Experimental Evaluation of the TCP Simultaneous-Send Problem in 802.11 Wireless Local Area Networks

Sumathi Gopal
WINLAB, Rutgers University
73, Brett Road
Piscataway, NJ 08854-8048
001 609 720 1202
sumathi@winlab.rutgers.edu

Dipankar Raychaudhuri
WINLAB, Rutgers University
73, Brett Road
Piscataway, NJ 08854-8048
001 732 445 0877
ray@winlab.rutgers.edu

ABSTRACT
This paper is an experimental follow up to our earlier paper [1] that investigated the TCP *simultaneous-send* problem which arises in infrastructure mode 802.11 wireless local area networks. In particular it was observed that for file transfer traffic, 802.11 wireless nodes have a sustained supply of packets to send and hence experience a relatively high rate of MAC contention. We showed that for TCP, this resulted in competition among data and ACK packets for channel access which caused considerable deterioration in flow throughput. Simulations of TCP ACK skipping as an alleviation to the problem, showed improvements as high as 100% when MAC retries were disabled. There were gains in other scenarios too albeit more moderate.

We evaluate the same TCP *simultaneous-send* problem with real world experiments on a wireless-cum-wired network testbed called ORBIT [2] at WINLAB, Rutgers University. ORBIT makes it feasible to conduct controlled and reproducible experiments in a wireless network scenario. The same network setup scenarios evaluated in simulations were considered here., particularly – scenarios with and without MAC retries, multiple TCP flows and multiple skipped ACKs. However not all scenarios could be reproduced in experiments for logistical reasons. In all, the experimental results confirm the original hypothesis on the detrimental effects of *simultaneous-send* and corroborate the advantages of ACK skipping, However the percentage gains in TCP throughput are far more moderate as compared to those observed in NS simulations. A reason could be differing TCP implementations, particularly with not all TCP optimizations implemented in NS. We share the experiences and challenges faced, particularly given that this work is among the first of its kind for testbed evaluation of transport protocols over wireless networks.

SIGCOMM'05 Workshops, August 22–26, 2005, Philadelphia, PA, USA.

Categories and Subject Descriptors
C.4 [Performance of Systems] :Measurement techniques
C.2.1[Computer-Communication Networks]: Network Architecture and Design - wireless communication
C.2.5 [Local and Wide Area Networks]: TCP protocol performance over 802.11 wireless networks

General Terms
Measurement, Performance, Experimentation

Keywords
Wireless networking, Experimental evaluation, full-fledged wireless network testbed, controlled wireless environment, TCP, 802.11 MAC, DCF, skipped ACKs, delayed ACKs, simultaneous-send

1. INTRODUCTION
802.11 wireless networks are now widely deployed in offices, homes and hotspots, supporting channel rates up to 54 Mbps. They operate in the Infrastructure mode where packets between any pair of nodes are relayed through an access point (AP). The Distributed Coordination Function (DCF) mode of MAC access is typically used in infrastructure networks, where all nodes including the AP have the same priority for channel access. Majority of the traffic in today's networks constitute TCP flows employed by applications including email, file transfer, web browsing and database access that require reliable end-to-end transport.

Numerous research papers have been published in the area of wireless networks. Most of them test performance and new protocols with network simulators such as NS[3] and OPNET[4]. These tools are excellent sandboxes to check correctness and understand the detailed operation of protocols. However just this not sufficient. These higher layer simulation tools fail to capture the operation characteristics of the protocol in real systems particularly in wireless networks because of the difficulty in representing the physical medium. NS does not implement the physical layer of the network stack nor does it implement the characteristics of the physical medium. Further it supports just the basic features of MAC protocols such as IEEE 802.11, hence making it essential to evaluate protocols by real world experimentation.

This said, experimentation in wireless network scenarios poses distinct challenges. The inherent broadcast nature of wireless links makes them vulnerable to environmental factors such as materials composing the floor, ceiling and furniture, opening and closing of doors and even movement of people. Hence experimental results obtained in one indoor environment are rarely reproduced elsewhere. The ORBIT testbed [2] serves to alleviate this problem. The testbed operated remotely, facilitates a controlled and hence reproducible environment. It currently contains a grid of 64 high-end nodes with multiple wired and wireless interfaces. The two wired interfaces are used to network nodes in separate control and data planes. The wireless interfaces support 802.11 a/b/g networks. Each node runs Debian linux. The nodes are setup so that the user may reimage nodes with her choice of kernel and requisite software. The grid is a convenient tool to conduct wireless as well as wired experiments. We evaluate TCP throughput in a 802.11b Infrastructure network under a variety of traffic scenarios such as single/multiple flows, with/without MAC retries, and short-lived/long-lived transmissions.

There are several papers that report on experimental evaluation of protocols for wireless networks, such as Ramjee et. al's Ack Regulator [5] and Paul et. al's AIRMAIL [6]. These are for protocols over cellular networks (WWANs), and were tested over proprietary systems. Those that exist for 802.11 wireless networks [9] are difficult to reproduce in a generic setup. To the best of our knowledge ours is among the first experimental evaluations of transport protocols over a full fledged *open-access* wireless network testbed for IEEE 802.11 networks., that is in turn becoming the de facto sandbox for experimentation.

The rest of this paper is organized as follows. In Section 2 we review the TCP *simultaneous-send* problem presented in [1]. Section 3 presents the experimental setup on the ORBIT testbed. Results, analysis and comparison to NS results are presented in Section 4 and we conclude in Section 5.

2. PROBLEM DESCRIPTION

The 802.11 MAC may be operated in one of two modes – Distributed Coordination function (DCF) and Point Coordination Function (PCF). This paper considers the DCF mode. Here all nodes including the Access Point (AP) have equal priority for channel access. Channel contention happens on a per-packet basis. A node waiting to transmit a packet, first senses the channel to be idle for a certain duration (called DIFS [15]), then selects a backoff slot randomly, based on a uniform distribution in a contention window (CW). The packet is transmitted if the channel is still idle in the selected slot. Otherwise, the node waits till the channel is idle again, backs off only for the requisite slots before attempting to transmit. The node learns of a successful transmission when it receives an acknowledgement (MAC-ACK) from the destination.

We evaluate the likelihood of MAC failure when nodes operating in DCF mode have a consistent supply of packets to send, causing them to persistently contend for channel access. Throughput derivations of 802.11 MAC have typically assumed Poisson packet arrival per backoff slot [8]. Instead, suppose there are (N+1) nodes with a continuous supply of packets that causes them to contend for the channel far more consistently. A transmission is successful only if no other node transmits in the same backoff slot. This likelihood of all nodes selecting independent slots is

$$1*\left(\frac{CW-1}{CW}\right)\left(\frac{CW-2}{CW}\right)\cdots\left(\frac{CW-N}{CW}\right)$$

$$=\left(\frac{(CW-1)!}{(CW-N-1)!*(CW)^N}\right)$$

Hence the likelihood of at least two nodes selecting the same slot is

$$1-\left(\frac{(CW-1)!}{(CW-N-1)!*(CW)^N}\right) \qquad (1)$$

This is the likelihood of a failed transmission for nodes having a consistent supply of packets to send.

In this paper we evaluate TCP behavior in 802.11 DCF mode of operation. TCP is based on a sliding window protocol that enables several successive data segments to be transmitted before receiving an acknowledgement (TCP ACK) for an earlier segment. TCP generates data packets in bursts proportional to the increase in congestion window size (*cwnd)*. When ACK-bunching happens there are bigger bursts. Thus several packets may arrive at the wireless interface in the TCP sender node, causing a consistent supply of packets to transmit. With no loss of generality, we may assume that the number of TCP ACKs produced is approximately proportional to the number of data segments. Hence we may expect a consistent supply of packets available to transmit in the MAC layer, even when nodes are relaying TCP ACKs.

When there is a single TCP flow, the likelihood of same slot selection is simply (1/CW)*(1/CW)*CW. If the previous transmission was not a failure, CW=CWmin=32 and the likelihood is 3%. Otherwise, CW doubles (binary random backoff) and that likelihood halves to 1.5%. For N=3 (3 TCP flows), and CW=32, likelihood of same slot selection, and hence MAC failures is 17.6% from Equation (1).

In particular, we analyze the scenario where the mobile nodes are TCP data sources uploading files to remote receivers (via the access point (AP)) as depicted in Figure 1. Because of two way traffic involved in a TCP flow due to data and ACK segments traversing in opposite directions, a single TCP flow constitutes two contending wireless nodes in an 802.11 DCF Infrastructure network. One contending nodes is always the AP. In our case, the AP relays TCP ACKs from the remote TCP receiver to the mobile node.

As explained earlier, the nature of TCP traffic causes the 802.11 MAC to often have a continuous supply of packets to send. This increases the likelihood of two or more nodes selecting the same backoff slot. Hence TCP data and ACK packets compete for channel access causing MAC transmission failures. The problem is particularly significant with a single TCP flow.

With reference to the network in Figure 1, the AP and TCP source node often transmit to each other in the same backoff slot. Neither node detects the packets, since the hardware implementation prevents them from sending and listening at the same time. This is the *simultaneous-send* problem. With no channel errors, disabled

MAC retries and a single TCP flow, *simultaneous-send* is the sole cause of packet losses. For multiple nodes, the same-slot selection phenomenon also manifests as collisions where two or more nodes transmit in the same MAC backoff slot, and a third listener hears a combined garbed signal. If MAC retries are enabled, *simultaneous-send* problem is not experienced at the transport layer. Instead, packet losses now occur due to MAC queue overflow from TCP bursts. While we could confirm this latter problem in NS simulations with trace files, it is far more challenging to do so in kernel implementations of TCP. Writing to trace files introduces significant computational overhead that interferes in the proper operation of the experiment. Instead it requires complicated logging procedures, that are beyond the scope of this paper.

It was demonstrated in [1] about how *skipping TCP ACKs* alleviated the *simultaneous-send* problem. The reasoning was that with no MAC retries, the number of ACK segments competing for channel access reduced, hence reducing MAC contention. For the case with MAC retries, ACK skipping controlled the growth of the TCP congestion window while in slow start, reducing packet bursts and hence minimizing MAC queue overflows. The NS simulations showed drastic throughput improvement with and without MAC retries. It was a 100% gain with a single flow, no MAC retries and 1 ACK skip. For the case with MAC retries, and the case with a single flow saw 30% improvement. However with multiple ACK skips gain in TCP throughput dropped significantly, particularly when no MAC retries were used. This may be due to ACK-starving of the TCP sender and also due to packet losses from burstier TCP.

Ack skipping is similar to delayed ACKs of TCP. With the latter, the receiver delays sending an ACK so that its own data may be piggybacked. With ACK-skipping, a specific number of ACKs are skipped irrespective of the time lag between them. We use ACK skipping here, since the effect of reduced TCP control traffic is more easily tracked with this rather than with delayed ACKs.

Results in this paper of the same experiments in the ORBIT wireless testbed, show more moderate but consistent improvement in TCP throughput with ACK skipping. The highest gain is about 35% over when there is no ACK skipping.

3. EXPERIMENTAL SETUP

All experiments were conducted on the ORBIT wireless testbed

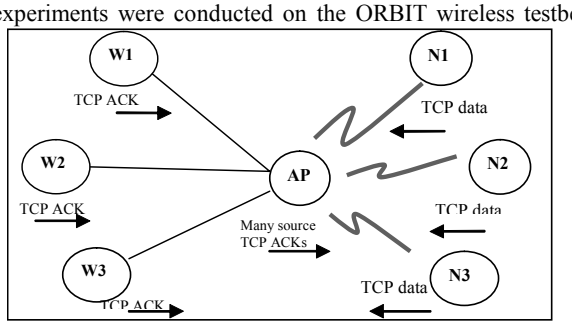


Figure 1: Network setup

[2]. It was comprised of 64 nodes placed in a square grid of 8 rows and 8 columns as depicted in Figure 2. Each node had multiple wired and wireless network interfaces. One wired interface was reserved for the control plane that enabled the grid to be operated remotely. The grid supported complete remote access and was supplemented by elaborate data logging and collection servers so that measurements did not interfere with the experiments. Each ORBIT radio node had a 1GHz VIA C3 processor, two experimenter-accessible 100BaseT Ethernet interfaces (for data and control). Every node had a dual-band (802.11a/b/g) radio interface that had either of Atheros-based or Intel-based chipsets. Some nodes had an additional radio interface with a Cisco Aironet 350 series-based PCMCIA 802.11b card. The nodes ran Debian Linux with 2.6.10 kernel version. The

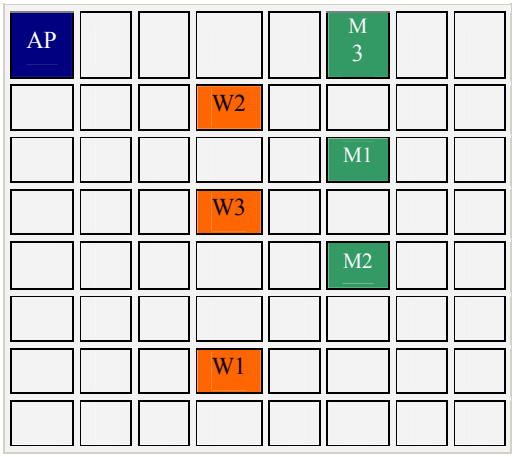


Figure 2: Experimental setup on the ORBIT grid; Wi are wired nodes; Mi are wireless nodes in Infrastructure mode deliver TCP data to Wi via the AP

control plane enabled nodes to be reimaged with new kernels as desired by the experimenter.

All experiments were carried out in an 802.11b Infrastructure network, so as to match those in the NS simulations. Despite the WiFi interoperability standard for 802.11, the network cards differed significantly in other features they provided. For example, of the three wireless cards supported in the ORBIT testbed, the Cisco cards alone supported change in MAC retries. Similarly for setting up access points and sniffers. Only the Atheros cards could be operated in the "Master" and "Monitor" modes required for access point and sniffing operations respectively. Since our experiments involved disabling MAC retries, we need to used Cisco cards for the client nodes. However the Atheros card interface had to be used for AP setup. It was hence not possible to disable MAC retries completely.

Both layer 2 and layer 3 settings were required to establish the wireless infrastructure network in the grid. Manual entries in routing tables of end nodes had to be made to ensure packet routing via the AP and IP forwarding had to be enabled in the node that was set up as AP. The infrastructure network depicted in Figure 1 was setup in the grid as shown in Figure 2. Wireless nodes were selected so that they were equidistant from the AP. The distance between AP and wireless nodes was approximately 2 meters. There was no other interfering traffic, or noise of channel fading during the experiments. Hence in all experiments, packet losses were attributed in full to MAC transmission failures.

TCP implementation in the kernel (version 2.6.10) was modified to incorporate ACK skipping. We ensured that this modification in TCP behavior did not interfere with regular error control mechanism of TCP. ACKs were not skipped when duplicate ACKs were to be sent, or a received data packet was out of order. Further, ACK skipping was set to start only after TCP operation reached steady-state. It is important to mention that, although the kernel modification was a single line, one had to be very careful to avoid introducing sub-optimality. For example, a single print statement to log skipped ACKs reduced TCP throughput to a 100^{th} of its previous value. This was due to the expensive per-packet file-write overhead in kernel operation. TCP segment sizes were set to 1000 bytes using TCP socket options.

Flows between multiple node pairs were overlapped by trial and error with the help of various scripts and calculated delays, in order to produce simultaneous flows. 6 long and 6 short flows were carried out back to back during each test run. An average of these was obtained to cancel any channel capture effects that were reported to occur from variation in card sensitivity [7] . Each data point in the graph was an averaged value of test runs of multiple flows.

The physical rate was fixed at 11 Mbps. Fixing rates was yet another feature that not all wireless cards supported. They typically had autorate selection mechanisms built in that overran manual rate fixation [7]. To our advantage, Cisco cards allowed rate fixation. The RTS/CTS feature was turned off. These settings could be modified using the *iwconfig* tool.

Unlike in a simulator, status parameters in real network interfaces were difficult to be tracked. These parameters we desired to track included interface queue size, MAC contention rate, link quality etc. The ORBIT grid incorporated facilities to track several layer 1 and layer 2 parameters at runtime, although those options have not been availed in this work.

The following parameters were considered for experiments just like in the NS simulations: 1. Length of TCP flow 2. MAC retries 3. Number of ACKs skipped 4. Number of simultaneous flows.

To reflect short-lived and long lived TCP flows, we used data transfers of 100kB and 6MB respectively. Long-lived TCP connections tested the stability of the protocol adaptation and confirmed its validity, while short-lived TCP connections supplied a view of TCP operation during its transient state.

Finally, the default retry limit is set to 16 in Atheros and Cisco cards. In the 802.11 network, a failed MAC transmission is known only by the lack of a returning MAC ACK.

4. RESULTS AND ANALYSIS

Figures 3-6 present results from the experiments. Graphs from NS simulations reported in [1] are included in the Appendix for comparison. . First it is important to notice that the TCP throughput obtained with no adaptation was itself significantly higher than in simulations. This could be from differing TCP and 802.11 implementations. However, we identify that the primary reason for this discrepancy is from the implementation of the interface queue in NS2. With the default of 50 packets, TCP packets were lost while in slow-start due to MAC queue overflows. This caused timeouts in TCP that significantly degraded overall throughput. This phenomenon did not happen while operating real-world TCP, as the operating system in the

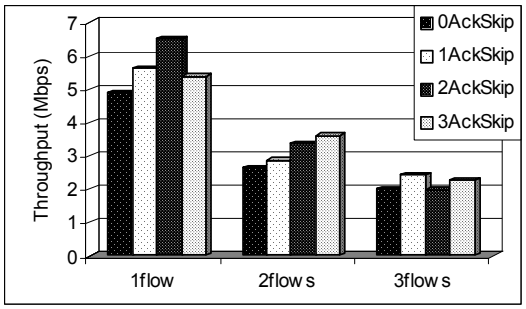


Figure 3: Short-lived TCP flow WITH MAC retries

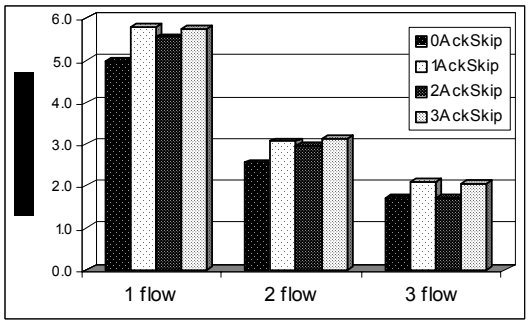


Figure 4: Long-lived TCP flow with MAC retries

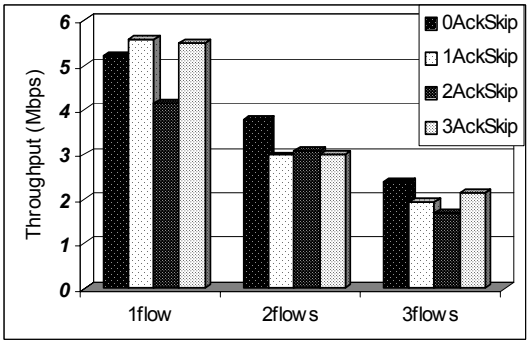


Figure 5: Short-lived TCP flow NO MAC retries

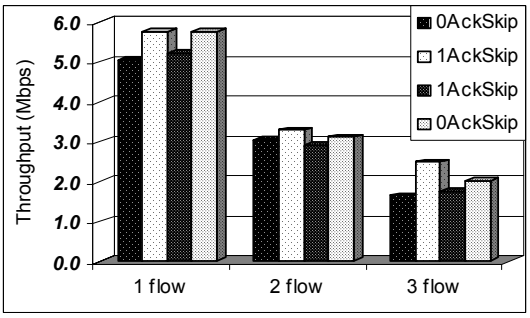


Figure 6: Long-lived TCP flow NO MAC retries

node acted as an intermediary between the TCP socket buffer and the interface queue in the network card. The OS delivered a packet from the TCP send-socket-buffer to the network interface queue, only when the interface driver set a memory availability flag. If the TCP socket buffer was full, no new bytes were

accepted from the application (the send() function returned an error in the application operating the TCP socket).

Since base throughputs in NS simulations differed from those in testbed experiments, we compare *patterns* in gains achieved rather than the actual gains themselves.

4.1 Case of enabled MAC retries

The Atheros cards had a default MAC retry setting of 16 that could not be changed. Hence that value was used even with Cisco cards. On the other hand, the maximum retries used in NS simulations was 8.

Just as in simulations, ACK skipping consistently improved TCP throughput (in this case with MAC retries) even with multiple simultaneous flows. MAC retransmissions were tracked by means of standalone sniffers. These sniffers comprised of Atheros cards in monitor mode and the *tcpdump* software. The particular gain patterns for short lived and long lived flows differed from those of NS simulations. This was probably due to the NS2 phenomenon of interface queue overflows during TCP slow start, that did not occur in real experiments. The throughput gain in real experiments seemed to come directly from reduced MAC contention due to fewer TCP ACK packets.

Long lived flows saw consistent throughput gain even with 3 skipped ACKs, whereas the gain dropped with such high ACK skips for short-lived flows. This could be because short-lived flows spent a higher percentage of their operation in slow start mode. In this mode, increase in TCP congestion window was proportional to the actual number of incoming ACK segments, even if they were cumulative ACKs. In the congestion-avoidance mode on the other hand, increase in congestion window was proportional to the number of data segments acknowledged. With three or more ACKs skipped, short-lived flows experienced ACK starving and hence had subdued throughput.

Overall, ACK skipping helped the case with MAC retries. Both short-lived and long-lived flows gain from this adaptation.

4.2 Case of disabled MAC retries

We reiterate that it was not possible to produce the case when MAC retries were completely disabled in the wireless infrastructure network, as this feature was not supported in Atheros cards that were used for AP. However MAC retries could be disabled in non-AP wireless nodes where Cisco cards were used. For the traffic scenario considered, this meant that MAC retries were disabled for TCP data segments, while the TCP ACK segments that were relayed by the AP enjoyed MAC retries. This was also confirmed with a standalone sniffer (Described above). In liu of this, results from simulations and experiments for this case cannot be compared.

From these results we infer that ACK skipping was more favorable for long lived rather than for short-lived flows, when MAC retransmissions were available only for TCP ACK packets.

4.3 Other Observations

The graphs indicate that for the default case with no ACK skips, TCP throughput was better with no MAC retries for TCP data segments than with MAC retries. This could be because link layer retransmissions produce variations in RTT for TCP, reducing its performance. This could possibly imply that TCP does a far better job handling MAC congestion by itself rather than with link layer retransmissions. This observation requires further study.

There was much better channel utilization with multiple simultaneous flows that with a single flow. This can be explained as due to the 802.11 backoff overhead. The 802.11 DCF backoff mechanism caused an average overhead of 300ms. With a few flows this overhead reduced as contention slots were staggered. When the likelihood of same slot selection was still reasonably small, there was a throughput improvement. However as the number of flows increased, the likelihood of same slot selection also increased, resulting in more MAC failures and subsequent degradation in throughput.

5. CONCLUSION

In this paper we have investigated TCP behavior in a 802.11b wireless Infrastructure network by means of experiments in a wireless network testbed. We have compared results obtained with similar experiments done in NS2 simulator. We conclude that TCP ACK skipping indeed improves TCP performance in real-life wireless LANs. Although the original goal was to explore the *simultaneous-send* problem reported in [1], several other insights were also obtained. The NS2 simulations showed the *simultaneous-send* problem manifest when MAC retries were disabled. However we were unable to reproduce the NO MAC retries case. Unlike in the NS2 simulator, various status indicators cannot be tracked in real experimentation. Instead the *simultaneous-send* phenomenon was observed using standalone network sniffers.

In summary, results in this paper corroborate the *simultaneous-send* problem where TCP data and ACK packets compete for channel access in 802.11 WLANs and cause significant throughput degradation.

This study has also lead to several other key insights, namely:

1. Its very important to confirm transport protocol performance observations, and validity of new protocols for wireless networks, with real-life implementations and testing on real testbeds.

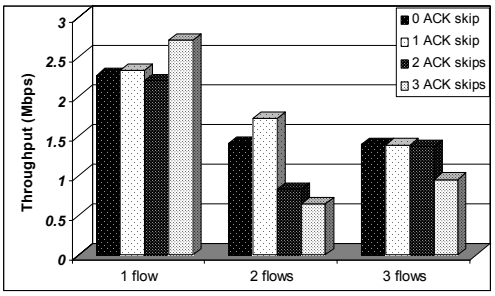
2. However, simulating the protocols in event-driven simulators such as NS, is also important, since they provide means to track protocol states while in operation, without affecting the operating plane. In real testbed scenarios, measuring protocol status in real time could significantly downgrade protocol performance as our experience with logging ACK sequence numbers in the kernel showed.

3. Last but not the least, we reiterate the importance of evaluating higher layer protocols in controlled and reproducible environments such as the ORBIT testbed. Simulators for these protocols fail to completely capture all phenomena associated with operating higher layer protocols over wireless links with medium access protocols.

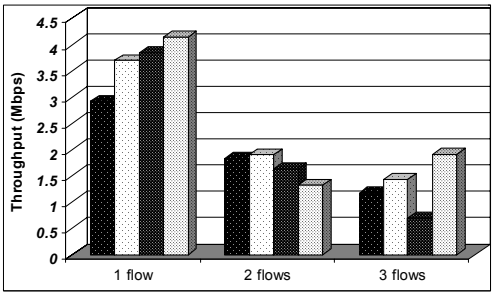
6. ACKNOWLEDGEMENTS

We would like to thank Faiyaz Ahmed and all the members in the ORBIT team at WINLAB, Rutgers University, for their immense help that made this work possible. We would also like to thank Vinay Iyengar for his help with data collection and presentation.

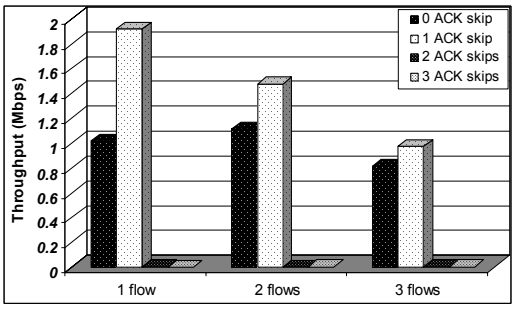
7. APPENDIX



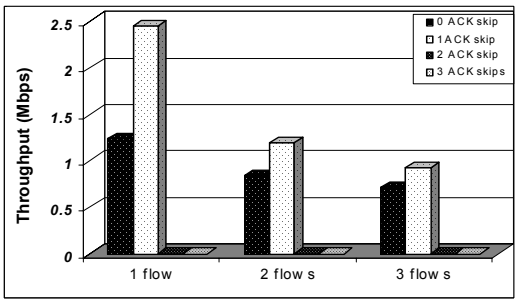
NS result: Short-lived TCP flow with MAC retries



NS Result: Long-lived TCP flow with MAC retries



NS Result: Short-lived TCP flow; NO MAC retries



NS result: Long-lived TCP flow NO MAC retries

8. REFERENCES

[1] Gopal, S.; Paul, S.; Raychaudhuri, D., "Investigation of the TCP Simultaneous-Send Problem in 802.11 Wireless Local Area Networks", IEEE International Conference on Communication (ICC) 2005, Seoul, South Korea, 16-20 May 2005

[2] ORBIT: Open Access Research Testbed for Next-Generation Wireless Networks www.orbit-lab.org

[3] Network Simulator – 2 (NS2): http://www.isi.edu/nsnam/ns/

[4] OPNET simulator: http://www.opnet.com/

[5] M C Chan and Ramchandran Ramjee:TCP/IP performance over 3G wireless link with rate and delay variation ACM mobicom 2002

[6] E. Ayanoglu, S. Paul, T. F. LaPorta, K. K. Sabnani, and R. D. Gitlin. AIRMAIL: A link-layer protocol for wireless networks. 1:47–60, February 1995.

[7] Haris Kremo, "IEEE 802.11 Medium Access Protocol: An Experimental Case Study", MS thesis, WINLAB, Rutgers University. April 2005

[8] Kamerman, A.; Aben, G., "Net throughput with IEEE 802.11 wireless LANs", IEEE WCNC. Sept 2000. Volume 2

[9] Garcia, M. Choque, J. Sanchez, L. Munoz, L. "An experimental study of Snoop TCP performance over the IEEE 802.11b WLAN", The 5th International Symposium on Wireless Personal Multimedia Communications, Oct 2002.

[10] Kherani, A.A.; Shorey, R.; "Performance improvement of TCP with delayed ACKs in IEEE 802.11 wireless LANs", Wireless Communications and Networking Conference, March 2004, Volume 3

[11] Eitan Altman, Tania Jiménez "Novel Delayed ACK Techniques for Improving TCP Performance in Multihop Wireless Networks" Personal Wireless Communications 03, Venice Italy.

[12] Shugong Xu; Tarek Saadawi; Myung Lee; "On TCP over wireless multi-hop networks", IEEE Military Communications Conference, Oct 2001.

[13] Wu, H.; Peng, Y.; Long, K.; Cheng, S.; Ma, J; "Performance of reliable transport protocol over IEEE 802.11 wireless LAN: analysis and enhancement", INFOCOM 2002.

[14] IEEE 802.11, 1999 Edition (ISO/IEC 8802-11: 1999) Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications