Energy-efficient reliable transport protocols for IP-based low power wireless networks

Ahmed Ayadi

N° d'ordre : 2012telb0234

Sous le sceau de l'Université européenne de Bretagne

Télécom Bretagne

En habilitation conjointe avec l'Université de Rennes 1

Ecole Doctorale – MATISSE

Energy-efficient reliable transport protocols for IP-based low-power wireless networks

---

Thèse de Doctorat

Mention : Informatique

Présentée par **Ahmed Ayadi**

Département : RSM

Directeur de thèse : Xavier Lagrange

Soutenue le 25 juin 2012

Jury :

M. César Viho, professeur à l'université Rennes 1 (Président)
M. Bernard Tourancheau, professeur à l'université de Lyon 1 (Rapporteur)
M. Andrzej Duda, professeur à Grenoble INP-Ensimag (Rapporteur)
M. Xavier Lagrange, professeur à Télécom Bretagne (Directeur de thèse)
M. Claude Chaudet, maitre de conférence à Télécom ParisTech (Examinateur)
M. Patrick Maillé, maitre de conférence à Télécom Bretagne (Examinateur)

Energy-efficient reliable transport protocols for IP-based, low-power multi-hop networks

Ahmed AYADI

Telecom Bretagne

September 6, 2012

Remerciment

Je voudrais remercier Monsieur Xavier Lagrange mon directeur de thèse, de m'y avoir accueilli et donné les moyens de mener à bien mes travaux.

Je remercie vivement mes deux encadrants David Ros et Patrick Maillé pour leur disponibilité et leurs précieux conseils qui m'ont permis d'enrichir mon travail, je les remercie également pour leur soutien tout au long du déroulement de ma thèse.

Je tiens à remercier profondément l'ensemble des doctorants et des stagiaires du département RSM avec lesquels j'ai eu des échanges scientifiques et culturels pendant toute la durée de la thèse.

Ma femme, ma fille, mes parents, et le reste de ma famille méritent bien plus qu'un remerciement pour m'avoir supporté et aidé pendant tout mon cursus, je leur dédié cette thèse.

Je ne saurais terminer ces remerciements sans penser aux membres du jury pour l'honneur qu'ils m'ont fait d'avoir voulu examiner et évaluer cette contribution et à toute personne qui a contribué, directement ou indirectement, à l'achèvement de ce travail.

Résumé en français

Introduction

Les réseaux à faible consommation d'énergie ont vécu une grande évolution depuis le début du XXIème siècle. De nombreux chercheurs se sont intéressés à l'étude de l'efficacité énergétique et ont proposé de nouvelles cartes réseaux sans fil à faible consommation d'énergie (par exemple les cartes IEEE 802.15.4). Cependant, la consommation énergétique d'un nœud mobile ne dépend pas seulement des protocoles des couches physiques et liaison de données, mais aussi des protocoles des couches supérieures. D'autre part, le déploiement de ces réseaux dans le monde réel a affronté d'autres obstacles que la consommation énergétique comme les problèmes de fiabilité et d'adressage. Une première solution a été proposée en 2003 par ZigBee alliance. La spécification ZigBee a complété la norme IEEE 802.15.4 en lui ajoutant quatre composantes principales: la couche réseau, la couche application, les périphériques ZigBee et les objets applicatifs. Toutefois, cette solution n'a pas obtenu un grand succès vu les problèmes d'évolutivité et d'intégration avec le grand réseau IP du monde Internet.

En 2005, un nouveau groupe de travail à l'IETF, nommé 6LoWPAN, a eu l'idée du déploiement d'IPv6 dans les réseaux à faible consommation d'énergie pour résoudre le problème d'adressage. Avec l'introduction d'IPv6, les nouveaux appareils sont devenus capables de communiquer aussi bien entre eux qu'avec tous les appareils IP à l'intérieur et à l'extérieur du réseau sans fil. Ces derniers, appelés aussi objets intelligents, ont changé le concept de l'Internet qui ne se limite pas qu'aux réseaux informatiques classiques, mais s'étend à tous les objets de la vie quotidienne. L'extension de l'Internet à tous les objets du monde réel représente la notion d'Internet des Objets.

Après le déploiement d'adressage IPv6, le regard des chercheurs s'est dirigé vers les couches supérieures du réseau à faible consommation d'énergie c'est-à-dire la couche transport des données et la couche application. Actuellement, l'UDP est le protocole de transport le plus utilisé dans les réseaux à faible consommation d'énergie. Il est vrai que l'UDP est utile pour les réseaux de faible consommation d'énergie car de nombreuses applications sont tolérantes aux pertes et qui ne demandent pas la fiabilité du transport des données. Toutefois, d'autres applications et services (tels que SSH et HTTP) ne sont pas tolérants aux pertes. Ce type d'applications nécessite un service fiable que l'UDP ne peut pas fournir. En outre, certains domaines d'application (tels que la santé, l'Armée, et la sécurité) imposent des contraintes de fiabilité fortes. Dans certains cas d'utilisation (par exemple, envoyer une mise à jour d'un capteur, ou l'envoi d'une requête demandant des informations spécifiques d'un capteur), il y a nécessité d'un moyen de transport fiable de données. D'autre part, le déploiement de TCP, le protocole de transport le plus utilisé dans les réseaux IP, dans les réseaux à faible consommation d'énergie rencontre diverses

difficultés telle que la consommation énergétique.

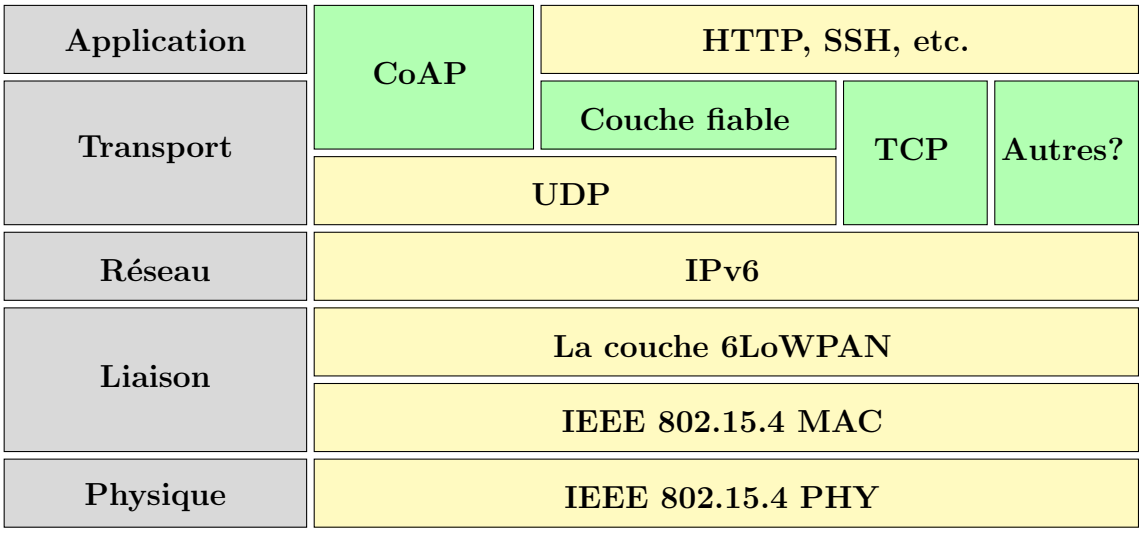


Figure 1: Les choix possibles d'un protocole de transport fiable au dessus des réseaux à faible consommation d'énergie

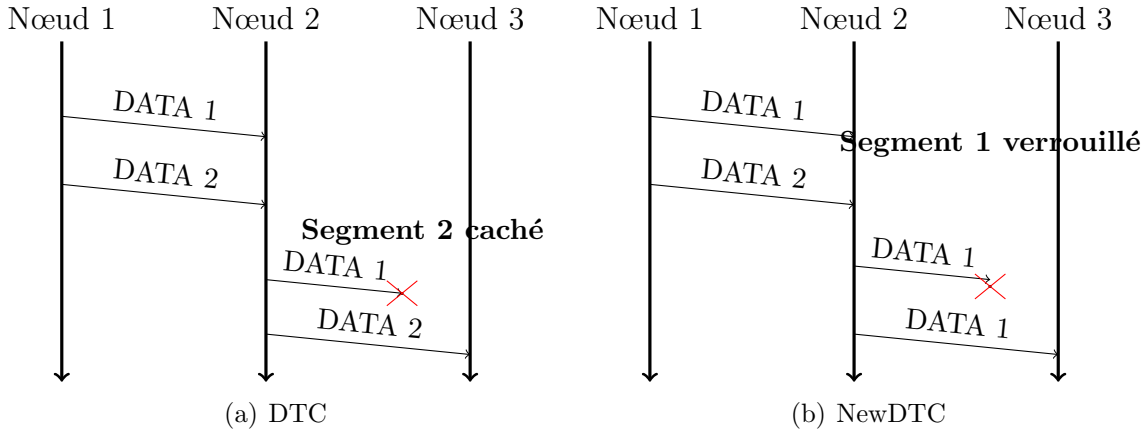
Dans ce contexte, nous pouvons imaginer quatre solutions possibles pour permettre un transfert fiable des données sur les réseaux de faible consommation d'énergie. Une première solution est de conserver l'un des protocoles de transport proposés dans la littérature pour les réseaux de capteurs et de l'adapter afin qu'il soit plus générique pour toutes sortes d'applications. Cependant, cette solution devra faire face au même problème de ZigBee. En effet, il ne sera pas facile d'intégrer cette solution à l'Internet et cela demandera le déploiement d'un proxy entre le monde sans fil et le monde filaire. Le proxy aura pour rôle de traduire les en-têtes du réseau IP (TCP ou UDP) en des en-têtes du protocole de transport du réseau sans fil. Une deuxième solution est de compléter l'UDP avec une nouvelle couche supérieure (comme le cas de RTP/RTCP). Cette solution doit offrir un transfert fiable de données qui n'est pas offert par l'UDP et doit ainsi implémenter des mécanismes de détection de pertes, de retransmissions, et de contrôle de congestion. Cependant, afin que la conception de ce protocole obéisse à toutes ces exigences, l'en-tête du protocole de transport doit comporter un numéro de séquence, un numéro d'acquittement, et d'autres champs qui sont déjà inclus dans l'en-tête TCP. Une troisième proposition, qui est actuellement en discussion au sein du groupe CORE à l'IETF consiste à laisser le contrôle de congestion et la récupération des pertes à la couche application. Toutefois, cette idée fournit une solution pour une catégorie limitée d'applications. Jusqu'à la rédaction de cette thèse, ce groupe de travail n'a prévu qu'une solution pour le HTTP sur l'UDP. Pour toutes ces raisons, nous avons choisi une quatrième proposition qui consiste à conserver TCP sur les réseaux à faible consommation d'énergie. Le choix de TCP nous permet de garder tous les

mécanismes implémentés par le protocole comme la retransmission des segments perdus et le contrôle de congestion. Dans ce travail, nous distinguons les limites de la mise en place de TCP sur les réseaux à faible consommation d'énergie, et nous proposons une solution pour chacun d'eux.

Dans nos travaux de recherche, nous nous sommes intéressés à l'étude de l'efficacité énergétique des protocoles de la couche de transport tout en conservant la fiabilité du transfert des données. Nos travaux se sont concentrés sur l'amélioration de l'efficacité énergétique du protocole de transport TCP dans les réseaux à faible consommation d'énergie. Ce chapitre présente un résumé des différents travaux de recherches réalisés en cours de thèse.

La retransmission des segments perdus

Afin de garantir la fiabilité des transferts de données, TCP retransmet de bout en bout les segments perdus. Dans les réseaux sans fil à multi sauts, la congestion n'est pas la seule cause de pertes car les mauvaises conditions du canal de transmission peuvent l'être aussi. Dans cet environnement, les performances de TCP peuvent se dégrader facilement et entraînent ainsi une baisse de débit et une augmentation du temps de transfert. De plus, dans le contexte des réseaux de faible consommation d'énergie, la perte d'un segment dans l'un des sauts du réseau sans fil entraîne une retransmission bout en bout de ce dernier alors que le paquet pourrait être à un ou deux sauts de la destination. Cette onéreuse retransmission de bout en bout peut être évitée si la retransmission se déclenche par le dernier nœud ayant reçu une copie du segment perdu. Cette idée suppose que les nœuds intermédiaires possèdent de l'espace mémoire afin de mémoriser les segments non acquittés. D'où l'idée de la retransmission de proche en proche des segments TCP perdus.



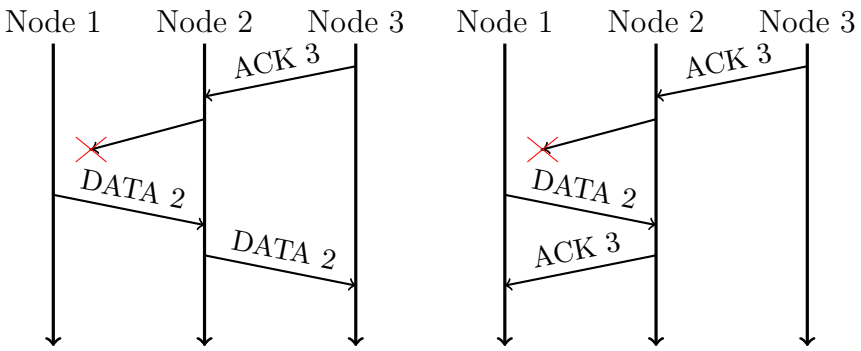
(a) DTC (b) NewDTC

Figure 2: La gestion du segment caché avec DTC et NewDTC

Distributed TCP Caching (DTC) est une des solutions qui a été proposées permettant aux nœuds intermédiaires de détecter les pertes et de retransmettre les segments perdus. Les auteurs présument que chaque nœud intermédiaire a assez de mémoire pour mettre en cache un seul segment TCP de données. Après la mise en cache d'un segment, un nœud DTC ne supprime un segment du cache que s'il reçoit un acquittement de la couche liaison de données. Un segment qui n'est pas acquitté par le prochain nœud est verrouillé dans la mémoire cache et retransmis après un délai avant retransmission (RTO). De plus, un segment de données verrouillé ne peut être écrasé par un autre segment TCP. Un segment caché est supprimé de la mémoire cache uniquement lorsque un acquittement (ACK) TCP est reçu.

Cependant, cet algorithme souffre de quelques lacunes. En effet, il a été proposé pour des réseaux TDMA alors que la majorité des réseaux à faible consommation d'énergie sont des réseaux CSMA-CD où le taux d'erreur est très élevé. De plus, la manière avec laquelle DTC gère le segment caché peut causer la perte de ce dernier sans qu'il soit bien reçu par la destination. Enfin, permettre aux nœuds de sauvegarder une copie du dernier segment et le retransmettre après un RTO peut créer de multiples retransmissions d'un seul segment TCP. Nous avons proposé une amélioration de DTC nommé Enhanced Distributed TCP Caching (EDTC). En effet, DTC propose de mettre en cache un segment reçu si la mémoire cache est vide. Il propose également de mettre en cache ce segment avec une probabilité de 50% si le cache n'est pas vide et non verrouillé. Un nœud DTC met en cache les segments les plus récents, ainsi les vieux devraient être retransmis. La figure 2 (a) montre que l'utilisation de cette approche conduit à une perte d'un segment si un nœud reçoit un nouveau segment TCP avant d'envoyer le segment déjà mis en cache. EDTC propose une meilleure gestion des segments cachés. Nous proposons une nouvelle approche permettant de remédier à ce problème. Après la réception d'un segment TCP données, un nœud NewDTC verrouille le segment reçu et l'envoie à la couche MAC essayant de l'envoyer à son tour au nœud prochain (voir la figure 2 (b)). A la réception d'un acquittement niveau deux (confirmant que le nœud suivant a bien reçu le segment), le nœud NewDTC déverrouille le segment reçu. Le segment mis en cache pourrait alors être remplacé par le prochain segment TCP ou supprimé si le même nœud reçoit un ACK TCP acquittant sa réception. Notre solution donne toute la priorité aux vieux segments s'ils ne sont pas envoyés avec succès à la couche liaison et la priorité au nouveau segment si le nœud est sûr que le suivant a bien reçu le segment mis en cache.

DTC propose que chaque nœud intermédiaire retransmette le segment mis en cache après un délai avant retransmission (RTO). Toutefois, les retransmissions locales de nœuds peuvent conduire à des retransmissions inutiles si le même segment est retransmis par plus qu'un nœud. Afin de réduire le nombre de retransmissions inutiles, un nœud NewDTC devrait recalculer le délai avant retransmission s'il reçoit un segment caché. Cette approche n'a aucun impact sur la retransmission des segments perdus. Un nœud NewDTC détecte



(a) DTC (b) NewDTC

Figure 3: TCP ACK loss recovery

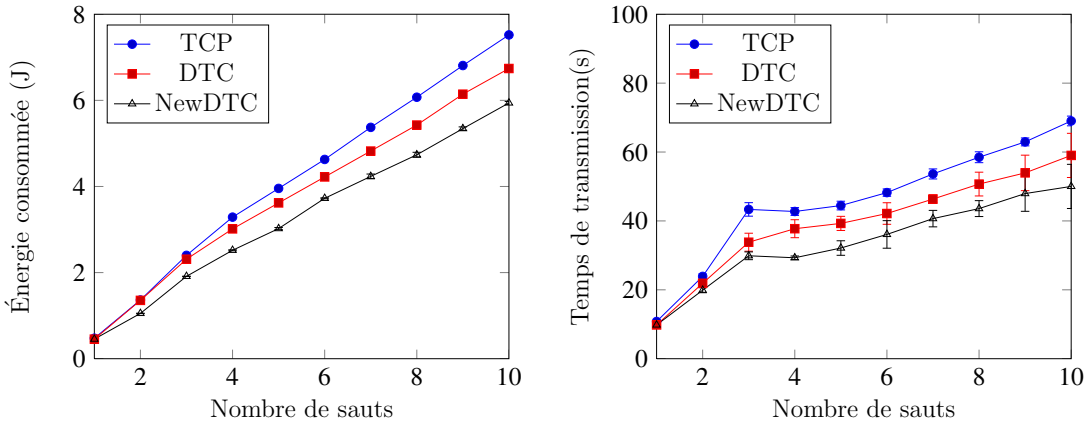
la perte d'un acquittement TCP en recevant un segment de données TCP qui a déjà été acquitté. Ainsi, un nœud NewDTC supprime le segment de données TCP reçu et régénère un acquittement. Cela exige que chaque nœud maintienne les états de toutes les connexions TCP.

La figure 3 montre un exemple d'une connexion TCP où un acquittement est perdu après qu'un segment ait été bien reçu. Le nœud 2 reçoit un acquittement TCP, mais ne parvient pas à l'envoyer au nœud 1. Ensuite, il reçoit un segment de données TCP déjà acquitté. Le nœud 2 supprime le segment TCP et régénère un acquittement TCP. Cette approche évite une nouvelle retransmission.

La figure 4 montre que DTC et NewDTC permettent de réduire la consommation totale d'énergie. Cela est dû aux retransmissions de proche en proche des segments perdus. Nous distinguons également que NewDTC réduit plus que DTC dans la consommation d'énergie en supprimant les segments TCP déjà acquittés. La figure 4 montre que NewDTC réduit la durée de transfert d'environ 35 à 60% par rapport à TCP et 14-45% par rapport à DTC. Cette performance est due à une meilleure gestion des segments cachés, tout en évitant les retransmissions inutiles suite aux pertes d'acquittements.

Réduire le taux d'acquittements

De nombreux chercheurs ont travaillé afin d'améliorer les performances de TCP dans les réseaux sans fil et ont proposé des algorithmes pour réduire le taux des acquittements par rapport aux segments de données. Ces derniers sont appelés des mécanismes d'acquittements retardés. L'idée principale de ces algorithmes est de retarder l'envoi d'un acquittement jusqu'à la réception de plusieurs segments de données (TCP permet de retarder l'acquittement de deux segments). Les deux mécanismes d'acquittements retardés



(a) Énergie consommée totale (b) Temps de transmission

Figure 4: Comparaison entre TCP, DTC et NewDTC en terme de consommation d'energie et temps de transmission

les plus connus sont TCP-TDA et TCP-DCA. Les deux algorithmes proposent qu'envoyer un acquittement TCP pour une serie de segments TCP reçus. L'algorithme TCP-TDA est plus simple à mettre en œuvre (par rapport au TCD-DCA) et ne nécessite pas que le récepteur calcule l'inter-arrivée des segments. En outre, le protocole TCP-DCA exige que l'expéditeur connaisse le nombre de sauts entre la source et la destination.

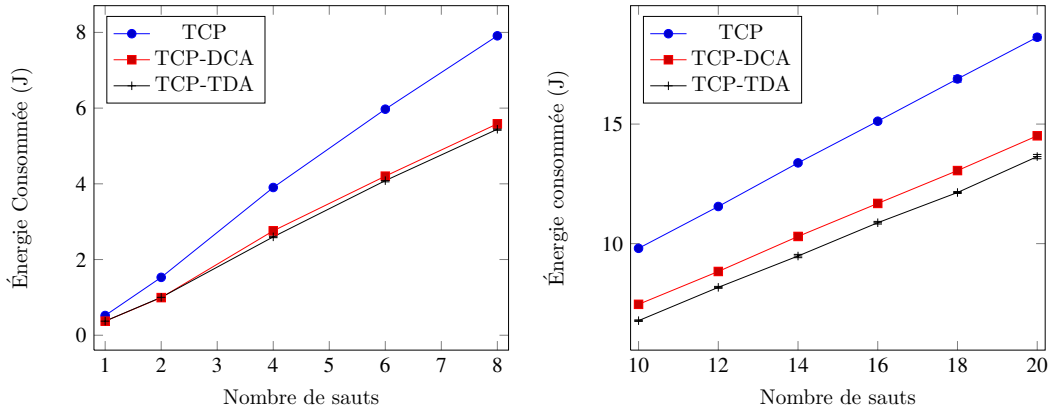


Figure 5: Comparaison entre TCP, TCP-TDA et TCP-DCA en terme d'énergie consommée

La figure 5 montre l'énergie consommée par tous les nœuds sans fil dans les différents algorithmes TCP. Nous pouvons voir que tous les algorithmes TCP d'acquittements retardés consomment moins d'énergie en les comparant au TCP et permettent de réduire l'énergie totale consommée de 40%. En outre, la figure 5 montre que le protocole TCP-TDA est plus économe en énergie que le protocole TCP-DCA. Cependant, l'inconvénient

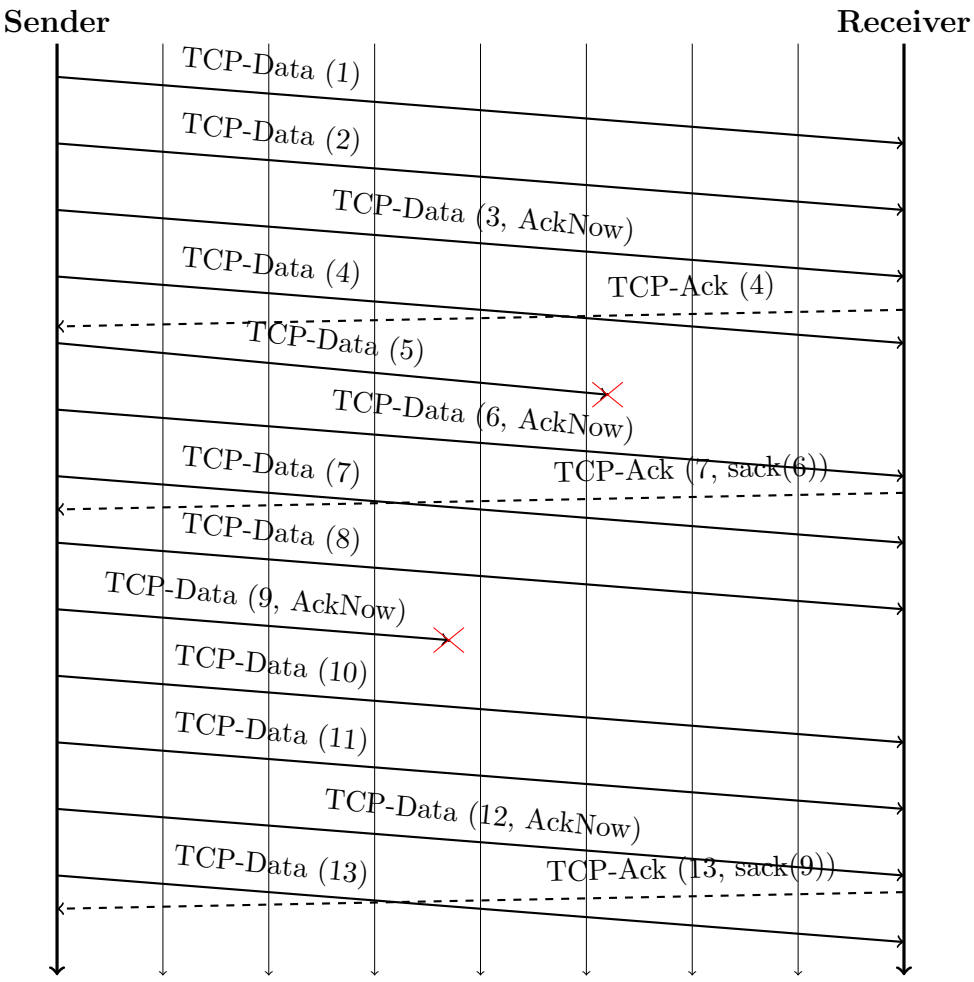


Figure 6: La récupération des acquittements TCP perdus

principal de TCP-TDA, c'est qu'il nécessite l'utilisation de la fenêtre demandée, qui devrait normalement informer l'expéditeur de la capacité du tampon de réception et du nombre de segments, successivement il doit envoyer un acquittement. Cette solution reste possible pour le transfert de données unidirectionnel et non pour les applications TCP très utilisées tels que SSH et HTTP. Nous proposons, à la place de l'envoi de la taille de la fenêtre de congestion, l'utilisation de l'un des bits réservés de l'en-tête TCP pour demander un acquittement. L'idée a été proposée auparavant par A. Oppermann [1] puis discutée par le groupe de travail TCPM dans [FARI10]. Le but était de réduire la congestion des acquittements et non de réduire la consommation d'énergie. La réduction du ratio des acquittements TCP aurait de mauvais impacts sur les performances TCP. La fenêtre de congestion TCP augmente d'une quantité constante à chaque acquittement reçu. Ainsi, un mécanisme

[1] http://www.ietf.org/mail-archive/web/tcpm/current/msg02356.html

d'acquittements retardés risque de diminuer le débit TCP en réduisant les acquittements. Dans [All03], Allman propose un autre mécanisme de contrôle de congestion pour faire face aux mécanismes d'acquittements retardés. L'idée principale était que la fenêtre de congestion TCP doit être augmentée en fonction du nombre d'octets acquittés par acquittements. Afin d'améliorer la performance TCA-TDA, nous proposons d'appliquer la même idée, même si le nombre d'acquittements est plus élevé que deux segments. En outre, l'option SACK TCP peut être utiliser afin de signaler si un ou plusieurs segments sont perdus. La figure 6 montre un exemple de scénario dans lequel l'un des segments envoyés est perdu. Le récepteur répond par un acquittement avec une option SACK pour informer l'expéditeur que le troisième segment n'a pas été reçu.

La compression d'en-tête TCP

L'en-tête TCP ajoute une surcharge importante par rapport à la taille totale d'une trame et surtout pour les petits paquets. Pour les acquittements TCP envoyés dans de petites trames (comme les trames IEEE 802.15.4 dont la taille maximale d'un paquet ne dépasse pas les 127 octets), la surcharge due aux en-têtes TCP représente plus de 70 % de la taille totale de la trame, ainsi, la transmission des champs inutiles ou redondants d'en-tête TCP doit être évitée.

Dans ce travail, nous proposons un nouveau mécanisme de compression d'en-tête TCP, appelé TCPHC, qui permet de réduire la taille de l'en-tête TCP jusqu'à six octets et ainsi augmenter le débit et réduire la consommation d'énergie. Nous considérons les réseaux 6LoWPANs comme un exemple de réseaux à faible consommation d'énergie dans notre étude.

TCPHC n'est pas seulement un algorithme de compression d'en-tête, mais fournit également un système permettant l'établissement de connexions TCP. Ces derniers ne sont pas seulement entre un nœud dans le réseau sans fil et un nœud externe, mais aussi entre les deux nœuds du réseau sans fil. Le premier type de connexion est réalisé à l'aide d'un routeur de bord (ER) qui relie le réseau sans fil au réseau IP externe. La figure 7 montre une topologie typique d'un réseau 6LoWPAN avec trois routeurs de bord qui créent trois ponts entre le réseau LoWPAN et le réseau IP externe.

Il existe trois types d'en-têtes TCPHC: un en-tête régulier (un en-tête normal, non compressé qui ne porte aucun identifiant de contexte (CID)), un en-tête complet (un en-tête non compressé qui rafraîchit le contexte. Il porte un CID pour identifier le contexte), et un en-tête compressé (un en-tête dans lequel tous les champs statiques sont éliminés, et tous les champs dynamiques sont envoyés compressés). Les mécanismes de compression et décompression ont été implémentés dans les routeurs de bord et les nœuds sans fil. Ainsi, les nœuds IP externes envoient et reçoivent des segments TCP réguliers, alors que

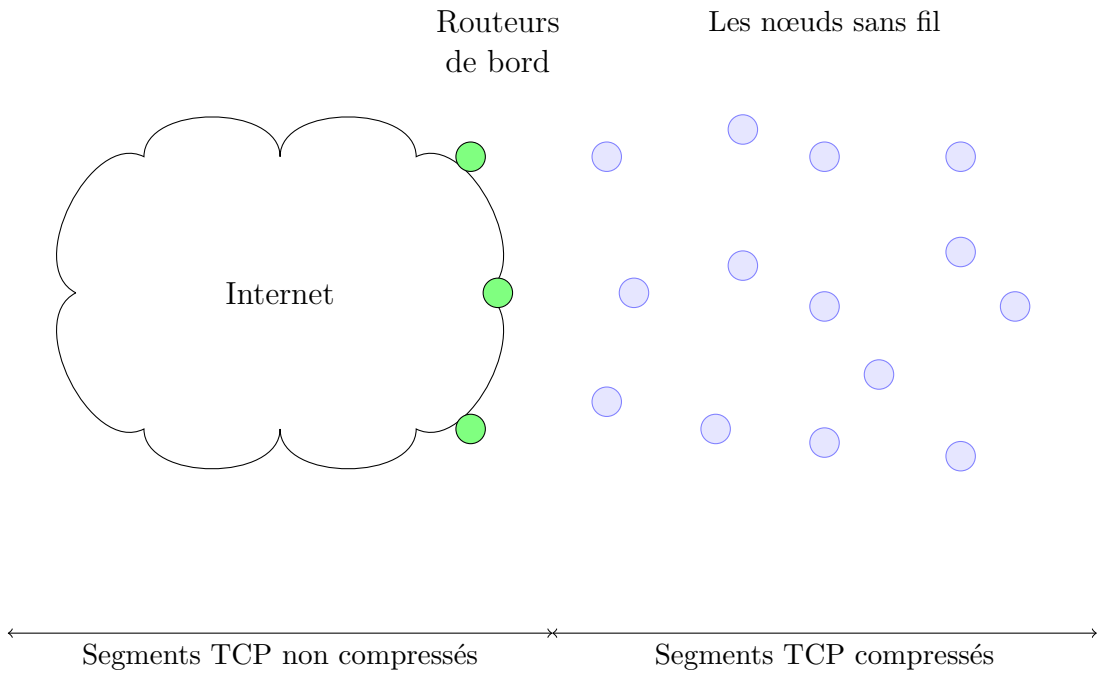
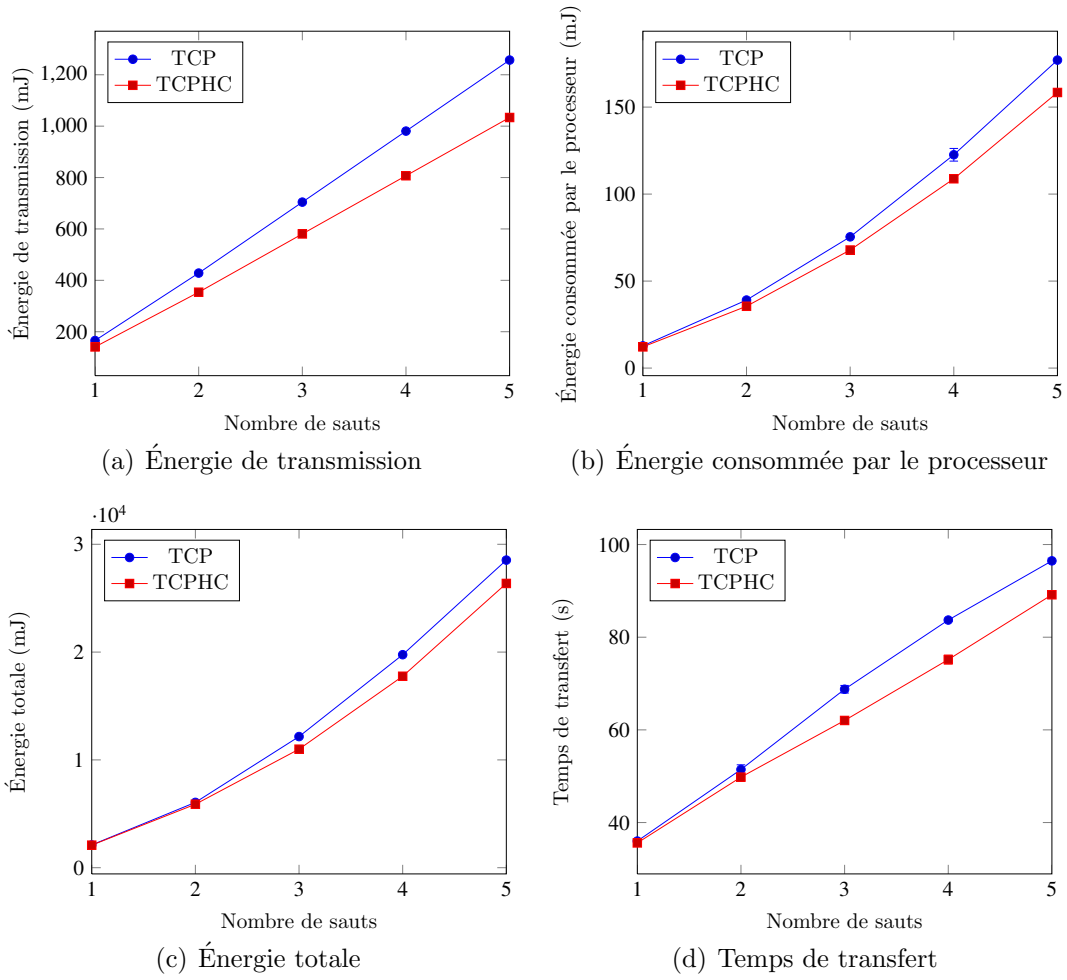


Figure 7: La compression d'en-tête TCP dans les réseaux 6LoWPANs

les nœuds sans fil envoient et reçoivent des segments TCP avec des en-têtes compressés ou des en-têtes complets.

Afin d'évaluer les performances de TCPHC, nous avons réalisé une évaluation expérimentale de TCPHC dans notre laboratoire. Nos résultats montrent que TCPHC réduit la consommation d'énergie de TCP. La figure 8 (a) montre que TCPHC réduit la consommation d'énergie de transmissions. Tout d'abord, nous ne distinguons pas une amélioration significative pour moins de trois sauts entre les deux nœuds. Toutefois, TCPHC réduit d'environ 17% l'énergie consommée par tous les nœuds de capteurs quand il y a 5 sauts entre l'expéditeur et le récepteur. La figure 8 (b) montre que l'algorithme de compression d'en-tête n'augmente pas l'énergie consommée par le processeur. Au contraire, l'énergie consommée par le processeur a été réduite par TCPHC. Ceci est dû à la nature de notre algorithme de compression qui n'ajoute pas les instructions nécessitant plus de calcul. La figure 8 (d) montre que le temps de transfert a diminué lorsque TCPHC a été déployé. TCPHC réduit la durée de transfert d'environ 9% par rapport à TCP et ainsi augmente le débit TCP. Enfin, la figure 8 (d) confirme les résultats précédents et montre que TCPHC réduit l'énergie totale par rapport à TCP.



(a) Énergie de transmission (b) Énergie consommée par le processeur

(c) Énergie totale (d) Temps de transfert

Figure 8: Les résultats expérimentaux de TCP et TCPHC dans les réseaux 6LoWPANs.

Impact de la taille des trames sur la consommation d'énergie

Nous avons étudié l'impact de la taille maximale des données dans un segment sur l'efficience énergétique de TCP selon les contraintes imposées par les couches inférieures. Nous avons pris les réseaux IEEE 802.15.4 comme un exemple de réseaux à faible consommation d'énergie. Comme été défini dans la norme, la taille maximale d'une trame IEEE 802.15.4 est de 127 octets. Le groupe 6LoWPAN de l'IETF a proposé une nouvelle couche protocolaire entre la couche IPv6 et la couche liaison de données IEEE 802.15.4.

La couche 6LoWPAN permet de compresser l'en-tête IPv6 (40 octets) et de fragmenter les paquets IP sous forme de petits fragments IEEE 802.15.4. Ainsi, un long segment

TCP (dont la taille dépasse les 127 octets) est fragmenté au niveau de la couche 6LoWPAN en de petits fragments IEEE 802.15.4 envoyés de façon indépendante. La perte d'un fragment induit la perte du segment TCP, et ainsi une nouvelle retransmission. D'autre part, l'utilisation de petits segments TCP permet d'envoyer un seul segment TCP dans une trame IEEE 802.15.4. L'envoi de petits segments TCP permet de régler le problème de la retransmission des différents fragments si l'un d'eux est perdu. Cependant, elle augmente la surcharge du nombre d'acquittements et la surcharge due aux en-têtes TCP dans tous les en-têtes. La figure 9 confirme qu'il est plus économe d'envoyer de courts segments lorsque le taux d'erreur est élevé, et d'envoyer de longs segments lorsque le taux d'erreur est faible.

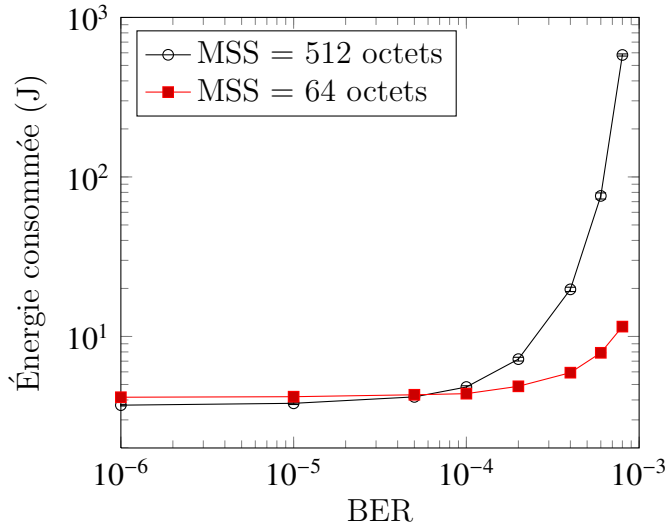
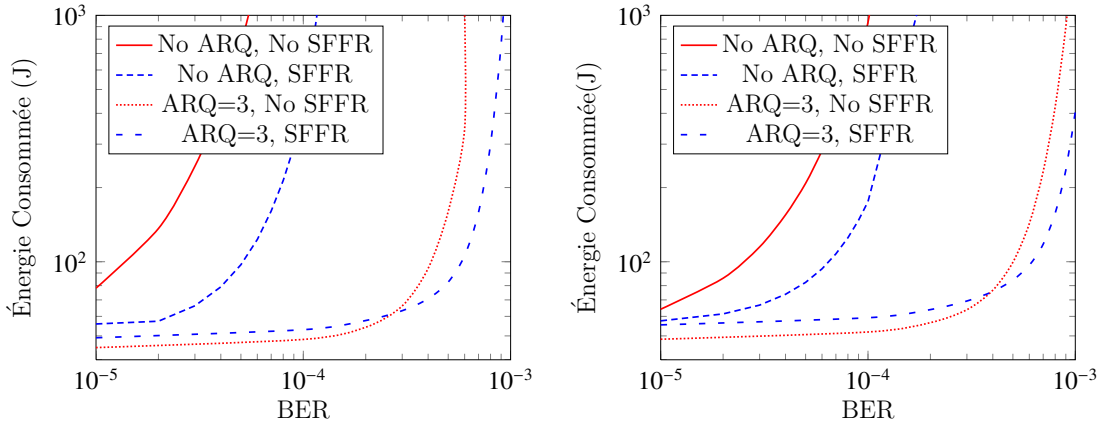


Figure 9: La consommation d'énergie avec des longs et des courts segments TCP en fonction du taux d'erreurs).

Comme l'estimation du taux d'erreur n'est pas si facile, nous avons étudié deux approches qui réduisent le taux d'erreur des segments. L'idée principale de la première approche, qui a été proposée par Thubert *et al.* [TH10], est de récupérer rapidement les fragments perdus avant une retransmission de la couche transport. L'algorithme, nommé SFFR, spécifie trois types de fragments: fragments récupérables (RFRAG), des fragments récupérables avec demande d'acquittement (RFRAG-AR) et des fragments accusé de réception (ACK-RFRAG). SFFR ajoute de nouveaux champs dans l'en-tête 6LoWPAN afin de spécifier le type et d'étiqueter les fragments pour pouvoir les rassembler ensuite.



(a) MSS = 1024 octets

(b) MSS = 512 octets

Figure 10: L'énergie consommée par l'ensemble des noeuds sans fils avec et sans SFFR (scenario avec 5 sauts).

Conclusion et Perspectives

Dans cette thèse, nous nous sommes intéressés à l'efficacité énergétique des protocoles de transport dans les réseaux de faible consommation d'énergie. Nous avons montré que le protocole TCP souffre de nombreuses limitations tels que la taille de son en-tête, le taux des acquittements et les retransmissions de bout en bout des segments perdus. Dans ce travail, nous avons essayé de proposer des améliorations/solutions simples pour chaque limitation. Nos contributions peuvent être facilement intégrées dans la version actuelle de TCP afin de le rendre plus viable pour les réseaux à faible consommation d'énergie. Dans ce qui suit, nous énumérons nos contributions.

En premier point, nous avons travaillé sur la réduction de la retransmission de bout en bout des segments perdus. Nous avons présenté un aperçu des algorithmes de récupération de proche en proche des segments perdus, puis, nous avons proposé une version améliorée de DTC, qui est l'un des algorithmes proposés pour TCP. Nos résultats de simulations ont montré que nos améliorations rendent DTC plus économe en énergie avec une meilleure gestion des segments en cache.

En second point, nous nous sommes concentrés sur la réduction du taux des acquittements TCP. Nous avons étudié deux algorithmes qui sont DCA et TDA. Ces deux algorithmes proposent d'envoyer un acquittement TCP pour une série de segments de données. Notre évaluation montre que les performances de TCP-TDA dépassent celles de TCP-DCA, non seulement en débit, mais aussi en efficacité énergétique.

En troisième point, nous avons proposé TCPHC, un nouvel algorithme de compres-

sion d'en-têtes TCP pour les réseaux de faible consommation d'énergie. TCPHC est un algorithme de compression d'en-tête robuste qui peut être mis en œuvre sur les systèmes d'exploitation embarqués tel que Contiki OS. TCPHC présente une nouvelle idée de compresser les champs dynamiques de l'en-tête TCP, en outre, il propose un moyen de compresser les options TCP. De plus, nous avons implémenté TCPHC sur Contiki OS et nous avons comparé ses performances à TCP. A partir de notre évaluation expérimentale, nous avons constaté que TCPHC est plus efficace que TCP au niveau la consommation d'énergie.

Un dernier point était d'étudier l'impact de la taille de segment TCP sur la consommation d'énergie. Notre étude montre qu'il est préférable d'envoyer de longs segments lorsque le taux d'erreur est faible, et de courts segments lorsque le taux d'erreur est élevé. Comme l'estimation du taux d'erreur n'est pas si facile, nous avons proposé deux approches qui réduisent le taux d'erreur des segments. L'idée principale de la première approche est de récupérer rapidement les fragments perdus avant une retransmission de la couche transport si elle existe. La deuxième approche consiste à résoudre le problème au niveau de la couche de transport et d'adapter dynamiquement la taille du segment sur la base des pertes observées. Les résultats des simulations confirment que ces deux approches permettent de réduire les retransmissions de bout en bout et ainsi la consommation d'énergie. Après cette étude nous sommes arrivés à conclure que TCP peut être adapté pour les réseaux de faible consommation d'énergie et mis en œuvre sur les nouveaux systèmes embarqués. En outre, les solutions proposées permettent de réduire la consommation d'énergie de TCP.

Ce travail est ouvert à plusieurs améliorations. Dans cette section, nous présentons les perspectives de chaque partie de ce travail qui sont les suivantes:

- Le réseau en chaîne était la seule topologie étudiée dans ce travail. Une future direction consiste à effectuer des simulations plus étendues afin d'évaluer les différents algorithmes de TCP dans des topologies plus complexes tels que les réseaux en grille. Avec ces nouvelles topologies, nos évaluations seront plus réalistes.

- Une autre perspective de ce travail est de combiner deux ou trois approches proposées afin de réduire la consommation d'énergie de TCP. Par exemple, l'idée d'employer l'algorithme de retransmission de proche en proche EDTC avec l'algorithme d'acquittements retardés TDA sera très intéressante parce qu'ils réduiraient les retransmissions de bout en bout et le nombre de segments d'acquittements échangés. Cependant, cette combinaison devrait tenir en compte plusieurs facteurs telle que la manière de calculer le délai avant retransmission.

- La compression d'en-tête de TCP pour les réseaux à faible consommation d'énergie est la contribution principale de cette thèse. Bien que les résultats d'évaluations montrent les bonnes performances de notre algorithme de compression, ce dernier reste encore ouvert à d'autres améliorations. D'une part, l'algorithme de compression

d'en-tête peut tenir en compte le déplacement d'un nœud sans fil et le changement de son routeur d'attachement. Après un changement de routeur d'attachement, le nouveau routeur demande le transfert des contextes de compression d'en-tête de TCP de l'ancien routeur. D'autre part, l'évaluation des performances de TCPHC réalisée dans ce travail est basée sur un scénario statique, où tous les nœuds sans fil sont fixes. Les futurs scénarios devraient prendre en considération la mobilité des nœuds sans fil, surtout que la nouvelle version de Contiki OS inclut le protocole de routage RPL.

- Le modèle analytique présenté dans ce travail peut être également amélioré, en particulier l'hypothèse que les taux d'erreur de tous les bits sont égaux. Le modèle de Gilbert-Elliott, qui est un modèle simple permettant de mieux modéliser les erreurs de transmission, peut être intégré dans notre modèle.

Abstract

Low power and Lossy Networks (LLNs) such as wireless sensor networks are currently used in many important applications fields such as remote environment monitoring and target tracking. This deployment has been enabled by the availability, especially in recent years, of embedded micro-controller devices that are smaller and cheaper. These devices are equipped with wireless interfaces, with which they can communicate with each other to form a network. In this thesis we focus on studying the energy consumption of reliable transport protocols over LLNs.

Recently, much research has been carried out to improve the reliability and the congestion control on low power networks. Some of these works have considered TCP inappropriate for this kind of networks. Indeed, the idea of deploying TCP was rejected due to its header overhead, its end-to-end retransmission mechanism, its large rate of acknowledgment, and the impact of the lower layers fragmentation on the energy consumption. Nonetheless, the use of standard TCP/IP protocols offers the advantage of a seamless connectivity between the wireless network and the Internet. TCP allows easily the use of standard applications (HTTP, SSH) for some tasks like reprogramming of nodes or firmware updates, without the need of deploying complex proxies in border routers.

In the first part of this work, we study the energy consumption of TCP and the ways that reduce its energy consumption. We study one of the proposed TCP algorithms to reduce the end-to-end retransmissions cost and we propose some improvements that allow it to reduce the energy consumption. Then, we study the compression of the TCP header over low-power and lossy networks and we consider IPv6 over Low power Wireless Personnel Area Networks (6LoWPAN) as an example. We propose a new TCP header compression algorithm that reduces the TCP header size to about six bytes.

In the second part, we propose a mathematical model that allows to estimate the energy consumption of wireless nodes. Using the model, we study the tradeoff between sending long and short TCP segments and their impact on the energy consumption. Finally, we study the impact of a new fragment recovery mechanism on the energy performance of TCP.

Keywords

Low power and Lossy Networks, 6LoWPAN, Transport Protocols, Energy Efficiency, TCP, Reliability.

Résumé

Les réseaux à faible consommation d'énergie tels que les réseaux de capteurs sans fil sont actuellement utilisés dans divers domaines militaires, environnementaux, médicaux, et commerciaux. Cette forte demande a été permise grâce à la disponibilité, surtout ces dernières années, de nouveaux microcontrôleurs qui sont plus petits, moins chers et plus intelligents. Ces nouveaux appareils sont équipés d'interfaces sans fil, avec lesquelles ils peuvent communiquer les uns avec les autres pour former un réseau. Dans cette thèse, nous nous concentrons sur l'étude de l'efficacité énergétique des protocoles de transport fiables pour ces réseaux.

Récemment, nombreux travaux de recherche ont été menées afin d'améliorer la fiabilité et le contrôle de congestion dans les réseaux à faible consommation d'énergie. Beaucoup de ces travaux ont considéré la pile protocolaire IP/TCP inadéquate pour ce type de réseaux. Initialement, l'idée a été rejetée à cause de plusieurs éléments, comme la surcharge de son en-tête, son mécanisme de retransmission de bout en bout, son taux élevé d'acquittements, et l'impact de la fragmentation des couches inférieures sur sa consommation d'énergie. Néanmoins, l'utilisation de la pile TCP/IP offre l'avantage d'une connectivité transparente entre le nouveau réseau sans fil et Internet. En effet, l'utilisation de TCP facilite l'utilisation des applications standards (e.g., HTTP, SSH) pour de nouveaux besoins comme la reprogrammation des nœuds ou la mise à jour du firmware sans une intervention humaine sur champ et sans un déploiement d'un proxy très complexe au niveau des routeurs de bord.

Dans la première partie de cette thèse, nous étudions la consommation d'énergie du protocole TCP et les manières qui permettent de réduire cette consommation. Nous étudions un des algorithmes TCP proposés afin de réduire le coût de la transmission de bout-à-bout et nous proposons une amélioration de ce dernier réduisant encore davantage plus la consommation d'énergie. Nous étudions ensuite la compression de l'en-tête TCP en prenant le réseau 6LoWPAN comme un exemple d'applications. Nous proposons un nouvel algorithme de compression qui permet aussi de réduire la consommation d'énergie. Dans une deuxième partie, nous proposons un modèle mathématique qui permet d'estimer la consommation d'énergie en fonction de plusieurs paramètres. A l'aide de ce modèle, nous étudions le compromis entre l'envoi de longs et courts segments TCP et leur impact sur la consommation d'énergie. Enfin, nous étudions ensuite l'impact d'un nouveau mécanisme de récupération des fragments sur la performance énergétique du protocole TCP.

Mots clés

Réseaux à faible consommation d'énergie, 6LoWPAN, Protocoles de transport, Efficacité énergétique, TCP, Fiabilité de transmission.

Contents

Remerciment **iii**

1 Introduction **1**
1.1 Context . . . 1
1.2 Motivation and Objectives . . . 3
1.3 Contributions . . . 4
1.4 Outline . . . 5

I Background **7**

2 Low power multi-hop wireless networks **9**
2.1 Introduction . . . 10
2.2 Low power networks . . . 10
2.3 Wireless Sensor Networks . . . 10
2.4 IPv6 over Low power Wireless Personal Area Networks . . . 12
2.4.1 IEEE 802.15.4 . . . 12
2.4.2 6LoWPAN . . . 13
2.4.3 Architecture . . . 13
2.4.4 Header compression . . . 14
2.4.5 Addressing . . . 15
2.5 Conclusion . . . 15

3 Reliable transport protocols over low power networks **17**
3.1 Introduction . . . 18
3.2 Transport protocols reliability for low power networks . . . 18
3.2.1 Congestion Control . . . 19
3.2.1.1 Congestion Detection . . . 20
3.2.1.2 Congestion Notification . . . 21
3.2.1.3 Congestion Avoidance . . . 21
3.2.2 Reliability . . . 24
3.2.2.1 Loss detection . . . 25
3.2.2.2 Retransmissions . . . 25
3.2.3 Energy Efficiency . . . 28
3.3 TCP/IP solutions for low power networks . . . 29
3.3.1 TCP caching and hop-by-hop recovery . . . 29
3.3.2 TCP dynamic delayed-acknowledgement . . . 30
3.3.3 Constrained Application Protocol (CoAP) . . . 31

3.4 Summary . . . 32
3.5 Conclusion . . . 34

II Reducing the energy consumption of TCP in multi-hop wireless networks 35

4 Making TCP more energy-efficient for low power networks 37
4.1 Introduction . . . 38
4.2 Why TCP for low power networks . . . 39
4.3 Distributed TCP Caching . . . 39
4.3.1 MAC Automatic Repeat reQuest . . . 40
4.3.2 Cache management . . . 41
4.3.3 Disabling unnecessary retransmissions . . . 42
4.3.4 ACK loss detection . . . 42
4.3.5 Round-Trip Time computation . . . 43
4.4 Reducing the TCP Acknowledgement ratio . . . 44
4.5 Performance evaluation . . . 45
4.5.1 Simulation environment . . . 46
4.5.2 Energy Model . . . 47
4.5.3 Comparison between NewDTC, TCP and DTC . . . 47
4.5.3.1 Number of hops . . . 48
4.5.3.2 Bit Error Rate . . . 49
4.5.4 Round Trip Time Computation . . . 49
4.5.5 Comparison between TCP, DCA, and TDA . . . 50
4.6 Conclusion . . . 51

5 TCP header compression for low power networks 53
5.1 Introduction . . . 54
5.2 Related work . . . 55
5.3 TCP Header Format . . . 56
5.4 TCP Header Compression . . . 58
5.4.1 Dynamic fields compression . . . 60
5.4.2 Context management . . . 63
5.4.3 Segment loss management . . . 63
5.4.4 LOWPAN_TCPHC Format . . . 64
5.4.4.1 TCP segments types . . . 64
5.4.4.2 LOWPAN_TCPHC Format . . . 65
5.4.5 TCP Option Compression . . . 66
5.4.5.1 SACK . . . 67

5.4.5.2 Timestamp . . . 67
5.4.6 Example of compressed TCP headers . . . 67
5.5 Experimental Setup . . . 68
5.5.1 Physical setup . . . 68
5.5.2 Hardware setup . . . 68
5.5.3 Software setup . . . 69
5.5.4 Energy consumption . . . 70
5.6 Results and Discussion . . . 71
5.6.1 TCPHC performance in loss-free environments . . . 71
5.6.2 TCPHC performance in lossy environments . . . 73
5.7 Conclusion . . . 73

III Impact of fragmentation on energy consumption 75

6 Impact of link layers fragmentation on the TCP energy consumption 77
6.1 Introduction . . . 78
6.2 Related Work . . . 79
6.3 TCP energy consumption model . . . 79
6.3.1 Link layer: one-hop model . . . 81
6.3.1.1 Link layer mechanisms . . . 81
6.3.1.2 Performance of one-hop transmissions . . . 81
6.3.2 Multi-hop model . . . 84
6.3.3 TCP performance . . . 86
6.4 Results and discussion . . . 88
6.4.1 Model assessment . . . 89
6.4.2 FEC redundancy ratio and energy consumption . . . 90
6.4.3 Selecting the TCP MSS to minimize energy consumption . . . 91
6.5 Conclusion . . . 92

7 Energy-efficient fragment recovery techniques for low power networks 95
7.1 Introduction . . . 96
7.2 The ARQ Error Control Mechanism . . . 96
7.3 Simple Fragment Forwarding and Recovery for 6LoWPANs . . . 97
7.3.1 Fragment Recovery . . . 98
7.3.2 An SFFR scenario . . . 98
7.4 Performance Evaluation and Discussion . . . 100
7.4.1 Impact of SFFR on TCP energy consumption . . . 100
7.4.2 Impact of SFFR on UDP energy consumption . . . 102
7.4.3 SFFR rounds *versus* energy efficiency . . . 104

7.4.4 When is it better to use SFFR? 104
7.5 Conclusion 105

8 Conclusion and Perspectives **107**

My Publications as a PhD Student **111**

List of Abbreviations and Acronyms **113**

Bibliography **115**

Index **122**

List of Figures

1 Les choix possibles d'un protocole de transport fiable au dessus des réseaux à faible consommation d'énergie vi
2 La gestion du segment caché avec DTC et NewDTC vii
3 TCP ACK loss recovery ix
4 Comparaison entre TCP, DTC et NewDTC en terme de consommation d'energie et temps de transmission x
5 Comparaison entre TCP, TCP-TDA et TCP-DCA en terme d'énergie consommée x
6 La récupération des acquittements TCP perdus xi
7 La compression d'en-tête TCP dans les réseaux 6LoWPANs xiii
8 Les résultats expérimentaux de TCP et TCPHC dans les réseaux 6LoWPANs. xiv
9 La consommation d'énergie avec des longs et des courts segments TCP en fonction du taux d'erreurs). xv
10 L'énergie consommée par l'ensemble des noeuds sans fils avec et sans SFFR (scenario avec 5 sauts). xvi

1.1 Choices of possible reliable transport protocols over low power networks . . 2

2.1 Overview of a Wireless Sensor Network 11
2.2 The 6LoWPAN architecture 13
2.3 LOWPAN_IPHC Header 14

3.1 ESRT: five characteristics on the normalized event reliability versus reporting frequency 23
3.2 RMST: a hop-by-hop recovery scenario 26

4.1 Hidden and exposed terminals problems in CSMA-CA networks 41
4.2 DTC and NewDTC cache management 42
4.3 Disabling unnecessary retransmissions 43
4.4 TCP ACK loss recovery 44
4.5 RTT computation: each node measures the RTT between itself and the receiver 44
4.6 TCP delayed acknowledgment recovery 46
4.7 Chain Topology 47
4.8 Comparison TCP, DTC, NewDTC in terms of consumed energy and transfer duration 48
4.9 Comparison of TCP, DTC and NewDTC with different bit error rates (number of hops=6) 49

4.10 Smoothed RTT computing improves TCP performance 50
4.11 Comparison between TCP, TCP-TDA and TCP-DCA in terms of Goodput 50
4.12 Comparison between TCP, TCP-TDA and TCP-DCA in terms of consumed energy 51

5.1 TCP header format 56
5.2 The IPv6 over Low power Wireless Personal Area Network Topology . . . 58
5.3 TCP connection initiation 59
5.4 Sequence number compression 61
5.5 Sequence number decompression 61
5.6 Comparison between TCP and CTCP in terms of sequence number compression 62
5.7 Different TCPHC packet format 64
5.8 TCP Header Encoding 65
5.9 TCP header option configuration 66
5.10 Compressed SACK option 67
5.11 Compressed TCP header encoding 67
5.12 The distribution of the wireless motes in the testbed 69
5.13 Crossbow Telos mote 70
5.14 Experimental results of multi-hop TCP vs. TCPHC over 6LoWPAN without concurrent CBR traffic. 72
5.15 Experimental results of multi-hop TCP vs. TCPHC over 6LoWPAN with a concurrent CBR traffic. 74

6.1 Failure and success scenarios for one link-layer transmission attempt. . . . 82
6.2 Failure and success scenarios in a multi-hop transmission. 85
6.3 Energy consumption with long or short TCP segments, as a function of the BER B (with $r = 3$). 88
6.4 Number of collisions in a multi-hop scenario. 90
6.5 Energy consumption with short or long TCP segments, as a function of the number of link layer attempts r (with $B = 5 \times 10^{-4}$). 91
6.6 Consumed energy using short or long TCP segment, as a function of the redundancy ratio α ($B = 3 \times 10^{-4}, h = 5$). 92
6.7 Long (MSS=512 bytes) versus short (MSS=64 bytes) in a multi-hop TCP transmission: prefer the short MSS above the curves, the long one below. ($\alpha = 0$) 93
6.8 Long (MSS=512 bytes) versus short (MSS=64 bytes) in a multi-hop TCP transmission: prefer the short MSS above the curves, the long one below. (with $r = 3$) 93

7.1 Recoverable Fragment Dispatch type and Header 97
7.2 Fragment Acknowledgement Dispatch type and Header 98
7.3 End-to-end simple fragment forwarding and recovery 99
7.4 **Analytical results:** Energy Consumption of a TCP data transfer with vs without SFFR (scenario with five hops). 101
7.5 **Simulation results:** Energy Consumption of a TCP data transfer with vs without SFFR (scenario with five hops). 102
7.6 Energy Consumption of a TCP data transfer with vs without SFFR (ARQ=3, $B = 5 \times 10^{-4}$). 103
7.7 Energy Efficiency of an UDP data transfer with vs without SFFR. 103
7.8 Energy Efficiency of an UDP data transfer with and without SFFR (ARQ=3, $B = 5 \times 10^{-4}$). 104
7.9 Energy Efficiency of an UDP data transfer with different SFFR rounds (ARQ=3, 5 hops). 105
7.10 SFFR in a multi-hop TCP transmission: prefer SFFR above the curves (ARQ=3). 106
7.11 SFFR in a multi-hop TCP transmission: prefer SFFR above the curves (MSS=1024). 106

List of Tables

2.1 IEEE 802.15.4 frequency bands and data rates 12

3.1 Classification of Reliable Transport Protocols 33

4.1 Simulation parameters 47
4.2 Energy model of wireless nodes 47

6.1 Notations used in this chapter; capital italics letters correspond to probabilities, bold letters to (expected) numbers of bits. 80
6.2 Default simulation parameters 89

7.1 Network parameters 100

Chapter 1

Introduction

1.1 Context

The success of wireless sensor networks (WSNs) is due to the small size of the wireless devices and the low power consumption of their wireless interface. The WSNs have been deployed in many applications fields: area monitoring (air pollution monitoring, forest fires detection, greenhouse monitoring), industrial monitoring (e.g., machine health monitoring), water/wastewater monitoring for agriculture, and so on. The wireless nodes in WSN are called *sensors* and the base station is called *sink*. Sensor nodes are small embedded devices, extremely basic in terms of their interfaces and their components. They usually consist of a processing unit with limited computational power and limited memory, sensors, a communication device, and a power source like battery. The base station is a gateway between the sensor nodes and the external network as it typically forwards data from the WSN to a server.

In this work, we do not limit our work to sensors (application layer), but we focus on general wireless multi-hop low power networks. These new networks are recently known as low power and lossy networks[1] or just **low power networks**. Nowadays, low power networks are becoming more and more popular and they are currently used in many industrial fields. Currently, most low power networks devices (e.g., sensors, actuators) use low-power wireless interfaces (e.g., Bluetooth, Low power IEEE 802.11 [IEE99], and especially IEEE 802.15.4 [IEE06]).

The IEEE 802.15.4 low power wireless personal area networks (LoWPAN) standard was released in 2003 by the Institute of Electrical and Electronics Engineers (IEEE). It provides the first global low-power standard. Afterwards, the ZigBee Alliance developed a solution for ad hoc control networks over IEEE 802.15.4. The ZigBee specification goes on to complete the standard by adding four main components: network layer, application layer, ZigBee device objects (ZDO's) and manufacturer-defined application objects. However, this solution has some problems with scalability, evolvability and Internet integration [Tit09, Max11].

In 2005, a new Internet Engineering Task Force (IETF) Working Group (WG) named 6LoWPAN started thinking about how to deploy and provide IPv6 addressing for all low power devices. After the introduction of IP, these devices are able to communicate not only between one another but also potentially with every IP device, inside and outside the

[1] See http://datatracker.ietf.org/wg/roll/charter

wireless network. These new devices, called IP Smart Objects[2] are changing the concept of the Internet. Internet becomes not only limited to a system of interconnected computer networks, but expands to an interconnection of all daily life objects leading to the concept of the *Internet of Things* (IoT) [KHS10]. The new IP systems will not be limited to PCs, laptops, routers, smart phones, but all wireless embedded devices that can be connected to Internet. We consider IoT as the third Internet revolution.

After the deployment of IP addresses everywhere, researchers in wireless networking started thinking about the upper layers: transport and application layers. It is true that UDP is useful for low power networks because many applications are fault-tolerant and do not require full reliability from the transport protocol. However, other applications and services (such as SSH and HTTP) are not fault-tolerant. These kinds of applications require a reliable service that UDP cannot provide. Moreover, some low power networks' application areas, such as health, military and security applications, impose strong reliability constraints. In some usage cases (e.g., sending a software update to a wireless node, or sending a query requesting specific information from the wireless node) there is a need for a reliable data transport. On the other hand, the deployment of TCP, which is currently the most used transport protocol in wireless IP-based networks, has many disadvantages such as the energy consumption.

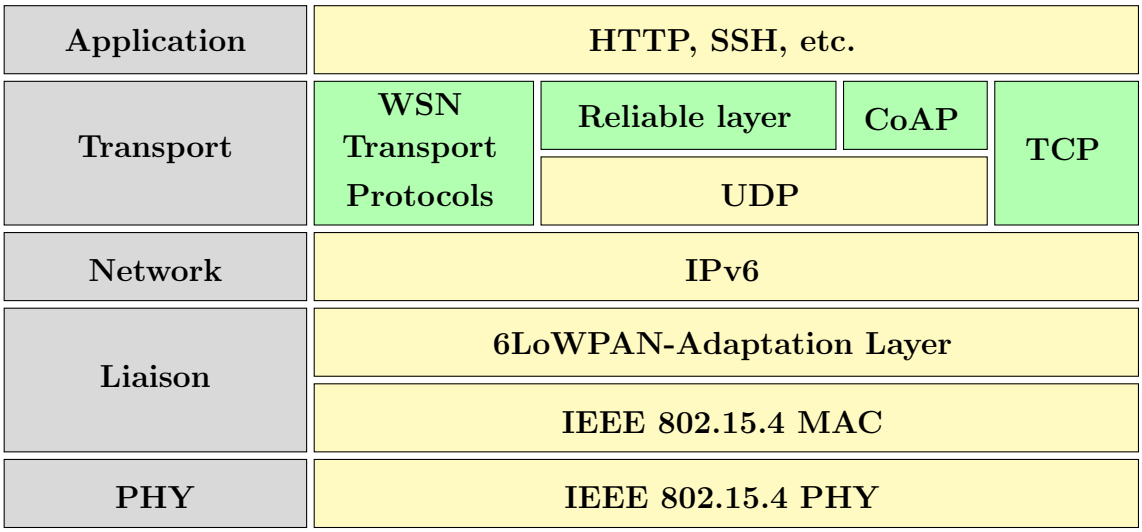


Figure 1.1: Choices of possible reliable transport protocols over low power networks

We can imagine four possible solutions to enable a reliable data transfer over low power networks (see Figure 1.1 for 6LoWPAN architecture). The first solution is to keep one of the recently proposed transport protocols for the WSN and to adapt it to be more general for all

[2]http://ipso-alliance.org/

kinds of applications. However, this solution will face the same problem as ZigBee. In fact, it will not be easy to integrate it with Internet, because it would require the deployment of a proxy between the wireless and the wired network. The proxy translates headers in the border router. A second solution is to provide a new transport layer over UDP. This solution should offer a reliable data transfer that is not offered by UDP, which consists of *data loss detection and retransmission*, and *congestion control and avoidance*. However, in order to design that protocol with all these requirements, the transport protocol header should include a sequence number, an acknowledgment number, and control flag and other fields that are already included in the TCP header. A third proposition, which is currently under development in the Constrained Restful Environments (CoRE) WG in the IETF, is to leave the congestion control and the loss recovery in the application layer. However, this solution provides a framework for a limited class of applications. At the moment of writing this thesis, the WG provides only a solution for HTTP over UDP. For all these reasons, in this thesis, we choose the fourth and last proposition, which consists of keeping TCP over the low power networks. The choice of TCP allows us to keep all mechanisms provided by TCP for loss recovery and congestion control. In this work, we distinguish most limitations of the TCP deployment over low power networks, and we propose a solution for each one.

1.2 Motivation and Objectives

In this thesis, we study the reliability and the energy-efficiency of data transmission in low power networks.

Most of these current propositions, which are proposed especially for wireless sensor networks, are not IP-based solutions (see Chapter 3). The non-TCP/IP reliable transport protocols need a complex algorithm on the Edge Router (ER), which is an IP router that interconnects the wireless networks to another IP network. Thus, the use of TCP reduces the complexity at the Edge Router. From the research perspective, investigating the use of TCP in new generation of wireless networks is of importance because the intersection of the TCP/IP protocol suite, the dominating communication protocol suite today, and low power wireless networks is a new area in computer networking research.

In this thesis, we focus on TCP over low power networks and especially over 6LoWPAN as an example of low power networks. The deployment of TCP allows current IP-based devices to communicate directly with LoWPAN devices using their TCP/IP stack. Deploying TCP, as it is currently implemented, inside the 6LoWPANs may not be the ”best” approach to solve the problem of reliability from 6LoWPANs to external-IP networks and vice versa. However, our improvements allow TCP to be more appropriated for this new context.

The objective of our thesis is to provide solutions for TCP over 6LoWPANs in order to

reduce the energy consumption and increase the throughput of TCP. The major problems of TCP over low power networks are the TCP header overhead, ratio of acknowledgment segments to data segments, end-to-end acknowledgments and retransmissions, and the TCP segment size.

1.3 Contributions

In this section, we list the contributions of our work. In addition, we show the main points that improve the energy efficiency of TCP in low power networks.

End-to-end acknowledgments and retransmissions

The end-to-end acknowledgment and retransmission schemes employed by TCP are not energy-efficient enough to be useful in low power networks. A single dropped packet requires an expensive retransmission from the original source. The low power wireless networks are often designed to be multi-hop, so, a single retransmission will incur transmission and reception costs at every hop through which the retransmitted packet will travel. The hop-by-hop retransmission reduces significantly the end-to-end retransmission by caching the not-yet acknowledged segments. In Chapter 4, we ameliorate a Distributed TCP Caching algorithm with more energy-efficient improvements.

TCP Acknowledgment ratio

TCP has serious performance problems in wireless networks in terms of energy efficiency. These problems are due to the high acknowledgment ratio (i.e. the number of TCP acknowledgments to the number of data segments). As described in [Pos81], the TCP receiver should acknowledge every data segment it receives. In fact, TCP acknowledgment segments represent half of all exchanged TCP segments, or about 33% if the TCP delayed acknowledgment mechanism is enabled. A delayed acknowledgment mechanism was proposed to halve the amount of TCP acknowledgment segments. So, the receiver should wait a short period before sending an acknowledgment. However, TCP delayed acknowledgments are not energy-efficient enough for low power networks because the ratio of acknowledgments to the data remains high. To be able to use TCP as a reliable transport protocol in 6LoWPANs, more energy efficient methods must be developed to decrease the TCP acknowledgment ratio and thus reduce significantly the consumed energy.

Header Overhead

TCP has a high overhead in terms of the protocol header size, particularly for small packets (such as IEEE 802.15.4). For small data packets, the TCP header overhead is over 70 % of

the link layer frame. Energy consumption is of prime importance in low power networks, thus transmission of unnecessary or redundant packet header fields should be avoided. In Chapter 5, we propose a new TCP header compression (TCPHC) that allows to reduce the TCP header down to six bytes. The TCP header compression algorithm for low power networks was proposed as an Internet draft for the 6LoWPAN WG in IETF. In this thesis, we present an experimental evaluation of TCPHC that was done in a testbed. Results show that TCPHC reduces the energy consumed due to TCP.

Adapting the TCP maximum segment size

In low power networks, link layer frames are generally small (e.g., 128 bytes in IEEE 802.15.4). Then, if the network layer does not provide fragmentation/reassembly mechanisms, TCP must send a short TCP segment that fits in a single frame. However, the use of a small TCP Maximum Segment Size (MSS) increases the number of TCP segments, and thus, the number of TCP acknowledgments and the header overhead. However, with the 6LoWPAN adaptation layer, TCP segments are split into fragments. The fragmentation reduces the TCP overhead. However, the loss of one of the link layer fragments leads to the loss of the original TCP segment, and thus to a new TCP retransmission. This fragmentation can make therefore imply a loss of energy efficiency.

In Chapter 6, we discuss the impact of link layer fragmentation on the TCP energy consumption by comparing the consumed energy by short versus long TCP segments. Then, we study a recently proposed fragment recovery algorithm for low power networks that allows to quickly recover the lost fragments in Chapter 7.

1.4 Outline

The manuscript is organized as follows:

In the first part, we present a background of related work on transport protocol reliability over multi-hop wireless networks. The first chapter gives some definitions and presents an overview of low-power multi-hop networks and introduces IPv6 over low-power networks. The second chapter is a survey of research works done on reliability, congestion control, and energy-efficiency in wireless sensor networks.

In the second part, we focus on how we can improve TCP performance and how we can make it more energy-efficient in low-power multi-hop wireless networks. The fourth chapter presents limitations of the deployment of standard TCP algorithms in 6LoWPANs. Thereafter, we present some improvements of TCP: we improve a distributed TCP caching algorithm and we present a better TCP delayed acknowledgment algorithm. In the fifth chapter, we introduce a new TCP header compression mechanism that allows us to reduce the TCP header overhead. We implement the proposed algorithm in the Contiki OS. From

the experimental results, we show that TCP with header compression reduces significantly the consumed energy.

In the third part, we study the impact of the link layer fragmentation on the energy consumption. The sixth chapter presents an analytical model that allows us to estimate the consumed energy. We study the tradeoffs involved in sending short versus long TCP segments. The seventh chapter study and evaluate a new fragment recovery algorithm that allows the link layer to recover quickly the lost fragments and to avoid a new TCP retransmission.

Finally, a concluding chapter summarizes the main achievements of the thesis and gives the perspectives of our work.

Part I

Background

Chapter 2

Low power multi-hop wireless networks

Contents

2.1 Introduction **10**

2.2 Low power networks **10**

2.3 Wireless Sensor Networks **10**

2.4 IPv6 over Low power Wireless Personal Area Networks . . . **12**

2.4.1 IEEE 802.15.4 12

2.4.2 6LoWPAN 13

2.4.3 Architecture 13

2.4.4 Header compression 14

2.4.5 Addressing 15

2.5 Conclusion **15**

2.1 Introduction

Low power Networks (such as Wireless Sensor Networks [YMG08]) are becoming more and more popular and they are currently being used in many industrial fields such as Remote Environment Monitoring (temperature and humidity) , Tracking Systems (especially for military purposes) [ARB+10], Health Monitoring [JEZ+05], etc. The evolution of the deployment of these networks is due to the main characteristics of their wireless devices: small size and weight, low power consumption and low cost.

In this chapter, we give an overview of low power networks and 6LoWPANs. Section 2.2 gives a definition of a low power network. Section 2.3 presents a detailed description of wireless sensor networks. Finally, Section 2.4 presents the IEEE 802.15.4 standard, then introduces the 6LoWPAN concept by describing its architecture.

2.2 Low power networks

Low power networks are made up of many wireless embedded devices with limited power, memory, and processing resources. They are interconnected by a variety of links, such as IEEE 802.15.4, Bluetooth, low power WiFi, wired or other low power Power Line Communication (PLC) links. Low power networks are transitioning to an end-to-end IP-based solution to avoid the problem of non-interoperable networks interconnected by protocol translation gateways and proxies.

Generally speaking, low power networks have at least four distinguishing characteristics:

- Low power networks operate with small, resource-constrained and highly portable operating systems.
- In most cases, low power nodes are energy-limited.
- Typical traffic patterns are not simply unicast flows (e.g. in some cases most if not all traffic can be multipoint to point).
- In most cases, low power networks will be employed over link layers with restricted frame-sizes, thus a routing protocol for low power networks should be specifically adapted for such link layers.

2.3 Wireless Sensor Networks

A Wireless Sensor Network (WSN) is a low power network that is composed of a large number of spatially distributed autonomous and embedded tiny sensor nodes, which consist

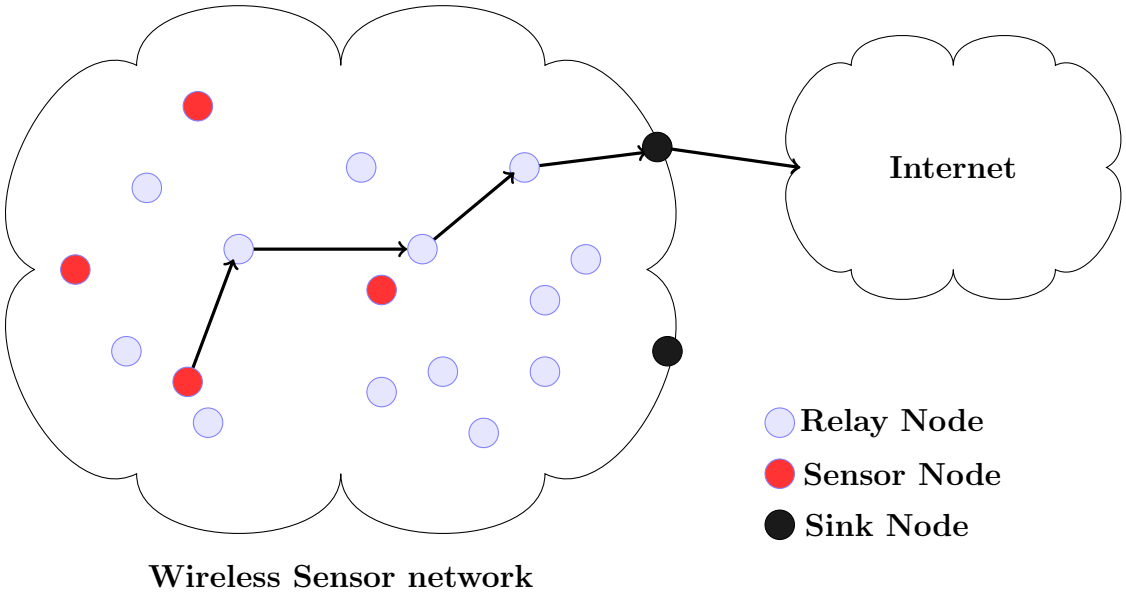


Figure 2.1: Overview of a Wireless Sensor Network

of sensing, data processing, and communicating components. The primary role of sensor nodes is to realize real-time monitoring, sensing and collecting information such as light intensity, temperature, humidity, noise and other physical phenomena. Then, sensor nodes process data and send it to sinks through wireless links. Motivated by military surveillance, WSNs are widely used in different areas like health, scientific research and security.

The data collected by each node (such as temperature, vibrations, sounds, movements etc.) are reported to a sink station in a hop-by-hop fashion using wireless transmissions. Intermediate nodes relay generated packets to a sink. Once received by the sink, such data can then be processed and analyzed for a better understanding of the monitored environment. Several successful deployments already denote the growing interest in this technology [BB08].

The small size and weight, the low cost of the hardware and the ease of deployment of such platforms enable the sensing of the environment in the least intrusive fashion. By spatially distributing tens or hundreds of such autonomous devices, a WSN can be built to cooperatively monitor physical or environmental conditions at different locations. The low power wireless transmitter usually mandates the collected data to be sent over multiple hops toward one or several sinks (Figure 2.1).

Initially, WSN applications did not require reliability because wireless sensor networks have been considered as fault-tolerant networks where sensor nodes collect environment information. However, new WSN applications like military applications (e.g., battlefield surveillance) require more and more reliability. Moreover, in [WCK05], the authors present

a need of re-tasking sensor nodes, and thus a need to send a binary file or a script file to sensor nodes. In our work, we mainly focus on the reliability of data transfer.

2.4 IPv6 over Low power Wireless Personal Area Networks

2.4.1 IEEE 802.15.4

Low power wireless personal area networks consist of devices that conform to the IEEE 802.15.4 standard. The common characteristics of IEEE 802.15.4 devices are short range, low bit rate, low power and low cost. Many of the devices are also limited in their memory and energy availability.

The IEEE 802.15.4 [IEE06] standard specifies MAC sub-layer and physical layer for Low power Wireless Personal Area Network (LoWPAN). The IEEE 802.15.4 standard defines low power wireless embedded radio communications at 2.4 GHz, 915 MHz and 868 MHz. The 802.15.4 standard provides 20–250 kbit/s data rates depending on the frequency. Table 2.1 summarizes the IEEE 802.15.4 frequency bands, the modulations, the spreading formats and the data rates. Channel sharing is achieved using carrier sense multiple access (CSMA), and acknowledgments are provided for reliability.

PHY (Mhz)	Frequency band (Mhz)	Spreading parameters		Data parameters		
		Chip rate	Modulation	Bit rate	Symbol rate	Symbols
868/915	868-868.6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
868/915 (opt.)	868-868.6	400	ASK	250	12.5	20-bit PSSS
	902-928	1600	ASK	250	50	5-bit PSSS
868/915 (opt.)	868-868.6	400	O-BPSK	100	25	16-ary Orthogonal
	902-928	1000	O-BPSK	250	62.5	16-ary Orthogonal
2450	2400-2483.5	2000	O-BPSK	250	62.5	16-ary Orthogonal

Table 2.1: IEEE 802.15.4 frequency bands and data rates

The physical layer payload is up to 127 bytes, with 72–116 bytes of payload available after link-layer framing, addressing, and optional security. The MAC protocol can be run in two modes: beaconless mode and beacon-enabled mode. Beaconless mode uses pure CSMA-CA channel access and operates quite like IEEE 802.11 without channel reservations. Beacon-enabled mode uses a hybrid time division multiple access (TDMA) approach, with the possibility of reserving time-slots for critical data.

2.4.2 6LoWPAN

6LoWPAN [KMS07] is an acronym of IPv6 over Low Power Wireless Personal Area Networks, and the name of an IETF Working Group (WG). The 6LoWPAN WG introduces a new layer on the TCP/IP stack in order to transport the IPv6 packets over IEEE 802.15.4 links (Figure 2.2). The 6LoWPAN WG of IETF defines encapsulation and header compression mechanisms that allow IPv6 packets to be sent and received over LoWPANs. The 6LoWPAN layer is required in order to adapt the size of the IPv6 packet (1280 bytes) to the link layer maximum transmission unit (MTU), which is the size (in bytes) of the largest protocol data unit that the layer can pass onwards. As described in [HT10], the new adaptation layer splits the IPv6 packet into small IEEE 802.15.4 fragments that should be sent to the receiver. In [SB09], the authors presented a straightforward technical definition of 6LoWPAN:

6LoWPAN standards enable the efficient use of IPv6 over low power, low-rate wireless networks on simple embedded devices through an adaptation layer and the optimization of related protocols.

2.4.3 Architecture

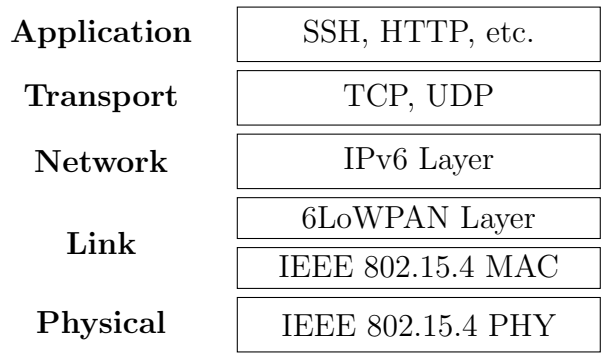


Figure 2.2: The 6LoWPAN architecture

Figure 2.2 shows the IPv6 protocol stack with 6LoWPAN. As mentioned in Section 2.4.2, the 6LoWPAN layer is between the IPv6 and the IEEE 802.15.4 MAC layer. 6LoWPAN supports only IPv6, for which a small adaptation layer (called the LoWPAN adaptation layer) has been defined to optimize IPv6 over IEEE 802.15.4 and similar link layers in [MKHC07].

Adaptation between the full IPv6 and the LoWPAN format is performed by routers at the edge of 6LoWPAN islands, called Edge Routers (ER). This transformation is transpar-

ent and stateless in both directions. LoWPAN adaptation in an edge router is typically performed as part of the 6LoWPAN network interface driver.

2.4.4 Header compression

The 6LoWPAN WG proposed LOWPAN_IPHC (IPHC) [MKHC07], a header compression mechanism for IPv6, to solve the problem of its big header (40 bytes) leaving little space for application data. With this mechanism, the 40 bytes can be often compressed to 3-5 bytes.

As described in [HT11], the algorithm is based on certain assumptions: IP version is 6, traffic class and flow label are both zero, the payload length can be inferred from lower layers from either the 6LoWPAN fragmentation header or the IEEE 802.15.4 header; the hop limit will be set to a well-known value by the source. These IPv6 fields are elided (e.g., the version field is always elided) or compressed (e.g., next header filed is one full byte, but has a number of values) or sent in-line (e.g., the hop limit was considered difficult to compress).

The LOWPAN_IPHC encoding utilizes 13 bits, 5 of which as dispatch of type. The encoding may be extended by another byte to support additional contexts (e.g., fragments recovery). Any information from the uncompressed IPv6 header fields carried in-line follow the LOWPAN_IPHC encoding, as shown in Figure 2.3. A detailed description of LOWPAN_IPHC can be found in [HT11].

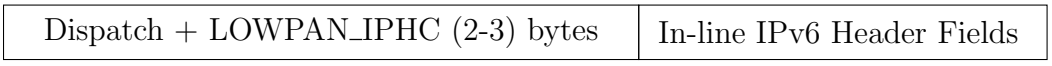


Figure 2.3: LOWPAN_IPHC Header

Currently, the User Datagram Protocol (UDP) is the most common transport protocol used with 6LoWPAN. In [MKHC07], there is only a UDP datagram compression method introduced (LOWPAN_NHC). Inside the LoWPAN, hosts and routers do not actually need to work with full IPv6 or UDP header formats at any point as all compressed fields are implicitly known by each node.

The transmission control protocol (TCP) is not commonly used with 6LoWPAN for performance, efficiency and complexity reasons (we focus on adapting TCP for 6LoWPAN in Chapter 4). The Internet control message protocol v6 (ICMPv6) is used for control messaging, for example ICMP echo, ICMP destination unreachable and Neighbor Discovery messages. In [O'F10], O'Flynn proposes LOWPAN_ICMPHC in a recent Internet draft, a header compression algorithm of ICMP messages. In [Bor11], Bormann proposes an Internet draft that provides a complete design for a simple addition to 6LoWPAN Header Compression that enables the compression of generic headers and header-like payloads.

Applications in 6LoWPANs are often specific and in binary format, although more standard application protocols are becoming available. Currently, the IETF CoRE WG is working on resource-constrained applications intended to run on IP-based low power networks. The Constrained Application Protocol (CoAP) functionality must operate well over UDP and UDP must be carried on wireless devices. There may be optional functions in CoAP that may be implemented over TCP [SSS+11].

2.4.5 Addressing

Addressing is required to differentiate between the nodes on a network. An IP adaptation layer involves at least two kinds of address: link-layer addresses and IP addresses. Low power wireless radio links typically make the use of flat link-layer addressing for all devices, and support both unique long addresses and configurable short addresses. IEEE 802.15.4 devices may use either of IEEE 64 bit extended addresses or, 16 bit addresses.

The IP addressing in 6LoWPAN works just like in any IPv6 network. IPv6 addresses are typically formed from the prefix of the LoWPAN and the link-layer address of the wireless interfaces. The difference in a LoWPAN is in the manner low power wireless technologies support link-layer addressing: a direct mapping between the link-layer address and the IPv6 address is used to reduce the address field size.

2.5 Conclusion

In this chapter, we gave an overview of LoWPAN. First, the low power network was introduced, followed by the definition of wireless sensor networks. Moreover, we showed that 6LoWPAN make viable to connect the low power wireless networks to Internet. In the next chapter, we focus on reliability and energy efficiency in low power networks and especially on wireless sensor networks. The next chapter compares the proposed reliable transport protocols for wireless sensor networks in terms of reliability, loss recovery, congestion control and energy efficiency.

Chapter 3

Reliable transport protocols over low power networks

Contents

3.1 Introduction 18

3.2 Transport protocols reliability for low power networks 18

3.2.1 Congestion Control 19

3.2.2 Reliability 24

3.2.3 Energy Efficiency 28

3.3 TCP/IP solutions for low power networks 29

3.3.1 TCP caching and hop-by-hop recovery 29

3.3.2 TCP dynamic delayed-acknowledgement 30

3.3.3 Constrained Application Protocol (CoAP) 31

3.4 Summary 32

3.5 Conclusion 34

3.1 Introduction

The need for a transport layer for low power networks to handle congestion and packet loss recovery has been recently debated. Researchers are working on the idea of a cheap, easily deployable network runs contrary to the costly and specialized transport layer for low power wireless networks. In this chapter, we focus especially on reliability in wireless sensor networks because most of the proposed works in the last years focused in sensing/monitoring applications. A transport protocol for low power networks should be reliable (or provide different levels of reliability for each kind of applications) and energy-efficient (reduce the amount of exchanged messages to reduce total consumed energy and thus increase the network lifetime). The reliability requires two essential mechanisms: **congestion control** (detection and avoidance of congestion), and **loss detection and recovery**.

This chapter presents a survey of reliable transport protocols for low power networks. The next section presents the needs of an energy-efficient and reliable transport protocols for wireless sensor networks and the recent proposed works in the literature. In Section 3.3, we present the related work on energy-efficiency of TCP over low power networks. Section 3.4 gives a summary of presented protocols in term of reliability, congestion control and energy efficiency.

3.2 Transport protocols reliability for low power networks

The transport protocol runs over the network layer, which generally provides a best-effort service. It uses end-to-end message transmission, where messages may be fragmented into several segments at the transmitter and reassembled at the receiver. The transport protocol can provide an unreliable service and datagrams may arrive out-of-order, appear duplicated, or go missing without notice. In this case, the transport protocol can assume that the error checking and correction are not necessary or are performed in the application. This allows to avoid the overhead of such processing at the transport level. On the other hand, the transport protocol can provide the following functions: orderly transmission, flow and congestion control, loss recovery, and possibly QoS guarantees such as timing and fairness. In this case, it is called a reliable transport protocol.

A reliable transport layer ensures the reliability at the receiver. Transport protocols in WSNs should support multiple applications and provide variable reliability levels, packet loss recovery and congestion control mechanisms. We can distinguish between three types of data in WSN:

- Data sent by sensor nodes to the sink (multipoint-to-point),

- Data sent by the sink to sensor nodes (point-to-multipoint),

- Data sent by the sink to a sensor node for different purposes (control, management, re-tasking, reprogramming) and data sent by a sensor node to the sink (point-to-point).

The development of a transport protocol should be generic and independent of the network layer. It may provide various reliability levels for different applications. WSNs like all low power networks suffer from a high loss rate. Packet loss may be due to bad radio communication, congestion, packet collision, full memory capacity, and node mobility or failure. Thus, a reliable transport protocol should provide two main functions: reliable data transport and congestion control.

A reliable application requires that all packets sent by a source arrive to the destination. Missing messages that may be lost in the wireless network should be recovered by reliable schemes. Congestion happens when the data packets generated by wireless nodes exceed the network capacity. When the network gets congested, intermediate nodes may drop packets. This leads to retransmissions of the dropped packets and thus a waste of energy, which is an important factor in wireless networks.

3.2.1 Congestion Control

The congestion control mechanism is an essential component for a reliable transport protocol even if some transport protocols (like PSFQ [WCK05], DTSN [MGN07], ERTP [TWPS09]) make an assumption that congestion is not likely to be a problem for wireless sensor networks. Others assume that all packet losses are due to congestion (such as ESRT [SAA03]).

Congestion occurs when a link or node is carrying more data than its capacity. Congestion leads to packet losses, and thus it has a significant impact on the performance of a reliable transport protocol.There are mainly two causes for congestion in WSNs. The first is due to the packet-arrival rate exceeding the packet-service rate. This is more likely to occur at the sensor nodes close to the sink, as they usually carry more combined upstream traffic. The second cause is the link-level performance aspects such as contention. It has some bad effects such as queuing delay and packet loss. Losses in wireless sensor networks are not only due to congestion but also channel conditions, collisions and interference. To distinguish between the two types of losses, an Explicit Congestion Notification (ECN) [RFB01] is used.

Congestion has a bad effect not only on the energy consumption but also on the network reliability. In fact, congestion in a relay node can lead to buffer overflow and thus larger queuing delay. This leads to packet losses and new retransmissions for the transport protocols. For these reasons, congestion in wireless networks must be well controlled and

avoided. In practice, in order to solve the congestion problem, three mechanisms should be provided: congestion detection, congestion notification, and congestion avoidance.

3.2.1.1 Congestion Detection

Congestion detection is the main component for congestion control. The congestion detection can be carried out in a distributed form (in wireless nodes) or in a centralized form (in the base station). In the distributed form, all sensor nodes detect congestion and then share the information. On the other hand, centralized solutions implement congestion detection on the base station.

A common mechanism would be to use **queue length** (e.g., Event to Sink Reliable Transport (ESRT) [SAA03]), **time to recover loss** (e.g., Rate-Controlled Reliable Transport protocol (RCRT) [PG07]), or the ratio of **packet service time** to **packet interarrival time** at the intermediate nodes.

Most protocols are based on buffer overflows to signal congestion using a congestion bit for the notification. For example, STCP [IGV05] specifies two thresholds: t_{lower} and t_{higher}. When the buffer reaches t_{lower}, the congestion bit is set with certain probability. When the buffer reaches t_{higher}, the node sets the congestion notification bit in every packet it forwards.

ESRT [SAA03] is one of the first transport protocols that seeks to achieve reliable event delivery with minimum energy expenditure. An ESRT node calculates $\Delta b = b_k - b_{k-1}$ after receiving a new packet, where b_k is the buffer size at the time k. If $b_k + \Delta b > B$ (where B is the buffer size) then the node signals a congestion by setting the congestion bit.

Interference-aware Fair Rate Control (IFRC) [RGGP06] is a shared congestion control algorithm. IFRC uses an exponentially weighted moving average of instantaneous queue length as a measure of congestion $avq_{t+1} = (1 - w) \times avg_t + w \times inst$, where $inst$ is the new queue length, avg is the average queue length, and w is a weight. The average queue length is updated whenever a packet is inserted into the queue. IFRC detects incipient congestion by using multiple buffer thresholds $U(k)$ where $U(k) = U(k-1) + I/2^{k-1}$ where k is a small integer and I is a constant increment of the queue length. Unlike STCP, IFRC uses multiple thresholds, thus, the rate halving becomes more frequent as the queue size increases. In this manner, a node continues to aggressively cut its rate until its queue starts to drain.

Rate-Controlled Reliable Transport protocol (RCRT) [PG07] is a multipoint-to-point reliable transport protocol for wireless sensor networks. RCRT uses an explicit end-to-end loss recovery and places all congestion detection, recovery and rate adaptation schemes in the base station. To distinguish between congestion and transmission losses, the RCRT congestion detection mechanism is based on the length of the losses (i.e., the number of lost packets). The sink node maintains a list of the out-of-order messages and computes the *Time to recover loss*. If this value exceeds 2× RTT (Round-Trip Time), a congestion is

then signaled. It assumes that the network is uncongested as long as end-to-end losses are recovered **quickly enough**. Thus, RCRT permits the sender to transmit at a light rate even if there are occasional end-to-end losses, since this rate can be recovered *quickly*.

For CSMA-like Medium Access Control (MAC) protocols, channel load can be measured and used as an indication of congestion. CODA [WEC03] is based on congestion detection by monitoring channel utilization and buffer occupancy at the receiver.

3.2.1.2 Congestion Notification

After detecting congestion, transport protocols need to propagate congestion information from the congested node to the upstream nodes or to the source nodes that contribute to congestion. In order to inform the source node about the congestion, two approaches can be used.

The first approach is by using a single binary bit in the data message (generally called **congestion notification** (CN) bit) (e.g., [SAA03]). This bit is set by a wireless node when its local buffer reaches a threshold to notify the destination of congestion. Then, the destination echoes the congestion indication to the source node by sending a control packet. Thereby, after receiving a control packet with CN bit set, the source node learns that the network is congested.

Another approach is to use into Explicit Congestion Notification (ECN) to disseminating congestion information. The explicit congestion notification uses control messages to notify the involved sensor nodes of congestion.

3.2.1.3 Congestion Avoidance

Generally, the prevention of network congestion and collapse requires two main components:

The first mechanism should be implemented in routers (in our case relay nodes) to drop packets under overload (e.g., **random early detection** (RED)). The second is end-to-end flow control mechanisms designed into the end-points, which respond to congestion and behave appropriately.

Then, upon receiving a congestion indication, a source node can adjust its transmission rate. If a single CN bit is used, **additive increase multiplicative decrease** (AIMD) schemes or their variants are usually applied, such as ESRT [SAA03].

In CODA, nodes receiving back pressure signals throttle down their transmission rates. In addition, a closed-loop mechanism operates on a longer time-scale. Based on acknowledgments received from the sink, sources regulate themselves. Lost acknowledgments result in reducing the rate at source.

Flush [KFD+07] is a reliable bulk transport protocol designed for WSNs. Flush supports only one data flow. Flush proposes also a rate allocation scheme for adapting dynam-

ically the sending rate of the sensor nodes. This scheme takes into acccount the broadcast nature of the medium and the interference between nodes. The rate allocation algorithm follows two basic rules:

- Rule 1: A node should only transmit when its downstream node is free from interference.
- Rule 2: A node's sending rate cannot exceed the sending rate of its successor.

These two rules reduce contention and thus collisions in the wireless network and minimize losses due to the queue overflows for all nodes.

In ESRT [SAA03], sensor nodes send event messages with an announced reporting frequency to the base station. Figure 3.1 shows the behavior of normalized reliability based on the frequency of reporting. This figure is obtained by tuning the reporting frequency from 10^{-1} to 10^{3} messages/second and computing the corresponding normalized reliability (i.e., the ratio of the observed event reliability to the desired event reliability).

Figure 3.1 shows that the event message reliability increases with the reporting frequency of sensor nodes. It is obvious that the more frequently sensor nodes send event messages, the more events arrive to the sink and thereafter more reliable are these events. However, after a certain reporting frequency F_{max}, the wireless network becomes congested. Therefore, the observed event reliability decreases while increasing the reporting frequency. Five characteristic regions are identified as follows:

- No Congestion, Low Reliability (NC, LR),
- No Congestion, High Reliability (NC, HR),
- Congestion, High Reliability (CHR),
- Congestion, Low Reliability (CLR),
- Optimal Operating Region (OOR).

An ESRT base station should regulate the reporting rate of sensors in response to a congestion detected in the network and try to maintain the reporting frequency in the OOR region. Congestion control mechanism is carried out in the base station, which informs all sensor nodes using a different wireless technology (e.g. IEEE 802.16) about the new reporting frequency. The use of a different MAC layer such as IEEE 802.16 allows the sink node to reach out all sensors, and does not interfere with the wireless network.

In IFRC [RGGP06], each node i includes the following information in the header of each outgoing transmission packet: its rate r_i, current average queue length, a bit indicating whether any child of i is congested, the smallest rate r_l among all its congested children, and l's average queue length. This information allow a wireless node to share it state to its parents. IFRC introduces two simple constraints:

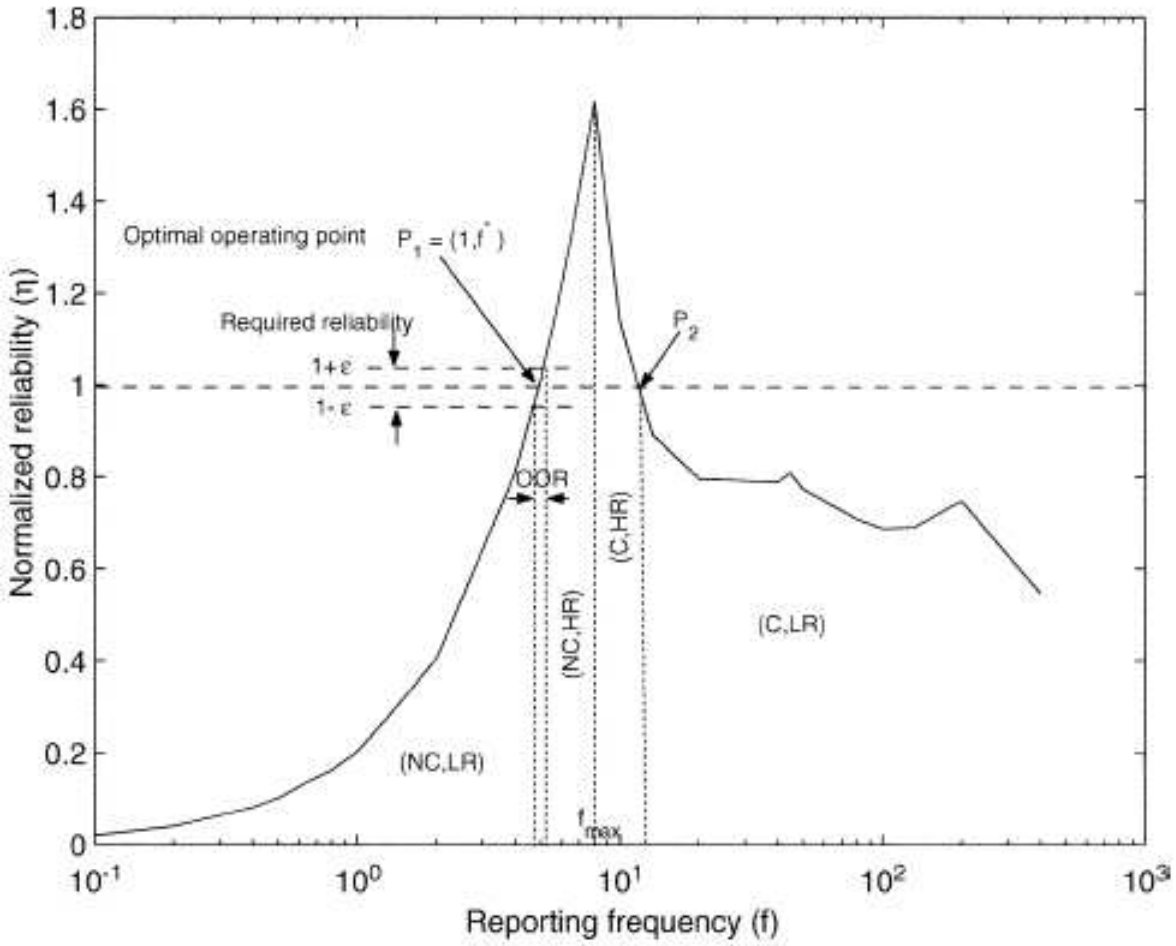


Figure 3.1: ESRT: five characteristics on the normalized event reliability versus reporting frequency

- Rule 1: r_i can never exceed r_j, the rate of i's parent j.
- Rule 2: Whenever a congested neighbor j of i crosses a buffer threshold $U(k)$, i sets its rate to the minimum of r_i and r_j. The same rule is applied for the most congested child l of any neighbor of i, i sets its rate to the minimum of r_i and r_l where l is the most congested child of i's neighbor.

All nodes start from a fixed rate r_{init}. IFRC implements multiplicative rate increase initially. After a **slow start** phase, an IFRC node increases its rate r_i every $1/r_i$ seconds. If node i is congested, then when threshold $U(k)$ is crossed, the node halves its current rate. The base station, even if it does not send messages, maintains its "rate" r_b and adapts using the same mechanism described below. Because the base station does not send messages,

it broadcasts a control message after the reception of five messages to share its rate r_b. IFRC presents a shared congestion control mechanism but it does not provide reliability guarantees. Moreover, IFRC adds an overhead in the header of the transport protocol and a significant amount of control messages.

RCRT [PG07] uses additive increase/multiplicative-decrease (AIMD [APS99]) algorithm to adapt the transmission rate of each source. Whenever the RCRT sink determines the network is congested, it applies the rate decrease and it computes the new rate for all flows.

- Increase : $R(t+1) = R(t) + A$
- Decrease : $R(t+1) = M(t) \times R(t)$

Where A is a constant and $M(t)$ is a function of loss rate, $M(t) = \frac{p_i(t)}{2 - p_i(t)}$ and $p_i(t)$ are the loss rate value of the source i at the instant t.

PORT [YLJH05] is another approach that minimizes the energy consumed by avoiding high communication costs with two schemes. The first scheme is based on the application-based optimization approach where the sink feedback the optimal reporting rates for source nodes. These source report feedbacks allow the sink to adjust the reporting rate of each data source. PORT adds a price for each node. A node price is the total number of transmission attempts made before a successful packet is delivered from the source to the sink. It is a metric used to evaluate the energy cost of communication. The sink adjusts the reporting rate of each source based on the source's node price and the information provided about the physical layer. The second scheme is based on feedback from the source node to the sink to inform it about the congestion and to increase the nodes costs. The sink uses the communication cost information to slow down the reporting rate of the appropriate source and to increase the reporting rate of other sources that have lower communication cost since reliability must be maintained.

3.2.2 Reliability

In the context of low power wireless network protocols, reliability properties specify the guarantees that the transport protocol provides with respect to the delivery of messages to the intended recipient. In fact, a reliable protocol should provide notifications to the sender as to the delivery of transmitted data.

The need for reliability is not the same for all applications but it depends on the importance of the application and even on the importance of certain packets. Some protocols such as PSFQ [WCK05], RMST [SH03], and RCRT [PG07] provide 100% reliability. On the other hand, ESRT [SAA03], ERTP [TWPS09], DTSN [MGN07] and STCP [IGV05]

provide classes of reliability for applications. Moreover, in the context of low power networks, the reliable transport protocol should be energy-efficient. Here we focus on the loss recovery that consists of **loss detection and notification** and **retransmissions**.

3.2.2.1 Loss detection

Loss detection methods differ from a protocol to another. However, the common mechanism is to include a sequence number in each packet header. The continuity of the sequence numbers can be used to detect packet loss. Then, the receiver uses gaps in the sequence numbers of received messages as a signal of packet losses.

There are two manners of loss detection: **end-to-end** and **hop-by-hop**. In the end-to-end approach, the receiver infers packet loss when it observes out-of-sequence arrivals. There are three ways for the notification: Acknowledgement (ACK), Negative ACK (NACK), Selective Acknowledgement (SACK), and Implicit ACK (IACK). The two first are explicit acknowledgments, which means that they consist of real control messages. However, IACK is a piggyback ACK, which means that if a message is overheard being forwarded again, this implies that the message has been successfully received. The use of IACK avoids control message overhead, so it is considered more energy-efficient.

However, in power-constrained networks, end-to-end recovery is considered not energy-efficient [PPG+07]. Therefore, most reliable transport protocols in WSNs use a hop-by-hop approach. In hop-by-hop loss detection, intermediate/neighbor nodes are responsible for loss detection and can enable local retransmissions.

3.2.2.2 Retransmissions

Like the loss detection, retransmissions of lost or damaged messages can be also either hop-by-hop or end-to-end. In the end-to-end approach, the source performs retransmissions. However, in hop-by-hop retransmissions, an intermediate node that receives a loss notification (e.g., NACK) searches the requested message in its local buffer. If it finds the lost message, it retransmits it, otherwise, it relays the NACK to the other nodes.

Reliable Multi-Segment Transport (RMST) [SH03] is the first transport layer with a hop-by-hop recovery scheme using caching mode. The main goal of RMST is to minimize the cost of end-to-end retransmissions. It offers two simple services: data segmentation/reassembly and guaranteed delivery. The RMST protocol provides two transmission modes: caching mode (with hop-by-hop recovery) and non-caching mode (with end-to-end recovery). In non-caching mode, only sources and sinks maintain a cache, and only sinks set timers to detect losses.

In caching mode, the RMST protocol assumes that each sensor node has a cache memory where recently received segments can be saved. RMST reduces end-to-end retransmissions

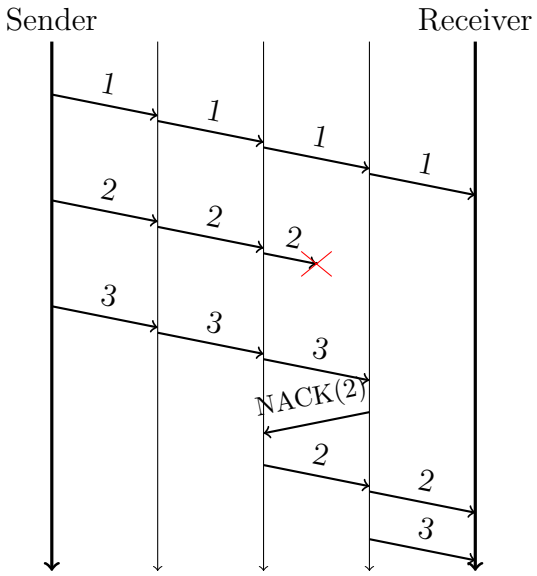


Figure 3.2: RMST: a hop-by-hop recovery scenario

by introducing hop-by-hop retransmissions from caches of neighbor nodes. In the link layer, lost packets are retransmitted using Automatic Repeat reQuest (ARQ) [FW02].

The RSMT receivers are the responsible for detecting losses and for trigging the recovery of the missing segments through the generation of Negative Acknowledgments (NACKs). The RSMT receivers are not only sinks, but they are also intermediate nodes. To handle losses, an RSMT intermediate node should store data traffic and construct a map of received segments. When an out-of-order segment is received, an RSMT receiver sends a NACK requesting retransmissions of lost messages. First, the one-hop neighbors process NACKs. Then, if one neighbor finds the missing segments in cache, it suppresses the NACK message and it retransmits the missing segments to the sink. Else, the NACK message is relayed to the next node toward the source. Figure 3.2 shows a detailed scenario of a RMST hop-by-hop recovery where the second message is lost in the third hop.

Pump Slowly, Fetch Quickly (PSFQ) [WCK05] mechanism is proposed for re-tasking/re-programming a group of sensors over-the-air. PSFQ is based on slowly injecting packets into the network **(pump operation)** and performing aggressive hop-by-hop recovery in case of packet losses **(fetch operation)**. Like RSMT, PSFQ provides a hop-by-hop error recovery mechanism in which intermediate nodes take the responsibility of loss detection and recovery. To enable a hop-by-hop loss recovery and in sequence data delivery, a data cache is created and maintained at intermediate nodes.

The PSFQ **"pump operation"** consists in a timely controlled data forwarding. In the intermediate nodes, when a packet is received in an out-of-order sequence, it is stored. However, instead of forwarding it, the intermediate node requests retransmissions of the

missing segment from its neighbors.

The PSFQ **"fetch operation"** is a proactive action of requesting a retransmission from neighboring nodes once the loss is detected at the receiving node. It corresponds to sending NACK for a retransmission request containing the sequence number of the missing segment. If the upstream neighbors do not posses the missing segment, they forward the NACK farther, until it reaches a node having the missing segments.

Hop-by-hop Reliability Support (HRS) [LKL06] is a hop-by-hop based reliable congestion control protocol. HRS proposes to use the packet sequence numbers to detect losses. This method speeds up the delay time in end-to-end transfer using a pair of an end-to-end sequence numbers and a hop-by-hop sequence numbers by which loss recovery is performed on missing hop-by-hop level. The one-hop sequence number allows a wireless node to detect a loss of a packet from its downstream node. Thus, with the hop-by-hop sequence numbers, an intermediate node can forward a received packet immediately to the next upstream node further and reduce the overall transfer delay by requesting the transmission only for the missing packet to the previous node. HRS uses a NACK-based approach and delayed ACK. A NACK message is sent if a gap is detected in the hop-by-hop sequence numbers. Delayed ACK is sent for the last packet after a short timeout.

Distributed Transport for Sensor Networks(DTSN) [MGN07] is an energy-efficient hop-by-hop reliable transport protocol using both ACK and NACK messages for delivery confirmation. A DTSN node analyzes the sequence numbers of the received packets and detects the losses by finding gaps.

Every source node sends an **Explicit Acknowledgment Request** (EAR) every one Acknowledgment Window (AW) to ask for an ACK or a NACK. The AW is the number of messages sent by the sender before requesting an acknowledgment. The sink node responds by an ACK message if no gap is detected or by a NACK message containing the sequence numbers of missing segments.

DTSN protocol is a hop-by-hop recovery protocol; all intermediate nodes cache received packets in their cache. Upon the reception of an ACK message, intermediate nodes delete acknowledged segments. Otherwise (i.e. reception of a NACK message), an intermediate node checks if its cache contains one of the missing segments.

A DTSN node retransmits missing segments and updates the NACK message. DTSN offers two types of service: total reliability service and differentiated reliability service. The difference between the two types of service is the probability of caching a segment in an intermediate node. For example, in full reliability scenario, all segments are cached in the intermediate nodes. DTSN is more energy efficient compared to PSFQ because it sends a ACK/NACK for AW messages. But, the DTSN algorithm does not treat congestion detection and control. Moreover, the algorithm does not tune the acknowledgment window size to reduce the ratio of acknowledgments to data.

Energy-efficient and Reliable Transport Protocol (ERTP) [TWPS09] is a hop-by-hop

recovery algorithm using implicit acknowledgments. ERTP requires that each node i after sending a packet to the next node to the sink overhears the next forwarding. The forwarding of a packet by node $i+1$ is considered as an implicit acknowledgment to node i.

The authors present a hop-by-hop reliability control, which adjusts the maximum number of retransmissions of a packet in each node based on the link loss rate. They present also an algorithm for computing the time in which node i is expected to "overhear" the forwarding packet of the node $i+1$.

In the results section, the authors show that the use of ERTP algorithm for computing the retransmission timeout (RTO) is better than Jacobson's algorithm. They show also that using ERTP gives a higher delivery ratio than using simple explicit acknowledgment. However, hearing all neighbor node traffics is not energy-efficient because listening consumes energy as well as sending.

3.2.3 Energy Efficiency

The cost of sensor components is a critical consideration in the design of sensor networks. It increases with the battery power of the devices. It is often economically advantageous to replace a sensor rather than to recharge it. By this reason, battery power is usually the important component in wireless devices. On the other hand, the lifetime of these devices depends on battery lifetime. Thus, energy efficiency is an important direction of low power networks investigations. As a result, it is important for the transport protocols to maintain high energy-efficiency in order to maximize system lifetime.

For loss-sensitive applications, packet losses lead to both retransmissions and the inevitable consumption of additional battery power. Therefore, several factors need to be carefully considered in the deployment of a transport protocol, including the number of packet retransmissions, the distance (e.g., number of hops) for each retransmission, and the overhead associated with control messages.

Transport protocols should provide reliability with the least number of exchanged messages. This constraint comes from the low capacity of energy of sensor node batteries. First, transport protocols should provide a mechanism to reduce the frequency of sending messages to reduce the total consumed energy for event-driven applications. Secondly, they should propose to use hop-by-hop recovery instead of end-to-end recovery to reduce retransmissions. Finally, the added control messages (e.g., ACK) must be used as rarely as possible to reduce their overhead.

The presented protocols have proposed various methods in order to reduce the energy consumption of the wireless nodes. For example, RMST, and PSFQ reduce the amount of exchanged messages by caching not already acknowledged segments in intermediate nodes and process a recovery once a loss is detected. DTSN proposes reducing the consumed energy by using **selective acknowledgment** (ACK and NACK) after an acknowledgment

window of messages, thus it reduces the control messages overhead. ERTP does not propose to use explicit acknowledgments but to use implicit acknowledgments. This approach needs a cross-layer mechanism between the link and the transport layers and permits to reduce the transport acknowledgments. All these works have tried to reduce the amount of control messages in the wireless sensor networks and thus increase the network lifetime.

3.3 TCP/IP solutions for low power networks

Transmission Control Protocol (TCP) is a reliable transport protocol that runs over IP networks, which provides end-to-end reliability and congestion control. However, in [MGN07, TWPS09, SH03], TCP was considered not well suitable for low power networks and especially wireless sensor networks. However, in order to make TCP viable for low power networks, several researchers are interested in reducing energy consumption of TCP and making it more energy-efficient [DVA04]. We present in this section related works to reducing loss recovery costs.

3.3.1 TCP caching and hop-by-hop recovery

In [DAV04], the authors present Distributed TCP Caching, a new scheme that uses segment caching and local retransmission in cooperation with the link layer for TCP/IP-based low power networks. DTC is an extension of the Snoop [BSAK95] idea towards multi-hop low power networks. The authors assume that each intermediate node is able to cache a single TCP data segment. DTC relies mainly on timeouts to detect packet losses. Thus, each DTC [DAV04] node measures the round-trip time (RTT) to the receiver and adapts a retransmission timeout RTO to 1.5 × RTT. The authors propose to compute the RTT in the TCP connection setup phase and to use RTO = 1.5 × RTT as a timeout value. In fact, the RTO may be too high which leads to lower throughput, or too low and then unnecessary end-to-end retransmissions may occur.

The wireless nodes cache the TCP segment that has the highest segment number seen with a probability of 50%. Authors justify the choice of this probability by a better distribution of cached segments than caching every new segment when the cache is not locked. An unacknowledged packet in the link layer should be locked and retransmitted after the timeout. Locked data segments should not be overwritten by a TCP segment with the higher sequence number. A locked segment is removed from the cache when a TCP ACK that acknowledges the cached segment is received, or when the segment times out.

DTC uses also the TCP SACK option [FMMP00, Mat96] for both packet loss detection on and signaling between DTC nodes. The TCP SACK option is used by wireless nodes to

inform other nodes about segments locked in their caches. To validate their schemes, the authors have implemented DTC on the OMNET++ simulator [Vag10]. Simulation results show that DTC enhances TCP performance for a chain topology and reduces the number of exchanged TCP segments.

In [BVD07], the authors present TCP Support for Sensor networks. TSS is a new layer between TCP and the network layer. TSS requires storing state information for each TCP connection that contains sequence numbers, acknowledgment numbers, and RTT. TSS uses Implicit ACK (IACK) for loss detection: a node is assumed to listen to packet forward of their neighbor to detect whether the next node has forwarded the TCP segment. A node using TSS always caches a packet until it is sure that the successor node towards the destination has received the segment. Retransmissions are mainly triggered by timeouts, which requires careful setting of timeout values. Like DTC [DAV04], the retransmission timeout is set to 1.5× RTT. To avoid congestion, a TSS node should stop forwarding its packets until it knows that all earlier packets have been received and forwarded by its successor node.

To validate their schemes, the authors used the OMNET++ simulator [Vag10]. The scenario used for their evaluation is very simple because it consists of a TCP source, a TCP sink and between them ten TSS wireless nodes. Simulation results show that TSS gives more throughput than TCP and less exchanged messages than DTC [DAV04]. However, hearing all neighbor nodes traffic is not energy-efficient because listening also consumes energy. In addition, a message transmission failure of one wireless node leads to stop the transmission of all its previous nodes.

3.3.2 TCP dynamic delayed-acknowledgement

Gerla *et al.* [GTB99] were the first to investigate the interaction between MAC layer and TCP in multi-hop wireless networks. Their study showed that TCP performance decreases significantly in CSMA networks when the distance between the sender and the receiver is larger than two hops. They explained this result by the hidden terminal problem caused by the collisions of TCP DATA segments and TCP ACK segments. In [FLZ+05], Fu *et al.* extended this work and showed that a good choice of a TCP window size depending on the number of hops between the source and receiver improves TCP throughput. Their simulations and analysis show that a TCP window of $\frac{h}{4}$ is the best choice in chain topology multi-hop wireless networks, where h is the number of hops between the source and the receiver.

Altman *et al.* [AJ03] studied the impact of delayed ACK on the TCP performance in multihop wireless networks. They showed that increasing the standard delayed ACK value ($d = 2$) up to $3 - 4$ packets increases the TCP throughput by around 50%. Moreover, they proposed a basic delayed ACK approach, which is called **Dynamic Delayed ACK**

(DDA). The TCP-DDA algorithm increases d gradually with the sequence number of the acknowledgment segment. The advertised window was limited to 4 packets.

De Oliveira and Braun [OB07] proposed a dynamic adaptive acknowledgement (DAA) strategy for minimizing the number of ACKs. The TCP-DAA algorithm proposes to adapt the TCP receiver window and the timeout interval. The TCP receiver computes a smoothed packet interval ($\overline{\delta}$) and then sets the timeout interval (t_i) as $t_i = (2+\kappa)*\overline{\delta}$, where κ is a timeout tolerance factor. κ defines how tolerant the receiver may be deferring its transmission beyond the second expected DATA packet. The TCP window size is initialized with one packet, and then increased by a startup speed factor ($\lambda = 0.3$ packet) until a certain threshold. After the startup phase, the TCP window size is increased by one packet size. TCP-DAA is more tolerant to packet delay variations by having the regular RTO (retransmission timeout) at the sender multiplied by a factor of 5. The authors simulation results show that TCP-DAA outperforms TCP-DDA in terms of goodput not only on chain topologies but also on grid topologies.

Chen *et al.* [CGLS08] proposed an adaptive delayed acknowledgment mechanisms called TCP-DCA (delayed Cumulative Ack). TCP-DCA removes the limit of delayed window of four packets and adapts the delay window limit depending of the path length (e.g., ping utility). The delay window limit is equal to the congestion window if the number of hops between the sender and the receiver is less than 3. Moreover, the delay window limit is equal to five if the number of hops is more than 3 and less than 9, and equal to three if the number of hops is more than nine hops. In addition, TCP-DCA algorithm puts the congestion window into the advertised window.

Chen *et al.* [CMSM09] proposed TCP-TDA, a new TCP algorithm that uses the ACK-delay timeout as a trigger for sending an acknowledgment. In fact, the TCP-TDA receiver always waits until this timeout event occurs to generate an ACK, no matter how many in-order packets it receives during the timeout period. TCP-TDA sets up a large delay window (equal to 25), thus it makes the ACK-delay timeout to be the main generator of ACKs. Moreover, the authors set the timeout time at 200 ms (default value is 100 ms). In addition, when the congestion window is small, TCP-TDA proposes to put the congestion windows in the advertised window, so it allows informing the receiver about the current value of the congestion window. The major difference between TCP-TDA and TCP-DCA is that it does not need to count the path length. Moreover, simulation results show that TCP-TDA improves TCP throughput up to 205% respect to standard TCP.

3.3.3 Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) [SHBF11] is a *simple*, *low-overhead* and *specialized for web* transfer protocol recently proposed by the CoRE WG in IETF. It was proposed to be used in resource-constrained IP networks and nodes for machine-to-machine

(M2M) applications such as smart energy and building automation. CoAP translates to HTTP for integration with the web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments. CoAP is a UDP-based protocol with optional reliability supporting unicast and multicast requests.

CoAP defines four types of messages: confirmable, non-confirmable, acknowledgement, and reset. As specified in [SHBF11], the CoAP reliability is provided by marking a message as confirmable (reliable message). A confirmable message is retransmitted using a default timeout and exponential back-off between retransmissions, until the receiver sends an ACK with the same message ID from the corresponding end-point. Instead, (i.e., the receiver is not able to process a confirmable message) the receiver replies with a reset message instead of an ACK. A reset message is sent if a confirmable message was received, but some context is missing. In [Cas11], Castellani proposes the Constrained Messaging Protocol (CMP), which is a message-layer protocol for CoAP. CMP adds to [SHBF11] some features such as delayed acknowledging, multi-datagram messaging, and partially ordered delivery of unconfirmed messages.

At the time of writing, only Eggert's work focused on congestion control for CoAP. In [Egg11], Eggert suggests a **simple windowing algorithm**; the CoAP stack should locally drop application-generated messages under overload situations. In fact, a CoAP node has a certain number of **message transmission credits** available during a time interval. Then if all message transmission credits have been sent, the CoAP stack should drop the application messages. After the arrival time, the CoAP node checks if all confirmable messages were acknowledged. If one of the confirmable messages is not acknowledged, the transmission credits are halved for the next time interval, else (i.e., acknowledgments have been received for all confirmable messages) the message transmission credits are increased by unit.

In [Egg11], Eggert discussed the use of Explicit Congestion Notification (ECN) [RFB01] for CoAP. The ECN bit can be used to decrease the message transmission credits if a CoAP node receives a message with an ECN bit.

3.4 Summary

Table 3.1 presents a summary of the listed protocols in Section 3.2. We can differentiate between transport protocols by the manner they recover losses (end to end recovery or hop by hop recovery), the use or not of caching in intermediate nodes, the kind of messages used for loss detection and recovery (ACK, NACK, IACK), and the level of reliability.

Transport Protocol	Direction	Type of flows	Congestion Control	Congestion Detection	End-to-End or Hop-by-Hop	Caching	ACK / NACK SACK / IACK
ESRT [SAA03]	Sensor to Sink	Continuous	Yes	Buffer size	End-to-End	No	-
PORT [YLJH05]	Sink to Sensor	Event-Driven	Yes	Packet Loss	-	-	
RSMT [SH03]	Sensor to Sink	Continuous	-	-	Hop-by-Hop End-to-End	Yes	NACK ACK
PSFQ [WCK05]	Sink to Sensor Sensor to Sink	Event-Driven	-	-	Hop-by-Hop	Yes	NACK
DTSN [MGN07]	Sensor to Sink	Continuous	-	-	Hop-by-Hop	Yes	ACK/NACK
HRS [LKL06]	Sensor to Sink	Continuous	-	-	Hop-by-Hop End-to-End	Yes	NACK ACK
CODA [WEC03]	Sensor to Sink	Event-Driven	Yes	Buffer size Channel load	End-to-End Hop-by-Hop	No	ACK
STCP [IGV05]	Sensor to Sink	Continuous Event-Driven	Yes	Buffer Size	End-to-Ends	Yes	ACK/NACK
Flush [KFD+07]	Sensors to Sink	Continuous	Rate control	No	End-to-End	No	NACK
IFRC [RGGP06]	Sensor to Sink	Continuous	Yes	Buffer size	-	No	-
RCRT [PG07]	Sensor to Sink	Continuous	Yes	Time to recover loss	End-to-End	No	NACK
ERTP [TWPS09]	Sensor to Sink	Continuous	-	-	Hop-by-Hop	-	IACK

Table 3.1: Classification of Reliable Transport Protocols

3.5 Conclusion

In this chapter, we presented a survey of recent works on reliability and congestion control in wireless sensor networks. We presented the different methods of loss detection and recovery, congestion detection and avoidance and energy-efficiency. We compared and classified in table 3.1 all these protocols in terms of reliability, congestion control and energy efficiency.

Part II

Reducing the energy consumption of TCP in multi-hop wireless networks

Chapter 4

Making TCP more energy-efficient for low power networks

Contents

4.1 Introduction 38

4.2 Why TCP for low power networks 39

4.3 Distributed TCP Caching 39

4.3.1 MAC Automatic Repeat reQuest 40

4.3.2 Cache management 41

4.3.3 Disabling unnecessary retransmissions 42

4.3.4 ACK loss detection 42

4.3.5 Round-Trip Time computation 43

4.4 Reducing the TCP Acknowledgement ratio 44

4.5 Performance evaluation 45

4.5.1 Simulation environment 46

4.5.2 Energy Model 47

4.5.3 Comparison between NewDTC, TCP and DTC 47

4.5.4 Round Trip Time Computation 49

4.5.5 Comparison between TCP, DCA, and TDA 50

4.6 Conclusion 51

4.1 Introduction

The deployment of IP on low power networks (e.g. 6LoWPANs) enables a direct interconnection of a low power network to external IP network without proxies or middleboxes. Such interconnection satisfies new application needs by allowing e.g. an external host on the Internet to communicate directly with a wireless device. In the scenario of controlling and managing sensors presented in the last chapter, this would likely mean using TCP as a reliable transport layer.

TCP uses end-to-end acknowledgements (ACKs) and retransmissions of lost packets to guarantee reliability. In multi-hop wireless IP networks, packet loss does not only happen because of congestion problem, but it may also be due to bad radio condition, node failure and frame collision due to simultaneous transmissions. Under such conditions, TCP performance may degrade, resulting in lower throughput and longer transfer times. In the context of low power networks, wireless losses and end-to-end retransmissions result in energy "wasted" due to packets being transmitted over one or more hops, only to be lost in a subsequent hop. The end-to-end retransmissions cost would be reduced by some hop-by-hop recovery and caching mechanisms.

Moreover, TCP suffers from its header overhead. The TCP header size is 20 bytes without options. The TCP header may be extended to 60 bytes by adding some options such as Selective Acknowledgment (SACK) [Mat96, FMMP00] and Timestamps [JBB92]. In addition, TCP requires that every TCP segment should be replied by a TCP acknowledgement from the receiver. Even with a delayed acknowledgment mechanism, the number of TCP acknowledgments is still high. The ratio of TCP acknowledgments to the TCP segments makes TCP not energy-efficient. One of the recently proposed ideas is to send a TCP acknowledgment for a block of TCP segments (see Section 3.3.2).

In this chapter, we show the reason of choosing TCP as a reliable transport protocol. We study Distributed TCP Caching (DTC) algorithm and we propose some simple modifications to improve the TCP energy-efficiency, which reduce both the consumed energy and the file transfer duration. Our modified algorithm is designed for CSMA-CA networks (e.g., IEEE 802.15.4 or IEEE 802.11) and it takes into account the presence of link-layer ARQ.

The remainder of the chapter is organized as follows. Section 4.2 presents our reasons for deploying TCP over low power networks. In Section 4.3, we study a hop-by-hop TCP recovery algorithm and present some improvements of DTC. In Section 4.4, we study how to improve the energy efficiency by reducing the TCP acknowledgment ratio. Section 4.5 presents performance evaluation, highlighting the improvement brought by our proposition, and the impact of different parameters.

4.2 Why TCP for low power networks

Several transport protocols have been proposed to ensure end-to-end reliable data transmission (see Chapter 3). To detect losses, these protocols have proposed to add a sequence number to identify all segments. An acknowledgment number is used to inform the source of the segments correctly received and the segments to be retransmitted. In addition, to manage congestion, other protocols have proposed a congestion notification bit to inform the source of congestion detected in the core network.

However, none of those protocols allows opening a connection between an IP node in an IP-based network and a wireless embedded node in the LoWPAN network. Indeed, to enable this communication, any new connection requires a proxy between the two networks to translate the transport protocol header of the wireless network to TCP and vice versa.

In this chapter, we propose to maintain TCP in the wireless network for several reasons. Currently, TCP is the most frequently used reliable transport protocol in the Internet. On the other hand, all the mechanisms, which have been proposed for wireless networks to ensure reliability, are already implemented in TCP (loss detection, retransmission, congestion control). And finally, the extension of TCP in wireless networks avoids redefining new specific applications for the wireless network. On the contrary, many applications and services would be quickly deployed for the wireless devices (HTTP, SSH, Telnet, etc.). However, TCP has to be improved and adapted to new constraints imposed by this new environment (CPU, memory, energy, bit rate, etc.).

4.3 Distributed TCP Caching

DTC is an extension of the Snoop [BSAK95] idea towards multihop wireless networks. The authors presume that each intermediate node has enough memory to cache a single TCP data segment. Segments that are not acknowledged at the link layer by the next hop are locked in the cache and retransmitted after a retransmission timeout (RTO). After caching a segment, DTC node unlock it if it receives a link layer acknowledgement. Locked data segments cannot be overwritten by other TCP segments. A locked segment is removed from the cache only when a TCP ACK or SACK [BAFW03] that acknowledges the cashed segment is received, or when the segment times out. DTC relies mainly on timeouts to detect packet losses. Thus, each node measures the round-trip time (RTT) to the receiver (sink) and adapts a retransmission timeout RTO to 1.5× RTT. The authors propose to compute the RTT in the TCP connection setup phase (they call this approach "flying start") and also after receiving a TCP ACK segment. Wireless nodes cache the TCP segment with the highest segment number seen if the cache is empty. Saving the segment with the highest segment number disadvantages old segments, which should be retransmitted. If the cache

is not empty and not locked, a DTC node caches the TCP segment with a probability of 50%.

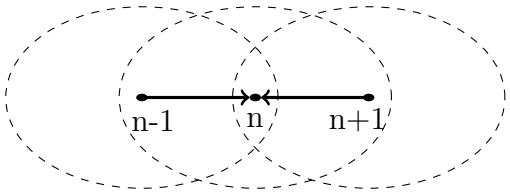
In the next section, we present our energy-efficient hop-by-hop distributed TCP caching algorithm, called NewDTC, which is based on DTC, with some improvements regarding the cache management. These improvements enable its deployment in CSMA-CA networks. DTC nodes should maintain a TCP state for each connection. Our DTC version is based on segment caching and local retransmission like the original DTC. However, an intermediate node saves a segment if the buffer cache is not locked. This reduces the congestion problem on intermediate nodes. We also suggest allowing intermediate nodes to generate an ACK message and sent it back to the sender if they receive a TCP data segment that was already acknowledged. The third point that we suggest is to adjust the timeout value that a node should wait before retransmitting a locked TCP data segment. A good estimation of round-trip time reduces the network congestion on intermediate nodes and avoids unnecessary retransmissions. We describe with more details our enhancements to the DTC algorithm in the following sections.

4.3.1 MAC Automatic Repeat reQuest

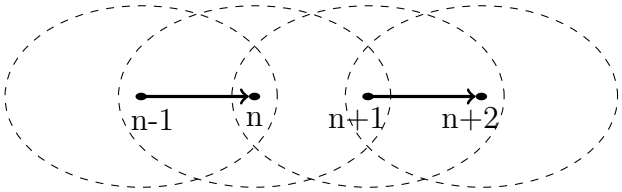
DTC was designed and validated for a TDMA network where losses are due to bit errors. However, in CSMA-CA networks, errors are not only due to bit errors but also to collisions between frames. Collisions in CSMA-CA networks are due to two problems: hidden and exposed terminals (see Figure 4.1). Using a single transmission in a high error link increases the RTT value, increases end-to-end retransmissions and reduces the connection throughput.

Unlike DTC, the NewDTC approach enables Automatic Repeat reQuest (ARQ) in MAC layer. This option already exists in CSMA-CA link layers like IEEE 802.15.4 and IEEE 802.11. Thereby, after receiving a TCP data segment, a wireless node sends the received message to its MAC layer buffer. The link layer uses ARQ to transmit TCP segments. If a wireless node does not receive an ACK, it retransmits the message. If the maximum retry number is reached, the MAC layer informs the DTC module about a link layer transmission failure. Then, The DTC module locks the received segment in its cache buffer and waits a retransmissions timeout $RTO = 1.5\times$ RTT before sending it again.

The use of the ARQ algorithm in the link layer allows wireless nodes to have a more reliable link layer. Furthermore, higher retransmissions are done in the link layer, lower hop-by-hop and end-to-end transport protocol retransmissions are needed.



(a) **Hidden terminal**: node n is visible for both node $n-1$ and node $n+1$. However, $n-1$ and node $n+1$ can not see each other. The collision happens at node n when nodes n and node $n+1$ send at the same time their frames to node n.



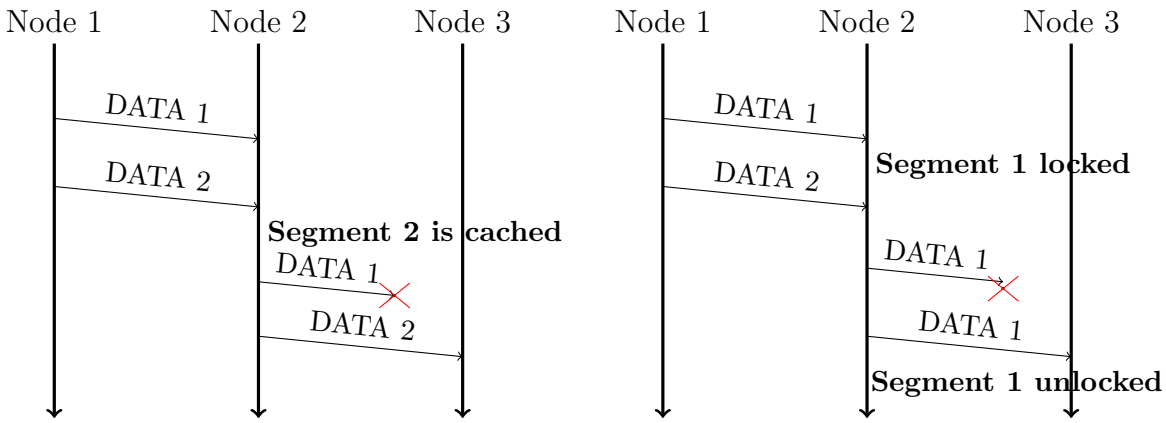
(b) **Exposed terminal:** node $n-1$ and node $n+1$ can not see each other. Node $n-1$ sends a frame to node n while node $n+1$ sends a frame to node $n+2$. The collision happens at node n because it hears both $n-1$ and $n+1$ frames.

Figure 4.1: Hidden and exposed terminals problems in CSMA-CA networks

4.3.2 Cache management

DTC proposes to cache a received segment if the cache buffer is empty. It also proposes to cache that segment with 50% probability if the cache is not empty and not locked. A DTC node caches the newest segments; this would deprive old segment, which should be retransmitted. Figure 4.2 (a) shows that the use of this approach leads to a segment loss if a wireless node receives a new TCP segment before sending the already cached segment. Node 2 does not receive a link layer acknowledgment and does not cache it. The segment 1 is overwritten by the segment 2 and then retransmitted from the source node.

We propose a new approach to cure this problem. We assume that wireless nodes are reliable. After receiving a TCP data segment, a NewDTC node locks the received segment and sends it to the link layer trying to send it to next hop node (see Figure 4.2 (b)). After receiving a link layer acknowledgment, the NewDTC node unlocks the received segment. The cached segment could then be overwritten by the next received TCP segment or deleted if the wireless node received a TCP ACK with an acknowledgement number greater than its sequence number or a TCP SACK containing a SACK block acquitting the cached segment.



(a) DTC: DATA 1 is lost and retransmitted from the source node

(b) NewDTC: DATA 1 is retransmitted from the node 2

Figure 4.2: DTC and NewDTC cache management

Our solution gives all priority to the old segment if it is not successfully sent at the link layer. On the other side (i.e., the source does not receive a link layer ack), NewDTC gives the priority to the new segment if the cached one is already acknowledged in the link layer (i.e., the wireless node is sure that its downstream neighbor has received the segment). This approach solves the problem of overwriting a cached segment before it could be sent to the next hop node and gives more chance to the old segment to be retransmitted.

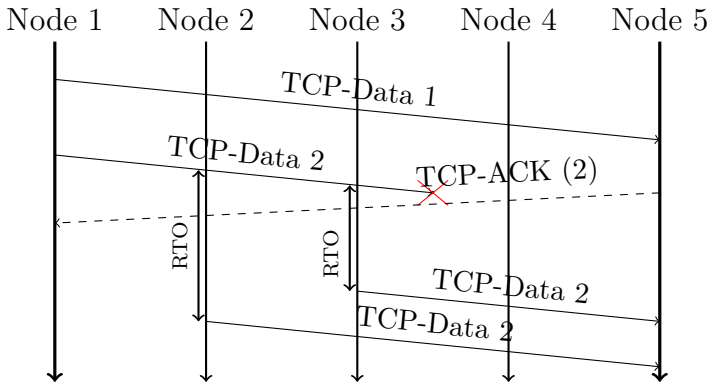
4.3.3 Disabling unnecessary retransmissions

DTC imposes that each intermediate node should retransmit the cached segment after an RTO. However, local retransmissions from wireless nodes can lead to unnecessary retransmissions if the same segment is retransmitted by more than one node (see Figure 4.3 (a)). To reduce the number of unnecessary retransmissions, a NewDTC node should not relay a TCP segment that just sent it. Moreover, a NewDTC node should update again the retransmission timeout if it receives a cached TCP data segment (see Figure 4.3 (b)).

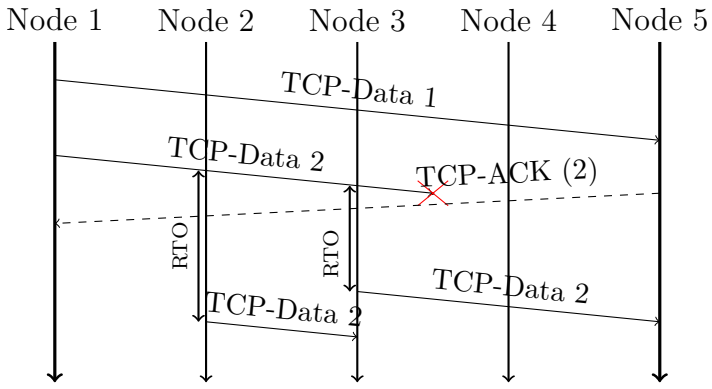
Further, when a wireless node receives a TCP data segment, it compares its sequence number to the cached one (if it exists). If they have the same sequence number, the wireless node should set the RTO value to 1.5× RTT. This approach has no impact on the retransmission of missing segments.

4.3.4 ACK loss detection

A NewDTC node detects the loss of a TCP acknowledgment by receiving a TCP data segment that has already been acknowledged by a TCP ACK. Thus, a NewDTC node



(a) DTC



(b) NewDTC

Figure 4.3: Disabling unnecessary retransmissions

deletes the received TCP data segment and replies by a new TCP ACK for the source. This requires that each node keeps connection state for all TCP connections that pass through the node by creating a context.

Figure 4.4 shows an example of a TCP ACK loss and a local TCP regeneration of an ACK after receiving an acknowledged data segment. Node 2 receives a TCP ACK but does not succeed in sending it to the downstream node. Then, it receives a TCP data segment already acknowledged. Node 2 deletes TCP segment number 2, regenerates a new TCP ACK. This approach avoids a new end-to-end retransmission from the sender to the receiver.

4.3.5 Round-Trip Time computation

DTC proposes to compute the RTT value after receiving a TCP ACK segment (see Figure 4.5), which acknowledges the cached segment. Then, the RTO value is chosen as

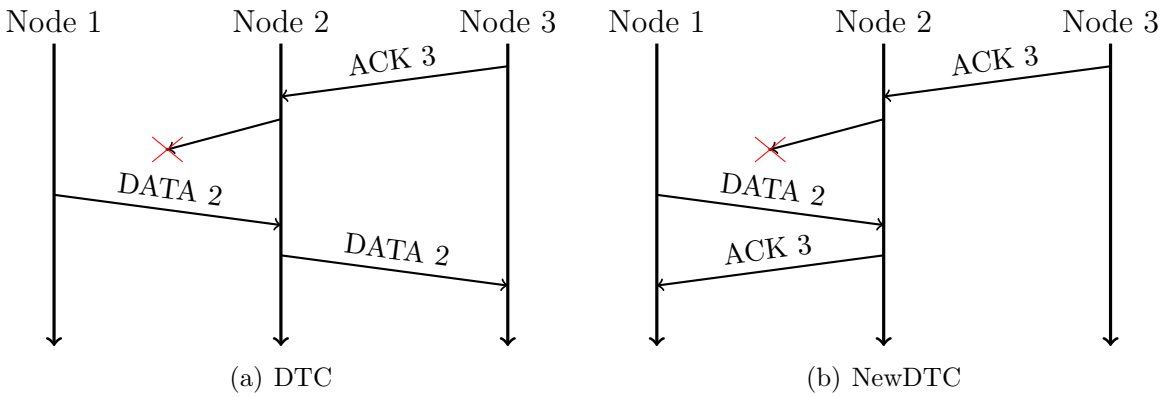


Figure 4.4: TCP ACK loss recovery

1.5 × RTT. The RTO computation is very important to reduce unnecessary end-to-end retransmissions and to get a better throughput.

In NewDTC, we conserve the flying start used in DTC, which consists of measuring RTT during the TCP connection set-up. However, we propose to smooth the measured RTT value as specified in [PA00] and to keep 1.5 × smoothed RTT as a retransmission timeout RTO.

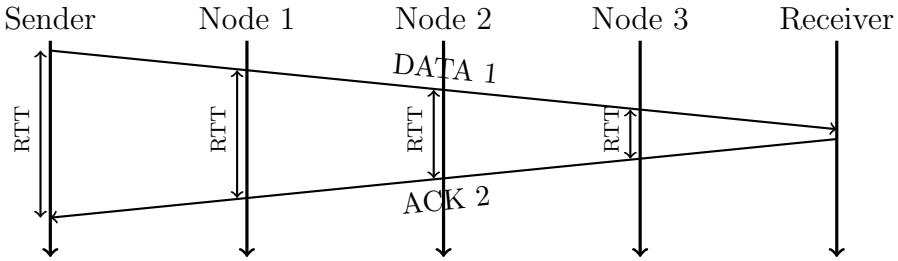


Figure 4.5: RTT computation: each node measures the RTT between itself and the receiver

4.4 Reducing the TCP Acknowledgement ratio

As it was well detailed in Section 3.3.2, recent researchers, that worked to improve the performance of TCP in multi-hop wireless networks, have proposed algorithms to reduce the number of TCP acknowledgments. The main idea of these algorithms is to delay the acknowledgment even after the reception of two segments.

The two more efficient delayed acknowledgment algorithms are TCP-TDA and TCP-DCA. TCP-TDA is simpler to implement (compared to TCP-DCA) and does not require

that the receiver computes the segment inter-arrival times (i.e., the time between consecutive segment arrival). Moreover, TCP-DCA requires that the sender knows the number of hops between itself and the receiver. For all these reasons, we focus more in this section on TCP-TDA.

The main drawback of TCP-TDA is that it requires the use of the advertised window, which should normally inform the sender about the receiver buffer capacity, to inform the receiver about the number of segments after which it must send an acknowledgement. The solution is possible for unidirectional data transfer (where one device can communicate with each other but only one direction at a time (i.e., not simultaneously)). These disable some of the most-used TCP-based applications such as SSH and HTTP. Instead of sending the size of the congestion window, we propose to use one of the reserved bits of the TCP header for requesting an acknowledgment. The idea was firstly proposed by A. Oppermann [1] and then discussed by [FARI10] to reduce the TCP acknowledgment congestion.

Reducing the TCP acknowledgement ratio would have a bad impact on TCP performance. The TCP congestion window increases by a constant amount for each arriving acknowledgment. The TCP delayed acknowledgement would reduce the TCP throughput by reducing the ACKs and then increase the transfer duration. In [All03], Allman proposes another congestion control mechanism for coping with delayed acknowledgments. The main idea was that the TCP congestion window should be increased based on the number of bytes acknowledged by the arriving ACKs. In order to improve delayed acknowledgment algorithms performance, we propose to apply the same idea, even if the number of delayed acknowledgments is higher than two segments. In addition, the TCP SACK option can be used to signal if one or more segments are lost by showing the received segments. Figure 4.6 shows an example of scenario where one of the sent segments is lost. The receiver responds by an ACK with a SACK option to inform the sender that the third segment was not received. We assume that the sender has a large window to send many segments before receiving an acknowledgment.

4.5 Performance evaluation

In this section, we presented a simulation evaluation of discussed ideas. In fact, we evaluate the performance of two TCP caching algorithms which are NewDTC and DTC. Then, we evaluate DCA and TDA and we compare their performance.

[1] http://www.ietf.org/mail-archive/web/tcpm/current/msg02356.html

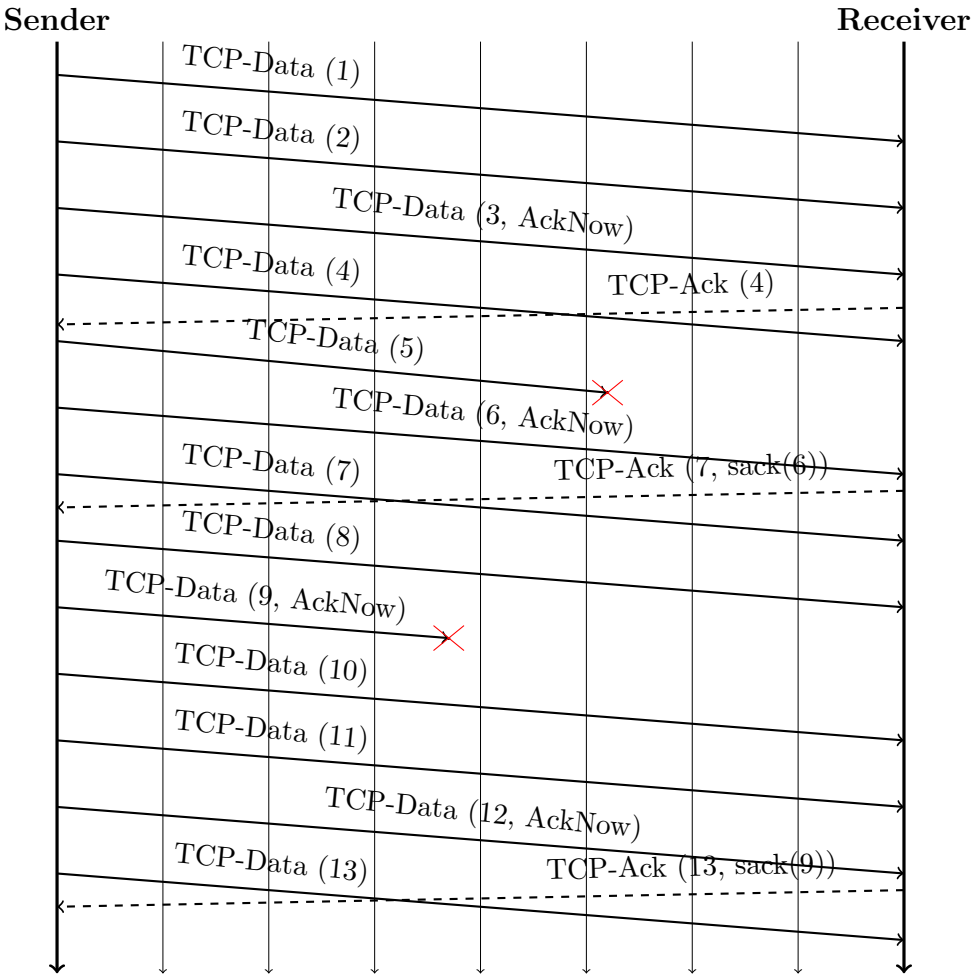


Figure 4.6: TCP delayed acknowledgment recovery

4.5.1 Simulation environment

To evaluate the performance of our improvements, we have implemented DCA, TDA, DTC and NewDTC on INETMANET [Vag11] a framework of the OMNET++ [Vag10] network simulator. We have performed simulations with a unidirectional TCP data transfer. We have used a chain topology as shown in Figure 4.7 where node n is in the transmission range of node $n-1$ and $n+1$. The distance between two neighbor nodes is 50 meters. Like the scenario in [DAV04], the TCP sender (source) sends 1000 TCP data segments (64 bytes) to the TCP receiver. Table 4.1 contains the values of all scenario parameters. Our simulations consist of 10 runs, and the reported results are the average of the 10 runs.

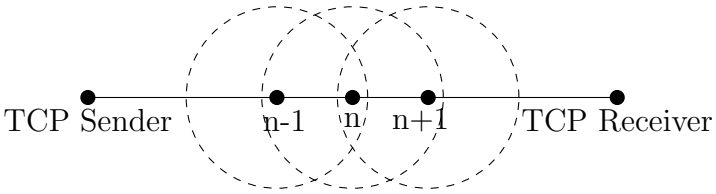


Figure 4.7: Chain Topology

Table 4.1: Simulation parameters

Parameters	Value
Maximum Segment Size	64 bytes
Routing Protocol	Static routing
MAC Layer	IEEE 802.15.4
PHY bitrate	250kb/s

4.5.2 Energy Model

To measure the amount of consumed energy by wireless nodes, we apply the following energy model. At a given moment, a wireless node is on one of four states: Transmit, Receive, Sleep or Idle. Table 4.2 contains the voltage and the current value of each state. These values have been obtained from CC2420 Datasheet and the Texas Instruments MSP430 Datasheet[2]. The energy consumed to transmit a link-layer frame equals

$$\text{Transmit energy} = \text{Voltage} \times \text{Transmit Current} \times \text{Transmit time}.$$

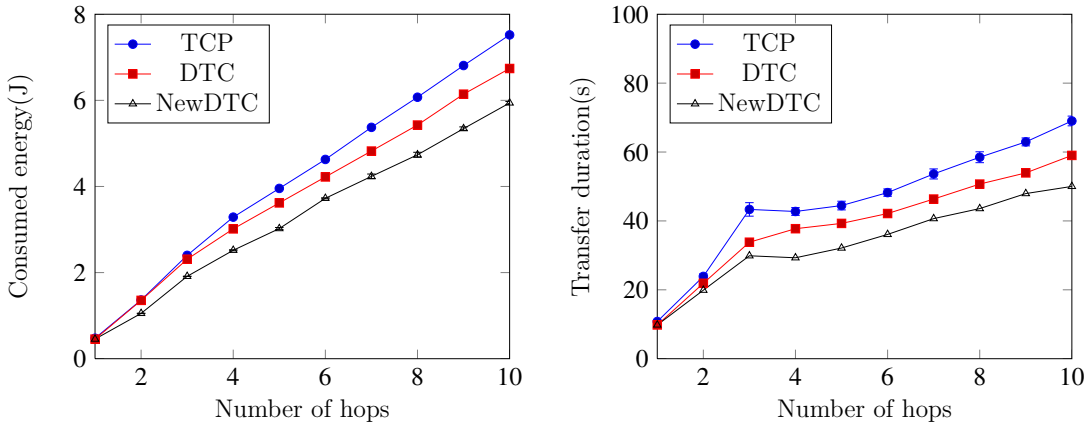
Table 4.2: Energy model of wireless nodes

Parameters	Values
Voltage	3V
Transmit Current	17.4 mA
Receive Current	19.7 mA
Idle Current	1.38 mA
Sleep Current	0.06 mA

4.5.3 Comparison between NewDTC, TCP and DTC

In this section, we compare the new version of DTC (i.e., NewDTC) which includes all our modifications to DTC and the original version of DTC described in [DAV04]. The compar-

[2]http://focus.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=msp430f1611



(a) Total energy consumption

(b) Transfer Duration

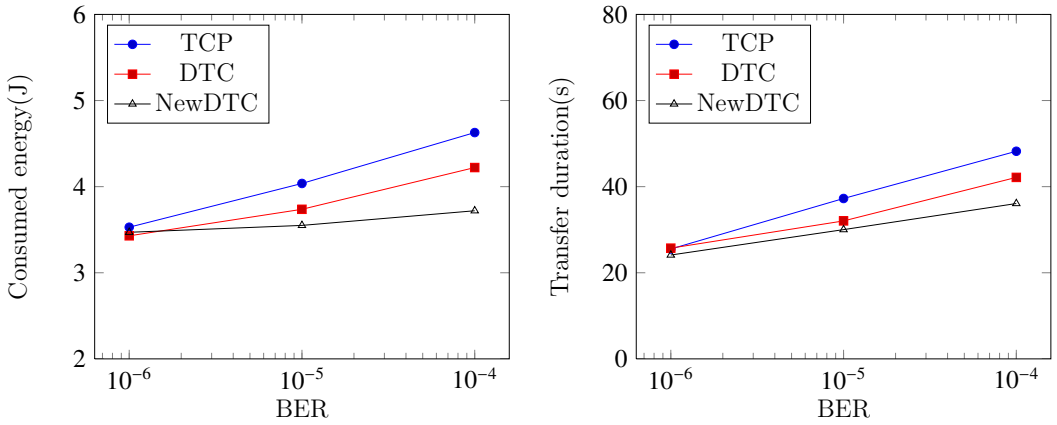
Figure 4.8: Comparison TCP, DTC, NewDTC in terms of consumed energy and transfer duration

ison metrics that we propose are: **total consumed energy** and **transfer duration**. We simulate two scenarios. In the first one, we increase the number of nodes in the network and thus the number of hops. In the second scenario we fix the number of nodes to seven and we simulate the same scenario with different bit error rate.

4.5.3.1 Number of hops

In order to justify our motivation for hop-by-hop retransmissions, we compare NewDTC and DTC to TCP. Figure 4.8 shows that DTC and NewDTC reduce the total consumed energy. This is due to the hop-by-hop recovery of lost packets. We also distinguish that NewDTC out-performs DTC in energy consumption by deleting already acknowledged TCP segments.

Sending a file is not the main purpose of low power network applications. However, if the transfer duration takes a long time, the performance of the WSN would decrease. Using the same topology as defined above, we compute the transfer duration of the same file using TCP, DTC and NewDTC. Figure 4.8 shows that in a CSMA-CA network, NewDTC reduces the transfer duration by about 35-60% compared to TCP and by 14-45% compared to DTC. This is due to the proposed cache management which reduces unnecessary end-to-end retransmissions, a scheme that disables sending multiple copies of the lost segments, and an acknowledgment detection mechanism that prevents an intermediate node to retransmit an acknowledged segment.



(a) Consumed energy

(b) Transfer duration

Figure 4.9: Comparison of TCP, DTC and NewDTC with different bit error rates (number of hops=6)

4.5.3.2 Bit Error Rate

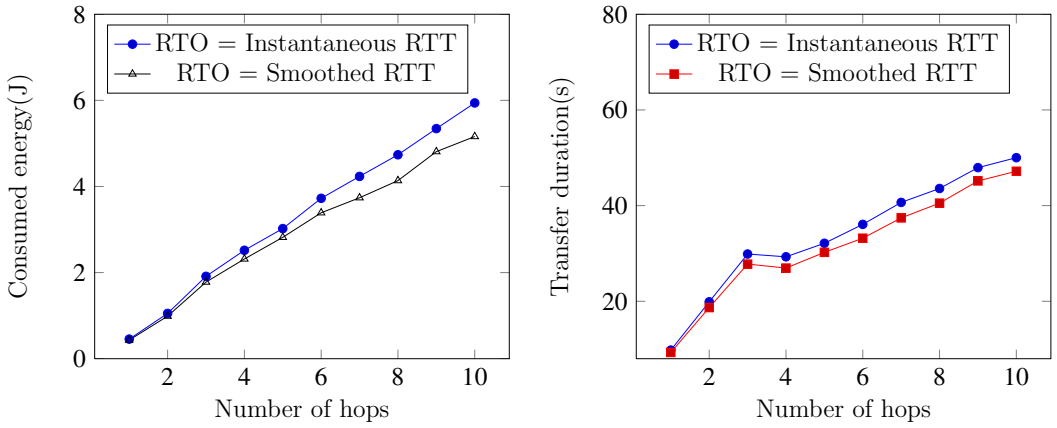
In this section, we study DTC performance with different bit error rate scenarios. Figure 4.9 compares the total consumed energy by TCP, DTC, and NewDTC nodes. It shows that NewDTC nodes consume 27.9% less than the TCP ones in high BER networks (10^{-4}) and about 23% less in low BER networks (10^{-6}).

Figure 4.9 shows that NewDTC also reduces the file transfer duration. Results show that NewDTC achieves higher throughput than TCP and DTC.

4.5.4 Round Trip Time Computation

DTC proposes to compute an RTT value from the time of reception a data segment to the time of reception of its ACK and then take 1.5 × RTT as RTO value. In Section 4.3.5, we explained that the TCP SACK option can also used to compute the RTT value. In fact, if an intermediate node receives a TCP segment with a SACK option acknowledging segment n, then, it computes the RTT of this segment and then updates the RTO.

In order to justify our motivation for the computing method of RTT, we compare NewDTC to the same version of NewDTC with RTT smoothing. We propose the use of Paxson and Allman's algorithm [PA00] for computing the RTO value. Figure 4.10 shows that a better RTO computing leads not only to reduce the total consumed energy by wireless nodes but also to decrease the transfer duration and thus to improve the TCP throughput.



(a) Consumed energy (b) Transfer duration

Figure 4.10: Smoothed RTT computing improves TCP performance

4.5.5 Comparison between TCP, DCA, and TDA

Now, we focus on studying the impact of delayed acknowledgments on the energy consumption. All previous studies compared between the proposed TCP algorithms based on the throughput. In this section, we present a first energy efficiency comparison of these TCP algorithms.

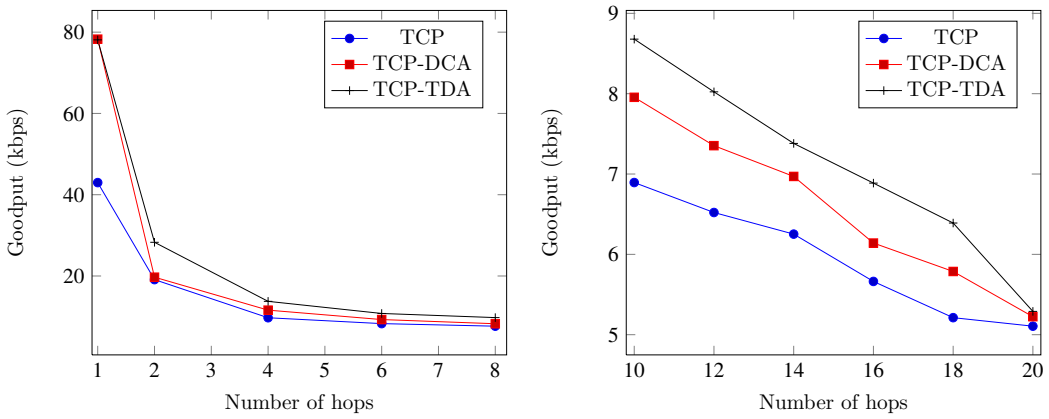


Figure 4.11: Comparison between TCP, TCP-TDA and TCP-DCA in terms of Goodput

Figure 4.11 shows that our results are close to the ones presented in [CMSM09]. In fact, TCP-TDA increases more the TCP throughput than TCP-DCA. TCP-TDA does not limit the delayed window size which allows the TCP source to send more segments before receiving an acknowledgment. We assume that the source node has a large sending buffer

to do that.

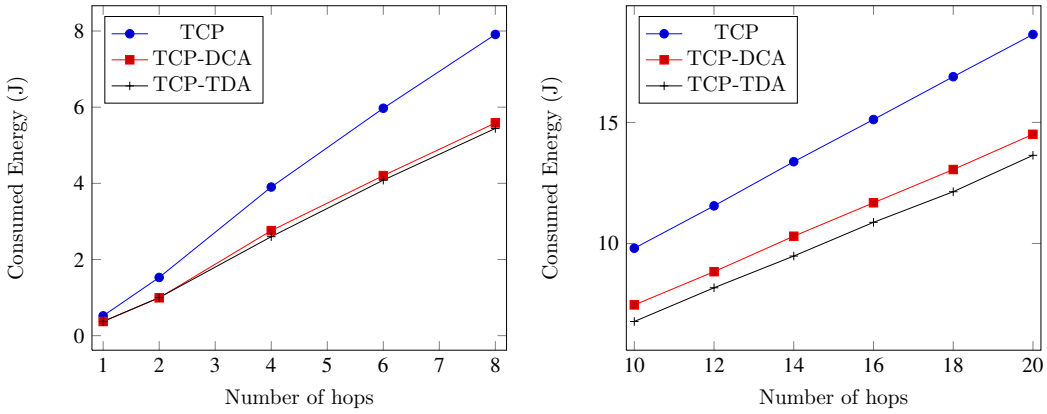


Figure 4.12: Comparison between TCP, TCP-TDA and TCP-DCA in terms of consumed energy

Figure 4.12 shows the consumed energy by all wireless nodes in different TCP algorithms. We can see that all TCP delayed acknowledgment algorithms are more energy efficient than standard TCP and reduce the total consumed energy by 40%. Moreover, Figure 4.12 shows that TCP-TDA is more energy-efficient than TCP-DCA because it reduces more the number of TCP acknowledgments segments.

4.6 Conclusion

In this chapter we have identified some points that should be improved to make TCP viable for low power networks. This chapter proposed some improvements to a distributed TCP caching algorithm for unicast transmission in low power networks. We have evaluated our contribution by simulation and verified that it reduces the amount of consumed energy by up to 30 percent and increases the throughput due to a better congestion management.

Chapter 5

TCP header compression for low power networks

Contents

5.1 Introduction 54
5.2 Related work 55
5.3 TCP Header Format 56
5.4 TCP Header Compression 58
5.4.1 Dynamic fields compression 60
5.4.2 Context management 63
5.4.3 Segment loss management 63
5.4.4 LOWPAN_TCPHC Format 64
5.4.5 TCP Option Compression 66
5.4.6 Example of compressed TCP headers 67
5.5 Experimental Setup 68
5.5.1 Physical setup 68
5.5.2 Hardware setup 68
5.5.3 Software setup 69
5.5.4 Energy consumption 70
5.6 Results and Discussion 71
5.6.1 TCPHC performance in loss-free environments 71
5.6.2 TCPHC performance in lossy environments 73
5.7 Conclusion 73

5.1 Introduction

In this chapter, we focus on TCP header compression for low power networks. Deploying TCP allows current IP-based devices to communicate directly with all wireless devices using their TCP/IP stack. Moreover, non-TCP/IP reliable transport protocols need a complex algorithm on the Edge Router. Thus the use of TCP reduces the complexity at the Edge Router (ER), which is an IPv6 router that interconnects the wireless network to another IP network.

Nevertheless, TCP performance is mainly harmed due to its header length. As an illustration, the maximum physical layer packet size in IEEE 802.15.4 networks [IEE06] is 127 bytes, and medium access layer and link layer security requirements leave only 81 bytes for upper layers data; finally, the use of IPv6 [Ste98] as a network protocol header (40 bytes) without compression leaves only 41 bytes for transport protocols. TCP uses 20 bytes in the header (if no TCP options are included), which leaves only 21 bytes for the application layer if no compression scheme is applied. Even with 6LoWPAN adaptation layer, a pure TCP acknowledgment (i.e., a TCP ACK carrying no data) without any TCP options represents 25% of the payload of an IEEE 802.15.4 MAC frame while TCP pure ACKs represent roughly 33% of the total number of segments exchanged in a TCP session (this figure may go up to roughly 50% if the Delayed ACK mechanism is not used). Therefore, a TCP header compression algorithm is needed for 6LoWPANs. Such a compression algorithm should respect some requirements: *efficiency* (the scheme must provide small header), *transparency* (the resulting header after a compression and decompression should be identical to the original header), and *disordering tolerance* (the scheme must be able to decompress compressed segments correctly even when segments arrive with a moderate disordering (1-2 packets)). Without these requirements the header compression algorithm can not be energy-efficient.

We propose LOWPAN_IPHC (TCPHC) [ART10], a new TCP header compression algorithm in order to reduce significantly the TCP header size for 6LoWPAN. The TCP header compression is performed in wireless nodes and the edge routers between the 6LoWPAN and the external IP network. Moreover, the TCPHC mechanism can be used in conjunction with the IPv6 header compression proposed by 6LoWPAN (IPHC) [HT10] and thus reduces all header overheads (i.e., IPv6 and TCP) to about seven to 10 bytes instead of 60 bytes.

The goal of this chapter is to introduce our new TCP header compression algorithm and to present an experimental evaluation of the TCPHC algorithm. The evaluation is done on our testbed in different environments (low-BER and high-BER environments). We compare the performance of the legacy TCP and of TCPHC based on two main metrics, namely transfer duration and the consumed energy. The results show that the TCPHC mechanism can reduce the size of the TCP header to 6 bytes in 95% of the cases and the consumed

energy by 9% to 16% depending on the scenario. Although these gains seem low, given the context of low power networks, our header compression algorithm would increase the lifetime of the low power networks.

The remainder of this chapter is organized as follows. Section 5.2 presents a brief overview of the related works in the area of TCP header compression. Section 5.3 presents the TCP header fields. Section 5.4 presents an overview of the TCPHC mechanism. We describe our testbed in Section 5.5. Section 5.6 shows experimental results.

5.2 Related work

This section presents prior work on TCP/IP header compression. In particular, we briefly describe three existing TCP header compression algorithms and discuss why they do not fit to low power networks. A more detailed discussion of these algorithms can be found in [JPS07].

One of the first TCP/IP header compression methods was Compressed TCP (CTCP), which has been proposed by Jacobson [Jac90]. Jacobson's header compression algorithm distinguishes dynamic fields from static fields. The static fields (i.e., fields that are expected to be constant throughout the lifetime of the packet stream such as source address and source port) are sent in two situations: when initiating a connection, and when refreshing the context (i.e., the state used by the compressor to compress a header, and by the decompressor to decompress a header) after a loss of synchronization. CTCP proposes to send the difference between the current and the previous value of dynamic fields (e.g., sequence number, acknowledgment number). When the synchronization is lost between the compressor and the decompressor (i.e., the destination does not success to decompress the compressed segments), the TCP sender sends a segment with a regular header to refresh the context. Experimental studies [PM97, SFRF01, Wan04] have shown that the performance of Jacobson's algorithm may degrade significantly in noisy/lossy network environments. An important disadvantage of CTCP is that it does not support TCP options, some of which are ubiquitous nowadays (e.g., SACK).

In [DNP99], Degermark *et al.* enhances Jacobson's TCP header compression by introducing a mechanism, called TWICE, to repair incorrectly-decompressed headers. [DNP99] also describes a mechanism for explicitly requesting the transmission of less-compressed or uncompressed headers. Such a mechanism is especially suited for pure TCP acknowledgments. Note however that Degermark's header compression algorithm does not currently provide a compression method for TCP options; changing option fields are carried in compressed headers, but without any compression. Also, the header request mechanism may be unsuited for 6LoWPAN networks, whose low bit rates and strong energy constraints are at odds with any additional signaling overhead. In addition, these two schemes are

based on sending a delta for the evolution of the sequence and acknowledgment fields. In case of de-sequencing, this latter leads to a complex reordering. TCPHC enhances the two previous algorithms by defining an adapted header compression of TCP for low power networks by sending the least significant bytes instead of sending a delta value. Moreover, TCPHC completes these solutions by defining header compression schemes for the most commonly used TCP options.

ROHC-TCP [JPS07] improves [DNP99] by providing a new method for compressing all TCP header fields, including the TCP options. ROHC-TCP proposes also to start compressing packets from the SYN segments, using parameters from previous or simultaneous connections. This may offer noticeable improvements in performance when most TCP flows are short-lived, i.e., composed of a small number of data segments. Nevertheless, the ROHC-TCP algorithm is fairly complex and its memory requirements may not be adequate in memory-constrained devices.

The TCPHC algorithm, described in this chapter, supports features like the compression of TCP options adapted for low power networks and at the same time it is relatively simple and easy to implement in memory- and CPU-constrained devices.

5.3 TCP Header Format

The TCP header consists of 11 fields, of which 10 are required. The 11th field is an optional extension field. Figure 5.1 shows the TCP header format. A detailed description of the TCP header fields can be found in [Pos81, RFB01, DNP99].

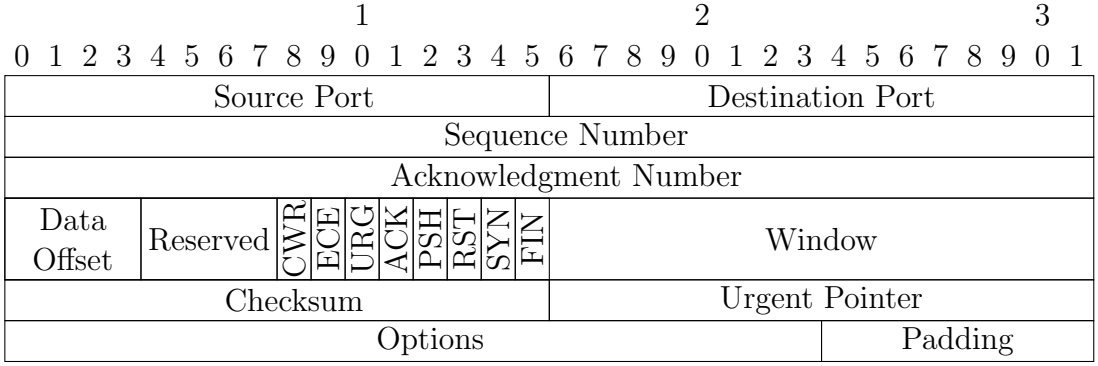


Figure 5.1: TCP header format

Source Port (16 bits): The source port number.

Destination Port (16 bits): The destination port number.

Sequence Number (32 bits): The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgment Number (32 bits): If the ACK control bit is set this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.

Data Offset (4 bits): The number of 32 bit words in the TCP Header. This indicates where the data begins. The TCP header (even one including options) is an integral number of 32 bits long.

Reserved (4 bits): Reserved for future use. Must be zero.

Control Bits (8 bits): from left to right

CWR: Congestion Window Reduced flag is set to indicate that it received a TCP segment with the ECE flag set.

ECE: if SYN flag is set (1) ECN-Echo indicates that the TCP node is ECN capable, else the packet it indicates that a packet with ECN flag set in IP header is received (added to header in [RFB01]).

URG: Urgent Pointer field significant.

ACK: Acknowledgment field significant.

PSH: Push Function.

RST: Reset the connection.

SYN: Synchronize sequence numbers.

FIN: No more data from sender.

Window (16 bits): The number of data octets beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to receive.

Checksum (16 bits): The 16 bit checksum field is used for error-checking of the header and data.

Urgent Pointer (16 bits): This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only be interpreted in segments with the URG control bit set.

5.4 TCP Header Compression

This section presents LOWPAN_TCPHC (TCPHC) , the TCP header compression mechanism for 6LoWPANs. The main purpose of TCPHC is to reduce the protocol header overhead, with the intent of reducing both bandwidth usage and energy consumption due to packet transmissions.

Indeed, TCPHC does not only introduce a header compression algorithm but also provides a scheme to allow establishing TCP connections between an external IP host and a LoWPAN host, and between two LoWPAN hosts. The former type of connection is performed by an Edge Router (ER), which links the 6LoWPAN to an external IPv6-based network. Figure 5.2 shows a typical 6LoWPAN topology with three edge routers, which create a gateway between the LoWPAN network and the external IP network.

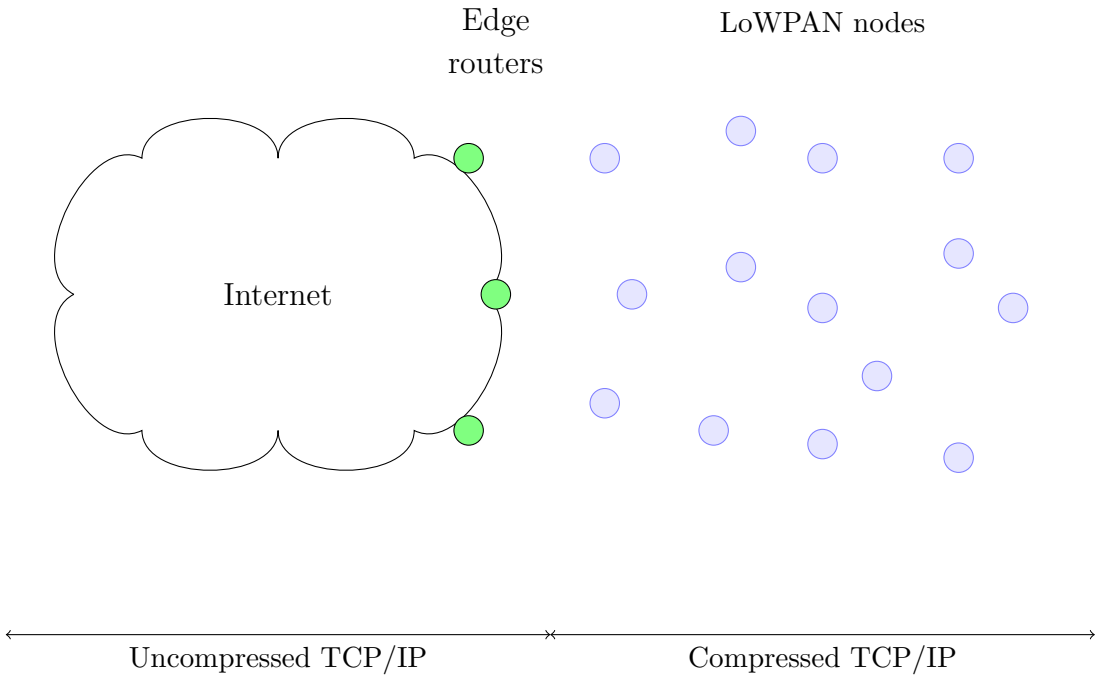


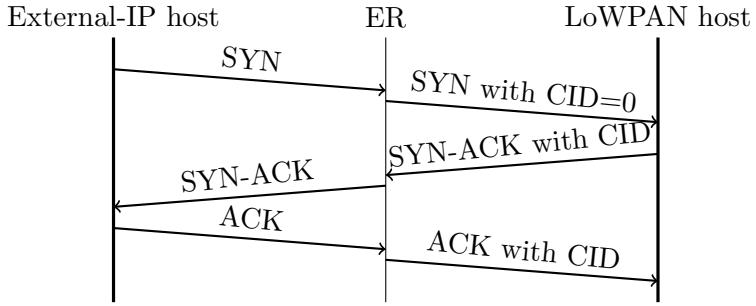
Figure 5.2: The IPv6 over Low power Wireless Personal Area Network Topology

There are three kinds of headers:

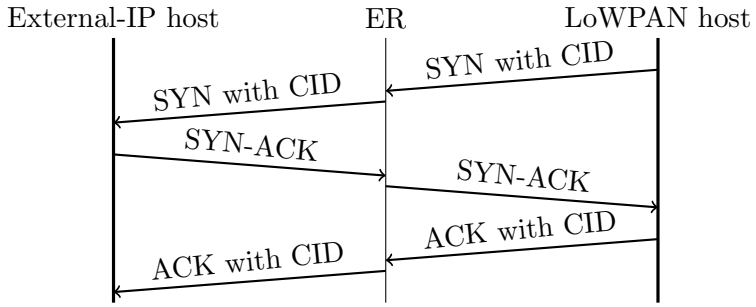
- regular header: a normal, uncompressed header that does not carry any context identifier (CID), which is a small unique number identifying the context that should be used to decompress a compressed header.
- full header: an uncompressed header that updates or refreshes the context for a packet stream. It carries a CID that will be used to identify the context,

- compressed header: a header in which all the static fields are elided, and all the dynamic fields are sent compressed.

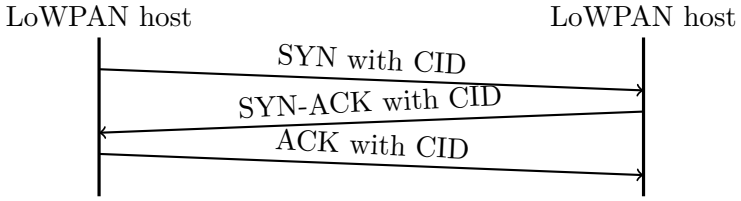
The compression and the decompression mechanisms are implemented on the edge routers and on the LoWPAN hosts. The external IP host sends and receives regular TCP segments, whereas the LoWPAN host sends and receives segments with compressed headers or full headers. The TCP connection can also be established between two LoWPAN hosts inside the same low power network for Machine-to-Machine (M2M) communications purposes.



(a) TCP connection initiated by an external IP-host



(b) TCP connection initiated by a LoWPAN host



(c) TCP connection between two LoWPAN hosts

Figure 5.3: TCP connection initiation

Thus, all segments inside the LoWPAN can be either full header segments or compressed header segments. The TCP header compression algorithm defines two kinds of headers. For the TCP opening phase or messages with the URG flag set to 1, full header segments are

sent. For the other situations, compressed header segments are sent. All these segments contain the Context Identifier (CID), which is used, with the IPv6 address of the first involved LoWPAN node, to identify the connection during the transfer phase and thus avoids to send on each packet the port numbers. The first LoWPAN node involved in the TCP connection assigns the CID value and its size. Figure 5.3 shows the exchanged TCP control segments in the TCP connection establishment phase.

5.4.1 Dynamic fields compression

In this section, we define the TCPHC specifications for TCP header compression for IEEE 802.15.4 networks. TCPHC initiates the compression algorithm by exchanging a context identifier at the beginning of the connection. The compressor and decompressor nodes save most fields of the first full headers as a context. The context consists of the header fields whose values are constant. These fields should be elided because they are the same or have few changes relative to the previous header. It is more efficient to send fewer bits, which represent the difference from previous value, compared to the sending of the absolute value. This mechanism is based on sending not all TCP fields, but only fields or parts of fields that do change from the last one sent. For example, the most significant bytes of the sequence number field can be elided if they are equal to those of the previous segment. The following paragraphs detail how the TCP dynamic fields are compressed.

- **Sequence and acknowledgment numbers:** the sequence number is the number of the first data byte in the segment (except for the first segment). The length of the sequence number field is four bytes.

  In a TCP connection, the sequence number is incremented for each segment by a value, which is between zero and the Maximum Segment Size (MSS). Thus, the two least significant bytes would change more frequently than the two most significant ones. For example, only the least significant byte (LSB) should be sent if the other bytes do not change. The decompressor module can deduce the elided bytes from the previously received segments. The sequence number can be elided if a receiver does not send data to the source and is acknowledging data segments. The same algorithm is used for the compression of acknowledgment number and only bytes, which are changed, should be carried in-line. If the TCP sink does not generate data, the four bytes of sequence number are omitted in all acknowledgment segments and only compressed acknowledgment fields should be sent.

  Figure 5.4 shows an example of a sequence number compression. In the first step, the source node compares the sequence number of the new TCP segment with the sequence number of the context. In this example, only one byte of the segment number (the less significant) has changed from the last segment sent. That byte

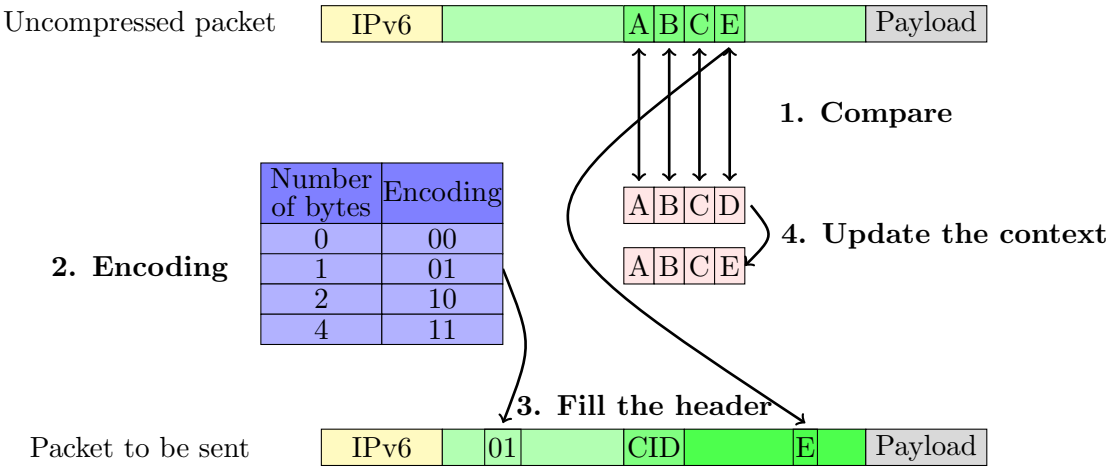


Figure 5.4: Sequence number compression

must be sent with the corresponding encoding. TCPHC fills the header with the right encoding in the LOWPAN_TCPHC header and the uncompressed header field with the uncompressed byte of the sequence number. Finally, TCPHC updates the context fields.

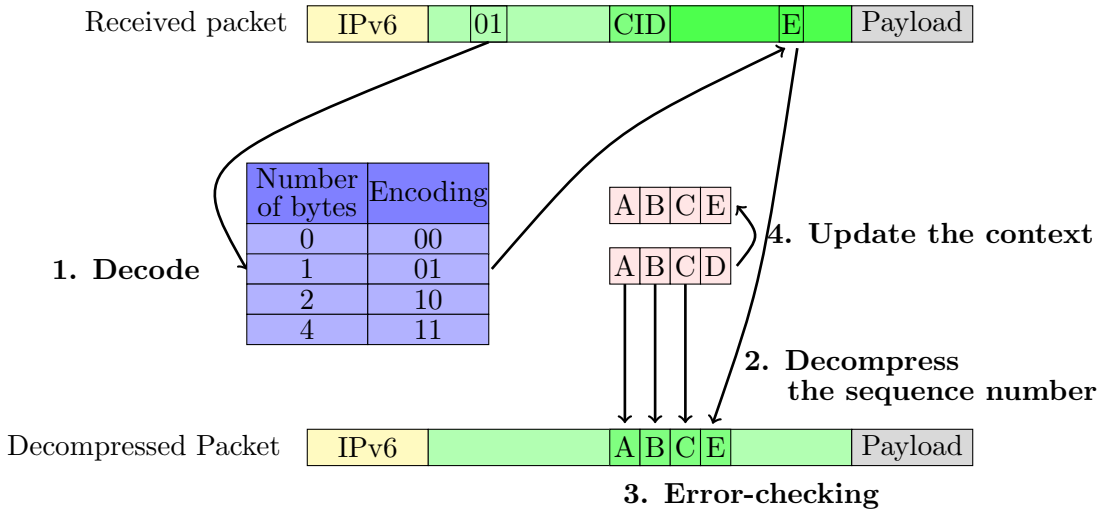
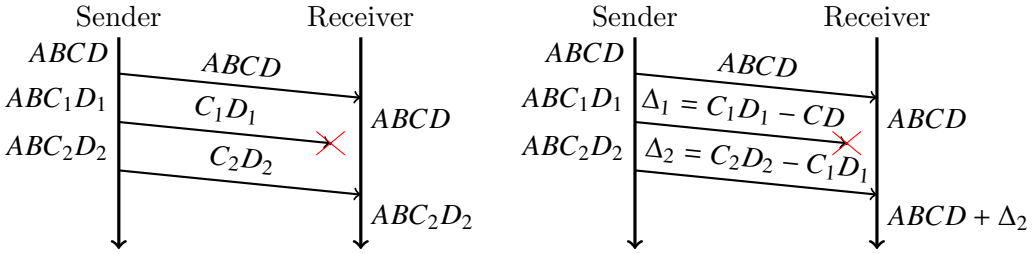


Figure 5.5: Sequence number decompression

Figure 5.5 shows an example of a sequence number decompression. In the first step, the receiver decodes the two byte of LOWPAN_TCPHC to know what are the fields that have been compressed and how they are compressed. In this example, only one byte of the sequence number (the less significant) has changed. That byte is

completed by three byte from the sequence number of the context. Then, TCP does an error-cheking in order to assure correctness. Finally, TCPHC updates the context fields.



(a) TCPHC: The receiver succeeds to decompress the third segment

(b) CTCP: The receiver fails to decompress the third segment

Figure 5.6: Comparison between TCP and CTCP in terms of sequence number compression

Our algorithm is more loss-tolerant than Jacobson's algorithm (CTCP). In fact, even if one TCP segment is lost, our TCP header compression algorithm has more chances to decompress the next compressed TCP segment. Figure 5.6 shows a scenario where the first segment is sent with-in full header. The second segment is lost and has not been by the receiver. CTCP sends detla in the segment 2, while TCPHC sends the two uncompressed bytes of the sequence number field. The third segment segment is correctly received in both cases, however, only TCPHC succeeds to uncompress the compressed segment and keeps the synchronization between the sender and the receiver.

- **The window** field can be omitted if it does not change in time. The TCP sender compares the window size in the TCP header and in the context. If only one byte is different, the different byte must be sent. Otherwise, if all the two bytes are the same, nothing must be sent. Else, if both of the two bytes are different, 2 bytes must be sent.

- **Flags** are omitted because TCP control messages are sent uncompressed, except SYN flag. The not compressed flags are: Push, Urgent, Congestion Window Reduced (CWR) and ECN-Echo (ECE). They are sent in the TCPHC encoding.

- **The urgent pointer** field is sent in uncompressed header format only if the urgent flag is set. Otherwise, this field is elided.

- **The checksum** is not compressed and is used by the receiver to check if the decompressed TCP segment is received correctly.

5.4.2 Context management

The CID management is an important key feature of LOWPAN_TCPHC. The CID is always allocated by the LOWPAN hosts. The Interface Identifier (IID) (e.g., the MAC address) of a LoWPAN host and the CID is utilized as a key by the wireless hosts and the edge routers in order to identify a TCP connection. Because the CID and context are precious resources for the sensor node, we tried to use them efficiently in our implementation.

Figure 5.3 (a) shows an example where an external host initials a TCP connection. Upon receiving the SYN segment, the ER does not create a new context but retransmits the SYN segment to the wireless host in full header format with an CID field equal to 0. If the wireless host accepts the connection, it creates a new context, and assigns to it the smallest number from the available CID numbers. When the ER receives the SYN-ACK, it creates a new TCP header compression context using the received information from the TCP header fields.

In a second case, where a wireless node initiates the TCP communication, the ER creates directly a new context when it receives the SYN segment. The SYN segment contains the CID chosen by the wireless node (see Figure 5.3 (b)). In the last case (i.e., a TCP connection between two wireless nodes), the CID is chosen by the node that initiated the connection (see Figure 5.3 (c)).

The edge router to which a LoWPAN node host is attached may change over time, due to route instability or to host mobility. However, this change should not break the TCP communication. To ensure the TCP communication despite the change of ER, the ERs should share the contexts of current connections. So, even if a 6LoWPAN node changes its attached ER, the new ER should continue to compress the segments using the same context. Context exchange and management between ERs are left for future work.

The ER should free a context when a TCP connection is finished (e.g., reception of FIN control messages). The edge router can also free a connection after a long silent period (i.e., when no messages are exchanged after a certain period of time). The ER can remove the context of a TCP connection after a long period of inactivity that may not be closed. In this case, after receiving a new data segment, the connexion continue but without any compression.

5.4.3 Segment loss management

Here, we present how the TCPHC mechanism should react when a segment is lost or is assumed to be lost. The loss is handled when the TCP ACK segment is not received within the retransmission timeout (RTO). The ER handles a retransmission by scanning the sequence numbers. The ER should send a retransmitted segment without compressing the dynamic fields. This mechanism allows updating the context on both sides after a

packet loss.

We assume that the 6LoWPAN has a low bit rate, and also that nodes are memory-constrained and thus the TCP window size is probably limited to a few segments. In this case, the loss of synchronization will likely not lead to a burst of losses. Therefore, this thesis does not present a refresh algorithm to update the context between the compressor and the decompressor.

5.4.4 LOWPAN_TCPHC Format

5.4.4.1 TCP segments types

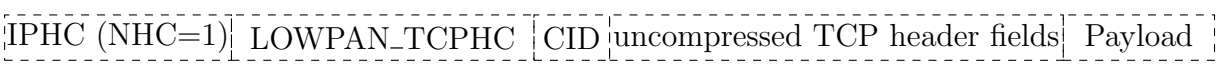
This section presents different types of TCP segments. In fact, three types of packets are used in a TCP session with header compression: regular header, full header and compressed header. Figure 5.7 shows the fields of different TCPHC packets. Figure 5.7 (a) shows the header stack of a regular TCP segment. The 6LoWPAN next header compression (NHC) flag of the 6LoWPAN encoding is equal to 0 and indicates that next header (i.e., TCP header) is not compressed.

Figure 5.7 (b) shows a full header TCP segment, an uncompressed header that updates or refreshes the context for a packet stream. It carries a dispatch equals to 00000001 and a CID that will be used to identify the context. Note that the NHC flag of the 6LoWPAN encoding is equal to 1 and indicates that the TCP header compressions is enabled.

Figure 5.7 (c) shows the header stack of a compressed TCP segment. The NHC flag of the 6LoWPAN header is equal to 1 and indicates that the TCP header is compressed. The TCPHC header carriers a LOWPAN_TCPHC fields of two bytes that describe how the TCP header fields are compressed or elided. This field is followed by a CID and the uncompressed TCP header fields. The next section describes with more details the LOWPAN_TCPHC encoding.

(a) Regular header TCP segment

(b) Full header TCP segment



(c) Compressed header TCP segment

Figure 5.7: Different TCPHC packet format

5.4.4.2 LOWPAN_TCPHC Format

Figure 5.8 shows the fields of LOWPAN_TCPHC.

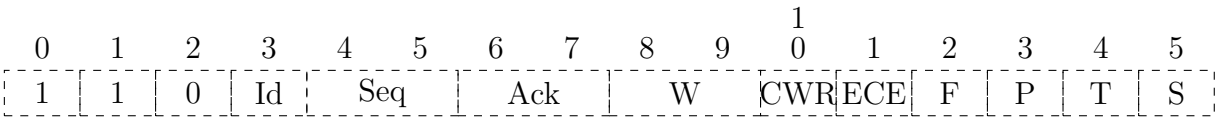


Figure 5.8: TCP Header Encoding

Id: Context Identifier size

- **0:** CID is coded in 8 bits.
- **1:** CID is coded in 16 bits.

Seq: Sequence Number:

- **00:** All 32 bits of Sequence Number are elided.
- **01:** The 8 less-significant bits of Sequence Number are carried in-line. The remaining 24 bits are elided.
- **10:** The 16 less-significant bits of Sequence Number are carried in-line. Last 16 bits of Sequence Number are elided.
- **11:** All 32 bits of Sequence Number are carried in-line.

Ack: Acknowledgment Number:

- **00:** All 32 bits of Acknowledgment Number are elided.
- **01:** The 8 less-significant bits of Acknowledgment Number are carried in-line. The remaining 24 bits are elided.
- **10:** First 16 less-significant bits of Acknowledgment Number are carried in-line. Last 16 bits of Acknowledgment Number are elided.
- **11:** All 32 bits of Acknowledgment Number are carried in-line.

W: Window:

- **00:** The Window field is elided.
- **01:** The less-significant byte of Window field is carried in-line. The second byte is elided.

10: The most-significant byte of Window field is carried in-line. The first byte is elided.

11: Full 16 bits for Window field are carried in-line.

F: Fin flag

P: Push flag

CWR: Congestion Window Reduced

ECE: ECN-Echo

T: Indicates if the TCP header contains Timestamp option

S: Indicates if the TCP header contains SACK option

Fields carried in-line (in part or in whole) appear in the same order as they do in the TCP header format. The TCP Length field must always be elided and it is inferred from lower layers using the 6LoWPAN fragmentation header or the IEEE 802.15.4 header.

5.4.5 TCP Option Compression

This section defines a compression method for the TCP options most likely to be used in 6LoWPAN. The TCP options are negotiated at the connection establishment phase. The ER can decide to allow or to deny an option sent in the SYN segment. This is compatible with standard TCP even if the TCP host in the external network does not know who refused the TCP options. LOWPAN_TCPHC compresses the mostly used TCP options : SACK and Timestamp. We assume that the SACK and Timestamp are enabled by default. The MSS option is sent uncompressed in the SYN segments. The Window Scale Option (WSO) is useless in 6LoWPAN because it is more performance to use small windows than large windows.

LOWPAN_TCPHC specifies two bits for SACK and Timestamp TCP options (see 5.8). Figure 5.9 shows the structure of a TCP segment including option compressed using LOWPAN_TCPHC. The size of the SACK option is 4 bytes and the size of Timestamp option is variable from 4 to 8 bytes.

Figure 5.9: TCP header option configuration

5.4.5.1 SACK

The SACK option [Mat96,FMMP00] should be negotiated in set-up phase, then the option may be used when dropped segments are detected by the receiver. This option is to be used to convey extended acknowledgment information over an established connection. LOWPAN_TCPHC allow to send only one block SACK. The left edge of the block can be replaced by the offset between the first byte of the segment and the right edge by the length of the block. The Left edge and the right edge will be coded in 16 bits (see Figure 5.10).

Figure 5.10: Compressed SACK option

5.4.5.2 Timestamp

This option carries eight-byte timestamp fields. If timestamp options [JBB92] are exchanged in the connection set-up phase, they are expected to appear on all subsequent segments. This overhead added by this option can be reduced: a TCP node, which does not sent data, is not interested in computing the RTT. And thus, it can reply by sending only Timestamp Echo Reply field (TSecr). However, the Timestamp Value field (TSval) is more important for TCP that send data.Then, if the T flag and ACK flag are set, it mean that the next 4 byte contain the TSecr. Otherwise, (i.e., if the ACK flag is not set) the 4 bytes contain the TSval.This optimization is only valid when a single TCP sends data. Otherwise, the two four-bytes should be sent.

5.4.6 Example of compressed TCP headers

In this section, we present an example of a compressed TCP header using LOWPAN_TCPHC. Figure 5.11 represents a header of a compressed TCP data segment. The first 16 bits are the LOWPAN_TCPHC encoding. The window field has not been changed compared to its antecedent, the two bytes of the lowest bytes of the sequence number that have been changed. The timestamp and Sack options bits are equal to 0 and indicates that "no TCP header option". The size of this header is seven bytes.

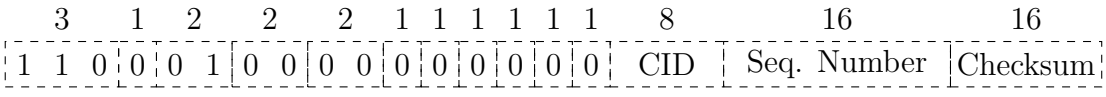


Figure 5.11: Compressed TCP header encoding

5.5 Experimental Setup

In this section, we describe our wireless testbed that we used in order to evaluate our header compression algorithm. Moreover, we present the hardware setup, the physical setup, and the placement of our wireless devices.

5.5.1 Physical setup

In our wireless testbed, seven wireless nodes are distributed with the same distance (between three and four meters) between each neighbor. The position of the wireless devices is shown in Figure 5.12. We can distinguish four types of wireless devices based on their functionalities:

1. The Edge Router (ER) is the border router that connects the wireless network to the IP-based wired network.

2. Wireless nodes (N_1, N_2, N_3, N_4, and N_5) can either be a Terminal Node (TN), or only a relay of data frames from the ER to the TN and vice-versa.

3. The External Node (EN) is an external sensor node in the same wireless network, which generates a CBR traffic to increase the packet loss ratio.

5.5.2 Hardware setup

In our testbed, all wireless devices are connected to a standard laptop by the USB port. This solution allows us to log the output messages from the wireless devices to the laptop.

The embedded devices used in our testbed are Crossbow TelosB [1] motes. They use Texas Instruments MSP430 microcontroller, which offers a 10kB RAM, and a 48 kB program flash memory. The Crossbow TelosB radio is CC2420 [2], which uses ISM frequency band (from 2400 MHz to 2483.5 MHz) and offers 250 kbps data rate.

Before starting our experiments, we explored the use of radio channels. We found that the radio environment is highly polluted by the IEEE 802.11 networks, because the IEEE 802.15.4 uses the same radio frequency as IEEE 802.11. To reduce this effect, we used the last channel of IEEE 802.15.4 (channel number 26) as described in [IEE06].

[1] http://www.hoskin.qc.ca/uploadpdf/Instrumentation/CrossBow/hoskin_TPR2400CA_42efb73715b8b.pdf
[2] www.flexipanel.com/Docs/CC2420_Data_Sheet_1_2.pdf

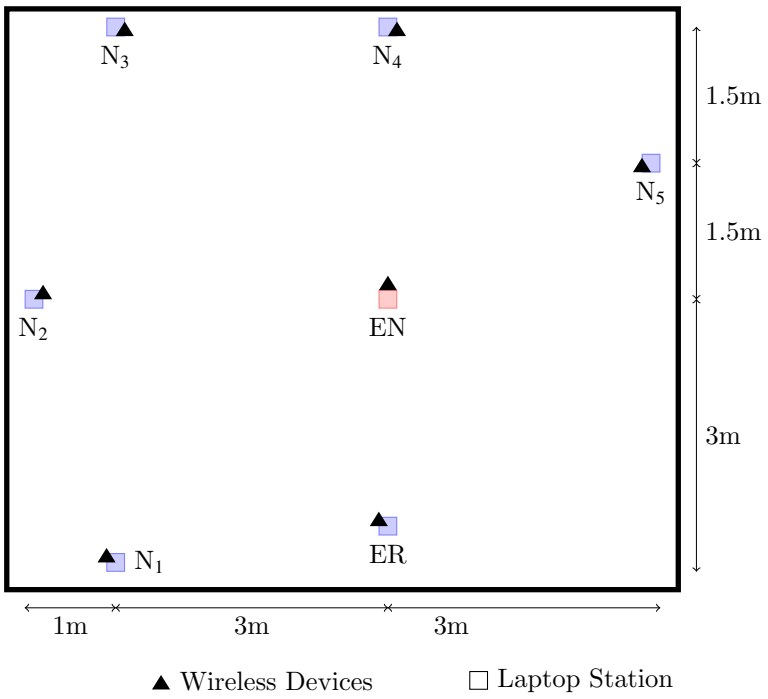


Figure 5.12: The distribution of the wireless motes in the testbed

5.5.3 Software setup

In our work, we use Contiki OS as the operating system for our wireless devices.

We have chosen Contiki OS because it already implements 6LoWPAN, UDP, and TCP. Contiki OS[3] is an open source operating system for networked embedded devices that includes the uIPv6 [Dun03] stack. Moreover, Contiki OS provides standard operating system features like threads, timers, random number generator, clocks, a file system, and a command line shell.

The 6LoWPAN implementation in Contiki OS is conformant to [HT10]. TCP is partially implemented on Contiki OS because of the memory-constraints of the wireless devices. In uIP, all RFC requirements that affect host-to-host communication are implemented, except for some mechanisms, such as soft error reporting.

A more detailed description of TCP implementation on Contiki OS is as follows:

- Maximum Segment Size (MSS): the default value of MSS in Contiki is 48 bytes. Fragmentation and recovery is not implemented, instead, a short MSS is utilised to send a TCP segment in one IEEE 802.15.4 frame.

[3] www.sics.se/contiki

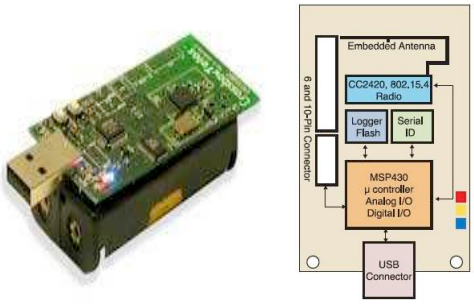


Figure 5.13: Crossbow Telos mote

- Retransmissions: driven by the periodic TCP timer. Since there is not any buffer to remember the packets sent previously, Contiki OS requires that the application takes an active part in performing the retransmissions.

- Flow Control: in uIPv6, the application cannot send more data than the receiving host can buffer.

- Congestion Control: there is no congestion control mechanism implemented on Contiki OS because uIPv6 can handle only one in-flight TCP segment per connection.

5.5.4 Energy consumption

Contiki OS provides a tool to compute the running time spent by a node on the *transmit* and *listen* radio states. Moreover, the Contiki OS provides an estimation of its CPU consumption.

Table 4.2 shows the energy power values, that have been obtained from CC2420 Datasheet and the Texas Instruments MSP430 Datasheet[4]. Based on those values, we are able to compute the consumed energy.

For example, the consumed energy E_{Listen} due to channel listening is equal to

$$E_{\text{Listen}} = T_{\text{Listen}} \times \text{Voltage} \times I_{\text{Listen}} \tag{5.1}$$

where T_{Listen} and I_{Listen} are respectively the time spent by a mote in listen mode and the listen current.

[4]http://focus.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=msp430f1611

5.6 Results and Discussion

In this section, we describe the TCP scenario in detail. A TCP source sends 48 kbytes of data to a TCP receiver. The maximum segment size is equal to 48 bytes, thus the TCP source sends 1000 TCP data segments to the receiver.

We design two kinds of experiments: loss-free scenario and lossy scenario. In our scenario, one node is a TCP source and the others are relay nodes. With respect to Figure 5.12, the routing is such that for a *i*-hops scenario, N_i is the destination node and nodes N_1 to N_{i-1} (in that order) relay TCP segments from ER to N_i.

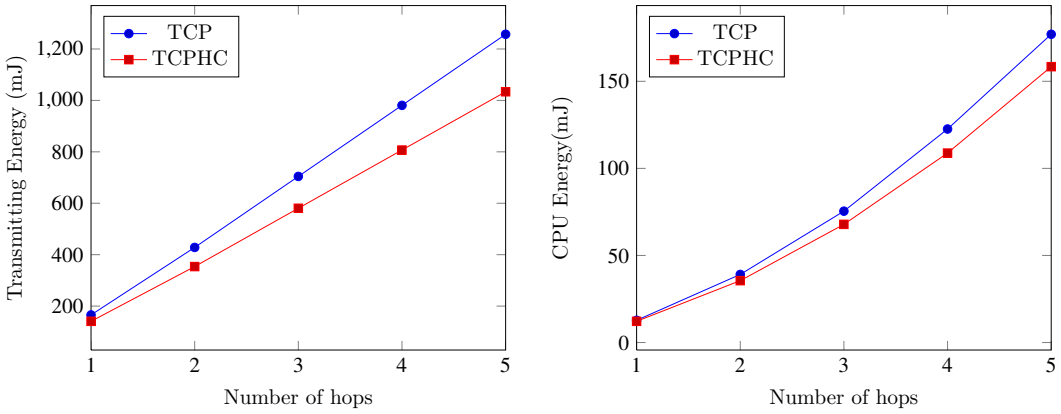
In the experiment, we are mainly interested in the energy consumed by the CPU, the radio transmission, and the radio listening during the TCP connection. We are also interested in the number of segment retransmissions and throughput.

We study the TCP performance in a multi-hop scenario from one to five hops in terms of both throughput and energy consumption. In our experiment, there are two traffics, one is a TCP connection, the other is a constant bit rate (CBR) traffic generated by a UDP traffic to increase the contention and to increase the bit error rate.

5.6.1 TCPHC performance in loss-free environments

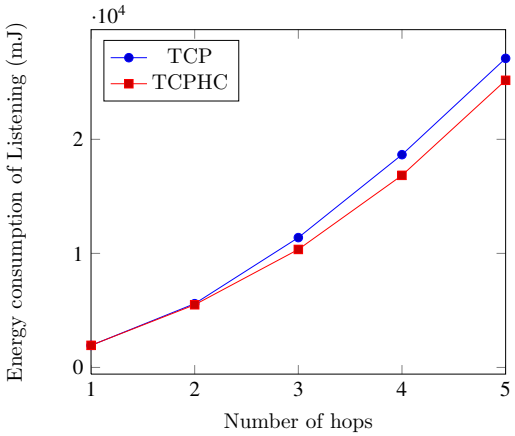
In a first scenario, we compare legacy TCP to TCP with header compression, in terms of transmitting energy consumption and transfer time for different numbers of hops. Figures 5.14 show the average and confidence interval of 10 runs. Figure 5.14 (e) shows that the transfer time decreases when TCPHC is deployed. TCPHC reduces the transfer duration by about 9% compared to legacy TCP, and thus increases the TCP throughput. Figure 5.14 (a) shows that TCPHC reduces the transmissions energy consumption. The TCPHC reduces by about 17% the energy consumed by all sensor nodes when there are 5 hops between the TCP sender and the TCP receiver.

Figure 5.14 (b) shows that the header compression algorithm does not increase the consumed CPU energy. On the contrary, the CPU energy has been reduced by TCPHC. This is due to the nature of our compression algorithm that does not add instructions requiring a large computing time. Figure 5.14 (c) shows that TCPHC does not reduce significantly the passive listening of the radio channel because 90% of this energy is dissipated by the MAC and PHY layers in a passive listening of the channel. Finally, Figure 5.14 (d) confirm previous results and shows that TCPHC reduces the total energy compared to the Contiki version of TCP.

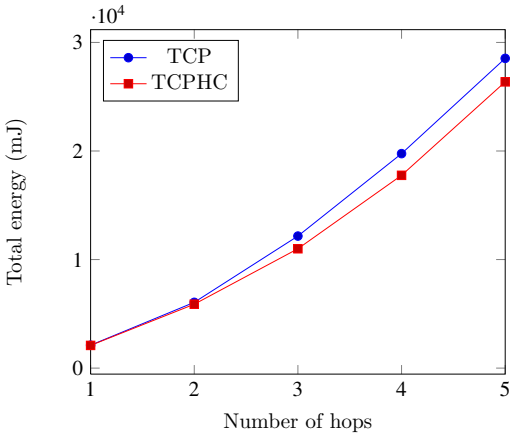


(a) Transmit Energy

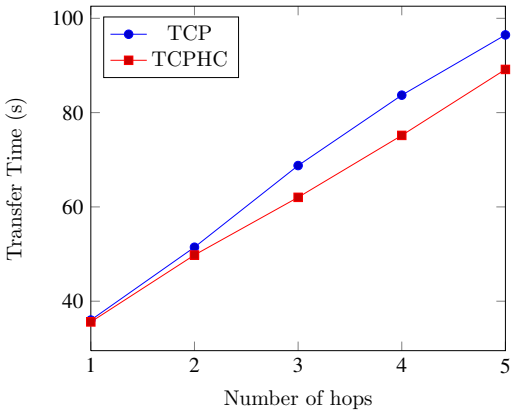
(b) CPU Energy



(c) Listening Energy



(d) Total Energy



(e) Transfer Time

Figure 5.14: Experimental results of multi-hop TCP vs. TCPHC over 6LoWPAN without concurrent CBR traffic.

5.6.2 TCPHC performance in lossy environments

Now, we compare legacy TCP to TCP with header compression in a lossy network. We add a new traffic generated by the External Node with the purpose of increasing the collisions and thus the packet error rate. To compare TCP performance with and without the header compression algorithm, we plot the energy consumption, the TCP end-to-end retransmission and transfer time with different number of hops between the TCP sender and TCP receiver. Figure 5.15 (f) shows that the transfer time decreases when TCPHC is deployed. The header compression algorithm reduces the transfer duration by about 15% compared to legacy TCP. The transfer time is reduced due to the short size of the new segments, which require less time to be sent and received. Moreover, Figure 5.15 (b) shows that TCPHC reduces the number of TCP end-to-end retransmissions compared to legacy TCP because the PER of the compressed segments is lower than PER of the normal segments. Figures 5.15 (a),(c) and (d) are similar to that observed in Figure 5.14 which confirm that TCPHC reduces the CPU and listening energy not only of loss-free environment but also in lossy environment. Finally, Figure 5.15 (e) shows that the header compression algorithm reduces the total energy consumption. As discussed in Figure 5.14, we do not observe a significant improvement when the number of hops between the two TCP hosts is less than or equal to two. However, the TCPHC reduces by about 15% the energy consumed by all sensor networks when there are five hops between the TCP sender and the TCP receiver. All those results show that TCPHC is a very interesting scheme for making TCP more energy-efficient and viable for low-power networks.

5.7 Conclusion

This chapter has presented experimental results of a TCP header compression algorithm over 6LoWPAN in a multi-hop scenario. Experimental results have shown that the larger the distance between the TCP sender and the edge router, the more header compression improves energy efficiency.

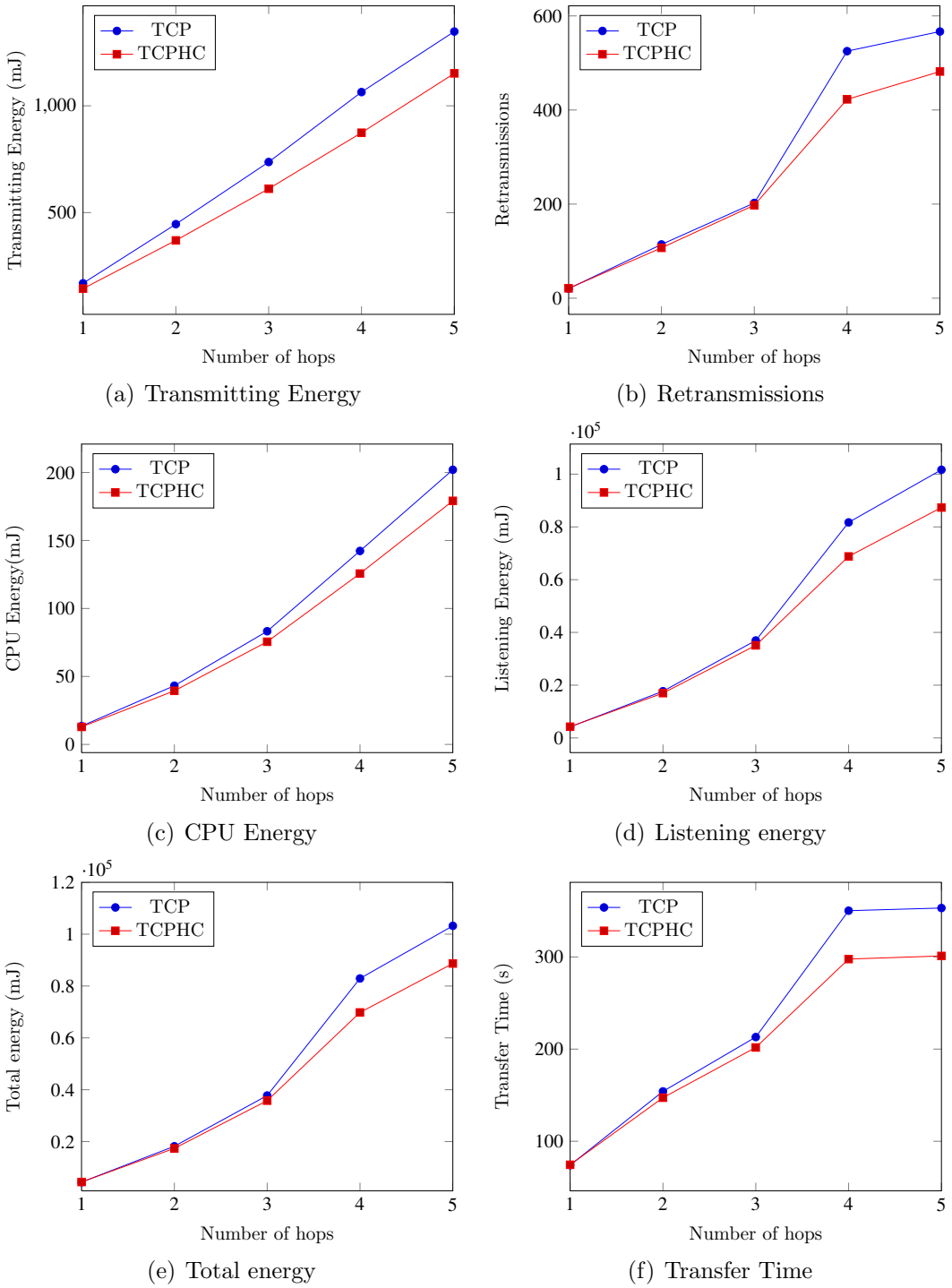


Figure 5.15: Experimental results of multi-hop TCP vs. TCPHC over 6LoWPAN with a concurrent CBR traffic.

Part III

Impact of fragmentation on energy consumption

Chapter 6

Impact of link layers fragmentation on the TCP energy consumption

Contents

6.1 Introduction 78

6.2 Related Work 79

6.3 TCP energy consumption model 79

6.3.1 Link layer: one-hop model 81

6.3.2 Multi-hop model 84

6.3.3 TCP performance 86

6.4 Results and discussion 88

6.4.1 Model assessment 89

6.4.2 FEC redundancy ratio and energy consumption 90

6.4.3 Selecting the TCP MSS to minimize energy consumption 91

6.5 Conclusion 92

6.1 Introduction

This chapter focuses on the energy cost of reliability when TCP is used in multi-hop wireless low power networks. In what follows, we will illustrate some main issues by considering the case of TCP in IPv6-enabled low power networks based on the 6LoWPAN protocols [KMS07].

Many parameters affect the energy consumption of a reliable transport protocol, like channel conditions (e.g., wireless losses, collisions), the level of link-layer reliability, the number of links/hops, or the maximum size of link-layer frames (e.g., 127 bytes for IEEE 802.15.4). The length of such frames may require the fragmentation of IPv6 datagrams before sending them. The 6LoWPAN layer compresses the IPv6 header and fragments IPv6 packets into short MAC frames. So, if the size of a TCP segment exceeds the maximum allowed length, the 6LoWPAN layer fragments it so it fits into small link-layer frames. Therefore, sending short TCP segments does not require any fragmentation. However, the use of a small maximum segment size(MSS) increases the number of TCP segments and, thus, the number of TCP acknowledgements and the corresponding MAC frames sent. Besides, with small MSS values the overhead due to TCP headers becomes larger; this, coupled with the small size of MAC frames, may result in very low protocol efficiency.

This chapter introduces a mathematical model aimed at predicting the energy consumed by the wireless nodes of a low power network in a bulk-data transfer scenario, with TCP used as a reliable transport layer. The model estimates TCP energy performance based on the bit error rate, the maximum number of retransmissions at the link layer, the number of hops between the sender and the receiver, the amount of Forward Error Correction (FEC), and the TCP maximum segment size.

The proposed model allows us to study the tradeoffs involved in sending short versus long TCP segments. We assume that the energy consumed in a data transfer depends mainly on the number of bits sent. Thus, our analytical model estimates the number of bits sent by all nodes in the network, taking into account the cumulated cost of all the link layer transmissions. Indeed, the number of MAC frames sent can be large with respect to the initial amount of data to transmit, because of link-layer retransmissions, the redundancy added for Forward Error Correction, and also due to end-to-end (i.e., transport layer) retransmissions when a TCP segment is lost before reaching the receiving node. We apply the model to study the energy efficiency of TCP over a low power network using 6LoWPAN and 802.15.4 protocols, and to study the effect of the TCP segment size, of the FEC redundancy ratio, and of the maximum link layer retransmission attempts on the total energy consumption.

The remainder of the chapter is organized as follows. Section 6.2 presents a short description of related work on modeling TCP energy consumption in wireless networks. Section 6.3 presents the derivation of our analytical model. In Section 6.4, we apply

the analytical results to derive the best TCP segment size strategy in terms of energy consumption.

6.2 Related Work

In [LS02], Lilakiatsakum and Senevirane propose an energy-efficiency metric to compare the performance of different versions of TCP. The metric is the ratio between the amount of bits sent by all nodes and the size of the application data. In this chapter, we adopt a variant of that metric (the total amount of bits sent); we use a fixed value for the application data size in all our numerical and simulation scenarios.

Bansal et al. [BSGM06] propose an analytical model to compute the energy consumed for carrying a TCP flow over a multi-hop wireless network. The authors assume that all wireless links have the same packet error rate and the same transmission power. They compute the energy spent by all nodes for sending a single TCP segment. The energy consumption is then a function of the packet error rate, the number of hops, and the maximum number of link-layer retransmissions. However, [BSGM06] does not take into account the cost of sending link-layer acknowledgements, nor the cost of transport-layer acknowledgements, which we add in the model introduced here. Besides, we explicitly consider the impact of lower-layer fragmentation and the impact of error correction, as well as different per-link error rates on the energy-efficiency of TCP.

Barman et al. [BMAA04] and Gallucio et al. [GMP03] present an analytical study of a TCP optimization problem in a hybrid wired/wireless network where the last hop is a wireless link. The authors of both papers define a utility function, which is the ratio of the throughput to the cost of a TCP connection. Our work completes these studies by introducing a multi-hop model for computing the energy consumption. Note that in this chapter we are not interested in the TCP throughput, because data rates are a secondary concern in low power networks; instead, we focus mainly on the energy costs of a TCP connection over multiple lossy links.

6.3 TCP energy consumption model

This section introduces a mathematical model to estimate the energy consumed by a TCP transmission in a wireless low power network. In order to simplify the model, we assume that such energy mainly corresponds to data emission and reception, and thus directly depends on the number of bits sent by all nodes.

We therefore compute the expected total amount of bits sent for a successful end-to-end TCP data transmission, as a function of several network parameters, namely: the bit error rate, the link-layer maximum number of attempts, the FEC redundancy ratio, the number

of hops between the source and receiver TCP hosts, and the TCP maximum segment size (MSS).

For the convenience of the reader, Table 6.1 lists most of the variables used in this chapter.

Table 6.1: Notations used in this chapter; capital italics letters correspond to probabilities, bold letters to (expected) numbers of bits.

Variable	Definition
h	Number of hops between source and destination
r	Maximum number of transmission attempts at the link layer
m	Number of fragments corresponding to a single TCP segment (due to link layer fragmentation)
α	FEC redundancy ratio
B	Bit error rate
P_{fail}	Probability of a failure in a link-layer transmission attempt of a data frame
P_{partial}	Probability of a link-layer data frame being correctly received and the corresponding acknowledgement being lost
P_{succ}	Probability of a successful link-layer transmission attempt (data+acknowledgement are correctly received)
F	Probability that a destination node does not receive a link layer data frame after r attempts
Q_s	Probability of an end-to-end packet transmission success
Q_f	Probability of an end-to-end packet transmission failure
$\mathbf{D}$	Link-layer data frame size
$\mathbf{A}$	Link-layer acknowledgement frame size
$\mathbf{H}_f$	Expected number of bits sent after r attempts knowing that the (one-hop) transmission has failed
$\mathbf{H}_s$	Expected number of bits sent within r attempts knowing that the (one-hop) transmission has succeeded
$\mathbf{E}_f$	Expected number of bits sent for an end-to-end packet transmission knowing that it has failed
$\mathbf{E}_s$	Expected number of bits sent for a successful end-to-end packet transmission knowing that it has succeeded
$\mathbf{S}$	Average number of bits sent for successfully transmitting a TCP segment

6.3.1 Link layer: one-hop model

We first focus on modeling the link-layer energy consumption, considering a CSMA-CA network with error correction control techniques combining Automatic Repeat reQuest (ARQ) [FW02] and Forward Error Correction (FEC).

6.3.1.1 Link layer mechanisms

Standard ARQ uses the cyclic redundancy check (CRC) error-detecting code that is added to the data: The receiver uses the error-detecting code number to check the integrity of the received data. After receiving a correct frame, the receiver replies by an ACK. If the sender does not receive any ACK before a timeout[1]— because either the original message or the ACK is lost, or they contain errors—, the sender retransmits again the same message. If the receiver sees that the frame is damaged, the receiver discards it and does not send an ACK. The ARQ algorithm continues until the sender receives an ACK or exceeds a predefined number of attempts r.

ARQ is the algorithm most frequently used by link-layer protocols to reduce the packet error rate. However, if the wireless network becomes very lossy, ARQ would increase the transmission delay between the source and the receiver. Abrupt increases of the end-to-end delay increase the round-trip time and may lead to a spurious TCP timeout. This can deteriorate the TCP performance.

An orthogonal approach consists in applying Forward Error Correction (FEC). FEC is a good solution for decreasing the packet error rate. The main idea of FEC is to add redundancy to the original frame, to allow the destination node to detect and correct some bit errors. In our case, if the size of a network datagram is greater than the maximum transmission unit (MTU) of the link layer, the datagram is divided into fragments of length $\mathbf{K}$ bits, and the FEC algorithm adds $(\alpha \times \mathbf{K})$ redundancy bits to form a frame of length $\mathbf{D}$. The ratio $\alpha = \frac{\mathbf{D}-\mathbf{K}}{\mathbf{K}}$ between the amount of redundancy added by FEC and the original frame length is called the *redundancy ratio*.

In what follows, we will consider an error-correction method like the well-known Reed-Solomon [RG60] (RS) algorithm. By adding $\mathbf{D}-\mathbf{K}$ bits to the $\mathbf{K}$ data, the RS algorithm can correct up to $(\mathbf{D}-\mathbf{K})/2$ bits.

6.3.1.2 Performance of one-hop transmissions

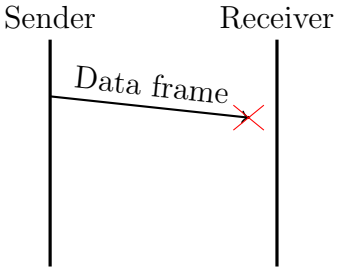
First, we only consider here a transmission between two immediate neighbors, without intermediate nodes. In CSMA-CA, two types of messages are used during data transmission: the data message (i.e., the message that contains the useful data), and the acknowledgement message sent by the receiving node. Figure 6.1 shows the three possible cases:

[1]Remark that we do not consider time issues in this model, thus, only losses can lead to retransmissions.

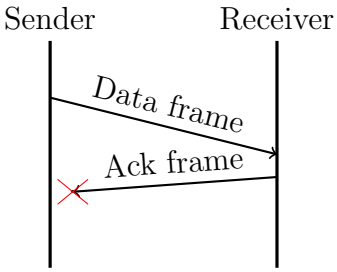
a. *Failure:* A failure due to the loss of the data frame. The sender will retransmit the frame.

b. *Partial failure:* The data frame is correctly received by the receiver, while the acknowledgement frame is lost[2]. Therefore, the sender will (needlessly) retransmit the data frame.

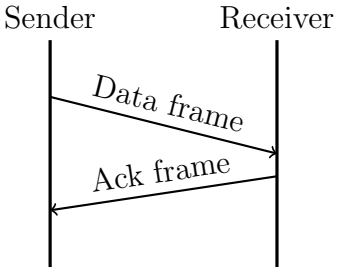
c. *Success:* A successful transmission of both the data and the acknowledgement frames.



(a) Failure.



(b) Partial failure.



(c) Success.

Figure 6.1: Failure and success scenarios for one link-layer transmission attempt.

The probability of a successful transmission of a link-layer frame depends on the bit error rate, the size of the message, and the redundancy ratio. Moreover, we assume that

[2]In a multi-hop setting, the receiver would nonetheless relay the frame to the next hop.

a frame transmission fails as soon as the number of bits erroneously received exceeds the number of correctable bits. The probability P_{fail} of having a *failure* (case (a) in Figure 6.1) is then the error probability for a data frame, i.e.,

$$P_{\text{fail}} := 1 - \sum_{i=0}^{c} \binom{\mathbf{D}}{i} B^i (1-B)^{\mathbf{D}-i}$$

where $\mathbf{D}$ is the number of bits of a data frame, $c = \frac{\mathbf{D}-\mathbf{K}}{2} = \frac{\alpha \mathbf{K}}{2}$ is the number of correctable bits, and for two integers i and j, $\binom{j}{i}$ is the number of possibilities of choosing i elements among j, i.e., $\binom{j}{i} = \frac{j!}{i!(j-i)!}$.

Likewise, denoting by $\mathbf{A}$ the size of an acknowledgement frame, the probability of a *partial failure* is the probability that the data frame is correctly received but the acknowledgement frame contains errors:

$$P_{\text{partial}} := (1 - P_{\text{fail}})(1 - (1-B)^{\mathbf{A}}),$$

whereas the probability of a *success* is

$$P_{\text{succ}} := (1 - P_{\text{fail}})(1-B)^{\mathbf{A}}.$$

Note that we assume that acknowledgement frames are sent *without* adding redundancy. This is a reasonable assumption, given the current practices in most wireless technologies.

In order to estimate the energy consumption, we compute the expected number of bits sent at the link layer. Remark that in cases of success or partial failure, $\mathbf{D}+\mathbf{A}$ bits are sent, whereas only $\mathbf{D}$ bits are sent in cases of failure. Recall however that the redundancy ratio determines the number of useful bits per frame, and thus the number of frames to send.

Now, we take into account the link-layer sending attempts, following the ARQ technique implemented by the MAC protocol. We denote by r the maximum number of attempts, after which the sender of a data frame considers the receiver is unreachable.

The *reception* failure probability of the data frame after r attempts is simply the probability F of having r successive failures, i.e.:

$$F := P_{\text{fail}}^r. \tag{6.1}$$

In that case, the total number of bits sent only comes from the sender (no acknowledgement is sent), and thus equals

$$\mathbf{H}_f := r \times \mathbf{D}. \tag{6.2}$$

With the probability $1 - F$, the receiver gets the data frame within the r link-layer attempts. In that case, the total number of bits sent depends on the number of failures and partial failures before a success, if any (we only know here that at least one attempt led to a success or a partial failure). There are two possibilities:

- either all r attempts were failures or partial failures, but at least one of them was a partial failure (since we are in the case where the receiver got the data),
- or the last of the $k \leq r$ attempts was a success (in the sense of case (c) in Figure 6.1).

Conditioned on the receiver getting the data frame within the r link-layer attempts, the expected total number of bits sent $\mathbf{H}_s$ can therefore be computed as follows:

$$\begin{aligned}
\mathbf{H}_s &= \frac{1}{1-F}\Big(\sum_{i=1}^{r}\binom{r}{i}P_{\text{partial}}^{i}P_{\text{fail}}^{r-i}(r\mathbf{D}+i\mathbf{A}) \\
&\quad + \sum_{k=1}^{r}P_{\text{succ}}\sum_{i=0}^{k-1}\binom{k-1}{i}P_{\text{partial}}^{i}P_{\text{fail}}^{k-1-i}(k\mathbf{D}+(i+1)\mathbf{A})\Big) \\
&= \frac{1}{1-F}\Big(r\mathbf{D}((P_{\text{fail}}+P_{\text{partial}})^{r}-P_{\text{fail}}^{r}) + r\mathbf{A}P_{\text{partial}}(P_{\text{fail}}+P_{\text{partial}})^{r-1} \\
&\quad + P_{\text{succ}}\sum_{k=1}^{r}(k\mathbf{D}+\mathbf{A})(P_{\text{fail}}+P_{\text{partial}})^{k-1} + \mathbf{A}\Big(\sum_{i=0}^{k-1}\binom{k-1}{i}P_{\text{partial}}^{i}P_{\text{fail}}^{k-1-i}i\Big)\Big) \qquad (6.3)
\end{aligned}$$

In (6.3), k stands for the index of the first success in r attempts (if any), and i is the number of partial failures among all attempts.

6.3.2 Multi-hop model

Let us now focus on the multi-hop case. An end-to-end transmission succeeds if the message reaches the destination after a certain number h of hops.

In this chapter, we assume that link layer transmissions on each hop are independent. We denote B_i the bit error rate and F_i the frame error rate of the i^{th} hop. F_i is computed as per (6.1), taking $B = B_i$. Therefore, the probability that a frame is correctly received by a destination node is simply the probability Q_s that all h one-hop transmissions succeed, i.e.,

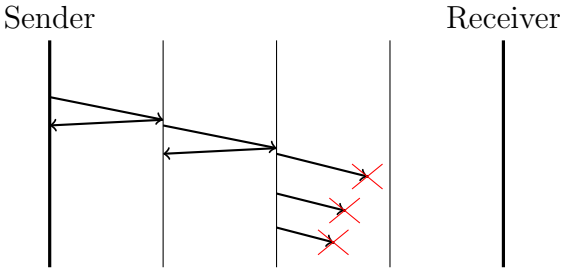
$$Q_s = \prod_{i=1}^{h}(1 - F_i) \qquad (6.4)$$

where h is the number of hops from the sender to the receiver.

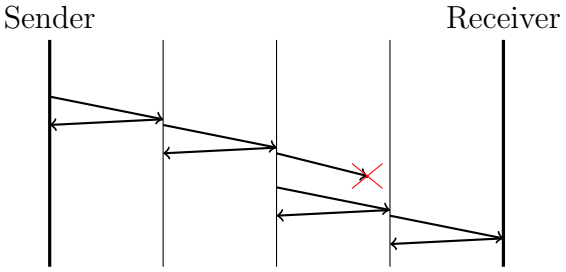
We assume here that the MAC layer is able to detect duplicate frames, such as those that are produced in case of a partial failure (case (b) in Figure 6.1). This can be implemented,

for instance, by using a sequence number in the MAC frame headers; when a node receives from one neighbor two successive frames with the same sequence number, it assumes that the corresponding Ack frame was lost and deletes the second frame. This avoids the propagation of several copies of the same data frame.

Figure 6.2 shows the two possibilities for the outcome of the end-to-end transmission of a frame.



(a) End-to-end failure scenario: the frame cannot be forwarded after r unsuccessful retransmissions.



(b) End-to-end success scenario: the frame arrives at the destination. This scenario may also include partial failures over one or more hops (not depicted).

Figure 6.2: Failure and success scenarios in a multi-hop transmission.

Again, we express the expected number of bits sent, conditioned on the success of the end-to-end transmission (encompassing possible link-layer retransmissions, within the limit of r total attempts per hop).

- Knowing that the destination node correctly receives the message, the expected number of bits sent by all network nodes is simply

$$\mathbf{E}_s := \sum_{i=1}^{h} \mathbf{H}_{s_i}. \tag{6.5}$$

- Knowing that the message was lost in one of the h hops, the number of bits sent

depends on the hop where the loss (i.e., the failure of all r attempts) occurs. The expected value then equals

$$\mathbf{E}_f := \frac{\sum_{k=1}^{h}(\sum_{i=1}^{k-1}\mathbf{H}_{s_i} + \mathbf{H}_{f_k})\prod_{j=1}^{k-1}(1-F_j)F_k}{1-Q_s} \tag{6.6}$$

where $\mathbf{H}_{s_i}$ and $\mathbf{H}_{f_i}$ are computed using (6.3) and (6.2), respectively.

6.3.3 TCP performance

We now study the energy consumption of TCP in a multi-hop wireless network, taking into account the fact that TCP segments may be fragmented by lower layers if the total size of data frames (that depends on the TCP maximum segment size (MSS)) exceeds the MTU of those layers.

We can intuitively think of several opposite effects of the MSS, which suggest that a trade-off has to be found:

i) If TCP segments are fragmented, the loss of one fragment leads to the loss of the original TCP segment and therefore a new TCP end-to-end retransmission. Further, using short TCP segments (i.e., a small MSS) saves CPU power associated to fragmentation and reassembly. Moreover, sending several fragments (treated independently by lower layers) increases the probability of collision between two TCP data fragments, due to the hidden-node problem. As an illustration of that phenomenon, one can refer to Figure 6.4, provided later on (see Section 6.4.1), where the number of collisions with long TCP segments is much larger than when short TCP segments are used.

ii) On the other hand, the use of long TCP segments reduces the number of TCP segments sent by the source, and thus, the TCP header overhead (which can be very large, for small MTU values) and the number of TCP acknowledgements.

To analyze such trade-off, we now apply the model presented in Section 6.3.2. Our objective here is to determine an MSS choice that optimizes TCP performance, in terms of energy consumption.

Let us consider that the MSS of TCP and the MTU of the MAC layer are such that each TCP segment is fragmented into m link-layer frames. This encompasses the particular case when no fragmentation occurs, that simply corresponds to $m = 1$. We will make the assumption that TCP ACKs always fit in a single MAC frame, though it would be easy to consider a more general case in which TCP ACKs are also fragmented.

To simplify the analysis, we assume that all fragments of a TCP data segment have the same total size $\mathbf{D}_{\text{DATA}}$, and all TCP acknowledgement frames have the same total

size $\mathbf{D}_{\text{ACK}}$. At the destination, the segment is reconstructed from the received fragments. The loss of a single link-layer frame induces the loss of the whole TCP segment, and thus the retransmission of all m fragments. Further, we consider that the number of TCP retransmissions is not limited; that is, the TCP source keeps on sending a segment until it receives a TCP acknowledgement from the TCP receiver.

From the previous analysis, the success probability P_s of a TCP segment transmission attempt is simply the probability that all m data fragments be correctly sent to the destination, and the TCP ACK be successfully sent back to the source:

$$P_s = Q_s^m \times Q_{s,\text{ack}} \ ,$$

where Q_s and $Q_{s,\text{ack}}$ denote the success probability of the multi-hop transmission of a TCP data fragment and of a TCP acknowledgement frame, respectively. Q_s and $Q_{s,\text{ack}}$ are simply obtained by applying (6.4), but replacing $\mathbf{D}$ by respectively $\mathbf{D}_{\text{DATA}}$ and $\mathbf{D}_{\text{ACK}}$.

In our model, each TCP data fragment transmission succeeds or fails independently of the others. Knowing that a transmission is successful *at the TCP level* (i.e., the TCP ACK is correctly received by the TCP source, which implies that all m fragments correctly reached the destination), the expected total number of bits sent by all nodes equals:

$$\mathbf{S}_s := \mathbf{E}_s \times m + \mathbf{E}_{s,\text{ack}}$$

As above, $\mathbf{E}_s$ and $\mathbf{E}_{s,\text{ack}}$ are obtained from (6.5) using $\mathbf{D}_{\text{DATA}}$ and $\mathbf{D}_{\text{ACK}}$, respectively, as the size of a frame. Likewise, we also define $\mathbf{E}_{f,\text{ack}}$ from (6.6).

Knowing that a TCP transmission attempt has failed, the expected number of bits sent end-to-end by all nodes is:

$$\mathbf{S}_f := \frac{1}{1-P_s}\Big[\underbrace{\mathbf{I}_f(1-Q_s^m)}_{\substack{\text{end-to-end transmission failure} \\ \text{of one or more of the } m \text{ fragments}}} + \underbrace{(\mathbf{E}_s \times m + \mathbf{E}_{f,\text{ack}})Q_s^m(1-Q_{s,\text{ack}})}_{\text{end-to-end transmission failure of the TCP ACK}} \Big],$$

where $\mathbf{I}_f$ is the expected total number of bits sent for the m fragments to reach the destination, knowing that they (i.e., at least one) finally fail. Formally,

$$\mathbf{I}_f := \sum_{k=1}^{m} \binom{m}{k} \Big(k\mathbf{E}_f + (m-k)\mathbf{E}_s\Big)(1-Q_s)^k Q_s^{m-k}.$$

After some algebra, we get

$$\mathbf{I}_f = m(1-Q_s)\mathbf{E}_f + m\mathbf{E}_s Q_s(1-Q_s^m).$$

To simplify the analysis, we will assume that the TCP window is equal to one TCP seg-

ment. This assumption is justified because a small window is a sensible choice for networks with a moderate number of hops [FLZ+05]. Moreover, such small windows are typical of current TCP implementations for low power networks (e.g., Contiki OS [DGV04]), since the memory and CPU constraints of wireless embedded devices make it difficult to fully implement TCP's congestion control mechanisms.

We can now compute the total number of bits that have to be sent by all nodes, to successfully transmit both a TCP segment and its corresponding TCP ACK. Since we assumed the number of TCP retransmissions is unbounded (i.e., the TCP sender keeps on retrying until the transmission succeeds), the mean number of transmissions for a given TCP segment equals $1/P_s$. This therefore corresponds to a total number of bits sent (per segment) of

$$\mathbf{S} := \mathbf{S}_f(1/P_s - 1) + \mathbf{S}_s.$$

Finally, to successfully send a given amount M of *application* ("useful") data, the expected number of bits sent by all wireless nodes is $\mathbf{S} \times \lceil M/\mathrm{MSS} \rceil$. It is that final value that will be considered as representative of the overall energy consumption of the TCP transmission.

6.4 Results and discussion

We will now compare the predictions of our analytical model to OMNET++ simulation results, and discuss the tradeoff between sending long or short segments in different scenarios.

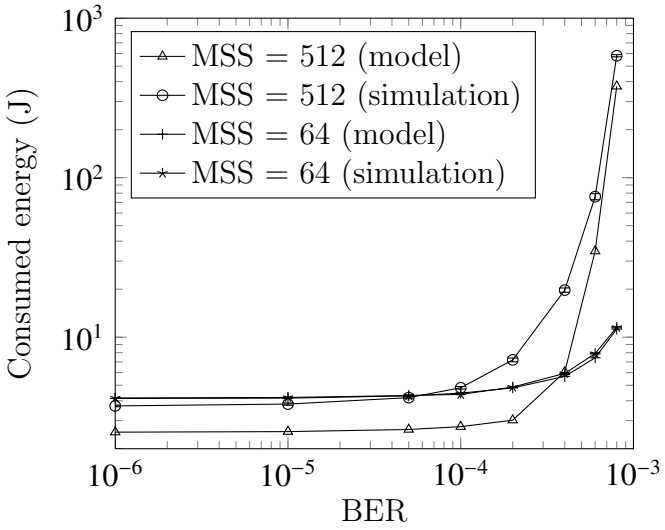


Figure 6.3: Energy consumption with long or short TCP segments, as a function of the BER B (with $r = 3$).

We apply our model to study the energy consumption of TCP on IPv6 over Low-Power Wireless Personal Area Networks [KMS07], considering IEEE 802.15.4 [IEE06] as the link-layer technology. The maximum link-layer frame size is thus equal to 127 bytes. The 6LoWPAN layer adapts the IPv6 datagrams to the link-layer MTU, as explained in [KMS07].

We consider two MSS choices (MSS=64 and MSS=512 bytes) for a given TCP session. When the MSS is 64 bytes, no fragmentation is performed by the 6LoWPAN layer, whereas in the case where MSS=512 bytes, this adaptation layer splits each segment into 8 frames.

6.4.1 Model assessment

Table 6.2: Default simulation parameters

Parameter	**Value**
h	5
r	3
α	0
BER B	3×10^{-4}
Link-layer Ack frame size	40 bits
Link-layer data frame header	120 bits
IP header	160 bits
TCP header	160 bits

We first validate the results of our analytical model, through the simulation of the following scenario. A TCP sender sends a short file (51.2 Kbytes) to a TCP receiver. We consider an average size of 20 bytes for IPv6 headers, thanks to the LOWPAN_IPHC [MKHC07] compression. The simulation results plotted are average values after 30 simulation runs. The TCP window size is set to 1, which implies that no congestion losses occur (we only consider one flow here).

As a result, retransmissions can only be due to transmission errors, as described in Section 6.3. For the sake of simplicity, all links have the same bit-error rate $B_i = B, i = 1, \ldots, h$. Unless indicated otherwise, the parameters used in simulations and in numerical computations correspond to the default values shown in Table 6.2. Finally, as in Section 6.3.3, the maximum number of transport-layer retransmissions is not bounded.

Figure 6.3 shows that analytical results closely match simulations results when a short TCP segment size is chosen. However, we can see that with long TCP segments, the model tends to underestimate the energy consumption. This difference comes from the presence of collisions between fragments of a given TCP data segment, that are not taken into account in the analytical model but do occur in simulation, as illustrated in Figure 6.4. In the

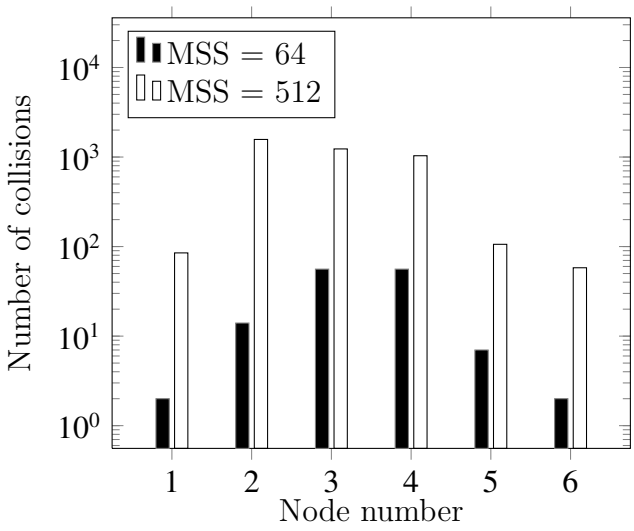


Figure 6.4: Number of collisions in a multi-hop scenario.

figure, nodes 1 and 6 are the source and destination TCP nodes, respectively, and nodes 2 to 5 are intermediate nodes.

Moreover, Figure 6.3 shows that using short TCP segments becomes interesting, from an energy point of view, when the bit-error rate is high (above 10^{-4}). For example, for a BER of 4×10^{-4}, the total consumed energy with MSS = 512 bytes is three times larger than with MSS = 64 bytes. For low values of B, the number of retransmissions is small, hence the energy consumption becomes roughly independent of B.

We remark on Figures 6.3 and 6.5 that simulation results fit better our analytical results when no fragmentation is performed by the link layer. This is again due to collisions that occur in the large-MSS case and are not encompassed by our model. Figure 6.5 also illustrates that increasing the number of link-layer attempts decreases the energy consumption, especially so when there is fragmentation (i.e., a large MSS). Indeed, giving the link layer more chances to send a data frame to its next-hop node reduces the discard probability of TCP segments, and therefore the number of end-to-end retransmissions. In the following, we study the impact of the other system parameters on the energy consumption, based on the model only. We should therefore stay aware that the energy consumption might be a little underestimated for large values of the MSS.

6.4.2 FEC redundancy ratio and energy consumption

We first study the impact of FEC on the energy consumption. Figure 6.6 shows that there seems to be an optimal amount of redundancy, in terms of energy consumption. Below the optimal value, adding redundancy reduces the probability of losses, and thus reduces the

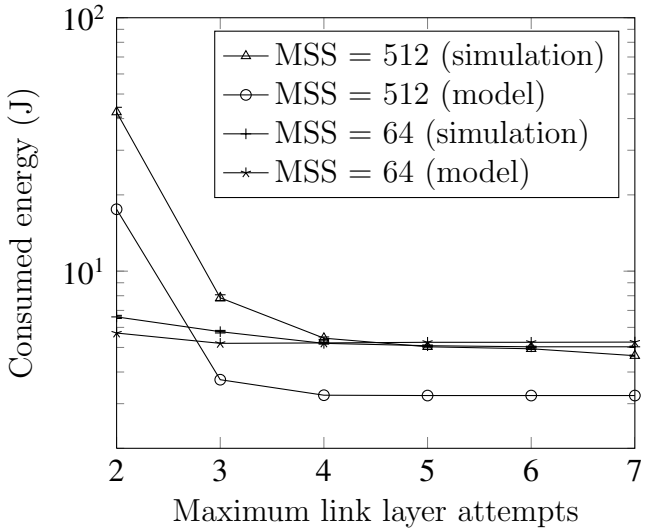


Figure 6.5: Energy consumption with short or long TCP segments, as a function of the number of link layer attempts r (with $B = 5 \times 10^{-4}$).

energy consumption. When α is above such optimal value, however, energy expenditure steadily increases because of the redundancy overhead.

Recall that the MTU being fixed, the number of frames to send depends on the redundancy ratio according to a stair step function, hence the discontinuities in the figure.

6.4.3 Selecting the TCP MSS to minimize energy consumption

Depending of the numerous parameters of a given scenario, it appears that the same MSS value is not always the most efficient one in terms of energy. We now intend to summarize the effect of all parameters, by focusing on the best MSS strategy to implement. In the following figures, we compare the two values $MSS = 64$ and $MSS = 512$, and we concentrate on the boundary values, that delimitate zones where one MSS value outperforms the other.

We plot those frontier curves in Figs. 6.7 and 6.8. In each figure, the area above the curve represents the case where a TCP MSS value of 64 bytes consumes less energy than an MSS of 512 bytes, while the opposite holds below the curves.

Figure 6.7 shows the two zones, depending on the transmission distance and the BER. We remark that for a given BER, short MSSs tend to outperform long MSSs when the distance grows: it is more and more interesting to use short MSS values instead of long ones. Indeed, for large networks the cost of end-to-end retransmissions will exceed the potential economy in terms of TCP header overhead. For a given distance, short MSSs are

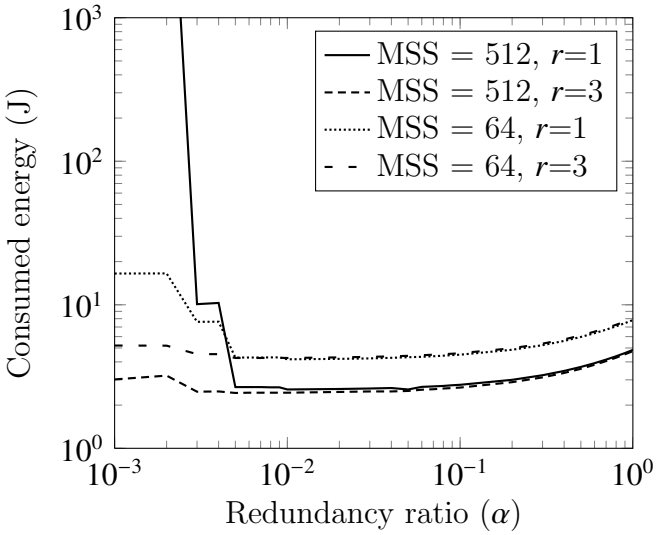


Figure 6.6: Consumed energy using short or long TCP segment, as a function of the redundancy ratio α ($B = 3 \times 10^{-4}, h = 5$).

better suited for large BER environments since (energy-spending) segment retransmissions tend to occur more frequently. As previously observed, increasing the maximum number of link layer attempts r reduces the one-hop transmission failures, and thus limits the effects of errors, hence favoring long MSSs.

Finally, Figure 6.8 illustrates the effect of FEC mechanisms. Not surprisingly (the FEC reducing the effect of transmission errors), redundancy makes large MSSs outperform small MSSs due to the overhead reduction they allow.

6.5 Conclusion

In this chapter, we have proposed an analytical model to estimate the number of bits sent by all wireless nodes in a TCP session in a low power network, in order to evaluate the overall energy consumption. The model has been validated through simulations, using the INETMANET framework [Vag11] of the OMNet++ network simulator [Vag10], in the context of TCP over 6LoWPANs.

Our main outcomes regard the choice of an energy-saving Maximum Segment Size (MSS) for TCP. We have shown that using a large TCP segment size is less energy consuming in small, low-error networks, while it becomes interesting to reduce the MSS when the network is large or very lossy. The impact of the number of attempts at the link layer, as well as the use of FEC, has also been studied.

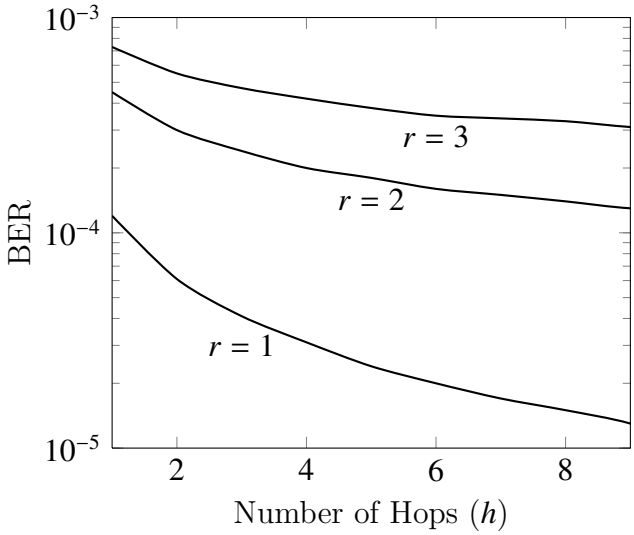


Figure 6.7: Long (MSS=512 bytes) versus short (MSS=64 bytes) in a multi-hop TCP transmission: prefer the short MSS above the curves, the long one below. ($\alpha = 0$)

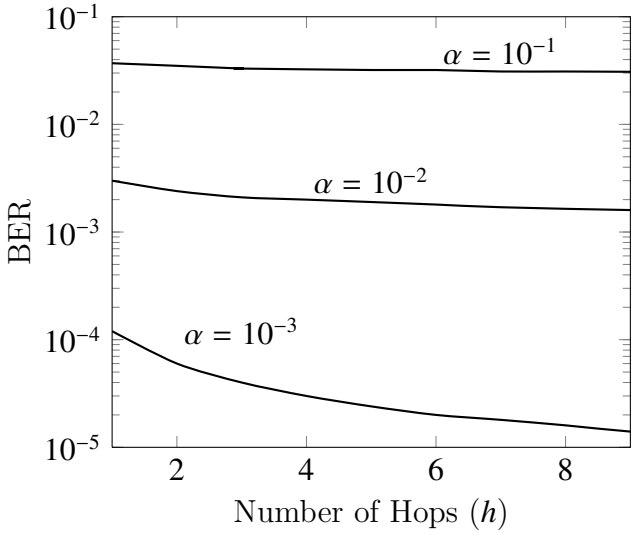


Figure 6.8: Long (MSS=512 bytes) versus short (MSS=64 bytes) in a multi-hop TCP transmission: prefer the short MSS above the curves, the long one below. (with $r = 3$)

Chapter 7

Energy-efficient fragment recovery techniques for low power networks

Contents

7.1 Introduction 96

7.2 The ARQ Error Control Mechanism 96

7.3 Simple Fragment Forwarding and Recovery for 6LoWPANs 97

7.3.1 Fragment Recovery 98

7.3.2 An SFFR scenario 98

7.4 Performance Evaluation and Discussion 100

7.4.1 Impact of SFFR on TCP energy consumption 100

7.4.2 Impact of SFFR on UDP energy consumption 102

7.4.3 SFFR rounds *versus* energy efficiency 104

7.4.4 When is it better to use SFFR? 104

7.5 Conclusion 105

7.1 Introduction

In the previous chapter, we provide a study of the impact of fragmentation on the energy consumption. Results show that it is not energy-efficient to send long packets when the bit error rate is high. A first possible solution is to reduce the transport layer payload to fit within the 802.15.4 maximum transmission unit (MTU). However, this solution adds a significant header overhead to each 802.15.4 frame because all link layer frames include the network and transport headers. A second solution is to use legacy link layer error control mechanisms such as Automatic Repeat reQuest (ARQ) [FW02]. However, if a single fragment is lost in transmission, all fragments may end up being retransmitted, further contributing to the congestion that might have caused the initial packet loss.

Legacy MAC layer error control techniques (e.g., ARQ) provide only one-hop error control solutions. These solutions are not sufficient for multi-hop environments, especially if the link layer does not support long frames. In this case, the IP packet is split into several fragments and the loss of one fragment in one hop leads to the retransmission of all fragments by the transport protocol.

Recently, Thubert and Hui introduced in [TH10] Simple Fragment Forwarding and Recovery (SFFR), a simple protocol to forward and recover individual fragments that might be lost over multiple hops between 6LoWPAN endpoints. SFFR complements hop-by-hop link layer recovery mechanisms for multi-hop environments. This chapter provides a performance evaluation of the simple fragment recovery based on energy consumption for UDP and TCP data transfers.

The remainder of the chapter is organized as follows. Section 7.2 gives a brief overview of the ARQ scheme and its drawbacks. Section 7.3 presents the principles of the SFFR mechanism. Section 7.4 shows our performance evaluation model and results.

7.2 The ARQ Error Control Mechanism

Currently, most error recovery algorithms are implemented in the link layer. In this section, we focus on the most common error recovery and correction techniques (e.g., Automatic Repeat reQuest (ARQ) [FW02, SCM84]).

As described in Section 6.3.1.1, ARQ is the most frequently used algorithm at the link layer to reduce the packet error rate. However, if the wireless network becomes very lossy, ARQ would increase the transmission delay between the source and the receiver. Abrupt increases of the end-to-end delay increase the round-trip time and may lead to a spurious TCP timeout. This can deteriorate the TCP performance.

In a multi-hop low power network, it makes little sense to reassemble the fragments at every hop. In a Mesh-Under low power network (that is, a Layer 2 switched mesh), each

fragment is an individual frame that follows its own-switched path along the mesh. In a Route-Over low power network (that is a Layer 3 routed mesh), the SFFR method that we describe in the next section consists in forwarding the fragments using a datagram tag as a label. In that case, all fragments follow the path found by the first fragment. In all cases, if an intermediate hop fails, a hop-by-hop recovery mechanism such as ARQ cannot reject a fragment back to the source to ask a retry and the loss will not be known at layer 2.

7.3 Simple Fragment Forwarding and Recovery for 6LoWPANs

The failure of successive ARQ attempts leads to the loss of a link layer frame in one of the hops between the source and the destination. In that case, reliable transport protocols such as TCP should retransmit the lost segment. Additionally, the transport protocol may send long segments that can be split by the link layer into small fragments if its MTU is smaller than the size of the network layer datagram. Therefore, the transport segment is sent by the link layer in small fragments, and the loss of one of these fragments leads to the loss of the original segment.

To resolve this point, a new simple end-to-end fragment forward and recovery algorithm has been proposed to allow the receiver to recover intelligently the lost fragments. In [TH10], the authors specify three types of link layer fragments: recoverable fragments (RFRAG), recoverable fragments with acknowledgment request (RFRAG-AR) and acknowledgment fragments (RFRAG-ACK).

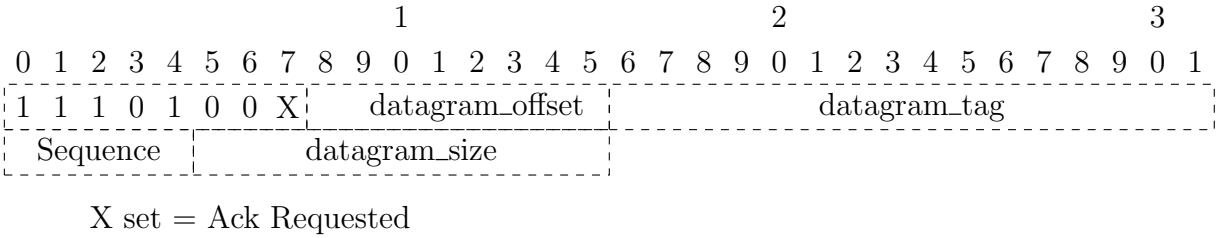


Figure 7.1: Recoverable Fragment Dispatch type and Header

The authors propose to add new fields such as *sequence*, *datagram_tag*, *datagram_offset* to the 6LoWPAN header. Figure 7.1 shows the RFRAG header. Moreover, they suggest adding a compressed acknowledgment bitmap to the acknowledgment. The bitmap can hold 32 bits, which is more than enough to index with a bit in the bitmap each possible

fragment of an IPv6 packet with the 6LoWPAN MTU of 1280 octets. Figure 7.2 shows the RFRAG-ACK header.

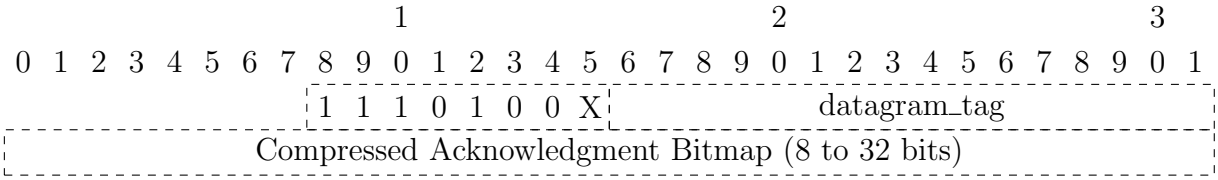


Figure 7.2: Fragment Acknowledgement Dispatch type and Header

7.3.1 Fragment Recovery

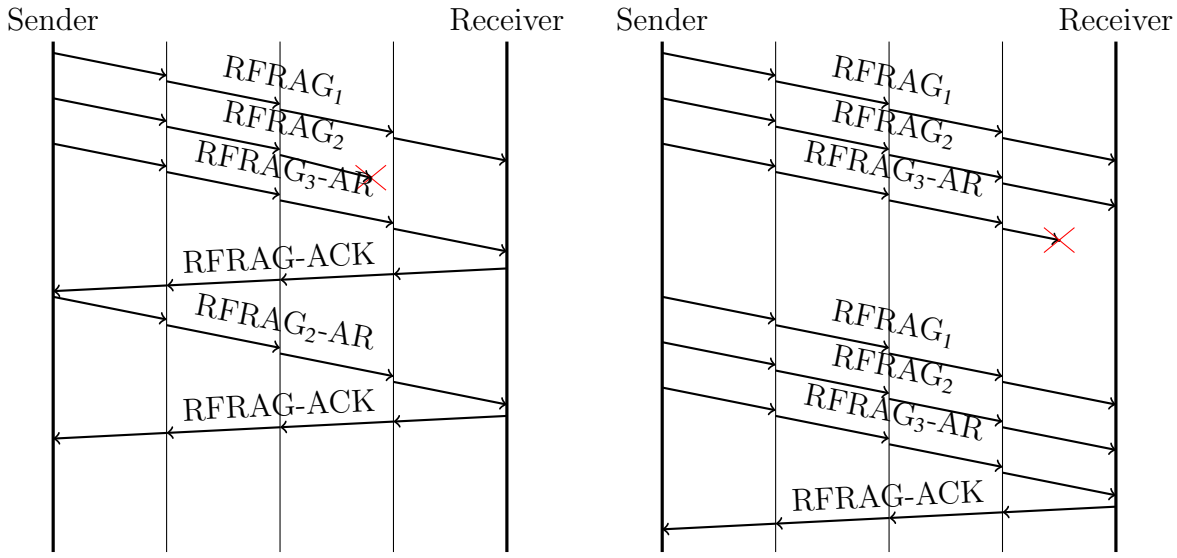
The 6LoWPAN sender controls the Fragment Acknowledgements. When the sender of the fragment knows that an underlying mechanism protects the Fragments it already may refrain from using the Acknowledgement mechanism, and it never sets the Ack Requested bit. The receiver must acknowledge the fragments it has received when it is asked to, and it may slightly defer that acknowledgement.

In the beginning, the sender issues a number of RFRAGs and it may flag the last fragment of a series with an Acknowledgment Request (AR). The receiver must reply by an RFRAG-ACK, with a bitmap that indicates which fragments are received, upon the reception of an RFRAG-AR. The bitmap is a 32-bits which accommodates up to 32 fragments. The bitmap is compressed as a variable length field formed by control bits and acknowledgement bits. For each fragment that was actually received, the corresponding bit is set in the compressed acknowledgment bitmap, so the acknowledgment enables the sender to know which fragments are lost or on the way and must potentially be recovered. This corresponds to one fragment recovery round.

Moreover, fragments are sent in a round robin fashion: the sender sends all the fragments for a first time before it retries any lost fragment; lost fragments are retried in sequence, oldest first. This mechanism enables the fragments on the way to be finally received and acknowledged before the sender decides to retry them. As specified in [TH10], a single round of fragment recovery is recommended, so it fits within the upper layer recovery timers; more than one round frame recovery may lead to a TCP timeout.

7.3.2 An SFFR scenario

Figure 7.3 (a) shows a first scenario of an end-to-end loss recovery using SFFR. Here, an IPv6 packet is split by the link layer protocol into three RFRAGs. The third fragment is sent with AR set. In this scenario, the second fragment ($RFRAG_2$) is lost before it



(a) Scenario 1: An RFRAG is lost

(b) Scenario 2: An RFRAG-AR is lost

Figure 7.3: End-to-end simple fragment forwarding and recovery

reaches the receiver. After receiving the $RFRAG_3$-AR fragment, the receiver replies by sending an RFRAG-ACK informing the sender that the first and the last fragments are correctly received, and requesting the retransmission of the lost fragment. The source sends a new copy of the lost fragment (i.e., $RFRAG_2$) with an acknowledgment request and removes copies of acknowledged fragments from the sending buffer ($RFRAG_1$ and $RFRAG_3$). Finally, the receiver ends the IPv6 packet transmission with an RFRAG-ACK, which acknowledges the reception of all the packet fragments. The source node deploying SFFR must have enough memory to save the sent fragments while waiting for the acknowledgment from the receiver. Figure 7.3 (b) shows a second scenario where the source node does not receive the RFRAG-ACK before the TCP retransmission timeout. In this case, the source node retransmits all fragments.

In this chapter, we assume that the link layer splits the original message into m fragments with the same size $\mathbf{D}/m + o$ where $\mathbf{D}$ is the original packet size and o is the overhead added by SFFR. At the destination, the message is reconstructed from the received fragments. The loss of one fragment makes impossible to reconstruct the original message. In addition, we assume that SFFR recovery rounds are limited to a single round for reliable transport protocols such as TCP. This work studies also the impact of the recovery rounds on the energy-consumption of a datagram transport protocol (e.g., UDP).

7.4 Performance Evaluation and Discussion

In this section, we present a performance study of SFFR in a multi-hop low-power network. The main metric that we take into account in our study is the energy consumption. Numerical results are obtained from the analytical model presented in Chapter 6, where on each radio channel, the bit transmission errors are assumed independent and identically distributed. Simulation results are obtained from OMNET++ simulator [Vag10] used in Chapter 4. The expected number of ARQ trials, as well as the probability of success, can then be computed. Likewise, the probability of end-to-end successful packet transmission and the expected number of bits sent at each end-to-end transmission attempt are calculable.

Table 7.1: Network parameters

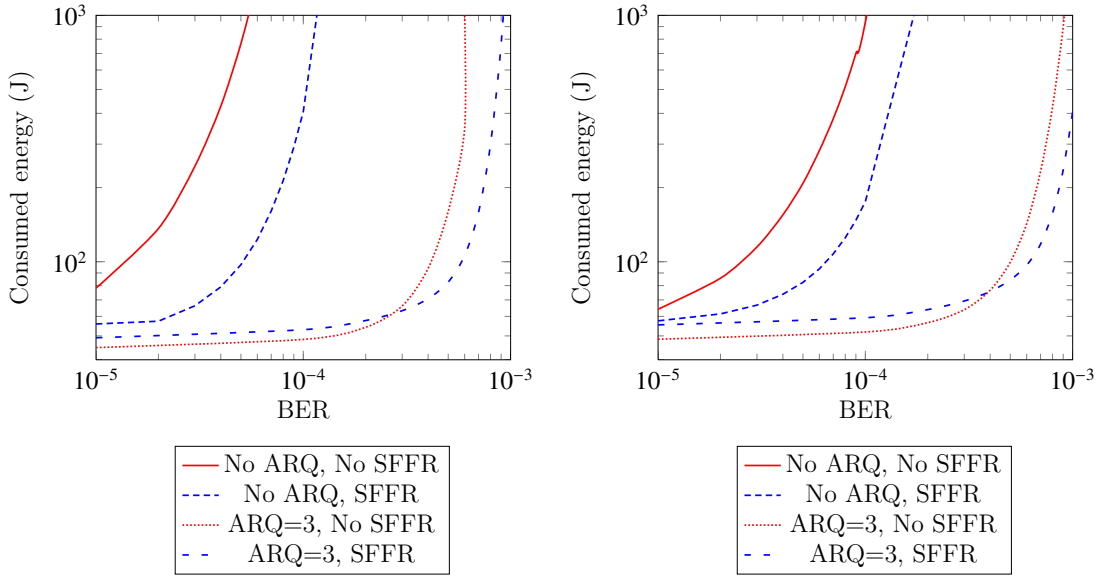
Parameter	**Value**
Hop number	5
Application data size	1048 kbytes
TCP MSS/ UDP payload size	512/1024 bytes
NHC header	1 bytes
TCPHC header	8 bytes
6LoWPAN header	3 bytes
IEEE 802.15.4 header	23 bytes
IEEE 802.15.4 acknowledgment size	10 bytes

We study the energy consumption of UDP and TCP on IPv6 over Low-Power Wireless Personal Area Networks [KMS07], considering IEEE 802.15.4 [IEE06] as the link-layer technology.

In our evaluation, we assume that the UDP header compression described in [HT10] is used to reduce the UDP header to 1 byte and our TCP header compression mechanism [ART10] is also used to reduce the TCP header to 8 bytes. Table 7.1 shows the default parameter values that, unless specified otherwise, have been used in our evaluation.

7.4.1 Impact of SFFR on TCP energy consumption

We first evaluate the SFFR performance in multi-hop wireless networks, through the following scenario. A TCP sender sends a file (1024 kbytes) to a TCP receiver. The number of sent TCP segments depends on the chosen MSS value (e.g., if the MSS value is equal to 1024, the TCP receiver should receive 1000 segments). We choose to study two cases: in the first case no link layer error recovery algorithm is used, and in the second case ARQ with a maximum of three link layer recovery attempts is implemented. We vary the bit



(a) MSS = 1024 bytes

(b) MSS = 512 bytes

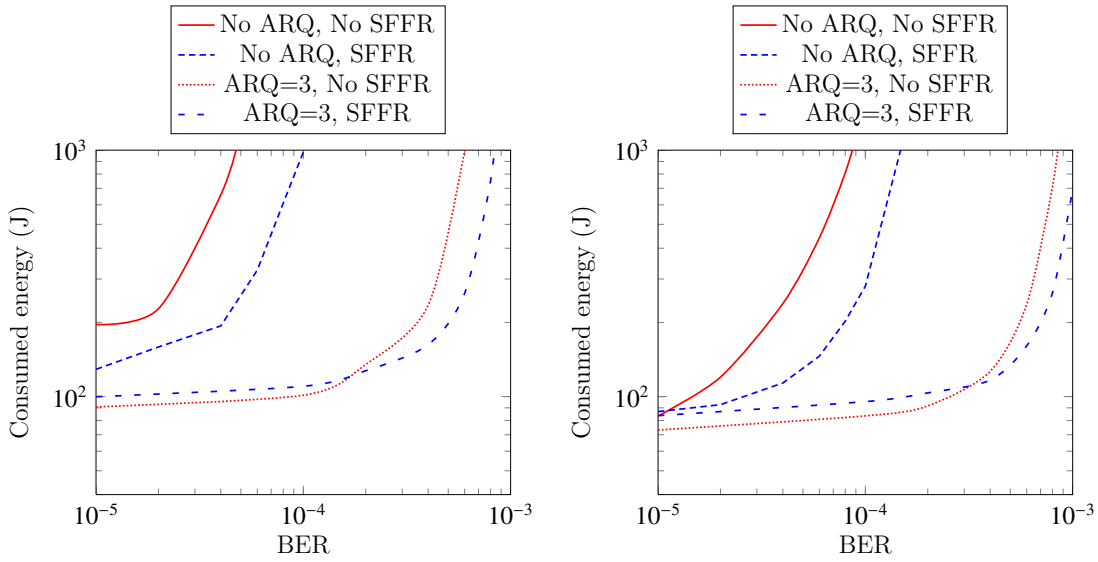
Figure 7.4: **Analytical results:** Energy Consumption of a TCP data transfer with vs without SFFR (scenario with five hops).

error rate, and we compute the consumed energy by all the wireless nodes in two scenarios, with and without SFFR.

Figure 7.4 displays the consumed energy by all wireless devices for two different MSS values, of 1024 and 512 bytes, with and without the simple fragment forward and recovery algorithm. Figure 7.4 shows that the SFFR significantly reduces the energy consumption if no ARQ is applied at the link layer. For example, when the Bit Error Rate (BER) is equal to 5×10^{-5}, the consumed energy without SFFR (767 J) is more than four times the consumed energy when SFFR is applied (160 J).

In the presence of ARQ, SFFR becomes less efficient in low bit error networks due to the added overheads and the new control messages (RFRAG-ACKs). However, it decreases significantly the consumed energy if the bit error rate is high (more than 2×10^{-4}). The SFFR improvement is less significant if the MSS is shorter. For example, Figure 7.4 (b) shows that SFFR is less energy efficient if the MSS is equal to 512 bytes. The longer the TCP segment, the more SFFR reduces the energy consumption because the segment recovery involves more fragments, i.e., more energy.

In Figure 7.6, we fix the bit error rate to 5×10^{-4} and the ARQ maximum number of retransmissions to three and we vary the number of hops from one to ten. Figure 7.6 shows that as the network size (the number of hops between the sender and receiver) increases, the improvement from SFFR becomes more significant. For example, when the



(a) MSS = 1024 bytes

(b) MSS = 512 bytes

Figure 7.5: **Simulation results:** Energy Consumption of a TCP data transfer with vs without SFFR (scenario with five hops).

MSS value is equal to 1024, SFFR reduces the consumed energy by about 30% for a three-hop transmission, and by 70% for a nine-hop transmission. This could also be expected: the larger the distance between the TCP sender and the TCP receiver, the more likely losses are and thus the more beneficial SFFR becomes.

7.4.2 Impact of SFFR on UDP energy consumption

Now, we study the impact of SFFR in datagram mode in a multi-hop scenario, where the receiver does not acknowledge the received segments and no end-to-end transport layer retransmissions are applied. To show the advantages of SFFR, we define an energy efficiency metric (λ), which is equal to the ratio of the received data bytes to the total bytes sent by all wireless nodes.

$$\lambda = \frac{\text{Received data bytes}}{\text{Total Sent bytes}}$$

In the first scenario, we fix the number of hops to five and we vary the bit error rate. As described in the previous section, we compute the energy efficiency of UDP packet transfer with and without SFFR and we plot the energy efficiency curves for two UDP payload sizes (1024 and 512 bytes). We compare the energy efficiency with only ARQ and with both ARQ/SFFR. Figure 7.7 shows that SFFR increases the energy efficiency when long UDP packets are sent. However, SFFR is not energy efficient when the size of the UDP

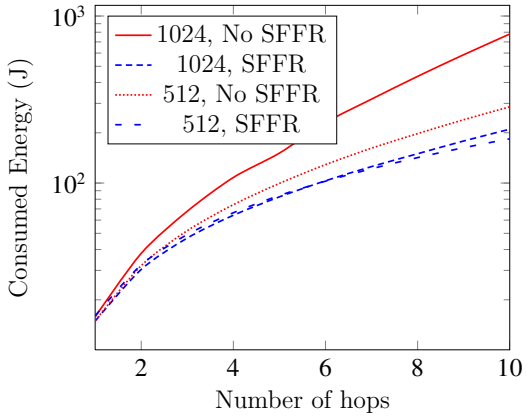
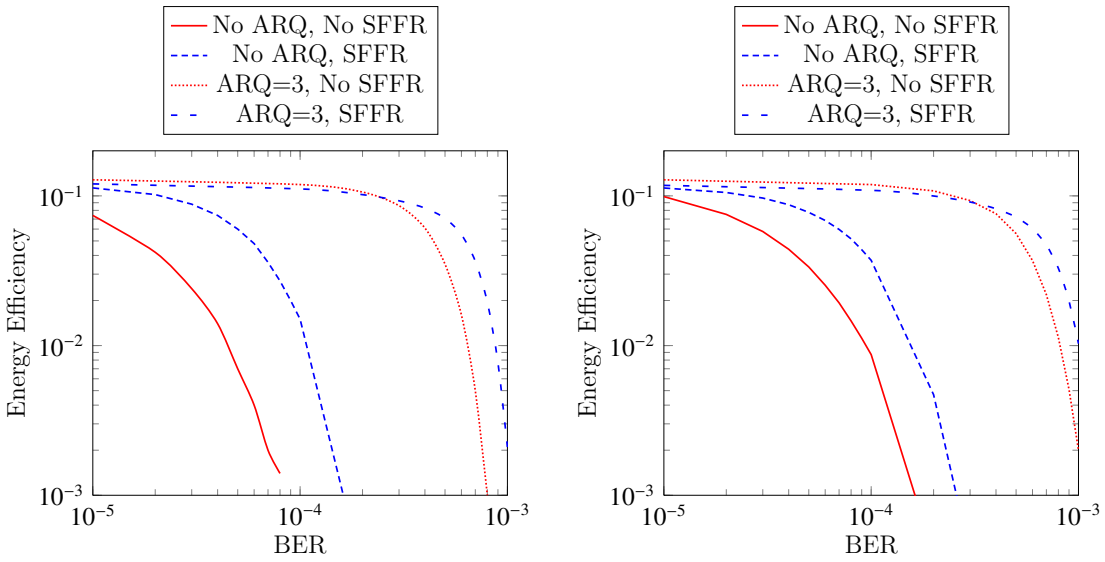


Figure 7.6: Energy Consumption of a TCP data transfer with vs without SFFR (ARQ=3, $B = 5 \times 10^{-4}$).

packet is short. On the contrary, for low BERs it is less efficient than ARQ due to the SFFR overhead.



(a) UDP payload size = 1024 bytes

(b) UDP payload size = 512 bytes

Figure 7.7: Energy Efficiency of an UDP data transfer with vs without SFFR.

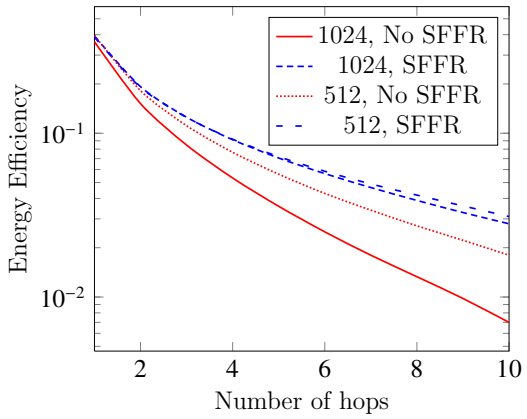


Figure 7.8: Energy Efficiency of an UDP data transfer with and without SFFR (ARQ=3, $B = 5 \times 10^{-4}$).

7.4.3 SFFR rounds *versus* energy efficiency

In this section, we study the impact of the number of fragment recovery rounds on the energy efficiency of SFFR in datagram mode. It is not recommended to add more than one fragment recovery round in TCP mode because, after a round-trip time, the TCP sender assumes that the TCP segment is lost and it must be retransmitted. However, in datagram mode, more than one SFFR round can be applied if the UDP application tolerates the corresponding delay. We study four scenarios with different values of SFFR: 0 (or no SFFR), 1, 2 or 3 SFFR rounds. Figure 7.9 shows the energy efficiency of all scenarios. The UDP payload size is equal to 1024 octets and the number of hops is equal to five.

Figure 7.9 shows that in low BER networks, all curves have the same behavior and we do not observe a significant improvement by increasing the SFFR recovery rounds. However, in high-loss networks (i.e., B is greater than 10^{-4}), Figure 7.9 shows that increasing the recovery rounds improves the energy efficiency. The SFFR recovery rounds recover more lost fragments in-network and then increases the delivery ratio.

7.4.4 When is it better to use SFFR?

Depending of the numerous parameters of a given scenario, it appears that using SFFR is not always the most energy-efficient solution. We now intend to summarize the effect of all parameters, by focusing on the Bit Error Rate, the number of hops, the TCP MSS value, and the ARQ retransmissions.

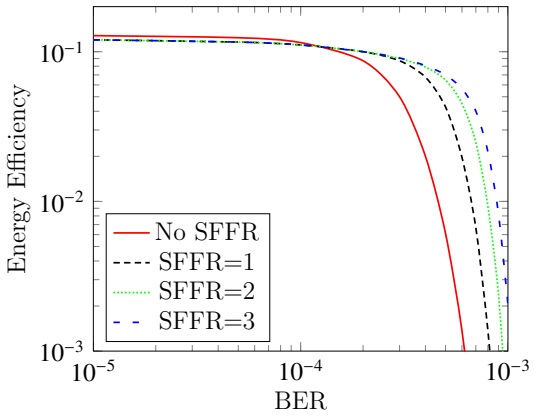


Figure 7.9: Energy Efficiency of an UDP data transfer with different SFFR rounds (ARQ=3, 5 hops).

In the following figures, we compare the energy consumption by a TCP connection with and without SFFR, and we concentrate on the boundary values, that delimitate zones where SFFR yields some improvement or it does not.

Figure 7.10 shows the two zones, depending on the number of hops and the BER. We remark that for a given BER, using SFFR increases the energy-efficiency when the distance grows: it is more and more interesting to use SFFR for large networks, as it was already observed in Figures 7.6 and 7.8.

We plot those frontier curves in Figures 7.10 and 7.11. In each figure, the area above the curve represents the case where the use of SFFR makes the TCP connection consume less energy, while the opposite holds below the curves.

Finally, Figure 7.11 illustrates the effect of ARQ mechanisms. Not surprisingly (the ARQ reducing the effect of transmission errors), ARQ retransmissions reduce the resulting per-hop packet error rate, and thus, they also reduce the energy improvement yielded by SFFR.

7.5 Conclusion

This chapter has studied the performance of a recently proposed end-to-end error correction algorithm that recovers lost link-layer fragments. The main advantage of the simple fragment forward and recovery scheme is to reduce the end-to-end transport layer retransmissions. Numerical results have shown that SFFR reduces the energy consumption for both TCP and UDP traffics.

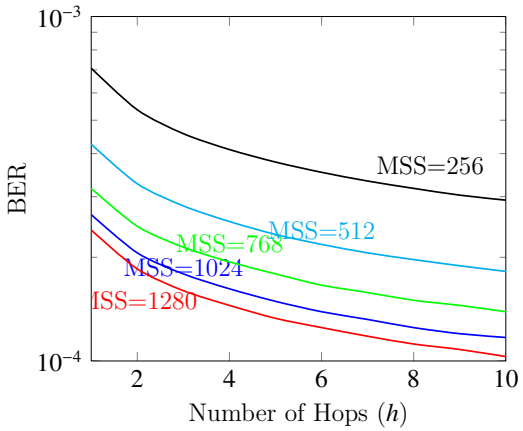


Figure 7.10: SFFR in a multi-hop TCP transmission: prefer SFFR above the curves (ARQ=3).

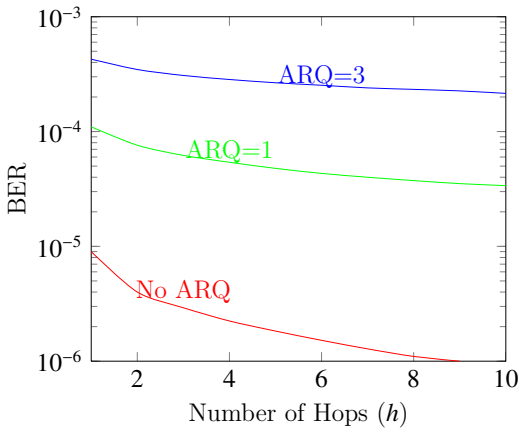


Figure 7.11: SFFR in a multi-hop TCP transmission: prefer SFFR above the curves (MSS=1024).

Chapter 8

Conclusion and Perspectives

The Internet of the Future will not be limited to PCs, routers, laptops, and smart phones but also to the new IP wireless embedded systems that will help to solve large problems in industry, health, home comfort, and energy management. The new embedded wireless systems use wireless connectivity that should be low power. The low energy consumption of the network interface issue has been solved by the new wireless interfaces such IEEE 802.15.4. However, having low-power wireless interfaces does not solve totally the issue if the higher layers are not adapted to the new context.

Low-power networks such as 6LoWPANs have an important role to play for in the Internet of the Future. The success of the 6LoWPAN devices deployment depends firstly on how they will be integrated into the current Internet wireless and wired infrastructure. Moreover, the success depends also on the manner the upper layers should be adapted to solve the tradeoff between energy-efficiency and reliability.

In this thesis, we have been interested in the energy efficiency of transport network protocols in low power networks and especially TCP over low power networks. We showed that TCP suffers from many limitations such as header overhead, high acknowledgment ratio, and end-to-end retransmissions. In this work, we tried to propose simple solutions and improvements for each limitation. Our contributions can be easily added to the current version of TCP in order to make it more viable for the resource-constraint environment without affecting the reliability of the protocol. In the following, we state some of these contributions.

The first point that we worked on is the reduction of the end-to-end retransmission of lost TCP segments. We presented an overview of the proposed TCP hop-by-hop recovery algorithms. We studied Distributed TCP Caching algorithm, then we proposed an enhanced version of it, named NewDTC. Our simulation results showed that our enhancements make DTC more energy-efficient than other TCP hop-by-hop recovery schemes due to a better congestion management.

In the second point, we focused on reducing the ratio of TCP acknowledgments. We studied two of the proposed algorithms to reduce the radio of acknowledgment segments to data segments, which are DCA and TDA. These two algorithms propose to send a TCP acknowledgment for more than 2 data segments. We have implemented DCA and TDA in OMNet++ simulator. Our simulation evaluation showed that TDA outperforms DCA not only on the throughput but also on the energy-efficiency.

We cannot reduce the energy consumption of TCP without reducing the header size. In this thesis we studied the proposed TCP header compression algorithms and we proposed

TCPHC, a new mechanism to compress TCP headers for low power networks (TCPHC). TCPHC aims to reduce the size of the TCP header and thus the energy consumption. TCPHC is a simple and robust header compression algorithm that can be implemented on resource-constrained operating systems such as Contiki OS. TCPHC presents a new idea to compress the dynamic fields of the TCP header that is more loss-tolerant, moreover, it proposes a new way to compress the TCP options. In order to evaluate our header compression algorithm, we implemented TCPHC on Contiki OS and compared its performance to legacy TCP. From our experimental evaluation, we found that no matter the loss level in the network, TCPHC decreases the energy consumption of TCP.

A final topic was to study the impact of the TCP segment size on the energy consumption. The TCP segment size has a great impact on the energy consumption and especially over 6LoWPANs where the link layer MTU is equal to 127 bytes. Based on an analytical model, our study shows that it is better to send long segment when the bit error rate is low; and short segment when the bit error rate becomes weak. Because the bit error rate estimation is not so easy to do, we studied an approach that allows the receiver to quickly recover the lost fragments before a new high layer retransmission. The main idea consist of sending a Results confirm that this approach allow a reduction of end-to-end retransmissions and thus the energy consumption.

After the implementation and evaluation, we reach the final conclusion: TCP can be adapted for the low-power networks and implemented on the new embedded wireless devices. Moreover, the proposed solutions help to reduce the overhead of the TCP header, hence saving the energy consumption.

Perspectives

This work is open to several areas for improvement. In this section, we present the possible perspectives of each part of this work.

The chain topology was the single topology used in our work. One possible future direction consists in performing extensive simulations to evaluate the performance of TCP versions in more complex topologies such as grid and cross topologies. We think that with these new models of topologies simulations will be much more realistic. Another perspective is to combine two or three approaches. For example, the ideas of using both the caching algorithm with TDA would be very interesting because it would reduce the end-to-end retransmissions and the number of exchanged acknowledgment segments. However, this combination should take into account several factors due to changes added by the two algorithms such as the way a TCP source should compute the retransmission timeout. Moreover, we would like to implement the discussed TCP algorithms on a real world testbed to not be limited to simulation results.

The TCP header compression for low power networks is the one of the main contribution of this thesis. Although the evaluation results show good performance, this algorithm is open to many improvements. On the one hand, the header compression algorithm would take into account the displacement of a wireless node and thus its router attachment change. After a change of an ER, the new ER requests the old one for the eventual TCP header compression contexts.

On the other hand, the performance evaluation of TCPHC done in this work is based on a static routing, where all wireless nodes are not mobile. Future scenarios should take into account the wireless node mobility for testing the ER change. This would be easier to integrate especially with the new version of Contiki OS that includes RPL as its default routing protocol.

The analytical model presented in this work, can also be improved especially the assumption that the bit error rate is independent. The Gilbert–Elliott model, which is a simple channel model for describing burst error patterns in transmission channels, can be integrated in our model.

My Publications as a PhD Student

Journal papers

- [Aya11] Ahmed Ayadi, "Energy-efficient and reliable transport protocols for wireless sensor networks: state-of-art", Wireless Sensor Network, Vol. 3, No. 3, pp. 106-113, march 2011.

Conference papers

- [AMR+11b] Ahmed Ayadi, Patrick Maillé, David Ros, Laurent Toutain and Pascal Thubert, "Energy-efficient fragment recovery techniques for low-power and lossy networks", In *Proc. 7th International Wireless Communications and Mobile Computing Conference (IWCMC 2011)*, Istambul, Turkey, 5-8 July 2011.

- [AMR+11c] Ahmed Ayadi, Patrick Maillé, David Ros, Laurent Toutain and Tiancong Zheng, "TCP over Low-Power and Lossy Networks: Tuning the Segment Size to Minimize Energy Consumption", In *Proc. 7th International Wireless Communications and Mobile Computing Conference (IWCMC 2011)*, Istambul, Turkey, 5-8 July 2011.

- [AMR11a] Ahmed Ayadi, Patrick Maillé, and David Ros, "TCP over Low-Power and Lossy Networks: Tuning the Segment Size to Minimize Energy Consumption", In *Proc. International Conference on New Technologies, Mobility and Security (NTMS'11)*, Paris, France, 07-10 february 2011.

- [ZAJ11] Tiancong Zheng, Ahmed Ayadi, and Xiaron Jiang, "TCP over 6LoWPAN for industrial applications: an experimental study", In *Proc. International Conference on New Technologies, Mobility and Security (NTMS'11)*, Paris, France, 07-10 february 2011.

- [AA10] Ahmed Ayadi and Azlan Awang, "Adaptive TCP segment size control for reducing energy consumption in 6LoWPANs", In *Proc. International Conference on Intelligent Network and Computing (ICINC'2010)*, Kuala Lumpur, Malaysia, 26-28 november 2010.

- [AMR10a] Ahmed Ayadi, Patrick Maillé, and David Ros, "Improving distributed TCP caching for wireless sensor networks". In *Proc. IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net'10)*, Juan les pins, France, 23-25 june 2010.

IETF Internet Drafts

- [ART10] Ahmed Ayadi, David Ros, and Laurent Toutain, "TCP header compression for 6LoWPAN", Internet Draft, *draft-aayadi-6lowpan-tcphc-01*, July 2010

TELECOM Bretagne Research Reports

- [AMR10b] Ahmed Ayadi, Patrick Maillé, and David Ros, "TCP over Low-Power and Lossy Networks: Tuning the Segment Size to Minimize Energy Consumption", In *Institut TELECOM/TELECOM Bretagne, (Collection des rapports de recherche de TELECOM Bretagne, RR-2010004-RSM)*, Brest, France, december 2010.

List of Abbreviations and Acronyms

6LoWPAN	IPv6 over Wireless Personel Area Network
ACK	Acknowledgement
ARQ	Automatique Repeat reQuest
BER	Bit Error Rate
BR	Border Router
CBR	Constant Bir Rate
CH	Compressed header
CID	Context Identifier
CPU	Central Processing Unit
CRC	Cyclical Redundancy Check
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
ECN	Explicit Congestion Notification
ER	Edge Router
FEC	Foward Error Correction
FH	Full Header
FTP	File Transfer Protocol
HC	Header Compression
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
ID	Internet Draft
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IID	Interface Identifier
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPSO	IP for Smart Objetcs (Alliance)
LAN	Local Area Network
LLN	Low-power and Lossy Network
LoWPAN	Low-power Wireless Personal Area Network
LSP	Least Significant Byte
M2M	Machine-to-Machine
MAC	Medium Access Control
MH	Mostly compressed Header
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit

NACK	Negative Acknowledgement
OS	Operating System
PC	Personal Computer
PER	Packet Error Rate
PHY	Physical Layer
PLC	Power Line Communications
QoS	Quality of Service
RAM	Random Access Memory
RFC	Requests For Comments
ROHC	Robust Header Compression
ROLL	Routing over Low-power and Lossy networks
ROM	Read Only Memory
RTT	Round-Trip Time
RTO	Retransmission Timeout
SACK	Selective Acknowledgement
SFFR	Simple Fragment Forward and Recovery
TCP	Transmission Control protocol
TCPHC	TCP Header Compression
TDMA	Time division Multiple Access
TTL	Time To Live
UDP	User Datagram Protocol
WG	Working Group
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WWW	World Wide Web

Bibliography

[AA10] A. Ayadi and A. Awang. Adaptive TCP segment size control for reducing energy consumption in 6LoWPANs. In *International Conference on Intelligent Network and Computing*, November 2010.

[AJ03] E. Altman and T. Jiménez. Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks. In *PWC*, pages 237–250, 2003.

[All03] M. Allman. TCP Congestion Control with Appropriate Byte Counting (ABC), February 2003.

[AMR10a] A. Ayadi, P. Maillé, and D. Ros. Improving distributed TCP caching for wireless sensor networks. In IEEE, editor, *IFIP Annual Mediterranean Ad Hoc Networking Workshop*, pages 1 – 6, Juan les pins, June 2010.

[AMR10b] A. Ayadi, P. Maille, and D. Ros. TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption. TELECOM Bretagne Research Report RR-2010004-RSM, October 2010.

[AMR11a] A. Ayadi, P. Maillé, and D. Ros. TCP over low-power and lossy networks: tuning the segment size to minimize energy consumption. In *Wireless Sensor Networks - theory and practice (WSN'2011)*, Paris, France, Febrary 2011.

[AMR+11b] A. Ayadi, P. Maillé, D. Ros, P. Thubert, and L. Toutain. Energy-efficient fragment recovery techniques for low-power and lossy networks. In *IEEE IWCMC: 7th International Wireless Communications and Mobile Computing Conference*, july 2011.

[AMR+11c] A. Ayadi, P. Maillé, D. Ros, T. Zheng, and L. Toutain. Implementation and evaluation of a TCP header compression for 6LoWPAN. In *IEEE IWCMC: 7th International Wireless Communications and Mobile Computing Conference*, july 2011.

[APS99] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control, April 1999.

[ARB+10] N. Ahmed, M. Rutten, T. Bessell, S.S. Kanhere, N. Gordon, and S. Jha. Detection and tracking using particle-filter-based wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(9):1332 –1345, sep. 2010.

[ART10] A. Ayadi, D. Ros, and L. Toutain. TCP header compression for 6LoWPAN. Internet Draft draft-aayadi-6lowpan-tcphc-01, work in progress, July 2010.

[Aya11] A. Ayadi. Energy-efficient and reliable transport protocols for wireless sensor networks: state-of-art. *Wireless Sensor Network*, 3(3):106 – 113, march 2011.

[BAFW03] E. Blanton, M. Allman, K. Fall, and L. Wang. A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP. RFC 3517, IETF, April 2003.

[BB08] Z. Bojkovic and B. Bakmaz. A survey on wireless sensor networks deployment. *WTOC*, 7:1172–1181, December 2008.

[BMAA04] D. Barman, I. Matta, E. Altman, and R. El Azouzi. TCP Optimization through FEC, ARQ, and Transmission Power Tradeoffs. In *Proceedings of WWIC*, pages 87–98, 2004.

[Bor11] C. Bormann. 6LoWPAN Generic Compression of Headers and Header-like Payloads, March 2011.

[BSAK95] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *MobiCom '95: Proceedings of 1st annual international conference on Mobile computing and networking*, pages 2–11, New York, 1995.

[BSGM06] S. Bansal, R. Shorey, R. Gupta, and A. Misra. Energy efficiency and capacity for TCP traffic in multi-hop wireless networks. *Wireless Networks*, 12(1):5–21, 2006.

[BVD07] T. Braun, T. Voigt, and A. Dunkels. TCP support for sensor networks. *Fourth Annual Conference on Wireless on Demand Network Systems and Services, WONS '07*, pages 162–169, Jan. 2007.

[Cas11] A. Castellani. Constrained Messaging Protocol: an UDP protocol extension useful for CoAP and other protocols., July 2011.

[CGLS08] J. Chen, M. Gerla, Y.-Z. Lee, and M. Y. Sanadidi. TCP with delayed ack for wireless networks. *Ad Hoc Networks*, 6(7):1098–1116, 2008.

[CMSM09] B. Chen, I. Marsic, H.-R. Shao, and R. Miller. Improved delayed ack for tcp over multi-hop wireless networks. In *WCNC*, pages 1772–1776, 2009.

[DAV04] A. Dunkels, J. Alonso, and T. Voigt. Distributed TCP Caching for Wireless Sensor Networks. In *Proceedings of 3rd Annual Mediterranean Ad-Hoc Networks Workshop*, 2004.

[DGV04] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.

[DNP99] M. Degermark, B. Nordgren, and S. Pink. IP Header Compression. IETF, RFC 2507, February 1999.

[Dun03] Adam Dunkels. Full TCP/IP for 8 Bit Architectures. In *Proceedings of First ACM/Usenix International Conference on Mobile Systems, Applications and Services (MobiSys 2003)*, San Francisco, May 2003.

[DVA04] Adam Dunkels, Thiemo Voigt, and Juan Alonso. Making TCP/IP Viable for Wireless Sensor Networks. In *Proceedings of First European Workshop on Wireless Sensor Networks (EWSN 2004), work-in-progress session*, Berlin, Germany, January 2004.

[Egg11] L. Eggert. Congestion Control for the Constrained Application Protocol (CoAP), January 2011.

[FARI10] S. Floyd, A. Arcia, D. Ros, and J. Iyengar. Adding Acknowledgement Congestion Control to TCP, February 2010.

[FLZ+05] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla. The Impact of Multihop Wireless Channel on TCP Performance. *IEEE Transactions on Mobile Computing*, 4(2):209–221, 2005.

[FMMP00] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC 2883, IETF, 2000.

[FW02] G. Fairhurst and L. Wood. Advice to link designers on link Automatic Repeat reQuest (ARQ). RFC 3366, IETF, 2002.

[GMP03] L. Galluccio, G. Morabito, and S. Palazzo. An Analytical Study of a Tradeoff Between Transmission Power and FEC for TCP Optimization in Wireless Networks. In *Proceedings IEEE INFOCOM*, pages 1765–1773, 2003.

[GTB99] M. Gerla, K. Tang, and R. Bagrodia. TCP performance in wireless multi-hop networks. In *Proceedings 2nd IEEE Workshop on Mobile Computer Systems and Applications (WMCSA)*, page 41, Washington, DC, USA, 1999.

[HT10] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams in 6LoWPAN Networks. Internet Draft draft-ietf-6lowpan-hc-11, work in progress, September 2010.

[HT11] J. Hui and P. Thubert. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, September 2011.

[IEE99] IEEE 802.11. IEEE 802.11-1999 Standard: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Computer Society., 1999.

[IEE06] IEEE Computer Society. IEEE Std. 802.15.4-2006, October 2006.

[IGV05] Y.G. Iyer, S. Gandham, and S. Venkatesan. STCP: a generic transport layer protocol for wireless sensor networks. In *Proceedings of 14th International Conference on Computer Communications and Networks*, pages 449–454, Oct. 2005.

[Jac90] V. Jacobson. Compressing TCP/IP headers for Low-Speed Serial Links. RFC 1144, IETF, Febrary 1990.

[JBB92] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance, May 1992.

[JEZ+05] R. Jafari, A. Encarnacao, A. Zahoory, F. Dabiri, H. Noshadi, and M. Sarrafzadeh. Wireless sensor networks for health monitoring. In *Proceedings of Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pages 479 – 481, jul. 2005.

[JPS07] L-E. Jonsson, G. Pelletier, and K. Sandlund. The RObust Header Compression (ROHC) Framework. RFC 5795, IETF, July 2007.

[KFD+07] S. Kim, R. Fonseca, P. Dutta, A. Tavakoli, D. Culler, Ph. Levis, S. Shenker, and I. Stoica. Flush: a reliable bulk transport protocol for multihop wireless networks. In *Proceedings of 5th international conference on Embedded networked sensor systems*, pages 351–365, New York, 2007. ACM.

[KHS10] M. Kranz, P. Holleis, and A. Schmidt. Embedded interaction: Interacting with the internet of things. *Internet Computing, IEEE*, 14(2):46 –53, 2010.

[KMS07] N. Kushalnagar, G. Montenegro, and C. Schumacher. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. RFC 4919, IETF, 2007.

[LKL06] H. Lee, Y. Ko, and D. Lee. A hop-by-hop reliability support scheme for wireless sensor networks. In *Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*, PERCOMW '06, pages 431–, Washington, DC, USA, 2006. IEEE Computer Society.

[LS02] W. Lilakiatsakun and A. Seneviratne. Enhancing TCP energy efficiency for mobile hosts. In *Proceedings of 10th IEEE International Conference on Networks*, pages 235 – 239, 2002.

[Mat96] M. Mathis. TCP Selective Acknowledgment Options. RFC 2018, IETF, October 1996.

[Max11] C. Maxfield. IPv4, IPv6, The Internet of Things, 6LoWPAN, and Lots of other Stuff, July 2011.

[MGN07] B. Marchi, A. Grilo, and M. Nunes. DTSN: Distributed Transport for Sensor Networks. *Proceedings of 12th IEEE Symposium on Computers and Communications*, pages 165–172, July 2007.

[MKHC07] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, IETF, September 2007.

[OB07] R.de Oliveira and T. Braun. A Smart TCP Acknowledgment Approach for Multihop Wireless Networks. *IEEE Transactions on Mobile Computing*, 6(2):192–205, 2007.

[O'F10] C. O'Flynn. ICMPv6/ND Compression for 6LoWPAN Networks. Internet Draft draft-oflynn-6lowpan-icmphc-00, work in progress, July 2010.

[PA00] V. Paxson and M. Allman. Computing TCP's Retransmission Timer, 2000.

[PG07] J. Paek and R. Govindan. RCRT: rate-controlled reliable transport for wireless sensor networks. In *Proceedings of 5th international conference on Embedded networked sensor systems*, pages 305–319, New York, 2007. ACM.

[PM97] S. J. Perkins and M. W. Mutka. Dependency Removal for Transport Protocol Header Compression over Noisy Channels. In *Proceedings of International Conference on Communications (ICC)*, pages 1025–1029, 1997.

[Pos81] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. Updated by RFC 3168.

[PPG+07] P. Pereira, P. R. Pereira, A. Grilo, F. Rocha, M. S. Nunes, A. Casaca, C. Chaudet, P. Almström, and M. Johansson. End-to-end reliability in wireless sensor networks: survey and research challenges, December 2007.

[RFB01] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 793 (Standard), September 2001.

[RG60] I. Reed and S. Golomb. Polynomial codes over certain finite fields. *Joint Society of Industrial and Applied Mathematics Journal*, 8(2):300–304, June 1960.

[RGGP06] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. Interference-aware fair rate control in wireless sensor networks. In *Proceedings of conference on Applications, technologies, architectures, and protocols for computer communications*, pages 63–74, New York, NY, USA, 2006. ACM.

[SAA03] Y. Sankarasubramaniam, B. Akan, and I. F. Akyildiz. ESRT: event-to-sink reliable transport in wireless sensor networks. In *Proceedings of 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 177–188, New York, 2003. ACM.

[SB09] Z. Shelby and C. Bormann. *6LoWPAN, the Wireless Embadded Internet*. WILEY, 2009.

[SCM84] L. Shu, D. Costello, and M. Miller. Automatic-repeat-request error-control schemes. *IEEE Communications Magazine*, 22(12):5 – 17, December 1984.

[SFRF01] A. Srivastava, R.J. Friday, M.W. Ritter, and W.S. Filippo. A study of TCP performance over wireless data networks. In *Proceedings of IEEE VTS 53rd Vehicular Technology Conference*, volume 3, pages 2265 –2269, 2001.

[SH03] F. Stann and J. Heidemann. RMST: reliable data transport in sensor networks. *Proceedings of First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 102–112, May 2003.

[SHBF11] Z. Shelby, K. Hartke, C. Bormann, and B. Frank. Constrained Application Protocol (CoAP), July 2011.

[SSS+11] Z. Shelby, M. Garrison Stuber, D. Sturek, B. Frank, and R. Kelsey. CoAP-Requirements and Features, May 2011.

[Ste98] E. Stephen. Internet Protocol, Version 6 (IPv6) Specification. IETF, RFC 2460, December 1998.

[TH10] P. Thubert and J. Hui. LoWPAN fragment Forwarding and Recovery. Internet Draft draft-thubert-6lowpan-simple-fragment-recovery-07, work in progress, June 2010.

[Tit09] J. Titus. 6LoWPAN goes where ZigBee can't, February 2009.

[TWPS09] L. Tuan, H. Wen, C. Peter, and J. Sanjay. ERTP: Energy-efficient and Reliable Transport Protocol for data streaming in Wireless Sensor Networks. *Computer Communications*, 32(7-10):1154 – 1171, 2009.

[Vag10] A. Vagas. OMNET++ 4.0, May 2010. http://www.isi.edu/nsnam/ns/.

[Vag11] A. Vagas. INETMANET framework, January 2011.

[Wan04] R. Wang. An experimental study of TCP/IP's Van Jacobson header compression behavior in lossy space environment. In *Proceedings of 60th IEEE Vehicular Technology Conference*, volume 6, pages 4046 – 4050, sep. 2004.

[WCK05] C.-Y. Wan, A.T. Campbell, and L. Krishnamurthy. Pump-slowly, fetch-quickly (PSFQ): a reliable transport protocol for sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4):862–872, April 2005.

[WEC03] Ch.-Y. Wan, S. B. Eisenman, and A. T. Campbell. CODA: congestion detection and avoidance in sensor networks. In *Proceedings of 1st international conference on Embedded networked sensor systems*, pages 266–279, New York, 2003. ACM.

[YLJH05] Z. Yangfan, M.R. Lyu, L. Jiangchuan, and W. Hui. PORT: a price-oriented reliable transport protocol for wireless sensor networks. In *Proceedings of 16th IEEE International Symposium on Software Reliability Engineering*, pages 10 pp.–126, Nov. 2005.

[YMG08] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292–2330, 2008.

[ZAJ11] T. Zheng, A. Ayadi, and X. Jiang. TCP over 6LoWPAN for industrial applications: An experimental study. In *Wireless Sensor Networks - theory and practice (WSN'2011)*, Paris, France, Febrary 2011.

Index

6LoWPAN, 1, 13

ACK loss detection, 42
AIMD, 21, 23
ARQ, 38, 81, 96, 97

Cache management, 41
CoAP, 15
Compressed TCP, 55
Congestion Control, 19
Congestion Detection, 20
Congestion Notification, 21
Constrained Application Protocol, 31
Context management, 63
CSMA, 12
CSMA-CA, 12, 81

DTC, 29
DTSN, 26

Edge Router, 13, 54
Energy Efficiency, 27
ERTP, 27
ESRT, 20
Explicit Congestion Notification, 32

FEC, 81, 90
Flush, 21

HRS, 26

ICMPv6, 14
IEEE, 1
IEEE 802.11, 1, 38
IEEE 802.15.4, 1, 12, 38, 54
IETF, 1, 13
IFRC, 20
Implicit ACK, 29
Internet of Things, 2
IP Addressing, 15
IPHC, 54, 55

Loss detection, 24
Low power Networks, 1, 10
LoWPAN, 1
LOWPAN_ICMPHC, 14
LOWPAN_IPHC, 14
LOWPAN_NHC, 14

MSS, 80, 91
MTU, 81

OMNet++, 29

PLC, 10
PSFQ, 26

Random Early Detection, 21
RCRT, 20
Reed-Solomon, 81
Reliability, 24
Remote Environment Monitoring, 10
RMST, 25
ROHC-TCP, 56

SACK, 38
Selective Acknowledgment, 28
SFFR, 97

TCP, 14, 28, 79, 86
TCP Header Compression, 58
TCPHC, 54, 58
TDMA, 12
TSS, 29
TWICE, 55

UDP, 14

Wireless Sensor Networks, 10

ZigBee, 1

www.telecom-bretagne.eu

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
Tél : + 33 (0)2 29 00 11 11
Fax : + 33 (0)2 29 00 10 00

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
Tél : + 33 (0)2 99 12 70 00
Fax : + 33 (0)2 99 12 70 19

Campus de Toulouse
10, avenue Édouard Belin
BP 44004
31028 Toulouse Cedex 04
France
Tél : + 33 (0)5 61 33 83 65
Fax : + 33 (0)5 61 33 83 75