Auto-protection of 802.11 networks from TCP ACK division

Andrés Arcia*
ae.arcia@telecom-bretagne.eu

David Ros
david.ros@telecom-bretagne.eu

Nicolas Montavont
nicolas@montavont.net

Institut TELECOM; TELECOM Bretagne
RSM, 2 rue de la châtaigneraie CS 17607
35576 Cesson Sévigné Cedex
Université européenne de Bretagne, France

1. INTRODUCTION AND MOTIVATION

Nowadays, Internet Mobile terminals are now all equipped with 802.11 devices. Moreover, a high and increasing number of hot spots are deployed in a variety of places, such as homes, cafes and airports. A typical mobile user uses TCP-based applications to access emails, browse the web or download files from the Internet. Thus the selfish interest for a user to have a better performing version of the Transmission Control Protocol (TCP).

In TCP, receivers usually delay the emission of acknowledgements (ACK) packets for efficiency purposes. However, just as TCP receiver may send less than one ACK per incoming data packet, it might also send more than one ACK per segment without breaking the fundamental ACK semantics. Savage et al. [5] studied first the so-called *ACK division* phenomenon (*divacks* for short), that consists in sending a massive number of ACKs per segment. Eventually, misbehaving receivers may attain an unfair share of the available bandwidth through the sending of *divacks*. On the other hand, using *divacks* has also been proposed to improve TCP's reaction when it faces abrupt bandwidth changes in the last hop of a wired-cum-wireless network [4]. In both cases, the idea is to take advantage of the increased TCP congestion window (*cwnd*) growth rate that could be achieved through the massive sending of *divacks*.

Using a medium access control technique such as distributed coordination function (DCF) in 802.11 and the RTS/CTS access mechanism, *divacks* and data packets have equal media access opportunities due to DCF contention nature [2]. We show that the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) mechanism with the random exponential back-off acts as a protection filter for the wired network from massive sending of *divacks*. This observation is valid as long as the bottleneck of the network remains in the wireless part. Thus, when the bottleneck is in the wired part of a wired-cum-wireless network, data packets are delayed and *divacks* gain access more frequently during the period of time in which the AP's buffer is empty.

* Also with University of Los Andes - Venezuela

ACM CoNEXT 2008 Student Workshop, December 9, 2008, Madrid, SPAIN. ISBN 978-1-60558-264-1

2. DISCUSSION

The mismatch in the way ACK information is handled by congestion control algorithms may lead to a severe form of unfairness, identified in [5]. A greedy TCP receiver may artificially increase the sender rate by simply sending *divacks*; the higher the number of *divacks* per TCP data segment, the higher the window growth rate and therefore the higher the throughput. Under certain conditions, this might allow a misbehaving TCP receiver to get an unfair share of the available bandwidth and could lead to congestion collapse. Savage et al. [5] illustrated this issue by an experiment using a modified Linux TCP stack, showing a reduction of 70% on the transfer time of a 65 *kbytes* file from a distant busy server.

Allman [1] introduced two byte-counting algorithms[1] to compensate the ACK rate impact on TCP *cwnd* performance. When ACKs are delayed like in a sender-controlled approach [3], the *cwnd* is increased proportionally to the number of acknowledged bytes. And at the same time, when the ACK rate is increased through *divacks*, byte-counting limits the growth of the *cwnd*. So it refrains the increasing of the *cwnd* in packet mode, and thus the *cwnd* is proportionally increased to the number of bytes acknowledged (i.e., closing the mismatch in the ACK interpretation). When this mechanism is deployed, *divacks* has no further effectiveness. Nevertheless, TCP byte-counting is no

[1]IETF Experimental RFC 3465.

longer active by default in current Linux distributions due to performance problems on short transfers (e.g., for RPC or telnet-like connections).

3. PRELIMINARY RESULTS

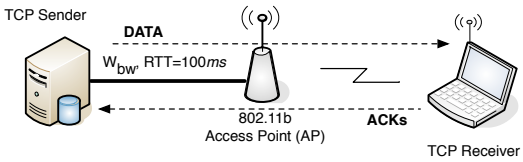

Figure 1: Wired-cum-wireless topology

In order to observe the impact of *divacks* on the congestion window we modified the BSD-derived implementation of TCP called Full-TCP in the *ns-2* simulator and implemented the *divacks* attack in the topology shown in Fig. 1. The experiment consisted on progressively decreasing the bandwidth at the wired part of the network while observing the impact on the *cwnd* dynamic. We used the default settings for the 802.11b access.

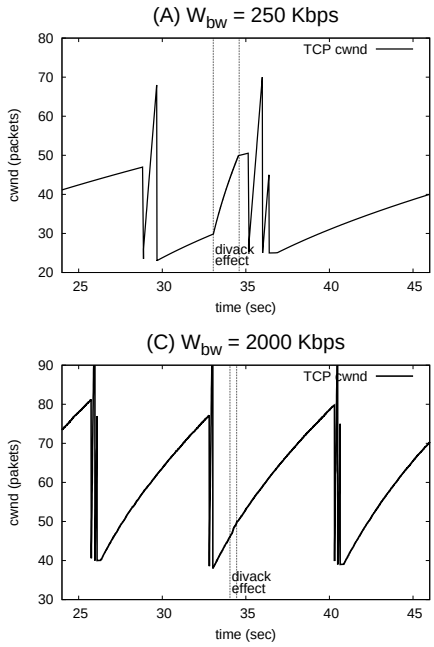

Figure 2: Effect of divack considering different wired-bottlenecks.

In the experiments described in Fig. 2 we sent 10 *divacks* per data packet for 80 in-order data packets (i.e., 800 *divacks* in total) during the congestion avoidance phase at the time indicated between the vertical bars. Although we are conscious that there is an important impact on the *slow start* phase, we know that for long transfers, the congestion avoidance phase is more important and thus our interest.

We tested for various rates of *divacks* per data packet, but for space reasons we show the relevant ones to illustrate our findings. Contrary to the common wisdom we found that a misbehaving receiver noticeably benefits from *divacks* only if the bottleneck is located at the wired part of the network, which does not represent the commonly found deployment scenario. Fig. 2.a shows how more transmission opportunities are given to the TCP receiver as long as the bandwidth of the wired part is smaller than the 802.11b access bandwidth. On the other hand, in Fig. 2.b, when the wired bandwidth is increased the effect on the *cwnd* is unnoticeable.

4. ON-GOING AND FUTURE WORK

We modified the Full-TCP version on *ns-2* and verified the behavior of the ACK division attack in a 802.11b access network. We found, contrary to the common wisdom, that the described attack is ineffective when the bottleneck is located at the wireless part of the wired-cum-wireless network.

We are currently working on the mathematical modeling of the media access impact on TCP *cwnd* when the *divack* attack is deployed. Moreover, we are extending our tests in a deployment with multiple clients using long-lived TCP traffic that perform download. In such scenarios where multiple clients are sharing the same AP, we believe that the effect of *divacks* will completely disappear because the increased competition for the medium access. Finally, we are also conducting a research for the impact of *divacks* facing TCP-congestion.

5. REFERENCES

[1] M. Allman. TCP Byte Counting Refinements. *SIGCOMM Comput. Commun. Rev.*, 29(3):14–22, 1999.

[2] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordinationfunction. *Selected Areas in Communications, IEEE Journal on*, 18(3):535–547, 2000.

[3] S. Floyd, A. Arcia, D. Ros, and J. Iyengar. Adding Acknowledgement Congestion Control to TCP. Internet Draft draft-floyd-tcpm-ackcc-04, work in progress, May 2008.

[4] M. Nakata. Receiver-based ACK Splitting Mechanism for TCP over Wired/wireless Heterogeneous Networks. Master's thesis, Graduate School of Information Science and Technology, Osaka University, Japan, 2006.

[5] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM SIGCOMM Computer Communications Review*, October 1999.