On the Uplink Performance of TCP in Multi-rate 802.11 WLANs

Naeem Khademi[1], Michael Welzl[1], and Renato Lo Cigno[2]

[1] Department of Informatics, University of Oslo, Norway
{naeemk,michawe}@ifi.uio.no
[2] DISI, University of Trento, Italy
locigno@disi.unitn.it

Abstract. IEEE 802.11 defines several physical layer data rates to provide more robust communication by falling back to a lower rate in the presence of high noise levels. The choice of the current rate can be automatized; e.g., Auto-Rate Fallback (ARF) is a well-known mechanism in which the sender adapts its transmission rate in response to link noise using up/down thresholds. ARF has been criticized for not being able to distinguish MAC collisions from channel noise. It has however been shown that, in the absence of noise and in the face of collisions, ARF does not play a significant role for TCP's downlink performance. The interactions of ARF, DCF and uplink TCP have not yet been deeply investigated. In this paper, we demonstrate our findings on the impact of rate fallback caused by collisions in ARF on the uplink performance of various TCP variants using simulations.

Keywords: TCP; 802.11 WLAN; Auto-Rate Fallback

1 Introduction

Rate adaptation is a mechanism that exploits the multi-rate capability of the physical layer defined in IEEE 802.11 standards to provide more robust communication in the presence of channel failures (e.g. noise, fading, interference). This is achieved by adapting the physical layer (PHY) rate selection based on the channel quality. Four (1∼11 Mbps) and eight (6∼54 Mbps) PHY rates are mandated by 802.11b and 802.11g standards, respectively. The choice of the specific adaptation algorithm is however left up to the vendor in the standard – but this is critical to the system performance.

Several rate adaptation algorithms have been proposed in the literature, most notably Auto-Rate Fallback (ARF)[6], which chooses the rate based on the number of consecutive successful or unsuccessful transmission attempts (up/down thresholds). For instance, it falls back to a lower rate after two consecutive transmission failures (e.g. ACK frame is not received) and increases the rate after ten successful transmission attempts in *Cisco Aironet 350* cards[1].

ARF is well known and widely adopted in wireless cards due to its simplicity, but it has been criticized for not being able to distinguish losses due to collision from losses due to channel noise which, in combination with DCF, may affect the overall system performance in typical multi-user wireless scenarios. There has been a significant amount of research on the interaction of DCF and ARF indicating ARF's poor performance at the MAC level, considering constant bit-rate UDP traffic scenarios [9, 12, 16]. However, less work has been done to study the inter-layer dependencies of ARF, DCF and TCP; exceptions are [1, 2]. These references reveal that, in scenarios with downlink traffic, TCP throughput hardly depends on the number of contending stations.

It has been shown that, in pure TCP downlink scenarios, the number of active stations participating in multiple access contention at an instant of time stays extremely low due to the n:1 traffic ratio of TCP ACK to data packets, resulting in a very low collision rate and therefore gaining the maximum achievable throughput. These results are valid for many typical client-server (e.g. http access) scenarios. Considering today's popular p2p file sharing, VoIP, email attachments and multimedia streaming applications, there is a need to study the overall behavior of the system in the presence of ARF in TCP uplink traffic scenarios, where several stations are contending to access the medium to upload large data packets simultaneously. In this paper we will address such scenarios.

The rest of this paper is organized as follows: in section II we provide some background on the current research trend of rate adaptation mechanisms and their pro's and con's compared to ARF as well as cross-layer interactions of ARF, DCF and TCP. In section III we will evaluate the uplink performance of TCP in presence of ARF with simulations; in section IV we extend our scope to high speed TCP variants and finally, in section V, we propose future work and conclude.

2 Background

Rate adaptation mechanisms in 802.11 WLANs adapt the PHY transmission rate based on the channel conditions to optimize network parameters such as application-level throughput or power consumption. The former is the main focus of this paper. There have been several proposed algorithms in the literature aiming to achieve this goal [6, 8, 9, 15, 5].

ARF [6] is the first published rate adaptation algorithm basically designed for WaveLan II devices and later employed by several 802.11 Wi-Fi NICs. The basic idea of ARF is to increase the PHY transmission rate after a certain number of successful attempts (*up* threshold) and falling back to a lower rate after a certain number of consecutive failures (*down* threshold) in addition to setting a timer. After fallback, when the number of per-frame ACKs reaches the *up* threshold or the timer expires, the PHY rate will be increased. If the first frame transmission after increasing the rate (*probing* frame) fails, ARF immediately switches back to the previous rate and restarts the timer.

Several problematic issues arise when deploying ARF. First: it takes frame loss as an indication of high channel noise and therefore mistakes collisions for noise. Second: it is too aggressive in rate reduction due to the normally small *down* threshold value (e.g. one or two) and it can therefore not utilize the total available bandwidth. Third: it attempts to increase the rate after each *up* threshold (e.g. ten) number of successful transmissions, but this may not be an indication that channel conditions have in fact improved.

Some work has been done to solve these problems [8, 9, 15]. The Collision-Aware Rate Adaptation (CARA) algorithm proposed in [8] exploits the RTS/CTS frames' functionality to differentiate frame collisions from frame transmission failures due to noise. The main idea behind CARA is that a transmission failure of a small RTS frame which is normally encoded at the lowest rate is less likely to be caused by channel noise and most probably the result of a collision, while the transmission failure of a larger data frame followed by a successful RTS/CTS handshake is probably due to channel noise. CARA's practicality in infrastructure-based WLANs is limited because of the mandatory usage of RTS/CTS.

Adaptive ARF (AARF) [9] is an extension of ARF which decreases the number of probing packet failures to use a higher rate by multiplying the default *up* threshold (e.g. 10) by two after a probing packet failure. The *up* threshold is set to its default value after a rate fallback. While AARF decreases the number of failed transmissions and retransmissions, it still inherits other weaknesses of ARF.

Rather than ARF, other rate adaptation mechanisms such as [5] require incompatible changes to the 802.11 standard, and therefore they are out of our scope. In this paper we focus on ARF only since it is one of the few open-source and 802.11-compatible mechanisms which is widely employed in wireless NICs.

As already mentioned, ARF's performance has been studied extensively at the PHY/MAC level. However, few works are available that investigate this at the transport layer level focusing on cross-layer interaction of ARF, DCF and TCP [1, 2]. In these references, it is claimed that ARF has a negligible impact of the performance of TCP thanks to TCP's self-clocking mechanism and the $1/n$ chance of the AP to gain access to the channel, which decreases the number of stations actively competing for channel access. Since the cases studied in these works are strictly based on download traffic scenarios, they lack a generic conclusion about whether ARF plays a significant role for TCP or not. The impact of ARF on uplink TCP traffic is discussed in the next section.

3 TCP performance in Multi-rate WLANs

To study the behavior of ARF in the presence of uplink and downlink flows, we performed a set of simulations using *ns-2* [14] and an extension [13] to provide Auto-Rate Fallback support (AARF)[1]. Ten stations are located at the same

[1] Performance evaluations of this paper are made using an improved variant of ARF (AARF) [9] and mentioned as ARF hereafter. Since AARF inherits the two major problems of ARF (being collision-reactive and aggressive in rate reduction) and only reduces the number of failed probing frames, any problem found in AARF applies to ARF as well.

distance of $10m$ from an access point to gain an equal signal power. Noise and interference effects are omitted from the scenarios in order to isolate the frame collision events from the other sources of frame loss. The main focal point of this work is to study the potential negative impact of collisions coupling with ARF on TCP performance when the channel condition is relatively good in a simplified and idealized scenario; therefore we have left this study under noisy channel condition which is harder to model as our future work. Common parameters used in different sets of simulations in this paper are brought in Table 1.

Table 1. Simulation parameters

MAC protocol	IEEE 802.11b	IEEE 802.11g
Packet size	1500 bytes	
Propagation model	TwoRayGround	
Interface queue	DropTail (50 packets size)	
CWMin	32	16
CWMax	1024	
SIFS	$10\,\mu s$	
DIFS	$50\,\mu s$	$28\,\mu s$
Slot time	$20\,\mu s$	$9\,\mu s$
RTS/CTS option	Disabled	

Figure 1(a) shows the per-second aggregate goodput of these stations under 802.11b channel when each of them is downloading unlimited FTP data carried by a TCP NewReno flow from a server which is connected to the AP via a 100 Mbps wired link[2]. As observed, ARF does not affect the performance of download flows for the whole simulation period keeping the aggregate goodput at the maximum level compared to when ARF is disabled. This phenomenon is due to TCP's self-clocking mechanism and the *1/n* probability ratio of the AP to gain access to the shared channel.

Simply put, data packets depart from the AP's downstream queue to the wireless stations. Upon successfully receiving a data packet at the destination's NIC, this station's queue will be backlogged with an ACK packet to be sent back after winning in medium access contention. Since the AP has equal probability of access to the channel as any other stations and it is in charge of transferring all data packets, stations' backlogged queues grow only as a consequence of AP's successful contention to access the channel. Collision occur only when already backlogged stations and the AP compete to concurrently access the channel. Using birth-death Markov chains in [1, 2], it has been shown that TCP's self-clocking mechanism keeps the number of backlogged stations (stations actively participating in contention) in equilibrium at 2-3 contending stations for scenarios with 2-100 wireless stations. This level of contention provides a very low

[2] Unless otherwise noted, the same parameter settings are also applied in all other simulations in this paper.

likelihood of collision, which mitigates the negative effect of ARF on downlink's TCP performance. In addition, the almost fixed collision rate causes what the authors called TCP's "scalable performance" for a varying number of nodes.

In contrast to these findings, in upload scenarios, wireless stations are participating in contention to transfer data to the AP and their queues are backlogged with data packets in accordance with their window size. A collision – in this case, between data packets – can be more harmful than in the download scenario, where ACK packets are colliding with each other or with data packets sent from the AP. The packet size also plays a role in determining the impact of collisions. The possible rate downshift triggered by ARF, affecting the transmission of a subsequent large data packet will lead to the under-utilization of the wireless channel bandwidth for a longer period of time compared to the transmission of a short ACK packet. Possibly, this may also result in the other stations' backlogged queue to grow larger resulting in a higher level of collisions.

Figure 1(b) shows the per-second aggregate goodput of uploading stations. It reveals that, when ARF is disabled, uploading stations perceive more fluctuations than the downloading stations because of the earlier mentioned difference in collision probability and impact. However, they achieve a better total performance because they more aggressively try to obtain access to the medium [7]. With enabled ARF in the upload scenarios, we observe a significant goodput reduction where the occurrence of collisions between data packets leads to the rate downshifts (falling to 15-20% of achievable goodput on average). This admits our argument about upload flows being more prone to ARF than download flows. To better understand the characteristics of cross-layer interaction

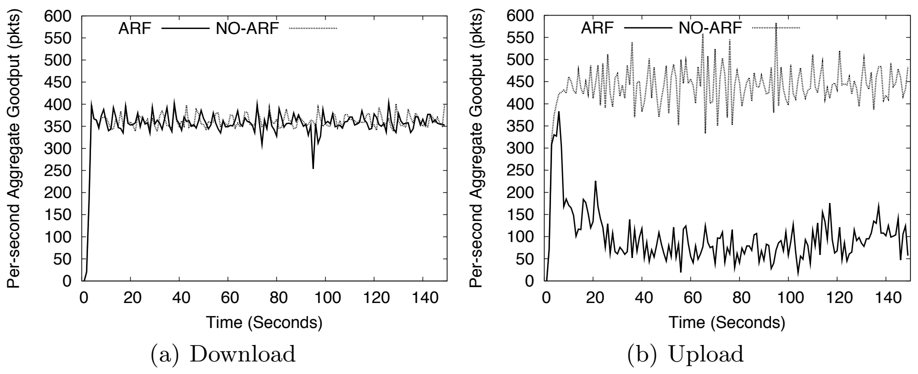


Fig. 1. Per-second aggregate goodput of 10 stations in 802.11b (wired delay= $50ms$).

between TCP and ARF, it is necessary to investigate the impact of collisions (packet losses) on TCP dynamism when ARF is enabled or disabled. The impact of the PHY transmission rate on the *cwnd* size for a single NewReno flow is plotted in Figure 2. Based on a rule-of-thumb the maximum *cwnd* size is equal to the bandwidth-delay product, which means that a lower PHY transmission rate (channel bandwidth) will lead to a lower maximum *cwnd* as depicted in Figure 2. Collisions at the MAC layer are perceived as packet errors (bit errors at the PHY layer). Hence, the impact of packet errors on *cwnd* of a single NewReno upload flow is investigated in Figure 3. The packet error rate perceived at the MAC layer of NICs was chosen to be high enough (PER=0.1) and uniformly distributed in order to present a clearly observable effect. An error model was defined at the MAC layer of AP and each of the wireless nodes to discard the received frames randomly with the uniform probability of e.g. 10%.

Based on Figure 3(a) it is obvious that frame losses due to packet error have a minimal impact on the *cwnd* size in the absence of ARF. This is because of the 802.11 MAC retransmission mechanism which ensures the receipt of a packet within a time that is normally shorter than a TCP retransmission timeout (RTO). Therefore an unsuccessful transmission of a MAC frame and/or its corresponding ACK frame will be compensated for by a frame retransmission, and consequently TCP *cwnd* will be kept almost intact at moderate values of the MAC frame error rate (PER of 0.1 is high here) . On the contrary, in the presence of ARF, the maximum *cwnd* drops to 60 packets (Figure 3(b)), indicating that the physical layer rate of 1 Mbps was mostly used as a result of ARF rate downshifts when PER=0.1.

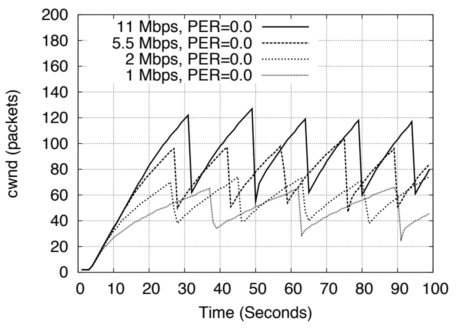
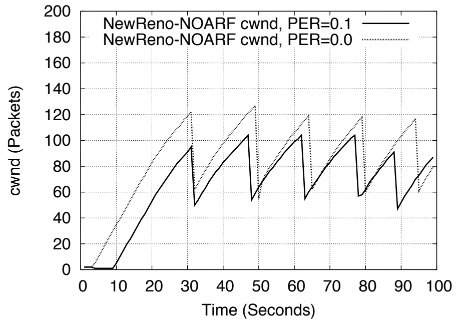
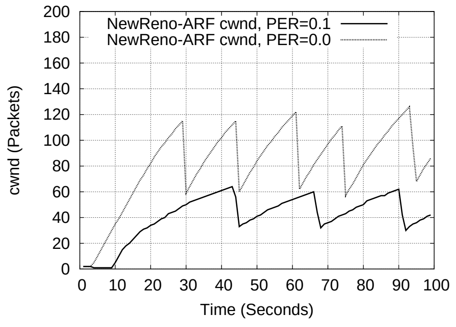


Fig. 2. *cwnd* of a single NewReno upload flow vs. PHY rate.

There is a difference in the sawtooth behavior of *cwnd*: with enabled ARF, *cwnd* growth is slower. This is because the delay (RTT) is increased by the MAC frame transmission and retransmission attempts with a reduced PHY rate; it therefore takes a longer time for a flow under ARF to fill up the AP buffer and experience a TCP packet drop. Figure 4 shows TCP's *RTT* for this scenario, revealing that retransmission attempts have a negligible impact on the *RTT* in the absence of ARF while rate downshifts significantly increase the *RTT* and consequently lead to slower *cwnd* growth in the presence of ARF.

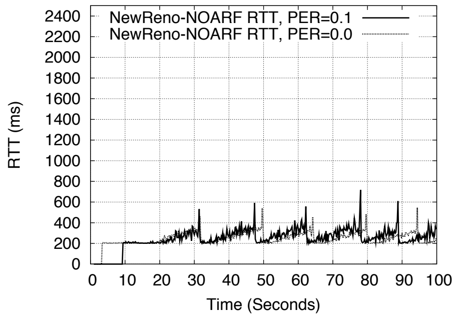


(a) Without ARF

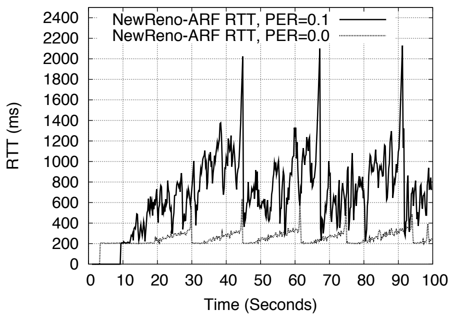


(b) With ARF

Fig. 3. *cwnd* of a single NewReno upload flow.



(a) Without ARF



(b) With ARF

Fig. 4. *RTT* of a single NewReno upload flow.

4 Performance of High-speed TCPs

TCP as the dominant transport protocol in the Internet has evolved in recent years to better utilize higher physical layer link speeds. Standard TCP performs poorly in networks with a large bandwidth$\times$delay product. There is a vast number of proposals for overcoming this limitation; some of them have been suggested for standardization in the Internet Engineering Task Force (IETF), and one of them (CUBIC [4]) is the default mechanism in Linux at the time of writing. These proposals are usually called *high speed TCP flavors*; in addition to CUBIC, we pick HTCP [10] and HSTCP [3] for our evaluation (because HTCP is commonly compared with CUBIC, and HSTCP is standardized). These variants can all achieve very large *cwnd* sizes compared to standard NewReno in long-distance networks.

To give a short overview of these TCP variants, in CUBIC the *cwnd* is a cubic function of time since the last congestion event with the inflection point set to the *cwnd* prior to the event. HTCP uses Additive Increase/Multiplicative Decrease (AIMD) to control TCP's *cwnd*. It increases its aggressiveness (in particular, the rate of additive increase) as the time since the previous loss increases. Finally, in HSTCP, when an ACK is received in the congestion avoidance phase, *cwnd* is increased by *a(w)/w*, and when a loss is detected via a triple duplicate ACK, *cwnd* is decreased by *(1-b(w))w*, where *w* is the current window size and the values of the functions *a* and *b* get larger and smaller with a growing *w*, respectively. We evaluated the performance of these high speed TCP flavors under the aforementioned uplink wireless-cum-wired scenario by using the ns-2 Linux TCP patch [11] with the original Linux kernel 2.6.16.3 source code.

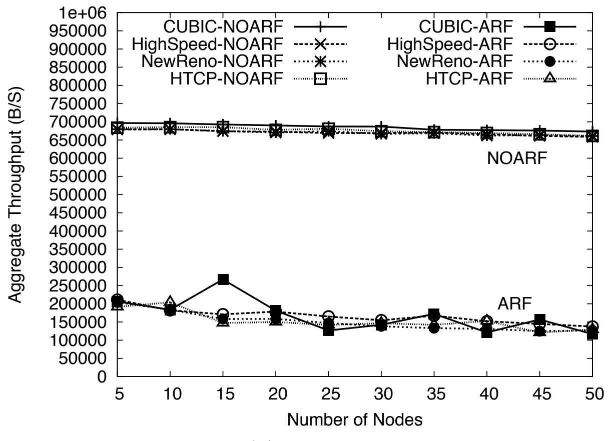
Figure 5 shows the average aggregate throughput of each TCP variant for a duration of 1000 seconds for a varying number of uploading stations where the wired delay is set to 100*ms*. In the absence of ARF, all high speed TCP variants perform almost the same and gain the maximum achievable throughput of 650-700 KB/s (Figure 5(a)) and 3.3 Mbps (Figure 5(b)) on average, which are the achievable throughputs associated to the maximum used wireless channel PHY rates of 802.11b and 802.11g (11 Mbps and 54 Mbps) respectively. Due to the limitation of the wireless channel bandwidth, they are not able to perform better than NewReno by letting the *cwnd* grow to a high value. The lines of HSTCP and New Reno are not distinguishable because their values are almost exactly the same.

When coupled with ARF, their aggregate throughput decline significantly, down to 100-200 KB/s on average in 802.11b (at the same level as NewReno), revealing the fact that rate downshifts as a consequence of collisions supersede the possible performance improvement of high speed TCP variants caused by their faster *cwnd* growth. It is worth to notice that these high speed TCP variants are designed to perform the same as NewReno for small *cwnd* sizes. Therefore, having an small *cwnd* because a low PHY rate (e.g. 1 Mbps in 802.11b) is used most of the time leads to the same achievable throughput as with NewReno. In 802.11g, HSTCP performs slightly better than the rest of TCP variants achieving only 1.4 Mbps on average while the throughput of other TCP variants decline to 1 Mbps on average.

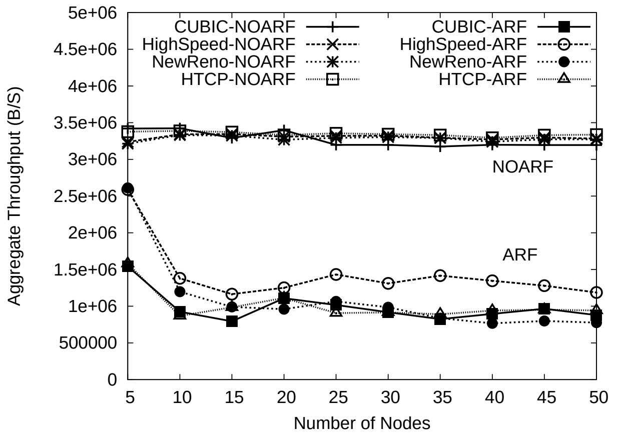
It is observable that the total aggregate throughput is almost invariant to the number of wireless nodes participating in contention due to "scalable performance" of TCP in uplink scenarios caused by TCP's self-clocking mechanism (similar to downlink scenarios). However, as justified previously, the impact of the collisions between data packets in uplink scenarios and their consequent effect of rate downshifts on *cwnd* size is totally different from the collisions between ACK packets in downlink scenarios.

To validate these results for larger bandwidth-delay products, where *cwnd* is expected to reach larger values, we repeated the simulations for different values of wired delay, ranging from 10 to 500*ms* with 10 uploading stations. Based on Figure 6 an extensive performance deterioration with enabled ARF is observable. Although different TCP variants perform almost the same in low bandwidth 802.11b wireless channel (Figure 6(a)), they behave differently under 802.11g channel with higher bandwidth where surprisingly, HSTCP and NewReno stand at the better throughput level compared to CUBIC and HTCP (Figure 6(b)).

Without ARF, however, CUBIC and HTCP are able to exploit almost the total provided bandwidth by 802.11g channel while the throughput of NewReno and HSTCP decline as wired delay increases. In 802.11b, all TCP variants gain the total achievable throughput. Conjointly, our results reveal the poor uplink TCP performance in general, and also for high speed TCP variants, in multi-rate 802.11 networks.

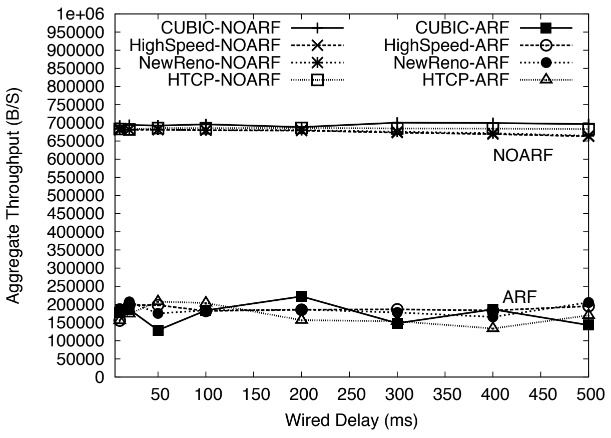


(a) 802.11b

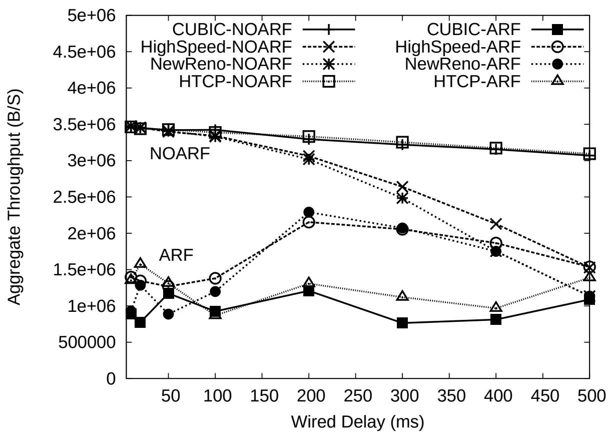


(b) 802.11g

Fig. 5. Aggregate throughput of TCP variants vs. number of nodes.



(a) 802.11b



(b) 802.11g

Fig. 6. Aggregate throughput of TCP variants vs. wired delay.

5 Conclusive remarks and future work

The uplink performance of TCP in multi-rate 802.11 networks has been studied and evaluated in this paper. It has been shown that ARF can significantly impede the TCP performance in uplink scenarios in contrast to the assumption that TCP's self-clocking behavior mitigates the impact of ARF in previous work. Real-life tests to validate these findings are left as our future work. Further, newer 802.11 standards with higher available bandwidth must be taken into consideration (e.g. 802.11n) and an analytical model will be proposed in the future. The impact of collisions on TCP performance with ARF under the noisy channel conditions should be studied analytically and experimentally. In addition, TCP could be modified in a way to perform better in the presence of ARF (e.g., we found that a buggy, severely rate limited version of HSTCP in the Linux kernel 2.6.16.3 sometimes performed better than the rest of the TCP variants). A performance improvement could therefore consider active queue management policies that make use of such a window size limitation; this will be investigated in the future.

References

1. Jaehyuk Choi, Kihong Park, and Chong kwon Kim. Cross-layer analysis of rate adaptation, DCF and TCP in multi-rate WLANs. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 1055 –1063, May 2007.
2. Sunwoong Choi, Kihong Park, and Chong-kwon Kim. On the performance characteristics of WLANs: revisited. *SIGMETRICS Performance Evaluation Review*, 33:97–108, June 2005.
3. Sally Floyd. Highspeed TCP for large congestion windows. RFC 3649 (Experimental), December 2003.
4. Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *SIGOPS Operating Systems Review*, 42:64–74, July 2008.
5. Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A rate-adaptive MAC protocol for multi-hop wireless networks. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, MobiCom '01, pages 236–251, New York, NY, USA, 2001. ACM.
6. Ad Kamerman and Leo Monteban. WaveLAN-II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, 2(3):118–133, August 1997.
7. Naeem Khademi and Mohamed Othman. Size-based and direction-based TCP fairness issues in IEEE 802.11 WLANs. *EURASIP Journal on Wireless Communications and Networking*, 2010.
8. J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. In *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1 –11, April 2006.
9. Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. IEEE 802.11 rate adaptation: a practical approach. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '04, pages 126–134, New York, NY, USA, 2004. ACM.
10. D. Leith and R. Shorten. H-TCP: TCP for high-speed and long-distance networks. In *Proc. PFLDnet, Argonne, 2004.*, 2004.
11. A Linux TCP implementation for NS2. `http://netlab.caltech.edu/projects/ns2tcplinux/ns2linux/`.
12. F. Maguolo, M. Lacage, and T. Turletti. Efficient collision detection for auto rate fallback algorithm. In *IEEE Symposium on Computers and Communications, ISCC 2008*, pages 25 –30, July 2008.
13. NS-2.29 Wireless Update Patch. `http://perso.citi.insa-lyon.fr/mfiore/`.
14. The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns/`.
15. Q. Pang, V.C.M Leung, and S.C. Liew. A rate adaptation algorithm for IEEE 802.11 WLANs based on MAC-layer loss differentiation. In *2nd International Conference on Broadband Networks. BroadNets 2005*, pages 659 – 667 Vol. 1, October 2005.

16. Yong Xi, Byung-Seo Kim, Ji bo Wei, and Qing-Yan Huang. Adaptive multirate auto rate fallback protocol for IEEE 802.11 WLANs. In *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pages 1 –7, October 2006.