ENSC–2013 N 495

THÈSE DE DOCTORAT DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

École Doctorale MATISSE

présentée pour obtenir le grade de

Docteur de l'École Normale Supérieure de Cachan

par

Aurore Junier

Analyses de performance et de stabilité des réseaux de télécommunication

---

Thèse présentée et soutenue au centre de recherche INRIA de Rennes le **16 décembre 2013** devant la commission d'examen composée de :

Monsieur Stefan Haar ..... *Président du jury*
Directeur de recherche, INRIA centre de Saclay, Île-de-France

Monsieur Nicolas Navet ..... *Rapporteur*
Associate professor, Université du Luxembourg

Monsieur Jean-Louis Boimond ..... *Rapporteur*
Professeur des Universités, Université d'Angers

Monsieur Ludovic Noirie ..... *Examinateur*
Ingénieur de recherche, Alcatel-Lucent Bell-labs Villarceaux

Monsieur Xavier Lagrange ..... *Examinateur*
Professeur des Universités, Télécom Bretagne

Monsieur Claude Jard ..... *Directeur de thèse*
Professeur des Universités, Université de Nantes

Madame Anne Bouillard ..... *Encadrante*
Maître de conférences, École Normale Supérieure

INRIA / IRISA équipe SUMO
Antenne de Bretagne de l'École Normale Supérieure de Cachan
Avenue Robert Schuman, 35170, BRUZ

À mon père.

L'AURORE
Je Remercie. . . !

Je remercie Claude Jard et Anne Bouillard qui ont accepté de diriger et d'encadrer cette thèse après mes six mois de stage de M2. Que soit aussi remercié Benoît Ronot pour l'intéressante collaboration ayant permis l'étude de véritables problèmes rencontrés par Alcatel-Lucent Bell-labs ainsi que pour son accueil sur le site de Villarceaux.

Je remercie les membres du jury, Stefan Haar, Nicolas Navet, Jean-Louis Boimond, Ludovic Noirie et Xavier Lagrange, qui ont accepté d'évaluer ce travail et pour leurs remarques constructives.

Je remercie également l'équipe DISTRICOM, devenue SUMO pour m'avoir accueillie. Je remercie particulièrement Albert Benveniste et Éric Fabre pour les utiles conseils quant à la suite de ma carrière. Un grand merci également à la toujours souriante Laurence Dinh pour l'aide précieuse afférente aux démarches administratives et aux créations de missions. Je ne saurais être exhaustive, mais je n'aurais garde d'oublier dans mes remerciements Carole et Rouwaida pour les pauses thé et déjeuner, les discussions que nous avons eu sur nos doutes et nos difficultés.

Je remercie maintenant mes amis en dehors du labo. Je pense, tout d'abord, à mes amies de Lycée : Marguerite, Élise, Katell et Tréfina. Malgré la distance qui nous sépare, nous avons réussi à maintenir notre amitié et elles m'ont soutenues tout au long de mes études. Je tiens également à remercier Florian, Guillaume ainsi que Laure et Philippe (pour toutes ces soirées très agréables), Matthieu Dorier (pour nos sessions musique), Claudine Cadran et Mathieu. Merci Jonathan pour nos quotidiennes conversations matinales et ton humour. Je souhaiterais terminer en remerciant Valérie, une amie comme on en rencontre pas beaucoup dans une vie. Merci pour nos *soirées filles* !

Je remercie tous les membres du club de danse sportives de Rennes et particulièrement mon incroyable danseur Eryck Le Lann. On a toujours réussi à s'entrainer, même dans les moments les plus intenses de ma thèse, et sans jamais se prendre la tête. Merci ! Danser et concourir avec toi était très agréable et intéressant.

Je remercie ma famille pour ses encouragements tout au long de mes études. Merci Alizé pour nos après-midi Twix - thé - boulot, ponctués de "PAU.SE", "QUOI ?", "PAU.SE" (référence Kaamelott saison 2 épisode 54, *La corde*) et les fameux "marreeuuuhhh" lorsque nos cerveaux n'en pouvaient plus. Merci aussi à mon papa pour son investissement et son aide depuis ... le primaire ! Papa, je te dédie cette thèse. Je remercie ma maman pour ses encouragements à chaque fois que j'ai eu des difficultés. Merci aussi à Éric, Chantal et Robinson.

Finally, I would like to thank Michael who has been here for me when my studies were driving me crazy this year. Thank you very much for your support. Our conversations and our multiple interests helped me to finish this PhD relaxed.

Table des matières

Introduction **9**

1 Préliminaires **13**
Quelques notations générales . . . 14
1.1 Topologie des réseaux de télécommunication et flux . . . 15
1.1.1 Topologie des réseaux . . . 15
1.1.2 Flux . . . 15
1.2 Le routage internet . . . 15
1.3 Le protocole OSPF . . . 16
1.3.1 Introduction . . . 16
1.3.2 La procédure de maintien des connexions entre voisins . . . 17
1.3.3 La procédure d'inondation . . . 17
1.4 Extraction de flux de données . . . 18

I Prévention de pannes **21**

2 Définition d'indicateurs de stabilité du réseau **25**
Notations du chapitre . . . 26
Introduction . . . 27
2.1 État de l'art . . . 28
2.2 Flux et contraintes . . . 30
2.2.1 Flux . . . 30
2.2.2 Définition de contraintes . . . 31
2.2.3 Rupture de contraintes . . . 33
2.3 Observation des flux : calcul d'indicateurs de stabilité . . . 34
2.3.1 Calcul des points critiques et sortant . . . 35
2.3.2 Étude fine du comportement d'un flux . . . 35
2.3.3 Quelques cas d'étude . . . 38
2.3.4 Étude de comportement à long terme d'un flux . . . 42
2.3.5 Discussion sur les algorithmes . . . 42
2.4 Résultats . . . 44
2.4.1 Mise en œuvre . . . 45
2.4.2 Application à la déviation de comportement . . . 46
2.4.3 Application au protocole OSPF . . . 47
Conclusion du chapitre . . . 53

3 Détection d'événements majeurs par corrélation d'alarmes rares **55**
Notations du chapitre . . . 56
Introduction . . . 57

3.1 État de l'art . . . 57
3.2 Graphes de dépendance des motifs pertinents . . . 60
3.2.1 Construction des motifs pertinents . . . 60
3.2.2 Dépendances des motifs pertinents . . . 62
3.3 Résultats . . . 63
3.3.1 Mise en œuvre . . . 64
3.3.2 Contexte d'étude . . . 64
3.3.3 Étude d'un premier élément de réseau . . . 65
3.3.4 Étude d'un second élément de réseau . . . 68
Conclusion du chapitre . . . 71

II Amélioration et contrôle des délais d'attente 73

4 Calcul des paramètres optimaux d'un protocole 77
Notations du chapitre . . . 78
Introduction . . . 79
4.1 Modélisation et simulation du processus d'inondation . . . 80
4.1.1 Réseau de Petri temporisé . . . 80
4.1.2 Modélisation du processus d'inondation . . . 80
4.1.3 Validation du modèle . . . 83
4.1.4 Mise en œuvre . . . 84
4.2 Étude de la longueur de la période . . . 85
4.2.1 Cas d'un trafic fluide . . . 85
4.2.2 Système congestionné . . . 85
4.2.3 Cas limite . . . 86
4.2.4 Condition suffisante de congestion . . . 86
4.3 Calcul de décalages initiaux optimaux . . . 88
4.3.1 Modèle de contraintes et notations . . . 88
4.3.2 Solution exacte par programmation linéaire . . . 90
4.3.3 Algorithme glouton . . . 91
4.3.4 Résultats de simulations avec décalages initiaux . . . 93
Conclusion du chapitre . . . 94

5 Calcul de bornes de performance déterministe avec priorités fixes 97
Notations du chapitre . . . 98
Introduction . . . 99
5.1 Le *network calculus* . . . 100
5.1.1 Flux et contrôle de flux . . . 101
5.1.2 Bornes caractéristiques . . . 106
5.1.3 Contraintes dans les réseaux complexes . . . 107
5.2 État de l'art . . . 109
5.2.1 Analyse de flux séparés . . . 109
5.2.2 Algorithme de programmation linéaire (LP) . . . 111
5.2.3 Priorités fixes . . . 112
5.3 Étude du délai maximal : politique de service à priorités fixes . . . 113
5.3.1 Modèle du réseau . . . 113
5.3.2 Condition nécessaire et suffisante pour délai borné . . . 115
5.3.3 Approche par programmation linéaire . . . 116
5.4 Exactitude des bornes calculées . . . 124
5.4.1 Étude complète d'un exemple . . . 124

5.4.2 Ajout de contraintes supplémentaires 125
5.4.3 Calcul de bornes exactes 125
5.4.4 Borne inférieure du pire délai 127
5.5 Résultats et comparaison aux méthodes existantes 127
5.5.1 Mise en œuvre 128
5.5.2 Étude d'un premier type de réseau 128
5.5.3 Étude d'une seconde topologie 129
Conclusion du chapitre 130

Conclusion et perspectives **131**

Introduction

Les réseaux de télécommunication se présentent aujourd'hui comme des assemblages complexes de composants. Ces composants, installés sur les routeurs, sont de plus en plus observés et commandés pour optimiser la qualité du service rendu de transport. Relativement simples à l'origine, ils ont progressivement évolué sous la pression des avancées technologiques qui ont autorisé une diversification croissante des informations transportées suscitant une appétence grandissante de la part des consommateurs.

Aujourd'hui, l'activité économique exige des réseaux fiables et performants. Z. Kerravala ([42]) a réalisé une étude estimant le coût d'une panne de réseau entre de 0.09 à 4.5 millions de dollars par heure selon les secteurs de marchés.

Les grandes entreprises elles-mêmes ne sont pas à l'abri de ces problèmes. Ainsi, l'échangeur central de l'entreprise BlackBerry subit-il une panne de 24 heures en 2011. Ce problème a pénalisé des millions d'utilisateurs dans le monde. Plus récemment, l'entreprise française France Telecom a du faire face à une panne de réseau sans fil durant 10 heures entre le 6 et le 7 juillet 2012. Durant une conférence de presse le Président-Directeur Général n'a pas voulu communiquer le coût de cette panne. Néanmoins, il a mentionné que des cadeaux (heures d'appel, téléphones, etc.) seraient offerts aux utilisateurs victimes de cette défaillance. Ces pannes prouvent la nécessité de renforcer les activités de gestion, suivi et prévention.

En particulier, maintenir la stabilité des réseaux est une tâche critique. Malheureusement, étant donné leur nature complexe, prédire leurs comportements n'est pas simple. L'ensemble des activités, méthodes ou procédures visant à les surveiller est connu sous le terme de *management* de réseaux. Il s'agit d'un domaine capital devant assurer leur bonne santé. Hélas, depuis quelques décennies, le management de réseau s'est considérablement compliqué. Il requiert l'implication d'experts et l'utilisation d'outils de haut-niveau. L'introduction de nouvelles technologies telles que les processus de réseaux autonomes et les *Software Defined Network* (nouvelle architecture réseau où le plan de contrôle et le plan de données sont complètement séparés) ne vont pas alléger les problèmes de management des réseaux émergents.

Afin de rendre les réseaux plus stables, les méthodes de management doivent pouvoir s'adapter à des systèmes toujours plus grands, devenir plus efficaces dans le traitement des informations et réagir plus rapidement aux pannes observées. Elles doivent donc répondre à des contraintes de fonctionnement très fortes. Deux des critères essentiels pour ces techniques s'appellent la *proactivité* et la *prédictivité*. La proactivité désigne un processus de détection des erreurs avant l'occurrence d'une panne et la prédictivité consiste à étudier le réseau sur une période relativement longue afin de prédire les prochaines anomalies et éviter de futures pannes. Ces critères impliquent notamment que les techniques développées effectuent des analyses en temps-réel. Enfin l'analyse proposée ne doit pas représenter une charge conséquente pour les éléments dans lesquels elle est implantée. Elle doit donc consommer peu de ressources (temps de calcul, espace mémoire). **Un premier aspect de la thèse défendue dans ce document est l'introduction de**

méthodes légères pour prédire les évolutions de performances.

La fiabilité d'un réseau repose également sur une bonne conception. Celle-ci peut être améliorée par la définition plus précise de bornes de performances générales : taille des files d'attente, charge du réseau, délai d'attente, etc. Depuis de nombreuses années, les chercheurs se sont intéressés au développement de techniques fournissant des garanties sur les performances générales des réseaux. Ce type de recherche débouche sur de nombreuses applications. Prenons l'exemple des algorithmes *control-rate* modulant le débit des flux de messages en fonction de l'occupation du réseau dont l'efficacité dépend directement des bornes de performances estimées. Beaucoup d'études ont été menées afin de trouver une méthode calculant, le plus précisément possible, cette occupation. Par exemple, les techniques de contrôle de flux, introduites dans les articles [6, 7], définissent la stabilité actuelle du réseau en fonction de la taille de la file d'attente du routeur le plus chargé. Plus tard, Kelly et al. ([41]) tentent d'améliorer cette méthode par le calcul de la bande passante libre. L'optimisation du trafic autoroutier représente un autre exemple d'application. Le système AHS (*Automated Highway System*) propose une amélioration du débit de circulation des véhicules par la formation assistée de pelotons ayant des espacements restreints entre chaque véhicules. La fiabilité de ce système est capitale afin d'assurer la sécurité des automobilistes. Malheureusement, Liu et al.([53]) montrent que des délais pouvant être longs, ou des rejets de messages corrompent la sécurité du réseau.

Les délais de messages dépendent de nombreux paramètres, établis il y a plusieurs années, qui n'ont jamais été remis en cause. Ainsi, d'importantes sur-estimations (pouvant porter sur des tailles de mémoires nécessaire, la charge maximale d'un routeur, etc.) ont été définies afin de garantir une qualité de service suffisante. De même, les paramètres des protocoles ont été très finement établis par des normes et n'ont plus été modifiés de peur de bouleverser l'équilibre du système. Cependant, les réseaux ont bien évolué depuis ces définitions et évolueront encore. Comment savoir alors si ces paramètres seront toujours adéquats ? **Le deuxième aspect développé dans cette thèse est la question de la sensibilité aux valeurs des paramètres.**

Dans cette thèse, nous présentons plusieurs approches afin de prévenir les risques de pannes des réseaux et d'évaluer leurs performances en fonction de paramètres.

Dans un premier temps, nous nous intéressons à l'élaboration de méthodes de management préventives étudiant la stabilité des réseaux.

Notre cahier des charges consistait à proposer des méthodes respectant les critères, discutés précédemment : de proactivité, prédictivité et faible consommation des ressources. A cela, les ingénieurs d'Alcatel-Lucent Bell-labs, avec lesquels nous avons collaboré, ont ajouté la nécessité de produire des méthodes génériques, pouvant s'appliquer à n'importe quel type de réseau.

Le principe des techniques élaborées consiste à émettre une alerte avant l'occurrence d'une défaillance. En effet, les pannes sont bien souvent précédées de signaux indiquant que le système sort de son état de fonctionnement normal. Cependant, ces indicateurs peuvent être difficiles à repérer face à la grande quantité de données à gérer.

Nous proposons une première méthode détectant des anomalies dissimulées dans des flux de messages quelconques par la définition de contraintes temporelles encadrant le flux. Dans un deuxième temps pour présentons une méthode se concentrant sur les flux d'alarmes. Ces messages ont été initialement créé pour indiquer qu'un équipement sort de son état de fonctionnement normal. Cette pratique s'avérant utile, de nombreux types d'alarmes virent le jour, n'informant désormais plus uniquement de menaces ([32]). Nous proposons une méthode établissant des graphes de corrélation entre les alarmes révélatri-

ces d'un danger imminent qui permettent de déterminer la cause d'une panne et de suivre l'évolution de problèmes apparus. Grâce à ces informations, un diagnostic des problèmes majeurs du réseau peut alors être établi.

Dans un second temps, nous essayons de mieux comprendre le comportement général des réseaux afin de minimiser certaines sur-estimations stipulées précédemment. Cette préoccupation soulève de nombreux questionnements : pourquoi certains paramètres de protocoles ont-ils été si finement établis ? Quels sont les critères de ces définitions ? Quelle est la conséquence d'une modification de ces valeurs ? Peut-on améliorer l'estimation des délais dans les réseaux ? ...

Pour répondre à cette problématique, nous nous penchons tout d'abord sur l'étude du comportement d'un protocole périodique : le protocole de routage OSPF (*Open Shortest Path First*). Afin de comprendre son fonctionnement, il est modélisé par un réseau de Petri. Puis, le problème de la définition de certains de ses paramètres et de son effet sur le caractère périodique du protocole est abordé.

Pour finir, nous étudions les problèmes de contrôle de performance des systèmes dynamiques critiques dans le cadre de serveurs munis d'une politique de service à priorités fixes. La méthode introduite calcule des délais pire-cas entre l'entrée et la sortie du système en fonction de la capacité de traitement des routeurs et de la vitesse d'arrivée des messages.

Chapitre 1

Préliminaires

Sommaire

Quelques notations générales 14
1.1 Topologie des réseaux de télécommunication et flux 15
1.1.1 Topologie des réseaux 15
1.1.2 Flux 15
1.2 Le routage internet 15
1.3 Le protocole OSPF 16
1.3.1 Introduction 16
1.3.2 La procédure de maintien des connexions entre voisins 17
1.3.3 La procédure d'inondation 17
1.4 Extraction de flux de données 18

Avant de présenter les travaux réalisés dans cette thèse, il convient d'en présenter quelques éléments généraux. Ainsi, ce chapitre introduit tout d'abord un ensemble de notations génériques, la représentation de la topologie des réseaux utilisée ainsi que la notion de flux considérée. Ensuite, le principe général du routage internet et la description du protocole d'étude OSPF sont présentés. Finalement, la plateforme de tests utilisée est introduite.

Quelques notations générales

La table 1.1 liste les notations utilisées dans ce manuscrit.

Notations	Significations
	Notations usuelles
$\mathbb{N}$	Ensemble des entiers naturels
f	Un flux
s, t	Dates
	Notations de topologie
$G = (V, E)$	Graphe orienté représentant la topologie du réseau
N_r	Nombre de routeurs du réseau
R_i	i^{ième} routeur du réseau
$\mathcal{V}(R_i)$	Ensemble des voisins de R_i
$\lvert\mathcal{V}(R_i)\rvert$	Nombre de voisins de R_i
	Notations spécifiques du protocole OSPF
d_i	Délai initial de R_i
$Hello$	Message de maintien des liaisons entre voisins
LSA_i	Message *link state advertisement* de R_i
ACK	Accusé de réception
T_r	Taille de la période de rafraîchissement
T_t	Temps de traitement d'un message
T_e	Temps d'envoi d'un message

TABLE 1.1 – Listes des notations générales.

1.1 Topologie des réseaux de télécommunication et flux

1.1.1 Topologie des réseaux

La topologie d'un réseau est modélisée par un graphe non-orienté $G = (V, E)$ où V est l'ensemble des routeurs du réseau et $E \subseteq V \times V$ est l'ensemble des arcs les reliant. Nous notons N_r le nombre de routeurs du réseau. Le $i^{ième}$ routeur du graphe est dénoté R_i et $V = \{R_1, \ldots, R_{N_r}\}$. Si $(R_i, R_j) \in E$ cela signifie que R_i et R_j sont voisins. L'ensemble des voisins de R_i, noté $\mathcal{V}(R_i)$, contient $|\mathcal{V}(R_i)|$ éléments.

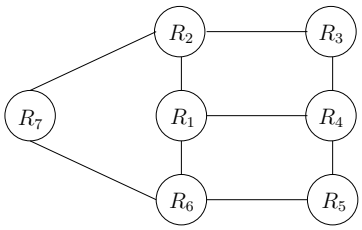


FIGURE 1 – Réseau à 7 routeurs.

La figure 1 représente un exemple de réseau composé de 7 routeurs : $V = \{R_1, R_2, R_3, R_4, R_5, R_6, R_7\}$. Le routeur R_1 a 3 voisins ($|\mathcal{V}(R_1)| = 3$) : $\mathcal{V}(R_1) = \{R_2, R_4, R_6\}$.

1.1.2 Flux

Dans cette thèse, nous étudions constamment des flux de messages. Un flux f est un ensemble de données ordonnées entrant ou sortant d'un système. Au fil des chapitres, différents aspects en sont considérés. Ainsi, dans le chapitre 2, un flux est défini par la suite croissante des dates d'arrivées des messages car l'étude se focalise sur leurs irrégularités. Nous analysons ensuite les alarmes et recherchons les corrélations entres-elles. Un flux désigne alors une suite composée des noms des alarmes. Dans le chapitre 4, les relations entre les messages d'un protocole sont étudiés. Ainsi, les flux ne sont plus considérés dans leur globalité et l'intérêt est porté sur les interactions entre les messages les composant. Finalement, l'étude du délai de transmission maximal d'un message d'un bout à l'autre du réseau, nous amène à reprendre la définition générale d'un flot et à considérer les effectifs des arrivées cumulées.

1.2 Le routage internet

L'internet est un réseau de communication mondial interconnectant de nombreuses machines. Afin de simplifier sa gestion, il est composé de sous-réseaux, appelés systèmes autonomes, reliés entre eux. Ceci implique que le routage internet s'articule autour de deux types de protocoles : ceux effectuant les échanges d'informations entre les systèmes autonomes (EGPs : *Exterior Gateway Protocols*) et ceux gérant la circulation des données à l'intérieur des systèmes autonomes (IGPs : *Interior Gateway Protocols*). Ces derniers sont classés en deux catégories.

- Dans les protocoles dits à *vecteur de distance*, tels que RIP ([55]), les routeurs disposent d'une vue restreinte du réseau. En effet, chacun maintient, dans sa table de routage, la distance le séparant de chaque destination atteignable ainsi que le nom du voisin permettant de lui envoyer un message. Afin de maintenir la base de données à jour, des messages contenant des paires (destination, distance) sont transmis périodiquement.

– Dans ceux dits à *état de liens*, tels que OSPF, les routeurs disposent tous de la même connaissance du réseau. Ici, les messages, envoyés périodiquement, informent de l'état des liens entre deux routeurs (d'où le nom d'état de lien qui lui est attribué). Chaque routeur collecte les informations de la topologie puis applique l'algorithme du plus court chemin de Disjkstra.

Les chapitres 13 et 27 du livre [14] et le livre [29] fournissent davantage de détails sur le routage, les protocoles et les architectures internet.

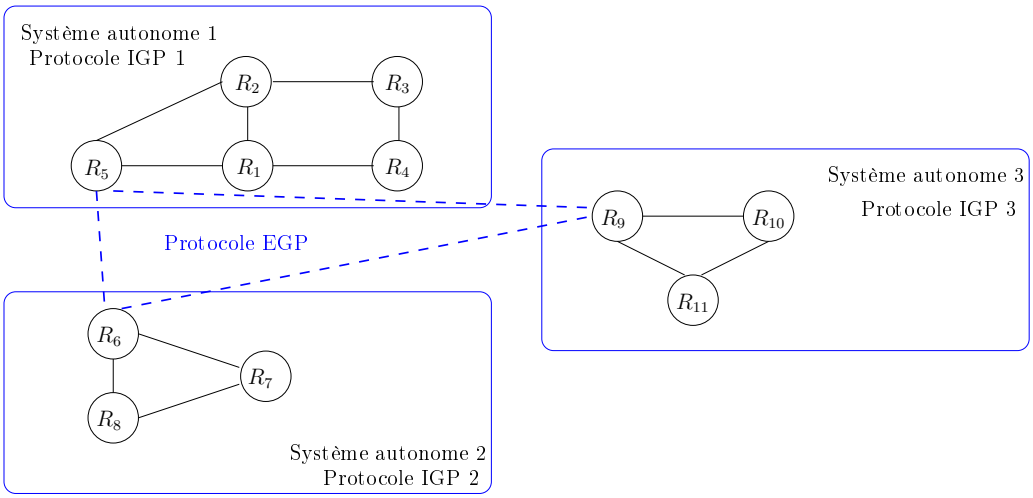


FIGURE 2 – Réseau divisé en 3 systèmes autonomes.

La figure 2 représente un exemple de réseau composé de 3 systèmes autonomes. Le routage de chacune des trois régions est effectué par un protocole IGP, qui peut être différent d'un système à l'autre. Les connexions entre systèmes sont régies par un protocole EGP.

1.3 Le protocole OSPF

1.3.1 Introduction

Le développement du protocole OSPF débuta en 1987 suite aux besoins de la communauté internet de disposer d'un protocole IGP utilisable sur des réseaux de grandes tailles. Le protocole RIP (*Routing Information Protocol*), utilisé jusqu'alors, présente les désavantages des protocoles de type vecteur de distance. Tout d'abord, les messages envoyés dans le réseau sont très gros, car ils contiennent la liste complète du contenu des bases de données. Ainsi, le traitement de ces messages requiert un pourcentage important des ressources de calcul des routeurs. Il fallut cependant attendre la deuxième version du protocole OSPF, en 1991, pour le voir finalement appliqué.

Le principe d'OSPF consiste à fournir la même vision du système autonome à chaque routeur. Le trafic induit est minimisé grâce à l'utilisation une méthode de mise à jour incrémentielle. De plus, il assure la fiabilité seul (pas d'implication du protocole TCP).

Dans cette thèse, le protocole OSPF est utilisé à plusieurs reprises afin d'effectuer les tests des méthodes proposées. Notamment, dans le chapitre 4 une méthode d'observation des protocoles périodiques s'établit à partir de ce cas d'étude. Il dispose de plusieurs avantages le distinguant comme protocole de test intéressant. En premier lieu, son comportement général simple permet de caractériser aisément les observations récoltées. De

plus, il s'agit d'un protocole bien connu qui a fait l'objet de nombreuses recherches. Notamment, dans l'article [24], Guérin et al. proposent d'y ajouter des mécanismes afin d'assurer la qualité de service. Dans [3], Basu and Riecke s'intéressent à l'analyse de sa stabilité. Plus tardivement, Rétvári et al. ([64]) le considèrent comme cas d'étude afin de réaliser une méthode de routage auto-adaptative.

Dans les paragraphes suivants, nous ne présentons que les spécificités d'OSPF abordées dans le manuscrit : la *procédure de maintien des connexions* des routeurs voisins et la *procédure d'inondation*. Tous les aspects du protocole sont détaillés dans la RFC [58].

1.3.2 La procédure de maintien des connexions entre voisins

La procédure de maintien des connexions est réalisée, entre voisins, par l'envoi de messages Hello. Elle s'assure aussi que les communications entre routeurs sont bien bidirectionnelles. Chaque routeur envoie régulièrement des messages Hello à travers toutes ses interfaces. La période d'envoi est généralement fixée à 10 secondes et est appelée *Hello Interval*. Si les voisins répondent avant que le temps imparti ne soit écoulé, la connexion entre les deux routeurs est maintenue. Sinon, le routeur est considéré comme mort. Le compteur définissant le moment à partir duquel un routeur est supprimé de la topologie est appelé *Router Dead Interval* et est généralement fixé à 30 secondes.

1.3.3 La procédure d'inondation

La procédure d'inondation assure que chaque routeur dispose d'informations récentes sur la topologie du réseau et que ces informations sont les mêmes pour tous.

Chaque routeur d'un réseau maintient une base de données contenant des informations telles que : les noms des éléments, le temps nécessaire pour envoyer un message vers un routeur donné, etc. Cette base de données est appelée *table de routage*. Elle doit être mise à jour périodiquement afin d'assurer le bon fonctionnement des envois de messages. Cette période de rafraîchissement, également appelée *période d'inondation*, débute toutes les T_r secondes (aux instants $\forall k \in \mathbb{N},\ kT_r$). La longueur de la période est généralement fixée à 1800 s.

Cette procédure fonctionne correctement si tous les routeurs disposent de la même base de données à la fin de chaque période. Cette propriété est appelée la *convergence des bases de données*. En général, nous dirons simplement que le protocole *converge*.

Chaque période d'inondation est réalisée de la façon suivante. Le routeur R_i transmet sa vision de l'état du réseau à tous ses voisins par l'envoi d'un message appelé LSA, pour *Link-State Advertisement*, noté LSA_i. Ce procédé génère un flux important de messages dans le réseau, d'où le nom d'inondation qui lui est attribué. Afin de réduire la charge instantanée du réseau, les envois de LSA sont décalés en chaque routeur par une horloge appelée *décalage initial*, noté d_i. Ainsi, R_i envoie LSA_i à la date $\forall k \in \mathbb{N},\ d_i + kT_r + T_t$.

Dans le chapitre 4, nous réalisons une étude du protocole OSPF. Afin de simplifier l'analyse, nous supposons que le temps de traitement ou de création (resp. le temps d'envoi) d'un message quelconque, noté T_t (resp. T_e), est fixe.

Supposons qu'un routeur R_j reçoive un LSA_i et qu'il commence son traitement à la date t. Le routeur R_j termine alors le processus de traitement de LSA_i à la date $t + T_t$. Durant ce temps, R_j analyse le contenu de LSA_i : il met sa base de données à jour, en conservant les informations les plus récentes. Si R_j a rafraîchi sa connaissance du réseau, il transmet sa nouvelle vision du réseau à tous ses voisins (sauf R_i) par le biais d'un nouvel LSA. Celui-ci est envoyé par R_j à la date $t + T_t$ et est reçu à la date $t + T_t + T_e$ par ces voisins.

À la réception de LSA_i, R_j effectue une deuxième opération : il envoie un accusé de réception à R_i noté ACK, de l'anglais *acknowledgement*. Ce dispositif témoigne de la bonne réception d'un message. En effet, lorsque R_i envoie son LSA_i, il attend un ACK de chacun de ses voisins. S'il ne le reçoit pas suffisamment tôt, R_i suppose que le LSA a été perdu et le retransmet à chaque voisins n'ayant pas encore répondu.

Le processus de rafraîchissement s'achève lorsque tous les routeurs ont convergé vers la même base de données. La figure 3 représente les échanges de messages entre R_i et R_j lors de l'émission de LSA_i.

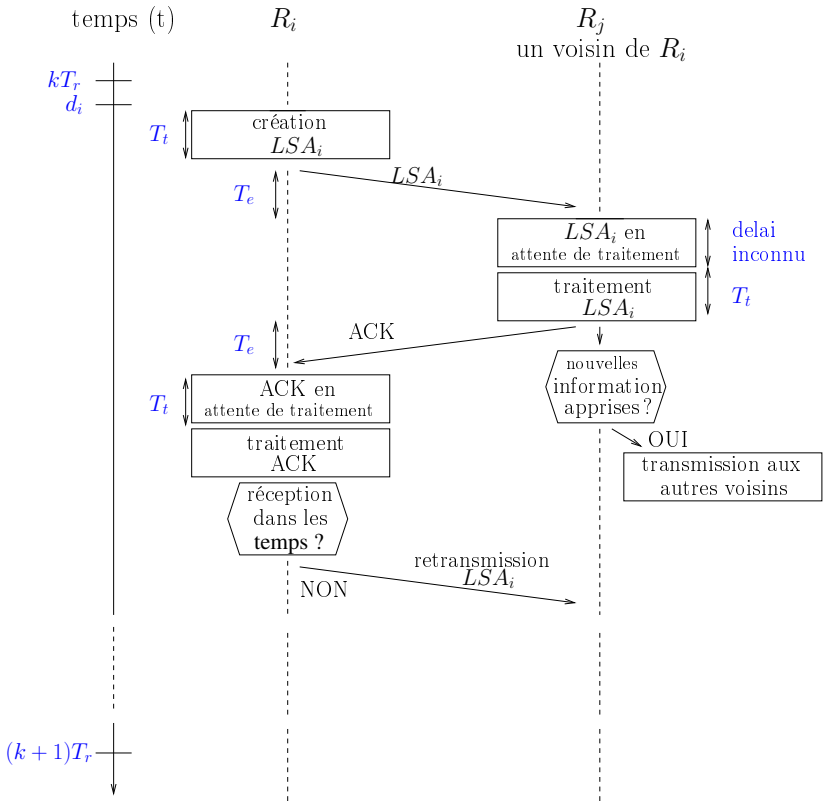


FIGURE 3 – Principe de la procédure d'inondation.

Remarque 1.3.1. *Un routeur prend en compte un LSA s'il respecte un ensemble de propriétés. La plus importante est l'âge du LSA. En effet, si un LSA est trop ancien, les données qu'il contient peuvent être obsolètes. Il est alors ignoré.*

Remarque 1.3.2. *En réalité, le protocole OSPF n'effectue pas le processus d'inondation entre tous les routeurs mais selon un arbre couvrant de la topologie du réseau.*

1.4 Extraction de flux de données

Afin d'obtenir des traces d'échanges de messages du protocole OSPF les plus réalistes possibles, nous avons émulé ce protocole grâce au logiciel de routage Quagga [61]. Les résultats des émulations sont enregistrés et transmis dans des fichiers de sortie connus sous le nom de fichier logs. Chaque nœud du réseau fonctionne à l'aide d'une machine Linux Ubuntu qui accueille une instance du logiciel de routage Quagga. Ceci procure une plateforme de test qui réagit comme un véritable réseau, pouvant être surveillé par des outils classiques de management réseau.

Les expériences étudiant le protocole OSPF présentées dans ce manuscrit sont réalisées avec la topologie du réseau de télécommunication des grandes villes allemandes. Il s'agit du réseau de 17 routeurs représenté en figure 4. Cette topologie et ses spécifications nous ont été fournies par Alcatel-Lucent Bell-labs. Elle a l'avantage d'être suffisamment grande pour fournir une étude significative de la performance de nos algorithmes dans un contexte réaliste, mais également suffisamment petite pour obtenir aisément une vision globale du comportement du réseau.

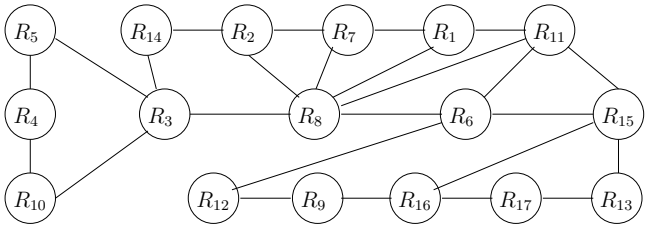


FIGURE 4 – Topologie allemande.

Première partie

Prévention de pannes

De nombreuses défaillances ne sont pas brusques mais résultent souvent d'une suite d'événements mineurs qui font progressivement dévier le réseau - ou certains éléments - de leur état de fonctionnement normal vers un état fautif. Hélas la détection de ces événements est difficile. En effet, il s'agit de rechercher l'apparition d'événements rares et mineurs, parfois cycliques, dans l'abondante quantité d'informations circulant dans un réseau.

Reprenons l'exemple (évoqué en introduction) de la panne subie dans les installations de France Telecom entre le 6 et le 7 juillet 2012. Un problème majeur réside dans le fait que l'équipe technique travaillant sur leurs réseaux n'a pas su déterminer la provenance de la panne. Elle a pu détecter l'équipement touché, mais pas la cause initiale du problème. Cela signifie qu'une anomalie est toujours cachée dans leurs systèmes et risque donc de déclencher à nouveau la panne apparue en 2012.

L'idée générale des méthodes développées dans cette partie, repose sur le fait qu'un marqueur d'instabilité se dissimule directement dans les flux de données échangées. Malheureusement, les méthodes de management actuelles ne repèrent pas encore ce genre d'erreurs. La gestion de l'évolution des réseaux nécessite donc la création de nouveaux concepts et algorithmes devant respecter de fortes contraintes (calcul en temps réel, adaptation à différents contextes, proactivité et prédictivité) afin de pouvoir être exploités. Il s'agit également de créer des méthodes traitant automatiquement les problèmes observés ou fournissant des résultats facilement interprétables par un humain.

Ainsi, dans un premier temps, nous nous intéressons à la détection d'anomalies cachées dans des flux de messages quelconques. À ce titre, l'algorithme introduit dans le chapitre 2 observe un flux de messages quelconques, effectue une analyse rapide de son comportement et en déduit des indicateurs de stabilité. Il s'agit d'une analyse proactive, prédictive et facilement interprétable permettant aux administrateurs réseaux de réagir avant une panne.

La nécessité de développer des études de management toujours plus performantes a motivé les industriels à fournir des outils permettant de meilleures observations des réseaux. Cette tendance a impliqué la mise en place des alarmes : messages développés afin d'informer les administrateurs de la santé d'un système. Aujourd'hui, les alarmes quotidiennement créées ne sont plus humainement gérables et nécessitent l'intervention de méthodes de corrélation et de diagnostic. Le chapitre 3 est voué à l'analyse de ce problème. Il propose une méthode prédictive visant à corréler les alarmes générées dans un réseau. L'idée est de fournir une étude permettant la détection de la cause initiale d'une perturbation. Cette méthode réutilise la technique introduite au chapitre précédent afin de guider l'expert réseau dans la détection du problème source, afin que celui-ci puisse réagir plus rapidement.

Chapitre 2

Définition d'indicateurs de stabilité du réseau

Sommaire

Notations du chapitre **26**
Introduction **27**
2.1 État de l'art **28**
2.2 Flux et contraintes **30**
2.2.1 Flux 30
2.2.2 Définition de contraintes 31
2.2.3 Rupture de contraintes 33
2.3 Observation des flux : calcul d'indicateurs de stabilité **34**
2.3.1 Calcul des points critiques et sortant 35
2.3.2 Étude fine du comportement d'un flux 35
2.3.3 Quelques cas d'étude 38
2.3.4 Étude de comportement à long terme d'un flux 42
2.3.5 Discussion sur les algorithmes 42
2.4 Résultats **44**
2.4.1 Mise en œuvre 45
2.4.2 Application à la déviation de comportement 46
2.4.3 Application au protocole OSPF 47
Conclusion du chapitre **53**

Le travail présenté dans ce chapitre résulte de travaux effectués en collaboration avec Anne Bouillard et Benoît Ronot et fait l'objet d'un article [85] présenté lors de la conférence CNSM'12 en 2012. Un rapport de recherche contenant davantage de résultats théoriques et de preuves a été publié [86]. Finalement, un brevet [87] a été déposé afin de protéger les algorithmes présentés dans ce chapitre.

Notations du chapitre

La table 2.1 liste les notations spécifiques à ce chapitre.

Symboles	Significations
	Définition de points
x_n	Date d'arrivée du n$^{\text{ième}}$ message
$P_n = (x_n, n)$	n$^{\text{ième}}$ point
$\underline{P}$ et $\underline{c}$	Point critique inférieur et numéro de ce point
$\overline{P}$ et $\overline{c}$	Point critique supérieur et numéro de ce point
P	Point courant
P_p	Point précédent le point courant
P_s	Point sortant des contraintes
	Contraintes
$\underline{\alpha}$	Courbe des arrivées inférieure
$\overline{\alpha}$	Courbe des arrivées supérieure
T	Délai maximal
σ	Nombre de messages arrivant simultanément
ρ	Taux d'arrivée à long terme
	Notations mineures
$FluxEntrée$	Flux d'entrée de l'algorithme 2
$FluxSortie$	Flux de sortie de l'algorithme 2
$\neg$	opérateur unitaire de négation
e_i	i$^{\text{ème}}$ exposant décrivant la structure des mot équilibrés
pp	Préfixe propre
f_{ex}	Exemple simple de flux
f_p	Exemple de flux périodique
f_{eq}	Exemple de flux équilibré
w_i et w'_i	i$^{\text{ème}}$ facteurs du mot équilibré
N_p	Nombre de messages dans la période d'un flux périodique
N_{niv}	Nombre de niveaux de l'algorithme 2.
N_{MAM}	Nombre de messages dans la fenêtre glissante de MAM

TABLE 2.1 – Listes des notations du chapitre 2.

Introduction

Il est admis depuis longtemps que les pannes des éléments d'un système ou les problèmes de configuration sont les plus fréquentes causes de dysfonctionnement. Cependant, les erreurs de codes définissent des anomalies cachées et imprévisibles qui doivent être prises en considération. Un exemple concret de ce type de problème est une mauvaise gestion de la mémoire d'un routeur par un processus qui oblige ce dernier à redémarrer 3 ou 4 fois par an. Ce type d'évènements génère des bugs qui suscitent des comportements imprévisibles de la part des éléments du réseau. Il est bien connu que les programmes contiennent de nombreuses erreurs pouvant impliquer des pannes. La NASA a calculé une moyenne de 4 bugs par millions de lignes de code. Sachant que le coût d'un bug représente 850$([68]), ce type d'erreur n'est pas négligeable. L'objectif de ce chapitre consiste à proposer une méthode détectant ce type d'erreurs. Face à la grande quantité d'information circulant dans le réseau, on ne peut imaginer une analyse exhaustive du contenu et de la signification de chaque message le traversant.

Nous nous intéressons aux flux de messages ayant, dans un réseau sain, un comportement général à long terme stable. Notre idée consiste à supposer qu'un flux présentant un des comportements suivants définit à long terme une menace pour la santé du réseau :

- fortes variations fréquentes ;
- déviations de la tendance générale.

Principe de la méthode développée Le principe de l'algorithme en-ligne proposé est de mettre en avant les changements de comportement de flux. L'objectif est d'en effectuer une observation à long terme et de détecter des évènements anormaux apparaissant périodiquement et/ou s'aggravant. Dans ce but, l'algorithme calcule une fenêtre temporelle autour du flux indiquant la vitesse moyenne des arrivées. Lorsqu'une donnée sort de ces contraintes, une nouvelle fenêtre est calculée afin de suivre les variations de comportements (voir figure 5). Le résultat transmis à l'utilisateur est l'ensemble des vitesses d'arrivée déterminées ainsi que l'intervalle durant lequel le flux vérifie ces contraintes.

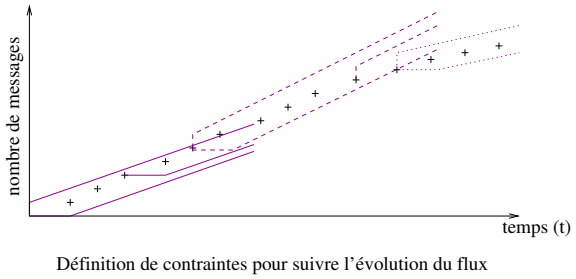


FIGURE 5 – Principe de notre algorithme.

Nous commençons ce chapitre en présentant les méthodes existantes pouvant être appliquées au problème que nous étudions (section 2.1). Nous expliquons les raisons pour lesquelles ces techniques ne satisfont pas les contraintes inhérentes au contexte de l'étude.

Dans la section 2.2 nous définissons une modélisation des flux de messages, appelée *contraintes*, afin d'observer les variations de comportement. Nous présentons alors un ensemble de définitions et de résultats théoriques inhérents à ces contraintes.

Ensuite, dans la section 2.3 nous présentons un algorithme qui pour un flux donné calcule une suite de contraintes. Nous établissons alors un ensemble de résultats théoriques dans le cas de flux périodiques et nous introduisons une version à plusieurs niveaux de l'algorithme initial qui apporte une amélioration significative. Ultérieurement, nous discutons quelques points concernant les choix d'implémentations et de paramètres effectués.

Finalement, la section 2.4 introduit un ensemble d'études réalisées sur l'architecture allemande (voir figure 4) que nous avons émulée dans les installations d'Alcatel-Lucent Bell-labs. Les résultats de l'algorithme montrent l'efficacité de la technique tant au niveau de la vitesse de calcul qu'au niveau de la simplicité de l'analyse des résultats. Nous montrons notamment que cet algorithme nous a permis de découvrir une erreur à laquelle nous ne nous attendions pas.

2.1 État de l'art

Les chercheurs ont tenté de nombreuses approches afin d'apporter une solution au problème de détection de congestions ou de pannes. Ce problème n'est pas récent ! En effet, les recherches remontent au début des années 70. Présentons une vue d'ensemble des méthodes développées.

Méthodes fondées sur l'analyse de données Les articles [33, 37, 38, 43] présentent des recherches concernant des erreurs spécifiques dans le volume de messages échangés. Par exemple, Hussain et al. ([33]) recherchent les *attaques DOS (Denial Of Service)* de dénis de service ayant pour but de rendre un service indisponible. Kim et Karp, [43], s'intéressent aux *attaques de vers (Worms)* qui se dupliquent sur les machines d'un réseau. Les méthodes spécifiques à un problème donné sont plus efficaces que celles étant génériques. Cependant, développer une méthode pour chaque erreur est difficilement envisageable. Qui plus est, davantage de problèmes apparaissent lorsque plusieurs erreurs sont combinées.

MIND (distributed index management method) [52] constitue une méthode d'indexation distribuée. Elle est basée sur une structure à deux composants :

- un ensemble de moniteurs distribués récoltent et enregistrent en continu des informations sur le trafic. Chaque moniteur génère des indices multi-dimensionnels afin de résumer les observations. Chaque dimension représente un paramètre du flux pris en considération et les indices représentent les différentes erreurs pouvant apparaître dans le réseau. On peut par exemple observer le nombre de sources distinctes envoyant de petits flots sur le port 3306 à destination du préfixe X ;
- un système de requêtes corrèle les données récoltées par différents moniteurs afin de détecter des motifs de trafic anormaux. Il est mis en place afin que l'utilisateur accède aux observations, résumées par des indices.

Néanmoins, la création des indices n'est pas automatisée : l'utilisateur doit manuellement créer un indice pour chaque ensemble de paramètres qu'il souhaite étudier conjointement. De plus, l'analyse du trafic à partir des résultats de requêtes n'est pas évidente, tout comme l'utilisation temps-réel de MIND.

L'analyse en composantes principales *PCA (Principal Component Analysis)* [35] est une technique bien connue. Il s'agit d'un procédé mathématique convertissant un ensemble d'observations corrélées en un ensemble de valeurs, non-corrélées, nommées *Composant Principaux (PCs)*. L'ensemble des PCs est divisé en deux espaces correspondant aux variations normales et anormales. L'objectif de PCA est de détecter des anomalies dans les flux d'un réseau à partir de l'espace normal construit. L'article [48] présente une utilisation de la méthode PCA pour la détection *d'anomalies de volume*. Il s'agit

d'erreurs brusques provenant de l'extérieur du réseau. Dans cette étude, les analyses sont effectuées sur un flux particulier, le flux *OD (Origine-Destination)*, qui correspond au trafic de l'entrée à la sortie de la dorsale internet (partie du réseau internet de très haut débit reliant des réseaux à l'intérieur d'un pays). Lakhina et al. montrent qu'en effet ce flux permet de mieux observer les anomalies de volume. Les méthodes [2, 45] s'attaquent également à ce problème. Elles sont cependant moins efficaces car elles appliquent des méthodes de décomposition temporelle à des résumés de flux IPs (qui peuvent être nombreux). Dans l'article [49], Lakhina et al. étendent leur précédent travail ([48]) afin de détecter des erreurs par l'observation des champs d'en-têtes de paquets IPs : une métrique d'entropie est utilisée afin de définir une distribution de trafic de chaque élément étudié et une matrice de données $n \times p$ est construite (où les lignes représentent les différentes observations et les colonnes les paramètres étudiés). Les éléments pris en compte étant corrélés, l'étude s'effectue sur un espace de dimension $m \ll p$. Néanmoins, cette méthode n'est pas temps-réel et nécessite la connaissance initiale des anomalies à rechercher. Les méthodes des articles [43, 44] définissent d'autres approches de détection d'anomalies par la création de motifs anormaux. L'efficacité de la méthode PCA à la détection d'anomalies reste cependant limitée [62]. Elle peut notamment détecter une faute qui n'en est pas une. Pire encore, l'espace normal peut être contaminé, par une anomalie importante, et mal fonctionner lors des observations suivantes, ce qui ne peut se produire avec la méthode que nous introduisons. Finalement, l'analyse pouvant s'appliquer à plusieurs flux, il est souvent difficile de trouver le flux présentant le comportement anormal détecté.

Citons finalement les méthodes présentées dans [25] et [26]. Leur principe consiste à extraire des motifs temporels d'un ensemble de données, connaissant la durée des évènements et les délais les séparant. Il s'agit d'études fines mais trop complexes pour être appliquées à des fins proactives. En effet, dans l'article [25], Guyet et Quiniou créent une méthode utilisant l'algorithme EM (*Espérance-Maximisation*) qui rend la méthode inefficace. Ainsi, dans [26], ils proposent une solution alternative regroupant des instances afin de construire des motifs "intéressants". Se heurtant à nouveau à des problèmes de complétude, ils continuent actuellement les recherches afin de proposer une solution plus efficace.

Méthodes de corrélation d'alarmes De nombreuses méthodes ont été développées afin de répondre au problème, complexe, de la corrélation d'alarmes. Les alarmes sont des messages spécifiques indiquant des caractéristiques précises du réseau. Par exemple, Chao et al. [12] établissent des corrélations probabilistes entre les alarmes et les fautes : à chaque alarme a_i est associée la probabilité $p_{i,j}$ que a_i soit émise à cause de la faute F_j. Cet article développe un mécanisme complexe basé sur un raisonnement hiérarchisé afin de corréler des alarmes.

Hélas, ce type d'approche n'est pas complètement automatisé et la surveillance des alarmes demeure encore un problème complexe. Aussi, une telle surveillance de réseau implique l'intervention d'un expert. Nous étudierons ce type d'approches en détail dans le chapitre 3 et nous présenterons un algorithme répondant aux enjeux actuels.

Méthodes pour un protocole particulier De nombreux travaux ont été réalisés pour rechercher les problèmes de congestion d'un protocole particulier. Par exemple, Han et al. [30] introduisent une évaluation des performances du protocole TCP. Notamment, une borne d'utilisation maximale des liens est calculée afin d'éviter les risques de congestion. Néanmoins, une méthode générique, pouvant s'appliquer dans n'importe quel contexte, est nécessaire.

Méthodes fondées sur une analyse statistique Willsky et Jones ([76]) présentent une méthode de filtrage adaptatif pour les systèmes linéaires soumis à des changements abrupts peu fréquents. De précédents travaux [74, 75] ont montré l'efficacité de ce modèle pour les systèmes sensibles aux pannes de composants ou aux petites non-linéarités. Cette étude est une modification et une généralisation des méthodes présentées dans [59, 66]. L'objectif est de construire un filtre adaptatif déterminant les changements abrupts du réseau. Ainsi, un filtre de Kalmann-Bucy modélise le système dynamique étudié et la méthode de taux de vraisemblance généralisée [71] détecte les changements abrupts à partir d'un seuil donné. Malheureusement, ces méthodes étudient uniquement les systèmes linéaires subissant des changements peu fréquents ce qui ne convient pas au contexte industriel que nous considérons.

Netscope ([22]) utilise également des statistiques afin de caractériser les liens d'un bout à l'autre du réseau. Cette approche, appelée *tomographie de réseau*, définit le pourcentage de perte d'informations dans les liens d'un réseau sans avoir à tous les surveiller. Des informations sur le réseau sont récoltées, puis utilisées, afin de calculer le pourcentage de perte de paquets des liens. L'objectif est de définir en temps-réel leur état de santé. Ceci est inféré à l'aide d'une représentation du réseau par un système linéaire, l'utilisation de moments de premier et deuxième ordres ainsi que des mesures d'un bout à l'autre du réseau.

Aujourd'hui, de nombreuses méthodes proposent des solutions pour la détection d'anomalies. Le défi principal réside dans la capacité à gérer une très grande quantité de données afin d'extraire les comportements anormaux (tels que des congestions ou des pannes) et de réagir rapidement avant que le système ne tombe totalement en panne. L'étude exhaustive de cet ensemble d'informations n'est pas envisageable. Ainsi, certaines méthodes proposent l'utilisation d'outils mathématiques qui restent actuellement trop complexes ou ne sont pas automatisés. D'autres étudient un protocole, un flux ou un problème particulier, ce qui ne permet pas une utilisation à grande échelle aisée. Finalement, certaines analyses utilisent des connaissances statistiques s'appliquant à un contexte donné ou calculant des valeurs telles que la probabilité de perte de messages. Dans ce chapitre, nous introduisons une méthode réactive permettant la détection déterministe d'anomalies cachées dans les réseaux quelle que soit sa topologie, sa configuration et le type de flux étudié.

2.2 Flux et contraintes

Dans cette partie, nous introduisons la définition des flux de message utilisés dans ce chapitre. Nous présentons ensuite les contraintes appliquées à ces flux. Elles seront utilisées dans la suite afin d'en modéliser le comportement.

2.2.1 Flux

Rappelons qu'un flux f est un ensemble de données ordonnées entrant ou sortant d'un système. Dans cette partie, un flux est représenté par la suite croissante $f = (x_n)_{n\in\mathbb{N}}$ correspondant aux dates d'arrivées des messages. La valeur x_n correspond à la date d'arrivée du $n^{\text{ième}}$ message. Nous supposons que $x_0 = 0$ et $\lim_{n\to\infty} x_n = \infty$.

Un flux est graphiquement représenté par une suite de points $(P_n)_{n\in\mathbb{N}} \in (\mathbb{R}_+ \times \mathbb{N})^{\mathbb{N}}$ où pour tout $n \in \mathbb{N}$, le $n^{\text{ième}}$ point est défini, par $P_n = (x_n, n)$. Le graphique représente alors les arrivées successives des messages du flux.

Exemple 2.2.1 (Flux). *La figure 6 représente les 5 premiers messages d'un flux f_{ex}. Durant 30 secondes les arrivées sont espacées de 10 secondes. Puis, les messages arrivent toutes les 20 secondes : $f_{ex} = \{x_1, x_2, x_3, x_4, x_5, \ldots\}$ avec $x_1 = 10\,s$, $x_2 = 20\,s$, $x_3 = 30\,s$, $x_4 = 50\,s$ et $x_5 = 70\,s$.*

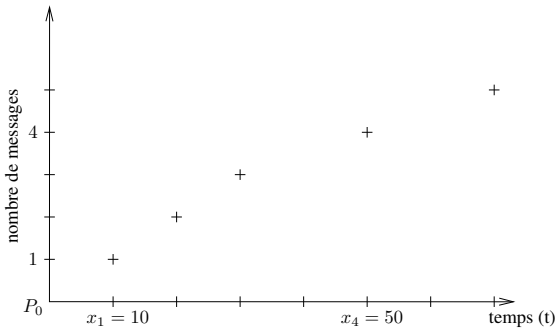


FIGURE 6 – Représentation graphique du flux f_{ex} introduit dans l'exemple 2.2.1.

2.2.2 Définition de contraintes

Le network calculus ([11, 50]) est une théorie développée afin de définir des bornes de performances déterministes pour les réseaux. Une des notions fondamentales de cette théorie est appelée *courbe d'arrivée*. Les courbes d'arrivées sont des fonctions qui permettent de modifier le fait que les arrivées de messages dans un flux sont contraintes. L'objectif est de donner une garantie sur le comportement du flux. Ces contraintes sont apportées de deux façons : par une contrainte inférieure et par une contrainte supérieure. La contrainte inférieure (resp. supérieure) borne la quantité minimale (resp. maximale) de données arrivant durant tout intervalle de temps. Ainsi, à partir de ces courbes, les arrivées de messages sont modulées afin de suivre les contraintes indiquées.

Dans ce chapitre, les flux de données sont contraints en reprenant le principe des courbes d'arrivée du network calculus. Cependant, le travail est réalisé dans le sens inverse : l'objectif est de rechercher les contraintes modélisant le flux. Ainsi, dans ce chapitre, nous ne présentons pas une analyse mathématique fine à l'aide du network calculus. Ceci fera l'objet du chapitre 5.

Un flux est dit contraint s'il suit la définition suivante.

Définition 2.2.2 (Flux contraint). *Soient $\underline{\alpha}, \overline{\alpha} : \mathbb{R}_+ \to \mathbb{R}_+$ deux fonctions croissantes et $m, n \in \mathbb{N}$, tels que $m < n$. Le flux (x_n) est dit $(\underline{\alpha}, \overline{\alpha})$-contraint sur l'intervalle de temps $[x_m, x_n]$ si $\forall m', n' \in \mathbb{N}$, tels que $m \leqslant m' < n' \leqslant n$, on a :*

Respect de la contrainte inférieure : $\underline{\alpha}(x_{n'} - x_{m'}) \leqslant n' - m'$;

Respect de la contrainte supérieure : $\overline{\alpha}(x_{n'} - x_{m'}) \geqslant n' - m'$.

Un flux est dit $(\underline{\alpha}, \overline{\alpha})$-contraint, s'il est contraint par $\underline{\alpha}$ et $\overline{\alpha}$ sur l'intervalle $[0, +\infty[$.

Graphiquement, un flux $(x_n)_{n\in\mathbb{N}}$ est $\underline{\alpha}$-contraint (resp. $\overline{\alpha}$-contraint) si pour tous $m, n \in \mathbb{N}$, $m < n$, P_n est au-dessus de $\underline{\alpha}(\cdot)$ (resp. en-dessous de $\overline{\alpha}(\cdot)$) dessinée à partir du point P_m, noté $P_m + \underline{\alpha}$ (resp. $P_m + \overline{\alpha}$).

Afin de contraindre un flux de la manière la plus pertinente possible, nous choisissons les courbes d'arrivées affines par morceaux suivantes :

$$\underline{\alpha}_{\rho,T}(t) = \rho(t - T)_+ \qquad \text{et} \qquad \overline{\alpha}_{\rho,\sigma}(t) = \sigma + \rho t.$$

La variable ρ définit la vitesse d'arrivée à long terme des messages, σ représente le nombre maximal d'arrivées simultanées et T correspond au délai maximal entre deux messages. Autrement dit, ρ représente la tendance générale du flux et σ, T définissent des bornes de tolérances. Elles introduisent une certaine souplesse lors du calcul des contraintes. La définition des courbes $\underline{\alpha}_{\rho,T}$ et $\overline{\alpha}_{\rho,\sigma}$ est motivée par le fait que la plupart des flux affichent théoriquement un comportement périodique. Ainsi, la définition de courbes linéaires ne paraît pas irréaliste.

Exemple 2.2.3 (Flux contraint). *Soient le flux f_{ex} introduit dans l'exemple 2.2.1 et les courbes d'arrivées $\underline{\alpha}_{0.1,10}$ et $\overline{\alpha}_{0.1,1}$. Les contraintes définies au fur et à mesure de l'arrivée des 5 premiers messages de f_{ex} sont représentées sur la figure 7.*

Le calcul montre que le flux est $(\underline{\alpha}_{0.1,10}, \overline{\alpha}_{0.1,1})$-contraint sur [0,50], mais pas sur [0,70]. En effet, $5 - 0 = 5 < \underline{\alpha}(x_5 - x_0) = \underline{\alpha}(70 - 0) = 6$.

Graphiquement, on remarque que les contraintes dessinées à partir des trois premiers points sont confondues avec celle représentée à partir de P_0. Elles sont donc équivalentes. Cependant la courbe $\overline{\alpha}$ dessinée à partir du point P_4 est en dessous des autres ce qui réduit la taille de la fenêtre temporelle pour les points suivants. Finalement, P_5 se situe en dessous de la contrainte inférieure, ce qui confirme le calcul montrant que f_{ex} est $(\underline{\alpha}_{0.1,10}, \overline{\alpha}_{0.1,1})$-contraint sur [0,50].

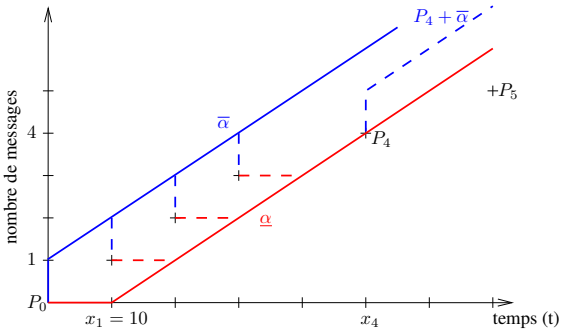


FIGURE 7 – Contraintes du flux f_{ex} introduit dans l'exemple 2.2.1.

Lemme 2.2.4 (Unicité). *Soit $(x_n)_{n\in\mathbb{N}}$ un flux de données. Alors :*

- *soit, il n'existe pas de pente ρ, telle qu'il existe σ et T, tels que (x_n) est $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-contraint ;*
- *ou, il existe une unique pente ρ et il existe σ, T tels que (x_n) est $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-contraint. De plus, pour tout $\sigma' \geqslant \sigma$ et $T' \geqslant T$, (x_n) est $(\underline{\alpha}_{\rho,T'}, \overline{\alpha}_{\rho,\sigma'})$-contraint.*

Démonstration. Il est évident que s'il existe ρ, σ, T tel que (x_n) soit $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-contraint, alors (x_n) est également $(\underline{\alpha}_{\rho,T'}, \overline{\alpha}_{\rho,\sigma'})$-contraint pour tout $\sigma' \geqslant \sigma$ et $T' \geqslant T$. La preuve est d'ailleurs apportée dans le livre [50].

Nous devons maintenant montrer qu'il n'existe pas deux pentes distinctes ρ, ρ' qui contraignent toutes les deux le flux. Supposons que (x_n) soit $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-contraint et $(\underline{\alpha}_{\rho',T'}, \overline{\alpha}_{\rho',\sigma'})$-contraint avec $\rho > \rho'$. Si l'affirmation précédente est vraie, alors (x_n) vérifie $\forall n \in \mathbb{N}$, $\rho(x_n - T) \leqslant n \leqslant \sigma' + \rho' x_n$, mais ceci ne peut pas être vrai pour $x_n > \frac{\rho T + \sigma'}{\rho - \rho'}$, ce qui termine la preuve. La figure 8 représente la contradiction exposée. En effet, à partir de $t > \frac{\rho T + \sigma'}{\rho - \rho'}$ le flux ne peut satisfaire les deux couples de contraintes car ils définissent des fenêtres temporelles disjointes. □

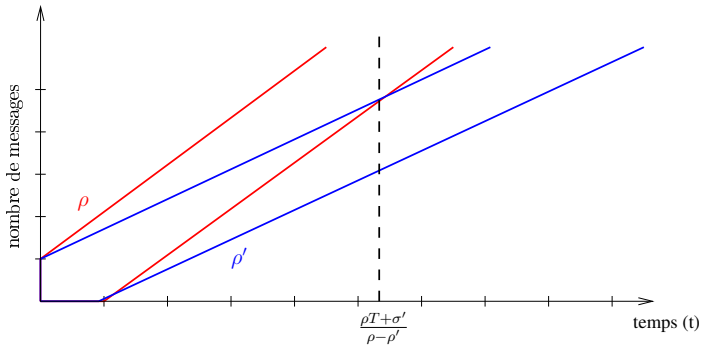


FIGURE 8 – Illustration de la preuve du lemme 2.2.4.

2.2.3 Rupture de contraintes

T et σ étant fixés, notre objectif est de trouver la pente correspondant à la vitesse moyenne d'arrivée des messages d'un flux observé. Si des changements de comportement apparaissent nous voulons faire varier la vitesse moyenne définie. Afin, de détecter efficacement ces évènements nous avons besoin de définir, à partir de contraintes données, la notion de point sortant et de points critiques.

La définition suivante établit la notion de point sortant. Il s'agit du premier point brisant les contraintes actuellement définies.

Définition 2.2.5 (Point sortant). *Soit (x_n) un flot de données, $\underline{\alpha}$ et $\overline{\alpha}$ des courbes de contraintes et m un entier. Supposons que (x_n) soit $(\underline{\alpha},\overline{\alpha})$-contraint à partir du $m^{ième}$ message arrivé. Le point sortant P_s est défini par*

$$s = \min\{p > m \mid (x_n) \text{ n'est pas } (\underline{\alpha},\overline{\alpha})\text{-contraint sur l'intervalle } [x_m, x_p]\}.$$

D'après la définition 2.2.2, vérifier que (x_n) est $\underline{\alpha}_{\rho,T}$-contraint, implique de vérifier pour tout couple m et n avec $m < n$, $\underline{\alpha}_{\rho,T}(x_n - x_m) \leqslant n - m$. En pratique, il n'est pas nécessaire d'étudier chaque point de (x_n), mais seulement le sous-ensemble fournissant les plus fortes contraintes. Nous les appelons *points critiques*. À chaque instant, il existe deux points critiques : celui définissant la plus forte contrainte supérieure et celui définissant la plus forte contrainte inférieure. Leur définition formelle est la suivante.

Définition 2.2.6 (Points critiques). *Soit (x_n) un flux de données, $\underline{\alpha},\overline{\alpha}$ des contraintes de pente $\rho \in \mathbb{R}_+$ et $m < n \in \mathbb{N}$. Supposons que (x_n) soit $(\underline{\alpha},\overline{\alpha})$-contraint sur $[x_m, x_n]$. Les messages critiques* inférieur *et* supérieur *considérant les messages P_m à P_n et les contraintes $\underline{\alpha}$ et $\overline{\alpha}$ sont $\underline{P}_n^{\rho,m} = (x_{\underline{c}_n^{\rho,m}}, \underline{c}_n^{\rho,m})$ et $\overline{P}_n^{\rho,m} = (x_{\overline{c}_n^{\rho,m}}, \overline{c}_n^{\rho,m})$ respectivement définis par :*

- $\underline{c}_n^{\rho,m} = \min\{p \in [m,n] \mid \max_{q\in[p,n]}(\frac{q}{\rho} - x_q) = \frac{p}{\rho} - x_p\}$;
- $\overline{c}_n^{\rho,m} = \min\{p \in [m,n] \mid \max_{q\in[p,n]}(x_q - \frac{q}{\rho}) = x_p - \frac{p}{\rho}\}$.

Afin de rendre la lecture plus agréable, lorsqu'il n'y a pas d'ambiguïté, les points critiques seront simplement appelés $\underline{P} = (x_{\underline{c}}, \underline{c})$ et $\overline{P} = (x_{\overline{c}}, \overline{c})$. Notons que la définition de ces points ne dépend que de ρ et est donc indépendante des valeurs σ et T choisies.

Exemple 2.2.7. *Reprenons le flux f_{ex} introduit dans l'exemple 2.2.1. Pour $q \in \{0,1,2,3\}$, $x_q - \frac{q}{\rho} = 0 < x_4 - \frac{4}{\rho} = 10$. Ainsi, P_0 est le point critique supérieur dans l'intervalle [0,30]. L'arrivée du quatrième message le remplace. Il s'agit de l'affirmation que nous avions intuitivement établie dans l'exemple précédent. En effet, graphiquement, cette définition implique que $P_4 + \overline{\alpha}$ est en dessous de $\overline{\alpha}$. On note $\overline{P}_4^{0.1,0} = P_4$.*

Proposition 2.2.8. *Soit $(x_n)_{n\in\mathbb{N}}$ un flux de données, $\underline{\alpha}, \overline{\alpha}$ des contraintes de pentes $\rho \in \mathbb{R}_+$ et $m < n \in \mathbb{N}$ tels que (x_n) soit contraint par $(\underline{\alpha}, \overline{\alpha})$ sur $[x_m, x_n]$. Alors le point critique inférieur peut être calculé par récurrence, à partir de P_m, en utilisant la formule suivante :*

$$\underline{c}_n^{\rho,m} = \begin{cases} m & \text{si } n = m \\ n & \text{si } \frac{n}{\rho} - x_n > \frac{\underline{c}_{n-1}^{\rho,m}}{\rho} - x_{\underline{c}_{n-1}^{\rho,m}} \\ \underline{c}_{n-1}^{\rho,m} & \text{sinon.} \end{cases}$$

Une formule similaire existe pour le point critique supérieur $\overline{c}_n^{\rho,m}$, en remplaçant le signe $>$ par $<$.

Démonstration. Les deux cas étant similaires, prouvons la formule sur $\underline{c}_n^{\rho,m}$. Si $n = m$, le point critique est le seul point existant : $\underline{c}_n^{\rho,m} = m$. Sinon, la définition 2.2.6 donne : $\underline{c}_{n-1}^{\rho,m} = \min\{p \in [m, n-1] \mid \max_{q\in[p,n-1]}(\frac{q}{\rho} - x_q) = \frac{p}{\rho} - x_p\}$. Donc, si $\frac{n}{\rho} - x_n > \frac{\underline{c}_{n-1}^{\rho,m}}{\rho} - x_{\underline{c}_{n-1}^{\rho,m}}$ on obtient $\underline{c}_n^{\rho,m} = n$. Sinon $\underline{c}_n^{\rho,m} = \underline{c}_{n-1}^{\rho,m}$.

□

À partir de la proposition précédente, si un flux (x_n) est $(\underline{\alpha}, \overline{\alpha})$-contraint sur $[x_m, x_n]$, vérifier qu'il l'est toujours à l'arrivée du message suivant (sur $[x_m, x_{n+1}]$) nécessite uniquement de vérifier les deux inéquations suivantes

$$\underline{\alpha}(x_{n+1} - x_{\underline{c}_n^{\rho,m}}) \leqslant n + 1 - \underline{c}_n^{\rho,m} \quad \text{et} \quad \overline{\alpha}(x_{n+1} - x_{\overline{c}_n^{\rho,m}}) \leqslant n + 1 - \overline{c}_n^{\rho,m}.$$

Finalement, nous introduisons le lemme 2.2.9 qui définit une relation simple entre le point sortant et les messages critiques.

Lemme 2.2.9. *Soient $(x_n)_{n\in\mathbb{N}}$ un flux de données, $\underline{\alpha}, \overline{\alpha}$ des contraintes d'arrivées de pente $\rho \in \mathbb{R}_+$ et $m < s \in \mathbb{N}$ tel que (x_n) est $(\underline{\alpha}, \overline{\alpha})$-contraint sur l'intervalle $[x_m, x_{s-1}]$ et P_s est le point sortant.*

- *si la contrainte inférieure est rompue, alors $\overline{c}_s^{\rho,m} = s$;*
- *si la contrainte supérieure est brisée, alors $\underline{c}_s^{\rho,m} = s$.*

Démonstration. Comme les deux cas sont symétriques, nous nous concentrons sur le premier. Soit, $\overline{P} = \overline{P}_{s-1}^{\rho,m}$ le message critique défini juste avant l'arrivée du message P_s. Comme le message $\overline{P}$ n'est pas le point sortant, on a $s - \overline{c} < \rho(x_s - x_{\overline{c}})$. D'après la proposition 2.2.8 on a $\overline{c} = s$, ce qui prouve le lemme. □

2.3 Observation des flux : calcul d'indicateurs de stabilité

La section précédente présente des fonctions affines par morceaux qui sont utilisées afin de contraindre un flux ainsi que les points nécessaires à l'étude des contraintes du flux. Notre objectif est de définir les pentes de ces fonctions afin qu'elles correspondent à la vitesse d'arrivée à long terme. Si le trafic est très régulier et que T, σ sont fixés suffisamment grands, il existe ρ tel que $(x_n)_{n\in\mathbb{N}}$ soit $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-contraint. Dans ce cas, nous souhaitons calculer cette pente. Si le trafic est variable, nous voulons calculer une suite de pentes exprimant les variations des arrivées de messages.

Dans ce but, quelques fonctions réalisant des calculs simples à partir de contraintes données sont tout d'abord présentées. L'algorithme central de ce chapitre, qui détecte les variations d'un flux accompagné des résultats théoriques de son comportement est introduit. Ensuite, une amélioration de notre algorithme est présentée et de quelques points concernant l'utilisation de celui-ci sont discutés.

2.3.1 Calcul des points critiques et sortant

Afin de faire varier les contraintes lorsque le comportement d'un flux oscille, il faut savoir calculer les points critiques et sortant présentés dans la section 2.2.

L'algorithme 1 présente des fonctions élémentaires qui étant données les contraintes $\underline{\alpha}_{\rho,T}$ et $\overline{\alpha}_{\rho,\sigma}$ et les points critiques courants, $\underline{P}$ et $\overline{P}$, traitent pour tout $n \in \mathbb{N}$, le n$^{\text{ième}}$ message d'un flux, $P = (x, n)$.

- La fonction `EstContraintInf` retourne Vrai si P est contraint par $\underline{\alpha}_{\rho,T}$;
- La fonction `EstContraintSup` retourne Vrai si P est contraint par $\overline{\alpha}_{\rho,\sigma}$;
- La fonction `MàJCritiques` met à jour les points critiques si P définit une contrainte supérieure ou inférieure plus forte que $\underline{P}$ ou $\overline{P}$. Rappelons que T et σ n'ont pas d'influence sur la définition des points critiques (définition 2.2.6) et par conséquent ne font pas partie des paramètres de cette fonction.

Algorithme 1 : Calculs élémentaires

Données : ρ, T, σ, $\underline{P}$, $\overline{P}$, P.

1 `EstContraintInf`$(\rho, T, \underline{P}, P) = (n \geqslant \rho(x - x_{\underline{c}} - T) + \underline{c})$

2 `EstContraintSup`$(\rho, \sigma, \overline{P}, P) = (n \leqslant \rho(x - x_{\overline{c}}) + \sigma + \overline{c})$

3 `MàJCritiques`$(\rho, \underline{P}, \overline{P}, P)$

4 **Si** $n < \rho(x - x_{\overline{c}}) + \overline{c}$ **alors** $\overline{P} \leftarrow P$;

5 **Sinon si** $n > \rho(x - x_{\underline{c}}) + \underline{c}$ **alors** $\underline{P} \leftarrow P$;

2.3.2 Étude fine du comportement d'un flux

L'algorithme 2 effectue un calcul de pentes successives pour modéliser les variations de vitesse d'arrivée des messages du flux (*FluxEntrée*) observé, où $\neg$ est l'opérateur unitaire de négation. Les paramètres T et σ sont fixés par l'utilisateur en fonction de la finesse de l'analyse souhaitée. Ils peuvent également être choisis en fonction des connaissances théoriques du flux observé. À chaque nouvelle arrivée de message, la boucle (lignes 22 à 25) est exécutée. Les fonctions de l'algorithme 1 sont utilisées afin de suivre l'évolution du flux. Lorsqu'un message brise une des contraintes, une nouvelle pente est calculée. Dans l'algorithme, $P = (x, n)$ représente le message actuellement traité et $P_p = (x_p, n_p)$ est le message précédent P.

Les lignes 17 à 20 réalisent l'initialisation de l'algorithme. Puis les lignes 22 à 25 représentent les étapes de calcul réalisées à l'arrivée de chaque messages.

La ligne 17 affecte P des coordonnées du premier message. Ensuite, la première pente est calculée (ligne 18), la valeur 2 est affectée à la variable n correspondant au nombre de messages analysés plus un, afin de préparer les coordonnées du prochain point (ligne 19) et P_p est initialisé à P en attendant l'arrivée du message suivant.

Les coordonnées du point P à analyser sont récupérées à la ligne 22.

La ligne 23 appelle la fonction `MàJPente` (définie entre les lignes 1 à 15) qui détermine si P satisfait les contraintes couramment définies (lignes 2 et 9). Si ce n'est pas le cas, une nouvelle pente, ρ, est calculée. Ce calcul dépend de la contrainte brisée. S'il s'agit de la contrainte inférieure, le calcul s'effectue entre P et $\overline{P}$ (lignes 3 et 4). S'il s'agit de la contrainte supérieure, le calcul est réalisé à partir de P et $\underline{P}$ (lignes 10 et 11). Aux lignes 6 et 7 (ou 13 et 14) la fonction détermine si P satisfait les nouvelles contraintes à partir de P_p. Si ce n'est pas le cas, une nouvelle pente est calculée entre ces points (ce choix sera discuté dans la section 2.3.5). Aux lignes 8 et 15, la nouvelle pente est ajoutée à la liste de celles précédemment calculées avec la date, x, à partir de laquelle la contrainte est

Algorithme 2 : Calcul de pentes

Données : $FluxEntrée$, T, σ.
Résultats : $ListePentes$, $FluxSortie$.

```
1  MàJPente(T, σ, ρ, P̄, P̲, P_p, P)
2  Si ¬EstContraintInf(T, ρ, P, P̲) alors
3      ρ ← (n − c̄)/(x − x_c̄);
4      P̲ ← P ; P̄ ← P ;
5      FluxSortie ← FluxSortie :: P;
6      Si ¬EstContraintInf(T, ρ, P, P_p) alors
7          ρ ← (n − n_p)/(x − x_p) ;
8      mémoriser(ListePentes, (ρ, x)) ;
9  Sinon si ¬EstContraintSup(σ, ρ, P, P̄) alors
10     ρ ← (n − c̲)/(x − x_c̲) ;
11     P̲ ← P ; P̄ ← P;
12     FluxSortie ← FluxSortie :: P;
13     Si ¬EstContraintSup(σ, ρ, P, P_p) alors
14         ρ ← (n − n_p)/(x − x_p) ;
15     mémoriser(ListePentes, (ρ, x)) ;
16 Début
17     P ← (dateArrivée(FluxEntrée), 1) ;
18     ρ ← 1/x;
19     n ← 2 ;
20     P_p ← P;
21     Tant que vrai faire
22         P ← (dateArrivée(FluxEntrée), n);
23         MàJPentes(T, σ, ρ, P̄, P̲, P_p, P);
24         MàJCritiques(ρ, P̄, P̲, P);
25         P_p ← P ; n ← n + 1;
26 fin
```

instaurée. Le flux $FluxSortie$ est un sous-flux de $FluxEntrée$. Il contient l'ensemble des messages sortants calculés. Il est construit au fur et à mesure de l'étude de $FluxEntrée$, lignes 5 et 12.

À la ligne 24 la fonction `MàJCritiques` (algorithme 1), qui met à jour les points $\underline{P}$ et $\overline{P}$, est appelée. Finalement, la ligne 25 incrémente le nombre n de messages étudiés et le point P_p est affecté à P.

Cet algorithme dispose d'un avantage très intéressant pour l'étude du comportement d'un flux. En effet, il permet de surveiller avec précision l'évolution des arrivées de messages dans le flux. Par conséquent, il permet de détecter des éventuelles anomalies cachées. Qui plus est, cette méthode calcule en temps constant et mémorise uniquement les pentes calculées, accompagnées des dates de calculs, et d'un petit ensemble de points. Ceci permet de respecter les contraintes industrielles fixées.

Notons que si un flux est $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$-contraint et que l'algorithme calcule ρ avec des paramètres au moins égaux à T et σ, alors on dit que l'algorithme a convergé. Alors, tous les messages suivants respecteront les contraintes et aucune nouvelle pente ne sera calculée.

Exemple 2.3.1 (Application de l'algorithme 2). *Considérons un flux (f_p) périodique de*

période 180 s (9 messages). Durant une période, les messages arrivent aux dates suivantes : $x_1 = 15\,s$, $x_2 = 35\,s$, $x_3 = 55\,s$, $x_4 = 80\,s$, $x_5 = 100\,s$, $x_6 = 120\,s$, $x_7 = 140\,s$, $x_8 = 160\,s$, $x_9 = 180\,s$. *Ce flux est contraint par* $\underline{\alpha}_{0.05,10}$ *et* $\overline{\alpha}_{0.05,1}$. *Notre objectif est de trouver* $\rho = 0.05$ *avec notre algorithme.*

La figure 9 représente les 12 premiers messages de ce flux ainsi que les pentes calculées par l'algorithme 2 utilisant les paramètres $T = 10$ *et* $\sigma = 1$. *La première pente calculée est* $\rho_1 = \frac{1}{x_1} = 0.06667$. *Le point* P_4 *est le premier point sortant* ($P_s^{(1)}$). *Il brise la contrainte* $\underline{\alpha}_{\rho_1,T}$. *Une nouvelle pente est alors évaluée entre* P_4 *et* $\overline{P}^{(1)} = P_3$. *Ceci donne* $\rho_2 = 0.04$. *Le point sortant suivant est* $P_s^{(2)} = P_{10}$. *Il franchit la contrainte supérieure. À cet instant le point critique inférieur est* $\underline{P}^{(2)} = P_9$. *La nouvelle pente est donc* $\rho = \dfrac{1}{195 - 180} = \rho_1$. *Comme le nouveau point sortant correspond, à une période près, au premier point sortant cela indique un comportement périodique. L'algorithme oscille entre le calcul des pentes* ρ_1 *et* ρ_2.

La figure 10 représente le comportement de l'algorithme lorsque le paramètre T *varie. Si* $T = 20$ *(courbes rouges à gauche), le point sortant est* $P_s = P_5$ *qui brise la contrainte inférieure. La nouvelle pente est calculée par la vitesse d'arrivée entre* P_5 *et* $\overline{P} = P_4$ *ce qui retourne* ρ. *L'algorithme converge. Cependant, si* $T = 60$ *(courbes bleues à droite), le point sortant* $P_s = P_{13}$ *brise la contrainte inférieure et la nouvelle pente est calculée entre* P_{13} *et* $\underline{P} = P_{12}$. *Ce cas est équivalent, à une période près, au premier exemple. L'algorithme oscille à nouveau.*

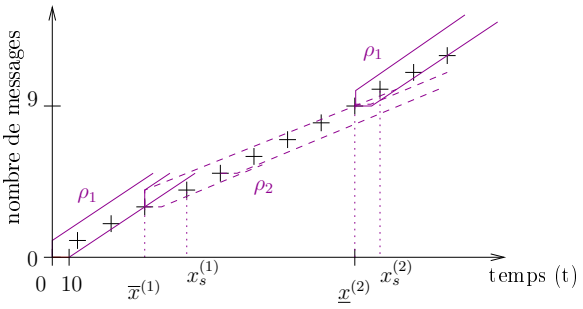


FIGURE 9 – Pentes calculées par l'algorithme 2 avec f_p, $T = 10$, $\sigma = 1$.

Vu les résultats obtenus dans l'exemple 2.3.1, nous aimerions déterminer les conditions assurant la convergence de l'algorithme sur les flux périodiques. Ceci fait l'objet du travail présenté dans le paragraphe suivant.

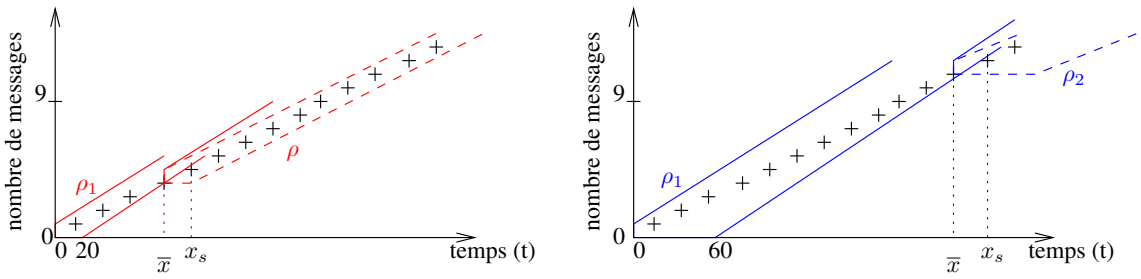


FIGURE 10 – Pentes calculées par l'algorithme 2 avec f_p, $T = 20$ et $T = 60$ ($\sigma = 1$).

2.3.3 Quelques cas d'étude

Étudions le comportement théorique de notre algorithme. Dans un premier temps, nous examinons le cas des flux périodiques quelconques puis nous étudions le cas particulier des flux équilibrés.

Flux périodiques

Définition 2.3.2 (Flux périodique). *Un flux* $(x_n)_{n\in\mathbb{N}}$ *est dit* N_p-périodique *si* $\forall n \in \mathbb{N}$

$$x_{(n+N_p)} - x_{(n+1+N_p)} = x_n - x_{n+1}.$$

La proposition suivante donne un résultat trivial sur les contraintes suivies par un flux périodique.

Proposition 2.3.3. *Soit* $(x_n)_{n\in\mathbb{N}}$ *un flux* N_p*-périodique. Il existe des valeurs* $\rho, T, \sigma \in \mathbb{R}_+$ *telles que* (x_n) *est* $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$*-contraint.*

Démonstration. Choisir les valeurs suivantes conduit immédiatement au résultat $\sigma = N_p$, $T = x_{N_p}$, et $\rho = x_{N_p}/N_p$. □

Étudions maintenant le comportement de notre algorithme sur ce type de flux.

Proposition 2.3.4. *Soit* $\rho, T, \sigma \in \mathbb{R}_+$. *Si* (x_n) *est* N_p*-périodique et* $(\underline{\alpha}_{\rho,T}, \overline{\alpha}_{\rho,\sigma})$*-contraint, alors l'application de l'algorithme 2 avec les paramètres* $((x_n), T, \sigma)$ *soit converge vers la pente* ρ *en un temps fini (et cette pente ne sera plus recalculée), soit suit un comportement périodique.*

Démonstration. D'après le lemme 2.2.4, s'il existe un instant à partir duquel l'algorithme ne calcule plus de pente, alors la pente ρ a été trouvée. Sinon, il existe toujours un message à venir qui rompt les contraintes actuelles.

Supposons que le taux d'arrivée moyen soit ρ et qu'il existe un instant tel que l'algorithme calcule une pente $\rho' < \rho$. Alors, le point critique inférieur sera mis à jour au moins une fois par période. En effet, au minimum le point critique inférieur actuel est mis à jour lors de l'arrivée du même message dans la période suivante. La figure 11 représente ce principe. Ainsi, le prochain calcul de pente sera $\frac{n-m}{x_n - x_m}$ avec $n - m < N_p$. La situation est symétrique avec $\rho' > \rho$.

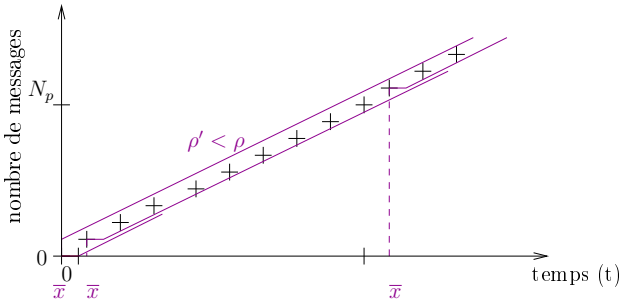


FIGURE 11 – Mise à jour du point critique inférieur pour un flux N_p-périodique.

En conséquence, le nombre de pentes calculées est fini et il existe $k, m, n \in \mathbb{N}$ tel que les paires de messages entrainant une mise à jour des pentes sont : (P_m, P_n) et (P_{m+kN_p}, P_{n+kN_p}), ce qui prouve un comportement périodique. □

L'exemple 2.3.1 illustre la précédente proposition. En effet, nous avons montré qu'utiliser l'algorithme 2, avec f_p (le flux périodique introduit dans l'exemple 2.3.1), $T = 10$ ou $T = 60$, sur le flux donné implique un comportement périodique où l'algorithme oscille entre les pentes $\rho_1 < \rho$ et $\rho_2 > \rho$. Cependant, fixer $T = 20$ entraine une rapide convergence des calculs.

Bien que la convergence ne soit pas assurée pour tout $T, \sigma \in \mathbb{R}_+$, le théorème suivant fournit une condition de convergence de l'algorithme. En effet, il montre que s'il existe un instant auquel l'algorithme 2 calcule une pente suffisamment proche de la vitesse d'arrivée moyenne des messages alors il converge en un temps fini.

Théorème 2.3.5 (Convergence). *Soit (x_n) un flux N_p-périodique, contraint par $(\underline{\alpha}_{\rho,T'}, \overline{\alpha}_{\rho,\sigma'})$. Il existe ϵ tel que si l'algorithme 2, utilisant les paramètres (x_n), $\sigma > \sigma'$ et $T > T'$, calcule une pente $\rho' \in [\rho - \epsilon, \rho + \epsilon]$, alors il converge vers la pente ρ en un temps fini.*

Démonstration. Il faut considérer les deux cas $\rho' < \rho$ et $\rho' > \rho$. Ceux-ci étant symétriques, focalisons nous sur l'étude du cas $\rho' > \rho$.

Comme $T > T'$, $\rho_1 = \rho \frac{x_{N_p} - T'}{x_{N_p} - T}$ est tel que $\forall \rho' \geqslant \rho_1$, les contraintes ne sont pas rompues durant la première période de (x_n) ($\forall t \in [0, x_{N_p}]$, $\underline{\alpha}_{\rho_1,T} \leqslant \underline{\alpha}_{\rho',T'}$).

L'ensemble $S = \{\rho x_n - n \mid n \in \mathbb{N}\}$ est fini. Soit $n_0 = \min\{n \mid \rho x_n - n = \min_{s \in S} s\}$ la plus petite valeur de l'ensemble S. Donc $n_0 \leqslant N_p$ et l'algorithme utilisant les pentes ρ' et ρ assigne le point P_{n_0} comme étant un point critique supérieur.

Concentrons nous sur les messages critiques supérieurs quand l'algorithme utilise la pente ρ'. En effet, lorsqu'il calcule la pente ρ, les points critiques ne sont plus mis à jour. Comme nous l'avons mentionné précédemment, ces points sont au moins mis à jour une fois par période et la nouvelle pente est calculée à partir de deux messages séparés au maximum d'une période. En conséquence, seulement un nombre fini de pentes peut être défini.

Prenons, $\rho_2 = \min\{\frac{x_n - x_m}{n-m} \mid 0 \leqslant m \leqslant n \leqslant N_p \text{ et } \frac{x_n - x_m}{n-m} > \rho\}$. Si $\rho \leqslant \rho' \leqslant \rho_2$, alors, lorsque le point critique supérieur est mis à jour la pente entre deux points critiques est exactement ρ. En effet, elle est forcément plus petite que ρ, mais ne peut pas être strictement inférieure à ρ par construction de S. Ceci prouve l'égalité.

D'après le lemme 2.2.9, la prochaine pente calculée est exactement ρ. Effectivement, il s'agit d'une pente définie à partir de deux points critiques supérieurs. Ainsi, trouver $\rho' \leqslant \min(\rho_1, \rho_2)$ garantit la convergence. □

Flux équilibrés Considérons un cas particulier des flux périodiques, les flux équilibrés, dont la définition est la suivante.

Définition 2.3.6 (Flux équilibrés). *Un flux (x_n) d'arrivée à long terme (pente) ρ est dit équilibré si $\forall n \in \mathbb{N}$,*

$$x_n = \lceil \frac{n}{\rho} \rceil.$$

Cette définition provient de celle des mots équilibrés où la n$^{\text{ième}}$ lettre du mot est définie à chaque intervalle de temps. Il y a un 0 si aucun message n'arrive durant un intervalle de temps et k fois le symbole 1 si k messages arrivent pendant une plage horaire. Davantage de détails sur les mots équilibrés sont présentés dans [54]. Afin d'observer des flux périodiques, on suppose que la pente est un nombre rationnel et que $\rho < 1$. En effet, si $\rho > 1$, plusieurs messages peuvent arriver simultanément.

La structure d'un mot équilibré ayant une pente rationnelle est la suivante : $\exists e_1, \ldots, e_k > 0$ tels que pour tout $\ell \in \{1, \ldots k-1\}$, le mot équilibré peut être décomposé suivant le couple de facteurs $w_\ell = w_{\ell-1}^{e_\ell} w'_{\ell-1}$ et $w'_\ell = w_{\ell-1}^{e_\ell - 1} w'_{\ell-1}$, avec $w_0 = 0$ et $w'_0 = 1$. Le motif périodique du mot est alors $w_k = w_{k-1}^{e_k - 1} w'_{k-1}$.

Exemple 2.3.7 (Flux équilibré). *Soit le flux équilibré (f_{eq}) représenté sur la figure 12. Les facteurs initiaux sont $w_0 = 0$ et $w'_0 = 1$. Alors, les facteurs suivant sont $w_1 = 0001$, $w'_1 = 001$, puis $w_2 = 0001001$ ($e_2 = 1$), $w'_2 = 001$. Finalement, le motif périodique est $w_3 = w'_3 = w_2w'_2 = 0001001001$ ($e_3 = 1$).*

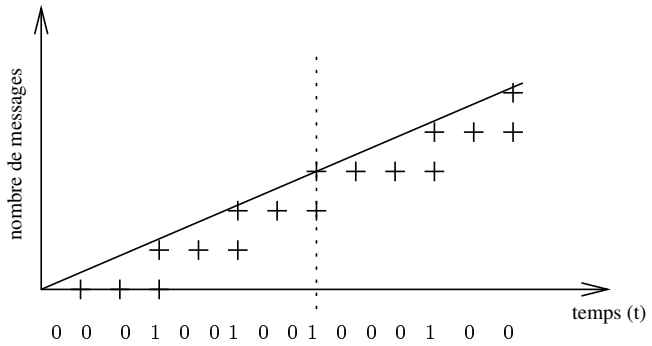


FIGURE 12 – Flux équilibré de vitesse moyenne d'arrivée $\rho = \frac{3}{7}$.

La pente d'un mot fini w est égale au nombre de 1 divisé par le nombre de 0 : $\rho(w) = \frac{|w|_1}{|w|_0}$, où $|w|_0$ (resp. $|w|_1$) est le nombre d'occurrence de 0 (resp. 1) dans le mot w. La pente d'un mot périodique est aussi la pente de ces motifs périodiques : $\rho(w^*) = \rho(w)$. Une simple récurrence montre que $\rho_\ell = \rho(w_\ell) \leqslant \rho(w'_\ell) = \rho'_\ell$.

Exemple 2.3.8 (Pentes des facteurs). *Reprenons le flux f_{eq} de pente $\rho = \frac{3}{7}$ présenté sur la figure 12. Les pentes des facteurs $w_1, w'_1, w_2, w'_2, w_3, w'_3$ sont respectivement $\rho_1 = \frac{1}{3}$, $\rho'_1 = \frac{1}{2}$, $\rho_2 = \frac{2}{5}$, $\rho'_2 = \frac{1}{2}$, et finalement $\rho_3 = \rho'_3 = \frac{3}{7}$.*

Lemme 2.3.9. *$\forall \ell$, il existe $\underline{w}$ et $\overline{w}$ tels que $w'_\ell = \underline{w}1$ et $w_\ell = \underline{w}0\overline{w}$.*

Démonstration. Le lemme est vrai pour $\ell = 0$. Supposons que le résultat est vrai pour $\ell - 1$. Comme, $w'_\ell = w_{\ell-1}^{e_\ell - 1} w'_{\ell-1}$ et $w_\ell = w_{\ell-1}^{e_\ell - 1} w_{\ell-1} w'_{\ell-1}$, alors le résultat est vrai pour ℓ. □

Lemme 2.3.10. *Soit pp un préfixe propre de w_ℓ (resp. w'_ℓ). Alors $\rho(pp) \leqslant \rho_\ell$ (resp. $\rho(pp) \leqslant \rho'_\ell$).*

Démonstration. Montrons cette inégalité par récurrence. Elle est vraie pour $\ell = 1$. Les facteurs w_ℓ et w'_ℓ s'écrivant de la manière suivante à partir de w_1 et w'_1 :

$$w_\ell = w_{\ell-1}^{e_\ell} w_{\ell-2}^{e_{\ell-1}-1} w_{\ell-3}^{e_{\ell-2}-1} \cdots w_1^{e_2-1} w'_1 \text{ et } w'_\ell = w_{\ell-1}^{e_\ell-1} w_{\ell-2}^{e_{\ell-1}-1} w_{\ell-3}^{e_{\ell-2}-1} \cdots w_1^{e_2-1} w'_1,$$

la propriété est également vérifiée pour un rang quelconque ℓ. □

Théorème 2.3.11. *Soient $\rho > 0$, $\sigma > \lceil \rho \rceil$ et $T \geqslant \lceil 1/\rho \rceil$. Si (x_n) est un flux équilibré périodique de pente ρ, alors l'algorithme 2 utilisant les paramètres σ et T converge vers ρ en un temps fini.*

Démonstration. Montrons par récurrence sur ℓ que le motif w_ℓ et/ou w'_ℓ correspond à la pente calculée par l'algorithme.

Initialisation : la première pente calculée est soit $\rho'_1 = \rho(w'_1)$ soit $\rho_1 = \rho(w_1)$.

Hypothèse de récurrence : la pente ρ_ℓ ou ρ'_ℓ est calculée au bout d'un temps fini.

Hérédité : afin de prouver ce résultat, nous utilisons le lemme 2.2.9. Si $\ell \neq k$, alors $\rho_\ell < \rho < \rho'_\ell$.

(1) Supposons que la pente qui a été calculée est ρ_ℓ. Alors, la contrainte supérieure est rompue par un point critique inférieur.

Le mot peut être décomposé en deux facteurs w_ℓ et w'_ℓ. Tant que les facteurs de w_ℓ sont trouvés, la contrainte supérieure est vérifiée et le point critique inférieur correspond au message avant la fin de la première occurrence de w_ℓ. Remarquons que cette première occurrence peut être un suffixe de w_ℓ. Comme $w_\ell = w_{\ell-1}w'_\ell$, ceci est aussi valide pour w'_ℓ.

La structure de w_ℓ et w'_ℓ montre que seulement le dernier message de w'_ℓ peut être critique. De plus, vu que $\sigma > \lceil\rho\rceil$, la contrainte ne peut être rompue par la première occurrence de w'_ℓ. Ainsi, la prochaine pente calculée correspond à la pente entre deux occurrences de w'_ℓ, et peut alors prendre les valeurs $\rho_{\ell+1}$ ou $\rho'_{\ell+1}$.

(2) Les mêmes arguments peuvent être utilisés si l'on suppose que l'algorithme obtient la pente ρ'_ℓ. La contrainte inférieure sera rompue et seul le premier message de w_ℓ peut être critique. En effet, la pente calculée étant $\rho'_\ell = \rho(w'_\ell)$, les messages de w'_ℓ ne peuvent être critiques. De même, ayant l'égalité $w_\ell = w_{\ell-1}w'_\ell$, le message critique ne peut provenir que de $w_{\ell-1} = w_{\ell-2}w'_{\ell-1}$. Or, $\rho'_{\ell-1} \geqslant \rho'_\ell$ et nous savons que la contrainte inférieure sera rompue. Donc le message critique supérieur provient de $w_{\ell-2}$. En appliquant récursivement l'argument précédent, le message critique ne peut qu'être le premier message de w_ℓ.

Comme $T \geqslant \lceil 1/\rho\rceil$, la contrainte ne peut être rompue par la première occurrence de w_ℓ. La prochaine pente est alors calculée entre deux occurrences de w_ℓ, et prend ainsi les valeurs $\rho_{\ell+1}$ ou $\rho'_{\ell+1}$. □

Exemple 2.3.12. *La figure 13 représente les pentes calculées par l'algorithme 2, appliqué à f_{eq} (de pente $\rho = \frac{3}{7}$ et représenté sur la figure 12), lorsque $T = 3$ et $\sigma = 1$. La première pente calculée est $\rho_1 = 1/x_1 = \frac{1}{3}$. Alors, P_6, arrivant à la date $x_s^{(1)}$, est le point sortant qui brise la contrainte supérieure. Le point critique inférieur associé à P_6 pour déterminer une nouvelle pente est $\underline{P}^{(1)} = P_5$, et alors la nouvelle pente est $\rho'_2 = 1/(x_6 - x_5) = \frac{1}{2}$. Le point sortant suivant est $P_s^{(2)} = P_{16}$ et son point critique supérieur est $\overline{P}^{(2)} = P_{13}$. Ceci permet le calcul de la pente $\rho_3 = 1/(x_{16} - x_{13}) = \frac{3}{7}$. Ainsi, la convergence est obtenue au bout de 38 s (temps correspondant à l'arrivée de 16 messages). Remarquons que les pentes calculées font toutes partie de l'ensemble des pentes des facteurs calculés dans l'exemple 2.3.8, comme énoncé dans le théorème 2.3.11.*

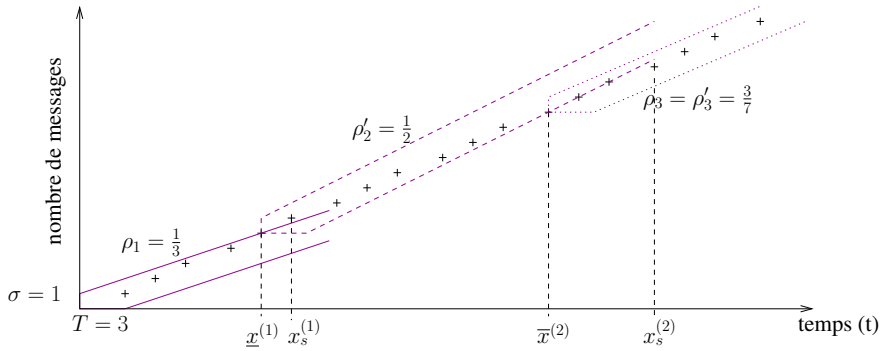


FIGURE 13 – Pentes calculées par l'algorithme 2 avec f_{eq}, $T = 3$ et $\sigma = 1$.

2.3.4 Étude de comportement à long terme d'un flux

Si l'algorithme 2 fournit une observation fine du comportement d'un flux, il sera cependant utile d'allier à cette étude précise une analyse procurant la tendance générale du flux. C'est le rôle de l'algorithme présenté dans ce paragraphe.

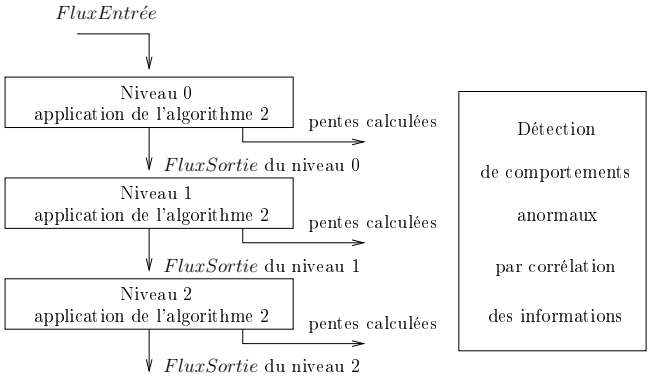


FIGURE 14 – Principe de l'algorithme à plusieurs niveaux.

La technique de calcul de comportement à long terme repose sur l'algorithme 2. L'invention consiste à développer une version à plusieurs niveaux de ce dernier. Chaque niveau fournit une vision du comportement du flux à plus long terme que le niveau précédent. Dans ce but, chaque niveau applique l'algorithme 2 sur le sous-flux calculé par le niveau inférieur (sauf pour le premier qui utilise $FluxEntrée$). L'utilisation de l'algorithme à plusieurs niveaux nécessite quatre types de paramètres : $FluxEntrée$, T et σ pour chaque niveau, et N_{niv} le nombre de niveaux désirés. La figure 14 représente le principe de cette méthode avec trois niveaux de calcul.

L'utilisation de cet algorithme présente l'intérêt de pouvoir observer un flux avec différents niveaux de détails. En effet, le niveau 0 fournit une étude fine et concise du flux alors que le niveau le plus élevé informe de la tendance générale de ce dernier. La corrélation de ces deux informations génère une meilleure analyse. La connaissance de la tendance à long terme permet de détecter aisément une déviation progressive du comportement d'un flux. Selon l'étude effectuée, cette information peut également être comparée au comportement théorique du flux observé, ce qui facilite une étude plus fiable.

Remarque 2.3.13. *Par abus de langage, nous parlerons souvent du niveau n de l'algorithme 2 pour faire référence au $n^{ième}$ niveau de la version multi-couche de celui-ci.*

Exemple 2.3.14 (Étude du comportement à long terme d'un flux)**.** *Reprenons le cas du flux f_p présenté dans l'exemple 2.3.1. La figure 15 représente les pentes calculées par l'algorithme à plusieurs niveaux lorsque $T = 10$ et $\sigma = 1$ pour chaque niveau et $N_{niv} = 2$. Les premiers points composants se niveau arrivent aux dates : $x_4 = 80$, $x_{10} = 195$, $x_{13} = 235$, $x_{19} = 375$, etc. La ligne temporelle supérieure représente les taux calculés par le niveau 0 et la ligne inférieure ceux obtenus par le niveau 1. Comme nous l'avons indiqué précédemment, le niveau 0 ne permet pas de converger vers la tendance générale du flux : $\rho = 0.05$. Cependant, le second niveau de calcul converge en deux étapes.*

2.3.5 Discussion sur les algorithmes

Étudions maintenant quelques points importants concernant les algorithmes : la définition des paramètres T et σ, les choix d'implémentation, et les solutions qui peuvent être apportées afin de garantir la convergence de l'algorithme.

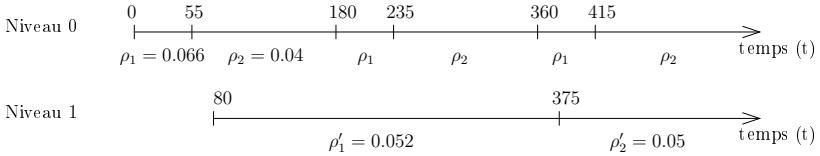


FIGURE 15 – Pentes calculées par l'algorithme à plusieurs niveaux avec f_p, $T = 10$, $\sigma = 1$ et $N_{niv} = 2$.

Définition des paramètres Dans le paragraphe précédent, nous avons expliqué que le choix des paramètres T et σ n'est pas d'une grande importance. En effet, aucune propriété évidente ne semble se dégager concernant la convergence. Dans l'exemple 2.3.1 nous avons montré qu'augmenter le paramètre T ne permet pas d'atteindre la convergence. Sur ce point, l'algorithme à plusieurs niveaux s'avère plus efficace.

Cependant, il incombe à l'utilisateur (tel qu'un administrateur réseau) de choisir ces paramètres afin de définir le niveau de tolérance acceptable pour les variations d'un flux. Ceci est particulièrement important pour le niveau 0 : le choix de petites valeurs impliquera une étude fine des variations. La connaissance théorique du flux étudié facilitera ce choix.

Finalement, la définition de ces paramètres peut être différente pour chaque niveau. Nous verrons dans la section 2.4 qu'il est utile de définir de petites valeurs pour les premiers niveaux (observations à court terme) et de plus grandes pour les suivants (observations à long terme).

Choix d'implémentations Justifions tout d'abord le choix des points critiques pour les calculs de pentes, puis discutons l'intérêt de la définition d'un second calcul de pente entre P et P_p lorsque ces derniers ne respectent pas les contraintes.

Calculs de pentes et points critiques : dans nos algorithmes, nous avons choisi de mettre à jour la pente avec le message critique supérieur, lorsque la contrainte inférieure est rompue et avec le message critique inférieur, lorsque la contrainte supérieure est rompue. Nous aurions pu définir le contraire et utiliser le point critique inférieur. Cependant, faire ce choix ne permet pas d'aboutir à la convergence en temps fini, considérant le lemme 2.2.9. Néanmoins, nous avons observé expérimentalement la convergence en temps infini en pratiquant ainsi. Cependant, au vu de l'étude réalisée, le choix d'une convergence en temps fini semble plus adéquat.

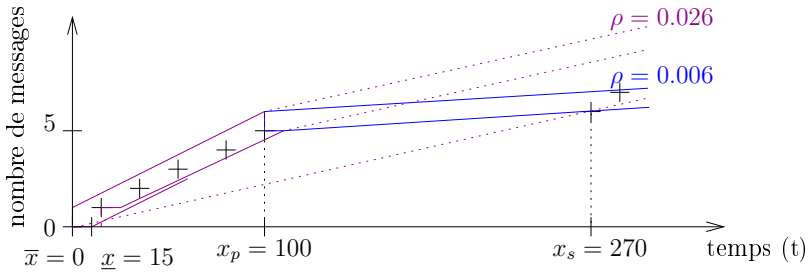


FIGURE 16 – Calcul de pente entre P et P_p.

Second calcul de pente : les lignes 6-7 et 13-14 de l'algorithme 2 présentent une petite amélioration. Elle repose sur l'idée suivante : lors du calcul d'une nouvelle pente on vérifie

que le point P, ayant rompu les contraintes, est également contraint par P_p (et pas uniquement $\underline{P}$ ou $\overline{P}$). Comme ce point représente le début d'une nouvelle tendance des arrivées de messages, faire ce calcul assure que les contraintes sont initialement respectées. D'un point de vue pratique, ce calcul se montre particulièrement utile dans le cas de messages n'arrivant plus durant un certain laps de temps (voir section 2.4). Illustrons ce principe sur le flux simple représenté sur la figure 16. Les messages arrivent en moyenne toutes les 10 s, durant 100 s. Le flux est initialement contraint par les courbes $\underline{\alpha}_{0.066,10}$ et $\overline{\alpha}_{0.066,1}$, dessinées en violet sur le graphe. Aucun message ne se présente entre les dates 100 et 270 s. Le message P_6, arrivant ensuite, rompt la contrainte inférieure : $P_s = (270, 6)$ et la pente $\rho = \frac{6-0}{x_6-x_0} = 0.026$ est alors calculée. Les nouvelles contraintes sont représentées par les courbes pointillées violettes. D'après la figure, il est évident que P_s ne satisfait pas à ces contraintes. Une seconde pente est alors définie entre $P_p = (100, 5)$ et P_s : $\rho' = 0.006$. Cette valeur est beaucoup plus faible que ρ. Nous verrons dans la section 2.4 qu'effectuer ce calcul permet de repérer aisément les périodes durant lesquelles aucun message n'est émis ainsi que l'importance de ce changement de comportement.

Convergence des algorithmes Afin d'assurer la convergence des algorithmes pour des flux quelconques, plusieurs améliorations peuvent être apportées.

- *Régression linéaire des moindres carrés* : cette méthode est très connue et définit une approximation d'un ensemble de points par une fonction linéaire. Dans notre étude, cette méthode peut s'avérer très efficace pour déterminer la pente moyenne sur le dernier niveau calculé. De plus, il s'agit d'une technique intéressante pour résoudre le problème de convergence des flux périodiques. En effet, nous n'avons pas trouvé d'exemple pour lequel l'algorithme à plusieurs niveaux ne converge pas. Cependant, étant donné un nombre de niveaux fixé il est pratiquement certain que ce phénomène se produit. Ainsi, la méthode de régression linéaire des moindres carrés permet de calculer une pente proche de ρ et de retrouver le contexte du théorème 2.3.5. Cela assure la convergence du dernier niveau.
- *Corrélation entre plusieurs niveaux* : une autre solution consiste à corréler les résultats des niveaux inférieurs et supérieurs. Ainsi, comme seul un sous-flux est considéré, même si le dernier niveau ne converge pas, les pentes calculées seront de plus en plus proches de ρ. D'après le théorème 2.3.5, fournir périodiquement ces pentes au niveau initial assure la convergence.
- *Mélange des implémentations* : l'algorithme 2 peut également assurer la convergence. En effet, il suffit simplement de faire fonctionner aléatoirement cet algorithme avec les adaptations proposées ci-dessus.

2.4 Résultats

La section précédente présente une méthode calculant des contraintes linéaires pour modéliser un flux de données. À chaque changement significatif du comportement du flux, la pente des contraintes est redéfinie. Dans cette partie, nous commençons par présenter la mise en œuvre de l'algorithme. Dans un second temps, nous testons ce dernier sur un ensemble de cas. Nous présentons tout d'abord l'analyse d'un flux simulé subissant une dégradation progressive. Ensuite, nous introduisons l'application de l'algorithme sur des flux OSPF émulés. L'observation des variations des pentes permet la détection aisée d'erreurs cachées dans les réseaux.

Toutes les applications de l'algorithme 2 utilisent 4 niveaux d'expertise avec $T = 10$, à chaque niveau, $\sigma = 1$ pour les niveaux 0 et 1 et $\sigma = 5$ pour les niveaux 2 et 3.

2.4.1 Mise en œuvre

Les résultats présentés dans cette section sont obtenus à partir d'un programme réalisé en Java. Il se compose de deux parties : la première réalise les calculs de l'algorithme 2 à plusieurs niveaux d'abstraction ; la seconde gère l'affichage des résultats.

L'interface graphique GUI (de l'anglais *Graphic User Interface*) est représentée sur la figure 17. Elle se compose de trois parties. La topologie du réseau est dessinée en haut à gauche, sa légende figurant à sa droite. Le bas de l'interface représente en temps-réel un flux de référence, le flux analysé, ainsi que les résultats de deux niveaux de l'algorithme.

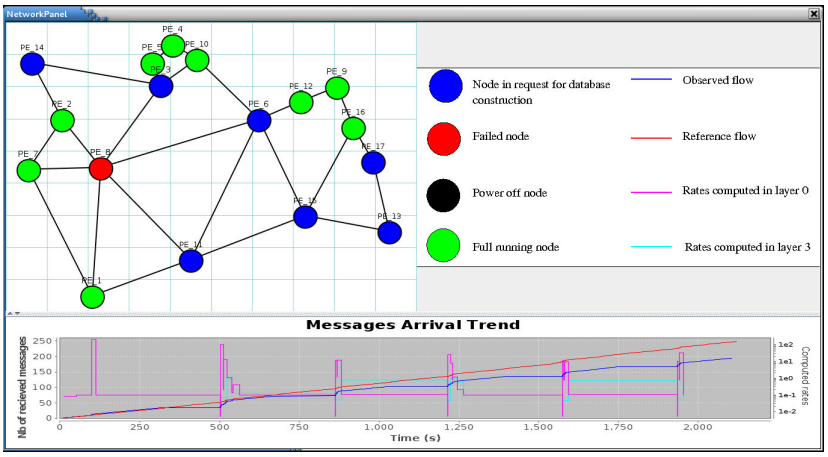


FIGURE 17 – Interface graphique.

Les éléments du réseau sont représentés par des cercles colorés et étiquetés. Ainsi le i$^{\text{ième}}$ élément est étiqueté PE_i, où PE signifie élément du panneau (*Panel Element* en anglais). Quatre couleurs représentent les différents états des éléments. Ces couleurs changent en temps-réel en fonction des évènements apparaissant dans le réseau.

- Un rond bleu indique qu'un élément effectue une synchronisation de sa base de données (échanges de LSAs). Cette couleur est présentée dans la légende par *Node in request for database construction* ;
- Un rond rouge informe qu'un élément est tombé en panne (*Failed node* dans la légende) ;
- Un rond noir représente un élément éteint (*Power off node*) ;
- Un rond vert montre qu'un élément est dans son état normal de fonctionnement où il échange périodiquement des messages Hello (*Full running node*).

Les liens entre les éléments sont simplement représentés par des traits pleins reliant un élément à un autre. Le bas de la figure représente quatre courbes :

- La courbe bleue affiche le flux actuellement étudié (appelé *Observed node* dans la légende) ;
- La courbe rouge affiche le flux de référence. Il s'agit d'un flux ne présentant pas d'anomalie (*Reference node*) ;
- La courbe magenta présente les pentes calculées par le niveau 0 de l'algorithme 2 (*Rates computed in Layer 0*) ;
- La courbe turquoise dessine les pentes calculées par le niveau 3 de l'algorithme 2 (*Rates computed in Layer 3*).

L'axe des abscisses de ces courbes est le temps en seconde (*Time (s)* sur la figure). L'axe des ordonnées pour les flux (courbes bleu et rouge) dispose d'une échelle linéaire qui compte le nombre de messages arrivés. Il est représenté à gauche du graphique et est

étiqueté par *Nb received messages*. L'axe des ordonnées pour les résultats de l'algorithme (courbes turquoise et magenta) utilise une échelle logarithmique. Il est représenté à droite du graphique et est étiqueté par *Computed rates*.

Cet outil, composé de quelques milliers de lignes de code, résulte d'une collaboration avec Benoît Ronot et fut utilisé tout au long de l'analyse de l'algorithme et notamment lors de sa présentation aux *Open Days* d'Alcatel-Lucent Bell-labs en mai 2012.

2.4.2 Application à la déviation de comportement

Étudions maintenant les résultats de l'algorithme 2 à plusieurs niveaux sur un flux transmettant des messages de plus en plus lentement. Ceci correspond au comportement classique d'un élément subissant une surcharge progressive, devenant de fait incapable d'envoyer les messages à la cadence initiale.

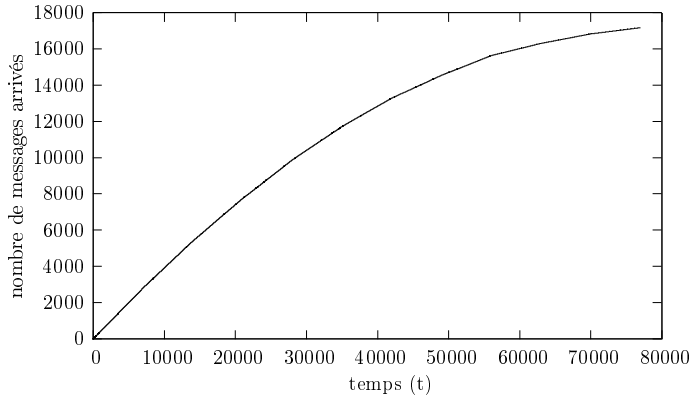


FIGURE 18 – Simulation du flux (f_ℓ) de messages arrivant de plus en plus lentement.

Ce phénomène courant n'est détecté par les méthodes de management qu'à partir du moment où le temps entre deux arrivées est supérieur à une borne critique définie par le protocole concerné. Cette détection est trop tardive car la surcharge risque d'être déjà trop importante pour qu'une intervention puisse avoir lieu avant l'occurrence d'une défaillance. De plus, les méthodes actuelles ne permettent pas d'observer la déviation progressive qui constitue une indication importante.

La figure 18 représente les arrivées de messages d'un tel flux, f_ℓ, durant approximativement une journée (21 heures). Le flux est simulé de sorte que le temps entre deux arrivées soit progressivement augmenté. Initialement, il est aléatoirement défini dans l'intervalle [1.6, 2.4] puis augmente jusqu'à être choisi dans [16, 24]. À chaque instant, le temps d'inter-arrivées est établi par une variable aléatoire suivant la loi uniforme (sur l'intervalle courant) afin de pouvoir simuler des petites variations dues à la vie courante du réseau.

La figure 19 représente les résultats du niveau 0 de notre algorithme sur le flux étudié. On remarque que les pentes calculées présentent de nombreuses variations qui cachent légèrement l'observation de la déviation progressive.

La figure 20 introduit les résultats du niveau 3 de l'algorithme. Cette analyse est très intéressante. En effet, elle permet de pratiquement supprimer toutes les fluctuations dues à la vie courante du réseau et l'observation de la déviation du comportement devient flagrante. De plus, les dates de changements d'intervalle définissant les temps d'inter-arrivées sont évidents et de nombreuses périodes sont représentées par une unique pente, correspondant à la vitesse moyenne d'arrivée des messages.

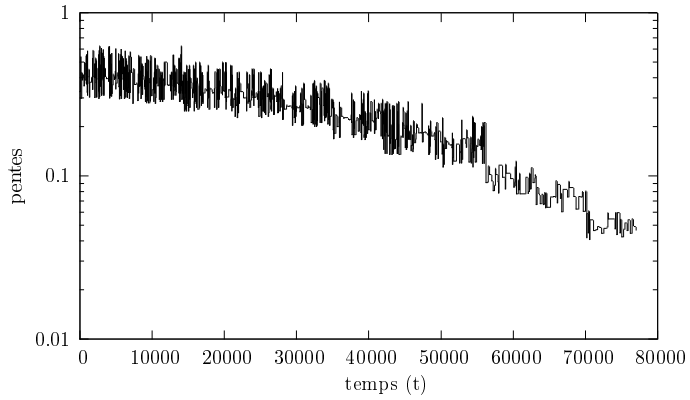


FIGURE 19 – Résultats du niveau 0 de l'algorithme 2 sur le flux f_ℓ dont les arrivées sont de plus en plus lentes.

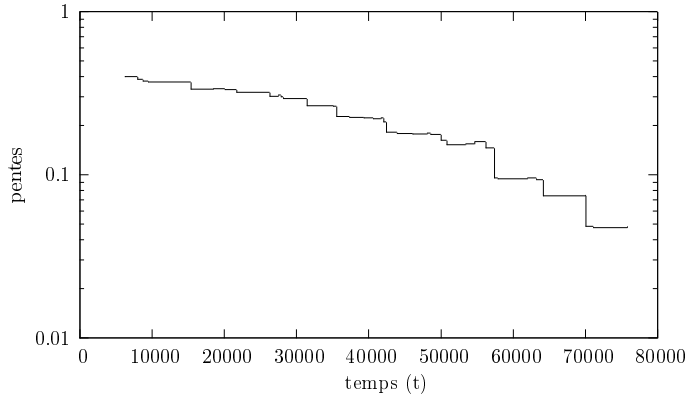


FIGURE 20 – Résultats du niveau 3 de l'algorithme 2 sur le flux f_ℓ dont les arrivées sont de plus en plus lentes.

Par conséquent, notre algorithme est très efficace pour mettre en avant les déviations de comportements d'un flux, même lorsqu'un bruit important est présent.

2.4.3 Application au protocole OSPF

Dans cette section, nous appliquons l'algorithme 2 à plusieurs niveaux sur des flux émulés du protocole OSPF provenant du réseau de télécommunication, de 17 routeurs, des grandes villes allemandes. Ce réseau est représenté sur la figure 4, page 19.

Nous analysons plusieurs scenarii définis sur ce réseau. Nous commençons par l'étude d'un réseau normal et stable doté d'un trafic OSPF fluide. Cette étude nous servira de référence pour les analyses suivantes. Alors, nous étudions le comportement malicieux d'un routeur tombant en panne périodiquement. Finalement, nous effectuons une étude du réseau lorsque les routeurs s'ajoutent progressivement, entrainant des problèmes de convergence du protocole OSPF.

Pour chaque scénario, nous surveillons tous les échanges de messages circulant dans la plateforme d'essai et nous montrons que les résultats prouvent l'intérêt de la méthode exposée. Les flux les plus révélateurs sont présentés dans les paragraphes suivants.

Dans l'exemple de référence, les figures présentant les résultats de l'algorithme introduisent également les résultats de la méthode de moyenne glissante : MAM (*Moving Average Method*). Il s'agit d'une méthode d'analyse statistique sur une série de données.

Son objectif, tout comme celui de notre algorithme, consiste à supprimer les fluctuations transitoires afin de conserver uniquement la tendance générale de la série observée. Une fois l'étude du scenario normal achevée avec notre algorithme, nous comparons les résultats obtenus à ceux rendus par la méthode MAM.

Trafic fluide

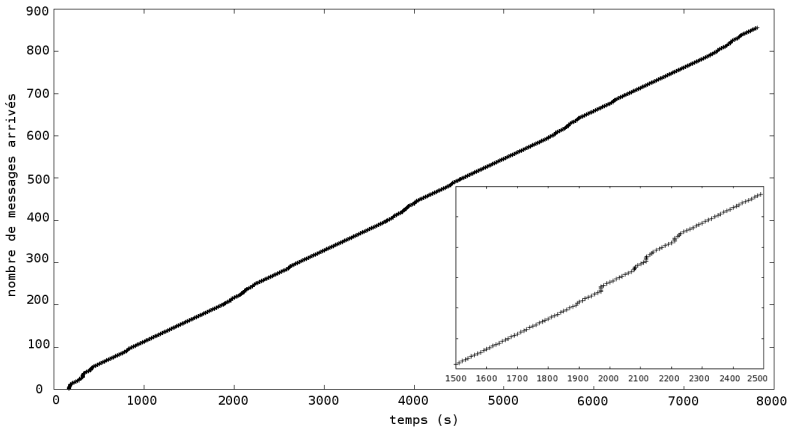


FIGURE 21 – Flux $f^{(1)}_{11\to 8}$ pendant 8000 s avec une période détaillée (entre 1500 et 2500 s) en bas à droite.

Étudions un flux OSPF dans un réseau ne présentant pas de perturbations. Le flot étudié ($f^{(1)}_{11\to 8}$) contient l'ensemble des messages transmis par R_{11} à R_8. La figure 21 représente les arrivées de messages de ce flux entre les instants 0 et 8000 s. L'encadré en bas à droite constitue un agrandissement des arrivées de messages, durant un peu plus d'une période, entre les instants 1500 s et 2500 s. À première vue le flux semble globalement linéaire. Ceci est dû aux arrivées des messages Hello toutes les 10 s. Cependant, une observation approfondie (période détaillée) permet d'observer des perturbations dans ce comportement linéaire (entre les instants 1900 et 2250 s). En effet, durant ce laps de temps l'arrivée des messages s'intensifie. Ce motif se repère aussi très légèrement sur le flux total. Ceci correspond aux opérations de mise à jour périodiques grâce aux messages LSAs. L'objectif est de mettre en évidence ces deux tendances par l'application de l'algorithme 2 en utilisant plusieurs niveaux de détails.

La courbe composée d'un trait continu sur la figure 22 représente les pentes calculées par le niveau 0 de notre algorithme sur le flux $f^{(1)}_{11\to 8}$. Cette courbe affiche très nettement un schéma périodique. En effet, les pentes exposent deux types de comportements :

- cas *stable* : un maximum de trois pentes est calculé durant 1500 s. Leurs valeurs sont très faibles, ne dépassant pas 1 message par seconde ;
- cas *perturbé* : durant un maximum de 400 s, environ 4 pentes sont calculées. Elles sont en général beaucoup plus élevées que dans le cas stable et varient entre 0.5 et 1000 messages par seconde.

Les résultats du niveau 3 de l'algorithme sont représentés par la courbe en trait plein de la figure 23. Une première observation permet de vérifier la présence des deux comportements. Cependant, le niveau 3 offre un niveau d'abstraction supérieur étant donné que chaque comportement est modélisé par une pente unique, référençant la vitesse moyenne d'arrivée des messages durant la période. Lorsque seuls les messages Hello sont transmis la pente définie est légèrement supérieure à 0.11. Puis, lorsque Hello et LSAs

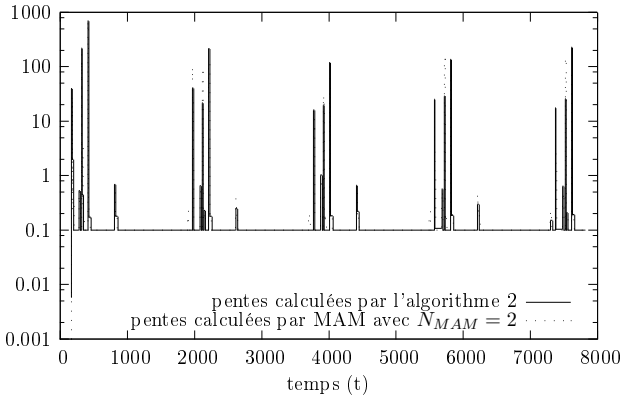


FIGURE 22 – Résultats du niveau 0 de l'algorithme 2 et de la méthode MAM de paramètre $N_{MAM} = 2$, sur le flux $f^{(1)}_{11\to 8}$ lors d'un trafic fluide.

sont transmis le taux d'arrivé moyen passe à 0.14. Une observation plus approfondie permet la détection d'une troisième pente définie durant quelques secondes entre le cas stable et le cas perturbé. Il s'agit d'un calcul de transition négligeable.

On peut remarquer la légère différence entre les premières pentes calculées et le reste de l'étude. Ceci est normal et correspond à l'initialisation du réseau.

En conclusion, l'algorithme 2 met en avant les deux comportements attendus d'après la connaissance théorique du protocole OSPF : le maintien des connexions entre routeurs par la réception de messages Hello, et le rafraîchissement des connaissances du réseau par l'envoi de messages LSAs (et Hello). De plus, notre algorithme représente chaque tendance par une pente, correspondant au taux d'arrivé moyen de chaque période. Ceci est très intéressant car une dégradation du comportement du réseau pourra ainsi être très facilement observable.

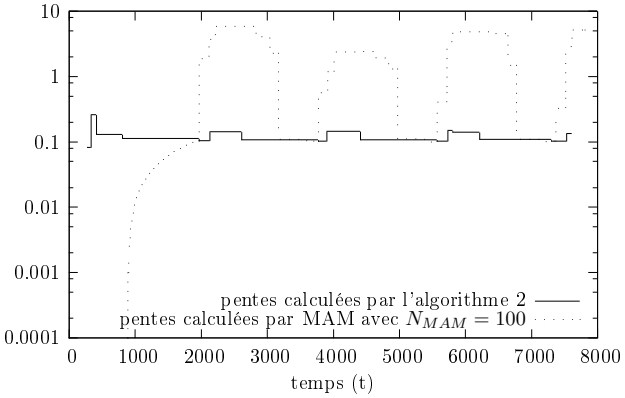


FIGURE 23 – Résultats du niveau 3 de l'algorithme 2 et de la méthode MAM de paramètre $N_{MAM} = 100$, sur le flux $f^{(1)}_{11\to 8}$ lors d'un trafic fluide.

Comparaison avec la méthode de moyenne glissante

Comparons maintenant notre algorithme avec la méthode de moyenne glissante MAM. Son principe consiste à analyser un flux de données afin de supprimer les fluctuations instantanées et d'extraire le comportement à long terme. Cette méthode utilise un paramètre, N_{MAM}, qui définit le nombre d'éléments à prendre en compte pour le calcul d'une moyenne. Afin de pouvoir comparer les résultats de MAM à ceux de notre algorithme, le

n$^{\text{ième}}$ élément étudié est $\frac{1}{x_n - x_{n-1}}$, ce qui correspond à l'inverse du temps d'attente entre l'arrivée des messages $P_{(n-1)}$ et P_n. De plus, MAM est appliquée sur le flux de messages avec le paramètre N_{MAM} fournissant les résultats les plus proches de ceux des niveaux 0 et 3.

La figure 22 présente les résultats du niveau 0 de notre algorithme (tracé plein) et ceux de MAM avec $N_{MAM} = 2$ (tracé en pointillé). On remarque que les résultats sont très proches, si bien que les pointillés sont souvent confondus avec le trait plein. La figure 23 introduit les résultats du niveau 3 de notre algorithme (tracé plein) et ceux de MAM avec $N_{MAM} = 100$ (tracé en pointillé). Ici, les résultats sont très différents. Le point commun entre ces résultats est qu'ils décrivent tous deux, deux mêmes comportements. Cependant, MAM est beaucoup moins précise. En effet, lorsque les LSAs sont émis, le changement de comportement indiqué par MAM dure trop longtemps (environ 1000 s) et, les moyennes calculées ne convergent pas vers le taux moyen d'arrivée des messages. De plus, le changement lent de comportement semble montrer que le temps d'émission des LSAs est plus long que celui de réception des Hello uniquement, ce qui n'est pas le cas. Par ailleurs, notre algorithme fournit les informations plus succinctement : seuls les dates de changements de comportements, et les pentes associées sont communiquées, alors que MAM fournit une valeur à chaque arrivée de messages. Ceci prouve l'intérêt de notre algorithme pour détecter les variations de comportements et sa concision quant aux informations transmises ce qui permet une étude plus rapide des flux en utilisant un espace mémoire bien plus restreint.

Finalement, on peut noter que les résultats de MAM dépendent fortement de la valeur choisie pour le paramètre N_{MAM}. Comme nous en avons précédemment discuté dans le paragraphe 2.3.5, ce n'est pas le cas de notre algorithme.

D'autres méthodes, utilisant le principe de la moyenne glissante telle que la moyenne glissante exponentielle, ne fournissent pas de meilleurs résultats.

Panne récurrente d'un routeur

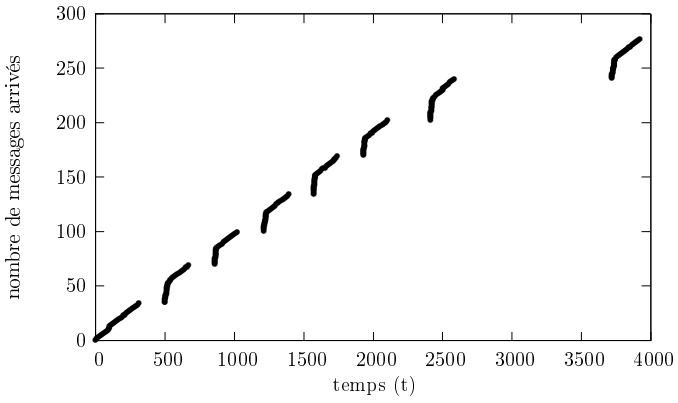


FIGURE 24 – Flux $f^{(2)}_{8\to 11}$ pendant 4000 secondes.

Présentons maintenant le cas d'un flux de messages OSPF où le routeur R_8 tombe en panne toutes les 6 minutes, durant 3 minutes. Ceci peut être assimilé à un défaut d'implémentation forçant le routeur à redémarrer fréquemment. Nous étudions le flux $f^{(2)}_{8\to 11}$ correspondant aux messages envoyés par R_8 à R_{11}. La figure 24 représente ce flux entre les instants 0 et 4000 s. Remarquons qu'au début de l'observation, les messages arrivent régulièrement. Ceci correspond aux arrivées des messages Hello. Ensuite, à la date 310 s et pendant 180 s plus aucun message n'arrive. Ceci représente la période de temps

durant laquelle le routeur R_8 ne peut plus envoyer de messages car il redémarre. Après quoi, à l'instant 500 s, la réception des messages recommence. La fréquence d'arrivée, tout d'abord plus intense, reprend ensuite le rythme précédent. Ces soudaines arrivées de messages correspondent aux envois de LSAs. Le routeur R_8 étant tombé en panne, dès qu'il est à nouveau opérationnel, il instaure une période de rafraîchissement afin de connaître les changements éventuels du réseau. Ensuite, les Hello sont échangés normalement jusqu'à la panne suivante.

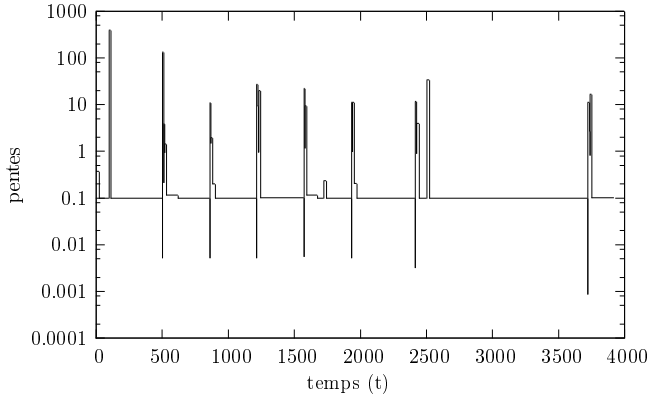


FIGURE 25 – Pentes calculées au niveau 0 de notre algorithme sur le flux $f^{(2)}_{8\to 11}$.

Les résultats du niveau 0 de l'algorithme 2 sont représentés sur la figure 25. La plupart du temps, les courbes affichent la pente $\rho = 0.1$. Elle correspond à la réception des messages Hello. À chaque fois que R_8 redémarre, un petit nombre de pentes très élevées est obtenu. Les valeurs s'échelonnent entre 10 et 400 messages par seconde, avec une valeur moyenne de 30. Avant chacune de ces pentes, on peut remarquer le calcul d'une pente extrêmement faible, oscillant entre 0.04 et 0.001. Elle représente le temps durant lequel aucun message n'est reçu (voir le paragraphe "Choix d'implémentation" de la partie 2.3.5).

La figure 26 représente les pentes calculées par le niveau 3 de notre algorithme. Les pentes obtenues sont plus lisses que celles du niveau 0. Ici, les valeurs ne dépassent pas 1 message par seconde, contre 400 pour le niveau 0. De plus, le nombre de changement de pente brusque est réduit. Malheureusement, dans le cas présent le niveau 3 n'apporte pas de connaissance supplémentaire sur le problème observé. Ce n'est cependant pas limitant car le niveau 0 fournit à lui seul toutes les informations nécessaires.

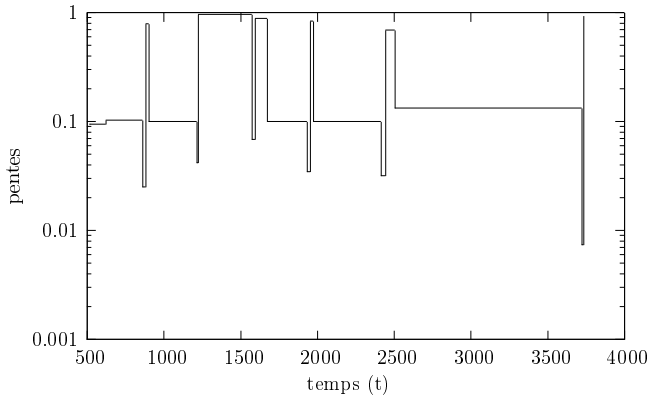


FIGURE 26 – Pentes calculées au niveau 3 de notre algorithme sur le flux $f^{(2)}_{8\to 11}$.

Perturbation de la convergence du protocole OSPF

Dans ce dernier cas d'étude, les nœuds démarrent les uns après les autres. Un nœud est lancé lorsque le nœud courant est totalement en fonction. L'ordre de démarrage est le suivant : $R_1, R_{10}, R_{11}, \ldots, R_{17}, R_2, \ldots R_9$, où les points de suspensions indiquent les noms des routeurs dans l'ordre croissant entre les bornes données. Cette expérience est réalisée afin d'observer le comportement de notre algorithme lorsque plusieurs routeurs doivent faire face à des changements simultanés. Ceci constitue un problème classiquement observé.

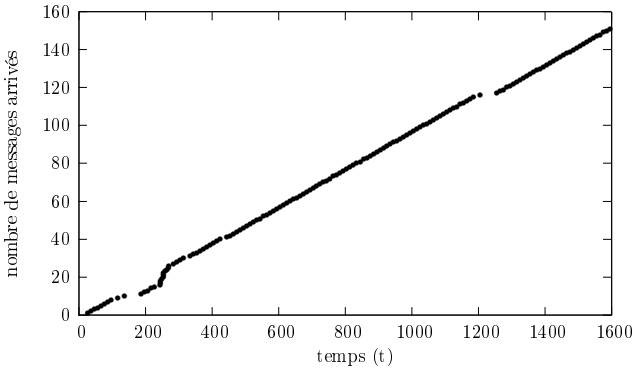


FIGURE 27 – Flux $f^{(3)}_{11\to 8}$ pendant 1600 secondes.

La figure 27 représente le flux ($f^{(3)}_{11\to 8}$) de messages OSPF envoyés au routeur R_8 par R_{11} dans ce contexte entre les instant 0 et 1600 s. Entre les dates 30 et 100 s, R_{11} envoie régulièrement des messages Hello vers R_8. Cependant, R_8 n'ayant pas encore démarré, il ne peut pas répondre. Par conséquent, entre 100 et 140 s, R_{11} réduit la vitesse d'envoi des Hello et finit par complètement cesser les envois. À l'instant 190 s, R_8 démarre et établit la connexion avec R_{11} par l'envoi d'un petit nombre de Hello (entre 190 et 230 s). Alors, ces deux routeurs partagent leurs connaissances du réseau (bases de données) par des échanges de LSAs. Ceci est graphiquement reconnaissable par l'arrivée soudaine de messages entre 230 et 271 s. Ensuite, les Hello sont envoyés périodiquement. Durant cette période, on peut noter la perte de plusieurs messages : aux instants 285, 335, 445 s. Ces déficits sont dus à la surcharge du routeur R_8 qui échange sa base de donnée avec les routeurs arrivants. Finalement, on peut observer une dernière perturbation entre 1200 et 1300 s. Elle s'explique par le fait que R_8 ne répond plus aux messages de R_{11} car il est occupé à se synchroniser avec R_9, qui vient d'arriver. Étant donné que R_9 et R_8 sont assez éloignés dans le réseau, cette perturbation se produit relativement longtemps après le démarrage de R_8.

Les résultats du niveau 0 de l'algorithme 2 sur le flux $f^{(3)}_{11\to 8}$ sont représentés sur la figure 28. On remarque immédiatement les deux perturbations mentionnées précédemment : le début des envois de messages de R_{11} vers R_8 entre les instants 0 et 450 s ainsi que la période d'intense activité du routeur R_8 entre 1200 et 1300 s.

Finalement, la figure 29 présente les résultats du niveau 1 de l'algorithme. Elle montre une amélioration flagrante par rapport au niveau précédent car la présence réitérée de perturbations s'accompagne d'un nombre restreint de détails. Les deux instants aux pentes extrêmement faibles indiquent les deux pertes de messages, et la pente beaucoup plus importante met en avant l'échange de LSA entre R_{11} et R_8.

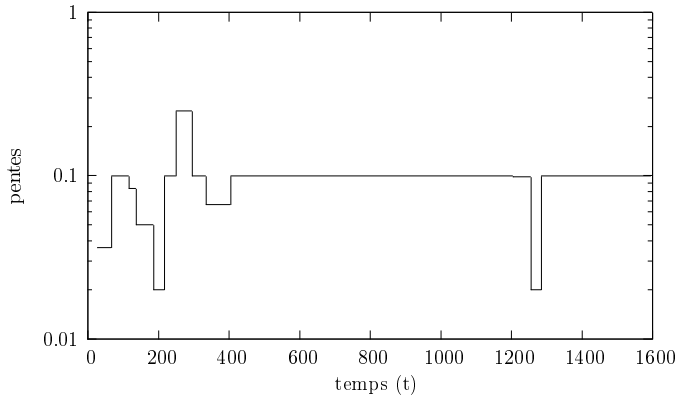


FIGURE 28 – Pentes calculées au niveau 0 de notre algorithme sur le flux $f^{(3)}_{11\to 8}$.

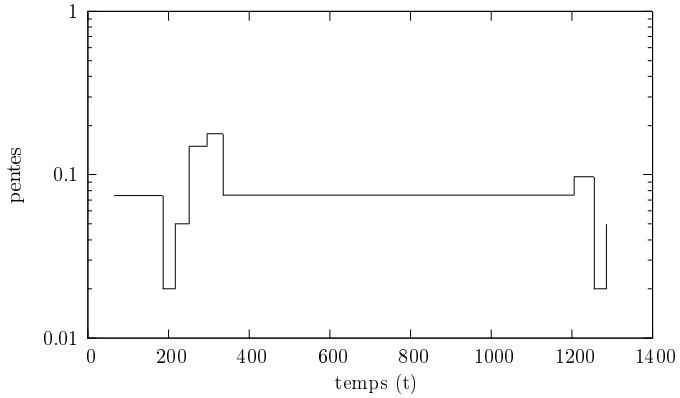


FIGURE 29 – Pentes calculées au niveau 1 de notre algorithme sur le flux $f^{(3)}_{11\to 8}$.

Complément sur les résultats de l'algorithme

L'algorithme a également prouvé son utilité et son efficacité lors de la mise en place du réseau de télécommunication allemand dans les installations d'Alcatel-Lucent Bell-labs. Dans le protocole OSPF, l'inondation des LSAs n'est pas effectuée entre tous les routeurs mais seulement sur un sous-ensemble décrivant un arbre couvrant du réseau. Lors d'un premier test du réseau allemand sur le cas de référence (présenté dans la section 2.4.3) nous nous sommes rapidement rendu compte d'un problème de configuration. En effet, lors de l'étude des flux de messages entre routeurs, nous nous sommes aperçu que les routeurs échangeant des LSAs ne décrivaient pas un arbre couvrant. Ceci prouve un problème de configuration du réseau.

Conclusion du chapitre

Dans ce chapitre, nous avons présenté un algorithme calculant des indicateurs de stabilité pour un flux de messages observé qui redéfini les contraintes à chaque fois que le modèle des arrivées varie. L'analyse de cette méthode préventive ne nécessite pas de connaissance particulière. Ceci permet une utilisation temps-réel et une amélioration de la proactivité des réseaux. De plus, le paramétrage très simple de cette méthode la rend facile à utiliser.

Par une étude théorique, nous avons montré que l'algorithme risque d'osciller entre le calcul d'un petit nombre de pentes dans le cas de flux périodiques. Cependant, si l'algorithme calcule une valeur suffisamment proche de la vitesse d'arrivée moyenne des

messages à l'algorithme alors ce dernier converge. Finalement, nous avons prouvé que dans le cas particulier des flux équilibrés la convergence est assurée.

Nous avons illustré le fonctionnement de notre algorithme sur un ensemble d'anomalies classiques pouvant mettre en danger la santé du réseau. Ces comportements ne sont à ce jour pas détectés par les méthodes de management actuelles. Nous avons alors prouvé que l'analyse des indicateurs de stabilités permet la mise en évidence de ces problèmes.

Les applications de l'algorithme, présentées dans ce chapitre, sont effectuées dans le contexte de flux de protocole. Nous avons particulièrement insisté sur l'analyse du protocole OSPF. Cependant, la méthode introduite ne fait aucune supposition sur la façon dont les messages doivent arriver. Elle peut donc être appliquée à des types de flux très variés. On pourrait notamment envisager l'application de cette méthode à des problèmes de sécurité. Dans un réseau stabilisé, les attaques seraient identifiées par les accroissements des arrivées de messages. On pourrait aussi envisager l'utilisation de cette technique sur les réseaux domiciles effectuant la surveillance des personnes âgées. Dans le chapitre 3, nous abordons le problème de la corrélation des alarmes. Nous montrerons que l'application du travail présenté dans ce chapitre permet une étude plus efficace des corrélations.

Néanmoins, ayant choisi de proposer une méthode très légère et adaptable à n'importe quelle situation, il paraît clair que les résultats obtenus ne présentent pas une étude du réseau aussi précise que les techniques réalisant une profonde analyse du système.

Chapitre 3

Détection d'événements majeurs par corrélation d'alarmes rares

Sommaire

Notations du chapitre **56**
Introduction **57**
3.1 État de l'art **57**
3.2 Graphes de dépendance des motifs pertinents **60**
3.2.1 Construction des motifs pertinents 60
3.2.2 Dépendances des motifs pertinents 62
3.3 Résultats **63**
3.3.1 Mise en œuvre 64
3.3.2 Contexte d'étude 64
3.3.3 Étude d'un premier élément de réseau 65
3.3.4 Étude d'un second élément de réseau 68
Conclusion du chapitre **71**

Le travail présenté dans ce chapitre résulte de travaux effectués en collaboration avec Anne Bouillard et Benoît Ronot. Il fait l'objet d'un article [89] présenté lors de la conférence CNSM'13 en 2013. Le rapport de recherche [88] introduit une version longue contenant davantage d'exemples et de résultats.

Notations du chapitre

La table 3.1 liste les notations spécifiques à ce chapitre.

Symboles	Significations
	Alarmes et flux
$\mathcal{A}$	Ensemble des alarmes
a, b, c, d, e, g	Des alarmes
$\mathcal{A}^*$	Ensemble des suites finies d'éléments de $\mathcal{A}$
$\mathcal{A}^+$	Ensemble des suites finies non-vides d'éléments de $\mathcal{A}$
$\|f\|$	Longueur du flux f
$\|f\|_a$	Nombre d'alarmes a dans f
$\overline{f}$	Support de f
	Définition des corrélations
M	Ensemble des alarmes fréquentes
θ	Seuil de sélection des alarmes fréquentes
$\mathcal{I}$	Ensemble des alarmes d'intérêt
$\iota_1, \ldots, \iota_\ell$	Motifs pertinents
$se(f)$	Séquence extraite de f
$ser(f)$	Séquence extraite de f et réduite
$\mathcal{P}(\mathcal{I})$	Ensemble des sous-ensembles de $\mathcal{I}$
f_{ex2}	Exemple de flux
(Re_1) et (Re_2)	Deux règles de réduction
$G_{dep} = (V_{dep}, E_{dep}, w_{dep})$	Graphe de dépendance pondéré
E_1 et E_2	Des éléments de réseau analysés

TABLE 3.1 – Listes des notations du chapitre 3.

Introduction

Devant la complexité croissante du management de réseau, les alarmes (messages d'information) furent créés afin d'aider les experts à analyser leur comportement. Ces messages, initialement envoyés lors de l'occurrence de problèmes critiques, n'informent désormais plus uniquement de menaces ([32]). Aujourd'hui, les réseaux pouvant facilement produire des millions d'alarmes par jour, la gestion manuelle de cette abondante ressource n'est plus envisageable. Cependant, le travail ([34]) réalisé par l'entreprise Control Arts Inc. montre que cette importante quantité d'informations est fortement redondante. En effet, une faute unique peut avoir plusieurs symptômes, se propager ou perdurer.

Le principe de la corrélation d'alarmes consiste à regrouper celles se rapportant à un même problème (afin de réduire le problème de redondance stipulé ci-dessus) et de mettre en avant celles correspondant à des fautes probables qui pourraient entraîner une instabilité du réseau. De nombreuses études se sont intéressées à cet enjeu ces dernières années (voir section 3.1). Malheureusement, les approches proposées restent limitées pour des applications proactives et temps-réel. La gestion automatique et efficace des alarmes demeure encore un défi.

Principe de la méthode développée L'idée développée dans ce chapitre repose sur le fait que l'enchaînement des alarmes caractérisant une faute possible décrit l'évolution des évènements dangereux apparaissant dans un réseau. Ainsi, détecter ces séquences permet de pratiquer une surveillance prédictive. Nous présentons ici une nouvelle heuristique construisant un graphe de dépendance statistique entre les alarmes. Grâce à ce graphe, nous extrayons des séquences significatives menant aux alarmes les plus représentatives d'une erreur critique.

Nous débutons ce chapitre, section 3.1, par la présentation des méthodes existantes en terme de corrélation d'alarmes. Nous verrons que l'état de l'art est très riche.

Puis, dans la section 3.2, nous présentons une nouvelle technique analysant un flux d'alarmes. Nous nous focalisons sur les signalements peu fréquents et nous proposons une méthode les extrayant du flux global sous la forme d'ensembles, dits *motifs pertinents*. Parmi cette collection d'éléments, nous distinguons les *alarmes rares* : alarmes très peu présentes risquant de conduire à des fautes critiques. Enfin, à partir des informations obtenues, nous établissons des graphes représentant les suites d'alarmes conduisant aux alarmes rares.

Enfin, dans la section 3.3, nous présentons une évaluation de la méthode proposée. Nous appliquons tout d'abord l'algorithme du chapitre précédent afin d'obtenir l'état de santé général du système. Ensuite, grâce à l'application de la nouvelle méthode, nous diagnostiquons les comportements dangereux et déterminons les problèmes apparaissant dans le réseau.

3.1 État de l'art

Depuis une trentaine d'années, de nombreuses études ont été proposées afin d'étudier les problèmes de corrélation d'alarmes. Dans cette partie, nous établissons une vue d'ensemble des méthodes existantes.

Méthode fondée sur des graphes de dépendances Kaztela et Schwartz ([40]) présentent une des premières heuristiques en matière de corrélation d'alarmes. Cette technique construit un graphe de dépendance statistique représentant les éléments du

réseau. Un lien de l'élément E_1 vers l'élément E_2 pondéré par $p_{1,2}$ indique qu'une panne apparaissant dans E_1 resurgit sur E_2 avec une probabilité $p_{1,2}$. À partir de ce graphe, le *domaine* de chaque alarme est établi : il s'agit de l'ensemble des éléments pouvant avoir généré cette alarme. Ce problème est une variation du *problème de la source unique* [18]. En fait, cet article présente et compare un ensemble d'algorithmes de localisation d'erreurs pouvant être appliqués à ce graphe. Cette étude ne cadre pas tout à fait avec nos objectifs. En effet, comme nous souhaitons une analyse proactive, il nous faut pouvoir observer l'évolution des problèmes majeurs.

Méthode définissant une architecture de management hiérarchisé Yemini et al. [81] décrivent un système de management de réseau (NMS : *Network Management Sytem*) effectuant la corrélation des évènements distribués. Il est appelé DECS pour *Distributed Event Correlation System*. Cette architecture récursive est composée de *domaines*, pouvant eux-même contenir des domaines. Chacun d'entre-eux dispose d'un *manageur* chargé de surveiller cette partie du réseau. Les analyses sont alors étagées afin de pouvoir être appliquées à des réseaux de taille conséquente. Le DECS comprend un algorithme de corrélation des évènements, connu sous le nom de *CODEBOOK*, construisant un graphe de causalité problèmes-symptômes. Afin de simplifier ce graphe complexe, les évènements communs aux fautes sont supprimés. Cette action affaiblit cependant les informations représentées dans le graphe. En effet, un évènement peut être significatif d'un problème mais insignifiant pour un autre. Ceci rend donc la détection de certaines fautes plus difficile. Quelques années plus tard, Chao et al. [12] ont proposé une nouvelle architecture également basée sur un raisonnement hiérarchisé et la création d'un graphe de causalité évènements-fautes, qui désormais est probabiliste. Suivant plusieurs paramètres, des indicateurs de confiances sont calculés afin de répertorier les alarmes entre *évènements significatifs et non-significatifs* pour chaque faute. Le système de management observe le réseau et diagnostique les problèmes apparus. Toutefois, ces travaux ne transmettent que les fautes diagnostiquées ce qui ne permet pas d'observer l'évolution des pannes et donc de réaliser une étude proactive.

Méthodes introduites pour une architecture spécifique Plusieurs méthodes proposent des solutions directement ajoutées dans le NMS. L'objectif est alors d'améliorer certains de ses aspects afin de réduire le flot brut d'alarmes observées en transmettant uniquement les informations pertinentes.

Par exemple, les méthodes introduites dans les articles [36] et [73] définissent une architecture d'alarmes utilisant une méthode de raisonnement à partir d'un modèle établi (*Model-Based Reasoning* [17]). Elles introduisent des règles groupant les alarmes symptomatiques d'un même problème. Plus précisément, IMPACT, présenté par Jacobson et Weissman dans [36], utilise un algorithme de règles de chaînage afin de réaliser la corrélation des alarmes. Le mécanisme de corrélation de Northern Telecom ([73]) a été implanté dans le langage *Smalltalk* et contient des mécanismes d'écriture de règles afin de maintenir la connaissance du réseau à jour (i.e. l'ensemble des règles détectant un problème spécifique).

Chyssler et al. [13], quant à eux, se focalisent sur la définition d'agents intelligents et implémentent un agent de corrélation basé sur l'architecture INFOSEC. Une alarme est générée si la somme des poids des alarmes apparues (attribués selon leurs sévérités) est supérieure à un seuil donné.

Dans [1], Amaral et al. proposent un système de corrélation divisé en trois niveaux. Le premier niveau réalise un pré-calcul. Il étudie les informations spatiales et temporelles afin de construire un ensemble d'alarmes compact appelé DLA (*Device Level Alarms*). Le

niveau de corrélation intervient ensuite. Il produit un graphe de dépendance des éléments du réseau et identifie la source et la destination des anomalies à l'aide d'une heuristique. Finalement, le niveau de représentation fournit une visualisation des chemins du graphe étant affectés par des anomalies. Le trafic est caractérisé par un modèle de management BLGBA (*Baseline for Automatic Backbone Management* [82]) qui génère un profil de comportement normal DSNS (*Digital Signature of Network Segment* [60]).

Toutes ces méthodes s'appliquent à des architectures spécifiques, ce qui ne permet pas une utilisation à grande échelle.

Méthode fondée sur de l'algèbre linéaire La technique proposée dans l'article [34] évalue mathématiquement la redondance des alarmes. Une alarme est dite redondante d'une autre alarme si elle apparaît, majoritairement, peu de temps après l'autre. Les occurrences des alarmes sont représentées par des gaussiennes centrées sur la date effective d'occurrence et les redondances sont alors détectées grâce à un calcul standard en algèbre linéaire appelé *décomposition en valeurs singulières*. Cependant, cette approche ne permet pas de considérer les alarmes apparaissant entre les alarmes redondantes, ce qui fournit une source d'information supplémentaire.

Méthodes fondées sur la définition de motifs Une autre façon d'appréhender le problème de la corrélation des évènements fautifs consiste à définir des motifs d'alarmes fréquents. C'est l'objectif des études [19] et [27]. Le principe consiste à procéder à une fouille de données sur le flux d'alarmes afin de définir des séquences temporellement corrélées. Ces méthodes sont assez populaires. Néanmoins, la recherche de motifs temporels est coûteuse. Par exemple, la méthode itérative décrite dans [19] est, au minimum, de complexité $O(i^5)$, où i est le nombre d'itérations effectuées. Ceci demeure trop complexe pour une méthode proactive. De plus, ces études se focalisent sur les motifs fréquents, alors que nous supposons que les alarmes importantes au contraire apparaissent peu.

Méthode fondée sur une machine à état fini probabiliste La méthode présentée dans [63] introduit des machines d'états finies probabilistes (PFSM : *Probabilistic Finite State Machine*) décrivant des erreurs de comportement afin de corréler les alarmes. Plus précisément, une PFSM représente la succession d'alarmes impliquant une erreur particulière. La faiblesse de cette approche repose sur le fait que l'ensemble des fautes possibles doit être connu à l'avance. En effet, une PFSM est construite pour chaque faute avant que le flux soit analysé. Ceci est gênant car les erreurs peuvent provenir de n'importe où et le spectre des fautes peut s'avérer très large.

Méthodes fondées sur des outils de modélisation Dans l'article [4], Benveniste et al. proposent une méthode en-ligne de diagnostique des systèmes asynchrones à événements discrets en effectuant des dépliages de réseaux de Petri. Ces réseaux représentent les liens entre les états des éléments du système et les occurrences d'évènements. Le dépliage du modèle renseigne l'ensemble des scenarii pouvant se produire. Cette méthode évite les problèmes d'explosions du nombre d'états du réseau de Petri déplié, qui constitue un problème classique lors de l'étude de ce type de systèmes, mais elle augmente le coût des calculs en-ligne. Les auteurs s'intéressent ultérieurement aux réseaux dont la topologie est susceptible de changer. Ils expliquent alors les limites du modèle jusqu'alors utilisé et proposent une nouvelle méthode fondée sur des grammaires de graphes ([28]).

Méthode fondée sur un coefficient de corrélation Dans l'article [80], Yang présente une méthode statistique hors-ligne. Les alarmes sont à nouveau représentées par des

gaussiennes et le coefficient de corrélation de Pearson [79] est appliqué. À la fin de l'étude, les liens entre les alarmes sont représentés à l'aide d'une matrice colorée [69].

Méthode fondée sur un algorithme génétique Weiss et Hirsh [72] proposent une méthode de prédiction des évènements rares. L'objectif consiste à rechercher des motifs maximisant les critères suivants :

- *efficacité* : pourcentage d'évènements prédits sur l'ensemble de ceux à détecter ;
- *précision* : pourcentage de prédiction correctes.

Cette méthode est élaborée à partir d'un algorithme génétique construisant des séquences temporisées d'événements rares. La population est initialisée avec un ensemble de motifs simples. À chaque itération, les éléments (parents) de la population sont "génétiquement" modifiés pour fournir des éléments fils. Le principe consiste à se rapprocher de la population maximisant les critères observés. Les expériences montrent que la solution maximale est atteinte après 2000 itérations. Ceci reste trop complexe pour une méthode proactive.

De nombreuses méthodes proposent des solutions concernant la corrélation des évènements majeurs des réseaux. Comme dans le chapitre précédent, l'enjeu réside dans la capacité à retrouver efficacement des informations significatives dans une grande quantité de données et de réagir rapidement suivant les résultats récoltés. Dans ce chapitre, nous proposons une nouvelle technique de corrélation basée sur la supposition que les alarmes significatives sont très peu représentées dans le flux.

3.2 Graphes de dépendance des motifs pertinents

Présentons maintenant notre méthode. Dans cette partie, nous étudions un flux d'alarmes, séquence finie ordonnée de noms d'alarmes, et nous extrayons des motifs pertinents, contenant des alarmes significatives. Ensuite, nous établissons les dépendances entre ces motifs afin de pouvoir effectuer une expertise des problèmes du réseau.

3.2.1 Construction des motifs pertinents

Dans la suite, nous utilisons les notations couramment appliquées à la théorie du langage : $\mathcal{A}$ étant un ensemble fini, $\mathcal{A}^*$ représente l'ensemble des suites finies d'éléments de $\mathcal{A}$, et $\mathcal{A}^+$ l'ensemble des suite finies non-vides d'éléments de $\mathcal{A}$. Étant donné un flux $f \in \mathcal{A}^*$, nous notons $|f|$ la longueur de f et $\forall a \in \mathcal{A}$, nous représentons par $|f|_a$ le nombre d'occurrences de l'alarme a dans f. Le support de f regroupe l'ensemble des alarmes apparaissant au moins une fois dans f. Il est défini par : $\overline{f} = \{a \in \mathcal{A} \mid |f|_a \geqslant 1\}$. Finalement, le symbole $\cdot$ désigne l'opérateur de concaténation : si $f_1 = a_1 \dots a_i$ et $f_2 = b_1 \dots b_j$ alors $f_1 \cdot f_2 = a_1 \dots a_i b_1 \dots b_j$.

Exemple 3.2.1. *Prenons $\mathcal{A} = \{a, b, c, d\}$ et $f = abcbca \in \mathcal{A}^*$ un flux très court. Alors $f \in \mathcal{A}^+$, $|f| = 6$, $|f|_a = 2$ et $\overline{f} = \{a, b, c\}$.*

Étudions un flux d'alarmes. Remarquons que dans notre étude la taille de $\mathcal{A}$ est approximativement 20 et que la longueur de f est arbitrairement longue. Dans nos tests, $|f| \simeq 10^5$, mais cette grandeur peut s'avérer bien plus importante. Dans ce type de flux, on constate que certaines alarmes sont très fréquentes alors que d'autres apparaissent très peu. Comme nous l'avons précisé en introduction, nous nous focalisons sur l'étude de ces dernières.

Dans cette section, notre objectif est de déterminer des groupes corrélés d'alarmes peu fréquentes. Nous procédons de la manière suivante.

- Identification des alarmes M les plus fréquentes ;
- Division de f en une séquence de motifs pertinents ensemblistes, $se(f)$;
- Étude de la corrélation entre les motifs pertinents et réduction de l'ensemble $se(f)$.

Identification des alarmes les plus fréquentes Cette tâche s'effectue simplement en utilisant un seuil θ et en comptant le nombre d'occurrences de chaque alarme $a \in \mathcal{A}$. L'ensemble M est alors défini par :

$$M = \{a \in \mathcal{A} \mid |f|_a/|f| \geq \theta\}.$$

Construction de motifs pertinents ensemblistes Dans la suite d'alarmes étudiée, certaines d'entre elles sont envoyées plusieurs fois. C'est notamment le cas lorsque plusieurs éléments de réseau détectent une même faute. Ainsi, lorsqu'une alarme a précède une alarme b, cela ne signifie pas que b est une conséquence de a ou que a fut produite avant b. De plus, les fichiers logs de ce type de flux présentent fréquemment des motifs de type $abababa\ldots$. Finalement, nous savons que le système de management ne retransmet pas immédiatement les alarmes reçues mais les enregistre pour les envoyer une fois rassemblées. L'ordre exact entre les alarmes ne s'avère donc pas très significatif et il n'est pas nécessaire de le maintenir entre les alarmes consécutives ou pratiquement consécutives. En conséquence, nous transformons f en une séquence d'ensembles d'alarmes. Considérons l'ensemble $\mathcal{I} = \mathcal{A}/M$ des alarmes d'intérêt peu fréquentes. Il existe alors une décomposition unique de f :

$$f = m_0 \cdot \iota_1 \cdot m_1 \cdot \iota_2 \cdots \iota_\ell \cdot m_\ell,$$

où $m_0, m_\ell \in M^*$, $m_1, \ldots, m_{\ell-1} \in M^+$, et $\iota_1, \ldots, \iota_\ell \in \mathcal{I}^+$.

Désormais, nous nous focalisons sur les occurrences des alarmes peu fréquentes (appartenant à $\mathcal{I}$). Dans l'écriture précédente les suites finies non-vides $\iota_1, \ldots \iota_\ell$ décrivent l'apparition des alarmes d'intérêt. Nous les nommons motifs pertinents et nous considérons la séquence extraite de f, $se(f)$, suivante.

$$se(f) = \overline{\iota_1} \cdot \overline{\iota_2} \cdots \overline{\iota_\ell} \in \mathcal{P}(\mathcal{I})^*.$$

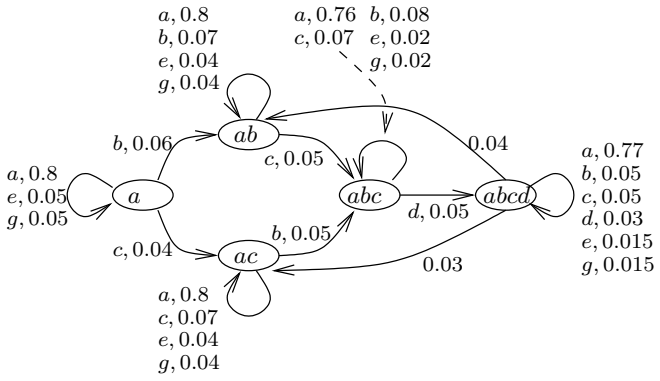


FIGURE 30 – Générateur markovien pour l'exemple 3.2.2.

Exemple 3.2.2. *Considérons un ensemble de 6 alarmes, $\mathcal{A} = \{a, b, c, d, e, g\}$. Supposons que a est une alarme fréquente et que l'occurrence des alarmes b et c induit l'occurrence de d. Lorsque d apparaît, la faute lui étant associée est réparée, ce qui implique l'arrêt*

des envois d'alarmes b et d mais pas a et c. Les alarmes e et g ne sont pas corrélées à la faute liée à l'alarme d et apparaissent constamment avec un faible taux d'arrivé.

Afin de générer un fichier log d'alarmes correspondant à ce type de phénomène, nous créons le générateur markovien de la figure 30. Les arcs sont étiquetés par les noms des alarmes associés à la probabilité de générer celle-ci. Notons que la probabilité de produire e ou g diminue pour modéliser le fait que la possibilité de générer b et c augmente.

Si le seuil θ utilisé pour séparer $\mathcal{A}$ en M et $\mathcal{I}$ est 0.5, alors $M = \{a\}$ et $\mathcal{I} = \{b, c, d, e, g\}$.

Utilisant ce générateur, nous avons produit un flux f_{ex} tel que $|f_{ex2}| = 1000$. Nous avons remarqué que d apparaissait majoritairement seule, mais aussi, assez fréquemment, dans les ensembles $\{b, d\}$ et $\{c, d\}$. Enfin, elle est représentée dans les ensembles $\{e, d\}$, $\{g, d\}$ ou encore, bien que moins souvent, dans des ensembles de tailles supérieurs. Ceci correspond au modèle énoncé.

Lors de l'expérience réalisée ci-dessus, nous avons obtenu $|se(f_{ex2})| = 180$. Ceci reste trop grand pour être aisément analysé. Notre but étant de prodiguer une détection rapide des problèmes éventuels, il faut trouver une méthode permettant une réduction supplémentaire de la séquence extraite $se(f)$. L'objectif de la troisième étape, décrite ci-après, est de fournir une vue d'ensemble plus schématique.

Réductions de l'ensemble des motifs Afin de réduire $se(f)$, nous définissons les deux règles suivantes. Soit $\iota_1, \iota_2 \in \mathcal{I}^+$.

$$(Re_1)\ \iota_1\iota_2\iota_1 \to \iota_1 \cup \iota_2 \qquad (Re_2)\ \iota_1\iota_2 \to \iota_1 \text{ si } \iota_2 \subseteq \iota_1.$$

En d'autres termes :
- si $se(f) = z_1 \cdot \iota_1\iota_2\iota_1 \cdot z_2$, alors la séquence extraite, $se'(f)$, obtenue après l'application de la règle (Re_1) est $se'(f) = z_1 \cdot (\iota_1 \cup \iota_2) \cdot z_2$;
- si $\iota_2 \subseteq \iota_1$ et $se(f) = z_1 \cdot \iota_1\iota_2 \cdot z_2$, alors la séquence extraite, $se''(f)$, obtenue après l'application de la règle (Re_2) est $se''(f) = z_1 \cdot \iota_1 \cdot z_2$.

Les règles peuvent être récursivement appliquées jusqu'à ce qu'aucune d'entre elles ne puissent plus l'être. La séquence extraite réduite de f alors obtenue est notée $ser(f)$. Cependant, notons que l'application de ces règles n'est pas commutative. Considérons la séquence de motifs ensemblistes $\{b, d\}\{b\}\{c\}\{b\}$, les règles (Re_1) et (Re_2) peuvent être appliquées. Néanmoins, utilisation de (Re_1) produit $\{b, d\}\{b, c\}$ et (Re_2) n'est plus applicable ; à l'inverse commencer avec (Re_2) donne $\{b, d\}, \{c\}\{b\}$ mais alors (Re_1) n'est plus utilisable. Malgré tout, choisir d'appliquer (Re_1) avant (Re_2) ou inversement implique une légère différence qui n'altère pas la qualité de l'étude. En conséquence, nous choisissons arbitrairement d'appliquer (Re_1) de la gauche vers la droite, puis (Re_2) jusqu'à ce qu'aucune règle ne puisse être utilisée.

Exemple 3.2.3. *L'application de ces règles sur $se(f_{ex2})$ de l'exemple 3.2.2, donne $|ser(f_{ex2})| = 93$. Sur cet exemple on ne peut observer une réduction drastique car le nombre d'alarmes est trop petit. Sur les cas considérés dans la section 3.3, nous montrerons que la réduction est bien plus intéressante.*

3.2.2 Dépendances des motifs pertinents

Discernons un type d'alarme particulier, sur lequel nous nous focaliserons désormais : les *alarmes rares*. Elles n'apparaissent qu'un petit nombre de fois dans un fichier log et correspondent aux alarmes révélatrices de comportements dangereux pour la santé du réseau. Généralement, une réparation est effectuée peu après leurs occurrences.

Utilisons maintenant l'heuristique introduite dans la section précédente afin de trouver des motifs conduisant aux alarmes rares. L'identification des motifs y conduisant permet la détection des causes premières des mauvais comportements avant qu'une panne ne se produise.

Afin d'effectuer cette opération, nous construisons un graphe de dépendance $G_{dep} = (V_{dep}, E_{dep}, w_{dep})$ des motifs ensemblistes trouvés à l'aide de notre heuristique : le graphe pondéré G_{dep} contient l'ensemble des motifs V_{dep} reliés entre eux par les arcs de l'ensemble E_{dep} pondérés par $w_{dep} : E_{dep} \to \mathbb{N}$ tels que :

- $V_{dep} = \overline{ser(f)}$ constitue l'ensemble des motifs de $ser(f)$ apparaissant au moins une fois ;
- $E_{dep} = \{(\iota_1, \iota_2) \mid \exists z_1, z_2 \text{ tel que } ser(f) = z_1 \cdot \iota_1\iota_2 \cdot z_2\}$;
- $w_{dep}(\iota_1, \iota_2) = |\{(z_1, z_2) \mid ser(f) = z_1 \cdot \iota_1\iota_2 \cdot z_2\}|$, le nombre d'occurrences de $\iota_1\iota_2$ dans $ser(f)$.

Une fois ce graphe construit, nous souhaitons déterminer les alarmes peu fréquentes conduisant à une alarme rare donnée.

Exemple 3.2.4. *Reprenons l'exemple 3.2.2. Étant donné que $|ser(f_{ex2})| = 93$, le graphe est trop grand pour être représenté. Concentrons nous sur l'ensemble des alarmes rares : $\{d\}$ et ses prédécesseurs. Le graphe ainsi obtenu est représenté sur la figure 31. Nous remarquons immédiatement que tous les prédécesseurs contiennent les alarmes b ou c (et très souvent les deux) cependant seulement trois présentent e ou g. Ceci concorde avec la description du générateur markovien de l'exemple 3.2.2.*

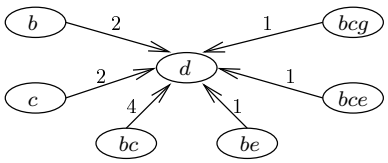


FIGURE 31 – L'alarme rare d et ses prédécesseurs.

La difficulté à analyser des flux d'alarmes réside dans le fait que les alarmes sont enregistrées sur de longues périodes (typiquement une année). En un an, de nombreux événements apparaissent dans un réseau, générant des évolutions de son comportement. En conséquence, le graphe, trop complexe, ne peut être étudié dans sa globalité. On pourrait envisager la découpe du fichier log observé en plusieurs périodes. Néanmoins, cette technique peut s'avérer dangereuse car déterminer l'endroit de la division n'est pas évidente et l'on risquerait de couper une période intéressante. Ceci ferait perdre des informations de corrélation. Nous choisissons d'étudier le flux complet et de définir un ensemble de sous-graphes se focalisant chacun sur une alarme rare donnée (une pathologie spécifique). Un sous-graphe représente une courte succession de motifs pertinents apparaissant avant et après une alarme rare. L'observation d'un sous-graphe fournit une analyse concise et précise des alarmes rares. Ceci définit des informations concernant l'origine de la faute émettant l'alarme rare et montre si une solution au problème a été apportée. L'utilisation de ce type de graphe permet à un expert réseau de connaître les alarmes risquant de conduire à un état dangereux du réseau mais aussi de déterminer à tout instant si un problème menace la santé du réseau et de réagir avant une panne.

3.3 Résultats

La section précédente présente une méthode réalisant une corrélation simple des alarmes d'un flux. Prouvons maintenant son efficacité ! Nous commençons par introduire

la mise en œuvre de l'algorithme. Puis, nous expliquons le contexte de l'étude. Finalement, nous réalisons l'expertise de deux des éléments du réseau étudié. Nous montrons que l'utilisation de l'algorithme introduit dans le chapitre précédent (chapitre 2) fournit de précieux renseignements. Il fut utilisé avec 4 niveaux d'expertise, lorsque $T = \sigma = 10$ pour les niveaux 0 et 1 et $T = 50$, $\sigma = 150$ pour les niveaux 2 et 3. Ensuite l'utilisation de l'algorithme de corrélation affine l'analyse en fournissant la source des évènements détectés.

3.3.1 Mise en œuvre

Les résultats présentés dans cette section sont obtenus à partir d'un programme réalisé en Python. Il nécessite un fichier contenant uniquement le nom des alarmes du flux observé. Il génère deux flux : le premier ne contient plus que des suites d'alarmes peu fréquentes ($se(f)$) ; le second ($ser(f)$) est obtenu après l'application des règles (Re_1) et (Re_2) sur le premier. Le second flux établit le graphe des dépendances entre motifs pertinents.

3.3.2 Contexte d'étude

Présentons le type de réseau étudié et un cas pathologique particulier.

Acronyme	Nom complet	E_1^{θ}	$E_1^{\theta'}$	$E_2^{\theta''}$
AIS	Ais	$\mathcal{I}$	$\mathcal{I}$	M
CFF	Cooling fan failure			$\mathcal{I}$
CP	Cabling problem	$\mathcal{I}$	$\mathcal{I}$	
CSF	Communication subsystem failure	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
DS	Degraded signal	$\mathcal{I}$	M	M
EBER	Excessive ber	$\mathcal{I}$	$\mathcal{I}$	M
FO	Frequency offset	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
HK	House keeping	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
LOF	Loss of frame			$\mathcal{I}$
LOS	Loss of signal	M	M	M
LOT	Loss of timing sources	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
MBM	Mib backup misaligned			$\mathcal{I}$
NI	Node isolation	$\mathcal{I}$	M	$\mathcal{I}$
NNF	Ne notification flooding			$\mathcal{I}$
RDI	Remote defect indication	$\mathcal{I}$	$\mathcal{I}$	M
RI	Resource isolation	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
RUM	Replaceable unit missing	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
RUP	Replaceable unit problem	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$
RUTM	Replaceable unit type mismatch			$\mathcal{I}$
SSF	Server signal failure	$\mathcal{I}$	$\mathcal{I}$	M
TIM	Trace identifier mismatch			$\mathcal{I}$
U	Unequipped	M	M	$\mathcal{I}$
URU	Underlying resource unavailable			M
UT	Unavailable time	$\mathcal{I}$	$\mathcal{I}$	M

TABLE 3.2 – Listes des acronymes d'alarmes.

Contexte général Afin de valider l'algorithme proposé, nous testons notre méthode sur des problèmes réels rencontrés par un opérateur (client chez Alcatel-Lucent Customer

Services). L'équipe du service client d'Alcatel-Lucent a pratiqué une profonde analyse du réseau en se basant sur son architecture et une expertise manuelle des technologies déployées. Cet opérateur dispose d'un réseau optique de hiérarchie numérique synchrone (SDH : *Optical Synchronous Digital Hierarchy*) faisant partie du niveau 1 du modèle de l'OSI. Le réseau étudié est complexe et contient plus de $6.2\,10^3$ éléments, tous capable d'émettre des alarmes. Actuellement, chaque élément analyse la masse totale d'information par des méthodes de management traditionnelles, ce qui constitue une tâche difficile.

Dans les paragraphes suivants, nous présentons l'analyse de deux éléments de réseau, E_1 et E_2. L'expertise est réalisée à partir de fichiers logs ayant enregistrés les alarmes émises durant une année. Ceci représente plus de $2.0\,10^5$ alarmes rien que pour l'étude d'un élément. Notons qu'afin de présenter des résultats intéressants et significatifs, les nœuds choisis sont ceux définis par le service client d'Alcatel-Lucent comme étant tombés en panne plusieurs fois.

Dans cette section, les flux sont des séquences composées de couples (nom d'alarme, date) afin de pouvoir appliquer l'algorithme du chapitre 2. De plus, f^i représente le flux d'alarmes émis par l'élément E_i et f^i_a le sous-flux de f^i ne contenant que les occurrences de l'alarme a. La table 3.2 liste les acronymes et les noms complets des alarmes envoyées par deux éléments de réseau que nous notons E_1 et E_2. Elle contient également la division entre alarmes fréquentes et peu fréquentes effectuée par l'algorithme de corrélation sur f^1 avec les paramètres θ et θ' et sur f^2 avec θ''. Une ligne vide indique que l'alarme en question n'est pas émise par E_1 ou E_2.

Cas d'étude Dans ce chapitre, nous étudions un problème fréquemment rencontré dans les réseaux SDH : la panne d'un laser. Plusieurs alarmes sont connues pour être symptomatiques de ce problème : LOS, AIS, LOF, RUP et RUM.

Énumérons quelques points importants de ce cas d'étude :

- Les alarmes LOS et LOF apparaissent très fréquemment. Ceci provoque l'envoi d'alarmes AIS signifiant que le signal d'origine est remplacé par l'émission de tous les bits à 1.
- les alarmes LOS, AIS et LOF peuvent concerner bien d'autres problèmes qu'une panne de composant d'un laser, cependant, RUP, RUM et TF sont véritablement significatifs de ce cas d'étude.

3.3.3 Étude d'un premier élément de réseau

Penchons nous tout d'abord sur l'analyse de l'élément de réseau E_1 ayant généré $36.0\,10^3$ alarmes de 17 types en un an.

La figure 32 représente les résultats du niveau 0 de l'algorithme du chapitre 2. On remarque que les pentes calculées sont faibles et durent relativement longtemps jusqu'à $1.0\,10^7$ s. Après cet instant, l'amplitude des pentes devient progressivement plus importante et le nombre de variations s'intensifie. Après $1.6\,10^7$ s, les fluctuations atteignent leur maximum et demeurent ainsi jusqu'à la fin de l'observation. On peut supposer qu'un problème majeur apparaît après $1.0\,10^7$ s.

Concentrons-nous maintenant sur l'étude de quelques alarmes particulières. Nous nous focalisons sur RUP, importante pour le cas d'étude, mais également sur EBER, FO, CSF et RI. Nous montrerons que ces deux dernières fournissent des informations significatives concernant le bon fonctionnement de E_1.

La courbe de la figure 33 (resp. figure 34) présente les pentes calculées par le niveau 0 de l'algorithme 2 sur les sous-flux f^1_{RUP} et f^1_{EBER} (resp. f^1_{RUP} et f^1_{FO}). Finalement, la figure 35 représente les résultats pour les sous-flux f^1_{CSF} et f^1_{RI}. Ces graphiques fournissent les informations suivantes :

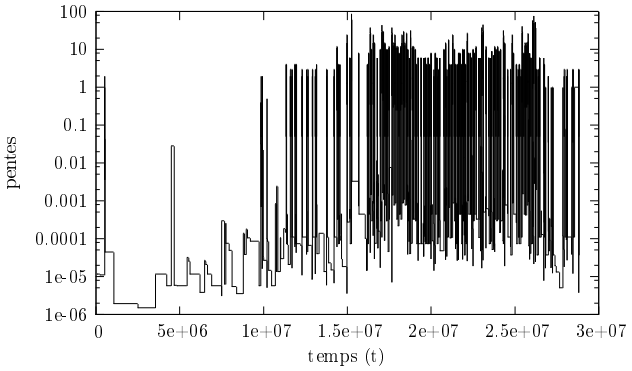


FIGURE 32 – Résultat du niveau 0 de l'algorithme 2 sur le flux f^1.

- RUP n'apparaît dans f^1 qu'au début de l'observation, durant les premières $7.0\,10^6$ s. Nous nous trouvons donc en présence du cas d'étude considéré ;
- EBER est générée pratiquement tout au long de l'observation. Néanmoins, on peut remarquer deux périodes d'arrivées majeures : lorsque RUP est présente (avant $1.0\,10^7$ s) et à la fin de l'année (après $2.0\,10^7$ s) ;
- FO n'apparaît qu'à trois reprises. Deux arrivées soudaines se manifestent aux dates $5.0\,10^5$ s et $1.0\,10^7$ s. Enfin, un nombre plus important d'occurrences se présente après $2.2\,10^7$ s ;
- CSF est en permanence générée par E_1. Cependant, un ensemble d'arrivées soudaines intervient entre $2.2\,10^7$ et $2.8\,10^7$ s ;
- RI se manifeste brièvement à la fin de l'observation.

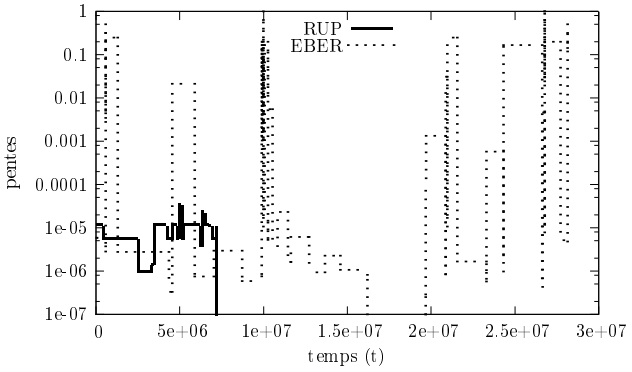


FIGURE 33 – Résultats du niveau 0 de l'algorithme 2 sur les sous-flux f^1_{RUP} et f^1_{EBER}.

À partir de ces observations, on remarque que EBER et FO se présentent majoritairement lorsque RUP apparaît. On peut conjecturer une corrélation. De plus, l'instabilité observée dans le flux global débute après les occurrences de RUP. On peut donc supposer que le problème majoritaire de E_1 ne provient pas du cas d'étude. À la fin de l'observation, les occurrences de RI succèdent à celles de CSF. On peut également supputer une corrélation entre ces dernières.

Concentrons nous maintenant sur l'heuristique présentée dans ce chapitre afin d'obtenir une étude précise de E_1. Testons la tout d'abord lorsque le seuil est $\theta = 0.3$. Ceci identifie deux alarmes comme fréquentes : $M = \{\text{U, LOS}\}$. Elles représentent à elles seules 97% du total des occurrences d'événements. La colonne E_1^θ de la table 3.2 indique la répartition des alarmes entre les ensembles M et $\mathcal{I}$.

La figure 36 représente le graphe des motifs pertinents de f^1 lorsque $\theta = 0.3$. Afin

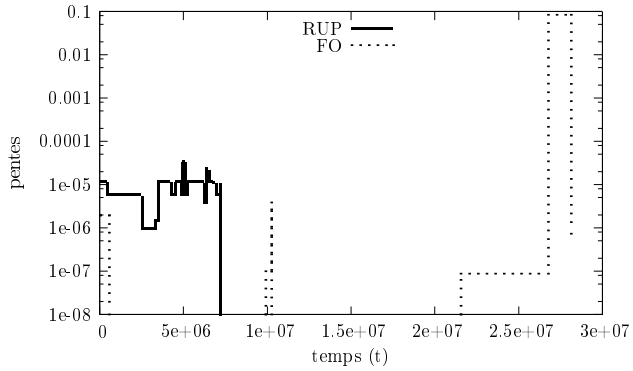


FIGURE 34 – Résultats du niveau 0 de l'algorithme 2 sur les sous-flux f^1_{RUP} et f^1_{FO}.

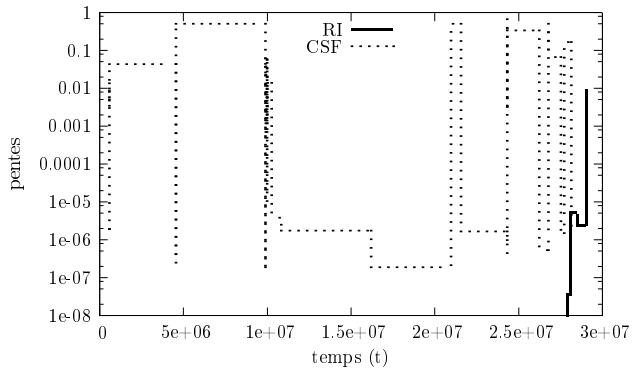


FIGURE 35 – Résultats du niveau 0 de l'algorithme 2 sur les sous-flux f^1_{CSF} et f^1_{RI}.

de simplifier ce graphe nous avons regroupé les alarmes DS, NI, CSF, EBER et RDI dans un ensemble appelé *Groupe*. Une double barre rouge est placée dans le graphe afin d'indiquer qu'une longue période sépare deux motifs. Ainsi, pendant un long moment, seules les alarmes de l'ensemble M apparaissent. En effet, les alarmes significatives d'un problème sont présentes uniquement au début et à la fin de f^1.

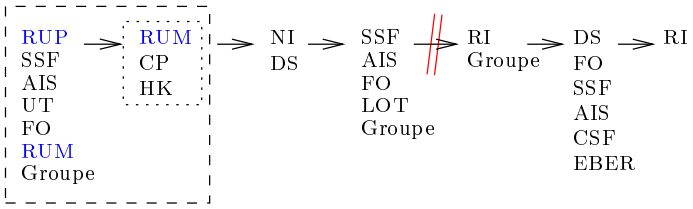


FIGURE 36 – Graphe acyclique des motifs pertinents de f^1 lorsque $\theta = 0.3$.

Les alarmes rares indiquant un problème critique sont RUM et RUP étant donné qu'elles apparaissent le moins souvent. Ainsi, l'observation des alarmes corrélées avec RUP et RUM permet de diagnostiquer le problème source présent. On peut également noter que HK et CP sont liées à RUM (voir l'encadré en pointillés de la figure 36). En effet, elles n'apparaissent qu'une seule fois dans le graphe, dans un motif contenant également RUM.

Néanmoins, le premier motif, contenant RUP, est très grand. Ceci peut cacher des corrélations importantes. Nous pratiquons alors une seconde étude du flux f^1 en utilisant le paramètre $\theta' = 0.003$ afin de réduire la taille de ce motif. L'ensemble des alarmes

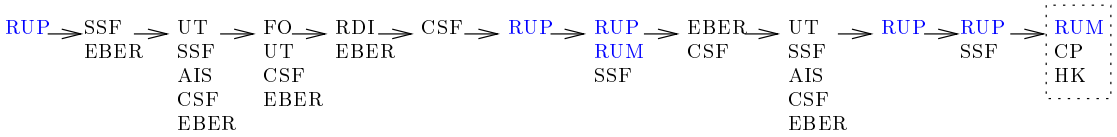


FIGURE 37 – Graphe partiel des motifs pertinents de f^1 lorsque $\theta' = 0.003$, représenté plus synthétiquement dans l'encadré de la figure 36.

majeures devient alors $M = \{\text{U, LOS, DS, NI}\}$. La nouvelle répartition entre M et $\mathcal{I}$ est présentée dans la colonne $E_1^{\theta'}$ de la table 3.2. Le graphe contient désormais 59 motifs pertinents ($|ser(f)| = 59$) au lieu de 7 précédemment. Celui-ci étant trop grand pour être entièrement représenté, nous avons exhibé la partie de ce graphe correspondant aux alarmes émises au début de l'étude sur la figure 37. L'utilisation de l'heuristique avec θ' permet d'obtenir un zoom sur les relations des alarmes représentées dans le rectangle en pointillés sur la figure 36. On observe alors que la corrélation entre RUM, HK et CP est maintenue. De plus, maintenant, RUP apparaît plusieurs fois, souvent précédée de SSF et CSF.

En conclusion, la présence d'EBER et UT dans le graphe de dépendance indique que le problème affecte un démodulateur du nœud. Ceci est confirmé par l'observation de FO avec ces dernières. À la fin de cette première partie du graphe de dépendance, RUM et CP montrent que l'élément défectueux a été remplacé. Ceci est confirmé par l'apparition de l'alarme HK. Nous remarquons également la présence d'un second problème affectant davantage E_1. D'après la figure 37, on peut déduire que le problème est lié à l'occurrence des alarmes CSF et SSF, étant donné quelles sont présentes dans chacun des motifs pertinents. Ceci indique qu'un problème provient d'un lien optique de E_1 ou de l'un de ses voisins. Plus précisément, CSF et SSF informent que la communication avec un élément est difficile. La réception de l'alarme RI permet de conclure que celui-ci ne peut finalement plus communiquer.

3.3.4 Étude d'un second élément de réseau

Focalisons nous maintenant sur l'étude de E_2 ayant généré $2.0\,10^5$ alarmes de 23 types durant une année.

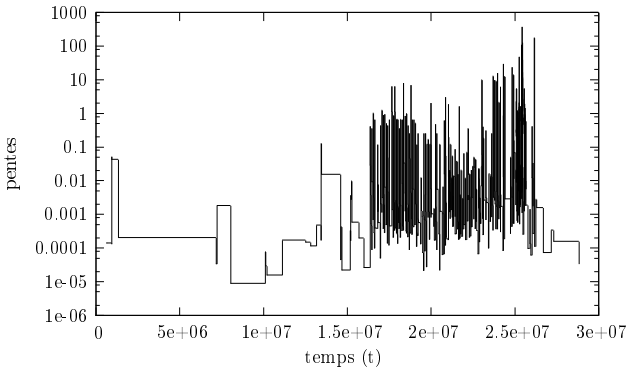


FIGURE 38 – Résultat du niveau 3 de l'algorithme 2 sur le flux f^2.

La figure 38 représente les pentes calculées au niveau 3 de l'algorithme 2. Nous remar-

quons immédiatement qu'entre $1.6\,10^7$ s. et $2.7\,10^7$ s. les arrivées s'intensifient et varient considérablement. Plus précisément, l'instabilité débute aux alentours de $6.0\,10^6$ s. Ceci s'explique simplement par une panne apparaissant très tôt dans l'élément et l'affectant progressivement. L'élément E_2 génère alors de plus en plus d'alarmes.

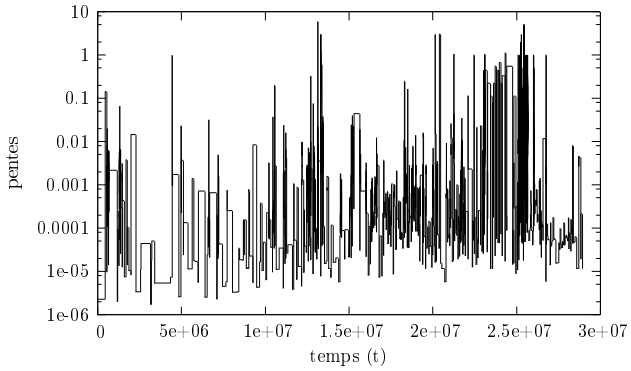


FIGURE 39 – Résultats du niveau 0 de l'algorithme 2 sur le sous-flux f^2_{AIS}.

Nous utilisons le niveau 0 de notre algorithme de détection sur le sous-flux f^2_{AIS}. Bien que AIS soit définie par notre heuristique de corrélation comme une alarme majeure, elle demeure un bon indicateur des problèmes d'un réseau. L'observation des pentes d'arrivée, figure 39, nous permet de déduire la présence permanente de AIS dans le flux f^2. Ceci sous-entend que E_2 est fortement perturbé. Notons qu'il s'agit d'une alarme mineure pour E_1.

Concentrons-nous à nouveau sur quelques alarmes importantes pour le cas d'étude : RUP et RUM. Nous analysons également les alarmes MBM et CSF, qui fournissent des informations complémentaires.

La figure 40 représente les pentes des arrivées calculées au niveau 0 pour les sous-flux f^2_{RUP} et f^2_{RUM}. L'alarme RUP se manifeste avant $6.5\,10^6$ s et après $2.5\,10^7$ s. Au début de la période observée, avant $5.0\,10^6$ s, RUM apparaît faiblement. Elle se présente à nouveau entre les instants $1.6\,10^7$ et $2.1\,10^7$ s, et encore entre $2.3\,10^7$ et $2.6\,10^7$ s.

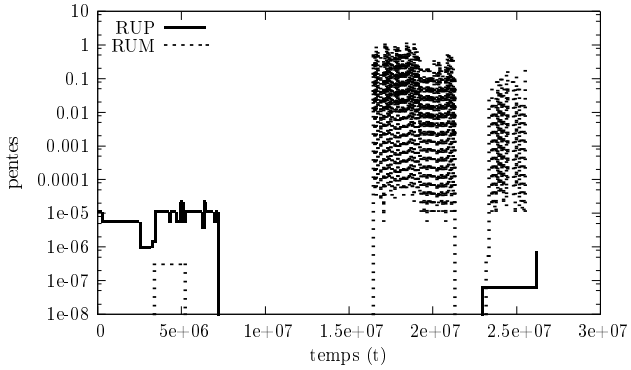


FIGURE 40 – Résultats du niveau 0 de l'algorithme 2 sur les sous-flux f^2_{RUP}, f^2_{RUM}.

Remarquons que le premier pic d'arrivées soudaines de RUP a exactement la même forme que celui observé dans E_1. Ceci semble indiquer, qu'avant $6.5\,10^6$ s., E_1 et E_2 observent la même pathologie du réseau. Cependant, E_2 reçoit une seconde éruption d'arrivées soudaines de RUP. Ceci n'est pas le cas pour E_1.

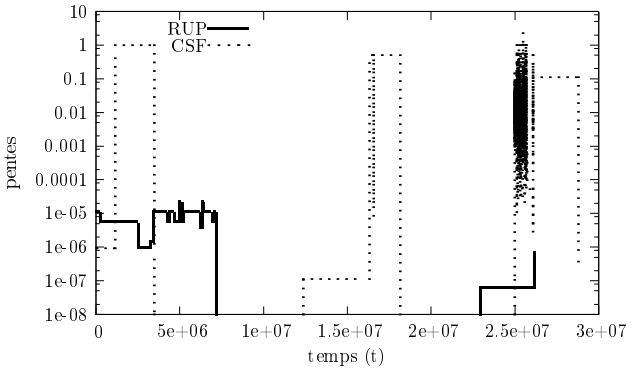


FIGURE 41 – Résultats du niveau 0 de l'algorithme 2 sur les sous-flux f^2_{RUP} et f^2_{CSF}.

Les pentes calculées par le niveau 0 de l'algorithme 2 sur le sous-flux f^2_{CSF} sont représentées sur la figure 41, accompagnées de celles du sous-flux f^2_{RUP}. Les résultats obtenus montrent trois périodes d'arrivées soudaines : la première se présente avant l'instant $4.0\,10^6$ s ; la seconde apparaît entre $1.3\,10^7$ et $1.7\,10^7$ s ; finalement, la dernière se manifeste après $2.5\,10^7$ s.

Finalement, la figure 42 représente les pentes calculées sur les sous-flux f^2_{MBM} et f^2_{RUP}. MBM présente également trois périodes d'arrivées soudaines : la première se manifeste entre $1.6\,10^7$ et $1.9\,10^7$ s ; la deuxième, plus petite, apparaît à l'instant $2.2\,10^7$ s ; finalement la dernière arrive aux alentours de $2.5\,10^7$ s.

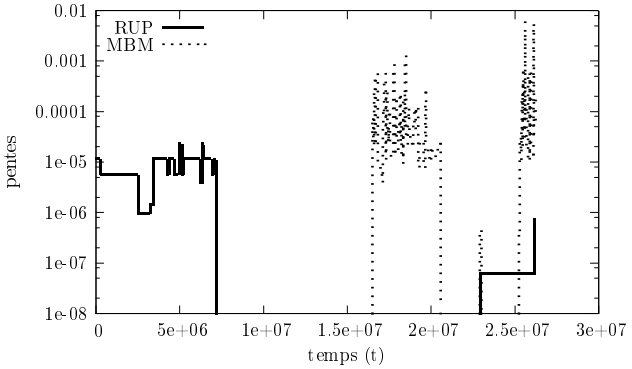


FIGURE 42 – Résultats du niveau 0 de l'algorithme 2 sur les sous-flux f^2_{RUP}et f^2_{MBM}.

De ces observations, on remarque que le second pic des arrivées de RUP est accompagné par de nombreuses occurrences des alarmes CSF, RUM et MBM.

Utilisons maintenant l'heuristique présentée dans ce chapitre avec $\theta'' = 0.01$. Ceci calcule l'ensemble des alarmes majeures $M = \{\text{AIS, LS, SSF, DS, RDI, URU, EBER, UT}\}$. La colonne $E_2^{\theta''}$ de la table 3.2 indique la répartition des alarmes entre les ensembles M et $\mathcal{I}$. Le nombre de motifs dans le graphe réduit est $|ser(f)| = 168$, ce qui fournit un graphe trop grand pour être entièrement représenté ici. Cependant, une observation rapide montre RUTM, CFF, HK, TIM et RI constituent l'ensemble des alarmes rares. En effet, elles apparaissent toutes au maximum 10 fois dans f^2. Les alarmes RUP, U et LOF peuvent également être considérées comme rares car elles se manifestent moins de 65 fois. Ceci reste très faible comparé à la taille du flux. Comme précédemment, nous nous focalisons sur l'occurrence des alarmes RUP et RUM.

La figure 43 représente un premier motif apparaissant au début de l'observation,

lorsque RUP se manifeste pour la première fois. La présence de RUP et RUM dans les motifs pertinents fait à nouveau ressortir le cas d'étude. Notons que CSF accompagne une fois encore les occurrences de RUP. C'est également le cas de NI et U, qui étaient des alarmes fréquentes pour E_1.

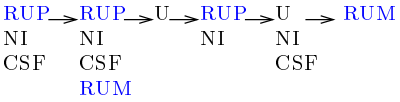


FIGURE 43 – Détail d'une première séquence de motifs pertinents pour f^2.

La figure 44 introduit une seconde suite de motifs, plus longue que la précédente. Elle apparaît presque à la fin de l'observation, peu avant la seconde occurrence de RUP. Cette séquence est particulièrement intéressante. En effet, toutes les alarmes rares, excepté RI, sont représentées dans ce motif. De plus, on peut remarquer que HK et CFF apparaissent à chaque fois ensemble. Il semble y avoir une corrélation entre ces dernières. Notons enfin la présence de CSF, NI et MBM.

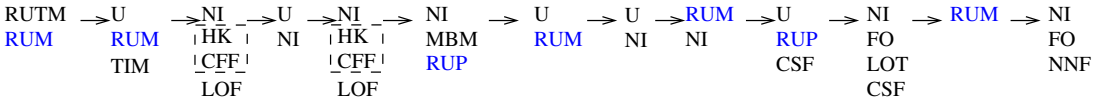


FIGURE 44 – Détail d'une seconde séquence de motifs pertinents pour f^2.

Concluons maintenant l'analyse de E_2.

La présence de RUP au début de l'observation indique qu'un élément du réseau ne fonctionne pas correctement et pourrait tomber en panne. La corrélation avec CSF montre de plus que l'élément subit des problèmes de communication. La figure 43 indique que lors de sa première arrivée, RUP n'est pas émise longtemps. De ce fait, nous pouvons conclure que le problème a été résolu.

Cependant, la seconde apparition de RUP, dans le motif représenté sur la figure 44, montre qu'un autre élément de réseau tombe en panne (ou l'élément ayant subi la panne précédente). Afin de réaliser une meilleure expertise du réseau, l'algorithme 2 à été utilisé sur le sous-flux f^2_{EBER}. Il est intéressant de remarquer que EBER n'apparaît qu'entre $2.1\,10^7$ et $2.6\,10^7$ s, ce qui correspond à la période durant laquelle RUM et RUP sont également émises. Cette corrélation indique qu'un problème affecte le démodulateur d'un laser.

Enfin, la figure 44 met en avant deux pathologies supplémentaires.

- On remarque que CFF et HK sont corrélées. L'alarme CFF indique que la température d'un élément dépasse le seuil autorisé. De la corrélation avec HK, nous déduisons que de nombreuses tentatives ont été réalisées afin de résoudre cette situation.
- Finalement, la présence combinée de MBM et CSF met en avant un problème de communication avec le contrôleur de carte.

Conclusion du chapitre

Dans ce chapitre, nous avons étudié le problème complexe de la corrélation des alarmes. Afin de permettre une analyse proactive, l'heuristique proposée est très légère aussi bien en temps de calcul qu'en espace mémoire et s'applique à n'importe quel réseau. Elle se déroule en plusieurs étapes. Dans un premier temps, les alarmes les plus fréquentes

sont définies. L'intérêt se porte alors sur la séquence de motifs d'alarmes non-fréquentes, dits *motifs pertinents*, extraite du flux analysé. À partir de cette séquence, les alarmes les moins présentes, appelées *alarmes rares*, sont mises en avant en tant que signal d'une menace sévère. Finalement, des graphes de dépendances de motifs pertinents conduisant aux alarmes rares sont construits.

Dans la partie résultats, nous avons analysé l'efficacité de notre heuristique sur les données d'un réseau réel préalablement étudié par l'équipe du service client d'Alcatel-Lucent Bell-labs. Nous avons montré qu'utiliser la méthode proposée permet de détecter rapidement les parties pertinentes du flux d'alarmes observé et d'en déduire une expertise fine.

Cependant, cette méthode n'est actuellement pas utilisée afin de traiter les données à la volée, or cela constitue un vif intérêt de la part des ingénieurs d'Alcatel-Lucent Bell-labs avec lesquels nous avons collaboré. Un travail intéressant consisterait à adapter cette technique en observant les flux des alarmes passées et courantes. On pourrait alors définir une analyse prédictive qui évaluerait en temps-réel la probabilité de réapparition des erreurs détectées. De plus, la méthode introduite ne peut traiter qu'un flux d'alarmes à la fois. Pour fournir une meilleure analyse et suivre l'évolution de la taille des réseaux, il serait également intéressant d'envisager une version générant une expertise automatique combinant les résultats de corrélation des flux de chaque élément.

Deuxième partie

Amélioration et contrôle des délais d'attente

Comment définir les paramètres d'un protocole afin d'éviter les congestions et réduire les délais d'attente des messages ? Quel est le délai maximal des messages d'un bout à l'autre d'un réseau sachant la capacité de traitement des routeurs et la vitesse d'arrivée des messages ? Telles sont, dans cette seconde partie, nos préoccupations.

Ainsi, dans le chapitre 4, nous présentons une étude concernant le risque de congestion des files d'attente (instabilité à cause de délais infini). Nous chercherons à obtenir des conditions sur les paramètres des réseaux évitant ce problème et diminuant le délai des messages dans les files d'entrée des routeurs. L'objectif consiste à réduire la quantité de paquets présents à chaque instant dans le système. Dans ce but, nous introduisons une nouvelle approche basée sur une modélisation des protocoles périodiques, que nous présentons sur le cas d'étude d'OSPF. Le but de cette modélisation est de permettre l'observation des échanges de messages entre routeurs et l'effet des modifications de certains paramètres sur ces échanges. Il s'agit également de réduire le risque de dangereuses synchronisations de messages qui pourraient conduire à la congestions du réseau. Prenant ces objectifs en considération, nous apportons une méthode affectant les paramètres du protocole OSPF pour s'approcher des valeurs optimales.

Enfin, dans le chapitre 5, nous introduisons une étude de calcul des délais pire-cas de messages d'un bout à l'autre du réseau lorsque la politique de traitement est à priorités fixes, en fonction de la vitesse des arrivées de messages et de la rapidité de traitement des routeurs. Nous apportons une méthode combinant l'utilisation d'une théorie développée pour ce domaine, appelée *network calculus*, et le maniement de programmes linéaires. Pour finir, nous évaluons les résultats de la technique développée à ceux calculés par les méthodes existantes. Nous montrons que les résultats sont plus précis que ceux obtenus jusqu'alors et qu'ils fournissent dans certains cas le délai pire-cas exact (pas d'approximation).

Chapitre 4

Calcul des paramètres optimaux d'un protocole : le cas d'étude d'OSPF

Sommaire

Notations du chapitre **78**
Introduction **79**
4.1 Modélisation et simulation du processus d'inondation **80**
4.1.1 Réseau de Petri temporisé 80
4.1.2 Modélisation du processus d'inondation 80
4.1.3 Validation du modèle 83
4.1.4 Mise en œuvre 84
4.2 Étude de la longueur de la période **85**
4.2.1 Cas d'un trafic fluide 85
4.2.2 Système congestionné 85
4.2.3 Cas limite 86
4.2.4 Condition suffisante de congestion 86
4.3 Calcul de décalages initiaux optimaux **88**
4.3.1 Modèle de contraintes et notations 88
4.3.2 Solution exacte par programmation linéaire 90
4.3.3 Algorithme glouton 91
4.3.4 Résultats de simulations avec décalages initiaux 93
Conclusion du chapitre **94**

Le travail présenté dans ce chapitre résulte de travaux effectués en collaboration avec Anne Bouillard et Claude Jard. Il fait l'objet d'un article [83] présenté lors de la conférence DCNET'13 en 2013.

Notations du chapitre

La table 4.1 liste les notations spécifiques à ce chapitre.

Symboles	Significations
	Réseau de Petri
$(Pl,Trans,Bwd,Fwd,M_0,\varphi)$	Un réseau de Petri
$Début_i$	Place initiale pour créer les messages LSA_is
$LSAenv_{i\to j}$	Place d'envoi LSA_i à R_j
$ACKenv_{i\to j}$	Place d'envoi un ACK de R_i à R_j
$LSArec_{j\to i}$	Place de réception d'un LSA provenant de R_j dans R_i
$ACKrec_{j\to i}$	Place de réception d'un ACK de R_j dans R_i
$Processeur$	Place représentant la ressource de calcul de R_i
$borne$	Place bornant le nombre de retransmission par période
b_i	Nombre de retransmissions possible par voisin de R_i
$Retransmission$	Place de retransmission d'un LSA
$Destruction$	Place de destruction d'un LSA
$N_{MR}(j)$	Nombre de messages reçus par R_j durant T_r
$N_{MT}(j)$	Nombre de messages traités par R_j durant T_r
	Heuristique
$[\delta_i, \Delta_i[$	Temps de séjour d'un message dans R_i
$T_t^{(i,j)}$	Temps de transmission entre R_i et R_j
$I_{i,j} = [\zeta_{i,j}, \psi_{i,j}[$	Intervalle de réception du premier LSA_i en R_j
$\zeta = (\zeta_{i,j})$	Matrice des valeurs $\zeta_{i,j}$
$\psi = (\psi_{i,j})$	Matrice des valeurs $\psi_{i,j}$
$D = (D_{i,j})$	$D_{i,j} = [\zeta_{i,j}, \gamma_{i,j}[$ avec $\gamma_{i,j} = \psi_{i,j} + \Delta_j$
$Q = (Q_i)$	Matrice des tailles de files d'attente maximale des routeurs
$b_{i,k,j}$ et B	$b_{i,k,j} = \gamma_{i,j} - \zeta_{k,j}$ et $B = \max_{i,k,j} b_{i,k,j}$
$C = (c_{i,k})$	$c_{i,k} = \max_{k\in\mathbb{N}_n} b_{i,k,j}$
$w(i,j)$	Poids de l'arc de R_i vers R_j
W	Poids d'un cycle hamiltonien

TABLE 4.1 – Listes des notations du chapitre 4.

Introduction

Bon nombre de protocoles de routage (dits de routage dynamique) travaillent dans un environnement dynamique dans lequel ils doivent constamment opérer des modifications. C'est notamment le cas des protocoles RIP [55] et OSPF [58]. Cette fonctionnalité est implémentée localement sur chaque routeur par une boucle générant des comportements réguliers. Ainsi, la *période de rafraîchissement* de la table de routage met à jour les chemins les plus courts. Le protocole bien connu OSPF, utilisé dans les réseaux internet, est facile à analyser. Nous l'avons utilisé précédemment et une définition partielle en est présentée dans la section 1.3. Rappelons qu'il s'agit d'un protocole à état de lien, générant fréquemment des messages Hello (vérifiant les connexions) et périodiquement des LSAs permettant la mise à jour des tables de routage. Il constituera le protocole d'étude de ce chapitre.

Un travail important à déjà été dédié au problème de stabilité des réseaux. Un problème classique de stabilité concerne par exemple la panne d'un lien pour laquelle il faudra que tous les routeurs du réseau convergent en temps fini vers la nouvelle topologie. Ce problème s'avère complexe lorsque le changement provient d'une surcharge d'un routeur, toujours possible avec l'extension OSPF-TE [39]. En effet, si la réponse à une congestion engendre l'envoi d'une plus grande quantité de messages, le problème devient rapidement critique. Néanmoins, Basu et Riecke mettent en évidence la robustesse d'OSPF-TE face à ce type de situation ([3]). Dans l'article [57], Moreau montre de plus que les flux d'information permettant la mise à jour de la topologie à un impact important sur les systèmes dynamiques. (Il propose ensuite des conditions assurant la convergence dans le cadre de systèmes multi-agents.)

Dans ce chapitre, nous étudions un problème associé à celui énoncé ci-dessus. En l'occurrence, nous nous intéressons au risque de congestion des files d'attente d'entrée des routeurs dû aux envois de messages LSAs et aux délais d'attente subis par les messages. En effet, nous croyons qu'il existe des situations pour lesquelles le comportement cyclique des protocoles de routage peut induire un effondrement des performances du réseau.

Principe de la méthode développée Les utilisations courantes du protocole OSPF définissent un cycle de rafraîchissement, très long, des tables de routage, appelé période d'inondation. Cependant, dans ce chapitre, nous considérons des réseaux OSPF dont la topologie change fréquemment. Dans ce genre de situation, il est intéressant d'augmenter la fréquence des périodes afin d'améliorer le temps de réponse aux changements. Dans ce chapitre, nous étudions l'effet du rapprochement des périodes sur les arrivées de messages aux routeurs et la taille de leurs files d'attente. Nous montrons des cas possibles de divergence et nous donnons une condition suffisante impliquant la surcharge d'un routeur.

Même lorsque les routeurs ne sont pas congestionnés par le trafic "classique" d'un protocole, l'accumulation des messages, que nous nommons synchronisation dangereuse, peut mettre en péril la santé du réseau. Ils deviennent en effet rapidement obsolètes. Les protocoles disposent de mécanismes retransmettant ces messages. C'est notamment le cas pour OSPF qui retransmet les LSAs trop âgés. Ceci génère un trafic plus dense que nous souhaitons éviter. Ainsi, nous discutons de mesures pouvant être prises afin de réduire les risques de synchronisation (et par conséquence les délais d'attente) en apportant des décalages entre les cycles de chaque routeur.

La section 4.1 définit la modélisation abstraite de la procédure d'inondation à l'aide d'un réseau de Petri. Nous validons ensuite ce modèle en comparant les résultats de simulation du réseau de Petri et ceux obtenus par l'émulation dans le cas de la topologie allemande de 17 nœuds (voir figure 4), page 19.

Dans la section 4.2, le modèle de réseau de Petri, représentant la topologie allemande, est simulé avec différents degrés de stress du réseau. Nous montrons notamment un cas présentant une accumulation de messages d'une période à l'autre. Nous fournissons finalement une condition suffisante de congestion.

Enfin, dans la section 4.3, nous présentons une méthode de calcul optimale des décalages initiaux (permettant de réduire le risque de synchronisation). Ce calcul est réalisé à l'aide d'un programme linéaire. Malheureusement, cette technique ne s'applique pas dans la majorité des cas, notamment pour la topologie allemande. Nous présentons alors une heuristique afin d'approcher une solution optimale. Pour finir, nous montrons que l'utilisation de temps de décalage permet une réduction efficace des accumulations de messages à l'entrée des routeurs.

4.1 Modélisation et simulation du processus d'inondation

Cette section introduit tout d'abord la définition d'un réseau du Petri temporisé, modèle théorique que nous utilisons afin de décrire la procédure d'inondation. Nous présentons ensuite la méthode de modélisation. Finalement, nous validons le modèle grâce à la comparaison de la simulation de ce dernier avec l'émulation du même réseau.

4.1.1 Réseau de Petri temporisé

Un réseau de Petri est un modèle standard permettant la modélisation de systèmes concurrents à évènements discrets. Le choix de ce modèle provient de l'observation de véritables traces OSPF dans lesquelles le temps de traitement varie très faiblement. Dans notre étude, nous utilisons la définition suivante du réseau de Petri :

Définition 4.1.1 (Réseau de Petri temporisé)**.** *Un réseau de Petri temporisé est un tuple (Pl, Trans, Bwd, Fwd, M_0, φ) dans lequel :*

- *Pl représente l'ensemble fini de places ;*
- *Trans l'ensemble fini de transitions ;*
- *$Bwd : P \times T \to \mathbb{N}$ la fonction d'incidence arrière, qui étant donné une place p_i et une transition t_j donne le poids de l'arc reliant p_i à t_j ;*
- *$Fwd : T \times P \to \mathbb{N}$ la fonction d'incidence avant, qui étant donné une transition t_j et une place p_i donne le poids de l'arc reliant t_j à p_i ;*
- *$M : P \to \mathbb{N}$ la fonction de marquage ;*
- *$\varphi : T \to \mathbb{Q}$ la fonction des contraintes temporelles des transitions.*

L'ensemble des places précédent (resp. suivant) une transition t_j est notée ${}^\bullet t_j$ (resp. $t_j{}^\bullet$). Les jetons progressent dans le réseau d'une place à une suivante en franchissant une transition par un processus appelé *tir*. Une transition t_j est dite *activée* si $\forall p_i \in {}^\bullet t_j, M(p_i) \geqslant Bwd(p_i, t_j)$. Le tir de t_j débute lorsque la transition s'active. Si t représente cet instant alors le processus de tir s'achève à la date $t + \varphi(t_j)$. À ce moment, le marquage des places évolue : $\forall p_i \in P, M(p_i) = M(p_i) + Fwd(t_j, p_i) - Bwd(p_i, t_j)$.

4.1.2 Modélisation du processus d'inondation

Présentons maintenant la modélisation du processus d'inondation d'un réseau. Rappelons d'abord les points importants de ce mécanisme (la section 1.3 le définit exhaustivement).

- L'inondation est un procédé répété toutes les T_r secondes ;

- à chaque période, le routeur R_i transmet sa vision du réseau par l'émission de LSA_i avec un décalage initial d_i. Autrement dit, il effectue des envois aux dates $\forall k \in \mathbb{N},\ d_i + kT_r + T_t$;
- si R_j débute le traitement de LSA_i à la date t, il termine à la date $t + T_t$, où T_t marque le temps nécessaire au traitement d'un message. À cette date :
  - il envoie un ACK à R_i ;
  - si R_j a recueilli de nouvelles informations grâce à LSA_i, il envoie un nouveau LSA à tous ses voisins (sauf R_i). Ce message est alors reçu à la date $t + T_t + T_e$, où T_e correspond au temps nécessaire pour envoyer un message.

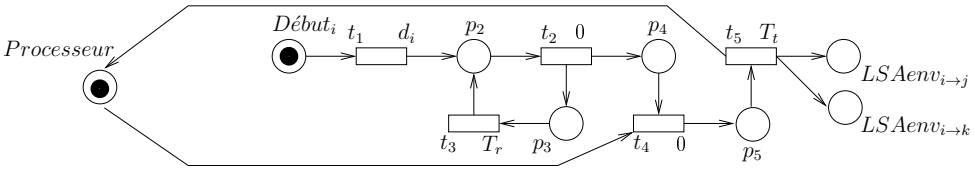


FIGURE 45 – Création de LSA_i dans R_i, représenté par un réseau de Petri.

Modélisation d'un routeur Modéliser le mécanisme d'inondation au niveau d'un routeur R_i à l'aide d'un réseau de Petri, nécessite quatre paramètres : d_i, T_r, T_t et $(b_i)_{i\in\{1,\dots,N_r\}}$ (que nous introduisons un peu plus loin). La création de LSA_i, le traitement d'un message reçu (LSA ou ACK) et la retransmission d'un LSA si nécessaire représentent les fonctionnalités d'un tel réseau. Notons que chaque routeur traite un seul message à la fois et de manière non-interruptible. Les paragraphes suivants définissent les différentes fonctionnalités énoncées.

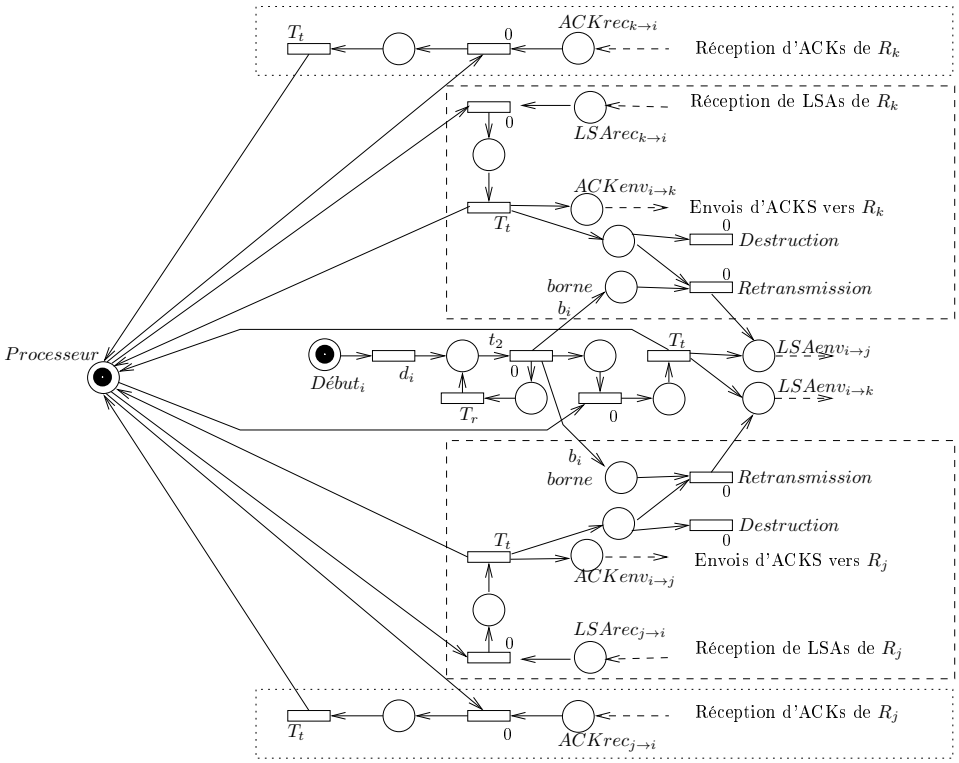


FIGURE 46 – Réseau de Petri d'un routeur R_i ayant deux voisins, R_j et R_k.

La place Processeur : elle contient initialement un jeton. Ce dernier représente la ressource de traitement du processeur permettant d'analyser des LSAs et ACKs. Il modélise le comportement de la file d'attente de R_i, où un seul message est traité à la fois. Quel que soit le type de message, le traitement suivant s'effectue : une transition instantanée est tirée pour réserver la ressource du routeur R_i. Notons que cette transition n'est tirable que si un message est en attente d'analyse. La transition suivante peut être tirée après un temps d'attente d'exactement T_t secondes. Ceci modélise le temps nécessaire au routeur R_i pour étudier n'importe quelle donnée. À ce moment, la ressource de la place *Processeur* est libérée, permettant le traitement d'un nouveau message. Ce mécanisme est illustré sur la figure 45 par le cycle : $Processeur, t_4, p_5, t_5, Processeur$.

Création d'un LSA : la figure 45 représente la partie du réseau de Petri chargée de la création de LSA_i, dans R_i, aux instants $\forall k \in \mathbb{N},\ d_i + kT_r$. Initialement, la place $Début_i$ contient un jeton. La première transition, t_1, peut être tirée après d_i secondes. Ceci génère un jeton dans la place p_2 à l'instant d_i pour la première fois. Ensuite, le cycle p_2, t_2, p_3, t_3 génère un jeton dans la place p_4 aux instants $\forall k \in \mathbb{N},\ d_i + kT_r$. Ce jeton est alors traité par le processus décrit précédemment et celui-ci libère un jeton pour tout $j \in \mathcal{V}(R_i)$ dans les places $LSAenv_{i\to j}$.

Réception d'un ACK : les rectangles en pointillés sur la figure 46 représentent le mécanisme de traitement d'un ACK. Un jeton dans la place $ACKrec_{j\to i}$ indique qu'un ACK a été envoyé par R_j à R_i. Ce message est alors traité suivant le procédé décrit précédemment (page 18) sans production d'aucun message.

Réception d'un LSA : les rectangles en tirets sur la figure 46 représentent le traitement d'un LSA. Un jeton dans la place $LSArec_{j\to i}$ indique qu'un LSA a été envoyé par R_j à R_i. Il est alors traité suivant le même procédé que les autres messages et cela génère un ACK dans la place $ACKenv_{i\to j}$. Cette tâche peut également générer un LSA transmis à tous les voisins de R_i excepté R_j (tirage de la transition *Retransmission*). Cette action modélise la retransmission d'un LSA lorsque de nouvelles informations ont été apprises par R_i. Si cette action ne se réalise pas, la transition *Destruction* est tirée afin d'exprimer le cas contraire et de supprimer le LSA traité. Dans le processus d'inondation, LSA_j est retransmis seulement lors de la première réception, afin d'assurer la convergence des bases de données avant la fin de la période d'inondation. Afin de modéliser ce principe avec notre réseau de Petri, nous bornons le nombre de retransmission par période. Ainsi, R_i peut retransmettre b_i LSAs provenant de R_j. Ceci est modélisé par la création de b_i jetons à chaque début de période dans les places appelées *borne*, placées devant les transitions *Retransmission*. Ces jetons sont introduits par la présence d'arcs pondérés par b_i placés entre la transition t_2 et les places *borne* sur la figure 46.

Modélisation d'un réseau La figure 46 représente le comportement complet d'un routeur R_i ayant deux voisins R_j et R_k. Considérons maintenant un réseau composé de trois routeurs R_i, R_j et R_k où R_i est connecté à R_j et R_k. Afin d'achever le réseau il faut représenter les routeurs R_j et R_k puis effectuer les connexions. Le routeur R_i est lié à R_j de la façon suivante : la place $LSAenv_{i\to j}$ (resp. $ACKenv_{i\to j}$) est liée à la place $LSArec_{i\to j}$ (resp. $ACKrec_{i\to j}$) en insérant une transition $LSA_{i\to j}$ (resp. $ACK_{i\to j}$) ayant un délai d'envoi T_e. La figure 47 représente ce réseau de trois routeurs.

Remarque 4.1.2. *Les routeurs R_j et R_k n'ayant qu'un seul voisin le mécanisme consistant à retransmettre un LSA reçu à ses voisins, excepté le voisin émetteur, n'a pas lieu d'être.*

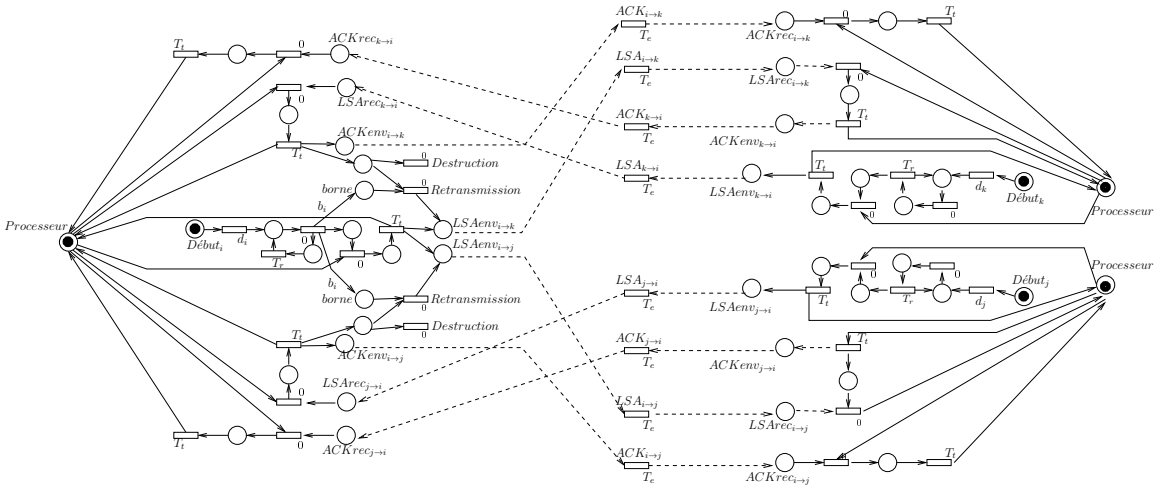


FIGURE 47 – Réseau de Petri d'une topologie de trois routeurs R_i, R_j et R_k.

4.1.3 Validation du modèle

Rappelons que les expériences réalisées dans ce chapitre se basent sur la topologie allemande (voir figure 4 page 19). Nous étudions le routeur R_8, ayant le plus grand nombre de voisins ($|\mathcal{V}(R_8)| = 6$).

Dans le réseau réel, les arrivées de messages LSAs et ACKs sont capturées par émulation grâce au logiciel *Quagga Routing Software* [61]. La figure 48 représente les arrivées de LSAs et ACKs au routeur R_8 durant 8000 s lorsque $T_r = 1800$ s.

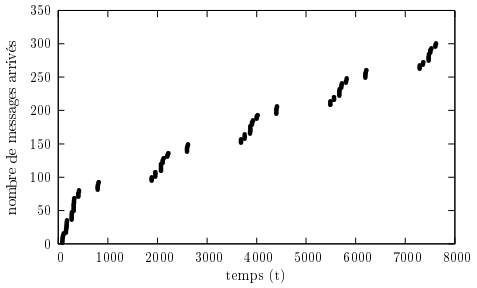


FIGURE 48 – Émulation des arrivées de messages au routeur R_8 lorsque $T_r = 1800$ s.

Les routeurs émulés disposent de processeurs de puissance 900 MHz. Un LSA comporte en moyenne 96 octets et un ACK 63, ce qui entraine un temps moyen de traitement de 0.8 μs pour un LSA et 0.5 μs pour un ACK. Finalement, le temps de transmission d'un LSA (resp. d'un ACK) est 96 ms (resp. 64 ms). Néanmoins, ces paramètres ne peuvent être appliqués dans le réseau de Petri. En effet, il modélise uniquement le processus d'inondation. Un routeur réel a évidemment bien plus de charge. Néanmoins, les paramètres du réseau de Petri sont définis afin de prendre en compte la charge complète des routeurs. En effet, le simulateur est calibré sur le cas classique $T_r = 1800$ s de sorte que la courbe d'arrivée de messages soit la même que celle obtenue par émulation du même réseau.

Les résultats présentés dans ce chapitre sont obtenus grâce au logiciel Renew [46] qui simule des réseaux de Petri temporisés. La figure 49 représente l'arrivée des messages lors de la simulation du réseau de Petri créé pour la topologie allemande lorsque $T_r = 1800$ s, $T_t = 15$ s et $T_e = 30$ s. Finalement, afin de correspondre à la quantité de messages émulés,

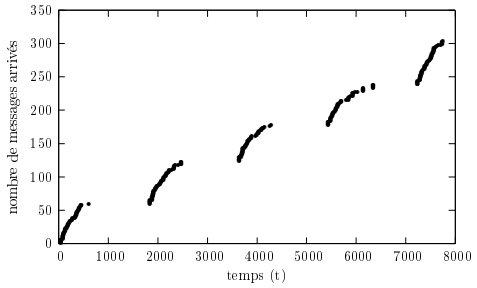


FIGURE 49 – Simulation par réseau de Petri des arrivées de messages au routeur R_8 lorsque $T_r = 1800\,\text{s}$.

le nombre de retransmissions maximales du routeur R_i est limité à $b_i = \dfrac{N_r - 1}{4|\mathcal{V}(R_i)|}$. Notons que pour l'instant les délais initiaux sont tous fixés à 0.

On remarque que les figures 48 et 49 se ressemblent beaucoup : les paramètres choisis ci-dessus définissent bien le comportement réel d'une procédure d'inondation. Ces deux courbes sont composées de périodes de 1800 s et montrent, sur chaque période, des arrivées soudaines durant approximativement 800 s. Puis, les arrivées de messages cessent jusqu'à la période suivante. Par conséquent, nous pouvons conclure que notre modèle abstrait capture efficacement le phénomène d'inondation.

Sauf mention contraire, à partir de maintenant nous attribuons aux paramètres $(b_i)_{i\in\{1,\dots,N_r\}}$, $(d_i)_{i\in\{1,\dots,N_r\}}$, T_r, T_t et T_e les valeurs énoncées ci-dessus.

4.1.4 Mise en œuvre

Afin d'analyser et de simuler aisément le modèle défini, nous avons implémenté un programme - codé en *C++* - générant automatiquement le réseau de Petri d'une topologie donnée. Il nécessite uniquement l'écriture d'un fichier texte très simple contenant les informations suivantes :

- le nombre de routeurs, N_r ;
- les paramètres T_r, T_t, T_e ;
- la liste des liens entre les routeurs.

Ce programme génère un second fichier texte, interprétable par Renew, construisant chaque routeur puis les liens entre routeurs suivant la méthodologie introduite dans la section 4.1.2. Le fichier est rédigé dans le langage de balisage *XML : Extensible Markup Language*. Il contient un petit ensemble de balises permettant de définir et de placer les éléments du réseau de Petri (places, transition, lien). Afin de permettre une observation aisée du réseau de Petri, les routeurs sont placés en cercle et sont définis selon la même structure.

Remarque 4.1.3. *Le programme est conçu afin de permettre l'expression du réseau de Petri dans différents formats par le biais de la modification d'un petit ensemble de lignes : les fonctions décrivant les éléments et la définition des espacements entre les éléments. Ceci permet de tester aisément notre modèle avec différents logiciels. Nous avons notamment réalisé ce travail afin d'utiliser le logiciel Roméo ([20]).*

Le logiciel Renew est configuré afin de récupérer des traces de simulations. Ces dernières contiennent pour chaque date de tirage le contenu des places correspondant à la file d'attente à l'entrée du (ou des) routeur(s) d'intérêt. Plus précisément, si l'on s'intéresse au routeur R_i, il s'agit pour tout $j \in \mathcal{V}(R_i)$ des places $LSArec_{i\to j}$ et $ACKrec_{i\to j}$. Renew produit un fichier texte par place observée, contenant l'historique de cette dernière.

Deux nouveaux programmes C++, gérant ces fichiers ont ainsi été développés. Le premier fournit les dates d'arrivées de message dans la file d'attente et le second présente l'évolution de la taille de la file.

4.2 Étude de la longueur de la période

Étudions l'effet de la taille de la période sur les arrivées de messages et la longueur de la file d'attente. Nous discutons tout d'abord du cas actuellement utilisé dans les réseaux, où $T_r = 1800\,\text{s}$. Puis, nous présentons un cas congestionné lorsque $T_r = 514\,\text{s}$. Finalement, nous étudions un cas à la limite de la congestion lorsque $T_r = 1000\,\text{s}$.

4.2.1 Cas d'un trafic fluide

Nous avons précédemment étudié les arrivées de messages lorsque $T_r = 1800\,\text{s}$ (voir la figure 49). La figure 50 représente la taille, soumise à de nombreuses fluctuations, de la file d'attente du routeur R_8 durant $10^5\,\text{s}$ (environ 1 journée) lorsque $T_r = 1800\,\text{s}$. Au début de chaque période, R_8 reçoit et traite des messages. Cependant, la vitesse d'arrivée est bien plus importante que la vitesse de traitement. Ceci implique un accroissement de la file d'attente. Puis, les envois cessent mais R_8 continue d'analyser les messages stockés. Ainsi, la taille de la file d'attente diminue jusqu'à être vide.

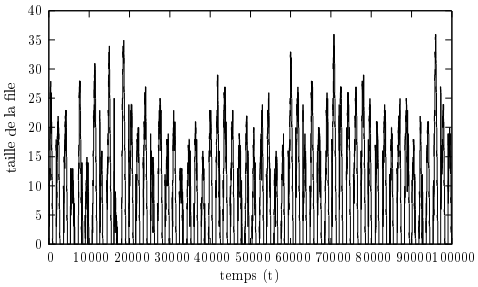


FIGURE 50 – Taille de la file d'attente du routeur R_8 lorsque $T_r = 1800\,\text{s}$.

4.2.2 Système congestionné

La figure 51 représente, à gauche, les arrivées de messages au routeur R_8 durant $8000\,\text{s}$ et, à droite, la taille de sa file d'attente durant $10^5\,\text{s}$, lorsque $T_r = 514\,\text{s}$. On peut remarquer que les messages arrivent sans discontinuer. En conséquence, R_8 n'a jamais la possibilité de vider sa file d'attente et la quantité de messages présents augmente en permanence.

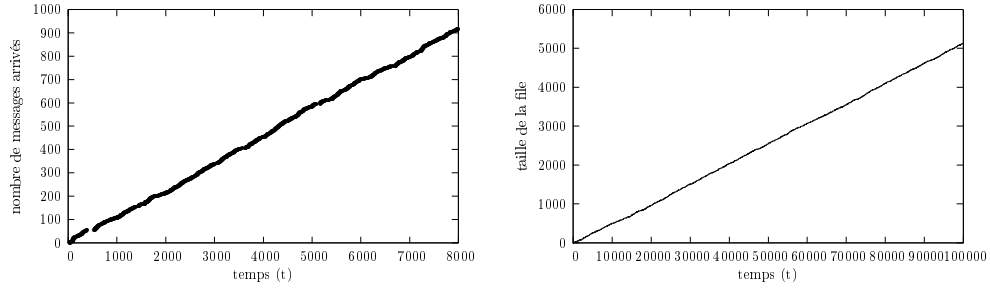


FIGURE 51 – Résultats de simulation de R_8 lorsque $T_r = 514\,\text{s}$: arrivées de messages (à gauche), taille de la file d'attente (à droite).

4.2.3 Cas limite

La figure 52 représente, à gauche, les arrivées de messages au routeur R_8 durant 8000 s et, à droite, la taille de sa file d'attente durant 10^5 s, lorsque $T_r = 1000$ s. Cette fois-ci les arrivées de messages d'une période ne sont pas immédiatement suivies des arrivées de la période suivante. La longueur de la période, suffisamment importante, permet à R_8 de traiter des messages de sa file d'attente avant le début de la période suivante. La courbe présentant les variations de la taille de la file de R_8 présente en effet des fluctuations symptomatiques de cet état. Cependant, la file d'attente ne se vide pas à la fin de chaque période. De plus, on peut remarquer que la quantité de messages présents dans la file à la fin des périodes varie. En conséquence, la stabilité de ce routeur ne peut être certifiée.

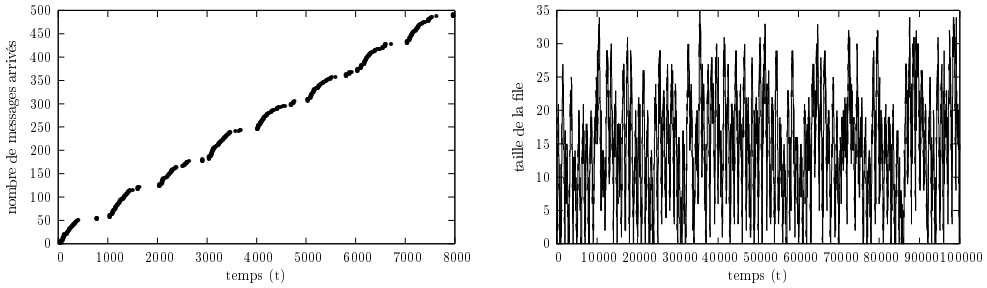


FIGURE 52 – Résultats de simulation de R_8 lorsque $T_r = 1000$ s : arrivées de messages (à gauche), taille de la file d'attente (à droite).

4.2.4 Condition suffisante de congestion

Dans cette partie, nous nous plaçons dans le pire cas, où chaque routeur acquiert de nouvelles informations de chaque autre routeur. Pour tout $i \in \{1, \ldots, N_r\}$ l'affectation $b_i = \dfrac{N_r - 1}{4|\mathcal{V}(R_i)|}$ n'est donc plus requise. Focalisons nous sur la quantité de messages reçus durant une période par un routeur.

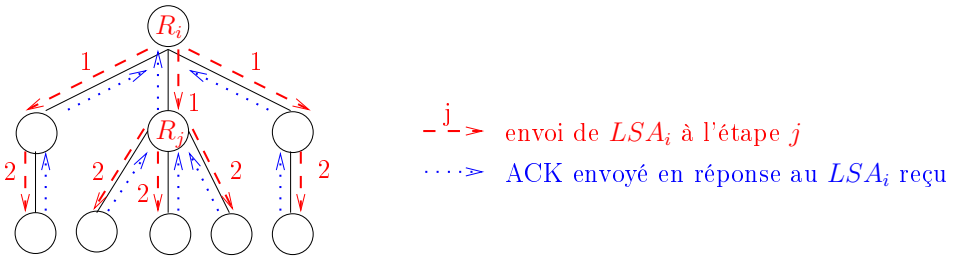


FIGURE 53 – Inondation de LSA_i : transmission des messages pour une topologie en arbre

Théorème 4.2.1. *Notons $N_{MR}(j)$ le nombre de messages reçus par le routeur R_j durant une période de longueur T_r. Alors,*

$$N_{MR}(j) \geqslant N_r|\mathcal{V}(R_j)|.$$

Démonstration. Concentrons-nous tout d'abord sur le cas des réseaux ayant une topologie en arbre et prouvons que l'inégalité est en fait une égalité. Comptons le nombre de

messages émis par R_i pour R_j. Dans le réseau de Petri, nous considérons deux sortes de messages : les LSAs et les ACKs. Si R_i constitue la racine de l'arbre, R_j reçoit LSA_i une seule fois : lors de son émission par R_i. Alors, R_j renvoie LSA_i à ses enfants. Il recevra alors un ACK de la part de chacun d'entre eux (voir la figure 53). Ainsi, le nombre de messages reçus pour l'inondation de LSA_i dans le réseau correspond au nombre de voisins de R_j. Considérons maintenant l'envoi de LSA_j dans le réseau. Le routeur R_j envoie ce LSA à tous ses voisins et reçoit en échange un ACK de chacun d'entre eux. En conséquence, R_j reçoit exactement $N_r|\mathcal{V}(R_j)|$ messages.

Étudions maintenant un réseau de topologie quelconque. Rappelons que la transmission de LSA_i se fait selon un arbre couvrant du graphe. En effet, (R_j, R_k) est un arc de l'arbre couvrant si R_k reçoit en premier le LSA_i transmis par R_j. Donc, dans le cas général, R_j reçoit au moins autant de LSA_i que lorsque la topologie représente l'arbre couvrant. Ceci fournit l'inégalité donnée. □

Le nombre de messages traités par R_j durant une période s'élève à $1 + N_{MR}(j)$. En effet, il traite les messages reçus ainsi que LSA_j. Notons $N_{MT}(j)$ ce nombre :

$$N_{MT}(j) = N_r|\mathcal{V}(R_j)| + 1.$$

Si un routeur n'a pas la capacité de traiter tous les messages de sa file avant la fin de la période, une congestion se forme à ce niveau du réseau. Ainsi, une congestion se produira si la borne suivante sur T_r est satisfaite.

Lemme 4.2.2. *Si $T_r < T_t N_{MT}(j)$, alors la file d'attente de R_j tend vers l'infini.*

Démonstration. La preuve est immédiate à partir du théorème 4.2.1. □

Exemple 4.2.3. *Considérons la topologie en arbre de la figure 53. Le théorème 4.2.1 assure que le nombre de messages reçus par R_j ($|\mathcal{V}(R_j)| = 4$) est $N_{MT}(j) = 9\times4+1 = 37$. Alors, si T_t est fixé à 15 s et si $T_r < 15 \times 37 = 555\,s$ le réseau est congestionné. La figure 54 représente la simulation du réseau de Petri présentant cette topologie durant 4.10^5 s lorsque les paramètres suivants : $T_r = 554\,s$, $T_t = 15\,s$, $T_e = 30\,s$. Elle présente parfaitement la congestion attendue : la taille de la file de R_j augmente effectivement d'une période à l'autre.*

Remarquons que la simulation ayant été réalisée avec la plus large période assurant la congestion, le routeur a le temps de traiter des messages une fois la réception achevée. En conséquence, la taille de la file varie beaucoup.

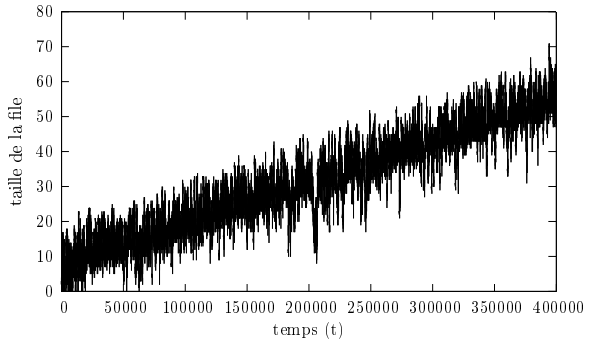


FIGURE 54 – Taille de la file d'attente de R_j pour une topologie en arbre, lorsque $T_r = 554\,s$.

4.3 Calcul de décalages initiaux optimaux

Dans la section 4.1.2, nous avons construit le modèle de réseau de Petri simulant le phénomène d'inondation du protocole OSPF. Cette modélisation nous aide à définir la longueur de la période permettant un maximum de rafraîchissement des informations de la topologie, sans impliquer de congestion. Maintenant, nous souhaitons affecter les décalages entre les périodes d'inondation de chaque routeur afin de minimiser la taille de leurs files d'attente et donc de réduire le temps d'attente des messages.

Nous avons tenté une première approche représentant les décalages initiaux par des paramètres du réseau de Petri. Une analyse de *model-checking à paramètres* a été réalisé afin de définir les paramètres de ce modèle. Ces expériences ont été mise en place grâce au logiciel Roméo ([20]). Cette étude calcule l'enveloppe convexe des valeurs de paramètres fournissant la solution optimale. Hélas, nous nous rendons compte que la trop grande complexité de ce problème ne permet pas une telle étude. Une deuxième approche fut alors tenté : effectuer une série de model-checking sur le réseau de Petri en faisant varier les valeurs des paramètres à chaque test, puis rechercher l'enveloppe convexe sur les résultats obtenus. Ces expériences sont réalisées grâce au logiciel DDD ([16]). Malheureusement, là encore, nous nous heurtons à une trop grande complexité.

Finalement, le problème est résolu en inférant des contraintes sur les paramètres, puis en définissant une méthode les calculant. Ce problème complexe est simplifié en adoptant le raisonnement suivant. Seuls les messages contribuant au processus d'inondation sont pris en compte : un LSA provenant de R_j n'est retransmis par R_i que s'il le reçoit pour la première fois. Ainsi, nous ne modélisons ni les ACKs, ni les LSAs arrivant après le premier LSA du même type (en chaque routeur).

4.3.1 Modèle de contraintes et notations

Notre objectif est de réaliser les inondations les plus proches possibles les unes des autres en minimisant les interactions d'une période sur l'autre. Nous dirons que deux périodes interagissent, si en au moins un routeur, le premier LSA provenant d'une période se retrouve dans une file d'attente en même temps qu'un LSA identique provenant d'une autre période.

Afin de résoudre ce problème, nous considérons encore une fois le graphe représentant la topologie du réseau $G = (V, E)$, où $V = \{R_1, \ldots, R_{N_r}\}$ et $E \subseteq V \times V$ est l'ensemble des liens entre routeurs. Si $(R_i, R_j) \in E$, alors $\mathrm{T}_t^{(i,j)}$ représente le temps de transmission entre R_i et R_j, et $\mathrm{T}_t^{(i,j)} = \infty$ si $(R_i, R_j) \notin E$. Le temps de séjour d'un message dans R_i, entre sa réception et son envoi, appartient à l'intervalle $[\delta_i, \Delta_i[$. Ce dernier est également défini pour la source du message.

Commençons par calculer $I_{i,j}$, l'intervalle d'arrivée d'un LSA émis par R_i pour R_j si l'inondation commence à la date 0. Si $i = j$, alors $I_{i,i} = [0, 0]$, sinon $I_{i,j} = [\zeta_{i,j}, \psi_{i,j}[$ où $\zeta_{i,j} = \min_{k \in \{1,\ldots,N_r\}} \zeta_{i,k} + \delta_k + \mathrm{T}_t^{(k,j)}$ et $\psi_{i,j} = \min_{k \in \{1,\ldots,N_r\}} \psi_{i,k} + \Delta_k + \mathrm{T}_t^{(k,j)}$.

Les valeurs $\zeta_{i,k} + \delta_k$ et $\psi_{i,k} + \Delta_k$ représentent respectivement la date minimum et maximum de départ de R_k d'un LSA émis par R_i. Nous effectuons le calcul de $\zeta_{i,j}$ et $\psi_{i,j}$, grâce à un algorithme de plus court chemin sur les graphes respectivement muni des longueurs d'arc $(\delta_i + \mathrm{T}_t^{(i,j)})$ et $(\Delta_i + \mathrm{T}_t^{(i,j)})$ et nous notons $\zeta = (\zeta_{i,j})$ et $\psi = (\psi_{i,j})$ les matrices des plus courts chemins. Elles peuvent par exemple être calculées en appliquant l'algorithme de Floyd-Warshall. Les messages émis par R_i sont reçus par R_j dans un intervalle de temps inclus dans $D_{i,j} = [\zeta_{i,j}, \gamma_{i,j}[= [\zeta_{i,j}, \psi_{i,j} + \Delta_j[$. La matrice de ces intervalles est notée $D = (D_{i,j})$.

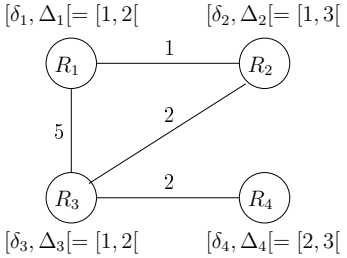


FIGURE 55 – Exemple simple de topologie à 4 routeurs.

Exemple 4.3.1 (Temps de séjour dans les routeurs). *La figure 55 représente une topologie simple ayant 4 routeurs. La matrice D est :*

$$D = \begin{pmatrix} [0,2[& [2,6[& [5,9[& [8,14[\\ [2,6[& [0,3[& [3,7[& [6,12[\\ [5,9[& [3,7[& [0,2[& [3,7[\\ [9,14[& [7,12[& [4,7[& [0,3[\end{pmatrix}.$$

Si l'inondation provenant du routeur R_i débute à la date d_i, son premier LSA reçu par R_j est au pire présent dans ce serveur durant l'intervalle $d_i + D_{i,j} = [d_i + \zeta_{i,j}, d_i + \gamma_{i,j}]$.

Ainsi, les interférences entre périodes d'inondation sont évitées en R_j, si les familles d'intervalles $(d_i + D_{i,j})_{i \in \{1,\dots,N_r\}}$ sont deux à deux disjointes et il n'y a aucune interférence si la condition suivante est respectée

$$\forall i, j, k \in \{1, \dots, N_r\},\ i \neq k \Rightarrow d_i + D_{i,j} \cap d_k + D_{k,j} = \emptyset.$$

Autrement dit, si :

$$\forall i, j, k \in \{1, \dots, N_r\},\ i \neq k \Rightarrow \begin{cases} d_i + \gamma_{i,j} \leq d_k + \zeta_{k,j} & \text{ou} \\ d_k + \gamma_{k,j} \leq d_i + \zeta_{i,j}. \end{cases}$$

Pour chaque triplet (i, j, k), les deux contraintes énoncées ci-dessus sont exclusives. En effet, comme $\gamma_{i,j} > \zeta_{i,j}$, si l'une des contraintes est vérifiée la deuxième ne peut l'être.

Si l'on considère plus d'une période, nous devons étudier les interférences entre la première et la deuxième inondation dans chaque routeur. En effet, s'il y a pas d'interférence entre ces deux-là, il n'y en aura jamais. La contrainte est alors la suivante.

$$\forall i, j, k \in \{1, \dots, N_r\},\ \begin{array}{ll} d_i + \gamma_{i,j} \leq d_k + \zeta_{k,j} & \text{ou} \\ d_k + \gamma_{k,j} \leq d_i + \zeta_{i,j} & \text{et} \\ d_k + \gamma_{k,j} \leq d_i + T_r + \zeta_{i,j} & \text{et} \\ d_i + \gamma_{i,j} \leq d_k + T_r + \zeta_{k,j} & \end{array} \tag{4.1}$$

Les deux cas sont illustrés sur la figure 56. Notons qu'en fonction des contraintes satisfaites, l'une des deux dernières inégalités est automatiquement satisfaite : par exemple si $d_k + \gamma_{k,j} \leq d_i + \zeta_{i,j}$, alors évidemment $d_k + \gamma_{k,j} \leq d_i + T_r + \zeta_{i,j}$.

Le problème à résoudre est de trouver les valeurs $(d_i)_{i \in \{1,\dots,N_r\}}$ de sorte que toutes les contraintes soient satisfaites et que T_r soit minimisé.

Théorème 4.3.2. *Soient $(\zeta_{i,j})_{i,j \in \{1,\dots,N_r\}}$, $(\gamma_{i,j})_{i,j \in \{1,\dots,N_r\}}$ et T_r. Le problème consistant à trouver des valeurs $(d_i)_{i \in \{1,\dots,N_r\}}$ satisfaisant les contraintes de l'équation (4.1) est NP-complet.*

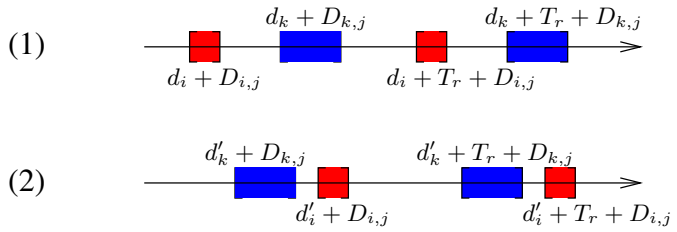


FIGURE 56 – Différents cas possible à partir des contraintes.

Démonstration. Manifestement, le problème réside dans NP. En effet, à partir de l'affectation des $(d_i)_{i\in\{1,\dots,N_r\}}$ et de la période T_r, il est possible de vérifier en temps polynomial si les contraintes sont satisfaites (il y a $O(N_r^3)$ contraintes).

Pour montrer que le problème est NP-difficile nous allons le réduire au problème du voyageur de commerce satisfaisant les inégalités triangulaires.

($\Rightarrow$) Supposons un graphe complet pondéré dont les poids, positifs, des arcs (de valeur $w(\ell,n)$) satisfont à l'inégalité triangulaire : pour tous nœuds ℓ, m, n, $w(\ell,m) + w(m,n) \leq w(\ell,n)$. Soit $\gamma_{i,j} = \max_{k\in\{1,\dots,N_r\}} w(k,i)$ et $\zeta_{i,j} = \gamma_{k,j} - w(i,k)$. Ces variables sont affectées de sorte que s'il existe j satisfaisant $d_i - d_k \geq \gamma_{k,j} - \zeta_{i,j}$, la même contrainte s'appliquera à tout j, car $\gamma_{k,j} - \zeta_{i,j} = w_{i,k}$.

Soit $(d_i)_{i\in\{1,\dots,N_r\}}$ et T_r une solution de notre problème. Ainsi, il existe un cycle hamiltonien de poids $W \leq T_r$ dans le graphe. En effet, supposons sans perte de généralité que $d_1 \leq d_2 \leq \cdots \leq d_{N_r}$. Ainsi, $w(1,2) + w(2,3) + \cdots + w(N_r,1) \leq (d_2 - d_1) + (d_3 - d_2) + \cdots + (d_1 - d_{N_r} + T_r) = T_r$.

($\Leftarrow$) Réciproquement, supposons qu'il existe un cycle hamiltonien de poids W, correspondant, sans perte de généralité, au cycle $1, 2, \dots, N_r$. Posons $d_1 = 0$ et $d_i = d_{i-1} + w(i-1,i)$. Chaque contrainte est respectée pour tout $(d_i)_{i\in\{1,\dots,N_r\}}$ et $T_r = W$ est une période possible : si $k > i$, $d_k - d_i = w(i,i+1) + \cdots + w(k-1,k) \geq w(i,k)$. De plus, $(d_i + W) - d_k = w(k,k+1) + \cdots + w(N_r,i) + \cdots + w(i-1,i) \geq w(k,i)$.

Ainsi, il existe un cycle hamiltonien, dans le plus défavorable des cas, d'une longueur T_r si et seulement si l'on peut trouver une solution au problème avec une période d'au plus T_r. Le problème est donc NP-complet. □

4.3.2 Solution exacte par programmation linéaire

Le problème étudié peut se résoudre à l'aide d'un programme linéaire utilisant des variables entières et non-entières. Un programme linéaire est constitué d'un ensemble de règles exprimant un problème d'optimisation et d'un objectif définissant l'optimisation à résoudre.

Ici, la partie difficile consiste à exprimer les contraintes suivantes

$$\begin{array}{l} d_i + \gamma_{i,j} \leq d_k + \zeta_{k,j} \quad \text{ou} \\ d_k + \gamma_{k,j} \leq d_i + \zeta_{i,j} \end{array}$$

dans un programme linéaire. C'est la raison pour laquelle on utilise des variables entières.

Tout d'abord, l'ensemble des contraintes peut être réécrit de la manière suivante :

$$d_k - d_i \geq b_{i,k,j} \qquad \text{ou} \qquad d_i - d_k \geq b_{k,i,j}$$

lorsque $b_{i,k,j} = \gamma_{i,j} - \zeta_{k,j}$. Posons $B = \max_{i,j,k} b_{i,k,j}$.

Lemme 4.3.3. *Il existe une solution de ce problème telle que pour tout $i \in \{1,\dots,N_r\}$, $d_i \in [0, BN_r]$.*

Démonstration. L'affectation $d_i = (i-1)B$ est une solution évidente. En effet, $\forall i < k$, $\forall j \in \{1, \ldots, N_r\}, d_k - d_i = (k-i)B \geq B \geq b_{i,k,j}$. De plus, $\forall i, k, j$, $d_k - d_i = (N_r - k + i)B \geq B \geq b_{i,k,j}$. □

Lemme 4.3.4. *Les ensembles de contraintes suivants sont équivalents.*

(i) $d_i, d_k \in [0, BN_r]$ *et* $(d_k - d_i \geq b_{i,k,j}$ *ou* $d_i - d_k \geq b_{k,i,j})$

(ii) $d_i, d_k \in [0, BN_r]$, $q \in \{0, 1\}$ *et* $d_k - d_i + (1-q)BN_r \geq b_{i,k,j}$ *et* $d_i - d_k + qBN_r \geq b_{k,i,j}$.

Démonstration. ($\Rightarrow$) Supposons que les contraintes (i) sont satisfaites. Alors,
- soit l'inéquation $d_k - d_i \geq b_{i,j,k}$ est vérifiée et donc les contraintes de (ii) sont satisfaites avec $q = 1$ (car nous avons $d_k - d_i \geq b_{i,j,k}$ et $d_i - d_k + BN_r \geq BN_r \geq b_{k,i,j}$) ;
- soit l'inéquation $d_i - d_k > b_{k,i,j}$ est vérifiée et les contraintes de (ii) sont satisfaites avec $q = 0$.

($\Leftarrow$) Supposons maintenant que les contraintes (ii) sont assurées. Si $q = 1$, alors, il est trivial que $d_k - d_i \geq b_{i,j,k}$ et si $q = 0$, alors $d_i - d_k \geq b_{k,i,j}$. Ce qui conclut la preuve. □

En conséquence, nous obtenons le programme linéaire suivant.

$$\text{Objectif : minimiser } T_r$$

$$\text{Contraintes : } \forall i, j, k \in \{1, \ldots, N_r\},\ i \neq k, 0 \leq d_i \leq BN_r,\ q_{i,k,j} \in \{0, 1\}$$

$$\begin{cases} d_k - d_i + (1 - q_{i,j,k})BN_r \geq b_{i,k,j} \\ d_i - d_k + q_{i,j,k}BN_r \geq b_{k,i,j} \\ d_k - d_i \leq T_r - \max_{j \in \mathbb{N}_{N_r}} b_{k,i,j} \end{cases}$$

Exemple 4.3.5. *Avec l'exemple simple de la figure 55, on obtient* $T_r = 28\,s$, *lorsque* $d_1 = 0\,s$, $d_2 = 21\,s$, $d_3 = 14\,s$ *et* $d_4 = 5\,s$.

Malheureusement, le calcul de la solution exacte a deux inconvénients. Tout d'abord, comme le problème est NP-difficile, le calcul des décalages initiaux pour de plus grands réseaux est trop long. De plus, si le programme linéaire calcule une période T_r et que la période cible est $T'_r > T_r$, il est plus efficace de modifier les valeurs $d_i - d_k$ en $(d_i - d_k)T'_r/T_r$. Ceci n'est hélas pas assuré par la solution trouvée. Dans le paragraphe suivant nous montrons comment calculer une solution prenant en compte ces contraintes supplémentaires.

4.3.3 Algorithme glouton

Simplifions le problème en observant uniquement les plus fortes contraintes. Posons $c_{i,k} = \max_{k \in \mathbb{N}_{N_r}} b_{i,k,j}$. Les contraintes deviennent :

$$c_{i,k} \leq d_k - d_i \leq T_r - c_{k,i} \qquad \text{ou} \qquad c_{k,i} \leq d_i - d_k \leq T_r - c_{i,k}. \tag{4.2}$$

Lemme 4.3.6. *Si* $(d_i)_{i \in \{1,\ldots,N_r\}}$ *est une solution des contraintes de l'équation. (4.2) avec une période* T_r, *alors pour une période* $T'_r > T_r$, *une solution du même problème est* $(\frac{T'_r}{T_r} d_i)$.

Démonstration. Si $c_{i,k} \leq d_k - d_i \leq T_r - c_{k,i}$, alors comme $\frac{T'_r}{T_r} \geq 1$, on a $\frac{T'_r}{T_r}(d_k - d_i) \geq d_k - d_i \geq c_{i,k}$. Deuxièmement, $\frac{T'_r}{T_r}(d_k - d_i) = \frac{T'_r}{T_r}(T_r - c_{k,i}) = T'_r - \frac{T'_r}{T_r} c_{k,i} \leq T'_r - c_{i,k}$. □

Algorithme 3 : Calcul de décalages initiaux.

Données : $c_{i,j}$.
Résultats : $d_1, \ldots, d_{N_r}$, T_r.

```
1  Début
2      U ← ∅ ;
3      O ← {1, ..., N_r};
4      Pour tous i ∈ O faire d_i ← 0;
5      Tant que O ≠ ∅ faire
6          o ← Argmin_{i∈O} d_i;
7          O ← O \ {o};
8          Pour tous i ∈ O faire d_i ← max(d_i, d_o + c_{o,i});
9          Pour tous i ∈ U faire T_r ← max(T_r, d_o − d_i + c_{o,i});
10         U ← U ∪ {o};
11 fin
```

La résolution de ces contraintes reste un problème NP-complet (la preuve du théorème 4.3.2 est en effet toujours valide). L'algorithme 3, de type glouton, peut être utilisé afin de définir les décalages initiaux et une longueur de période. À chaque étape, l'algorithme affecte un décalage initial en choisissant la plus petite valeur possible satisfaisant aux contraintes et en tenant compte des décalages initiaux précédemment définis.

Lemme 4.3.7. *À chaque étape de l'algorithme, les contraintes (4.2) telles que $i, k \in U$ sont satisfaites.*

Démonstration. Montrons ce résultat par récurrence. Lorsque $U = \emptyset$ ou $|U| = 1$, le résultat est trivial puisque aucune contrainte n'est impliquée. Supposons que l'affirmation est vérifiée pour U et posons o le prochain élément ajouté à U par l'algorithme. D'après la ligne 8, nous savons que $d_o \geq \max_{i \in U} d_i + c_{i,o}$. Alors, pour tout $i \in U$, $d_o - d_i \geq c_{i,o}$. D'après la ligne 9, pour tout $i \in U$, $T_r \geq d_o - d_i + c_{o,i}$, alors $d_o - d_i \leq T_r - c_{o,i}$. Donc les contraintes impliquant o sont satisfaites. Finalement, comme T_r peut uniquement augmenter, si les contraintes entre i et j, $i, j \in U$ sont satisfaites à une étape de l'algorithme, elles reste satisfaites pour les étapes suivantes. □

Exemple 4.3.8 (Application de l'algorithme 3). *Sur l'exemple simple de la figure 55, nous avons*

$$C = (c_{i,j}) = \begin{pmatrix} 0 & 8 & 11 & 14 \\ 6 & 0 & 9 & 12 \\ 9 & 7 & 0 & 7 \\ 14 & 12 & 9 & 0 \end{pmatrix}.$$

Au début de l'algorithme, tous les décalages sont initialisés à 0 ($\forall i \in \{1,2,3,4\}$ $d_i = 0$). Ainsi, la première évaluation de la ligne 6 peut choisir n'importe quel décalage initial. Supposons que R_1 est choisi en premier. Les valeurs sont alors mises à jour à $d_1 = 0\,s$, $d_2 = \max(0, d_1 + c_{1,2}) = 8\,s$, $d_3 = 11\,s$ et $d_4 = 14\,s$; $T_r = 0\,s$. Ensuite, R_2 est choisi et l'on obtient les mises à jour suivantes $d_3 = \max(d_3, d_2 + c_{2,3}) = 17\,s$ et $d_4 = 20\,s$; $T_r = \max(T_r, d_2 - d_1 + c_{2,1}) = 14\,s$. Ceci donne finalement, $d_1 = 0\,s$, $d_2 = 8\,s$, $d_3 = 17\,s$, $d_4 = 24\,s$ et $T_r = 38\,s$.

Ce problème peut également être résolu en utilisant un programme linéaire (de variables entières) en remplaçant les variables $q_{i,k.j}$ du programme linéaire décrit au paragraphe précédent par $q_{i,k}$ (le paramètre j est oublié). Ceci conduit aux mêmes contraintes

que l'équation (4.2). Dans ce cas, nous trouvons $T_r = 36\,\mathrm{s}$, avec $d_1 = 0\,\mathrm{s}$, $d_2 = 30\,\mathrm{s}$, $d_3 = 11\,\mathrm{s}$ et $d_4 = 18\,\mathrm{s}$. Sur cet exemple, notre heuristique est donc proche de la solution optimale.

Dans le lemme suivant, nous supposons que la période cible a une longueur $T'_r < T_r$. Autrement dit, l'algorithme ne trouve pas de solution qui assure qu'un seul message soit présent dans la file d'attente de chaque routeur à tout instant. Nous supposons également que le temps de séjour d'un message ne dépend pas de la taille de la file.

Lemme 4.3.9. *Soient une période de longueur T_r et $(d_i)_{i\in\{1,\dots,N_r\}}$ une solution pour le calcul des décalages initiaux dans ce cas. Si la période est modifiée à $T'_r < T_r$, la même affectation assure qu'il n'existe pas plus de $\lceil \frac{T_r}{T'_r} \rceil$ messages présents dans la file d'attente de chaque routeur.*

Démonstration. Soit $\lceil \frac{T_r}{T'_r} \rceil = q$ et numérotons m_i^j le $j^{ième}$ message provenant du routeur R_i. Pour $\ell \in \{0, \dots, q-1\}$, dans chaque routeur, il ne peut y avoir simultanément plusieurs messages de l'ensemble $(m_i^{kq+\ell})_{k\in\mathbb{N}, i\in\mathbb{N}_{N_r}}$, car $qT'_r \geq T_r$. En conclusion, il ne peut y avoir plus de q messages dans un routeur. □

4.3.4 Résultats de simulations avec décalages initiaux

Dans cette section nous présentons les résultats de simulation du réseau de Petri modélisant la topologie allemande de 17 nœuds lorsque les décalages initiaux sont calculés par l'algorithme 3 et que $T_r = 1800\,\mathrm{s}$. Nous discuterons également des résultats obtenus dans le cas limite : $T_r = 1000\,\mathrm{s}$.

Définition des paramètres de l'algorithme 3

Avant de pouvoir effectuer des simulations du modèle, il nous faut définir les temps de transmission et de séjour à utiliser dans l'algorithme 3.

- Le temps de transmission a déjà été fixé dans le réseau de Petri (section 4.1.3) : $\forall (i,j) \in E,\ \mathrm{T}_t^{(i,j)} = T_e = 30\,\mathrm{s}$.
- Pour chaque routeur R_i, le temps de séjour est au minimum égal au temps de traitement. Donc $\forall i \in \mathcal{V}(R_i),\ \delta_i = T_t = 15\,\mathrm{s}$. Il s'agit du cas où la file d'attente est vide. Le temps de séjour maximal est extrait d'une longue simulation (environ 3 jours, soit $3.10^5\,\mathrm{s}$) du réseau de Petri lorsque tous les décalages initiaux sont nuls. Durant cette simulation, la taille maximale de la file de R_i est $\mathcal{Q}_i$. Alors le temps de séjour maximal est $\Delta_i = \mathcal{Q}_i T_t$.

Remarque 4.3.10. *Calculer le temps de séjour par ce biais permet de prendre en compte tous les messages circulant dans le réseau durant le processus d'inondation et pas uniquement le premier LSA émis par chaque routeur.*

Listons la taille maximale de la file d'attente de chaque routeur :

$$\mathcal{Q} = (7, 8, 13, 2, 2, 17, 8, 37, 4, 5, 13, 2, 2, 3, 13, 6, 2).$$

Simulation lorsque $T_r = 1800\,\mathrm{s}$

L'algorithme 3 calcule les décalages initiaux suivants :

$$d = (0, 105, 1200, 810, 75, 255, 420, 1335, 1035, 1080, 1155, 1530, 630, 330, 780, 330, 1680).$$

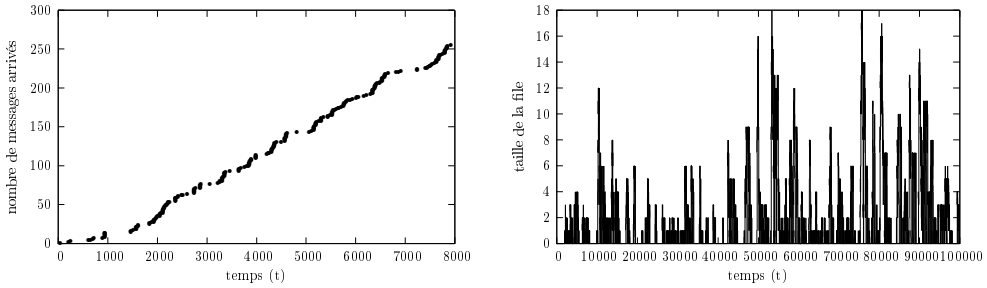


FIGURE 57 – Résultats de simulation de R_8 lorsque $T_r = 1800\,\text{s}$ avec décalages initiaux : arrivées de messages (à gauche), taille de la file d'attente (à droite).

La figure 57 représente l'évolution de la taille de la file d'attente du routeur R_8 lorsque $T_r = 1800\,\text{s}$ et que les décalages sont ceux calculés précédemment. La taille maximale est maintenant $Max_8 = 25$. Ceci apporte une amélioration importante. En effet, sans les décalages on avait obtenu $Max_8 = 37$. De plus, la taille de la file est désormais majoritairement inférieure à 10.

Simulation lorsque $T_r = 1000\,\text{s}$

La figure 58 représente les arrivées de messages ainsi que la taille de la file d'attente de R_8 durant une simulation de $10^5\,\text{s}$ lorsque les décalages initiaux sont calculés par l'heuristique si $T_r = 1000\,\text{s}$. Ici, l'étalement des messages sur la période complète, n'améliore pas la taille de la file d'attente de ce routeur.

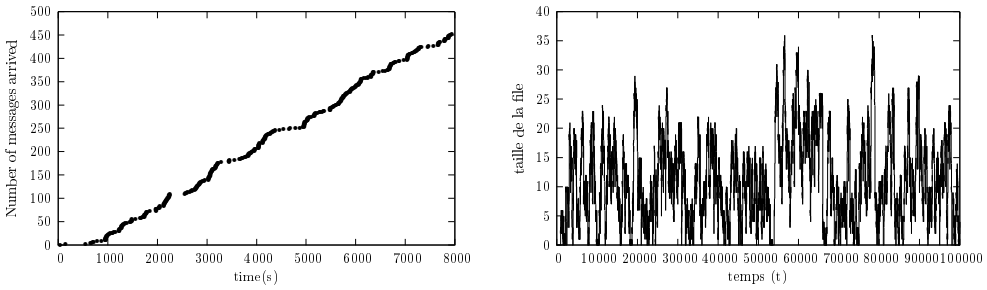


FIGURE 58 – Résultats de simulation de R_8 lorsque $T_r = 1000\,\text{s}$ avec décalages initiaux : arrivées de messages (à gauche), taille de la file d'attente (à droite).

Rappelons que $T_r = 1000\,\text{s}$ correspond au cas limite entre la congestion et le fonctionnement normal. Ainsi, la longueur de période est définie de sorte que le routeur le plus chargé, R_8, ait juste le temps de traiter tous les messages d'une période avant le début de la suivante. En conséquence, décaler les envois n'a pas d'intérêt pour ce routeur.

Conclusion du chapitre

Dans ce chapitre, nous avons étudié l'impact de la modification des valeurs de paramètres de protocoles périodiques sur les risques de congestion des files d'attente et les délais de messages. L'objectif consistait à proposer une méthode affectant ces paramètres afin de diminuer la longueur du processus périodique observé tout en maintenant la stabilité du protocole.

Nous avons tout d'abord présenté une méthode de simulation du processus d'inondation du protocole d'étude, OSPF, afin de mieux comprendre son fonctionnement. Une abstraction du processus a été apporté en définissant un réseau de Petri. Les simulations ont montré que ce modèle correspond effectivement au processus réel. Nous avons ensuite analysé l'effet de la longueur de la période d'inondation sur la congestion des files d'attente et nous avons présenté une condition suffisante assurant cet état.

Dans un second temps, nous avons introduit un algorithme glouton définissant les décalages initiaux afin d'éviter le risque de synchronisations dangereuses. Sur un exemple, nous avons montré que le résultat obtenu s'approche de la solution optimale. Cette méthode a également prouvé son efficacité sur l'exemple classique (T_r=1800 s) en affichant une réduction significative de la taille de la file d'attente du routeur le plus chargé.

Néanmoins, à l'heure actuelle, le risque d'instabilité du protocole OSPF reste très faible : la convergence est toujours assurée. Ce n'est cependant pas le cas de tous les protocoles périodiques. On peut notamment citer le protocole de routage inter systèmes autonomes BGP (*Border Gateway Protocol* [77]). En effet les problèmes d'instabilités de ce dernier sont toujours à l'étude ([47, 78]). En particulier, des erreurs de configuration ou des variations fréquentes de la topologie du réseau gênent la convergence du protocole et résultent en une surconsommation des ressources du réseau. Il serait donc intéressant d'observer le résultat de notre méthode appliquée à BGP.

Chapitre 5

Calcul de bornes de performance déterministe avec priorités fixes

Sommaire

Notations du chapitre 98
Introduction 99
5.1 Le *network calculus* 100
5.1.1 Flux et contrôle de flux 101
5.1.2 Bornes caractéristiques 106
5.1.3 Contraintes dans les réseaux complexes 107
5.2 État de l'art 109
5.2.1 Analyse de flux séparés 109
5.2.2 Algorithme de programmation linéaire (LP) 111
5.2.3 Priorités fixes 112
5.3 Étude du délai maximal : politique de service à priorités fixes 113
5.3.1 Modèle du réseau 113
5.3.2 Condition nécessaire et suffisante pour délai borné 115
5.3.3 Approche par programmation linéaire 116
5.4 Exactitude des bornes calculées 124
5.4.1 Étude complète d'un exemple 124
5.4.2 Ajout de contraintes supplémentaires 125
5.4.3 Calcul de bornes exactes 125
5.4.4 Borne inférieure du pire délai 127
5.5 Résultats et comparaison aux méthodes existantes 127
5.5.1 Mise en œuvre 128
5.5.2 Étude d'un premier type de réseau 128
5.5.3 Étude d'une seconde topologie 129
Conclusion du chapitre 130

Le travail présenté dans ce chapitre résulte de travaux effectués en collaboration avec Anne Bouillard, et fait l'objet d'un article [84] présenté lors de la conférence Valuetools'11 en 2011.

Notations du chapitre

La table 5.1 regroupe les notations utilisées dans ce chapitre.

Symboles	Significations
	Description de la topologie
N_f	Nombre de flux dans le réseau
$\mathcal{F}$	Ensemble des fonctions croissantes et continues à gauche
F^{in}	Fonction des arrivées cumulées d'un routeur R
F^{out}	Fonction des départs cumulés d'un routeur R
$prev_i(j)$	Routeur traversé par f_i avant R_j
$next(j)$	Routeur traversé après R_j
$Fl(j)$	Ensemble des flux traversant R_j
$maxFl(j)$	Flux de priorité minimale traversant R_j
$F_i^{(j)}$	Flux des arrivées de f_i en R_j
$last_i$	Dernier routeur traversé par f_i
	Contraintes et bornes
α et α'	Courbes d'arrivée et courbe de sortie
ρ	Arrivée à long terme
σ	Nombre de messages arrivant simultanément
β	Courbes de service
r	Taux de service
L	Latence maximale avant début du service d'un routeur
$\mathcal{S}_{\mathcal{T}}(\beta)$	Ensemble des trajectoires admettant β de type $\mathcal{T}$
d_p	Délai pur
$d(t)$ et D_{max}	Délai d'un donnée entrant à l'instant t et délai maximal
$start(t)$	Début de période chargée incluant t
$hDev$	Déviation horizontale
	Contraintes linéaires
λ	Collection finie de programmes linéaires
opt_λ	Valeur optimale des programmes linéaires de λ
t_u	Date d'entrée de la donnée d'intérêt dans le réseau
t_j	Date de début de période chargée de R_j
$path_i$ et ℓ_i	Chemin de f_i de longueur ℓ_i
$c_i^{(j)}$	Dates intermédiaires
C	Capacité d'un routeur
V_t et V_f	Ensemble des variables temporelles et fonctionnelles
g_i	Valeurs extraites d'un flux
$H, \tilde{H}, L, \tilde{L}$	Des charges du réseau

TABLE 5.1 – Listes des notations du chapitre 5.

Introduction

Un réseau est qualifié de *critique* lorsque l'occurrence d'une panne peut avoir des conséquences dramatiques : personnes blessées, dégâts matériels, dommages environnementaux, etc. Aujourd'hui, de plus en plus de dispositifs informatiques sont composés de réseaux critiques. En conséquence, l'évaluation de leurs performances pire-cas devient un enjeu capital.

La théorie du network calculus fut développée afin de répondre à ce type de besoins. Elle calcule des bornes de performances déterministes tels que les délais ou les charges des réseaux de communication. Le network calculus s'applique à la certification de réseaux (particulièrement en avionique) et son utilisation est à l'étude pour les réseaux embarqués. Voici quelques exemples d'application.

- Le réseau avionique AFDX (*Avionics Full DupleX switched ethernet*) est un réseau Ethernet redondant et fiabilisé qui sert de support aux communications internes des avions A380. Le network calculus a été appliqué à ce réseau pour garantir les délais d'attentes. La thèse [23] a bien amélioré les bornes calculées en prenant en compte les contraintes physiques de ce réseau.
- Le réseaux Switched Ethernet est un réseau de type Ethernet qui s'organise autour de communicateurs ([21]).
- Flex-Ray et CAN sont des réseaux de communication embarqués dans les automobiles (resp. [15] et [31]).

Le network calculus utilise des courbes, afin de paramétrer le réseau. Rappelons, qu'une *courbe d'arrivée* borne la quantité maximale de données pouvant arriver durant un intervalle de temps quelconque. De même, une *courbe de service* définit une garantie sur la quantité minimale de données traitées par un routeur. Le network calculus introduit une théorie élégante maniant les opérateurs de l'algèbre min-plus.

Hélas, de récentes études ont montré les limites de cette théorie. Notamment, l'application directe de ces résultats conduit à des bornes très pessimistes. Également, Schmitt et al. [67] exhibent un phénomène connu par sa dénomination anglaise *Pay Multiplexing Only Once (PMOO)*. Une méthode prenant en compte PMOO présente en général une amélioration significative. Malheureusement, dans certain cas, une méthode ne prenant pas PMOO en considération peut afficher de meilleurs résultats.

La programmation linéaire est une méthode mathématique visant à résoudre un problème d'optimisation. Le problème se présente sous la forme de contraintes linéaires. Une fonction particulière du programme est appelée *l'objectif*. C'est cette dernière qu'un solveur prend en compte pour rechercher la solution optimale au problème. Cette méthode a été utilisée par Bouillard et al., dans [9], afin de calculer les bornes exactes à l'aide du network calculus. Les auteurs montrent que dans le cas acyclique le problème est NP-difficile mais qu'il se réduit à une complexité polynomiale pour les réseaux en *tandem* (réseaux dont le graphe orienté réduit est un chemin orienté sans raccourci).

D'autres études se sont concentrées sur une politique de service particulière. On peut notamment citer les travaux de Lenzini et al.[51] qui traitent le cas de la politique *FIFO : premier entré, premier servi*, dans les réseaux en tandem et les réseaux *sink-tree* (réseau dans lequel un flux principal traverse un ensemble de routeurs, il peut être rejoint par d'autres flux qui l'accompagnent alors jusqu'à sa sortie du réseau).

Dans ce chapitre, nous nous intéressons aux réseaux dont les routeurs sont soumis à une politique de service à priorités fixes. Plusieurs études se sont déjà focalisées sur ce cas. Thiele et al. [70] présentent une méthode utilisant une variante du network calculus, appelée *RTC : Real-Time Calculus*. Afin d'atteindre de meilleurs résultats que le network calculus, la méthode RTC utilise des contraintes maximale et minimale sur les arrivées

et les services. Cette politique est également étudiée dans l'article [31]. L'objectif est de dimensionner et de paramétrer le réseau de communication automobile CAN (*Controller Area Network*). L'application du network calculus est améliorée grâce aux résultats de simulations définissant le comportement des tâches temporelles souples (dont les délais impartis ne doivent pas être obligatoirement respectés). Hélas, cette étude ne prend en compte qu'un ensemble restreint de topologies. Finalement, Saidane et al. [65] utilisent la théorie des trajectoires afin d'analyser les politiques à priorités fixes, mais ceci n'est pas l'objet de ce chapitre.

Principe de la méthode développée Dans ce chapitre nous proposons une méthode efficace calculant une borne maximale du délai d'un flux dans des réseaux de topologie quelconque disposant d'une politique de service à priorités fixes, en étendant les travaux présentés dans [9]. Nous montrons les limites de cette méthode par un ensemble d'exemples, puis nous l'améliorons en la combinant avec une technique existante. Nous présentons notamment deux types de réseaux pour lesquels notre méthode atteint exactement le délai pire-cas.

Le début du chapitre (section 5.1) décrit formellement la théorie du network calculus. Nous introduisons ainsi les opérateurs de convolution et de déconvolution dans l'espace (min,+), puis les contraintes sur les arrivées de données des flux et les services des routeurs. Enfin, nous introduisons les calculs permettant de déterminer les contraintes d'arrivée d'un flux en un routeur quelconque ainsi que le service résiduel d'un routeur après le traitement d'un certain nombre de flux.

Dans la section 5.2 nous établissons l'état de l'art des techniques calculant des bornes maximales du délai pire-cas. Nous expliquons comment les méthodes, présentées dans [67] et [70], utilisent les principes du network calculus pour déterminer ces bornes. Nous présentons également la technique qui calcule une borne maximale du délai en traduisant les contraintes du network calculus en un programme linéaire, présenté dans l'article [9].

La partie 5.3 développe le cœur de notre travail dans ce domaine. Nous détaillons la méthode utilisée pour encoder les priorités en un routeur et définissons les contraintes linéaires nécessaires, tout d'abord pour les topologies en arbre, puis pour une topologie générale.

Dans la section 5.4 nous étudions les limites de la méthode proposée et nous apportons une amélioration. Nous présentons alors deux cas dans lesquels notre méthode calcule exactement le délai maximal.

Finalement, la section 5.5 présente les résultats des différentes méthodes introduites sur un ensemble de cas.

5.1 Le *network calculus*

Introduisons les notions de flux, de courbes d'arrivée et de service, constituant le cœur de la théorie du network calculus, ainsi que les bornes de performance usuellement étudiées à l'aide de cette théorie : la charge d'un réseau et le délai subi par les données d'un flux. Commençons par définir l'ensemble des fonctions étudiées ainsi que les opérateurs de convolution et de déconvolution.

Soit $\mathcal{F}$ l'ensemble de fonctions définies par : $\mathcal{F} = \{h : \mathbb{R}_+ \to \mathbb{R}_+ \cup \{+\infty\} \mid h$ est croissante, continue à gauche et $h(0) = 0\}$. Dans la suite, nous supposerons toujours que les fonctions étudiées appartiennent à cet ensemble. Les opérateurs de convolution et de déconvolution sont présentés dans le semi-anneau ($\mathcal{F}$, min, $\otimes$), où min est la fonction minimum et $\otimes$ l'opérateur de convolution.

Définition 5.1.1 (Convolution). *Soit h_1, $h_2 \in \mathcal{F}$ alors la convolution de h_1 et h_2 est :*

$$\forall t \in \mathbb{R}_+ \ (h_1 \otimes h_2)(t) = \inf_{0 \leqslant s \leqslant t} \{h_1(s) + h_2(t-s)\}.$$

Nous définissons l'opération de déconvolution comme suit :

Définition 5.1.2 (Déconvolution). *Soit h_1, $h_2 \in \mathcal{F}$ alors la déconvolution de h_1 et h_2 est :*

$$\forall t \in \mathbb{R}_+ \ (h_1 \oslash h_2)(t) = sup_{s \geqslant 0} \{h_1(t+s) - h_2(s)\}.$$

Dans la suite, nous emploierons souvent ces deux opérateurs dans le calcul de bornes caractéristiques de réseaux.

5.1.1 Flux et contrôle de flux

Flux

Dans ce chapitre, un flux f est simplement un ensemble de données ordonnées entrant ou sortant d'un routeur et nous considérons F^{in} (resp. F^{out}) la fonction des arrivées cumulées (resp. des départs cumulés). De manière intuitive, F^{in} et F^{out} appartiennent à l'ensemble des fonctions croissantes $\mathcal{F}$. La figure 59 représente un routeur R traversé par un flux de données transmettant un flux de sortie.

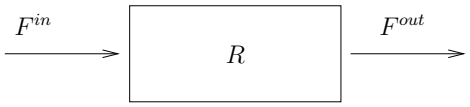


FIGURE 59 – Système simple ayant un flux d'entrée et transmettant un flux de sortie.

Définition 5.1.3 (Trajectoire). *Un couple (F^{in}, F^{out}) est appelé une trajectoire du routeur R.*

Nous supposons que les routeurs ne suppriment ni n'ajoutent de données. En conséquence, la quantité de bits sortant du routeur à un instant donné est toujours inférieure à celle y entrant. Ceci est appelé la propriété de causalité dont la définition formelle est la suivante.

Définition 5.1.4 (Causalité). *Les fonctions F^{in} et F^{out} sont causales si $\forall t \in \mathbb{R}_+$, $F^{out}(t) \leqslant F^{in}(t)$.*

Courbe d'arrivée

Les courbes d'arrivées fournissent des garanties sur la façon dont les données arrivent dans un routeur. On définit une courbe d'arrivée comme suit (voir aussi la définition 2.2.2 page 31) :

Définition 5.1.5 (Courbe d'arrivée). *Soit $\alpha \in \mathcal{F}$. On dit que F^{in} est α-contrainte si et seulement si pour toutes dates s et t telles que $s \leqslant t$,*

$$F^{in}(t) - F^{in}(s) \leqslant \alpha(t-s).$$

Si un flux arrivant en un routeur R est α-contraint, alors quels que soient deux instants s et t choisis, la quantité de données arrivée en R entre ces deux moments est inférieure à $\alpha(t-s)$. Ce principe est représenté sur la figure 60 lorsque la courbe d'arrivée est $\alpha(t) = \sigma + \rho t$. La fonction α fournit ainsi une borne sur la quantité de données pouvant arriver entre deux instants.

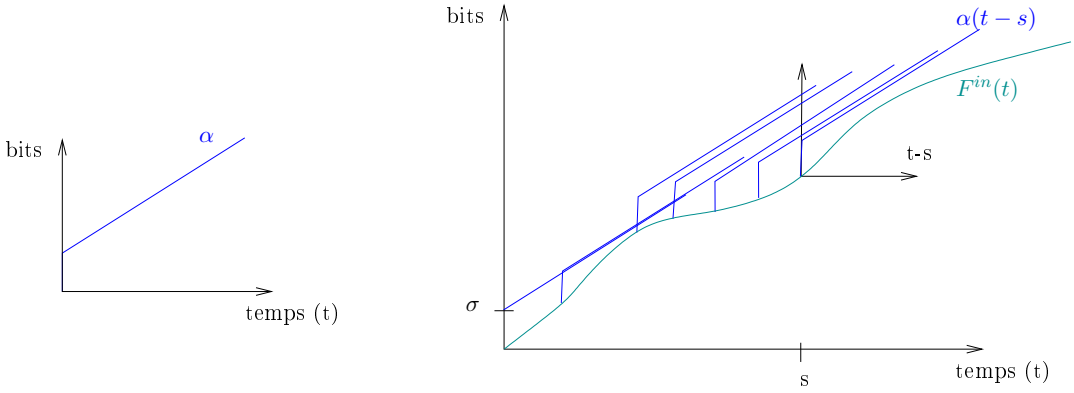


FIGURE 60 – Flux α-contraint lorsque $\alpha(t) = \sigma + \rho t$.

Parmi toutes les courbes d'arrivée, on distingue une classe : les courbes d'arrivées affines par morceaux et concaves. Un exemple d'une telle courbe est donnée en figure 61. Elle peut être exprimée comme un minimum fini de fonctions affines : $\exists n \in \mathbb{N}$, une suite de pentes $\rho_1 > \rho_2 > \ldots > \rho_n$ et une suite de valeurs $\sigma_1 < \sigma_2 < \ldots < \sigma_n$, tels que $\alpha(t) = \min_{i\in\{1,\ldots,n\}}\{\sigma_i + \rho_i t\}$. Dans la suite, nous notons $|\alpha|$ le nombre de fonctions linéaires définissant α.

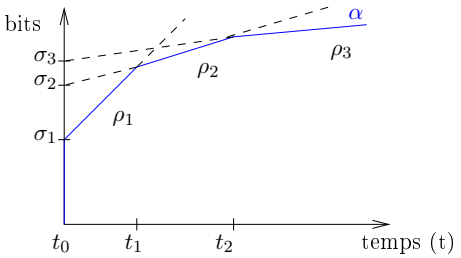


FIGURE 61 – Courbe d'arrivée concave affine par morceaux.

Remarque 5.1.6. *Nous verrons ultérieurement que cette définition permet d'exprimer ce type de fonctions par des contraintes linéaires.*

Les courbes d'arrivée affines définies par $\alpha_{\sigma,\rho}(t) = \sigma + \rho t$ forment un cas particulier des fonctions affines par morceaux concaves. Le routeur est alors observé comme un saut percé, d'où le nom *Leaky Bucket* qui leur est attribué. Il peut traiter des données à une certaine vitesse ρ (vitesse d'écoulement du seau) et dispose d'une file d'attente de taille σ (taille du seau). Si les données arrivent plus vite que le routeur ne peut les traiter, elles remplissent le seau. Si le contenant est plein, les gouttes d'eau suivantes débordent du seau et sont alors perdues. Ceci correspond donc à un bon moyen de contraindre les arrivées des données des flux dont on ne veut pas perdre d'information. Un exemple de fonction Leacky Bucket est représenté sur la figure 62. Cette modélisation de flux

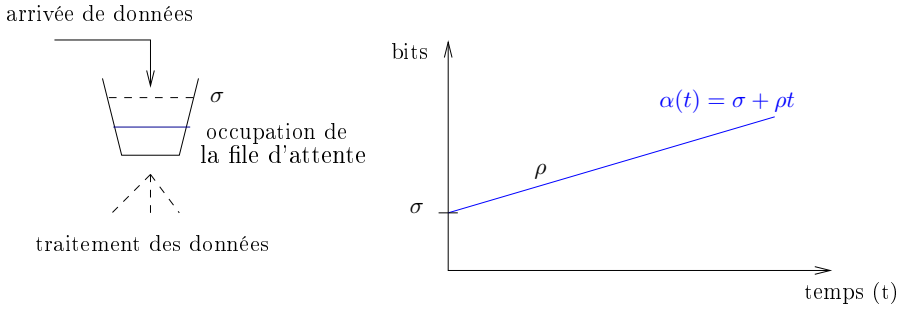


FIGURE 62 – Analogie Leaky Bucket.

implique que le taux d'arrivée des données à long terme est ρ et que le nombre maximal de messages pouvant arriver brusquement est σ.

Courbe de service

Les courbes de service donnent des garanties sur le traitement des flots. En effet, elles vont garantir la vitesse à laquelle un nœud peut analyser des données. De manière formelle une courbe de service est définie comme suit :

Définition 5.1.7 (Courbe de service)**.** *Le routeur R offre au flot une courbe de service $\beta \in \mathcal{F}$ si et seulement si $F^{out} \geqslant F^{in} \otimes \beta$. De plus, un flot de sortie est dit minimal si $F^{out} = F^{in} \otimes \beta$.*

La figure 63 illustre le principe des courbes de service. La figure de gauche représente la courbe de service β, d'un routeur R, correspondant à un simple retard pur d_p. La figure de droite représente les arrivées cumulées de R (en bleu) ainsi que deux fonctions de sorties cumulées F_1^{out} et F_2^{out} (en rouge) qui constituent deux sorties possibles de R, ce dernier étant β-contraint. On remarquera que F_2^{out} est la fonction des départs cumulés minimale étant donné qu'elle vérifie l'équation $F_2^{out} = F^{in} \otimes \beta$. Les trajectoires de sortie sont obtenues à partir de F^{in}, en vérifiant l'équation $F^{out}(t) \geqslant F^{in}(t + d_p)$.

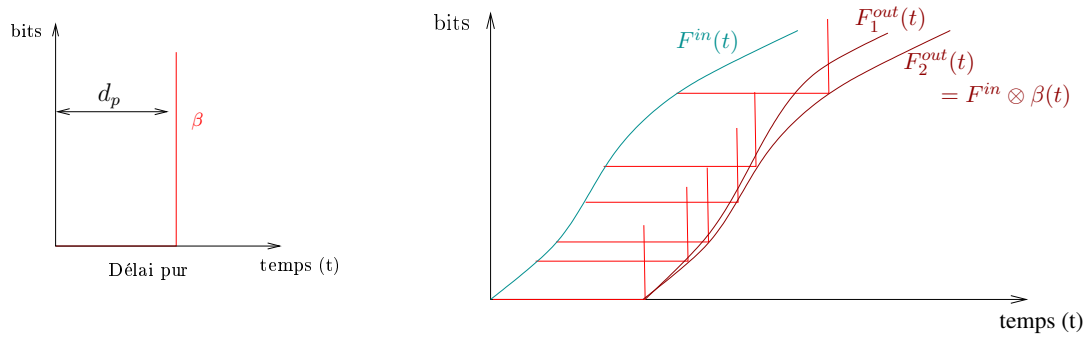


FIGURE 63 – Principe des courbes de service.

Il existe différentes définitions de courbes de service qui ne sont pas toujours équivalentes. Elles sont répertoriées dans l'article [8]. Pour chaque type $\mathcal{T}$ de courbe de service, étudié dans cette thèse, nous définissons pour tout $\beta \in \mathcal{F}$, l'ensemble $\mathcal{S}_{\mathcal{T}}(\beta)$ des trajectoires admettant β comme courbe de service de type $\mathcal{T}$. Un routeur R fournit une *courbe de service β de type $\mathcal{T}$*, si les conditions de $\mathcal{S}_{\mathcal{T}}(\beta)$ sont respectées pour toute trajectoire (F^{in}, F^{out}). On note alors $R \subseteq \mathcal{S}_{\mathcal{T}}(\beta)$.

– Courbe de service simple :

$$\mathcal{S}_{simple}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid F^{in} \geqslant F^{out} \geqslant F^{in} \otimes \beta\};$$

– Courbe de service strict :

$$\mathcal{S}_{strict}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid F^{in} \geqslant F^{out}, \forall \mathcal{PC} :]s,t[, s \leqslant t,$$

$$F^{out}(t) - F^{out}(s) \geqslant \beta(t-s)\},$$

où $\mathcal{PC}$ est une *période chargée*. Il s'agit d'un intervalle durant lequel le routeur dispose en permanence de données à traiter. Cette notion est formalisée par la définition 5.1.8 suivante ;

– Nœud à capacité variable (*Variable Capacity Node (VCN)*) :

$$\mathcal{S}_{vcn}(\beta) = \{(F^{in}, F^{out}) \in \mathcal{F} \times \mathcal{F} \mid \exists C \in \mathcal{F}, \forall t \geqslant 0,$$

$$F^{out}(t) = \inf_{0 \leqslant s \leqslant t} F^{in}(s) + C(t) - C(s), \text{ et } \forall 0 \leqslant s \leqslant t,\ C(t) - C(s) \geqslant \beta(t-s)\}.$$

On remarque facilement que la définition d'une courbe de service strict β est plus restrictive qu'une courbe de service simple. Ainsi, si β est une courbe de service strict, β est également une courbe de service simple. Cependant, le contraire n'est pas vrai. Dans l'article [8], Bouillard et al. prouvent que $\mathcal{S}_{vcn}(\beta) \subseteq \mathcal{S}_{strict}(\beta) \subseteq \mathcal{S}_{simple}(\beta)$.

Définition 5.1.8 (Période chargée)**.** *La période chargée est un intervalle de temps $I \subset \mathbb{R}_+$ durant lequel la charge du routeur n'est pas nulle. Autrement dit, $\forall s \in I, F^{in}(s) - F^{out}(s) > 0$. Le début de période chargée contenant une date t quelconque, notée $start(t)$, est définie par : $start(t) = \sup\{s \leqslant t | F^{in}(s) = F^{out}(s)\}$.*

Les fonctions des arrivées cumulées F^{in} et F^{out} étant continues à gauche, nous avons $F^{in}(start(t)) = F^{out}(start(t))$. De plus, si $F^{in}(t) = F^{out}(t)$ alors $start(t) = t$.

Remarque 5.1.9. *Dans les définitions précédentes, c'est le service qui est de type $\mathcal{T}$ et non la courbe β. Le type de service définit la façon dont le routeur se comporte étant donnée une courbe de service. Par abus de langage on parlera souvent de courbe de service de type $\mathcal{T}$ pour signifier qu'un routeur dispose d'un service $\mathcal{T}$ de paramètre β.*

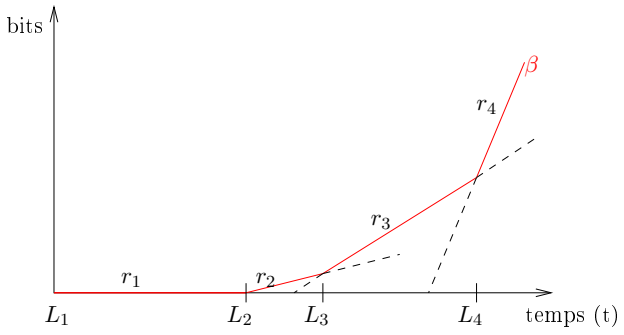


FIGURE 64 – Courbes de service convexe affine par morceaux.

Parmi toute les courbes de service, on distingue les courbes exprimant un délai pur, définies par $\beta_{d_p}(t) = 0$ si $t < d_p$, sinon $\beta_{d_p}(t) = +\infty$ et les courbes de service convexes affines par morceaux définies par une collection de vitesses de traitement croissantes. Elles peuvent être exprimées comme un maximum fini de fonctions affines comme suit : $\exists n \in \mathbb{N}$, $r_1 < r_2 < \ldots < r_n$, et une suite de valeurs $L_1 < L_2 < \ldots < L_n$ tels que

$\beta(t) = \max_{i\in\{1,\dots,n\}}\{r_i(t - L_i)\}$. Dans la suite, nous notons $|\beta|$ le nombre de fonctions linéaires définissant β_j. Un exemple d'une courbe de service convexe affine par morceaux est représenté sur la figure 64. Dans notre étude, sauf mention contraire, nous supposons que les routeurs sont paramétrés par des courbes de service convexes affines par morceaux.

Les courbes de service *Rate-Latency*, définies par $\beta_{r,L} = r(t - L)_+$ illustrent un cas particulier des fonctions convexes affines par morceaux. Rappelons que la notation a_+ représente $\max(a, 0)$.

La figure 65 présente des fonctions d'arrivées et de départs cumulés possible pour un routeur muni d'une courbe de service $\beta_{r,L}(t)$. Le repère de droite représente la fonction de sortie minimale F_1^{out} dans le cas d'un service strict, le repère de gauche, quant à lui, représente la fonction de sortie minimale F_2^{out} dans le cas d'un service simple. La fonction F_1^{out} est égale à tout instant à la contrainte la plus stricte de la période chargée alors que $F_2^{out} = F^{in} \otimes \beta$ est égale à chaque instant au minimum des contraintes β représentées en chaque point de la courbe F^{in}.

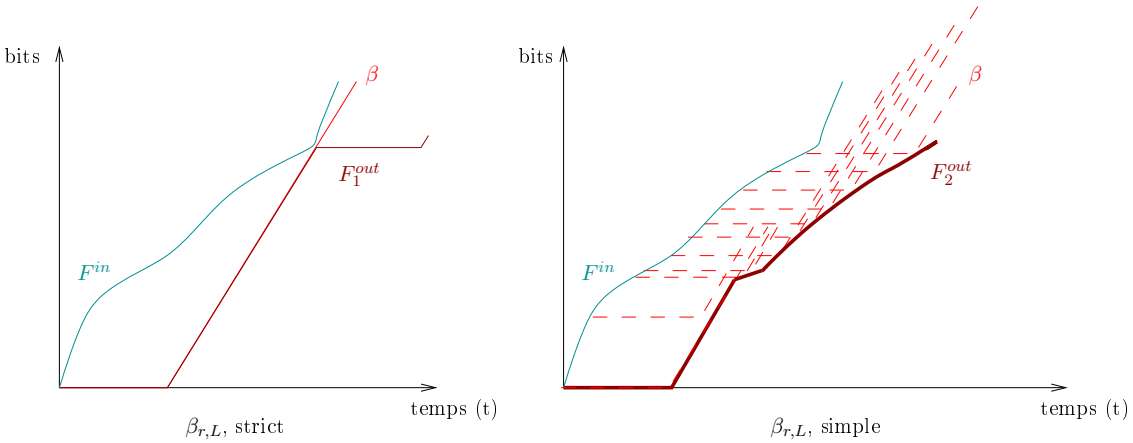


FIGURE 65 – Courbes de service Rate-Latency, flots d'entrée et de sortie.

En général, les courbes de service des réseaux utilisées sont de la forme $\beta_{r,L}$. Cette modélisation du service d'un routeur indique que les messages sont en moyenne traités avec une vitesse r mais que leur traitement débute par une latence de L secondes.

Trajectoire admissible

Une trajectoire est dite admissible si elle respecte les propriétés du network calculus. Elle est déterminée formellement par la définition suivante.

Définition 5.1.10 (Trajectoire admissible)**.** *Soit F^{in} (resp. F^{out}) une fonction des arrivées (resp. sorties) cumulées en un routeur R disposant d'un service strict et soient $\alpha, \beta \in \mathcal{F}$ deux fonctions croissantes. Alors (F^{in}, F^{out}) est une trajectoire admissible de R si :*

- *Croissance de F^{in} et F^{out} ;*
- *Causalité de F^{in} et F^{out} : $F^{in} \geqslant F^{out}$;*
- *F^{in} est α-contraint : $F^{in}(t) - F^{in}(s) \leqslant \alpha(t - s)$;*
- *(F^{in}, F^{out}) est strictement β-contraint : $(F^{in}, F^{out}) \in \mathcal{S}_{strict}(\beta)$.*

5.1.2 Bornes caractéristiques

En introduction, nous avons cité les bornes de performance classiquement évaluées par le *network calculus* (*charge* et *délai*) ainsi que la notion de *flot d'entrée et de sortie*. Dans cette partie, nous en donnons les définitions précises.

Charge

La charge correspond à la quantité de données présente dans le routeur étudié.

Définition 5.1.11 (Charge). *La charge de R à la date t est définie par $F^{in}(t) - F^{out}(t)$.*

Remarque 5.1.12. *Quelque soit $t \in \mathbb{R}_+$, l'intervalle $]start(t), t]$ est une période chargée.*

La page 28 du livre [50] nous fournit le résultat ci-dessous.

Théorème 5.1.13 (Propriété de la charge). *Soient α la contrainte sur F^{in} et β la courbe de service du routeur. Alors la charge, à un instant donné t, vérifie :*

$$F^{in}(t) - F^{out}(t) \leqslant sup_{s \geqslant 0}\{\alpha(s) - \beta(s)\} = \alpha \oslash \beta(0).$$

Délai

La définition ci-dessous concerne le délai subi par une donnée entre son entrée et sa sortie d'un routeur R. Cette information permet de définir si un réseau est surchargé (délai infini), ou si une donnée est traitée suffisamment rapidement.

Définition 5.1.14 (Délai). *Soit une trajectoire (F^{in}, F^{out}) de R, le délai subi par la donnée entrant dans R à l'instant t est*

$$\begin{aligned} d(t) &= \inf\{s \geqslant 0 | F^{in}(t) \leqslant F^{out}(t+s)\} \\ ou\ d(t) &= \sup\{s \geqslant 0 | F^{in}(t) > F^{out}(t+s)\}. \end{aligned}$$

Soit $hDev(h_1, h_2) = \sup_{t \geqslant 0}\{\inf_{d \geqslant 0}\{h_1(t) \leqslant h_2(t+d)\}\}$ la distance horizontale maximum entre deux fonctions h_1 et h_2. Le délai maximal d'une trajectoire est défini par $D_{max} = \sup_{t \geqslant 0} d(t) = hDev(F^{in}, F^{out})$.

Le délai de traitement maximal d'une donnée entrant dans le routeur R correspond au délai maximum subi par les données de toutes les trajectoires de R. Nous pouvons ainsi établir le théorème suivant.

Théorème 5.1.15 (Propriété du délai). *Soit un flot α-contraint traversant un routeur R ayant une courbe de service simple β. Alors $D_{max} \leqslant hDev(\alpha,\beta)$.*

Le délai maximal d'un routeur R peut être obtenu uniquement à partir de α et de β. En effet le délai maximal est borné par la plus grande déviation horizontale séparant α de β, ce que nous représentons sur le schéma 66.

Exemple 5.1.16. *Calculons le délai maximal D_{max} d'un flux $\alpha_{\sigma,\rho}$-contraint, passant dans un routeur $\beta_{r,L}$-contraint. D'après le théorème ci-dessus,*

$$D_{max} \leqslant hDev(\alpha, \beta) = \sup_{t \geqslant 0}\{\inf_{d \geqslant 0}\{\alpha(t) \leqslant \beta(t+d)\}\}.$$

Commençons par calculer $\alpha_{\sigma,\rho}(t) \leqslant \beta_{r,L}(t+d)$:

$$\begin{aligned} \sigma + \rho t &\leqslant r(t + d - L) \\ \frac{\sigma + rL}{r} + \frac{(\rho - r)t}{r} &\leqslant d. \end{aligned}$$

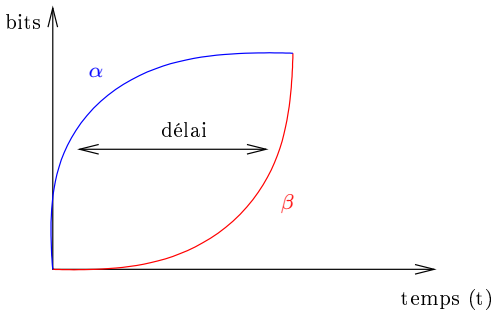


FIGURE 66 – Représentation du délai d'un routeur β-contraint, traversé par un flux α-contraint.

Donc,

$$\begin{aligned} D_{max} &\leqslant hDev(\alpha_{\sigma,\rho}, \beta_{r,L}) \\ Si\ (\rho - r) \leqslant 0,\ alors\ D_{max} &= \frac{\sigma + rL}{r}; \\ Si\ non\ D_{max} &= +\infty. \end{aligned}$$

5.1.3 Contraintes dans les réseaux complexes

Considérons maintenant des réseaux plus complexes pouvant être composés de plusieurs flux et routeurs. Nous introduisons ainsi la contrainte sur un flot à sa sortie d'un routeur, *la notion de capacité résiduelle* et de *service global.*

Flot de sortie

Nous avons précédemment introduit F^{out}, la fonction des départs cumulés d'un routeur R traversé par un flot dont la fonction des arrivées cumulées est F^{in}. Nous donnons ici un résultat présenté dans [11] que nous utiliserons ultérieurement.

Théorème 5.1.17 (Flot de sortie)**.** *Soit un flot de courbe d'arrivée α passant dans un routeur β-contraint. Alors le flot sortant est α'-contraint par :*

$$\alpha' = \alpha \oslash \beta.$$

Exemple 5.1.18. *Calculons la contrainte de sortie d'un flux d'arrivée $\alpha_{\sigma,\rho}$-contraint traversant un routeur $\beta_{r,L}$-contraint dans l'éventualité où $r \geq \rho$. Cette hypothèse correspond au cas généralement considéré dans lequel le délai est borné, comme montré précédemment.*

$$\begin{aligned} \alpha'(t) &= ((\sigma + \rho t) \oslash r(t - L)_+) \\ &= \sup_{s \geqslant 0} \{\sigma + \rho(t + s) - r(s - L)_+\} \\ &= \sup_{0 \leqslant s \leqslant L} \{\sigma + \rho(t + s)\} \vee \sup_{s \geqslant L} \{\sigma + \rho t - (r - \rho)s + rL\} \\ &= \sigma + \rho(t + L) = \alpha(t + L). \end{aligned}$$

La charge d'un routeur de courbe de service β traversé par un flux α-contraint étant $(\alpha \oslash \beta)(0)$ (d'après le théorème 5.1.13), la charge de R est $\alpha'(0) = \alpha(L)$.

Capacité résiduelle

La capacité de calcul d'un routeur une fois qu'un ou plusieurs flux ont été traité s'appelle la capacité résiduelle d'un routeur, définie ci-dessous.

Théorème 5.1.19 (Capacité résiduelle). *Considérons deux flux f_1 et f_2 traversant un routeur R de courbe de service stricte β tel que f_1 est α_1-contraint. Alors, f_2 dispose d'un service minimal simple $(\beta - \alpha_1)_+$.*

Le théorème précédent ne s'applique que si le service associé au routeur R est strict. Le service résiduel étant simple, ceci empêche son utilisation sur les courbes résiduelles. Cependant, le théorème suivant affirme que la courbe résiduelle reste stricte dans le cas de politique de service à priorités fixes.

Théorème 5.1.20 (Capacité résiduelle ([8] p20)). *Considérons deux flux f_1 et f_2 traversant un routeur R de courbe de service stricte β tel que f_1 est α_1-contraint. Si f_1 a priorité sur f_2, le service minimal strict offert à f_2 est $(\beta - \alpha_1)_+$.*

On peut ainsi calculer la capacité de traitement associée aux flux les moins prioritaires.

Exemple 5.1.21. *Soit un routeur R de courbe de service strict $\beta_{r,L}$ traversé par deux flux f_1 et f_2 de courbes d'arrivées respectives α_{σ_1,ρ_1}, α_{σ_2,ρ_2}, et supposons que f_1 est plus prioritaire que f_2. Ce réseau est représenté sur la figure 67. Le service résiduel β' après le traitement de f_1 est :*

$$\begin{aligned} \beta'(t) &= (\beta_{r,L} - \alpha_{\sigma_1,\rho_1})_+(t) \\ &= ((r(t-L)_+ - \sigma_1 + \rho_1 t)_+ \\ &= (r-\rho_1)(t-L')_+, \end{aligned}$$

où L' est la date à laquelle $(\beta_{r,L} - \alpha_{\sigma_1,\rho_1})$ s'annule : $L' = \frac{\sigma_1 + rL}{r-\rho_1}$. Le flux f_2 est donc servit par R avec une courbe de service strict $\beta'(t) = (r-\rho_1)(t - \frac{\sigma_1+rL}{r-\rho_1})_+$.

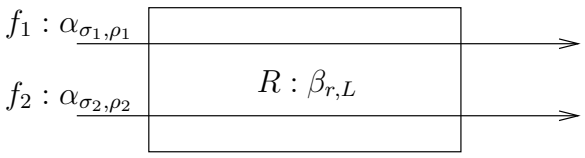


FIGURE 67 – Système traversé par deux flux.

Service global

Le service global correspond à la capacité totale de service d'un réseau composé de plusieurs routeurs.

Définition 5.1.22 (Concaténation). *Soit un flot traversant un routeur R_1 puis un routeur R_2. Supposons que le routeur R_i dispose d'une courbe de service β_i pour $i = \{1, 2\}$. Alors le réseau peut être représenté par un routeur unique résultant de la concaténation de R_1 et R_2 disposant d'une courbe de service minimale $\beta_1 \otimes \beta_2$.*

Exemple 5.1.23. *Étudions un réseau composé de deux routeurs, R_1 et R_2 dont les courbes de services associées sont respectivement β_{r_1,L_1} et β_{r_2,L_2}, traversés par un flux. Par un calcul simple (livre [50] p35) on obtient que la courbe de service β_{Tot} du réseau est :*

$$\beta_{Tot}(t) = \beta_{r_1,L_1} \otimes \beta_{r_2,L_2} = (\min(r_1, r_2)(t - L_1 - L_2))_+.$$

Ainsi, lorsqu'un (ou plusieurs) flux traverse(nt) un ensemble de routeurs, on peut imaginer qu'il franchit un routeur unique dont la courbe de service résulte en la convolution des courbes de service des routeurs traversés. Cependant, ce principe n'est valable que si tous les flux traversent l'ensemble des routeurs dont on fait la concaténation. En effet, si ce n'est pas le cas, le problème devient plus complexe et fait l'objet, entre autre, de l'étude [67], présentée ultérieurement (paragraphe 5.2.1).

Dans les calculs futurs, nous appliquerons directement les résultats des exemples précédents.

5.2 État de l'art

Dans cette partie nous faisons l'inventaire des méthodes existantes permettant de calculer le délai maximal d'un flux passant à travers un réseau. Nous commençons par présenter des techniques introduites dans les articles [9] et [67], où aucune supposition n'est faite sur la politique de service (*multiplexage aveugle*). Tous les flux sont servis ensemble par le routeur dont on connaît la capacité de traitement minimale. Enfin, nous présentons une procédure de calcul prenant en compte des priorités sur les flux ([70]).

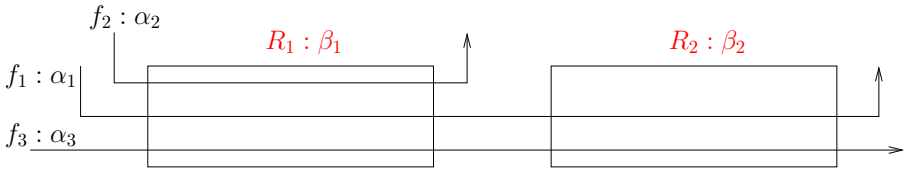


FIGURE 68 – Réseau de deux routeurs et trois flux.

Nous illustrons ces différentes méthodes sur le réseau simple représenté sur la figure 68. Ce réseau est composé de 2 routeurs (R_1 et R_2) et trois flux (f_1, f_2 et f_3). Les routeurs disposent de courbes de service strict respectives β_1 et β_2 et les flux sont contraints par les courbes d'arrivées respectives α_1, α_2 et α_3. Les flux f_1 et f_3 traversent le réseau entier alors que f_2 ne passe que par R_1.

5.2.1 Analyse de flux séparés

Nous nous intéressons dans cette partie aux réseaux dont les flux sont regroupés et où aucune supposition n'est faite sur la politique de service. C'est ce que l'on appelle une politique aveugle car l'on ne peut pas déterminer quelle donnée est traitée en premier. Comme on ne sait pas quels flux sont servis en premier, l'analyse de Schmitt et al., présentée dans l'article [67], se place dans le pire cas : chaque flux est étudié comme s'il était traité en dernier dans chaque nœud du réseau. Ainsi, Schmitt et al. commencent par calculer les services résiduels en chaque nœud pour chaque flux considéré séparément. Autrement dit, ils calculent la bande passante de chaque nœud pour chaque flux. Ces services résiduels sont ensuite concaténés pour obtenir le service minimal d'un flux sur le réseau complet. Alors, le délai maximum, subi par ce flux, se calcule par la déviation

horizontale entre sa courbe d'arrivée et la courbe de service résiduelle du réseau complet. Cette méthode s'appelle *Separate Flow Analysis* (SFA).

Exemple 5.2.1. *Le délai maximal pour le flux f_1 de la figure 68 est :*

$$D_{max(1)} = hDev(\alpha_1, \beta'_1 \otimes \beta'_2),$$

où $\beta'_1 = [\beta_1 - \alpha_2 - \alpha_3]_+$ est le service résiduel de R_1 après traitement de f_2 et f_3, $\beta'_2 = [\beta_2 - \alpha'_3]_+$ celui de R_2 après le traitement de f_3 dont la courbe d'arrivée $\alpha'_3 = (\alpha_3 \oslash (\beta_1 - \alpha_1 - \alpha_2)_+)_+$, correspond au flux de sortie de f_3 après R_1 dont le service résiduel pour f_3 est $(\beta_1 - \alpha_1 - \alpha_2)_+$.

Cette approche peut être améliorée dans le cas de politiques de services à priorités fixes. Dans ce cas, il suffit de retrancher le service apporté aux flux plus prioritaires uniquement.

Exemple 5.2.2. *Sur l'exemple précédent, si l'on suppose que f_1 est le flux le plus prioritaire et f_3 le moins prioritaire, le délai de f_1 est $D_{max(1)} = hDev(\alpha_1, \beta_1 \otimes \beta_2)$, et celui de f_3 est $D_{max(3)} = hDev(\alpha_3, [\beta_1 - \alpha_1 - \alpha_2]_+ \otimes [\beta_2 - (\alpha_1 \oslash [\beta_1 - \alpha_2 - \alpha_3]_+)_+]_+)$.*

Dans le calcul fait par SFA, le multiplexage des flux qui interfèrent est compté plusieurs fois. Cette vision n'est pas très réaliste. En effet, une fois que les flux ont traversé une premier routeur, les coûts de multiplexage sont moindre pour les suivants. Une meilleure approche, prenant en compte un phénomène est appelé *PMOO : Pay Multiplexing Only Once*). Elle amortis ces coûts sur l'ensemble des routeurs. Cette technique fut appliquée à SFA et a donné naissance à une nouvelle méthode : PMOO-SFA.

Pour calculer le délai subi par les données d'un flux f, la méthode PMOO-SFA effectue d'abord la concaténation des routeurs, puis retranche la quantité de traitement nécessaire au passage des données des autres flux. Ceci fournit la courbe de service résiduelle totale permettant le traitement des données d'un flux f. Le délai maximum est enfin obtenu par la déviation horizontale entre sa courbe d'arrivée et la courbe de service résiduelle ainsi calculée.

Cependant, l'analyse PMOO-SFA s'applique à un ensemble restreint de topologies. En effet, elle n'a de sens que si tous les flux du réseaux suivent le même chemin. Ainsi, elle ne peut être appliquée pour le réseau de la figure 68.

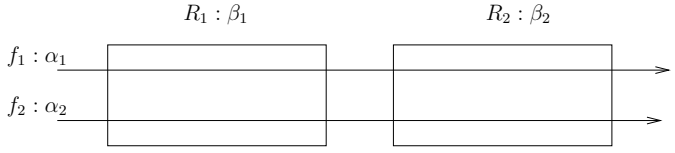


FIGURE 69 – Réseau de deux routeurs et deux flux.

Exemple 5.2.3. *Appliquons néanmoins cette analyse dans le cas d'un réseau de deux flux (f_1, f_2) traversant deux routeurs (R_1, R_2) (figure 69). Le délai subi par le flux f_1 est :*

$$D_{max(1)} = hDev(\alpha_1, (\beta_1 \otimes \beta_2 - \alpha_2)_+).$$

Dans certains cas pathologiques, PMOO-SFA rend de moins bons résultats que SFA ([67]). Considérons à nouveau le réseau de la figure 69 lorsque pour tout $i \in \{1, 2\}$ $\alpha_i(t) = \sigma_i + \rho_i t$ et $\beta_i(t) = r_i(t - L_i)_+$. Les méthodes SFA et PMOO-SFA définissent les formules suivantes :

$$d_{SFA} = L_2 + \frac{\sigma_1}{\min\{r_1, r_2\} - \rho_2} + \frac{\rho_2 L_2}{r_2 - \rho_2},$$

$$d_{PMOO-SFA} = L_2 + \frac{\sigma_1 + \rho_2 L_2}{\min\{r_1, r_2\} - \rho_2}.$$

Ces formules impliquent que $d_{PMOO-SFA} - d_{SFA} \geqslant 0$. Ainsi, en fonction des paramètres choisis, on peut obtenir une différence arbitrairement large. Ce problème résulte de la définition de l'opérateur de convolution qui est commutatif dans l'algèbre min-plus. Ce qui rompt une information donnée par la topologie.

5.2.2 Algorithme de programmation linéaire (LP)

Bouillard et al. apportent, dans l'article [9], une nouvelle méthode de calcul des bornes de performance des réseaux en utilisant la programmation linéaire. Cette technique encode les contraintes du network calculus (service, arrivées, causalité, ...) par un ensemble de programmes linéaires.

Théorème 5.2.4. *Soit un réseau contenant N_r routeurs et N_f flux. Si le réseau est acyclique alors pour un flux f_i(resp. un routeur R_j) il existe une collection finie λ de programmes linéaires avec pour valeur optimale opt_λ telle que* $\max(opt_\lambda)$ *soit le pire délai d'un bout à l'autre du réseau (resp. la pire charge).*

Rappelons qu'un programme linéaire définit, par un ensemble fini de contraintes, en un ensemble fini de dates, les trajectoires admissibles d'un réseau. L'objectif est de déterminer le pire délai d'une donnée du flux que l'on étudie. Ainsi, on imagine étudier la donnée du flux entrant dans le routeur au pire moment, et l'objectif consiste à déterminer cet instant. Cette donnée est appelée *donnée d'intérêt*. L'ensemble de dates contient tout d'abord la date t_u correspondant au moment où la donnée étudiée sort du réseau. De plus, cet ensemble regroupe les débuts des périodes chargées de chaque routeur tels que la donnée étudiée est présente dans le nœud durant cette période chargée. Les contraintes expriment, en chacune des dates considérées, les règles du network calculus : la croissance des flux, la causalité de chaque flux entre l'entrée et la sortie de chaque routeur ainsi que les contraintes α et β. Il faut ajouter au programme linéaire les contraintes définissant l'ordonnancement des dates.

Un solveur résout ensuite l'ensemble des contraintes. Ceci permet de tester toutes les fonctions d'arrivées et de départs cumulés et de trouver la donnée subissant le pire délai. Cette méthode offre le gros avantage de retourner le délai exact.

Malheureusement, dans le cas acyclique, le temps de calcul est NP-difficile. Cependant, les auteurs de l'article [9] montrent que le temps de résolution est polynomial pour les réseaux en tandem.

Exemple 5.2.5. *Les contraintes linéaires correspondant au réseau de la figure 68 sont énumérées ci-dessous. On notera, pour tout $i \in \{1,2,3\}$, $F_i^{(0)}$ les arrivées cumulées des données de f_i en R_1, $F_i^{(1)}$ les départs cumulés des données de f_i en R_1, et pour tout $i \in \{1,3\}$ $F_i^{(2)}$ le flux de données de f_i sortant de R_2. On dispose de quatre dates importantes : t_1 début de période chargée de R_1, t_2 celle de R_2, t_u (resp. t_3) date à laquelle la donnée de f_3 étudiée rentre (resp. sort) du réseau.*

- *Objectif : max $t_3 - t_u$;*
- *Croissance des dates : $t_1 \leqslant t_2 \leqslant t_3$;*
- *Contrainte sur la date d'arrivée, t_u, de la donnée d'intérêt :*
  - $t_1 \leqslant t_u \leqslant t_3$;
  - $F_3^{(0)}(t_u) - F_3^{(2)}(t_3) \geqslant 0$;
  - $F_3^{(0)}(t_u) - F_3^{(0)}(t_1) \geqslant 0$ *et* $F_3^{(0)}(t_3) - F_3^{(0)}(t_u) \geqslant 0$;
  - $F_3^{(0)}(t_u) - F_3^{(0)}(t_1) \leqslant \alpha_3(t_u - t_1)$ *et* $F_3^{(0)}(t_3) - F_3^{(0)}(t_u) \leqslant \alpha_3(t_3 - u)$;

- *Début de période chargée :*
  - $F_i^{(0)}(t_1) = F_i^{(1)}(t_1),\ \forall i \in \{1,2,3\}$;
  - $F_i^{(1)}(t_2) = F_i^{(2)}(t_2),\ \forall i \in \{1,3\}$;
- *Croissance des flux :*
  - $F_i^{(0)}(t_1) \leqslant F_i^{(0)}(t_2) \leqslant F_i^{(0)}(t_3),\ \forall i \in \{1,2,3\}$;
  - $F_i^{(1)}(t_1) \leqslant F_i^{(1)}(t_2) \leqslant F_i^{(1)}(t_3),\ \forall i \in \{1,2,3\}$;
  - $F_i^{(2)}(t_1) \leqslant F_i^{(2)}(t_2) \leqslant F_i^{(2)}(t_3),\ \forall i \in \{1,3\}$;
- *Causalité des flux :*
  - $F_i^{(0)}(t_1) \geqslant F_i^{(1)}(t_1) \geqslant F_i^{(2)}(t_1),\ \forall i \in \{1,3\}$ *et* $F_2^{(0)}(t_1) \geqslant F_2^{(1)}(t_1)$;
  - $F_i^{(0)}(t_2) \geqslant F_i^{(1)}(t_2) \geqslant F_i^{(2)}(t_2),\ \forall i \in \{1,3\}$ *et* $F_2^{(0)}(t_2) \geqslant F_2^{(1)}(t_2)$;
  - $F_i^{(0)}(t_3) \geqslant F_i^{(1)}(t_3) \geqslant F_i^{(2)}(t_3),\ \forall i \in \{1,3\}$ *et* $F_2^{(0)}(t_3) \geqslant F_2^{(1)}(t_3)$;
- *Contraintes α du flux $f_i, i \in \{1,2,3\}$:*
  - $F_i^{(0)}(t_2) - F_i^{(0)}(t_1) \leqslant \alpha_i(t_2 - t_1)$;
  - $F_i^{(0)}(t_3) - F_i^{(0)}(t_2) \leqslant \alpha_i(t_3 - t_2)$;
  - $F_i^{(0)}(t_3) - F_i^{(0)}(t_1) \leqslant \alpha_i(t_3 - t_1)$;
- *Contraintes β :*
  - $F_1^{(1)}(t_2) + F_2^{(1)}(t_2) + F_3^{(1)}(t_2) - F_1^{(1)}(t_1) - F_2^{(1)}(t_1) - F_3^{(1)}(t_1) \geqslant \beta_1(t_2 - t_1)$;
  - $F_1^{(2)}(t_3) + F_3^{(2)}(t_3) - F_1^{(2)}(t_2) - F_3^{(2)}(t_2) \geqslant \beta_2(t_3 - t_2)$.

5.2.3 Priorités fixes

La méthode introduite par Thiele et al. dans l'article [70] se nomme *Real Time Calculus-fixed priorities* (RTC). Cette technique de calcul vise à apporter un ordonnancement efficace dans le cas de réseaux à tâches périodiques et apériodiques, en temps réel ou sans contrainte de temps, à l'aide des priorités fixes. On veut donc garantir l'arrivée de celles étant temps-réel avant leurs échéances tout en permettant l'exécution du plus grand nombre de tâches non-contraintes possible. Pour cela, les tâches temps-réel disposent des priorités les plus fortes.

Son principe est similaire à celui de la méthode SFA sauf que RTC suppose que les flux sont soumis à des priorités. Ceci permet donc d'affiner la borne calculée. Le service résiduel est obtenu en soustrayant progressivement les ressources nécessitées par les flux de priorités plus élevées.

RTC diffère néanmoins de SFA car elle associe deux courbes des arrivées à un flux de données et deux courbes de service à un routeur. La courbe α^{min} (resp. α^{max}) correspond à la contrainte minimum (resp. maximum) sur les arrivées de données dans le routeur à chaque instant. De manière similaire, un routeur R dispose de deux courbes services β^{min} et β^{max} correspondant respectivement au service minimum/maximum que le routeur peut fournir. La formule suivante donne le délai maximal subi par une donnée entrant dans R :

$$D_{max} = hDev(\alpha^{max}, \beta^{min}).$$

Afin de comparer cette méthode à celles introduites précédemment les courbes α^{min} et β^{max} sont oubliées.

La figure 70, représente le réseau de la figure 68 vu par RTC, lorsque f_1 dispose de la plus forte priorité et f_3 de la plus faible. Ainsi, f_1 traverse R_1 disposant du service β_1. Le flux f_2 traverse également ce premier routeur. Cependant, f_1 et f_2 sont représentés comme s'ils étaient traités par différents routeurs. En effet, f_2 étant moins prioritaire que f_1, ces données sont traitées une fois f_1 entièrement servi. Ce qui équivaut à considérer que f_2 traverse un autre routeur de courbe de service $\beta_1' = (\beta_1 - \alpha_1)_+$.

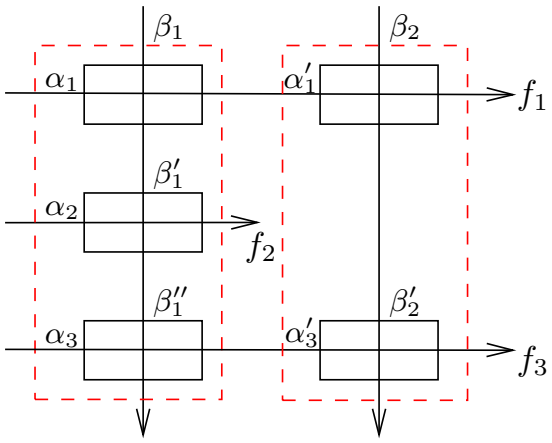


FIGURE 70 – Application de la méthode RTC au réseau de la figure 68.

Le multiplexage étant réalisé au fur et à mesure de l'obtention des courbes résiduelles, les coûts de multiplexage sont comptés plusieurs fois et le phénomène PMOO n'est pas assurée. Par conséquent, les bornes calculées par cette méthode sont sur-estimées.

Exemple 5.2.6 (Application numérique). *En utilisant les courbes* $\alpha_1(t) = 1+t$, $\alpha_2(t) = 1+t$, $\alpha_3(t) = 1+2t$, $\beta_1(t) = 6(t-1)_+$ *et* $\beta_2(t) = 4(t-2)$, *les courbes résiduelles associées à* f_3 *pour chacune des méthodes sont* $\beta_{SFA}(t) = 3(t-6.556)_+$, $\beta_{LP}(t) = 3(t-5.133)_+$ *et* $\beta_{RTC}(t) = 3(t-6)_+$. *Alors les délais obtenus par les méthodes sont :* $d_{SFA} = 6.889$, $d_{LP} = 5.467$ *et* $d_{RTC} = 6.333$. *On remarque immédiatement que LP rend un meilleur résultat que les deux autres méthodes sans prendre en compte la politique de service. Notre objectif est de fournir une méthode calculant des bornes plus fines que les techniques existantes, dans le cadre d'une politique de service à priorités fixes.*

5.3 Étude du délai maximal : politique de service à priorités fixes

Introduisons notre méthode de calcul de délai pire-cas à l'aide de contraintes linéaires pour les politiques de service à priorités fixes. Définissons tout d'abord les notations utilisées ainsi qu'une condition nécessaire et suffisante pour obtenir des délais bornés.

5.3.1 Modèle du réseau

On considère désormais que les flux sont soumis à des priorités. Par la suite, les flux sont indexés selon leur priorité et chaque priorité est différente :

$$\text{(PRIO)} \quad i < j \quad \Leftrightarrow \quad f_i \text{ a une priorité plus élevée que } f_j.$$

Ainsi, f_1 est le flux le plus prioritaire.

Considérons un réseau contenant N_r routeurs tels que le routeur R_k dispose d'une courbe de service strict minimale β_k convexe affine par morceaux et N_f flux tels que le flux f_i est contraint par une fonction α_i concave affine par morceaux.

Chaque flux f_i suit un chemin acyclique de longueur ℓ_i dans le réseau que l'on note : $path_i = \langle j_1, \ldots, j_{\ell_i} \rangle$. On nomme également $prev_i(j)$ le routeur traversé par le flux f_i juste avant d'entrer dans R_j. Si R_j est le premier routeur traversé par f_i alors $prev_i(j) = 0$.

Pour chaque routeur R_j, l'ensemble $Fl(j) = \{i | f_i$ traverse le routeur $R_j\}$ définit l'ensemble des flux traversant le routeur et $maxFl(j) = \max\{i | i \in Fl(j)\}$ représente le flux traversant le routeur ayant la plus petite priorité. Par convention, $Fl(0) = [1, N_f]$.

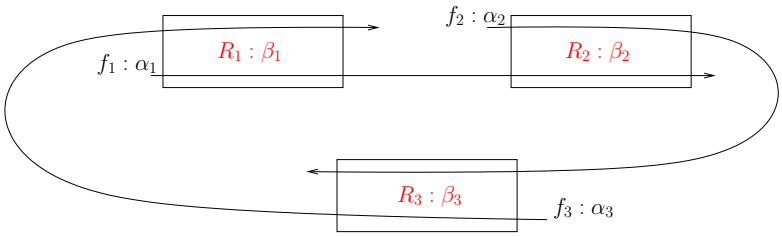


FIGURE 71 – Réseau cyclique de trois routeurs et de trois flux.

Exemple 5.3.1. *Par exemple, sur la figure 71, le flux f_1 dispose d'une courbe d'arrivée α_1. Le chemin parcouru par ce flux est $path_1 = \langle 1, 2\rangle$, et donc $prev_1(2) = 1$. De plus, f_1 est plus prioritaire que f_2, f_3 se voyant servi en dernier. Le routeur R_1 dispose d'une courbe de service strict β_1. Les flux traversant ce routeur sont $Fl(1) = \{1, 3\}$ dont le moins prioritaire est $maxFl(1) = 3$. Enfin la trajectoire du réseau, $(F_1^{(1)}, F_3^{(1)}, F_1^{(2)}, F_2^{(2)}, F_2^{(3)}, F_3^{(3)})$ est admissible en R_1 si, entre autre, $(F_1^{(0)} + F_3^{(3)}, F_1^{(1)} + F_3^{(1)}) \in \mathcal{S}_{strict}(\beta_1)$.*

Si l'on s'intéresse au délai maximal subi par un flux quelconque de priorité m, f_m, alors il n'est pas nécessaire de considérer les flux de priorités inférieures. En effet, considérons une date t à laquelle le réseau reçoit des données à traiter provenant des flux f_m et $f_{m'}$, $m' > m$. Alors, les données de $f_{m'}$ ne seront traitées qu'après celle de f_m. Donc, les flux de priorités inférieures ne peuvent pas augmenter le délai d'un flux de priorité supérieure. Au contraire, comme les routeurs disposent souvent d'un temps de latence correspondant au temps nécessaire au routeur pour démarrer, ils peuvent accélérer le traitement des données des flux de priorités supérieures. En effet, supposons une date t à laquelle le flux f_m présente au réseau des données à traiter. Si le réseau est vide avant cette date (partie gauche de la figure 72), les données ne sont traitées qu'après un temps de latence (L), à partir de la date $t + L$. Cependant, si à l'instant t le réseau est occupé à servir un flux moins prioritaire (partie droite de la figure 72), cette latence n'a pas lieu. Ainsi dans la suite on supposera que le flux ayant la plus faible priorité est celui dont on souhaite calculer le délai et on le notera f_{N_f}.

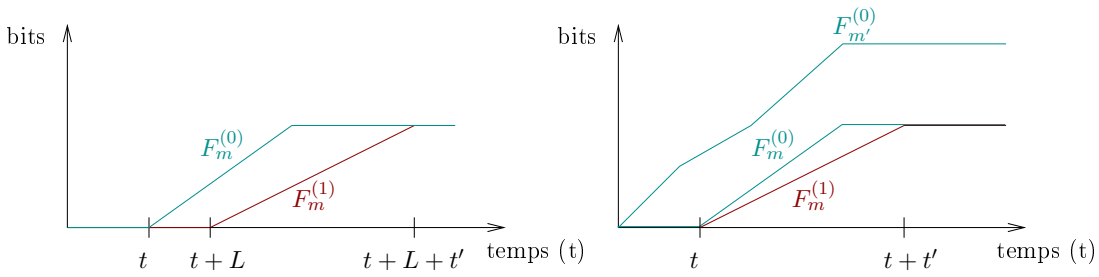


FIGURE 72 – Effet d'un flux moins prioritaire sur le temps de traitement des données d'un flux plus prioritaire.

5.3.2 Condition nécessaire et suffisante pour délai borné

L'algorithme 4 peut être utilisé afin de calculer une borne supérieure du délai dans un réseau quelconque. Cet algorithme reprend le principe de la méthode SFA [67], introduite dans la partie 5.2, lorsque les flux sont soumis à des priorités ou la méthode RTC [70], présentée dans le même paragraphe, lorsque l'on utilise uniquement les courbes α_{max} et β_{min}.

Algorithme 4 : Méthode SFA avec priorités

Données : Description du réseau : $path_i$, α_i, β_j.
Résultats : $\alpha_i^{(j)}$ courbe d'arrivée de $F_i^{(j)}$, $\beta_j^{(i)}$ courbe de service strict de f_i en R_j.

Début
  Pour tous $j = 1$ *à* N_r **faire**
    $\beta'_j \leftarrow \beta_j$;
  Pour tous $i = 1$ *à* N_f **faire**
    Soient $path_i = \langle j_1 \dots j_{\ell_i} \rangle$; $j_0 = 0$;
    $\alpha_i^{(0)} \leftarrow \alpha_i$;
    Pour tous $k = 0$ *à* $\ell_i - 1$ **faire**
      $\alpha_i^{(j_{k+1})} \leftarrow \alpha_i^{(j_k)} \oslash \beta'_{j_k}$;
      $\beta_{j_k}^{(i)} \leftarrow \beta'_{j_k}$;
      $\beta'_{j_k} \leftarrow (\beta_{j_k}^{(i)} - \alpha_i^{(j_k)})_+$;
fin

Définition 5.3.2 (Stabilité). *Un réseau est dit* stable *si et seulement si chaque flux traversant le réseau subi un délai borné.*

Grâce au théorème suivant nous définissons une condition nécessaire et suffisante pour assurer la stabilité d'un réseau.

Théorème 5.3.3 (Condition de stabilité). *Soit un réseau composé de N_r flux et N_f routeurs tels que f_i est contraint par $\alpha_i(t) = \sigma_i + \rho_i t$, R_j dispose d'une courbe de service stricte $\beta_j(t) = r_j(t - L_j)_+$ et la propriété (PRIO) sur les priorités est respectée. Le réseau est stable si et seulement si*

$$\forall j \in [1, N_r], \qquad r_j \geqslant \sum_{i \in Fl(j)} \rho_i.$$

Démonstration. Nous démontrons ce résultat par récurrence sur le nombre de flux dans le réseau.

Hypothèse de récurrence (H_k) : le délai de $f_1, \dots, f_k$ est fini et pour tout routeur R_j, la courbe de service résiduelle pour les flux $f_{k+1}, \dots, f_{N_f}$ prend la forme $\beta_j^{(k)}(t) = r_j^{(k)}(t - L_j^{(k)})_+$ avec

$$r_j^{(k)} \geqslant \sum_{i \in Fl(j) \cap [k+1, N_f]} \rho_i \text{ et } L_j^{(k)} < \infty.$$

L'hypothèse (H_1) est vraie : d'après l'hypothèse du théorème, pour tout $j \in \{j_1, \dots, j_{\ell_1}\}$, chemin de f_1, $r_j \geqslant \rho_1$. Le délai maximal de f_1 est borné (voir exemple 5.1.16). De plus, les flux de sorties de f_1 ont tous un taux ρ_1 (voir exemple 5.1.18) donc, le taux

de service résiduel du routeur R_j est $r_j^{(1)} = r_j - \rho_1$ et $L_j^{(1)} = \dfrac{\sigma_1 + r_j L_j}{r_j - \rho_1} < \infty$ (d'après l'exemple 5.1.21). Pour tous les autres routeurs, quel que soit $j \notin \{j_1, \ldots, j_{\ell_1}\}$, la courbe de service est inchangée et donc $r_j^{(1)} = r_j$ et $L_j^{(1)} = L_j$. En conclusion, pour tout routeur R_j on obtient : $r_j^{(1)} \geqslant \sum_{i \in Fl(j) \cap [2, N_f]} \rho_i$ et $L_j^{(1)} < \infty$.

Supposons que l'hypothèse (H_{k-1}) soit vraie : considérons le flux f_k. Par hypothèse de récurrence, pour tout $j \in \{j_k, \ldots, j_{\ell_k}\}$, $r_j^{(k-1)} \geqslant \sum_{i \in Fl(j) \cap [k, N_f]} \rho_i \geq \rho_k$ donc le délai subi par f_k est borné (voir l'exemple 5.1.16). De plus, le service résiduel de chaque routeur traversé par f_k est $r_j^{(k)} = r_j^{(k-1)} - \rho_k \geqslant \sum_{i \in Fl(j) \cap [k+1, N_f]} \rho_i$ et la capacité de service des autres routeurs est inchangée. Donc, $r_j^{(k)} = r_j^{(k-1)} \geqslant \sum_{i \in Fl(j) \cap [k+1, N_f]} \rho_i$ et la borne $L_j^{(k)} < \infty$ est satisfaite (d'après l'exemple 5.1.18).

Finalement, (H_{N_f}) prouve le théorème.

□

5.3.3 Approche par programmation linéaire

Présentons maintenant le cœur de notre contribution : un programme linéaire calculant les délais maximum dans les réseaux munis de politique de service à priorités fixes. Nous commençons par introduire les contraintes encodant les priorités en un routeur linéaire. Puis, nous extrapolons aux réseaux ayant une topologie en arbre. Finalement, nous terminons le travail en proposant une solution dans le cas des topologies générales.

Encodage des priorités en un routeur linéaire

Considérons le routeur R, représenté sur la figure 73, disposant d'une courbe de service strict β. Ce routeur est traversé par N_f flux $f_1, \ldots, f_{N_f}$ satisfaisant à la propriété $(PRIO)$. Chaque flux f_i est α_i-contraint et $F_i^{(0)}$ (resp. $F_i^{(1)}$) représente la fonction des arrivées cumulées (resp. des départs cumulés) de f_i.

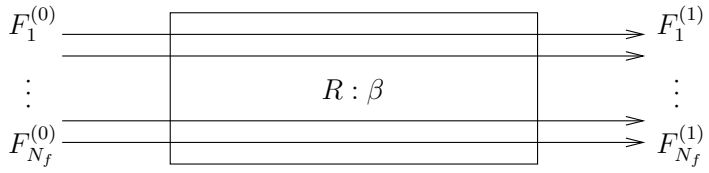


FIGURE 73 – Système R traversé par N_f flux.

Présentons les contraintes assurant les priorités pour une période chargée donnée. Étant donné une date t, nous étudions la période $[start(t), t]$. Dans ce but, il nous faut définir un ensemble de dates intermédiaires, notées $c_i, 0 \leqslant i \leqslant N_f$, telles que $c_0 = t$, $c_{N_f} = start(t)$ et pour tout $1 < i < N_f$

$$c_i = \sup\{s \leqslant t \mid \forall k \in [1, i],\ F_k^{(1)}(s) = F_k^{(0)}(s)\}.$$

La date c_i représente le dernier instant auquel f_{i+1} peut être servi. Autrement dit, cette date correspond au dernier moment de l'intervalle $[start(t), t]$ où tous les flux de priorités supérieures à f_{i+1} n'étaient pas chargés.

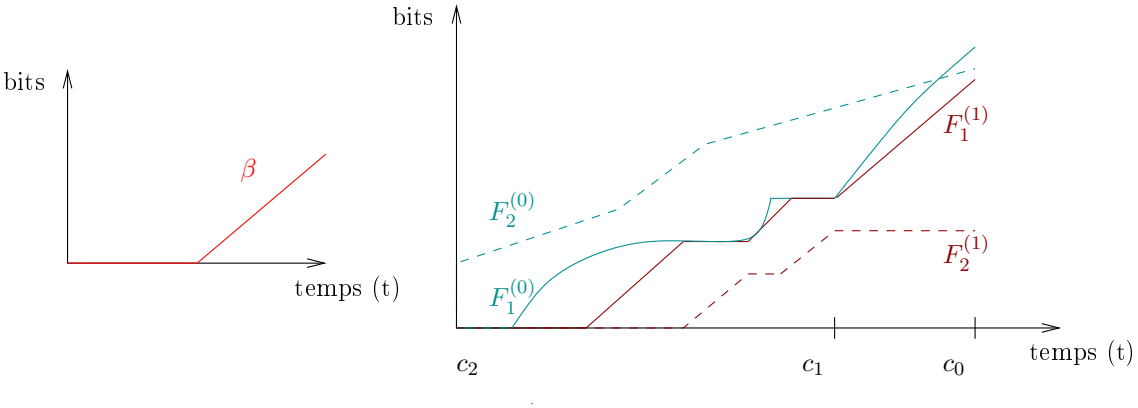


FIGURE 74 – Illustration des dates $(c_i)_{i\in[0,2]}$.

Exemple 5.3.4. *Considérons un routeur R composé de deux flux f_1 et f_2 soumis à la propriété (PRIO) et illustrons le principe des dates $(c_i)_{i\in[0,2]}$. La partie droite de la figure 74 représente deux fonctions possibles des entrées et des sorties cumulées de f_1 et f_2 et la partie gauche représente la courbe de service strict de R. Soit un instant donné t. Alors $c_0 = t$ et $c_2 = start(t)$. La date c_1 correspond au dernier moment de l'intervalle $[start(t), t]$ où les données de f_2 peuvent être traitées.*

Propriété 5.3.5. *La définition des dates intermédiaires ainsi que les caractéristiques afférentes aux priorités des flux fournissent les propriétés suivantes :*

$$c_0 \geqslant c_1 \geqslant \cdots \geqslant c_{N_f}; \tag{5.1}$$

$$\forall i \in [0, N_f], \forall k \leqslant i, F_k^{(0)}(c_i) = F_k^{(1)}(c_i); \tag{5.2}$$

$$\forall i \in [0, N_f], \forall k \geqslant i+2, F_k^{(1)}(c_i) = F_k^{(1)}(c_{i+1}). \tag{5.3}$$

Démonstration. D'après la définition de la date c_i, il est immédiat que $\forall k \in [1, i]$, $F_k^{(0)}(c_i) = F_k^{(1)}(c_i)$. La propriété (5.2) est vérifiée.

Supposons (5.2) vraie : les flux f_1 à f_i sont vidés à la date c_i. Donc, $F_{i+1}^{(0)}(c_i) \geqslant F_{i+1}^{(1)}(c_i)$. Les flux $f_{i+2}, \ldots, f_{N_f}$ ne peuvent donc pas être traité (d'après la propriété (PRIO)). Ceci démontre (5.3).

Finalement, d'après (PRIO), on a $c_{i+1} = \sup\{s \leqslant t \mid \forall k \in [1, i+1],\ F_k^{(1)}(s) = F_k^{(0)}(s)\} \leqslant \sup\{s \leqslant t \mid \forall k \in [1, i],\ F_k^{(1)}(s) = F_k^{(0)}(s)\} = c_i$. La propriété (5.1) est satisfaite. □

Réciproquement, le lemme suivant montre qu'à partir d'une trajectoire admissible satisfaisant aux propriétés (5.1) à (5.3), on peut construire une trajectoire admissible et respectant les priorités aux dates $(c_i)_{i\in[0,N_f]}$.

Lemme 5.3.6. *Soit $(F_i^{(0)}, F_i^{(1)})_{1\leqslant i\leqslant N_f}$ une trajectoire admissible. Supposons qu'il existe $c < c'$ dans une même période chargée et i_0 tels que*

$$\forall i \leqslant i_0, \quad F_i^{(0)}(c) = F_i^{(1)}(c); \tag{5.4}$$

$$\forall i < i_0, \quad F_i^{(0)}(c') = F_i^{(1)}(c'); \tag{5.5}$$

$$\forall i > i_0, \quad F_i^{(1)}(c) = F_i^{(1)}(c'). \tag{5.6}$$

Alors il existe une trajectoire admissible $(F_i^{(0)}, \tilde{F}_i^{(1)})_{1\leqslant i\leqslant N_f}$ *telle que* $\forall 1 \leqslant i \leqslant N_f$, $\tilde{F}_i^{(1)}(c) = F_i^{(1)}(c)$, $\tilde{F}_i^{(1)}(c') = F_i^{(1)}(c')$, $\forall t \in [c, c']$, $\sum_{1\leqslant i\leqslant N_f} \tilde{F}_i^{(1)}(t) = \sum_{1\leqslant i\leqslant N_f} F_i^{(1)}(t)$ *qui respecte les priorités entre les dates* c *et* c'.

Afin de prouver le lemme 5.3.6, nous utilisons l'équivalence $\mathcal{S}_{vcn}(\beta) = \mathcal{S}_{strict}(\beta)$ lorsque β est une fonction affine par morceaux et convexe et le résultat suivant démontré dans l'article [8] sur les nœud à capacité variable.

Lemme 5.3.7. *Soient* $F^{in}, C \in \mathcal{F}$ *et* $F^{out} : \mathbb{R}_+ \mapsto \mathbb{R}$ *tels que* $\forall t \geqslant 0, F^{out}(t) = \inf_{0\leqslant s\leqslant t} F^{in}(s) + C(t) - C(s)$. *Alors,* $F^{out} \in \mathcal{F}$ *et* $F^{out}(t) = F^{in}(start(t)) + C(t) - C(start(t))$.

Démonstration du lemme 5.3.6. Soit C la capacité de service du routeur. Comme c et c' appartiennent à une même période chargée, on a : $\forall c \leqslant t \leqslant c', C(t)-C(c) = \sum_i F_i^{(1)}(t) - F_i^{(1)}(c)$. Autrement dit, $C(t) - C(c)$ est le service global fourni entre les instants c et t.

De cette quantité de service on peut définir la trajectoire $\tilde{F}$ qui respecte les priorités. Comme les flux $f_1, \ldots, f_i$, $i \leqslant i_0$, sont plus prioritaires que $f_{i+1}, \ldots, f_{N_f}$ et que c marque le début de période chargée des flux $f_1, \ldots, f_i$ (d'après la propriété 5.4), on définit :

$$\sum_{1\leqslant k\leqslant i} \tilde{F}_k^{(1)}(t) = \inf_{c\leqslant s\leqslant t} \sum_{1\leqslant k\leqslant i} F_k^{(0)}(s) + C(t) - C(s). \tag{5.7}$$

Si $i = N_f$, cette formule assure que $\sum_{1\leqslant k\leqslant N_f} \tilde{F}_k^{(1)} = \sum_{1\leqslant k\leqslant N_f} F_k^{(1)}$.

Le service offert aux flux $f_i, \ldots, f_{N_f}$ entre les dates s et t est $C_i(t) - C_i(s) = C(t) - \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(t) - (C(s) - \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(s))$. De plus, on a $\tilde{F}_i^{(1)}(t) = \inf_{c\leqslant s\leqslant t} F_i^{(0)}(s) + C_i(t) - C_i(s)$. En effet, cette équation équivaut à l'équation suivante d'après la définition de $C_i(t) - C_i(s)$ précédente :

$$\begin{aligned}
\tilde{F}_i^{(1)}(t) + \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(t) &= \inf_{c\leqslant s\leqslant t} F_i^{(0)}(s) + C(t) - C(s) + \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}(s) \\
\sum_{k=1}^{i} \tilde{F}_k^{(1)}(t) &= \inf_{c\leqslant s\leqslant t} F_i^{(0)}(s) + C(t) - C(s) + \inf_{c\leqslant t'\leqslant s} \sum_{k=1}^{i-1} F_k^{(0)}(t') + C(s) - C(t'), \\
&\qquad \text{en utilisant l'équation 5.7} \\
\sum_{k=1}^{i} \tilde{F}_k^{(1)}(t) &= \inf_{c\leqslant t'\leqslant s\leqslant t} F_i^{(0)}(s) + \sum_{k=1}^{i-1} F_k^{(0)}(t') + C(t) - C(t') \\
\sum_{k=1}^{i} \tilde{F}_k^{(1)}(t) &= \inf_{c\leqslant s\leqslant t} \sum_{k=1}^{i} F_k^{(0)}(s) + C(t) - C(s), \\
&\qquad \text{car le minimum est atteint pour } s = t'.
\end{aligned}$$

On retrouve donc effectivement le service définit par l'équation 5.7.

L'équation 5.7 indique que $C_i = C - \sum_{k=1}^{i-1} \tilde{F}_k^{(1)}$ est croissante. En conséquence, d'après les propriétés des nœuds à capacité variable ([50]), $\tilde{F}_i^{(1)} \leqslant F_i^{(0)}$ et $\tilde{F}_i^{(1)}$ est croissante.

Ainsi, la trajectoire $(F_i^{(0)}, \tilde{F}_i^{(1)})$ est admissible et satisfait aux priorités dans l'intervalle $[c, c']$. □

Par conséquent, en appliquant le lemme 5.3.6 aux dates c_i et c_{i-1}, $1 \leqslant i \leqslant N_f$, on construit, à partir des valeurs des trajectoires en un ensemble fini de dates et des contraintes définies, une trajectoire dans la période chargée d'intérêt qui respecte les priorités.

Contraintes linéaires pour une topologie en arbre

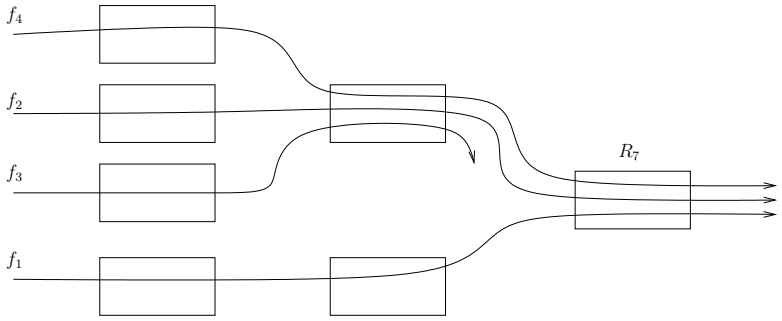


FIGURE 75 – Exemple de topologie en arbre.

Nous venons de définir les dates permettant d'assurer les priorités sur une période chargée. Dans ce paragraphe, nous définissons les contraintes linéaires dans le cas d'un réseau ayant une topologie en arbre. Commençons par introduire les notations spécifiques à l'étude de ce type de topologie.

Modèle du réseau ayant une topologie en arbre Nous nous focalisons sur les réseaux ayant une topologie en arbre, lorsque les flux sont dirigés vers la racine de l'arbre. Notre objectif est de calculer le pire délai du flux f_{N_f} lorsque celui-ci se termine à la racine de l'arbre. Dans la suite nous utilisons les notations suivantes :

- R_{N_r} est le routeur à la racine de l'arbre ;
- $next(j)$ dénote le numéro du routeur suivant R_j. Si, $j = N_r$ alors $next(j) = N_r+1$;

Rappelons que $prev_i(j)$ fournit le nom du routeur traversé par f_i avant de rentrer en R_j.

La figure 75 représente un exemple de topologie en arbre composé de 4 flux et 7 routeurs.

Contraintes linéaires Étudions par la théorie du network calculus le pire délai subi par un flux, lorsque les flux du réseau sont soumis à des priorités. Notre méthode reprend et étend les principes des programmes linéaires introduits en [9] en ajoutant les contraintes liées aux priorités entre les flux. En effet, la sous-section 5.3.3 montre que pour assurer les priorités entre les flux, il faut introduire des dates supplémentaires respectant un ensemble de propriétés. Nous présentons d'abord les variables du programme, puis l'objectif et enfin les contraintes linéaires appliquées aux variables.

L'ensemble des variables temporelles, noté V_t, rassemble les dates soumises à des contraintes. Nous les notons $c_i^{(j)}$ et t_u. Ceci fournit l'ensemble suivant d'au plus $N_r(N_f+1)$ dates.

- t_u désigne la date à laquelle la donnée d'intérêt (subissant le pire délai) entre dans le réseau ;
- $\forall j \in [1, N_r]$, $\forall i \in Fl(j) \cup \{0\}$, $c_i^{(j)}$ représentent les dates définissant les priorités sur la période chargée d'intérêt.

L'ensemble des variables fonctionnelles, noté V_f, correspond aux fonctions des arrivées et des sorties cumulées des flux du réseau en chaque routeur pour l'ensemble fini de dates V_t. L'ensemble V_f contient au pire $2N_f(N_f+1)N_r$ variables définies par :

- $F_m^{(0)}(t_u)$ la quantité de données de f_{N_f} entrée dans le routeur à la date t_u ;
- $\forall j \in [1, N_r]$, $\forall i \in Fl(j)$, $\forall k \in Fl(j) \cup \{0\}$, nous définissons les variables $F_i^{(0)}(c_k^{(j)})$, $F_i^{(j)}(c_k^{(j)})$ et $F_i^{(j)}(c_k^{(prev_i(j))})$.

L'objectif étant de calculer le délai maximal de f_{N_f}, on cherche à maximiser le temps entre l'arrivée et la sortie de la donnée étudiée : $\max c_0^{(N_r)} - t_u$.

Les contraintes linéaires sont listées dans les catégories suivantes afin de représenter les différentes contraintes du problème à résoudre.

- *Contrainte assurant que le délai calculé correspond au délai subi par une donnée* : $F_{N_f}^{(0)}(t_u) \geqslant F_{N_f}^{(N_r)}(c_0^{(N_r)})$;
- *Contraintes temporelles* : $\forall j \in [1, N_r],\ i, i' \in Fl(j) \cup \{0\},\ i > i'$,
$$c_i^{(j)} \leqslant c_{i'}^{(j)} \text{ et } c_0^{(j)} = c_{maxFl(next(j))}^{(next(j))} \qquad \text{(au plus } N_r(N_f+1) \text{ contraintes)};$$
- *Contraintes de causalité et de croissance sur les flux* : $\forall j, j' \in [1, N_r], j \geqslant j',\ \forall t \leqslant t' \in V_t$ et $F_i^{(j)}(t), F_i^{(j')}(t), F_i^{(j)}(t') \in V_f$,
$$F_i^{(j)}(t) \leqslant F_i^{(j')}(t) \qquad \text{(au plus } 2N_rN_f(N_f+1) \text{ contraintes)};$$
$$F_i^{(j)}(t) \leqslant F_i^{(j)}(t') \qquad \text{(au plus } 2N_rN_f^2 \text{ contraintes)};$$
- *Contraintes d'arrivées* : $\forall c_k^{(j)} \geqslant c_{k'}^{(j')} \in V_t,\ i \in Fl(j) \cap Fl(j')$,
$$F_i^{(0)}(c_k^{(j)}) - F_i^{(0)}(c_{k'}^{(j')}) \leqslant \alpha_i(c_k^{(j)} - c_{k'}^{(j')}) \qquad \text{(au plus } \sum_{i=1}^{N_f} |\alpha_i|(N_r(N_f+1))^2 \text{ contraintes)};$$
- *Contraintes de service* : $\forall j \in [1, N_r], \forall k, k' \in Fl(j) \cup \{0\},\ k < k'$,
$$\sum_{i \in Fl(j)} F_i^{(j)}(c_k^{(j)}) - F_i^{(j)}(c_{k'}^{(j)}) \geqslant \beta_j(c_k^{(j)} - c_{k'}^{(j)}) \qquad \text{(au plus} \sum_{j=1}^{N_r} |\beta_j|(N_f)^2 \text{ contraintes)};$$
- *Contraintes sur les périodes chargées* : $\forall j \in [1, N_r], \forall i \in Fl(j)$,
$$F_i^{(prev_i(j))}(c_{maxFl(j)}^{(j)}) = F_i^{(j)}(c_{maxFl(j)}^{(j)}) \qquad \text{(au plus } N_fN_r \text{ contraintes)};$$
- *Contraintes de priorités* : $\forall j \in [1, N_r], \forall k \in Fl(j) \cup \{0\}, \forall i \in Fl(j)$,
$$\forall i \leqslant k,\ F_i^{(j)}(c_k^{(j)}) = F_i^{(prev_i(j))}(c_k^{(j)}) \qquad \text{(au plus } N_rN_f(N_f+1) \text{ contraintes)};$$
$$\forall i \geqslant k+2,\ F_i^{(j)}(c_k^{(j)}) = F_i^{(j)}(c_{k+1}^{(j)}) \qquad \text{(au plus } N_rN_f(N_f+1) \text{ contraintes)}.$$

Remarque 5.3.8. *L'entier $k+2$ de la contrainte ci-dessus correspond à l'indice du flux deux fois moins prioritaire que f_k passant dans le routeur R_j. Cette grandeur n'existe pas toujours : lorsque $k = N_f$ et $k = N_f + 1$. Ceci constitue un abus de notation.*

Remarque 5.3.9. *On peut remarquer que $c_0^{(j)}$ et $c_{maxFl(next(j))}^{(next(j))}$ expriment la même date et que par conséquent il ne serait pas nécessaire de les conserver toutes les deux. Néanmoins cela simplifie l'expression des contraintes. Nous noterons également cette date $t_{next(j)}$ ultérieurement.*

Finalement, nous notons λ_{prio} l'ensemble de toutes les contraintes linéaires énoncées ci-dessus et d_{prio} la solution optimale obtenue par le solveur résolvant l'ensemble de contraintes λ_{prio}.

Le nombre total de contraintes est polynomial par rapport à la taille de la description du réseau étudié (nombre de flux et de routeurs).

Solution optimale Les contraintes linéaires établies définissent une trajectoire du réseau en un ensemble fini de points, correspondant à une solution optimale de l'objectif. Montrons maintenant que cette trajectoire satisfait aux contraintes du network

calculus et aux priorités. Nous avons déjà montré (lemme 5.3.6) qu'à partir d'un ensemble de points satisfaisant aux contraintes linéaires, on peut construire une trajectoire vérifiant les priorités durant une période chargée. Nous devons maintenant prouver que l'on peut extrapoler des fonctions des arrivées cumulées $F_i^{(0)}$ qui soient α_i-contraintes à chaque instant d'une période chargée et que chaque routeur R_j offre un service minimal β_j.

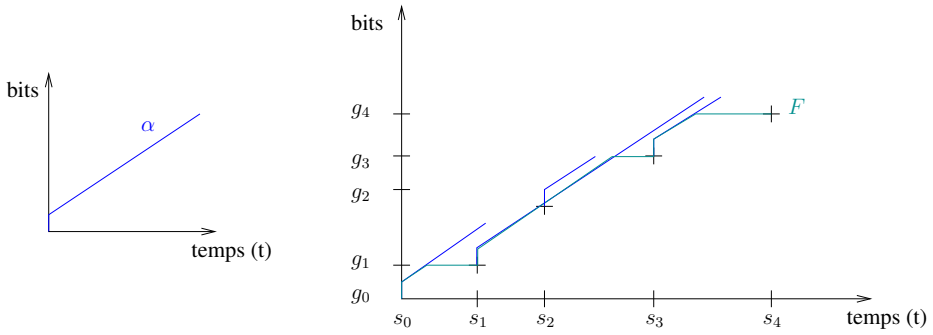


FIGURE 76 – Fonction des arrivées extrapolée d'un ensemble fini de points.

Lemme 5.3.10. *Soit α une fonction concave de $\mathcal{F}$, $\exists n \in \mathbb{N}$, $s_0 \leqslant s_1 \leqslant \cdots \leqslant s_n$ un ensemble fini de dates et $\{h_i\}_{0\leqslant i\leqslant n}$ un ensemble tels que*

$$i_1 < i_2 \Rightarrow 0 \leqslant h_{i_2} - h_{i_1} \leqslant \alpha(s_{i_1} - s_{i_2}).$$

Alors, on peut extrapoler de la suite h_i une fonction F qui soit croissante, α-contrainte telle que $F(s_i) = h_i$, $0 \leqslant i \leqslant n$.

Démonstration. Soit la fonction F telle que

$$F(t) = \min\{\min\{h_i + \alpha(t - s_i) \mid s_i \leqslant t\}, \min\{h_i \mid s_i \geqslant t\}\}.$$

La figure 76 représente un exemple de fonction F ainsi définie.

Remarquons que pour tout i, $i \in \{1, \ldots, n\}$, tel que $s_i \leqslant t$, on a $F(t) - h_i \leqslant \alpha(t - s_i)$. De plus, $F(s_i) = \min_{i\geqslant j} h_j + \alpha(s_i - s_j) \leqslant h_i$, dont le minimum est atteint pour $i = j$. Donc on obtient $F(s_i) = h_i$.

Posons deux dates quelconques t', t'' telles que $t' < t''$. Si $F(t') \neq F(t'')$, deux cas peuvent se produire :

- il existe i tel que $t' \leqslant s_i$ et $F(t') = h_i$. Alors, $F(t'') - F(t') = F(t'') - h_i \leqslant \alpha(t'' - s_i)$;
- ou bien, il existe i tel que $F(t') = h_i + \alpha(t' - s_i)$. Alors,
  $F(t'') - F(t') = F(t'') - h_i - \alpha(t' - s_i) \leqslant \alpha(t'' - s_i) - \alpha(t' - s_i)$.

Dans les deux cas $F(t'') - F(t') \leqslant \alpha(t'' - t')$. Par conséquent, la fonction F est croissante et α-contrainte. □

Ce lemme s'applique immédiatement aux fonctions des arrivées cumulées. Ce qui prouve qu'à partir des contraintes linéaires définies on peut extrapoler des fonctions $F_i^{(0)}$ qui soient croissantes et α_i-contraintes.

Il nous reste à prouver que le service fourni par un routeur quelconque R est une courbe de service strict β.

Lemme 5.3.11. *Soit β une fonction convexe appartenant à $\mathcal{F}$,$\exists n \in \mathbb{N}$, $s_0 \leqslant s_1 \leqslant \cdots \leqslant s_n$ une suite de dates et $\{h_i\}_{0\leqslant i\leqslant n}$ un ensemble tels que :*

$$i_1 < i_2 \Rightarrow h_{i_2} - h_{i_1} \geqslant \beta(s_{i_2} - s_{i_1}).$$

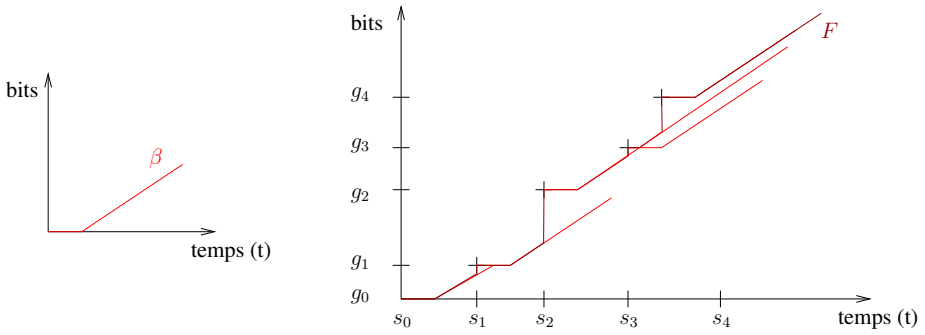


FIGURE 77 – Fonction des sorties cumulées extrapolée d'un ensemble fini de points.

Alors, on peut extrapoler de la suite h_i une fonction F qui soit croissante et β-contrainte avec $F(s_i) = h_i$, $0 \leq i \leqslant n$.

Démonstration. Soit la fonction F telle que :

$$F(t) = \max\{h_i + \beta(t - s_i) \mid t \geqslant s_i\}.$$

La figure 77 représente un exemple de fonction F ainsi définie.

Remarquons que pour tout $i \in \{1, \ldots, n\}$ tel que $s_i \leqslant t$, on a $F(t) - h_i \geqslant \beta(t - s_i)$. De plus, $F(s_i) = h_i$.

Posons deux dates quelconques t', t'' telles que $t' < t''$. Si $F(t') \neq F(t'')$. Il existe i tel que $F(t') = h_i + \beta(t' - s_i)$. Alors,
$F(t'') - F(t') = F(t'') - h_i - \beta(t' - s_i) \geqslant \beta(t'' - s_i) - \beta(t' - s_i) \geqslant \beta(t'' - t')$. Donc F est croissante et β-contrainte. □

Finalement ces calculs viennent de montrer le théorème suivant :

Théorème 5.3.12. *Soit un réseau ayant une topologie en arbre traversé par des flux ordonnés par des priorités. À partir de l'ensemble de contraintes λ_{prio}, on peut construire une trajectoire du réseau satisfaisant aux priorités, aux contraintes d'arrivées de chaque flux et aux contraintes de service de chaque routeur sur la période chargée durant laquelle la donnée d'intérêt réside dans le routeur. De plus, l'ensemble de contraintes est polynomial en le nombre de routeurs et de flux composant le réseau.*

Exemple 5.3.13. *La figure 68 représente le plus petit réseau permettant d'observer une différence entre le multiplexage arbitraire et celui à priorités fixes. En utilisant les propriétés $\sigma_1 = \sigma_2 = \sigma_3 = \sigma$, $\rho_1 = \rho_2 = \rho_3 = \rho$, $r_1 - \rho > r_2$ et $L_2 = 0$, on obtient les formules de délai*

$$\begin{aligned} d_{blind} &= \frac{r_2}{r_2 - \rho}\frac{\sigma + r_1 L_1}{r_1 - \rho} + \frac{2\sigma}{r_2 - \rho}, \\ d_{prio} &= \frac{2\sigma}{r_2 - \rho} + \frac{\sigma + r_1 L_1}{r_1 - 2\rho}, \end{aligned}$$

où d_{blind} (resp. d_{prio}) est le délai dans le cas d'un multiplexage aveugle (resp. avec priorités fixes).

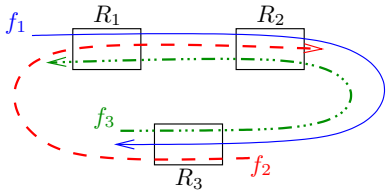


FIGURE 78 – Exemple de réseau de topologie quelconque.

Généralisation à une topologie quelconque

Traitons maintenant le cas d'un réseau de topologie quelconque. L'idée générale consiste à transformer la topologie de ce réseau afin d'obtenir une forme d'arbre par une méthode de dépliage. Les routeurs sont dupliqués afin de recréer une topologie en arbre.

La figure 78 représente un réseau composé de 3 routeurs et de 3 flux tel que : $path_1 = \langle 1, 2, 3\rangle$, $path_2 = \langle 3, 1, 2\rangle$, $path_3 = \langle 3, 2, 1\rangle$. La figure 79 représente le même réseau une fois déplié. L'idée générale du dépliement est la suivante. On s'intéresse au délai subi par le flux le moins prioritaire f_3. Pour cela, on doit considérer chaque routeur traversé par ce flux dans l'ordre indiqué par $path_3$ et calculer le service fourni à f_3 par chacun d'entre eux. Il correspond au service global du routeur auquel on soustrait le service alloué aux flux plus prioritaires. Or, le service fourni par un routeur R à un flux f dépend du chemin parcouru par f avant de rentrer dans R ainsi que des flux plus prioritaires. Par exemple, le service rendu à f_3 en R_3 dépend des arrivées de f_2 et f_1. Une borne des arrivées de f_2 se calcule immédiatement car il s'agit du premier routeur qu'il traverse, et dépend uniquement de α_2. Pour f_1, il faut considérer l'historique de ce flux et donc dupliquer tous les routeurs préalablement traversés. Ce chemin (et les routeurs dupliqués) sont représentés dans le rectangle en pointillé de la figure 79.

Le nom des routeurs dans un réseau déplié est plus complexe, comme ils sont dupliqués, et nécessite la définition d'un nouveau type de chemin : *chemin de priorité décroissante* (CPD). Un CPD $\langle (j_0, f_{i_0}), \ldots, (j_p, f_{i_p})\rangle$ est une séquence finie de paires de la forme (nom de routeur, nom de flux) telle que :

$$\begin{cases} \langle j_0, \ldots, j_p\rangle \text{ est un chemin du réseau,} \\ \forall 0 \leqslant k \leqslant p,\ f_{i_k} \in Fl(j_k), \\ \forall 0 \leqslant k < p,\ j_k = prev_{i_k}(j_{k+1}) \\ \forall 0 \leqslant k < p,\ i_k = \max\{i \leqslant k_{k+1} \mid i \in Fl(j_k)\},\ \text{est la priorité la plus faible des flux} \\ \quad \text{traversant le routeur } R_{j_k}. \end{cases}$$

Le nombre de CPD d'un réseau quelconque est fini. Les routeurs sont alors étiquetés par des CPDs terminant par le couple $(last_{N_f}, f_{N_f})$, où $last_{N_f}$ correspond au dernier routeur traversé par f_{N_f}. Afin de simplifier les notations sur la figure 79 les routeurs sont étiquetés par le premier couple du CPD. Par exemple, le CPD du premier routeur en haut à gauche est $\langle (1, f_1), (2, f_1), (3, f_3), (2, f_3), (1, f_3)\rangle$.

Soit $\langle j_1, \ldots j_{\ell_i}\rangle$ un chemin pour le flux f_i dans un réseau de topologie quelconque. Dans le réseau déplié, le flux f_i suit le chemin $\langle \Pi_1, \ldots \Pi_q\rangle$ tel que $\Pi_k = \langle (j_k, f_{i_k}), \ldots, (j_q, f_{i_q}), \Pi\rangle$ et Π est soit un chemin vide, ou $q = \ell_i$, ou $\Pi = \langle (j', f')\ldots\rangle$ avec $j' \neq j_{q+1}$ et $i_0 > i$. Par exemple, dans le réseau représenté sur la figure 78 le flux f_2 suit le chemin $\langle \Pi_1, \Pi_2\Pi_3\rangle$ tel que $\Pi_1 = \langle (3, f_3), (2, f_3), (1, f_3)\rangle$, $\Pi_2 = \langle (3, f_2), (1, f_2), (2, f_3), (1, f_3)\rangle$ et $\Pi_3 = \langle (3, f_2), (1, f_3)\rangle$.

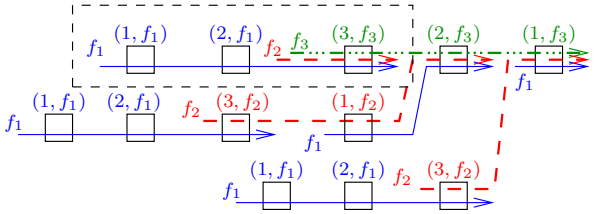


FIGURE 79 – Dépliement du réseau de la figure 78.

5.4 Exactitude des bornes calculées

Dans les paragraphes précédents, nous avons déterminé un moyen de calculer des délais de manière plus fine que dans le cas d'une politique de service aveugle. Néanmoins nous n'avons pas encore abordé le problème de l'exactitude des résultats calculés. Dans un premier temps, nous présentons un cas qui malheureusement ne retourne pas une borne exacte et nous montrons que l'algorithme 4 rend alors un meilleur résultat. Puis, nous introduisons une technique permettant d'améliorer la borne calculée. Nous finissons par mettre en évidence des cas pour lesquels notre méthode calcule une valeur exacte, ainsi qu'une méthode déterminant une borne minimale du délai.

Dans la suite nous notons d_{prio} (resp. d_{sfa}) le délai du flux f_{N_f} calculé par la méthode de contraintes linéaires avec priorités fixes (resp. la méthode SFA).

5.4.1 Étude complète d'un exemple

Dans cette partie, nous nous concentrons sur l'étude de l'exemple simple de la figure 80 afin de présenter les avantages et les limites de la méthode que nous venons d'introduire. Le réseau est composé de trois routeurs et trois flux où f_1 et f_3 traversent la totalité du réseau et f_2 ne passe qu'à travers R_2.

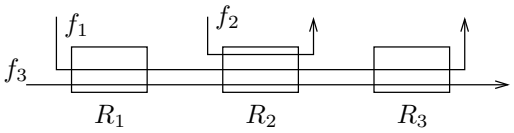


FIGURE 80 – Réseau de trois routeurs et de trois flux ayant une topologie en arbre.

Comparons nos résultats à ceux de la méthode SFA lorsque des priorités sont affectés aux flux, que l'on calcule avec l'algorithme 4. Dans tous les cas que nous avons étudiés dans la section 5.5 les délais obtenus par la méthode de programmation linéaire sont plus faible que ceux calculés par l'algorithme 4. Par exemple, avec les fonctions $\alpha_1(t) = 2t+2$, $\alpha_2(t) = 3t+3$, $\alpha_3(t) = 2$, $\beta_1(t) = 4(t-5)_+$, $\beta_2(t) = 8(t-4)_+$, $\beta_3(t) = 3(t-4)_+$, le délai calculé avec notre méthode est $d_{prio} = 52$ alors que l'on obtient $d_{sfa} = 60.6$ avec l'algorithme 4.

Cependant, si l'on utilise les fonctions $\alpha'_1(t) = t+1$, $\alpha'_2(t) = 2$, $\alpha'_3(t) = 1$, $\beta'_1(t) = 2t$, $\beta'_2(t) = 4t$, $\beta'_3(t) = 2t$. Notre méthode calcule $d_{prio} = 2$ alors que SFA obtient $d_{sfa} = 5/6$. En effet, la méthode SFA avec priorités peut fournir de meilleurs résultats que notre algorithme car notre méthode prend en compte les contraintes de service pour chaque routeur en une seule période chargée. Par exemple, les contraintes de service de R_1 sont étudiées uniquement durant la période chargée entre t_1 et t_2. Ainsi, entre les instants t_2

et t_3, les contraintes de service ne sont pas exprimées ce qui peut permettre une charge du flux f_1 qui ne devrait pas être autorisée.

Afin d'affiner la borne calculée par notre méthode, nous encodons les contraintes d'arrivées de chaque flux en chaque routeur qu'il traverse grâce aux résultats fournis par la méthode SFA lorsque les flux sont soumis à des priorités. Ainsi, cette méthode rendra une borne plus faible que la méthode SFA et que la méthode exacte pour les politiques aveugles. Néanmoins, il est peu probable d'obtenir une borne exacte vu que le principe consiste a mixer deux méthodes inexactes. Ajoutons les contraintes suivante : $\forall j \in [1, N_r],\ \forall i \in Fl(j),\ \forall k, k' \in Fl(j) \cup \{0\}, k \geq k'$,

$$F_i^{(j)}(c_{k'}^{(j)}) - F_i^{(j)}(c_k^{(j)}) \leq (\alpha_i^{(j)}(c_{k'}^{(j+1)} - c_k^{(j+1)})),$$

où $\alpha_i^{(j)}$ est calculée par l'algorithme 4. Nous dénotons λ_{sfa} cet ensemble de contraintes.

Théorème 5.4.1. *Soit $d_{prio-sfa}$ la solution optimale de l'ensemble de contraintes $\lambda_{prio} \cup \lambda_{sfa}$. Alors,*

$$d_{prio-sfa} \leq d_{sfa} \text{ et } d_{prio-sfa} \leq d_{prio}.$$

Le calcul du délai de la figure 80 donne $d_{prio-sfa} = 49$ avec le premier ensemble de fonctions et $d_{prio-sfa} = 5/6$ pour le deuxième (ce qui correspond effectivement au délai pire-cas).

5.4.2 Ajout de contraintes supplémentaires

Dans cette partie, nous tentons de calculer des bornes exactes en exprimant les contraintes de service et de priorités en chaque routeur durant chaque période chargée. Précédemment, les contraintes de service de R_j étaient uniquement exprimées entre les instants t_j et $t_{next(j)}$.

Ainsi, dans l'exemple de la figure 80, nous exprimons maintenant les contraintes de service de R_1 entre t_2 et t_3. Les contraintes n'ont pas besoin d'être introduite pour les routeurs R_1 et R_2 entre les instants t_3 et t_4 vu que durant cette période les routeurs doivent fournir un délai infini afin de maximiser le délai.

Ainsi il faut étudier la période chargée incluant la date t_3. Ceci requiert la définition d'une nouvelle date $t'_2 = start_1(t_3)$, où $start_1(t)$ correspond au début de période chargée du routeur R_1 incluant la date t. De plus, il faut introduire la date $c_1^{(1)'}$ afin d'exprimer les priorités dans R_1 entre t'_2 et t_3. Mais alors plusieurs cas peuvent se produire :

- $t_1 \leq c_1^{(1)} \leq t_2 \leq t'_2 \leq c_1^{(1)'} \leq t_3$, ou
- $t_1 = t'_2 \leq c_1^{(1)} = c_1^{(1)'} \leq t_2 \leq t_3$, ou
- $t_1 = t'_2 \leq c_1^{(1)} \leq t_2 \leq c_1^{(1)'} \leq t_3$.

Ces dates peuvent s'entremêler avec les valeurs $c_1^{(2)}$ et $c_2^{(2)}$, ce qui résulte en un total de 31 ordres envisageables. Hélas, tester ces différentes hypothèses avec un unique programme linéaire s'avère irréalisable. Il faudrait en effet créer un programme pour chaque ordre éventuel. Cette solution se révèle inappropriée étant donné que le nombre d'ordres présumés explose même pour des réseaux de petite taille.

5.4.3 Calcul de bornes exactes

Nous présentons ici deux cas pour lesquels la borne calculée grâce aux contraintes linéaires λ_{prio} est exacte. Nous nous cantonnons aux cas des topologies en arbre et nous supposons que les routeurs sont numérotés de sorte que $j < next(j)$.

Plus Proches Destination en Premier

La politique de *Plus Courtes Destination en Premier* (en anglais Shortest Destination First) (SDF) [56] est une politique à priorités fixes où un flux dispose d'une priorité plus élevée que tous les flux dont la destination dépasse sa propre destination. Les priorités sont définies comme suit :

$$\text{(SDF)} \qquad i_1 < i_2 \Rightarrow last_{i_1} \leq last_{i_2}.$$

Montrons que cette politique correspond à la pire politique de service pour notre modèle.

Théorème 5.4.2. *Le pire délai d'un réseau ayant une topologie en arbre et une politique aveugle peut être atteint lorsque le réseau est muni de la politique SDF.*

Démonstration. Soit $(F_i^{(j)})$ une trajectoire qui atteint le pire délai possible du réseau et modifions cette trajectoire, en une trajectoire $(\tilde{F}_i^{(j)})$, qui satisfait la politique SDF. Soit t_{N_r+1} la date à laquelle la donnée d'intérêt du flux f_{N_f} sort du réseau et $t_j = start_j(t_{next(j)})$.

(1) Avant le temps t_j, les ancêtres R_k du routeur R_j agissent comme des routeurs infinis : $\forall t \leq t_j$, $\tilde{F}_i^{(k)}(t) = \tilde{F}_i^{(k-1)}(t)$;

(2) Après l'instant $t_{next(j)}$, les descendants R_k du routeur R_j agissent comme des routeurs infinis : $\forall t \geq t_{next(j)}$, $\tilde{F}_i^{(k)}(t) = \tilde{F}_i^{(k-1)}(t) = F_i^{(0)}(t)$;

(3) Durant la période chargée $]t_j, t_{next(j)}]$, le service global du routeur R_j est le même que celui fourni à la trajectoire $(F_i^{(j)})$ et respecte les priorités définies par SDF.

La date t_j étant inchangée, le délai calculé est inchangé également. De plus, si la trajectoire est modifiée par (1) et (2), elle reste admissible. En effet, la charge de l'unique période chargée d'un routeur à un instant donné ne peut qu'augmenter comparée à celle de la trajectoire initiale. Ainsi on peut supposer que $(F_i^{(j)})$ satisfait déjà aux points (1) et (2). Il reste à prouver que la trajectoire $(\tilde{F}_i^{(j)})$ est admissible, autrement dit que $]t_j, t_{next(j)}]$ est une période chargée du routeur R_j.

Montrons par induction sur le nombre de routeurs que la charge transmise est au moins égale à celle de la trajectoire initiale. On note $\mathrm{Fl}(j)_{\geq k} = \mathrm{Fl}(j) \cap [k, N_f]$. On montre donc que pour un routeur R_j, quel que soit k,

$$\sum_{i \in \mathrm{Fl}(j)_{\geq k}} (\tilde{F}_i^{(prev_i(j))}(t_j) - \tilde{F}_i^{(j)}(t_j)) \geq \sum_{i \in \mathrm{Fl}(j)_{\geq k}} (F_i^{(prev_i(j))}(t_j) - F_i^{(j)}(t_j)).$$

Si R_j est une feuille de l'arbre l'inéquation est vraie. En effet, on a $prev_i(j) = 0$, les arrivées cumulées sont inchangées et $]t_j, t_{next(j)}]$ est la première période chargée de cette branche de l'arbre.

Hypothèse de récurrence : supposons que l'inégalité est vérifiée pour tout routeur appartenant à l'ensemble $\cup_{i \in Fl(j)} prev_i(j)$. Pour prouver que l'inégalité est toujours vraie pour le routeur R_j, il suffit de montrer que pour tout routeur $R_{j'}$, $j' = prev_i(j)$, l'inéquation est assurée à l'instant t_j (et non au temps $t_{prev_i(j)}$). Soit un indice arbitraire k, nous notons H, $\tilde{H}$, L et $\tilde{L}$ la charge transmise à l'instant $t_{j'}$ au routeur $R_{j'}$ pour les flux plus (moins) prioritaires que f_k pour la trajectoire $(F_i^{(j)})$ $((\tilde{F}_i^{(j)}))$.

Par induction, on sait que $L \leqslant \tilde{L}$ et $H + L \leqslant \tilde{H} + \tilde{L}$. Si, $\tilde{H} - H \geqslant 0$, alors le service fourni aux flux plus prioritaires augmente (grâce à SDF) et celui pour les flux moins prioritaires diminue. Sinon, la différence de service pour les flux plus prioritaires entre les trajectoires $(F_i^{(j)})$ et $(\tilde{F}_i^{(j)})$ est inférieure à $H - \tilde{H}$. Ainsi, le service apporté au flux moins prioritaires ne peut qu'augmenter de $H - \tilde{H} \leqslant \tilde{L} - L$, ce qui termine la preuve. □

Plus courte destination en dernier pour des chemins de longueur 2

Présentons un second cas d'étude pour lequel la borne calculée est exactement le pire délai : tous les flux traversent au maximum 2 routeurs et une priorité plus élevée est accordée au flux sortant plus tard que d'autres ($i_1 \leq i_2 \Rightarrow last_{i_1} \geq last_{i_2}$). Cette politique est nommée *Shortest Destination Last of size 2* (2SDL). Un exemple d'un tel réseau est représenté en figure 81.

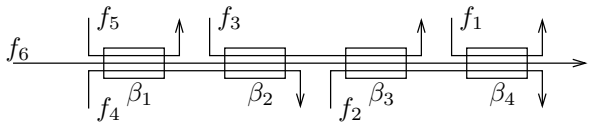


FIGURE 81 – Exemple de réseau soumis à la priorité 2SDL.

Théorème 5.4.3. *La solution optimale de λ_{prio} est exactement le délai pire-cas lorsque le réseau remplit les conditions de la politique de service à priorités fixes (2DSL).*

Démonstration. Soit $(F_i^{(j)})$ une trajectoire construite à partir de l'ensemble des contraintes linéaires λ_{prio}. Le délai calculé est inchangé lorsque le routeur R_j fournit un service infini après la date $t_{next(j)}$. Alors, le service est inchangé entre les instants $[t_j, t_{next(j)}[$ et le service fourni aux flux de priorités maximales est également inchangé puisqu'il s'agit d'un nouveau flux pour le routeur. Le changement apporté à la trajectoire est double : un service plus important peut être fourni à l'instant t_j pour les flux de priorités intermédiaires et les flux sortent du réseau à ce routeur R_j. Ceci n'implique alors aucune modification sur la charge transmise aux routeurs suivants.

Ainsi on peut construire une trajectoire admissible dont le délai est exactement la solution optimale de λ_{prio}. □

5.4.4 Borne inférieure du pire délai

Finalement, afin de déterminer si les délais que nous calculons sont proches des bornes exactes, nous définissons une méthode simple calculant une borne inférieure de ce délai. Ainsi, nous supposons que tous les routeurs fournissent un service infini après la première période chargée. Le délai calculé par cette méthode est noté d_{lbound}.

Pour définir un service infini pour chaque routeur après leur première période chargée, nous utilisons la contrainte linéaire suivante :

- $\forall j \in [1, N_r],\ \forall i \in [1, N_f]$ if $i \in Fl(j)$, $\forall k \in Fl(next(j)) \cup \{0\}, F_i^j(c_k^{(next(j))}) = F_i^0(c_k^{(next(j))})$.

Remarquons que cet ensemble de contraintes est polynomial et donc n'augmente pas la complexité de la méthode introduite.

Exemple 5.4.4. *Le délai calculé par cette méthode dans le cas du réseau représenté sur la figure 80, avec les courbes α'_1, α'_2, α'_3, β'_1, β'_2, β'_3, définies précédemment, est $d_{lbound} = 39.25$. Avec la meilleure méthode (qui combine les priorités fixes et la méthode SFA) on obtenait $d_{prio-sfa} = 49$. Cet exemple nous permet d'observer un écart non-négligeable entre la borne inférieure et la meilleure méthode.*

5.5 Résultats et comparaison aux méthodes existantes

Évaluons maintenant l'ensemble des méthodes proposées sur quelques exemples sur lesquels nous faisons varier la taille des réseaux, ainsi que leurs charges.

5.5.1 Mise en œuvre

Les travaux présentés dans ce chapitre ont été accompagné de l'implémentation d'un ensemble de programmes *Caml* générant automatiquement les contraintes linéaires exprimant une méthode donnée pour un réseau donné. Les approches étudiées sont listées ci-dessous. La notation utilisée pour désigner le délai calculé par chacune d'entre-elles est indiquée entre parenthèses.

- Méthode SFA à priorités fixes (d_{sfa}) ;
- Politique de service aveugle (d_{blind}) ;
- Politique de service à priorités fixes (d_{prio}) ;
- Politique de service à priorités fixes et contraintes SFA ($d_{prio-sfa}$) ;
- Méthode de calcul de borne inférieure (d_{lbound}).

L'unique paramètre utilisé est un fichier texte contenant une définition concise de la topologie ainsi que les courbes d'arrivées et de service du réseau à étudier. Ces programmes reprennent et étendent celui crée pour la réalisation des recherches présentées dans [9]. De part la construction du programme initial, celui-ci n'accepte actuellement que la définition de topologie en tandem.

Finalement, nous utilisons le solveur *LPSolve* [5] afin de résoudre les contraintes linéaires générées et ainsi calculer le délai maximal.

5.5.2 Étude d'un premier type de réseau

Étudions un premier type de réseaux composé de N_r routeurs et $N_r + 1$ flux. Nous appelons ce type de réseau, «réseau en tandem à flux emboités». Le flux f_i traverse les routeurs R_1 à R_{N_r-i+1} et f_{N_r+1} traverse le réseau entier. La figure 82 représente une telle topologie lorsque $N_r = 4$.

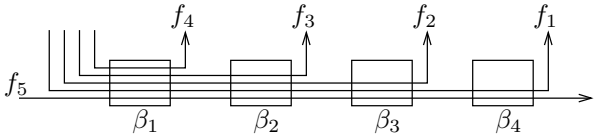


FIGURE 82 – Réseau de 4 routeurs en tandem à flux emboités.

Commençons par étudier l'effet du taux d'utilisation des routeurs sur la précision des résultats. Nous définissons la courbe de service du routeur R_j par $\beta_j(t) = 3(N_r - j + 1)(t - 0.1)_+$. Le temps de latence est 0.1 s et le taux de service est inversement linéaire au nombre de flux traversant le réseau. Chaque flux f_i (sauf le dernier) dispose des mêmes caractéristiques. Les messages arrivent en moyenne à une vitesse qui varie entre 0.3 Mbits/s et 2.9 Mbits/s. De plus, le flux instantané de message peut atteindre la valeur de 1 Mbits : $\alpha_i(t) = 1 + \rho_i t$, avec $\rho_i \in [0.3, 2.9]$. Le flux f_{N_r+1} transmet au réseau une donnée unique de 1 Mbits : $\alpha_{N_r+1}(t) = 1$.

La figure 83 représente les bornes maximales du délai du flux f_{N_r+1} calculées par ces méthodes lorsque le réseau est composé de 8 routeurs et que l'on fait varier la quantité de données arrivant dans le réseau entre 0.3 Mbits/s et 2.9 Mbits/s.

La figure 84 modélise la variation des délais calculés avec ces méthodes lorsque la taille du réseau varie de 2 à 10 routeurs. La courbe d'arrivée du flux f_i est $\alpha_i(t) = 1 + 2.5t$ si $i \neq N_f$ et $\alpha_{N_f}(t) = 1$.

Remarquons tout d'abord que la méthode SFA est bien plus pessimiste que les méthodes de programmation linéaire. Elle peut rendre des résultats 5 fois plus élevés que d_{prio}. De plus, l'inégalité $d_{prio-sfa} \leq d_{prio} \leq d_{blind}$ est satisfaite. En outre, l'écart entre la borne inférieure et $d_{prio-sfa}$ est assez faible. Ce qui illustre, dans ce cas, la finesse des

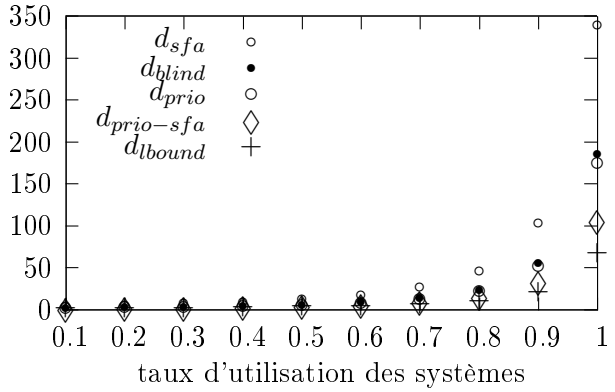


FIGURE 83 – Bornes maximales du délai pour un réseau de 8 routeurs en tandem composé de flux emboités.

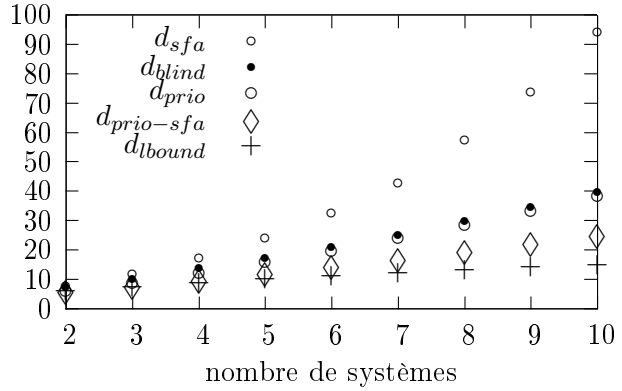


FIGURE 84 – Bornes maximales du délai pour un réseau en tandem composé de flux emboités lorsque le nombre de routeurs varie de 2 à 10.

bornes calculées par notre méthode, car nos calculs ne sont alors pas très éloignés du délai pire-cas.

5.5.3 Étude d'une seconde topologie

Présentons un second cas d'étude. Nous avons déjà partiellement étudié ce type de topologie dans la partie 5.4.3 où nous avons prouvé (théorème 5.4.3) que notre méthode atteignait le délai pire-cas. Ainsi, cette étude ne présente que les résultats d_{sfa}, d_{blind} et d_{prio} ($d_{prio} = d_{prio-sfa} = d_{lbound}$).

Le réseau est composé d'une succession de N_r routeurs et de $N_r + 2$ flux. Chaque flux traverse exactement deux routeurs, sauf le premier et l'avant-dernier qui n'en traversent qu'un et f_{N_f} qui traverse l'ensemble des routeurs. Chaque flux débute en un routeur différent, et les flux sont numérotés de sorte que $i_1 \leq i_2 \Rightarrow last_{i_1} \geq last_{i_2}$. La figure 85 représente un tel réseau composé de 4 routeurs.

Dans cette analyse, le routeur R_j est muni d'une courbe de service $\beta_j(t) = 10(t-5)_+$, le flux f_i dispose de la courbe d'arrivée $\alpha_i(t) = 1 + 2t$ si $i \neq 1$, $\alpha_1(t) = 1 + 4t$ sinon.

Les bornes maximales de délai calculées par les méthodes pour cet exemple sont représentées sur la figure 86.

La méthode SFA montre toujours une sur-approximation importante. De plus, d_{blind}

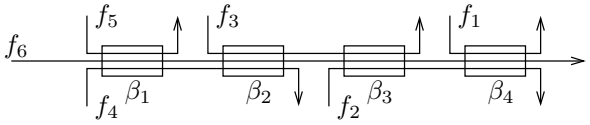


FIGURE 85 – Réseau de 4 routeurs soumis à la propriété 2SDL.

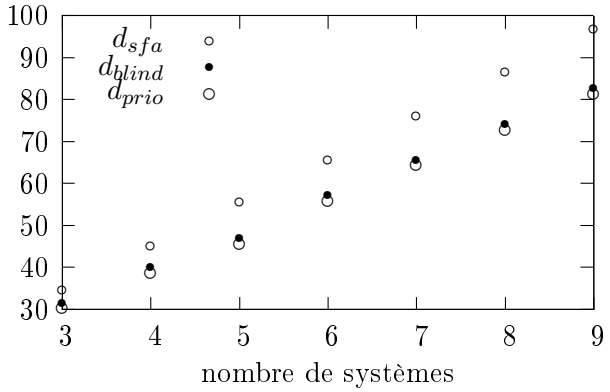


FIGURE 86 – Bornes maximales du délai pour un réseau soumis à la propriété 2SDL dont la taille varie.

est très proche du délai pire-cas (d_{lbound}).

Ceci montre que la programmation linéaire avec priorité et contrainte SFA constitue une méthode efficace pour calculer le délai pire-cas des réseaux dont la politique de service est à priorités fixes.

Conclusion du chapitre

Dans ce chapitre nous avons développé plusieurs méthodes appliquant la technique de programmation linéaire à la théorie du network calculus pour les réseaux à politique de service à priorités fixes. Pour les topologies en arbre les méthodes fournissent de bons résultats, même si elles calculent parfois une sur-approximation. Dans le cas général, les routeurs étant dupliqués afin de maintenir les contraintes de tous les flux, les résultats sont plus pessimistes.

Les algorithmes proposés sont de complexité temporelle polynomiale. Néanmoins, les résultats peuvent être difficiles à obtenir pour de grandes topologies.

Notre étude se restreint au contexte de courbes d'arrivées concaves affines par morceaux et de courbes de service convexes affines par morceaux. Même s'il s'agit d'un cas très intéressant qui correspond à une modélisation réaliste des arrivées et des services des réseaux, il serait intéressant de prévoir l'application de d'autres types de fonctions.

Finalement, cette étude ne considère que les politiques de service à priorités fixes. Un travail intéressant serait d'appliquer cette méthode à d'autres politiques de services (GPS (*Generalized Processor Sharing*), FQ (*Fair Queuing*), etc.). Ceci à été récemment étudié par Bouillard et Stea ([10]) dans le cas de la politique de service FIFO.

Conclusion et perspectives

Dans ce travail de thèse, nous nous sommes intéressés à l'apparition de comportements dangereux dans les réseaux de télécommunication. Ce travail a été motivé par le fait que les opérateurs ont encore bien du mal à garantir la stabilité de leurs réseaux, ce qui leur cause un fort préjudice pécuniaire. Dans notre cheminement, nous avons développé deux axes majeurs.

Tout d'abord, nous nous sommes penchés sur la prédiction de comportements menaçant la santé des réseaux.

Nous avons établi une première méthode analysant des flux de messages quelconques. Cette approche, aussi rapide en calcul que légère en espace mémoire, permet de détecter les comportements d'intérêts par le calcul d'indicateurs de stabilité en temps-réel. Ceci autorise une analyse proactive qui permet à l'administrateur réseau de réagir durant l'observation d'une instabilité, avant qu'une défaillance n'apparaisse.

Dans une deuxième phase, nous avons approfondi notre étude sur les flux d'alarmes. Le défi résidait dans la capacité à extraire de ces données opulentes des informations concises. Nous avons construit une méthode de corrélation prédictive, extrayant des séquences d'ensembles d'alarmes significatives. Ces séquences permettent de déterminer les sources de problèmes et d'observer leur évolution.

Nous nous sommes ensuite intéressés au lien entre l'affectation de certains paramètres et les délais de messages.

Notre première préoccupation était de caractériser les relations entre les messages des protocoles périodiques et d'en déduire des conditions sur leurs paramètres pouvant mener à la congestion du réseau (délai infini). Il s'agissait également de rechercher la définition de certains paramètres réduisant les délais des messages. Dans ce but, nous avons présenté une modélisation du protocole OSPF par réseau de Petri temporisé. Une approche combinant la simulation du modèle défini à une heuristique recherchant une définition de paramètres approchant la solution optimale a été présenté.

Une seconde préoccupation consistait à améliorer l'estimation des délais maxima, pour les réseaux dont les serveurs sont soumis à une politique de service à priorités fixes, en fonction de la capacité de traitement des routeurs et de la vitesse d'arrivée des messages. Cette approche consistait à encoder les contraintes de la théorie du network calculus dans un programme linaire. Notre méthode comparée aux solutions existantes obtient des résultats significativement meilleurs. Une amélioration complémentaire a été apportée par la composition de notre travail et de travaux complémentaires existants.

De nombreuses perspectives apparaissent à la suite de ce travail de thèse. Nous souhaitons en présenter ici certaines, relatives aux limites des travaux développés.

Les méthodes fournissant des expertises du comportement de réseaux ont prouvé leur efficacité sur un ensemble de cas d'étude. Cependant, nous n'avons pas étudié l'étendue de leurs capacités. On ne peut décemment pas affirmer qu'elles détectent précisément toutes les anomalies possibles. En effet, il semble naturel que la qualité des résultats obtenus soit moins précise que celle fournie par des études proposant une analyse profonde s'appliquant, par exemple, à un contexte particulier. Il est également clair que l'ensemble des erreurs potentielles ne peut pas être répertorié et testé. On pourrait envisager de créer une plateforme de test sur laquelle les paramètres pourraient être facilement modifiés afin d'observer de nombreuses éventualités. Mais étant donné l'étendue des comportements des réseaux, obtenir ainsi une expertise des capacités de nos techniques constitue une tâche complexe.

Ce travail ayant été réalisé dans le but d'être appliqué par les opérateurs réseaux, une tâche importante consiste également à développer ces méthodes dans des réseaux de compagnies de télécommunication. Ceci permettrait également de fournir une première observation de l'amélioration de la stabilité des réseaux mais également de déterminer des instabilités résiduelles. Comment déterminer alors les flux fournissant les meilleures observations ? Faut-il s'attacher à un protocole particulier ? Vaut-il mieux observer des flux regroupant un ensemble de protocoles ? Telles sont les questions qu'il faudra alors prendre en compte pour cette étude.

Une autre limite concerne le fait que ces méthodes n'étudient qu'un seul flux à la fois. Si l'on veut analyser les flux entre routeurs cela entraine une certaine restriction. En effet, l'observation des indicateurs étant manuelle, l'analyse de plusieurs flux devient vite ingérable. Afin de pallier cette lacune, il faut développer des méthodes automatiques afin que l'administrateur réseau n'ait qu'une seule source d'information à observer. Par exemple, pour la méthode calculant les indicateurs de stabilités, une première solution possible consisterait à corréler les résultats de chaque flux afin de transmettre des indicateurs de stabilité du réseau complet. On pourrait également envisager une méthode analysant les résultats et avertissant uniquement d'un danger réel. Par exemple, elle pourrait reconnaître les comportements déviants, ou de fréquentes et fortes variations. Malheureusement, conserver une approche générique pourra difficilement détecter toutes les observations d'un administrateur réseau. On pourrait trancher entre ces arguments et proposer une solution pour les flux à caractère périodique. Celle-ci pourrait disposer d'une simple connaissance initiale constituée des indicateurs de stabilité calculés durant une période non perturbée. Ainsi, l'algorithme calculerait des indicateurs de stabilité en temps-réel, les comparerait à ceux initialement établis (valeurs de références) et émettrait une alerte lorsque l'écart entre ces valeurs dépasserait un seuil donné.

Définir des plages de performances optimales de paramètres d'un protocole est un problème NP-complet auquel nous nous sommes heurtés. Nous avons donc développé une heuristique afin d'approcher des valeurs optimales. Mais une grande marge de manœuvre semble encore possible. En effet, la taille de période d'inondation actuellement définie par l'heuristique est très supérieure à celle couramment utilisée. Ceci est dû aux simplifications du problème initial que nous avons apportées et à l'emploi de l'heuristique. On pourrait rechercher des améliorations de cette dernière en employant par exemple une méthode de recuit simulé. S'agissant d'une définition statique des paramètres, envisager une étude longue n'est pas un problème et pourrait apporter une amélioration importante.

Actuellement, l'affectation des paramètres proposée s'effectue lors de l'établissement du réseau. Cependant, les études présentées dans la première partie de cette thèse montrent que les comportements de ces derniers varient fréquemment. En conséquence, il

serait intéressant de prévoir une variation de cette méthode afin de pouvoir modifier la définition des paramètres en temps-réel. On pourrait également envisager une combinaison avec l'algorithme 2 afin de déterminer les instants auxquels des changements de paramètres sont utiles.

Le calcul de délai maximal à l'aide de la théorie de network calculus et de la programmation linéaire montre une grande amélioration comparée aux précédentes méthodes. Un écart important demeure cependant entre la valeur trouvée et la borne inférieure. Ainsi, déterminer la précision de la borne obtenue reste un problème ouvert.

Finalement, les résultats présentés dans la première partie de cette thèse a suscité un vif intérêt de la part de nos collaborateurs d'Alcatel-Lucent Bell-labs. Ils nous ont d'ailleurs accompagnés dans la création d'un brevet afin de protéger l'invention présentée dans le chapitre 2. Cette éventualité a également été formulée pour la technique introduite dans le chapitre suivant. Elle n'a cependant pas abouti par manque de temps. Monsieur Benoît Ronot travaille actuellement à l'intégration de ces travaux dans les produits des clients d'Alcatel-Lucent Bell-labs. Il semble particulièrement enthousiaste face aux résultats de corrélation d'alarmes qui fournissent, selon lui, des analyses complémentaires aux techniques actuellement employées. Malheureusement, les résultats présentés dans la seconde partie de cette thèse éveillent bien moins l'intérêt des ingénieurs d'Alcatel-Lucent Bell-labs qui n'y voient pas d'application immédiate répondant aux besoins de l'entreprise.

Bibliographie

[1] A. Amaral, B. Zarpelão, L. de Souza Mendes, J. Rodrigues, and M. Proença. Inference of network anomaly propagation using spatio-temporal correlation. *Journal of Network and Computer Applications*, 35(6) :1781–1792, 2012.

[2] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proc. of the 2nd ACM SIGCOMM Workshop on Internet measurment (IMW'02)*, pages 71–82, 2002.

[3] A. Basu and J. Riecke. Stability issues in OSPF routing. In *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*, pages 225–236, New York, NY, USA, 2001. ACM.

[4] A. Benveniste, E. Fabre, C. Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE TRANS. ON AUTOMATIC CONTROL*, 2001.

[5] M. Berkelaar, K. Eikland, and P. Notebaert. lp_solve : Open source (mixed-integer) linear programming system. *Eindhoven U. of Technology*, 2004.

[6] J.-C. Bolot and A. Shankar. Dynamical behavior of rate-based flow control mechanisms. *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'90)*, 20(2) :35–49, 1990.

[7] F. Bonomi, D. Mitra, and J. Seery. Adaptive algorithms for feedback-based flow control in high speed, wide area atm networks. *IEEE Journal on Selected Areas in Communications*, 13 :1267–1283, 1995.

[8] A. Bouillard, L. Jouhet, and É. Thierry. Comparison of different classes of service curves in network calculus. Technical report, 2010.

[9] A. Bouillard, L. Jouhet, and É. Thierry. Tight performance bounds in the worst case analysis of feed forward networks. In *Proc. of the International Conference on Computer Communications (INFOCOM'10)*. IEEE, 2010.

[10] A. Bouillard and G. Stea. Exact worst-case delay for FIFO-multiplexing tandems. In *Proc. of International Conference on Performance Evaluation Methodologies and Tools (Valuetools'12)*, pages 158–167, 2012.

[11] C. Chang. *Performance Guarantees in Communication Networks*. TNCS, Springer-Verlag, 2000.

[12] C. Chao, D. Yang, and A. Liu. An automated fault diagnosis system using hierarchical reasoning and alarm correlation. *Journal of Network and Systems Management*, 9 :183–202, 2001.

[13] T. Chyssler, S. Burschka, M. Semling, T. Lingvall, and K. Burbeck. Alarm reduction and correlation in intrusion detection systems. In *Proc. of the Detection of Intrusions and Malware Vulnerability Assessment (DIMVA'04)*, pages 9–24, 2004.

[14] D. Comer. *Computer Networks and Internets*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2008.

[15] FlexRay Consortium et al. Flexray communications system-protocol specification. *Version*, 2(1) :198–207, 2005.

[16] J.-M. Couvreur, E. Encrenaz, E. Paviot-Adet, D. Poitrenaud, and P.-A. Wacrenier. Data decision diagrams for Petri net analysis. In *Proc. of the 23th International Conference on Application and Theory of Petri Nets (ICATPN'02)*, volume 2360, pages 101–120. Springer Verlag, 2002.

[17] R. Davis, H. Shrobe, W. Hamscher, K. Wieckert, M. Shirley, and S. Polit. Diagnosis based on description of structure and function. In *Second National Conference on Artificial Intelligence (AAAI, 1982)*, pages 137–142, 1982.

[18] Y. Dinitz, N. Garg, and M. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19(1) :17–41, 1999.

[19] C. Dousson and T. Vu Duong. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *Proc. of the International Joint Conference on Artifical Intelligence (IJCAI'99)*, pages 620–626, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[20] G. Gardey, D. Lime, M. Magnin, and O. Roux. Roméo : A Tool for Analyzing time Petri nets. In *Proc. of the 17th International Conference on Computer Aided Verification (CAV'05)*, pages 418–423. Springer, 2005.

[21] J.-P. Georges, E. Rondeau, and T. Divoux. Evaluation of switched ethernet in an industrial context by using the network calculus. In *Proc. of 4th IEEE Workshop on Factory Communication Systems (WFCS'02)*, pages 19–26, 2002.

[22] D. Ghita, H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran. Netscope : Practical network loss tomography. In *Proc. of the International Conference on Computer Communications (INFOCOM'10)*, pages 1 –9. IEEE, 2010.

[23] J. Grieu. *Analyse et évaluation de techniques de communication Ethernet pour l'interconnexion des systèmes avioniques.* PhD thesis, INPT, Toulouse, 2004.

[24] R. Guerin, A. Orda, and D. Williams. QoS routing mechanisms and OSPF extensions. In *Proc. of the Global Communication Conference Exhibition and Industry Forum (GLOBECOM'96)*, pages 1903–1908, 1996.

[25] T. Guyet and R. Quiniou. Mining temporal patterns with quantitative intervals. In *Proc. on the 4th International Workshop on Mining Complex Data, (ICDM'08) Workshop*, page 10. IEEE, 2008.

[26] T. Guyet and R. Quiniou. Extracting temporal patterns from interval-based sequences. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI'11)*. Morgan Kaufmann Publishers Inc., 2011.

[27] T. Guyet and R. Quiniou. Incremental mining of frequent sequences from a window sliding over a stream of itemsets. In *Actes des 6èmes Journées Intelligence Artificielle Fondamentale*, pages 153–162, 2012.

[28] S. Haar, A. Benveniste, E. Fabre, and C. Jard. Fault diagnosis for distributed asynchronous dynamically reconfigured discrete event systems. In *In Proc. of the 16th IFAC World Congress*, 2005.

[29] B. Halabi. *Internet Routing Architectures.* Cisco Press, Indianapolis, IN, 1997.

[30] H. Han, C. Hollot, D. Towsley, and Y. Chait. Synchronization of TCP flows in networks with small droptail buffers. In *Proc. of the 44th IEEE CDC-ECC*, pages 6762 – 6767, 2005.

[31] T. Herpel, K.-S. Hielscher, U. Klehmet, and R. German. Stochastic and deterministic performance evaluation of automotive CAN communication. *Journal of Computer Network*, 53(8) :1171–1185, 2009.

[32] B. Hollifield and E. Habibi. *The Alarm Management Handbook : Seven Effective Methods for Optimum Performance.* ISA, 2007.

[33] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, pages 99–110. ACM, 2003.

[34] Control Arts Inc. Alarm system engineering. In *http ://www.controlartsinc.com/Support/Publications.html*, 2010.

[35] J. Jackson and G. Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21(3) :341–349, 1979.

[36] Gabriel Jakobson and Mark Weissman. Alarm correlation. *Network, IEEE*, 7(6) :52–59, 1993.

[37] J. Jung, B. Krishnamurthy, and M. Rabinovich. Flash Crowds and Denial of Service Attacks : Characterization and Implications for CDNs and Web Sites. In *Proc. of the 11th International World Wide Web Conference (WWW'02)*, 2002.

[38] J. Jung, V. Paxson, A. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy*, 2004.

[39] M. Katz, K. Kompella, and D. Yeung. Traffic Engineering (TE) Extensions to OSPF Version 2, 2003. Updated by RFC 4203.

[40] I. Katzela and M. Schwartz. Schemes for fault identification in communication networks. *IEEE/ACM Transactions on Networking*, 3(6) :753–764, December 1995.

[41] F. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks : shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, volume 49, 1998.

[42] Z. Kerravala. As the value of enterprise networks escalates, so does the need for configuration management. *Enterprise Computing and Networking, The Yankee Group*, 2004.

[43] H. Kim and B. Karp. Autograph : toward automated, distributed worm signature detection. In *Proc. of the 13th conference on USENIX Security Symposium (SSYM'04)*, page 19. USENIX Association, 2004.

[44] C. Kreibich and J. Crowcroft. Honeycomb : creating intrusion detection signatures using honeypots. *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04)*, 34 :51–56, 2004.

[45] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection : methods, evaluation, and applications. In *Proc. of the 3rd ACM SIGCOMM conference on Internet measurement (IMC'03)*, pages 234–247. ACM, 2003.

[46] O. Kummer, F. Wienberg, M. Duvigneau, M. Kohler, D. Moldt, and H. Rolke. Renew-the reference net workshop. In *Proc. of the 24th International Conference on Application and Theory of Petri Nets (ATPN'03)*, 2003.

[47] C. Labovitz, G. R. Malan, and F. Jahanian. Origins of internet routing instability. In *Proc. of the International Conference on Computer Communications (INFOCOM'99)*, pages 218–226, 1999.

[48] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04)*, 34 :219–230, 2004.

[49] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'05)*, 35 :217–228, 2005.

[50] J.-Y. Le Boudec and P. Thiran. *Network Calculus : A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, 2001. revised version 4, 2004.

[51] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. *Performance Evaluation*, 63(9-10) :956–987, 2006.

[52] X. Li, F. Bian, H. Zhang, C. Diot, R. Govindan, W. Hong, and G. Iannaccone. MIND : A distributed multi-dimensional indexing system for network diagnosis. In *Proc. of the International Conference on Computer Communications (INFOCOM'06)*. IEEE, 2006.

[53] X. Liu and A. Goldsmith. Effects of communication delay on string stability in vehicle platoons. In *IEEE Proc. Intelligent Transportation Systems Council (ITSC'01)*, 2001.

[54] M. Lothaire. *Algebraic combinatorics on words*. Cambridge university press, 2002.

[55] G. Malkin. RFC 2453 RIP v2, 1998.

[56] M. Mavronicolas. Stability in routing : Networks and protocols. *Bulletin of the EATCS*, 74 :119–134, 2001.

[57] L. Moreau. Stability of multiagent systems with time-dependent communication links. 50(2) :169–182, 2005.

[58] J. Moy. RFC 2328 OSPF v2, 1998.

[59] O. Oyerinde and S. H. Mneney. Subspace tracking-based decision directed cir estimator and adaptive cir prediction. *Proc. of the IEEE Transactions on Vehicular Technology*, 61(5) :2097–2107, 2012.

[60] M. Proença, B. Zarpelão, and L. de Souza Mendes. Anomaly detection for network servers using digital signature of network segment. In *Proc. of the AICT/SAPIR/ELETE*, pages 290–295, 2005.

[61] Quagga. A routing software package for TCP/IP networks, http ://www.nongnu.org/quagga/.

[62] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. *Proc. of the SIGMETRICS'07 Performance Evaluation Review*, 35 :109–120, 2007.

[63] I. Rouvellou and G. Hart. Automatic alarm correlation for fault identification. In *Proc. of the International Conference on Computer Communications (INFOCOM'95)*, pages 553–561. IEEE, 1995.

[64] G. Rétvári, F. Németh, R. Chaparadza, and R. Szabó. OSPF for implementing self-adaptive routing in autonomic networks : A case study. In Yacine Ghamri-Doudane, editor, *MACE*, volume 5844 of *Lecture Notes in Computer Science*, pages 72–85. Springer, 2009.

[65] L. Saidane, S. Azzaz, S. Martin, and P. Minet. FP/FIFO scheduling : deterministic versus probabilistic QoS guarantees and p-schedulability. In *Proc. of the IEEE International Conference on Communications (ICC'07)*, 2007.

[66] P. Sanyal and C. Shen. Bayes decision rule for rapid detection and adaptive estimation scheme with space applications. In *Proc. of the IEEE Trans. Automat. Contr.*, volume AC-19, pages pp 228–231, 1974.

[67] J. Schmitt, F. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing. Technical report, University of Kaiserslautern, 2007.

[68] B. Swaminathan. Agile methodologies overview. http ://www.slideshare.net/Siddhi/intro-to-agile, 2007.

[69] A. Tangirala, S. Shah, and N. Thornhill. PSCMAP : A new tool for plant-wide oscillation detection. *Journal of Process Control*, 15 :931–941, 2005.

[70] L. Thiele, S. Chakraborty, M. Gries, A. Maxiaguine, and J. Greutert. Embedded software in network processors models and algorithms. In *Proc. of International conference on embedded software (EMSOFT'01)*. ACM, 2001.

[71] H. Van Trees. *Detection, estimation, and modulation theory*. Wiley. com, 2004.

[72] G. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 359–363. AAAI Press, 1998.

[73] T. White and N. Ross. An architecture for an alarm correlation engine. In *Object Technology*, 1997.

[74] A. Willsky, J. Deyst, and B. Crawford. Adaptive filtering and self-test methods for failure detection and compensation. In *Proc. of the Joint American Control Conference*, pages 637–645, 1974.

[75] A. Willsky and H. Jones. A generalized likelihood ratio approach to state estimation in linear systems subjects to abrupt changes. In *Proc. of the 13th IEEE Conference on Decision and Control (CDC'74)*, pages 846 –853, 1974.

[76] A. Willsky and H. Jones. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Transactions on Automatic Control*, 21(1) :108 – 112, 1976.

[77] T. Li Y. Rekhter. RFC 1771 A Border Gateway Protocol 4 (BGP-4), 1995.

[78] Houssame Yahiaoui. Sabis : Simulation des instabilités du protocole bgp. Technical report, 9ième Atelier en Evaluation de Performances, 2008.

[79] F. Yang, S.L. Shah, and D. Xiao. Correlation analysis of alarm data and alarm limit design for industrial processes. In *Proc. of the American Control Conference (ACC'10)*, pages 5850–5855, 2010.

[80] Fan Yang. Improved correlation analysis and visualization of industrial alarm data. In *18th IFAC World Congress*, pages 12898–12903, 2011.

[81] S. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. High speed and robust event correlation. *IEEE communications Magazine*, 34(5) :82–90, 1996.

[82] B. Zarpelão, L. de Souza Mendes, M. Proença, and J. Rodrigues. Parameterized anomaly detection system with automatic configuration. In *Proc. of the Global Communication Conference Exhibition and Industry Forum (GLOBECOM'09)*, pages 1–6, 2009.

Bibliographie personnelle

[83] Anne Bouillard, Claude Jard, and Aurore Junier. Some synchronization issues in OSPF routing. In *Proc. of DCNET'13*, 2013.

[84] Anne Bouillard and Aurore Junier. Worst-case delay bounds with fixed priorities using network calculus. In *Proc. of Valuetools'11*, 2011.

[85] Anne Bouillard, Aurore Junier, and Benoît Ronot. Hidden anomaly detection in telecommunication networks. In *Proc. of CNSM'12*, 2012.

[86] Anne Bouillard, Aurore Junier, and Benoît Ronot. Hidden anomaly detection in telecommunication networks. Rapport de recherche RR-7979, INRIA, 2012.

[87] Anne Bouillard, Aurore Junier, and Benoît Ronot. Patent : Monitoring of a communication device over time. In *ALCATEL-LUCENT / INRIA*, 2012.

[88] Anne Bouillard, Aurore Junier, and Benoît Ronot. Alarms correlation in telecommunication networks. Rapport de recherche RR-8321, INRIA, 2013.

[89] Anne Bouillard, Aurore Junier, and Benoît Ronot. Impact of rare alarms on event correlation. In *Proc. of CNSM'13*, 2013.