TCP et réseaux ad hoc
L'évitement de la congestion

Mathias Péron

Rapport de stage MIM2 - 2004

Résumé

A l'heure actuelle 80% du trafic dans l'Internet est basé sur le protocole de transport TCP. La question de son utilisation et de ses performances dans un environnement ad hoc va de soi. Plusieurs recherches récentes tendent à montrer que TCP se comporte mal dans ce type de réseau.

Après une introduction au protocole TCP ce rapport s'attelle à cerner les problèmes que TCP rencontre dans les réseaux ad hoc, par le biais entre autres d'un étude du médium radio. On parlera alors des solutions envisagées jusqu'à présent, dont les architectures inter-couches. Enfin, après diverses constatations, un nouvel algorithme de contrôle de la congestion pour TCP est proposé et ses résultats exposés.

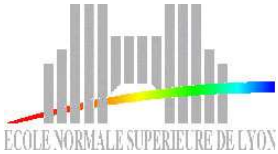


Table des matières

1 Introduction

Le début des années 1970 voit, au sein du projet militaire Américain DARPA[1], la naissance des premiers réseaux utilisant le médium radio. Ce n'est qu'avec l'arrivée du protocole 802.11 de l'IEEE[2] qui permet de bâtir des réseaux sans fil autour de bases fixes, que la recherche civile s'empare à la fin des années 90 des problématiques liées à ces réseaux.

La couche MAC[3] de la norme 802.11 porte en elle une méthode d'accès au médium totalement distribuée, c'est à dire ne nécessitant en fait aucune base pour que les éléments du réseau communiquent entre eux. En utilisant cette méthode dite DCF[4], ces réseaux, nommés *ad hoc*, ne nécessitent donc aucune infrastructure fixe, et chacun de ses nœuds peut à la fois agir comme client et comme routeur, le tout dans un environnement mobile.

De par leur facilité et rapidité d'auto-organisation, les réseaux ad hoc se trouvent des applications de plus en plus nombreuses, avec par exemple la mise en communication de véhicules ou encore la prise de mesures à l'aide de capteurs (sensors networks).

Dans ce champ d'étude récent, la recherche s'est focalisée sur la conception de protocoles de routage, dont témoigne la création du groupe de travail MANET[5] au sein de l'IETF[6] qui s'occupe de standardiser ces protocoles. Les autres couches du modèle commencent à attirer l'attention, et plusieurs recherches pointent de nombreuses défaillances. Au niveau de la couche MAC nous avions par exemple mis à jour de nombreux problèmes d'équité d'accès au médium et proposé une solution [Pér03].

Au niveau de la couche transport, le protocole TCP[7] montre un comportement inattendu par rapport à certaines de ses propriétés lorsqu'il est utilisé dans un cadre ad hoc. C'est ce problème que nous nous sommes proposés d'étudier, du fait que l'existence d'un service de transport fiable est nécessaire aux communications entre applications, et donc au futur des technologies ad hoc.

Après la section deux dédiée à un bref rappel du fonctionnement de TCP, nous essayerons dans la section suivante de discerner les problèmes que rencontre TCP en ad hoc. La section quatre rappelera au lecteur l'ensemble des solutions qui ont été jusqu'alors envisagées. Enfin la dernière section

[1] The Defense Advanced Research Projects Agency
[2] Institute of Electrical and Electronics Engineers
[3] Medium Access Control
[4] Distributed Coordination Function
[5] Mobile Ad hoc NETworking
[6] The Internet Engineering Task Force
[7] Transmission Control Protocol

présente notre algorithme dont les performances seront étudiées.

2 Vue d'ensemble de TCP

Le protocole TCP, élaboré par Vinton Cerf en 1973[8], fonctionne en commutation par paquets et détient les propriétés suivantes :

- il est *orienté à la connexion*, c'est à dire qu'une communication fiable est établie entre les deux entités communicantes. Ceci est assuré par une phase de connexion et de déconnexion (three-way handshakes).
- il est *fiable* c'est à dire qu'il assure la délivrance de la totalité des données envoyées, de façon ordonnée et exempte d'erreur.
- c'est un protocole *bout à bout* (end-to-end) c'est à dire que tout le contrôle réside uniquement chez l'émetteur et le destinataire.
- il est *indépendant* vis à vis *des données*
- il régule son débit à travers un mécanisme de *contrôle de flux*, qui prend en compte le volume de données maximum que le récepteur peut recevoir, et un *contrôle de congestion* qui s'active lors de saturation dans le réseau.
  Ceci répond au problème de limitation des ressources dans le réseau en terme d'occupation des buffers.

Les deux sous sections suivantes décrivent les mécanismes mis en place par TCP pour assurer ces différentes propriétés.

Le but n'est pas de donner une description exhaustive de TCP, mais juste de fournir au lecteur toutes les clés essentielles à la compréhension de la problématique. Pour ceux qui souhaiteraient approfondir, nous les renvoyons à un ouvrage de référence [Com96].

2.1 Fiabilité de la transmission

La mise en oeuvre de la fiabilité passe par l'utilisation de paquets d'acquittement (ACK) et de numéros de séquences (conceptuellement le numéro du premier octet contenu dans un paquet). Lorsqu'un nœud reçoit un paquet, il émet un acquittement en renseignant le numéro de séquence jusqu'auquel il a reçu *tous* les octets. On parle d'acquittement cumulatif.

Il est alors possible que l'on obtienne un second acquittement pour un même numéro de séquence (acquittement dupliqué), qui signifie qu'un paquet précédent a été détruit dans le réseau ou qu'il y est retardé.

Cependant TCP décidera de réémettre le paquet en question qu'à l'expiration (TimeOut) d'un délai avant retransmission, le RTO[9], propre à chaque paquet qui court au moment de l'émission de celui-ci.

[8] La spécification de TCP se trouve dans le RFC 793 (Request For Comments) publié en 1981

[9] Retransmission TimeOut

Le RTO, est calculé à partir du temps de transmission aller-retour, le RTT[10]. Ce dernier est une estimation du temps qui s'écoule entre l'émission d'un paquet et l'arrivée de *son* acquittement.

Les retransmissions du côté de l'émetteur comme le réordonnancement du côté du récepteur imposent naturellement la présence de buffers des deux côtés de la connexion.

La figure 1 propose un résumé de ces mécanismes en illustrant l'échange de quelques paquets.

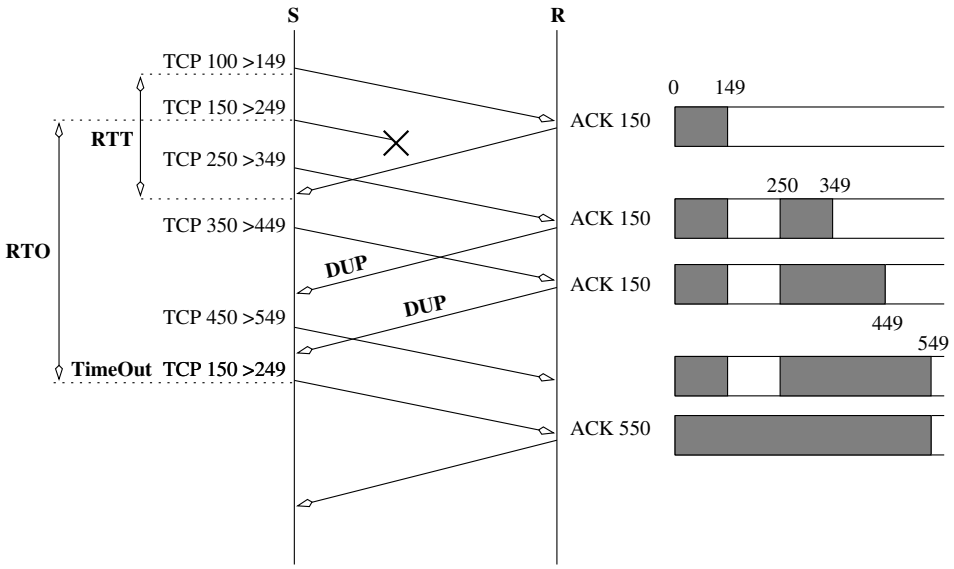


FIG. 1 – TCP : échange de paquets illustrant les mécanismes de fiabilité

2.2 Régulation du débit

TCP régule ses envois à travers un système de *fenêtre glissante* (sliding windows) qui définit la quantité de paquets pouvant être envoyés sans être acquittés, en terme de numéro de séquence. Cette fenêtre, dite *fenêtre de transmission*, que l'on abrègera en TWND[11], est calculée à partir de deux autres fenêtres.

– la *fenêtre permise*, notée AWND[12]. Elle permet au récepteur d'annoncer le nombre de segments qu'il est capable actuellement de recevoir – qui dépend, outre l'occupation du buffer, de la vitesse à laquelle l'application traite les paquets reçus. L'envoi de cette valeur dans chaque paquet constitue le contrôle de flux.

[10]Round-Trip Time
[11]Transmission WiNDow
[12]Allowed sender WiNDow

– La *fenêtre de congestion*, notée CWND[13]. Celle-ci est maintenue par le contrôle de la congestion qui s'effectue au niveau de l'émetteur. Son comportement est détaillé dans la sous-section suivante.

La fenêtre de transmission est alors régie par la formule :

$$min(\text{CWND},\ \text{AWND})$$

La figure 2 illustre son fonctionnement au moment où dans la figure précédente le dernier acquittement n'est pas encore parvenu.

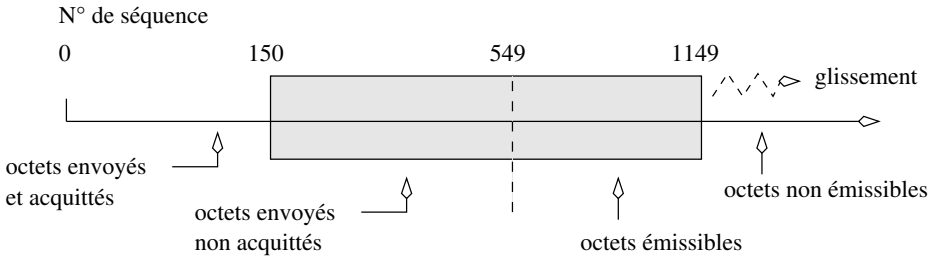


FIG. 2 – TCP : fenêtre de transmission de 1000 octets

2.2.1 Contrôle de la congestion

Dans les réseaux filaires, la congestion prend racine dans la destruction ou la rétention des paquets au niveau des buffers (seulement 1% des pertes de paquets dans l'Internet est dû à l'altération du contenu).

La fenêtre de congestion doit donc être en mesure de s'adapter à l'état courant des buffers, et ce sans surestimer la congestion au risque d'influer négativement sur le débit de la connexion.

Pour atteindre ce but, TCP va au début d'une connexion envoyer un paquet, puis, de plus en plus en augmentant CWND de façon exponentielle jusqu'à recevoir un signe de saturation dans le réseau. C'est la phase de *départ lent*, notée SS[14]. En réponse à l'apparition de congestion, ce schéma est répété, mais cette fois s'arrête à certaine valeur (SSThreshold) de CWND, inférieure au dernier pic qu'elle a atteint, pour laisser place à une seconde phase dite d'*évitement de la congestion*, notée CA[15], où le paramètre CWND va maintenant augmenter de façon linéaire.

Un graphique schématique du comportement de CWND est donné en figure 3.

[13]Congestion WiNDow
[14]Slow Start (aux initiales pas forcément heureuses)
[15]Congestion Avoidance

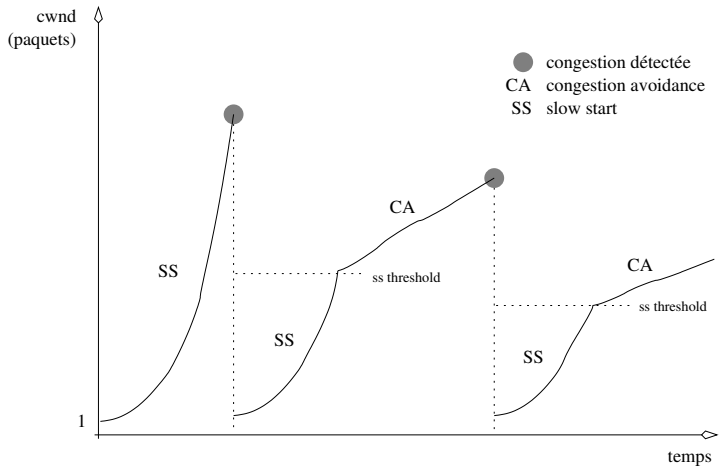


FIG. 3 – TCP : comportement de la fenêtre de congestion

L'idée consiste donc à jauger au plus près la congestion, et à diminuer drastiquement le débit dès son apparition. Les signes que TCP utilise pour la détecter sont les acquittements dupliqués et les timeouts (Algorithme 1). Ceci implique que les algorithmes d'estimation du RTT et du RTO peuvent jouer un rôle néfaste sur le débit, s'ils ne sont pas conçus avec soin.

```
cwnd := 1MSS;
/* Maximum Segment Size : taille max. du paquet en bytes */ ss_threshold
= 2^16;
max_dup_ack = 3;
état = SS;
while est_connecté do
    if reception_ack then
        switch état do
            case SS
                cwnd += MSS;
            case CA
                cwnd += (MSS^2 / cwnd);
        if cwnd > ss_threshold then
            état = CA;
    if timeout || dup_ack then
        if timeout || #dup_ack consécutifs == max_dup_ack then
            cwnd = 1MSS;
            état = SS;
        ss_threshold = max(1/2 TWND, 2MSS);
```

Algorithme 1 : Gestion de la fenêtre de congestion de TCP

Cette section nous aura rappelé quelles sont les propriétés de TCP et comment il les assure, spécialement celle du contrôle de la congestion. Il est à noter qu'il existe une foule de variantes apportées à TCP – calcul du RTT, calcul de l'accroissement en phase CA – mais qui respectent le schéma standard présenté ici.

Nous verrons quelques modifications importantes de TCP à la section quatre. A présent nous allons nous replacer dans le contexte ad hoc.

3 TCP dans le cadre ad hoc

Placer TCP dans un environnement ad hoc suppose de choisir au préalable les protocoles que l'on va utiliser au niveau des couches inférieures – au sens du modèle TCP/IP – qui sont la couche réseau et la couche de liaison (MAC).

Si le choix de 802.11[16] est naturel pour la couche MAC, il existe de nombreux protocoles de routage à notre disposition, que l'on peut classer dans deux familles différentes. D'un côté les *réactifs*, qui innondent le réseau de messages pour découvrir à la demande le chemin entre deux nœuds, et de l'autre les *proactifs* qui par l'échange régulier de messages construisent des tables de routage (directement inspiré du protocole RIP[17] d'IP). L'IETF a standardisé, respectivement dans chacune de ces familles, les protocoles AODV[18] et DSDV[19].

Nous avons opté pour le protocole AODV, car la majeure partie de nos simulations se sont faites dans des réseaux peu denses dans lesquels il génère moins de paquets de signalisation qu'un protocle proactif.

Toutes les simulations de ce rapport ont été obtenues à partir de NS2[20], un simulateur au fonctionnement validé et reconnu par la communauté scientifique. Projet *open source*, il présente l'avantage de permettre des modifications aisées à tous les niveaux.

Des cinq propriétés de TCP que nous avions listées (page 4), on remarque que seul l'évitement de la congestion est dépendant du réseau. On acceptera aisément qu'en ad hoc les autres propriétés sont préservées.

Si l'on souhaite montrer que TCP est défaillant quant à la régulation de son débit, on note de prime abord la difficulté à évaluer ce qu'est une bonne régulation du débit par rapport à la congestion du réseau.

Cela suppose d'abord d'avoir une *définition* de la congestion, et de savoir si celle-ci est identique en ad hoc et en filaire. Il s'agit ensuite d'avoir, selon

[16] Sous-entendu 802.11b, opérant à un meilleur débit théorique de 11 Mbps
[17] Routing Information Protocol
[18] Ad hoc On Demand Distance Vector
[19] Destination Sequence Distance Vector
[20] Network Simulator 2. La page internet du projet : `http://www.isi.edu/nsnam/ns/`

celle-ci, des indicateurs de l'efficacité de l'évitement de la congestion. On pense au taux d'occupation des buffers des routeurs, au nombre de paquets détruits et pourquoi, ou encore au débit global que l'on atteint pendant une connexion TCP.

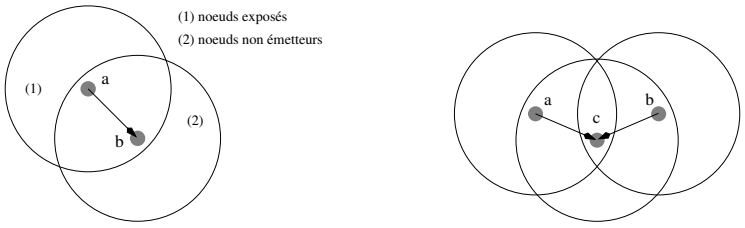
Afin de pouvoir répondre à ces questions il nous faut au préalable comprendre les caractéristiques des réseaux ad hoc. C'est ce que propose la section suivante.

3.1 Particularités d'un environnement ad hoc

> Note : une description exhaustive de l'ensemble de la couche MAC est donnée dans le rapport de stage [Pér03] à la section intitulée "Le protocole de l'IEEE 802.11". Nous invitons le lecteur à s'y reporter pour de plus amples explications concernant l'exposé de faits de cette section.

Le médium radio, à contrario du filaire, n'est pas en mesure de détecter les collisions. Il doit donc les éviter et utilise dans ce but un temps aléatoire avant d'accéder au médium, pour réduire la probabilité de collision avec un voisin et des paquets d'acquittement pour s'assurer de la bonne transmission d'un paquet.

La diffusion radio, par exemple entre les noeuds a et b de la figure 4(a), impose un comportement aux nœuds environnants. Les nœuds se trouvant dans la zone (1) ne peuvent être récepteur et ceux se trouvant dans la zone (2) ne peuvent émettre sans quoi une collision aurait lieu au nœud b. Remarquons que ces deux zones s'inversent lors de l'échange de l'acquittement.



(a) la diffusion radio

(b) le problème des noeuds cachés

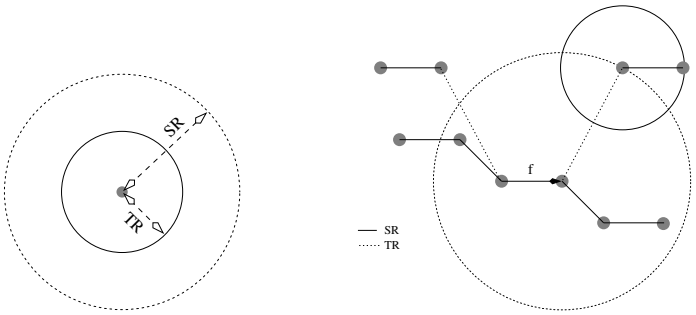
FIG. 4 – Conséquences du médium radio

La figure 4(b) représente le scénario classique où un noeud ne respecte pas la contrainte (2). Les nœuds a et b, sont trop éloignés l'un de l'autre

pour comprendre la vrai raison des collisions en c et de fait diminuent drastiquement leur débit à tort afin d'éviter cette situation.

Un mécanisme de réservation du canal a donc été mis en place : le nœud a envoie une demande d'émission, le RTS[21], auquel c répond par l'affirmative en envoyant un CTS[22] qui va imposer le silence au nœud b durant l'émission du paquet et de son acquittement.

Il reste à prendre en compte que la diffusion radio d'un paquet engendre des interférences bien au delà de la zone où le paquet est en mesure d'être décodé. Ainsi on distingue la zone de transmission, notée TR[23] et la zone de détection de signal, notée CS[24]. La détection d'un signal dans cette dernière zone impose au nœud de respecter la contrainte (1). Comme le suggère la figure 5(a), on estime que la fin de la zone CS est deux fois plus éloignée que celle de la zone TR.



(a) Atténuation du signal

(b) ensemble des flux en compétition d'accès avec le flux f

FIG. 5 – L'espace de compétition d'accès dans un environnement ad hoc

Enfin, si on considère un flux f, de par les particularités qui précèdent on peut déterminer l'ensemble des flux, et donc des nœuds, qui sont en compétition d'accès au médium avec f. Ces flux sont représentés de façon schématique à la figure 5(b).

Ces particularités donnent aux réseaux ad hoc les propriétés désavantageuses suivantes :

– un coût important d'accès au médium, dû à l'échange des RTS/CTS et à un délai aléatoire avant de pouvoir émettre.

[21] Request To Send
[22] Clear To Send
[23] Transmission Range
[24] Carrier Sense

– un espace de compétition d'accès au médium extrêmement étendu.
C'est dans ce cadre que TCP va devoir réguler son débit.

Théorie de l'information (parenthèse) Dans la description de l'environnement ad hoc que l'on vient de fournir, on a tendance à oublier qu'elle repose sur une des visions possibles de mode d'opération. Par exemple, toute interférence est traitée comme du bruit, ou encore tout paquet est entièrement décodé avant d'être retransmis.

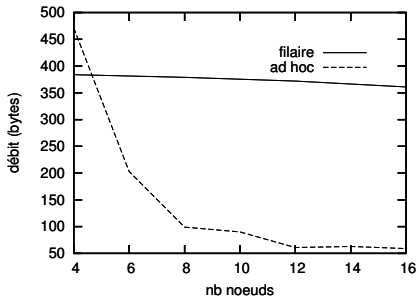
Or, comme le souligne [XK02], les interférences peuvent être de l'information ! Dans le cas où un nœud reçoit la superposition d'un signal fort et d'un signal très faible, il peut être en mesure de décoder le premier et par soustraction obtenir le second.

Répondre à la question de la quantité d'informations qui peut transiter dans un réseau ad hoc, ou encore du meilleur modèle à appliquer, sont des interrogations qui intéresseront les lecteurs qui, en fin de lecture, auront un avis pessimiste sur la résolution des problèmes en ad hoc.

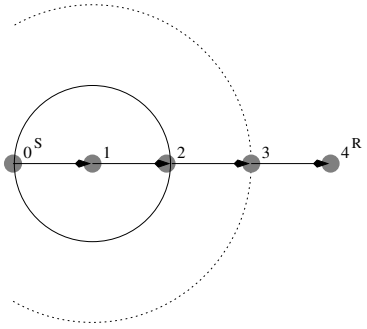
3.2 Questionnement

Notre point de départ a été de regarder comment TCP se comportait dans des scénarios simples que sont les chaînes de nœuds où l'émetteur et le récepteur se trouvent en bout (figure 6(b)).

Les résultats obtenus, visibles à la figure 6(a), montrent que le débit de la connexion diminue exponentiellement avec la longueur de la chaîne pour finalement se stabiliser à un débit extrêmement faible de l'ordre de 70 KBps.



(a) débit par rapport à la longueur de la chaîne



(b) les chaînes de nos simulations. Exemple d'une 5-chaîne

FIG. 6 – Comportement comparé dans les chaînes de TCP filaire et ad hoc

Nous avons mis en regard le comportement de TCP dans des réseaux filaires équivalents qui sans surprise est linéaire (on constate une très légère baisse due au temps de recherche de la route et surtout, de par l'allongement du RTT, un accroissement légèrement plus faible de la fenêtre de congestion).

Dans les deux environnements, et quelle que soit la longueur de la chaîne, aucun buffer n'est mis à mal. Mais si aucun paquet n'est donc perdu en filaire ce n'est pas le cas de l'environnement ad hoc ou en moyenne 5% des paquets n'arrivent pas à destination. Cependant ceci ne représente pas de telles pertes au point d'expliquer une baisse exponentielle du débit.

Une question qui surgit alors est celle des performances d'un protocole de transport qui n'assure pas de régulation de son débit. C'est le cas du protocole UDP[25], qui n'assure pas non plus de propriété de fiabilité, mais qui, à l'instar de TCP, fonctionne en commutation par paquets et avec une sémantique bout-à-bout. Nous avons alors comparé les performances de ces deux protocoles dans une 8-chaîne. Les résultats sont donnés à la table 1.

	UDP	TCP
débit (KBps)	173	99
paquets perdus (%)	60.74	5.16
débordement de buffers	62	0

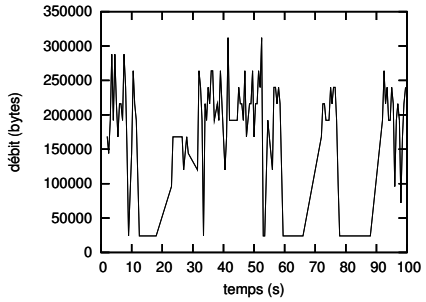
TAB. 1 – Différents indicateurs de TCP et UDP dans la 8-chaîne

UDP, en surchargeant le réseau perd beaucoup de paquets et cause des débordements au niveau des buffers, mais cependant atteint 70% de débit supplémentaire par rapport à la connexion TCP[26]. TCP devrait être en mesure de s'approcher au moins d'un débit équivalent.

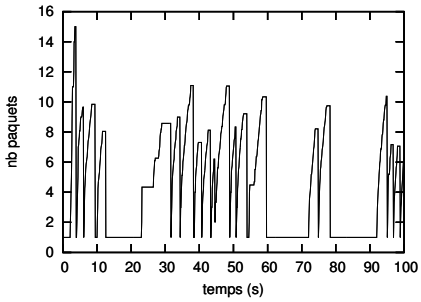
Le mécanisme de régulation de débit est le premier suspect. La figure 7, qui montre l'évolution du débit et de la fenêtre de congestion de TCP dans ce scénario, nous dit qu'il s'agit du coupable.

[25] User Datagram Protocol

[26] Ce qui ne signifie pas 70% de "données" supplémentaires, par exemple dans le cas d'un transfert de fichiers, puisque UDP n'est pas fiable



(a) évolution du débit



(b) évolution de CWND

FIG. 7 – Performances de TCP dans la 8-chaîne

Le débit souffre de grandes pauses au nombre de trois, d'une dizaine de secondes chacune, directement corrélées à une stagnation de CWND à 1 paquet. On remarque de plus que hors de ces phases TCP arrive à atteindre un débit moyen aux alentours de 200 KBps.

Le blocage de la fenêtre de congestion en phase de départ lent a diverses origines. Dans le cas présent le premier est dû à trois timeouts successifs, les deux suivants sont dus à une absence de route au moment du recalcul de celle-ci par le protocole de routage.

L'algorithme d'évitement de la congestion de TCP a été conçu pour répondre aux problèmes des goulots d'étranglement dans les réseaux filaires. La nature du médium radio, avec son espace de compétition d'accès large, fait que cette notion devient diffuse faisant de la perte d'un paquet ou de son retardement dans le réseau une conséquence de nombreux faits.

Nous nous sommes proposés de lister dans la section suivante l'ensemble des facteurs qui engendrent des pertes lors d'une connexion TCP dans les réseaux ad hoc actuels.

Au final, nous avons vu que les réseaux ad hoc montrent un mauvais comportement quant à un simple flux. Si celui-ci paraît être inhérent au médium radio, il semble cependant possible de faire en sorte que TCP évite le blocage de ses envois en prenant en compte ce qu'est réellement la congestion en ad hoc. On obtiendrait alors un meilleur débit qu'avec UDP, rendant TCP viable pour les réseaux ad hoc.

3.3 Facteurs de pertes

Une foule d'études propose des évaluations de ces facteurs, comme par exemple [DB01] qui s'intéresse aux performances de TCP en présence des trois protocoles de routage standards. Leur défaut réside dans le fait que tant

les études basées sur les simulations que les approches théoriques se doivent de fixer de nombreux paramètres qui souvent constituent les principales influences sur le résultat qu'elles nous donnent.

Ainsi nous ne nous sommes pas attachés à quantifier les pertes générées mais plutôt à répertorier l'ensemble des mécanismes qui pour nous constituent *l'ensemble de congestion ad hoc*, qui rassemble tant les pertes que la création d'indices de pertes (timeouts, etc).

3.3.1 La mobilité

Le caractère mobile du réseau implique que deux noeuds puissent à tout moment être déconnectés, créant ainsi de nombreuses pertes de paquets, notamment d'acquittements, qui pour TCP vont engendrer de nombreux timeouts consécutifs.

Le problème réside dans la transition qui doit s'effectuer lorsqu'un mobile change de position. Ce problème, connu sous le nom de *handoff* intervient à plusieurs niveaux réseau et concerne plusieurs mobiles. Le nœud en mouvement devra peut-être changer d'adresse IP s'il atteint un nouveau réseau, alors qu'un autre nœud qui se servait de celui-ci pour router des données devra invoquer une recherche de route.

Ainsi on souhaite minimiser la perte de paquets, ce que tente de faire les *smooth handoff* ou encore réduire l'augmentation du délai d'arrivée des paquets, rôle proposé par les *fast handoff*.

3.3.2 Le protocole de routage

Recherche de route La construction des tables de routage implique l'envoi de nombreux paquets de signalisation. Dans le cas des protocoles réactifs, durant une période assez longue le médium va leur être quasiment dédié.

Le fort coût d'accès au médium en temps et en espace fait que cette période peut-être suffisament longue pour produire des timeouts sur certaines connexions.

Cependant le problème le plus fréquent, se situe au niveau de l'un des nœuds, dans la perte d'une route pendant la phase de recherche distribuée, ce qui peut provoquer des destructions de paquets conséquentes selon la façon dont le protocole réagit.

Multi-route Certains protocoles de routage maintiennent plusieurs routes pour une même destination dans le but de diminuer la fréquence de recherche de routes ou encore d'améliorer la connectivité dans un environnement très mobile.

Les différentes routes que vont emprunter les paquets ne sont pas équivalentes en terme de délais, ni de compétion d'accès subit, ce qui va résulter au niveau du récepteur de la connexion en une arrivée plus désordonnée des paquets. La

conséquence de ces paquets dits *out-of-sequence* est bien sûr la génération de nombreux acquittements dupliqués qui sont des indices de congestion, mais également un point de difficulté pour le calcul du RTO.

Zones grises Sous 802.11b les paquets diffusés sont envoyés avec une vitesse de transmission de 1/2 Mbps alors que les paquets envoyés en *unicast* le sont à 11 Mbps.

Les protocoles de routage utilisent typiquement la diffusion pour envoyer leurs paquets de signalisation lors de recherche de routes. Par contre l'envoi des paquets de données se fait en unicast. La portée du signal décroissant avec l'augmentation de la vitesse, le protocole de routage aura obtenu des routes dont certaines, dans la zone grise, ne seront pas accessibles lors de l'envoi des données, créant là encore une source de pertes.

3.3.3 La couche d'accès au médium

Altération Le médium radio est connu pour avoir un fort taux d'erreur par bit, BER[27], c'est à dire de paquets au contenu altéré, dû aux interférences, ce qui constitue encore des destructions de paquets.

Retransmissions La couche MAC effectue des retransmissions des paquets, dans la limite de quatre, lorsqu'elle n'a pas reçu d'acquittement. Ce mécanisme qui a pour but d'améliorer la fiabilité rentre en conflit avec celui de retransmission de TCP.

Tout d'abord il influe considérablement sur le calcul du RTT qui est crucial quant au problème de la congestion. Enfin, il n'est pas rare que TCP et la couche MAC effectuent une retransmission d'un même paquet. Ceci va avoir pour conséquence la génération d'acquittement dupliqués. D'après [DCY93] en deça d'un certain taux d'erreur la plupart des paquets sont retransmis par les deux couches causant alors une perte de l'ordre de 30% du débit par rapport au scénario où TCP est seul garant de la fiabilité.

3.3.4 Probabilité de destruction

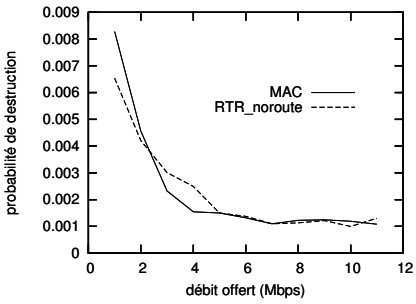
Dans l'excellent article [FZL+03], les auteurs décrivent l'évolution de la probabilité qu'un paquet soit détruit par la couche MAC, à l'aide d'une vue simplifiée du réseau qui leur permet de proposer une formule pour la calculer, classiquement à l'aide de chaînes de Markov. Leur résultat tend à montrer que cette probabilité augmente avec le débit offert, pour finir par stagner.

Pour clore et illustrer cette section, nous avons calculé la probabilité de destruction d'un paquet durant nos simulations dans la 8-chaîne en fonction

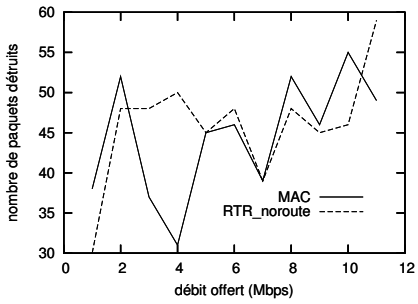
[27] Bit Error Rate

du débit offert. Le résultat, à la figure 8(a), différencie les destructions effectuées par la couche MAC à cause de la compétition d'accès, et celles par la couche réseau à cause d'un défaut de route.

On observe que le résultat est pour ce scénario à l'inverse du résultat théorique précédent, puisque la probabilité de perte décroît avec le débit offert avant de se stabiliser. Cependant on soupçonne tout simplement que le nombre de paquets détruits soit quasiment constant (au vue du nombre de paquets transitants allant de 4500 pour un débit de 1 Mbps à 45000 pour un débit de 11 Mbps) quel que soit le débit proposé, ce que nous montre la figure 8(b).



(a) Probabilité de destruction selon le débit offert



(b) Nombre de paques détruits selon le débit offert

FIG. 8 – Probabilité de destruction dans la 8-chaîne sous TCP

On conclura uniquement sur le fait que les réseaux ad hoc ont un comportement de destruction très différent des réseaux filaires (où la probabilité de destruction d'un paquet va être de 1 dès l'apparition d'un buffer plein) qui se concrétise en un certain nombre de paquets détruits indicateur de la compétition d'accès subit, et ce dès le début de la connexion.

3.4 Conclusions

L'ensemble de congestion comme nous l'avons définit, prend sa source dans toutes les couches réseaux inférieures à la couche de transport. En plus d'imposer une congestion constante, des pics de pertes sont produits lors de handoffs ou d'actions du protocole de routage.

Les dernières améliorations apportées ces dernières années à TCP, telles les implémentations Vegas, Reno et NewReno ainsi que le système SACK[28]

[28]Selective ACKnowledgment. Les intervalles des numéros de séquence des paquets reçus

n'ont pas montré de résultats significatifs dans le cadre ad hoc lors de nos simulations, tout au plus une légère augmentation du débit global, loin de celle espérée à la fin de notre questionnement.

Si l'on souhaite améliorer les performances de TCP dans un cadre ad hoc, plusieurs voies s'offrent à nous.

- Face aux nombreux timeouts et acquittements dupliqués générés, on peut souhaiter différencier leurs causes et selon, invoquer ou non l'algorithme d'évitement de la congestion. L'idée essentielle est de préserver la vision classique de la congestion et le mécanisme standard de TCP.
- On peut chercher à revoir la conception des couches MAC et réseaux, afin de minimiser les pertes. Cette voie est essentielle et si des nouveaux résultats sont nécessaires, ils ne permettront pas de s'affranchir de la nature intrinsèque du médium radio et ne résoudront pas le problème de TCP dans cet environnement.
- On peut penser que TCP place dans le réseau beaucoup trop de paquets à la fois. L'idée est alors de minimiser le nombre de paquets dans le réseau afin de diminuer les problèmes de compétition d'accès, et donc les destructions.

La première voie suppose que TCP se voie offrir des informations ou qu'il puisse accéder aux mécanismes des couches inférieures. Quelques protocoles basés sur cette approche sont présentés à la section suivante.

Nous n'explorerons pas la seconde voie. Cependant on notera au niveau MAC l'existence du mécanisme ARQ[29] qui effectue des retransmissions si un paquet a été altéré (au regard du CRC ou alors couplé au FEC[30] basé sur l'ajout de bits de parité). L'article [AB04] montre que ARQ/FEC améliorent la latence de transfert de TCP, alors que [CZ99] affirme que ce système peut être néfaste au débit de TCP, ARQ augmentant le RTT et FEC diminuant la bande passante.

Enfin, la dernière voie est celle que nous avons ouverte, à travers notre algorithme présenté à la section 5.

4 Vers des conceptions inter-couches

L'architecture actuelle des protocoles réseaux, que ce soit le modèle OSI[31] ou le modèle TCP/IP, repose sur un ensemble de modules devant chacun

sont placés par le récepteur dans les options de l'entête IP, informant ainsi exactement l'émetteur des paquets qu'il doit retransmettre.

[29] Automatic Repeat reQuest

[30] Forward Error Correction

[31] Open Systems Interconnection

fournir un service précis et ce de façon autonome. Ces modules on été organisés hiérarchiquement dans une pile, chaque module se trouvant au dessus du service dont il a besoin pour fournir le sien (figure 9(a)).

Une telle conception a l'avantage de fournir un niveau d'abstraction permettant de découper le problème en sous-problèmes, plus facile à aborder, et par là même d'améliorer la compréhension que l'on a du problème entier.

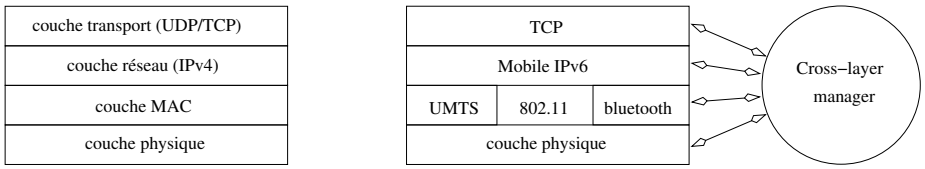
Le succès de TCP/IP réside certainement plus dans la conception de son architecture que dans les algorithmes qu'il met en oeuvre.

Cependant, dès l'apparition d'influences entre deux couches, l'envie est forte que ces couches puissent communiquer entre elles un ensemble de variables, statistiques, qui aideront à la résolution d'un problème. C'est ce que proposent les architectures inter-couches (cross-layers) notamment à travers des retours d'informations (en anglais, feedbacks).

De nombreux *feedbacks* ont été proposés pour les réseaux ad hoc, comme par exemple la couche physique informant la couche MAC de la qualité de la liaison permettant à ce dernier de décider d'utiliser ou non des mécanismes de contrôle d'erreur. Au niveau de la consommation d'énergie, très importante dans le cas des réseaux de capteurs, on souhaite par exemple que la couche MAC adapte sa portée de transmission ou encore que la couche réseau préfère des noeuds à d'autres lors du routage.

On notera tout de suite que ceci accroît la difficulté pour TCP de fournir son propre service. En effet, la considération précédente sur l'état des batteries mène par exemple directement à une sorte de handoffs.

Au final, comme le prophétise [CRR04] les réseaux sans fil devraient s'orienter vers une architecture comptant un module *cross-layer* auquel les évènements de chaque couche seraient relayés, et qui aurait pour rôle de base de données des variables du réseaux. La figure 9(b) illustre cette vision.



(a) les 4 premières couches du modèle OSI actuel

(b) future architecture proposée pour les réseaux sans fil

FIG. 9 – Architecture des protocoles réseaux

Nous allons voir à présent quelques protocoles cross-layers représentatifs

qui ont été inventés pour améliorer les performances de TCP en ad hoc. Le récent rapport de recherche [AEP04] donne un état de l'art en la matière dans lequel le lecteur trouvera une liste exhaustive de ce type de protocoles.

4.1 Contrôle des échecs de routage

L'idée consiste à imposer à un nœud qui détecte la perte d'une route l'envoi d'un paquet de *feedback*. A la réception de celui-ci l'émetteur TCP, que ce soit dans le protocole TCP-F[32] [CRVP97] ou ELFN[33] [HV99], va bloquer son fonctionnement : gel des horloges, des émissions et des fenêtres de régulation de débit. Ainsi à la fin du blocage (TCP-F attend un paquet de notification de reconstruction de la route alors que ELFN teste par l'envoi de paquets spéciaux si la route est de nouveau disponible) TCP va pouvoir reprendre immédiatement ses envois à un débit concordant avec la réalité du réseau et donc ne pas souffrir d'un départ lent de sa fenêtre de congestion.

On remarquera la proposition de [KTC01] dans TCP-BuS[34] d'imposer aux nœuds du réseau une bufferisation des paquets en cours de transit lorsqu'ils reçoivent le *feedback* de perte de route. Ainsi ces paquets ne seront pas perdus et en augmentant suffisamment le RTO de ces paquets ils ne provoqueront pas de timeouts, ce qui rend le blocage de TCP encore plus transparent en rapport à l'échec de routage.

TCP-BuS, dont l'idée de mise en cache des paquets est à souligner, a montré de bons résultats au vu des échecs de routage en atteignant 30% de débit supplémentaire en moyenne par rapport à TCP (10% par rapport à TCP-F).

4.2 Contrôle de la congestion

Venu du monde filaire, le mécanisme ECN[35] permet à un nœud routeur, selon l'occupation moyenne de son buffer, de positionner un bit dans l'entête IP[36] du paquet qu'il est en train de traiter, pour signifier qu'une congestion est probable. Le paquet, en arrivant au récepteur de la connexion TCP va positionner le bit ECN-echo dans quelques-uns des acquittements suivants. l'ECN-echo va avoir pour effet d'inhiber l'accroissement de CWND à l'arrivée de l'acquittement au niveau de l'emetteur TCP.

[32] TCP-Feedback

[33] Explicit Link Failure Notification

[34] TCP Buffering capability and Sequence information

[35] Explicit Congestion Notification, définit par le RFC 3168. L'idée fait suite au mécanisme RED (Random Early Detection) de gestion active des buffers, où tout simplement au lieu de positionner le drapeau CE, le paquet est détruit

[36] En l'occurence le bit CE (Congestion Experienced), septième drapeau TOS (Type Of Service)

D'après [CMP00][37] ECN atteint de bonnes performances en prévenant la congestion dans les réseaux ad hoc.

ATCP[38], proposé par [LS99] est l'un des protocoles *cross-layers* le mieux conçu actuellement. Il inclut une nouvelle couche entre la couche réseau et la couche TCP qui a pour but de filtrer les signes de congestion émanants du réseau.

Celle-ci garde trace des acquittements reçus et des horloges de retransmissions. Ainsi lorsque trois[39] acquittements dupliqués arrivent, ou lorsque un timeout est proche de s'effectuer, ATCP bloque TCP et effectue la retransmission lui-même des paquets nécessaires, jusqu'au moment où un acquittement non dupliqué arrive. TCP est alors débloqué et l'acquittement lui est délivré.

Enfin ATCP inclut également le mécanisme ECN et un contrôle d'échecs de routage similaire à ceux vus précédemment. L'avantage de ATCP est sa transparence tant au niveau du nœud où il opère que pour l'ensemble du réseau. Malheureusement, si des simulations ont été entreprises, elles ne sont pas de nature[40] à rendre compte de l'amélioration réelle apportée, et nous ne pouvons donc reporter ici de résultats.

Une dernière approche qui nous a paru prometteuse est celle de [KKFT02] nommée Split-TCP. Le protocole élit un ensemble de *proxys* tout au long d'une connexion TCP qui vont acquitter, par un paquet nommé LocalACK, l'arrivée du paquet au proxy précédent. Un proxy fait suivre les paquets au débit auquel il reçoit les acquittements LocalACK du proxy suivant. Afin de préserver la sémantique bout-à-bout de TCP les acquittements standards sont conservés.

Au final, la fenêtre de congestion de l'émetteur TCP est calculée à partir de l'ajout de la fenêtre bout-à-bout (ancienne CWND) et de la fenêtre locale variant selon le débit du proxy suivant, cette dernière étant implémentée également au niveau de chaque proxy.

L'équipe de recherche annonce une amélioration de 5 à 30 % du débit par rapport à TCP. Le plus grand inconvénient réside dans le trafic généré par les acquittements LocalACK et par l'élection des proxys, surtout dans un environnement mobile. Cependant on insistera sur le fait que c'est le seul protocole qui ne propose pas de bloquer TCP mais de réguler son débit selon ce qu'expérimente la connexion sur des tronçons du réseau, à savoir entre deux proxys voisins.

[37] Les auteurs ont d'ailleurs écrit en 2001 un Internet-Draft à l'IETF pour proposer la standardisation d'ECN dans les réseaux sans fil

[38] Ad Hoc TCP

[39] variable max_dup_ack de l'algorithme 1

[40] Les simulations n'incluaient pas de protocoles de routage et s'effectuaient en filaire !

Conclusion Le plus grand nombre de protocoles cherchant à améliorer TCP utilise des mécanismes *cross-layers*. Ceux-ci demandent une coopération de plus en plus importante des nœuds intermédiaires – bufferisation par exemple – et nécessitent souvent l'utilisation de paquets de *feedbacks*.

Le fait qu'une partie du mécanisme soit implémentée dans les nœuds intermédiaires implique que la totalité du réseau implémente le protocole TCP proposé, tant pour son fonctionnement que pour éviter de créer des situations inégalitaires. Or ceci va à l'encontre de la philosophie première des réseaux ad hoc où on souhaite s'affranchir d'une telle homogénéité. Enfin, dans un médium souffrant d'un haut BER, de forte compétition et de nombreuses déconnexions, l'ajout de paquets de signalisation n'est pas souhaitable.

On ne peut douter des résultats positifs sur le débit de TCP mais la dérive vers des architectures de plus en plus complexes où l'on colle des "rustines" sur chaque problème est à craindre. Si l'utilisation de feedbacks paraît inévitable sur certains points en ad hoc comme les ressources en énergie, il serait souhaitable que la recherche se mette d'accord sur l'ensemble des variables qui pourraient être communiquées sans obscurcir la compréhension à la fois générale et particulière de l'ensemble des couches réseaux.

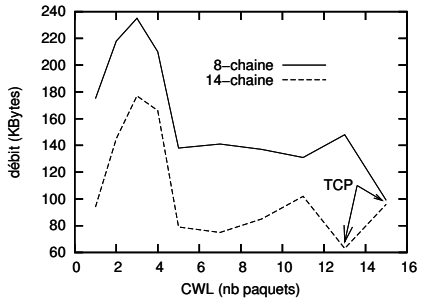
5 TCPToK : évitement de *l'ensemble de congestion*

5.1 Point de départ

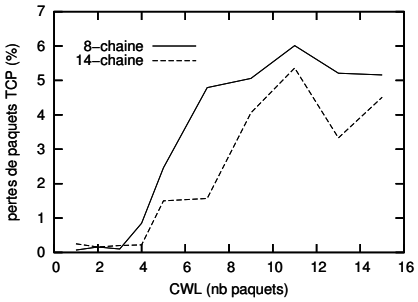
Face à l'ensemble de congestion nous avions émis l'avis que la réduction de paquets introduits dans le réseau à un même moment pourrait être bénéfique.

Dans [FZL+03], les auteurs – M. Gerla – proposent de regarder l'effet d'une borne supérieure imposée à la variable CWND lors du contrôle de congestion. Nous avons effectué nos propres simulations en ajoutant cette borne, nommée CWL[41], à l'algorithme 1. Le débit atteint et le pourcentage de perte selon la valeur de CWL dans la 8-chaîne et la 14-chaîne sont exposés à la figure 10. Ces résultats sont similaires à ceux exposés dans leur article.

[41] Congestion Window Limit



(a) débit atteint par rapport à CWL



(b) pertes subies par rapport à CWL

FIG. 10 – Effets de la borne Congestion Window Limit

On remarque immédiatement une variation extrême dans les résultats tant au niveau du débit qu'au niveau des pertes. En effet pour une valeur de CWL aux environs de 3, le débit par rapport à TCP est doublé et les pertes, quant à elles, sont devenues quasi normales – de l'ordre de 0,1 %.

Il est à noter que si on augmente fortement le débit, l'écart de performance se situe dans une fourchette de 30 à 40 % et non 100 % – on se souvient en effet que la probabilité de pertes diminue avec le débit (cf. section 3.3.4).

Ainsi limiter les envois à une très petite valeur annule une grande partie de l'ensemble de congestion. En effet la probabilité de destruction, qui est l'un de ses indicateurs, est de $2,5.10^{-4}$ pour la 8-chaîne et de $1,4.10^{-4}$ pour la 14-chaîne, lorsque CWL est égale à 3. Ce résultat est à mettre en rapport avec la probabilité de 10^{-3} que nous avions mis à jour à la figure 8(a), et nous indique donc que 10 fois moins de paquets sont perdus à cause d'un défaut de route ou de la compétition d'accès au médium.

Considérations sur CWL L'explication d'un tel résultat se trouve dans ce que l'équipe de M. Gerla appelle la *réutilisation spatiale*. Celle-ci consiste pour une connexion, dans une configuration donnée d'un réseau, en la maximisation du nombre de transmissions parallèles. Les émissions parallèles possibles sont régies exactement par l'espace de compétition d'accès au médium[42] présenté au début de ce rapport à la figure 5(b) ce qui implique,

[42]Des transmissions parallèles sont possibles dans cet espace de compétition, à condition qu'elles puissent se synchroniser afin de ne pas interférer. Cette possibilité concerne uniquement deux flux dont les émetteurs ou les récepteurs sont en opposition comme le souligne [AA03], qui propose un algorithme de synchronisation

dans le cas des chaînes qu'un seul flux peut être actif sur quatre liens radios ou encore qu'un nœud peut émettre tous les cinq nœuds [43].

Les auteurs concluent alors que la valeur de CWL optimale dans une chaîne est de $\frac{h}{4}$ où h représente le nombre de liens dans la chaîne – il s'agit donc d'une $(h+1)$-chaîne.

On remarque que le contrôle de congestion de TCP devient très simple pour les chaînes de moins de cinq nœuds. Dans [CXN03], qui se base sur ces travaux, il est proposé d'ajuster CWL à l'aide du nombre de liens aller-retour de la connexion, le RTHC[44], selon la formule $\frac{1}{5}$ RTHC. Les résultats montrent jusqu'à 15% d'amélioration en terme de débit (il est passé sous silence comment on calcule le RHTC ...). Le facteur de la formule est fixé à partir de simulations et on remarque qu'il correspond au calcul $\frac{2\,h}{5}$ dans la chaîne.

Ceci défait le raisonnement précédent sur la réutilisation spatiale. En fait, vouloir que CWL augmente avec la longeur de la chaîne est une erreur. Comme l'a expérimenté [CXN03] et comme nous le montrent nos simulations la valeur optimale de CWL se situe entre 3 et 5 et doit rester dans cet intervalle.

L'idée sous-jacente est qu'une fois que l'ensemble des nœuds participant à une connexion sont tous en possession de quelques paquets à délivrer, CWL devrait être positionnée à la valeur un, la chaîne fonctionnant alors comme un *pipeline*. Cependant ceci ne prend pas en compte le délai d'arrivée des acquittements qui conditionne les transmissions à travers le système de fenêtre glissante, ni les aléas du réseau qui font que finalement une fenêtre de transmissions de quelques paquets seulement permet de ne pas casser le comportement en flux tendu que l'on souhaite avoir dans la chaîne. Aller au delà de ces quelques paquets, c'est engendrer une augmentation rapide de la probabilité de destruction et donc des pertes comme l'illustre la figure 10(b).

5.2 La régulation du débit dans TCPToK

Fort de ces connaissances, nous avons voulu construire un algorithme qui régule CWND aux valeurs que l'on a montré optimales pour les performances de TCP. Il s'agit donc de concevoir un nouvel algorithme de régulation de la fenêtre de congestion, et ce si possible, sans recourir à des architectures cross-layers.

[43] Cette analyse dépend bien sûr de la distance au nœuds des zones TR et CS. Ici on se base sur les distances des cartes les plus courantes, comme l'équipe de M. Gerla, ce qui nous donne les chaînes de la figure 6(b)

[44] Round-Trip Hop-Count

On constate tout d'abord que maintenir CWND en dessous de 5 implique de laisser de côté le schéma exponentiel du départ lent, voire même d'abandonner l'algorithme premier de TCP. En effet on souhaiterait plutôt avoir *deux mécanismes*, l'un qui régule la fenêtre de congestion en fonction de la capacité du réseau en terme de réutilisation spatiale et l'autre qui offre une réponse à l'apparition des indices de congestion tels les timeouts, l'idée étant que le second sera d'autant plus performant si le premier accomplit parfaitement sa tâche.

Fondement Il nous est apparu qu'un indice de la réutilisation spatiale était offert à l'émetteur TCP par le premier nœud qui relaye les paquets que ce premier émet. Sous l'hypothèse d'une mobilité faible, lorsque ce nœud va faire suivre le paquet TCP, l'émetteur de la connexion TCP étant dans son voisinage va également le recevoir, puisqu'il se trouve en situation de nœud exposé.

Deux constatations. Premièrement, lorsqu'on émet un paquet TCP, il est quasiment certain que l'on devra laisser l'équivalent de deux temps d'émission à des nœuds de notre voisinage, l'un étant dû au routage du paquet de données vers sa destination – qui peut donc ne pas avoir lieu si le paquet à été pris par un nœud qui s'est très éloigné de l'émetteur TCP avant de router le paquet – et l'autre dû à l'arrivée de l'acquittement – qui lui est certain.

Deuxièmement, le fait que les nœuds de mon voisinage puissent introduire *rapidement* le paquet de données dans la suite du réseau montre que celui-ci tend à avoir une bonne réutilisation spatiale, dont l'étendue va bien au delà des quatre liens de l'espace de compétition, si l'on considère en effet que chaque nœud avait également un paquet à envoyer et qu'il a pu l'effectuer dans ce même délai.

Régulation C'est sur ces indices que notre algorithme TCPToK va se baser pour réguler son débit, à travers un simple système d'échange de jetons[45] avec son voisinage, ces jetons étant les paquets de données eux-mêmes. L'idée est que lorsque l'émetteur TCP va effectuer une transmission de paquets, il va lui en coûter un nombre équivalent de jetons. Sur ces paquets, un certain nombre va être transmis au noeud suivant, et l'émetteur TCP en les recevant également, va obtenir un jeton pour chacun d'eux.

On souhaite avoir constamment un nombre de jetons équivalent à CWND, et donc lorsque une fenêtre à été envoyée et que tous les jetons ne sont pas revenus l'algorithme décide que la réutilisation spatiale est mise à mal et que CWND doit s'adapter au nombre de jetons reçus puisqu'ils constituent une mesure de ce qui est actuellement transférable dans le réseau.

[45] En anglais *tokens*, d'où le nom de l'agorithme

Congestion Il reste à définir le comportement à tenir face aux indices de congestion. Là encore, le fait de réinitialiser la fenêtre de congestion n'a plus vraiment de sens, surtout que l'on souhaite qu'elle soit liée à l'échange de jetons. A l'instar de TCP, on souhaite diminuer de façon forte le débit. La limite CWL nous est apparu comme un excellent moyen de poser une barrière à l'évolution de CWND, à une valeur où elle génère de la congestion.

Si TCPToK va positionner CWL lorsqu'il détecte de la congestion, il est possible que la valeur de cette borne puisse augmenter pour prendre en compte un changement de comportement du réseau. Cependant afin que les effets du changement de CWL soient effectifs, une période de stabilisation est imposée durant laquelle CWL reste inchangée.

5.2.1 Algorithme complet

Les mécanismes étant exposés, on souhaite ici discuter l'algorithme TCPToK de façon exhaustive.

L'algorithme 2 a pour but de récupérer les messages qui consituent des jetons et implique donc des modifications au niveau MAC et réseau afin de porter le jeton à la connaissance de TCP.

```
/* ns-2/mac/mac-802_11.cc, recv_timer(), appellée à l'arrivée d'un
paquet */
if c'est un paquet de données ET je ne suis pas le destinataire MAC then
    marquer le paquet;
    transmettre à la couche supérieure;
    retour;
/* ns-2/aodv/aodv.cc, recv(), appellée à l'arrivée d'un paquet */
if le paquet est marqué then
    if l'adresse IP de l'émetteur est la mienne ET le nombre de
    retransmissions est de 1 then
        transmettre à la couche supérieure;
    else
        détruire le paquet;
    retour;
```

Algorithme 2 : TCPToK, niveau MAC et réseau

L'algorithme 3 constitue la première partie du code modifié au niveau de TCP. On y trouve le comptage basique des jetons et les initialisations.

La variable avg_cwnd contiendra une estimation de la valeur moyenne de CWND. stable consiste en un compteur de jetons reçus et constitue l'avancement de la période de stabilité : lorsque sa valeur dépasse seuil_stable, CWL peut être réévaluée. Enfin last_cwl est la mémoire de la valeur maximale que CWND a atteint depuis le dernier changement de CWL.

```
/* ns-2/tcp/tcp.cc, TcpAgent(), constructeur d'un agent TCP */
tokens = 1;
last_cwl = 1;
avg_cwnd = 1;
cwnd = 1;
stable = 0;
/* ns-2/tcp/tcp.cc, output(), envoi d'un paquet */
tokens–;
/* ns-2/tcp/tcp.cc, recv(), réception d'un paquet */
if le paquet est marqué then
    tokens++;
    stable++;
    détruire le paquet;
```

Algorithme 3 : TCPToK, niveau TCP, partie un

L'algorithme 4 s'occupe de la régulation de CWND et du nombre de jetons, et de fixer la valeur de CWL.

Lorsqu'un acquittement arrive, l'état Congestion Avoidance est imposé. Si l'ensemble des jetons envoyés est revenu le calcul de CWND est identique à celui de TCP. Dans le cas contraire on assigne à CWND le nombre de jetons courants. Le nombre de jetons alors à disposition de TCP est égal à la nouvelle valeur de CWND moins les jetons qui vont arriver en retard.

Ainsi dans TCPToK le fait qu'un acquittement soit délivré n'implique pas une augmentation de CWND mais sa réévaluation. Enfin, si la connexion est stable et si l'écart entre CWL et l'évaluation moyenne de CWND est faible, on considère que l'on peut être en mesure d'envoyer plus de paquets et donc CWL est augmenté.

Le calcul de CWND invoqué par les timeouts et les acquittements dupliqués n'est pas modifié. Comme précédemment le changement de CWND impose le recalcul du nombre de jetons alloués. Par contre c'est ici qu'intervient le calcul de la limite CWL. Si l'écart entre celle-ci et l'évaluation de CWND est important, CWL va être placée entre ces deux valeurs. Par contre si la différence est faible, un tel mécanisme mènerait à une stagnation de CWL, et donc celle-ci est placée légèrement en dessous de l'évaluation moyenne de CWND.

```
/* ns-2/tcp/tcp.cc, opencwnd(), augmenter CWND, appellée à
l'arrivée d'un ACK non dupliqué */
cwnd_before = cwnd;
état = CA;
/* calcul de "increment" selon le standard TCP */
if tokens == cwnd then
    cwnd += increment;
    tokens = partie_inf(cwnd);
else if tokens < partie_inf(cwnd) then
    cwnd = tokens;
    tokens = cwnd - (partie_inf(cwnd_before) - tokens);
    if tokens < 1 then
        cwnd = 1;
        tokens = 1;
else
    tokens = partie_inf(cwnd);
avg_cwnd = (0,9 × avg_cwnd) + (0,1 × cwnd);
if cwnd > last_cwl then
    last_cwl = cwnd;
if stable > seuil_stable then
    diff = 1/2 (last_cwl - avg_cwnd);
    if diff < 0,5 then
        cwl = avg_cwnd + diff;
        stable = 0;
/* ns-2/tcp/tcp.cc, slowdown(), réduire CWND, appellée au
déclanchement d'un timeout ou à l'arrivée d'ACK dupliqués */
cwnd_before = cwnd;
/* calcul du nouveau CWND selon le standard TCP */
avg_cwnd = (0,9 × avg_cwnd) + (0,1 × cwnd);
tokens = partie_inf(cwnd) - (partie_inf(cwnd_before) - tokens);
if tokens < 1 then
    cwnd = 1;
    tokens = 1;
diff = 1/2 (last_cwl - avg_cwnd);
if diff < 0,5 then
    cwl = avg_cwnd - diff;
else
    cwl = avg_cwnd + diff;
last_cwl = cwnd;
stable = 0;
```

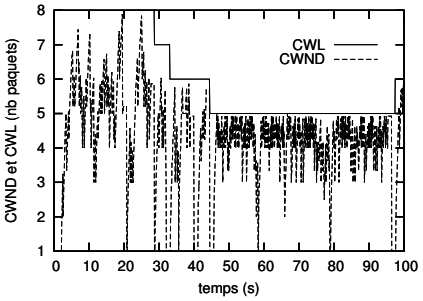
Algorithme 4 : TCPToK, niveau TCP, partie deux

5.3 Résultats

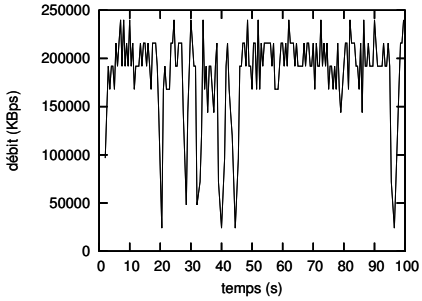
Pour illustrer le fonctionnement de TCPToK nous avons choisi la 14-chaîne. Le comportement de la fenêtre de congestion donné en figure 11(a) est celui attendu : l'échange de jetons conduit à une fluctuation de CWND aux alentours de 5 jusqu'à la seconde 20 où un timeout est déclenché. La borne CWL rentre alors en action et au fur et à mesure des timeouts (secondes 28, 33, 39, 44) se réactualise de façon coordonnée à l'évolution de CWND. On remarque que jusqu'à la seconde 45 l'échange de jetons est quasiment le seul à intervenir dans la valeur de CWND. Puis finalement CWL est bloquée à la valeur 5, et le mécanisme des jetons vient en butée sur cette borne.

L'algorithme a alors réussi à fixer la limite des émissions à laquelle il obtient un débit maximal pour un minimum de congestion. Ce résultat est très visible à la figure 11(b) qui montre l'évolution du débit dans ce scénario.

On remarquera enfin que la limite CWL est réhaussée un peu avant la centième seconde, de par un retour efficient des jetons.



(a) évolution de CWND et de CWL dans la 14-chaîne avec TCPToK



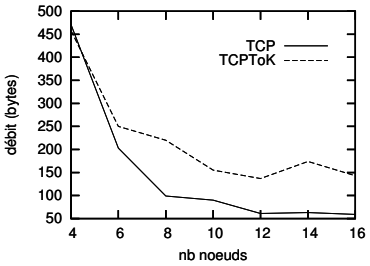
(b) débit dans la 14-chaîne avec TCPToK

Fig. 11 – Performances de TCPToK dans la 14-chaîne

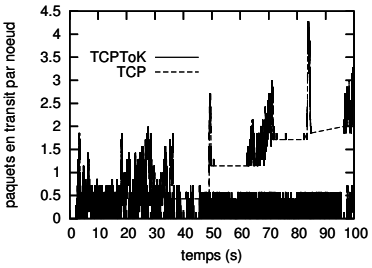
TCPToK permet un débit dans ce scénario de 174 KBps, et souffre de 0,27% de perte sur les paquets qu'il émet. Ceci est à mettre en regard, à la fois avec les performances de TCP, soit un débit de 63 KBps et un pourcentage de perte de 6,19%, et les performances de TCP avec CWL constante, qui sont de 177 KBps et de 0,20% de perte.

Ainsi on a quasiment triplé le débit de TCP, et atteint notre objectif de coller à l'optimal du schéma où l'on positionne la limite CWL. Si la différence entre TCP et TCPToK est dans ce scénario la plus forte que nous ayons, la plupart de nos simulations montre un gain minimal de 35% sur le débit pour TCPToK, ce qui constitue un très bon résultat.

Pour rendre compte du changement de comportement que TCPToK apporte, nous avons comparé les débits avec l'accroissement de la longueur de la chaîne. Ceux-ci sont mis en parallèle à ceux de TCP dans la figure 12(a). Outre la forte amélioration du débit, le résultat caché réside dans le fait que les chaînes de longueur 10, 12 et 16 pourraient atteindre un meilleur débit (au vue des simulations) de telle sorte que celui-ci soit – quasiment – indépendant de la longueur de la chaîne, à l'instar du filaire. Savoir si cela était possible, faisait partie de notre questionnement initial.



(a) débit par rapport à la longueur de la chaîne



(b) nombre de paquets en moyenne en attente dans les nœuds routeurs (exemple : 14-chaîne)

FIG. 12 – Comportement de TCPToK

Enfin, au niveau de la réutilisation spatiale, on a regardé le nombre de paquets en attente de transmission à un temps `t` au niveau de chaque nœud. La figure 12(b) trace la moyenne sur tous les nœuds de ces informations pour TCP et TCPToK. On remarque que le nombre de paquets en transit dans le réseau sous TCPToK est presque constant, ce qui n'est pas le cas de TCP où les nœuds vont se retrouver avec de plus en plus de paquets à envoyer, marque d'une compétition d'accès forte. La régulation du débit de TCPToK permet bien une avancée des paquets de façon continue dans la chaîne.

Déchaîner Une question incontournable est de savoir comment se comporte TCPToK dans d'autres configurations, comme la grille, la croix – deux flux s'entrecoupants – ou encore en présence de nœuds n'ayant pas la même capacité de débit (goulot d'étranglement). Le tableau 2 donne un récapitulatif des résultats expérimentés dans ces différents scénarios.

	débit TCP (KBps)	débit TCPToK (Kbps)
(4,4)-grille 1 flux diagonal	171	200 (+17%)
(4,4)-grille 2 flux diagonaux	97	131 (+35%)
8-chaîne 4-goulot	139	187 (+35%)
12-chaîne 8-goulot	86	131 (+52%)
(12,4)-croix au noeud 6	(12-chaîne) 63 (4-chaîne) 208	72 (+14%) 271 (+30%)
8-chaîne handoff 7s	73	154 (+111%)

TAB. 2 – TCPToK à l'épreuve de différents scénarios

On remarquera, dans le scénario de la croix, la forte inégalité au niveau du débit entre les deux flux, accentuée par TCPToK. [XBLG02] se sont intéressés au problème de l'équité au niveau TCP.

Un dernier résultat important de TCPToK est qu'il ne crée pas de gigue – variation dans le délai de transfert d'un paquet, ce qui constitue l'un des critères sine qua non pour fournir de la qualité de service (QoS) dans un réseau.

5.4 Futur travail

Comme le montre la sous-section précédente, les résultats de TCPToK sont encourageants. Cependant il faut être conscient que sa conception souffre de beaucoup de manques. Voici quelques considérations qui devront être prises en compte.

- on ne s'est pas préoccupé des acquittements. Dans certaines simulations on remarque que l'algorithme a tendance à faire fluctuer CWND à des valeurs légèrement trop grandes, or bien que les acquittements soient de petite taille, le coût fixe de prise du médium est à prendre en compte. De plus il y a de fortes chances pour que les acquittements soient en compétition avec les paquets de données.
- si l'émetteur détient plusieurs connexions cela implique qu'il puisse attribuer les jetons aux bonnes connexions. Pour ce faire, on peut se baser sur le numéro de séquence.
- le nœud voisin peut être amené à retransmettre le paquet TCP. Ceci génère un jeton superflu alors qu'il s'agit d'un indice de congestion. Pour repérer le jeton dupliqué on pourra garder trace de tous les numéros de séquence des jetons en attente. Enfin il faudra définir comment on prend en compte cet événement.
- ce nœud peut tout aussi bien être amené à détruire un paquet et donc un jeton. La question est celle de l'utilisation d'un feedback.
- Un problème difficile sera celui de la mobilité des nœuds qui risquent en s'éloignant les uns des autres de perdre des jetons. Il s'agit encore

d'un problème de handoff.
– L'un des problèmes majeurs que TCPToK peut rencontrer est de ne pas être en mesure de pouvoir recevoir les jetons, par exemple s'il est exposé, en plus d'être exposé à l'émission des jetons, à un autre flux générant alors des collisions.

6 Conclusion

Dans ce rapport nous avons essayé de caractériser le comportement de TCP dans un environnement ad hoc. Nous avons noté que sa propriété de régulation de débit était basée sur une vision de la congestion en filaire, qui est toute autre dans les réseaux ad hoc. Nous avons alors exhibé l'ensemble de congestion ad hoc qui prend source dans les couches MAC et réseau.

Il était alors naturel de se tourner vers des architectures inter-couches pour capturer le phénomène de congestion. Cependant outre les doutes sur les *effets secondaires* de ce type d'architecture, nous avons vu que le problème du comportement de TCP prenait source en lui même, à cause d'une régulation de son débit inadaptée au médium ad hoc et ses contraintes.

Nous nous sommes donc dirigés vers la conception d'un algorithme capable de s'adapter à un maximum de réseaux différents, et ce en testant sa capacité de réponse. TCPToK change le niveau de performances que l'on peut atteindre et réduit la congestion à un niveau où elle peut être appréhendée efficacement.

Au vue de l'importance démontrée d'un bon contrôle de débit et de congestion, ces mécanismes ne pourront pas être occultés, par exemple dans la conception d'un nouveau protocole de transport.

La communauté scientifique travaillant sur ces problématiques devrait être interessée par la modélisation de TCPToK, ou encore son amélioration et son évaluation et espérons que ceci la mènera sur la voie d'une adaptation réussie de TCP au monde ad hoc.

Références

[AA03] Sorav Bansal Arup Acharya, Archan Mistra, *Maca-p : A mac for concurrent transmissions in multi-hop wireless networks*, IEEE PerCom (2003).

[AB04] Raja Abdelmoumen and Chadi Barakat, *Analysis of tcp latency over wireless links supporting fec/arq-sr for error recovery*, IEEE International Conference on Communications (2004).

[AEP04] Al Hanbali Ahmad, Altman Eitan, and Nain Philippe, *A survey of tcp over mobile ad hoc networks*, Tech. report, INRIA maestro, May 2004.

[CMP00] Shiduan Cheng, Jian Ma, and Fei Peng, *A proposal to apply ecn to wireless and mobile networks*, Internet Society INET (2000).

[Com96] Douglas Comer, *Tcp/ip : architecture, protocoles, applications*, 3ème ed., Mai 1996.

[CRR04] Gustavo Carneiro, José Ruela, and Manuel Ricardo, *Cross-layer design in 4g wireless terminals*, IEEE Wireless Communications **11** (2004), no. 2.

[CRVP97] Kartik Chandran, Sudarshan Raghunathan, S. Venkatesan, and Ravi Prakash, *A feedback based scheme for improving tcp performance in ad-hoc wireless networks*, International Conference on Distributed Computing Systems (1997).

[CXN03] Kai Chen, Yuan Xue, and Klara Nahrstedt, *On setting tcp's congestion window limit in mobile ad hoc networks*, IEEE ICC (2003).

[CZ99] A. Chockalingam and M. Zorzi, *Wireless tcp performance with link layer fec/arq*, IEEE ICC (1999).

[DB01] Thomas D. Dyer and Rajendra V. Boppana, *A comparison of tcp performance over three routing protocols for mobile ad hoc networks*, MobiHoc (2001).

[DCY93] A. DeSimone, M. Chuah, and O. Yue, *Throughput performance of transport-layer protocols over wireless lans*, GLOBECOM **1** (1993).

[FZL+03] Zhenghua Fu, Petros Zerfos, Haiyun Luo, Songwu Lu, Lixia Zhang, and Mario Gerla, *The impact of multihop wireless channel on tcp throughput and loss*, IEEE INFOCOM (2003).

[HV99] Gavin Holland and Nitin Vaidya, *Analysis of tcp performance over mobile ad hoc networks*.

[KKFT02] S. Kopparty, S.V. Krishnamurthy, M. Faloutsos, and S.K. Tripathi, *Split tcp for mobile ad hoc networks*, IEEE GLOBECOM (2002).

[KTC01] Dongkyun Kim, C.-K. Toh, and Yanghee Choi, *Tcp-bus : Improving tcp performance in wireless ad hoc networks*, IEEE Communications Society Journal on Communications and Networks **3** (2001), no. 2.

[LS99] Jian Liu and Suresh Singh, *Atcp : Tcp for mobile ad hoc networks*, IEEE Journal on Selected Areas in Communications **19** (1999), no. 7.

[Pér03] Mathias Péron, *Etude de l'équité dans les réseaux ad hoc*, Master's thesis, Ecole Normale Supérieure de Lyon, 2003.

[XBLG02] Kaixin Xu, Sang Bae, Sungwook Lee, and Mario Gerla, *Tcp behavior across multihop wireless networks and the wired internet*, ACM WoWMoM (2002).

[XK02] Liang-Liang Xie and P. R. Kumar, *A network information theory for wireless communication : Scaling laws and optimal operation*, IEEE Transactions on Information Theory (2002).

Equipe d'accueil

Ce stage s'est déroulé au sein de l'équipe systèmes distribués DSG[46], l'un des groupes de recherche en informatique du Trinity College à Dublin. Le groupe, qui compte plus de cent personnes, se concentre sur quatre grands thèmes, les architectures middleware, la programmation distribuée, l'informatique omniprésente et enfin les réseaux mobiles.

C'est dans cette dernière thématique que M. Stephan Weber a encadré ce stage.

Il est à noter que le laboratoire est en train d'installer dans la ville de Dublin un réseau ad hoc fixe comportant une quinzaine de nœuds, nommé WAND[47]. Ce réseau devrait vite devenir un banc d'essai apprécié. Malheureusement, seul quatre nœuds sont pour l'instant opérationnels, dans Westland Row[48] où se trouve le laboratoire. J'ai été chargé d'effectuer un tracé de la topologie actuelle du réseau afin de pouvoir mener des comparaisons avec le simulateur ns2.

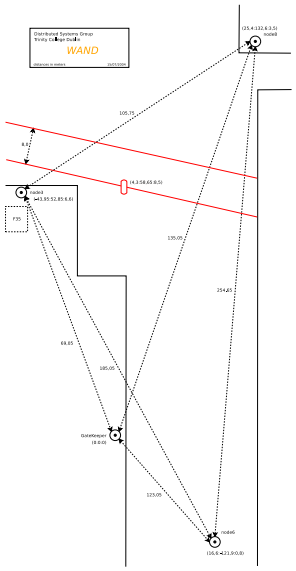


FIG. 13 – Topologie du projet WAND, côté Westland row

[46]Distributed Systems Group
[47]WAN Dublin
[48]Oscar Wilde est né au numéro 21 de cette rue en 1854

Table des figures

Liste des tableaux

Liste des Algorithmes

Ressources éléctroniques disponibles à l'adresse
`http://perso.ens-lyon.fr/mathias.peron/`