Etudes de la consommation d'énergie de TCP Tahoe, Reno, New-Reno, SACK, Vegas et WestwoodNR dans les réseaux ad hoc

Alaa Seddik Ghaleb*, Yacine Ghamri-Doudane** et Sidi-Mohammed Senouci*

*Laboratoire d'Informatique de Paris 6 (LIP6)
8 rue du Capitaine Scott, 75015 Paris

**Institut d'Informatique d'Entreprise (IIE-CNAM)
18 Allée Jean Rostand, 91025 Evry CEDEX

Résumé – **La consommation d'énergie est un facteur très important dans les réseaux ad hoc, puisque les nœuds ad hoc représentent des terminaux mobiles dont le support énergétique est souvent une batterie de capacité limitée. D'autre part, plusieurs travaux ont montré qu'une grande partie de cette consommation est due seulement aux coûts de traitement liés aux transmissions réseau. Parmi ces traitements, nous nous intéressons dans ce travail à ceux effectués par la couche transport TCP (Transport Control Protocol). TCP a été conçu principalement pour les réseaux filaires où les taux d'erreur ne sont pas trop élevés et les ruptures de liens sont très rares. Contrairement aux réseaux filaires, les mécanismes de fiabilisation introduits par TCP entraînent un nombre important de retransmissions de paquets dans les réseaux sans-fil, qui par conséquent aboutissent à plus d'énergie consommée. Plusieurs variantes de TCP (Tahoe, Reno, New-Reno, SACK, Vegas, et WestwoodNR) ont été proposées dans l'unique but de concevoir des protocoles de transport réagissant au mieux devant la perte de paquets. Dans ce papier, nous analysons l'effet de chacune de ces variantes sur la consommation d'énergie dans les réseaux ad hoc statiques. Grâce à un ensemble de simulations nous avons remarqué que TCP New-Reno est, dans la plupart des scénarios étudiés, le plus performant comparé aux autres variantes.**

Mots clés – **réseaux ad hoc, consommation d'énergie, variantes de TCP.**

1. Introduction

Les réseaux ad hoc, sont des réseaux radioélectriques qui se déploient facilement voire automatiquement entre personnes souhaitant communiquer entre elles, sans qu'il y ait besoin de développer toute une infrastructure pour ce faire. Fonctionnant un peu sur le principe du pair-à-pair (P2P), il suffit que deux terminaux soient à proximité pour qu'ils puissent communiquer entre eux si besoin est. C'est une technologie idéale pour les applications militaires ou civiles caractérisées par une absence, ou la non-fiabilité, d'une infrastructure préexistante (opérations de secours, missions d'exploration, etc.). L'activité du groupe MANET (*Mobile Ad hoc Networks*) de l'IETF montre que le développement de ces réseaux est en plein essor.

Un des grands challenges, pour ce type de réseaux, réside dans l'autonomie restreinte des stations mobiles le constituant. Effectivement, une des principales contraintes dans les communications sans fil est la durée de vie limitée des terminaux mobiles dont le support énergétique représente souvent une batterie dont la capacité est limitée. Cette contrainte devient davantage considérable pour les réseaux ad hoc, où les stations ont de surcroît la fonction de routage. De ce fait, il devient important d'économiser l'énergie dans un but de prolonger la survivabilité de tels réseaux. Cette énergie consommée dans un nœud est une combinaison de l'énergie consommée dans chacun des niveau de la pile protocolaire (physique, liaison données, réseau, transport et la couche application) [1][2]. Plus particulièrement, nous

nous intéressons, dans ce papier, à la consommation d'énergie due à la couche transport TCP (Transport Control Protocol). TCP étant la couche transport la plus utilisée dans le réseau Internet.

En dépit de sa popularité, TCP ne convient pas aux réseaux ad hoc. TCP a été conçu pour les réseaux filaires où les taux d'erreur, BER (Bit Error Rate), ne sont pas élevés et où les congestions dans le réseau sont les principales causes de pertes de paquets. Par contre, dans un réseau ad hoc, plusieurs sortes d'événements peuvent causer des pertes de paquets. En effet, un paquet peut être perdu à cause d'une congestion dans le réseau, à cause d'un taux d'erreur qui peut être élevé ou encore à cause d'un lien rompu suite à la mobilité d'un ou plusieurs nœuds. Or, TCP ne sait pas distinguer entre ces différentes causes de perte de paquets. Ces différentes causes de pertes de paquets peuvent fortement influer sur le fonctionnement de TCP (c-à-d le fonctionnement de ses différentes variantes) et par conséquent sur la consommation d'énergie qui en découle. L'objectif principal de notre travail consiste à étudier les performances de chacune des variantes de TCP et son impact sur la consommation d'énergie dans un réseau ad hoc.

Le reste du papier est organisé comme suit. Après une présentation des différentes variantes de TCP dans la section 2, nous abordons dans la section 3 les différents problèmes de performances de TCP dans les réseaux ad hoc. L'évaluation par simulation de la consommation d'énergie de chacune des variantes de TCP est présentée dans la section 4. Pour finir, la section 5 récapitule les principales conclusions auxquelles nous avons aboutit et présente quelques perspectives.

2. Les variantes de TCP

Plusieurs variantes du protocole TCP existent dans la littérature. Ces différentes variantes ont été proposées dans un seul but : permettre à TCP de réagir au mieux aux pertes de paquets. Dans cette section nous donnons un aperçu du fonctionnement des principales variantes de TCP.

2.1 TCP de base (Old-Tahoe)

Dans les premières versions de TCP, l'émetteur envoie plusieurs segments consécutivement. Le nombre de ces segments lui est indiqué par le destinataire sous forme de taille de fenêtre à utiliser. Le récepteur acquitte alors uniquement le dernier segment reçu correctement par un acquittement accumulatif. En cas de perte d'un segment, il n'existe aucun moyen de l'indiquer à l'émetteur. Pour palier cela, dans ses dernières versions, TCP utilise un délai d'attente de retransmission, RTO (*Retransmission TimeOut*), que l'émetteur doit attendre avant de s'apercevoir de la perte d'un segment et pouvoir le récupérer[1]. À cause de ce temps perdu avant de s'apercevoir de la perte d'un segment, l'émetteur va fonctionner à vide pendant un certain temps et va ensuite envoyer tous les paquets suivants même s'il les a déjà envoyé. Par conséquent, la route entre les deux nœuds sera sous-utilisée et la durée de la communication sera plus grande. Les auteurs de [4] ont prouvé que lorsque la durée de la communication est grande, la consommation de l'énergie est plus élevée. Ainsi, dans cette variante de TCP, la consommation de l'énergie est très élevée quand les taux d'erreur sont importants dans le réseau. Ceci est du au temps, RTO, passé avant de retransmettre les paquets perdus.

Cette version initiale de TCP était donc inadéquate à la structure de l'Internet et a été rapidement remplacée. Elle ne sera donc pas analysée dans la suite de ce papier.

[1] Le calcul du RTO est détaillé dans [3].

2.2 TCP Tahoe

TCP Tahoe date de 1988 et n'est plus utilisée aujourd'hui. Dans cette version, de nouveaux algorithmes de contrôle de congestion sont rajoutés : *Slow-Start*, *Congestion Avoidance* et *Fast Retransmit* [5][6].

Les algorithmes de *Slow-Start* et de *Congestion Avoidance* sont indépendants et ont des objectifs différents. Le *Slow-Start* augmente la taille de la fenêtre de congestion rapidement afin d'atteindre, le plus tôt possible, le débit de transmission maximal. Quant au *Congestion Avoidance*, il augmente la taille de la fenêtre lentement afin d'éviter, le plus longtemps possible, la perte de paquets. En pratique, ces deux algorithmes sont utilisés conjointement. L'algorithme *Slow-Start* est utilisé en premier pour permettre à une connexion TCP d'atteindre rapidement une certaine valeur de débit. Une fois cette valeur atteinte, c'est l'algorithme *Congestion Avoidance* qui prendra le relais pour continuer à augmenter progressivement le débit.

L'algorithme *Fast Retransmit* vise à retransmettre les paquets perdus sans avoir à attendre un délai d'attente de retransmission, RTO. En effet, après avoir reçu un petit nombre d'acquittements dupliqués (*duplicate ACKs*) pour le même segment TCP (typiquement trois), l'émetteur infère qu'un paquet est perdu et retransmet le paquet sans attendre l'expiration du RTO. Ainsi, le *Fast Retransmit* assure une retransmission de paquets plus rapide, ce qui améliore le débit et qui devrait économiser l'énergie consommée en même temps. Notons qu'après chaque retransmission, TCP Tahoe repasse par la phase *Slow-Start*.

2.3 TCP Reno

TCP Reno est la variante la plus populaire. Cette version est similaire à TCP Tahoe, excepté qu'il modifie l'algorithme *Fast Retransmit* pour y inclure le mécanisme *Fast Recovery* [5][7]. Le nouvel algorithme permet d'éviter que le canal de communication ne soit vide après un *Fast Retransmit* évitant ainsi le besoin de démarrer le *Slow-Start* pour le remplir de nouveau après la perte d'un seul paquet [8].

La différence entre TCP Tahoe et TCP Reno réside dans la façon d'estimer la perte de paquets. TCP Tahoe considère chaque perte de paquet comme un problème sérieux dans le réseau conduisant à diminuer le débit de transmission après chaque perte de paquet. Au contraire, TCP Reno peut différencier entre les deux cas suivants : (i) perte de paquet aperçue par le RTO (le réseau subit une congestion sévère) et (ii) perte de paquet aperçue par des acquittements dupliqués (la congestion dans le réseau n'est pas sévère). Dans le premier cas, TCP Reno diminue son débit de transmission à une valeur minimale et entre dans la phase *Slow-Start*. Dans le second cas, TCP Reno diminue son débit de transmission de moitié sans repasser par la phase *Slow-Start*.

2.4 TCP New-Reno

La version de TCP New-Reno est l'une des versions les plus utilisées actuellement. TCP New-Reno apporte une nouvelle modification permettant d'améliorer la réaction de TCP Reno face à la perte de plusieurs paquets d'un seul segment de données en même temps (perte en rafale) [9][10]. Les changements concernent le cas ou l'émetteur ne reçoit qu'un acquittement partiel (*partial ACK*), après avoir retransmis des paquets perdus en rafale en utilisant l'algorithme *Fast Recovery*. Un acquittement partiel permet au récepteur de rendre compte à l'émetteur du fait que parmi les paquets retransmis une première fois, certains ont à nouveau été perdus. Dans ce cas, avec New-Reno un accusé de réception partiel ne fait pas quitter l'algorithme *Fast Recovery* [8]. Dès lors, à la différence de Reno, cette version n'attend pas un RTO pour retransmettre ces paquets perdus.

2.5 TCP SACK

Dans [11], les auteurs proposent d'étendre le contenu des accusés de réception. Alors que les accusés de réception ne permettaient pas de savoir avec certitude le numéro du ou des

segments perdus lors d'une congestion du réseau, TCP SACK (*Selective Acknowledgement*) apporte une solution à ce problème en utilisant un champ particulier '*SACK'* dans la donnée d'ACK. Via ce champ, les accusés contiennent des informations, appelés 'blocs', permettant d'indiquer les discontinuités entre les paquets réceptionnés. Les blocs contiennent ainsi l'information permettant de savoir quel est, ou quels sont les paquets à retransmettre. Comme il s'agit d'une option de TCP, l'algorithme implémentant SACK inclut les versions de TCP Reno et New Reno.

2.6 TCP Westwood

TCP WestwoodNR est une variante modifiée de TCP New-Reno. En fait, il existe deux variantes de TCP Westwood : une basée sur TCP Reno et l'autre basée sur TCP New-Reno. Nous nous intéressons à la seconde variante dans ce papier. Elle est connue dans la littérature sous le nom TCP WestwoodNR.

Dans [12], TCP WestwoodNR est décrit comme étant une version efficace pour le transport d'informations pouvant circuler sur des réseaux sans fil. Cette version se base sur une estimation de la bande passante disponible pour régler les mouvements de la fenêtre de congestion en adaptant les seuils présents dans les algorithmes de traitement de congestion sur base de ces estimations. L'objectif de l'estimation de la bande passante est de permettre de différencier entre les pertes dues aux erreurs liées au support de transmission, des pertes liées aux congestions.

2.7 TCP Vegas

TCP Vegas est une extension de TCP Reno. Dans TCP Vegas, l'émetteur enregistre les temps auxquels un segment est émis ainsi que les temps auxquels on reçoit l'accusé de réception correspondant. Quand un acquittement arrive à l'émetteur, TCP Vegas calcule le RTT en se basant sur l'horloge du système. Il utilise ensuite cette valeur pour décider s'il doit retransmettre le dernier segment émis ou non [13]. Par exemple, lorsque le RTT estimé est plus grand que la valeur du RTO, il retransmet le paquet suivant directement sans avoir à attendre les 3 acquittements dupliqués. Autrement dit, TCP Vegas traite la réception d'un acquittement comme un indice pour voir s'il est nécessaire de déclencher le RTO pour les paquets suivants ou non. Ainsi, TCP Vegas détecte la perte de paquets de manière plus rapide que les autres variantes [13]. L'objectif mis en évidence dans l'algorithme de Vegas, est d'essayer d'obtenir un taux de transmission plus élevé avec moins de retransmissions.

3. Problèmes de performance de TCP dans les réseaux ad hoc

Outre la nature de l'environnement sans-fil, les performances du protocole TCP sont fortement affectées par la nature du réseau ad hoc lui-même : mobilité des nœuds, routage et limitation des ressources (bande passante et batterie) [14][15]. TCP montre, en effet, un comportement peu désirable concernant la consommation de l'énergie à cause de sa fiabilité [16]. Les performances de TCP dépendent de différents facteurs tels que le modèle de mobilité, le modèle de trafic, la topologie du réseau, la position des obstacles, etc. Afin de mieux comprendre l'effet de tels facteurs sur les performances d'un réseau ad hoc, nous les avons classé en deux catégories : (i) facteurs liés à l'environnement sans-fil, et (ii) facteurs liés à la nature dynamique des réseaux ad hoc.

3.1 Facteurs liés à l'environnement sans-fil

Dans les réseaux sans-fil, il existe deux facteurs à ne pas négliger : les taux d'erreur (BER) relativement élevés et le routage multi-routes. Le premier facteur, peut engendrés des taux de perte non négligeables de segments TCP sur les liens sans fil. Ceci cause naturellement une consommation d'énergie pour la retransmission des paquets perdus. Cependant, dépendamment de la variante de TCP utilisée, ces pertes peuvent également provoquer une consommation d'énergie complémentaire et inutile. Le deuxième facteur est provoqué par les protocoles de routage qui maintiennent plus d'une seule route entre chaque paire

source/destination, afin d'éviter la recherche d'une nouvelle route lorsque c'est nécessaire [15][17]. Ces protocoles de routage sont utilisés afin d'avoir un équilibre de la consommation de l'énergie à travers tout le réseau, en envoyant le trafic de données sur des routes différentes [15]. Cependant, ceci entraîne également le fait qu'un certain nombre de paquets n'arrivent pas dans le même ordre au niveau du récepteur ; ce qui oblige le récepteur à envoyer des acquittements dupliqués qui vont forcer l'émetteur à invoquer l'algorithme de contrôle de congestion [18] causant ainsi une consommation d'énergie inutile provoquée par les retransmissions des paquets hors séquence.

3.2 Facteurs liés à l'environnement ad hoc mobile

En plus de l'effet de l'environnement sans-fil, la nature des réseaux ad hoc (mobilité des nœuds, partitionnement du réseau) a également une conséquence importante sur les performances de TCP. Ainsi, lorsqu'une route vers la destination ne devient plus valide, l'algorithme de routage tente d'en trouver une nouvelle. Cependant, il est possible que la découverte de cette nouvelle route prenne plus de temps que le RTO du côté émetteur. Dans ce cas, la source retransmet les données perdues et initialise l'algorithme de contrôle de congestion. Le débit de transmission sera faible même après la découverte de la nouvelle route et la connexion TCP inefficace. Dans un réseau ad hoc avec une grande mobilité (découvertes de route fréquentes), la connexion TCP ne pourra donc pas transmettre à un débit maximum (le débit de transmission sera faible comparé à celui annoncé par la destination) [18]. Nous remarquons ainsi qu'il est préférable d'arrêter momentanément la transmission et ne reprendre que lorsqu'une nouvelle route vers la destination soit disponible.

Il peut également arriver que le réseau ad hoc soit partitionné pendant un laps de temps plus ou moins important. Si la source et la destination se retrouvent dans des partitions différentes, tous les paquets transmis seront perdus et la source initialisera l'algorithme de contrôle de congestion. La situation du partitionnement du réseau risque d'isoler plus longtemps la source de la destination que la simple coupure d'un lien suivi d'une re-découverte de route. Lorsque le partitionnement dure longtemps (beaucoup plus longtemps que le RTO), il cause un phénomène appelé « *serial timeouts* ». Ce phénomène consiste en la situation où le même segment est envoyé consécutivement plusieurs fois à la destination, qui n'est malheureusement plus rattachée à la source. Ici également, l'idéal aurait été de marquer un temps d'arrêt avant chaque retransmission. Ceci permettra d'éviter d'inonder le réseau par des paquets qui n'arrivent jamais à la destination.

Il existe également d'autres facteurs influençant les performances de TCP dans les réseaux ad hoc [19]. Par exemple, une haute mobilité n'est pas toujours un handicape dans les réseaux ad hoc. Des travaux ont même montré que la mobilité peut améliorer les performances, puisqu'elle répartie le trafic uniformément dans le réseau [20][21]. Un autre facteur est la densité du réseau. En effet, dans le cas où les nœuds ad hoc sont proches, les paquets de données n'auront pas besoin de faire plusieurs sauts. Cependant, les réseaux à forte densité souffrent de problèmes d'interférence et de contention. Il faut également noter que les murs ou d'autres obstacles, empêchant les transmissions des ondes radios, diminuent largement l'effet de la densité du réseau.

4. Etude comparative de la consommation d'énergie des différentes versions de TCP

Dans cette section, nous décrivons les résultats que nous avons obtenus pour l'étude comparative de la consommation d'énergie des différentes versions de TCP. Ces simulations ont été réalisées en utilisant le simulateur NS-2 (Network Simulator version 2) [22]. La topologie de simulation consiste en un réseau de 20 nœuds dispersés sur une aire de (670x670m²). Lors des 400s de simulations, 14 connexions TCP sont établies (trafic ftp avec une taille de paquets égale à 512 octets et une taille de fichier à transférer infinie). Initialement, la capacité de la batterie de chaque noeud est de 10 joules. Cette énergie initiale est réduite progressivement par l'émission, la réception et la re-émission des paquets de données sur

l'interface sans fil. Dans le cadre de notre travail, nous considérons uniquement le cas le plus simple où la transmission et la réception d'un paquet consomme une quantité fixe d'énergie. Lorsque, la quantité d'énergie restante au niveau d'un nœud atteint zéro joules, le dit nœud sera considéré comme mort et ne pourra plus prendre part aux communications.

Dans le cadre de nos scénarios de simulation, nous considérons trois différentes valeurs de taux d'erreurs bits (5%, 10% et 15%) dues aux imperfections du canal radio. Nous considérons également le cas de rupture d'un lien dans le réseau (LL - *Link Loss*) due au départ ou à la mobilité d'un nœud. Notons que la rupture d'un lien provoque dans la plupart des cas des pertes en rafale pour les connexions TCP. Ces scénarios de simulation sont appliqués à chacune des versions de TCP. Deux paramètres de performances, liées au fonctionnement de TCP, sont étudiés. Le premier de ces deux paramètres est l'énergie consommée par l'émission, la réception et la re-émission des paquets de données par les nœuds du réseau divisée par le nombre de bits correctement reçus par les destinataires des connexions TCP. Ce paramètre est dénoté dans la suite par énergie consommée par bit reçu. Le second paramètre est le temps moyen des connexions TCP. En effet, il a été démontré dans [4] que l'énergie consommée au niveau de chaque nœud pour l'écoute du canal radio ainsi que celle consommée au niveau du CPU par l'exécution des algorithmes de TCP (*Fast Retransmit, Slow-Start* et *Congestion Avoidance*) sont proportionnelles à ce temps moyen.

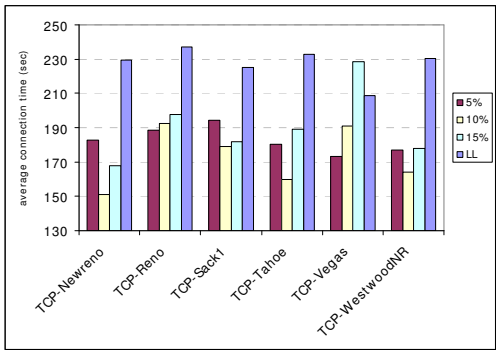

Figure 1. Le temps moyen de connexion TCP.

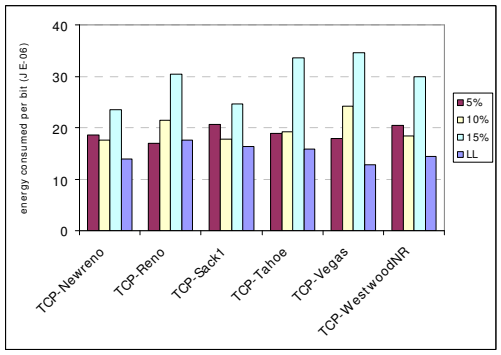

Figure 2. Energie consommée par chaque bit reçu.

4.1 La consommation d'énergie de TCP Tahoe

À partir de la Figure 2, nous pouvons conclure que nos résultats confirment bien ceux obtenus dans [23]. Nous pouvons en effet conclure que TCP Tahoe fonctionne mieux face aux pertes en rafale (LL) que dans le cas de pertes probabilistes dues aux imperfections du canal radio. Du point de vue de l'énergie consommée par bit reçu, TCP Tahoe a un comportement conservateur, car après chaque perte en rafale il arrête la transmission « *backs off* ». Ce comportement est logique : en effet lorsque une route est perdue à cause d'un de ses liens qui a été rompu, la connexion TCP doit s'arrêter le temps qu'une nouvelle route soit découverte. Dans [23], les auteurs n'ont cependant pas pris en considération le fait que cet arrêt de transmission introduit un délai, qui va prolonger le temps de connexion et, par la même, augmenter la consommation d'énergie en mode inoccupé. La Figure 1 explique l'effet du « *backing off* » de TCP sur le temps moyen de connexion. Donc, même si l'algorithme de « *backing off* » permet d'économiser l'énergie consommée par bit reçu dans le cas des pertes en rafale, la consommation d'énergie en mode inoccupé due à cet algorithme est quant à elle augmentée. Lorsque les taux d'erreurs dans le réseau sont élevés, TCP Tahoe consomme beaucoup d'énergie à cause du grand nombre de retransmissions. De plus, comme après chaque perte, TCP Tahoe est obligé d'entrer dans la phase *Slow-Start*, celui-ci diminue son débit de transmission et par conséquent augmente considérablement le temps de connexion moyen.

4.2 La consommation d'énergie de TCP Reno

À partir de la Figure 2, les résultas de simulation démontre que l'énergie consommée par TCP Reno face aux pertes probabilistes est inférieure à l'énergie consommée par TCP Tahoe dans des conditions similaires. Ceci est dû à l'algorithme *Fast Recovery* qui n'existe pas dans TCP Tahoe. L'algorithme *Fast Recovery* permet d'éviter que le canal de communication ne soit vide après un *Fast Retransmit*. Pour ce faire, l'algorithme *Fast Recovery* évite de démarrer le *Slow-Start* après la perte d'un seul paquet.

Dans le cas de pertes en rafale (LL), TCP Tahoe consomme moins d'énergie par bit reçu car il arrête la transmission (*backs off*) face à la première perte de paquets. Par ce fait, dans le cas des pertes en rafales, TCP Tahoe augmente les chances d'une retransmission correcte après l'arrêt de la transmission. En effet, comme les pertes en rafales sont dues à la rupture d'un lien ou à la mauvaise qualité de ce dernier, l'arrêt de transmission pendant un laps de temps permet à l'algorithme de routage ad hoc de reconstruire de nouvelles routes vers la destination ou d'attendre le rétablissement du lien perdu. Par conséquent, ceci aide à éviter des retransmissions inutiles pendant la période ou aucune route convenable n'est disponible dans le réseau. TCP Reno quant à lui n'arrête sa transmission qu'après avoir eu connaissance d'une deuxième perte consécutive à la première. Ce qui cause, dans le cas des pertes en rafale, une consommation d'énergie supplémentaire à TCP Reno par rapport à TCP Tahoe.

Parallèlement à cela, la Figure 1 montre que le temps moyen de connexion de TCP Reno est plus long comparé aux autres variantes de TCP. Ceci est d'autant plus marquant lorsque le taux d'erreur dans le réseau est élevé. Ceci est dû au fait que TCP Reno n'est pas capable de gérer plus d'une seule perte de paquet perdu. Or lorsque le taux d'erreur dans le réseau est élevé, deux voire plusieurs paquets peuvent être perdus consécutivement. Dans ce cas, TCP Reno diminue son débit de transmission de moitié à chaque perte de paquet. Par conséquent, après deux essais de recouvrement rapide après perte (*Fast Recovery*), TCP Reno atteint quasiment le même débit de transmission que TCP Tahoe. Après trois essais de recouvrement rapide après perte, TCP Reno doit attendre pendant un RTO pour s'apercevoir d'une nouvelle perte de paquets, alors il « *backs off* » et entre dans la phase de *Slow-Start* tout comme TCP Tahoe. Par conséquent, TCP Reno met plus de temps pour gérer les paquets perdus. La consommation d'énergie en mode inoccupé est ainsi supérieure à celle de TCP Tahoe. En effet, et comme mentionné précédemment, le temps moyen de connexion est proportionnel à l'énergie consommée dans le CPU pour exécuter les algorithmes de TCP plus celle consommée en écoutant les canaux radios. On peut donc conclure que, dans TCP Reno, l'exécution de l'algorithme *Fast Recovery* augmente la consommation d'énergie en mode inoccupé. Tous cela nous amène à conclure que TCP Reno consomme probablement plus d'énergie totale que les autres variantes de TCP dans la plupart de cas.

4.3 La consommation d'énergie de TCP New-Reno

À partir de la Figure 2, nous remarquons que TCP New-Reno économise l'énergie consommée par bit reçu dans quasiment tous les cas examinés et par rapport à toutes les autres variantes de TCP. Ceci est dû aux acquittements partiels utilisés. Grâce à l'utilisation de ces acquittements, TCP New-Reno n'a pas besoin d'attendre un RTO afin de retransmettre les paquets de données perdus. Nous pouvons ainsi remarqué (Figure 2) que dans le cas de perte en rafale, TCP Tahoe agit mieux que TCP Reno (comme déjà expliqué), et TCP New-Reno agit légèrement mieux que TCP Tahoe. Ceci est dû à la fois au fait que TCP New-Reno n'est pas obligé d'attendre le RTO avant de retransmettre les paquets perdus, mais également au fait que son débit de transmission accroît plus rapidement que dans le cas de TCP Tahoe.

Analysons maintenant la consommation d'énergie en mode inoccupé. À partir de la Figure 1, nous pouvons constaté que grâce aux acquittements partiels introduits par TCP New-Reno, le temps moyen de connexion de cette variante de TCP est moindre en comparaison à ceux de TCP Tahoe et TCP Reno. Ceci est effectivement vérifié dans les deux cas de figures étudiés : pertes en rafale et taux d'erreur élevés dans le réseau.

On conclut donc à partir des Figures 1 et 2 que TCP New-Reno possède le meilleur comportement en terme de consommation d'énergie en comparaison aux deux autres variantes de TCP déjà analysées : TCP Tahoe et TCP Reno. À partir des Figures 1 et 2, nous pouvons même conclure que TCP New-Reno possède globalement le meilleur comportement, en terme de consommation d'énergie, en comparaison à toutes les autres variantes de TCP. Nous allons en expliquer les raisons dans les sous-sections suivantes.

4.4 La consommation d'énergie de TCP SACK

La Figure 2 montre que TCP SACK ne consomme pas beaucoup d'énergie. Nous pouvons ainsi remarquer que TCP New-Reno agit légèrement mieux dans la plupart des cas. En fait, lorsque l'on compare les performances de TCP SACK et TCP New-Reno, dans les réseaux ad hoc, et plus particulièrement en terme de débit utile (*goodput*[2]), on remarque que TCP SACK se comporte mieux que TCP New-Reno dans la plupart des cas (c-à-d, possède un *goodput* plus élevé). Ce comportement de TCP SACK est du aux acquittements sélectifs qui lui permettent de terminer la retransmission des paquets perdus plus rapidement que TCP New-Reno (qui ne connaît pas précisément quels paquets ont été réellement perdus et donc quels paquets retransmettre impérativement). Cependant, en ce qui concerne la consommation d'énergie, le gain obtenu par TCP SACK est neutralisée. On pense que c'est à cause de l'en tête introduit par le champ SACK dans les paquets d'acquittements. En effet, les paquets d'Acquittements Sélectifs (SACK[3]) peuvent, dans certains cas, atteindre deux fois la taille[4] d'un paquet d'acquittement TCP (ACK). En retour, l'énergie consommée par chaque SACK envoyé sera plus élevée.

La Figure 1 quand à elle montre deux résultats distincts :

- Le premier est que TCP SACK à un temps moyen de connexion court dans le cas des pertes en rafale (LL). Ceci est évident vu que TCP SACK ne retransmet que les paquets perdus. Ainsi, les paquets suivants une perte en rafale, qui auront utilisé une nouvelle route (découverte par le protocole de routage) que celle ayant provoquée la perte en rafale, ne seront pas retransmis. Parallèlement à cela, TCP New-Reno doit quant à lui retransmettre tous les paquets suivants une perte en rafale. En outre, TCP New-Reno ne peut retransmettre qu'un seul paquet par RTT. Ainsi, lors d'une perte de rafale dans le réseau, on attend une période de temps égal au nombre de paquets perdus multiplié par la valeur du RTT. Donc, plus on a de paquets a retransmettre plus long sera la période de retransmission. Ce qui n'est pas le cas avec TCP SACK.

- Le deuxième résultat constaté à partir de la Figure 1 est que, dans le cas de taux d'erreur élevés, le temps moyen de connexion de TCP New-Reno est inférieur à celui de TCP SACK. En effet, lorsqu'un paquet retransmis est perdu encore une fois, ce qui est commun si les taux d'erreur dans le réseau sont élevés, TCP SACK n'aperçoit cette perte qu'après un RTO. Ainsi, il retransmet le paquet et entre dans la phase *Slow-Start*. Ceci n'est pas le cas avec TCP New-Reno. TCP New-Reno n'entre pas dans la phase *Slow-Start*, il diminue son débit de transmission de moitié.

Nous noterons également que l'énergie consommée en mode inoccupé par unité de temps due à l'exécution de TCP SACK est plus élevée en comparaison à celle consommée par TCP New-Reno. En effet, en plus des algorithmes liés au fonctionnement traditionnel de TCP,

---

[2] Quantité de données correctement reçues par le destinataire pour un laps de temps donné.

[3] La taille des SACK = IP Header + TCP ACK Header + SACK option = 20 octets + 20 octets + 40 octets = 80 octets. 40 octets est la taille maximale de l'en tête d'une option de TCP. SACK peut les utiliser entièrement pour envoyer les acquittements sélectifs « *Selective Acknowledgements*». Cette taille dépend du nombre de segments à acquitter par chaque SACK.

[4] La taille normale d'un paquet d'acquittement TCP = IP en tête + TCP ACK en tête = 20 octets + 20 octets = 40 octets.

TCP SACK doit maintenir (mémorisation, algorithme de mise à jour, etc.) une liste des paquets considérés comme perdus en fonction des acquittements sélectifs (SACK) reçus. Par conséquent, même si le temps moyen de connexion de TCP SACK est légèrement inférieur dans le cas des pertes en rafale, nous pensons que la quantité d'énergie consommée par TCP SACK est probablement plus élevée que celle consommée par TCP New-Reno.

4.5 La consommation d'énergie de TCP WestwoodNR

La Figure 1 montre que le temps moyen de connexion dans TCP WestwoodNR est le même que dans TCP New-Reno dans le cas d'une perte en rafale. En fait, lorsqu'un lien est rompu dans le réseau, la connexion TCP éprouve des pertes en rafale et les deux variantes de TCP attendent un temps RTO avant de retransmettre les données perdues. Comme ces deux variantes possèdent le même comportement par rapport aux pertes en rafale, elles ont quasiment les mêmes performances en ce qui concerne le temps moyen de connexion pour ce type de pertes. La Figure 1 montre également que lorsque le taux d'erreur dans le réseau est faible (5%), le temps moyen de connexion de TCP WestwoodNR est inférieur au temps moyen de connexion de TCP New-Reno. Ceci est dû au fait qu'il est capable d'adapter son débit de transmission selon la bande passante estimée dans le réseau au lieu de le diminuer automatiquement de moitié comme c'est le cas de TCP New-Reno. Cependant, lorsque les taux d'erreur sont élevés (10% et 15%), TCP New-Reno a un meilleur comportement, en ce qui concerne le temps moyen de connexion, en comparaison à TCP WestwoodNR. En effet, lorsque le taux d'erreur est élevé, un nombre important d'acquittements est perdu. Or l'algorithme d'estimation de la bande passante de TCP WestwoodNR est basé sur le calcul des temps d'aller-retour RTT et donc sur la réception des paquets d'acquittement. La perte de plusieurs paquets d'acquittement cause donc des disfonctionnement de l'algorithme d'estimation de la bande passante. Ces disfonctionnements font qu'à la fois le temps moyen de connexion et l'énergie consommée par bit reçu (Figure 2) sont plus élevés par rapport à TCP New-Reno.

À partir de la Figure 2, nous pouvons remarquer que TCP WestwoodNR et TCP New-Reno consomment quasiment la même quantité d'énergie par bit reçu dans la plupart des cas examinés (pour les mêmes raisons illustrés précédemment). En effet, lorsque les taux d'erreur dans le réseau sont de 5% ou 10%, nous avons remarqué que la proportion entre le nombre de bits reçus par la destination et le nombre total de bits transmis incluant les bits perdus, reste presque la même. Ceci reste vrai en dépit du fait que les temps moyens de connexion sont différents pour les deux variantes de TCP : TCP WestwoodNR et TCP New-Reno. Ceci est dû au fait que les deux variantes de TCP introduisent des délais de façon différente dans la retransmission de paquets perdus. Plus précisément, nous pensons que la taille de la fenêtre de congestion n'est pas bien dimensionnée dans les deux variantes. Autrement dit, par moment c'est TCP WestwoodNR qui a le meilleur débit de transmission et par moment c'est TCP New-Reno.

4.6 La consommation d'énergie de TCP Vegas

À partir des Figures 1 et 2, nous remarquons que lorsque les taux d'erreur ne sont pas importants dans le réseau (5%), TCP Vegas, en comparaison aux autres variantes de TCP, possède le meilleur comportement aussi bien pour la consommation d'énergie par bit reçu que pour le temps moyen de connexion. Ceci est dû au fait que TCP Vegas vérifie l'évolution du timeout après chaque acquittement reçu et décide de retransmettre des paquets qu'il va considérer comme perdus. Ceci permet un dimensionnement adéquat de la valeur des timeout en comparaison avec l'utilisation pur et simple de la valeur RTO dans les autres variantes de TCP. Cela conduit à gérer plus rapidement les pertes de paquets. Cependant, lorsque les taux d'erreur sont importants dans le réseau (10% et 15%), TCP Vegas possède les performances les plus mauvaises parmi toutes les variantes de TCP. En effet, dans le cas où les taux d'erreur sont élevés, plusieurs acquittements sont perdus et TCP Vegas va désigner certains paquets comme perdus avant même la réception d'acquittements dupliqués. Ces paquets ayant

cependant été bien reçus par la destination et retransmis quand même. Ceci mène naturellement à une consommation d'énergie supplémentaire par rapport aux autres variantes de TCP.

Dans le cas des pertes en rafale, nous remarquons que TCP Vegas possède de meilleures performances en comparaison aux autres variantes de TCP. Ceci est valable aussi bien pour le temps de connexion moyen (Figure 1) que pour l'énergie consommée par bit reçu (Figure 2). Ceci est dû à sa capacité à bien estimer le timeout avant chaque retransmission. En effet, TCP Vegas décide d'une retransmission d'un paquet immédiatement après la réception d'un ACK. Ainsi, dans le cas où un lien est rompu et que les routes sont symétriques (ce qui est le cas dans nos scénarios), on ne peut recevoir de nouveau des ACK que si une nouvelle route est à nouveau disponible. Par conséquent, le temps d'arrêt de la transmission dans TCP Vegas, lors des ruptures de liens, est dimensionné de façon plus optimale en comparaison aux autres variantes de TCP. Ceci permet donc à la fois de minimiser le temps moyen de connexion mais également d'utiliser les routes plus efficacement (plus longtemps). Ceci augmente la proportion des paquets correctement reçus et par la même diminue la consommation d'énergie par bit reçu.

4.7 Synthèse des résultats obtenus

Les résultats de simulation obtenus montrent que lorsque les taux d'erreur sont élevés dans le réseau (15%), TCP New-Reno est le plus performant suivi par TCP SACK. Lorsque les taux d'erreur ne sont pas élevés dans le réseau (5% et 10%), TCP New-Reno consomme moins d'énergie par bit reçu que les autres variantes. En outre, TCP Vegas et TCP New-Reno ont la meilleure consommation d'énergie (la plus basse) en cas de perte en rafale dans le réseau. Quand nous comparons le temps moyen de connexion pour les différentes variantes, nous trouvons que lorsque le taux d'erreur est à 10% ou 15%, TCP New-Reno a le meilleur comportement, suivie par TCP WestwoodNR. Dans le cas des pertes en rafale, c'est TCP Vegas et TCP SACK suivies par TCP New-Reno qui ont les meilleures performances en ce qui concerne le temps moyen de connexion. Cependant, pour des taux d'erreur bas (5%), les meilleures performances en terme de temps moyen de connexion sont celles de TCP Vegas et TCP WestwoodNR. Ces deux variantes de TCP sont en effet capables à mieux adapter leur débit de transmission pour ce scénario de perte. Pour résumer nous pouvons dire que nos résultats montrent que dans la plupart des cas examinés, TCP New-Reno est le plus performant dans un réseau ad hoc statique. C'est prouvé également que l'algorithme de contrôle de congestion de chaque variante TCP a un effet sur l'énergie consommée dans les réseaux ad hoc.

5. Conclusion

Même si TCP a été à l'origine conçu pour les réseaux filaire, nous avons constaté lors de cette étude qu'il pouvait également être très efficace dans les réseaux ad hoc. Nous pouvons même dire, après notre étude, que TCP a un comportement conservateur en terme d'énergie lorsqu'il est face à des pertes en rafale. Notons que ce cas de figure est très commun dans les réseaux ad hoc car pouvant résulter de la rupture d'un lien dans le réseau. Lors de notre étude comparative des différentes variantes de TCP, nous avons également trouvé que TCP New-Reno offre le meilleur comportement globale en terme de consommation d'énergie en comparaison aux autres variantes de TCP. En effet, TCP New-Reno gère efficacement (par rapport à la consommation d'énergie) tous les cas de perte dans un réseau ad hoc statique. Cependant, l'inconvénient majeur de TCP New-Reno est que l'émetteur ne retransmet qu'un seul paquet par RTT. Par conséquent, lorsque plusieurs paquets sont perdus, TCP New-Reno les récupère après une période d'attente importante qui n'est dans certains cas pas nécessaire (cas de pertes en rafale).

Dans les scénarios étudiés ici, nous ne prenons pas en considération ni la mobilité des nœuds dans le réseau ni l'effet des protocoles de routage. Après avoir étudié le comportement des différentes variantes de TCP dans un réseau ad hoc statique sous différents modèles de perte de paquets, nous pensons dans un travail future étudier l'effet des autres paramètres

influant la consommation d'énergie dans un réseau ad hoc (choix du protocole de routage et mobilité des nœuds).

6. Références

[1] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J.–C. Chen, "A survey of energy efficient network Protocols for wireless Networks", *ACM/Baltzer Journal on Wireless Networks*, vol. 7, no. 4, pp. 343-358, 2001.

[2] S. Senouci, and G. Pujolle, "Energy efficient consumption in wireless ad hoc networks", *IEEE ICC'2004 (International Conference on Communications)*, Paris, June 2004.

[3] G. R. Wright and W. R. Stevens, "TCP/IP Illustrated, Volume I (The Protocols)", *Addison Wesley*, 1994.

[4] K. Pentikousis, "TCP in wired-cum-wireless environments", *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, Fourth Quarter 2000.

[5] V. Jacobson., "Congestion avoidance and control", *ACM SIGCOMM*, Stanford, CA, pp. 314-329, August 1988.

[6] M. Allman, V. Paxon, and W. Stevens, "TCP Congestion Control", *IETF RFC 2581*, April 1999.

[7] V. Jacobson, "Modified TCP Congestion avoidance Algorithm", *Email to the end2end-interest mailing list*, ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt, April 1990.

[8] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and sack TCP", *ACM computer communications review*, vol. 26, no. 3, pp. 5-21, July 1996.

[9] J. Hoe, "Start-up Dynamics of TCP's Congestion Control and Avoidance Scheme", *Master's thesis, MIT*, June 1995.

[10] D. D. Clark and J. Hoe, "Start-up Dynamics of TCP's Congestion Control and Avoidance Scheme", *Presentation to the Internet end2end Research Group*, June 1995.

[11] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP Selective Acknowledgement option", *IETF RFC 2018*, October 1996.

[12] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth Estimation for enhanced transport over wireless links", *International conference on mobile computing and networking (Mobicom'2001)*, pp. 287-297, Rome, Italy, Jul. July 2001.

[13] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson "TCP Vegas: New Techniques for Congestion Detection and Avoidance", *ACM SIGCOMM*, pp. 24-35, London, U.K., October 1994.

[14] V. Ramarathinam, M. A. Labrador, "Performance Analysis of TCP over Static Ad Hoc Wireless Networks", *ISCA Conference on Parallel and Distributed Computing Systems (PDCS'2002)*, pp. 410-415, September 2002.

[15] S. Senouci, "Application de techniques d'apprentissage dans les réseaux mobiles", *Phd. Thesis*, University of Paris 6, October 2003.

[16] V. Tsaoussidis, A. Lahanas and C. Zhang, "The Wave and Probe Communication mechanisms", *Journal of Supercomputing*, vol.20, no.2, pp.115-135, September 2001.

[17] M. K. Marina, S. R. Das, "Ad hoc on-demand multipath distance vector routing", *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 3, pp. 92–93, July 2002.

[18] J. Liu, S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks", *IEEE Journal on Selected Areas in Communications (JSAC'2001)*, vol. 19, no. 7, pp. 1300-1315, July 2001.

[19] O. Westin, "TCP Performance in Wireless Mobile Multi-hop Ad Hoc Networks", *Master's Thesis*, SICS - Swedish Institute of Computer Science, December 2003.

[20] N. Gupta and S. R. Das. "A capacity and utilization study of mobile ad hoc networks", *IEEE LCN'2001 (Conference on Local Computer Networks)*, pp. 576-583, Tampa, Florida, November 2001.

[21] W.-H. Liao, J.-P. Sheu, and Y.-C. Tseng, "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks", *Telecommunication Systems*, vol. 18 no. 1, pp. 37-60, September 2001.

[22] Network Simulator – NS-2, **www.isi.edu/nsnam/ns/**.

[23] M. Zorzi and R. Rao, "Energy Efficiency of TCP in a local wireless environment", *Mobile Networks and Applications*, vol. 6, no. 3, July 2001.