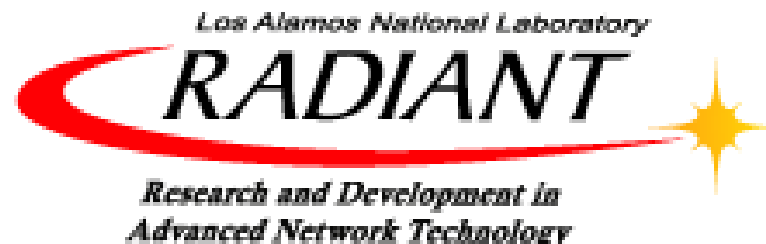


# Dynamic Right-Sizing: A Simulation Study

---

**Eric Weigle and Wu-Chun Feng**

{ehw,feng}@lanl.gov



<http://www.lanl.gov/radiant>

# Presentation Overview

---

- Introduction
- TCP
  - Background
  - Problems
- DRS
  - The Solution
  - Algorithm
  - Configuration
- Simulations
- Results
  - All static flows
  - One static/dynamic
  - Mostly static
  - Equal static/dynamic
  - Mostly dynamic
- Conclusion
  - (Implementation)

# Introduction

---

- Goal: Well performing TCP over 6 orders of magnitude:
  - 56K Modem = 56,000 =  $5.6 \times 10^4$  bps
  - 1Gbps Enet = 1,000,000,000 =  $1 \times 10^9$  bps
- TCP has many problems limiting performance; in particular:
  - Flow and congestion control mechanisms
- DRS addresses one of them
  - Static flow windows!

# TCP Background

---

- Reliability via local buffering until acknowledgement
- Limits data transmitted per Round Trip Time (RTT)
- Ideal size comparable to the bandwidth $\times$ delay product
  - These range from a few bytes and a few megabytes
    - 56Kbps  $\times$  5ms  $\rightarrow$  36B
      - Wastes 99% memory (36B/64KB = 0.05%)
    - 622.08Mbps  $\times$  100ms  $\rightarrow$  7.8 MB
      - Wastes 99% bandwidth (64KB/7.8MB = 0.80%)
- Maximal default buffer size (without window scaling)
  - 64KB

# Problems With TCP

---

- TCP Reno AIMD and slow start restart
  - Long additive increase period
    - For 8MB bandwidth×delay product, convergence requires four and a half minutes!
      - $8\text{MB} \times \frac{1}{2} \times 1024^2\text{B}/\text{MB} / 1500\text{B}/\text{segment} \times 1\text{segment}/\text{RTT} \times 0.100 \text{ s}/\text{RTT} = 266\text{s}$
- Bursty, chaotic, self similar network traffic
  - Bandwidth and latency vary dramatically
- Flow windows cannot effectively be set by hand
- No loss differentiation (Wireless)
- No OS bypass mechanism

# Dynamic Right-Sizing

---

- Variables:
  - $\{e,f,c\}wnd$ : Sender's
    - effective transmission/flow/congestion window
  - $awnd$ : Receiver's advertised window
  - $maxwnd$ : Send/Recv maximal window (memory)
- So  $ewnd = \min(fwnd, cwnd, awnd, maxwnd)$
- Currently,  $cwnd$  changes while  $fwnd$  is static;
  - Transmission should be limited **only** by congestion!
- DRS: Adapts  $fwnd$ ; solves the problem
  - Is interoperable (no breakage of TCP semantics)
  - Requires no user intervention or administration
  - Is fair

# DRS Algorithm

---

- Bandwidth maximized when  $fwnd \geq cwnd$
- Memory use minimized when  $fwnd = cwnd$
- At the source,
  - $fwnd = \min(cwnd, awnd, maxwnd)$
- At the destination,
  - $awnd \geq 2 \times cwnd$ , If sender in slow start
  - $awnd \geq cwnd + 1$ , If sender in additive increase
- In Simulation:
  - Set  $fwnd = cwnd$
- In Implementation:
  - Set  $|cwnd - fwnd| < \delta$

# DRS Fairness

---

- Outside observer:
  - Cannot distinguish DRS from static with large *fwnd*
  - "If you can't tell, it doesn't matter"
- New fairness measure:
  - Flows should achieve bandwidth proportional to the resources allocated
- For example:
  - If a static flow allocates a 44 packet (64KB) *fwnd*, and a DRS flow allocates a 700 packet (1MB) *fwnd*
  - Then the DRS flow should receive:
    - $(700+44)/44 = 16$  times the performance

# DRS Configuration

---

- Original knobs:
  - Default / maximal buffer size, window scaling factor
  - Set based on **technical** requirements
  - Impossible for a human to effectively set
- New knobs:
  - Default / maximal buffer size, window scaling factor
  - Set based on **administrative** requirements
  - Easy to set; little harm from extreme values
    - Internally, kernel code automatically tunes
  - *Interpretation of parameters has changed*

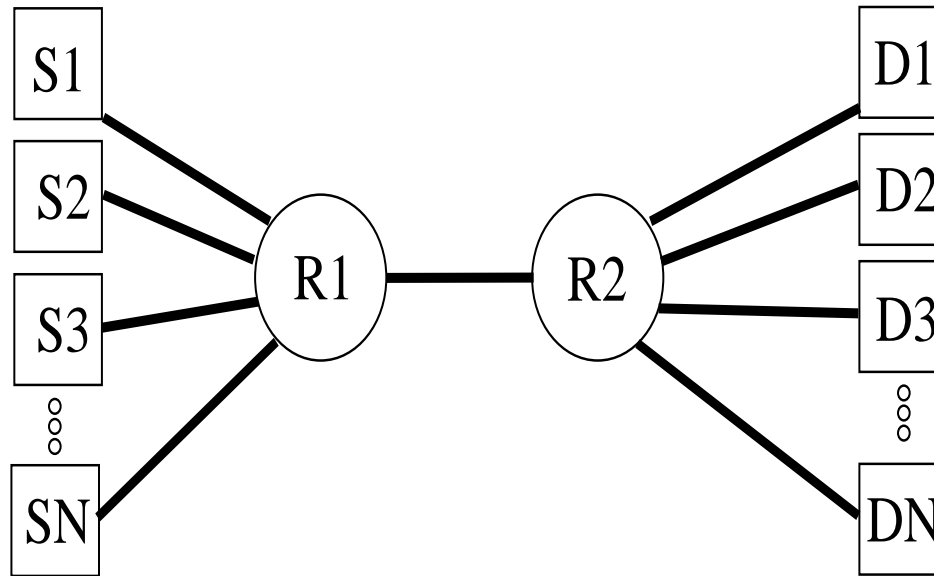
# Simulations

---

- Run TCP connections and see what happens. Vary:
  - Proportion of static flow window and dynamic flow window (DRS) TCP connections in the network.
    - All static; today's Internet.
    - Some static, some dynamic; incremental adoption
    - All dynamic; the future
  - Router queue sizes: 100, 500, and 16,384\* packets
    - approximately 145KB, 730KB, 23.4MB
- Hold constant (variations considered; insignificant)
  - Traffic generators: infinite file (Poisson & Pareto.)
  - Traffic flow: one way (bidirectional.)
  - Router queuing: Drop tail (RED.)
  - Stock TCP buffer: 64KB

# Simulation Topology

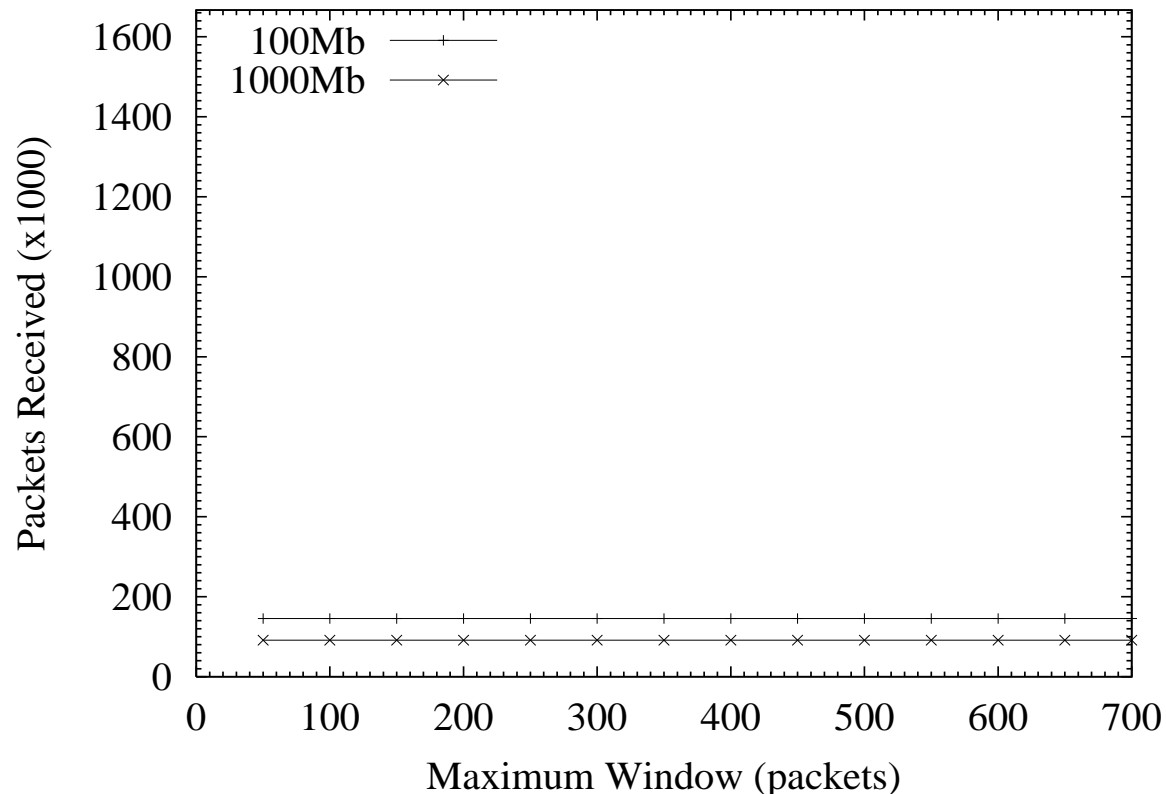
---



- All Links:
  - 100Mbps/10ms delay (60ms RTT) OR
  - 1Gbps/16ms delay (96ms RTT)
- Simulations were run for 200 seconds.
  - On 100Mbps topology, at most 1,666,666 packets (2.3GB) can be sent

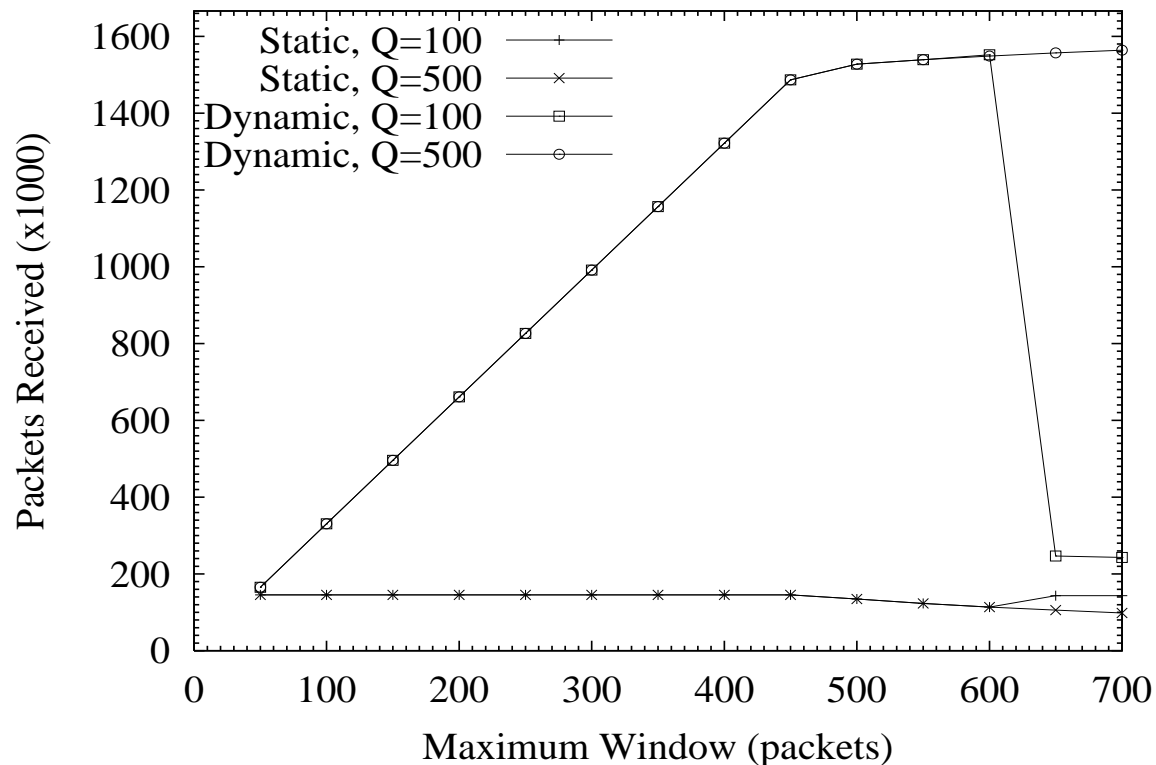
# Results: Static Flows

---



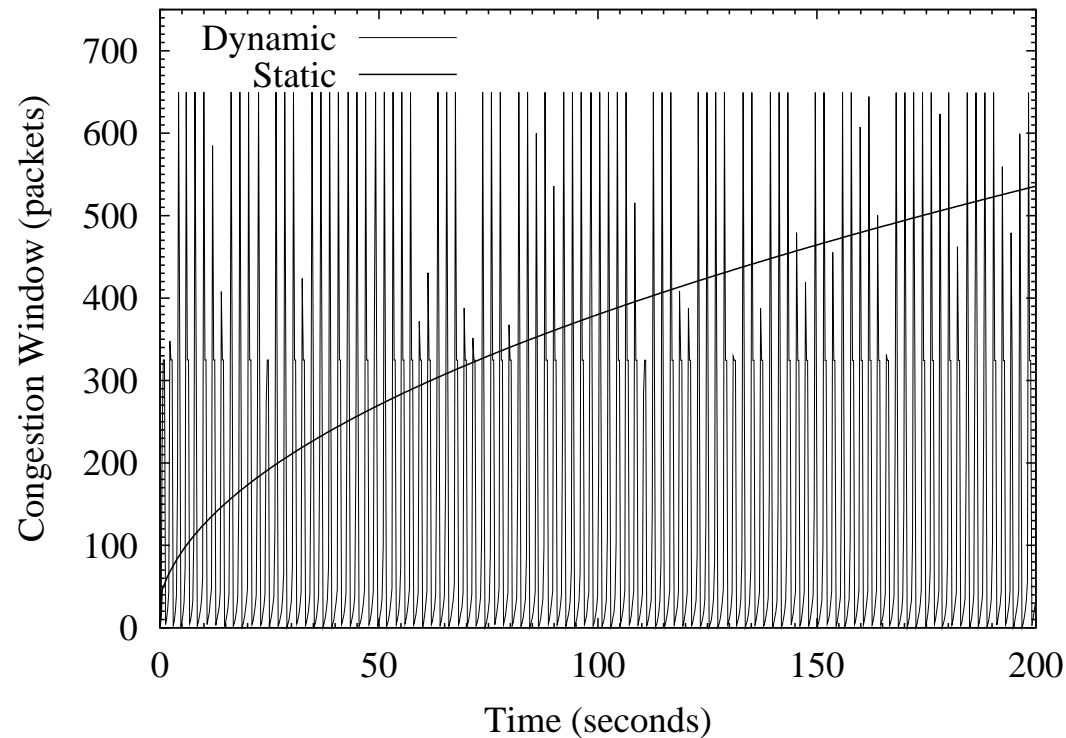
- Static flow windows inadequate in a WAN.
  - 100Mb: Throttle to 8.7% of available bandwidth
  - 1Gb: to 0.55% of available bandwidth

# 1 Static vs. 1 Dynamic



- $fwnd$  varies from 73KB to 1MB
- $maxwnd \approx 650$  is significant
- Actual bandwidth $\times$ delay for this network.

# *cwnd* Over Time



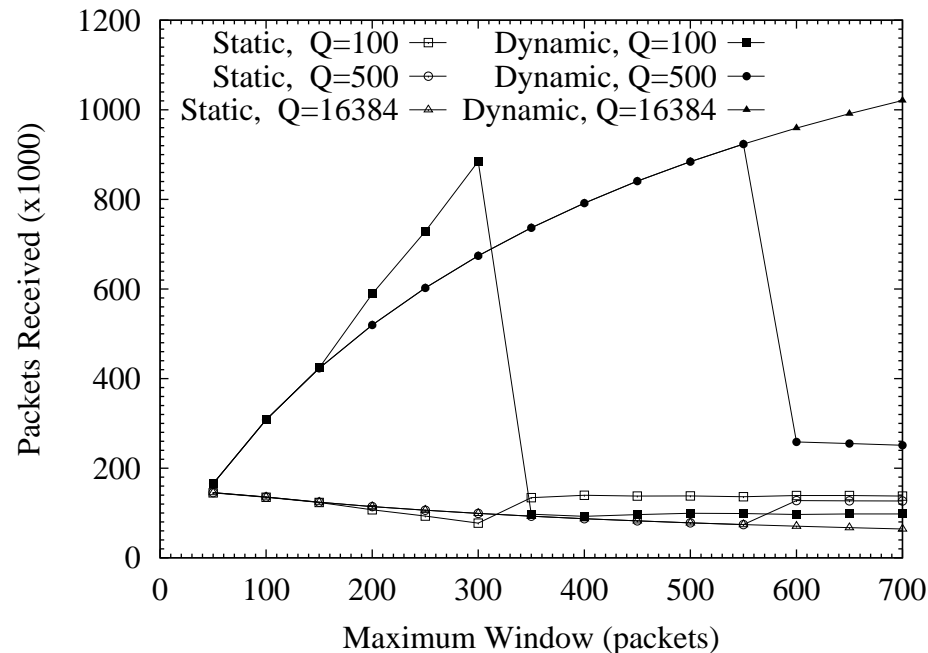
- DRS' variability due to use of slow start restart
  - *cwnd* grows to 650 packets, allowing a large burst; induces loss rates of up to 27%!
- Problem with TCP Reno AIMD, slow start restart
  - Also seen with large static flow windows.

# Possible Solutions

---

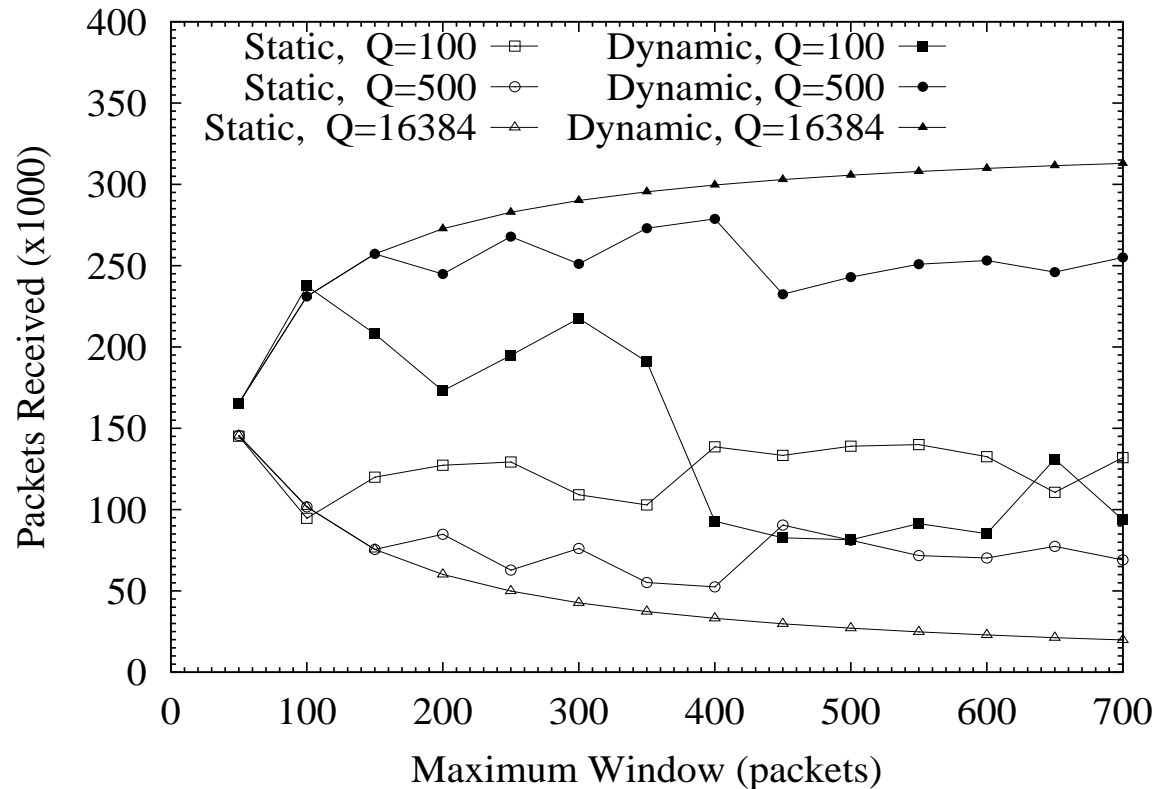
- Increase buffering in routers (least realistic):
  - Here, 500 packets is fine; gives 93.8Mbps
- Use fractional backoff
  - Generalized AIMD: additive increase takes too long
- Explicit window adaption:
  - Use maxwnd heuristic – fwnd as congestion control
- Other possibilities:
  - ECN, TCP Vegas, TCP Pacing, etc.
- In practice (our implementation):
  - Detect incipient congestion via increasing RTTS

# Results: Mostly Static



- Similarities with prior section:
  - Performance drops when queueing exhausted
  - Large queues give utilization over 99%
  - DRS flow outperforms any static flow
- Differences from prior section:
  - One case where static flows better than DRS flow
  - Sublinear performance increase with queue sizes
    - Feedback loop increasing delay with queue length

# Results: Equal Static/DRS



- Erratic behavior for small queue sizes
- Reno Highly sensitive to initial conditions
- Large windows emphasize behavior
- Still fair!

# Fairness: Equal Static/DRS

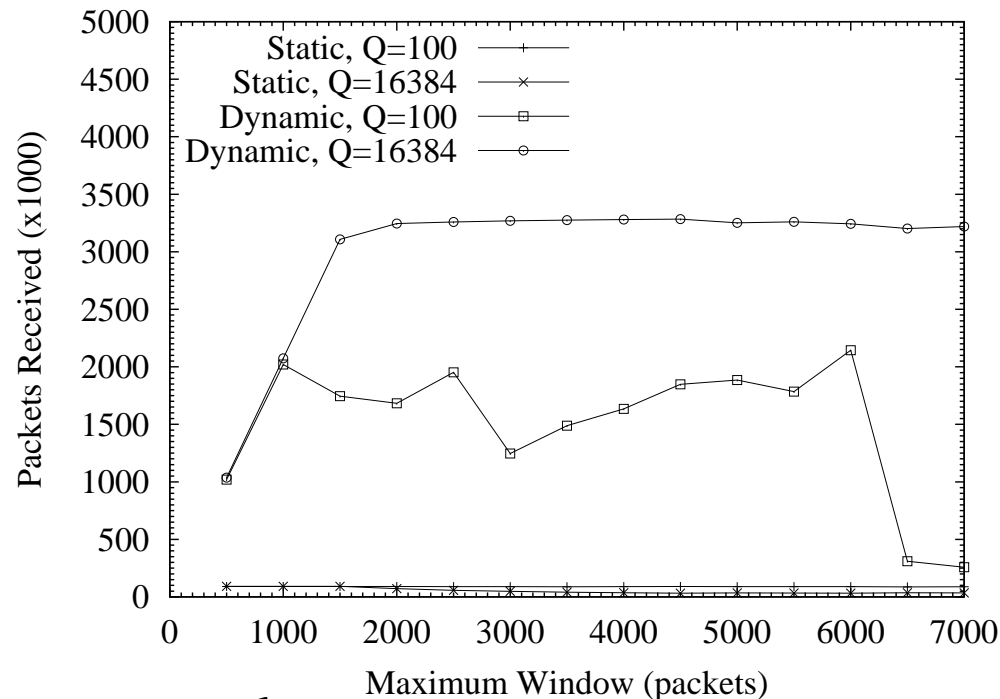
---

- Among flow types:
  - 100Mb network,  $Q=16384$ ,  $\text{maxwnd}=700$ :
    - 5 DRS flows each allocate 700 segments
    - 5 static each allocate 44 segments
    - Same as our earlier example—Results match!
- Among individual flows:
  - 100Mb network,  $Q=100$ ,  $\text{maxwnd}=650$ :
    - Higher variability per flow—*short term*

Flow #	1	2	3	4	5
Static	108	112	113	112	108
Dynamic	389	66	55	71	74

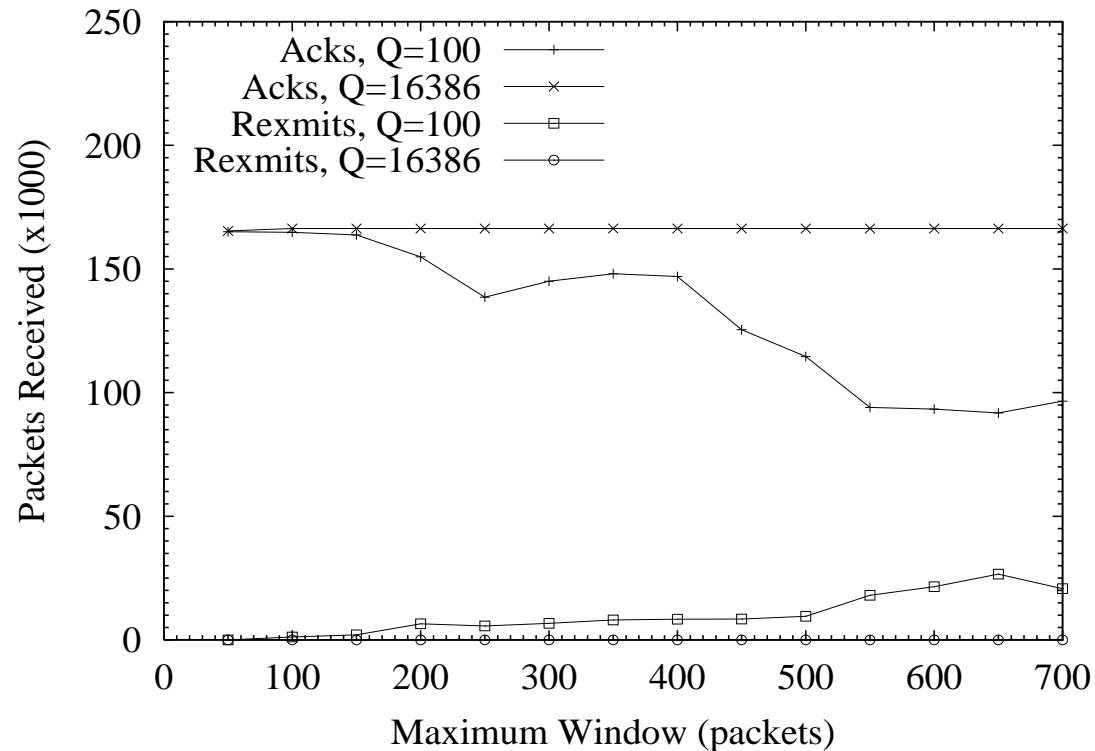
- Some of this is a simulator artifact
  - Deterministic, packet queues, link number breaks ties 18

# Gigabit: Equal Static/DRS



- Small maxwnd:
  - Linear pattern as earlier
- Larger maxwnd:
  - DRS utilizes almost 100 times BW of static
  - Queue of 16384
    - DRS  $\approx$  200Mbps versus static  $\approx$  2Mbps
    - High utilization; over 99%

# Results: Mostly/All DRS



- 10 DRS flows, 100Mbps net, queues of 100 or 16,384
- Shows instance of the large losses
  - Slow start/slow–start restart problem again!
- Horizontal line for larger queue size–
  - Fair bandwidth is 10Mb per flow
  - Requires only 50 to 100 packets unacknowledged

# Conclusion

---

- In simulation, we have shown:
  - DRS provides good performance:
    - Can improve throughput by orders of magnitude
  - DRS behavior is fair
  - 'No knobs':
    - Administrative versus technical configuration
  - TCP has 'issues' to address
    - Slow start restart, AIMD, self similarity
- Similar work has been done at PSC
  - BSD versus Linux
  - Alters TCP semantics (advertised window)
- We have an implementation, verifying these results.

# Implementation- Mike Fisk

---

- Patch for Linux 2.2.19 completed, 2.4.x in progress
  - Uses RTT estimates to avoid 'dropoffs' seen in simulation
  - Interoperates with stock (static) TCP implementations
  - Currently, only increases fwnd size (i.e. Scales up only.)
  - Will be publicly available (GPL, after SC2001)
- Results: over gigabit Ethernet with 100ms delay
  - Factor of 7 improvement in throughput!
- Shameless Plugs for SC 2001; Come see our Demos—
  - DRS in Linux 2.2.19 (perhaps 2.4.x?)
  - DRS in userspace (FTP)
  - Other work (TICKET, MAGNeT...)

**→<http://www.lanl.gov/radiant>**