données à transmettre, celle qui donne la priorité au flux ayant été le moins servi (`LAS` pour Least Attained Service), minimise stochastiquement le nombre de flux actifs dans le système lorsque la distribution de la taille des flux est hyperexponentielle (DHR pour Deacreasing Hazard Rate), voir [119]. Nous renvoyons le lecteur vers [120] pour une étude plus approfondie sur les résultats issus de la théorie de l'ordonnancement.

Toutes ces études proposent une amélioration *localement*, néanmoins, le comportement obtenu *globalement* lors d'un déploiement complet sur un réseau à l'échelle d'Internet n'est pas clairement caractérisé. Par exemple, si la modification d'un protocole implique

une augmentation de la bande passante pour un ensemble de flux, ceci implique une diminution des ressources pour les flux restants. Ceci nous incite à suivre une approche différente.

Nous nous concentrons sur deux nœuds, l'un *origine* et l'autre *destination* au sein du réseau, et nous développons une solution pour implémenter une politique d'ordonnancement de flux TCP de manière non intrusive et transparente pour le reste du réseau. En d'autres termes, les segments TCP d'un couple *origine-destination* sont ordonnancés sans modifier la bande passante allouée par le réseau à cet ensemble de flux. Ceci nous permet de construire des ordonnanceurs qui visent à améliorer la performance d'un ensemble de flux partageant les mêmes ressources réseaux, sans pénalité aucune pour les autres flux d'Internet.

Nos travaux se rapprochent de ceux menés dans [101, 115–117], mais on soulignera deux différences majeures. La première provient du fait que notre approche garantit que la bande passante allouée à un couple *origine-destination* reste inchangée par rapport à une utilisation standard de TCP. Le second point résulte dans la capacité à supporter un ordonnancement arbitraire, pas nécessairement basé sur la taille des flux. Dans notre approche, les fonctions de contrôle de congestion et celle de l'ordonnancement sont découplées et pour tout couple *origine-destination*, nous sommes libres de choisir comment les flux sont ordonnancés. L'utilisation de la taille des flux est séduisante par le fait qu'elle nous permet de dériver des propriétés mathématiques intéressantes, mais d'autres solutions pourraient être envisagées, telles que donner la priorité à un flux temps réel, multimédia, interactif ou encore de supervision sur un flux dit *best-effort*.

4.3 Découpler contrôle de congestion et ordonnancement

Dans cette section, nous montrons qu'il est possible d'ordonnancer les segments d'un ensemble de mesures ou flux applicatifs partageant la même route, sans modifier la bande passante qu'une implémentation standard de TCP aurait acquise. Dans ce qui suit, nous utiliserons le terme *tcp* pour faire référence à une implémentation standard du protocole, sans tenir compte d'une version spécifique.

4.3.1 Une première intuition

Afin de mieux établir la notion d'ordonnancement non intrusif par route nous proposons l'exemple illustré par la figure 4.1. Dans cet exemple, deux stations sources communiquent avec deux stations destinations différentes et partagent un même goulot

d'étranglement. La première station transfère une mesure (flux 1) d'une taille de 40 unités de données. Pour l'expérience, la seconde station transfèrera deux mesures (flux 2 et 3) respectivement de taille 15 et 4 unités. Le goulot d'étranglement ne peut transmettre que 2 unités de données par unité de temps et les sources démarrent de manière séquentielle (1, 2 puis 3). Pour l'illustration nous négligerons les phases de contrôle de congestion. Dans le cas de TCP, à l'arrivée de chaque flux, la bande passante est divisée et une part égale est donnée à chaque flux. Cette situation est illustrée dans le cadre supérieur droit. Une première erreur serait d'ordonnancer les flux applicatifs de la station 2 au travers d'une seule *socket*, c'est le cas illustré dans le cadre inférieur gauche. En effet, la bande passante acquise par la source 2 diminue avec le nombre de flux générés par la source 1 utilisant le goulot d'étranglement. Au contraire, un ordonnancement non intrusif par route est illustré dans le cadre inférieur droit de la figure 4.1. Comme on peut le constater, la bande passante allouée au flux 1 est inchangée tout le long de son transfert. Néanmoins, nous avons transférer la mesure 3 avant la mesure 2 ce qui a réduit le temps moyen de transfert pour la source 2.

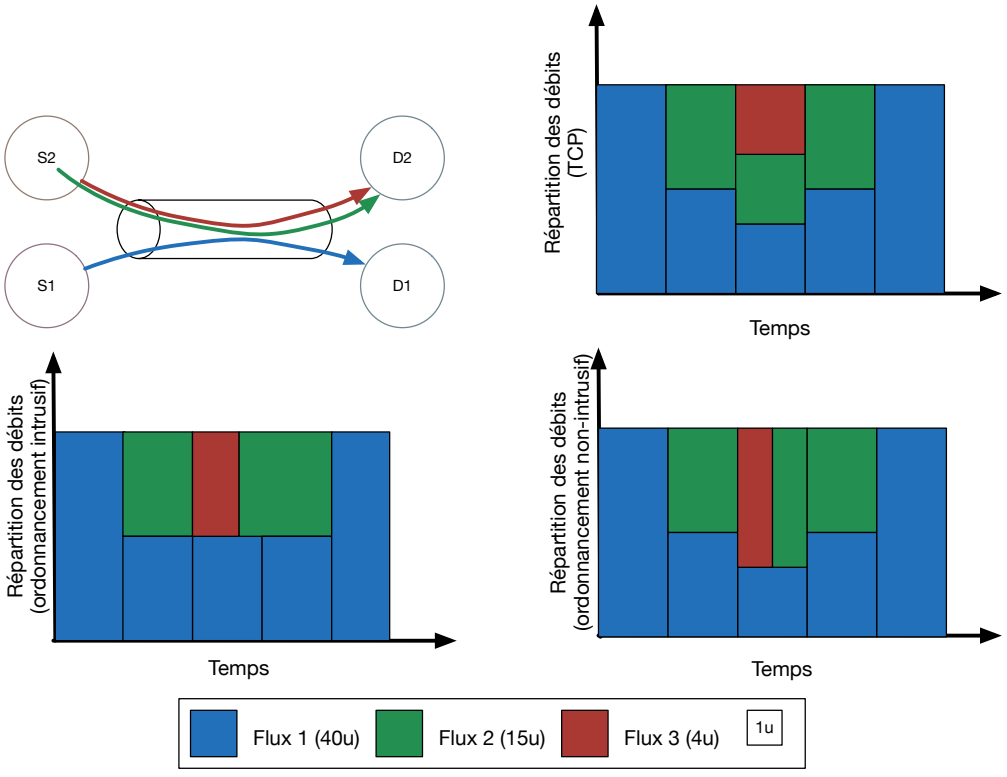


FIGURE 4.1: Un exemple simplifié d'ordonnancement non-intrusif

En d'autres termes, si $V^{tcp}(t)$ représente le volume de trafic qui a été introduit par TCP dans une route donnée jusqu'au temps t, nous montrerons que sous la condition d'avoir des *buffers* d'émission et de réception infinis, un contrôle de congestion plus général *gtcp* peut changer de manière arbitraire le contenu des segments en utilisant une politique

π, de telle sorte que $V^{gtcp_\pi}(t) = V^{tcp}(t)$, $\forall t$. Cette dernière propriété implique que *gtcp* est non intrusif relativement au protocole de référence *tcp*, ceci laissant la possibilité d'optimiser la qualité de service perçue par cet ensemble de flux, sans infliger aucune modification sur les autres flux du réseau. Afin de construire un tel protocole *gtcp*, nous aurons besoin d'être capable de *simuler* à tout instant, l'état des flux si l'on avait utilisé *tcp*. Réciproquement, ceci permettra à *gtcp* d'introduire les segments dans le réseau exactement aux mêmes moments que l'aurait fait *tcp*. Nous insistons sur le fait que l'encapsulation de plusieurs flux applicatifs dans un seul ou plusieurs flux TCP sans attention particulière ne permet pas de satisfaire ce but.

Dans le but pour *gtcp* d'être capable d'implémenter un ordonnancement arbitraire π, *gtcp* aura besoin de conserver deux représentations de l'état d'un flux : la représentation *virtuelle* qui garde en mémoire l'état sous *tcp* et la représentation *réelle*. À chaque fois qu'un paquet sera envoyé (ce qui dépendra de l'état virtuel), l'ordonnanceur π choisira le flux qui sera servi en réalité (ce qui pourra dépendre de l'état *réel*, *virtuel* et d'autres informations de contexte telles que la qualité de service). Pour atteindre cet objectif, nous montrerons qu'il est suffisant pour *gtcp*, d'encapsuler dans tous ses segments, une information sur le flux TCP virtuel (ce qui permettra de conserver la valeur de l'état *virtuel* d'un flux), et sur le flux réel (ce qui permettra de mettre à jour l'état *réel* de ce flux).

Le reste de la section est organisé de la manière suivante. En sous-section 4.3.2 nous décrivons comment *gtcp* peut être construit de manière non intrusive, pour n'importe quelle politique d'ordonnancement π en décrivant en détail les algorithmes correspondant au traitement des événements réseau, à savoir l'arrivée d'une donnée applicative (*wri*), la réception d'un acquittement (*ack*), et l'expiration d'un timer de retransmission (*rto*). En sous-section 4.3.3, nous énonçons le résultat formel, dans lequel l'établissement d'une politique π telle que $V^{gtcp}(t) = V^{tcp}(t)$, $\forall t$ peut-être implémentée, sous réserve d'avoir des tailles de *buffer* non bornées.

4.3.2 Une Approche algorithmique : maintenir l'état de *buffers* virtuels

Tel que nous l'avons illustré dans le plan supérieur de la figure 4.2, TCP peut se modéliser comme un système de plusieurs files et de plusieurs serveurs, où chaque serveur est associé à une file. Notre approche est de découpler les files des serveurs avec une politique d'ordonnancement π tout en garantissant que l'activité des serveurs reste inchangée. Dans notre convention, un service (ou numéro de séquence) est la transmission d'un segment, il démarre avec la première tentative de transmission et se termine sur la

réception de son acquittement (*ack*). Le service peut être préempté sur réception de trois *ack*'s dupliqués, ce qui déclenchera π pour choisir le segment (pas nécessairement le même) à transmettre. Par conséquent, un serveur offre plusieurs services en parallèle, dont la durée et l'ordre de terminaison dépendent du réseau et de *tcp*. Le volume de trafic total produit par les serveurs est capturé par la fonction $V^{tcp}(t)$.

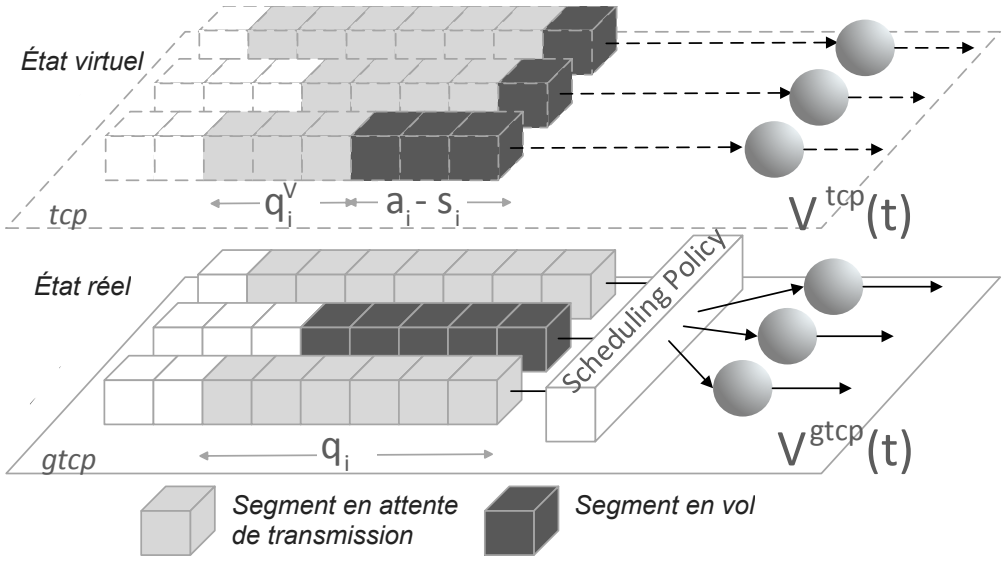


FIGURE 4.2: Les deux plans de *gtcp*. Alors que dans le plan inférieur, la seconde file est prioritaire, nous maintenons dans le plan supérieur l'état virtuel des files.

Nous considérons que la connexion est établie, sans option particulière activée. Dans un souci de clarté, nous faisons l'hypothèse que les tailles l des arrivées dans les files d'attente sont des multiples de la taille maximum de segment mss (de l'anglais Maximum Segment Size). En conséquence, tous les paquets ont une taille maximum de segment. Cette hypothèse n'est pas vraiment un prérequis pour construire l'algorithme de *gtcp*, mais simplifie grandement l'exposé. La façon de gérer des segments de taille variable est discutée en remarque 4.2.

Pour *tcp*, la possibilité pour un serveur i de transmettre un nouveau segment dépend de ses variables d'état qui sont :

- q_i : La taille de la file de transmission i
- s_i : Le plus grand numéro de séquence en vol
- a_i : Le plus grand numéro d'acquittement cumulatif
- $cwnd_i$: La taille de la fenêtre de congestion
- $rwnd_i$: La taille de la fenêtre de réception (sur l'hôte destinataire)

De ces variables d'état, le standard RFC-5681[121] dérive un crédit d'émission par flux. Le crédit d'émission du flux i, noté cr_i, est tel que $cr_i = \min(cwnd_i, rwnd_i) + a_i - s_i$. Pour être conforme au standard, un nouveau segment ne peut être envoyé par le serveur i si $cr_i = 0$ ou si $q_i = 0$. Ici, Avec *gtcp*, nous découplons les files des serveurs, en conséquence,

q_i dépendra de π. Pour pallier ce phénomène, nous introduisons en plus de q_i, la variable q_i^v, la taille virtuelle de la file i, celle qui renferme la valeur de q_i telle qu'elle l'aurait été sous *tcp*.

Un serveur i évalue alors son état pour offrir de nouveaux services, sur l'arrivée d'événements qui peuvent être soit (i) $wri_i^l : l$ octets arrivent dans la file i ou (ii) $ack_{i,j}^{s,s'}$: le segment s' de la file j est acquitté avec le service s du serveur i. L'algorithme 5 détaille les actions à entreprendre sur l'occurrence de l'événement wri_i^l.

Algorithme 5 on_wri(i, l)

$q_i^v \Leftarrow q_i^v + l$ // $q_i \Leftarrow q_i + l$
while $cr_i > 0$ and $q_i^v > 0$ **do**
  serve(i, s_i)
  $s_i \Leftarrow s_i + mss$ // $cr_i \Leftarrow cr_i - mss$
  $q_i^v \Leftarrow q_i^v - mss$ // $q_j \Leftarrow q_j - mss$
end while

Lors de l'arrivée de l données dans la file i (pour rappel, l est proportionnel à mss), le serveur i met à jour la taille de sa file virtuelle q_v^i pour compenser l'arrivée dans sa file d'attente réelle q_i, puis, il offre de nouveaux services jusqu'à atteindre sa limite en se basant sur sa file virtuelle. Après chaque service, le serveur i met à jour son état comme suit : il incrémente son numéro de séquence (ce qui diminue son crédit d'émission), décrémente la taille de sa file virtuelle q_i^v. Cette dernière opération signifie que le serveur i transmet virtuellement les données de la file i et considère que la file réellement servie reste inchangée. Les actions entreprises dans le cas où un service s est offert par le serveur i sont données dans le pseudo-code de l'algorithme 6.

Algorithme 6 serve(i, s)

$j, s' \Leftarrow$ schedule() // implemente π
$P_{i,j}^{s,s'} \Leftarrow$ tcp_head(i, s) | sched_head(j, s') | data()
send$(P_{i,j}^{s,s'})$ // produit l'événement $sen_{i,j}^{s,s'}$
set_timer$(P_{i,j}^{s,s'})$ // planifie l'événement rto_i^s
associate$((i, s), (j, s'))$
mark(j, s')

Quand le serveur i est sur le point de transmettre un segment s, il demande en tout premier lieu à la fonction externe `schedule`, quel segment s' de quelle file j il doit réellement servir. La fonction `schedule` implémente la politique de scheduling π spécifiée par *gtcp*. Par la suite le serveur forge un paquet contenant l'identification du service (i,s), l'identification de la donnée (j, s') et la donnée elle-même. Les deux identifiants sont associés et le segment de données est marqué comme étant *en vol* pour l'ordonnanceur. Nous associons à la transmission, la production de l'événement $sen_{i,j}^{s,s'}$. Notons ici que

la politique d'ordonnancement de *tcp* est un cas particulier de *gtcp* où l'ordonnanceur π assigne systématiquement $(j, s') := (i, s)$.

Côté récepteur, le pendant de l'événement $send_{i,j}^{s,s'}$ est l'événement $rcv_{i,j}^{s,s'}$. Lorsque cet événement survient, le récepteur produit un événement d'acquittement $ack_{i,j}^{s,s'}$ qui annonce par la même occasion la taille de sa fenêtre de réception $rwnd_i$. En plus de cet acquittement, il garde l'association (i, s) et (j, s'). Dans le cas d'une réception multiple d'un même service (perte d'acquittement, mauvaise estimation du temps d'aller-retour (RTT pour round-trip time), la même association est immédiatement acquittée. Comme nous l'avons expliqué précédemment, l'événement $ack_{i,j}^{s,s'}$ peut déclencher une transmission de paquet par le serveur i, la réaction à un tel événement est décrite dans l'algorithme 7.

Algorithme 7 on_ack(i, s, j, s')

```
ack_type ⇐ up_cong_state()
if ack_type = third_dupack then
  serve(i, s)
else
  if ack_type = cumulack then
    unmark_all_associations_of((i, s))
    remove_all_associations((i, s))
    free_tx_buf((j, s'))
  end if
  while cr_i > 0 and q_i^v > 0 do
    serve(i, s_i)
    s_i ⇐ s_i + mss              // cr_i ⇐ cr_i − mss
    q_i^v ⇐ q_i^v − mss          // q_j ⇐ q_j − mss
  end while
end if
```

Sur réception d'un *ack*, le serveur i met à jour les variables d'état liées à la congestion. En particulier, le serveur i modifie les valeurs de $cwnd_i, rwnd_i, a_i$ (et par conséquent cr_i), mais réalise également d'autres actions telles que l'annulation de *timer* de retransmission, l'estimation du RTT etc. Étant donné que ces actions dépendent de la version de TCP, nous les encapsulons dans la fonction `up_cong_state` qui retourne le type d'acquittement (autrement dit le nombre de duplications). Si l'acquittement correspond à une troisième duplication, une retransmission est déclenchée, ce qui signifie que le service est préempté. Ainsi, nous pouvons servir une autre application et associer un segment donné supplémentaire à ce service. Si l'acquittement n'est pas une troisième duplication, de nouveaux services sont offerts avec la fonction `serve`. Dans le cas d'un acquittement cumulatif, le *segment de données* acquitté est libéré. Tous les autres segments qui étaient jusqu'alors associés au service sont démarqués et seront reconsidérés par l'ordonnanceur pour une transmission future.

Enfin nous avons introduit dans les algorithmes précédents l'événement rto_i^s (pour retransmission time-out), qui correspond à l'expiration d'un *timer* de retransmission. Cet événement est traité comme la détection d'une troisième duplication d'acquittement par l'émetteur.

4.3.3 Conditions de faisabilité d'un ordonnancement arbitraire

En se basant sur le cadre algorithmique que nous venons de décrire, nous établissons la proposition suivante.

Proposition 4.1. *Quel que soit la séquence d'événements considérée, il est possible d'adopter une politique d'ordonnancement arbitraire π telle que l'on ait $\forall t, \quad V^{gtcp}(t) = V^{tcp}(t)$, si et seulement si les buffers d'émission et de réception ne sont pas bornés.*

Démonstration. Soit $\mathcal{T}^{tcp}(t) = \{t_i \in [-\infty, t[\quad i \in \mathbb{N}\}$ l'ensemble des temps d'occurrence des événements jusqu'au temps t sous l'algorithme *tcp*, avec la convention que $\forall p < q$, $t_p \leq t_q$. De la même manière, notons $\mathcal{T}^{gtcp}(t)$ l'ensemble des temps d'occurrence pour *gtcp*. Pour spécifier quel événement survient à ces instants, nous définissons une fonction $E^{tcp} : \mathcal{T}^{tcp}(t) \to \mathcal{E}^{tcp}$ où $\mathcal{E}^{tcp}$ est l'ensemble des événements possibles qui peuvent survenir au temps t. La fonction E^{gtcp} est définie de manière équivalente. Pour définir $\mathcal{E}^{gtcp}$, nous considérons seulement les événements qui ont un impact sur le comportement de TCP. En conséquence $\mathcal{E}^{gtcp} = \{wri_j^l, ack_{i,j}^{s,s'}, rto_i^s, send_{i,j}^{s,s'}, dat_{i,j}^{s,s'}\}$ où i, j sont respectivement des identifiants de files et de serveur au temps t et $l, s, s' \in \mathbb{N}^3$. $\mathcal{E}^{tcp}$ est identique à $\mathcal{E}^{gtcp}$ avec nécessairement $s = s'$ et $i = j$. Il est aisé de voir que nous avons $V^{gtcp}(t) = |\{t' \in \mathcal{T}^{gtcp}(t) \mid \exists (i,s),(j,s') \text{ s.t. } E^{gtcp}(t') = send_{i,j}^{s,s'}\}| \times mss$, qui signifie que le volume total de trafic introduit jusqu'au temps t sur une route soit égal au nombre de segments envoyés avant t multiplié par leur taille. $V^{tcp}(t)$ est défini de manière similaire.

Par construction des algorithmes 5-7, quelle que soit la politique π de *gtcp*, nous avons $\mathcal{T}^{gtcp}(t) = \mathcal{T}^{tcp}(t)$ et l'équivalence suivante :

$$\begin{aligned} &(\exists \ (i,s) \mid E^{tcp}(t') = send_i^s) \\ \iff &(\exists \ (i,s),(j,s') \mid E^{gtcp}(t') = send_{i,j}^{s,s'}). \end{aligned}$$

Par conséquent, nous avons $V^{gtcp}(t) = V^{tcp}(t)$, $\forall t$, indépendamment de la politique π implémentée par *gtcp*. Afin que cet argument puisse tenir, les buffers d'émission et de réception doivent être infinis. Si les *buffers* étaient de taille finie, il serait possible de

trouver une séquence d'événements telle que sous la politique d'ordonnancement π certains *buffers* soient pleins, entraînant ainsi des pertes alors que leurs *buffers virtuels* eux restent presque vides. Cette situation peut se rencontrer au niveau de l'émetteur, pour une file de faible priorité qui est significativement moins servie qu'avec *tcp*. Au niveau récepteur, la même chose peut arriver pour une file d'attente prioritaire qui reçoit une quantité d'information bien plus grande qu'avec *tcp*, la fenêtre du récepteur devenant potentiellement inférieure à la fenêtre de congestion. Une telle situation modifie la séquence d'arrivée des événements futurs de telle sorte que les algorithmes leur correspondant ne sont pas déclenchés et il n'est alors plus possible de savoir si l'on préserve $\mathcal{T}^{gtcp}(t) = \mathcal{T}^{tcp}(t)$ à partir de ce moment-là. □

Remarque 4.2 (Sur la gestion d'arrivées qui ne sont pas des multiples de mss). Dans ce cas, nous pourrions rencontrer la situation ou la taille d'un service est plus grande que la taille de la file prioritaire. Nous devrions donc encapsuler plusieurs segments de données issus de files différentes au sein d'un même service. Une telle gestion requiert un mécanisme amélioré pour associer un service à un ensemble (ou potentiellement plusieurs ensembles dans le cas de retransmission) de segments de données ayant des tailles différentes.

Dans la prochaine section, nous étudions les possibilités d'appliquer un ordonnancement spécifique, basé sur la taille des flux ou mesures transférées. En se basant sur des résultats classiques d'ordonnancement, nous en déduisons l'optimalité de certaines politiques.

4.4 Ordonnancement par taille de flux : optimalité de politiques

En section 4.3 nous avons montré que le contrôle de congestion et l'ordonnancement sont deux aspects de TCP qui peuvent être complètement découplés, chacun pouvant être géré par un composant différent. Par conséquent, un hôte source peut efficacement ordonnancer des segments TCP sans avoir aucun impact sur les autres connexions actives du réseau. Les propriétés de l'ordonnanceur étant masquées derrière la fonction `schedule` utilisée dans l'algorithme 6.

L'objectif de cette section est d'étudier comment la prise en compte de la taille des flux par un ordonnanceur, pourrait améliorer la performance perçue par les applications. En particulier, nous nous intéresserons à la réduction du nombre de flux actifs sur une route. Dans le cadre de cette section, nous supposerons que toutes les données d'un flux sont disponibles pour être ordonnancées, autrement dit qu'une mesure arrive entière et que par conséquent, sa taille est connue. Si une application transmet des données par le biais

d'une connexion TCP de manière discontinue (telle que pour une application interactive), nous considérerons chacune de ces transmissions comme l'arrivée d'un *nouveau* flux.

4.4.1 Équivalence avec un serveur à capacité variable dans le temps

Notre problème d'ordonnancement est un peu différent de ceux étudiés classiquement dans la littérature. En général, les tâches (ici les flux applicatifs) reçoivent un service fiable et *continu* et partent dès que leur temps de service requis a été reçu. Dans notre cas, les segments sont ordonnancés au moment de la transmission, et un peu plus tard, quand un acquittement est reçu, nous considérons que l'information est bien transmise et qu'une quantité discrète de service a été attribuée.

Dans le but d'établir un résultat d'optimalité, nous devons considérer un canal sans perte ni dé-séquencement. Les résultats de la section 4.5, en présence de pertes et de dé-séquencements, montrent qu'un ordonnancement basé sur la taille des flux amène un gain de performance notable.

Étant donné qu'aucun segment de données ni acquittement n'est réordonné ou perdu, nous pouvons maintenant considérer qu'un segment est transmis avec succès dès qu'il quitte le buffer d'émission et nous pouvons alors dire que le flux quitte le système lorsque son dernier segment de donnée est envoyé. Soit $\mathcal{N}^\pi(t)$ l'ensemble des flux ayant des segments de données à transmettre au temps t et soit $N^\pi(t) = |\mathcal{N}^\pi(t)|$ le nombre de ces flux à ce même moment. L'hypothèse qu'aucun segment ne soit perdu ou déséquencé implique directement que la minimisation de $N^\pi(t)$ est équivalente à la minimisation du nombre de flux actifs mesurés sur réception d'acquittements.

Comme expliqué précédemment (section 4.3), notre approche permet de conserver le nombre total de segments transmis au temps t, $V^{gtcp}(t)$, indépendamment de la politique d'ordonnancement π de *gtcp*. Soit t_i, $i = 1, 2, \ldots$, les moments de transmission de chaque segment sous *tcp*. Nous définissons alors une fonction de capacité $C(t) = mss \times \sum_{i:t_i \leq t} \delta(t - t_i)$ ou $\delta(\cdot)$ est le delta de Dirac. La figure 4.3 illustre la fonction $C(t)$ pour un exemple particulier.

La fonction $C(t)$ peut-être vue comme une fonction capacité variable dans le temps, et capture le taux *instantané* de transmission d'une donnée. Il est clair à présent que pour tout t, $\int_{-\infty}^{t} C(s)\mathrm{d}s = V^{tcp}(t)$, donc en considérant la fonction $C(t)$, un segment est *instantanément* transmis aux temps t_i, $i = 1, 2, \ldots,$.

Le problème de déterminer un algorithme d'ordonnancement optimal π peut maintenant se transformer en un problème d'ordonnancement optimal au sein d'un seul serveur dont

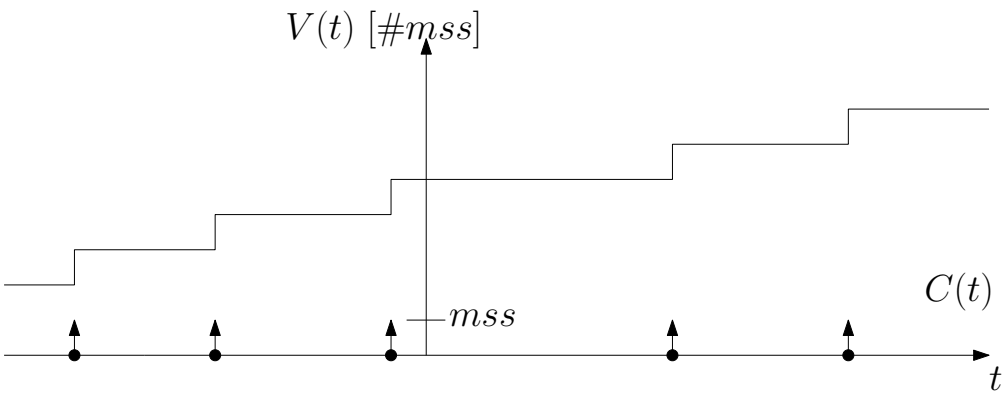


FIGURE 4.3: Fonction $C(t)$ qui capture la transmission instantanée des segments de données. Chaque point représente un moment pour lequel une transmission a lieu.

la capacité est variable dans le temps : $C(t)$. Comme nous le montrerons dans la sous-section 4.4.3, ceci nous permet d'utiliser les travaux classiquement admis dans ce domaine pour dériver des résultats d'optimalité. Il est à noter que nous avons laissé de côté la dépendance de $C(t)$ vis-à-vis de π pour rester cohérent avec la proposition 4.1. Nous introduisons à présent les principaux algorithmes d'ordonnancement dont nous aurons besoin pour le reste de ce chapitre.

4.4.2 Politiques d'ordonnancement

La politique implémentée par *gtcp*, notée π, spécifie le flux servi à un temps t. Le flux choisi recevra un service instantané de valeur $C(t)$. Puisque $C(t)$ est une somme de delta de Dirac, si $t \neq t_i$, alors la valeur du service reçue est 0, et si $t = t_i$, un segment de taille mss est envoyé. En fonction de l'information disponible pour l'ordonnanceur, deux types de politiques peuvent être adoptées : (i) en connaissant le temps de service requis, ou bien (ii) en connaissant le service déjà offert. Les notions de *service requis* et de *service offert* doivent être interprétées ici comme la quantité de données ou *le nombre de segments qu'un flux doit transmettre* et le *nombre de segments déjà transmis par un flux*, respectivement. Nous dérivons des politiques les plus connues les possibilités suivantes :

— Servir le moins servi (`LAS` pour Least Attained Service) est une politique de type (ii) qui sert le flux ayant reçu le moins de service au moment de la décision, ce qui correspond dans notre cas au flux TCP ayant envoyé le moins de segments.
— Servir le plus faible besoin (`SRPT` pour Shortest-Remaining-Processing-Time) est une politique de type (i) qui sert le flux ayant le temps de service restant le plus court.
— Servir pour un temps de séjour *équitable* (`FAIR`) est une politique de type (i) servant le flux qui aurait terminé en premier avec *tcp*.

`FAIR` s'est inspiré de la politique du temps de séjour équitable (FSP pour Fair Sojourn Protocol) issue de [122]. En l'absence de perte et de déséquencement de segment, si l'ordonnanceur connait la taille de la mesure et le comportement de *tcp*, il est possible pour *gtcp* de déterminer, en se basant sur l'évolution du crédit d'émission quel est la mesure qui aurait été transférée en premier avec *tcp*.

4.4.3 Résultat d'optimalité

La proposition 4.3 ci-dessous résume les principaux résultats théoriques qui illustrent comment un ordonnancement basé sur la taille des flux pourrait réduire le nombre de flux actifs sur une route. Nous aurons besoin de la notion de *hazard rate*. Soit $\mathbb{P}(S = k)$ la probabilité qu'un flux requière la transmission de k segments. L'*hazard rate* d'une distribution est alors donné par $h(k) = \mathbb{P}(S = k)/\mathbb{P}(S \geq k)$, et lorsque $h(k)$ diminue avec k, on dit que la distribution est de type DHR (pour *Decreasing Hazard Rate*). La distribution de la taille des flux sur Internet est usuellement modélisée par une distribution de type DHR. C'est le cas par exemple dans [123].

Proposition 4.3. *Sous réserve que pour tout ordonnanceur π implémenté par gtcp l'on ait pour tout t, $V^{gtcp}(t) = V^{tcp}(t)$, aucune perte, et aucun déséquencement, alors nous avons :*

- *Si l'ordonnanceur est `SRPT`, alors $\{N^{\mathtt{SRPT}}(t)\}_{t\geq 0} \leq \{N^{gtcp}(t)\}_{t\geq 0}$, où gtcp peut implémenter n'importe quelle politique arbitraire de type (i).*
- *Si la distribution de la taille des flux est de type DHR, alors $\mathbb{P}(N^{\mathtt{LAS}}(t) > k) \leq \mathbb{P}(N^{gtcp}(t) > k), \forall k$, où gtcp peut implémenter n'importe quelle politique arbitraire de type (ii).*
- *Si l'ordonnanceur est `FAIR`, alors $T_i^{\mathtt{FAIR}} \leq T_i^{tcp}$, où T_i est le temps de transfert du i-ème flux.*

Démonstration. Les preuves se basent sur l'application de résultats connus à notre système *modifié* ayant une capacité de transfert $C(t)$ et nous en fournissons une ébauche.

Pour être plus spécifique, considérons le cas de `SRPT`. Puisque `SRPT` est de type (i), l'ordonnanceur a une information précise sur la quantité de segments que chaque flux a besoin de transmettre au total. Supposons que *gtcp* implémente un ordonnanceur π et que $\mathbf{R}^{gtcp}(t) = (R_n^{gtcp}(t); n \in \{1, 2, \ldots\})$ représente le vecteur *ordonné* du nombre de segments non envoyés au temps t, avec par convention $R_1^{gtcp}(t)$ étant la plus grande valeur. Posons à présent $R^{gtcp}(t)$, le nombre total de segments non envoyés au temps t. Puisque le nombre total de segments $V^{gtcp}(t) = V^{tcp}(t)$ est indépendant de la politique

d'ordonnancement déployée par *gtcp*, il s'ensuit que le nombre total de segments non envoyés, $R^{gtcp}(t) = R^{tcp}(t)$ est également indépendant de π.

En s'appuyant maintenant sur [124, Proposition 1], qui est valable pour n'importe quelle fonction de capacité $C(t)$, on peut montrer que `SRPT` préserve au cours du temps, une relation sur le nombre total cumulé des segments non envoyés. Nous avons donc pour n'importe quelle politique π que pourrait implémenter *gtcp*, $k \in \{1, 2, \ldots\}$:

$$\sum_{n=k}^{\infty} R_n^{\texttt{SRPT}}(t) \leq \sum_{n=k}^{\infty} R_n^{gtcp}(t). \tag{4.1}$$

Nous avons à présent un argument direct pour montrer l'optimalité de `SRPT`. Supposons que $N^{gtcp}(t) = n$. Alors, par (4.1), nous avons $\sum_{j=n+1}^{\infty} R_j^{\texttt{SRPT}}(t) \leq \sum_{j=n+1}^{\infty} R_j^{gtcp}(t) = 0$,, impliquant directement que $N^{\texttt{SRPT}}(t) \leq n = N^{gtcp}(t)$.

La preuve pour `LAS` est une adaptation directe de [119, Theorem 2.1]. L'hypothèse d'une distribution de type DHR pour la taille des flux implique que plus un flux recevra de service, plus son temps moyen de service restant sera grand. Ceci explique pourquoi `LAS` est optimal, car en servant un flux ayant reçu moins de service, l'ordonnanceur *devine* quel est le flux supposé terminer en premier. La preuve de [119, Theorem 2.1] utilise une contraposée, montrant que toute politique ne suivant pas `LAS` à un moment donné est nécessairement sous-optimal.

Le résultat de `FAIR` se fait par construction, c'est une application directe de [122]. Sous `FAIR`, tous les flux termineront leur transmission plus tôt que ce que cela aurait été pour *tcp*, excepté pour le flux transmettant le dernier segment avant une période d'inactivité, qui lui terminera à la même date. Sous la politique `FAIR`, le flux choisi transmettra à un rythme bien plus élevé que sous *tcp*. En fait, sous cette politique, le flux choisi transmet à tous les instants disponibles pour une transmission jusqu'à ce qu'il satisfasse son besoin. Il est donc trivial d'observer que tous les flux termineront plus tôt que ce qu'ils l'auraient fait avec *tcp*. Une exception est faite pour le dernier flux, lorsque $R^{\texttt{FAIR}}(t) = 1$, ceci implique qu'il y a un seul flux actif et que nécessairement nous ayons $R^{tcp}(t) = 1$. Ce dernier flux terminera sa transmission au même instant sous `FAIR` et sous *tcp*. □

Remarque 4.4 (Sur la nécessité de l'absence de perte et de déséquencement). Ces hypothèses sont nécessaires pour établir les résultats d'optimalité de la proposition 4.3. À titre d'explication, considérons le cas de `SRPT` et que, à un instant donné, il y ait deux flux, l'un tel que $R_1(t) = 1$ et l'autre tel que $R_2(t) = 2$. `SRPT`, ordonnancera le flux 1, mais si le segment est perdu ou déséquencé, le flux 2 pourrait recevoir ses acquittements plus tôt, et par conséquent finir sa transmission également plus tôt. Ceci impliquerait que `SRPT` ne soit optimal pour cette suite d'événements au sens que nous lui donnons

dans la proposition 4.3. Pour `LAS`, nous pouvons construire des contre-exemples similaires. En ce qui concerne `FAIR`, en présence de pertes et de déséquencements, il n'est pas possible de prédire exactement quel est le flux qui est censé finir en premier avec *tcp*. Par conséquent, on ne pourra plus dire que `FAIR` réduit le temps de transmission de tous les flux. Néanmoins, les résultats de la section 4.5 montrent qu'en présence de perte et de déséquencement, le gain de performance de `SRPT`, `LAS` et `FAIR` par rapport à *tcp* est considérable.

4.5 Quelques considérations techniques

Cette section étudie dans quelles mesures les hypothèses nécessaires pour établir les propositions 4.1 et 4.3 sont réalistes. Nous aborderons notamment, les possibilités d'ordonnancement, la capacité des buffers, la taille des segments et l'absence de perte et de déséquencement.

4.5.1 Les possibilités d'ordonnancement du trafic

L'ordonnancement des flux applicatifs tel que nous l'abordons n'est envisageable que lorsque plusieurs mesures doivent être transférées en même temps. Pour une utilisation plus générale à un ensemble de flux, il faut que plusieurs files de transmission ayant une même destination soient simultanément non vides. Afin d'obtenir une idée de la quantité de trafic concerné par ce cas de figure, nous avons analysé 30 minutes de trafic TCP/IPv4 capturé entre Seattle et Chicago en mars 2014 sur le lien optique (0C192) d'un fournisseur de service Tiers 1. [125]. Étant donné qu'il n'est pas possible de connaitre l'état des buffers de transmission des sources TCP sur Internet, nous approximons le nombre de files non vides par le nombre de flux actifs. Un flux est considéré actif entre les temps de capture de son premier segment et de son dernier segment. On dira que deux flux sont concurrents sur une route si leurs périodes d'activités se chevauchent. La table 4.1 résume les résultats obtenus où nous représentons le nombre de routes ayant des flux concurrents, le nombre total de flux en concurrence et le volume correspondant à ces flux. Malgré le fait que seulement 18.1% des flux sont en concurrence, ils représentent 58.2% du

Table 4.1: Part du trafic ordonnançable

	Total	Partagé/Concurrent	Ratio
Routes (nombre)	1788796	102981	5.7%
Flux (nombre)	5155554	937033	18.1%
Flux (volumes in KB)	261769672	152530580	58.2%

trafic en volume de donnée. Ceci illustre que la transmission de fichiers volumineux peut se révéler être un domaine d'application important de notre approche. En particulier, on peut penser au déploiement de Cloud, à la réplication de base de données ou à de l'analyse distribuée comme considérée dans nos travaux, mais aussi dans [115].

4.5.2 La capacité des buffers

Du côté émetteur, bénéficier de *buffers* infinis n'est nécessaire que pour les flux non prioritaires interactifs, et peut s'émuler par un appel bloquant au niveau de l'accès au service correspondant. Ainsi, si une application de faible priorité est capable d'attendre comme dans le cas d'un transfert de fichier, nous pouvons considérer que les *buffers* d'émission sont infinis. Au niveau du récepteur, il est raisonnable de considérer qu'un *buffer* est infini lorsque la fenêtre de congestion est toujours inférieure à la fenêtre du récepteur. Ce cas là arrive naturellement lorsque la congestion se situe au niveau du réseau et non pas au niveau de l'hôte de réception. Par exemple, une approche comme [126], dans laquelle la fenêtre du récepteur est dynamiquement ajustée, pourrait garantir que la fenêtre de réception soit toujours plus grande que la fenêtre de congestion.

4.5.3 L'interface de programmation et option TCP

Afin de conserver une compatibilité avec les applications déjà existantes, il est nécessaire de pouvoir conserver la même façon d'interagir avec la couche de transport. L'interface de programmation (API) de *gtcp* doit donc être compatible avec celle de *tcp*, autrement dit celle des *socket* de Berkeley. Parmi les hypothèses et modifications les plus importantes que *gtcp* implique nous avons évoqué : la connaissance de la taille des flux, l'existence d'un niveau de priorité entre les flux et la possibilité de choisir un ordonnanceur. Afin de prévoir ces fonctionnalités tout en conservant la même API nous proposons d'ajouter simplement de nouvelles options réglables au niveau de chaque *socket*. Ces options sont :

— Le type d'ordonnanceur : `scheduler_type` (par défaut, TCP)
— Le niveau de priorité : `priority` (par défaut, *best effort*)
— La taille des données en attente en dehors du système :`pending_outside`

Lorsqu'un client fait une demande de connexion, le numéro de priorité ainsi que le type d'ordonnanceur est négocié avec le serveur par l'intermédiaire du champ d'options. Comme nous l'avons expliqué précédemment, si le flux est ordonnancé, deux niveaux de numéros de ports et de numéros de séquences devront être gérés, l'un applicatif et l'autre de congestion. Le protocole TCP prévoit jusqu'à 40 octets de champs d'options, ce champ est suffisant pour contenir les numéros de séquences et numéro de ports de

la donnée réellement contenue dans le paquet. Le type d'ordonnanceur lui, est défini au niveau de l'implémentation système pour chaque nœud.

4.5.4 La taille des segments

Dans la remarque 4.2, nous avons expliqué qu'il été possible de gérer plusieurs tailles de segments en contrepartie d'une complexité supplémentaire. En se basant sur les statistiques mises à la disposition du public par CAIDA [127], nous représentons sur la figure 4.4 la distribution de la taille des segments TCP/IPv4, pour les années 2002 et 2014. Nous observons que la très grande majorité des segments ont une taille corres-

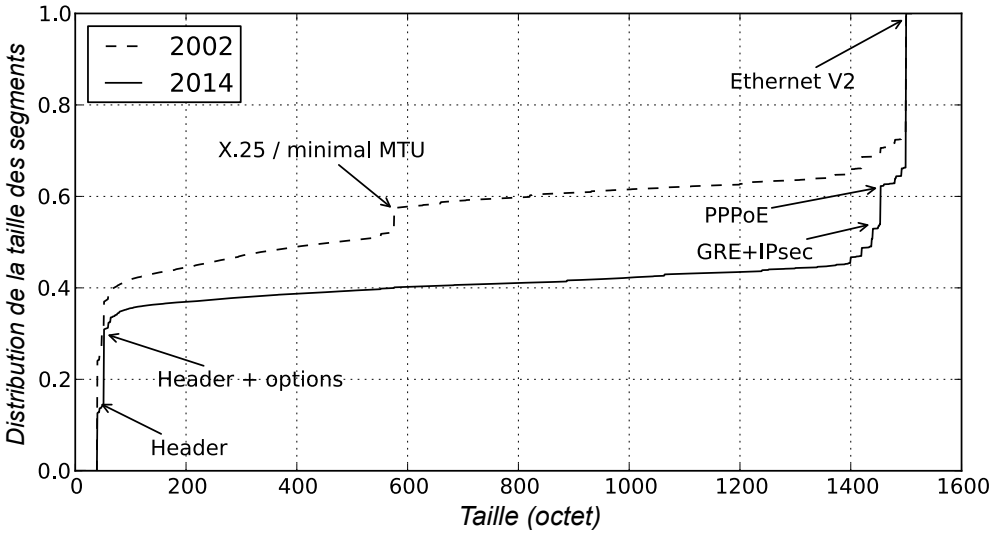


FIGURE 4.4: Distribution des la taille des segments TCP/IPv4 en 2002 et en 2014 issue de tous les points de capture de CAIDA

pondant à une unité de transmission maximale (MTU pour Maximum Transmission Unit) d'une technologie Internet. On estime facilement qu'au moins 60% des segments de données sont des segments de taille maximale.

4.5.5 Absence de pertes et de déséquencement

Ces hypothèses étaient nécessaires à la proposition 4.3 pour que le résultat d'optimalité soit valable. En présence de pertes et de déséquencements, il n'est pas possible de prouver le résultat optimal. Nous nous attendons en revanche à ce que *gtcp* améliore la performance de *tcp*. Nous faisons également remarquer que, dans le cas où un mécanisme de notification de la congestion est employé [107](ECN pour Early Congestion Notification), la preuve d'optimalité tiendrait même en présence de pertes. D'un point de vue d'une perte, le pire scénario est celui de la perte du dernier acquittement pour un flux. Dans un tel cas, un serveur ne détecterait une perte que lorsque l'événement *rto*

surviendrait et que tous les segments envoyés depuis le paquet concerné n'auraient en réalité pas été issus d'un flux prioritaire.

4.6 Implémentation et résultats expérimentaux

Nous présentons une implémentation de *gtcp* accompagnée des résultats expérimentaux pour plusieurs politiques d'ordonnancement.

4.6.1 `SCHED_TCP`, une implémentation de *gtcp*

L'implémentation d'un algorithme de contrôle de congestion peut soit réutiliser un protocole de transport déjà existant, soit être complètement développée. D'un point de vue implémentation, la solution la plus directe afin d'obtenir un contrôle de congestion *gtcp* qui mimique l'algorithme 5 de la section 4.3 est de maintenir autant de connexions actives que l'aurait fait *tcp*, et laisser chacune de ces connexions utiliser le contrôle de congestion de *tcp*. En retour, ceci nous garantit que nous répliquons la suite d'événements de *tcp* et qu'à ce moment-là $V^{gtcp}(t) = V^{tcp}(t)$, $\forall t$. La couche d'ordonnancement de *gtcp* peut-être déployée au-dessus de ces connexions *tcp* comme une couche intermédiaire entre la couche de transport et la couche applicative. Comme énoncé en section 4.2, cette approche suit la tendance actuelle en terme d'architecture protocolaire pour les couches hautes du modèle OSI présenté dans [111].

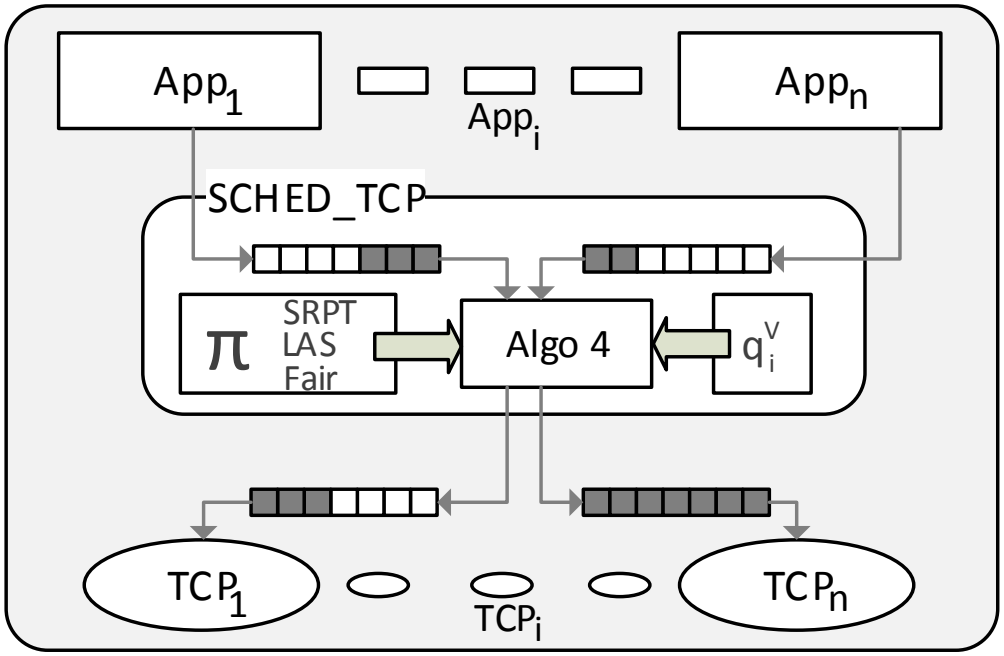


FIGURE 4.5: Architecture de `SCHED_TCP` pour n applications et n connections TCP.

L'implémentation de *gtcp* appelé `SCHED_TCP` et illustrée par la figure 4.5 a été implémentée en C sous Linux 3.2 (également testée sous Linux 2.6) en se basant sur l'interface de programmation (API pour Application Programming Interface) des *sockets* de Berkeley. `SCHED_TCP` multiplexe les flux applicatifs entre les connexions TCP du côté de l'émetteur et les démultiplexe du côté récepteur. Le composant d'ordonnancement π est en charge de choisir le flux prioritaire tandis que le composant principal encapsule et désencapsule les données applicatives avec un entête qui identifie le flux applicatif prioritaire et le transfère à la première connexion TCP disponible. L'algorithme 8, est une solution implémentable inspirée de l'algorithme 5 et 6, où la fonction `schedule` fait référence à la politique d'ordonnancement π devant être implémentée. Lorsqu'une connexion TCP est prête pour accepter de nouvelles données dans son *buffer* d'émission, une vérification de l'état des files virtuelles est faite et la file la plus prioritaire est servie en fonction de la politique π.

Dans l'état qui est représenté en figure 4.5, la connexion TCP_1 a de la place dans son buffer d'émission et devient donc prête à accepter de nouvelles données issues de l'application choisie par π.

Algorithme 8 On_connection_ready(i)

if $q_i^v > 0$ **then**
  $j, s' \Leftarrow$ schedule() // implémente π
  $P_j^{s'} \Leftarrow$ sched_head(j, s') | data()
  enqueue(i,$P_j^{s'}$) // *équivalent à* $send_{i,j}^{s,s'}$
  $q_i^v \Leftarrow q_i^v - mss$ // $q_j \Leftarrow q_j - mss$
end if

Étant donné que `SCHED_TCP` est implémenté dans l'espace utilisateur, il existe des limitations en comparaison avec le comportement théorique de *gtcp*. En particulier, la fiabilité et le contrôle de congestion sont des services fournis par les connexions TCP. Par conséquent, la retransmission, et une partie des *buffers* d'émission ne sont pas partagés par les flux. Ceci implique que le crédit d'émission est géré par TCP et il n'est pas possible de préempter la retransmission d'un segment perdu en transmettant un segment de plus haute priorité. Puisque `SCHED_TCP` sera utilisé sur un vrai réseau, nous nous attendons également à des pertes et/ou des déséquencements de paquets, menant à des retransmissions. Chaque retransmission peut potentiellement retarder l'envoi d'un segment de plus haute priorité. Dans ce cas, il n'est pas possible de reproduire exactement le comportement de la politique d'ordonnancement π, puisqu'au temps t de la transmission, nous ne choisissons pas le segment envoyé sur le réseau, mais plutôt celui qui sera donné à une connexion TCP. L'objectif de cette section est de montrer qu'en dépit des différences qu'il existe entre `SCHED_TCP` et *gtcp*, `SCHED_TCP` obtient un gain de

performance notable vis-à-vis de *tcp*. La fonction `schedule` de l'algorithme 8 implémente *tcp*, ainsi que les politiques `SRPT`, `LAS`, et `FAIR` décrites en section 4.4.3.

4.6.2 Résultats expérimentaux et comparaison de politiques

Dans cette sous-section, nous présentons des mesures faites sur un réseau réel avec `SCHED_TCP` déployé entre deux hôtes, formant ainsi un couple *origine-destination*. La source est située dans un laboratoire de recherche et la destination est située dans un réseau domestique rural. Il est important de noter que sur un vrai réseau, il n'est pas possible de comparer les performances de deux protocoles sous les mêmes conditions de trafic. Dans le but de contrer la non-reproductibilité des expériences, nous répétons chaque expérience 100 fois et en prenons la moyenne.

En figure 4.6, nous traçons le débit moyen des flux pour *tcp* et pour la politique `SRPT`. Nous avons généré trois flux démarrant à des dates prédéfinies et comme expliqué ci-dessus nous avons répété cette expérience 100 fois. Sur la figure supérieure, les trois flux sont gérés par *tcp*. Dans la figure inférieure, `SCHED_TCP` gère les flux *long* et *court* alors que le flux *témoin* est géré par *tcp*. À son arrivée à la seconde 5, `SCHED_TCP` donne la pleine priorité au flux *court* et par conséquent, celui-ci termine sa transmission aux environs de la 11-ème seconde. D'un autre côté, le flux *long* termine sa transmission au bout de 25 secondes dans les deux cas. Le débit obtenu par le flux *témoin* est similaire à tout instant pour les deux cas, ce qui illustre la non-intrusivié de `SCHED_TCP`.

Dans notre implémentation, nous nous situons en couche session. De ce fait, même si le flux le plus court est prioritaire, il est soumis au mécanisme de slow-start de TCP, et lorsque son buffer d'émission est plein il profite du buffer d'émission du flux long. Pour une implémentation en couche transport de notre mécanisme, l'effet du slow-start disparaitrait et nous aurions l'intégralité de la bande passante déjà acquise par le premier flux qui serait allouée aux flux le plus court. Une telle approche résoudrait le problème connu de la sous-allocation des petits flux dans TCP.

Dans le but de comparer la performance des politiques d'ordonnancement de la section 4.4.3, nous considérons maintenant un scénario de trafic plus réaliste expérimenté sur un réseau local (LAN pour Local Area Network) de 100 Mbps. L'arrivée des mesures suit un processus de Poisson de taux λ. Nous avons supposé que la taille des mesures a une distribution de Pareto. Soit S la taille d'une mesure, nous considérons que $\mathbb{P}(S > x) = \frac{1}{(1+cx)^{\alpha}}$, $\forall x \geq 0$. La distribution de Pareto est de type DHR, et a été souvent utilisée pour décrire des distributions de tailles de fichiers sur Internet. Nous choisissons $\alpha = 3$ et $c = 10^{-7}$, ce qui donne une taille moyenne de mesure de $\mathbb{E}(S) = 5$ MB. La capacité du lien est $C = 12$ Mbps et la taille maximale d'un segment est 1448

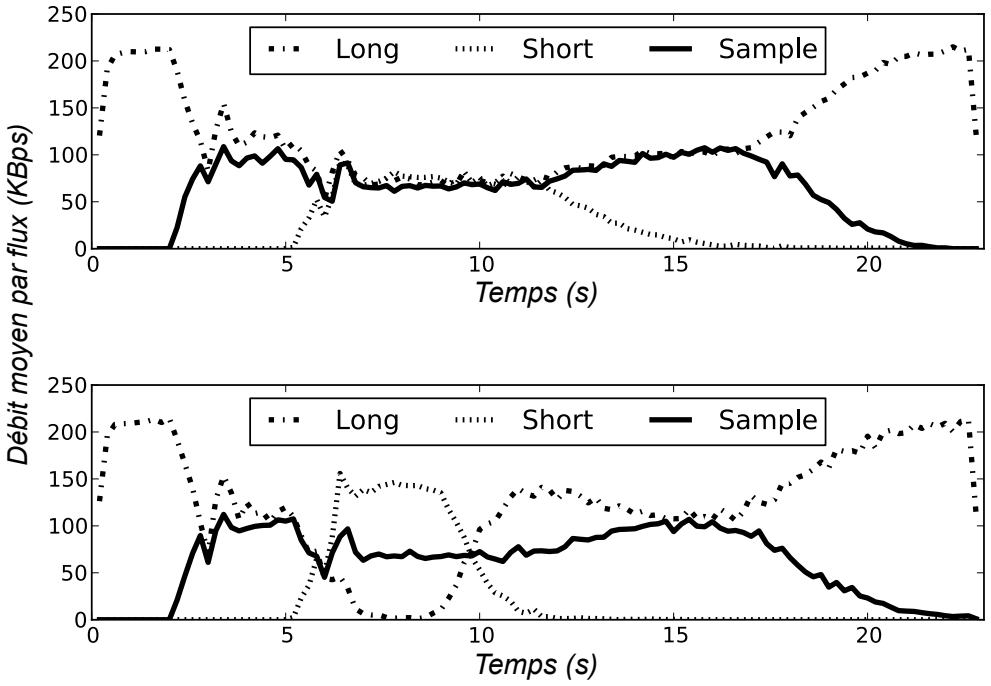


FIGURE 4.6: Débits moyens pour deux flux ordonnancés par *tcp* (figure supérieur) et par `SRPT` (figure inférieure) en concurrence avec un flux *témoin* géré par la politique *tcp*.

B. La valeur de λ est obtenue en fixant la charge du système à 0.9, soit $\frac{\lambda \mathbb{E}(S) \cdot 1448}{C} = 0.9$. Les résultats que nous détaillons sont issus de 100 expériences indépendantes.

La figure 4.7 montre la distribution du nombre de connexions actives. Nous observons que `SRPT` et `FAIR` tendent à minimiser le nombre de connexions actives. Malgré une performance supérieure à celle de *tcp*, `LAS` offre des performances plus en retrait par comparaison avec les deux premières politiques d'ordonnancement. Ceci montre que le fait d'être informé de la taille des mesures offre un avantage conséquent à `SRPT` et `FAIR`.

La figure 4.8 illustre la distribution des temps de transfert par mesure. Comme dans le cas du nombre de connexions actives, les politiques de type (i) comme `SRPT` et `FAIR` offrent un gain de performance supplémentaire. Il est à noter que `SRPT` et `FAIR` sont meilleurs que `LAS` et *tcp* quelle que soit la durée de transfert. Nous remarquons aussi que `LAS` améliore le temps de transfert des petits flux. Cependant, pour les flux durant plus de 6 secondes, `LAS` est moins efficace que *tcp*. Ceci était prévisible puisque `LAS` donne la priorité aux nouveaux flux aux dépens des flux plus anciens.

Dans la figure 4.9, nous avons représenté le temps de transfert pour chaque mesure. Un

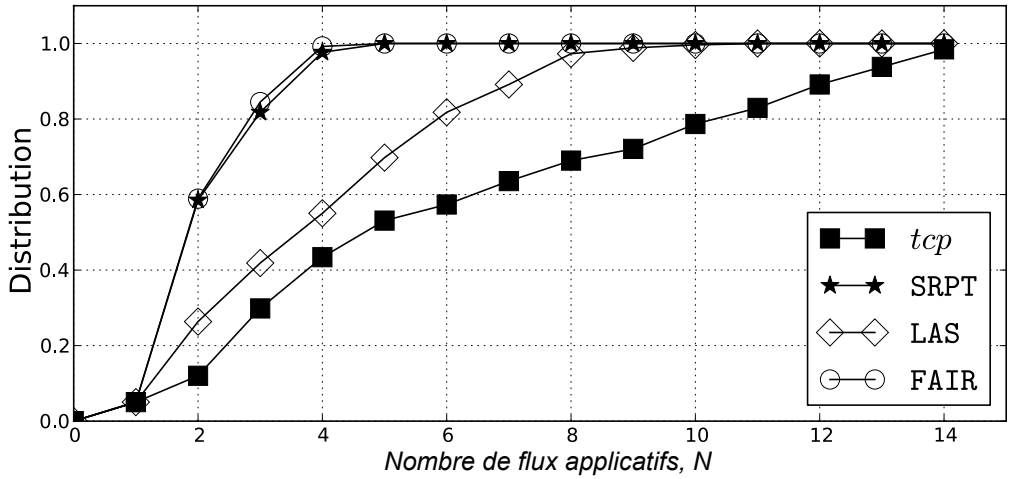


FIGURE 4.7: Distribution du nombre de connexions actives (sans considération de la taille) sous les politiques *tcp*, `SRPT`, `LAS` and `FAIR`.

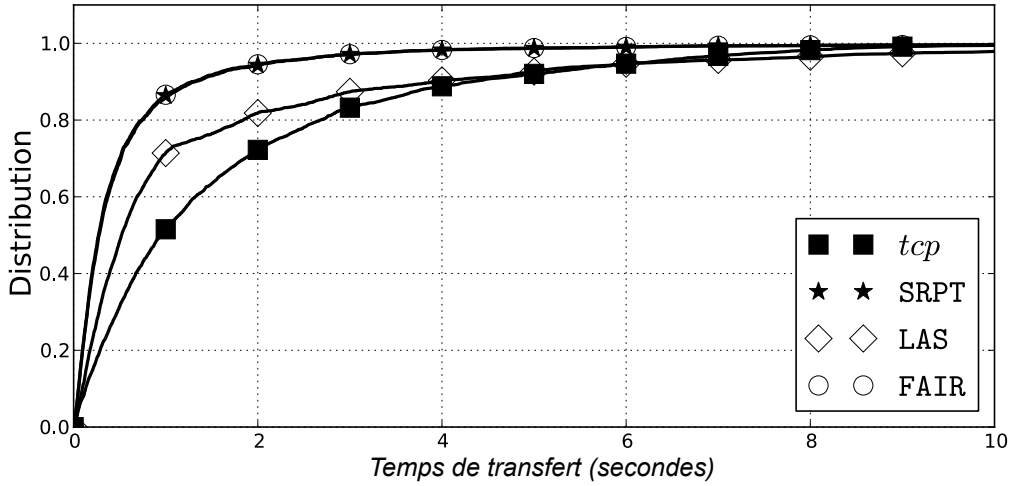


FIGURE 4.8: Distribution des temps de transfert des flux sous différentes politiques d'ordonnancement.

point (x, y) signifie qu'une mesure de taille x a été transféré en un temps y. Nous observons que la volatilité du temps de transfert augmente avec la taille d'une mesure. Sous les politiques `SRPT` et `FAIR`, cette volatilité reste néanmoins plus réduite que pour les politiques `LAS` et *tcp*. Cela signifie que les politiques de type (i) sont non seulement meilleures en terme de temps de transfert, mais elles améliorent également la *prédictibilité* de leur performance, autrement dit, toutes les mesures d'une même taille ont des temps de transfert similaires, indépendamment de leur date d'arrivée.

La table 4.2 résume plusieurs résultats statistiques sur les temps de transfert. En prenant *tcp* comme une référence, nous détaillons le pourcentage des transferts qui terminent en *avance* et en *retard*. Nous pouvons voir qu'avec `SRPT` et `FAIR`, plus de 88% des transferts terminent plus *tôt* qu'avec *tcp* et que seulement 3% des flux terminent plus *tard*. `LAS` réduit le temps de transfert pour environ 60% des flux, alors qu'il l'allonge pour 25%

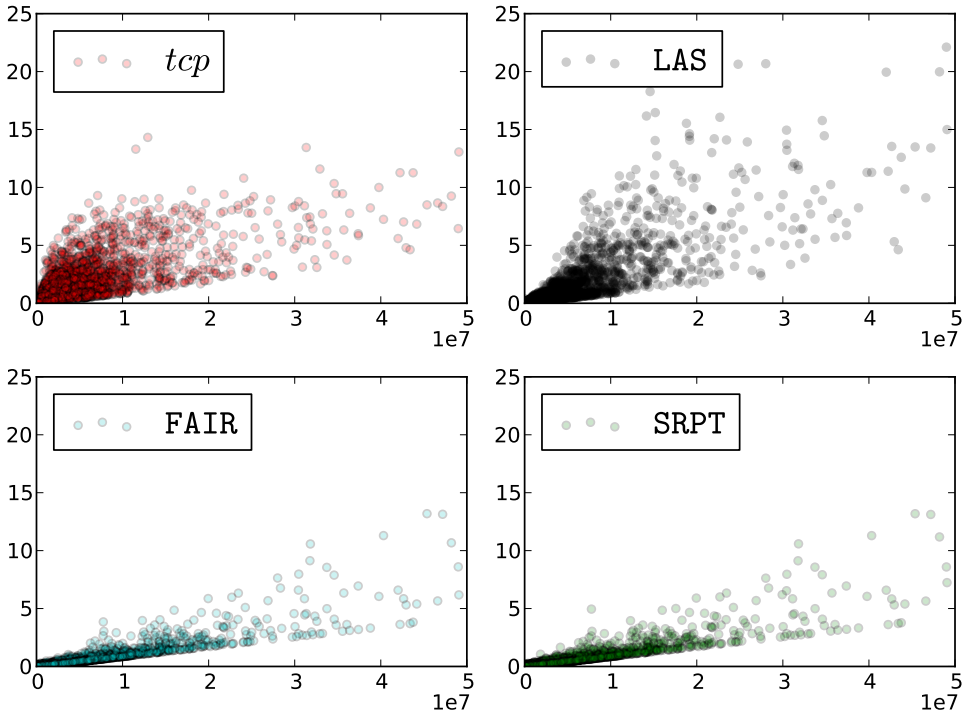


FIGURE 4.9: Temps de transfert en secondes en fonction de la taille d'une mesure en MB. Un point en (x, y) représente une mesure de taille xMB transféré en y seconds.

d'entre eux. Nous donnons également le *gain moyen* et la *dégradation*, qui font référence à la *réduction* (resp. *augmentation*) moyenne du temps de transfert pour les mesures qui terminent plus tôt (resp tard).

La tendance générale est encore une fois en faveur de `SRPT` et `FAIR` qui offrent des gains de performance similaires et que le fait de disposer d'une information sur la taille de la mesure est un avantage considérable.

TABLE 4.2: Statistique sur le temps de transfert pour différentes politiques d'ordonnancement.

	Avance	*Retard*	*Identique*	$\overline{Gain}$	$\overline{Perte}$
`LAS`	61%	25.6%	13.2%	-0.57s	+314ms
`SRPT`	88.8%	3.2%	8%	-1.11s	+28ms
`FAIR`	89.6%	2.4%	7.9%	-1.13s	+33ms

4.7 Conclusions et travaux futurs

L'objet principal de ce chapitre est de montrer comment il est possible de réduire le temps d'acquisition d'une opération de mesure distribuée en travaillant au niveau du transfert de données entre 2 agents. Pour ce faire, nous avons concentré nos travaux sur

le protocole TCP pour lequel nous avons cherché à ordonnancer les flux partageant une même route. Nous avons montré que l'*ordonnancement* et le *contrôle de congestion* de TCP sont des composantes qui peuvent être découplées. Nous avons ainsi introduit *gtcp*, un nouveau protocole qui permet d'implémenter toute politique d'ordonnancement entre les flux partageant une même route, tout en préservant l'impact général que ceux-ci ont sur le réseau.

L'ordonnanceur de *gtcp* permet de donner une priorité arbitraire entre les flux. En ce qui concerne les flux de mesures distribués, les mesures connues pour être assortatives peuvent devenir plus prioritaires que les autres. Une priorité inférieure (*less than best effort*) peut s'appliquer aux flux de mesures actives estimant délai et bande passante afin de réduire leur intruisivité. Enfin, la priorité peut être donnée à certains flux de contrôle et de gestion au sein du réseau pour une meilleure réactivité.

Afin de dériver des résultats généraux, nous nous sommes intéressés aux politiques d'ordonnancement se basant sur la taille des mesures. Ce choix nous a permis de dériver des résultats d'optimalité pour *gtcp* notamment au niveau de la réduction de la latence moyenne. Nos expérimentations avec `SCHED_TCP` sur un réseau réel ont montré que le gain de performance global est notable, et ceci, sans aucune dégradation pour les autres flux. `SCHED_TCP` pourrait être déployé de manière incrémentale, améliorant la performance de flux sur certaines routes, sans influencer la performance obtenue sur une autre route du réseau.

Ce travail nous amène à réfléchir à la notion d'équité. Dans le cas de TCP, l'équité est gérée par flux. Cela signifie qu'une application utilisant plusieurs flux TCP acquiert plus de bande passante qu'une application partageant son goulot d'étranglement (si celui-ci est en cœur de réseau). Il en est de même au niveau d'un utilisateur ou d'une station. Face à l'impossibilité de changer radicalement le modèle OSI, nous avons choisi de conserver la façon d'allouer la bande passante entre les stations. Néanmoins, en ajoutant une couche de contrôle de congestion entre les couches réseaux et transport du modèle OSI la notion d'équité serait appliquée aux stations et chaque station pourrait choisir comment utiliser la ressource qui lui est allouée sur une route. Une priorité pourrait être donnée à un certain type de trafic comme un trafic interactif, ou temps réel face à un autre. De manière plus générale, l'utilisateur final pourrait définir son ordonnanceur π, visant à maximiser la définition de sa propre performance, telle que par exemple nous la définirons dans le chapitre suivant.

Après avoir étudié la distribution et le transfert d'une mesure, nous nous intéresserons dans le chapitre suivant à ce que celle-ci peut représenter.

Chapitre 5

Évaluation et analyse distribuée de la performance

5.1 Motivations

Dans les chapitres précédents, nous avons étudié l'amélioration de la diffusion et du transfert d'une information de mesure dans un réseau. Nous avons défini différents services, à savoir `distribution` et `transport`. Nous avons également défini une classe `Mesure` représentant la donnée échangée. Dans ce chapitre nous nous intéressons à ce que cette classe peut représenter, notamment en terme de *performance* et de *contexte* afin que les nœuds d'un réseau puissent être capable d'évaluer la performance de leur système et de comprendre les facteurs environnementaux qui l'impactent.

En fonction du point de vue depuis lequel on l'observe, un réseau peut s'avérer plus ou moins performant. Par exemple du point de vue d'un utilisateur, le réseau est performant si son débit est suffisant. Du point de vue d'un opérateur, la performance est synonyme de rentabilité économique. Un premier observateur externe à ce réseau pourra le juger performant pour son faible impact environnemental tandis qu'un second l'évaluera sous l'angle de l'équité dans le partage des ressources. Afin d'évaluer la performance des réseaux, différentes approches ont donc été utilisées, nous pouvons donner les exemples suivants. Dans [128] les auteurs proposent d'évaluer la performance d'un réseau de commutation de paquets par classes d'utilisateurs. Pour chaque classe, différents critères sont observés comme le taux de rejet entre deux usagers, le temps moyen de transport d'un message ainsi que sa variance maximale. Dans [129] la performance d'un réseau de capteurs sans fil est évalué par le débit moyen des données remontées à la passerelle, le nombre moyen de time-slots pour transmettre un message et la consommation énergétique par time-slot. Enfin, nous avons nous-même utilisé dans le chapitre précédent le délai de transfert moyen d'un flux ainsi que le nombre de transfert concurrent afin d'évaluer la performance des algorithmes d'ordonnancement de TCP.

Au final, concepteurs et évaluateurs définissent divers objectifs ou notions d'équité tout en utilisant un vocabulaire et des outils différents. De plus, en rassemblant divers systèmes au sein d'un même réseau, ces notions de performance deviennent d'autant plus complexes. Comment faire pour composer avec des applications (ou des utilisateurs) lorsque leurs intérêts divergent et cherchent à optimiser des critères différents ? Comment évaluer si un système est performant ou non et analyser les facteurs environnementaux qui impactent cette performance ? Comment intégrer cette évaluation et cette analyse au sein du réseau lui-même. C'est à ces questions que nous tentons de répondre dans ce chapitre.

Pour ce faire nous proposons de modéliser un réseau comme un système multi-agents pour lequel nous définissons la notion de *performance* et de *contexte*. Nous définirons la *performance* d'un système comme la distribution des utilités de ses agents, et le

contexte comme la distribution de leurs variables d'état. Dans un second-temps nous nous intéressons à la façon dont les nœuds d'un réseau peuvent mesurer la *performance* de leur système, et leur *contexte*. Nous utiliserons pour cela les outils de propagation d'information décrits dans les chapitres précédents pour superviser une distribution à temps variable. Pour cela, la classe `Mesure` sera instanciée par un dictionnaire, pouvant ainsi représenter un espace vectoriel quelconque.

Le reste du chapitre est organisé de la sorte. Dans la prochaine section nous présentons notre modèle pour l'évaluation de la performance. Après avoir expliqué l'importance des distributions dans ce modèle, nous montrons dans la section 5.3 comment les nœuds du réseau peuvent superviser une telle distribution à temps variable lorsque la topologie est dynamique. Enfin nous présentons dans la section 5.3 deux cas d'études, le premier illustre la mesure distribuée de la *performance* et du *contexte* dans un réseau dynamique sans fil tandis que le second explique comment tirer partie de ces informations pour analyser l'évolution de la performance.

5.2 Modélisation et évaluation des systèmes distribués

Dans cette section, nous proposons de modéliser un réseau comme un système multi-agent (SMA), pour lequel nous définissons les notions de performance et de contexte. Ce modèle a pour but de donner un vocabulaire commun aux concepteurs de systèmes comme à ceux qui les évaluent. En définissant dès la conception, les objectifs de performance d'un système, l'intégration des mécanismes d'auto-évaluation et d'auto-gestion au sein de celui-ci sera facilitée.

A la base du modèle de système multi-agents élaboré dans cette section, se trouve la notion fondamentale de *variable*. La variable sera l'élément constitutif des états, des événements et des agents de notre modèle. Une variable (ou attribut) x est un élément de description qui est un élément du domaine $\mathcal{X}$. Nous utiliserons de manière interchangeable, *domaine*, *type* ou plus généralement *classe*. Les variables, leurs domaines et leurs représentations peuvent être variés et complexes. Nous ne posons pas de limite sur leur domaines. Celles-ci peuvent aller d'un simple entier, à une collection d'agents en passant par une fonction ou un *tuple*. Un vecteur ou *tuple* $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ est une collection ordonnée de n variables aussi appelées attributs avec potentiellement $n = \infty$. Un *tuple* $\mathbf{x}$ est naturellement associé à un domaine $\mathcal{X}$ qui est le produit cartésien du domaine de ses attributs, soit $\mathcal{X} = \prod_{i=1}^{n} \mathcal{X}_i$. Pour des raisons d'intelligibilité, lorsque les éléments $x_1, x_2, \ldots$ d'un *tuple* seront aussi des *tuple*s, nous les considérerons comme des variables simples sans entrer dans le détail de leurs attributs.

5.2.1 Agent et utilité

Les agents sont les entités principales d'un système. Ils interagissent entre eux, observent, analysent et prennent des décisions. Un agent a est défini par un quadruplet $(\mathcal{X}_a, \mathcal{E}_a, u_a, \pi_a)$ dont les éléments sont respectivement : l'espace d'état, l'interface, l'utilité et la politique d'évaluation de la performance.

Espace d'état : Un agent peut être appréhendé comme une machine à états qui évolue au sein d'un environnement ou monde extérieur. L'état $\mathbf{x}_a \in \mathcal{X}_a$ d'un agent est un *tuple* de variables d'état décrivant chacun de ses aspects.

Interface : Nous supposons l'existence d'un domaine universel $\mathcal{E}$ d'événements. Un événement $e \in \mathcal{E}$ est un *tuple* $(src, id, champs)$ dont les éléments sont :

— src : l'agent qui en est le responsable
— id : l'identifiant de l'événement qui définit le *domaine* du champs
— $champs$: un *tuple* définissant les variables de l'événement

Un événement est équivalent à une trame (ou suite de champs) dont le premier élément est la source et le second est le type. En terme d'implémentation, un événement correspond à l'appel d'une méthode d'un objet ou l'envoi d'un message de sa part. L'interface $\mathcal{E}_a$ d'un agent a constitue l'ensemble de ses méthodes et de ses messages. Elle est définie par $\mathcal{E}_a = \{(src, type, champs) \in \mathcal{E} | src = a\}$.

Utilité : La notion d'utilité ou encore de fonction objectif, est communément utilisée en théorie des jeux, dans les systèmes décisionnels ou lorsque qu'une recherche d'optimalité est nécessaire. Nous définissons l'utilité $u_a : \mathcal{X}_a \mapsto [0, 1]$ d'un agent a comme une fonction évaluant la valeur de son état, autrement dit, la satisfaction d'un agent d'être dans un état donné. Lorsque $u_a(\mathbf{x}) = 1$ (resp. 0), un agent est pleinement (resp. nullement) satisfait de sa situation, de son état. L'utilité peut se voir comme le ratio de satisfaction d'un agent.

Performance : La notion de performance chez un agent est ce qui le conduit à agir au sein de son environnement. Face à deux actions possibles, un agent évalue les issues de ces actions et choisit celle qu'il préfère. Seules les issues en terme d'utilité comptent dans ce choix, à savoir :

— La valeur des utilités pouvant survenir à l'issue de ce choix.
— Les probabilités d'occurrence de ces valeurs.

Ceci se résume par une *performance* $p \in \mathcal{P}$ qui est une distribution de probabilité sur l'espace probabilisable $([0,1], \wp([0,1]))$ [1]. Une telle distribution peut être utilisée pour décrire la valeur d'une utilité en présence de risque (comme celle de l'état futur d'un agent) ou encore l'utilité d'une population d'agents de même importance. La meilleure (resp. pire) des performances est noté δ_1 (resp. δ_0), elle représente le cas dans lequel la probabilité qu'un agent soit pleinement (resp. nullement) satisfait vaut 1.

Politique d'évaluation de la performance : Une politique d'évaluation de la performance est une fonction π telle que $\pi : \mathcal{P} \mapsto [0,1]$. Cette fonction évalue une performance en la notant sur l'intervalle $[0,1]$. L'évaluation d'une performance est assimilable à celle d'une distance. En effet, évaluer une performance revient à mesurer sa distance avec la meilleure (resp. pire) des performances : δ_1, (resp. δ_0). Un moyen simple est de se baser sur l'espérance. C'est ce qui est fait par un agent rationnel au sens de Von-Neumann et Morgensten [130], qui définit l'évaluation d'une performance comme l'utilité moyenne. Cependant une telle politique est connue pour ne pas modéliser correctement le comportement utilisateur [131]. En effet, il est parfois naturel pour un utilisateur, de se baser sur l'utilité minimale garantie plutôt que sur l'utilité moyenne. Une politique d'évaluation de la performance définit la sensibilité au risque d'un agent. Il est donc possible d'utiliser d'autres politiques basées par exemple sur une divergence statistique (f-divergence, entropie de Rényi ou une distance énergétique), ou sur un quantile de la distribution. Un agent possède donc sa propre politique d'évaluation de performance π_a qui lui permettra de juger l'utilité d'un état futur ou celle d'une population d'agents.

5.2.2 Système multi-agents et performance

Un système multi-agents (SMA) est relatif à un *tuple* $\mathbf{a} = (a_1, a_2, \ldots, a_m)$ d'agents. C'est un agent dont l'état $\mathbf{x_a}$ est le vecteur $\mathbf{x_a} = (\mathbf{x}_{a_1}, \mathbf{x}_{a_2}, \ldots, \mathbf{x}_{a_n})$ des états de ces agents et pour un SMA donné, il est possible de définir un vecteur des utilités $(u_{a_1}(\mathbf{x}_{a_1}), u_{a_2}(\mathbf{x}_{a_2}), \ldots, u_{a_n}(\mathbf{x}_{a_n}))$.

L'interface et l'utilité d'un SMA dérivent également de celles de ses agents. Pour un SMA composé des agents $(a_1, a_2, \ldots, a_n)$ l'interface et l'utilité sont telles que :

- $\mathcal{E}_{\mathbf{a}} = \bigcup_{i=1}^{n} \mathcal{E}_{a_i}$
- $\exists f : [0,1]^n \mapsto [0,1]$ telle que $u_{\mathbf{a}}(\mathbf{x_a}) = f(u_{a_1}(\mathbf{x}_{a_1}), u_{a_2}(\mathbf{x}_{a_2}), \ldots, u_{a_n}(\mathbf{x}_{a_n}))$

Plus simplement l'interface d'un SMA est l'union des interfaces de ses agents tandis que son utilité est une fonction du vecteur des utilités de ses agents.

1. Nous notons $\wp(.)$ l'opérateur ensemble des parties

Évaluer l'état d'un système, revient donc à évaluer le vecteur des utilités de ses agents. Cette fois-ci, c'est la fonction utilité qui est assimilable à une fonction distance. Évaluer l'état d'un système revient à mesurer la distance du vecteur des utilités de ses agents avec le meilleur (resp. pire) vecteur d'utilité dont toutes les composantes valent 1 (resp. 0).

Nous nous intéressons aux politiques qui ne sont pas sensibles à l'échelle (nombre d'agents dans le système) et qui considèrent que tous les agents ont une importance égale. Dans un tel cas, évaluer l'état d'un système revient à évaluer la distribution des utilités de ses agents [2]. Dès lors, $u_{\mathbf{a}}$ devient une politique d'évaluation de la performance. Comme nous l'avions déjà énoncé auparavant, une performance peut être utilisée pour décrire la valeur de l'état d'un agent en présence de risque mais aussi l'utilité d'une population d'agents de même importance, ce qui est le cas ici.

Exemple 5.1 (Évaluation d'un algorithme d'allocation de ressources)**.** Afin d'illustrer les éléments que nous venons de définir, nous considérons le modèle simplifié d'un SMA de n agents soumis à l'allocation d'une ressource réseau. Ce problème d'allocation de ressource est celui rencontré lors d'un accès au médium dans les réseaux sans fil mais aussi dans le partage d'une route par plusieurs flux tel que nous l'avons étudié dans le chapitre précédent.

Nous caractérisons simplement l'état d'un agent par un vecteur de deux éléments (b, r), symbolisant respectivement son besoin et sa ressource. En implémentant un algorithme d'allocation de ressource distribué chaque agent reçoit une certaine quantité de ressource r. Dans le cas d'un contrôle d'accès au médium, la ressource correspond par exemple à un temps de parole des stations. Dans le cas d'un contrôle de congestion distribué, la ressource correspond à une bande passante. Pour l'expérience, tous les agents ont la même utilité définie par $u((b, r)) = \min(1, \frac{r}{b})$ que nous représentons dans la figure 5.1.

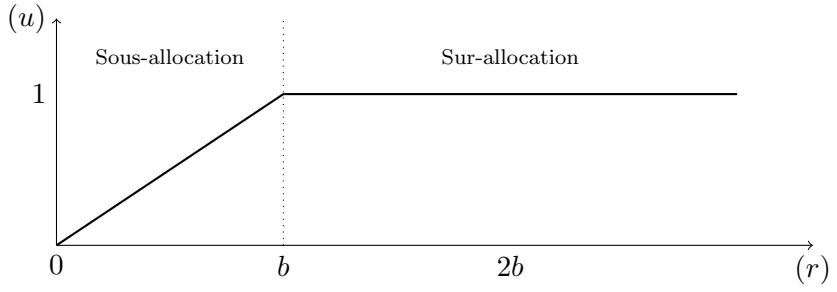


FIGURE 5.1: Utilité d'un agent en fonction de sa ressource

Nous étudions 4 algorithmes d'allocation de ressources parmi les plus connus qui sont :

2. Il est possible de généraliser pour k classes d'agent, en utilisant k performances

— Parts égales (PE) : chaque agent reçoit la même quantité de ressource (cas d'un algorithme TDMA classique ou de TCP)
— Proportionnel au besoin (PB) : chaque agent reçoit une quantité de ressource proportionnel à son besoin.
— Max-min équitable (Mm) : initialement, chaque agent reçoit la même quantité de ressource et les surplus sont successivement transférés vers l'agent ayant le plus petit besoin non-satisfait.
— Maximum d'efficacité (ME) : Les agents sont servis pleinement par besoin croissant.

Chaque algorithme va mener à une performance (distribution des utilités) différente, le rôle d'une politique d'évaluation est de déterminer la meilleure loterie, ce qui correspond dans ce modèle-jouet, à décider du meilleur algorithme. Nous menons l'expérience pour 100 agents lorsque la taille de leur besoin suit une loi exponentielle et que la ressource à allouer ne peut couvrir qu'une fraction $\alpha = 0.5$ du besoin total des agents. Le graphe ci-dessous décrit la fonction de répartition des utilités au sein du système pour chaque algorithme. Nous spécifions également dans le tableau trois caractéristiques des performances qui sont la moyenne, la variance ($\sigma^2(.)$) et l'indice d'équité de Jain ($\mathcal{J}(.)$)[3], et qui peuvent servir à évaluer la performance des algorithmes. Il est à noter que ces résultats ne dépendent pas du paramètre λ de la loi exponentielle (le détail du calcul est donné en annexe).

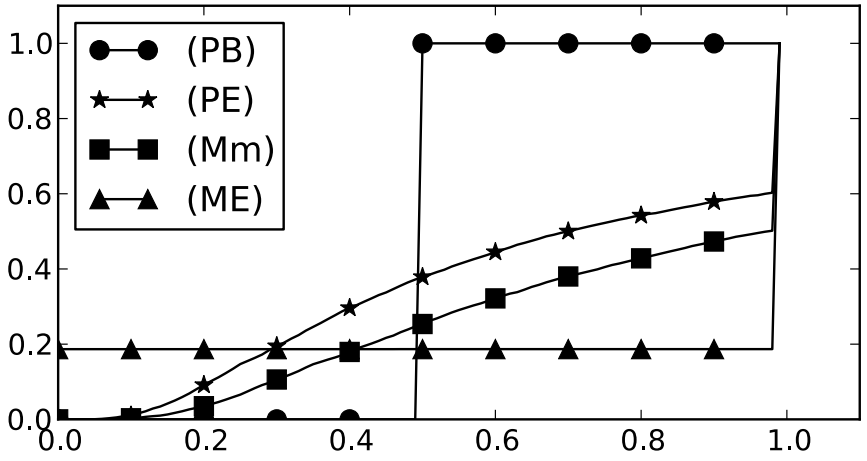


FIGURE 5.2: Performance ou fonction de répartition des utilités pour différents algorithmes d'allocation de ressource

Ici évaluer la performance, c'est choisir un compromis entre la satisfaction moyenne des agents et l'équité. Il est clair dans l'expérience ci-dessus, que l'algorithme (ME) est le plus performant en terme de satisfaction moyenne mais il est parmi les moins équitables. Bien

3. $\mathcal{J}(\mathbf{u}) = \frac{(\sum u_i)^2}{n.\sum u_i^2}$

Algorithme	$\bar{\mathbf{u}}$	$\sigma^2(\mathbf{u})$	$\mathcal{J}(\mathbf{u})$
(PB)	0.5	0	1
(PE)	0.67	0.10	0.81
(Mm)	0.76	0.08	0.87
(ME)	0.81	$(0.19, 1, 1)$	0.81

TABLE 5.1: Quelques indicateurs de performance

entendu, il est logique de préférer l'algorithme (Mm) à l'algorithme (PE) car sa fonction de répartition lui est toujours inférieure[4]. Enfin, l'algorithme (PB) est parfaitement équitable (au sens des utilités) mais se trouve être le moins performant en terme d'utilité moyenne. Cette notion d'équité au sens des utilités nous mène à considérer l'utilité comme une ressource *virtuelle* partageable. Si l'algorithme (PE) *gaspille* en quelque sorte des ressources réseau en donnant trop de ressources à certains agents qui ne les utilisent pas (comme c'est le cas pour une allocation de type TDMA), l'algorithme (PB) *gaspille* lui de l'utilité en donnant de la ressource à celui qui la valorise le moins (celui qui a le plus gros besoin).

5.2.3 Observations et contexte

Dans le souci de pouvoir analyser la performance d'un système, nous proposons de suivre son évolution et de la confronter avec celle de son *contexte*. Dans le langage courant, le contexte comprend à la fois un voisinage temporel (états et événements passés, présents ou futurs, etc.) et un voisinage spatial (états et événements des agents voisins). Nous proposons donc d'agréger les états et les événements des agents afin de définir leur *contexte*.

Observations et trace du système : Pour un SMA donné, une observation est un triplet $(temps, agent, evenement)$, spécifiant qu'un agent a observé un événement à un instant donné. Une telle information est atomique dans le sens où sa signification ne dépend pas d'un autre contenu. Elle peut être vue comme une trame échangeable sur un réseau dans un but de collaboration entre agents. En terme d'implémentation, une observation correspond à une fonction de rappel (callback). La trace d'un système correspond à l'ensemble des observations classées par ordre chronologique. On parlera également de flux d'observation.

Remarque 5.1. Le format de trame d'une observation permet de nombreuses flexibilités telles que l'encapsulation d'un événement comme :

— La mise à jour d'une variable (métrique classique, utilité, etc.)

4. Dominance statistique d'ordre 1

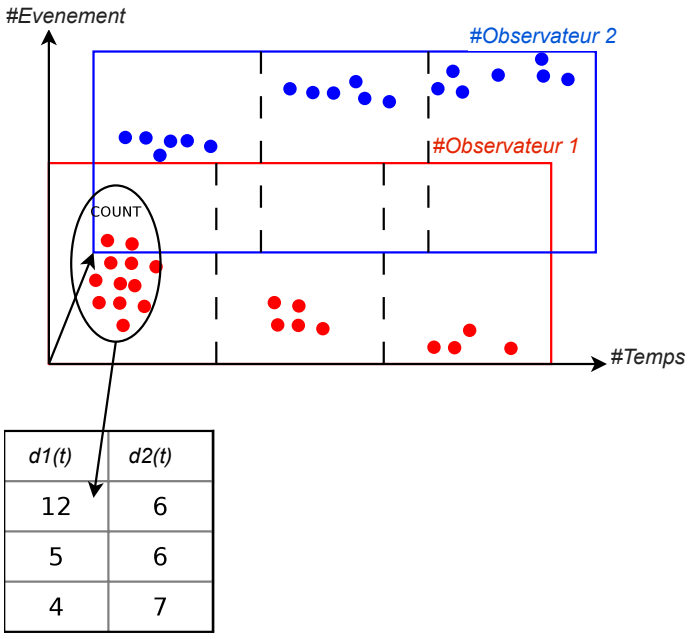


FIGURE 5.3: Exemple de série temporelle de descripteurs

— Une autre observation.
— Le calcul d'un *descripteur* (voir ci-dessous)

Groupement des observations par propriétés et extraction de flux : Afin de relier l'évolution de la performance avec les observations faites sur le système nous proposons la méthode d'agrégation suivante :

1. Partitionner les observations par ressemblance : pour ce faire, la similarité des dates, des agents ou des événements peut être exploitée par un algorithme de partitionnement.
2. Appliquer à chaque partition une fonction d'agrégation pour calculer un *descripteur* : des fonctions telles que `min`, `max`, `average`, `count` seront considérées.

Pour notre étude, nous avons groupé les observations par observateur et par type d'événement. Puis nous avons construit des sous-groupes par intervalle de temps. C'est sur ces sous-groupes que nous avons appliqués des fonctions d'agrégation (telles que le cardinal ou la moyenne sur le champ d'un événement etc...). Nous pouvons ainsi créer des séries temporelles de *descripteur*. Un exemple parlant de *descripteur* est le nombre d'événements observés par un agent et par unité de temps, que nous avons illustré dans la figure 5.3. Ce traitement d'observation peut donc être effectué aussi bien en ligne que hors ligne par les agents. Nous verrons dans la section 5.4 comment exploiter ces *descripteurs* pour analyser l'évolution de la performance. Par construction, un *descripteur* peut donc faire partie de l'état d'un agent.

Exploitation des variables d'état et définition du contexte : Un descripteur agrège les événements dans le temps et permet en quelque sorte d'intégrer l'histoire d'un agent. Nous nous intéressons à l'agrégation de ces informations au sein d'un réseau. Dans le modèle de système multi-agents que nous avons proposé, un état est un *tuple* de variables. Chaque variable est un élément descriptif de son état. À titre d'exemple, celles-ci peuvent être :

— Une niveau d'énergie
— Une valeur de capteur
— Un compteur d'événement ou un autre *descripteur*
— La position géographique
— Un niveau de congestion
— Une liste de label (ou *tags*)
— ...

Nous proposons de définir le contexte comme la distribution de ces valeurs au sein du réseau. Ceci peut donc correspondre à la densité géographique des nœuds dans un réseau sans-fil, comme à la distribution du niveau de congestion dans un voisinage réseau ou encore à la distribution d'un descripteur intégrant ainsi le temps et le voisinage réseau. Une autre approche intéressante pour la représentation du contexte est celle du *sac de mots* (aussi appelé sémantique vectorielle). Cette représentation est couramment exploitée par les algorithmes de classification de documents. Chaque agent peut maintenir à jour un ensemble de labels (`goodSNR`, `lowEnergy`, `highlyMobile`, `multiCore`, `IPv6`, `videoStream`, `rfidTag` ...) ou méta-données munis d'un poids (initialement unitaire). Chacun de ces labels peut décrire un élément de contexte comme par exemple : la configuration matérielle, la version d'une librairie, les protocoles utilisés, le besoin applicatif ou encore l'ensemble des objets environnants. La distribution des labels dans un voisinage réseau, peut, au même titre que dans un document, décrire la situation actuelle de ce réseau, ce qui constitue autant d'informations exploitables à des fins d'autonomie de la gestion.

Dans cette section, nous avons proposé un modèle d'évaluation et d'analyse de la performance des systèmes multi-agents. Dans ce modèle la *performance* d'un système ainsi que la représentation de son contexte repose essentiellement sur les notions de vecteurs et de distribution multivariés. Afin de pouvoir rendre un réseaux autonome, il est nécessaire que ses agents soient à même d'évaluer la performance de leur système et de la confronter au contexte afin de comprendre ce qui l'impacte. Nous proposons d'exploiter les services de `diffusion` et de `transport` développés dans les chapitres précédents pour mesurer dans les réseaux dynamiques ces deux informations. Ceci passera donc par l'évaluation d'une densité de probabilité à temps variable.

5.3 Évaluation distribuée de la performance dans les réseaux dynamiques

Dans cette section nous étudions comment une distribution peut-être évaluée de manière collaborative entre les agents, notamment en se basant sur les outils de mesure que nous avons développés dans les chapitres précédents. La mesure d'une telle distribution sera exploitable pour l'évaluation de la performance d'un système ainsi que celle de son contexte.

5.3.1 Estimation paramétrique et non paramétrique de densité de probabilité

Lorsque l'on mesure une métrique dans un réseau, en connaître la moyenne peut s'avérer utile. Néanmoins il est souvent dit que la valeur moyenne d'une population n'est pas très parlante sans son écart type. En fait, ces paramètres sont seulement utiles si l'on a une connaissance a priori sur le phénomène observé. Par exemple, le couple moyenne et écart-type caractérise pleinement une loi normale, mais si la distribution est méconnue et s'avère en réalité être multi-modale, ces paramètres peuvent être mal interprétés. Par ailleurs, si l'on veut manipuler des données qualitatives (lorsque la métrique relevée n'a rien de numérique), la moyenne n'est pas calculable et d'autres approches comme celle du mode doivent être utilisées.

Estimation par la méthode des noyaux : Les techniques d'estimation non-paramétriques peuvent surmonter ce problème. Nous avons souvent tendance à les associer aux diagrammes en bâtons (données qualitatives) ou aux histogrammes (données quantitatives et continues). En échange d'une complexité de calcul supplémentaire, l'estimation par noyau (KDE pour Kernel Density Estimation) englobent ces méthodes en estimant une densité de probabilité (*pdf* pour probability density function) sans avoir de connaissance préalable sur celle-ci.

Dans le cas de mesures dans $\mathbb{R}^d$, pour un ensemble de P points de $\mathbb{R}^d$ issus d'une *pdf* méconnue f, l'estimation par noyau $\hat{f}$ de f peut être définie par

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{P} \alpha_k K'_k(\mathbf{x}), \tag{5.1}$$

où α_k et K'_k sont respectivement le poids et la fonction noyau mis à l'échelle, associés à l'échantillon $\mathbf{p}_k$. La fonction noyau mis à l'échelle K'_k est définie par

$$K'_k(\mathbf{x}) = |\mathbf{H}_k|^{\frac{-1}{2}} K_k(\mathbf{H}_k^{\frac{-1}{2}}(\mathbf{x} - \mathbf{p}_k)). \tag{5.2}$$

Cette fonction K'_k peut être vue comme la contribution de l'échantillon $\mathbf{p}_k$ à l'estimation de la valeur de f au point $\mathbf{x}$. Cette contribution est définie par la bande-passante $\mathbf{H}_k$ qui est une matrice carrée de dimension d et par la fonction noyau K_k qui est symétrique et dont l'intégrale sur son ensemble de définition vaut 1 [132]. Dans bien des cas, $\mathbf{H}_k$ et K_k sont les mêmes quelque soit k. De plus $\sum_{k=1}^{P} \alpha_k = 1$ et en général nous avons $\alpha_k = \frac{1}{P}$. Il est important de noter que le choix de la bande passante ainsi que celui de la fonction noyau est crucial sur la qualité des estimations. Néanmoins, ces aspects dépassent le cadre de nos travaux. Dans un souci de simplicité, nous considérons un noyau Gaussien $G(\mathbf{p}) = (2\pi)^{-d/2} e^{\frac{-\mathbf{p}^T\mathbf{p}}{2}}$, et chaque nœud estime localement la bande passante optimale en se basant sur l'algorithme donné par Scott dans [133]. En pratique ce choix est ouvert. Pour des données qualitatives ou catégoriques, les points échantillons appartiennent non pas à $\mathbb{R}^d$, mais à un ensemble fini et discret de labels ou catégories. Dans ce cas particulier, nous pouvons utiliser le noyau suivant :

$$K'_k(\mathbf{x}) = K_k(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{p}_k = \mathbf{x}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3}$$

Algorithme décentralisé d'une distribution : Comme dans le cas centralisé, nous pouvons aborder le problème de l'estimation décentralisée d'une distribution de façons paramétrique ou non-paramétrique.

Dans le premier cas, des hypothèses sont faites sur la distribution étudiée qui peut alors être déterminée par un ensemble de paramètres. L'objectif est de calculer les valeurs de paramètres qui optimisent un critère donné, comme le maximum de vraisemblance. Des algorithmes spécifiques pour calculer de manière distribuée un modèle de mélange de Gaussienne (GMM pour Gaussian Mixture Model) sont donnés par Nowak dans [134] et par Jiang et collab. dans [135]. D'autres solutions proposées par Kowalczyk, Vlassis and Sfakianakis utilisent un protocole par propagation de rumeurs [136, 137]. En ce qui concerne les estimations non-paramétriques, une estimation distribuée par la méthode des noyaux est donnée par Hu et collab. dans [138]. Ces derniers travaux proposent également une méthode de compression (avec perte d'information) pour considérer les nœuds dont les ressources sont limitées. Tous ces travaux ont prouvé leur efficacité, néanmoins ils sont soit spécifiques au problème considéré, soit difficiles à mettre en

œuvre ou bien émettent des hypothèses fortes sur le modèle réseau sous-jacent qui ne colle pas forcément à la réalité des réseaux sans-fil dynamiques.

Toutefois, nous avons vu que l'évaluation de la performance d'un réseau passe par celle d'une distribution, celle des *performances* ou *satisfactions* de ses agents. De par sa nature, celle-ci évolue au cours du temps. À des fins de supervision distribuée, nous avons donc besoin d'évaluer des distributions à temps variable. Par ailleurs, le problème de l'estimation décentralisée d'une distribution à temps variable n'a pas encore été traité à notre connaissance et l'adaptation des travaux précédents à cette généralisation peut s'avérer significativement complexe. Nous proposons alors d'exploiter les outils développés dans les chapitres précédents pour résoudre ce problème.

5.3.2 Estimation décentralisé d'une distribution à temps variable

Estimer une densité de probabilité relève d'un problème statistique. Répartir une tâche entre différents agents est un problème algorithmique. Les algorithmes existant approchent l'estimation distribuée d'une densité de probabilité d'un seul bloc et fournissent des algorithmes spécifiques à un type de réseau (topologie statique et graphe complet) et à un type de densité de probabilité (somme de gaussienne, probabilité fixe dans le temps etc). Adapter ces algorithmes pour prendre en compte à la fois une topologie dynamique et une densité changeante dans le temps nécessite de lourdes modifications. Nous préconisons une autre approche dans laquelle nous transformons le problème initial en un problème de calcul de moyenne que nous savons résoudre et optimiser en fonction de la topologie. Une fois transformé, le problème revient à trouver les données locales que chaque nœud doit échanger afin de converger vers la solution.

5.3.2.1 Mise à profit de la théorie du consensus

Comme nous l'avons étudié dans le chapitre 3, nous sommes capables de suivre la moyenne des métriques d'un ensemble de nœuds alors que le graphe de communication est dynamique et que cette moyenne évolue dans le temps. Il suffit pour cela que les nœuds utilisent un algorithme de type FODAC (pour First Order Dynamic Average Consensus). Pour rappel, une itération de l'algorithme FODAC peut s'écrire ainsi :

$$x_i(t+1) = \delta s_i(t) + \sum_{j=1}^{n} w_{ij} x_j(t), \tag{5.4}$$

Dans cette équation x_i et s_i sont respectivement l'estimation de $\bar{s}$ du nœud i et son état, alors que $\delta s_i(t) = s_i(t) - s_i(t-1)$.

Notons à présent que l'équation (5.4) a juste besoin que les états appartiennent à un espace vectoriel. Comme nous l'avions fait remarqué dans le chapitre 3, une classe de `Mesure` se doit d'implémenter une somme et un produit. Notre approche est donc de transformer le problème d'estimation décentralisé d'une densité de probabilité à temps variable en un *simple* problème de suivi de moyenne. Pour ce faire, nous définissons l'état de chaque agent comme une densité de probabilité, de sorte que leur moyenne soit égale à la densité que nous souhaitons estimer. Les principaux avantages d'une telle approche sont l'immédiateté de la preuve de convergence, et d'un point de vue pratique, la trivialité d'une implémentation orientée objet. Nous expliquons à présent la manière dont nous avons défini l'état des agents afin que leur moyenne représente la distribution à estimer.

5.3.2.2 Consensus de la moyenne pour des fonctions noyaux :

Observons un instant l'équation (5.1),

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{P} \alpha_k K'_k(\mathbf{x}).$$

Il est clair que l'estimation d'une densité par noyau n'est autre qu'une moyenne pondérée de fonctions. De plus, si nous posons $\alpha_k = \frac{1}{P}$, $\forall k$, celle-ci devient une moyenne simple. Ainsi, si nous définissons l'état d'un agent avec la fonction adaptée (comme par exemple la fonction K'_k), il s'ensuit que l'algorithme FODAC traquera la valeur de $\hat{f}$.

Valeurs initiales et convergence L'état s_i d'un agent sera une fonction dans $\mathbb{R}^d$. Cet état ou fonction peut changer au cours du temps. Cette variation peut-être provoquée par exemple, par l'évolution de l'ensemble des échantillons observés. La valeur du consensus sera x_i et par l'algorithme FODAC, nous aurons $x_i \approx \bar{s}$. Dans un souci de clarté, nous simplifions la notation en utilisant s_i, x_i, $\hat{f}_i$ et $\hat{f}$ respectivement à la place de $(s_i(t))(\mathbf{p})$, $(x_i(t))(\mathbf{p})$, $(\hat{f}_i(t))(\mathbf{p})$ et $(\hat{f}(t))(\mathbf{p})$.

Soit un ensemble de P points répartis entre les agents, où chacun d'entre eux possède un sous-ensemble de P_i points ($P = \sum_{i=1}^{N} P_i$). Nous notons $K'_{i,r}$ le noyau mis à l'échelle que l'agent i associe à son r-ème échantillon. En utilisant ces notations, l'estimation KDE centralisée de la distribution lorsque tous les échantillons ont le même poids devient

$$\hat{f} = \frac{1}{P} \sum_{i=1}^{N} \sum_{r=1}^{P_i} K'_{i,r} \tag{5.5}$$

Si nous définissons à présent l'état de l'agent i par $s_i = \sum_{k=1}^{P_i} K'_{i,k}$, puisque la valeur de consensus x_i du nœud i suit la valeur de $\bar{s}$, nous aurons alors

$$x_i \approx \bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{P_i} K'_{i,k}, \quad \forall i. \tag{5.6}$$

En identifiant $\hat{f}$ dans le membre de droite de l'équation (5.6), nous obtenons

$$x_i \approx \bar{s} = \frac{P}{N}\hat{f}. \tag{5.7}$$

Cette dernière équation établit que chaque nœud participant à l'algorithme FODAC estime asymptotiquement la valeur de $\hat{f}$ avec un coefficient multiplicateur $\frac{P}{N}$, qui n'est autre que le nombre moyen d'échantillons par agent.

En organisant les noyaux dans un multi-ensemble (ou dictionnaire) $(\mathcal{K}', m)$ où $\mathcal{K}' = \bigcup_{i,r}\{K'_{i,r}\}$ est le support et m est la fonction de multiplicité, nous pouvons alors écrire

$$\begin{aligned} \hat{f} &= \frac{1}{P}\sum_{i=1}^{N}\sum_{r=1}^{P_i} K'_{i,r} = \frac{1}{P}\sum_{K'\in\mathcal{K}'} m(K')K', \\ \text{où } P &= \sum_{i=1}^{N} P_i = \sum_{K'\in\mathcal{K}'} m(K'). \end{aligned} \tag{5.8}$$

Et nous avons alors :

$$x_i \approx \bar{s} = \frac{P}{N}\hat{f}. = \sum_{K'\in\mathcal{K}'} \frac{m(K')}{N}K' \tag{5.9}$$

Ceci signifie que chaque nœud tend à avoir un ensemble de couple $(K', \frac{m(K')}{N})$ qui est identifiable à un couple *(clé,valeur)* dans une représentation en dictionnaire de $\hat{f}$. De ce fait, la somme des valeurs de ce dictionnaire est $\sum_{K'\in\mathcal{K}'} \frac{m(K')}{N} = \frac{P}{N}$. Ainsi, chaque nœud peut calculer localement le coefficient multiplicateur en sommant toutes les valeurs d'un dictionnaire. Nous expliquons à présent comment un dictionnaire permet de représenter de manière informatique de telles fonctions.

5.3.2.3 Représentation des données par un dictionnaire

Comme expliqué précédemment, nous cherchons à représenter de manière informatique, une fonction particulière afin de pouvoir l'additionner à d'autres et la multiplier par un scalaire. Nous décrivons donc ici une implémentation de la classe `Mesure`, pour laquelle nous devons définir l'addition interne et la multiplication par un scalaire.

D'une manière générale, l'espace des fonctions de $\mathbb{R}^d$ dans $\mathbb{R}$ est un espace vectoriel. Quelque soit l'espace vectoriel considéré, un vecteur peut toujours être représenté par un ensemble de couples (e, α) ou e est l'élément d'une base de l'espace vectoriel et α est un scalaire non nul. Par conséquent, nous représentons un vecteur u par un dictionnaire (ou tableau associatif), $D_u : \mathcal{K}_u \to \mathcal{V}_u$. Ici une clé $c \in \mathcal{K}_u$ est un élément d'une base de l'espace vectoriel tandis que sa valeur $v \in \mathcal{V}_u \subset \mathbb{R}^*$ est le coefficient non nul correspondant.

Dans notre cas, nous considérons l'espace des fonctions dont la base est l'ensemble des noyaux mis à l'échelle. Un noyau mis à l'échelle K'_k est un élément de cette base, et il est pleinement caractérisé par un triplet $(K_k, \mathbf{H}_k, \mathbf{p}_k)$. Nous représentons donc une fonction f par un dictionnaire (ou tableau associatif), $D_f : \mathcal{K}_f \to \mathcal{V}_f$, où une clé $c \in \mathcal{K}_f$ est un triplet caractéristique d'un noyaux à l'échelle.

Dans ce qui suit, nous considérons D_f,D_g, D_h comme les représentations de trois fonctions f, g, h. Le dictionnaire vide sera noté $D_\emptyset$.

Opération d'addition Nous définissons l'addition de deux vecteurs (ou dictionnaires) de la manière suivante :

$$\begin{aligned} & h = f + g \\ & \iff \\ & \mathcal{K}_h = \mathcal{K}_f \cup \mathcal{K}_g - \{c \in \mathcal{K}_f \cap \mathcal{K}_g \,|\, D_f(c) + D_g(c) = 0\}, \\ & \quad \text{et } \forall c \in \mathcal{K}_h \text{ nous avons} \\ & D_h(c) = \begin{cases} D_f(c) & \text{si } c \in \mathcal{K}_f \cap \overline{\mathcal{K}}_g \\ D_g(c) & \text{si } c \in \overline{\mathcal{K}}_f \cap \mathcal{K}_g \\ D_f(c) + D_g(c) & \text{sinon.} \end{cases} \end{aligned} \tag{5.10}$$

Les deux dictionnaires sont fusionnés, et pour chaque clé commune, les valeurs sont additionnées, si la somme est nulle, la clé est supprimée.

Multiplication par un scalaire Pour $\gamma \in \mathbb{R}^*$ nous définissons la multiplication par un scalaire de la manière suivante :

$$\begin{aligned} & h = \gamma f \iff \mathcal{K}_h = \mathcal{K}_f, \text{ et } \forall c \in \mathcal{K}_h, \quad D_h(c) = \gamma D_f(c), \\ & h = 0f \iff \mathcal{K}_h = \emptyset, \end{aligned} \tag{5.11}$$

ce qui signifie que pour chaque clé, la valeur est multipliée par le scalaire et si le scalaire est nul alors le dictionnaire est vidé. La façon dont nous avons défini l'addition et la

multiplication par un scalaire, satisfait les 8 axiomes des espaces vectoriels. Nous ne détaillerons pas les preuves qui sont triviales [5]

5.3.2.4 Un exemple simple de deux agents

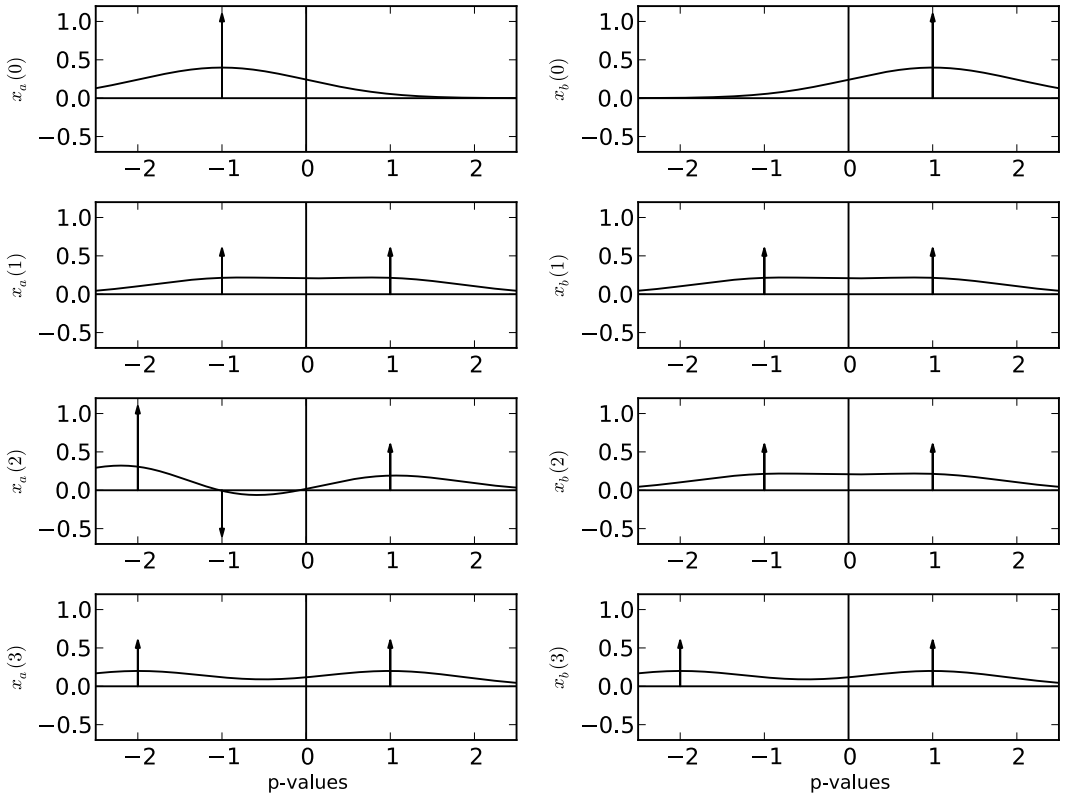


FIGURE 5.4: Quatre premières estimations d'un réseau de 2 nœuds

Pour illustrer notre algorithme, nous considérons le scénario trivial où deux agents a et b sont toujours connectés. Dans cette configuration, la matrice de pondération est constante : $\mathbf{W}(t) = \frac{\mathbf{1}\mathbf{1}^T}{2}$. Les deux agents échangent leurs valeurs et calculent la moyenne. À chaque instant $t \in \{1, 2, \ldots\}$, a et b observent leur propre et unique échantillon $p_a(t)$ et $p_b(t)$ dans $\mathbb{R}$. Pour l'illustration nous prenons les valeurs suivantes : $p_a(t) = -1, -2, -2, -2, \ldots$ et $p_b(t) = 1, 1, 1, 1, \ldots$. La figure 5.4 montre l'estimation gaussienne x_a (gauche) et x_b (droite) des agents pour les quatre premières étapes de l'algorithme et les dictionnaires correspondants. Dans la représentation des dictionnaires, une clé est illustrée par la position d'une flèche, tandis que sa valeur est représentée par sa longueur ainsi que sa direction. Dans la première étape, chaque agent possède sa propre estimation. L'estimation devient commune en $t = 2$ alors que l'agent a vient de changer d'état. À $t = 3$, l'estimation de l'agent a est mis à jour pour être communique à l'agent b. Enfin, pour $t = 4$ les deux agents ont convergé vers la bonne estimation.

5. Ou si besoin en annexe

Remarque 5.2. Notons que l'algorithme de consensus donné par l'équation (3.1) introduit une différence entre les états (équivalent à l'introduction de noyaux négatifs), δs_i. Il résulte de cette différence que x_i n'est pas nécessairement une fonction positive et par conséquent n'est pas une mesure de probabilité (comme défini de manière usuelle). En effet nous avons considéré l'espace des densités de probabilité comme un espace vectoriel alors que celui-ci n'en est pas un. Cependant, ceci ne pose pas de soucis puisque x_i est une estimation de $\hat{f}$. Ce phénomène est principalement dû à l'état transitoire des nœuds et disparait lorsque l'ensemble des échantillons se stabilise comme illustré pour $x_a(t)$, lorsque $t > 1$ dans la figure 5.4

5.3.2.5 Le compromis précision-ressource

Mis à part les ressources de calcul, un algorithme de consensus a besoin de mémoire et de bande-passante réseau. La mémoire nécessaire est directement liée à la taille de la représentation d'une valeur de consensus. La bande-passante réseau, elle, peut s'évaluer rapidement en multipliant cette taille par la fréquence d'échange des messages et le nombre de nœuds dans le réseau. Nous suggérons d'adapter ces paramètres qui par la même occasion affectent la précision de l'estimation. Par exemple, la réduction de la fréquence des messages économise de la ressource réseau. Cependant, si cette fréquence est trop basse, l'algorithme FODAC fait face à une situation de sous-échantillonnage et l'erreur évaluée en utilisant la borne supérieure de [90] sera importante. Réduire la taille de la représentation du consensus est une autre solution pour économiser à la fois de la ressource de communication et de la mémoire. Cette réduction implique une perte d'information et dégrade également l'estimation.

La taille de la représentation de la densité de probabilité suit une progression en $\mathcal{O}(|\mathcal{K}'|)$. Intuitivement, nous pouvons dire que l'ensemble des noyaux d'une distribution constante est collecté en un nombre d'itérations égal au diamètre du réseau, augmentant très rapidement la taille des messages échangés. Ceci est particulièrement vrai dans le cas d'une distribution variant dans le temps, où de nouveaux noyaux sont introduits, tandis que les anciens peuvent ne jamais disparaitre (leurs coefficients tendant néanmoins vers zéro). Pour contrer cette effet néfaste, Hu et coll. suggèrent dans [138] l'usage d'un algorithme de compression entre les échanges. Si ce type de méthode est acceptable pour une distribution constante, il doit être appliqué avec une grande précaution dans notre cas. Utiliser un algorithme de compression revient à changer l'addition de notre espace vectoriel, et par la même occasion perdre son associativité au moment de la compression [6]. Par conséquent les propriétés de l'algorithme FODAC ne sont plus garanties, laissant

6. Ainsi, lorsqu'un nœud effectue une somme de trois distributions f, g, h, on aura $(f + g) + h \neq f + (h + g)$ puisque l'addition perd de l'information.

possiblement les nœuds s'éloigner progressivement de la moyenne de leur états. Nous proposons une autre façon de procéder qui conserve l'associativité de l'addition en limitant la taille de l'ensemble des noyaux, autrement dit la dimension de notre espace vectoriel. En effet la taille de la représentation du consensus augmente uniquement avec l'introduction d'un nouveau noyau. Ainsi, utiliser un ensemble fini de noyaux limitera cette croissante. Notre approche est alors de borner et partitionner l'ensemble des fonctions noyaux, ce qui est équivalent à classer les échantillons dans des catégories comme pour un histogramme.

Remarque 5.3 (Apprentissage automatique distribué). Dès lors qu'un problème initial est de calculer une combinaison linéaire des informations locales à chaque nœud, la stratégie du consensus peut être exploitée. Certains problèmes de classification peuvent être résolus ainsi comme ceux des *méthodes d'ensemble* (voir [139]). La démarche est alors non plus de trouver un algorithme permettant de distribuer le calcul initial, mais plutôt de transformer celui-ci en un calcul de moyenne, de sorte à pouvoir le résoudre par un algorithme de consensus.

5.4 Cas d'études

Notre méthode a été conçue à la base pour superviser des réseaux. Nous l'évaluons au travers de simulations NS-3. Cette section étudie deux scénarios différents. Dans le premier, nous illustrons l'évaluation distribuée de la *performance* et du *contexte*. Dans le second scénario, nous montrons comment le modèle de la section 5.2 peut être utilisé pour analyser l'évolution de la *performance*.

5.4.1 Mesure du contexte et de la performance

Après avoir décrit les paramètres expérimentaux, nous spécifions les opérations de mesures distribuées du réseau et enfin nous détaillons les résultats obtenus.

5.4.1.1 Paramètres expérimentaux

Nous simulons un réseau sans-fil mobile de type ad-hoc pendant 100 secondes. Pour ces simulations, nous découpons géographiquement le réseau en zones. Nous avons deux zones carrées dites primaires et nous avons placé 60 nœuds dans chacune d'elles. La taille des zones sont telles que les nœuds d'une même zone sont (la majorité du temps) dans un même composant connexe du graphe de communication. En effet, dans notre cas, le graphe de communication peut être modélisé par un graphe Euclidien (précédemment

étudié), et la probabilité d'avoir un composant unique dépend de la portée de communication, de la géométrie (taille et forme) des zones ainsi que du nombre de nœuds. Les deux zones primaires sont séparées par une zone de transition comme illustré figure 5.5.

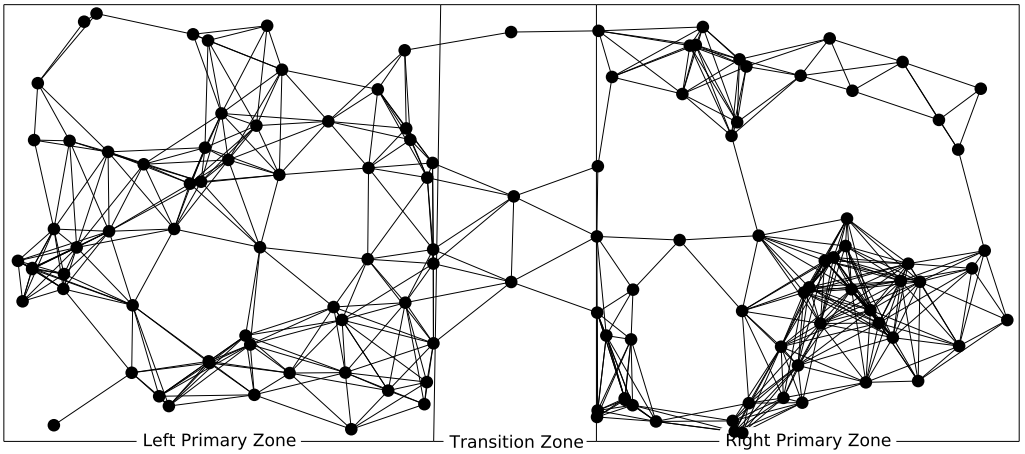


FIGURE 5.5: Topologie réseau considérée avec 3 nœuds de la zone de transition

Deux nœuds appartenant à des zones primaires différentes ne peuvent communiquer sans l'aide d'un nœud dans la zone de transition. Tous les nœuds se déplacent selon un modèle de mobilité de type *random waypoint* dans la limite de leurs zones. Durant la simulation ils se déplacent entre une et trois fois à la vitesse moyenne de 4.0m/s. Un tel zonage géographique et une telle mobilité nous permettra d'étudier la robustesse de la mesure distribuée. Les nœuds ont une unique interface sans fil, leur couche physique et d'accès au médium sont simulées en utilisant le modèle YansWifi de NS-3 où les contrôleurs sont configurés en mode ad-hoc et utilisent un algorithme d'adaptation automatique de débit sans qualité de service. Les routes sont découvertes au moyen d'AODV qui est communément utilisé dans les topologies dynamiques. En terme de trafic, chaque nœud choisi 3 destinations vers lesquelles il envoie un trafic intermittant (de type *ON/OFF*) à débit constant (CBR pour Constant Bit Rate) en utilisant UDP. Durant la période *ON*, toutes les sources ont le même débit qui est de 10KB/s, ce trafic nous permettra de définir l'utilité de chacun des nœuds.

5.4.1.2 Utilité des agents et contexte

Nous supposons que chaque seconde, un nœud relève sa position et collecte des métriques qui sont dérivées de son trafic local (et en pratique pourront être extraites des compteurs de leur MIB (pour Management Information Base). À chaque période de consensus τ, les nœuds communiquent les informations requises pour mettre à jour leur valeur de

consensus (degré et dictionnaire). Les métriques liées au trafic sont le nombre d'octets traités par la couche IP issus des sources UDP locales (*sent*), le nombre d'octets issus des sources d'autres nœuds (fwd) et le nombre d'octets rejetés (*dropped*). Pour décrire le niveau local de satisfaction, nous dérivons de ces trois variables d'état une valeur de *performance* ou satisfaction simple $sat = \frac{sent+fwd}{sent+fwd+dropped}$, qui décroit lorsqu'un nœud n'est pas capable de traiter un paquet (problème de route, surcharge...). Nous conduisons trois consensus dynamiques, le premier traque la valeur moyenne de la métrique *sat*, le second traque sa distribution dans l'ensemble du réseau. Dans le dernier consensus, nous estimons la distribution (multivariée) géographique de la population des nœuds.

5.4.1.3 Résultats de simulation

Nous expliquons à présent les trois consensus et illustrons chacun d'eux par une figure.

Estimation de la satisfaction moyenne : Une solution simple pour un nœud d'estimer la valeur moyenne d'une métrique est d'utiliser sa valeur locale. Dans ce cas, l'erreur est donnée par la déviation absolue moyenne. Dans un premier consensus, chaque nœud estime la valeur moyenne de leur satisfaction. La figure 5.6 montre comment l'erreur absolue moyenne est réduite par l'algorithme FODAC et illustre le compromis qu'il existe entre la précision et la consommation de la bande passante. Sur la figure supérieure

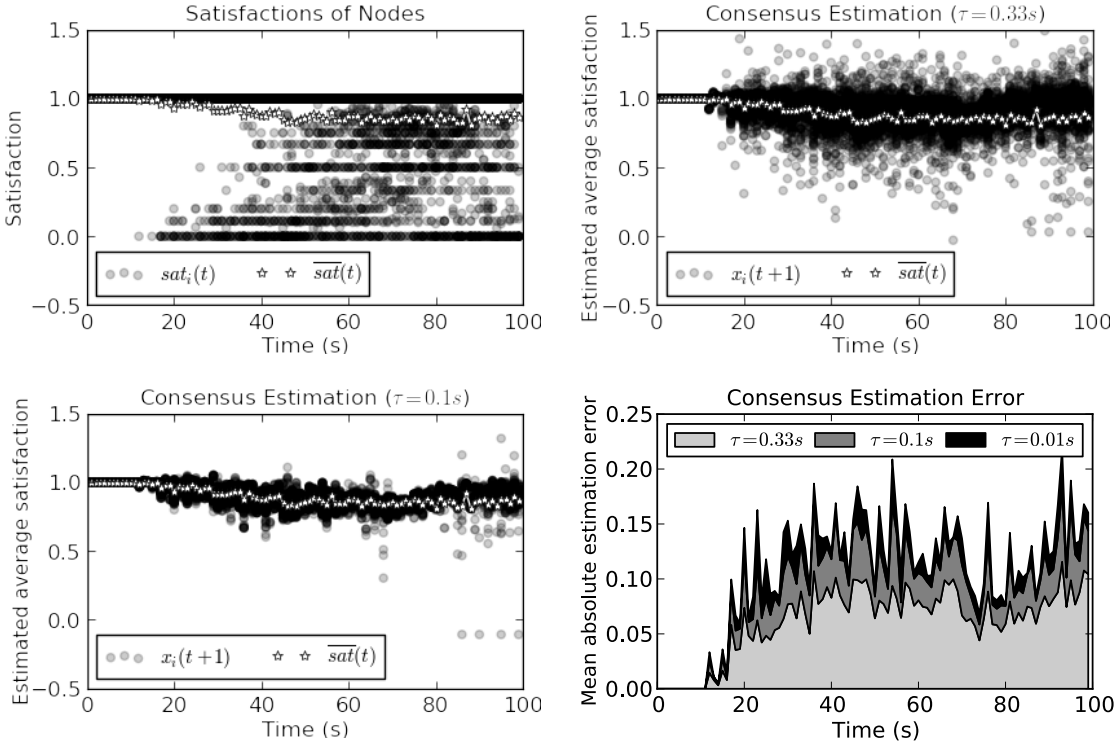


FIGURE 5.6: Estimation de la satisfaction moyenne

gauche, nous traçons la satisfaction $sat_i(t)$ de chaque nœud i toutes les secondes ainsi que leur moyenne. La satisfaction moyenne décroit progressivement à cause des déplacements des nœuds et du démarrage des sources UDP. Les figures supérieure droite et inférieure gauche représente l'estimation $x_i(t)$ par l'algorithme FODAC, de cette moyenne pour des périodes de consensus de $\tau = 0.33s$ et $\tau = 0.1s$, respectivement. Nous pouvons remarquer la réduction de l'erreur moyenne avec la période de consensus. Durant les 20 dernières secondes, un nœud estime la moyenne à zéro, ce nœud était un nœud de la zone de transition n'ayant plus de voisins. Dans la dernière figure, nous détaillons l'erreur absolue moyenne $||\mathbf{x}(t+1) - \mathbf{1}.\bar{x}(t)||_1/n$ pour différents ordres de grandeur de période de consensus. De cette figure, nous pouvons dire intuitivement qu'il existe un point de fonctionnement pour lequel une réduction supplémentaire de l'incertitude ne vaut pas le surplus de bande-passante qui lui est nécessaire. Ce compromis peut être formulé comme un problème d'optimisation (que nous laissons non résolu) pour lequel la ressource à maximiser serait le Shannon.

Estimation de distribution des satisfactions : Une des contributions principales de ce chapitre est d'estimer de manière décentralisée une distribution variant dans le temps. Nous illustrons cette estimation au travers de la figure 5.7 avec une période de consensus de $\tau = 0.1s$. La figure supérieure est l'estimation de la distribution au cours du temps (qui est similaire à celle de l'estimation centralisé) pour un nœud de la zone de gauche. La figure inférieure est l'estimation de cette même distribution pour le nœud de la zone de transition subissant des problèmes de connexion durant les 20 dernières secondes. Au début de la simulation, les nœuds estiment que tout le réseau est pleinement satisfait. Cette distribution évolue au cours du temps et les deux nœuds sont au fait de cette évolution et sont en accord sur l'estimation. Comme attendu, leurs estimations diffèrent sur les 20 dernières secondes car le nœud de la zone de transition est déconnectée. Cependant, comme l'hypothèse (H.2) est respectée, ce nœud corrige son erreur aussitôt en rejoignant le réseau.

Estimation de la densité géographique : Enfin, nous considérons l'estimation d'une densité multivariée à temps variable. Pour ce dernier cas, nous considérons la position géographique des nœuds sur la carte. Comme nous l'avions remarqué en 5.3.2.5, la taille des messages croit avec le cardinal de l'ensemble des noyaux. Vu que la plupart des méthodes de compression ne peuvent être utilisées sans perdre l'associativité de l'addition, notre technique est de limiter l'ensemble des noyaux. Dans ce cas, les nœuds considèrent un ensemble de 216 noyaux dont les positions sont uniformément réparties sur la carte ($\approx 9 \times 10^{-4}$ noyaux/m^2). Au lieu de communiquer le noyau associé à leur position exacte, les nœuds utilisent le noyau le plus proche dans l'ensemble des noyaux.

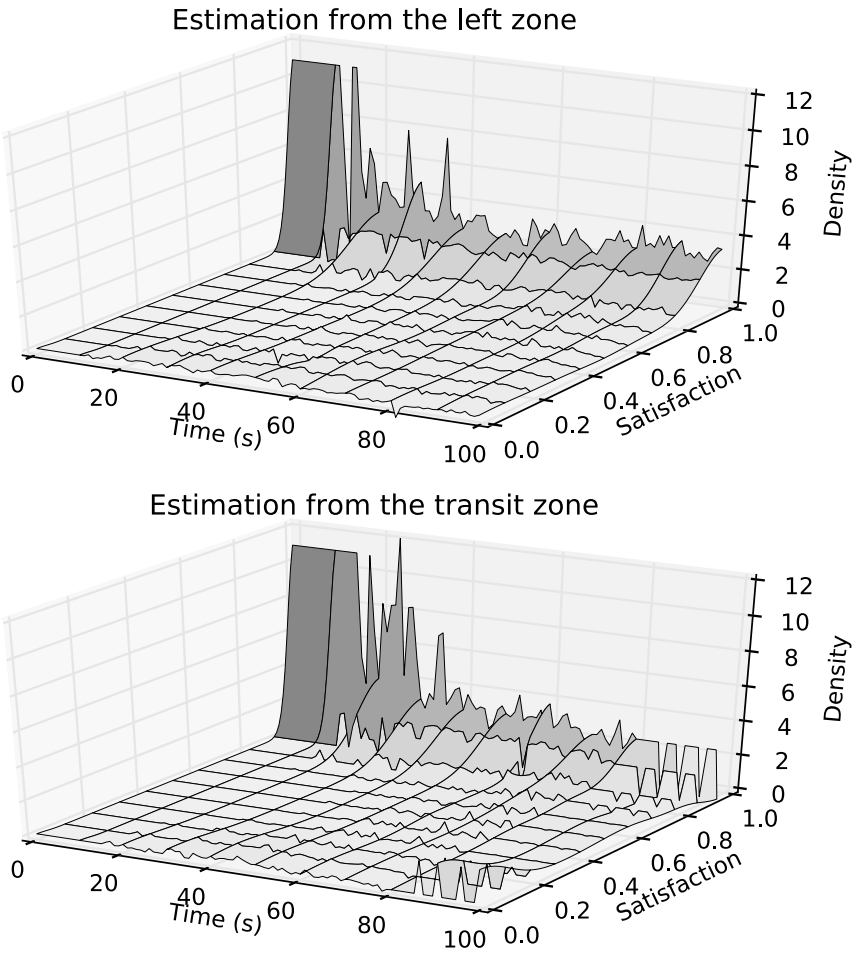


FIGURE 5.7: Estimation de la distribution des satisfactions

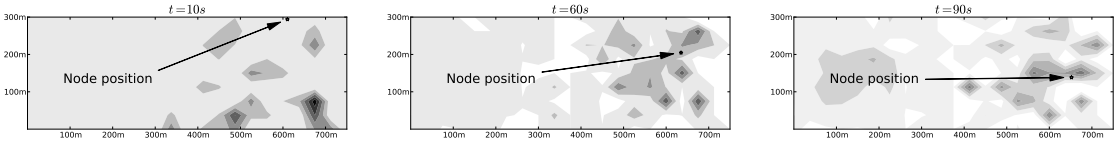


FIGURE 5.8: Estimation de la densité géographique avec 216 noyaux à $t = 10s$, $t = 60s$ et $t = 90s$, pour un nœud évoluant dans la zone de droite.

La figure 5.8 est la densité géographique estimée d'un nœud situé dans la zone de droite à différents instants. Durant la simulation, la zone de transition relaie l'information entre les zones, et le nœud améliore sa vue de la carte complète au cours du temps, devenant capable de distinguer les différentes zones. Un point intéressant ici et que le nœud possèdera une vue précise de son voisinage tandis qu'il aura une vue plus approximative des zones distantes. En ce sens, un nœud sera donc apte à suivre l'évolution de son contexte en donnant plus d'importance à son voisinage.

5.4.2 Analyse de flux d'observation

Dans ce cas d'étude, nous considérons le même type de réseau que dans la section 5.4, mais que nous restreignons à quelques nœuds et pour un scénario particulier. Nous décrirons dans un premier temps les conditions de l'expérience, puis l'instanciation du modèle de SMA. Dans un dernier temps, nous montrons comment ce modèle peut être exploité dans l'analyse de la performance.

5.4.2.1 Paramètres expérimentaux

Le scénario considéré est illustré par la figure 5.9. Au début du scénario, toutes les sources sont stoppées, et les nœuds sont tous satisfaits. Les sources de trafic commencent à

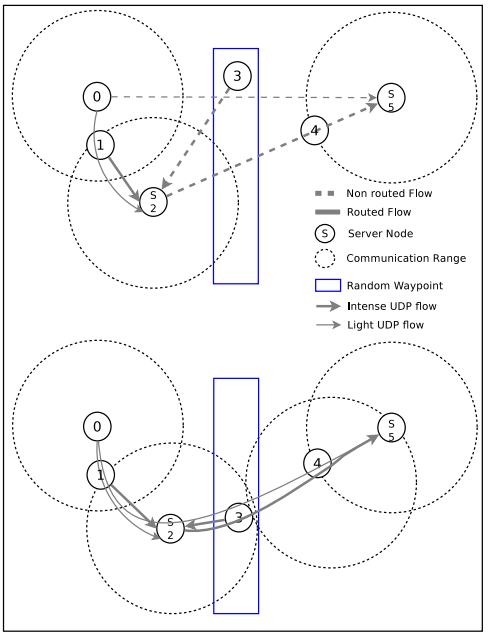


FIGURE 5.9: Dans ce scénario, les nœuds 2 et 5 sont des destinations UDP. Le nœud 3 est mobile. Dans la figure supérieure, le nœud 2 peut subir une surcharge alors que dans figure ci-dessous il n'y a plus de route vers le nœud 5.

transmettre au bout de 2 secondes et l'indice de performance commence brutalement à se dégrader. Puis pour chaque mouvement significatif du nœud 3, les routes IP sont perdues et recouvrées, impactant significativement l'indice de performance. Lorsque les routes sont établies, les fluctuations peuvent s'expliquer par la variation du délai. Lorsque le nœud 2 rencontre des difficultés à transmettre, son nombre de retransmissions augmente et a un impact direct sur le délai moyen. Même si seulement certains paquets sont concernés, ils peuvent avoir un impact important sur la satisfaction du nœud source si ces paquets sont les seuls qu'il envoie. Puisque l'indice de performance considère tous

les nœuds sans aucun regard sur le volume de ses sources, nous percevons facilement un impact sur celui-ci.

5.4.2.2 Instanciation du modèle de SMA

De par sa conception, NS-3 est un simulateur à événements discrets, il offre donc des propriétés intéressantes pour instancier notre méthode d'analyse. Parmi ces propriétés son système de trace nous facilite l'implémentation des observations d'événements. Dans notre cas, le temps est un flottant dont l'origine est le début de la simulation et chaque nœud est identifié de manière unique en se basant sur son adresse MAC.

Événements considérés : Nous avons définis plusieurs types d'événements. Certains sont locaux à un agent (au sens des automates) comme par exemple son déplacement, la modification de sa table de routage etc... et ne sont pas observables depuis d'autres agents. D'autres sont publics (aussi dit en sortie), ce sont principalement des événements de type `Paquet`, observables par d'autres agents. La table 5.2 détaille d'avantage les différents types d'événements considérés.

Table 5.2: Description des différents types d'événements

Type	**Information**
Paquet	Paquet capturé en mode promiscuité accompagné des champs *radio-tap*
Rtam	Un attribut de la table de routage a été modifié (nombre d'entrées valides, chemin le plus long etc...)
Déplacement	Modification de la position
Ipv4Drop	Paquet éliminé pour une erreur de routage
PhyRxError	Erreur au niveau Physique lors de la réception d'une trame
PhyRxDrop	Une Trame est éliminée en réception
MacTxDrop	Paquet éliminé avant d'avoir été en attente de transmission
MacRxDrop	Paquet éliminé après la couche physique
MacTxDataFailed	Échec de la Transmission du Paquet de donnée au niveau de la couche MAC
MacTxRtsFailed	La demande de transmission a échoué au niveau de la couche MAC.
MacTxFinalDataFailed	Le nombre de retransmissions consécutives (MacTxDataFailed) a dépassé la limite.
MacTxFinalRtsFailed	Le nombre de demandes de transmissions consécutivement échoué (MacTxRtsFailed) a dépassé la limite.

Utilité des agents : Comme nous l'avons expliqué, nous pouvons extraire l'indice de performance de la trace des observations. Nous conservons le même pas de temps que

dans l'exemple précédent afin d'agréger les observations et de calculer les descripteurs (1 seconde). Cette valeur est suffisamment fine pour suivre l'évolution de la performance tout en restant assez grande pour avoir une signification statistique à l'échelle temporelle du paquet de communication. Dans ce cas d'étude nous ne gardons pas le même indice de performance que le précédent qui était directement lié à un ratio du nombre d'événements par unité de temps. Nous nous basons plutôt sur le délai de transit d'un paquet. Chaque paquet généré par une source UDP possédera un score en fonction de son délai. La fonction de score est la suivante :

$$score(pqt) = max\Big(0, \frac{seuil - delai(pqt)}{seuil}\Big)$$

Le score d'un paquet décroit linéairement avec son délai jusqu'à atteindre la valeur nulle. Il vaut 1 pour un délai nul, et 0 si le délai dépasse un certain seuil (ou si le paquet est perdu). Nous réglons une valeur arbitraire de seuil à 10ms. Le délai associé à un paquet est la différence entre l'estampillage de la première observation du paquet sur le médium sans fil et celui de la première observation par sa destination. La satisfaction ou performance d'un nœud pour une période de temps est donnée par la moyenne des scores des paquets générés par ses sources UDP durant cet intervalle. Enfin nous choisissons d'utiliser la performance moyenne des agents comme politique d'évaluation du système.

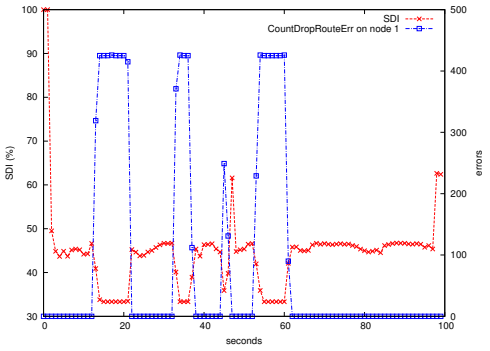
Construction des descripteurs : Nous avons construit les descripteurs en groupant les observations par observateur et par seconde. Nous avons construit plus de 20 descripteurs mais nous présentons en table 5.3 uniquement les plus significatifs.

Nom	**Description**
AvgnbGateway	Nombre moyen de nœuds passerelles dans la table de routage
AvgnbValid	Nombre moyen d'entrées valides dans la table de routage
CountPhyRx	Nombre de trames reçues
CountAllRetry	Nombre de trames capturées avec un drapeau de retransmission
CountMyRetry	Nombre de trames émises avec un drapeau de retransmission
CountMyIpFlow	Nombre de destinations des sources locales
CountMyUdpSrc	Nombre de flux locaux actifs
CountAllFlow	Nombre de flux routés
CountPhyRxError	Nombre d'événements PhyRxError
CountPhyRxDrop	Nombre d'événements PhyRxDrop
CountDropRouteErr	Nombre de paquet supprimés suite à une erreur de routage

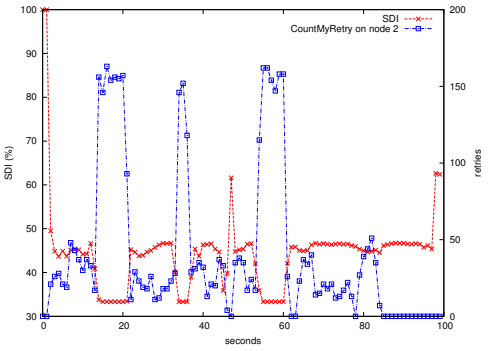
TABLE 5.3: Détail des descripteurs

5.4.2.3 Analyse de séries temporelles

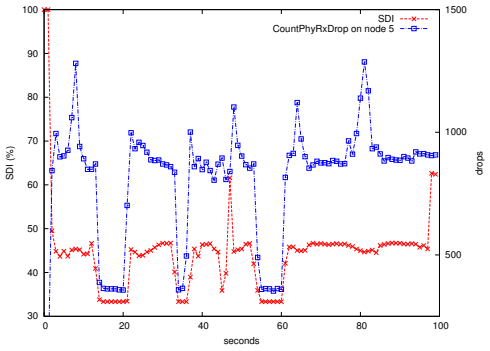
Après avoir regroupé les événements observables par similarité, nous en avons extrait des séries temporelles de descripteur. Nous avons étudié la relation statistique entre les différents *descripteurs* et la performance. Pour une fenêtre glissante d'échantillons, nous calculons le coefficient de Pearson $r_{i,p}$ entre le descripteur d_i et la performance p. Le but est de trouver pour chaque nouvel échantillon de temps le coefficient ayant la plus grande valeur absolue afin de déceler un possible lien avec l'évolution de la performance. Après



(A) CountDropRouteErr on node 1



(B) CountMyRetry on node 2



(C) CountPhyRxDrop on node 5

FIGURE 5.10: Évolution temporelle de la performance et de trois descripteurs

avoir calculé les coefficients de corrélation, nous trouvons des valeurs significatives pour trois descripteurs qui sont illustrés en figure 5.10. Le nombre d'erreurs de routage sur le nœud 1, le nombre d'erreurs de réception sur le nœud 5 et le nombre de retransmissions sur le nœud 2 ont respectivement les scores de corrélation -0.92, 0.79, -0.88. La figure 5.10a illustre clairement que les principales fluctuations de l'indice de performance sont liés aux erreurs de routage. En effet, le nœud 1 ne peut pas trouver de route vers le nœud 5 puisque le nœud 3 a changé de place. Les retransmissions au niveau du nœud 2 sont détaillées dans la figure 5.10b. Il y a effectivement un lien avec l'indice de performance lorsque la route est disponible et que les nœuds 2 et 3 ont des difficultés à se joindre. De prime abord, on pourrait penser que les retransmissions du nœud 2 sont liées à l'élimination des paquets par la couche physique du nœud 5. En fait ces

deux descripteurs sont corrélés négativement. En effet, ces nœuds ne peuvent pas ce joindre du fait de leur distance respective. Puisque le nombre d'éliminations de paquet est bien plus grand (1000) que le nombre de retransmissions (100), on peut se douter qu'il n'y pas de relations entre les observations sous-jacentes à ces descripteurs. En revanche l'élimination de paquets pourrait provenir du fait que le nœud 5 soit toujours en limite de portée du nœud 2, suffisamment pour capturer son trafic mais pas assez pour l'interpréter. La figure 5.10c, confirme que le nœud 5 n'élimine pas de paquet pour des indices de performances faibles, puisque le nœud 2, ne trouvant pas de route, fait face à des erreurs de routage et ne génère plus de trafic.

Remarque 5.4 (descripteurs et consensus). Dans un soucis de collaboration chaque nœud peut maintenir à jour un histogramme de ces descripteurs dans lequel chaque poids représente un coefficient de corrélation. En appliquant un consensus sur cette histogramme, chaque nœud peut avoir une représentation actualisée des phénomènes les plus liés avec l'évolution de la performance.

5.5 Conclusion

Après avoir étudié dans les chapitres précédents la diffusion et le transfert d'une information dans les réseaux, nous nous sommes concentrés sur le contenu d'une telle information. Nous avons proposé dans ce chapitre une modélisation des systèmes distribués permettant de prendre en compte les notions de *performance* et de *contexte*. Motivé par la supervision distribuée de cette performance au sein d'un réseau dynamique, nous avons exploité les algorithmes de consensus présentés précédemment pour estimer une distribution multivariée à temps variable de manière décentralisée. En exploitant une représentation des espaces vectoriels par des dictionnaires, nous avons transformé ce problème en un problème de suivi de moyenne ce qui nous a permis de conserver les mêmes propriétés de convergence que celles établies par la théorie du consensus. Grâce à notre approche, chaque nœud peut suivre l'évolution de la performance et accéder à une représentation du contexte de son réseau qui est d'autant plus précise pour un voisinage proche. Enfin nous avons illustré par un cas d'étude comment les nœuds pouvaient exploiter ces informations pour analyser l'évolution de la performance.

Chapitre 6

Conclusion et perspectives

6.1 Rappel du contexte

Ces quinze dernières années, le développement des technologies de communication sans fil a révolutionné le monde des réseaux. Elles ont créé un accès en continue à l'information mais aussi une dépendance à celle-ci, se rendant par la même occasion indispensable. Avec un nombre de terminaux mobiles croissant, les réseaux de communication ont pu voir leurs structures évoluer vers des topologies de plus en plus dynamiques. Au delà de ce changement de structure, le progrès de la micro-électronique a également permis de développer des objets communicants adaptés à des besoins spécifiques. Ces objets aux ressources hétérogènes, sont de nouvelles sources d'information mais aussi de nouvelles destinations qu'il faut peu à peu intégrer à nos réseaux. Gérer un réseau est donc devenu une tâche complexe qu'il devient nécessaire de déléguer aux objets eux-même.

Avant de se lancer dans une tâche de gestion, une première étape est celle de la prise d'information, la prise de *mesure*. Mesurer nos réseaux est donc une étape cruciale. Outre le fait que sans celle-ci aucune autonomie n'est possible, mesurer les réseaux aujourd'hui c'est accéder à une masse d'information qui nous permet de comprendre une partie de notre société. L'information de mesure n'est plus seulement un délai, ou un débit, c'est aussi une donnée qu'un objet personnel renferme et qui reflète un mode de vie.

Ce cas d'utilisation transverse de la mesure distribuée des réseaux aura été le fil conducteur de cette thèse. Tout en permettant une représentation riche d'une mesure, nous avons cherché à améliorer sa diffusion, son transfert mais aussi à donner du sens à celle-ci.

6.2 Méthodologie et approche adoptée

Face à la complexité de notre problème, nous avons choisi de décomposer celui-ci en différentes étapes qui sont : l'acquisition d'une mesure, sa représentation, son partage, et son analyse. Nous nous sommes concentrés essentiellement sur les trois dernières étapes.

Pour chaque étape nous avons donné un modèle approché mais suffisamment précis pour pouvoir raisonner de manière formelle. Ainsi nous avons raisonné dans des cadres comme ceux de l'algèbre linéaire, de la théorie des graphes, de l'algorithmique, ou encore de la statistique. Puis, en utilisant ces représentations, nous avons exploité des outils issus de différents domaines scientifiques pour avoir des premiers résultats théoriques. À ce titre nous nous sommes inspirés des domaines de la théorie du contrôle (chapitre 3), de l'ordonnancement (chapitre 4) et de la statistique (chapitre 5). Enfin, nous avons procédé à des expérimentations afin de pouvoir appliquer nos modèles théoriques et de vérifier dans quelles mesures, ceux-ci sont acceptables. Nous avons pour cela suivi différentes stratégies en fonction de la faisabilité de l'expérimentation, et nous avons notamment effectué du calcul scientifique (chapitre 3), de l'expérimentation réelle (chapitre 4), et de la simulation par événement discret (chapitre 5). Cette démarche nous aura permis d'apporter des contributions sur différents points de la mesure distribuée dans les réseaux. Nous rappelons à présent ces contributions.

6.3 Résumé des contributions

En premier lieu nous avons donné au problème de la mesure distribuée une forme algébrique. Nous avons montré que ce problème pouvait être modélisé par un produit matriciel et nous avons défini les notions de précision, de latence, et de coût. Ceci nous a permis de proposer une conception en trois couches que nous avons nommées `Mesure`, `Diffusion`, `Transport`. Au niveau de la couche de `Diffusion`, nous avons étudié le cas des topologies dynamiques pour lequel nous proposons l'utilisation d'un algorithme de consensus de la moyenne. Pour cet algorithme nous avons élaboré l'heuristique SOS qui offre une meilleure précision que ses concurrentes pour un coût et une latence égale. Nous avons également étudié l'impact des conditions initiales pour de tels algorithmes et nous avons montré l'importance du biais d'échantillonnage dans les graphes où la mesure est assortative.

Nous avons dans un second temps travaillé au niveau de la couche `Transport`. Sur des réseaux utilisant le protocole TCP, nous avons proposé un mécanisme pour découpler l'ordonnancement des flux et le contrôle de congestion. Par ce découplage nous offrons

sur chaque route une différenciation de service des flux qui nous permet de maitriser les ressources allouées aux flux de mesure, à ceux de contrôle et à ceux de données. Nous offrons ainsi une classe de trafic *less than best effort* pour la mesure active. Par ailleurs lorsque plusieurs mesures sont supervisées en parallèle, nous pouvons donner la priorité aux mesures importantes, où à celles assortatives. En utilisant les résultats classiques de la théorie de l'ordonnancement nous avons également montré puis vérifié expérimentalement qu'en appliquant des politiques basées sur la taille des flux nous pouvions réduire le temps moyen de transfert des mesures.

Enfin, après s'être concentrés sur la manière de transférer et de diffuser une mesure, nous nous sommes penchés sur la mesure elle-même. Nous avons proposé une modélisation des systèmes distribués permettant de prendre en compte les notions de *performance* et de *contexte*. Nous avons défini ces deux notions respectivement en terme de distribution des utilités et distribution de métriques. Ces distributions étant variables dans le temps, nous avons proposé un algorithme d'estimation décentralisé d'une densité de probabilité à temps variable. Pour ce faire nous avons montré que nous pouvions transformer ce problème en un consensus de la moyenne et nous avons exploité les résultats des chapitres précédents pour réaliser cette estimation. Nous avons vérifié par simulation avec le simulateur NS-3, la validité de nos algorithmes et nous avons montré comment chaque nœud pouvait suivre l'évolution de la performance et accéder à une représentation du contexte de son réseau qui donne d'avantage d'importance à son voisinage proche. Enfin nous avons illustré par un cas d'étude comment les nœuds pouvaient exploiter ces informations pour analyser l'évolution de la performance.

6.4 Pistes de recherches et dernières remarques

Alors que nous arrivons à la fin de ce manuscrit, nous proposons des pistes de recherches pouvant donner suite aux travaux que nous avons menés. Nous étudions ces pistes par chapitre.

En ce qui concerne le chapitre 3, nous avons cherché à réduire l'erreur moyenne dans les étapes de consensus. Cette optimisation aurait pu être exprimée en terme d'utilité, où chaque agent chercherait à réduire son erreur d'estimation. Se concentrer sur la réduction de l'erreur moyenne peut être vu comme une politique d'évaluation de la performance et nous pourrions étudier d'autres indicateurs. Par exemple il serait possible d'étudier et de comparer les algorithmes sur la base de la distribution des erreurs. Comme dans d'autres domaines, nous pourrions nous attendre à devoir faire un compromis entre l'erreur moyenne (le biais) et la variance. En ce sens, nous pouvons considérer l'information, ou la mesure comme une *ressource* qui doit être partagée et qui nécessite un compromis

entre partage efficace (réduction de l'erreur moyenne) et partage équitable (réduction de la variance de l'erreur)

En ce qui concerne le chapitre 4, nous nous sommes concentrés sur le protocole de transport TCP car c'est celui qui est le plus répandu actuellement. Nous avons proposé une implémentation de notre mécanisme par une couche session dans l'espace utilisateur. Il serait envisageable de passer ce mécanisme comme une option de TCP, au même titre que les acquittements sélectifs. Outre cet aspect d'implémentation, un second travail peut-être fait au niveau de l'algorithme d'ordonnancement. Nous avons spécifié des ordonnanceurs basés sur la taille des flux. Néanmoins nous pourrions fournir à un ordonnanceur de plus amples informations, comme par exemple sur l'utilité de chacune des applications. Un ordonnanceur au fait de cette information serait en mesure d'optimiser l'évaluation d'une performance dans le sens que nous avons défini au chapitre 5.

Enfin dans le chapitre 5, nous avons cherché à suivre l'évolution de la performance du réseau ainsi que celle de *descripteurs*. La construction de *descripteurs* pertinents est une voie qui mérite d'être explorée. C'est dans cette direction qu'il sera possible de disposer des descripteurs les plus représentatifs du contexte permettant à un nœud de choisir la configuration réseau à adopter. Aussi l'analyse de la trace des observations est une seconde piste. Nous avons en effet décrit le réseau comme une machine à état, pour laquelle nous n'avons pas défini de transition. Il serait intéressant, à des fins de prédiction de reconstruire ces règles de transition à partir de la trace des observations.

Notre dernière remarque se tourne vers les limites théoriques de nos travaux, Comme nous l'avons fait remarqué, il existe un compromis entre la précision et l'occupation de ressources d'un algorithme de supervision. La limite de ce compromis est fixée par la dynamique du signal, ou de la métrique que nous souhaitons superviser. Ceci signifie qu'il ne sera pas possible de mesurer des phénomènes dont l'échelle de temps est trop faible. Néanmoins, lorsque les phénomènes à observer ont une échelle humaine, superviser un système distribué par un réseau de communication nous semble une application parfaitement envisageable pour la mesure distribuée et la construction de systèmes intelligents.

Annexe A

Heuristiques non paracontractante et topologies cyclique

Dans cette annexe, nous donnons un contre-exemple, prouvant que les heuristiques NW et MDW ne sont pas para-contractante. Ainsi le fait qu'une matrice de pondération W soit stochastique double et que la chaine de Markov associée soit apériodique et irréductible ne suffit pas à garantir la convergence la convergence du consensus. Les heuristiques NW et MDW ne convergent donc pas toujours comme cela avait été affirmé respectivement par [96] et [87]. Nous prenons l'exemple d'un graphe $G = (\mathcal{N}, \mathcal{E})$ cyclique de taille paire $n > 3$. Pour ce type de graphe, les heuristiques NW et MDW proposent une pondération constante qui est :

$$w_{ij} = \frac{1}{2} \text{ si } (i,j) \in \mathcal{E} w_{ij} = 0 \text{ sinon}$$

Dans ce cas $w_{ii} = 0, \forall i$ et la matrice de pondération correspondante est de la forme

$$L = \begin{bmatrix} 0 & \frac{1}{2} & \dots & \dots & \dots & \dots & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & \dots & \dots & \dots & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \dots & \dots & 0 \\ \vdots & \vdots & \dots & \dots & \ddots & \dots & \frac{1}{2} \\ \frac{1}{2} & \dots & \dots & \dots & \dots & \frac{1}{2} & 0 \end{bmatrix}$$

Il s'avère que -1 est une valeur propre de W. La valeur propre associée est le vecteur $[1, -1, \dots, 1, -1]^T$.

Dans ce cas $\rho(W) \geq 1$. La matrice de pondération n'est alors pas *paracontractante* :

$$W.x \neq x \implies ||Wx||_2 < ||x||_2$$

.

Si l'on se réfère à [87], le taux de convergence est :

$$r_{asym}(W) = \rho(W - \frac{\mathbf{1}\mathbf{1}^T}{n}).$$

Il peut être exprimé par :

$$\rho(W - \frac{\mathbf{1}\mathbf{1}^T}{n}) = \max\lambda_2(W), -\lambda_n(W),$$

où λ_i désigne la i-ème plus grande valeur propre.

Ainsi, si -1 est la plus petite valeur propre de W alors $r_{asym}(W) = 1$, si ce n'est pas le cas alors $r_{asym}(W) > 1$.

Par conséquent, $r_{asym}(W) \geq 1$ et la condition de convergence $r_{asym}(W) < 1$ n'est pas respectée. La figure ci-dessous illustre une situation dans laquelle la convergence ne peut être atteinte.

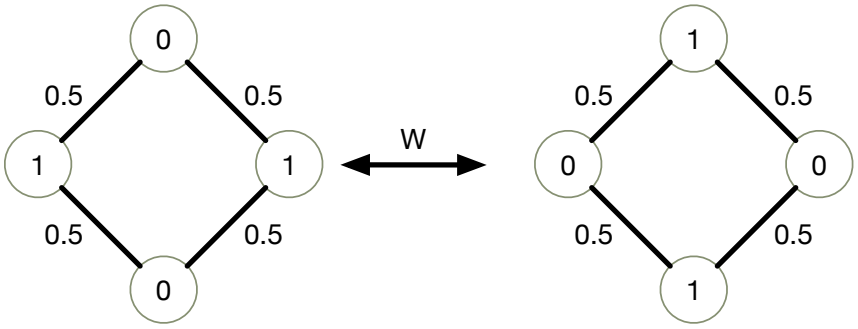


FIGURE A.1: Cycle de 4 nœuds avec les poids NW et MDW pour une état initial donné

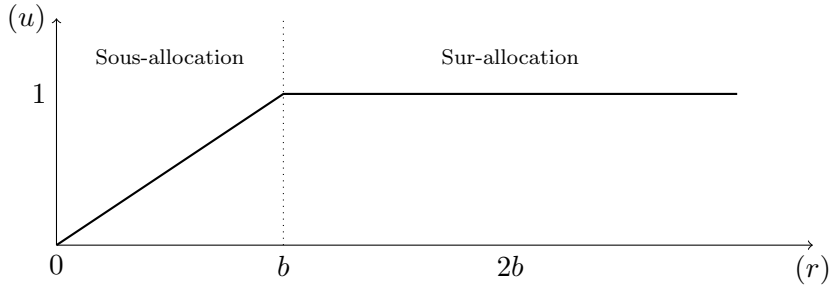
On voit clairement que dans cette situation les valeurs de consensus des nœuds sont $[1, 0, 1, 0]^T$ et qu'à chaque itération les nœuds permutent ces valeurs en appliquant l'heuristique. Une convergence est alors impossible.

Annexe B

Complément du chapitre 5

B.1 Suite de l'exemple 5.1

Dans cette section, nous reprenons l'exemple 5.1 qui traite de l'allocation d'une ressource entre n agents. L'état d'un agent est caractérisé par un vecteur de deux éléments (b, r), symbolisant respectivement son besoin et sa ressource. En fonction de l'algorithme d'allocation de ressource appliqué, chaque agent peut recevoir une certaine quantité de ressources. L'utilité d'un agent sera définie par $u((b, r)) = \min(1, \frac{r}{b})$. Cette utilité est tracée ci-dessous, pour une valeur fixe de b.



Tous les agents appartenant à la même classe, la distribution de leur utilité caractérise la performance du système. Nous étudions donc cette distribution pour 4 algorithmes d'allocation de ressources qui sont :

1. Part égales (PE) : chaque agent reçoit la même quantité de ressource.
2. Proportionnel au besoin (PB) : chaque agent reçoit une quantité de ressource proportionnelle à son besoin.
3. Max-min équitable (Mm) : initialement, chaque agent reçoit la même quantité de ressource et les surplus sont successivement transférés vers l'agent ayant le plus petit besoin non-satisfait.

4. Maximum d'efficacité (ME) : Les agents sont servis pleinement par besoin croissant.

Nous supposons que les besoins sont exponentiellement distribués (paramètre λ) et que les ressources disponibles ne couvrent qu'une part α de la somme de ces besoins. Sous ces conditions, nous montrons pour ces 4 algorithmes, que la distribution des utilités (qui dans ce cas reflète la performance du système) dépend uniquement de α et non de λ. Dans ce qui suit, nous calculons analytiquement F, la fonction de répartition des utilités pour chaque algorithme. Ceci revient à choisir un agent du système et de calculer la probabilité que son utilité u soit inférieure à un seuil x donné, $x \in [0, 1[$. D'une manière générale nous avons :

$$\forall x \in [0,1[, \quad F(x) = \mathbb{P}(u < x) = \mathbb{P}(\frac{r}{b} < x) = 1 - \mathbb{P}(b < \frac{r}{x}) = e^{-\frac{\lambda r}{x}} \tag{B.1}$$

Pour l'exercice, nous normalisons le nombre d'agent à 1, ainsi le besoin moyen est $\frac{1}{\lambda}$, et la quantité de ressource disponible $\frac{\alpha}{\lambda}$ (λ correspond donc à notre échelle)

Algorithme (PE) : Pour cet algorithme, la valeur de r est la même pour chaque agent. En moyenne chaque agent ayant besoin de $\frac{1}{\lambda}$, nous avons donc $r = \frac{\alpha}{\lambda}$. En injectant cette valeur dans l'équation B.1 nous obtenons

$$\forall x \in [0,1[, \quad F(x) = e^{-\frac{\lambda r}{x}} = e^{-\frac{\alpha}{x}}.$$

Algorithme (PB) : Pour cet algorithme, la valeur de r est proportionnelle au besoin de l'agent. Nous avons donc $r = \alpha \cdot b$. Nous obtenons directement la valeur de $u = \frac{\lambda r}{x} = \alpha$. Nous obtenons directement la fonction de répartition :

$$\forall x \in [0,1[, \quad F(x) = \begin{cases} 0 & \text{si } x < \alpha \\ 1 & \text{sinon.} \end{cases}$$

Algorithme (ME) : Pour cet algorithme, seuls les besoins les plus faibles sont servis. Il existe une valeur β en dessous de laquelle un agent est servi, et au dessus de laquelle il ne l'est pas. La valeur de r est donc définie de la manière suivante :

$$r = \begin{cases} b & \text{si } b < \beta \\ 0 & \text{sinon.} \end{cases}$$

La somme des besoins inférieurs à β étant égale aux ressources disponibles, nous pouvons déterminer la valeur de β en résolvant l'égalité suivante.

$$\int_0^\beta x \cdot e^{-\lambda x} \mathrm{d}x = \frac{\alpha}{\lambda}$$

En intégrant par partie, nous obtenons l'équation :

$$e^{-\lambda\beta} \cdot (\lambda\beta + 1) + \alpha - 1 = 0.$$

Les solutions de cette équation utilisent les fonctions de Lambert, et l'on obtient :

$$\beta = \frac{-W_{-1}(\frac{\alpha-1}{e}) - 1}{\lambda},$$

Une seule solution respecte les contraintes de valeur pour β, et correspond au coefficient -1. Nous avons alors :

$$\mathbb{P}(u = 1) = \mathbb{P}(b < \beta) = 1 - e^{W_{-1}(\frac{\alpha-1}{e})+1}$$

De ce fait la fonction de répartition est constante. Elle est telle que :

$$\forall x \in [0, 1[, \quad F(x) = e^{W_{-1}(\frac{\alpha-1}{e})+1}$$

Algorithme (Mm) : Pour cet algorithme, tous les besoins sont servis au même rythme, et contrairement à l'algorithme (PE), les ressources allouées à un besoin ne dépassent jamais celui-ci. Par construction, il existe une valeur β en dessous de laquelle un agent est servi à hauteur de son besoin, et au dessus de laquelle, il reçoit β. Ceci se traduit par l'égalité suivante :

$$\frac{\frac{\alpha}{\lambda} - \int_0^\beta x\lambda e^{-\lambda x} \mathrm{d}x}{1 - \int_0^\beta \lambda e^{-\lambda x} \mathrm{d}x} = \beta \tag{B.2}$$

La valeur β représente la quantité de ressources attribuées à chaque agent dont le besoin est supérieur à cette valeur. Le numérateur constitue la somme des ressources allouées à l'ensemble de ces agents, tandis que le dénominateur représente leur nombre. En effet, après normalisation, 1 est la totalité des agents, et $\frac{\alpha}{\lambda}$ la totalité des ressources. On peut ainsi déterminer la valeur de β. Le calcul du numérateur s'effectue en intégrant par partie, tandis que le numérateur est une forme classique d'intégrale. On obtient donc l'équation suivante.

$$\frac{\alpha + (\lambda\beta + 1)e^{-\lambda\beta} - 1}{\lambda e^{-\lambda\beta}} = \beta$$

En multipliant par λ cette équation et en factorisant par le membre de gauche, cela donne :

$$\alpha + e^{-\lambda\beta} - 1 = 0 \rightarrow \beta = \frac{-\ln(1-\alpha)}{\lambda}$$

.

En injectant β dans l'équation B.1 nous obtenons la fonction de répartition suivante :

$$\forall x \in [0,1[, \quad F(x) = 1 - \mathbb{P}(b < \frac{\beta}{x}) = e^{-\frac{\ln(1-\alpha)}{x}} = (1-\alpha)^{\frac{1}{x}}$$

Bibliographie

[1] The Computer for the 21st Century. URL http://www.scientificamerican.com/article/the-computer-for-the-21st-century/.

[2] SmartDustBAA97-43-Abstract.pdf. URL http://robotics.eecs.berkeley.edu/~pister/SmartDust/SmartDustBAA97-43-Abstract.pdf.

[3] Manuel Ruiz-Sandoval, Tomonori Nagayama, and BF Spencer Jr. Sensor development using berkeley mote platform. *Journal of Earthquake Engineering*, 10(2) : 289–309, 2006.

[4] G. Chen, M. Fojtik, Daeyeon Kim, D. Fick, Junsun Park, Mingoo Seok, Mao-Ter Chen, Zhiyoong Foo, D. Sylvester, and D. Blaauw. Millimeter-scale nearly perpetual sensor system with stacked battery and solar cells. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 288–289, February 2010. doi : 10.1109/ISSCC.2010.5433921.

[5] Davy Preuveneers, Jan Van den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, Yolande Berbers, Karin Coninx, Viviane Jonckers, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In *Ambient intelligence*, pages 148–159. Springer, 2004.

[6] Carlos Ramos, Juan Carlos Augusto, and Daniel Shapiro. Ambient intelligence—the next step for artificial intelligence. *Intelligent Systems, IEEE*, 23(2) : 15–18, 2008.

[7] Diane J Cook, Juan C Augusto, and Vikramaditya R Jakkula. Ambient intelligence : Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4) :277–298, 2009.

[8] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot) : A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7) :1645–1660, 2013.

[9] *Cisco Visual Networking Index : Global Mobile Data Traffic Forecast Update, 2012-2017*. Cisco, February 2013. Published : White Paper.

[10] *More than 50 billion connected devices*. Cisco, February 2011. Published : White Paper.

[11] Ioe economy. URL http://www.cisco.com/web/about/ac79/docs/innov/IoE_Economy.pdf.

[12] Jacques Bughin, Michael Cui, and James Manyika. Ten it-enabled business trends for the decade ahead. May 2013.

[13] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things : A survey. *Computer networks*, 54(15) :2787–2805, 2010.

[14] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelfflé. Vision and challenges for realising the internet of things. 2010.

[15] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things : Vision, applications and research challenges. *Ad Hoc Networks*, 10(7) :1497–1516, 2012.

[16] URL http://ec.europa.eu/programmes/horizon2020/.

[17] Guillaume Kremer. *Metrology of wireless networks : At the signal and the digital frontier*. Theses, Universite Toulouse III Paul Sabatier, October 2014. URL https://tel.archives-ouvertes.fr/tel-01104964.

[18] ThousandEye. Best practices for saas adoption in the enterprise. Technical report. URL www.thousandeyes.com. Published : White Paper.

[19] S. Ickin, K. Wac, M. Fiedler, L. Janowski, Jin-Hyuk Hong, and A.K. Dey. Factors influencing quality of experience of commonly used mobile applications. *Communications Magazine, IEEE*, 50(4) :48–56, April 2012. ISSN 0163-6804. doi : 10.1109/MCOM.2012.6178833.

[20] The DAG Project. URL http://dag.cs.waikato.ac.nz/.

[21] John W Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo. NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing. In *Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on*, pages 160–161. IEEE, 2007.

[22] Comparison of network monitoring systems, March 2015. URL http://en.wikipedia.org/w/index.php?title=Comparison_of_network_monitoring_systems&oldid=650131421. Page Version ID : 650131421.

[23] CAIDA : Center for Applied Internet Data Analysis. URL http://www.caida.org/home/.

[24] Home | Internet2. URL http://www.internet2.edu/.

[25] RIPE Network Coordination Centre, . URL https://www.ripe.net/.

[26] GlobalNOC. URL https://noc.net.internet2.edu/i2network/research-data.html.

[27] RIPE Atlas - RIPE Network Coordination Centre, . URL https://atlas.ripe.net/.

[28] Home | SamKnows. URL https://www.samknows.com/.

[29] Shafqat Ur Rehman, Thierry Turletti, and Walid Dabbous. Benchmarking in Wireless Networks. Interne, PLANETE - INRIA Sophia Antipolis / INRIA Grenoble Rhône-Alpes, Oct 2010. URL http://hal.inria.fr/inria-00530329.

[30] Clément Burin des Roziers, Guillaume Chelius, Tony Ducrocq, Eric Fleury, Antoine Fraboulet, Antoine Gallais, Nathalie Mitton, Thomas Noél, and Julien Vandaele. Using senslab as a first class scientific tool for large scale wireless sensor network experiments. In Jordi Domingo-Pascual, Pietro Manzoni, Sergio Palazzo, Ana Pont, and Caterina Scoglio, editors, *NETWORKING 2011*, volume 6640 of *Lecture Notes in Computer Science*, pages 147–159. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-20756-3. doi : 10.1007/978-3-642-20757-0_12. URL http://dx.doi.org/10.1007/978-3-642-20757-0_12.

[31] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, 36(SI) :255–270, December 2002. ISSN 0163-5980. doi : 10.1145/844128.844152. URL http://doi.acm.org/10.1145/844128.844152.

[32] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *IN PROCEEDINGS OF THE IEEE WIRELESS COMMUNICATIONS AND NETWORKING CONFERENCE (WCNC*, pages 1664–1669, 2005.

[33] Ehsan Nourbakhsh, Jeff Dix, Paul Johnson, Ryan Burchfield, S. Venkatesan, Neeraj Mittal, and Ravi Prakash. Assert : A wireless networking testbed, 2011.

[34] Kefeng Tan, Daniel Wu, An (Jack) Chan, and Prasant Mohapatra. Comparing simulation tools and experimental testbeds for wireless mesh networks. *Pervasive and Mobile Computing*, 7(4) :434 – 448, 2011. ISSN 1574-1192. doi : 10.1016/j.pmcj.2011.04.004. URL http://www.sciencedirect.com/science/article/pii/S157411921100040X.

[35] S. Lohier, A. Rachedi, E. Livolant, and I. Salhi. Wireless sensor network simulators relevance compared to a real ieee 802.15.4 testbed. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pages 1347–1352, 2011. doi : 10.1109/IWCMC.2011.5982734.

[36] Nathan Eagle and Alex (Sandy) Pentland. Reality Mining : Sensing Complex Social Systems. *Personal Ubiquitous Comput.*, 10(4) :255–268, March 2006. ISSN 1617-4909. doi : 10.1007/s00779-005-0046-3. URL http://dx.doi.org/10.1007/s00779-005-0046-3.

[37] Philippe Owezarski. *HDR : Contribution de la métrologie Internet à l'ingénierie des réseaux*. PhD thesis, September 2006. URL http://www.laas.fr/owe/PUBLIS/owe_HDR.pdf.

[38] Peng Qin, Bin Dai, Benxiong Huang, Guan Xu, and Kui Wu. A survey on network tomography with network coding. *Communications Surveys Tutorials, IEEE*, 16 (4) :1981–1995, Fourthquarter 2014. ISSN 1553-877X. doi : 10.1109/COMST.2014.2320096.

[39] Y. Vardi. Network tomography : Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91(433) :pp. 365–377, 1996. ISSN 01621459. URL http://www.jstor.org/stable/2291416.

[40] Claudia Tebaldi and Mike West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442) :pp. 557–573, 1998. ISSN 01621459. URL http://www.jstor.org/stable/2670105.

[41] B. Yu, J. Cao, D. Davis, and S. Vander Wiel. Time-varying network tomography : router link data. In *Information Theory, 2000. Proceedings. IEEE International Symposium on*, pages 79–, 2000. doi : 10.1109/ISIT.2000.866369.

[42] Matt Sargent, Jakub Czyz, Mark Allman, and Michael Bailey. On The Power and Limitations of Detecting Network Filtering via Passive Observation. In *Passive and Active Measurement Conference*, March 2015.

[43] Jakub Czyz, Mark Allman, Jing Zhang, Scott Iekel-Johnson, Eric Osterweil, and Michael Bailey. Measuring IPv6 Adoption. In *ACM SIGCOMM*, August 2014.

[44] Pedro Casas, Johan Mazel, and Philippe Owezarski. Unsupervised Network Intrusion Detection Systems : Detecting the Unknown without Knowledge. *Comput. Commun.*, 35(7) :772–783, April 2012. ISSN 0140-3664. doi : 10.1016/j.comcom.2012.01.016. URL http://dx.doi.org/10.1016/j.comcom.2012.01.016.

[45] Eric Bloedorn, Alan D. Christiansen, William Hill, Clement Skorupka, Lisa M. Talbot, and Jonathan Tivel. Data mining for network intrusion detection : How to get started. Technical report, The MITRE Corporation, 2001.

[46] Abhishek Mairh, Debabrat Barik, Kanchan Verma, and Debasish Jena. Honeypot in network security : A survey. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, pages 600–605, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0464-1. doi : 10.1145/1947940.1948065. URL http://doi.acm.org/10.1145/1947940.1948065.

[47] Kyle Schomp, Tom Callahan, Michael Rabinovich, and Mark Allman. Assessing dns vulnerability to record injection. In Michalis Faloutsos and Aleksandar Kuzmanovic, editors, *Passive and Active Measurement*, volume 8362 of *Lecture Notes in Computer Science*, pages 214–223. Springer International Publishing, 2014. ISBN 978-3-319-04917-5. doi : 10.1007/978-3-319-04918-2_21. URL http://dx.doi.org/10.1007/978-3-319-04918-2_21.

[48] Zakir Durumeric, James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, and Vern Paxson. The matter of heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, pages 475–488, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3213-2. doi : 10.1145/2663716.2663755. URL http://doi.acm.org/10.1145/2663716.2663755.

[49] Martin Arlitt and Carey Williamson. An Analysis of TCP Reset Behaviour on the Internet. *ACM SIGCOMM Computer Communication Review*, 35 :37–44, 2005.

[50] AT&T. The AT&T Labs Research, 2013. URL http://www.research.att.com/projects.

[51] *IMC '14 : Proceedings of the 2014 Conference on Internet Measurement Conference*, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3213-2.

[52] *IMC '13 : Proceedings of the 2013 Conference on Internet Measurement Conference*, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1953-9. 423133.

[53] *IMC '12 : Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1705-4. 423123.

[54] *IMC '11 : Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1013-0. 523113.

[55] *IMC '10 : Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0483-2. 523103.

[56] *IMC '09 : Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-771-4. 532093.

[57] *IMC '08 : Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-334-1. 532082.

[58] Lora-alliance. URL http://lora-alliance.org/.

[59] Dobroslav Tsonev, Stefan Videv, and Harald Haas. Light fidelity (li-fi) : towards all-optical networking. In *Proc. SPIE*, volume 9007, pages 900702–900702–10, 2013. doi : 10.1117/12.2044649. URL http://dx.doi.org/10.1117/12.2044649.

[60] Vincent Liu, Aaron Parks, Vamsi Talla, Shyamnath Gollakota, David Wetherall, and Joshua R. Smith. Ambient backscatter : Wireless communication out of thin air. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 39–50, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2056-6. doi : 10.1145/2486001.2486015. URL http://doi.acm.org/10.1145/2486001.2486015.

[61] Edwin A. Hernandez, Matthew C. Chidester, and Alan D. George. Adaptive sampling for network management. *J. Netw. Syst. Manage.*, 9(4) :409–434, December 2001. ISSN 1064-7570. doi : 10.1023/A:1012980307500. URL http://dx.doi.org/10.1023/A:1012980307500.

[62] Graham Cormode. Continuous distributed monitoring : A short survey. In *Proceedings of the First International Workshop on Algorithms and Models for Distributed Event Processing*, AlMoDEP '11, pages 1–10, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0922-6. doi : 10.1145/2031792.2031793. URL http://doi.acm.org/10.1145/2031792.2031793.

[63] A.G. Prieto and R. Stadler. A-gap : An adaptive protocol for continuous network monitoring with accuracy objectives. *Network and Service Management, IEEE Transactions on*, 4(1) :2–12, June 2007. ISSN 1932-4537. doi : 10.1109/TNSM.2007.030101.

[64] D. Jurca and R. Stadler. H-gap : estimating histograms of local variables with accuracy objectives for distributed real-time monitoring. *Network and Service Management, IEEE Transactions on*, 7(2) :83–95, June 2010. ISSN 1932-4537. doi : 10.1109/TNSM.2010.06.I8P0292.

[65] Mohammad Sadegh Talebi, Ahmad Khonsari, Amin Mohtasham, and Ali Abbasi. Cost-aware monitoring of network-wide aggregates in wireless sensor networks. *Computer Networks*, 55(6) :1276 – 1290, 2011. ISSN 1389-1286. doi : http://dx.doi.org/10.1016/j.comnet.2010.11.012. URL http://www.sciencedirect.com/science/article/pii/S1389128610003634.

[66] Ozlem Incel, Amitabha Ghosh, Bhaskar Krishnamachari, and Krishna Chintalapudi. Fast data collection in tree-based wireless sensor networks. *IEEE Transactions on Mobile Computing*, 11(1) :86–99, January 2012. ISSN 1536-1233. doi : 10.1109/TMC.2011.22. URL http://dx.doi.org/10.1109/TMC.2011.22.

[67] Nadia Battat, Hamida Seba, and Hamamache Kheddouci. Monitoring in mobile ad hoc networks : A survey. *Computer Networks*, 69(0) :82 – 100, 2014. ISSN 1389-1286. doi : http://dx.doi.org/10.1016/j.comnet.2014.04.013. URL http://www.sciencedirect.com/science/article/pii/S1389128614001662.

[68] Wendi Rabiner Heinzelman, Joanna Kulik, and Hari Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, pages 174–185, New York, NY, USA, 1999. ACM. ISBN 1-58113-142-9. doi : 10.1145/313451.313529. URL http://doi.acm.org/10.1145/313451.313529.

[69] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1) :2–16, February 2003. ISSN 1063-6692. doi : 10.1109/TNET.2002.808417. URL http://dx.doi.org/10.1109/TNET.2002.808417.

[70] Narayanan Sadagopan, Bhaskar Krishnamachari, and Ahmed Helmy. Active query forwarding in sensor networks. *Ad Hoc Networks*, 3(1) :91 – 113, 2005. ISSN 1570-8705. doi : http://dx.doi.org/10.1016/j.adhoc.2003.08.001. URL http://www.sciencedirect.com/science/article/pii/S157087050300074X.

[71] Mohse Gabel, Assaf Schuster, and Daniel Keren. Communication-efficient distributed variance monitoring and outlier detection for multivariate time series. In *Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium*, IPDPS '14, pages 37–47, Washington, DC, USA, 2014. IEEE

Computer Society. ISBN 978-1-4799-3800-1. doi : 10.1109/IPDPS.2014.16. URL http://dx.doi.org/10.1109/IPDPS.2014.16.

[72] Jie Yang, Jun Wang, Maarten Clements, Johan A Pouwelse, Arjen P de Vries, and Marcel Reinders. An epidemic-based p2p recommender system. *LSDS-IR*, page 11, 2007.

[73] Young Myoung Ko and Natarajan Gautam. Epidemic-based information dissemination in wireless mobile sensor networks. *IEEE/ACM Transactions on Networking (TON)*, 18(6) :1738–1751, 2010.

[74] Ayalvadi J Ganesh, A-M Kermarrec, and Laurent Massoulié. Peer-to-peer membership management for gossip-based protocols. *Computers, IEEE Transactions on*, 52(2) :139–149, 2003.

[75] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482–491. IEEE, 2003.

[76] Zygmunt J Haas, Joseph Y Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking (ToN)*, 14(3) :479–491, 2006.

[77] Norbert Tölgyesi and Márk Jelasity. Adaptive peer sampling with newscast. In Henk Sips, Dick Epema, and Hai-Xiang Lin, editors, *Euro-Par 2009 Parallel Processing*, volume 5704 of *Lecture Notes in Computer Science*, pages 523–534. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03868-6. doi : 10.1007/978-3-642-03869-3_50. URL http://dx.doi.org/10.1007/978-3-642-03869-3_50.

[78] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service : Experimental evaluation of unstructured gossip-based implementations. In Hans-Arno Jacobsen, editor, *Middleware 2004*, volume 3231 of *Lecture Notes in Computer Science*, pages 79–98. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-23428-9. doi : 10.1007/978-3-540-30229-2_5. URL http://dx.doi.org/10.1007/978-3-540-30229-2_5.

[79] Patrick Denantes. Performance of Averaging Algorithms in Time-Varying Networks. Technical report, 2007.

[80] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2) :374–382, April 1985.

[81] V. Borkar and P.P. Varaiya. Asymptotic agreement in distributed estimation. *Automatic Control, IEEE Transactions on*, 27(3) :650–655, Jun 1982.

[82] R.O. Saber and R.M. Murray. Consensus protocols for networks of dynamic agents. In *American Control Conference, 2003. Proceedings of the 2003*, volume 2, pages 951–956, June 2003.

[83] V.M. Preciado and George C. Verghese. Synchronization in generalized erdös-rényi networks of nonlinear oscillators. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 4628–4633, Dec 2005.

[84] Chengzhong Xu and Francis C. Lau. *Load Balancing in Parallel Computers : Theory and Practice*. Kluwer Academic Publishers, Norwell, MA, USA, 1997. ISBN 079239819X.

[85] S. Martinez. Distributed representation of spatial fields through an adaptive interpolation scheme. In *American Control Conference, 2007. ACC '07*, pages 2750–2755, July 2007.

[86] R. Olfati-Saber, J.A. Fax, and R.M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1) :215–233, Jan 2007. ISSN 0018-9219. doi : 10.1109/JPROC.2006.887293.

[87] Lin Xiao and Stephen Boyd. Fast Linear Iterations for Distributed Averaging. *Systems and Control Letters*, 53 :65–78, 2003.

[88] Alex Olshevsky and John N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM J. Control Optim.*, 48(1) :33–55, February 2009.

[89] Ludwig Elsner, Rafael Bru, and Michael M. Neumann. Convergence of infinite products of matrices and inner-outer iteration schemes. *Electronic Transactions on Numerical Analysis*, 2 :183–193, 1994. ISSN 1068-9613.

[90] Minghui Zhu and Sonia Martínez. Discrete-time dynamic average consensus. *Automatica*, 46(2) :322 – 329, 2010.

[91] R.A. Freeman, Peng Yang, and K.M. Lynch. Stability and convergence properties of dynamic average consensus estimators. In *Decision and Control, 2006 45th IEEE Conference on*, pages 338–343, Dec 2006. doi : 10.1109/CDC.2006.377078.

[92] M. El Chamie, Ji Liu, and T. Basar. Design and analysis of distributed averaging with quantized communication. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 3860–3865, Dec 2014. doi : 10.1109/CDC.2014.7039988.

[93] Heath J. LeBlanc and Xenofon D. Koutsoukos. Consensus in networked multi-agent systems with adversaries. In *Proceedings of the 14th International Conference on Hybrid Systems : Computation and Control*, HSCC '11, pages 281–290, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0629-4. doi : 10.1145/1967701.1967742. URL http://doi.acm.org/10.1145/1967701.1967742.

[94] Jozef Kenyeres, Jozef Kenyeres, Martin Kenyeres, Markus Rupp, and Peter Farkas. Wsn implementation of the average consensus algorithm. In *Wireless Conference 2011 - Sustainable Wireless Technologies (European Wireless), 11th European*, pages 1–8, April 2011.

[95] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1) :66–77, January 1983. ISSN 0164-0925. doi : 10.1145/357195.357200. URL http://doi.acm.org/10.1145/357195.357200.

[96] K. Avrachenkov, M. El Chamie, and G. Neglia. A local average consensus algorithm for wireless sensor networks. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–6, June 2011. doi : 10.1109/DCOSS.2011.5982199.

[97] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC2582, April 1999.

[98] S. Ha, I. Rhee, and L. Xu. CUBIC : a new TCP-friendly high-speed TCP variant. *Operating Systems Review*, 42(5) :64–74, 2008.

[99] K. Tan, J. Song, Q. Zhang, and M. Sridharan. A Compound TCP approach for high-speed and long distance networks. In *Proceedings of IEEE INFOCOM*, 2006.

[100] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC2475, February 1998.

[101] B. Schroeder and M. Harchol-Balter. Web servers under overload : How scheduling can help. In *ITC 2003*, 2003.

[102] E. Biersack, B. Schroeder, and G. Urvoy-Keller. Scheduling in practice. *SIGMETRICS Perform. Eval. Rev.*, 34(4) :21–28, March 2007. ISSN 0163-5999. doi : 10.1145/1243401.1243407. URL http://doi.acm.org/10.1145/1243401.1243407.

[103] I.M. Verloop, S.C. Borst, and R. Núñez-Queija. Stability of size-based scheduling disciplines in resource-sharing networks. *Performance Evaluation*, 62 :247–262, 2005.

[104] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's initial window. RFC3390, October 2002.

[105] V. Padmanabhan and R. Katz. TCP fast start : A technique for speed-ing up web transfers. In *IEEE GLOBECOM*, November 1998.

[106] K.K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC3168, September 2001.

[107] S. Floyd. TCP and explicit congestion notification. *ACM Computer Communication Review*, 24(5) :10–23, 1994.

[108] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An extension to the selective acknowledgement (sack) option for tcp. RFC2883, July 2000.

[109] N. Dukkipati, M. Mathis, Y. Cheng, and M. Ghobadi. Proportional rate reduction for TCP. In *IMC*, pages 155–170, 2011.

[110] M. Saverio et al. TCP westwood : Bandwidth estimation for enhanced transport over wireless links. In *ACM MOBICOM 2001*, 2001. ISBN 1-58113-422-3. doi : 10.1145/381677.381704. URL http://doi.acm.org/10.1145/381677.381704.

[111] B.Ford et al. Breaking up the transport logjam. In *HOTNETS'08*, 2008.

[112] M. Handley ands O. Bonaventure A. Ford, C. Raiciu. TCP extensions for multipath operation with multiple addresses. RFC6824, January 2014.

[113] W. Noureddine and F. Tobagi. Improving the performance of interactive TCP applications using service differentiation. In *Proceedings of IEEE INFOCOM*, pages 31–40, 2002.

[114] L. Massoulié and J. Roberts. Arguments in favour of admission control for TCP flows. In *Proceedings of ITC-19*, pages 33–44, 1999.

[115] M. Chowdhury, M. Zaharia, J. Ma, M.J. Jordan, and I. Stoica. Managing data transfers in computer clusters with orchestra. *SIGCOMM Comput. Commun. Rev.*, 41(4) :98–109, August 2011. ISSN 0146-4833. doi : 10.1145/2043164.2018448. URL http://doi.acm.org/10.1145/2043164.2018448.

[116] I. Rai, G. Urvoy-Keller, and E. Biersack. LAS scheduling approach to avoid bandwidth hogging in heterogeneous TCP networks. In *Proceedings of IEEE HSNMC*, 2004.

[117] K.E. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation between short and long TCP flows : Predictability of the response time. In *Proceedings of INFOCOM*, 2004.

[118] L.E. Schrage. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16(3) :687–690, 1968.

[119] R. Righter and J.G. Shanthikumar. Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. *Prob. Eng. Inf. Sci.*, 3 :323–334, 1989.

[120] G. Weiss. A tutorial in stochastic scheduling. In *Scheduling Theory and Its Applications*. Wiley, 1995.

[121] M. Allman and V. Paxson. TCP congestion control. RFC5681, 2009.

[122] E. Friedman and S. Henderson. Fairness and efficiency in processor sharing protocols to minimize sojourn times. *Proceedings of ACM SIGMETRICS*, pages 229–337, 2003.

[123] M. Crovella, M. Taqqu, and A. Bestavros. A practical guide to heavy tails. chapter Heavy-tailed Probability Distributions in the World Wide Web, pages 3–25. Birkhauser Boston Inc., Cambridge, MA, USA, 1998. ISBN 0-8176-3951-9. URL http://dl.acm.org/citation.cfm?id=292595.292596.

[124] S. Aalto and U. Ayesta. SRPT applied to bandwidth-sharing networks. *Annals of Operations Research*, 170 :3–19, 2009.

[125] The caida ucsd anonymized internet traces 2014, equinix-chicago, dira, 03-20-2014. "http ://www.caida.org/data/passive/passive_2014_dataset.xml".

[126] P. Ravis, J. Manish, and C. Dovrolis. Socket buffer auto-sizing for high-performance data transfers. *J. of Grid Computing*, 1 :361–376, 2003. ISSN 1570-7873. doi : 10.1023/B:GRID.0000037554.67413.52. URL http://dx.doi.org/10.1023/B%3AGRID.0000037554.67413.52.

[127] The web site of the cooperative association for internet data analysis (caida). http ://www.caida.org/.

[128] C. Monma and Diane Sheng. Backbone network design and performance analysis : A methodology for packet switching networks. *Selected Areas in Communications, IEEE Journal on*, 4(6) :946–965, Sep 1986. ISSN 0733-8716. doi : 10.1109/JSAC.1986.1146400.

[129] C.-F. Chiasserini and M. Garetto. Modeling the performance of wireless sensor networks. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages –231, March 2004. doi : 10.1109/INFCOM.2004.1354496.

[130] John Von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944. ISBN 0691119937. URL http://jmvidal.cse.sc.edu/library/neumann44a.pdf.

[131] Amos Tversky and Daniel Kahneman. Advances in prospect theory : Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, 5(4) :297–323, 1992. ISSN 0895-5646. doi : 10.1007/BF00122574. URL http://dx.doi.org/10.1007/BF00122574.

[132] Tarn Duong and Martin Hazelton. Plug-in bandwidth matrices for bivariate kernel density estimation. *Journal of Nonparametric Statistics*, 15(1) :17–30, 2003. doi : 10.1080/10485250306039. URL http://dx.doi.org/10.1080/10485250306039.

[133] David W. Scott. *Multivariate density estimation : theory, practice, and visualization*. Wiley series in probability and mathematical statistics. Wiley, 1992.

[134] R.D. Nowak. Distributed em algorithms for density estimation and clustering in sensor networks. *Signal Processing, IEEE Transactions on*, 51(8) :2245–2253, Aug 2003.

[135] Hongbo Jiang and Shudong Jin. Scalable and robust aggregation techniques for extracting statistical information in sensor networks, 2006.

[136] Wojtek Kowalczyk and Nikos Vlassis. Newscast em. In *In NIPS 17*, pages 713–720. MIT Press, 2005.

[137] Nikos Vlassis, Yiannis Sfakianakis, and Wojtek Kowalczyk. Gossip-based greedy gaussian mixture learning. In *10th Panhellenic Conf. on Informatics*, 2005.

[138] Yusuo Hu, Jian-Guang Lou, Hua Chen, and Jiang Li. Distributed density estimation using non-parametric statistics. In *Distributed Computing Systems, 2007. ICDCS '07. 27th International Conference on*, pages 28–28, June 2007.

[139] ThomasG. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67704-8. doi : 10.1007/3-540-45014-9_1. URL http://dx.doi.org/10.1007/3-540-45014-9_1.