Removing the MAC Retransmissions Times from the RTT in TCP

Eugen Dedu, Sébastien Linck, François Spies
{Eugen.Dedu,Sebastien.Linck,Francois.Spies}@pu-pm.univ-fcomte.fr
Laboratoire d'Informatique de l'Université de Franche-Comté (LIFC)
IUT de Belfort-Montbéliard, 4 place Tharradin, BP 427, 25211 Montbéliard, France

KEYWORDS
Wireless network, MAC retransmission, RTT, TCP.

Abstract

MAC retransmission of packets (for example in a video transfer) raises the RTT and leads to throughput decrease on TCP versions based on RTT for their sending rate. This is not the appropriate effect, because retransmissions are given by temporary interference, which often appear in wireless links. This paper treats effects of MAC retransmissions. A new TCP option is proposed, which fully takes into account the effect of MAC retransmissions. Wireless network cards have a timer which is initialised with the value of the option once the packet is sent the first time, and is put in this option each time the packet is retransmitted. The exact time of retransmissions is then known by the source. Simulations show that the proposed mechanism increases throughput.

INTRODUCTION AND MOTIVATION

It is now an evidence that TCP is not adapted to wireless links. The main concern is that losses/delays are given by *temporary* interferences, hence no congestion mechanisms should be taken.

Many studies treat effects of packet *loss* in wireless links (802.11) on TCP, but only few treat effects of packet *delay* during retransmission. It is worthwhile to note that successful MAC retransmissions appear much more frequently than MAC losses.

On wireless links, MAC retransmissions are the effect of radio interferences. Because these are temporary and appear randomly, they do not give any useful value, but mislead network mechanisms. Therefore, several works exist for avoiding their effect. But works treating the MAC retransmission concentrate generally on the influence the RTT has on RTO (retransmission timer) of TCP [11, 7, 6].

However, there are several TCP versions that use the RTT for their *sending rate*, for instance Vegas [1] uses all the RTT samples, and Westwood+ [4] and TIBET [2] use smallest RTT. RTP/RTCP over UDP, common for video streaming for example, is another case where RTT values are needed. Other mechanisms which need correct RTT may arise. [10] shows that in certain cases a gain of 10% in throughput appear if the real RTT is replaced by the adjusted RTT.

This article proposes a mechanism that allow senders to take into account the time taken by retransmissions. A TCP option is added, with one field. Also, a timer is added to the network card. Each time a packet is transmitted, the network card puts in its option the time taken by its retransmission, using the internal timer. The field is echoed back to the source in an ACK packet. The source is hence aware of the time lost in all the MAC retransmissions. A similar mechanism can be applied to non TCP flows too.

BACKGROUND

802.11 is a protocol which is used with two access modes: PCF (Point Coordination Function) and DCF (Distributed Coordination Function).

In DCF mode, each machine can access the network when it wants. However, in order to reduce collisions, machines must choose randomly a value, called *backoff*, in a given range, called collision window (Contention Window CW). CW doubles at each retransmission (with an upper bound equal to a power of 2 minus 1). The initial window lies between 0 and 31 ($= 2^5 - 1$) units. Units are called *time slots*, with $t_s = 20\mu s$ as defined by the standard.

In PCF mode the AP (access point) is the master of conversations. It gives the "access" in turn to all the mobiles. Mobiles can send packets only when the AP gives them the right. There is no backoff. It is worthwhile to note that in the mode PCF each period of time is divided in two parts: the PCF part and the DCF part. This allows machines which implement only DCF mode to have the possibility to access the network. The method proposed in this article works in both DCF and PCF modes.

802.11 is a protocol which relies on ARQ (Automatic Repeat Request). After sending a packet, the network card listens for an acknowledgement from the receiver's network card. If it does not arrive, then the sender's network card considers that the packet was lost and it will retransmit it. There is a limit in the number of retransmissions.

Packet loss is frequent in wireless networks. They appear generally when an external interference occurs in

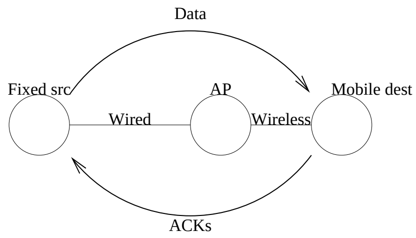

Figure 1: A TCP transmission between a wired machine and a wireless one.

the network, but also when the mobile machines exits from the covered area of the network. Interferences are known to be temporary and appearing at random times. A detailed explanation of the 802.11 standard can be found in [3].

PRINCIPLE

Due to frequent losses, 802.11 allows MAC retransmission. Hence, 802.11 transform a network with losses and predictible delay into a network with no loss and variable delay. On the other hand, packet losses are generally due to interferences. As they are *temporary*, the RTT should not be influenced by them. We therefore propose a mechanism to remove the delay taken by retransmissions.

In order to implement this idea we were faced to several choices. Is the number of retransmissions or the total retransmission times which is sent to the sender? Is this information put at the TCP or at the IP level? Is this information sent by the AP directly to the source, or passed to the destination which in turn sends it to the source? Several solutions exist, we present only the solution we judge the most appropriate.

Our solution is the following (see figure 1). An option is added to TCP, called `timelost`, containing a field, called `rets`. Each wireless network card has a timer. The timer is initialised to the value of the `rets` field of the packet for the first transmission of a packet (after the backoff). Each time it sends a packet on the wireless link, the value of the timer is put in the `rets` field. Thus the timer reflects the *exact* time loss due to retransmissions. When the source receives a packet, it takes the appropriate action, for example it subtracts the value of the `rets` field from the RTT of the packet.

The reminder of this section details this mechanism in TCP. A similar mechanism can be used for non TCP flows too, for example in an RTP/RTCP transmission, the receiver puts in RTCP packets going to the source the time lost in retransmissions of data packets.

Retransmission Time Computing

As specified, each network card has a timer. Each time a new packet is processed, its `rets` is used to initialise the timer. This allows to sum several times, for example when the packet is an ACK and it contains already a time, the lost time of the corresponding *data* packet. During each (re)transmission, the value of the timer is put in the `rets` of the packet.

The modification which needs to be done at the network card level is the addition of a timer. As nowadays timers exist on all the network cards (used for backoff for example), this is not an issue.

The MAC-level fragmentation does not influence our mechanism. Indeed, when an IP packet is fragmented the time loss is null if each fragment arrives at destination without retransmission.

Transmitting the Information to the Source

We propose an option, called `timelost`, to be added to TCP. This option has only one field, containing a time value. If the field has 2 bytes and the measurement unit is the time slot t_s given previously, then the field will overflow at a time $t = 65536 t_s = 65536 \times 20 \mu s \approx 1.3s$. In the 802.11b standard the maximum CW is 1023, hence 2 bytes are sufficient. If the field has 4 bytes, it will overflow after $t \approx 65536 \times 1.3s \approx 1$ day, which is largely sufficient.

The source sets this field to zero. During the trip, the field may be modified by a wired-wireless machine. The receiver echoes back the value to the source in an ACK packet, exactly like the TCP timestamp option [5]. During the return trip, the field may further be increased. The source receives the ACK packet and computes the retransmission time.

This mechanism allows incremental deploying. It gives useful values only if the sender, the receiver and the AP know about it. If the sender is not aware of this option, nothing happens. Elsewhere, it adds the option and set the field to 0. If the AP and/or the receiver do not know this option, the sender either receives no option, or an option with value 0, which do not change anything either.

Actions Taken by the Source

It is up to the source to take appropriate actions. For instance, we suggest that a TCP Vegas source would use the new RTT in its formulae, or to a video server this would allow to better know the network jitter.

Drawbacks

In this mechanism, network cards have access to level 4 (TCP) information. It is worthwhile to note that several papers propose such methods.

Also, it needs to be further evaluated if the gains obtained by this mechanism for a higher range of transfers overtake the changes needed to implement it.

SIMULATIONS

The NS2 simulator [8] version 2.28 is used to simulate our mechanism. We have modified NS2. All the modifications to NS2 and the source of the tests shown in this article are available on-line[1].

Modifications to NS2

We add only a new field, called `rets`, to TCP header. It is easier to implement and does not change results. `rets` is of type `double`. We modify:

1. The sending part of TCP/Vegas, which initialises `rets` to 0.
2. The sending part of the MAC 802.11 protocol, which adds retransmission time to `rets` field.
3. The sending part of TCPSink (whose role is to send ACKs), which echoes `rets` field of the data packet in the ACKs it sends.
4. The receiving part of TCP/Vegas, which computes RTT as RTT minus `rets`. There is no other action to take at source, because it sees the modified RTT.

The second item is used/applied only when `useRets` is 1. `useRets` is an NS2 parameter which can be used in user's `.tcl` files. It defaults to 0.
At 802.11 level, each time a packet is retransmitted, its retransmission time is computed. The value used is the time between the packet sending time and the reception of its acknowledgement.

Shadowing propagation model in NS2

Currently, three wireless propagation models are implemented in NS2 [9]: Free space, Two ray ground and Shadowing. The two first models are of type "all or nothing": If distance d between the mobiles is smaller than a certain value, all packets are received. If d is greater, no packet is received. These are not appropriate in our case, since we need retransmissions from time to time.
In the shadowing model, packets are always received for $d < s1$, always lost for $d > s2$ and received with a probability for $s1 < d < s2$.

Simulations

The simulated network in given in figure 1: a wired TCP Vegas source, an AP, and a mobile TCP destination. Note that both AP and the mobile use retransmission times. The mobile destination moves away linearly from the AP, starting at $t = 0$. At $t = 150s$, it stops moving and stay there until the end of simulation. It is near the covered area of the AP, where most of the retransmissions appear. Figure 2 details the movement of the mobile.
Two simulations are executed: one with original version of TCP and one with modified version of TCP. The results of the simulation are given in figure 3. Several stages can be seen in the figure:

- During the first 5 seconds, the routing protocol initialises and no data exchange appear.
- In the shadowing model, no packet loss/retransmission appear until a certain distance. This is clearly seen in the figure, as the curves of two versions are identical for $t < 100$. Gains can be seen only on retransmissions.
- For $t > 150$, retransmissions are common. The mobile does not move. An average improvement of 4.5% is obtained.
- The maximum increase in the bandwidth over a 100 seconds interval is obtained for $150 < t < 250$: approximately 15%.

[1] At `http://lifc.univ-fcomte.fr/~dedu/publi/euromedia/`

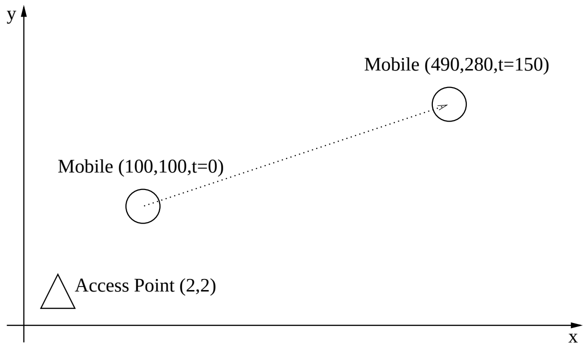

Figure 2: The movement of the mobile.

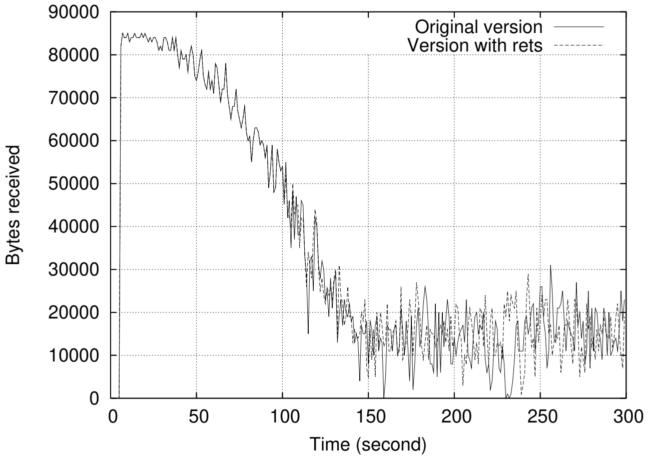

Figure 3: Simulation results.

RELATED WORK

From the implementation point of view, there are several works on modifying the TCP specification and hence the sender and the receiver.

Several works treat packet loss in wireless networks.

[11] presents an analytical model that predicts the RTO of TCP for given network parameters. It concludes that delay variations are critical only when they are of order of seconds.

[7] presents a model where certain packets are artificially delayed at link-layer. The RTT increases, and the RTO of TCP with it, but the number of false TCP retransmissions decreases, leading to a higher overall throughput.

In [6], the wireless access point (AP) encapsulates each packet in a header containing a timestamp, which is echoed by the receiver in its ACKs. This allows AP to discover lost packets even if they are out-of-order. When such a loss is detected, the AP sets a bit (ERN, Explicit Retransmission Notification) in the ACK packet which inform the sender to retransmit the packet without taking congestion actions.

The negative effect of MAC retransmission on TCP is treated in [10], where the TCP connection between a wired machine and a wireless machine is divided in two TCP connections by the AP in the middle. The AP buffers data received from the wired end and retransmit it to the wireless end if it was not received. Also, the time spent in the AP is subtracted from the TCP timestamp option. Contrary to the method proposed in this article, the time spent at the AP is not accurate (the timestamp granularity depends on the source machine [5]), the AP needs to buffer data and it works only with TCP timestamp option.

CONCLUSIONS AND FUTURE WORK

This article presents a method to remove the effect of MAC retransmissions on the RTT computing on source-side in TCP. In adds an option to TCP containing a time value. This value is increased by wireless network cards whenever the packet is retransmitted. The field is propagated back to the source, like the TCP timestamp option [5]. This allows the sender to know exactly the time passed in retransmissions.

Simulations have shown an improvement of throughput with our method. We plan to do real experiments in order to validate our simulations. If successful, we plan to give a complete proposal for improving TCP in case of retransmissions on wireless links.

ACKNOWLEDGEMENTS

Authors are grateful to Dominique Dhoutaut for his insights in the 802.11 protocol.

References

[1] L. Brakmo and L. Peterson. TCP Vegas: end to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communication*, 13(8):1465–1480, Oct. 1995.

[2] A. Capone, L. Fratta, and F. Martignon. Bandwidth estimation schemes for TCP over wireless networks. *IEEE Transactions on Mobile Computing*, 3(2):129–143, Apr.-June 2004.

[3] D. Dhoutaut. *Etude du standard 802.11 dans le cadre des réseaux ad-hoc : de la simulation à l'expérimentation.* PhD thesis, Ecole Nationale Supérieure de Lyon, Dec. 2003.

[4] L. A. Grieco and S. Mascolo. TCP Westwood and Easy RED to improve fairness in high-speed networks. In *IEEE/IFIP Seventh International Workshop on Protocols for High-Speed Networks*, pages 130–146, Berlin, Germany, Apr. 2002.

[5] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance, May 1992. RFC 1323.

[6] Z. Jing and N. Zhisheng. A reliable TCP-aware link layer retransmission for wireless networks. In *International Conference on Communication Technology*, volume 1, pages 900–905, Beijing, China, Aug. 2000.

[7] N. Möller, K. H. Johansson, and H. Hjalmarsson. Making retransmission delays in wireless links friendlier to TCP. In *43rd IEEE Conference on Decision and Control*, Bahamas, Dec. 2004.

[8] Network simulator — ns-2. `http://www.isi.edu/nsnam/ns/`.

[9] The NS manual. `http://www.isi.edu/nsnam/ns/ns-documentation.html`.

[10] K. Ratnam and I. Matta. Effect of local retransmission at wireless access points on the round trip time estimation of TCP. In *The 31st Annual Simulation Symposium*, pages 150–156, Boston, MA, USA, Apr. 1998.

[11] M. Scharf, M. Necker, and B. Gloss. The sensitivity of TCP to sudden delay variations in mobile networks. In *3rd IFIP-TC6 Networking Conference*, pages 76–87, Athens, Greece, May 2004. LNCS 3042.