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

Sumathi Gopal Dipankar Raychaudhuri
WINLAB, Rutgers University

EWIND 2005
ACM SIGCOMM Workshop on Experimental Approaches to Wireless Network Design and Analysis

22nd August 2005

Outline of talk

- Introduction
- NS results
- Testbed Experiments
- Results and analysis
- Conclusion

Introduction

IEEE 802.11 DCF mode

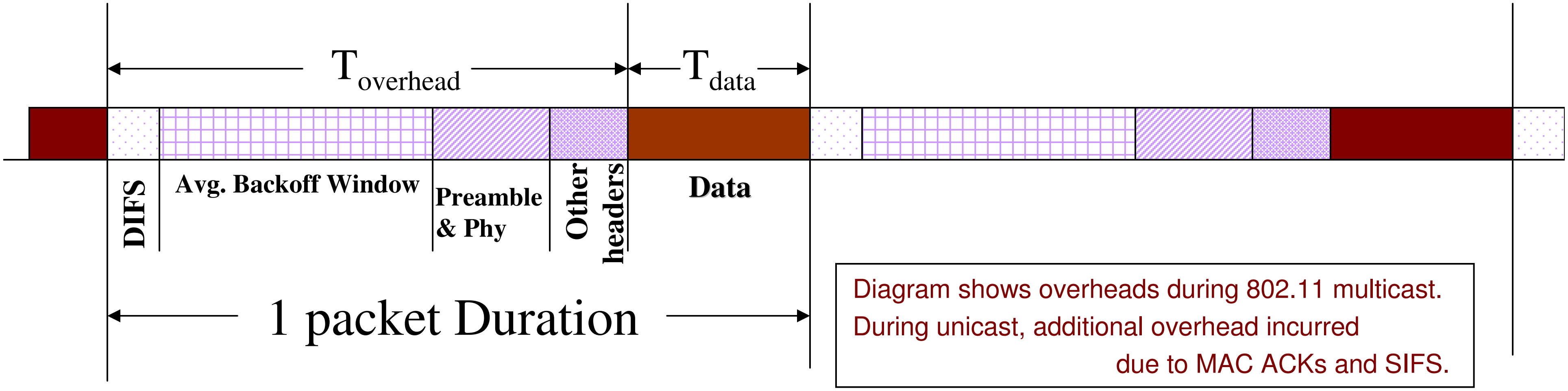


Diagram shows overheads during 802.11 multicast.
During unicast, additional overhead incurred due to MAC ACKs and SIFS.

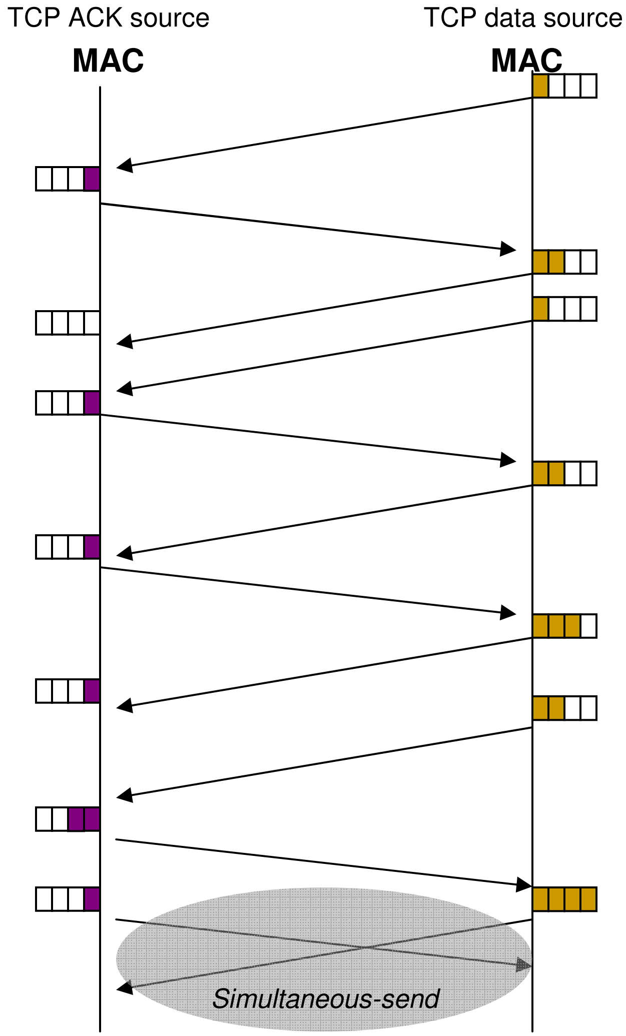
- Distributed access; equal priority for all nodes
- Large overhead per packet;
  - MAC contention (random backoff) primary contributor
  - Preamble and Phy headers transmitted at 1/2Mbps
  - DIFS
  - Other headers – LLC, IP, TCP
- For a 1kB packet, channel usage efficiency < 54%; < 2% for a 16Byte packet!

The TCP *simultaneous-send* problem

first reported in

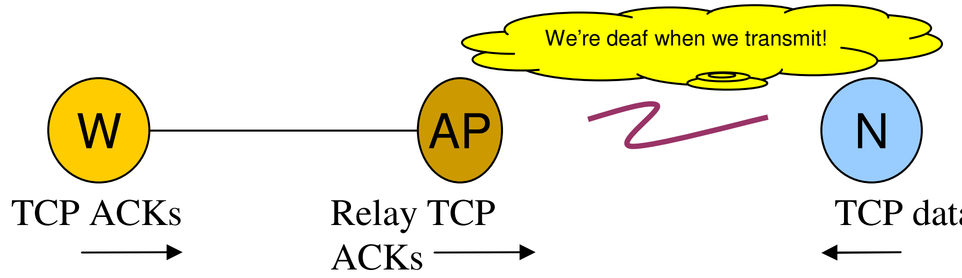
S Gopal, S Paul, D Raychaudhuri, "Investigation of the TCP Simultaneous-Send Problem in 802.11 wireless Local Area Networks", ICC May 2005, Seoul, South Korea

Persistent channel contention with TCP



- With TCP, packet arrival at 802.11 MAC is *not Poisson* and MAC contention happens far more often
- TCP slow-start causes a continuous supply of packets in the MAC queue
- Channel contention for every packet

TCP Simultaneous-send problem in 802.11 wireless networks



- TCP causes persistent MAC contention in 802.11 (Shown in previous slide)
- Hardware implementation – cannot send and receive at the same time
- Likelihood of at least 2 nodes of N nodes selecting the same backoff slot is:

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

- ***Simultaneous-send*** problem occurs with a 3% likelihood
- TCP packet losses, and hence retransmission timeouts, reduce throughput

Alleviating simultaneous-send with *TCP ACK skipping*

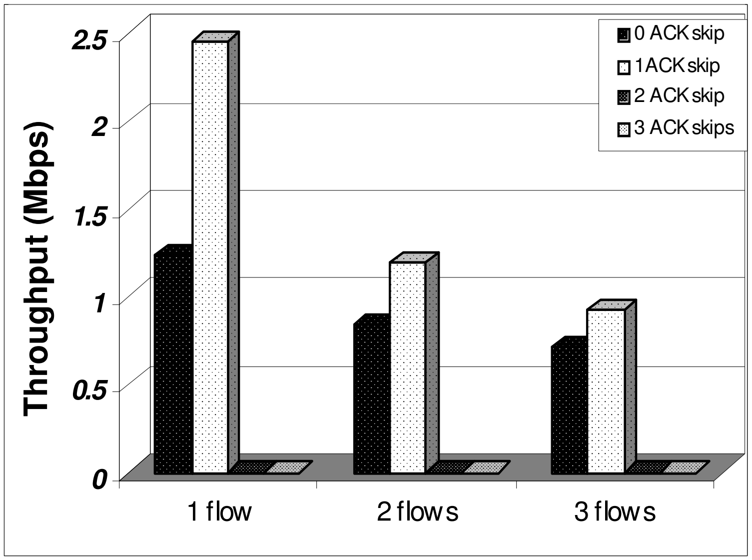
- Assumptions:
  - Delayed ACK adaptation not enabled;
  - There is typically one ACK for every data packet
- TCP congestion window can tolerate skipping to some extent due to cumulative nature of TCP ACKs.
- ACK skipping reduces contention for data packets
  - 1 ACK skip reduces AP load by half
- However skipping too many ACKs is detrimental to TCP throughput due to constrained growth of the congestion window.

NS results

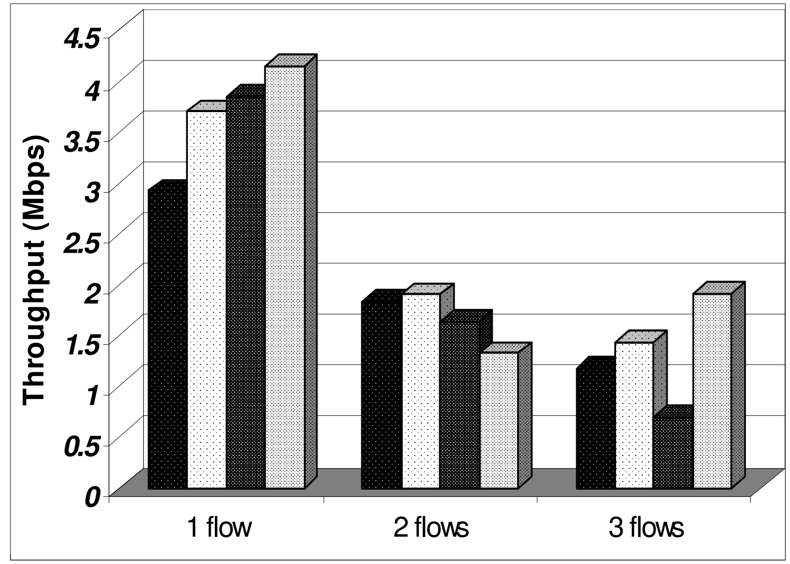
first reported in

S Gopal, S Paul, D Raychaudhuri, "Investigation of the TCP Simultaneous-Send Problem in 802.11 wireless Local Area Networks", ICC May 2005, Seoul, South Korea

NS Results



Disabled MAC retries

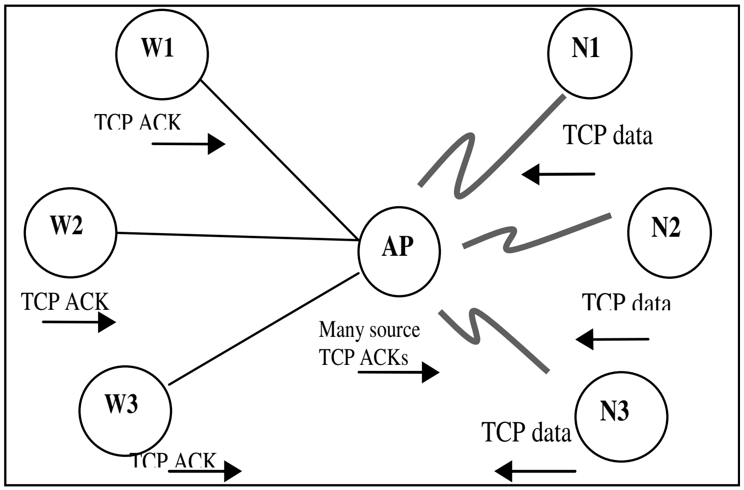


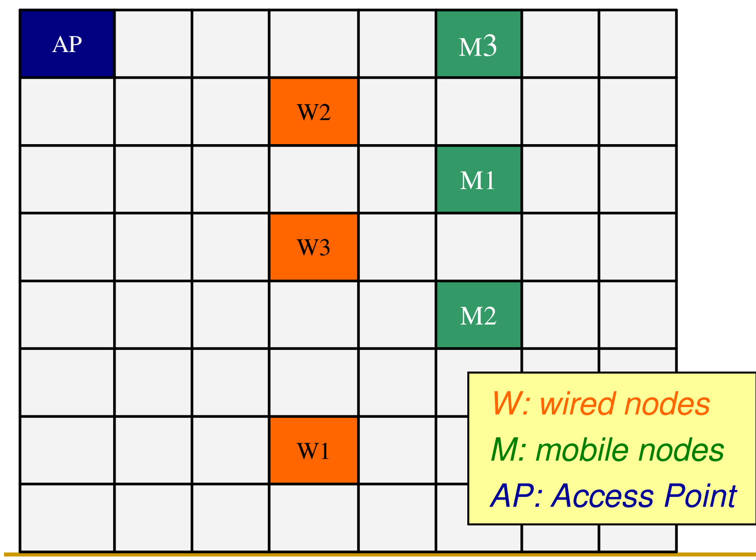
Enabled MAC retries

	Disabled MAC Retries	Enabled MAC Retries
Throughput Gain on 1 ACK skip	100 %	30%
Cause of gain with ACK skip	Reduced contention	Slower growth of cwnd during slow start
Higher ACK skips	Lost ACKs too costly; ACK starving degrades throughput	MAC retries compensate for ACK losses

ORBIT testbed experiments

Testbed Experiment setup





- Cisco and Atheros cards for wireless interfaces
- Configuration settings in Layer 2 and Layer 3
- All nodes in hearing range of each other
- No interfering traffic or noise, hence all packet losses due to MAC contention
- TCP code in kernel modified for ACK skipping

Some setup details..

- Only Atheros cards operated in “Master” and “Monitor” modes
- Only Cisco cards support MAC retry modification; Atheros cards do not.
- AP and Sniffer: Atheros cards; Wireless nodes: Cisco cards
- In the Disabled MAC Retries case: Retries disabled only for TCP data. Full retries for TCP ACKs.
- Cisco cards allow rate fixation; auto rate adaptation disabled
- Settings made with *iwconfig*

Setup details…

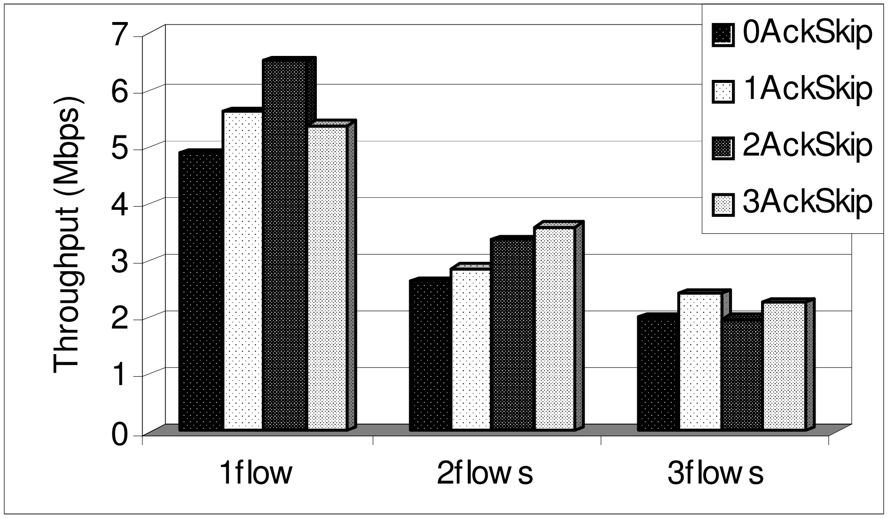
- Excellent TCP code in kernel 2.6.10 – well commented.
- Stevens “TCP/IP Illustrated” not useful. Code structure is very different
- Control plane of ORBIT testbed very handy for kernel modifications

Testbed Experiments

- Each throughput point in the graph is an average of 6 trials, and the simultaneous flows
- Short-lived flows - 100kB; Long-lived flows - 6MB file transfer
- Phy rate fixed at 11Mbps
- RTS/CTS disabled
- Maximum 16 MAC retries (default setting in Atheros cards)

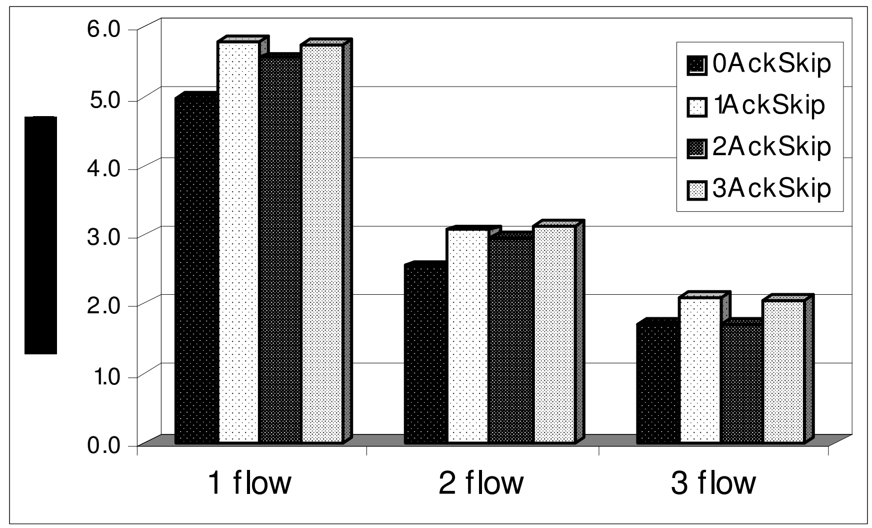
Testbed Results

Results (MAC Retries Enabled)



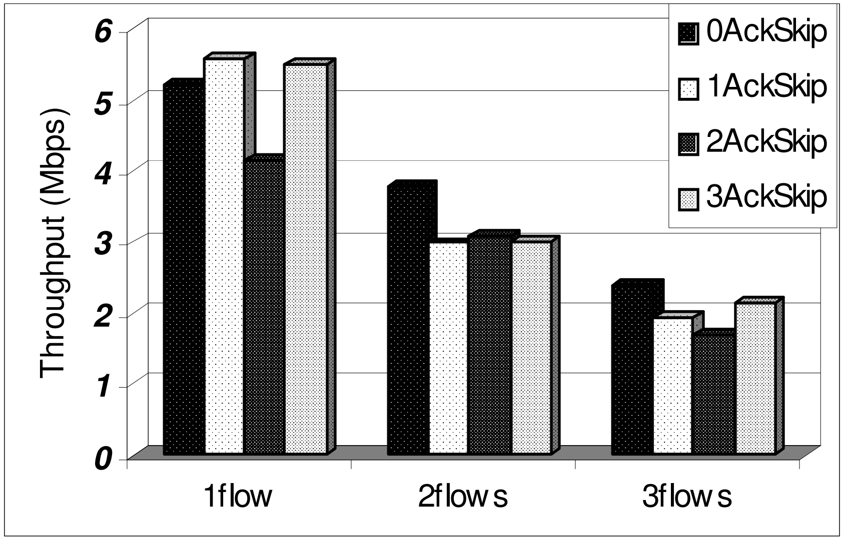
Short-lived flows

Consistent throughput gains with ACK Skipping



Long-lived flows

Results (Disabled MAC retries)

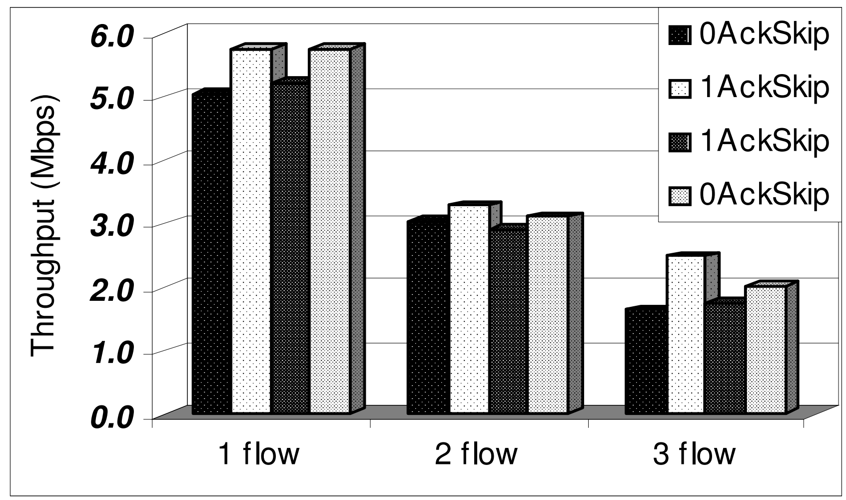


Short-lived flows

Wireless Nodes (TCP data) retry disabled

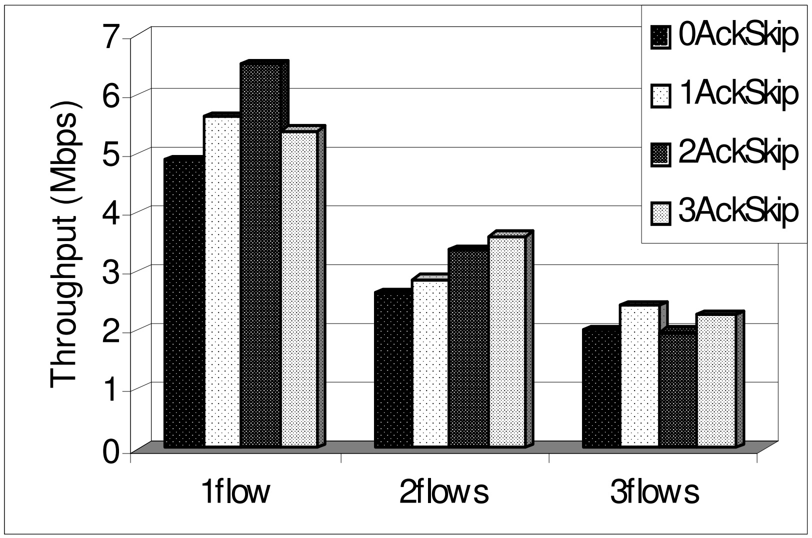
AP (TCP ACKs) retry NOT disabled

Sniffer showed TCP retx even with a single flow

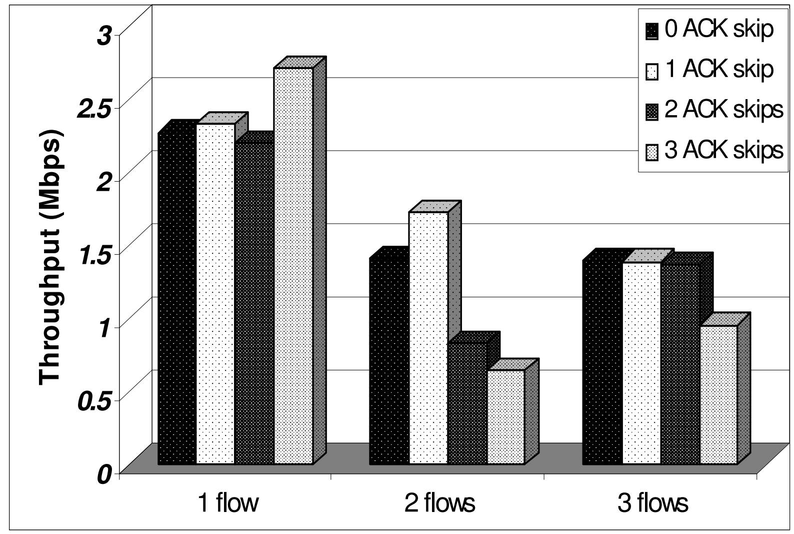


Long-lived flows

Comparing Testbed and NS results (Enabled MAC retries)



Testbed result



NS result

Testbed results:

- Much higher base throughput (4.8 vs. 2.2Mbps)
- Consistent gains with ACK skipping, although moderate.

Observations

- Testbed results confirm the TCP simultaneous-send problem
- Skipping 1 ACK consistently improves TCP throughput although for different reasons in different cases.
- Testbed results differ from NS results with respect to base throughputs and gains
- In the case with MAC retries: TCP slow start does not cause MAC queue overflows in real systems because of OS intervention.
- Status of variables hard to observe in real-time in testbed experiments; Sniffers used to observe packet flow

Conclusion

Conclusion

- Main insights:
  - Complex interaction of TCP with 802.11 MAC
  - TCP control packets interfere with transmission of data packets over 802.11 WLANs causing overall throughput degradation
  - Simple TCP adaptation of skipping alternate ACKs achieves significant gains
- NS simulations required to evaluate protocol correctness and observe status parameters in real-time
- Real-life evaluation of transport protocols essential to understand operation along with other layer protocols.
- ORBIT enables repeatability of wireless experiments
- Better instrumentation of the network stack required to observe real-time status.

Done! ☺

Questions… ?