Table des matières

Présentation des travaux de thèse **1**

1 Introduction *1*

1.1 Description des sujets étudiés *1*

1.2 Contributions de la thèse *3*

2 Modèles d'inférence pour l'estimation de caractéristiques réseaux *5*

2.1 La file d'attente $M+M/M/1/K$ *6*

2.2 La file d'attente $M+M/D/1/K$ *7*

2.3 Estimation des paramètres *7*

2.4 Analyse des résultats *8*

2.5 Conclusion *9*

3 Estimation de la taille de groupes multipoints *9*

3.1 Estimation optimale à base de filtre de Kalman *10*

3.2 Estimation optimale à base de filtre de Wiener *11*

3.3 Estimation efficace à base de filtre linéaire d'ordre 1 . . . . . *13*

3.4 Validation des modèles *13*

3.5 Conclusion *14*

4 Analyse de deux mécanismes de communication dans un environnement à code mobile *14*

4.1 Mécanisme distribué à base de répéteurs *15*

4.2 Mécanisme centralisé à base de serveur de localisation *16*

4.3 Validation des modèles et évaluation des performances *16*

4.4 Extension au cas de multiples paires source-agent *18*

4.5 Conclusion *19*

5 Conclusions et perspectives *19*

Présentation des travaux de thèse

1 Introduction

"Ce qui a débuté comme un exercice de paranoïa militaire est devenu un moyen de communications globales"[1] [2].

L'Internet est un réseau mondial reliant des réseaux d'ordinateurs et plusieurs centaines de millions d'utilisateurs de par le monde. La croissance exponentielle de l'Internet en nombre de réseaux ou d'utilisateurs pose de réels problèmes en termes de performance et de contrôle du réseau. L'un des principaux concepts de base qui ont contribué à la popularité de l'Internet et à son développement est connu sous le nom de "l'argument de bout-en-bout". Cette philosophie consiste à garder le réseau le plus simple possible et à déployer les fonctionnalités complexes aux extrémités de celui-ci [108, 37, 66]. Bien que ce principe de base fut en grande partie responsable du succès de l'Internet, il rend problématique d'obtenir des informations sur l'état interne du réseau.

Si jamais une fonctionnalité devait être déployée dans l'Internet, deux possibilités s'offrent au concepteur : soit de l'ajouter au cœur du réseau, soit de l'implémenter dans la couche application. Il est très coûteux de changer quoi que ce soit à l'intérieur du réseau, puisque ceci implique de modifier tous les routeurs de l'Internet. D'autre part, cette tâche devient encore plus ardue quand on sait que l'Internet est très hétérogène. Des réseaux câblés, sans fil et même satellitaires sont reliés entre eux, et forcément les couches les plus basses sont différentes d'un sous-réseau à l'autre. Pour toutes ces raisons, nous estimons qu'il est préférable de garder le réseau tel quel, et tout au long de cette thèse, nous n'allons nous intéresser qu'à des cas où les fonctionnalités désirées sont à implémenter au niveau applicatif.

1.1 Description des sujets étudiés

Tout au long de cette thèse, nous nous intéresserons à résoudre des problèmes propres à certains types d'applications, et pour chacun de ces problèmes, nous adopterons une approche de bout-en-bout.

Le premier sujet que nous aborderons est lié au problème de contrôle de congestion dans les applications point-à-points. Celles qui utilisent le protocole de transmission TCP

[1] En anglais dans le texte.

(pour Transmission Control Protocol [98]) pour assurer le transport des informations d'un site à un autre ne sont pas concernées par ce problème puisque le protocole TCP s'en charge déjà [67]. Les applications utilisant le protocole UDP (pour User Datagram Protocol [97]) pour le transport des données n'ont pas ce privilège et doivent contrôler leurs débits afin d'éviter de congestionner le réseau. Il est d'ailleurs souhaitable que ce type d'applications puisse se comporter de manière "civilisée" en évitant d'accaparer toutes les ressources disponibles, en d'autres termes, il faudrait que ces applications puissent être *TCP-friendly* [4].

Pour pouvoir adapter le débit d'envoi des données sur le réseau, l'application devrait avoir une idée sur l'état actuel du chemin utilisé. Une façon de faire consiste à estimer des caractéristiques propres au lien le plus congestionné de ce dernier. Par exemple, la bande passante disponible sur ce lien permet de faire un contrôle de congestion, alors que sa bande passante permet juste de contrôler le débit de l'application. Dans ce contexte, nous proposons une méthodologie de bout-en-bout qui fournit à l'application des estimations sur l'état du nœud le plus congestionné, rendant possible l'utilisation d'un mécanisme d'adaptation. À part le problème de contrôle de congestion, notre approche peut servir à équilibrer la charge des routeurs ou bien encore à la planification du réseau, puisque la méthodologie proposée permet d'identifier les nœuds congestionnés et ceux peu utilisés.

Le deuxième sujet que nous traiterons concerne les applications multipoints et, plus précisément, l'estimation de la taille des sessions multipoints en nombre de terminaux connectés à la session. Suite à l'apparition de matériels informatiques sophistiqués permettant de bien reproduire tant l'image que le son, les applications multimédia deviennent de plus en plus populaires et le trafic généré par ce type d'applications ne cesse d'augmenter [119]. Nous pensons que, tôt ou tard, les grands événements médiatiques finiront par être diffusés sur l'Internet via des sessions multipoints. De plus en plus de chaînes de télévision voudront diffuser leurs programmes via le multipoint. Il est alors essentiel pour ces sources d'information d'évaluer la popularité du contenu émis tant pour l'aménagement des programmes futurs que pour des raisons publicitaires. Il faudrait donc concevoir une méthode pour estimer le nombre de récepteurs dans une session multipoint à un instant donné. À noter que cette information peut également être utilisée par les fournisseurs d'accès dans l'établissement des factures : au lieu de faire payer les sources sur la base de leurs débits, le fournisseur d'accès peut choisir de les faire payer sur la base de leurs nombres de récepteurs.

Pour estimer la taille d'une session multiploint, la façon la plus directe serait de déployer une fonctionnalité de comptage dans le réseau lui-même. Toutefois, cette solution n'est pas très adaptée à la situation. Premièrement, elle impliquerait que chaque routeur maintienne un compteur par session multipoint, ce qui est évidemment coûteux en ressources et supporte mal le passage à l'échelle. Deuxièmement, cette modification devrait être répercutée sur tous les routeurs assurant le multipoint, ce qui est évidemment une tâche énorme. Troisièmement, l'estimation fournie serait toujours en dessous de la réalité puisque les routeurs d'accès des réseaux locaux n'ont aucun moyen de savoir le compte exact de terminaux connectés à une session multipoint donnée, la seule information disponible étant l'existence d'au moins un récepteur dans ladite session [43]. C'est pourquoi nous avons concentré nos

efforts sur l'estimation de la taille d'une session multipoint en utilisant une approche de bout-en-bout.

Le dernier sujet que nous étudierons est propre aux applications dites à code mobile. Un agent mobile est un programme qui peut changer de lieu d'exécution quand il le décide. Partant de cette définition, a été développé un paradigme appelé "code mobile" [117] qui fait de la mobilité une partie intégrante d'une application d'où le nom "application à code mobile". Les champs d'application de ce paradigme sont nombreux ; citons notamment l'équilibrage de charge, le commerce électronique, la recherche d'information et l'analyse de données [33]. Dans ce dernier cas, il s'agit d'analyser de grandes bases de données situées sur un site lointain ; le fait de déplacer le code vers le site en question permet d'économiser de la bande passante puisque le code est typiquement de petite taille contrairement aux données.

Les problèmes intrinsèques au paradigme de code mobile concernent les communications entre agents : il faudrait que ceux-ci puissent communiquer entre eux malgré la mobilité. Dans la plupart des cas, les librairies de code mobile fournissent, en sus de la mobilité, des mécanismes de communication entre agents. Parmi les mécanismes les plus répandus, citons (*i*) un mécanisme réparti qui fait usage d'objets appelés "répéteurs", chargés de transférer tout message à son destinataire ; (*ii*) un mécanisme centralisé qui s'appuie sur un serveur de localisation pour assurer les communications. Ce serveur garde à jour une base de données relative à l'emplacement de tous les agents mobiles. La question qui se pose tout naturellement est la suivante : quel mécanisme est le plus performant? C'est à fin de répondre à cette question que nous nous sommes intéressés au problème. À noter qu'aucune étude formelle dans ce cadre n'a été entreprise auparavant. Les concepteurs de librairies à code mobile, fournissant un ou plusieurs mécanismes de communication, ne justifient pas en général leur choix du mécanisme adopté et se basent souvent sur des raisonnements intuitifs (voir à titre d'exemple [84]). Nous verrons par la suite, lors de la comparaison formelle entre les deux mécanismes étudiés, que la réponse à la question "quel mécanisme est le plus performant?" n'est pas toujours évidente.

1.2 Contributions des travaux de thèse

Les contributions de cette thèse sont multiples et variées, allant de la théorie des files d'attentes, à l'environnement à code mobile, en passant par l'ingénierie des réseaux.

Dans le premier chapitre de cette thèse, nous proposons deux modèles de files d'attente basés sur les files $M/M/1/K$ et $M/D/1/K$. Pour chacune de ces files nous trouvons des expressions pour la probabilité de perte, l'utilisation du serveur, le temps de réponse, etc.. Pour la première fois, des quantités telles que la probabilité de perte conditionnelle et la probabilité de succès conditionnelle sont calculées et utilisées pour la file $M+M/M/1/K$. Par ailleurs, c'est aussi la première fois que l'analyse de la file $M/D/1/K$ est aussi détaillée. Des explications précises sont fournies pour le calcul de la distribution stationnaire de l'occupation de la file d'attente, ce qui permet le calcul de la probabilité de perte, de

l'utilisation du serveur et du temps de réponse de la file d'attente. À partir des modèles étudiés, nous proposons plusieurs schémas permettant l'estimation des caractéristiques internes du réseau. Les performances de ces divers schémas sont alors évaluées et comparées grâce à des simulations faites avec `ns-2` [83].

Dans le second chapitre de cette thèse, nous nous intéressons à l'estimation de la taille des groupes multipoints. Dans un premier temps, nous modélisons le groupe multipoint par une file d'attente $M/M/\infty$. Dans ce cas, le nombre de récepteurs dans le groupe est représenté par le nombre de serveurs occupés dans la file d'attente $M/M/\infty$. En trafic fort et après normalisation, le nombre de récepteurs dans le groupe converge en distribution vers le processus d'Ornstein-Ühlenbeck. La dynamique de ce processus étant linéaire, nous pouvons utiliser le filtre de Kalman pour résoudre le problème d'estimation. Notre objectif étant de trouver l'estimateur optimal sous les hypothèses les plus générales possibles, nous utilisons un filtre de Wiener pour trouver l'estimateur optimal dans le cas d'une file d'attente $M/M/\infty$ soumise à un trafic quelconque. À noter que la théorie de Wiener s'applique pour le cas plus général de la file $M/G/\infty$ mais le calcul des paramètres du filtre n'est malheureusement possible que pour le cas $M/M/\infty$. Par ailleurs, nous construisons un filtre linéaire d'ordre 1, qui est optimal parmi tous les filtres linéaires d'ordre 1, dans le cas où le groupe multipoint est modélisé par une file $M/H_L/\infty$ (H_L désigne une loi hyperexponentielle d'ordre L). L'estimateur résultant de ce filtre pour $L = 2$ et ceux découlant des filtres de Kalman et de Wiener sont comparés à des traces synthétiques et des traces réelles, en vue d'évaluer leurs performances et de déterminer le meilleur estimateur parmi ceux proposés. Malgré le fait que les hypothèses de base des modèles ne sont pas vérifiées dans les traces réelles, nous observons de bonnes performances : tous les estimateurs proposés reflètent bien l'évolution dans le temps de la taille des groupes multipoints. Nous insistons sur le fait que c'est la première fois que des traces réelles (audio et vidéo) sont utilisées pour la validation d'estimateurs de la taille du groupe, puisque jusqu'à présent seules des traces synthétiques ont été utilisées.

Les travaux présentés dans le troisième chapitre sont très originaux du fait même qu'ils se trouvent au croisement de deux domaines de recherche très différents : le monde de la modélisation et celui du code mobile. Les chercheurs travaillant dans le premier domaine n'ont souvent pas connaissance des problèmes rencontrés dans le deuxième domaine. De la même façon, les chercheurs venant du monde du code mobile ne sont souvent pas à l'aise avec des techniques de modélisation telles les chaînes de Markov. Comme conséquence, il n'y a presque eu aucune étude d'évaluation de performance de mécanismes relatifs au code mobile jusqu'à présent. Nous développons des modèles markoviens pour deux implémentations de mécanismes de communications entre agents. Le mécanisme à base de répéteurs est modélisé par une chaîne de Markov en temps continu et à espace d'états infini. Pour résoudre les équations d'équilibre et trouver la distribution stationnaire des états, nous utilisons des transformées en z qui rendent fini le nombre d'équations à traiter. Quant au mécanisme centralisé, nous le modélisons par une chaîne de Markov en temps continu et à espace d'états fini et les équations de Chapman–Kolmogorov sont résolues numériquement. Dans chacun des modèles, nous trouvons une expression pour l'espérance du temps de

communication, et dans le cas du mécanisme à base de répéteurs, nous trouvons également une expression pour le nombre moyen de répéteurs. Les deux modèles proposés sont alors validés à travers des simulations et des expérimentations conduites sur un réseau local et sur un réseau régional. Nous trouvons que les résultats analytiques et expérimentaux sont assez proches ce qui montre que nous pouvons utiliser les formules analytiques donnant les temps de réponse des deux mécanismes pour comparer formellement leur performances. Comme conséquence, nous pouvons tester une large gamme de conditions ce qui n'était pas possible lors des expérimentations. Les résultats de cette comparaison formelle ont révélés qu'aucun mécanisme n'est toujours le meilleur, ce qui rend notre analyse encore plus intéressante.

Dans les sections suivantes, nous allons traiter à part chacun des problèmes évoqués dans la section *1.1*.

2 *Modèles d'inférence pour l'estimation de caractéristiques réseaux*

Pour le transport de ses informations d'un point à un autre dans le réseau, une application s'appuie soit sur le protocole TCP, soit sur le protocole UDP. Ces deux protocoles de transport font partie de la famille de protocoles appelée "TCP/IP". Alors que TCP fournit un mécanisme de contrôle de congestion, le protocole UDP transmet les paquets de l'application sans aucun contrôle, au risque de congestionner le réseau. Une autre grande différence entre ces deux protocoles est que TCP retransmet les paquets perçus comme perdus, alors qu'UDP n'offre aucune garantie sur l'intégralité des informations transmises. Dans ce contexte, les applications, utilisant UDP pour le transport des informations et souhaitant avoir un minimum de garanties, devraient pouvoir régler elles-mêmes le débit avec lequel elles injectent des paquets dans le réseau. Afin d'obtenir un taux de perte faible, ces applications devraient veiller à ce que leur débit ne dépasse pas la bande passante disponible. Il faudrait aussi qu'elles aient un moyen de connaître la taille maximale des rafales de paquets qu'elles pourraient vouloir injecter dans le réseau. Ce qu'il faut donc c'est que les applications puissent avoir des estimations sur ces paramètres.

Nous supposons par la suite qu'il n'existe qu'un seul goulot d'étranglement le long d'une connexion. Nous proposons de modéliser une connexion par une simple file d'attente, ayant (i) une vitesse de traitement égale à celle du goulot d'étranglement de la connexion (qu'on notera μ) et (ii) un tampon de taille finie égale à K. Nous proposons également que l'application sonde le réseau avec des paquets de taille exponentiellement distribuée, à des intervalles de temps exponentiellement distribués, afin de déterminer l'état du réseau. Ce flux "sonde" est donc un processus de Poisson injecté dans la file d'attente décrite avant, et nous noterons son débit par γ. Les paquets de données de l'application, ainsi que les paquets provenant de tous les autres flux traversant le goulot d'étranglement de la connexion, seront représentés par un seul flux de débit λ et appelé "flux transverse".

Pour pouvoir s'adapter aux conditions du réseau, il faudrait donc que l'application

puisse avoir une estimation de la vitesse du serveur μ et de l'intensité du trafic transverse λ. La taille du tampon K étant inconnue, il faudrait l'estimer aussi. Nous allons supposer que le flux transverse peut être modéliser par un processus de Poisson de paramètre λ. Nous savons que cette hypothèse n'est pas vérifiée en général [95], nous ne l'avons considérée que pour la facilité mathématique qu'elle amène dans le calcul[2]. Au vu des hypothèses introduites jusqu'à présent et selon la notation de Kendall–Lee [72, 78]), nous pouvons dire que la connexion est modélisée par une file d'attente de type $M{+}M/G/1/K$ (voir la figure 1.2, page 9).

La méthodologie que nous proposons pour estimer μ, λ et K se résume en quatre points :

1. enregistrer des informations de base relatives aux paquets du flux sonde (instants d'entrée dans le réseau, instants d'arrivée à la destination, indication sur les paquets perdus, etc.) ;

2. utiliser ces informations pour estimer le taux de perte des paquets sonde, le délai moyen induit par le réseau, etc. ;

3. exprimer formellement les mesures de performance estimées en étape 2 en fonction des paramètres μ, λ et K, en utilisant le modèle de la connexion ;

4. inférer les valeurs des paramètres μ, λ et K en se basant sur les expressions trouvées en étape 3 et les mesures de performance estimées en étape 2.

2.1 La file d'attente $M{+}M/M/1/K$

Dans un premier temps, nous considérons que les temps de service sont indépendants entre eux et suivent une loi exponentielle de paramètre μ. La distribution stationnaire de l'occupation de la file d'attente est bien connue [75]. Nous pouvons alors facilement exprimer la probabilité de perte P_L, l'utilisation du serveur U et l'espérance du temps de réponse de la file d'attente R en fonction des paramètres μ, λ, γ et K (voir les équations (1.3), (1.5) et (1.10) respectivement, pages 10–11). Pour exprimer les probabilités conditionnelles de perte q_L et de succès q_N, nous calculons la probabilité conditionnelle d'avoir k paquets dans la file à l'instant t sachant qu'il y en avait i à l'instant 0 et étant donné qu'il n'y a pas de flux sonde ($\gamma = 0$). Le calcul de la transformée de Laplace de cette probabilité conditionnelle est détaillé dans la proposition 1.4.1. Les probabilités conditionnelles q_L et q_N s'expriment simplement en fonction de cette transformée de Laplace, et après calcul nous obtenons les équations (1.18) et (1.21).

[2]Au moment de valider nos modèles nous avons considéré des flux non-Poissonniens pour tester la sensibilité des modèles à cette hypothèse.

2.2 La file d'attente $M+M/D/1/K$

Nous considérons à présent que les temps de service sont déterministes et valent $\sigma = 1/\mu$. Pour trouver la distribution stationnaire de l'occupation de la file d'attente, nous nous basons sur l'analyse de la file $M/G/1/K$ qu'a faite J. W. Cohen dans [39]. Un calcul intermédiaire est nécessaire : il faut trouver les coefficients de la série de Taylor d'une fonction dépendant de la transformée de Laplace–Stieltjes de la distribution des temps de service. La distribution stationnaire de l'occupation de la file d'attente s'exprime alors très simplement en fonction de ces coefficients que nous noterons α_j (voir les équations (1.28)–(1.31)). Le lecteur intéressé est vivement encouragé à consulter la section 1.5.1 pour tous les détails sur le calcul de ces coefficients. Ayant exprimé la distribution stationnaire du nombre de paquets dans la file, il est facile de trouver des expressions pour la probabilité de perte P_L, l'utilisation du serveur U et l'espérance du temps de réponse R (voir les équations (1.37), (1.38) et (1.40) respectivement, pages 21–22).

2.3 Estimation des paramètres

Pour le modèle $M+M/M/1/K$, nous avons trouvé des expressions pour cinq mesures de performance (P_L, U, R, q_L et q_N) en fonction des paramètres de la file d'attente μ, λ, γ et K. Le débit du trafic sonde γ est connu de l'application, mais pas les trois autres paramètres. Pour les estimer, il suffit d'estimer trois mesures de performance et utiliser leurs expressions pour inférer les valeurs des paramètres μ, λ et K. Comme nous avons le choix entre cinq différentes mesures de performance, nous obtiendrons au total 10 schémas différents selon les trois mesures choisies. À noter toutefois, que le schéma qui utilise les trois mesures relatives aux pertes (P_L, q_L et q_N) n'est pas valide car ces dernières ne sont pas indépendantes. Nous nous retrouvons donc avec 9 schémas d'estimation.

Pour le modèle $M+M/D/1/K$, nous avons trouvé des expressions pour trois mesures de performance (P_L, U et R) en fonction des paramètres inconnus de la file d'attente μ, λ et K. Pour estimer ces derniers nous n'avons qu'un seul choix possible : utiliser les expressions de P_L, U et R et des estimations de ces mesures de performance, et inverser le système de trois équations à trois inconnues obtenu pour trouver les estimateurs $\hat{\mu}, \hat{\lambda}$ et $\hat{K}$. La table 1.1 (page 23) présente les équations à utiliser dans ce schéma, ainsi que celles à utiliser dans les neuf schémas obtenus précédemment.

Jusqu'à présent nous n'avons pas tenu compte d'un fait important : il existe plusieurs méthodes permettant l'estimation de la vitesse du goulot d'étranglement, autrement dit, de μ. Citons notamment [21, 31, 94, 77, 46]. Dans toutes ces références, les techniques utilisées pour l'estimation de μ se basent sur la dispersion observée sur une paire ou un train de paquets sonde (ou même sur les deux types d'envoi comme dans [46]). Si nous considérons maintenant que la vitesse du goulot d'étranglement μ peut être estimée à l'aide de *pathrate* [46], PBM [94] ou bien ROPP [77], il ne nous reste plus qu'à estimer λ et K et il est suffisant

dans ce cas d'utiliser deux mesures de performances. Pour le modèle $M{+}M/M/1/K$, nous aurons 10 schémas d'estimation possible mais pour le modèle $M{+}M/D/1/K$, nous n'en aurons qu'un seul (au lieu de 3). Nous avons bien trouvé des expressions pour 3 mesures de performances (P_L, U et R) ce qui offre 3 possibilités pour le choix de deux mesures parmi ces trois mesures, mais l'expression trouvée pour l'espérance du temps de réponse R comporte plusieurs inconnues et ne peut pas être utilisée que si celles de P_L et U le sont également. Les 11 schémas d'estimation ainsi obtenus sont présentés brièvement dans la table 1.2 (page 24).

Étant donné que les expressions trouvées pour les mesures de performances ne sont pas linéaires, il n'est pas certain qu'un système de deux ou trois de ces équations ait une solution unique. Nous avons pu formellement montrer l'unicité de la solution pour certains schémas uniquement. Toutefois, dans toutes nos expérimentations, nous avons toujours obtenu une solution unique pour un système donné.

2.4 Analyse des résultats

Nous avons réalisé une cinquantaine de simulations à l'aide de `ns-2`. Dans 20 de ces simulations, nous avons veillé à ce que le scénario simulé soit le plus proche possible du modèle $M{+}M/M/1/K$. Nous pouvons ainsi observer lesquelles parmi les cinq mesures de performance obtenues pour ce modèle sont le mieux estimées. Rappelons que ces cinq mesures sont estimées grâce aux informations collectées sur les paquets sonde (instants d'arrivée et de départ du réseau, indication des paquets perdus, indication des paquets ayant subi des délais d'attente; voir les équations (1.81)–(1.85), page 37). Nous avons trouvé que l'estimateur $\hat{q}_L$ est très bruité et que $\hat{P}_L$ a une faible variance mais une erreur relative pouvant dépasser la valeur 0.2. Quant aux estimateurs $\hat{U}$, $\hat{R}$ et $\hat{q}_N$, ils ont une faible variance et une faible erreur relative. À noter toutefois, que certains estimateurs convergent plus rapidement que d'autres selon la condition du réseau simulé. Ainsi, pour le cas congestionné, la majorité des paquets subit un délai d'attente, et la première estimation valide de l'utilisation U est assez tardive puisqu'elle coïncide avec le premier paquet n'ayant pas subi de délai (il faut que $\hat{U} < 1$).

La trentaine de simulations qui ne sont pas de type $M{+}M/M/1/K$ se divise entre des simulations où le service est déterministe pour chaque flux (il y a 100 flux exogènes et un flux sonde) ; des simulations où le trafic transverse est constitué de 100 ou de 250 flux de type On/Off dans lesquels la durée des périodes d'activité et d'inactivité sont de distribution Pareto ; et des simulations où le trafic transverse est constitué de 250 ou de 1000 flux FTP sur TCP.

Nous allons d'abord présenter les résultats des schémas dans le cas où la vitesse du goulot d'étranglement μ est connue. Sur toutes les simulations réalisées, nous avons observé que le schéma utilisant la probabilité de perte P_L et l'espérance du temps de réponse R retourne les meilleures estimations de λ (intensité du trafic transverse) et K (la taille de la

file d'attente). Nous observons que la qualité des estimations s'améliore quand le nombre de flux en arrière plan augmente, que ce soit pour les flux On/Off ou FTP sur TCP. Sur les 50 simulations et après une durée de simulation de 500 secondes, l'erreur relative sur λ est inférieure à 1% (respectivement 5% et 9%) dans 26 simulations (respectivement 38 et 49 simulations), alors que l'erreur relative sur K est inférieure à 1% (respectivement 5% et 9%) dans 20 simulations (respectivement 35 et 42 simulations).

Les résultats des schémas d'estimation quand la vitesse du goulot d'étranglement μ doit également être estimée ne sont pas aussi intéressants. Quand nous utilisons une estimation d'une mesure de performance à la place de celle-ci, nous induisons une erreur sur l'estimation finale. Il est alors évident que l'utilisation d'un nombre supérieur de mesures de performances conduit à une erreur plus importante en sortie. C'est ainsi que nous observons que le meilleur schéma, qui est celui utilisant P_L, U et R, ne donne de bons résultats que dans les simulations de type $M{+}M/M/1/K$.

Finalement, nous avons conduit six simulations dans lesquelles le flux sonde traversait quatre liens en cascade (voir la figure 1.17, page 59) au lieu de n'en traverser qu'un seul (cas des 50 simulations déjà présentées). Les résultats obtenus à l'aide du schéma utilisant P_L et R sont assez satisfaisants. Comme précédemment, nous avons considéré des flux transverses de type On/Off (une simulation) et FTP sur TCP (trois simulations). L'erreur relative sur l'intensité du trafic transverse est très faible quelque soit la position de goulot d'étranglement. En ce qui concerne l'estimation de la taille de la file d'attente, nous obtenons de bons résultats sauf pour le cas où le trafic transverse est de type TCP et que le goulot d'étranglement se situe près de la destination. Nous observons dans ce cas que la qualité de l'estimation s'améliore quand le nombre de flux en arrière plan augmente. Tous les détails sur ces simulations se trouvent dans les tables 1.11 et 1.12 (page 60).

2.5 Conclusion

Nous avons présenté deux modèles d'inférence qui permettent d'estimer simultanément l'intensité du trafic transverse et la capacité du tampon au nœud le plus congestionné d'une connexion. Nous avons trouvé que la meilleure façon de faire consiste à estimer le taux de perte et le temps de réponse de la connexion, et utiliser ces estimations pour inférer les valeurs des paramètres cités ci-dessus. Cette technique a été testée sur diverses simulations et les résultats obtenus sont satisfaisants. Enfin, ce travail a en partie été publié dans [16].

3 Estimation de la taille de groupes multipoints

Le protocole IP multipoint [43, 44] a été introduit pour permettre à plusieurs terminaux de recevoir les mêmes informations sans pour autant surcharger le réseau. Les applications pouvant tirer profit de ce protocole sont nombreuses et variées, telles la visioconférence, les cours à distance, la vidéo à la demande, la retransmission des événements

sportifs et les diffusions à grand public (cours des monnaies et de la bourse, télévisions, radios). Dans ces derniers exemples, une estimation du nombre de récepteurs peut être très utile aux fournisseurs de contenu qui peuvent ainsi adapter leurs services aux préférences de leur public.

Pour estimer le nombre de membres dans un groupe multipoint, la façon la plus simple (et la plus sûre) consiste à demander à tous les membres de signaler leur présence par l'envoi d'un acquittement à la source du groupe. Toutefois, ceci aura un impact négatif tant sur le réseau que sur la source qui sera submergée par les acquittements (nous nous positionnons dans le cas de groupes multipoints très larges). Il est donc préférable (et souhaitable) que les envois d'acquittement soient probabilistes. Pour ceci, nous proposons que chaque récepteur, à la demande de la source, envoie un acquittement avec une probabilité notée p à des intervalles de temps réguliers ; soit S cet intervalle. De cette façon, la source reçoit un certain nombre d'acquittements toutes les S unités de temps. Nous montrerons par la suite comment ces mesures bruitées peuvent être filtrées pour donner une estimation de la taille du groupe qui suit son évolution aussi efficacement que possible.

Un groupe multipoint peut être simplement modélisé par une file d'attente $G/G/\infty$. Les inter-arrivées des membres du groupe sont représentées par les inter-arrivées des paquets dans la file d'attente, les temps de séjour des récepteurs dans le groupe sont représentés par les temps de service dans la file d'attente, et enfin, le nombre de récepteurs dans le groupe, autrement dit la taille du groupe, est représenté par le nombre de serveurs occupés dans la file $G/G/\infty$. Nous noterons par $N(t)$ la taille du groupe multipoint. Dans les sections suivantes, nous nous appuyons sur la théorie des filtres adaptatifs pour estimer au mieux $N(t)$.

3.1 Estimation optimale à base de filtre de Kalman

Dans un premier temps, nous supposons que les inter-arrivées des membres du groupe suivent une loi exponentielle de paramètre λ, et que les durées de vie des récepteurs dans le groupe suivent également une loi exponentielle mais de paramètre μ. Dans ce cas, la taille du groupe $N(t)$ n'est autre que le nombre de serveurs occupés dans la file d'attente $M/M/\infty$. Nous savons donc que $N(t)$ est une variable poissonnienne de paramètre $\rho := \lambda/\mu$ [75].

Sous l'hypothèse d'un trafic fort (i.e., un groupe multipoint très large), la variable $N(t)$ tend vers l'infini. Mais par contre, si nous normalisons $N(t)$ par rapport à sa trajectoire limite, le processus obtenu (voir l'équation (2.8)) converge en distribution vers un processus de diffusion (voir l'équation (2.9)) [103]. Le processus de diffusion, dit également processus d'Ornstein-Ühlenbeck, est caractérisé par une dynamique linéaire : sa valeur à un instant donné n'est autre qu'une fraction de sa valeur à l'instant précédent à laquelle s'ajoute un bruit blanc.

Considérons maintenant le nombre d'acquittements $Y(t)$ reçus à un instant donné t.

Quand la taille du groupe multipoint tend vers l'infini, cette variable $Y(t)$ va elle-même tendre vers l'infini. Pour ceci, nous normalisons également la variable $Y(t)$ autour de sa trajectoire limite. Le processus obtenu (voir l'équation (2.16)) converge en distribution vers un processus qui est une fonction linéaire du processus de diffusion déjà mentionné (voir l'équation (2.27)). Ainsi, la valeur de ce processus à un instant donné n'est autre qu'une fraction de la valeur du processus de diffusion au même instant à laquelle s'ajoute un bruit blanc.

Le filtre de Kalman est un filtre linéaire d'ordre 1 qui est optimal parmi tous les filtres mesurables [107]. L'optimalité réside dans le fait que la variance de l'erreur quadratique est minimisée. Pour pouvoir utiliser ce filtre, il est nécessaire que la dynamique de l'état du système et des mesures soit linéaire, ce que nous venons juste de montrer pour le cas normalisé. Trois équations décrivent le filtre de Kalman ; ce sont (*i*) l'équation de Riccati qui permet de calculer la variance de l'erreur quadratique (qui est minimale, rappelons-le), (*ii*) l'équation donnant le gain du filtre et (*iii*) l'équation donnant l'estimation de l'état du système qui, dans notre cas, est normalisé. Cette dernière équation met en jeu deux termes dont le premier prend en compte l'estimation précédente et le deuxième l'actualise grâce à la nouvelle mesure normalisée. Pour obtenir l'estimation de la taille du groupe multipoint, il suffit de prendre l'équation (*iii*) et dénormaliser les variables représentant le système et les mesures (nombre d'acquittements reçus à la source). L'équation finale est donnée en (2.38) (page 79).

Nous avons testé l'estimateur donné en (2.38) et noté $\hat{N}_n$ sur des traces synthétiques et réelles. Dans les deux cas, nous observons de très bonnes performances (voir à titre d'exemples les figures 2.3–2.6, pages 81–82 et 85–86). L'estimateur semble être très robuste puisque les résultats sont satisfaisants malgré le fait que les distributions des traces utilisées ne sont pas exponentielles. Nous avons effectivement utilisé des lois Pareto pour générer les traces synthétiques et avons observé que les lois des inter-arrivées et des temps de séjour dans les traces réelles sont sous-exponentielles (loi de Weibull, loi Lognormale). Nous obtenons également de bons résultats quand le groupe est de petite taille (une quarantaine de récepteurs). Rappelons que cet estimateur de la taille du groupe multipoint a été obtenu sous des hypothèses de lois exponentielles et de trafic fort. Nous savons qu'il est optimal sous ces conditions et avons observé de bonnes performances dans des cas plus généraux. Nous aimerions toutefois construire un estimateur optimal sous des hypothèses plus générales. C'est ce que nous essayons de faire dans les prochaines sections.

3.2 Estimation optimale à base de filtre de Wiener

Nous considérons à présent que les temps de séjour des récepteurs dans le groupe suivent une loi de distribution générale à l'exception des lois dites à queue lourde. Autrement dit, la somme des autocovariances du processus $N(t)$ est finie. L'autocovariance de la version stationnaire de $N(t)$ est donnée en (2.39) [41, équation (5.39)]. Dans la théorie de Wiener,

le processus du signal à estimer et celui des mesures bruitées ont une espérance nulle. Nous allons donc centrer les variables $N(t)$ et $Y(t)$ autour de leurs moyennes respectives qui sont ρ et $p\rho$. Les variables ainsi obtenues sont notées $\nu(t)$ et $y(t)$.

Par la suite, nous considérons ces variables à l'état stationnaire en temps discret. La réponse impulsionnelle du filtre optimal qui transforme y_n en $\hat{\nu}_n$ vérifie l'équation de Wiener-Hopf donnée en (2.43) [61]. Une alternative consiste à calculer la fonction de transfert du filtre optimal. Pour ceci, il faut procéder aux étapes suivantes:

1. calculer $S_y(z)$, le spectre de puissance du processus des mesures y_n ;
2. factoriser $S_y(z)$ de la façon suivant $S_y(z) = \sigma^2 G(z)G(z^{-1})$ où $G(z)$ est la partie de $S_y(z)$ ayant tous ses zéros et pôles dans le cercle unité ;
3. calculer $S_{\nu y}(z)$, le spectre de puissance croisé des processus ν_n et y_n ;
4. écrire le rapport $S_{\nu y}(z)/G(z^{-1})$ sous forme de somme de fractions et isoler les fractions ayant tous ses zéros et pôles dans le cercle unité ;
5. prendre le résultat obtenu en étape 4 et le diviser par $\sigma^2 G(z)$; ceci donne la fonction de transfert du filtre optimal.

L'utilisation du filtre de Wiener est conditionnée par l'aptitude à factoriser $S_y(z)$ comme présenté dans l'étape 2. Malheureusement, cette factorisation canonique n'est possible que pour le cas où la distribution des temps de séjour des récepteurs est exponentielle. Nous nous retrouvons donc avec un modèle $M/M/\infty$ pour le groupe multipoint, mais nous nous sommes affranchis de l'hypothèse de trafic fort considérée dans la section *3.1*. Effectuant les étapes 1 à 5 décrite ci-dessus et inversant la fonction de transfert pour retrouver la réponse impulsionnelle du filtre, nous obtenons une équation aux différences d'ordre 1 donnant l'estimateur centré $\hat{\nu}_n$ en fonction de sa valeur à l'instant précédent et du processus centré des mesures y_n. Il ne nous reste plus qu'à remplacer les processus centrés par ceux non-centrés pour obtenir l'équation donnant $\hat{N}_n$ (voir (2.47)).

L'estimateur obtenu grâce au filtre de Wiener est identique à celui obtenu dans la section *3.1* obtenu grâce au filtre de Kalman. Ce résultat s'explique par le fait que les deux approches considérées minimisent la variance de l'erreur quadratique. Le filtre de Kalman est toujours linéaire d'ordre 1, celui de Wiener est toujours linéaire, et dans le cas exponentiel, il s'est avéré qu'il est d'ordre 1 également. Il est donc tout naturel que les estimateurs fournis par ces deux méthodes de filtrage soient les mêmes pour le modèle $M/M/\infty$. À noter toutefois que l'utilisation du filtre de Kalman nécessite une hypothèse de trafic fort, ce qui n'est pas le cas dans la théorie de Wiener.

Utilisant les spectres de puissance $S_y(z)$ et $S_{\nu y}(z)$ et la fonction de transfert du filtre optimal, nous calculons la variance de l'erreur quadratique (qui est minimale). De toute évidence, l'expression trouvée est la même que celle donnée par l'équation de Riccati de la théorie de Kalman.

3.3 Estimation efficace à base de filtre linéaire d'ordre 1

Nous considérons à nouveau le modèle $M/G/\infty$, autrement dit, les temps de séjour sont généralement distribués, et comme précédemment, nous excluons les lois à queue lourde. Dans cette section, nous nous donnons une équation aux différences d'ordre 1 de la forme $\hat{\nu}_n = A\hat{\nu}_{n-1} + By_n$ et calculons les paramètres A et B qui minimisent la variance de l'erreur quadratique donnée par $\epsilon := \mathbf{E}[(\nu_n - \hat{\nu}_n)^2]$. Il suffit donc de résoudre le système d'équations donné donné par : $\partial\epsilon/\partial A = 0$ et $\partial\epsilon/\partial B = 0$. Nous montrons que ce système possède une solution unique, vérifiant $0 < A < 1$, ce qui garantit la stabilité du filtre donné ci-haut. Il est évident que si les temps de séjour sont exponentiellement distribués, alors l'estimateur obtenu sera identique à celui donné par le filtre de Wiener ou de Kalman. Nous considérons par la suite que les temps de séjour suivent une loi hyperexponentielle ayant L stages. Quand $L = 2$, une solution numérique est facilement obtenue. Nous noterons l'estimateur obtenu par $\hat{N}_n^{H_2}$ par opposition à $\hat{N}_n^E$ qui dénotera l'estimateur de la taille du groupe quand les temps de séjour des récepteurs sont exponentiellement distribués. À noter que $\hat{N}_n^E$ est optimal parmi tous les estimateurs dans le cas où les lois des inter-arrivées et des temps de séjour sont exponentielles, et que $\hat{N}_n^{H_2}$ est optimal parmi tous les estimateurs d'ordre 1 dans le cas où les inter-arrivées suivent une loi exponentielle et les temps de séjour sont hyperexponentiellement distribués.

3.4 Validation des modèles

Nous avons testé les estimateurs $\hat{N}_n^E$ et $\hat{N}_n^{H_2}$ sur des traces réelles de sessions vidéos. Pour chaque trace, les valeurs de la probabilité d'acquittement p et de l'intervalle d'acquittement S sont choisies de façon à obtenir une faible variance de l'erreur quadratique et un petit volume d'acquittements générés (se référer à la section 2.7 pour tous les détails). Sur les quatre traces considérées (voir à ce titre les figures 2.8–2.11, pages 105–106), nous observons de bonnes performances pour les deux estimateurs. Le niveau de qualité de l'estimation est presque la même si on utilise $\hat{N}_n^E$ ou $\hat{N}_n^{H_2}$ (les détails sont dans les tables 2.8 et 2.9, pages 103–104). Toutefois nous remarquons que l'estimateur $\hat{N}_n^E$ donne de meilleurs résultats quand la taille du groupe multipoint change abruptement, à l'inverse de l'estimateur $\hat{N}_n^{H_2}$ qui donne de meilleurs résultats quand il y a peu de changements dans le groupe. Mais ce qui avantage clairement l'utilisation de $\hat{N}_n^E$, c'est qu'elle nécessite la connaissance *a priori* de deux paramètres seulement : la taille moyenne du groupe et la valeur moyenne des temps de séjour des récepteurs. Quant à l'estimateur $\hat{N}_n^{H_2}$, son utilisation nécessite l'identification de quatre termes (la taille moyenne du groupe, les valeurs moyennes des temps de séjour des récepteurs dans les deux stages de la loi hyperexponentielle et la probabilité d'un des stages de cette loi) ce qui est plus difficile à assurer.

L'estimateur $\hat{N}_n^E$ a été testé sur des traces où les lois sont sous-exponentielles (voir la

section *3.1*), mais qu'en est-il de l'estimateur $\hat{N}_n^{H_2}$? Ce dernier a été testé sur des traces de sessions vidéos uniquement. Nous avons identifié les lois des inter-arrivées et des temps de séjour de ces traces comme étant soit Lognormales soit de Weibull (voir la table 2.10 page 107). Ceci indique que l'estimateur $\hat{N}_n^{H_2}$ est assez robuste puisque l'estimation donnée, dans des cas autres que celui où il a été développé (modèle $M/H_2/\infty$), est très satisfaisante.

Enfin, pour estimer la taille moyenne du groupe ρ et le temps moyen de séjour $1/\mu$ (les paramètres pré-requis pour utiliser $\hat{N}_n^E$), nous proposons trois méthodes différentes qui nécessitent l'envoi d'acquittements supplémentaires : (*i*) soit les récepteurs envoient des acquittements probabilistes au moment de leurs arrivées dans le groupe, ce qui permet d'estimer λ ; (*ii*) soit les récepteurs envoient des acquittements au moment de quitter le groupe, ce qui permet d'estimer μ ; (*iii*) soit les deux ensembles. Nous proposons au lecteur intéressé de consulter la section 2.9 pour plus de détails et un test sur une trace réelle.

3.5 Conclusion

Nous avons proposé plusieurs estimateurs de la taille du groupe multipoint en suivant successivement trois approches distinctes. Nous avons testé les estimateurs sur des traces synthétiques et réelles et avons observé de bonnes performances. Nous pensons que le meilleur de ces estimateurs est celui développé pour le modèle $M/M/\infty$ et qui nécessite la connaissance de la taille moyenne du groupe et du temps moyen de séjour. Ainsi, nous avons indiqué comment la source pourrait estimer ces deux paramètres. Ce travail a en partie été publié dans [12].

4 Analyse de deux mécanismes de communication dans un environnement à code mobile

Dans un environnement à code mobile, il ne suffit pas qu'une librairie de code fournisse les éléments nécessaires à la mobilité. Il est nécessaire de fournir également des moyens pour assurer les communications entre les divers agents d'une application. Deux mécanismes de communication sont très répandus, or jusqu'à présent, le choix d'utilisation de l'un ou de l'autre de ces mécanismes n'a pas été formellement justifié. Nous proposons de choisir le mécanisme le plus rapide, autrement dit, celui qui délivre un message à l'agent mobile le plus rapidement possible. Nous allons décrire et modéliser chacun de ces deux mécanismes dans les deux prochaines sections.

Nous considérons par la suite une application à code mobile dans laquelle un objet immobile qu'on désignera par "source" essaie de communiquer avec un objet mobile qu'on désignera indifféremment par "agent mobile" ou simplement "agent". Les communications entre objets sont avec *Rendez-vous*, autrement dit, tout objet ayant envoyé un message à un autre objet se retrouve bloqué tant que le message n'a pas atteint sa destination.

4.1 Mécanisme distribué à base de répéteurs

Le premier mécanisme considéré met en jeu des objets spéciaux appelés "répéteurs". Quand un agent mobile migre, il laisse sur l'ancien site un objet – un répéteur – chargé de lui transmettre tout message lui étant destiné. Au fur et à mesure que l'agent migre, une chaîne de répéteurs se forme entre le site initial de l'agent et le site où l'agent se trouve actuellement. La source possède une référence sur le site initial de l'agent et, à chaque fois qu'un message parvient à l'agent après avoir été transmis par un répéteur, l'agent envoie un message de mise à jour à la source. Quand ceci arrive, la chaîne de répéteurs n'est plus utilisée lors de l'envoi de futurs messages. La chaîne est dite "court-circuitée" et, de proche en proche, ses répéteurs sont éliminés car plus référencés.

Pour modéliser ce mécanisme, nous devons prendre en considération la distance séparant le message (ou la source en l'absence de celui-ci) de l'agent en nombre de sauts, l'état de l'agent (inactif/migrant) et celui de la source (inactive/communiquant). Nous obtenons ainsi un triplet qui définit l'état du système. Sous des hypothèses d'indépendance et de lois exponentielles pour les variables en jeu (durée de migration, délai inter-sites, etc.), l'état du système est représenté par une chaîne de Markov en temps continu (voir la figure 3.3, page 121). L'espace d'états de cette chaîne est infini à cause de la première composante : la distance séparant la source de l'agent peut potentiellement atteindre l'infini s'il n'y a pas de limite sur le nombre de hôtes que l'agent peut visiter.

Pour trouver la condition de stabilité de cette chaîne de Markov et exprimer la distribution stationnaire des probabilités d'état, nous commençons par écrire les équations d'équilibre des flux en entrée et en sortie de chaque état. Nous introduisons ensuite une transformée en z de la distribution stationnaire. En utilisant cette transformée et en manipulant judicieusement les équations d'équilibre, nous nous retrouvons avec un nombre fini d'équations facile à résoudre. La condition de stabilité implique que l'agent ne doit pas se déplacer plus rapidement que le message qui essaie de le joindre. Ceci est évident puisque la distance entre le message (ou la source) et l'agent croît quand ce dernier migre, et décroît quand un répéteur transmet le message au site en aval le long de la chaîne des répéteurs.

Pour quantifier la performance de ce mécanisme, nous nous intéressons à deux mesures : le temps moyen de communication, noté T_F et donné en équation (3.33) (page 128), et le nombre moyen de répéteurs. Dans ce dernier cas, nous faisons la distinction entre le nombre moyen de répéteurs entre la source (qui est immobile) et l'agent, noté N_s et donné en équation (3.37) (page 129), et le nombre de répéteurs entre le message et l'agent, noté N et donné en équation (3.50) (page 131). Si l'agent ne migre pas entre le moment d'émission d'un message et le moment où le message lui parvient, nous obtiendrons $N_s = N$. Or un agent mobile peut migrer entre ces deux moments, ce qui fait que le message aura à traverser un nombre plus important de répéteurs. C'est pour cette raison que nous avons $N > N_s$. À noter que N_s peut être utilisé pour évaluer la tolérance aux pannes du mécanisme [20].

4.2 Mécanisme centralisé à base de serveur de localisation

Le deuxième mécanisme considéré se base sur une entité centralisée pour assurer les communications. À la fin de chaque migration, l'agent envoie un message de mise à jour à un serveur de localisation. Celui-ci maintient une base de donnée ayant les positions actuelles de tous les agents mobiles. Quand une source désire communiquer avec l'agent, elle utilise la dernière position connue de celui-ci et envoie le message au site correspondant. Si l'agent a migré depuis la dernière communication, l'essai de communication échoue. La source s'adresse alors au serveur de localisation pour obtenir la position actuelle de l'agent, qui peut bien ne pas être la bonne : pendant l'envoi de la réponse par le serveur, l'agent peut avoir migré.

Le serveur de localisation n'a pas le même fonctionnement d'une librairie à une autre. Nous allons décrire celui adopté dans la librairie de code mobile *ProActive* [101]. Dans cette librairie, le tampon du serveur est partitionné afin que chaque paire source-agent ait son propre espace de mémoire. Ainsi, les interactions entre les différentes paires sont éliminées. Un ordonnanceur cyclique est utilisé et une seule requête est servie à chaque fois. Quand plusieurs requêtes sont en attente dans le même tampon, le serveur de localisation traite en priorité les messages de mise à jour de l'agent. La politique de service est non-préemptive puisque *ProActive* est une libraire Java. Quand le serveur finit de traiter une requête de la source et qu'une requête de mise à jour de l'agent est en attente, le serveur ne va pas envoyer la position de l'agent trouvée dans la base de donnée (car devenue caduque) à la source mais va plutôt remettre la requête dans le tampon pour la retraiter ultérieurement. Pour finir, les requêtes de mise à jour d'un agent remplacent toute requête de mise à jour du même agent qui serait en attente.

Pour modéliser ce mécanisme, nous nous restreignons au cas d'une seule paire source-agent et montrons en section *4.4* comment étendre notre modèle au cas plus général de multiples paires source-agent. L'état du système ainsi simplifié est entièrement défini par l'état du serveur (notamment le type de requêtes s'y trouvant), celui de l'agent (inactif/migrant) et celui de la source (inactive/communiquant avec l'agent/communiquant avec le serveur). Sous des hypothèses d'indépendance et de lois exponentielles des variables en jeu (temps de service, latence réseau, inter-migrations, etc.), le système est représenté par une chaîne de Markov en temps continu et espace d'états fini (voir la figure 3.7, page 136).

Pour trouver la distribution stationnaire des probabilités d'état, nous écrivons les équations d'équilibre pour chaque état et l'équation de normalisation (la somme des probabilités vaut 1). La résolution du système matriciel ainsi obtenu se fait numériquement. De plus, nous exprimons le temps moyen de communication, noté T_S et donné en équation (3.51) (page 137).

4.3 Validation des modèles et évaluation des performances

Les modèles développés en sections *4.1* et *4.2* considèrent que les lois des variables

régissant le fonctionnement des mécanismes sont exponentielles et indépendantes entre elles. Or, des expérimentations conduites sur un réseau local et un autre régional montrent que ces variables ont plutôt une loi de distribution de Weibull, à l'exception des temps de services qui sont relativement constants. À noter que les temps d'inactivité de la source et de l'agent dépendent de l'application uniquement et peuvent donc suivre des lois arbitraires.

Dans un premier temps, nous validons les modèles a l'aide de simulations. Nous considérons plusieurs scénarios dans lesquels toutes les variables sauf une sont exponentiellement distribuées. Dans ces simulations, nous observons une très bonne robustesse des modèles à des temps d'inactivité déterministes de l'agent, à des délais inter-sites ayant une loi de distribution de Weibull et à des temps de services déterministes. Les modèles semblent être plus sensibles aux distributions des temps d'inactivités de la source et des durées de migration. Nous obtenons ainsi une erreur relative allant jusqu'à 17% quand les temps d'inactivité de la source sont déterministes, et allant jusqu'à 16% quand les durées de migration suivent une loi de Weibull. Pour terminer, nous considérons des scénarios où toutes les variables ne sont pas exponentiellement distribuées. Les résultats sont acceptables puisqu'en moyenne l'erreur relative est de 14%. Pour plus de détails, se référer à la table 3.3 (page 140).

Dans les simulations effectuées, les hypothèses d'indépendance étaient satisfaites. Pour tester la robustesse des modèles aux cas où celles-ci ne le sont pas, nous menons des expérimentations sur un réseau local et sur un réseau régional. En effet, dans ces conditions, les durées de migration et les délais inter-sites sont particulièrement corrélés. Malgré cela, nous observons que les résultats prédits par les modèles sont assez proches des résultats expérimentaux, à l'exception des expérimentations sur un réseau régional où le taux de migration de l'agent est élevé (voir les figures 3.8 et 3.9, pages 3.8 et 3.9). Dans ces cas, nous avons rencontré des difficultés lors de l'estimation du délai moyen inter-sites (latence du réseau régional), or ce paramètre a une grande influence sur les résultats théoriques. D'autre part, nous observons que le temps de communication offert par le mécanisme centralisé est plus court que celui offert par le mécanisme réparti sur un réseau local. C'est exactement l'inverse qui se produit sur un réseau régional où le mécanisme des répéteurs est plus performant (voir à ce titre la figure 3.10, page 145).

Les modèles proposés dans les sections *4.1* et *4.2* sont valides, puisque les résultats théoriques sont proches des résultats expérimentaux. Nous pouvons donc utiliser les modèles pour prédire les performances des mécanismes dans des cas généraux. En étudiant le signe de la différence entre les temps moyens de communication offerts par les deux mécanismes, plus précisément le signe de $\Delta T = T_F - T_S$, nous pouvons prédire lequel des deux mécanismes sera le plus performant. Ainsi quand ΔT est positif, le mécanisme centralisé est plus performant, et quand ΔT est négatif, c'est le mécanisme réparti qui est meilleur. Le signe de ΔT est représenté dans la figure 3.11 (page 147) pour plusieurs valeurs des paramètres des modèles. Nous retrouvons notamment les observations faites lors les expérimentations (le mécanisme centralisé est plus performant sur un réseau local et le mécanisme distribué est plus performant sur un réseau régional) et observons plusieurs comportements imprévisibles que l'intuition seule n'aurait pu prédire. Ainsi, il apparaît que, pour un taux de migration donné, le mécanisme centralisé est préférable sauf pour des taux de communication de

valeurs moyennes (dans ce cas, $\Delta T < 0$; voir les figures 3.11(a), (b) et (d), page 147). Une étude plus approfondie montre que pour des taux de communication de valeurs moyennes, les deux mécanismes offrent des performances presque égales, avec toutefois un temps moyen de communication légèrement plus faible pour le mécanisme des répéteurs (la différence est de l'ordre des dixièmes de millisecondes).

4.4 Extension au cas de multiples paires source-agent

La question du passage à l'échelle ne se pose que pour le modèle du mécanisme centralisé. Pour modéliser le mécanisme réparti, nous avons pris en compte la distance séparant une source d'un agent et les états desdits source et agent. Ceci revient à étudier la dynamique d'une seule chaîne de répéteurs. Or il n'y a pas d'interaction entre les différentes chaînes puisque, par définition, une chaîne est exclusive a une paire source-agent. Ceci n'est pas le cas du mécanisme centralisé. La politique de service élimine les interactions entre les requêtes quand celles-ci sont en attente, mais il n'empêche qu'il n'y a qu'un seul serveur pour traiter toutes les requêtes. Ainsi, quand le serveur traite une requête relative à une certaine paire source-agent, il sera vu comme étant en vacances par les requêtes générées par d'autres paires source-agent. Une alternative consiste à considérer que le serveur est en fait plus lent à traiter les requêtes. Ainsi le temps passé à traiter les requêtes d'autres paires sera comptabilisé comme faisant partie du temps de service des requêtes de la paire considérée.

Dans cette optique, nous allons remplacer dans le modèle la vitesse réelle du serveur par une vitesse réduite tenant compte du nombre de paires source-agent. Le temps moyen de communication trouvé n'est alors qu'une approximation. Cette simplification n'est toutefois plus valable quand le délai inter-sites est important ou que le taux de communication est très élevé. Le fait est que, dans ces cas, le serveur reste bloqué souvent (grand nombre de réponses à envoyer) et pendant un long laps de temps à chaque fois (grande latence du réseau). Pour évaluer la qualité de cette approximation, nous avons conduit quatre séries de simulations (une centaine de simulations dans chaque série, correspondant à certaines valeurs des taux de communication et de migration) dans lesquelles nous avons tenu à garder des variables de loi exponentielle. Le but est d'isoler l'effet de l'approximation dans les résultats théoriques.

Nous trouvons ainsi que la qualité de l'approximation dépend des valeurs choisies pour les taux de communication et de migration. Ainsi nous observons que l'erreur relative sur le temps moyen de communication reste en dessous de 10% tant que l'utilisation du serveur reste en dessous de 0.7 pour un taux de migration égal à 1, et en dessous de 0.5 pour un taux de migration égal à 10 (pour plus de détails, se référer à la table 3.7, page 153). À noter qu'un nombre de paires source-agent donné n'induit pas la même utilisation du serveur, celle-ci dépendant également des valeurs choisies pour les taux de communication et de migration. Ainsi, quand le taux de migration vaut 1 et que le taux de communication vaut 1 aussi, 92 paires source-agent sont nécessaires pour que le serveur soit utilisé à 70%,

alors qu'une cinquantaine suffit si le taux de communication vaut 10.

4.5 Conclusion

Nous avons proposé deux chaînes de Markov pour modéliser deux mécanismes de communication entre agents dans un environnement à code mobile. Grâce à ces modèles, nous avons pu trouver des expressions pour le temps moyen de communication donné par chacun des mécanismes et pour le nombre moyen de répéteurs utilisé dans le mécanisme réparti. Nous avons validé nos modèles à l'aide de simulations et d'expérimentations sur un réseau local et un autre régional. Ayant observé une bonne concordance entre les résultats théoriques et expérimentaux, nous avons utilisé les formules théoriques pour prédire la performance des mécanismes dans des cas plus généraux. Nous avons ainsi développé un moyen sûr pour choisir le meilleur mécanisme selon l'environnement de l'application.

5 Conclusions et perspectives

Nous avons, dans ce travail, considéré la même philosophie pour résoudre certains problèmes propres à certaines applications. Cette philosophie consiste d'abord à proposer des modèles mathématiques du système étudié ; ensuite, des estimateurs ou des mesures de performance sont dégagés des modèles proposés ; et finalement, les résultats analytiques sont comparés à des résultats simulés ou expérimentaux.

Dans la section *2*, nous proposons deux modèles basés sur les files d'attente $M/M/1/K$ et $M/D/1/K$ et considérons onze schémas différents permettant l'estimation desdits paramètres. Nous testons ces schémas sur des simulations et trouvons que la meilleure façon d'estimer l'intensité du trafic transverse et la taille de la mémoire au nœud le plus congestionné repose sur l'utilisation du taux de perte et du temps moyen de réponse. En ce qui concerne l'estimation de la vitesse du goulot d'étranglement, nous avons proposé des schémas permettant l'estimation des trois paramètres simultanément, mais les résultats montrent que les schémas d'estimation ne sont pas robustes. Ainsi, il est préférable d'estimer la vitesse du goulot d'étranglement en utilisant *pathrate* [46], PBM [94] ou ROPP [77], et ensuite estimer l'intensité du trafic transverse et la taille de la mémoire au nœud le plus congestionné. Toutefois, nous n'avons pas étudié l'impact d'utiliser une estimation de la vitesse du goulot d'étranglement au lieu d'utiliser la valeur correcte comme nous l'avons fait. Toute erreur sur ce paramètre va forcément influer sur les résultats donnés en section *2*. La quantification de cette influence n'a toujours pas été faite et reste une question ouverte.

Dans la section *3*, nous modélisons le groupe multipoint par une file d'attente $M/M/\infty$ et résolvons le problème d'estimation à l'aide d'un filtre de Kalman (cas d'un trafic fort) et un filtre de Wiener (cas général). Ensuite, nous construisons un estimateur dans le cas d'un modèle $M/H_2/\infty$. Nous testons ces estimateurs sur des traces synthétiques et sur des traces réelles (audio et vidéo) et observons de bonnes performances. Toutefois, d'autres modèles,

plus réalistes, pourront être proposés. Nous pensons notamment à des modèles $M/W/\infty$ ou $M/L/\infty$ puisque les lois de distributions observées sur les traces réelles (audio et vidéo) sont soit de Weibull soit Lognormales. Une façon de procéder consiste à s'imposer une équation aux différences d'une forme donnée (d'ordre 1 par exemple) reliant l'estimateur au processus de mesure, et à calculer les coefficients de cette équation de façon à minimiser la variance de l'erreur quadratique. Une autre perspective de travail concerne le choix de la probabilité d'acquittement et de l'intervalle d'acquittement. Un critère possible consiste à limiter le volume d'acquittements générés pendant un laps de temps fixe (dans cette thèse nous avons limité le volume d'acquittements générés à chaque intervalle d'acquittement).

Dans la section *4*, nous modélisons deux mécanismes de communication entre objets à l'aide de chaînes de Markov et évaluons leurs temps de réponse. Nous validons nos modèles par des simulations et des expérimentations sur des réseaux réels (local et régional). De nos expérimentations, il est apparu que le mécanisme centralisé est le plus performant sur un réseau local alors que le mécanisme réparti est le plus performant sur un réseau régional. Les expressions des temps moyen de communication sont en fait un outil de prédiction qui permettent de sélectionner le mécanisme le plus performant étant donné le contexte d'utilisation. Le modèle du mécanisme centralisé pourrait bien être revu et amélioré pour mieux tenir compte du passage à l'échelle. D'autre part, il existe plusieurs autres mécanismes de communication qui n'ont toujours pas été modélisés ni évalués. Citons notamment un mécanisme mixte qui met en jeu un serveur de localisation et des répéteurs à durée de vie limitée. Deux variantes existent selon qui parmi les répéteurs et les agents envoient des messages de mise à jour au serveur. Il n'est effectivement pas nécessaire que répéteurs et agents mettent à jour le serveur. Concernant la conception de ces mécanismes, plusieurs questions sont sans réponse, telles "quelle est la durée de vie optimale des répéteurs?" et "au bout de combien de migrations l'agent doit-il mettre à jour le serveur?" Cette dernière question ne se pose que si les répéteurs ne font pas de mise à jour avant de devenir inactifs.