RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Doctorat de l'Université de Toulouse

préparé à l'ENAC

Apport de l'IA pour la caractérisation réseau et la gestion de la ressource SATCOM

Thèse présentée et soutenue, le 5 mars 2026 par

Paul GRISLAIN

École doctorale
EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse

Spécialité
Informatique et Télécommunications

Unité de recherche
ENAC-LAB - Laboratoire de Recherche ENAC

Thèse dirigée par
Emmanuel LOCHIN et Jean-Yves TOURNERET

Composition du jury
M. Pham CONGDUC, Président, Université de Pau et des Pays de l'Adour
M. Marcelo DIAS DE AMORIM, Rapporteur, Sorbonne Université
M. Jérôme LACAN, Examinateur, ISAE-SUPAERO
M. Emmanuel LOCHIN, Directeur de thèse, ENAC

Membres invités
M. Jean-Yves TOURNERET, INP Toulouse

Remerciements

Cette thèse est issue d'une réflexion personnelle, mais qui n'aurait pas pu aboutir sans le soutien d'Emmanuel Lochin et Jean-Yves Tourneret. Mes remerciements vont avant tout à eux, car ils ont largement contribué aux réflexions développées ici. Bien plus que cela, durant ces trois ans de montagnes russes, ils ont su poser les bonnes questions pour calmer l'euphorie (surtout en début de thèse) et me motiver à avancer dans les moments creux. Je remercie aussi les nombreuses personnes qui ont aussi contribué à cette thèse par des remarques judicieuses, des questions pertinentes, de bons conseils ou leur amitié. Jean-Philippe Condomines m'avait proposé des pistes intéressantes, ainsi que Gentian Jakllari et Dino Lopez, membres du CSI. Cédric Baudoin et Nicolas Kuhn de Thalès ont donné à cette thèse le cadre des réseaux satellites et m'ont permis de mieux connaître les mégaconstellations. Du laboratoire ENAC, je remercie plus particulièrement Alain Pirovano et mes collègues forts sympathiques Jiayi, Valencia, Mehdi, Hakim, Gabriel, Audric, Anne-Laure. Pendant ces années, le laboratoire TéSA a toujours été un lieu où il fait bon vivre, agréable et propice à la réflexion. J'en remercie Corinne, la directrice, ainsi que tous les membres : Isabelle, Raoul, Philippe, Jacques et Patrice. J'y ai rencontré des gens fantastiques : Kareth, Hamish, Kin, Victor, Corentin, Yousra, Evelyne, Gaston, Valérian, Marta, Joan, Esteban, Younes, Ayoub, Maurine, Geoffroy, Nicolas et d'autres encore ...

Au cours de cette thèse, j'ai été porté par de bons amis rencontrés à Toulouse et à l'ENAC. Plus spécialement Jihanne qui a aussi effectué sa thèse à TéSA et Robin qui y poursuit la sienne. Aussi, ma présentation orale n'aurait jamais été telle sans l'entraînement énergique de Gabriel (actuellement doctorant en histoire). Je leur souhaite le meilleur ainsi qu'à ma famille qui m'a constamment encouragé.

Enfin, Oana Hotescu, Jérôme Lacan et Emmanuel Lochin m'ont donné l'opportunité de découvrir ce monde de la recherche par un projet en fin d'école d'ingénieurs. À travers eux, je remercie tous les enseignants qui, à tour de rôle, m'ont fait progresser dans ce parcours académique en attisant ma curiosité pour les sciences. L'intérêt en valait largement la peine.

Table des matières

Liste des contributions

Publications pendant la thèse

Paul Grislain, Emmanuel Lochin, and Jean-Yves Tourneret. Partitionnement de graphe pour l'identification de goulots d'étranglement partagés. In XXXe Colloque GRETSI-Traitement du Signal et des Images, 2025.

Paul Grislain, Emmanuel Lochin, Jean-Yves Tourneret, and Nicolas Kuhn. Procédé de cartographie d'un réseau à file d'attente IP, basé sur l'analyse passive en un point d'un réseau, avril 2025.

Publications avant thèse

Paul Grislain, Alexia Auddino, Anna Barraqué, François Lamothe, Oana Hotescu, Jérôme Lacan, José Radzik, and Emmanuel Lochin. Rethinking leo constellations routing. International Journal of Satellite Communications and Networking, 2024.

Paul Grislain, Nicolas Pelissier, François Lamothe, Oana Hotescu, Jérôme Lacan, Emmanuel Lochin, and José Radzik. Rethinking leo constellations routing with the unsplittable multi-commodity flows problem. In 2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC), pages 1–8. IEEE, 2022.

Chapitre 1

Introduction

Une thèse connaît de nombreux hauts et bas. Cependant un déclic s'est produit au début de la seconde année et a donné un tour particulier à cette thèse. L'élément déclencheur est la lecture, ou plus exactement l'interprétation d'un passage d'une œuvre d'un philosophe du Moyen-Âge qui repose actuellement aux Jacobins à Toulouse. Le philosophe Saint Thomas d'Aquin n'a pas seulement laissé ses initiales au blason du Stade Toulousain, il a aussi écrit la Somme Théologique, imposant traité de doctrine catholique sur Dieu. Mais à quoi sert-il d'en parler si Dieu n'existe pas ? Il commence donc par une preuve de l'existence de Dieu. En résumé, son raisonnement comporte deux étapes : constater la présence d'un ordre intelligible dans la nature, et y voir l'œuvre d'un être supérieur intelligent (Dieu) qui en est à l'origine. Ce principe s'applique directement au problème que nous allons aborder dans cette thèse : **l'observation d'un ordre dans le flux de paquets permet-il d'inférer l'existence d'un système organisé (i.e., une file d'attente) à l'origine de cette régularité ?** À l'inverse, l'absence d'ordre dans le flux observé exclut la présence d'une file d'attente, selon la définition que nous en proposons. Nous verrons que cette étape est essentielle pour la **caractérisation et la gestion des ressources des mégaconstellations** qui est l'objectif ultime de ce travail. Ce manuscrit développe cette intuition pour en faire un outil d'inférence topologique, posant ainsi les bases d'une solution au problème étudié

1.1 Pourquoi étudier ce problème et pourquoi au sein de mégaconstellations ?

Depuis quelques années les réseaux internet connaissent une belle effervescence avec le déploiement de mégaconstellations. Une mégaconstellation est avant tout un déploie-

ment massif de satellites de télécommunications en orbite basse. Actuellement, il y a deux mégaconstellations en service : OneWeb, qui compte une centaine de satellites, et Starlink, composé de près de 6000 satellites.

Figure 1.1 : Illustration de mise en orbite de satellites Starlink

1.1.1 Le défi de la gestion des mégaconstellations

Ces constellations permettent un accès ubiquitaire d'internet : *Internet access for everyone, everywhere*[1]. Une constellation de satellite peut couvrir toute la surface terrestre parce que son déploiement n'a pas les contraintes d'une infrastructure au sol et que chaque satellite peut couvrir une région beaucoup plus grande qu'une station sol. Les mégaconstellations offrent donc un moyen de communiquer à échelle mondiale, couvrant les territoires isolés jusqu'à présent. Il y a deux catégories de mégaconstellations, caractérisées par des déploiements massifs. Les "petites", comme les projets IRIS2 ou Telesat limités à 300 satellites, pourront adopter une gestion centralisée. Les "grosses" sont composées de milliers de satellites, devront opter pour une gestion distribuée, tel que c'est le cas pour internet à échelle mondiale. Ces derniers chiffres sont impressionnants comparés à ce qui existe jusqu'à présent dans le monde spatial. Actuellement, le projet de Starlink est de placer 14000 satellites en orbites, tandis que Amazon Leo (anciennement Kuiper) en prévoit 18000. Mais il est important de mettre ces chiffres en relation avec le sol : actuellement Starlink dessert 7 millions de personnes dans le monde entier avec 6000 satellites. En France métropolitaine, nous avons près de 63000 stations 4G (auxquelles il faudrait ajouter les stations 5G) pour 70 millions d'habi-

[1] www.internetsociety.org

tants. Ce déploiement, important en comparaison des précédents, est en fait minime comparé au déploiement terrestre.

1.2 Le problème de la congestion

Les mégaconstellations sont une avancée majeure dans le déploiement d'internet, rendue possible par de nombreuses innovations pour l'accès à l'espace et les télécommunications. Comme représenté figure 1.2, une mégaconstellation fait intervenir trois acteurs principaux : les utilisateurs, les satellites, et le segment sol constitué de stations relais et de centres de contrôle.

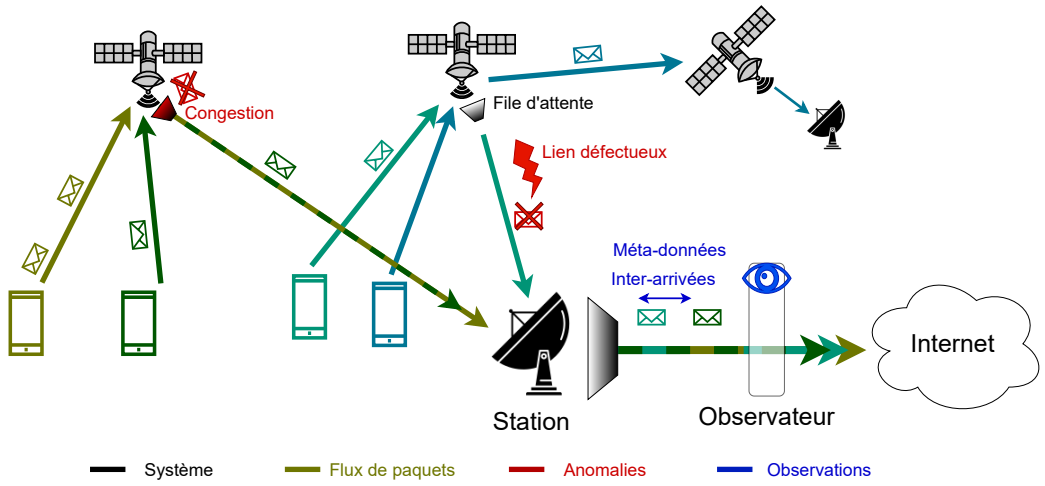


FIGURE 1.2 : Schéma d'accès à internet par satellite

Comme dans les réseaux terrestres, différents dysfonctionnements peuvent impacter ce réseau. Les auteurs de [MTGB22] rapportent notamment de problèmes de congestion et d'erreurs lien. Le gestionnaire du réseau doit donc repérer les anomalies pour les corriger.

Le gestionnaire du réseau devrait en toute logique repérer ces anomalies pour les corriger, cependant cette tâche se complexifie grandement et devient combinatoire plus la dimension de la constellation augmente. De plus, il semble évident que les mégaconstellations ne pourront être gérées uniquement par un seul et même gestionnaire de réseau ce qui complexifie de nouveau un partage et une allocation de ressources cohérentes. Pour preuve, devant l'apparition de phénomène de congestion incontrôlable, Starlink a fait le choix d'appliquer un principe de *network pricing* : la *congestion*

charge[2] qui est une solution de régulation de la congestion (jusqu'à un certain point) basée sur une pénalisation financière.

Le défis de gestion et de supervision d'un tel réseau sont multipliés par le nombre de satellites routeurs et de stations sol, et la grande distance qui les sépare. De plus, la mobilité des satellites par rapport à la Terre impose de renouveler sans cesse les connexions. Pour toutes ces raisons, une approche centralisée semble impossible à mettre en œuvre sans générer une latence qui pénaliserait l'ensemble du réseau.

Des choix forts quant à la supervision, la gestion des ressources et de partage des tâches entre stations sol et satellites sont encore débattus, mais dans tous les cas l'approche pour gérer un tel réseau devra être décentralisée, comme dans le système d'internet actuel.

1.2.1 Une solution : inférer la topologie pour superviser passivement l'écoulement du réseau

Dans l'hypothèse où un contrôle centralisé ne soit pas envisageable, ou juste à des fins de *reverse ingeniering* ou de supervision pour détecter des anomalies, une mesure passive est souvent une solution simple et non invasive à mettre en œuvre. C'est ce que nous proposons dans cette thèse. Un algorithme de ce type permettrait à n'importe quel gestionnaire, partenaire ou un simple utilisateur de pouvoir sonder l'état du réseau à distance via un outil de mesure de performance et de le cartographier. Dans le cas du gestionnaire réseau un tel outil de mesure permettrait de corroborer des observations d'écoulement de trafic avec un système de gestion basé sur des protocoles de supervision standards via des outils basés sur le protocole SNMP [CMPS02] est disponible. Enfin, bien que développé dans un contexte satellitaire, cet outil métrologique trouve son application dans n'importe quel type de réseau filaire ou sans-fil.

1.3 Présentation de la démarche de la thèse

Le sujet de la thèse contient deux volets : d'une part la caractérisation du réseau, c'est-à-dire s'informer sur sa situation et détecter des anomalies de fonctionnement ; d'autre part proposer des solutions pour améliorer la gestion des ressources.

Cette thèse s'est initialement focalisée sur le développement de méthodes passives pour déceler des anomalies de fonctionnement de façon passive et avec une contrainte de trafic utilisateur chiffré. Concrètement, l'idée était de déceler des anomalies dans le réseau à partir de l'étude de séries temporelles d'arrivées de paquets IP. Nous avons

[2] https://dishycentral.com/starlink-congestion-charge

donc mis en place des techniques de *machine* ou *deep learning*, mais les résultats peu probants nous ont incité à changer d'approche. Nous nous sommes donc posé la question suivantes : quelles informations pouvons-nous extraire de la série temporelle des paquets ? Les 2ème et 3ème années de cette thèse ont donc été dédiées à la mise au point d'une méthode pour mettre en évidence la présence de portions de chemin communes à des ensembles de flux de paquets. En parallèle, cette méthode caractérise le débit des nœuds, ce qui ouvre la voie pour estimer un taux d'utilisation des routeurs, détecter la congestion, et finalement collecter des informations utiles à la gestion du réseau.

Nous montrons dans ce manuscrit qu'il est possible (sous certaines conditions) de recueillir de façon passive des informations de topologie du réseau, débit des liens, et potentiellement saturation. Ces informations pourront ensuite être utilisées pour détecter des anomalies et finalement optimiser la gestion des ressources du réseau.

Le manuscrit est organisé en 4 chapitres principaux :

- Le premier est un chapitre d'état de l'art sur le fonctionnement d'une constellation de satellite et la détection de congestion dans les réseaux internet. Cette partie débute par la description du mouvement des satellites, des aspects de conception d'une mégaconstellation et enfin une comparaison de performance entre les différents types de réseau. Cela nous amène au problème de la congestion, et nous abordons plus en détail ses causes et les techniques mise en œuvre pour y remédier. Dès lors, la seconde partie de l'état de l'art présente des méthodes pour la détecter. Beaucoup de méthodes basées sur l'analyse du trafic utilisateur tombent en désuétude à cause du chiffrement total des protocoles. Pour cette raison, nous cherchons d'autres pistes, d'abord dans les méthodes actives par émission de trafic sonde, puis dans les méthodes d'analyse d'agrégats de flux réseaux. C'est cette approche par l'étude des agrégats qui est approfondie dans la thèse.

- Le deuxième chapitre a pour objectif la réalisation d'un jeu de données pour tester les méthodes de détection d'anomalies, et plus particulièrement de congestion. Trois modèles sont proposés. Le premier est un modèle mathématique des files d'attentes. Les hypothèses simplificatrices nécessaires pour modéliser un réseau de façon analytique ne permettent pas d'aboutir à une solution satisfaisante, et nous avons eu recours à un simulateur réseau. La simulation permet de modéliser le système beaucoup plus finement et d'obtenir des résultats très précis. La principale difficulté rencontrée est un problème de lenteur de simulation. En remarquant que le système peut être simplifié au voisinage d'une station sol, le

nombre de systèmes à simuler est largement réduit et rend l'émulation très accessible. Finalement, nous utilisons un émulateur simplifié qui a fourni rapidement des données en quantité suffisante.

- Le troisième chapitre contient la proposition d'inférence de topologie. Tout commence par l'élaboration d'une méthode de détection de présence d'une file d'attente. Nous comparons cette méthode avec les résultats de l'état de l'art. Un algorithme complet est ensuite présenté pour identifier la situation de différents réseaux émulés. Certains modèles sont tirés de la littérature.

- Le dernier chapitre aborde plus en détail le problème fondamental de partitionnement à résoudre dans le cadre de l'inférence de topologie. Ce problème complexe est ramené à un problème de partitionnement de graphe, mieux couvert dans la littérature. Un court état de l'art justifie le choix de la méthode utilisée. Le second volet de ce chapitre tente d'apporter une solution quant à la fiabilité des résultats obtenus. Nous présentons différents modèles de la littérature et en proposons un nouveau, plus adapté à notre problème. Nous montrons comment ce dernier modèle peut être utilisé pour quantifier la fiabilité d'une solution de partitionnement, donc de topologie.

- Enfin, nous terminons par un retour sur l'algorithme utilisé et ses points d'amélioration. Cette thèse présente donc une méthode de connaissance du réseau, qui ouvre de nouvelles perspectives en détection d'anomalies et gestion des ressources.

Chapitre 2

État de l'art

Ces dernières années ont montré une petite révolution dans le domaine des télécommunications spatiales avec l'arrivée des mégaconstellations. Jusqu'à récemment, les communications spatiales étaient l'affaire de quelques satellites massifs en orbite géostationnaire (GEO). De nouveaux acteurs spatiaux ont contribué à réduire le coût de l'accès à l'espace en améliorant les processus et en passant à une logique plus industrielle. La masse en orbite a considérablement augmenté au cours de la dernière décennie et de très nombreux satellites de communications ont été lancés en orbite terrestre basse *Low Earth Orbit* (LEO), formant ainsi des mégaconstellations qui concurrencent les satellites géostationnaires.

Dans cet état de l'art, nous étudions d'abord ce premier changement majeur qui renouvelle les communications spatiales et les réseaux en général. Nous présentons succinctement les raisons du déploiement de ces mégaconstellations, quelques aspects importants de leur conception et enfin les défis auxquelles elles sont actuellement confrontées. Nous verrons qu'à l'instar des réseaux terrestres, elles restent sujettes à la congestion.

Nous abordons ensuite un second changement profond des réseaux qui concerne leur fonctionnement. Le paradigme collaboratif des premiers temps d'internet est abandonné au profit de la sécurité. Le chiffrement des protocoles force donc à repenser la gestion des réseaux. Cette seconde partie de l'état de l'art traite donc des métriques utilisées pour décrire l'état d'un réseau et des outils qui permettent de les estimer.

2.1 Les mégaconstellations face au problème de la congestion

Une mégaconstellation est un nouveau type de réseau de télécommunications. Nous expliquons dans un premier temps ce qu'est une mégaconstellation, et pourquoi elles sont déployées. Puis nous abordons plus spécifiquement le problème de la congestion. Ce problème touche tous les réseaux et diverses méthodes ont été proposées pour la contenir, mais ne semblent pas le résoudre totalement. La taille et la mobilité d'une mégaconstellation apportent des difficultés supplémentaires qui rendent le rendent plus complexe à gérer.

2.1.1 Description des mégaconstellations et quelques raisons du succès de leur déploiement

Une mégaconstellation est un déploiement massif de satellites de télécommunications LEO. Elle permet de communiquer depuis n'importe où sur la Terre, tout en maintenant des performances proches des systèmes terrestres en termes de latence et débit. Après un rappel des limites des systèmes préexistants, nous présentons succinctement leur fonctionnementrencontrées et concluons par une étude comparative de leurs performances et de leurs défis actuels.

2.1.1.1 Les limites des systèmes préexistants

L'arrivée des mégaconstellations s'explique notamment par les contraintes des systèmes préexistants. Les satellites ont très tôt été utilisés dans les télécommunications pour pallier la portée limitée des réseaux terrestres. Des moyens de télécommunication par satellite GEO ont été déployés, mais leur performance est bridée par la distance des

satellites GEO. Les mégaconstellations LEO sont un compromis intéressant qui offre des solutions à ces deux problèmes.

Une couverture mondiale Comparé à un réseau composé d'infrastructures au sol, les satellites procurent les avantages suivants :

- couverture étendue ;
- indépendance des infrastructures au sol.

Le principal avantage des communications par satellite est la portée d'un satellite. Un satellite peut couvrir une zone plus grande que n'importe quelle station sol (sauf à très basse fréquence, mais les débits très faibles font que ce n'est pas comparable). Ainsi, il suffit de 3 satellites géostationnaires pour couvrir le monde, pôles exceptés. Les satellites ont rendu possibles les télécommunications à l'échelle du globe et répandu l'usage de la télévision. Les satellites à haute altitude sont donc très adaptés aux applications de *broadcast*.

Le second avantage des satellites est leur indépendance aux événements terrestres. Les communications par satellites restent fonctionnelles en cas de catastrophe naturelle, de guerre ayant entraîné la destruction des infrastructures au sol, et dans les opérations militaires ou scientifiques en environnement hostile. Dans un contexte plus large, ils couvrent les territoires peu peuplés où le maintien d'équipements au sol est difficile ou peu rentable, mers et zones à faible densité de population. Les communications par satellite sont efficaces si le ciel est dégagé d'obstacles et simplifient les communications dans différents moyens de transports, trains, bateaux et avions.

La réduction de latence Les raisons évoquées ci-dessus ne suffisent pas à expliquer l'essor des mégaconstellations par rapport aux satellites géostationnaires. En fait, la diminution de la distance sol-satellite s'accompagne de deux avantages cruciaux :

1. réduction de la latence ;
2. amélioration du bilan de liaison.

Un satellite géostationnaire est situé à près de 36000 km à la verticale de l'équateur. Ainsi, une onde radio émise par un utilisateur au sol atteindra le satellite au bout de 120 ms, légèrement plus à nos latitudes. Dans le cas où l'utilisateur effectue une requête vers un serveur sur Terre à travers le satellite, le signal devra parcourir 4 fois cette distance Terre-Satellite, donc le temps aller-retour théorique minimum est de l'ordre de 480ms. Cela convient pour beaucoup d'usages d'internet, est gênant pour des applications

temps-réel comme la téléphonie, mais est totalement inacceptable dans le cas de jeux vidéo. Les protocoles de transport d'internet, Transmission Control Protocol (TCP) en premier lieu, subissent aussi une baisse de performance à cause de cette latence qui rend les utilisateurs moins réactifs et rend moins optimale l'utilisation de la bande passante.

Enfin, le bilan de liaison est très lourd sur une telle distance car la puissance d'un signal radio s'atténue en fonction du carré de la distance. En passant à des distances plus courtes, il devient possible d'utiliser des équipements moins puissants, des antennes moins directives ou de qualité moindre. Les dernières avancées montrent même qu'un téléphone standard peut communiquer directement avec un satellite LEO. La simplification du bilan de liaison permet donc de démocratiser l'accès aux télécommunications par satellite.

En conclusion, le déploiement de mégaconstellations de satellites LEO répond à un besoin de notre société qui a besoin de moyens de télécommunication plus performants. L'arrivée des mégaconstellations est possible grâce aux avancées industrielles (réduction des coûts unitaires par le nombre de lancements, la réutilisation des lanceurs la fabrication en chaîne des satellites) et la résolution de problèmes techniques. Contrairement aux satellites LEO, les satellites géostationnaires sont fixes par rapport au sol. Pour garantir une meilleure communication, l'utilisateur doit orienter son antenne et viser un satellite en mouvement. D'autre part, cela pose des défis de routage que nous ne développons pas dans cette thèse.

2.1.1.2 Qu'est-ce qu'une mégaconstellation ?

Nous décrivons maintenant succinctement le fonctionnement d'une mégaconstellation.

Les satellites Commençons par aborder le mouvement d'un satellite avant de décrire leur équipement de communication.

La mécanique spatiale étudie le mouvement des corps célestes et décrit toutes les interactions entre le satellite et son environnement. Nous pouvons considérer ici le simple problème à deux corps : le satellite est en orbite autour d'une Terre ronde, et ne subit que son attraction. La trajectoire du satellite est alors parfaitement déterminée. Elle peut être complètement identifiée en connaissant sa position et son vecteur vitesse à un instant donné. D'après les lois de Képler, cette trajectoire est elliptique et peut être décrite par les paramètres orbitaux :

- excentricité de l'orbite, qui est souvent nulle dans le cas de satellites en orbite basse.

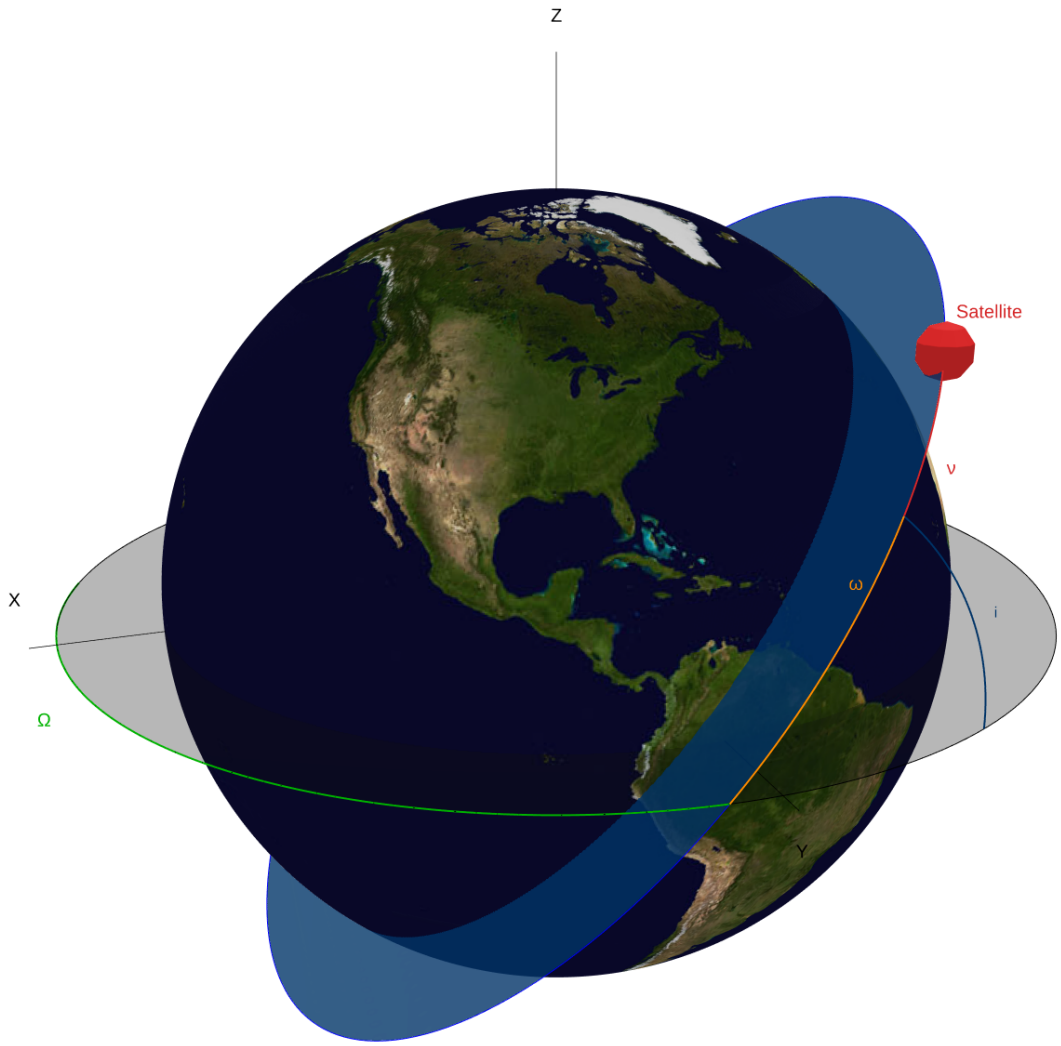


FIGURE 2.1 : La trajectoire d'un satellite en orbite basse est décrite par les éléments orbitaux (seuls les angles sont illustrés ici).

- demi grand-axe, qui est ici le rayon de l'orbite circulaire du satellite
- 4 angles (Ω, i, ω, ν) qui permettent d'orienter l'orbite dans un repère géocentrique. Voir figure 2.1

Un satellite en orbite basse circulaire se déplace à vitesse constante, en fonction de son altitude. Typiquement, un satellite à 600 km d'altitude effectue une révolution en près de 96 minutes. Plus le satellite est bas, plus sa vitesse de rotation est élevée. Le mouvement d'un satellite en orbite basse provoque de nombreuses difficultés techniques à résoudre pour assurer les télécommunications à travers la mégaconstellation. Sur le

plan physique, le déplacement du satellite et ses variations de vitesse relative par rapport à son correspondant sur Terre affectent le signal. Cet effet Doppler modifie la fréquence du signal et doit être (pré-)compensé en émission ou réception. Cela engendre de nombreuses difficultés dans le partage des bandes de fréquence. Aussi, un satellite en orbite basse traverse le cône de visibilité d'un utilisateur en une dizaine de minutes. Un utilisateur devra donc fréquemment changer de satellite. Ce problème s'applique de la même façon aux stations sol. Cependant, les satellites *Medium Earth Orbit* (MEO) et LEO ne sont pas tous toujours à portée d'une station sol. Pour maintenir la connexion, les satellites hors portée peuvent être équipés de moyens de communiquer entre eux sans repasser par une station sol. Par exemple, le projet Telesat spécifiait 4 têtes optiques par satellite pour des communications laser avec 4 autres satellites : 2 pour les communications avec des satellites sur la même orbite, 2 avec des satellites sur des orbites adjacentes.

Les difficultés techniques sont donc résolues par des choix techniques et de dimensionnement de la constellation. Allons plus loin en ce sens. L'orbite d'un satellite est définie dans le repère géocentrique (O, X, Y, Z), dont l'origine est attachée au centre de la Terre et dont les axes pointent vers des étoiles lointaines supposées fixes. La Terre tourne autour de l'axe des pôles, tandis que le satellite tourne sur son orbite, considérée fixe. Ces deux rotations font balayer aux satellites de larges régions du monde. Les zones survolées par un satellite dépendent donc de l'inclinaison de l'orbite. Lorsque l'inclinaison est proche de 90°, le satellite survole potentiellement toute la Terre. À l'inverse, si elle vaut 0°, le satellite survole l'équateur et ne passera jamais à la verticale des régions à latitude élevée.

Comme expliqué précédemment, la surface éclairée au sol d'un satellite dépend de son altitude. Un satellite a au moins une antenne orientée vers le sol pour communiquer avec les utilisateurs et stations. En plus de l'altitude, la directivité de l'antenne, son gain, et son orientation permettront d'assurer une bonne communication. Réciproquement, l'utilisateur au sol ne peut communiquer qu'avec les satellites disponibles dans son champ de vision. Starlink indique qu'un satellite est disponible à partir de 15° d'élévation (le guide de configuration indique que l'antenne a un champ de vision de 110° et peut être inclinée de 20°).

Dimensionnement d'une mégaconstellation Nous entrons ainsi dans la conception d'une mégaconstellation, résultat d'un compromis entre performance et complexité de la flotte de satellites. Nous avons vu que les performances diminuent avec la distance et l'accroissement de l'altitude des satellites, mais cela engendre d'autres difficultés qui rendent les communications plus contraignantes.

Le premier paramètre à définir est l'altitude de la constellation, choisi en fonction du bilan de liaison souhaité. Le second paramètre est l'inclinaison, selon le choix des opérateurs de couvrir les pôles ou non. Le choix d'orbites polaires permet d'assurer une couverture globale, mais la constellation perd en efficacité car la densité de satellites est élevée près des pôles, alors que la majorité de la population, donc des clients potentiels, vit à des latitudes plus faibles. Pour résoudre ce compromis, certains opérateurs ont recours à plusieurs orbites à différentes altitudes. Par exemple des orbites polaires assurent une couverture mondiale, tandis que des orbites moins inclinées, plus basses et une densité de satellites plus élevée fournissent la connexion aux utilisateurs situés en majorité à des latitudes plus basses.

Le service sera garanti à condition qu'un utilisateur soit toujours à portée d'au moins un satellite, plus pour assurer les transferts (*handovers*) et une certaine robustesse. Cette dernière contrainte de conception de la constellation impose un nombre de plans et de satellites par orbite. Ces contraintes jouent aussi sur la création de liens inter-satellites afin de relayer les paquets qui circulent dans la constellation sans retour sol.

Le segment sol joue une part primordiale dans la performance de la constellation. Le réseau ne sera compétitif que si les données sont acheminées rapidement vers les serveurs terrestres, d'où la nécessité de positionner de nombreuses stations sol pour faire rapidement redescendre les flux de données. La multiplicité des stations sol est aussi un moyen de réduire l'engorgement et augmenter la capacité du réseau.

La continuité et la qualité du service découlent donc de tous ces paramètres : performance individuelle des satellites, leur nombre et répartition sans oublier le segment sol.

2.1.1.3 Les défis actuels des mégaconstellations : concurrence externe et difficultés internes de gestion

Nous abordons maintenant des aspects de performance des mégaconstellations. Nous verrons que leur arrivée bouleverse le monde des télécommunications, stimule la concurrence et modifiera à terme les habitudes de certains utilisateurs.

Comparaison des mégaconstellations LEO avec les systèmes GEO et la téléphonie mobile Actuellement le plus gros opérateur de constellations LEO est Starlink. Ce dernier a lancé une nouvelle course à l'espace et nous prendrons ses chiffres comme référence. Autour de lui la concurrence peine à s'organiser : OneWeb ne fournit pas de service aux particuliers, Iridium NEXT est beaucoup plus petite. Les autres

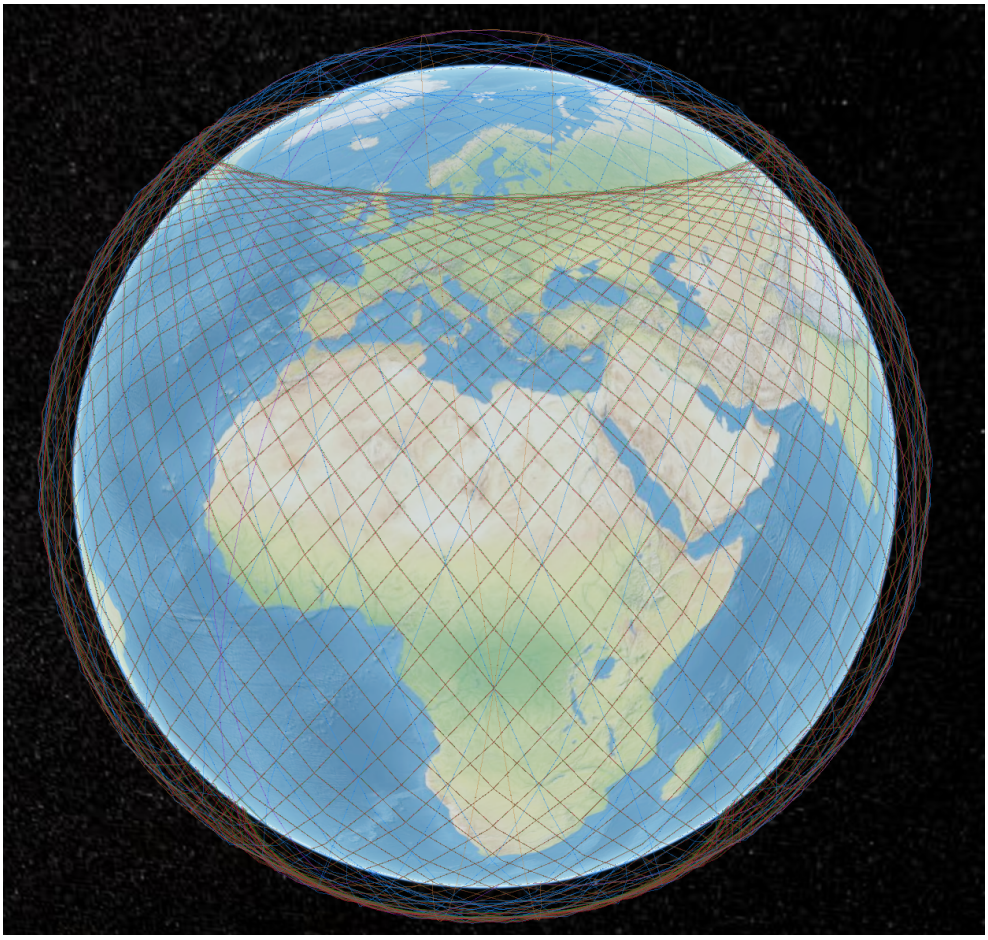


Figure 2.2 : Le projet Starlink Gen 1, en grande partie déployé, est composé de 5 couches situées entre 540 et 570km d'altitude. Les couches à haute inclinaison assurent la couverture des pôles, tandis que les couches à inclinaison réduite, plus dense à latitude plus faible, assurent une meilleure qualité de service aux utilisateurs. Image générée avec CesiumJS

(Amazon Leo, Guowang ...) ont à peine commencé le déploiement et sont à l'état de projet.

Les mégaconstellations entrent en concurrence avec les solutions par satellite géostationnaire pour des aspects de performance. Les opérateurs subissent donc la concurrence des mégaconstellations qui proposent des débits plus élevés et une latence réduite. Les opérateurs de communication par satellite géostationnaire ont donc revu leur service à la hausse et proposent de meilleurs débits comme le note une étude Ookla.

En raison de la longue distance avec l'orbite géostationnaire, la latence des offres par satellite géostationnaire sera toujours beaucoup plus élevée que les autres. La comparaison des offres Wifi de plusieurs compagnies aériennes témoigne qu'en pratique, les

services fournis par Starlink sont meilleurs que ceux de la concurrence géostationnaire.

Les réseaux mobiles et fibre resteront indispensables dans les métropoles, où le déploiement d'un système terrestre sera toujours plus efficace, et où les bâtiments gênent la visibilité. Au contraire, les territoires peu peuplés ou escarpés sont favorables au satellite. La concurrence est plus vive dans les zones intermédiaires.

Aux Etats-Unis, les réseaux mobiles conservent une courte avance en débit et latence sur les réseaux par satellite, d'après speedtest et Ookla. Cependant, les performances de ces derniers continuent de s'améliorer progressivement avec le déploiement de nouveaux satellites. Les résultats de ces différentes études sont compilés dans le tableau comparatif suivant :

Fournisseur	**Débit descendant (Mbps)**	**Débit montant (Mbps)**	**Latence (ms)**
Réseau fixe	288.75	44.22	12
Réseau mobile	176.12	11.07	27
Starlink	104.71	14.84	45
HughesNet	47.79	4.44	683

TABLE 2.1 : Comparaison des performances médianes Internet aux USA (Q1 2025, sources : Ookla Speedtest)

Les performances réseau restent inégales selon les régions. Les pays africains restent actuellement moins bien desservis d'après les données Ookla. Nous rappelons ici que la performance du système repose en grande partie sur le segment sol. Le déploiement de nouvelles stations améliorera considérablement les résultats pour se rapprocher de ceux obtenus aux États-Unis.

Un épisode de congestion L'arrivée des mégaconstellations a provoqué un véritable engouement en 2022 et 2023. Le besoin d'accès à internet a été amplifié avec les confinements, et certains états (avec les États-Unis en tête) ont proposé des plans pour favoriser l'accès aux services numériques.

Le résultat ne s'est pas fait attendre : Starlink a été submergé de demandes et ses performances ont chuté considérablement dans certaines régions fin 2022 et en 2023, toujours selon Ookla.

Un peu plus tôt, l'étude de Michel et al. [MTGB22] avait montré ces quelques faiblesses. D'après ses auteurs, la latence sur un court trajet passe de 20ms à quelques centaines de ms en cas de congestion sévère. D'autre part, les chercheurs ont effectué des tests de congestion avec des transferts de fichiers par QUIC. Celle-ci s'est manifestée par la présence de pertes de paquets élevées, de l'ordre de 2%. L'analyse des pertes révèle deux types de pertes : des pertes isolées, souvent observées en upload et causées

par des erreurs liens, et des pertes de longues rafales de paquets, souvent attribuées à des épisodes de saturation des liens, donc de congestion.

Quelques mois après l'étude de Michel et al, le réseau a été massivement touché par la congestion. Les débits ont été réduits de moitié comme l'attestent des articles de presse et posts de 2022[1]. Diverses mesures de restrictions ont été prises, comme le refus de nouveaux clients dans certaines zones en Amérique du Nord. Un temps l'instauration d'un quota de données à 1 To a été envisagé, cette mesure a été finalement abandonnée. Actuellement, ces problèmes existent toujours dans certaines régions où la demande est telle que Starlink impose des frais supplémentaires à l'inscription. La congestion reste donc un problème majeur pour les mégaconstellations.

2.1.2 La congestion

La congestion est un phénomène récurrent dans les réseaux. Elle touche tous les réseaux, et plus spécialement ceux victimes de leur succès. Nous revenons dans cette partie sur la congestion : causes, conséquences et outils déployés pour la corriger. Nous verrons par la suite comment la détecter.

2.1.2.1 Causes et conséquences de la congestion

La congestion correspond à une saturation des ressources réseaux, provoquée par une demande excessive. Nous expliquons la congestion d'un point de vue théorique par le mécanisme de file d'attente, puis établissons le lien avec les conséquences qu'elle engendre sur le trafic des utilisateurs. Enfin nous l'abordons du point de vue réseau avec la notion de goulot d'étranglement.

Les files d'attente Le mathématicien Erlang, père de la théorie des files d'attente, a développé ce concept pour la gestion des lignes téléphoniques. La file d'attente permet de modéliser un système effectuant des tâches répétitives, comme le processus de transmission de paquet dans un routeur. Elle est composée de trois éléments :

1. Un buffer qui permet de stocker les données et dont la taille est limitée.
2. Une discipline, qui priorise et ordonne les tâches à traiter
3. Un service, qui traite les tâches à un débit donné

Décrivons plus en détail ces trois composants. La file d'attente est un convertisseur de pertes en délai dont la capacité est définie par un buffer. Le buffer détermine le

[1] par exemple https://www.jeffgeerling.com/blog/2022/starlinks-current-problem-capacity

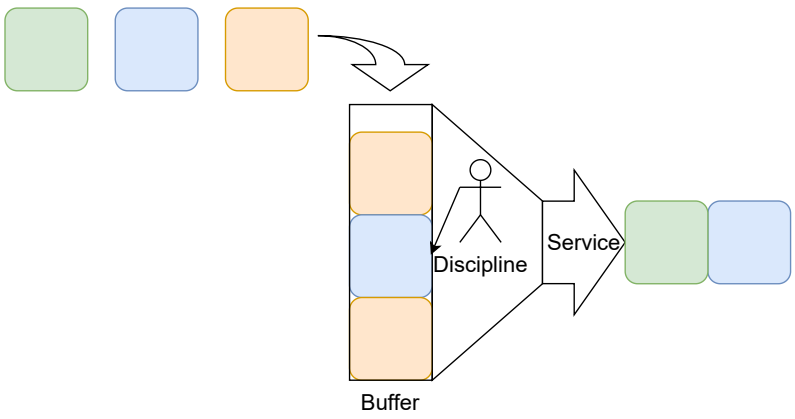


FIGURE 2.3 : Illustration d'une file d'attente : une discipline sélectionne dans un buffer le prochain paquet à transmettre immédiatement quand le service est libre

nombre maximal de paquets stockés en attente et plus il est long, plus la file d'attente est capable d'absorber des pics de charge importants et provoque des délais imposants. Au contraire, un buffer de capacité très faible n'absorbera pas les pics de trafic IP, engendrant potentiellement beaucoup de pertes.

Les disciplines sont des algorithmes de priorisation des paquets. La discipline de base est la *First In First Out*, Premier arrivé, Premier Servi. D'autres disciplines plus équitables existent, par exemple *Round Robin* et *Stochastic Fair Queuing*. D'autres, au contraire, permettent de prioriser certains flux. C'est le cas de la *pfifo_fast*, nom de la discipline par défaut dans de nombreux systèmes linux.

Le service le plus simple transmet successivement et sans perte de temps les paquets présents dans le buffer. Il existe des systèmes plus complexes constitués de files de serveur en parallèles, modélisés par des files d'attente à temps inter-espacement aléatoire. Ces types de services peuvent être aussi considérés comme des systèmes moins optimaux, caractérisés par une sous-utilisation des ressources. Par exemple, lorsque plusieurs émetteurs entrent en concurrence dans un réseau géré par le protocole Carrier Sense Multiple Access (CSMA), l'un sera transmis immédiatement tandis que les autres seront réémis plus tard, au bout d'un temps aléatoire. Ainsi, un réseau local piloté par le protocole CSMA et contenant un grand nombre d'utilisateurs peut être considéré comme une file d'attente avec un service de loi exponentielle. Le concept de file d'attente est donc général, il s'adapte à tout système de transmission ayant une limite de débit.

Définitions de la congestion De nombreux critères ont été avancés pour définir la congestion. L'auteur de [VKF+03] cite le taux de pertes, le débit moyen, la variance du débit ou encore une métrique nommée facteur du délai (*delay factor*), coefficient associé au taux d'occupation d'une file d'attente. Ces métriques forment un tableau d'indicateurs de l'état du réseau permettant d'évaluer la situation perçue par les utilisateurs et de se former un avis sur leur ressenti et la présence de congestion. Nous verrons les différentes implémentations de méthodes centrées utilisateur ou depuis l'intérieur du réseau, par la mesure du remplissage de la file d'attente.

Ces critères peuvent être utilisés pour définir un critère de congestion, en tenant compte des caractéristiques du réseau. La congestion dépend avant tout du contexte, elle est une situation d'engorgement où le surnombre d'utilisateurs sature des routeurs et provoque une chute des performances des communications :

- chute du débit des flux ;
- accroissement de la latence, issue d'une accumulation des paquets en attente de transmission ;
- taux de pertes élevé, car le débit entrant dans les routeurs excède largement leur capacité.

La congestion est indissociable de la théorie des files d'attente. Le réglage d'une file d'attente est un choix de compromis entre le délai ajouté aux paquets et le taux de pertes lorsque le débit entrant dépasse la vitesse de traitement des paquets. Si la rafale est de courte durée, le délai infligé impactera peu de paquets et une retransmission côté utilisateur est évitée, permettant de gagner un temps considérable. Au contraire, si la rafale est de longue durée, le délai infligé aux paquets dans le système s'accumule, générant de la congestion et dégradant la qualité de la communication.

Ces points de vue divergents sont réunis par la théorie des files d'attente. La loi de Little montre que le délai moyen passé dans une file est indépendant de sa discipline, mais dépend du débit entrant, du service et de la taille de buffer. Nous aborderons la théorie des files d'attente § 3.2 pour établir le lien entre la distribution des inter-espacements en entrée du système, en sortie, la taille du buffer et le débit de la file d'attente.

En conclusion de ces différents points de vue, le réseau altère le trafic entrant en lui infligeant des pertes et des délais. La congestion se manifeste par une altération excessive et inhabituelle des flux entrants. Les utilisateurs observent alors une baisse du débit et une hausse de la latence.

Congestion aux extrémités ou dans le cœur ? L'emplacement du point limitant a des conséquences très importantes sur le ressenti des utilisateurs. Lorsque le point limitant du réseau est l'utilisateur lui-même, on ne parle pas de congestion. Un utilisateur se retrouve bridé en débit par contrat avec son opérateur ou simplement par ses interfaces réseaux. Dans ces conditions, l'écoulement à l'intérieur du réseau reste fluide, tandis que les débits et la latence restent stables. Il n'y a donc pas de baisse inattendue du débit, qui correspond à celui du goulot d'étranglement habituel. La stabilité du débit assure la bonne performance d'un algorithme de contrôle de congestion. En estimant correctement le débit, l'algorithme peut adapter les émissions et ainsi minimiser les pertes, voire les délais.

Les difficultés apparaissent avec l'accroissement du nombre d'utilisateurs. Les goulots se déplacent alors vers le cœur du réseau, sur les points de rencontre des agrégats de flux. Les utilisateurs perçoivent alors une baisse de performances : augmentation du délai puis apparition de pertes. La variabilité des flux des utilisateurs rend moins précise l'estimation du débit effectuée par l'algorithme de contrôle de congestion. Sans connaissance a priori et sans moyen de se synchroniser entre eux, les utilisateurs du réseau ne peuvent plus mesurer fiablement les débits et latence du réseau. Pour limiter les pertes, chaque émetteur doit augmenter la marge de sécurité et l'algorithme de contrôle de congestion réduire encore le débit. La congestion est donc d'autant plus sévère qu'elle se situe au cœur du réseau, touchant un grand nombre d'utilisateurs. Le partage de bande passante ne laissera qu'un faible débit à chaque utilisateur, et ceux-ci auront des difficultés à l'estimer, donc à l'exploiter et généreront de la latence et des pertes.

La première chose à faire pour l'administrateur du réseau est donc de connaître les goulots d'étranglement du réseau. Un suivi attentif des points limitants permettra de s'informer de la présence de congestion dans le réseau. Lorsqu'elle apparaît, le gestionnaire de réseau pourra tenter de minimiser les délais inutiles provoqués par le remplissage des files d'attente, agir sur les délais en réglant les tailles des buffers des files d'attente (en particulier le goulot), et aider ainsi les algorithmes de contrôle de congestion dans leur estimation du débit.

2.1.2.2 Mécanismes de contrôle de congestion

Nous entrons ici plus dans le détail des mécanismes proposés pour mitiger la congestion. La première technique est le contrôle de congestion, mis en place par les utilisateurs. La seconde est celle de l'Active Queue Management (AQM) qui consistent à régler les buffers du réseau.

Nous avons mentionné dans l'introduction l'existence d'autres solutions à plus long terme comme les algorithmes de routage, l'ajout de nouveaux routeurs ou le remplacement des équipements. Ces solutions sortent du cadre de cette thèse.

Mécanismes d'*Active Queue Management* L'AQM est un compromis de gestion entre pertes et délais dans la file d'attente. Toute la difficulté est de savoir s'il faut conserver les paquets entrants car le débit va bientôt baisser, ou s'il vaut mieux les perdre immédiatement afin de limiter les conséquences de la congestion. Il faut trouver la réponse optimale à une rafale de paquets. Trop anticiper la congestion provoque des pertes de paquets inutiles et une sous-utilisation du réseau. Ne pas l'anticiper du tout revient à ajouter des délais de traitement et rend le réseau inutilisable.

Dès les débuts d'internet, Sally Floyd alerte sur le problème de bufferbloat et conseille d'implémenter des techniques d'AQM. La technique Random Early Detection (RED) [FJ93] consiste à provoquer des pertes aléatoires avant le remplissage total de la file d'attente pour informer les flots TCP du remplissage de la file d'attente et les inciter à réduire le débit. Plus simplement, certains ont conseillé de mieux régler la taille du buffer [AKM04], en fonction du nombre de flots en compétition.

Des améliorations ont été proposées pour s'adapter à des réseaux plus performants réunissant plus d'utilisateurs [GN12] ou simplifier le réglage des files d'attente en gérant la latence plutôt que la longueur de file d'attente. Les techniques restent similaires [KLM14] et les paramètres sans cesse être réadaptés aux variations du nombre d'utilisateurs, aux nouvelles générations de matériel, à la configuration du réseau, du Round Trip Time (RTT) moyen des flots ou encore des algorithmes de contrôle de congestion que nous introduisons maintenant.

Le contrôle de congestion des utilisateurs Si l'AQM optimise les réglages des files d'attente à l'intérieur du réseau et réduit les effets de la congestion, les algorithmes de contrôle de congestion joue un rôle complémentaire à l'extérieur du réseau, en contrôlant les flux issus des utilisateurs.

L'algorithme de contrôle de congestion a pour objectifs de maximiser le débit et minimiser la latence d'un flux, sous la contrainte d'un partage équitable de la bande passante. Ces deux objectifs sont contradictoires car maximiser le débit implique d'émettre beaucoup de paquets, donc de remplir les files d'attentes du système et finalement augmenter la latence. L'article [YMH+18] compare un grand nombre d'algorithmes et montre que quelques-uns offrent de meilleures performances absolues. L'étude indique aussi clairement que ces meilleures performances dépendent fortement des conditions du réseau, et que le compromis débit/latence est très variable.

Le RFC 9743 [DF25] résume la situation actuelle d'internet. Actuellement la solution au problème de congestion dépend du bon comportement de chaque utilisateur. Le partage des ressources est garanti par l'utilisation d'un algorithme de contrôle de congestion similaire pour tous. Cet algorithme doit

- protéger contre l'effondrement des performances en cas de congestion sévère en réduisant le débit au minimum ;
- protéger contre le bufferbloat en évitant de remplir les buffers et causer une latence démesurée nuisible à tous ;
- protéger contre de fortes pertes de paquets, notamment lorsque ces pertes sont dues à un débit trop élevé ;
- garantir l'équité entre les utilisateurs et converger vers un partage équitable ;
- gérer les connexions courtes et s'adapter à de nombreux flots courts qui mettent fin à leur connexion avant même de finir la phase de *slow start*.

Le respect de ces exigences repose sur l'interprétation d'observations du réseau. À chaque paquet de données émis, le destinataire répond par un paquet Acknowledgement (ACK) indiquant son arrivée. L'émetteur mesure le RTT grâce au retour du correspondant par un paquet ACK. S'il n'est pas reçu au bout d'une certaine marge de délai, le paquet émis est supposé perdu. L'émetteur adapte ainsi son débit et en fonction des mesures relevées sur les paquets précédents. À petite échelle, ces décisions sont efficaces et la congestion est évitée aux extrémités du réseau. Mais à grande échelle l'influence de l'utilisateur se noie dans la masse et la congestion est beaucoup plus compliquée à gérer dans le cœur du réseau. Sans connaissance de l'état du réseau, une masse d'utilisateurs aura des difficultés à se coordonner pour améliorer globalement la situation. D'où l'intérêt des méthodes d'AQM et l'importance pour le gestionnaire réseau de connaître l'état du réseau de façon globale.

Les algorithmes de contrôle de congestion ont historiquement deux phases. Au cours de la première phase de *slow start*, le débit croît exponentiellement jusqu'à atteindre un débit proche du partage équitable de la bande passante. La seconde phase dite de *congestion avoidance* débute alors, l'algorithme conserve un débit stable et s'adapte aux conditions du réseau. Il teste régulièrement une augmentation du débit et diminue le débit si les performances évoluent de façon défavorable. Actuellement il existe plusieurs versions d'algorithmes de contrôle de congestion. La plus utilisée est TCP Cubic [HRX08], configurée par défaut dans les systèmes Linux et Windows. Les équipes de Google ont proposé l'algorithme BBR [CCG+17] de l'anglais *Bottleneck Bandwidth and*

RTT qui estime donc la bande passante de la connexion et le RTT. L'estimation de ces paramètres lui permet d'éviter de remplir les buffers et ainsi améliorer les performances, notamment en termes de délai. Les deux algorithmes Cubic et BBR (en considérant les dernières versions de BBR [CSB25]) réagissent aux pertes sur le principe de l'AIMD. Lorsque les conditions du réseau sont bonnes dans la phase de *congestion avoidance*, le débit augmente linéairement, mais il décroît exponentiellement lorsque des pertes apparaissent. Ce principe de l'AIMD assure le partage équitable des ressources entre les utilisateurs.

En conclusion, les utilisateurs ont une bonne connaissance de la performance du système, mais ne sont pas toujours capables d'agir. Au contraire, remarquons que le gestionnaire réseau a plus de moyens d'action contre la congestion (et autres problèmes du réseau), mais il a besoin pour cela de connaître l'état du réseau.

2.2 Détecter une baisse de performances des communications des utilisateurs

Cette seconde partie décrit les moyens de connaissance de l'état du réseau du point de vue de son gestionnaire. Nous détaillons des méthodes de connaissance du réseau, détection de congestion et localisation des sources de pertes par des techniques de métrologie. Nous nous appuyons sur l'étude d'Owezarski et al. [OL06] pour présenter ces techniques. Nous commençons par des études centrées sur la performance des utilisateurs, puis poursuivrons par des méthodes dédiées à la connaissance de l'état du réseau.

2.2.1 Approches individuelles des flux de trafic

Ces premières méthodes consistent à identifier la présence de congestion par ses effets directs, c'est-à-dire une baisse de performance des communications entre les utilisateurs du réseau. Nous cherchons dans un premier temps à identifier cette baisse de performance dans les communications des utilisateurs eux-mêmes. La seconde partie présente des moyens dédiés, déployés par les gestionnaires de réseaux ou d'autres entités pour mesurer la performance des réseaux.

2.2.1.1 Analyse du trafic des utilisateurs

La détection d'anomalies se base sur l'analyse du comportement d'un flux et sa comparaison avec un comportement de référence, qu'il soit attendu ou anormal. Cette analyse se base sur un ensemble d'observations effectuées depuis nœud du réseau, qui collecte

des informations sur les événements qui s'y déroulent : qui fait quoi, où et quand. Avant de présenter les méthodes d'analyse du trafic des utilisateurs, nous étudions la sécurisation des réseaux qui pose actuellement un défi d'observabilité du réseau et force à proposer de nouvelles méthodes d'analyse du trafic.

Le défi d'observabilité du réseau Plusieurs tentatives ont été proposées au début des années 2000 pour favoriser la collaboration entre les membres du réseau. Par exemple les routeurs peuvent modifier le champ Type of Service (ToS) pour inciter les émetteurs à réduire leur débit. Le protocole XCP [KHR02] pousse encore plus la collaboration avec la mise en place de protocoles d'échange d'informations réseau. Ces protocoles n'ont pas connu de déploiement à grande échelle.

Internet s'est éloigné de cette dynamique de Beaucoup de méthodes développées pour l'étude des performances dans le réseau extraient les informations partagées en clair par les protocoles, notamment les numéros de séquence et d'acquittement des en-têtes de TCP.

Cependant, la sécurisation progressive des protocoles, largement soutenue par les fournisseurs de contenu, limite de plus en plus les informations lisibles par les nœuds intermédiaires du réseau. Le déploiement de HyperText Transfer Protocol version 2 (HTTP/2 ou HTTPS) a protégé les utilisateurs finaux des intermédiaires mal intentionnés et permis l'essor du commerce en ligne. Ce protocole est actuellement le plus répandu sur internet avec le protocole TCP. Mais ce dernier est progressivement remplacé par QUIC avec l'appui de géants du web (Google, Meta) et HTTP/2 abandonné au profit de HTTP3.

Avec QUIC, le nœud intermédiaire n'a accès qu'aux informations nécessaires pour router (adresses du protocole IP et numéros de port des protocoles TCP ou User Datagram Protocol (UDP)). QUIC implémente le spin bit qui permet d'estimer le RTT mais son adoption a suscité des remous et reste limitée [KSW23].

TCP n'est pas le seul protocole touché, une version sécurisée du protocole Domain Name System (DNS) a été implémentée avec DNS over HTTPS (DoH) : désormais les noms des sites web ne circulent plus en clair.

Le RFC 8546 [TK19] introduit finalement le concept d'observabilité du réseau avec la notion de "wire image", afin de garantir la vie privée et minimiser les connaissances que peuvent extraire les intermédiaires. L'image réseau correspond aux informations que peut capturer un intermédiaire réseau. En résumé, il reste actuellement les informations de source et destination (adresse IP et port), la taille des paquets et leurs instants d'arrivées. Ces instants d'arrivée des paquets, souvent résumés à la distribution des temps d'inter-arrivées des paquets, sont caractéristiques d'un comportement

de l'utilisateur. Sa caractérisation peut ouvrir la voie à des méthodes de classification du trafic et de détection d'anomalies.

Analyse du trafic des utilisateurs L'idée ici est donc de se reposer sur les utilisateurs pour identifier les anomalies. Les flux contrôlés estiment les conditions du réseau et réagissent en fonction de leur évolution. Il suffit alors d'identifier les flux, collecter les informations que transmettent ces protocoles et observer leurs réactions pour identifier la présence d'anomalies dans le réseau. Ce principe est possible à condition que le comportement de l'utilisateur soit prévisible pendant la période d'étude. Une connexion très courte ne sera pas une bonne candidate pour cette analyse. Plusieurs études montrent qu'il est possible de les identifier par lecture des en-têtes des paquets et analyse des séries temporelles des paquets.

La technique nécessite de connaître et caractériser l'application au préalable. Ce sujet qui relève des techniques de Network Traffic Monitoring and Analysis (NTMA) a largement intéressé la communauté des réseaux. L'état de l'art sur ce sujet a beaucoup évolué, passant du *Machine Learning* [NA09] à des techniques de *Deep Learning* bien plus puissantes (Long Short Term Memory (LSTM), Attention, Large Language Model (LLM)) [XYH+19], [AST21]. Ainsi, [LMCSEL17] prédit si le flot observé provient d'Office 365, d'un service Microsoft, si c'est une vidéo YouTube, un flux QUIC, etc. L'entraînement de ces classificateurs nécessite une vérité terrain, obtenue par exemple par *Deep Packet Inspection*. Des travaux similaires portent sur les applications mobiles [ZPH19], [CLMS17], les utilisateurs de téléphone portables étant désormais majoritaires dans internet.

YouTube est un bon candidat, car les connexions durent suffisamment longtemps et l'application est fréquemment utilisée dans les pays occidentaux en soirée : elle compte pour près de 10% du trafic internet selon le dernier rapport Sandvine. L'article [PPRL24] collecte de nombreuses statistiques de mesures de débits et de tailles de paquets des mesures de débit afin de prédire la définition de la vidéo et identifier des épisodes de blocage (*stalling*) dans ces flux chiffrés. La définition de la vidéo est liée à la qualité moyenne de la communication, tandis que les épisodes de blocage indiquent des dégradations subites du réseau.

Des résultats intéressants peuvent être obtenus au niveau de la couche transport par l'étude de flux de protocole TCP, protocole le plus répandu dans internet actuellement. Divers articles ont proposé des méthodes de calcul du RTT par les premiers échanges SYN-ACK et lors de la phase de Slow Start [JD02]. Sur un flot cette étude n'est pas utile, mais une étude statistique des RTT de tous les flots pourrait révéler la présence de congestion, par exemple en montrant un accroissement général du RTT.

Dans certaines versions d'algorithme de contrôle de congestion, le RTT peut aussi être mesuré passivement lors de la phase de congestion avoidance [VLL05]. Dans la même lignée, l'article [TFA+24] détecte un comportement anormal par la transition entre *slow start* et congestion avoidance. Si, en sortie de *slow start*, la diminution de la taille de la fenêtre de l'algorithme de contrôle de congestion marque l'apparition de pertes imprévues. Au contraire, son augmentation est corrélée à la présence d'une gigue élevée et de pertes faussement détectées. Ces deux situations sont anormales, TCP est leurré et l'utilisation du réseau n'est pas optimale. L'apparition de nouveaux algorithmes de contrôle de congestion pourrait compromettre cette méthode, mais l'article présente aussi une méthode d'identification de cet algorithme, notamment pour le différencier du plus moderne BBR.

2.2.1.2 Benchmarks du réseau

Les méthodes passives/collaboratives développées précédemment rencontrent des difficultés à cause de la tendance actuelle au chiffrement des communications. Nous présentons donc ici l'arsenal des méthodes directes pour mesurer les performances du réseau. Des dégradations de ces performances peuvent être dues à divers facteurs, notamment la congestion.

Sondes de test Les méthodes de base pour caractériser le réseau sont des méthodes d'émission de sonde dans le réseau. L'outil le plus simple et le plus répandu est probablement la commande ping qui permet de tester la présence d'un correspondant du réseau et mesurer le RTT. Elle repose sur le protocole Internet Control Message Protocol (ICMP), assez largement implémenté et accepté sur internet. Plus évolué, Iperf mesure le débit, le RTT et le taux de pertes du transfert d'un gros fichier comme le ferait l'utilisateur. Il nécessite d'avoir accès à deux machines du réseau, l'un émettant le flux de données, l'autre le recevant. Les deux entités échangent des informations pour mesurer les performances de la communication. De tels outils ont été déployés à grande échelle par différents acteurs privés comme QoSi, Ookla pour évaluer la rapidité d'une connexion standard. Enfin la littérature des années 2000 témoigne d'une activité de recherche intense sur l'utilisation de sondes pour estimer des capacités de débit sur des chemins comme le montre le tableau 1 de [MBD04]. L'un des premiers outils est Bprobe de Carter et Crovella [CC96]. Il consiste à émettre des salves de paquets ICMP à débit maximal puis mesurer la différence d'arrivée entre deux paquets successifs. L'écart entre les paquets observé par le récepteur sera dimensionné par le lien dont le débit est le plus faible, comme représenté figure 2.4.

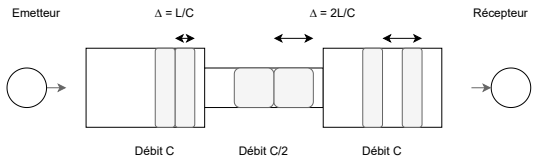


Figure 2.4 : Identification d'un lien à débit limité par observation du temps d'inter-arrivée d'une paire de paquets. En notant L la taille d'un paquet, on remarque que l'écart Δ qui sépare les paquets prend la valeur du lien à débit minimal.

La distribution des inter-arrivées est largement impactée par ce lien dont la capacité est réduite. Bprobe estime le débit de ce lien limitant par le mode principal des inter-arrivées. L'outil nommé Cprobe calcule le ratio entre le débit à l'émission d'une salve de paquets ICMP et le débit mesuré à la réception et mesure ainsi la congestion. Les outils se sont multipliés sur ce même concept d'analyse de paires de paquets pour déterminer la bande passante disponible et la congestion. Nettimer [LB01] filtre la densité des inter-arrivées avec un noyau gaussien tandis que Pathrate de [DRM01] améliore le choix du mode de la distribution des inter-arrivées pour estimer la bande passante disponible. Paxson analysait avec précision des histogrammes d'inter-arrivées dans sa thèse [Pax97] et attribue les pics de densité aux différents liens. L'article [KKB+04] poursuit l'étude de ces différents modes et montre la possibilité d'identifier d'identifier la congestion sur plusieurs liens intermédiaires, à différents débits en zoomant plus ou moins dans l'histogramme. Enfin, l'outil Spruce est un outil de détection de congestion sur un lien qui estime la bande passante disponible en connaissant a priori la capacité maximale [SKK03].

Généralisation Les outils précédemment décrits doivent être utilisés à l'échelle d'un réseau pour en tester les multiples chemins et détecter la congestion. Aussi, l'article [FOE22] illustre une étude menée chez différents opérateurs mobiles scandinaves. Un ensemble de terminaux est déployé avec les outils QProbe et TSLP, estimant respectivement la bande passante disponible et le RTT. Plus spécifiquement, QProbe émet des séries de paquets sonde dont l'analyse révèle la présence de congestion, tandis que TSLP détecte des variations de RTT. Ces deux outils sont combinés pour créer des métriques de performances utilisées pour déceler des conditions dégradées pour les utilisateurs et la congestion de façon générale. Un tel réseau est complexe à entretenir, l'article [MHL+14] minimise le nombre de collaborateurs nécessaires et les place correctement pour estimer les caractéristiques de tous les liens sans exception. Il minimise ensuite le nombre de mesures nécessaires entre ces collaborateurs et complète les mesures manquantes de délai, bande passante ou présence de lien défectueux entre toutes

les paires de collaborateurs [MZS20].

Ainsi, des mesures centrées utilisateurs peuvent être décomposées pour estimer les propriétés des différents liens du réseau dans une formulation duale où les variables ne sont plus les utilisateurs mais les différents liens du réseau. Cette inversion nécessite de connaître la topologie du réseau, c'est-à-dire tous ses nœuds et liens. Différents articles ont montré des liens forts entre performance des liens et topologie. En effet, les conditions de traitement de deux paquets passés par un même nœud à un faible intervalle de temps seront similaires. L'article [KKP22] utilise la covariance de séries temporelles de délai ou pertes, tandis que [SJSB22] utilise des cumulants (moments d'ordre supérieur) pour caractériser la distribution de la somme des pertes ou de la latence. Ces deux articles utilisent ensuite des techniques de décomposition différentes afin de retrouver les liens communs entre les flux.

2.2.2 Approche du point de vue réseau

Les méthodes présentées ci-dessus adoptent un point de vue réseau, visant la caractérisation des utilisateurs pour détecter des problèmes réseaux. Ces problèmes peuvent être abordés d'un point de vue complémentaire, en considérant l'état des routeurs et des liens du réseau plutôt que la performance individuelle des utilisateurs. Nous présentons donc différentes méthodes permettant de connaître cet état du réseau, puis concluons sur les différentes approches de caractérisation des utilisateurs et de l'état du réseau.

2.2.2.1 Tests de présence de goulot et estimation de l'état d'une file d'attente

Une information capitale pour éviter la congestion dans le réseau est de connaître ses points limitants. Nous abordons ici le problèmes de détection, localisation et caractérisation des goulots d'étranglement. Deux familles de méthodes permettent de résoudre ce problème : les méthodes basées sur des corrélations de flux puis sur des analyses d'agrégats.

Méthodes par corrélation De nombreuses méthodes centrées utilisateurs ont été élaborées pour connaître l'état du réseau. Elles constituent la base de l'état de l'art en inférence de topologie et détection de goulot d'étranglement. Nous présentons ces méthodes d'inférence de topologie et de caractérisation des files d'attente.

La plupart de ces travaux sont basés sur des méthodes de corrélation de taux de pertes ou de délai [HLM+04]. Le lien dimensionnant étant la principale cause de pertes

de paquets, des détecteurs basés sur la corrélation des pertes ont été expérimentées [HBB00]. Toutefois, leur précision laisse à désirer parce que les pertes sont rares dans les réseaux standards.

La seconde caractéristique interne des files d'attente, le délai, est donc mise à contribution. Le délai total expérimenté par les paquets d'un même flot est la somme des contributions de chaque file d'attente. Chacune est modélisée comme un processus aléatoire, dont les caractéristiques de délai et de taux de perte évoluent peu sur un court intervalle de temps. Deux paquets partageant un chemin commun subiront des conditions similaires dans chaque file d'attente commune. Rubenstein [RKT00] exploite cette propriété dans une méthode basée sur l'émission de flux de trafic sonde. Les instants de réception et d'émission des paquets sont collectées pour créer des séries temporelles du temps de traversée des paquets à travers le réseau. Une corrélation forte entre deux séries temporelles de deux flux différents indique que les paquets ont probablement traversé les mêmes files d'attente. Cette technique permet donc de détecter les goulots communs [RKT02]. La technique sur les séries temporelles de délais a été optimisée dans des travaux suivants, par exemple avec une transformée en ondelettes [KKS+08] avant de calculer la corrélation et valider ou non la présence d'un goulot commun.

Lorsque le réseau comporte plusieurs flots, les auteurs de [YWY08] proposent d'utiliser une décomposition SVD de la matrice des statistiques de corrélation des flots 2 à 2. Plutôt qu'un test de corrélation, Hayes a proposé d'utiliser un mélange de différentes statistiques [HFW14] pour regrouper les flots partageant un goulot commun. Il en compare les performances avec une méthode de décomposition en ondelettes et différents algorithmes de partitionnement [HWF+20].

Finalement, toutes ces méthodes estiment une valeur interne des files traversées (pertes ou délai ajouté par chaque file) qui ne peut être estimée sans collaboration entre la source et l'observateur de la communication. Ces approches nécessitent des mesures de délai ou de pertes, donc d'avoir accès aux numéros des paquets. Il ne semble donc pas possible de construire une méthode purement passive sur ces hypothèses, à cause du chiffrement des protocoles.

Étude de la distribution des temps inter-arrivée La seconde famille de méthodes centrées réseau est basée sur l'étude de la distribution des inter-espacements des paquets en sortie de la file d'attente.

Une méthode simpliste, tirée des méthodes actives est d'observer le minimum ou le mode principal de la distribution des inter-arrivées [CC96]. Ces méthodes naïves ne sont pas très robuste, la première ne résiste pas aux altérations du trafic générées

par une file d'attente intermédiaire entre l'observateur et les goulots d'étranglement, tandis que la seconde est difficile à interpréter, la distribution des inter-arrivées étant complexe à analyser. La clef du problème réside donc avant tout dans l'interprétation de cette distribution qui contient l'information de présence de files d'attente, comme l'ont montré certains auteurs [Pax97] [LB01].

Varga a tenté d'utiliser le lien entre délai dans la file d'attente et distribution des inter-arrivées des paquets pour indiquer la présence de congestion. Le *delay factor* [VKF+03] est un rapport entre le délai mesuré dans une file d'attente et le délai attendu. Ce délai attendu est estimé grâce à une formule qui relie la distribution en sortie avec le temps de séjour en connaissant le débit de la file d'attente, pourvu que la loi des paquets en entrée soit une loi de Poisson (modèle Markovien). Cette méthode est efficace mais l'application est limitée : pour mesurer la congestion d'une file d'attente, il faut se placer à sa sortie immédiate, connaître son débit et mesurer le débit du trafic sortant.

Aussi Varga poursuit l'étude avec des moments d'ordre 3 (kurtosis et asymétrie) dans la distribution des inter-arrivées pour mesurer la congestion [Var06]. D'autres auteurs ont eu recours à des études temporelles. L'article [HPH+09] propose une solution originale : les auteurs mesurent le trafic sortant de file d'attente de différents débits standards et calculent la transformée de Fourier du flux sortant. Ils fabriquent ainsi un dictionnaire de signatures qu'ils comparent ensuite avec la transformée de Fourier d'un flux à étudier. Si le spectre d'un flux matche avec une entrée du dictionnaire, la présence de congestion est détectée et d'un même coup la capacité du lien limitant est mesurée.

Enfin, la méthode la plus efficace de l'état de l'art semble celle proposée par Katabi [KB01]. Le principe est simple : une file d'attente congestionnée émet de longues séries de paquets à débit constant, produisant des pics caractéristiques du débit dans l'histogramme des inter-arrivées des paquets. À l'inverse, les inter-espacements entre les paquets sont quelconques lorsque les files d'attentes ne se remplissent pas. Ces pics sont mis en valeur par la mesure de l'entropie de Rényi qui est une généralisation de l'entropie de Shannon. Les résultats obtenus avec cette métrique sont généralement très bons, quelques cas posent problème comme lorsque deux files d'attente de même débit saturent en même temps. Nous reviendrons sur ce problème. Finalement, Katabi montre que l'entropie des temps inter-arrivée d'un agrégat de flux issus d'un même goulot est faible. Trouver les goulots revient à classer les flux en fonction du goulot traversé et donc trouver la partition C des flots qui minimise la somme des entropies H de la distribution des temps inter-arrivée de chaque agrégat de flux C_k

$$\arg\min_{C} = \sum_{k=0}^{K} H(C_k) \tag{2.1}$$

2.2.2.2 Conclusion de l'état de l'art sur la nécessité d'adopter un point de vue du réseau pour l'optimisation du réseau

Nous achevons cet état de l'art par une comparaison des approches visant à caractériser le réseau. La comparaison nous semble plus favorable à l'approche du point de vue réseau, dont nous détaillons ensuite les possibilités.

La différence des points de vue utilisateur et réseau Les méthodes basées sur l'analyse de flux individuels ont révélé plusieurs limites. Les méthodes d'analyse passive ont actuellement des difficultés liées au chiffrement du protocole ; les méthodes actives fonctionneront toujours, mais leur déploiement est une contrainte et leur utilisation est défavorable aux clients du système. Finalement, ces méthodes permettent d'accéder à des métriques qui reflètent la performance rencontrée par les utilisateurs du réseau, mais n'indiquent pas directement si le réseau fonctionne correctement et est utilisé à son plein potentiel. Le schéma de pensée appliqué dans ces méthodes est de chercher à satisfaire les utilisateurs. Cela se traduit en termes de Quality of Experience (QoE), Quality of Service (QoS) et débouche sur l'analyse d'un flux à la fois. Ainsi, dans [KKB+04] comme dans la plupart des articles, les flux sont analysés individuellement afin d'en extraire une information sur l'état de chaque connexion. Une synthèse de ces résultats permet de déduire des informations sur l'état du réseau et connaître son fonctionnement, tandis que les anomalies seront exprimées par une chute de performance (débit, latence) de l'utilisateur moyen.

En comparaison, l'objectif de l'approche réseau est avant tout le bon fonctionnement du réseau ; la satisfaction des utilisateurs découlera de l'optimisation apportée au fonctionnement du réseau. Les anomalies sont exprimées différemment : on voudra savoir si des équipements sont défaillants, des paquets mal routés ou encore si l'attente dans les files est excessive. Ces anomalies pourront être détectées par comparaison entre l'état actuel du réseau et son état attendu. La méthode d'analyse du trafic change avec l'objectif. Contrairement au cas précédent, l'approche réseau inverse les deux étapes décrites précédemment : les flux partageant une portion de trajet commun sont d'abord agrégés, puis l'agrégat global est analysé. L'analyse de l'agrégat de flux permet de mieux mettre en valeur les caractéristiques de la portion de trajet commune et atténuer tout ce qui relève des portions spécifiques à chaque flux.

Estimation de l'état du réseau En conclusion, nous adoptons une démarche selon l'approche réseau illustrée dans la figure 2.5. Les travaux de cette thèse concernent uniquement la zone grisée.

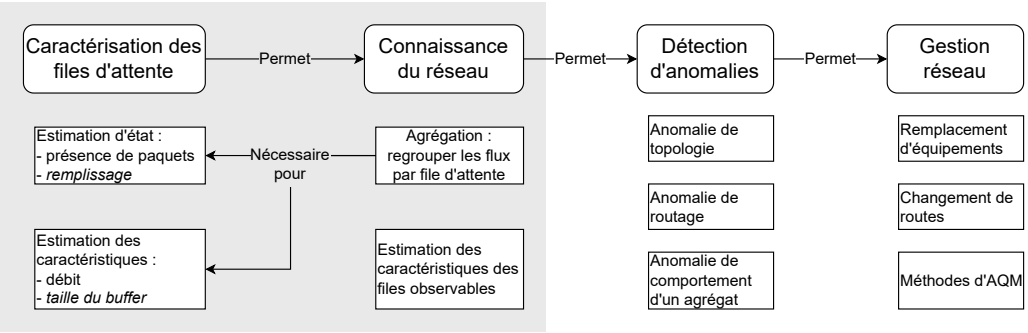


FIGURE 2.5 : La méthode de la thèse résumée

Nous aborderons de façon approfondie la description d'un réseau dans la partie suivante, qui traite de modélisation et simulation de réseaux. Ici, en quelques mots, il s'agit de connaître les nœuds, les liens, le routage et finalement l'état des files d'attente. Une file d'attente est décrite par

- son flux entrant, associé à un taux de pertes ;
- une latence ou remplissage de son buffer ;
- son flux sortant, principalement caractérisé par la distribution des inter-espacements des paquets sortants.

Par définition, l'agrégat issu de la file d'attente ne contient pas d'information sur le taux de pertes ou le remplissage du buffer. Les méthodes passives par analyse de l'agrégat sortant ne permettront donc pas d'obtenir ces informations internes à la file d'attente sans autre connaissance sur les flux entrants dans le réseau. L'agrégat contient deux informations : le débit de sortie de la file d'attente et la saturation, qui est la mesure de présence de paquets en attente. Le schéma 2.5 indique qu'obtenir ces informations implique d'observer le bon agrégat de flux, donc de connaître la topologie. À cause de cette interdépendance, la méthode doit donc réaliser une estimation conjointe des flux partageant une file commune et le débit de cette file commune. La thèse n'approfondit donc pas la mesure de saturation et la détection des goulots du réseau qui sont laissés en perspective.

Le gestionnaire du réseau a accès aux équipements et peut déployer un système de collecte des informations de toutes les files d'attente. Il peut se constituer une base de

données qui servira à comparer l'état du réseau avec son état attendu pour détecter les anomalies. La constitution d'une telle base de données simplifierait le fastidieux travail d'analyse du trafic que nous proposons dans cette thèse. L'avantage d'un système passif est qu'il ne nécessite pas de droits d'accès ni de déployer un protocole qui pourrait être complexe à mettre en œuvre dans les mégaconstellations à cause de la taille du réseau ou des fréquents *handovers*.

Chapitre 3

Modélisations d'un réseau internet

Le jeu de données est la matière du travail de cette thèse qui consiste à retrouver l'état du réseau à partir de l'analyse d'une partie des événements qui s'y sont produits. Le jeu de données doit contenir ces informations de façon fiable et précise pour bien comprendre le problème, tester des hypothèses et valider des résultats. Les événements d'un réseau sont décrits dans des fichiers de journalisation (aussi nommés *logs* ou traces réseau) par point de passage qui contiennent les métadonnées des paquets : instant de passage et en-tête de chaque paquet contenant la source, la destination, le type de protocole et des informations échangées par le protocole.

Nous nous focalisons ici sur l'inter-espacement des paquets uniquement. En effet, ces dernières informations échangées par les protocoles sont progressivement chiffrées, notamment avec le déploiement de QUIC. L'état de l'art montre qu'il est possible d'obtenir des résultats intéressants à partir de l'analyse des instants et différences d'instants d'arrivée des paquets, comme montré précédemment dans l'état de l'art 2.2.2.1. Leur précision sera donc cruciale pour parvenir à de bons résultats. Le second critère de qualité du jeu de données est la connaissance de l'état du réseau : sa topologie, les débits des liens et le remplissage des files d'attente. Différentes alternatives sont présentées dans ce chapitre pour obtenir un bon jeu de données. Cela relève du parcours d'obstacles : nous n'avions pas à disposition de jeu de données réelles complet et avons décidé d'en générer un. Différentes techniques ont été envisagées. La première est d'utiliser un modèle mathématique de file d'attente, qui se révèle trop imprécis temporellement. La seconde option est un simulateur, dont la précision donne satisfaction, mais sa lenteur nous a conduits à tester l'émulation. Cette dernière méthode, basée sur des conteneurs docker, nous permet finalement de générer rapidement de bons jeux de données dans des cas simplifiés.

3.1 L'alternative entre données réelles et génération de données

Au moment de débuter véritablement le travail de thèse, deux options se présentent à nous : travailler sur des données réelles ou en générer. Nous présentons donc les avantages, inconvénients et principes de ces méthodes.

Données réelles La meilleure option, qui est aussi la plus simple est de travailler avec des données réelles. Trois obstacles s'opposent à cela : le déploiement d'un tel système est complexe ; les entreprises ne veulent pas forcément partager leurs données ; enfin les données privées des clients sont confidentielles, la législation de protection des données privée est contraignante. Le déploiement de protocoles chiffrés s'inscrit dans cette optique de protection de la vie privée des clients et de non-divulgation du fonctionnement du réseau. Cela conduit à redéfinir ce qu'est une méthode passive. À terme, seules les méthodes basées sur des mesures de temps inter-arrivées pourront être qualifiées de passives. Ici, la qualité d'un jeu de données se mesure d'abord à la précision des instants d'arrivée des paquets. Dans le cadre d'une démarche supervisée, nous avons besoin d'informations complémentaires sur l'état du réseau. Les traces réseau sont collectées par capture de trafic avec des logiciels comme Wireshark. Cet outil capture toutes les trames du réseau local, même celles qui ne sont pas destinées à l'hôte. La capture de trafic dans une infrastructure en service ne doit pas avoir d'impact sur l'écoulement du trafic. Aussi la capture n'est pas effectuée directement sur le

routeur qui voit passer l'écoulement, mais un premier équipement duplique le trafic sans ralentir ni altérer le trafic client et dirige la copie vers l'équipement de capture du trafic. Enfin une chaîne de traitement filtre les paquets, tronque le contenu inutile et anonymise les adresses IP pour empêcher la reconnaissance des utilisateurs. Un tel dispositif n'est donc pas simple à mettre en œuvre et maintenir ; rares sont les plateformes qui mettent à dispositions les données collectées. En pratique, ces informations sont souvent accompagnées de la topologie et des débits des liens. L'état du réseau peut être connu à partir des traces collectées en tous points du réseau, ou par des statistiques d'utilisation des files d'attente générées par la commande tc (*traffic control*).

La base de données MAWI [CMK00] contient des enregistrements effectués régulièrement dans un réseau universitaire au Japon. Ces données sont faciles d'accès mais ne sont pas exploitables dans notre cas pour deux raisons : elles manquent de précision temporelle ; les mesures ont été effectuées sur un lien en bordure du réseau dont le débit relativement faible ne laisse pas passer suffisamment d'information pour espérer identifier un goulot dans le réseau avec les méthodes testées. Le site CAIDA regroupe des données hétéroclites issues de différents réseaux. Certaines sont disponibles librement, d'autre sont soumises à autorisation. Seules ces dernières auraient éventuellement pu nous intéresser, nous n'avons pas effectué la démarche.

Données issues d'un modèle Contrairement aux données réelles, les données issues d'un modèle ne contiennent pas d'aberration ou d'imprévu. Le scénario d'une simulation est parfaitement déterminé et sa bonne maîtrise permet de connaître à l'avance les points limitants du réseau et les liens défectueux provoquant des pertes. Le choix des paramètres et la maîtrise des logs simplifient largement le travail de compréhension du réseau et la définition de la réalité terrain. On peut ainsi comparer l'efficacité des méthodes de contrôle de congestion et les différentes configurations du réseau sur des résultats très fiables. La labellisation des données est particulièrement facile dans ce contexte et ouvre une voie royale pour un apprentissage supervisé et plus généralement la détection d'anomalies.

Le contrôle de tous les éléments du réseau est un avantage considérable qui simplifie les mesures et la labellisation des données. À l'inverse, seules les arrivées des paquets seront observables dans scénario réel.

Le réalisme est le souci majeur de la simulation depuis les débuts d'internet [FP01]. Différents modèles d'abstraction ont été envisagés pour déceler la congestion d'une file d'attente :

1. des modèles mathématiques de files d'attente ;

2. un simulateur à événements discrets, précis mais complexe ;

3. un émulateur, qui s'est avéré être le meilleur compromis de simplicité et réalisme.

Nous présentons ces différents modèles, en portant une attention particulière aux inter-espacements des paquets en sorties. Comme l'ont montré les techniques de l'état de l'art, la distribution des inter-arrivées contient des informations qui nous permettront de déceler les goulots.

3.2 Modèles probabilistes

Un réseau effectue deux types d'opérations sur les flux de paquets IP. Le premier type est le routage, qui divise selon leur destination des flux réunis en un point. À l'inverse, le second type est le multiplexage qui agrège dans une file d'attente des flux provenant de sources différentes. Nous nous intéressons à ce second mécanisme qui, intuitivement, en rassemblant les flux issus des différentes régions du réseau, collecte aussi les informations. Le fil conducteur de cette étude probabiliste est la recherche d'expression analytique de l'état interne d'une file d'attente en fonction du flux sortant, en posant un minimum d'hypothèses sur le flux entrant. La théorie des files d'attente permet ainsi de mettre en équation la distribution des inter-espacements des paquets en entrée et en sortie, le remplissage de la file d'attente, le délai subi par les paquets et finalement la présence de congestion. La plupart des résultats théoriques présentés sont extraits des ouvrages [GSTH11] et [GP99].

Nous avons vu que la congestion se manifeste du point de vue interne au réseau selon deux critères : hausse du taux de pertes ou allongement des délais d'une file d'attente. Nous partons d'une étude générale des files d'attente pour relier le flux observé en sortie, le flux entrant et l'état d'une file d'attente. Malgré l'ajout d'hypothèses supplémentaires, nous montrerons que les possibilités par l'étude analytique restent limitées. Nous continuerons ce chapitre par l'étude de méthodes de génération de données, tandis que le chapitre suivant traitera de l'exploitation de ces données.

3.2.1 Estimation du temps d'attente

Nous commençons par nous focaliser sur l'estimation du temps d'attente dans la file pour mesurer la congestion. Une première analyse dans un cas général montre la nécessité de connaître l'entrée de la file d'attente et son service. Nous ajoutons alors l'hypothèse de la loi d'inter-arrivée exponentielle qui simplifie l'étude et remplace la variable interne des instants d'arrivée des paquets par l'observation de la sortie.

3.2.1.1 Estimation du temps d'attente dans le cas général

Il existe de nombreuses variantes de files d'attente, comme nous le verrons dans un premier temps avec la notation de Kendall. Dans un second temps, nous verrons la loi de Little, qui s'applique à toutes les files d'attente indistinctement de ses paramètres.

Une classification des modèles de file d'attente La théorie des files d'attente, est née au début du $XX^{\text{ème}}$s de l'imagination du mathématicien Erlang qui voulait optimiser les réseaux téléphoniques. Historiquement, une file d'attente met en scène des clients qui se présentent au standard téléphonique, patientent le temps que le standardiste leur réponde, puis sont connectés à leur correspondant. Le client désire attendre le moins possible, tandis que l'opérateur a intérêt à minimiser la quantité de standardistes et de lignes tout en assurant un bon service aux utilisateurs. Les ressources pourront être ajustées en fonction de la demande lors d'un pic de charge afin de satisfaire les clients. La satisfaction des clients pourra enfin être mesurée par la probabilité de ne pas dépasser un certain temps d'attente.

De nombreuses variations peuvent être apportées à ce problème. La notation de Kendall les classe sur la base de la série de paramètres suivante, notée sous la forme Loi d'arrivée/Loi de service/nombre de serveurs parallèles/capacité maximale/nombre de clients/discipline.

Les lois définissent des instants d'entrée dans le système ou de sortie d'un paquet. Parmi les plus utilisées figurent la loi exponentielle définie par la lettre M (markovien). Il existe aussi la loi déterministe D. La lettre G est utilisée dans un cas Général, lorsque la loi n'a pas d'influence sur une formule énoncée, ou que cette formule s'adapte quel que soit le résultat. La théorie des files d'attente s'adapte aux systèmes composés de multiples serveurs et permet de comparer la performance d'un gros serveur rapide avec plusieurs petits serveurs en parallèle. La capacité du système définit le nombre de clients maximal en attente. Dans certains systèmes les clients rebouclent et il y a un nombre maximal de clients dans le système. Dans notre cas où les clients sont des paquets de données, cette information n'a pas beaucoup d'intérêt. Enfin, la discipline définit l'ordre de traitement des paquets. La plus basique est "premier entré - premier sorti". Il existe encore d'autres variantes de files d'attente qui incluent d'autres paramètres, par exemple l'impatience des clients qui décident de quitter une file trop longue.

Propriétés d'une file d'attente Nous rappelons maintenant quelques propriétés fréquemment utilisées dans les files d'attente.

La première propriété est l'ergodicité. Par ergodicité, la moyenne temporelle du remplissage de la file d'attente converge vers la moyenne de l'état stationnaire de la file d'attente, qui est le point d'équilibre du remplissage de la file d'attente. Par la suite nous nous intéresserons principalement à l'étude de ces états d'équilibre.

La première est la loi de Little. Elle définit une relation entre le nombre de clients N en attente dans la file d'attente, le débit d'arrivée des clients λ dans le système et le temps d'attente moyen d'un client W.

$$E(N) = \lambda W \tag{3.1}$$

Ce résultat, valable dans un cas général, montre que le temps d'attente moyen d'un paquet dans la file d'attente est indépendant de l'ordre de transmission des paquets imposé par un ordonnancement (i.e. FIFO, *fair queuing*). L'avantage procuré aux paquets priorisés par une discipline quelconque est contrebalancé par la pénalisation des paquets non prioritaires. Nous pouvons donc faire abstraction de l'ordonnanceur dans notre étude de la congestion des files d'attentes.

Cette formule met en évidence que pour mesurer la congestion la relation entre le débit du flux entrant dans la file et son remplissage. Cette équation montre que cette métrique n'est pas accessible directement par observation de la sortie, mais qu'il faut absolument accéder à une variable interne à la file d'attente ou ajouter une hypothèse supplémentaire. Nous formulons donc hypothèse sur le flux entrant.

3.2.1.2 Estimation du temps d'attente sous hypothèse markovienne

Après avoir justifié l'ajout de l'hypothèse markovienne sur la loi d'arrivée des paquets, c'est-à-dire en supposant que la différence entre des instants d'arrivée de deux paquets successifs suit une loi exponentielle, nous montrons qu'il est possible d'estimer le temps d'attente en fonction du débit de sortie des paquets.

Agrégation de flux indépendants : une loi sans mémoire ? L'hypothèse markovienne est une simplification couramment utilisée dans la littérature des files d'attente.

Le théorème de Palm Khintchine montre que le nombre de paquets émis par N émetteurs indépendants et de loi stationnaire pendant un intervalle de temps donné converge vers une loi de Poisson. De façon équivalente, la loi des intervalles entre paquets consécutifs issus de ces émetteurs suit une loi exponentielle. Cette hypothèse simple correspond à beaucoup de situations, ce qui en fait une loi fréquemment utilisée pour modéliser les entrées d'une file d'attente. Néanmoins, ce théorème est valable mathématiquement sous des conditions très strictes, notamment :

- les émetteurs ont un comportement stationnaire ;
- l'absence d'un émetteur prédominant ;
- les émetteurs sont indépendants ;
- l'approximation n'est valable que pour une courte période de temps : des phénomènes périodiques seront mis en évidence dans certains cas si l'observation dure longtemps.

L'article [Lin06] explique cela avec plus de rigueur mathématique. En résumé, il faut un nombre d'utilisateurs indépendants suffisamment grand et observer pendant un intervalle de temps suffisamment faible par rapport à l'intervalle caractéristique qui sépare deux paquets d'un même flux pour considérer que l'écoulement est poissonien.

Nous supposons maintenant que l'hypothèse exponentielle - sans mémoire est acquise, ce qui simplifie considérablement l'étude des files d'attente.

PASTA La propriété PASTA découle de la loi exponentielle et de la propriété d'indépendance temporelle entre les paquets. Les calculs seront de ce fait largement simplifiés.

Une file d'attente est décrite par la variable aléatoire du nombre de paquets en cours de traitement, c'est-à-dire en attente ou en cours de transmission. L'espérance temporelle du temps de traitement W est définie par

$$E(W) = \lim_{u \to \infty} \int_0^u \frac{W(t)}{u} dt \tag{3.2}$$

tandis que la moyenne des temps de traitement des paquets est définie par l'équation

$$\bar{W} = \lim_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n} W_i \tag{3.3}$$

Dans un cas général, on a

$$E(W) \neq \bar{W} \tag{3.4}$$

Par exemple, si on considère que l'entrée de la file d'attente est un flux CBR dont le débit d_e est inférieur au débit de service d_s, donc $\bar{W} < E(W)$: le temps d'attente d'un paquet est toujours nul et inférieur à l'espérance du temps de traitement. En effet, en régime établi, chaque paquet entrant voit une file d'attente vide parce que le paquet précédent a été transmis.

Au contraire, si on considère un flux de paquets de même débit moyen d_e mais qui est très irrégulier, une rafale de paquet de débit localement supérieur au débit de

service d_s remplira la file d'attente et les paquets suivants seront pénalisés. Ainsi, on observera que $\bar{W} > E(W)$.

Selon la propriété PASTA, un flux poissonien vérifiera que $\bar{W} = E(W)$. Cette propriété découle de l'absence de mémoire du flux poissonien : dans un intervalle de temps donné, l'instant d'arrivée de chaque paquet suit une loi uniforme. Son temps d'attente moyen correspondra donc au temps moyen d'attente dans la file, comme l'indique la propriété.

Les explications de ces propriétés sont plus détaillées en 1.7 et 1.9 de [GSTH11].

L'égalité entre la moyenne des temps de transmission des paquets et la moyenne temporelle du temps de transmission d'un paquet nous permet de faire le lien entre temps d'attente moyen des paquets, remplissage temporel moyen et débit μ de sortie de la file d'attente :

$$W = \frac{E(N) + 1}{\mu} \tag{3.5}$$

Ce dernier résultat (3.5) et le précédent (3.1) peuvent être combinés pour éliminer le débit entrant ou le remplissage des paquets, mais nous ne pourrons pas estimer le temps d'attente sans l'un des deux. Une étude statistique ne suffit donc pas à détecter la présence de congestion de façon passive et nous continuons l'étude en formulant des hypothèses supplémentaires sur le fonctionnement des files d'attente.

3.2.2 Analyse de modèles de files d'attente

Les équations précédentes montrent qu'il est possible d'estimer sous des hypothèses raisonnables le temps passé dans une file d'attente à partir de la mesure du débit entrant et de la connaissance du paramètre de débit de la file d'attente. Dans le cadre d'une méthode de supervision passive, nous aimerions détecter la congestion uniquement à partir du flux sortant. Nous étudions dans un premier temps le modèle de file M/M/1. Cependant, il révèle rapidement ses limites et nous nous penchons ensuite sur des modèles à temps discret, qui apporteront davantage d'information par l'analyse des temps inter-arrivée des paquets IP.

3.2.2.1 Files M/M/1

Évaluation du temps d'attente moyen En combinant les formules précédentes de la loi de Little (3.1) et de la propriété PASTA (3.5), nous pouvons facilement estimer le remplissage d'une file d'attente en fonction de ses débits entrant λ et sortant μ. Notons que le débit observé en sortie de file sera $\mu_{\text{obs}} = \lambda$ car, en l'absence de pertes, les paquets sont conservés.

$$E(N) = \frac{\lambda}{\mu - \lambda} \tag{3.6}$$

$$W = \frac{1}{\mu - \lambda} \tag{3.7}$$

Ces équations sont souvent écrites dans la littérature en faisant apparaître $\rho = \frac{\lambda}{\mu}$.

Au regard des hypothèses précédentes, la file M/M/1 est au premier abord le meilleur modèle. Cette file d'attente modélise un flux de paquets de loi d'inter-arrivée exponentielle traité par un seul serveur de loi exponentielle.

Notons que, par conservation des paquets, c'est-à-dire tant que le débit entrant λ est inférieur au service μ, la loi des temps d'inter-arrivée des paquets en sortie de la file d'attente est aussi une loi exponentielle de débit λ.

La formule de Pollaczek-Khintchine généralise cette équation à tout type de service (file M/G/1) :

$$W = \frac{1 + C_B^2}{2} \frac{\lambda}{\mu(\mu - \lambda)} + \frac{1}{\mu} \tag{3.8}$$

Le temps de service est modélisé par une variable aléatoire S, telle que $E(S) = \mu$ et $C_B^2 = \frac{Var(S)}{E(S)^2}$. En conclusion, ce modèle permet d'estimer un niveau de congestion (ou taux d'occupation) par la mesure du débit sortant de la file d'attente à condition d'avoir une connaissance préalable du service et de sa gigue.

Ces formules sont incomplètes cependant car elles n'intègrent pas les pertes générées par une file d'attente. Pour aller plus loin nous devons donc modéliser plus finement la file d'attente et considérer sa taille limitée.

Évaluation du taux de pertes Dans une file d'attente, les paquets qui ne peuvent pas être transmis immédiatement sont stockés jusqu'à remplir un buffer de capacité C (modèle M/M/1/C). Les paquets qui arrivent alors que le système est rempli sont perdus. La file d'attente est modélisée par un vecteur P de taille $C + 1$ dont chaque élément p_i est la probabilité que la file d'attente contienne i paquets. Ce vecteur contient la distribution du nombre N de paquets à un instant t et nous définissons la matrice stochastique Π de taille $(C + 1) \times (C + 1)$ pour passer à l'instant $t + \delta$. La loi d'arrivée des paquets est supposée stationnaire, les probabilités de transition $\pi_{i,j} = P(N_{t+\delta} = i | N_t = j)$ sont constants et indépendants du temps.

Ainsi, la distribution à l'instant $t + \delta$ est calculée par $P_{t+\delta} = \Pi P_t$. La matrice stochastique est valable sur un pas de temps, par exemple la durée de transmission d'un paquet. Le temps de transmission d'un paquet n'est pas constant dans le mo-

dèle M/M/1 et on a plutôt recours au générateur infinitésimal Q associé à la matrice stochastique P.

Notons qu'il existe un unique distribution d'état à l'équilibre de la file qui vérifie $P^* = \Pi P^*$ et vers lequel convergera la distribution du nombre de paquets dans la file d'attente.

La solution dans le cas d'un modèle M/M/1/C dépend de la capacité C de la file d'attente et du ratio $\rho = \frac{\lambda}{\mu}$ entre débit entrant et taux de service. Elle est donnée par

$$P(N = n) = \frac{1 - \rho}{1 - \rho^{C+1}} \rho^n \tag{3.9}$$

Cette équation nous permet de calculer le taux de perte qui est la probabilité qu'un paquet soit rejeté, ou encore la probabilité que le système soit rempli. Le taux de perte est donc donné par

$$P(N = C) = \frac{1 - \rho}{1 - \rho^{C+1}} \rho^C \tag{3.10}$$

On peut lier ce taux de pertes au débit observé en sortie avec $\mu_{\text{obs}} = \lambda(1 - P(N = C))$. Il n'y a plus conservation des paquets à travers la file d'attente, et le débit observé en sortie est inférieur à celui entrant.

En $n = 0$ on a la probabilité que le système soit vide, donc le taux d'utilisation du serveur vaut

$$1 - P(N = 0) = 1 - \frac{1 - \rho}{1 - \rho^{C+1}} \tag{3.11}$$

Notons enfin que ces formules peuvent être étendues dans le cas d'une file d'attente M/M/1 en modélisant un vecteur de taille infinie, ce qui nous donne la distribution suivante :

$$P(N = n) = \frac{\mu - \lambda}{\mu} \left(\frac{\lambda}{\mu}\right)^n \tag{3.12}$$

Dans ce cas, il n'y a pas de pertes et on retrouve ainsi la distribution du temps d'attente dans une file qui est une loi exponentielle de paramètre $\mu - \lambda$:

$$w(t) = (\mu - \lambda) e^{-(\mu - \lambda)t} \tag{3.13}$$

Conclusion sur les modèles de file d'attente de la loi d'entrée est exponentielle Ces modèles montrent que

- sous l'hypothèse de loi exponentielle en entrée de file d'attente
- avec la connaissance de la loi du service et de ses paramètres,

- en mesurant la sortie de la file d'attente,

il est possible de retrouver

- le paramètre de débit de la loi exponentielle en entrée de la file d'attente,
- l'état interne de la file d'attente : temps d'attente dans la file dans un cas général, ainsi qu'une connaissance du taux de pertes et de la taille de la file d'attente dans le cas d'une loi exponentielle,

à partir des équations définies décrites précédemment.

Les résultats sont bien ceux attendus pour détecter la congestion. Cependant, les formules mettent en évidence l'importance de connaître la topologie, l'état du réseau et de bien caractériser les flux entrants des utilisateurs. Nous supposons que ces résultats pourraient être généralisés à d'autres lois d'entrée de la file d'attente à condition de réaliser des calculs complexes sur les distributions. Cette stratégie impliquerait de connaître à l'avance les utilisateurs, et caractériser leurs comportements variables. Cette méthode semble donc difficile à mettre en œuvre et son succès sera incertain.

D'autre part, la loi exponentielle a un avantage calculatoire et un inconvénient sur les informations récupérées. La loi exponentielle permet d'étendre facilement les calculs à un réseau d files d'attente. En effet, l'agrégat de flux dont les lois d'inter-espacement des paquets sont exponentielles suit une même loi d'inter-espacement des paquets exponentielle. Supposer des lois d'entrée et de service exponentielles simplifie largement les calculs, mais alors la topologie n'a plus d'impact sur le mélange des flux. L'inconvénient de cette loi est qu'elle transporte peu de données. Elle n'est caractérisée que par un paramètre et nous contraint à la mesure des débits. Observer la distribution des paquets en sortie du réseau sous cette hypothèse ne nous donne donc aucune information sur la topologie.

Sous l'hypothèse de la loi exponentielle, il faut connaître le système préalablement et observer absolument tous les paquets sortants. L'absence d'observation de paquets perdus ou ayant été déroutés introduira une erreur d'estimation des débits, pertes et donc des temps d'attente dans chaque file du système, réduisant ainsi la possibilité de détecter un goulot d'étranglement. Ainsi, les hypothèses de flux poissonien réduisent l'étude du réseau à de simples mesures de débits, largement insuffisantes pour détecter la plupart des anomalies qui pourraient survenir dans un réseau réel, où l'observation du trafic est très partielle.

3.2.2.2 Flux à temps discret

Nous avons montré que le modèle M/M/1 et tous les autres modèles à entrée Markovienne apportent peu d'information sur la topologie. Nous en déduisons que pour retrouver l'état du réseau à partir de l'analyse de mesures d'instants d'arrivée de paquets, il faut ajouter une autre information.

Nous éliminons d'office l'idée de mieux caractériser la distribution des paquets en entrée du réseau, car cette idée serait impossible à mettre en œuvre en pratique. Néanmoins nous pouvons essayer de caractériser une partie des utilisateurs. Ainsi, nous proposons dans un premier temps d'étudier les flux constants qui traversent le réseau. Nous testons ensuite une seconde hypothèse, de changer le type de service et marquer la dépendance entre l'état de la file d'attente et l'inter-espacement en sortie de file. Nous étudions donc le changement de paradigme proposé par O. Brun avec le modèle de file M/D/1.

Flux constants L'idée développée serait d'utiliser des flux dont on détecte le modèle pour recueillir des informations sur le réseau. Concrètement, peut-on détecter des anomalies à partir de mesures sur des flux Constant Bit Rate (CBR) ?

La démarche se rapproche des scénarios de sondes de test proposés par Rubenstein puis Hayes : dans les deux cas on introduit une information temporelle dans des paquets sonde pour identifier l'état des files d'attente.

L'article [BBG06] met en œuvre un flux à débit constant qui traverse plusieurs files d'attente successives et est à chaque fois mis en concurrence avec un agrégat de flux de loi d'inter-arrivée exponentielle. Ces files d'attente accroissent progressivement la gigue du flux et le résultat est une formule complexe qui montre que la gigue totale mesurée est composée de la somme des contributions de chaque file traversée. D'autres papiers se sont intéressés à la gigue ajoutée à des flux CBR, par exemple l'article [AHA+21] avec des files M/D/1 et l'étude de [DGS12], basée des files d'attente de modèle M/M/1.

Tous ces papiers permettent d'établir un lien entre la gigue ajoutée et l'état des files d'attente, mais introduisent un terme supplémentaire lié aux autres flux dont la gigue est inconnue. Ce terme supplémentaire implique de caractériser l'agrégat des autres flux et ne simplifie pas vraiment le problème. La présence de flux CBR ne nous aidera donc pas et nous passons à la seconde hypothèse sur le service de la file d'attente.

Le modèle M/D/1 La notation de Kendall ne permet pas de rendre compte du fonctionnement précis de la file d'attente car elle introduit une indépendance entre la loi en entrée et la loi de traitement des paquets.

Ainsi, on peut comprendre la notation M/D/1 comme une file d'attente dont la loi d'arrivée des paquets est un flux poissonien (inter-espacement des paquets en entrée de loi exponentielle) et dont le service est déterministe, au sens où chaque paquet est traité en un temps déterminé. Bien entendu, dans ce cas la distribution des inter-espacements des paquets en sortie est une loi exponentielle.

L'approche Brun est originale de ce point de vue car ce n'est pas la loi du temps de service qui est caractérisée, mais la loi de la distribution des inter-espacements en sortie. L'article [GBG02] décrit ainsi une file d'attente qui teste la présence de paquets en attente à intervalle régulier, et en transmet un le cas échéant. Contrairement aux modèles précédents, ce type de service définit une loi conditionnée sur le remplissage de la file d'attente : si la file est remplie, le temps de transmission sera de la durée de l'intervalle de transmission de la file d'attente ; au contraire si un paquet arrive entre deux instants de transmission et trouve la file vide, son temps de transmission sera réduit à l'intervalle de temps qui le sépare de la prochaine échéance.

Le modèle M/D/1 définit une file d'attente de taille finie dont les paquets sont émis à intervalles réguliers. De la même façon que pour l'étude du modèle M/M/1/C, ce modèle est représenté par un vecteur d'état du remplissage de la file d'attente, et une matrice de transition permet d'estimer le remplissage de la file d'attente en un intervalle de temps d'émission d'un paquet.

Ce modèle met en avant met en avant la présence de cet intervalle de temps caractéristique dans la distribution des inter-espacements du flux en sortie de file et une baisse de l'entropie (nous y reviendrons par la suite) d'un agrégat au passage de la file. Ces deux aspects ne peuvent cependant pas être exploités car distribution des inter-espacements des paquets en sortie de file est discrète par construction. Ainsi, la différence de temps entre deux instants d'émission sera toujours un multiple de la périodicité de la file d'attente.

Comparé aux modèles M/M/1 précédents, ce modèle est le premier à introduire de façon mathématique une explication de la présence de la durée de transmission standard d'un paquet dans la distribution des paquets en sortie de file d'attente. Néanmoins cette valeur d'inter-arrivée est beaucoup trop présente dans la distribution des inter-arrivées modélisée par rapport à ce que nous mesurons en pratique.

D'autre part, ce modèle possède un second inconvénient qui le rend impossible à utiliser pour modéliser un réseau. En effet, il suppose que la loi de la distribution des inter-arrivées en entrée est exponentielle, mais que sa sortie ne l'est pas. Cette seconde particularité rend le modèle impossible à utiliser dans un réseau composé de plusieurs files d'attentes.

Ce modèle ouvre une piste pour exploiter mathématiquement les inter-espacements

caractéristiques du temps de transmission d'un paquet. Malheureusement, ce modèle souffre de divers défauts qui rendent son utilisation impossible pour déterminer des informations sur le réseau.

Nous reprendrons au chapitre suivant l'idée introduite ici de variable latente de présence de paquet dans la file d'attente afin de détecter la saturation de la file d'attente.

3.3 Le Simulateur Hypatia

Les modèles analytiques mathématiques montrent rapidement leurs limites pour l'analyse d'un système complexe. Nous avons donc utilisé Hypatia [KBÁ+20], qui est un simulateur de mégaconstellation à événements discrets basé sur NS3. Il simule des flux de paquets internet qui transitent à travers une constellation de satellites et sont relient deux points du globe. Ce simulateur à temps discret est à notre connaissance le premier et le seul en libre accès qui modélise une mégaconstellation et implémente un algorithme de routage des paquets à travers celle-ci.

3.3.1 Implémentation

Hypatia est composé du simulateur NS3 complété par des modules spécifiques et un ensemble de scripts permettant de définir les paramètres de la simulation. Ces scripts ont été largement remaniés dans notre version d'Hypatia. Nous regroupons les principaux paramètres du simulateur en deux classes. La première décrit le réseau, c'est-à-dire les positions, les liens et caractéristiques de ces liens dans le réseau. La seconde les paramètres de simulation. Nous suivrons ce plan pour présenter le fonctionnement du simulateur puis des points importants de la simulation.

3.3.1.1 Définition d'un réseau de satellites

Dans son état actuel, Hypatia permet de simuler une constellation simple, ou plutôt une unique *shell*. La communication entre satellites dont les orbites sont à des altitudes différentes n'est pas à l'ordre du jour.

Les différents objets Il y a quatre classes d'objets dans notre version d'Hypatia :

- les satellites, en orbite basse autour de la Terre ;
- les utilisateurs ;
- les stations sol ;

- les serveurs et routeurs au sol.

Les satellites en orbite basse suivent un mouvement quasi-circulaire autour du globe. Les positions des satellites sont régulièrement publiées au format TLE dans des bases de données publiques comme Celestrak. Leur mouvement est implémenté à partir de différentes bibliothèques de propagation orbitale. Le code python repose sur le module python-sgp4 qui implémente le modèle de perturbations simplifiées SGP4. La précision de ce modèle sera d'autant meilleure que la date à prévoir est proche de la date du TLE. Les satellites décrivent des orbites dont les plans orbitaux sont répartis à intervalles réguliers autour de la Terre. De nombreuses constellations déployées (OneWeb, Starlink) ou en projet sont disponibles. Nous nous baserons ici une proposition de constellation pour le projet Telesat Lightspeed, une constellation de 28 plans de 27 satellites à 700km d'altitude. L'inclinaison orbitale est de 65°, assurant une bonne couverture au-dessus des zones les plus habitées. Cette couche orbitale ne suffit pas à assurer une couverture mondiale, celle-ci devra être assurée par une deuxième couche couvrant les pôles.

Dans cette version d'Hypatia, les utilisateurs sont positionnés autour des 100 villes les plus peuplées du monde, suivant une loi Kent afin de les disperser un peu autour de ces villes. Cette configuration est une méthode simplifiée pour générer des utilisateurs selon la densité de population. Enfin, des stations sol sont placées dans les plus grandes villes et des serveurs placés en périphérie, à faible distance (quelques millisecondes de délai).

Liens du réseau Les objets décrits précédemment sont reliés entre eux afin de créer un réseau schématisé figure 3.1 qui illustre le fonctionnement d'Hypatia.

Les plus simples sont les liens sols. Une station sol (*gateway*) est reliée à un ou plusieurs serveurs. ces liens du segment sol sont statiques, définis avec un débit élevé et une latence faible.

À l'inverse, les liens sol-espace sont beaucoup plus complexes à gérer. Il en existe plusieurs types, définis entre un satellite et un utilisateur ou station sol. Hypatia permet désormais des liens asymétriques permettant de distinguer les utilisateurs, dont les débits sont limités par rapport aux satellites ou aux stations sol. Ces dernières ont un débit et une portée plus importants. Typiquement, dans nos simulations, un utilisateur ne communique qu'avec les 2-3 satellites les plus proches dont l'élévation dépasse 35° tandis que la station sol en a 5 à disposition et peut atteindre des satellites jusqu'à 10° d'élévation au-dessus du sol. La connexion entre un objet sol et un satellite est définie dans le simulateur par l'élévation du satellite relativement à l'objet au sol ou, de

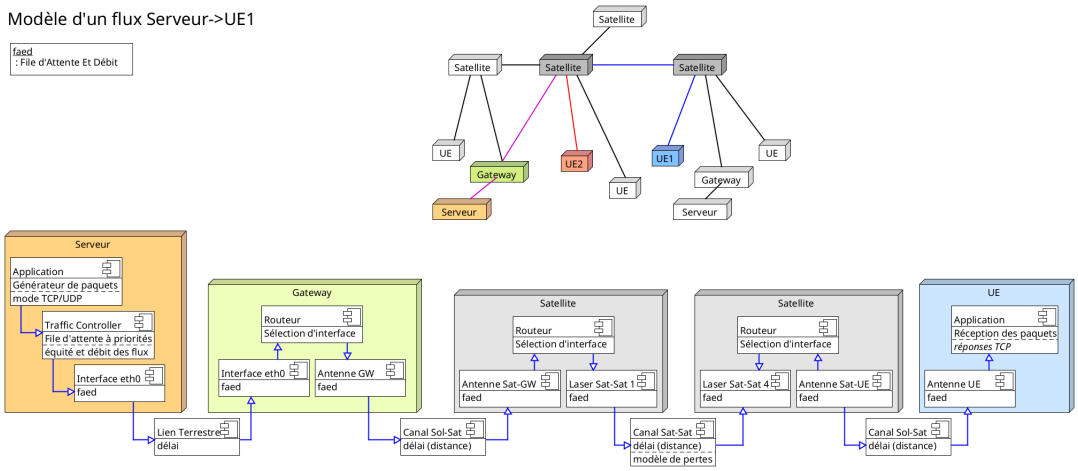


Figure 3.1 : Représentation des interfaces réseau

façon équivalente, par l'angle d'ouverture de l'antenne du satellite. La latence dépend simplement de la distance entre les objets, donc de l'élévation du satellite par rapport à son client au sol. Nous n'implémentons pas de modèle de télécommunication en fonction de la portée, mais le débit de l'agrégat des flux sortant d'une interface est bridé par son débit maximal, indépendamment de la destination. Ainsi, plusieurs satellites se partagent la bande passante d'une station sol. En résumé, un utilisateur possède une unique interface vers les satellites, une station sol au moins 2 interfaces (une vers les satellites, les autres à destination d'un ou plusieurs serveurs) et un satellite 6 interfaces : une vers les utilisateurs, une vers les stations sol et 4 vers 4 satellites voisins que nous décrivons maintenant.

Ces dernières interfaces constituent les liens inter-satellites, essentiels pour assurer une couverture mondiale. Le premier système Starlink n'en disposait pas et chaque paquet internet reçu par un satellite devait être retransmis immédiatement au sol. À l'instar d'autres constellations LEO ou MEO comme O3b, les satellites communiquent entre eux par radio ou en optique. Les liens optiques offrent un débit beaucoup plus élevé en espace libre, mais nécessitent une bonne précision de pointage et donc un contrôle d'attitude très fin. Les progrès ont permis de déployer cette technique plus largement. Ces liens vérifient simplement un débit maximal, constant et défini sur la durée de la simulation, ainsi qu'une latence mise à jour en fonction des positions des satellites.

Par défaut un satellite est connecté à quatre autres satellites : deux satellites voisins sur la même orbite, un sur l'orbite de droite et un autre sur l'orbite voisine de gauche. Ces voisins sont des satellites survolant le globe dans le même sens. En effet,

la connexion avec un satellite parcourant la Terre dans le sens opposé s'avérerait peu efficace à cause des vitesses relatives élevées.

À chaque passage par son apogée, les orbites adjacentes d'un satellite sont interverties. Les communications avec ces satellites adjacents ne seront plus possibles et cesseront le temps de recaler les satellites. Les liens sont donc désactivés temporairement.

Diverses causes peuvent amener à des pertes de paquets (bagotage des liens intersatellites, interférences ...). Des modèles de pertes, tel que celui de Gilbert-Elliott, peuvent être activés sur les certains liens.

3.3.1.2 Le routage des paquets dans la constellation

La mobilité des satellites impose de résoudre deux problèmes de routage. Le premier est un problème de modélisation du réseau qui consiste à régulièrement mettre à jour les liens du réseau. Le second paragraphe traite réellement du routage et du choix d'un chemin optimisé.

La mise à jour des liens du réseau Les interfaces d'Hypatia sont très simplifiées. Une fois le paquet passé par la file d'attente et en attente de transmission effective, les positions des correspondants sont mises à jour ainsi que le délai du lien par un calcul de la distance qui sépare les correspondants. L'instant de réception est alors planifié. Le modèle de perte peut entrer en jeu à ce moment et décider de la perte d'un paquet. Les liens entre deux objets sol ou deux satellites sont fixes et n'évolueront pas en fonction du temps (sauf anomalie). En revanche, la gestion des interfaces dans un contexte sol-espace est bien plus difficile car les correspondants changent constamment. Le réalisateur d'Hypatia a testé différentes solutions. La première a été de créer un réseau spécifique pour chaque paire équipement-sol/satellite. Cette solution, basée sur l'activation/désactivation d'interfaces, complexifie inutilement le système avec un surnombre d'interfaces.

Le développeur d'Hypatia a imaginé de contourner la difficulté technique de simulation de connexion/déconnexion des interfaces par la création d'un bus unique par lequel passent toutes les communications sol-espace. Périodiquement, au terme d'un pas de temps défini par défaut à 2s, le plan de routage du réseau est mis à jour et un algorithme de routage recalcule toutes les routes des différents flux. Dans le simulateur, les connexions entre interfaces sont maintenues, mais les flux de paquets suivent les instructions des tables de routage et n'empruntent pas de chemin impossible physiquement.

En pratique, tous les nœuds puisent les informations de routage dans la même base de donnée pré-calculée par Hypatia avant d'effectuer la simulation. Hypatia n'intègre donc ni protocole de routage, ni de *handover*, ni de gestion des clients comme dans un réseau mobile. Le contrôle est donc opéré uniquement par AQM et TCP pour les flux concernés. Tous les utilisateurs étant définis avant le lancement d'une simulation, le calcul des tables de routage peut intégrer la disposition des utilisateurs afin d'éviter de saturer certains liens.

L'algorithme de routage Le routage dans une constellation est un problème complexe dont nous effleurons ici quelques aspects. Les algorithmes de contrôle de congestion (notamment TCP) supportent mal les délais et la gigue et on s'attend à des performances dégradées lorsque la latence et la gigue du réseau sont plus élevées.

Afin de maximiser les performances de TCP et minimiser ce souci, le routage choisi initialement est un plus court chemin. Celui-ci est contraint car un flux de paquets ayant atteint la constellation ne doit pas redescendre sporadiquement par un quelconque utilisateur ou station sol avant de remonter vers un autre satellite et ainsi "sauter" une orbite. Le routage par plus court chemin peut donc être décomposé en 3 morceaux : un tronçon source-premier satellite, un tronçon dans la constellation et enfin le lien ente le dernier satellite et la destination. À l'intérieur de la constellation l'algorithme de Floyd-Warshall donne une solution de routage par plus court chemin. Cependant, la distance entre satellites raccourcit près des pôles, et les utilisateurs émettent beaucoup de paquets à grande distance, ces flux passeront par les pôles formant un rond-point. De même, il est préférable pour ces flux longue distance de passer au-dessus des grandes de passer par des satellites critiques reliés à des stations sol saturées. Une autre stratégie de routage implémentée dans [GPL+22] résout un problème d'optimisation de minimisation de la surcharge des liens. Les résultats ont montré des progrès spectaculaires pour les flux UDP par rapport à un algorithme de plus court chemin. Cependant, ces résultats ne sont pas aussi probants dans le cas de flux TCP, probablement à cause des délais supplémentaires causés par ces chemins détournés. Aussi nous n'utiliserons pas dans ces expériences de routage à optimisation de charge réseau.

En revanche, le choix du premier et du dernier satellite s'avère critique. Le graphe de routage dans la constellation peut être représenté par un tore. Deux satellites proches géographiquement et allant dans la même direction sont voisins dans la topologie. En revanche, deux satellites proches géographiquement mais opposés en direction seront très éloignés dans la topologie. Ainsi, pour une paire source-destination dont la longitude est proche, le choix du satellite le plus proche amène à choisir un satellite montant

ou descendant du pôle Nord sans distinction. Source et destinations ont statistiquement une chance sur 2 de se retrouver très éloignés (si les satellites choisis tournent en sens contraire) dans la topologie. Dans ce cas, l'algorithme fait tourner le flux de paquet autour de la Terre pour rejoindre le satellite circulant en sens opposé. Cela augmente considérablement le délai, entraîne une chute des performances, et provoque une chute de performance car, pour minimiser la distance, tous ces flux se retrouvent à tourner autour des pôles. L'étude [ABH+22] a pointé du doigt cette faiblesse d'Hypatia et proposé une correction en évaluant les k (à définir) plus proches satellites. Plus simplement, nous réunissons tous les nœuds (satellites, utilisateurs, stations sol et serveurs) dans un unique graphe dont les arêtes sont les liens techniquement possibles, pondérés par le délai entre deux nœuds. Les liens sol-satellites sont largement pénalisés pour dissuader les flux de paquets de redescendre au sol.

En réalité, ce choix du premier satellite est beaucoup plus complexe. En plus des considérations de puissance des émetteurs, portée du signal et débit utile, il faudrait prendre en compte le temps restant de visibilité du satellite. La durée d'une fenêtre de communication dépend de l'altitude du satellite et de la directivité des antennes. Avec des satellites LEO, la fenêtre de communication dure quelques minutes. Il peut donc être judicieux de sélectionner un satellite plus éloigné mais en approche pour limiter le nombre de handovers.

Notons que dans Hypatia le calcul des route est centralisé, tous les flux sont connus à l'avance ainsi que les positions géographiques des sources et destinations. Ce ne sera pas toujours le cas dans un vrai réseau et les résultats de performance d'Hypatia ne peuvent pas être exploités tels quels. En revanche, les hypothèses de construction du réseau, même parfois peu réalistes, suffiront à étudier les problèmes de liens congestionnés ou de pertes aléatoires sur des liens défectueux.

3.3.2 Étude de la congestion dans la constellation

Après avoir décrit le fonctionnement du simulateur Hypatia, nous effectuons des expériences pour mettre en avant des anomalies réseau. L'étude présente un cas de congestion qui nous permettra d'évaluer les capacités d'Hypatia. Nous commençons par décrire la préparation de l'expérience avant d'en interpréter les résultats.

3.3.2.1 Description des expériences : une montée en charge du réseau

La montée en charge du réseau se fait en deux temps. Nous analyserons l'utilisation des liens à l'issue d'une première expérience contenant un faible nombre d'utilisateurs, puis en déployons un nombre beaucoup plus important pour mettre en évidence la

congestion. Nous commençons par donner les détails techniques des utilisateurs, puis présentons les paramètres du réseau. Ces derniers restent inchangés dans les expériences.

Campagne d'expériences Hypatia est surtout intéressant pour tester différentes architectures et configurations faisant ressortir l'influence des différents paramètres. Des campagnes de tests ont donc été lancées faisant varier l'architecture du réseau, l'importance du routage, le nombre d'utilisateurs et les débits des liens.

Une campagne d'Hypatia est un ensemble d'expériences similaires où quelques paramètres seulement varient, ce qui permet de comparer différents réglages. Nous présenterons les résultats d'une simple campagne de deux expériences faisant varier le nombre d'utilisateurs de 200 à 950. Les autres paramètres de la constellation ne varient pas. Nous nous intéresserons plus particulièrement dans le cas présent à la congestion provoquée par un surnombre d'utilisateurs. Ici les utilisateurs émettront des flux TCP vers différents serveurs. Les statistiques collectées sur les liens incluent le nombre et la taille des paquets transmis, permettant de mesurer l'activité d'une interface. Les simulations de TCP permettent de recueillir en plus des fichiers de logs d'évolution du RTT, l'avancement de la transmission de données et l'évolution de la fenêtre de congestion du protocole TCP.

L'algorithme de contrôle de congestion *New Reno* est appliqué sur tous les flux de la simulation. Seules des files *bfifo*, sans ordonnanceur sont utilisées, les tailles des buffers varient selon les expériences.

NS3 est un simulateur à temps discret qui maintient une file d'événements. Il n'est pas possible de paralléliser ces événements, ce qui implique des temps de simulation très longs. Nous présentons les résultats d'une très courte expérience de 10 s. Arrivé à terme de la période de temps, nous considérons que le réseau est dans un état suffisamment stable pour mesurer la congestion dans le réseau et déterminer les emplacements des goulots.

Paramètres des liens Hypatia permet de définir les paramètres de lien suivants :

1. le débit (fixe, limité par l'interface émettrice) ;
2. une *queuing discipline* (qdisc), par exemple une simple file FIFO ou FQ-Codel ;
3. la latence, constante dans les liens terrestres ou variable en fonction des positions des correspondants ;
4. un modèle de pertes de paquets pour simuler des liens défectueux.

Le choix des paramètres des liens est rendu difficile pour plusieurs raisons :

- les opérateurs de constellations dévoilent peu leurs caractéristiques ;
- actuellement les performances évoluent rapidement avec le temps et les nouvelles générations de satellites ;
- Hypatia n'est pas complètement parallélisable, des expériences à haut débit proches de la réalité semble impossibles à réaliser pendant une durée convenable. Il faut donc réduire les débits.

Pour toutes ces raisons, nous avons choisi arbitrairement les valeurs suivantes pour les différents liens du système.

Source	utilisateur	satellite	autre objet sol
Débit (Mbps)	6	70	600

TABLE 3.1 : Débits des liens

La définition de ces valeurs détermine les points faibles du réseau. Le lien sol est surdimensionné afin que le point limitant du réseau soit dans la constellation et non autre part. Aussi nous n'observons que le trafic en provenance des utilisateurs. Dans la simulation effectuée, le trafic descendant vers les utilisateurs est limité au strict minimum. Ainsi il n'affecte pas le trafic montant des utilisateurs.

3.3.2.2 Les goulots se rapprochent du cœur lorsque la congestion apparaît

L'expérience met en œuvre deux situations. Dans le premier cas, les utilisateurs peu nombreux sont bridés par le débit de leurs interfaces. La seconde situation met en œuvre un grand nombre d'utilisateurs. Les performances individuelles baissent, et nous montrons que les goulots d'étranglement se sont déplacés à proximité de la station sol.

Cas avec peu d'utilisateurs La première simulation met en œuvre 200 utilisateurs qui émettent des flux TCP. Une image de l'utilisation des liens dans la constellation est montrée figure 3.2.

Dans cette représentation, les liens non utilisés ne sont pas affichés. De 0 à 90% d'utilisation, les liens suivent un dégradé allant du vert au rouge. Au-delà, ils deviennent noirs jusqu'à 100% d'utilisation. La figure montre donc que les liens les plus utilisés sont ceux reliant les utilisateurs aux satellites. Les utilisateurs sont donc principalement bridés par leur interface. Le tableau 3.2 nous donne des chiffres plus précis :

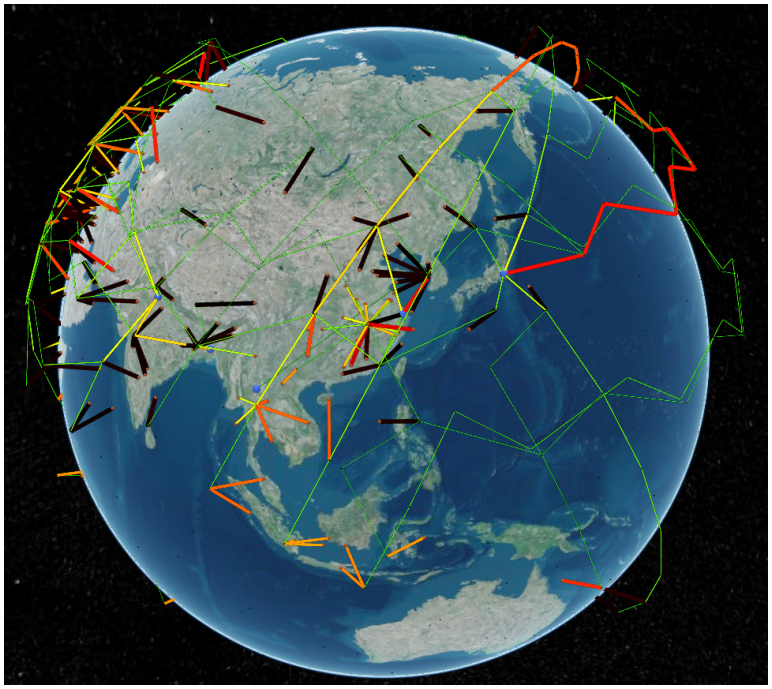


Figure 3.2 : Utilisation des liens de la constellation pour 200 utilisateurs

le taux d'utilisation est calculé par la somme des temps de transmission des paquets ayant transité sur le lien pendant un intervalle de temps de 300 ms. La quatrième colonne du tableau indique la fraction de liens ayant un taux d'utilisation supérieur à 90%. Les liés mesurés sont uniquement les liens actifs pendant la période considérée. Leur nombre est donné dans la 5$^{\text{ème}}$ colonne. Cette première expérience avec un faible nombre d'utilisateurs confirme que les liens *utilisateur→satellite* sont ceux limitant dans la configuration donnée, la moyenne de leurs taux d'utilisation est de 80%, et plus de 50% des liens sont utilisés plus de 90% du temps.

source	**destination**	**taux d'utilisation**	**fraction utilisation> 90%**	**nb de liens**
utilisateur	satellite	0.7996	0.5200	200
satellite	satellite	0.0833	0.0000	524
satellite	station	0.4508	0.1071	28
station	routeur	0.2262	0.0000	7
routeur	serveur	0.0609	0.0000	26
serveur	routeur	0.0012	0.0000	26
routeur	station	0.0044	0.0000	7
station	satellite	0.0044	0.0000	7
satellite	utilisateur	0.0135	0.0111	90

Table 3.2 : Utilisation des différents types de liens, cas avec 200 utilisateurs

Les taux d'utilisation des autres types de liens sont bien plus bas, ce qui nous indique que dans cette configuration, les goulots d'étranglement du réseau sont les utilisateurs eux-mêmes, chaque utilisateur est bridé à sa performance habituelle.

Cas avec un grand nombre d'utilisateurs La situation change radicalement en augmentant le nombre d'utilisateurs comme illustré figure 3.3.

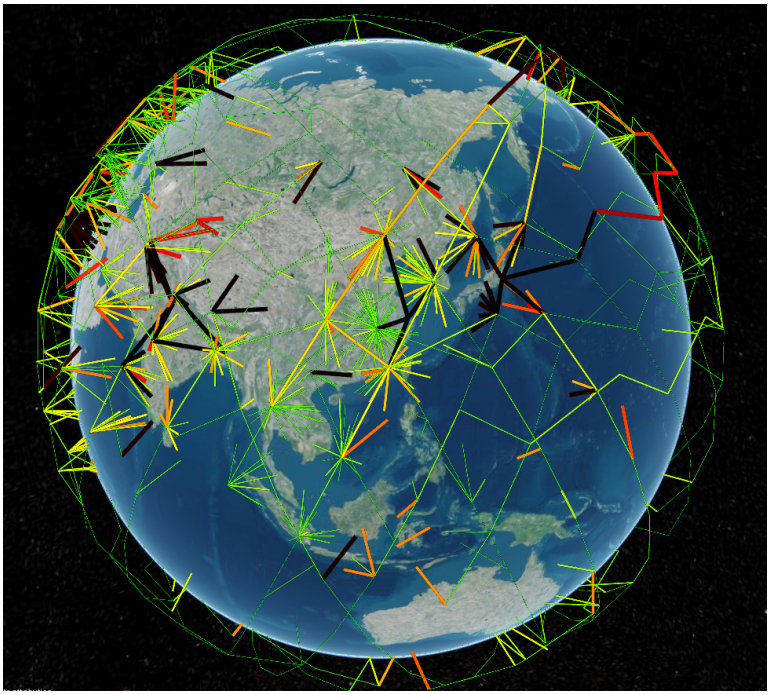


FIGURE 3.3 : Utilisation des liens de la constellation pour 950 utilisateurs

Cette fois, les liens congestionnés sont les liens *satellite→station*, comme nous le constatons autour des stations de Shanghai et Tokyo. Ces liens sont utilisés en moyenne à 85% et près de 77% de ces liens sont utilisés à plus de 90%, comme indiqué dans la table 3.3. En augmentant le nombre d'utilisateurs, le fonctionnement du réseau change complètement. Désormais les liens utilisateurs sont en moyenne moins sollicités tandis que la congestion se forme autour des stations sol, les satellites étant les points limitants du réseau.

Ici, les utilisateurs du réseau expérimentent une situation dégradée, leur débit chute de moitié. Remarquons que les liens satellites sont à peine plus chargés. Il y a plus de liens inter-satellites utilisés (1028 comparé à 512 précédemment), mais le taux d'utilisation moyen de ces liens reste bas. Nous observons sur la figure quelques exceptions, notamment un lien à destination de Tokyo en provenance d'un satellite au-dessus du Pacifique.

source	destination	taux d'utilisation	fraction utilisation> 90%	nb de liens
utilisateur	satellite	0.3368	0.0645	946
satellite	satellite	0.0717	0.0117	1028
satellite	station	0.8517	0.7692	26
station	routeur	0.4528	0.0000	7
routeur	serveur	0.1057	0.0000	30
serveur	routeur	0.0021	0.0000	30
routeur	station	0.0091	0.0000	7
station	satellite	0.0091	0.0000	7
satellite	utilisateur	0.0255	0.0229	218

Table 3.3 : Utilisation des différents types de liens, cas avec 950 utilisateurs

Ces expériences illustrent le résultat assez prévisible qu'en fonction du nombre d'utilisateurs et des différentes capacités des liens, la congestion se produira à des endroits différents. Ici les liens *satellite→sol* ont un débit de 70 Mb/s. Rappelons qu'il y en a maximum 5 reliés à une station dont le lien avec le routeur est limité à 600 Mb/s, celui-ci est largement surdimensionné. En choisissant des valeurs différentes, ce dernier lien aurait pu être congestionné de préférence au premier. Le gestionnaire du réseau est donc maître, dans une certaine mesure, de l'emplacement des liens limitants du réseau et donc des goulots et de la congestion.

Enfin, nous voyons dans l'étude de la constellation que nous pouvons restreindre à l'étude des stations sol individuellement. Il suffira de se placer sur les liens à plus gros débit pour décomposer l'étude du réseau en autant de sous-problèmes. On pourra s'attendre à des conditions similaires dans chaque cas : goulot proche de la station ou des utilisateurs.

3.3.3 Conclusion sur l'usage d'un simulateur

Nous dressons finalement un bilan d'Hypatia : sa force consiste en la possibilité de modéliser facilement un réseau complexe. Cependant, la lenteur de la simulation nous bride dans l'étude des réseau et nous préférerons passer à l'émulation. Nous n'avons pas eu de mauvaise surprise quant au fonctionnement de TCP dans la constellation simulée, les performances sont restées similaires à celles d'un réseau statique. Surtout, nous y voyons une opportunité pour simplifier encore le réseau et générer des données dans des réseaux simplifiés.

3.3.3.1 Avantages et limites d'Hypatia

Hypatia a été développé pour observer le fonctionnement d'une mégaconstellation au niveau des couches réseau et transport. Hypatia, basé sur NS3, implémente ces couches de façon précise, tandis que les couches plus basses (comme les modèles de propagation) ou plus hautes sont largement simplifiées ou inexistantes et évite des calculs inutiles. D'autre part, le simulateur NS3 a l'avantage de pouvoir ajouter des logs adaptés facilement et partout où on en a besoin, ce qui permet de bien comprendre tout ce qui se passe dans le réseau.

En revanche, plusieurs limites issues de NS3 jouent en défaveur d'Hypatia. Le gros inconvénient d'Hypatia est sa lenteur. Son fonctionnement par événements discrets, héritage de NS3, le rend très précis mais très lent car les opérations ne peuvent pas être parallélisées facilement. Cela implique de réduire la capacité des liens entre les équipements et in fine à des simulations moins réalistes.

Le second est que NS3 n'implémente pas d'identifiant unique des paquets. Un tel identifiant peut être ajouté aux paquets UDP et permet de connaître les causes des pertes avec précision. Dans le cas de TCP, c'est beaucoup plus compliqué, il n'est pas possible d'ajouter un tel champ car TCP retransmet le même paquet sur demande, sans nouvel identifiant. Afin de connaître avec précisions les causes des pertes de paquets, il faut donc utiliser une combinaison de variables : numéro de segment, numéro de ACK, numéro de séquence, voire l'option *timestamp* quand cela est possible. Des ambiguïtés subsistent et nous avons des difficultés à les résoudre totalement.

Enfin, l'avenir d'Hypatia semble compromis car il est bloqué sur une ancienne version de NS3 qui, contrairement aux versions plus récentes, implémente le mouvement des satellites. Pour faciliter la mise à jour d'Hypatia, un module de mouvement des satellites devra donc être réimplémenté.

3.3.3.2 Résultats obtenus

Hypatia a été développé avant tout pour obtenir une idée de la capacité et des performances d'un réseau basé sur une mégaconstellation. Il a été utilisé principalement au début de cette thèse d'abord pour découvrir les réseaux puis comme un outil de simulation d'un réseau d'abord bien réglé puis subissant de la congestion ou des pertes sur certains liens.

Notre étude a d'abord porté sur l'analyse des temps inter-arrivée des flux individuels TCP ou UDP. Ainsi :

- on a su détecter des pertes dans un flux UDP déterministe (il suffit d'identifier

la périodicité des paquets de ce flux) ;

- on n'a pas su détecter des pertes dans un flux aléatoire dont l'inter-espacement en entrée du réseau est de loi exponentielle. Des pertes aléatoires ne modifient pas dans ce cas précis la loi d'arrivée des paquets mais seulement les paramètres de cette loi. En rappelant que la loi d'un agrégat converge vers une loi exponentielle (voir §3.2.1.2), nous en déduisons qu'il est impossible d'identifier la présence de pertes dans un agrégat de flots aléatoires dans un cas général ;

- on n'a pas su identifier de façon fiable les pertes dans un flux TCP. Nous n'obtenons pas de meilleurs résultats que l'état de l'art, qui à notre connaissance se résume à l'analyse de la phase *slow start* (par exemple, voir [TFA+24]).

Diagnostiquer la perte de paquet (l'identifier puis déterminer sa cause : lien défectueux ou congestion) est donc un exercice difficile et semble mener à une impasse. Les résultats des expériences précédentes incitent à partir dans la voie de la détection des goulots du réseau, et plus particulièrement leur localisation.

Dans nos simulations, nous avons choisi de connecter la plupart des utilisateurs (90%) à un serveur proche. Par construction, les flux internet émis par les utilisateurs convergent vers la station la plus proche. Une station sol a donc une aire d'influence contenant les satellites alentour et capte la majorité dans flux entrant dans cette zone. Nous pouvons donc considérer qu'un réseau est la somme de l'ensemble de ces sous-réseaux et simplifions dans la suite l'étude à une unique station sol avec son voisinage.

3.4 Émulation par conteneurs Docker

Après avoir débuté la modélisation d'une mégaconstellation avec de simples équations mathématique, puis poursuivi par un simulateur complexe, nous aboutissons au modèle intermédiaire de l'émulateur. Nous présentons son fonctionnement puis les expériences que nous y avons réalisées et dont les données seront analysées dans le chapitre suivant.

3.4.1 L'émulateur Dockem

Ce simple émulateur (dans le dépôt git privé de l'ENAC) est un simple script permettant d'automatiser le déploiement d'un petit réseau émulé avec des conteneurs docker. Nous motivons les raisons de ces choix, puis entrons dans les détails de l'implémentation. Dans un premier temps, le script lit les paramètres de déploiement du réseau des conteneurs dockers, puis les actions qui y seront effectuées par les utilisateurs.

3.4.1.1 Les choix techniques de l'émulateur

Deux choix techniques forts ont été réalisés dans dockem. Le premier est de déployer de petits réseaux et non de grandes mégaconstellations. Nous justifions ce choix par les résultats des expériences précédentes réalisées avec Hypatia. Le second choix est plus technologique, nous utilisons une approche par conteneurs docker.

Une topologie statique en entonnoir Le modèle simplifié de topologie est centré autour des stations sol qui sont ici les points d'agrégation du trafic : il y a beaucoup moins de station sol que de satellites ou d'utilisateurs. Nous choisissons donc d'étudier un réseau local tel que présenté figure 3.4. Un réseau étudié a une forme d'entonnoir, aussi nommé Y-topologie par [RKT00]. Nous ajouterons des points de fuite dans notre réseau, car tout le trafic ne sera pas toujours observé dans des cas réels et que nous retrouvons ce type de topologie en *parking Lot* dans la littérature.

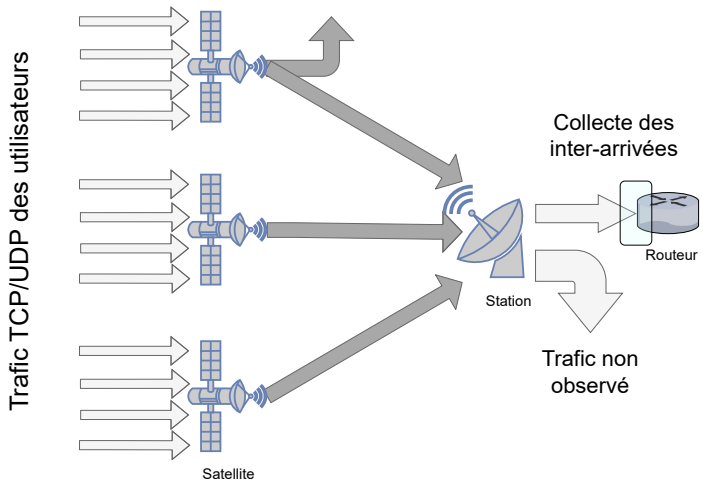


FIGURE 3.4 : Réseau simplifié : les objets sont statiques et seul le voisinage de la station sol est conservé.

Dans une constellation, les vitesses relatives des satellites créent de la gigue, et au bout de quelque temps les satellites changent de station et la topologie évolue. Cependant, sur une courte période de simulation, inférieure à la minute, nous pouvons considérer que la topologie n'aura pas évolué et que les protocoles utilisés (TCP notamment) auront eu le temps de converger vers un état d'équilibre. Ainsi nous simplifions l'étude à un réseau statique, sans prendre en compte les connexions-déconnexions des satellites, ni même leur mouvement.

Ayant simplifié largement le problème étudié, nous proposons d'utiliser un émulateur plutôt qu'un simulateur. Un réseau émulé est constitué d'un mixte de composants réels et simulés. Contrairement à un simulateur qui définit de façon centralisée tous les événements qui se produisent dans le réseau, ici chaque composant agit et interagit indépendamment avec les autres composants. Ce système est donc à la fois plus réaliste parce qu'il utilise des applications réelles, et plus performant parce qu'il est parallélisé par nature.

Cette parallélisation nous permettra de réduire le temps de génération de données et travailler sur des débits plus élevés, similaires à ceux expérimentés par les utilisateurs de réseaux.

Dockem Dockem est un simple script inspiré de GoNetEm. GoNetEm est un outil pédagogique de l'ENAC qui déploie un réseau virtuel de conteneurs dockers et apprend à le gérer en configurant chaque machine individuellement en ligne de commande.

Dockem est un script python qui automatise ces étapes. Il crée des conteneurs dockers, déploie des réseaux, configure le routage par plus court chemin et enfin lance des commandes pour démarrer les serveurs et créer des flux de paquets depuis les clients. En plus d'avoir un fichier de configuration décrivant la topologie, Dockem contient un fichier décrivant les différentes actions de mise en écoute d'un serveur TCP, capture de paquets et émission de flux. Sa configuration est divisée en deux fichiers décrivant la topologie et les actions des utilisateurs.

Une simulation est décrite par deux fichiers. Le premier contient la topologie du réseau et les configurations des interfaces. Le second fichier contient la séquence des commandes effectuées par les différents nœuds du réseau.

3.4.1.2 Mise en place d'une topologie : création des nœuds et liens

L'émulateur repose avant tout sur deux briques techniques : le conteneur docker et l'émulation de liens par les interfaces virtuelles. Nous détaillons ces deux notions puis abordons brièvement le routage.

Présentation des nœuds : un mot sur les conteneurs docker L'émulateur débute par le déploiement de nœud et la création de réseaux docker pour les relier. Chaque nœud peut être considéré comme un ordinateur à lui tout seul et être représenté par une machine virtuelle. Cependant, les machines virtuelles consomment beaucoup de ressources et des solutions moins gourmandes en temps de calcul de processeur ont été développées.

Il existe dans le noyau linux une fonctionnalité appelée espace de noms (*namespace*) qui permet de lancer des processus dans un contexte spécial. Un processus lancé dans un tel contexte ne verra pas toute l'arborescence des fichiers, ne pourra pas se rendre compte de la présence d'autre processus. Docker est un outil facile d'utilisation qui permet de créer des conteneurs. Chaque conteneur simule un OS dans un contexte soigneusement isolé et protégé, qui ne se rend pas compte qu'il est hébergé par un hôte dont il n'a pas accès à toutes les ressources.

Le conteneur peut héberger des applications qui sont des scripts client/serveur TCP, émetteur/récepteur UDP et de capture de paquets (tshark). Nous différencions ainsi les clients (émetteurs de paquets), des serveurs (réception des paquets) et satellites (routeur). En plus de cette spécification, chacun peut être un point de capture de trafic et enregistrer le trafic qu'il verra passer.

En créant de nombreux conteneurs et en les autorisant à communiquer entre eux grâce à des interfaces virtuelles, on modélise un véritable parc informatique et son réseau internet.

Contrairement à un simulateur à événements discrets, l'émulateur fonctionne en temps réel. La charge ne peut donc pas être lissée dans le temps comme en simulation, mais est répartie entre les multiples cœurs du processeur. Une expérience émulée est donc bien plus rapide qu'une expérience simulée, mais le nombre et les débits des interfaces émulées est bridé par la fréquence du processeur et le nombre de cœurs. Pour éviter d'atteindre ces limites, nous vérifions que les débits échangés restent bien en deçà du débit du processeur. Le scénario restera donc simple, le nombre de satellites largement réduit par rapport à la taille d'une constellation.

La configuration des interfaces virtuelles Les nœuds sont ensuite regroupés en sous-réseaux. Un routeur est un nœud appartenant à plusieurs sous-réseaux. À chaque définition d'un sous-réseau, les nœuds qui en font partie se voient assigner une nouvelle interface qui peut communiquer avec tous les nœuds du même sous-réseau.

Les interfaces émulées sont bridées et configurées grâce à la commande tc, le couteau suisse de la configuration d'interfaces réseau. Cet outil multifonction, largement utilisé dans tous les serveurs linux, permet d'imposer des lois de contrôle du trafic et réaliser les opérations suivantes :

- **Shaping** : Limiter le débit du trafic par mise en file d'attente.
- **Policing** : Limiter le débit du trafic par marquage ou rejet de paquets.
- **Ordonnancement** : Prioriser certains types de trafic QoS.

- **Classification** : Trier les paquets selon des critères (IP, port, etc.) pour leur appliquer des règles.
- **Filtrage** : Appliquer des règles selon les en-têtes des paquets.

Des chaînes de commandes tc permettent de créer des structures complexes de partage de la bande passante. Dans le cadre de l'émulation de lien, nous avons avant tout besoin de brider les débits. Différentes implémentations permettent de limiter le débit des flux à travers une interface :

- tc tbf : *token bucket filter*, qui fonctionne sur un mode continu de génération de jetons, à rapprocher du modèle M/D/1. L'explication des timers dans la documentation ne semble pas à jour.
- tc hfsc dont la documentation semble la plus à jour et explique les timers dans linux.
- tc netem

Nous avons systématiquement utilisé la commande *tc netem* car elle permet de brider de débit du lien et ajouter de la latence en même temps, en plus d'autres options comme un modèle de gigue ou la génération de pertes.

Le routage Ayant décrit les nœuds et établi leurs spécificités, les clients pourront émettre des paquets sur le réseau. À chaque nœud intermédiaire se pose la question du prochain nœud où devra aller le paquet. Cette réponse est automatiquement donnée par les tables de routage, qui doivent être configurées pour indiquer la bonne destination.

Il existe des algorithmes de routage comme OSPF [Moy98] où les routeurs échangent des informations, construisent une carte du réseau et remplissent automatiquement leur table de routage. À l'instar d'Hypatia, nous avons préféré calculer directement les plus courts chemins (en terme de nœuds) et remplir les tables de routage de façon statique. Cet algorithme calcule les chemins entre les générateurs et récepteurs de paquets et les tables de routage sont déployées au début de la simulation.

Quand les interfaces sont configurées et que les tables de routage déployées, le réseau est prêt pour une simulation.

3.4.1.3 Les commandes de génération et capture de flux de trafic

La simulation proprement dite est définie par les actions des nœuds du réseau. Elle commence avec le déploiement des serveurs TCP et récepteurs UDP. Des points de

capture de paquets sont mis en place avec tshark. Enfin les utilisateurs commencent leur séquence d'émission de paquets : clients TCP et émetteurs UDP. Nous abordons ici quelques point importants de ces différents scripts.

Émission - réception UDP L'émission-réception de paquets est simple à gérer. Côté client, il suffit de définir d'ouvrir un socket en mode datagramme. Il suffit de spécifier adresse et port destination et le système est paré à envoyer. La loi d'émission d'attente avant l'émission du prochain paquet est alors définie. Un *timer* est réglé pour commander l'envoi d'un paquet à intervalles de temps fixe ou suivant une loi, par exemple la loi exponentielle. Les paquets ont une taille constante prédéfinie. Un champ contenant le numéro du paquet est inséré à la suite de l'en-tête UDP dans l'espace laissé libre. Ce numéro de séquence sera incrémenté pour chaque nouveau paquet afin de détecter les pertes.

Un socket peut être ouvert dans le récepteur à l'autre extrémité du réseau au port spécifié par l'émetteur. Cela permet de réaliser des logs à l'autre bout pour mesurer le débit reçu et estimer un taux de pertes. Ces logs n'ont d'intérêt que pour vérifier le bon fonctionnement du système et mesurer la qualité de service. Nous n'en avons pas besoin dans le cadre de l'inférence de topologie.

Émission - réception TCP La gestion est plus complexe dans TCP.

Voici le fonctionnement : un socket TCP est ouvert en mode écoute côté serveur sur un port prédéfini, un autre socket TCP est créé côté client. Le socket client se connecte au serveur qui accepte sa demande selon le protocole TCP. Le serveur transfère la connexion vers un nouveau port et libère ainsi le socket de connexion pour d'autres clients. Chaque client a initialement une certaine quantité de données à transmettre. Son buffer est rempli régulièrement pour continuer à émettre pendant une durée spécifiée. En utilisant les sockets du système, nous sommes contraints à l'utilisation de l'algorithme de contrôle de congestion du système, *Cubic* dans notre cas.

Dans une connexion TCP, seul l'émetteur a une parfaite connaissance des paquets émis et des pertes occasionnée. En effet, lorsqu'un paquet n'est pas acquitté, il est réémis avec le même numéro de séquence. Il est donc plus difficile de labelliser précisément les séries temporelles de paquets et noter que certains paquets ont été perdus. De même que dans le cas UDP, le récepteur TCP peut enregistrer la progression de la connexion. Nous ne collectons pas ici de mesures de RTT ou de fenêtre de congestion côté client.

Capture de paquets Nous utilisons l'outil tshark, version sans interface graphique de wireshark pour capturer les paquets. Cette capture peut être effectuée en tout nœud du réseau, sur chaque interface. Pour accélérer le traitement, la capture est limitée à l'en-tête du paquet. La précision de l'instant de capture est très importante. Elle dépend principalement des composants de la carte réseau. Les mesures doivent permettre d'identifier la présence d'un délai de l'ordre de la durée d'émission d'un paquet. Par défaut Wireshark enregistre les valeurs avec une précision de l'ordre de la microseconde, qui correspond au délai d'émission d'un paquet de taille standard 1500o émis à 12Gb/s. Nos simulations sont donc limitées à cause de la précision des mesures à 1,2Gb/s. S'il y a des paquets plus petits dans le réseau, par exemple des acks simples de taille 42o, le débit d'un lien doit être de l'ordre de 33Mb/s pour conserver une bonne précision. Ce débit étant très faible, nous pouvons soit augmenter la précision du timestamp des paquets à la nanoseconde près, soit ignorer ces petits paquets.

Notons que nous aurions aussi pu collecter directement des statistiques. La commande tc permet d'enregistrer des statistiques du remplissage de la file d'attente et du nombre de paquets transmis et perdus. Cet outil ne permet pas un suivi en temps réel, mais peut être utilisé à intervalle régulier pour suivre l'évolution de la file d'attente. Nous ne l'avons pas exploité car ces informations sont contenues de façon plus précise dans les fichiers de capture des paquets en entrée et sortie de la file d'attente.

3.4.2 Réseau émulé

Après avoir présenté l'émulation dans ses grandes lignes et les principaux aspects à considérer, nous passons à la pratique. Nous décrivons la topologie du réseau émulé, puis les mesures d'un réseau.

3.4.2.1 Description de la topologie du réseau émulé

Nous présentons figure 3.5 le réseau émulé qui servira pour nos expériences.

Ce réseau contient

- 7x12 = 84 utilisateurs représentés par les symboles ◯
- 11 routeurs "satellites" nommés aX, bX et cX selon leur proximité à la station obs1
- trois routeurs "terrestres" obs1, obs2, obs3
- 7 serveurs non représentés sur ce schéma, dont 5 situés en aval de obs1 et un seul derrière obs3

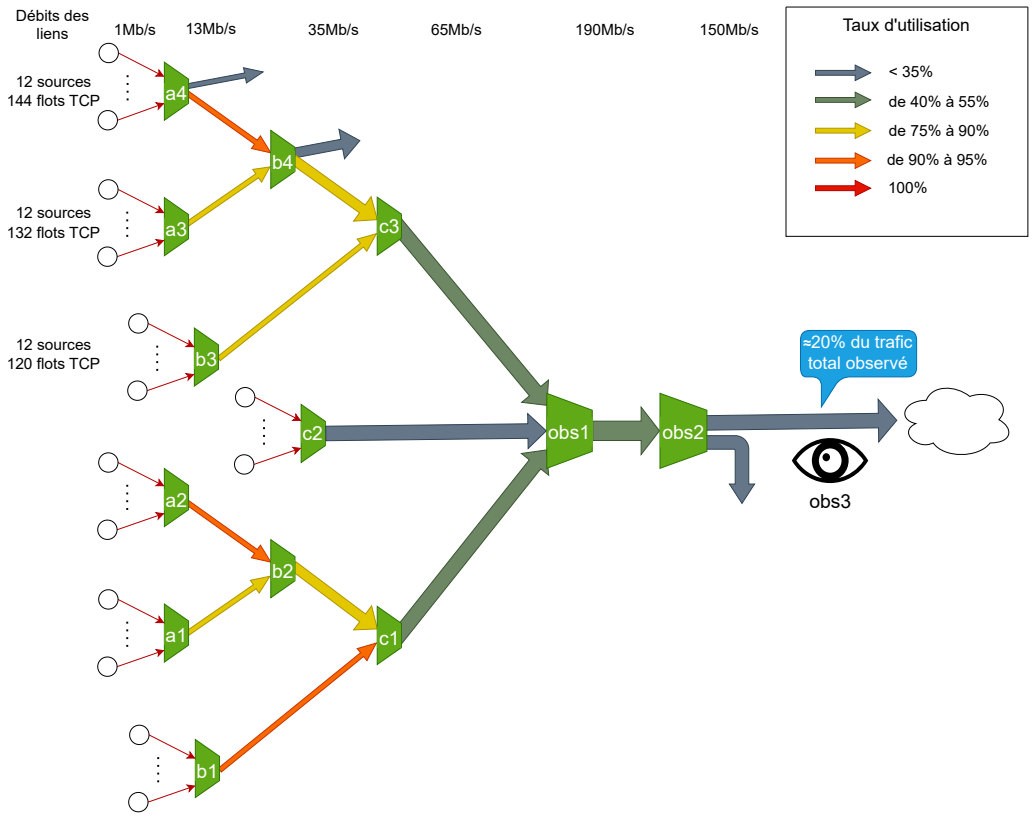


FIGURE 3.5 : Représentation du réseau émulé

Chaque nœud a un débit limité, les ordres de grandeurs sont donnés sur la figure. Les valeurs précises sont dans le fichier des paramètres de la topologie, de même que les valeurs de délai ajoutées. Notons qu'il n'y pas de gigue ajoutée sur le chemin montant des utilisateurs vers les serveurs, mais ces derniers en introduisent.

Concernant les actions, chaque utilisateur émet 2 flux TCP vers chacun des serveurs en aval : les utilisateurs a1, a2, bX et cX sont connectés à 5 serveurs, les utilisateurs a3 ont 6 serveurs en aval tandis que les utilisateurs a4 en ont 7.

3.4.2.2 Collecte et analyse des captures de trafic

Nous présentons ici quelques aspects pratiques de la capture et les motivations du scénario. Dans un second temps, nous analysons les contraintes de simulation et la pertinence des données collectées.

Mesure du trafic L'expérience dure un peu plus de deux minutes pendant lesquelles nous capturons les paquets sur les liens du réseau.

Contrairement à ce qui était fait dans le simulateur, nous n'avons pas enregistré de statistiques de remplissage des files d'attente. Cependant, des points de capture ont donc été disposés sur tous les nœuds pour capturer les instants d'arrivée des paquets avec *tshark*.

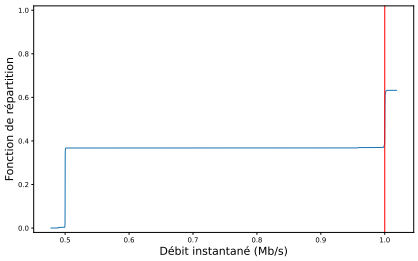
Ces traces servent ensuite à établir l'état du réseau, notamment le taux d'utilisation des files d'attente. Le taux d'utilisation correspond au rapport de la période où la file d'attente transmet des paquets divisée par le temps total. Ce ratio est mesuré en Erlang, unité de mesure sans dimension. Ce taux d'utilisation met en évidence que, dans le scénario étudié, seuls les liens issus des utilisateurs sont saturés, leur charge atteint 100%.

L'intuition que nous voulons vérifier à travers cette expérience est que les techniques de détection de goulot sont en fait des techniques de détection de file d'attente qui mesurent la saturation d'une file d'attente, ou plus précisément la proportion de paquets ayant attendu dans la file d'attente. Nous ne présentons donc pas un scénario de congestion de la station sol, mais un écoulement nominal dans un réseau en bon état. Nous conservons dans ce scénario l'intuition que l'information est contenue entièrement dans les inter-arrivées des paquets qui reflètent les débits des liens. Nous formulons donc l'hypothèse que cette information ne sera pas perdue à condition de conserver intacts une bonne partie des inter-espacements créés par une file d'attente. Cette condition est vérifiée si la capacité des liens est croissante dans le sens de parcours des paquets, et si la saturation des liens en aval n'est pas trop élevée. Ces raisons nous ont donc poussé à utiliser un tel réseau en forme d'arbre, dont les sources de flux sont situés aux extrémités sur des liens à faible capacité. Le point d'observation est situé à sa base, et le réseau émulé ne présente pas de congestion. Enfin, de nombreux flux ne sont pas directement observés, ce qui permettra de vérifier la robustesse du concept.

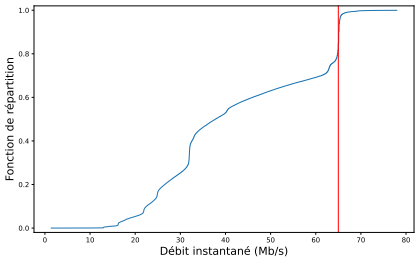
Limites techniques de l'expérimentation La réalisation de l'expérience se heurte à deux limites techniques. La première est une limite physique liée à la capacité de traitement du processeur. Le processeur doit pouvoir supporter un lien individuellement, ce qui signifie qu'un cœur doit avoir une capacité de traitement supérieure au débit d'un lien. De plus, les opérations de transmission des paquets sur les différents liens doivent être réparties entre les processeurs : la somme des débits des liens devra rester inférieure au débit de traitement du processeur. De façon préventive, nous définissons aussi des liens dont les débits restent bien inférieurs à la fréquence du processeur (dans

notre cas 1,8 GHz). Nous vérifions aussi pendant le déroulement de l'expérience que le processeur n'est pas surchargé.

La seconde difficulté concerne la différence entre la simulation et la pratique. En pratique une interface physique est bridée, son débit instantané ne peut pas dépasser une certaine valeur, qui découle de la fréquence d'un oscillateur interne. Cette contrainte est différente dans une interface virtuelle, dont l'intervalle de régulation peut être supérieur au temps de transmission d'un paquet. Nous reviendrons sur cela en début du chapitre d'inférence de topologie. Ainsi, dans certains cas la contrainte de temps de transmission qui est systématiquement présente peut être violée dans l'émulateur lorsque plusieurs paquets seront transmis en bloc.



(a) Débits instantanés sur un lien ***utilisateur→a1***



(b) Débits instantanés sur le lien ***c1→obs1***

FIGURE 3.6 : Courbes permettant de comparer les débits instantanés en sortie de deux files d'attente. Les limites de débit émulé sont représentées par les lignes rouges : toute valeur qui dépasse ces limites de débits indique une erreur. Contrairement à la file illustrée à droite, celle de gauche illustre un échec de limitation du débit par une chaîne tc.

Cette difficulté est illustrée figure 3.6. Cette figure présente des observations de débit instantané sur deux liens du réseau. Ces courbes sont mesures de ratio $\frac{p}{\Delta T}$ où p est la taille d'un paquet et ΔT la différence de temps qui sépare l'instant de fin de réception du paquet précédent avec l'instant de fin de réception du paquet étudié. Le lien étudié sur la figure de droite (fig 3.6b) est correctement modélisé : la fonction de répartition des débits instantanés montre que ceux-ci restent bien inférieure à la limite fixée, malgré quelques erreurs dues à des paquets de petite taille. En revanche, la fonction de répartition présentée à gauche (fig 3.6a) révèle que près de 40% des paquets ont été transmis trop rapidement par rapport au débit indiqué. Malgré un débit instantané souvent au-dessus de la limite, le débit global observé sur l'intervalle de temps de la simulation est proche de 1Mb/s. Nous nous sommes donc rendus compte que les limites de débit instantanées ne sont pas respectées sur tous ces petits liens

situés en amont de la topologie. En revanche tous les autres liens sont correctement émulés.

3.4.2.3 Conclusion

En conclusion, cet émulateur basé sur docker a été rapide et simple à développer comparé à la prise en main d'Hypatia. En ce qui concerne les performances, l'émulateur est aussi bridé par la fréquence du processeur mais tire parti des nombreux cœurs du réseau. Il existe d'autres émulateurs, notamment Mininet. Mininet peut fonctionner sur une VM qui a l'avantage de ne pas interférer avec le système en place, mais provoque une large chute des performances, comparé à la version installée localement sur un système linux. Dans ce second mode de fonctionnement, mininet permet probablement d'atteindre de meilleures performances car nos conteneurs docker sont plus lourds que de simples processus dans des *namespaces*.

Toutefois, les conteneurs docker présentent un bon compromis et leur facilité de déploiement et d'utilisation nous a permis de construire facilement un scénario adapté à nos besoins et générer des données rapidement en quantité suffisante.

Les données produites peuvent être considérées réalistes sur un intervalle de temps court. Pour plus de réalisme, il faudra utiliser des modèles de trafic et implémenter le mouvement des satellites et les changements de connexions. Toutes ces considérations ne devraient pas avoir trop d'impact sur les résultats. Comme nous l'avons mentionné plus haut, le réalisme vient surtout du modèle de file d'attente. Si le modèle ne correspond pas à ce qui et attendu, les données sont inutilisables (ou le modèle, selon les cas !). Dans l'émulateur, une question cruciale est celle du choix de la commande tc utilisée et de son implémentation. Selon la qdisc choisie, les résultats pourront être très différents.

Après avoir testé des modèles de file d'attente de plus en plus complexes, nous avons utilisé un simulateur qui l'était plus, alimentant des questions finalement éloignées de l'objectif. Par exemple, comment répartir les utilisateurs sur le globe ? Quel trafic simuler ? etc. Nous sommes revenus de tout cela avec un émulateur simplifié, mettant en évidence l'importance du choix de la file d'attente ainsi que, nous le verrons, de la topologie et des débits des liens.

Finalement, nous avons obtenu un jeu de données d'observation du trafic et distinguons deux modes de fonctionnement distincts de la file d'attente. Cela aura des conséquences dans l'analyse des données. Nous pouvons désormais étudier la congestion, ou plutôt la saturation dans un réseau.

Chapitre 4

Une méthode passive d'inférence de topologie dans un réseau IP

Avant d'aborder ce chapitre, reprenons le fil de cette thèse depuis le début. Thalès Alenia Space, fabricant de satellites géostationnaires, voit le déploiement de la constellation de satellites LEO Starlink. Cette constellation offre un service d'accès à internet bien plus performant que les traditionnels fournisseurs d'accès à internet par satellite géostationnaire. Le marché d'internet par satellite géostationnaire s'effondre donc petit à petit face au nouvel entrant et Thalès tente de participer à des projets concurrents : HAPS (le Stratobus, idée de dirigeable stratosphérique), OneWeb, Telesat et maintenant IRIS2 (un projet porté par l'Union Européenne). Il y a deux types de mégaconstellations :

- les petites, contenant près de 300 satellites et représentées par les projets énumérées précédemment. Leur gestion peut être centralisée.

- les grandes, dont l'ordre de grandeur est proche de 15000 satellites à terme, sont représentées par Starlink et Amazon Leo (porté par Amazon). Leur gestion ne pourra probablement pas être centralisée en raison de leur taille.

Nous étudions ces grandes constellations qui fourniront un meilleur service à condition de les exploiter convenablement. Comme dans les réseaux terrestres, ces réseaux sont sujets à des problèmes de congestion. Le chapitre précédent a permis de les modéliser et a rappelé que la congestion se produit lorsque les goulots d'étranglement du réseau se situent près du cœur réseau. Dans le cas d'une mégaconstellation, détecter la congestion revient donc à identifier la présence d'un goulot à proximité d'une station sol.

Ce chapitre présente la contribution majeure de la thèse qui est une méthode pour reconstruire la topologie du réseau depuis une station sol. Les instants d'arrivées des paquets IP, initialement groupés par flux, sont analysés et progressivement regroupés selon leur provenance de plus en plus précise en remontant dans la topologie depuis le point d'observation jusqu'à l'émetteur. Nous commençons par expliquer le concept par une présentation simplifiée du fonctionnement d'une file d'attente, et comparons la contrainte exercée par une file d'attente avec la notion d'entropie utilisée par Katabi. Un test de détection de file d'attente commune est ensuite implémenté pour créer une méthode complète d'inférence de topologie, qui est ensuite illustré à travers un exemple pas à pas. Enfin, les résultats de cet algorithme et de la méthode Katabi sont présentés dans divers cas d'application pratiques, émulés à partir de l'outil présenté au chapitre précédent (§ 3.4.1).

4.1 Méthodes de détection d'une file d'attente

Nous avons vu qu'il existe de nombreuses méthodes pour détecter les files d'attente présentes dans un réseau. Nous cherchons dans un premier temps à caractériser la présence d'une file d'attente par la distribution des inter-espacements en sortie de file. Nous présentons ensuite une nouvelle approche de détection de file d'attente.

Les principales variales utilisées dans cette section de détection de file d'attente

sont notées dans le tableau 4.1.

Symbole	Description
y	indice d'une file d'attente.
x	indice d'un paquet.
b_x	taille du paquet x.
T	durée de transmission d'un paquet.
T_{xy}	durée de transmission du $x^{\text{ième}}$ paquet en sortie de la file y.
μ_y	débit de la file y.
$t_{x,y}$	instant de fin d'émission du $x^{\text{ième}}$ paquet en sortie de file y.
$e_{x,y}$	inter-espacement du paquet x avec le paquet $x-1$ en sortie de file y. Cet intervalle de temps est divisé par b_x.

Table 4.1 : Tables de notations utilisées dans l'analyse des files d'attente

4.1.1 Méthodes de la littérature et analyse des données d'une file d'attente

Nous revenons sur deux méthodes de la littérature qui ont attiré notre attention : le minimum et l'entropie. Ces méthodes sont mises en relation avec le modèle de file d'attente pour expliquer leur fondement, et ainsi élaborer une méthode alternative.

4.1.1.1 L'estimateur du minimum pour déterminer le débit de la dernière file d'attente traversée

On s'intéresse dans ce premier cas à un problème simplifié, afin d'identifier la différence entre un agrégat de flux et la sortie d'une file d'attente. Identifier cette différence implique de bien définir la file d'attente, et cela nous amène à la première piste du minimum des inter-arrivées. Nous introduisons ensuite la notion d'intervalle de régulation comme une extension au concept de file d'attente.

Le débit instantané bridé Parmi les modèles de file d'attente présentés précédemment, le modèle le plus réaliste est le modèle déterministe. Le temps de transmission d'un paquet est simplement le ratio entre la taille b_x du paquet x et le débit μ_y de la file d'attente y.

$$T_{xy} = \frac{b_x}{\mu_y}$$

Chaque file d'attente ne traite qu'un seul paquet à la fois, les paquets sont traités successivement. La différence des instants de fin de réception des paquets vérifie :

$$t_{by} - t_{ay} > T_{by} \tag{4.1}$$

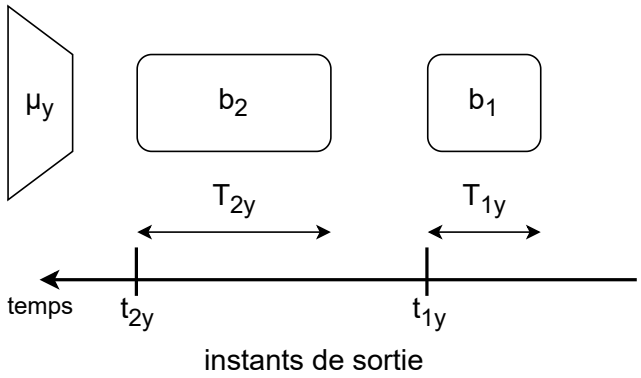


FIGURE 4.1 : Mesure de différence d'instants de sortie de file d'attente

Ainsi, pour mettre en évidence la présence d'une file d'attente, on mesure les écarts pondérés - inverse du débit instantané - entre deux paquets successifs en sortie de la file y. Ces mesures sont définies par

$$e_{xy} = \frac{t_{x,y} - t_{x-1,y}}{b_x} \tag{4.2}$$

Dans un premier temps, nous travaillerons avec des paquets de taille constante unitaire et les illustrations seront réalisées avec des simulations de paquets suivant une loi de trafic entrant est exponentielle. L'extension à des cas plus complexes sera abordée après avoir présenté le concept de détection de file. Ainsi, T_{xy} est constant et vaut $T_y = \frac{1}{\mu_y}$ pour tout x.

Enfin, les valeurs e_{xy} sont définies dans l'intervalle $[\frac{1}{\mu_y}, \infty)$, indépendamment de la loi du trafic entrant. Détecter la présence d'une file d'attente reviendra à montrer l'existence d'une borne basse (T_{Xy}) à l'intervalle de définition de la variable aléatoire e_{Xy}. La démonstration de l'existence de cette file implique d'en estimer le débit grâce à l'estimateur $\hat{T}_y = \min_x e_{xy}$ qui converge vers $T_y = \frac{1}{\mu_y}$.

Cette hypothèse reste valide à condition que la régulation de trafic soit effectuée instantanément, comme nous allons le voir.

Intervalle de régulation du trafic Toute série de paquets est émise par un dispositif y dont le débit maximal est limité, et nous pouvons estimer cette valeur en sortie immédiate grâce à l'estimateur $\hat{T}_y = \min_x e_{xy}$, où x prend les valeurs des indices des paquets observés en sortie de la file y. Cependant le mélange avec des flux issus d'autres files d'attente et le passage à travers des files intermédiaires perturbe cet estimateur ? Nous devons donc nous baser sur des mesures d'inter-espacement entre deux

paquets successifs potentiellement bruitées, qui ne reflètent plus forcément le temps de transmission minimal T_y observé en sortie de la file y. Le choix de mesurer des écarts pondérés fonctionne ici car le débit instantané d'une interface est bridé physiquement par la technologie. Le débit sortant ne dépassera donc pas μ quel que soit l'intervalle de temps considéré. Nous observons cela dans la distribution des inter-espacements.

En simulation ou émulation, certains mécanismes de limitation du débit régulent le trafic plus grossièrement, sur un intervalle de temps plus grand. L'exemple typique est le token bucket, dont l'intervalle de temps de régulation correspond à l'intervalle de temps de remplissage d'un bucket. Dans une telle file d'attente les paquets de taille inférieure au token sont transmis théoriquement à vitesse infinie, en pratique à la vitesse maximal du matériel. L'étude de la distribution des temps d'inter-arrivées ne convient donc pas pour montrer l'existence d'une telle file d'attente.

Enfin, les files d'attente M/M/1 ont une durée de régulation probabiliste : leur débit ne peut pas dépasser tel seuil avec telle probabilité, donnée par la loi de sortie de la file d'attente.

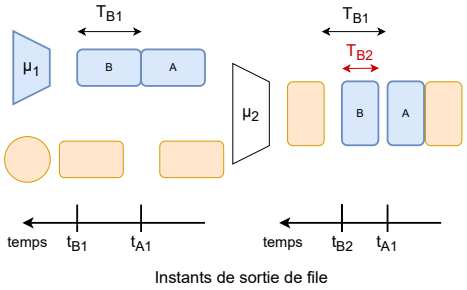
En conclusion, cette méthode du minimum des inter-arrivées pondérées permet de détecter la dernière file d'attente traversée par l'agrégat et estimer son débit. Mais pouvons-nous savoir ce qui se passe en amont de cette dernière file ?

Dans la suite, nous détecterons la présence de files d'attente à régulation de débit instantanée en observant les écarts entre deux paquets consécutifs issus de cette file. Nous n'avons pas étudié les cas avec intervalle de régulation, mais l'approche développée ici pourrait leur être adaptée en vérifiant l'existence d'un intervalle de régulation : au-dessus de l'intervalle de régulation, le débit reste plafonné. Au contraire, il faudrait montrer que les mesures de débit ne cessent d'augmenter en réduisant la durée de l'intervalle de régulation.

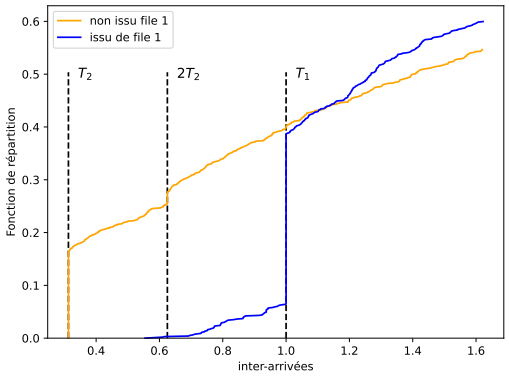
4.1.1.2 Utiliser le trafic issu d'une file pour estimer son débit

Dans la plupart des cas, l'estimation du minimum des écarts pondérés entre paquets successifs convergera vers le débit de la dernière file d'attente commune. Mais nous aimerions savoir ce qu'il y a en amont de cette dernière file.

Transformation de la distribution des inter-espacements Nous illustrons cette étude de ce qui se passe en amont de la dernière file d'attente avec la figure 4.2a. Voici la situation : un flux de loi exponentielle traverse une première file 1 de débit μ_1, puis est agrégé à un second flux de loi exponentielle avant de traverser une file 2 de débit μ_2.



(a) Mesure de différence d'instants de sortie de paquets issus d'une file d'attente



(b) En sortie de file 2, comparaison entre trafic issu et non issu de la file 1.

FIGURE 4.2 : Flux traversant plusieurs files d'attente successives

Nous observons en sortie du système le flux issu de la file 1 pour espérer y déceler une marque de la file 1. Pour simplifier, les résultats présentés figure 4.2b sont obtenus par de simples simulations avec des paquets de taille constante.

Nous voyons sur cette figure que la piste précédente du minimum des inter-arrivées s'évanouit : le minimum des inter-espacements, qui vaut T_1 en entrée, converge vers T_2 (durée de transmission d'un paquet de la file 2) à mesure que la durée d'observation augmente. Si $\mu_1 > \mu_2$, la convergence est immédiate : aucune inter-arrivée formatée par la première file d'attente ne sera conservée par la seconde file d'attente. Dans le cas contraire, la convergence sera plus ou moins lente selon le trafic concurrent dans la seconde file. En revanche, nous constatons ici qu'une grande proportion des inter-arrivées entre paquets issus de la file 1 ont été formatées à la valeur T_1 et qu'elles ont été conservées bien que les paquets soient passés ensuite à travers la file 2. Nous retrouvons ainsi l'étude des modes de la distribution des inter-arrivées, préconisée par de nombreux auteurs dans l'état de l'art ([Pax97], [LB01]). Observons que le mode lié au temps de transmission de la file est très marqué dans la fonction de répartition de l'agrégat des flux sortant de la file d'attente, mais peut l'être beaucoup moins si nous observons un sous-ensemble de cet agrégat.

L'entropie Plutôt que d'observer la distribution selon l'axe des abscisses, l'entropie mesure selon l'axe des ordonnées. L'entropie de Rényi est une généralisation de l'entropie de Shannon définie par la formule

$$H_q = \frac{1}{1-q} \log_2 \left(\int f^q(x) dx \right) \tag{4.3}$$

L'entropie mesure le désordre dans le trafic. Intuitivement, un flux entrant dans le système est désordonné. La file d'attente filtre les petites inter-arrivées et régule le trafic. Moins les valeurs sont dispersées, plus il y a d'ordre et donc plus il est probable que le flux ait été régulé par une file d'attente.

Nous pouvons utiliser l'entropie pour agréger les flux en fonction des files d'attente traversées. Plus une file d'attente a un impact fort sur le trafic observé (en étant souvent saturée), plus l'agrégat qui s'en écoule sera ordonné. Agréger des flots permettra de réduire l'entropie à condition que les inter-espacements entre ces flots soient ordonnés par une file d'attente commune. Katabi montre qu'en minimisant l'entropie, on regroupera les flux par la file d'attente principale qui aura impacté le plus les inter-espacements du trafic [KB01]. Toutefois, cette file peut être qualifiée à tort de goulot du réseau : en considérant l'illustration 4.2a, la file 2 peut avoir une grande capacité et être souvent remplie, masquant ainsi une file 1 sous-dimensionnée qui est le vrai goulot du réseau, à l'origine de nombreuses pertes.

L'entropie ne mesure donc pas vraiment la congestion mais la saturation d'une file d'attente. De plus, son implémentation souffre de quelques difficultés techniques que nous présentons ici. Tout d'abord, nous n'avons pas accès à la densité de probabilité f mais à des données discrètes. Nous utilisons une version discrète de l'entropie de Rényi en remplaçant l'intégrale par une somme et la loi de densité par l'histogramme.

Ensuite, il faut déterminer deux paramètres pour calculer l'entropie. Le premier est le paramètre q qui règle l'importance à accorder à un mode de la distribution. Plus q est élevé, plus l'attention est portée sur la hauteur du mode principal. En pratique, Katabi recommande des valeurs entre 4 et 5. Le second paramètre est le pas de l'histogramme : qu'il soit trop petit ou trop grand, aucun pic ne ressortira de l'histogramme car les valeurs seront soit trop dispersées, soit trop regroupées. Idéalement ce réglage est défini en fonction du débit de la file d'attente à mettre en évidence. Un bon pas sera trouvé entre le $100^{\text{ème}}$ et le $10^{\text{ème}}$ de cette valeur. De manière encore plus empirique, il semble qu'un réglage lié à un petit quantile des données, par exemple $\frac{\text{quantile}(0,1)}{100}$ fonctionne correctement. Cela se justifie par le fait que la file d'attente sera avant tout visible dans les petites inter-arrivées de la distribution. Le pas doit être suffisamment grand pour obtenir un nombre convenable de mesures (le choix du minimum des inter-arrivées est mauvais de ce point de vue) mais suffisamment petit pour observer des variations.

Enfin, l'entropie ne prend en compte que les valeurs sur l'axe des ordonnées ce qui la rend sensible au débit. Dans un réseau, la loi qui représente le désordre des paquets n'est pas une loi uniforme des inter-arrivées des paquets mais une loi exponentielle. L'entropie d'un trafic de loi exponentielle de débit λ vaut $H_q = \frac{1}{1-q} \log\left(\frac{\lambda^{q-1}}{q}\right)$. Avec $q > 1$, plus λ est élevé, plus l'entropie sera faible, conduisant à le supposer issu d'une

file d'attente, à l'instar d'un flux de débit faible présentant un pic caractéristique d'une file d'attente.

4.1.2 Méthodes par comparaison avec un flux de référence aléatoire

Les méthodes de la littérature permettent de détecter certaines files, nous proposons ici des méthodes alternatives pour identifier et caractériser les files d'attente du réseau. Une file d'attente sera observable à condition de modifier une part significative du trafic qui la traverse et que cette marque de la file d'attente soit conservée jusqu'au point d'observation. Un critère simple est de pouvoir observer des temps inter-arrivées formés par chacune des files d'attente en amont, donc que les débits des liens du réseau soient croissants et que les goulots du réseau soient situés en amont. Nous proposons deux méthodes différentes, dans l'ordre où nous les avons eues au cours de la thèse.

4.1.2.1 Détection et mesure du débit de la première file commune à un agrégat

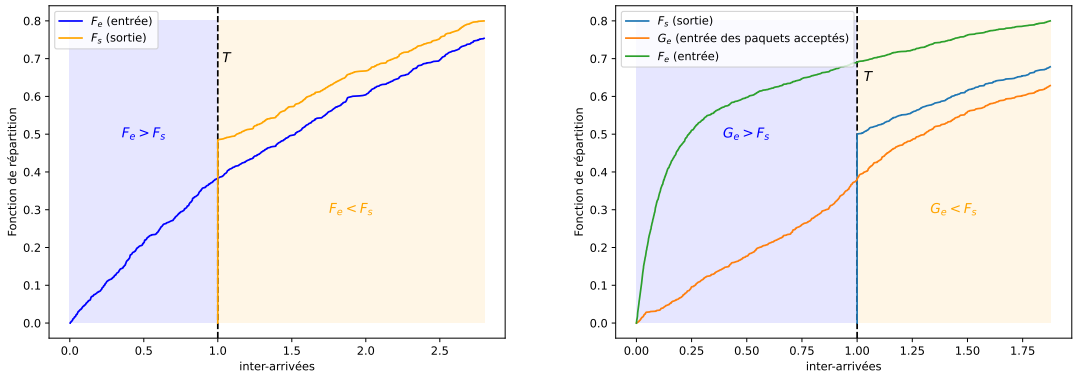
La première méthode consiste à déceler dans un agrégat la file d'attente de plus faible débit ayant impacté l'agrégat. Cette file d'attente sera la première file détectable traversée par l'agrégat.

Comparaison entre trafic entrant et sortant Pour illustrer le problème, nous présentons un simple flux qui traverse une file d'attente. Les paquets sont de taille constante et espacés suivant une loi exponentielle.

Nous obtenons les résultats présentés figure 4.3. La file d'attente filtre effectivement les inter-arrivées comme exprimé avec le minimum. La fonction cumulée croissante des inter-arrivées des paquets est donc inférieure en sortie par rapport à l'entrée ($F_e > F_s$). La file stocke les paquets et, par conservation du débit, transmet les paquets entrés à débit élevé au débit de la file d'attente. Cela produit un pic de densité (réduisant l'entropie) et un saut de la fonction de répartition au-dessus de la fonction de répartition en entrée $F_e < F_s$ sur une certaine plage, avant convergence éventuelle des deux fonctions de répartition.

Ainsi, la mesure de l'impact d'une file d'attente peut se quantifier en comparant les distributions des inter-arrivées des paquets :

$$\max_x |F_e(x) - F_s(x)| \tag{4.4}$$



(a) En absence de pertes (b) En présence de pertes

Figure 4.3 : Comparaison entrée/sortie d'une file d'attente

Ce maximum sera atteint en T, le temps de transmission d'un paquet par la file d'attente. Notons qu'à cette valeur s'opère la transition du signe de $F_e - F_s$. Nous avons donc ici un moyen de quantifier l'impact de la file d'attente en mesurant la proportion d'inter-espacements modifiés, ainsi que de déterminer son débit. Nous précisons qu'il s'agit d'une comparaison entre trafic "entré" et "sorti". En effet, lorsque la file d'attente est totalement remplie, des pertes sont générées. Ces paquets jetés avant l'entrée dans la file d'attente auraient aussi bien pu ne pas exister, la sortie n'aurait pas été différente. La figure 4.3b illustre cette comparaison entre les flux de trafic en entrée et en sortie de la file.

Toute la difficulté consiste donc à générer un trafic équivalent à celui en entrée de la file d'attente.

Génération de trafic aléatoire La méthode proposée repose sur la comparaison entre

- La distribution des inter-arrivées du trafic sortant, observé dans la file d'attente ;
- La distribution des inter-arrivées du trafic entrant, les paquets finalement non observés, que ce soit pour cause de rejet par la file, perte ou parce qu'ils ont emprunté un autre chemin.

N'ayant pas accès au trafic sortant, nous avons besoin de reconstituer ce trafic entrant. Il existe une dépendance entre certains événements créée en sortie de file d'attente qui est supposée ne pas exister en entrée. Pour reconstruire le trafic entrant nous utilisons la simplification préférée en probabilités : une hypothèse d'indépendance des événements.

La fonction de répartition des temps d'inter-arrivée en entrée est reconstruite à partir de la sortie en posant une hypothèse d'indépendance parmi les suivantes :

- indépendance des paquets ;
- indépendance des flots.

Indépendance des paquets L'indépendance des paquets consiste à supposer que deux paquets successifs sont indépendants. Cela correspond à supposer que les paquets suivent une loi d'entrée de file d'attente exponentielle. La loi exponentielle est définie avec le seul paramètre de débit λ dont il suffit de La loi des événements indépendants étant la loi exponentielle, on cherchera la loi exponentielle qui minimise l'erreur avec la distribution observée.

La fonction cumulée croissante estimée en entrée est donc :

$$F_e(t) = 1 - e^{-\lambda t} \tag{4.5}$$

Indépendance des flots Nous savons que les utilisateurs d'internet utilisent fréquemment un algorithme de contrôle de congestion ou émettent à un débit constant. Ici nous prenons en compte que deux paquets issus du même utilisateur peuvent être corrélés, mais pas deux paquets issus de deux utilisateurs différents.

Dans le cas de l'observation d'un agrégat composé de deux flots, la variable étudiée est U le temps d'attente avant le prochain paquet. Cette variable est définie par $U = \min(U_1, U_2)$ qui sont les temps d'attente avant l'arrivée du prochain paquet issus respectivement du flot 1 ou du flot 2. Comme les deux flots sont supposés indépendants, la fonction cumulée croissante reconstruite en entrée est la suivante :

$$F_e(t) = P(U < t) = 1 - P(U_1 > t)P(U_2 > t) \tag{4.6}$$

Où les lois de U_1 et U_2 sont obtenues en traçant l'histogramme de chaque variable individuellement.

4.1.2.2 Détection de file supplémentaire à un sous-ensemble d'un agrégat de flux

L'implémentation de la méthode précédente est basée sur la métrique d'entropie qui est difficile à maîtriser. Le résultat dépend beaucoup de la saturation de la file d'attente, des files intermédiaires de la proportion de trafic observé. Plus cette proportion est faible, moins les résultats seront fiables et concluants.

Reprenant l'expérience illustrée figure 4.2, nous cherchons un moyen de vérifier si un ensemble de paquets partage une file d'attente supplémentaire par rapport aux autres flux. Les paquets issus de la file 1 sont plus rarement espacés d'un écart inférieur à T_1 que les autres. Nous exploitons cette propriété qui reste vrai sur un sous-ensemble de l'agrégat initial.

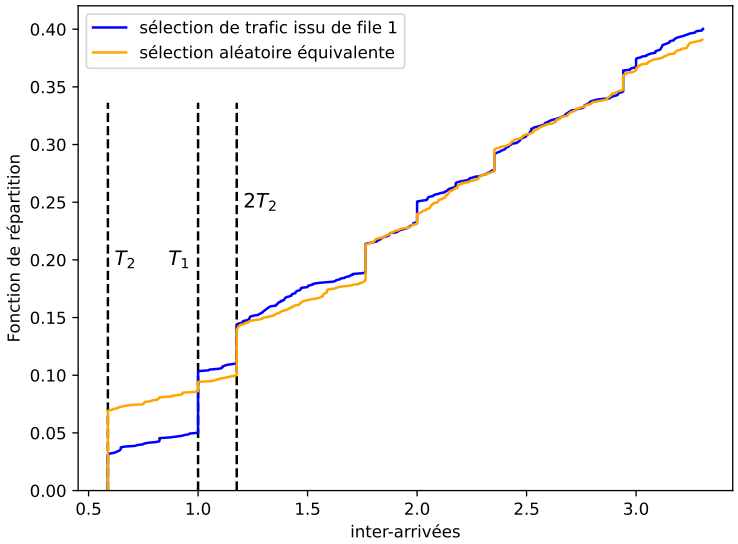


Figure 4.4 : Comparaison entre une sélection de paquets ayant traversé deux files d'attente et un même nombre de paquets choisis aléatoirement (voir schéma 4.2a).

La figure 4.4 illustre ces résultats où, en sortie de file 2, un sous-ensemble de paquets issus de la file 1 est comparé à un choix aléatoire de paquets issus de la file 2. Lorsque le nombre d'observations est suffisamment conséquent, on met en évidence que la fonction cumulée croissante des paquets est inférieur à la fonction cumulée croissante aléatoire dans l'intervalle $]T_2, T_1[$. La comparaison n'est possible qu'à débit équivalent, aussi pour tester si un ensemble de flux partage une file supérieure commune par rapport aux autres flux, nous comparerons les inter-arrivées de ces paquets avec un même nombre de paquets tirés aléatoirement. Pour plus de précision nous pouvons faire du *bagging* : recommencer l'expérience plusieurs fois sur de nouvelles sélections aléatoires.

Les résultats obtenus avec ce test se sont révélés bien plus fiables pour vérifier qu'un agrégat partage une file d'attente de débit donné plutôt que la comparaison entre ce débit et le débit estimé par la méthode précédente.

4.2 Algorithme d'inférence de topologie

Après avoir présenté le concept de test de présence de file d'attente, nous l'utilisons dans un algorithme d'inférence de topologie. Cette introduction apporte une vue globale de l'algorithme, dont nous décrivons de manière succincte les différentes étapes. La seconde partie apporte des précisions sur des points critiques de l'implémentation et les hypothèses de fonctionnement de la méthode développée.

Aperçu d'un algorithme d'inférence de topologie Ce dernier test est implémenté dans un algorithme d'inférence de topologie. Cet algorithme produit un arbre (graphe sans boucle), dont les arêtes et nœuds correspondent aux liens et files d'attente du réseau. Le réseau est découvert en remontant progressivement les nœuds de l'arbre. Partant de l'agrégat complet au point d'observation, on divise l'agrégat en groupes de flux partageant une file supplémentaire. On recommence ensuite sur chaque groupe de flux identifiés comme provenant d'une même file d'attente.

Plus en détail, il faut :

1. identifier le débit (ou temps de transmission T d'un paquet de taille standard) de la file d'attente commune à l'agrégat ;
2. effectuer des tests de présence de file d'attente supérieure pour une valeur de temps de transmission supérieure à $T <$. En dessous de cette valeur, une file d'attente supérieure ne sera pas mise en valeur, tandis que si nous choisissons une valeur trop élevée, nous risquons de manquer la détection d'un étage de l'arbre ;
3. partitionner effectivement l'agrégat en fonctions des résultats obtenus à l'étape précédente.

4.2.1 Étape 1 - Estimation du niveau d'analyse T

Entrée : agrégat complet des paquets au point d'observation.

Sortie : $\hat{T}$, estimation de T, temps de transmission et inverse du débit de la file commune à l'agrégat.

Algorithme Étant donné un agrégat de flots :

1. Calculer la fonction de répartition des temps d'inter-arrivée de l'agrégat complet ;
2. Lisser la fonction de répartition et la dériver ;

3. Définir $\hat{T}$ par l'abscisse du plus haut saut de la fonction de la fonction de répartition, c'est-à-dire au pic de densité de la distribution des temps d'inter-arrivée.

Des erreurs peuvent apparaître dans la recherche du pic de densité :

- Présence d'un pic à un quantile très grand ⇒ borner la recherche du saut de la fonction de répartition à la zone des valeurs inférieures à la médiane ;
- Le saut de la fonction de répartition n'est pas significatif ⇒ choisir une valeur, par défaut le quantile 10% des temps d'inter-arrivée de l'agrégat.

Remarques : Lisser la fonction de répartition permet d'éviter des erreurs numériques lors de la dérivation. Le filtre de lissage est ici un noyau de convolution gaussien d'écart-type $\sigma = \frac{n}{120}$ avec n le nombre de paquets observés.

Le premier type d'erreur se produit lorsqu'un sous-ensemble significatif des paquets passe par une même file d'attente fortement saturée. Elle masque alors les autres files d'attente, y compris celle de l'agrégat complet. Le risque est de ne pas détecter les autres files d'attente. Le second se produit lorsqu'il n'y a pas de file d'attente commune à l'agrégat complet. Une valeur arbitraire (par exemple 10%) permettra probablement de mettre en évidence des sous-ensembles de flots issus d'une même file en amont.

4.2.2 Étape 2 - Construction de fonctions de répartitions pour toute paire de flots (i, j)

Entrée : agrégat complet et indices des flots (i, j).

Sortie : fonction de répartition F_{ij} des inter-flots de l'agrégat (i, j) et fonction de répartition de référence F_{ij}^{ref}.

La démarche suivante sera répétée sur chaque paire de flots.

Les notations utilisées au cours de cette étape sont données dans le tableau 4.2. Notons ici l'utilisation d'inter-flots à la place des inter-arrivées. Les inter-flots de l'agrégat (i, j) sont les temps d'inter-arrivée entre deux paquets p_n et p_{n+1} consécutifs dans l'agrégat tels que l'un appartient au flot i et l'autre au flot j, $i \neq j$.

Remarque : nous aborderons dans la suite le sujet des inter-flux et intra-flux qui permet d'éviter certains problèmes, notamment ceux liés à la disparité des débits des flux. Ici le problème ne se pose pas parce que ces étapes sont illustrées avec des résultats de simulation sur des flux de loi exponentielle : la différence que nous présenterons par la suite n'existe pas ici.

Symbole	Description
(i, j)	indices d'une paire de flots
N_{ij}	$(N_i + N_j)$ somme des nombres de paquets des flots i et j
F_{ij}	fonction de répartition des inter-flots de l'agrégat des flots (i, j)
F_{ij}^{ref}	fonction de répartition de référence pour les flots i et j
K	nombre de simulations de Monte-Carlo. Nous utilisons $K = 200$
$F_{ij}^{k,\text{ref}}$	fonction de répartition des temps d'inter-arrivée de l'échantillon k

TABLE 4.2 : Notations utilisées

Démarche Étant donné une paire de flots (i, j) :

1. sélectionner les N_{ij} paquets appartenant aux flots i et j ;
2. calculer F_{ij}, la fonction de répartition des inter-flots de l'agrégat (i, j) ;
3. construire la fonction de référence par méthode Monte-Carlo en k itérations :
   a) construire K groupes de n_{ij} paquets choisis aléatoirement dans l'agrégat complet ;
   b) calculer pour tout $k \in 1, .., K$ la fonction de répartition $F_{ij}^{k,\text{ref}}$ des temps d'inter-arrivée des paquets du groupe k ;
4. calculer la fonction de référence F_{ij}^{ref} qui est la moyenne des $\{F_{ij}^{k,\text{ref}}, k \in 1, .., K\}$.

4.2.3 Étape 3 - Construction de la matrice d'adjacence

Entrée : ensemble des $(F_{ij}, F_{ij}^{\text{ref}})$

Sortie : matrice d'adjacence binaire $\boldsymbol{A}$

La matrice d'adjacence $\boldsymbol{A} = (a_{ij})$ binaire compare les fonctions de répartition des $(F_{ij}, F_{ij}^{\text{ref}})$ au niveau $\hat{T}$. Pour chaque paire de flots (i, j),

$$a_{ij} = \begin{cases} 1 & \text{si } F_{ij}(\hat{T}) < F_{ij}^{\text{ref}}(\hat{T}) \\ 0 & \text{sinon} \end{cases} \tag{4.7}$$

4.2.4 Étape 4 - Partitionnement

Entrée : matrice d'adjacence $\boldsymbol{A}$.

Sortie : liste de clusters.

Un algorithme de partitionnement est utilisé pour trouver le partitionnement optimal de l'agrégat des flots et corriger les erreurs éventuelles dans la matrice d'adjacence $\boldsymbol{A}$. Cet algorithme fournit un ensemble de classes contenant les flots contenant les flots issu d'un même nœud. Par rapport au nœud commun à l'agrégat complet, ces nœuds

- Sont situés en amont dans la topologie du point de vue de l'observateur
- Ont un débit inférieur

Nous reviendrons en détail sur ces aspects de partitionnement au chapitre suivant.

4.2.5 Étape 5 - Récursion

Les étapes précédentes ont permis d'obtenir la liste des clusters de flots qui composent l'agrégat. Ces mêmes étapes sont répétées sur chaque cluster de flots trouvé afin de construire récursivement la topologie.

Des informations sur le réseau pourraient être utilisée ici ou dans les étapes précédentes pour évaluer ou corriger les résultats :

- Adresses IP source/destination et ports source/destination des paquets
- Nombre maximal d'interfaces d'un équipement (c'est le nombre maximal de clusters à trouver)
- Débits des interfaces, aide dans l'estimation de $\hat{T}$

4.3 Hypothèses et précisions sur le fonctionnement de l'algorithme

Nous revenons ici sur divers problèmes rencontrés dans l'exécution de l'algorithme. Nous commençons par la fin, et notamment expliquons pourquoi il est important de simplifier le partitionnement et l'intérêt de le réduire à un problème de partitionnement de graphe. En conséquence, nous devons réaliser des tests de présence de file d'attente uniquement sur des paires de flux, et montrons l'importance d'utiliser les inter-flux plutôt que la distribution complète des inter-arrivées de l'agrégat des deux flux. Nous analysons ensuite le cas d'observation partielle du flux sortant de la file d'attente et expliquons l'importance de l'hypothèse de taille constante des paquets circulant dans la topologie.

4.3.1 Le partitionnement de graphe

La division d'un agrégat de flux en un nombre inconnu de sous-ensemble de flux est un problème difficile. L'idéal serait d'évaluer chaque sous-ensemble possible et sélectionner les meilleurs candidats. La complexité est bien trop élevée, aussi nous préférons tester les flux deux à deux et trouver une solution, même partielle, plutôt que pas. Pour N flux, nous restreignons donc le nombre de tests à réaliser à $\binom{N}{2} = \frac{N(N-1)}{2}$. Le nombre de tests est donc réduit à une complexité polynomiale en $O(N^2)$, ce qui est beaucoup plus acceptable. Cette simplification a un coût : la précision des tests est beaucoup moins bonne. Nous devons donc croiser les résultats des tests pour corriger les erreurs et résoudre ainsi un problème de partitionnement. Ce problème de partitionnement est connu, il a été largement abordé dans la littérature même s'il n'existe actuellement pas de garanties de performances sur celui-ci. Nous reviendrons sur ce problème dans le chapitre suivant de partitionnement de graphes.

4.3.2 Inter- et intra-flux des paquets

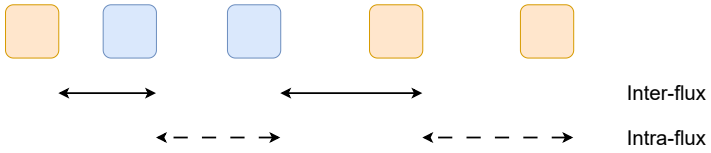


FIGURE 4.5 : Décomposition des temps inter-arrivées des paquets

La série temporelle des inter-arrivées de deux flux de paquets se décompose en deux types de mesures :

- Les intra-flux, durées qui séparent deux paquets successifs de l'agrégat appartenant au même flux
- Les inter-flux, durées qui séparent deux paquets successifs de l'agrégat appartenant à des flux différents

Quelle information contiennent les intra-flux ?

Les intra-flux sont des écarts entre paquets ayant parcouru le même chemin tout au long du réseau, contrairement aux inter-flux. Les intra-flux ont donc probablement été plus soumis aux contraintes des files d'attentes en amont et seront en moyenne théoriquement plus élevés. Nous observons ce comportement en simulation avec des flux de loi d'inter-espacement exponentielle. Plusieurs effets contradictoires vont influencer les valeurs des intra-flux :

- Le mode de fonctionnement des files d'attente
- La présence de rafales de paquets dans le flux initial
- L'accroissement des débits des files d'attente

Lorsque la file d'attente bride le débit de façon instantanée, la distribution des intra-flux reflètera cette limite de débit, qui restera potentiellement visible jusqu'au point d'observation. Ce n'est pas le cas avec nos données émulées car la première file d'attente, au contact de l'émetteur fonctionne sur le principe du *token bucket filter*. Le fonctionnement de l'algorithme de contrôle de congestion, et plus largement la méthode d'émission des paquets modifie considérablement les résultats. Intuitivement, on pourrait penser que la moyenne des inter-flux soit inférieure à celle des intra-flux. Mais lorsque le trafic est émis en rafales, la sortie de la file d'attente fera plus souvent apparaître des paires de paquets issus du même utilisateur. Contrairement à l'intuition précédente, la moyenne des intra-flux pourrait dans ce cas être supérieure à celle des inter-flux. Les rafales de paquets seront plus ou moins bien conservées par les files d'attente successives. Les rafales de paquets seront plus probablement morcelées si le débit de la nouvelle file d'attente est très supérieur au débit de la dernière file traversée par le flux.

Pourquoi seuls les inter-flux sont intéressants ?

Lorsque les flux de paquets sont multiplexés, les rafales de paquets sont bien morcelées : l'agrégat sortant est formé de rafales de paquets appartenant aux différents flux et on mesure beaucoup d'inter-flux à la valeur du temps de transmission de la file d'attente. Les inter-flux sont alors représentatifs de la première file où se sont rejoints les différents flux. Contrairement aux tests en simulation avec des flux de loi exponentielle où la différence était peu marquée, les tests sur émulateur ont contribué à largement améliorer les performances du test de file supplémentaire supérieure. Les inter-flux, qui sont les plus susceptibles de contenir l'information de présence d'une file supérieure commune, sont masqués dans les inter-arrivées deux flux de débits très différents.

4.3.3 Résultats en cas d'observation partielle des paquets

Nous n'avons pas indiqué au cours de l'algorithme l'impact d'une observation partielle du trafic. En effet, tous les paquets échangés dans le réseau ne seront pas observables en un unique point central et l'existence de certains paquets ayant contribué à remplir les files d'attentes ne sera pas connue.

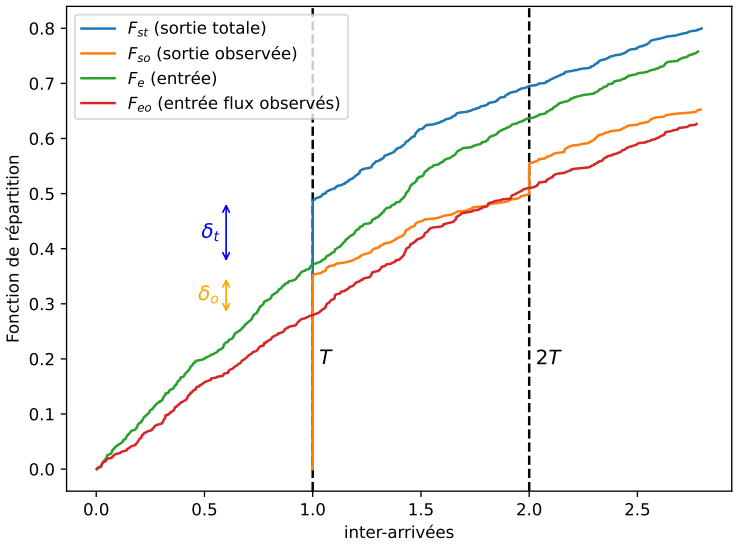


FIGURE 4.6 : Inter-arrivées de flux de paquets à taille constante ayant traversé une file d'attente

La comparaison des flux entrant et sortant de la file d'attente permet toujours de déceler la présence de la file d'attente. Cependant la marge de détection est plus faible. La figure 4.6 illustre une simple expérimentation de trafic poissonien traversant une file d'attente, mais dont seule une partie des paquets serait observée. Les fonctions de répartition F_e et F_{st} sont calculées en entrée et sortie de la file d'attente à partir de l'ensemble des paquets ayant traversé la file (qui ne génère pas de pertes), tandis que les fonctions F_{eo} et F_{so} sont calculées à partir d'une sélection aléatoire de 70% des paquets. La présence d'une file d'attente de niveau T est détectée en mesurant la valeur $\delta_o = F_{so}(T) - F_{eo}(T)$. La même opération dans une situation de connaissance totale des paquets ayant traversé la sortie donne $\delta_t = F_{st}(T) - F_e(T)$, et on constate que $\delta_t > \delta_o$. En théorie, la détection de file d'attente par observation partielle du trafic reste possible (c'est ce que nous faisons en réalité avec les tests sur des paires de flux) mais la différence est moins marqué donc la performance du test de présence de file sera moins bonne.

Enfin, en cas d'observation partielle de la sortie de la file d'attente, nous observons des sauts harmoniques de la fonction de répartition des *inter-flux* (ou *inter-flux* pondérés si les paquets ne sont pas de taille constante). Ces sauts, aussi détectable par des modes de la distribution, sont formés par l'absence d'un ou plusieurs paquets successifs non observés. La hauteur cumulée de ces harmoniques correspond au taux de saturation de la file d'attente présenté en début de chapitre. On peut donc ainsi

compenser la perte d'information causée par les paquets manquants et mesurer le taux de saturation de la file d'attente.

4.3.4 L'hypothèse de taille constante des paquets

Les difficultés apparaissent lorsqu'on observe des flux dont les paquets sont de taille variable en aval du réseau, après plusieurs files d'attente. Dans ce cas, les harmoniques ne pourront plus être mises en évidence, même en divisant par la taille des paquets, parce que la taille des paquets intermédiaires non observés est inconnue.

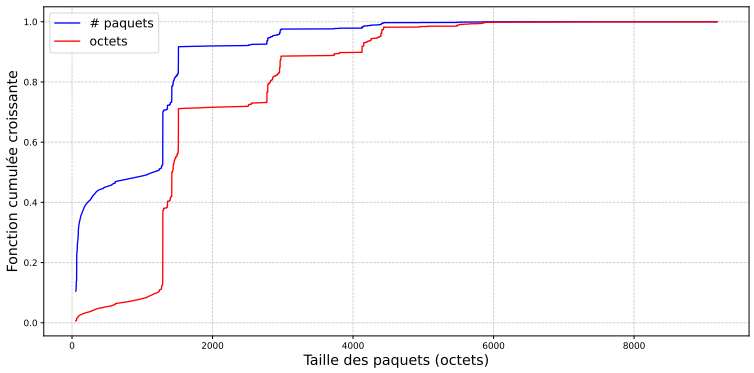


Figure 4.7 : Répartition de la taille des paquets, données MAWI.

La situation dans internet est un entre-deux : on ne peut pas dire que tous les paquets sont de taille constante, mais l'approximation reste plausible. Comme illustré figure 4.7 avec des données extraites un jour d'avril 2023 dans le réseau MAWI, les réseaux transportent globalement 2 types de paquets :

- de très petits paquets (SYN, ACK, ARP, DHCP ...) qui échangent des informations de protocole
- des gros paquets de transport de données

Ces petits paquets représentent en masse moins de 10% du trafic. Près de 60% est constitué de paquets de taille égale à la MTU. Enfin les autres sont de taille proche d'une multiple de la MTU. Dans une optique d'estimation de la saturation de la file d'attente, le manque de déterminisme de la taille des paquets pourra être plus ou moins compensé par ces inter-espacements caractéristiques et la connaissance de la répartition des différents paquets de taille proportionnelle à la MTU.

Nous supposons que cette répartition restera similaire dans tous les réseaux. Cette hypothèse sera mise à mal dans des contextes très particuliers. Par exemple, les traces

plus récentes de MAWI sont lourdement affectées par un projet de scan d'internet qui génère un nombre élevé de paquets ICMP.

Dans la suite du travail, dans nos simulations les paquets sont presque tous de même taille constante égale à la MTU.

4.4 Cas d'application

4.4.1 Présentation globale

Afin d'illustrer la méthode, nous illustrons son application sur l'expérience émulée dans le chapitre de génération des données. Les détails du réseau sont donnés §3.4.1. L'utilisation des *inter-flux pondérés* est la seule modification effectuée par rapport à l'algorithme §4.2, afin de s'adaptation aux spécificités des flux TCP-Cubic de l'émulateur. Nous la reproduisons figure 4.8, en la simplifiant au premier niveau de la topologie.

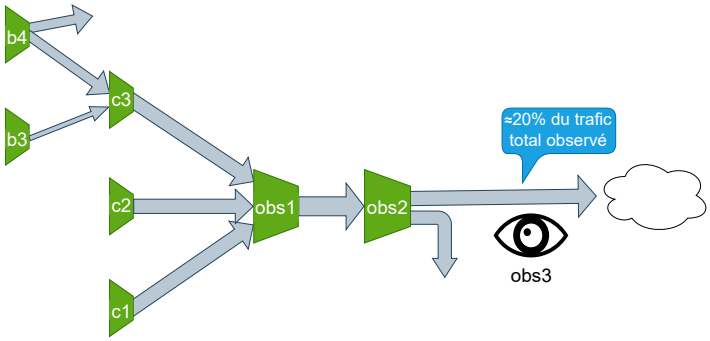


FIGURE 4.8 : Représentation du réseau émulé (idem fig 3.5)

Nous cherchons à diviser selon leur provenance **c1**, **c2** ou **c3** les flux de paquets observés en **obs3**. La figure 4.9 illustre les contraintes du test de file supérieure commune. Nous vérifions que les temps inter-arrivées observés d'agrégats de flux issus respectivement des files supérieures **c1**, **c2** et **c3** prennent rarement des valeurs inférieures aux temps de transmissions T_Y respectifs de ces files d'attente, et que ces files présentent généralement un saut de la fonction de répartition marqué en cette même valeur T_Y.

Au contraire, nous observons dans la distribution de l'agrégat complet, issu du multiplexage de tous les flots, la présence d'une relativement grande proportion de valeurs d'inter-arrivées. La zone de "choix de seuil" indique cette partie de la distribution essentiellement constituée d'inter-arrivées entre paquets issus de deux files d'attentes différentes.

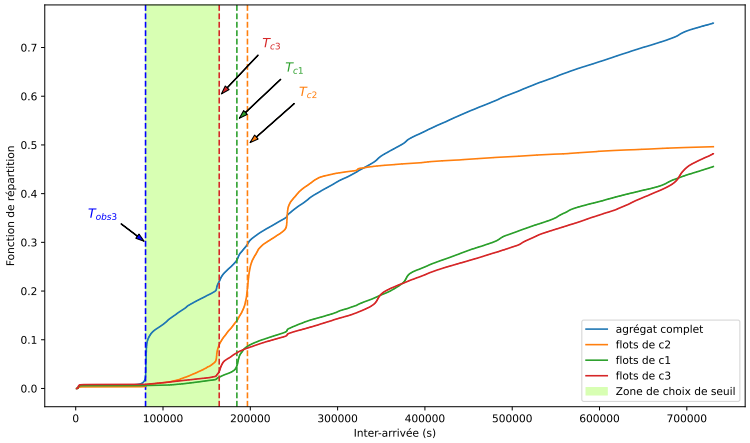


Figure 4.9 : Le multiplexage des flots génère de nombreuses inter-arrivées

Cette zone est délimitée à gauche par la file d'attente de plus faible débit commune à l'agrégat. La valeur de T $obs3$ est à relier au débit de la file d'attente **obs2**, qui masque par son débit plus faible la file **obs1**. À droite, cette zone est délimitée par la file supérieure de plus haut débit, dans notre cas la file $c3$.

Si le choix du niveau d'analyse est dans cette zone, nous pourrons marquer une différence entre un agrégat de flux aléatoire et un agrégat de flux issu d'une même file d'attente supérieure.

Notons que la file **obs1** n'est pas observable.

4.4.2 Étape 1 - Estimation du niveau d'analyse

Le niveau d'analyse $\hat{T}$ est estimé à partir de l'agrégat observé en **obs3**. Il sera choisi comme la valeur du temps d'inter-arrivée associée au plus haut saut de la fonction de répartition des temps d'inter-arrivée de l'agrégat des flots observés. Le saut le plus haut correspond à l'abscisse de la densité la plus élevée. Cette densité présente des Diracs et n'est pas simple à représenter. Cette difficulté technique est contournée en lissant la fonction de répartition (fig 4.10) avant de la dériver.

L'abscisse du plus haut saut est en pratique estimée au minimum de l'inverse de la densité (voir figure 4.11) pour réduire le risque d'erreur numérique.

Le résultat de cette étape est $\hat{T} = 80e3$ns.

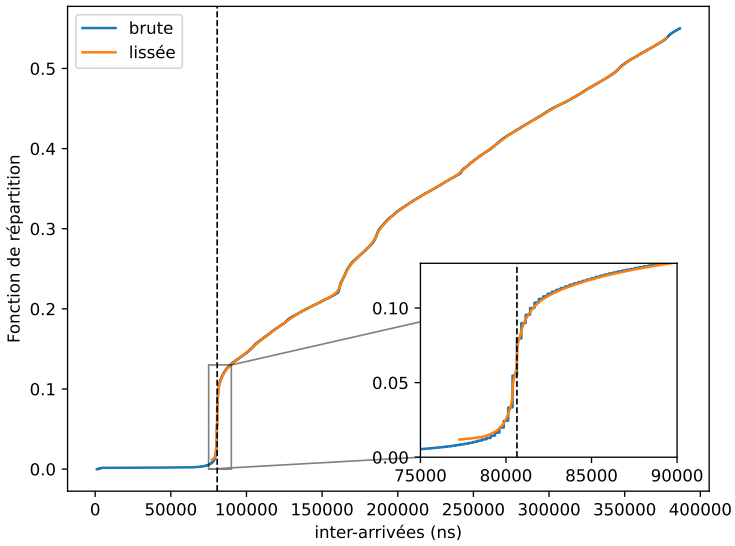


FIGURE 4.10 : Le lissage de la fonction de répartition des temps d'inter-arrivée de l'agrégat permet d'éviter des pics de densité infinie causés par des temps d'inter-arrivée identiques.

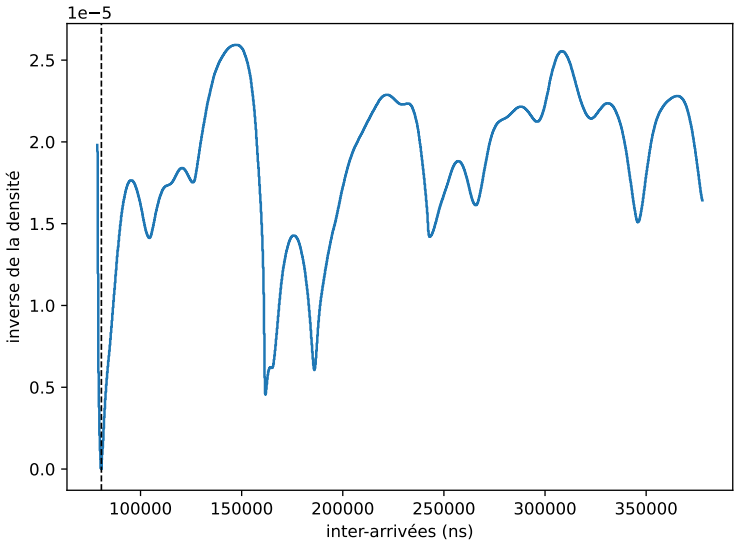


FIGURE 4.11 : Le niveau $\hat{T}$ est l'abscisse du minimum de l'inverse de la densité associée à la fonction de répartition lissée.

4.4.3 Etape 2 - Construction des fonctions de répartition référentes

Dès que le niveau $\hat{T}$ de la file d'attente est estimé, il est possible de vérifier si deux flots partagent ou non une file d'attente supplémentaire. Étant donnés deux flots, on calcule

la fonction de répartition observée de leurs temps d'inter-arrivée, puis une référence construite selon l'hypothèse d'indépendance des flots. Comme représenté précédemment figure 4.9, il existe toute une zone dans laquelle un test est possible. Cependant nous ne connaissons pas les extrémités de cette zone car le débit le plus élevé des files supérieures n'est pas connu. Il est donc préférable de comparer les deux fonctions de répartition au niveau $\hat{T}$ estimé précédemment.

La figure 4.12 illustre le cas de deux flots qui ne partagent pas de file d'attente supérieure commune au-dessus de **obs1** (non observable avec cette technique). À l'abscisse $\hat{T}$ donnée, la fonction de répartition des inter-flots observés (F_{ij}) est supérieure à la fonction de référence F_{ij}^{ref} calculée.

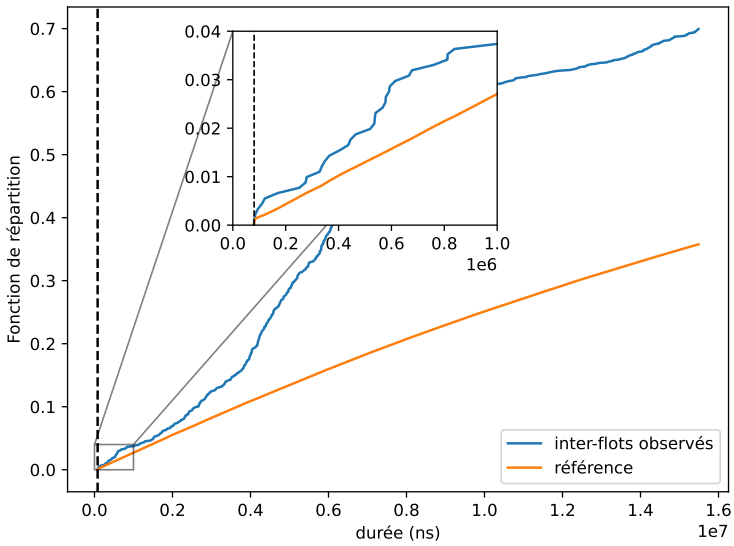


Figure 4.12 : Comparaison des fonctions de répartition de deux flots issus de **a1** et **b1** qui se rejoignent en **c1**. À $t = 80.10^3$, les flots sont indépendants et vérifient $F^{\text{ref}}(t) < F^{\text{obs}}(t)$.

La figure 4.13 illustre au contraire deux flots qui partagent une file d'attente supplémentaire commune au-dessus de **obs1** (non observable avec cette technique). À l'abscisse $\hat{T}$ donnée, la fonction des inter-flots observés (F_{ij}) est inférieure à la fonction de référence F_{ij}^{ref} calculée.

Cette étape renvoie les résultats des tests de présence d'une file supplémentaire commune au niveau $\hat{T}$ pour chaque paire de flot (i, j).

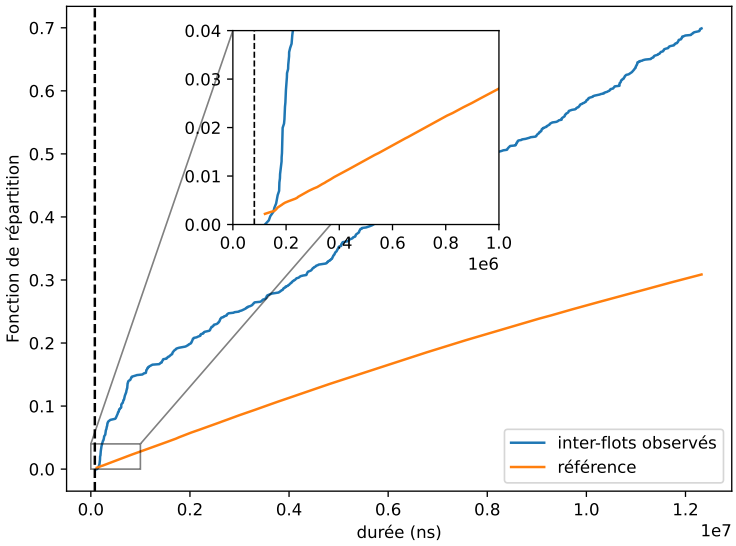


FIGURE 4.13 : Comparaison des fonctions de répartition de deux flots issus de **a1** et **b1** qui se rejoignent en **c1**. À $t = 80.10^3$, les flots sont corrélés et vérifient $F^{\text{ref}}(t) > F^{\text{obs}}(t)$.

4.4.4 Etape 3 - Construction de la matrice d'adjacence

Les calculs de l'étape précédente servent à construire un graphe dont la matrice d'adjacence est représentée figure 4.14. Pour rappel,

$$a_{ij} = \begin{cases} 1 & \Rightarrow (i,j) \text{ sont liés dans le graphe et ont probablement un goulot commun} \\ 0 & \Rightarrow (i,j) \text{ ne sont pas liés, et n'ont probablement pas de goulot commun} \end{cases} \tag{4.8}$$

La matrice théorique figure 4.14 montre les valeurs prises par les clusters. La matrice récupérée en pratique (à droite) contient beaucoup d'erreurs qu'un algorithme de partitionnement permettra de corriger (ou non selon la situation).

4.4.5 Etape 4 - Partitionnement

Pour cette étape, nous utilisons l'algorithme de partitionnement spectral, qui projette les nœuds d'un graphe dans un espace de vecteurs propres choisis, comme montré figure 4.15. En utilisant deux vecteurs propres, chaque nœud du graphe est placé par ses deux coordonnées et nous montrons ainsi visuellement que le problème est simple à résoudre. Cette projection est une étape intermédiaire qui montre le rapprochement spatial des flots issus du même goulot. Les couleurs ont été ajoutées pour mettre en évidence les différences entre les clusters, nous n'y avons pas accès en réalité. Le détail

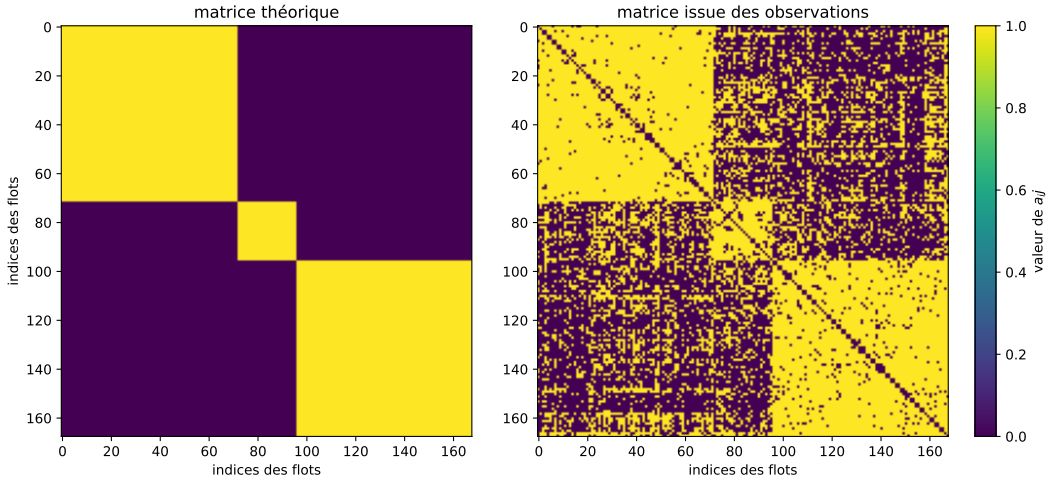


Figure 4.14 : Les flots partageant un goulot commun sont liés dans le graphe. Cela crée des clusters formant des blocs sur la diagonale de la matrice d'adjacence.

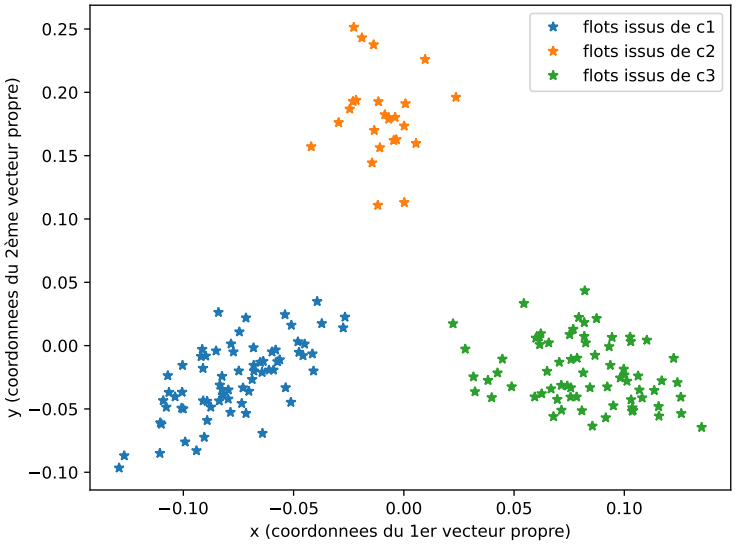


Figure 4.15 : Les flots liés dans la matrice d'adjacence sont proches spatialement dans l'espace des vecteurs propres choisis selon la méthode spectrale.

de l'algorithme sera abordé plus loin.

4.4.6 Etape 5 - Récursion

La même technique sera reproduite sur chaque cluster identifié. Le cluster des flots passés par **c3** a été correctement identifié à l'étape précédente. Le niveau d'analyse calculé à l'étape 1 sur cet agrégat de flots vaut $\hat{T} = 165632$ns. La figure 4.16 représente la matrice d'adjacence obtenue au cours de l'étape 4. Les blocs de la diagonale, clairement identifiables, sont associés respectivement à l'agrégat issu de **b3** (petit cluster) et celui de **b4** (gros cluster). Le partitionnement ne présente donc aucune difficulté.

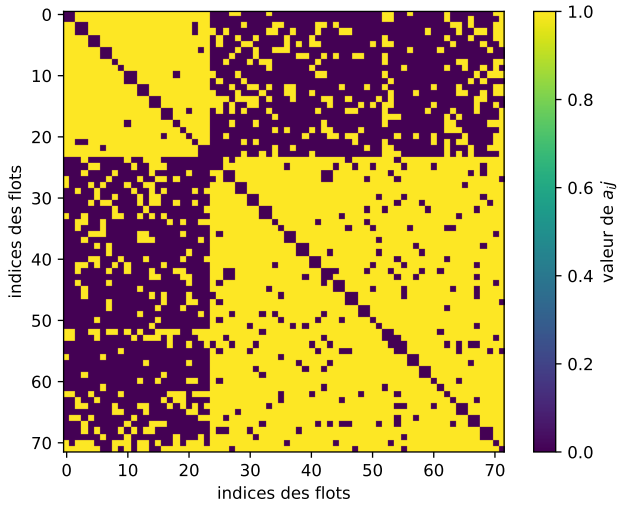


FIGURE 4.16 : Matrice d'adjacence des flots ayant traversé **c3** au niveau $\hat{T} = 165e3$ns qui met en évidence l'embranchement des files **b3** et **b4** (cf fig. 4.8).

En conclusion, la progression de l'algorithme permettra de remonter de plus en plus en amont dans le réseau. La combinaison de tous les résultats est une reconstruction de la topologie.

Des critères d'arrêt de l'algorithme pourront être spécifiés avec le nombre de flots, un niveau $\hat{T}$ à ne pas dépasser et une métrique (à définir) de partitionnement.

4.5 Comparaison avec l'état de l'art

Nous comparons les résultats obtenus avec l'entropie de Rényi (voir plus haut 4.1.1.2) et la méthode de test de file d'attente supérieure commune. Les deux méthodes sont comparées d'abord dans la situation du réseau employé par Katabi, qui est notre référence de la littérature. Dans un second temps, nous employons un réseau émulé

qui nous semblait plus intéressant à mettre en œuvre dans le cadre plus complexe de l'inférence de topologie.

4.5.1 L'expérience de Katabi

En plus d'expliquer une méthode originale pour la détection de goulot, Katabi décrit avec précision le réseau utilisé sur lequel ses résultats ont été obtenus. Les traces réseaux qu'elle a utilisé ne sont pas disponibles en ligne, mais nous pouvons aujourd'hui émuler facilement le fonctionnement d'un tel réseau et reconstituer son expérience dans un premier temps avant de comparer les résultats des différentes méthodes.

La topologie figure 4.17 est extraite de l'article [KBY01]. Au total 20 flux TCP et 2 flux UDP à débit constant saturent les goulots B1 (1,5 Mb/s), B2 (1,5 Mb/s) et B3 (1 Mb/s). Cette topologie permet de créer 3 goulots très marqués, qui masquent par leur faible débit les liens en amont.

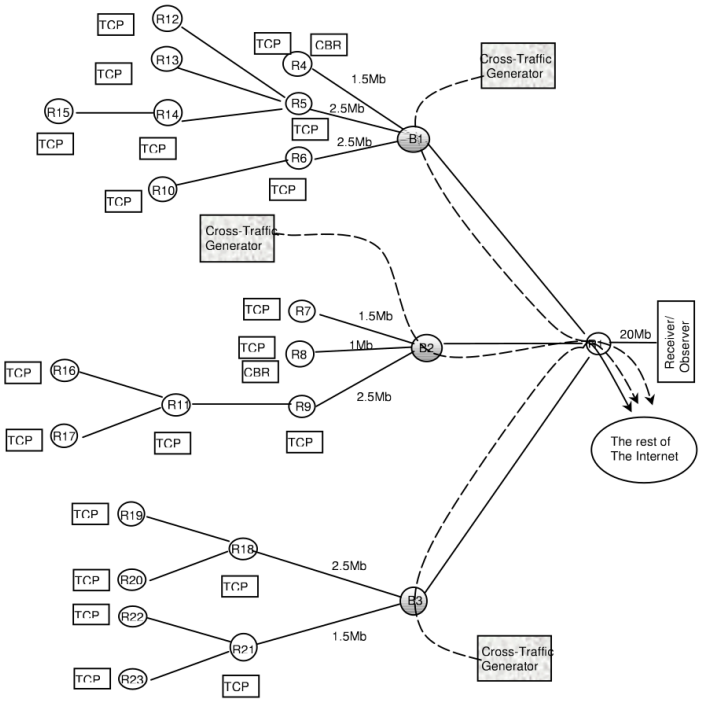
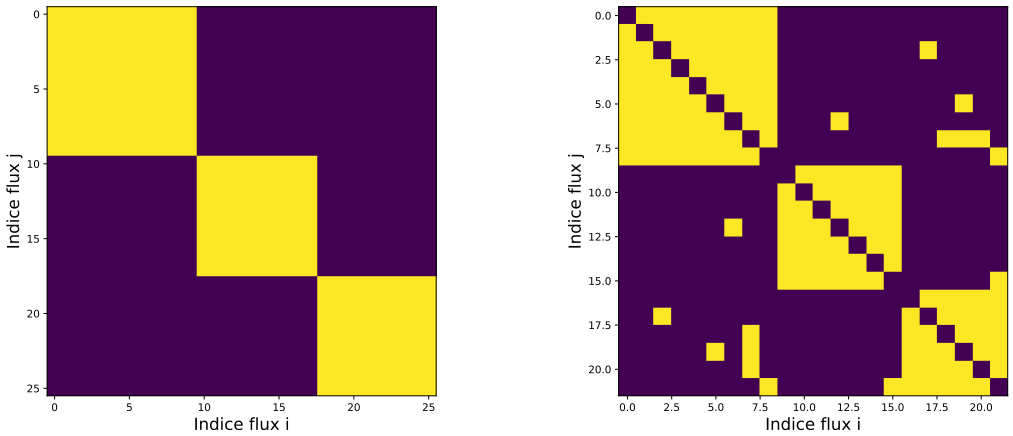


Figure 4.17 : Topologie à 3 goulots avec trafic non observé, extraite de l'article [KBY01], utilisée pour comparer la méthode par entropie de Katabi et notre méthode de détection de file d'attente.

Les flux de paquets circulent dans la topologie, sont identifiés par un numéro unique et observés au niveau de l'observateur à droite.



(a) Matrice théorique des paires de flots partageant un goulot commun

(b) Matrice résultat des tests de présence de file d'attente supérieure commune

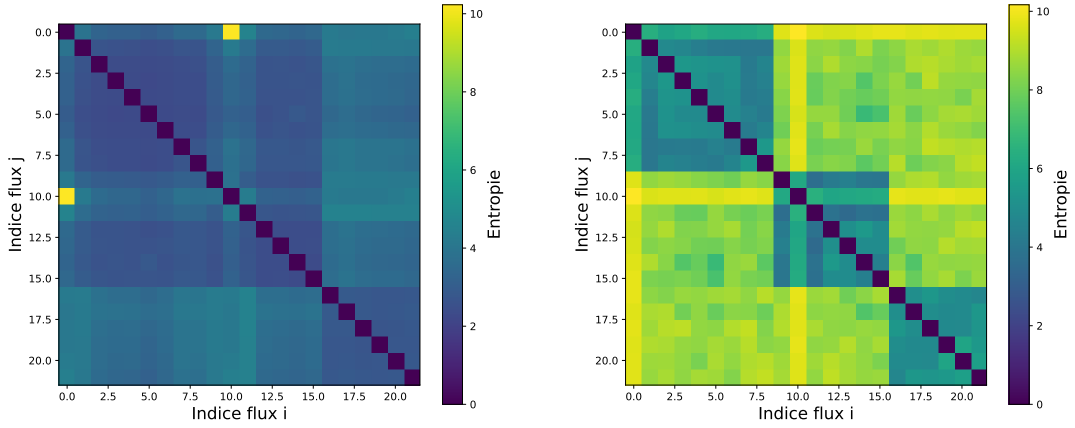
FIGURE 4.18 : Matrices d'adjacence sur les paires de flots de l'expérience fig 4.17

Le résultat attendu d'un test de présence de goulot ou file d'attente supérieur commune est représenté figure 4.18a. Comme précédemment, les résultats sont présentés par paire de flot et mettent en évidence une matrice diagonale par blocs. Les trois blocs correspondent aux trois goulots **B1**, **B2**, **B3**.

Les résultats obtenus avec le test de goulot supérieur laissent apparaître cette diagonale (voir fig 4.18b), et les résultats obtenus après classification montrent qu'effectivement les flots sont bien classés. Les calculs d'entropie de Rényi sur les agrégats de paire de flots sont présentés sur la figure 4.19. Plus l'entropie est basse, plus les flots sont supposés être ordonnés et donc la paire de flot provenir d'un même goulot. Au contraire, une entropie élevée signale que les deux flots proviennent de deux goulots distincts. Les résultats montrent que ce n'est pas si simple : à gauche, l'entropie a été calculée sur les inter-arrivées et la diagonale de bloc n'apparaît pas distinctement, contrairement à la figure de droite, où l'entropie a été calculée sur les inter-flux (voir distinction § 4.3.2). Plus spécifiquement, l'entropie confond les goulots **B1** et **B2** dont le débit de sortie est identique. Les débits identiques de ces deux files d'attente créent des temps inter-arrivées identiques entre paquets du même flux. Ces intra-flux font donc baisser l'entropie de la distribution des inter-arrivées de la paire de flux. Les inter-flux sont beaucoup moins sensibles à ce problème de files d'attente ayant un même débit.

Cependant, le calcul de l'entropie est sensible à la différence de comportement des flux. Cela est flagrant dans le cas du calcul des inter-flux où les deux flux UDP, numérotés 0 et 10, génèrent beaucoup plus d'entropie que les autres flux. Ce défaut de

l'entropie a été signalé plus haut (cf §4.1.1.2).



(a) Entropie de la distribution des inter-arrivées

(b) Entropie de la distribution des inter-flux

Figure 4.19 : Entropie calculées sur les paires de flots de l'expérience fig 4.17

Ces résultats sont donc plutôt favorables à la méthode de test de goulot supérieur. Cependant nous ne pouvons pas remonter plus en amont dans cette topologie car les débits des goulots sont inférieurs aux débits des liens en amont.

4.5.2 Comparaison sur la topologie proposée

Nous montrons ici quelques résultats de l'entropie sur la topologie introduite dans la thèse (voir figure complète 3.5 ou simplifiée 4.8).

Deux réglages différents des débits des files d'attente ont été adoptés pour montrer des comportements différents. Dans la première expérience, les débits des files d'attentes sont presque doublés à chaque étape, ce qui simplifie théoriquement l'estimation des débits des liens. Dans la seconde expérience, les débits des liens croissent moins vite, rendant plus difficile l'estimation des débits des liens.

	Source	a	b	c	obs1	obs2
Expérience 1	1	13	35	65	190	150
Expérience 2	1	13	20	30	90	50

Table 4.3 : Tableau des débits des différents étages de files d'attente

Dans ces deux expériences les débits des flux sont bridés par les liens source. Il n'y a donc pas de goulot à proprement parler, mais des files d'attente que nous tâchons de détecter.

4.5.2.1 Expérience 1 - forte croissance des débits des liens

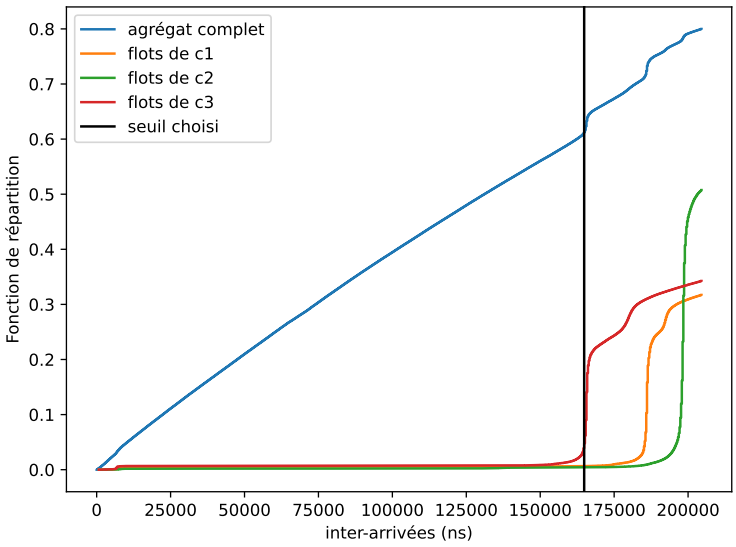
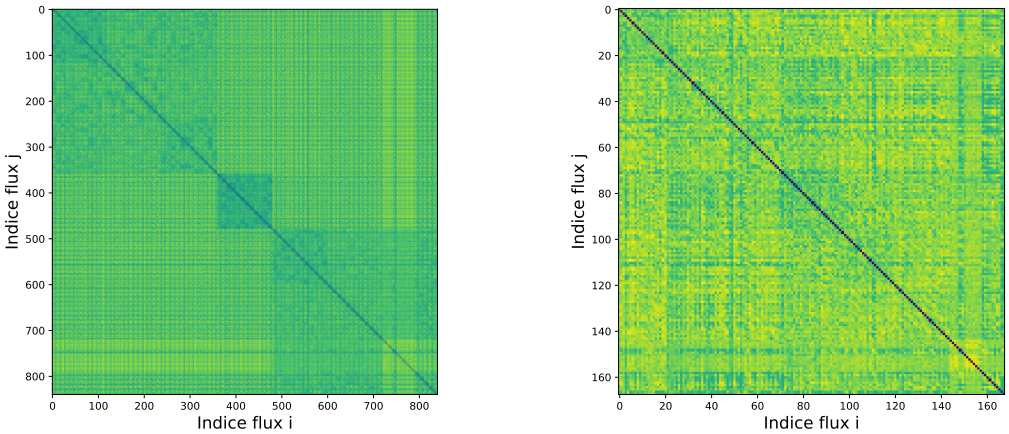


FIGURE 4.20 : Distributions des inter-arrivées au point **obs1**



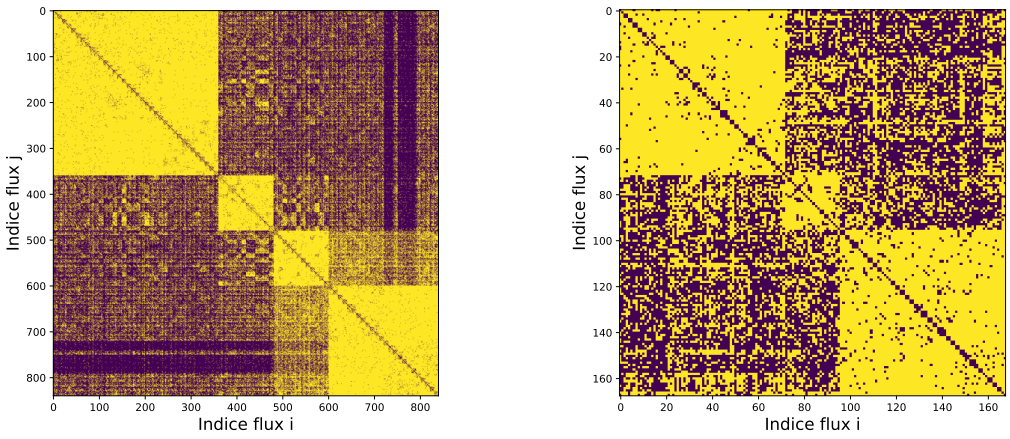
(a) Depuis **obs1**

(b) Depuis **obs3**

FIGURE 4.21 : Entropie de distributions inter-flux, données issues de l'expérience fig 3.5

Dans cette première expérience, les débits des liens sont facilement identifiables. Les données collectées au point **obs3** ont été présentées figure 4.9 ; ceux collectés au point **obs1** le sont figure 4.20.

Étonnamment, l'entropie ne permet pas d'obtenir de bons résultats. Les files d'attente saturent peu, seule une faible proportion des inter-arrivées sont modifiées par



(a) Depuis **obs1** (b) Depuis **obs3**

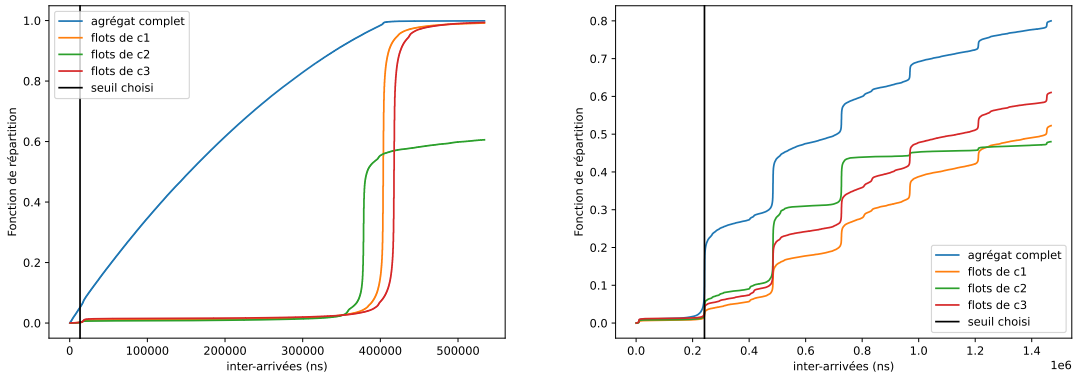
Figure 4.22 : Test de goulot commun sur des inter-flux, données issues de l'expérience fig 3.5

les différentes files d'attente. Cette faible quantité de pics dans la distribution des inter-arrivées rend l'entropie beaucoup moins efficace, et justifie son utilisation pour la recherche de goulots plutôt que de files d'attente en général. Dans cette situation, l'entropie ne permet pas de comprendre la topologie, que ce soit en **obs1** (où seul le cluster de flux passés par le point **c2** reste identifiable) ou en **obs3**. Les matrices d'entropie des inter-arrivées de paires de flux sont présentées figure 4.21.

Au contraire, cette situation est très favorable à la méthode par test de présence de goulot commun. La figure 4.22 montre des blocs bien délimités sur la diagonale de la matrice, qui correspondent aux différents clusters. Notons une petite "erreur" dans le cas des données observées en **obs1**. Ici le premier saut dans la distribution des inter-arrivées correspond à l'agrégat issu de **c3**. Cette file d'attente est donc effacée partiellement en choisissant un seuil de test au niveau de ce saut (voir figure 4.20) et nous observons des sous-clusters issus des files d'attente en amont dans la figure 4.22.

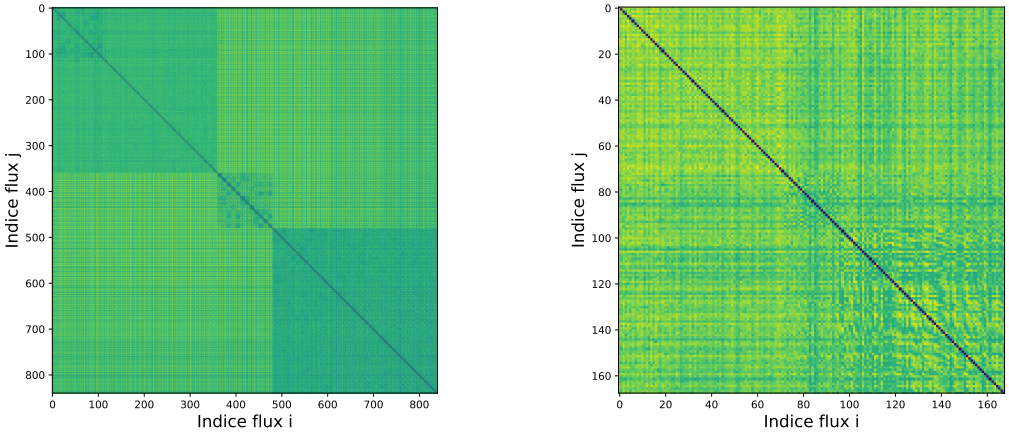
4.5.2.2 Expérience 2 - faible croissance des débits des liens

La figure 4.23 montre que les inter-arrivées collectées aux emplacements **obs1** et surtout **obs3** ont été largement modifiées par la traversée des files d'attentes successives. Ce scénario convient beaucoup plus à l'entropie de Rényi comme nous le voyons figure 4.24. Les clusters sont désormais identifiables au point **obs1**. En revanche, il reste très difficile de les observer après une seconde file d'attente, comme illustré au point **obs3**. Dans ce second cas, la dernière file d'attente discrétise les inter-arrivées et l'entropie dépend principalement du débit des flux.



(a) Depuis **obs1** (b) Depuis **obs3**

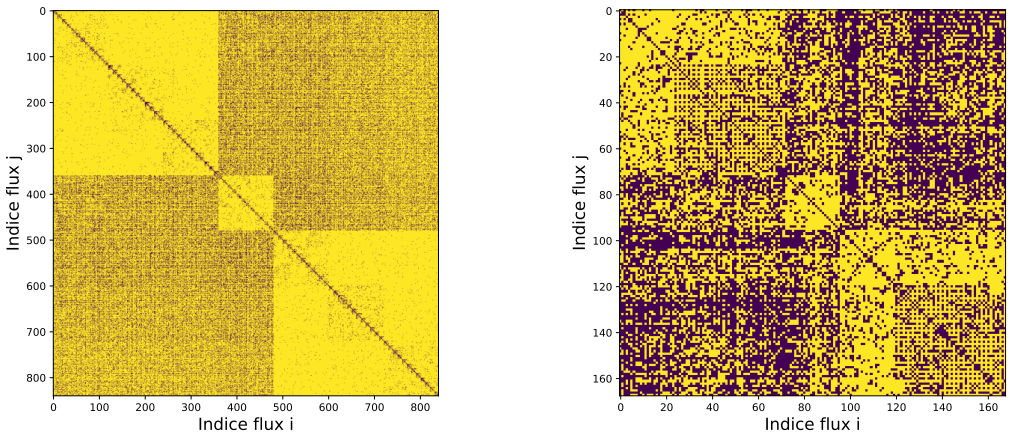
FIGURE 4.23 : Distributions des inter-arrivées



(a) Depuis **obs1** (b) Depuis **obs3**

FIGURE 4.24 : Entropie de distributions des inter-flux dans la topologie fig 3.5. Voir les débits des liens dans le tableau 4.3.

Les résultats des tests de présence de file d'attente, sont un peu moins clairs que les résultats précédents, mais restent tout à fait exploitables. Ils permettent d'identifier les clusters de flots et permettront de remonter dans la topologie comme illustré dans la figure 4.25.



(a) Depuis **obs1**

(b) Depuis **obs3**

Figure 4.25 : Test de file commune dans la topologie fig 3.5. Voir les débits des liens dans le tableau 4.3.

4.6 Récapitulatif

En résumé, il semble que les seules données utiles non chiffrées dont nous disposerons à terme dans un réseau seront les adresses IP et les ports, qui permettent de regrouper les paquets en flux. De plus, nous observons la sortie du réseau mais n'avons pas d'information a priori sur les flux entrants. Pour connaître l'état du réseau, nous avons besoin d'une information supplémentaire. Nous avons proposé un modèle de transformation fondé sur deux hypothèses :

- l'indépendance des flots, qui implique que des petites inter-arrivées se formeront au cours du mélange des flux ;
- la vitesse de traitement limitée de la file d'attente, qui interdit l'observation en sortie de petites inter-arrivées.

Nous avons vérifié cette dernière hypothèse dans le cas d'une file d'attente à traitement instantané et vitesse de traitement constante. Ce modèle pourrait éventuellement être étendu à des cas plus complexes en considérant le débit sur un intervalle de régulation de trafic donné - ou à découvrir.

En tous cas, il semble que ces hypothèses sur l'arrivée des paquets et le fonctionnement d'une file d'attente soient insuffisantes pour obtenir plus d'information sur l'état du système, notamment la taille et le remplissage des buffers. Si cela est vérifié, la détection des goulots d'étranglement devra faire appel à une autre définition, et relier la congestion au taux de paquets retardés par une file d'attente, par exemple.

Expérience	Méthode basée sur	
	l'entropie	le test de file
Scénario Katabi	succès de détection	
Scénario 1	échec	succès
Scénario 2	détection partielle	succès

TABLE 4.4 : Tableau des résultats des algorithmes

Les premiers résultats (voir tableau 4.4) de l'inférence de topologie ont été obtenus en simulation avec des flux de loi poissonienne, puis l'utilisation des intra-flux a permis de débloquer la situation dans un cas plus général.

Enfin, les dernier résultats obtenus dans des réseaux émulés avec des flux TCP montrent la possibilité de retrouver la topologie du réseau avec un tel algorithme. La difficulté du problème de classification des flux a été déplacée vers un problème de partitionnement de graphe que nous étudions à présent.

Chapitre 5

Partitionnement de graphe pour la classification de flux de paquets IP

Au cours du chapitre précédent, nous avons présenté une méthode passive d'inférence de topologie pour connaître le réseau. Cette méthode vise à détecter les goulots d'étranglement à proximité d'une station sol, ce qui serait synonyme de congestion et donc de dégradation de performance pour les utilisateurs. Fondamentalement, il s'agit de classer des flux de trafic IP par chemin commun. Partant de la station sol, on commence par diviser les flux IP en fonction du satellite dont ils proviennent, puis on recommence en divisant à chaque fois les flux en fonction de leur provenance. Chaque étape nécessite donc la résolution d'un problème complexe de partitionnement des flux. La méthode de Katabi, qui nous a servi de référence au chapitre précédent pour la résolution d'un partitionnement des flux, manipule de longues séries d'instants d'arrivées de paquets et effectuer de nombreuses et coûteuses opérations de tri. Cette approche ne passe pas à l'échelle lorsque les nombres de flux et de paquets sont très élevés. Nous avons donc proposé au chapitre précédent de simplifier le problème en combinant uniquement les paires de flux. Cette simplification réduit considérablement la taille des données à manipuler, mais ne diminue pas la complexité du problème de partitionnement, comme nous le verrons.

Pour résoudre le partitionnement d'un ensemble de flux IP, nous passons donc par le partitionnement d'un graphe binaire, dont les nœuds sont associés aux flux tandis que les arêtes sont le résultat d'un test de présence de nœud supérieur commun. Ce problème a été largement étudié dans la littérature. Nous présentons en première partie différentes techniques de la littérature et choisissons celle qui nous semble avoir le meilleur compromis entre rapidité et performance.

Les deux parties suivantes sont dédiées à l'étude de fiabilité d'une solution : la seconde partie compare ainsi des modèles de graphes qui altèrent un graphe idéal partitionné par ajout ou suppression d'arêtes selon une loi de probabilité. Enfin la dernière partie est une application des modèles de graphe : on évalue la qualité d'un partitionnement en comparant le résultat de l'algorithme proposé en première partie avec un résultat issu du partitionnement "optimal", obtenu en appliquant le modèle de graphe à un graphe expérimental.

5.1 Algorithmes de partitionnement de graphe

Un algorithme de partitionnement effectue en boucle deux étapes :

1. proposer une partition, selon une méthode de partitionnement ;
2. évaluer la partition, selon une fonction objectif.

Ces deux étapes sont répétées jusqu'à atteindre un critère d'arrêt.

La fonction objectif traduit le compromis entre le nombre de classes et la dispersion des données. Lorsque le nombre de classes est connu d'avance, le problème est largement simplifié, la complexité du problème de partitionnement est polynomiale. Sa complexité est exponentielle dans notre cas car ce nombre de classes n'est pas connu d'avance. Trouver la solution exacte nécessiterait d'évaluer toutes les partitions possibles. Cette option n'est pas envisageable et la littérature présente des heuristiques pour n'évaluer que des partitions d'intérêt. Nous étudions ces méthodes dans une première partie avant

de présenter les fonctions objectifs et plus généralement des critères de partitionnement dans une seconde partie. Cette partie de revue de la littérature aboutit à la sélection d'un algorithme de partitionnement de graphe, et donc de classer les flux IP (alias nœuds du graphe) par nœud réseau commun (alias classes de partitionnement).

5.1.1 Méthodes de partitionnement

Le partitionnement est un problème largement abordé dans la littérature comme en témoigne la diversité des algorithmes. L'article [For10] référence de nombreuses méthodes ; de même scikit-learn [PVG+11] en récapitule un certain nombre, présentées dans la figure 5.1. Nous observons dans ce tableau que le partitionnement spectral est le seul à retrouver les classes (*clusters*) les plus intuitives pour les 5 premières expériences, mais qu'il échoue dans le dernier cas où il n'existe aucune structure dans les données.

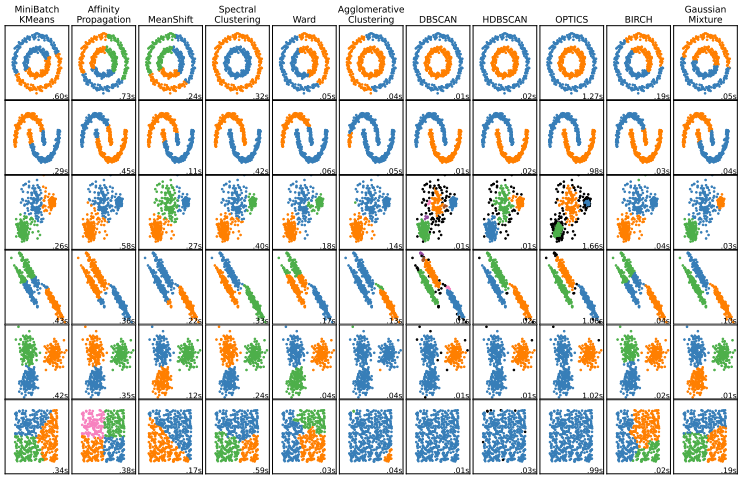


FIGURE 5.1 : Comparaison d'algorithmes de partitionnement, figure extraite de l'article Clustering scikit-learn

Nous n'étudierons pas séparément ces nombreuses méthodes, mais présentons les différentes stratégies utilisées dans ces méthodes de partitionnement. Nous traitons dans un premier temps de ces stratégies permettant la résolution du problème à proprement parler, puis décrivons le partitionnement spectral, une technique de projection dans un espace de dimension réduite qui simplifie le problème.

5.1.1.1 Approches de classification par regroupement ou partitionnement de nœuds

Il existe trois grandes familles méthodes gloutonnes dans la littérature. La première famille de méthodes "ascendante" procède par agglomérations successives (e.g. [BCGH18] ou Katabi [KBY01]) : les nœuds initialement isolés sont agrégés progressivement. La deuxième famille fait exactement le contraire. Dans cette approche descendante, tous les nœuds appartiennent initialement à une même classe qui est progressivement décomposée. Le graphe est découpé par bissections successives (par ex [New06b]). Ces deux premières familles de méthodes ont un point commun : toutes deux aboutissent à la création d'un dendrogramme, arbre décrivant les fusions ou divisions de classes, et résolvent le problème en trouvant la meilleure coupe de ce dendogramme.

La troisième famille de méthodes regroupe tous les algorithmes qui explorent les partitions de proche en proche. Partant d'une première classification des nœuds, ces algorithmes testent différentes partitions plus ou moins proches et sélectionnent la meilleure partition ou un ensemble de bonnes partitions pour finalement déterminer la meilleure option. Ce processus peut comporter une part d'aléatoire, comme dans les algorithmes génétiques, de recuit simulé [DCM18], ou les différentes techniques d'échantillonnage. À chaque étape une nouvelle solution sera proposée et choisie à partir d'un tirage aléatoire de solutions voisines. Ce choix aléatoire peut être pondéré selon une fonction score. Au contraire, certains algorithmes comme K-means associent directement les nœuds à la classe la plus probable dans un contexte géométrique. Ils n'intègrent pas de choix aléatoire mais aboutissent directement à la meilleure solution voisine.

Il est possible de combiner l'une des deux premières méthodes avec la troisième. Par exemple, Katabi commence par une initialisation agglomérative et termine par une méthode d'exploration aléatoire permettant de revenir sur de mauvais choix initiaux. Dans nos expériences le coût de calcul de l'entropie de Rényi utilisée par Katabi devient rapidement trop élevé : chaque nouveau calcul implique de trier les instants d'arrivée des paquets et calculer la distribution en segmentant les inter-arrivées sur de petits intervalles. Plutôt que de calculer une fonction de coût globale, nous avons préféré simplifier le problème de partitionnement des flux avec une unique fonction objectif en un problème de partitionnement de graphe.

Toutes ces techniques proposent d'explorer un sous-ensemble plausible des partitions du graphe. Elles ne fournissent donc pas de garantie d'optimalité.

5.1.1.2 Le partitionnement spectral : transformations graphe - espace euclidien

À l'origine, le partitionnement spectral est une technique de classification de points définis par des coordonnées réelles. Cette technique consiste à projeter les données dans un nouvel espace porté par des vecteurs orthogonaux qui maximisent la dispersion des données et simplifieront la classification. Nous décrivons dans un premier temps la méthode complète avant d'insister sur l'étape cruciale de calcul du spectre.

La méthode de partitionnement spectrale transpose un problème géométrique dans un graphe avant de revenir à un problème géométrique simplifié. Cette méthode effectue les étapes suivantes :

1. calcul d'une matrice d'affinité, simplifiée en une matrice d'adjacence associée à un graphe pondéré ;
2. calcul du spectre (valeurs et vecteurs propres) de la matrice d'adjacence puis projection des nœuds dans un espace euclidien de dimension réduite, dont les axes sont les vecteurs propres sélectionnés (nous reviendrons plus loin sur ce choix) ;
3. Le partitionnement est réalisé dans ce nouvel espace à l'aide d'une méthode parmi celles citées précédemment, par exemple K-means.

Nous apportons quelques précisions sur ces étapes.

Transposition de points définis par des coordonnées euclidiennes dans un graphe pondéré La première étape de projection des points dans un graphe peut être effectuée de la façon suivante. Tout d'abord, une matrice est créée en calculant une métrique d'affinité f_{ij} sur toutes les paires de points (i, j). Cette métrique est souvent un noyau, qui fait intervenir la norme d'une transformation linéaire des coordonnées (x_i, x_j) associées aux points (i, j). La fonction de base radiale gaussienne d'hyperparamètre γ est couramment utilisée :

$$f_{ij} = \exp(-\gamma |x_i - x_j|^2). \tag{5.1}$$

Remarquons que le choix de la métrique dans cette première étape doit être effectué en cohérence avec celui de la dernière étape du partitionnement spectral. En effet, le choix de la métrique est une hypothèse implicite sur la répartition des points. En proposant ce noyau gaussien, on considère implicitement que les données suivent des lois normales dont on cherchera à déterminer les moyennes. Maximiser la log-vraisemblance

d'observations obéissant à des lois normales revient à minimiser une somme de variances. Nous entrons ici dans le choix de la fonction objectif que nous aborderons plus en détail par la suite. Ainsi, dans l'implémentation l'algorithme de partitionnement spectral proposée par scikit-learn, le regroupement final est réalisé par défaut à l'aide de K-means, qui peut être vue comme une version "dure" de GMM, où les points ne sont assignés qu'à un seul cluster et dont on suppose que les covariances sont égales et isotropes. L'utilisation d'autres métriques pousserait à passer au cas plus général d'un algorithme d'Estimation-Maximisation (*Expectation-Maximisation, EM*). La matrice d'affinité ainsi calculée est simplifiée pour construire un graphe pondéré dont la matrice d'adjacence $A = (a_{ij})$ ne conserve que les arêtes de poids fort. Des techniques pour optimiser conjointement l'hyperparamètre γ et le choix du seuil de conservation des arêtes sont déployées dans les problèmes de classification par machines à support de vecteur *Support Vector Machine*. L'article [HCL+03] propose une méthode pratique, tandis que le chapitre 6 du livre [BN06] détaille les métriques d'affinité.

La première étape de construction du graphe n'est pas nécessaire dans notre cas, car nous avons déjà la matrice d'adjacence. Le calcul de cette matrice d'adjacence entre des flux IP a été décrit au chapitre précédent dans la méthode d'inférence de topologie passive.

Projection des nœuds d'un graphe dans un espace euclidien de dimension réduite Nous entrons dans la partie élégante du partitionnement spectral qui projette les nœuds d'un graphe associé à une matrice d'adjacence A (telle que calculée précédemment) dans un espace euclidien de dimension proche du nombre de classes.

Dans la matrice d'adjacence, chaque ligne peut être vue comme une nouvelle représentation des coordonnées d'un nœud, dont la position s'exprime en fonction des autres nœuds du graphe. Le calcul du spectre de la matrice d'adjacence projette les nœuds dans un nouvel espace, dont les axes sont les vecteurs propres de la matrice d'adjacence. Le calcul des vecteurs propres, utilisé dans le cadre de l'Analyse en Composantes Principales (ACP), consiste à projeter successivement les nœuds selon des directions orthogonales, en maximisant à chaque fois la variance selon l'axe de projection.

Dans cette nouvelle base, le vecteur propre i contient la $i^{\text{ème}}$ coordonnées de chaque nœud. Seules les D premiers vecteurs sont conservés pour partitionner les nœuds dans un nouvel espace $\mathbb{R}^D$ de dimensions réduites.

Le premier vecteur propre calculé est un vecteur unitaire normalisé. Il ne contient pas d'information et n'est pas utilisé. Le second vecteur propre est connu comme le vecteur de Fiedler. Sa valeur propre est la deuxième plus faible et détermine la coupe maximale, qui divise le graphe en deux sous-ensembles les plus éloignés possibles. Les

vecteurs propres suivants divisent aussi le graphe, mais la séparation sera de moins en moins nette. Dans ce nouvel espace, chaque classe sera identifiée par un vecteur qui détermine sa direction. Le nombre de classes dépend directement de D ; on observe un maximum de $K = D + 1$ classes après projection du graphe.

Les articles [SM00] et [VL07] recommandent de travailler avec le Laplacien normalisé plutôt que la matrice d'adjacence brute. En effet, les classes sont projetées selon diverses directions dans le nouvel espace défini par les vecteurs propres. Dans ce nouvel espace, les classes sont identifiables par la direction de leur vecteur moyen. La normalisation *"random walk"* a l'avantage d'homogénéiser la norme des coordonnées des points dans le nouvel espace de projection. Cela améliore la performance d'un algorithme comme K-means qui est sensible à la distance et non à la direction.

Le Laplacien normalisé *"random walk"* est défini dans l'équation (5.2) :

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1}(\boldsymbol{A} - \boldsymbol{D}). \tag{5.2}$$

Cette équation fait intervenir la matrice identité $\boldsymbol{I}$ et la matrice diagonale des degrés des nœuds $\boldsymbol{D}$. $\boldsymbol{D}$ est une matrice dont tous les termes sont nuls exceptés ceux de la diagonale qui sont définis par :

$$\boldsymbol{D}_{ii} = \sum_j a_{ij} \tag{5.3}$$

L'algorithme de partitionnement spectral d'un graphe Les optimisations et choix présentés dans cette partie sur le partitionnement spectral ont abouti à l'algorithme 1.

Algorithme 1 : Partitionnement spectral [SM00]

Data : matrice d'adjacence du graphe $\boldsymbol{A}$, nombre de classes K
Calculer le Laplacien normalisé $\boldsymbol{L}$ (voir eq (5.2))
Calculer $\boldsymbol{U}$ la matrice des K premiers vecteurs propres $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$ associés aux plus petites valeurs propres du Laplacien
Chaque ligne i de $\boldsymbol{U}$ correspond aux coordonnées projetées du nœud i dans un espace de dimension K.
Partitionner les nœuds en K classes avec l'algorithme K-means.

La vitesse d'exécution de l'algorithme de spectrale doit beaucoup à une initialisation très efficace, nommée Kmeans++ [AV06]. Le choix de D, crucial pour identifier les classes, n'est a priori pas connu. Son choix est débattu dans la littérature. Plusieurs méthodes font référence aux valeurs propres pour choisir ce nombre. Typiquement, le

dernier vecteur propre sélectionné sera celui dont la différence avec la valeur propre suivante sera la plus élevée. Une autre méthode, proposée dans l'article [For10] et décrite dans l'algorithme propose d'estimer ce nombre de classes a posteriori, en repartant du graphe non projeté. L'algorithme fonctionne donc sur la base d'une fonction objectif qui définit la meilleure partition obtenue.

5.1.2 Métriques d'évaluation d'un partitionnement

Cette partie décrit des métriques permettant d'évaluer la qualité d'un partitionnement. Nous abordons dans un premier temps le critère de Fisher et des modèles de vraisemblance utilisés pour partitionner le graphe dans l'espace euclidien projeté, puis la modularité, qui est un critère de partitionnement de graphe, et enfin le Indice de Rand Ajusté (ARI), qui permet de comparer deux partitions.

5.1.2.1 Le critère de Fisher

L'ACP projette successivement les données selon des axes qui maximisent la variance des points projetés. Nous montrons qu'elle permet de facilement séparer les données appartenant à des classes différentes. Les premiers algorithmes de partitionnement de graphes utilisaient une fonction objectif définie à l'aide d'une métrique de coupe minimale (théorème *max-flow min-cut*) [KL70]. La coupe minimale est donnée par le premier vecteur propre (dit de Fiedler) de la matrice d'adjacence. Selon la coordonnée, positive ou négative de chaque nœud dans le premier vecteur propre, le nœud sera classé dans l'une ou l'autre classe. Deux classes sont ainsi formées et l'opération recommence pour chaque cluster.

L'ACP met en évidence une corrélation entre les données et permet de mettre en évidence un premier partitionnement. Une technique complémentaire à l'ACP, nommée Analyse Discriminante Linéaire (ADL), peut être utilisée pour évaluer la pertinence et affiner le partitionnement. L'ADL recherche les axes qui maximisent une fonction de coût, le critère de Fisher, qui est un ratio entre des variances inter- et intra-classe. Les axes trouvés, la valeur de ce critère peut être employée pour mesurer la séparation des données. Nous introduisons d'abord les notations suivantes :

K : le nombre de classes, avec $K \geq 2$.

W_i : l'ensemble des éléments de la classe i.

m_i : le centroïde (moyenne) des éléments de la classe i.

m : le centroïde global de tous les éléments.

n_i : le nombre d'éléments dans la classe i.

La matrice de dispersion intra-classe est définie par :

$$S = \sum_{i=1}^{K} S_i \quad \text{où} \quad S_i = \sum_{x \in W_i} (x - m_i)(x - m_i)^t \tag{5.4}$$

La matrice de dispersion totale est donnée par :

$$T = \sum_{i=1}^{K} \sum_{x \in W_i} (x - m)(x - m)^t = S + \sum_{i=1}^{K} n_i (m_i - m)(m_i - m)^t \tag{5.5}$$

Enfin, la distance entre moyenne entre les classes est définie par

$$B = \sum_{i=1}^{K} n_i (m_i - m)(m_i - m)^t \tag{5.6}$$

Le critère de Fisher vise à maximiser la séparation entre les classes tout en minimisant la dispersion intra-classe. Cela revient à maximiser la fonction suivante, sous contrainte que $||u|| = 1$: $I(u) = \frac{u^t B u}{u^t T u}$. Cela est équivalent à maximiser $J(u) = \frac{u^t B u}{u^t S u}$.

Les vecteurs de séparant les classes sont trouvés par un nouveau calcul de valeur propre sous contrainte que $||u|| = 1$.

$$S^{-1} B u = \lambda u \tag{5.7}$$

Où λ est une valeur propre et u un vecteur propre qui lui est associé et qui indique une direction de séparation entre des classes.

Enfin, le critère de Fisher est souvent formulé comme :

$$J = \text{trace}(S^{-1} B) \tag{5.8}$$

L'article [SM00] montre qu'optimiser la coupe normalisée *Ncut* revient à trouver les valeurs propres du Laplacien normalisé défini équation (5.2), et finalement optimiser le critère de Fisher. Les deux articles [SM00] et [VL07] indiquent que ce partitionnement est biaisé et favorise des classes de tailles égales.

5.1.2.2 K-means et la méthode EM

L'implémentation de l'algorithme de partitionnement spectral repose souvent sur l'algorithme k-means, qui peut être interprété comme un cas spécifique de l'algorithme

EM [DLR77]. Dans ce contexte, K-means correspond à un modèle de mélange de gaussiennes isotropes (variances identiques et matrices de covariance sphériques).

L'algorithme EM est composé de deux phases :

- la phase d'Espérance (E) calcule les probabilités d'appartenance de chaque point à chaque classe, en utilisant les paramètres actuels du modèle.
- la phase de Maximisation (M) met à jour les paramètres du modèle (centroïdes, variances, etc.) pour maximiser la vraisemblance des données, en utilisant les probabilités calculées lors de la phase E.

Dans le cas le plus simple, comme K-means, maximiser la vraisemblance revient à associer chaque point à la classe dont le centroïde est le plus proche. Cette hypothèse est cohérente avec certains modèles de graphes, comme les Stochastic Block Models que nous aborderons par la suite et où les nœuds d'une même classe ont une probabilité similaire d'être connectés. Bien que cette méthode soit simple et robuste, il est possible de la généraliser en utilisant des modèles plus complexes, comme le Modèle de Mélange de Gaussiennes (GMM), qui permet d'estimer des variances différentes pour chaque classe et ainsi de modéliser des classes de tailles et de formes variées.

5.1.2.3 La modularité

La modularité est une fonction objectif de partitionnement de graphe définie par Newman [New06b]. Cette fonction est basée sur une intuition de parcours des nœuds d'un graphe pondéré dans une marche aléatoire. Partant d'un nœud i, la probabilité d'arriver sur un autre nœud sera plus élevée si ce nœud est de même classe que i que s'il ne l'est pas.

Nous introduisons les notations suivantes pour définir ensuite la modularité Q.

N : variable aléatoire associée au choix d'un nœud du graphe.

i : indice d'un nœud du graphe.

C : variable aléatoire associée au choix d'un nœud appartenant à une certaine classe.

c_i : indice de la classe du nœud i.

A : matrice d'adjacence du graphe.

a_{ij} : poids de l'arête entre les nœuds (i, j).

$$Q = \sum_i P(N = i, C = c_i) - P(N = i)P(C = c_i) \tag{5.9}$$

La qualité d'un partitionnement pourra être évaluée à partir de la modularité : plus elle est grande, meilleur est le partitionnement. Nous explicitons les différents termes de la modularité.

Une marche aléatoire sur un graphe pondéré est un processus stochastique discret dans lequel, à chaque étape, la probabilité de passer à un nœud voisin est proportionnelle au poids de l'arête qui relie les deux nœuds. On suppose que le graphe est connexe, c'est-à-dire qu'il existe un chemin entre n'importe quelle paire de nœuds. Alors, la probabilité d'être dans un nœud i converge vers un état stationnaire tel que la probabilité d'être dans un nœud est donnée par la fraction du poids de ses arêtes sur le poids total du graphe.

$$P(N = i) = \frac{\sum_j a_{ij}}{\sum_{j,l} a_{jl}} \tag{5.10}$$

De la même façon, on peut définir la probabilité de choisir aléatoirement un nœud appartenant à la même classe c_i qu'un nœud i peut se définir par la fraction entre le poids des arêtes reliant des nœuds de la classe c_i divisé par le poids total du graphe.

$$P(C = c_i) = \frac{\sum_{j,l \in c_i^2} a_{jl}}{\sum_{j,l} a_{jl}} \tag{5.11}$$

En fait, la probabilité observée de choisir un nœud de la même classe que i est définie par le ratio entre le poids des arêtes du nœud i qui sont connectées à des nœuds de même classe avec le poids total des arêtes de i :

$$P(C = c_i | N = i) = \frac{\sum_{j \in c_i} a_{ij}}{\sum_j a_{ij}} \tag{5.12}$$

Enfin, la probabilité, de sélectionner un nœud i et de rester sur un nœud de même classe c_i en une étape est donnée par :

$$P(N = i, C = c_i) = P(C = c_i | N = i)P(N = i) \tag{5.13}$$

$$= \frac{\sum_{j \in c_i} a_{ij}}{\sum_{j,l} a_{jl}} \tag{5.14}$$

$$= \frac{\text{Nombre d'arêtes de i internes au cluster}}{\text{Nombre total d'arêtes}} \tag{5.15}$$

La modularité a démontré son efficacité sur de nombreux problèmes et ses résultats

sont cohérents avec ceux obtenus par la méthode des valeurs et vecteurs propres. La relation entre ces deux méthodes est approfondie dans l'article [New06a].
Lorsque la matrice d'adjacence suit un modèle probabiliste défini, une fonction objectif de référence est obtenue par maximum de vraisemblance. Malheureusement, ces modèles font intervenir des paramètres qui ne sont pas toujours connus a priori et qu'il est souvent difficile d'estimer conjointement avec le partitionnement. Notons que la modularité peut être obtenue à partir de la vraisemblance d'un modèle probabiliste appelé *Planted Partition Model* [New16], nous reviendrons sur ce sujet plus loin.

La modularité n'est pas parfaite, mais souffre de quelques inconvénients. [BDG+08] remarque qu'elle forme un plateau local autour de la bonne solution : dans certaines situations, la modularité n'est pas suffisamment convexe et certains nœuds demeurent difficiles à classer. Conjointement avec l'article [GdMC10], ils notent une difficulté à détecter les petites communautés, il y a une sorte de limite de résolution dans la modularité. Enfin, la structure des classes ne modifie pas le score de modularité, seule les proportions d'arêtes intra et inter-classes sont prises en compte. Est-ce problématique ? Ce résultat est contre-intuitif.

Constatant que le modularité ne prend en compte que le saut suivant, [MSH15] et [GSA+15] étendent cette définition de la modularité à des marches aléatoire plus longues. Sa modularité généralisée est donc une différence de proportion entre les chemins reliant deux nœuds de façon aléatoire ou parcourant uniquement les nœuds d'un même cluster.

Adjusted Rand Index L'ARI (Indice de rand Ajusté), défini dans l'article [GA17] est un indice de la performance d'un partitionnement. Il ne repose pas sur l'analyse d'un graphe, mais compare la solution trouvée avec la partition attendue par un comptage des paires de nœuds correctement classées. Les valeurs de ce score sont comprises entre -1 (partition complètement différente) et 1 (partition conforme à la vérité terrain), sachant que $\text{ARI} = 0$ indique un partitionnement aléatoire. La matrice d'adjacence de référence associée au graphe idéal est présentée dans la figure 5.2a. Cette matrice est utilisée dans toutes les expériences de cet article. Les matrices d'adjacence estimées seront toujours présentées avec le même ordre des nœuds pour mettre en évidence les classes correspondant à des blocs sur la diagonale.

Il existe d'autres métriques de score (la silhouette [Rou87], entre autres) que nous ne détaillons pas.

Après avoir proposé un algorithme de partitionnement de flot, deux étapes sont encore nécessaires :

1. évaluer la performance de l'algorithme.

2. estimer la fiabilité d'un résultat

L'évaluation des performances doit être réalisée en pratique, mais si possible aussi en théorie avec un calcul de bornes. Selon les conditions de l'expérience, les files d'attente du réseau seront plus ou moins observables. La difficulté du problème n'est pas facile à régler. L'utilisation de modèles de graphes dont on peut paramétrer le bruit permet construire des abaques de performances, établir des bornes et ainsi contourner le problème. En sens inverse, pour estimer la fiabilité d'un résultat, on peut chercher les paramètres du modèle le plus vraisemblable et calculer la performance de l'algorithme sur ce modèle.

5.1.3 Choix d'un algorithme de partitionnement spectral

Nous utilisons une version d'un algorithme de partitionnement spectral contrôlée par la modularité Q, reprenant une idée de [WS05]. Les auteurs ont proposé une première version lente, qui évalue le partitionnement de tous les nombres de classes, puis une seconde version rapide par division et évaluation de cluster. Nous reprenons la première version et ajoutons à la fin un algorithme de ré-échantillonnage 3 des classes pour améliorer la solution.

Algorithme 2 : Algorithme mixte

Data : matrice d'adjacence du graphe $\boldsymbol{A}$, Nombre maximal de classes $K_{\max}$

Initialisation :
Soit *partitions* une liste de taille $K_{\max}$
Soit *scores* une liste de taille $K_{\max}$
for $k = 2$ **to** $K_{\max}$ **do**
  Initialiser $\mathcal{P}_0$ par le résultat du partitionnement spectral de $\boldsymbol{A}$ avec k classes.
  Soit $\mathcal{P}$ la partition trouvée par rééchantillonnage des classes et S le score associé.
  $partitions[k] \leftarrow \mathcal{P}$
  $scores[k] \leftarrow S$
end
$meilleure\_partition \leftarrow partitions[\arg\max_k scores]$
return *meilleure_partition*

L'algorithme de rééchantillonnage des classes est décrit dans l'algorithme 3. Cet algorithme parcourt tous les nœuds, propose de nouvelles partitions par déplacement des nœuds entre les classes, et renvoie la meilleure partition trouvée. Notons que la

modularité Q peut prendre des valeurs négatives. Pour éviter cela, on utilise une loi de probabilité de déplacement d'un nœud qui vaut $softmax((g_k)_{k=1,...K_{\max}})$.

Algorithme 3 : Algorithme de ré-échantillonnage des classes

Data : matrice d'adjacence du graphe $\boldsymbol{A}$, Partition initiale $\mathcal{P}_0$, Nombre d'itérations maximal N_{iter}

Initialisation :
Soit $K_{\max}$ le nombre de classes de $\mathcal{P}_0$
Soit $N_{\max}$ le nombre de nœuds du graphe, égal au nombre de lignes de $\boldsymbol{A}$
$\mathcal{P} \leftarrow \mathcal{P}_0$
$meilleure\_partition \leftarrow \mathcal{P}_0$
$meilleur\_score \leftarrow Q(\boldsymbol{A}, \mathcal{P})$
for $t = 1$ **to** N_{iter} **do**
  for $n = 1$ **to** $N_{\max}$ **do**
    Soit g un vecteur de taille $K_{\max}$
    for $k = 1$ **to** $K_{\max}$ **do**
      Créer $\mathcal{P}_k$ sur la base de $\mathcal{P}$ en déplaçant le nœud n dans la classe k.
      $g[k] \leftarrow Q(\boldsymbol{A}, \mathcal{P}_k)$
    end
    if $\max(poids) > meilleur\_score$ **then**
      $meilleur\_k \leftarrow \arg\max_k g$
      $meilleur\_score \leftarrow g[meilleur\_k]$
      $meilleure\_partition \leftarrow \mathcal{P}_{meilleur\_k}$
    end
    Tirer $k^* \sim softmax(g)$
    $\mathcal{P} \leftarrow \mathcal{P}_{k^*}$
  end
end
return $meilleure\_partition$, $meilleur\_score$

Des variantes des algorithmes 2 et 3 sont utilisées par la suite, mettant en œuvre une fonction objectif de vraisemblance à la place de la modularité Q. Dans ces conditions, la probabilité de choisir la vraisemblance du partitionnement, la probabilité de déplacer le nœud n sélectionné dans le cluster k vaut $\frac{g_k}{\sum_k^{K_{\max}} g_k}$.

En pratique, l'étape la plus lente est le calcul de la matrice d'adjacence $\boldsymbol{A}$. Cet algorithme se révèle assez lent. Le calcul des vecteurs propres est rapidement limité par la taille de la matrice, mais dans notre cas le calcul des matrices d'adjacence s'est révélé tout aussi lent et préférons fiabiliser les résultats.

5.2 Modèles de graphes aléatoires

Le choix de la fonction objectif est crucial. Plusieurs auteurs ont tenté de recourir à des modèles de graphes pour trouver la fonction objectif optimale. Dans nos expériences, les résultats obtenus en pratique avec ces méthodes se sont révélés souvent moins bons que ceux obtenus avec la modularité. Il y a deux applications aux modèles de graphes :

1. Connaissant les paramètres du modèle, générer des graphes.

2. Étant donné un graphe, estimer les paramètres du modèle générateur.

Pour le premier point, la connaissance des paramètres du modèle nous permet de calculer a posteriori la vraisemblance d'un graphe généré. La solution idéale sera obtenue en maximisant la vraisemblance, qui est la meilleure fonction objectif. Si l'algorithme de partitionnement basé sur cette fonction converge, la solution obtenue sera la meilleure possible et sert de référence. Les erreurs de partitionnement des autres algorithmes basés sur la modularité ou d'autres fonctions objectif sont comparées aux erreurs de l'algorithme référence pour mesurer une marge de progression. En effet, l'erreur d'un algorithme basé sur une autre fonction objectif sera supérieure à celle de l'algorithme du maximum de vraisemblance.

Ayant ainsi caractérisé dans un premier temps la performance d'un algorithme de partitionnement, il est possible d'estimer la qualité d'une solution. Cette fois, l'entrée du problème est un graphe et son partitionnement. Nous pouvons alors estimer les paramètres du modèle le plus vraisemblable. Ces paramètres sont directement liés à la physique du problème et peuvent contenir des informations utiles. Par exemple, dans notre cas précis, on s'attend à ce que les paramètres des nœuds du graphe soient corrélés au débit et à la variabilité des flux du réseau. La connaissance de ces paramètres permet aussi, grâce à l'étape 1, de re-générer de nombreux graphes simulés et partitionner ces graphes simulés. La variance de la solution de ce dernier sur un grand nombre de simulations du même modèle de graphe peut être considérée comme une borne. On mesure ainsi la pertinence d'une solution de façon générale. Finalement, ces résultats peuvent être utilisés pour estimer la fiabilité de la classification de chaque nœud individuellement par une mesure de sensibilité des paramètres du modèle.

La difficulté est donc de trouver un modèle permettant de produire des données réalistes, mais qui puisse être inversé afin de pouvoir effectuer cette seconde étape d'estimation de précision d'une solution.

5.2.1 Notations

Nous utilisons les notations suivantes pour définir les modèles de partitionnement. Le partitionnement d'un graphe consiste à répartir les N nœuds d'un graphe en K classes $C_1, ..., C_K$. Nous considérons des graphes non orientés et non pondérés associés à une matrice d'adjacence symétrique $\boldsymbol{A} = (a_{ij})$, avec $(i, j) \in \{1, ..., N\}^2$, où $a_{ij} \in \{0, 1\}$ indique si les nœuds i et j appartiennent à la même classe, i.e., $a_{ij} = 1$ si ces nœuds appartiennent à la même classe et $a_{ij} = 0$ sinon.

La solution du partitionnement est une matrice $\boldsymbol{Z}$ de dimensions (K, N) dont les valeurs sont 0 ou 1. Chaque indice z_{kn} indique l'appartenance du nœud n à la classe C_k et pour établir une classification stricte, on vérifie que $\sum_{k=0}^{K} z_{kn} = 1$. La matrice d'adjacence théorique du graphe nommée $\boldsymbol{V}$ vérifie $\boldsymbol{V} = \boldsymbol{Z}^T\boldsymbol{Z}$. Ainsi, deux nœuds (i, j) sont connectés et appartiennent à la même classe si $v_{ij} = 1$.

Le problème de partitionnement consiste à retrouver la matrice de partitionnement $\boldsymbol{V}$ ou de façon équivalente la matrice $\boldsymbol{Z}$. Nous disposons pour cela d'une observation bruitée qui est la matrice $\boldsymbol{A}$ obtenue à partir de tests de présence de file commune.

5.2.2 Modèles de graphes aléatoires de la littérature

Les articles de la littérature utilisant des modèles de graphes aléatoires utilisent principalement le Modèle Stochastique par Bloc, *Stochastic Block Model* (SBM) ou une version que nous appelons ici *Planted Partition Model*.

5.2.2.1 *Planted Partition Model*

Nous nommons *Planted Partition Model* une version simplifiée du SBM. Son partitionnement optimal est strict et sa vraisemblance peut être utilisée comme fonction objectif sans hypothèse sur le nombre de classes. Elle s'écrit de la façon suivante en définissant les paramètres du modèle $p = P(a_{ij} = 1 | v_{ij} = 1)$ et $q = P(a_{ij} = 1 | v_{ij} = 0)$:

$$L(\boldsymbol{V}; p, q, \boldsymbol{A}) = \prod_{i,j} p^{a_{ij}v_{ij}}(1-p)^{(1-a_{ij})v_{ij}} \times \prod_{i,j} q^{a_{ij}(1-v_{ij})}(1-q)^{(1-a_{ij})(1-v_{ij})}. \quad (5.16)$$

L'estimateur du maximum de vraisemblance de $\boldsymbol{V}$ (qui définit le partitionnement) s'obtient en maximisant (5.16). Maximiser la vraisemblance nécessite de connaître les probabilités p et q, comme dans [CDZ24]. Lorsque les valeurs diffèrent significativement des paramètres fixés par l'algorithme, les résultats sont mauvais et très inférieurs à ceux obtenus avec la modularité. Dans [CSX14], les valeurs de p et q sont également fixées et un terme pénalisant le nombre de classes est ajouté à la vraisemblance.

Un autre modèle plus fin qualifié d'*extended Planted Partition Model* a été étudié dans [CCT12]. Ce modèle ajoute un unique paramètre à chaque nœud qui joue sur le degré du nœud et rend plus importants certains nœuds. Ce paramètre unique implique toutefois de conserver un même rapport $\theta = \frac{p_n}{q_n}$ pour tous les nœuds.

5.2.2.2 *Stochastic Block Model*

Le SBM, introduit dans [HLL83], est décrit par une matrice $\boldsymbol{C}$ de taille $K \times K$ où K est le nombre de classes. Chaque élément c_{kl} définit la probabilité de présence d'une arête entre un nœud du groupe k et un autre du groupe l. Ce modèle a été largement utilisé dans la littérature de partitionnement comme en témoigne l'article [LW19]. Maximiser une fonction de vraisemblance de ce modèle permet de réaliser une classification assortative, c'est-à-dire qui rassemble dans une même classes les nœuds dortement reliés entre eux. À l'inverse, les équations de vraisemblance (du *Planted Partition Model* (5.16) ainsi que des autres modèles) laissent la possibilité de rassembler dans une même classe des nœuds peu reliés entre eux, en choisissant une valeur de p plus faible que q.

Le SBM permet donc de décrire des structures diverses, où une classe peut présenter des liens forts avec certaines classes mais pas d'autres. Dans ce modèle, le nombre de paramètres augmente avec le nombre de classes, sans ajout de contrainte supplémentaire. Donc plus il y a de classes, plus le nombre de paramètres augmente, plus le modèle se rapproche des données et la vraisemblance augmente. De ce point de vue, le meilleur modèle présentera toujours autant de classes que de nœuds. Tout un pan de la littérature tente de résoudre cette difficulté. Les approches sont nombreuses : par exemple le modèle du restaurant chinois limite le nombre de classes en fonction du nombre de nœuds. D'autres recherchent un coude dans l'accroissement de la vraisemblance en fonction du nombre de classes. Enfin, Peixoto introduit le *microcanonical* SBM [Pei17], qui minimise la probabilité d'obtenir le graphe observé pour des paramètres donnés.

Il existe une extension nommée Modèle Stochastique par Blocs avec correction du degré (DCSBM) qui ajoute un paramètre pour s'adapter au degré des nœuds [KN11], comme vu précédemment avec le *Planted Partition Model*.

5.2.3 Notre contribution : le modèle par nœuds

Nous présentons ici le modèle par nœuds qui est la contribution majeure de ce chapitre. Après avoir décrit les paramètres du modèle, nous présentons une méthode simple pour

estimer les paramètres de ce modèle à partir d'un graphe aléatoire, en supposant acquise la connaissance des clusters.

5.2.3.1 Description du modèle par nœuds

Le *Planted Partition Model* a l'avantage de modéliser systématiquement un partitionnement strict mais ne convient pas aux données réseaux. En effet, les probabilités de détection de goulots et de fausse alarme dépendent de paramètres propres aux nœuds (débit, variabilité des temps d'inter-arrivée des flux de paquets, ...). Aussi, nous proposons un modèle probabiliste plus général, appelé « Modèle par Nœuds » dans lequel chaque nœud n du graphe est associé à des attributs p_n, q_n tels que $1 \geq p_n > q_n \geq 0, \forall n$. On définit la probabilité de présence d'une arête entre deux nœuds (i, j) par une moyenne de leurs attributs :

$$\begin{aligned} P(a_{ij} = 1 | v_{ij} = 1) &= \tfrac{p_i + p_j}{2} \triangleq p_{ij}, \\ P(a_{ij} = 1 | v_{ij} = 0) &= \tfrac{q_i + q_j}{2} \triangleq q_{ij}. \end{aligned}$$

Ce modèle contient plus de paramètres que les modèles précédents qui lui permettent de générer des matrices plus complexes et de ressembler plus finement aux matrices obtenues dans les réseaux, comme nous le verrons par la suite. La vraisemblance associée à ce modèle a la forme suivante :

$$\begin{aligned} L(\boldsymbol{V}; \boldsymbol{A}, \boldsymbol{p}, \boldsymbol{q}) = &\prod_{i,j} p_{ij}^{a_{ij} v_{ij}} (1 - p_{ij})^{(1 - a_{ij}) v_{ij}} \\ &\times \prod_{i,j} q_{ij}^{a_{ij}(1 - v_{ij})} (1 - q_{ij})^{(1 - a_{ij})(1 - v_{ij})} \end{aligned} \tag{5.17}$$

avec $\boldsymbol{p} = (p_1, ..., p_N)^T$ et $\boldsymbol{q} = (q_1, ..., q_N)^T$.

5.2.3.2 Estimation de paramètres

Simuler des réalisations d'un modèle de graphe est très simple. En revanche, l'étape inverse qui consiste à déterminer les paramètres en connaissant les classes de données peut amener à résoudre un système d'équations très complexes. Cette seconde étape est utile pour une analyse de sensibilité des résultats. Après avoir déterminé les paramètres associés à l'expérience, en supposant que celle-ci correspond bien au modèle de graphe utilisé, nous pourrons revenir sur le résultat de partitionnement trouvé et estimer leur pertinence.

La définition de la probabilité de présence d'une arête par une somme plutôt qu'un produit, simplifie considérablement le problème d'estimation des paramètres du mo-

dèle. On doit choisir les vecteurs de coefficients $\boldsymbol{p}$ et $\boldsymbol{q}$ tels que $0 \leq \boldsymbol{p} \leq 1$ et $0 \leq \boldsymbol{q} \leq 1$

$$a_{ij} \approx v_{ij}\frac{p_i + p_j}{2} + (1 - v_{ij})\frac{q_i + q_j}{2}. \tag{5.18}$$

Pour minimiser l'erreur $\sum_{i,j} e_{ij}^2$ avec

$$e_{ij} = a_{ij} - v_{ij}\frac{p_i + p_j}{2} - (1 - v_{ij})\frac{q_i + q_j}{2}. \tag{5.19}$$

On aboutit aux systèmes d'équations suivants, définis pour tout i :

$$\mathrm{Card}(C_k)p_i + \sum_{j \in C_k} p_j = 2 \sum_{j \in C_k} a_{ij} \tag{5.20}$$

$$\mathrm{Card}(\bar{C}_k)q_i + \sum_{j \in \bar{C}_k} q_j = 2 \sum_{j \in \bar{C}_k} a_{ij}. \tag{5.21}$$

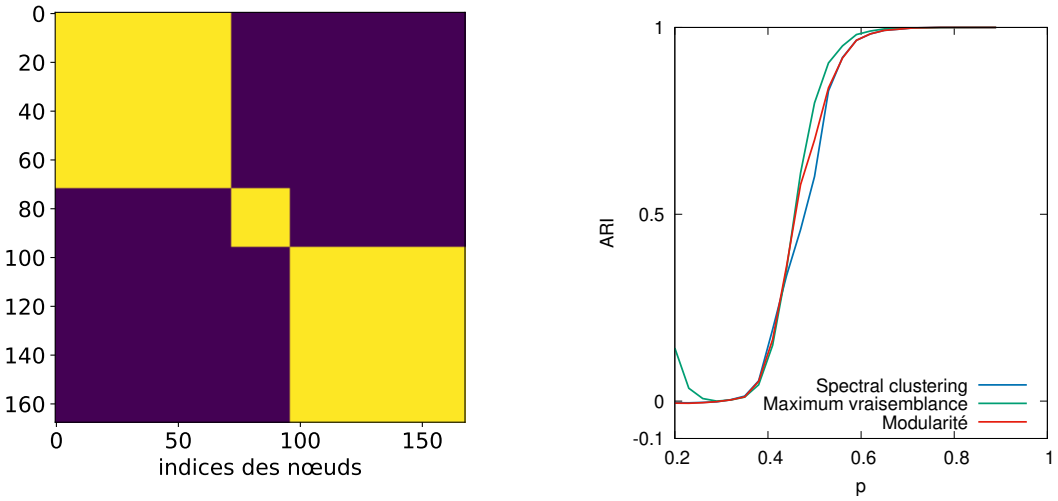
En pratique, la solution n'est pas exacte et nous minimisons l'erreur quadratique de ces équations avec les contraintes $0 \leq q_i, p_i \leq 1, \forall i$. Ce problème peut être mis sous une forme linéaire et est résolu facilement par des bibliothèques spécialisées, par exemple cvxpy en python.

5.3 Évaluation de l'algorithme

Cette section étudie la performance de l'algorithme 2 maximisant la modularité. Nous comparons ses résultats avec ceux obtenus par l'algorithme 1 de partitionnement spectral (sachant le nombre de classes) et la vraisemblance lorsque les paramètres associés au partitionnement sont connus.

5.3.1 Données synthétiques

Les algorithmes sont d'abord testés avec le Planted Partition Model. Il existe une symétrie dans la formule qui se retrouve dans les résultats. Nous avons choisi de simuler des cas avec $q = 0.3$ et p variable, sachant que les conclusions sont généralisables. Pour chaque valeur de p, 60 matrices différentes ont été simulées sur la base de la matrice théorique 5.2a. La figure 5.2b montre que les solutions obtenues par partitionnement spectral, modularité et maximum de vraisemblance (calculé avec rééchantillonnage des classes, en remplaçant la modularité par la vraisemblance dans l'algorithme 3) s'améliorent lorsque p augmente. Toutes les méthodes donnent des résultats similaires. Cela confirme que la modularité est une bonne approximation de la vraisemblance pour ce Planted Model.



(a) Vraie matrice d'adjacence.

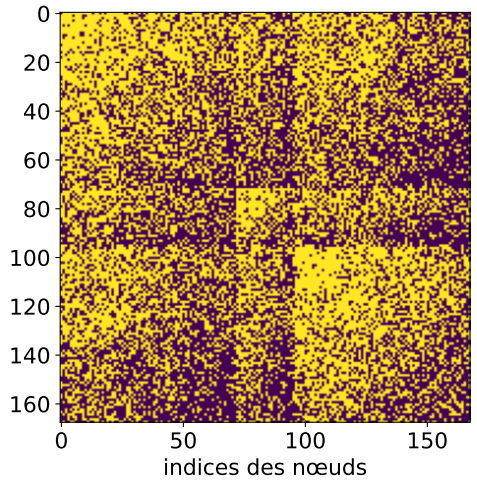
(b) ARIs moyens.

Figure 5.2 : Matrice d'adjacence et ARI pour le Planted Model.

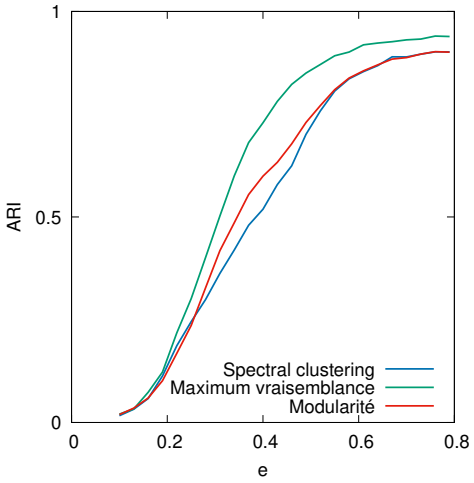
La figure 5.2b montre aussi que lorsque e diminue, donc quand $p < q$, la probabilité de lien intra-classe devient inférieure à celle d'un lien inter-classe. La méthode par modularité, qui fait justement l'hypothèse inverse, reste équivalente à un partitionnement aléatoire. La symétrie du problème se retrouve dans la symétrie de la vraisemblance qui permet d'obtenir de meilleurs résultats à mesure que $|p - q|$ augmente.

Les expériences suivantes s'intéressent au Modèle par Nœuds. Les paramètres du Modèle par Nœuds sont trop nombreux pour effectuer une analyse de performance exhaustive à l'aide de tous les vecteurs $\boldsymbol{p}$ et $\boldsymbol{q}$ du modèle. Nous choisissons donc d'évaluer les différentes méthodes de partitionnement sur des vecteurs $\boldsymbol{p}$ et $\boldsymbol{q}$ générés selon un paramètre $e = \sup_n\{p_n - q_n\}$, qui est le maximum des écarts entre les vecteurs $\boldsymbol{p}$ et $\boldsymbol{q}$. Ce paramètre contrôle la difficulté du partitionnement. Pour chaque valeur de e, 60 matrices d'adjacence sont simulées. Les résultats obtenus avec le Modèle par Nœuds sont représentés dans la figure 5.3b. La performance du partitionnement s'améliore lorsque e augmente (comme prévu avec la figure 5.2b), même si certains nœuds, dont l'écart $p_n - q_n$ est trop faible, sont mal classés.

La meilleure méthode est la vraisemblance, toujours supérieure aux deux autres. En revanche, la modularité est parfois meilleure que le partitionnement spectral. La connaissance du nombre de classes permet de faire un bon choix d'espace de projection. Néanmoins, les résultats montrent que parfois la modularité aboutit à une meilleure classification des flux selon le critère ARI malgré un nombre de classes différent. La connaissance a priori du nombre de classes est donc pratique pour obtenir rapidement



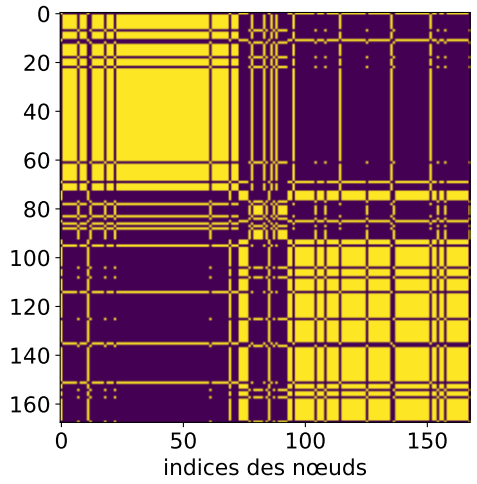
(a) Matrice d'adjacence simulée, $e = 0.4$.



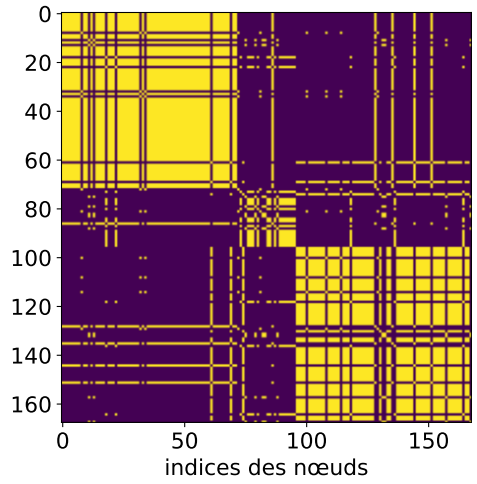
(b) ARIs moyens.

FIGURE 5.3 : Matrice et ARI pour le Modèle par Nœuds.

une bonne estimation de la partition par le nombre de classes, mais n'apporte pas d'information pour la résolution du problème car on peut atteindre un meilleur résultat en maximisant la modularité sans cette connaissance.



(a) Modularité & Spectral.



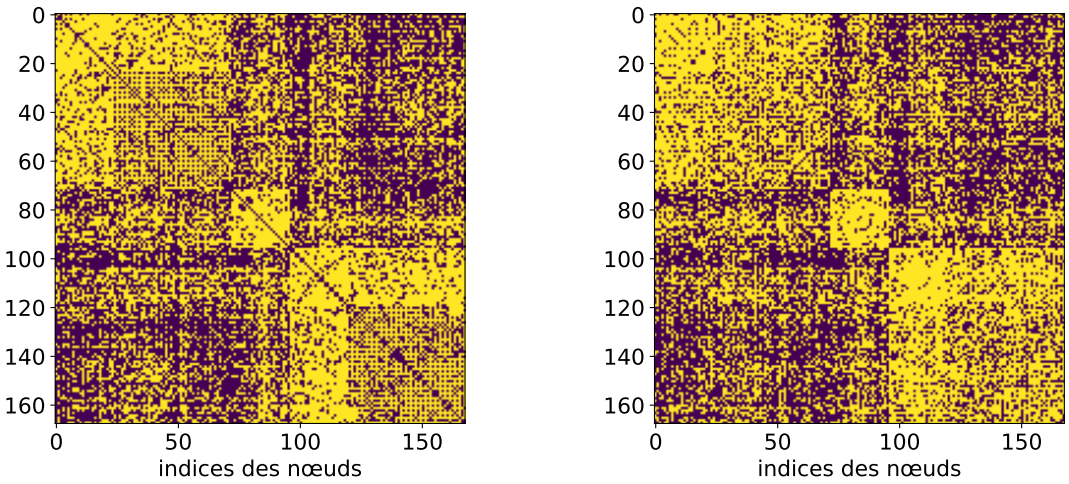
(b) Maximum de vraisemblance.

FIGURE 5.4 : Matrices solutions pour la matrice simulée 5.3a.

La figure 5.3a montre une matrice d'adjacence obtenue avec le Modèle par Nœuds avec $e = 0.4$. Les résultats de partitionnement sont présentés dans la figure 5.4. À première vue, les résultats obtenus avec le partitionnement spectral et avec la modu-

larité sont identiques et un peu moins bons que ceux obtenus avec la vraisemblance. Dans le détail, on se rend compte que la connaissance du nombre de classes utilisée par le partitionnement spectral aboutit à une moins bonne solution. La sélection des premiers vecteurs propres tronque l'information et la solution perd en qualité face à la celle obtenue par la modularité. Un mauvais choix de classes permet donc parfois de mieux classer les flots.

5.3.2 Données émulées



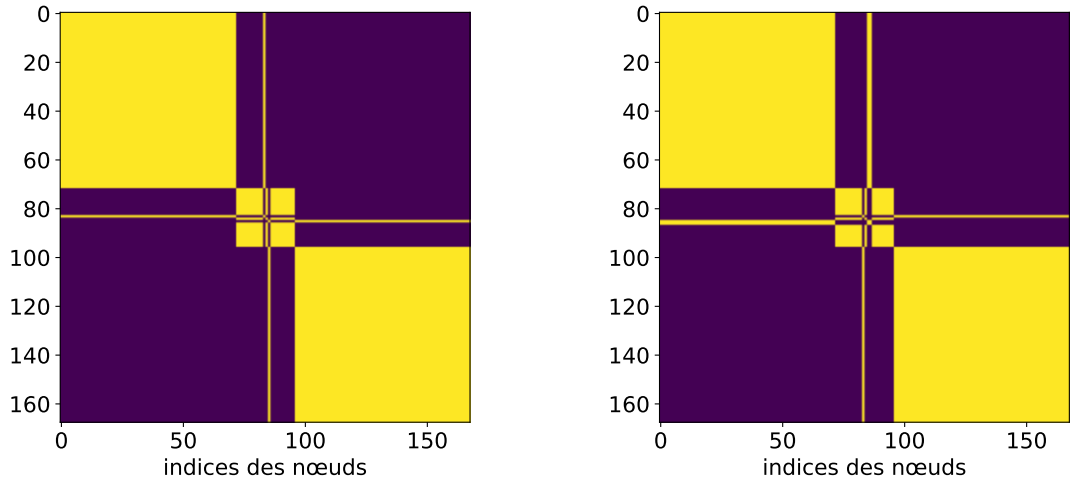
(a) Données émulées.

(b) générée avec le Modèle par Nœuds.

Figure 5.5 : Matrices d'adjacences.

Cette partie considère des données issues d'une expérience effectuée dans un réseau émulé. Nous devons trouver 3 goulots, i.e., 3 classes de flux qui ont traversé des files d'attentes différentes. D'autres files d'attente en amont régulent le trafic et forment des sous-classes diagonales moins connectées. La figure 5.5a illustre la matrice des tests de présence de goulots communs pour les données émulées. En comparaison, la figure 5.5b montre une matrice simulée à partir du Modèle par Nœuds avec des paramètres $\boldsymbol{p}$ et $\boldsymbol{q}$ optimisés pour ressembler à la figure 5.5a. Le modèle par nœud ne conserve pas les sous-classes visibles sur la diagonale, mais les matrices des deux figures 5.5a et 5.5b sont très similaires, confirmant l'intérêt du Modèle par Nœuds.

Les résultats obtenus par maximisation de la modularité et la méthode de partitionnement spectral sont très bons et identiques : 2 nœuds seulement sont mal classés dans les données émulées et 3 sur les données simulées. Les matrices d'adjacence ob-



(a) Données émulées.

(b) Modèle par Nœuds.

FIGURE 5.6 : Partitionnements des matrices d'adjacence.

tenues sont montrées sur les figures 5.6a et 5.6b. En comparaison, le maximum de vraisemblance commet une erreur de classification sur la matrice simulée.

5.4 Récapitulatif

Nous avons proposé un nouveau modèle probabiliste appelé « Modèle par Nœuds » pour le partitionnement de graphes. Grâce à ses nombreux paramètres, il modélise mieux les résultats de tests de présence de goulot que le *Planted Model*, tout en conservant sa simplicité de définition d'une partition optimale. Ce modèle permet d'étudier indépendamment la performance d'un algorithme de partitionnement et la présence d'un goulot d'étranglement. Un algorithme de partitionnement a été évalué sur ce modèle : les résultats obtenus sont bons sur des données émulées et proches de l'optimal dans le cas de données simulées.

Ces travaux pourront être étendus pour mieux établir la conformité du modèle aux données réelles. S'il s'avère que les paramètres du modèle peuvent être estimés autrement, il vaut mieux utiliser la fonction de vraisemblance et la fiabilité dépendra de la précision de l'estimation des paramètres du modèle. Au contraire, le partitionnement peut être l'étape nécessaire pour calculer ces paramètres et dans ce cas il serait intéressant de leur donner une signification physique.

Chapitre 6

Conclusion

6.1 Méthode

Cette thèse initialement orientée vers l'optimisation du réseau et la détection d'anomalies s'arrête finalement sur un outil de description du fonctionnement d'un réseau internet. Dans un contexte très théorique, sans données réelles, nous avons eu recours à la simulation pour modéliser des réseaux. Nous avons rapidement éprouvé des difficultés pour détecter des anomalies lorsque les paramètres du réseau variaient. La forte dépendance des observations aux conditions expérimentales nous a amené à nous poser des questions théoriques : est-il possible de retrouver les paramètres du réseau à partir d'une simple observation de trafic ? Pour répondre à cette question, nous sommes repartis de la définition d'une file d'attente, brique de base des réseaux. Cette définition est toute simple : la file d'attente limite le débit du flux sortant.

De cette observation découle un test, paramétré par une valeur d, indiquant l'absence d'une file d'attente de débit inférieur à d. Une analyse statistique de l'agrégat nous permet d'inverser le test et conclure quant à la présence probable d'une file d'attente de débit inférieur à d. En itérant sur différentes valeurs d choisies intelligemment en indiquant une file supérieure, on parvient à subdiviser l'agrégat des flux en fonction des files d'attentes traversées et ainsi retrouver l'architecture du réseau.

Cette méthode est en elle-même une forme de détection d'anomalie, qui consiste à définir un comportement normal et indiquer lorsqu'un comportement différent apparaît. Le comportement normal est ici une loi aléatoire d'arrivée des paquets, tandis qu'une série de paquets ayant traversé une file d'attente est ordonnée. Cette différence est quantifiée en mesurant une différence de fonction de répartition des temps inter-arrivées.

6.2 Résultats

Le résultat de cette thèse est une méthode (encore largement perfectible) permettant de connaître la topologie d'un réseau inconnu. Ainsi, les principaux partenaires du réseau (Google, un IX) pourront vraisemblablement connaître la topologie et obtenir un bon aperçu de l'état du réseau au prix d'un bon équipement de capture du trafic et de quelques calculs. Ces mêmes opérateurs partenaires pourront donc identifier des anomalies dans le réseau de l'opérateur, et réciproquement.

Nous conjecturons qu'un nœud est observable depuis un point d'observation du réseau à deux conditions :

- Qu'un agrégat de paquets issus du nœud à observer atteigne le point d'observation.
- Que tous les nœuds traversés par l'agrégat soient de débit supérieur à celui du nœud à observer.

La limite de compréhension du réseau fixe aussi l'horizon de détection des erreurs : là où l'action des files d'attente n'est plus discernable, il ne sera plus possible de détecter des erreurs sans connaissance supplémentaire.

L'impossibilité de remonter à la connaissance de la topologie du réseau aurait prouvé l'impossibilité de détecter les files d'attente, de comprendre le réseau, et finalement de détecter des anomalies dans du trafic chiffré. À l'inverse, en montrant que cette connaissance est possible, nous montrons que la détection d'anomalies dans un réseau sera possible de façon passive sur du trafic chiffré et sans connaissance a priori de la méthode d'émission du trafic ni de la topologie.

6.3 Perspectives

Les perspectives de cette thèse se déclinent en 4 pistes.

La première piste est de terminer la mise au point de l'algorithme d'inférence de topologie que nous avons esquissé. La principale question est celle de la fiabilité des solutions et du critère d'arrêt : à quel moment faut-il arrêter de remonter dans la topologie ? Dit autrement, jusqu'où pouvons-nous obtenir un partitionnement des flux qui soit suffisamment fiable ? Une suggestion a été faite dans le dernier chapitre de la thèse, mais elle n'a pas été implémentée ni testée. Les différents points de temps de calcul et validation du modèle de graphe sur des données réelles seront à vérifier soigneusement. D'autre part, la qualité des solutions pourrait être améliorée avec une

meilleure fonction objectif. Les résultats fournis par la modularité sont très bons, mais il existe encore une marge de progression avec la modularité généralisée. Cette fonction est cruciale car elle est aussi à la base du critère d'arrêt. Une bonne alternative de revenir à l'étude des agrégats et améliorer l'optimisation par un critère de pertinence basé directement sur les inter-arrivées, et non sur le graphe.

L'axe de travail suivant concerne l'exploitation de l'algorithme. L'estimation du débit d'une file d'attente peut servir à estimer le taux de saturation de chaque file en prenant en compte les paquets non observés. Cette estimation passera encore par l'analyse de l'agrégat qui en est issu, en interprétant les sauts de la fonction de répartition. Dans cette fonction de répartition, l'abscisse de chaque saut multiple du temps de transmission d'un paquet de la file indique des séquences de paquets saturées non observées entièrement. Il serait donc possible d'estimer la proportion de trafic observé à partir des mesures de différence d'ordonnée sur les sauts de la fonction de répartition.

Le troisième axe est celui de la détection d'anomalies. Dans un contexte de flux chiffrés, sans numérotation des paquets ni prévisibilité du comportement individuel d'un flux, la seule façon de détecter une anomalie sera de mesurer le débit du flux et d'observer une anomalie temporelle (par exemple variation anormale du débit du flux) ou spatiale (par comparaison avec les agrégats issus d'autres nœuds voisins).

Le dernier axe est l'optimisation globale du réseau. Nous voyons dans ces travaux que les goulots d'étranglement sont susceptibles de se déplacer dans le réseau. La gestion des files d'attente n'est donc pas une simple affaire locale, mais devrait être pensée de façon globale pour harmoniser la gestion du réseau. Maîtriser la congestion consistera à savoir et choisir l'endroit où celle-ci apparaît. Les points de congestion dépendront donc des équipements du réseau et de la topologie du réseau.

Enfin, ces choix auront un impact fort sur l'observabilité du réseau et donc les poins où l'algorithme proposé dans cette thèse pourra être déployé avec succès. Le choix d'un protocole de routage multichemin comme TCP *multipath* remet aussi en cause tout le travail effectué. L'inférence de topologie est impossible si les paquets d'un même flux peuvent être issus de chemins différents.

Glossaire

ACK *Acknowledgement*, numéro d'acquittement échangé dans le protocole TCP pour confirmer la réception des données. Ce mécanisme permet de garantir la fiabilité des communications en assurant que les paquets sont bien reçus par le destinataire. 21, 24, 57, 87

ACP Analyse en Composantes Principales. 108, 110

ADL Analyse Discriminante Linéaire. 110

AIMD augmentation additive/retrait multiplicatif (*Additive Increase Multiplicative Decrease*). 21

AQM *Active Queue Management*, gestion active de file d'attente. Les algorithmes AQM, comme RED, sont utilisés pour améliorer les performances des réseaux en évitant la congestion. 19–21, 49

ARI *Adjusted Rand Index*, indice de Rand ajusté. 110, 113, 122

CBR *Constant Bit Rate*, se dit d'un flot périodique où les paquets sont envoyés à intervalles réguliers. CBR est souvent utilisé pour décrire des flux multimédias ou des applications temps réel. 44

CSMA *Carrier Sense Multiple Access*, protocole de la couche liaison de données utilisé pour gérer l'accès au médium partagé dans les réseaux locaux. CSMA est souvent utilisé dans les réseaux Ethernet et Wi-Fi. 17

DCSBM *Degree Corrected Stochastic Block Model*, Modèle Stochastique par Blocs avec correction du degré des nœuds. 118

DNS *Domain Name System*, système permettant de traduction des noms de domaine en adresses IP pour faciliter la navigation sur Internet. 23

DoH *DNS over HTTPS*, protocole sécurisé encapsulant les requêtes DNS dans le protocole HTTPS pour améliorer la confidentialité et la sécurité des échanges. 23

DPI *Deep Packet Inspection*, inspection de paquet en profondeur : protocole et données. 24

FIFO *First In First Out* - premier entré, premier sorti. 38, 52

GEO En orbite géostationnaire, située à environ 35 786 km d'altitude au-dessus de l'équateur. Les satellites géostationnaires restent fixes par rapport à un point sur Terre et servent beaucoup pour la télévision et pour internet jusque récemment. 7–9, 13

GMM *Gaussian Mixture Model*, modèle de mélange de lois gaussiennes. 107

HTTP/2 *HyperText Transfer Protocol version 2*, protocole de communication utilisé pour transférer des données sur le Web, version sécurisée et améliorée de HTTP. 23

ICMP *Internet Control Message Protocol*, protocole couche réseau de test de connexion. 25, 88

K-means K-moyennes. 106, 107, 109

LEO *Low Earth Orbit*, en orbite basse. Les orbites basses (inférieure à 2 000 km d'altitude) sont notamment utilisées par les stations spatiales, l'observation de la Terre et désormais pour les télécommunications (téléphonie, IoT et internet). 7–10, 12, 13, 48, 51, 69

LLM *Large Language Model*, Grand Modèle de Langage, architecture de réseau de neurones profond utilisée pour comprendre et générer du langage naturel. Les LLM sont entraînés sur de grandes quantités de texte et sont capables de réaliser des tâches complexes comme la génération de texte ou la réponse à des questions. 24

LSTM *Long Short Term Memory*, architecture de réseau de neurones récursif conçue pour éviter le problème de la disparition du gradient. Les LSTM sont souvent utilisés pour des tâches nécessitant une mémoire à long terme, comme la reconnaissance vocale ou la traduction automatique. 24

MEO *Medium Earth Orbit*, en orbite à moyenne altitude. Les satellites en orbite moyenne (entre 2 000 et 35 786 km d'altitude) servent notamment dans le cadre de systèmes de navigation comme le GPS, Galileo ou BeiDou. 12, 48

MTU *Maximum Transmission Unit*, unité maximale de transmission. 88

NS3 *Network Simulateur 3*, simulateur de réseau IP à événements discrets. Il est largement construit à partir de la pile protocole linux et implémente le concept de nœuds mobiles dans un réseau. Voir le site ns3 ou le papier associé [np]. 46, 52, 56, 57

NTMA *Network Traffic Monitoring and Analysis*, ensemble des techniques d'identification des applications et d'analyse du trafic des utilisateurs. 24

PASTA *Poisson Arrivals See Time Averages* - la moyenne des mesures à des instants tirés selon un processus de Poisson convergent vers une moyenne temporelle. 40

qdisc *queuing discipline*, discipline de file d'attente. 52, 68

QoE *Quality of Experience*, qualité d'expérience. La QoE est une mesure subjective de la satisfaction de l'utilisateur final pour une application ou un service, souvent influencée par des facteurs comme la latence, la qualité vidéo, ou la fiabilité. 30

QoS *Quality of Service*, qualité de service. La QoS est un ensemble de technologies et de mécanismes utilisés pour garantir un niveau de performance spécifique pour les applications critiques dans un réseau. 30, 61

QUIC *Quick UDP Internet Connection*, protocole de transport fiabilisé devenu un standard dans le RFC 9000. QUIC combine les avantages de TCP et UDP, offrant une connexion rapide et sécurisée, souvent utilisée pour le protocole HTTP/3. 15, 23, 24

RED *Random Early Detection*, algorithme d'AQM (Active Queue Management) conçu pour éviter la congestion du réseau en rejetant aléatoirement des paquets avant que la file d'attente ne soit pleine. 20

RTT *Round Trip Time*, mesure du temps aller-retour d'un paquet entre l'émetteur et le récepteur. Le RTT est un indicateur clé de la latence dans un réseau et est souvent utilisé pour évaluer les performances des connexions. 20, 21, 23–26, 52, 63

SBM *Stochastic Block Model*, Modèle Stochastique par Bloc. 117, 118

tc *traffic control*, outil en ligne de commande de manipulation des files d'attente présent dans les systèmes linux [Hem25]. 35, 61, 63, 67

TCP *Transmission Control Protocol*, protocole de transport normalisé par le RFC 793. TCP est un protocole orienté connexion qui assure la fiabilité, le contrôle de flux, et la gestion des erreurs dans les communications réseau. 9, 20, 21, 23, 24, 49, 50, 52, 53, 57, 59, 60, 62, 63, 65, 102

TLE *Two Lines Element*. 46, 47

ToS *Type of Service*, champ présent dans le protocole IP pour prioriser certaines communications. Le champ ToS permet de spécifier des niveaux de priorité pour différents types de trafic réseau. 23

UDP *User Datagram Protocol*, protocole de transport non fiable documenté par le RFC 768. UDP est souvent utilisé pour les applications où la rapidité est plus importante que la fiabilité, comme la diffusion vidéo ou les jeux en ligne. 23, 50, 57, 60, 62, 63

Bibliographie

[ABH+22] Alexia Auddino, Anna Barraqué, Oana Hotescu, Jérôme Lacan, José Radzik, and Emmanuel Lochin. The nearest is not the fastest : On the importance of selecting in/out routing hops over a satellite leo constellation. In 2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall), pages 1–5. IEEE, 2022.

[AHA+21] Mohamed A Alkelsh, Walid KA Hasan, Emhamed A Adaba, Albahlool M Abood, and Johnson Agbinya. Design & simulation of voice qos performance in data network congestion for m/d/1 queuing model. Albahit journal of applied sciences, 2(1) :37–45, 2021.

[AKM04] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing router buffers. ACM SIGCOMM Computer Communication Review, 34(4) :281–292, 2004.

[AST21] Mahmoud Abbasi, Amin Shahraki, and Amir Taherkordi. Deep learning for network traffic monitoring and analysis (ntma) : A survey. Computer Communications, 2021.

[AV06] David Arthur and Sergei Vassilvitskii. k-means++ : The advantages of careful seeding. Technical report, Stanford, 2006.

[BBG06] Olivier Brun, Charles Bockstal, and Jean-Marie Garcia. Analytic approximation of the jitter incurred by cbr traffics in ip networks. Telecommunication Systems, 33 :23–45, 2006.

[BCGH18] Thomas Bonald, Bertrand Charpentier, Alexis Galland, and Alexandre Hollocou. Hierarchical graph clustering using node pair sampling. arXiv preprint arXiv :1806.01664, 2018.

[BDG+08] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. IEEE Transactions on Knowledge and Data Engineering, 2008.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning, volume 4. Springer, 2006.

[CC96] Robert L Carter and Mark E Crovella. Measuring bottleneck link speed in packet-switched networks. Performance evaluation, 27 :297–318, 1996.

[CCG+17] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr : Congestion-based congestion control. Communications of the ACM, 60(2) :58–66, 2017.

[CCT12] Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. Journal Machine Learning Research, pages 35–1, 2012.

[CDZ24] Mihai Cucuringu, Xiaowen Dong, and Ning Zhang. Maximum likelihood estimation on stochastic blockmodels for directed graph clustering. arXiv preprint arXiv :2403.19516, 2024.

[CLMS17] Mauro Conti, QianQian Li, Alberto Maragno, and Riccardo Spolaor. The dark side(-channel) of mobile devices : A survey on network traffic analysis. arXiv : Cryptography and Security, 2017.

[CMK00] Kenjiro Cho, Koushirou Mitsuya, and Akira Kato. Traffic data repository at the wide project. In 2000 USENIX Annual Technical Conference (USENIX ATC 00), 2000.

[CMPS02] J Case, R Mundy, D Partain, and B Stewart. Rfc3410 : Introduction and applicability statements for internet-standard management framework, 2002.

[CSB25] Neal Cardwell, Ian Swett, and Joseph Beshay. BBR Congestion Control. Internet-Draft draft-ietf-ccwg-bbr-04, Internet Engineering Task Force, October 2025. Work in Progress.

[CSX14] Yudong Chen, Sujay Sanghavi, and Huan Xu. Improved graph clustering. IEEE Trans. Inf. Theory, 60(10) :6440–6455, 2014.

[DCM18] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. Simulated annealing : From basics to applications. In Handbook of metaheuristics, pages 1–35. Springer, 2018.

[DF25] Martin Duke and Gorry Fairhurst. Specifying New Congestion Control Algorithms. RFC 9743, March 2025.

[DGS12] Hamza Dahmouni, André Girard, and Brunilde Sansò. An analytical model for jitter in ip networks. annals of telecommunications-annales des télécommunications, 67 :81–90, 2012.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society : series B (methodological), 39(1) :1–22, 1977.

[DRM01] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore. What do packet dispersion techniques measure ? In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213), volume 2, pages 905–914. IEEE, 2001.

[FJ93] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on networking, 1(4) :397–413, 1993.

[FOE22] Mah-Rukh Fida, Andres F. Ocampo, and Ahmed Elmokashfi. Measuring and localising congestion in mobile broadband networks. IEEE Transactions on Network and Service Management, 19(1) :366–380, 2022.

[For10] Santo Fortunato. Community detection in graphs. Physics reports, 486(3-5) :75–174, 2010.

[FP01] Sally Floyd and Vern Paxson. Difficulties in simulating the internet. IEEE/ACM Transactions on networking, 9(4) :392–403, 2001.

[GA17] Alexander J Gates and Yong-Yeol Ahn. The impact of random models on clustering similarity. Journal of Machine Learning Research, 18(87) :1–28, 2017.

[GBG02] Jean-Marie Garcia, Olivier Brun, and David Gauchard. Transient analytical solution of m/d/1/n queues. Journal of applied probability, 39(4) :853–864, 2002.

[GdMC10] Benjamin H Good, Yves-Alexandre de Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. Physical Review E, 2010.

[GN12] Jim Gettys and Kathleen Nichols. Bufferbloat : dark buffers in the internet. Communications of the ACM, 55(1) :57–65, 2012.

[GP99] Erol Gelenbe and Guy Pujolle. Introduction to queueing networks. John wiley & sons, New York, (NY), 2e éd. edition, 1999.

[GPL+22] Paul Grislain, Nicolas Pelissier, François Lamothe, Oana Hotescu, Jérôme Lacan, Emmanuel Lochin, and José Radzik. Rethinking leo constellations routing with the unsplittable multi-commodity flows problem. In 2022 11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC), pages 1–8. IEEE, 2022.

[GSA+15] Mohadeseh Ganji, Abbas Seifi, Hosein Alizadeh, James Bailey, and Peter J. Stuckey. Generalized modularity for community detection. null, 2015.

[GSTH11] Donald Gross, John F Shortle, James M Thompson, and Carl M Harris. Fundamentals of queueing theory, volume 627. John wiley & sons, Hoboken, (NJ), 4e éd. edition, 2011.

[HBB00] Khaled Harfoush, Azer Bestavros, and John Byers. Robust identification of shared losses using end-to-end unicast probes. In Proceedings 2000 International Conference on Network Protocols, pages 22–33. IEEE, 2000.

[HCL+03] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification. 2003.

[Hem25] Stephen Hemminger. tc - show / manipulate traffic control settings, March 2025. Part of the iproute2 package (version 6.14.0), a suite of Linux networking utilities including 'ip', 'ss', and 'tc'. iproute2 provides advanced features for configuring and monitoring network behavior in the Linux kernel.

[HFW14] David A Hayes, Simone Ferlin, and Michael Welzl. Practical passive shared bottleneck detection using shape summary statistics. In 39th

Annual IEEE Conference on Local Computer Networks, pages 150–158. IEEE, 2014.

[HLL83] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels : First steps. Social networks, 5(2) :109–137, 1983.

[HLM+04] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating internet bottlenecks : Algorithms, measurements, and implications. ACM SIGCOMM Computer Communication Review, 34(4) :41–54, 2004.

[HPH+09] Xinming He, Christos Papadopoulos, John Heidemann, Urbashi Mitra, and Usman Riaz. Remote detection of bottleneck links using spectral and statistical methods. Computer Networks, 53(3) :279–298, 2009.

[HRX08] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic : a new tcp-friendly high-speed tcp variant. ACM SIGOPS operating systems review, 42(5) :64–74, 2008.

[HWF+20] David A Hayes, Michael Welzl, Simone Ferlin, David Ros, and Safiqul Islam. Online identification of groups of flows sharing a network bottleneck. IEEE/ACM Transactions on Networking, 28(5) :2229–2242, 2020.

[JD02] Hao Jiang and Constantinos Dovrolis. Passive estimation of tcp round-trip times. ACM SIGCOMM Computer Communication Review, 32(3) :75–88, 2002.

[KB01] Dina Katabi and Charles Blake. Inferring congestion sharing and path characteristics from packet interarrival times. Technical report, MIT-LCS-TR-828, MIT, 12 2001.

[KBÁ+20] Simon Kassing, Debopam Bhattacherjee, André Baptista Águas, Jens Eirik Saethre, and Ankit Singla. Exploring the “Internet from space” with Hypatia. In ACM IMC, 2020.

[KBY01] Dina Katabi, Issam Bazzi, and Xiaowei Yang. A passive approach for detecting shared bottlenecks. In Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No. 01EX495), pages 174–181. IEEE, 2001.

[KHR02] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications, pages 89–102, 2002.

[KKB+04] Sachin Katti, Dina Katabi, Charles Blake, Eddie Kohler, and Jacob Strauss. Multiq : Automated detection of multiple bottleneck capacities along a path. In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, pages 245–250, 2004.

[KKP22] Grigorios Kakkavas, Vasileios Karyotis, and Symeon Papavassiliou. Topology inference and link parameter estimation based on end-to-end measurements. Future Internet, 14(2) :45, 2022.

[KKS+08] Min Sik Kim, Taekhyun Kim, Yong-June Shin, Simon S Lam, and Edward J Powers. A wavelet-based approach to detect shared congestion. IEEE/ACM Transactions on Networking, 16(4) :763–776, 2008.

[KL70] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. The Bell system technical journal, 49(2) :291–307, 1970.

[KLM14] Nicolas Kuhn, Emmanuel Lochin, and Olivier Mehani. Revisiting old friends : is codel really achieving what red cannot ? In Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop, pages 3–8, 2014.

[KN11] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics, 83(1) :016107, 2011.

[KSW23] Ike Kunze, Constantin Sander, and Klaus Wehrle. Does it spin ? on the adoption and use of quic's spin bit. In Proceedings of the 2023 ACM on Internet Measurement Conference, pages 554–560, 2023.

[LB01] Kevin Lai and Mary Baker. Nettimer : A tool for measuring bottleneck link bandwidth. In 3rd USENIX Symposium on Internet Technologies and Systems (USITS 01), 2001.

[Lin06] Benjamin Lindner. Superposition of many independent spike trains is generally not a poisson process. Physical Review E—Statistical, Nonlinear, and Soft Matter Physics, 73(2) :022901, 2006.

[LMCSEL17] Manuel López-Martín, Belén Carro, Antonio Sánchez-Esguevillas, and Jaime Lloret. Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access, 2017.

[LW19] Clement Lee and Darren J Wilkinson. A review of stochastic block models and extensions for graph clustering. Applied Network Science, 4(1) :1–50, 2019.

[MBD04] Mohammad Malli, Chadi Barakat, and Walid Dabbous. A Survey on Internet Topology Inference. Research Report RR-5439, INRIA, 2004.

[MHL+14] Liang Ma, Ting He, Kin K Leung, Ananthram Swami, and Don Towsley. Inferring link metrics from end-to-end path measurements : Identifiability and monitor placement. IEEE/ACM transactions on networking, 22(4) :1351–1368, 2014.

[Moy98] John Moy. OSPF Version 2. RFC 2328, April 1998.

[MSH15] David Mehrle, Amy Strosser, and Anthony Harkin. Walk-modularity and community structure in networks. Network Science, 2015.

[MTGB22] François Michel, Martino Trevisan, Danilo Giordano, and Olivier Bonaventure. A first look at starlink performance. In Proceedings of the 22nd ACM Internet Measurement Conference, pages 130–136, 2022.

[MZS20] Liang Ma, Ziyao Zhang, and Mudhakar Srivatsa. Neural network tomography. arXiv preprint arXiv :2001.02942, 2020.

[NA09] Thuy TT Nguyen and Grenville Armitage. A survey of techniques for internet traffic classification using machine learning. IEEE communications surveys & tutorials, 10(4) :56–76, 2009.

[New06a] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. Physical Review E, 2006.

[New06b] Mark EJ Newman. Modularity and community structure in networks. Proc. of the national academy of sciences, 103(23) :8577–8582, 2006.

[New16] Mark EJ Newman. Equivalence between modularity optimization and maximum likelihood methods for community detection. Physical Review E, 94(5) :052315, 2016.

[np] The ns 3 project. the ns-3 network simulator. Accessed : 2023-10-08.

[OL06] Philippe Owezarski and Nicolas Larrieu. Techniques et outils de métrologie pour l'Internet et son trafic. Editions TI, 2006.

[Pax97] Vern Edward Paxson. Measurements and analysis of end-to-end Internet dynamics. University of California, Berkeley, 1997.

[Pei17] Tiago P. Peixoto. Nonparametric bayesian inference of the microcanonical stochastic block model. Physical Review E, 2017.

[PPRL24] Matthieu Petrou, David Pradas, Mickaël Royer, and Emmanuel Lochin. Unveiling youtube qoe over satcom using deep-learning. IEEE Access, 2024.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. Journal of Machine Learning Research, 12 :2825–2830, 2011.

[RKT00] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. In Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pages 145–155, 2000.

[RKT02] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting shared congestion of flows via end-to-end measurement. IEEE/ACM Transactions On Networking, 10(3) :381–395, 2002.

[Rou87] Peter J Rousseeuw. Silhouettes : a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20 :53–65, 1987.

[SJSB22] Kevin D Smith, Saber Jafarpour, Ananthram Swami, and Francesco Bullo. Topology inference with multivariate cumulants : the möbius inference algorithm. IEEE/ACM Transactions on networking, 30(5) :2102–2116, 2022.

[SKK03] Jacob Strauss, Dina Katabi, and Frans Kaashoek. A measurement study of available bandwidth estimation tools. In Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, pages 39–44, 2003.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 22(8) :888–905, 2000.

[TFA+24] Ziad Tlaiss, Alexandre Ferrieux, Isabel Amigo, Isabelle Hamchaoui, and Sandrine Vaton. Automated slow-start detection for anomaly root cause analysis and bbr identification. Annals of Telecommunications, 79(3) :149–163, 2024.

[TK19] Brian Trammell and Mirja Kühlewind. The Wire Image of a Network Protocol. RFC 8546, apr 2019.

[Var06] Pal Varga. Analyzing packet interarrival times distribution to detect network bottlenecks. In EUNICE 2005 : Networks and Applications Towards a Ubiquitously Connected World : IFIP International Workshop on Networked Applications, Colmenarejo, Madrid/Spain, 6-8 July, 2005, volume 196, pages 17–29. Springer, 06 2006.

[VKF+03] Pál Varga, Gergely Kún, Péter Fodor, József Bíró, Daisuke Satoh, and Keisuke Ishibashi. An advanced technique on bottleneck detection. In IFIP WG6. 3 workshop, EUNICE 2003, 2003.

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. Statistics and computing, 17 :395–416, 2007.

[VLL05] Bryan Veal, Kang Li, and David Lowenthal. New methods for passive estimation of tcp round-trip times. In Passive and Active Network Measurement : 6th International Workshop, PAM 2005, Boston, MA, USA, March 31-April 1, 2005. Proceedings 6, pages 121–134. Springer, 2005.

[WS05] Scott White and Padhraic Smyth. A spectral clustering approach to finding communities in graph. SDM, 2005.

[XYH+19] Junfeng Xie, F. Richard Yu, Tao Huang, Renchao Xie, Jiang Liu, Chenmeng Wang, and Yunjie Liu. A survey of machine learning techniques applied to software defined networking (sdn) : Research issues and challenges. IEEE Communications Surveys and Tutorials, 2019.

[YMH+18] Francis Y Yan, Jestin Ma, Greg D Hill, Deepti Raghavan, Riad S Wahby, Philip Levis, and Keith Winstein. Pantheon : the training ground for

internet congestion-control research. In 2018 USENIX Annual Technical Conference (USENIX ATC 18), pages 731–743, 2018.

[YWY08] Muhammad Murtaza Yousaf, Michael Welzl, and Bülent Yener. Accurate shared bottleneck detection based on svd and outlier detection. University of Innsbruck, Institute of Computer Science, Tech. Rep. DPS NSG Technical Report, 1, 2008.

[ZPH19] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking : A survey. IEEE Communications Surveys and Tutorials, 2019.

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

Université de Toulouse

Titre : Apport de l'IA pour la caractérisation réseau et la gestion de la ressource SATCOM

Mots clés : communication satellite, congestion, QUIC, partitionnement

Résumé : Les mégaconstellations améliorent considérablement l'accès à Internet à l'échelle mondiale et bouleversent le secteur des télécommunications. Elles partagent avec les réseaux terrestres plusieurs défis, notamment celui de la congestion. Toutefois, la gestion d'un tel réseau est plus complexe en raison de sa taille et de la mobilité des satellites : leur déplacement relatif par rapport aux clients et aux stations au sol impose un renouvellement fréquent des connexions. Ce contexte particulier souligne l'importance d'outils de suivi de l'état du réseau. Cette thèse commence par aborder la modélisation d'un tel réseau, et se concentre sur l'identification des goulots d'étranglement. Les files d'attente de ces nœuds limitant le trafic sont susceptibles d'accumuler les paquets. Un réglage inadapté de ces files peut alors provoquer de la congestion et dégrader la qualité de service offerte aux utilisateurs. Nous présentons ensuite un algorithme passif d'inférence de topologie, qui constitue la contribution majeure de cette thèse. Les utilisateurs émettent dans le réseau des flux de paquets, identifiables par un quintuplet commun (adresses IP, ports, protocole). La méthode proposée consiste à analyser les intervalles d'arrivée des paquets appartenant à des ensembles de flux afin de caractériser les nœuds communs. L'enjeu principal réside dans le partitionnement correct des flux, permettant ainsi de mettre en évidence les différentes files d'attente du système. Ce point est développé plus en détail dans la dernière partie. Cette analyse des intervalles d'arrivée permet d'estimer la saturation des files d'attente et, in fine, d'identifier les goulots du réseau. L'avantage de la méthode proposée réside dans sa simplicité de déploiement : un unique point de capture dans le réseau suffit pour connaître l'état des files d'attente. Notons toutefois que cette étude se limite aux flux montants vers le point de capture. Cette approche se révèle particulièrement adaptée au contexte spatial, notamment à proximité d'une station au sol. Les résultats, obtenus sur un réseau émulé, montrent la possibilité de détecter les goulots dans une architecture centralisée.

Title: AI contribution to network characterization and SATCOM resource management

Key words: satellite communication, congestion, QUIC, clustering

Abstract: Megaconstellations significantly improve global Internet access and are transforming the telecommunications sector. They share several challenges with terrestrial networks, particularly congestion. However, managing such a network is more complex due to its scale and the mobility of satellites: their relative movement with respect to ground users and stations requires frequent handovers. This variable context underscores the importance of network monitoring tools. This thesis first addresses the modeling of such a network, focusing on the identification of bottlenecks. The queues at these nodes, which limit traffic, are prone to packet accumulation. Inadequate queue management can lead to congestion and degrade the quality of service provided to users. We then present a passive topology inference algorithm, which is the major contribution of this thesis. Users transmit packet flows in the network, identifiable by a common five-tuple (IP addresses, ports, protocol). The proposed method involves analyzing the inter-arrival times of packets belonging to sets of flows to characterize common nodes. The main challenge lies in correctly partitioning the flows, thereby revealing the various queues in the system. This aspect is explored in greater detail in the final section. This analysis of inter-arrival times enables the estimation of queue saturation and leads, ultimately, to identify the network bottlenecks. The main advantage of the proposed method is its deployment simplicity: a single capture point in the network is sufficient to determine the state of the queues in its neighborhood. It should be noted, however, that this study is limited to upstream flows toward the capture point. This approach is particularly well-suited to the spatial context, especially near a ground station. The results, obtained on an emulated network, demonstrate the ability to detect bottlenecks in centralized architectures.