Ns Tutorial: Case Studies

John Heidemann (USC/ISI)
Polly Huang (ETH Zurich)

March 14, 2002

Road Map

- Simple examples
  - TCP
  - web traffic

Provide an entry point

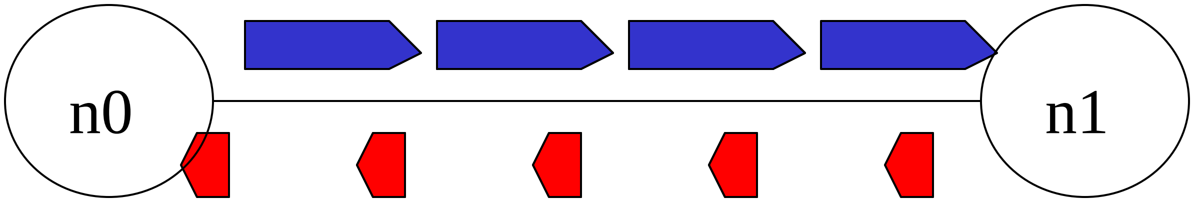
- Case Study
  - Impact of HTTP and TCP parameters to Web performance
  - Hidden structure behind aggregated Web traffic

Show case ns’s functionality and relevance to the CN program

Presentation Style

- Slides
- Script walk-through
- Live demos with nam (Network AniMator)

Example I: TCP



```
set ns [new Simulator]
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1.5Mb 10ms DropTail
set tcp [new Agent/TCP]
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n1 $tcpsink
$ns connect $tcp $tcpsink
```

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 0.2 "$ftp start"
$ns at 1.2 ”exit"
$ns run
```

Example II: Web Traffic

- A Web session – a series of page downloads
  - Number of pages
  - Inter-page time
  - Page size (number of embedded objects)
  - Inter-object time
  - Object size (KB)
- 5 random variables

Case Study I: Web Performance

- Impact of TCP and HTTP parameters
- Try to answer:
  - Will the proposed changes work in a variety of conditions?
  - Should TCP Sack be deployed?
  - Should persistency or pipelining be deployed?
  - Which parameters are more cost effective to tune?

Methodology

- Methodology
  - Select performance critical parameters
  - Use most commonly used values as the base case
  - Tune parameter values to compare to the base case
- Toward a systematic and exhaustive evaluation
- Enabled by ns
  - rich library of workload and protocol implementations
  - Contributed code from a huge user/developer community

Parameters and Values

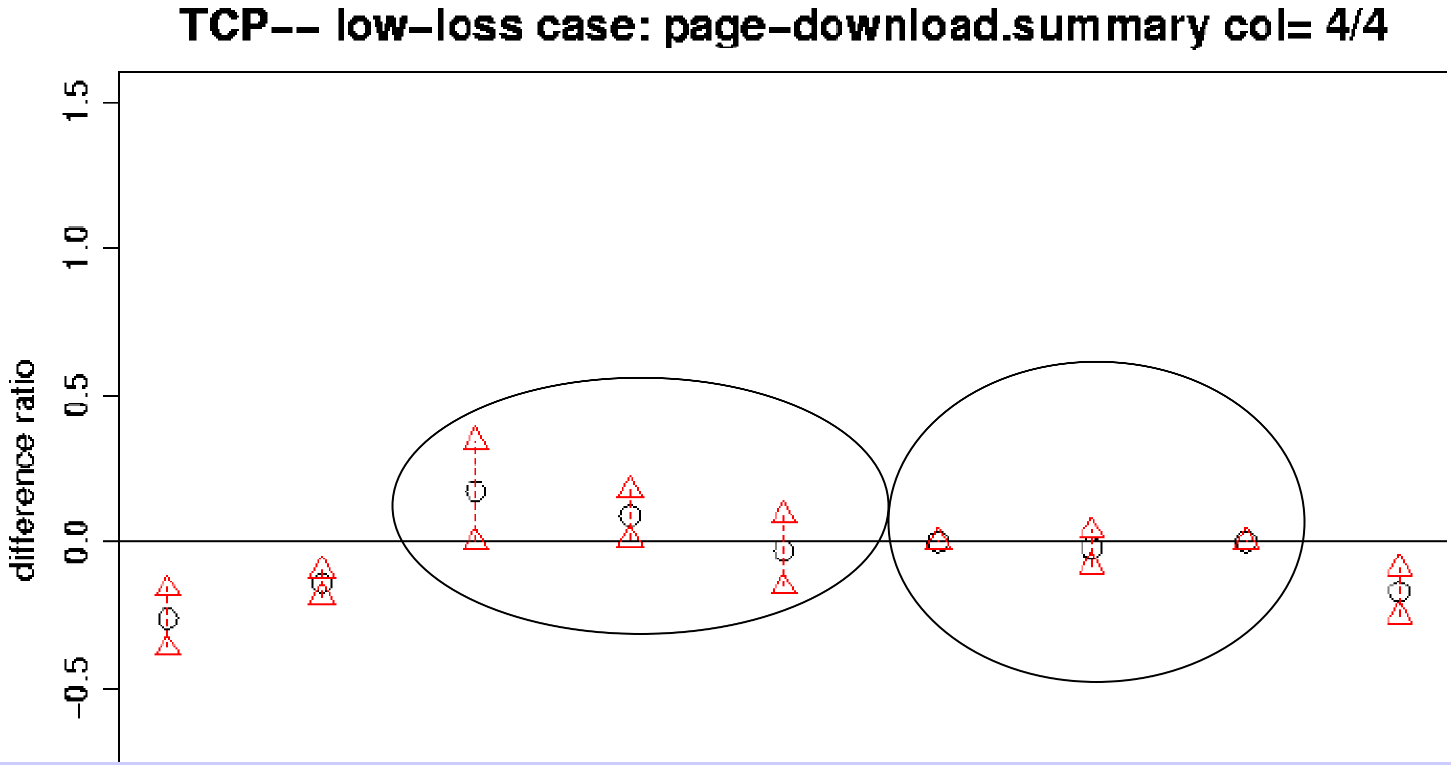
TCP

- Packet size
  - 576, 1460
- Delayed ack
  - on, off
- Congestion avoidance
  - NewReno, Tahoe, Reno, Sack
- Initial retransmission timeout
  - 3, 6 sec
- Timer granularity
  - 100, 500 msec
- Timestamp option
  - on, off
- Initial window size
  - 2, 4

HTTP

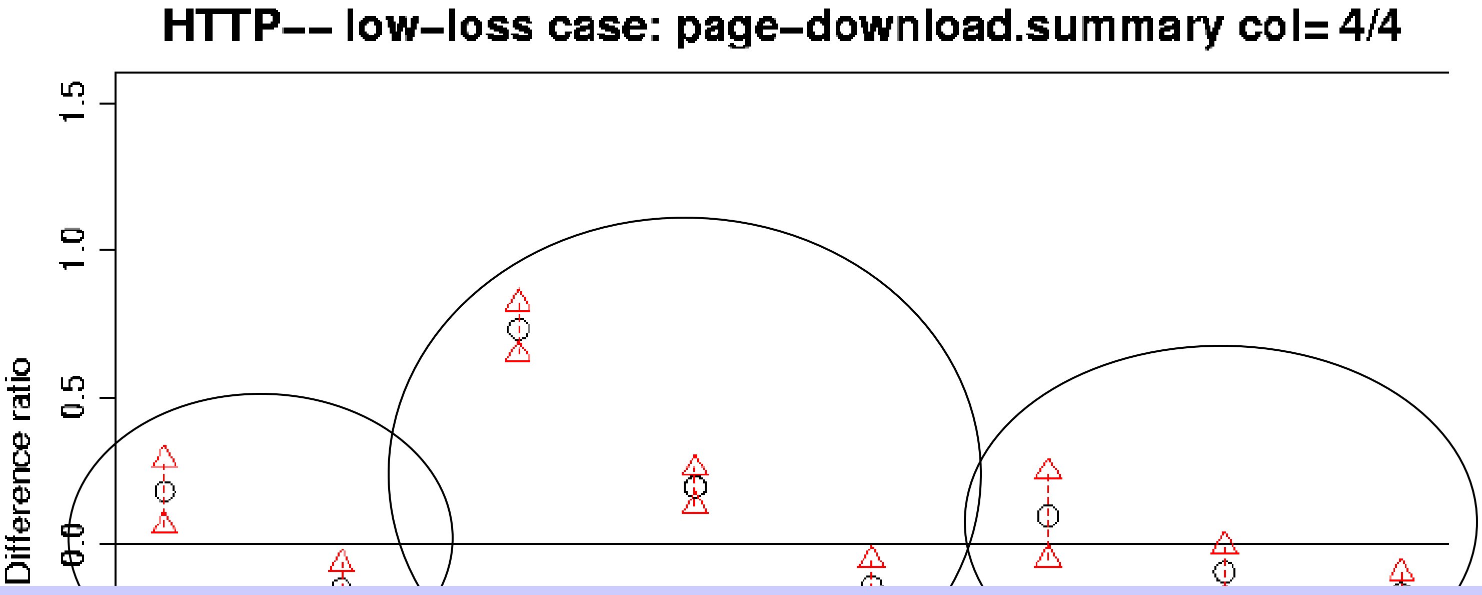
- Connection type
  - persistent, simple, pipelined
- Number of parallel connections
  - 2, 1, 4

Page Download Time – TCP



Sack, NewReno, Reno, Tahoe, gradually better
Timer-related parameters, no significant impact

Page Download Time - HTTP

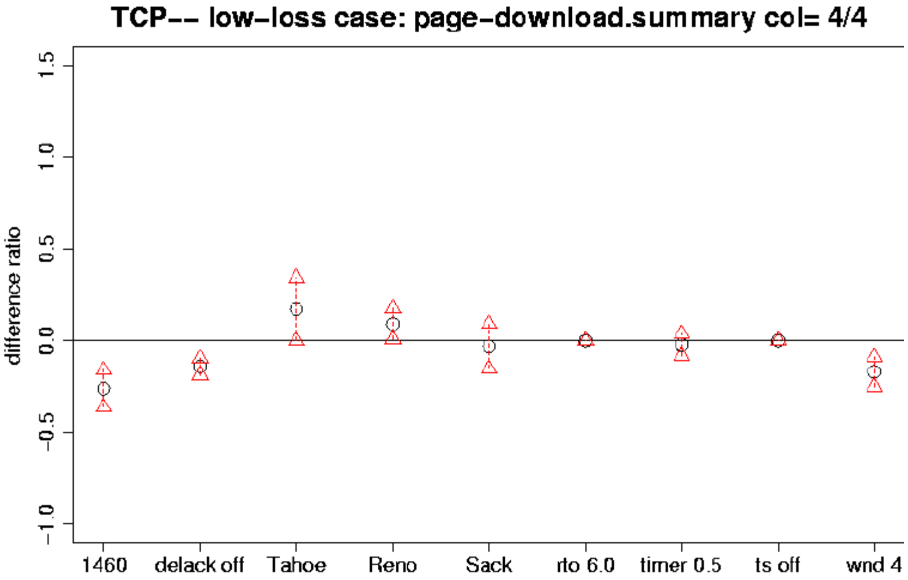


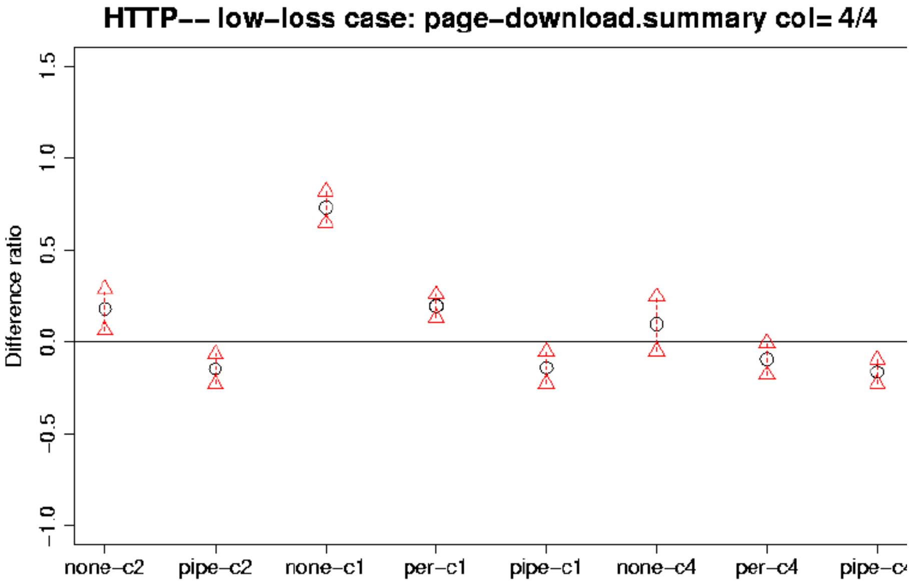
Simple, persistent, and pipelined connections, gradually better
Higher the number of parallel connections,
Smaller the range of improvement

TCP vs. HTTP

TCP

HTTP



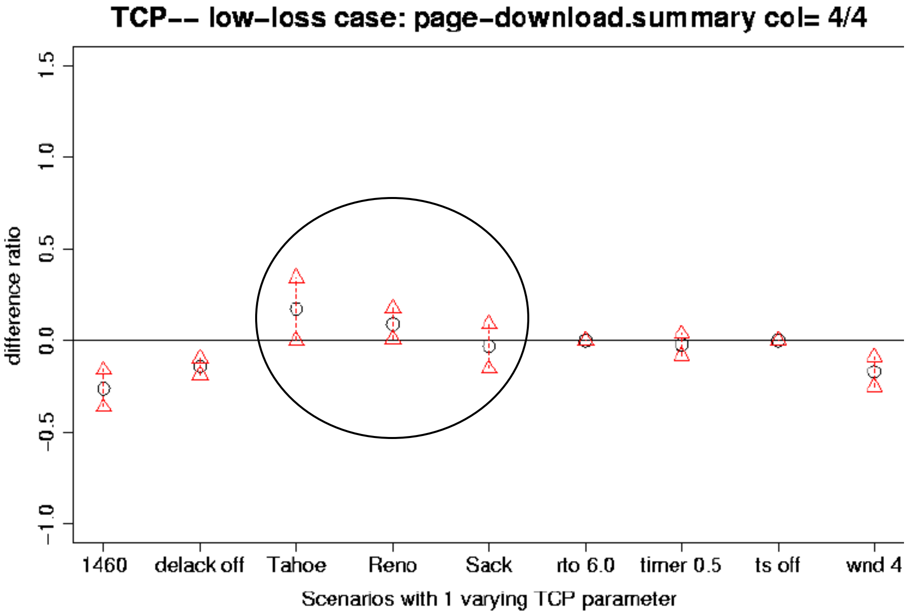


Impact of TCP Sack is relatively small.

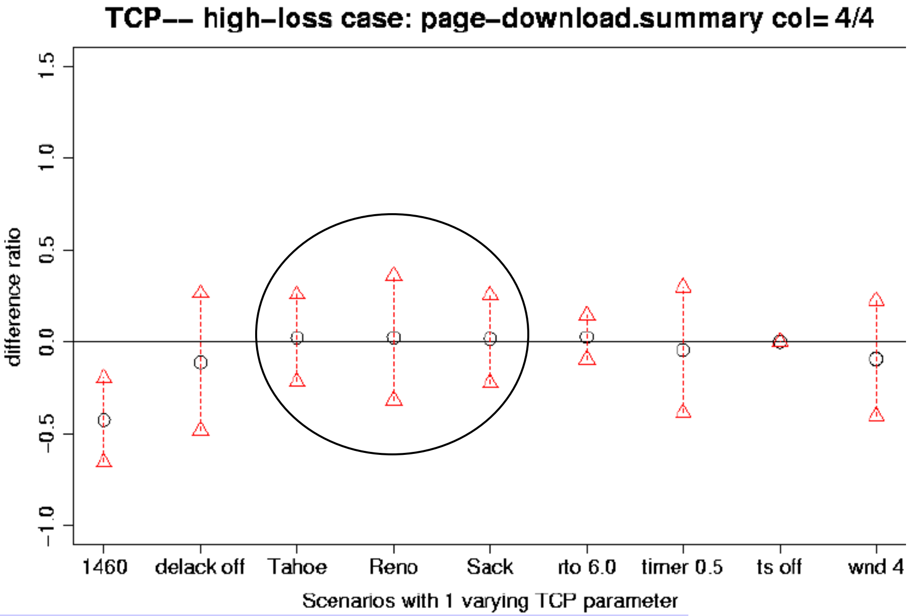
Low vs. High loss - TCP

Low loss

High loss



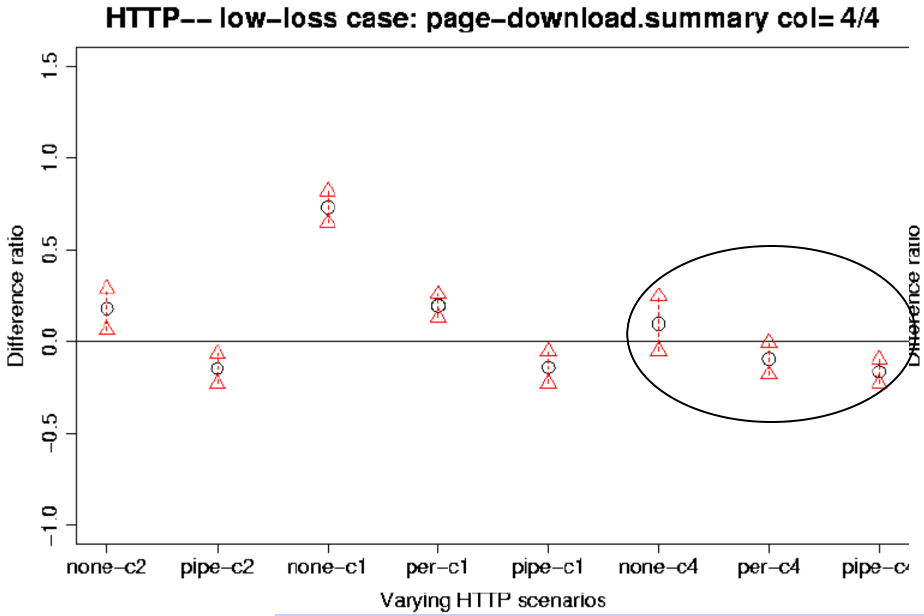
TCP-- low-loss case: page-download.summary col= 4/4

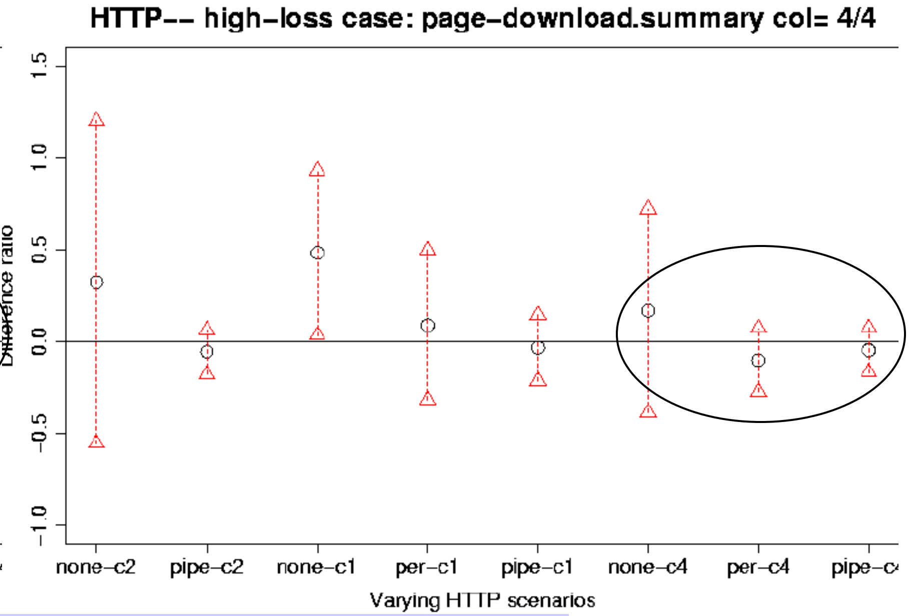


TCP-- high-loss case: page-download.summary col= 4/4

That tiny bit of advantage in TCP Sack disappears in high-loss case.

Low vs. High loss - HTTP


HTTP–– low–loss case: page–download.summary col= 4/4


HTTP–– high–loss case: page–download.summary col= 4/4

Pipelining loses its advantage when # of parallel connections is high.

Preliminary Findings

- Will the proposed changes work in a variety of conditions?
  - Not really
  - TCP Sack and HTTP pipelining
- Should TCP Sack be deployed?
  - Maybe not, if deployment cost is high
- Should persistency or pipelining be deployed?
  - Maybe yes, but doesn’t make sense to work with too many parallel connections

The Real Message

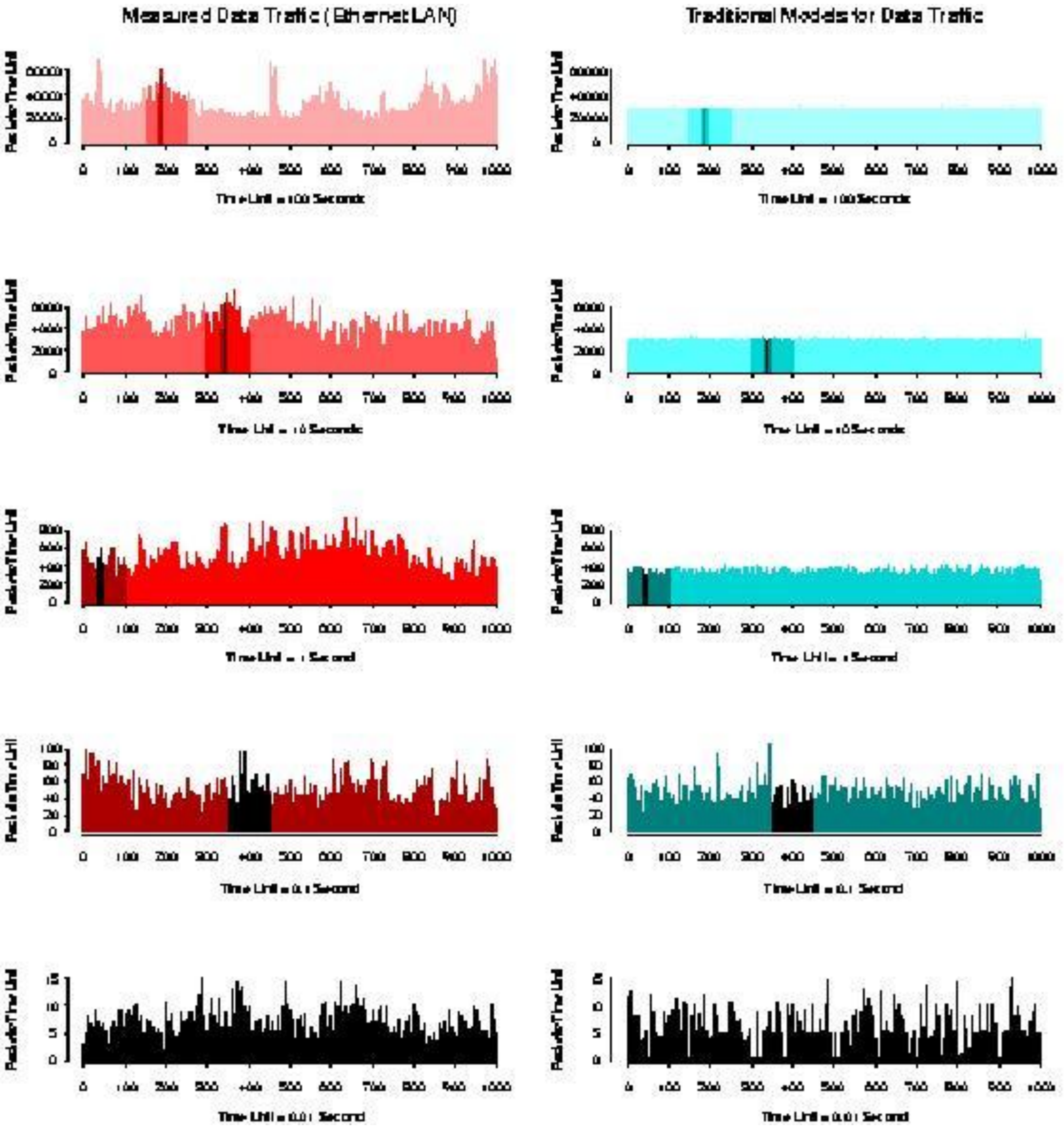
- Design decisions need to be validated in the context of the Internet.
- Layers of protocols, tremendous amount of unknown dynamics
- Simulation tools like ns can help us track the complexity (within a layer or across layers)
- Ns en-powers such studies
  - A rich library base
  - A large community contributing to the base

Case Study II: Web Traffic

- Web traffic is not exact self-similar
- How does it diverge from exact self-similar?
- Why is there this divergence?

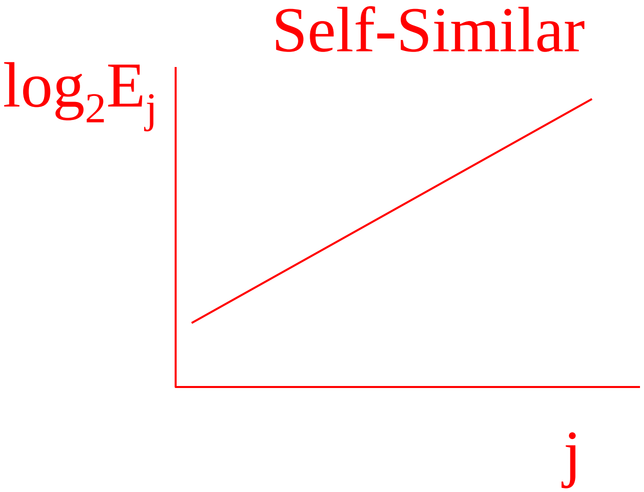
Self-similarity

- Distributions of #packets/time unit look alike in different time scale

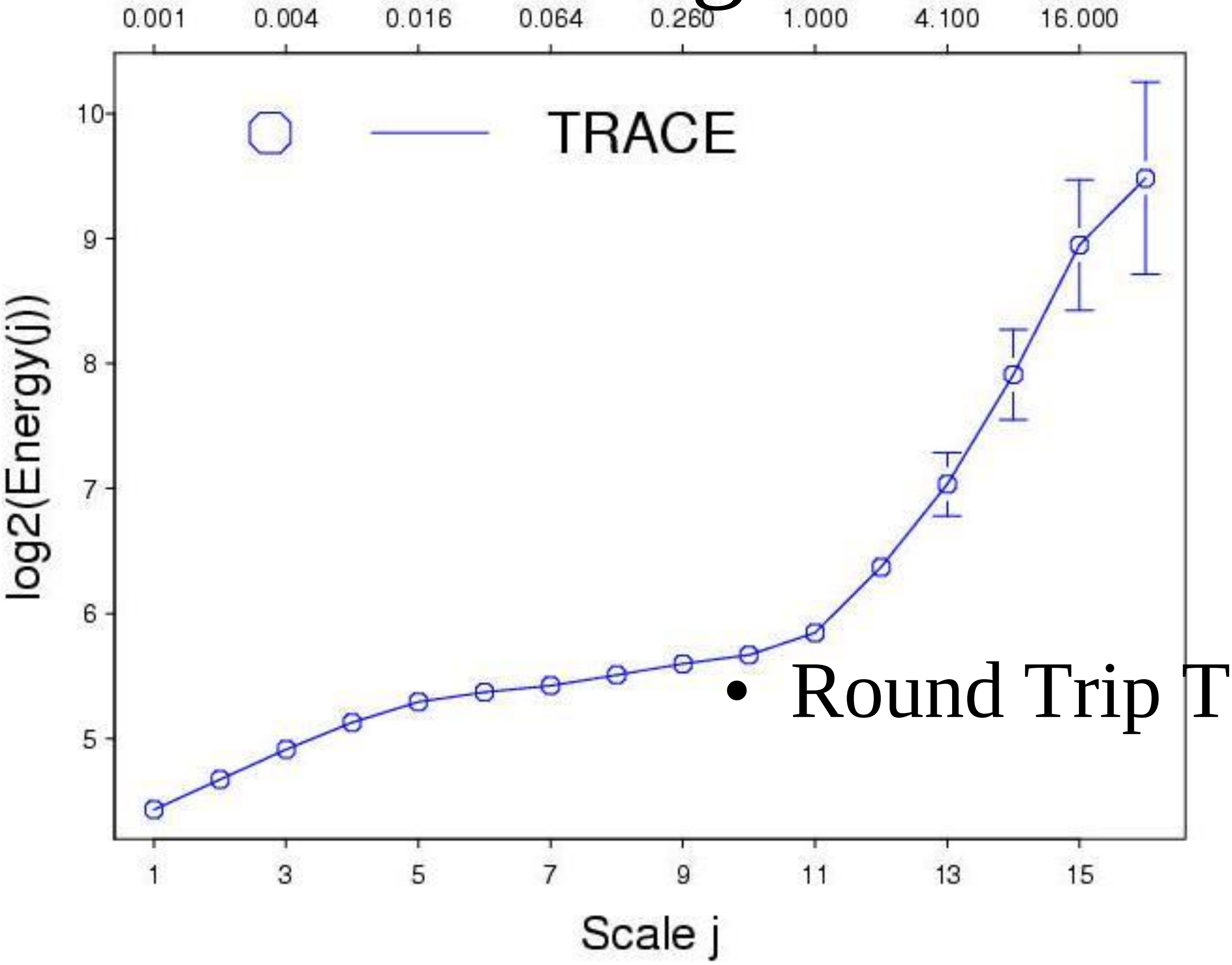

Measured Data Traffic (Ethernet LAN)
Traditional Models for Data Traffic
Packets/Time Unit
60000
40000
20000
0
6000
4000
2000
800
600
400
200
100
80
60
40
20
15
10
5
0 100 200 300 400 500 600 700 800 900 1000
Time Unit = 100 Seconds
Time Unit = 10 Seconds
Time Unit = 1 Second
Time Unit = 0.1 Second
Time Unit = 0.01 Second

Wavelet Analysis

- FFT - frequency decomposition d_j
- WT - frequency and time decomposition $d_{j,k}$
- $\sum_k (d_{j,k}^2) / N_j \equiv E_j = 2^{j(2H-1)} C$
- $\log_2 E_j = (2H-1)\, j + \log_2 C$

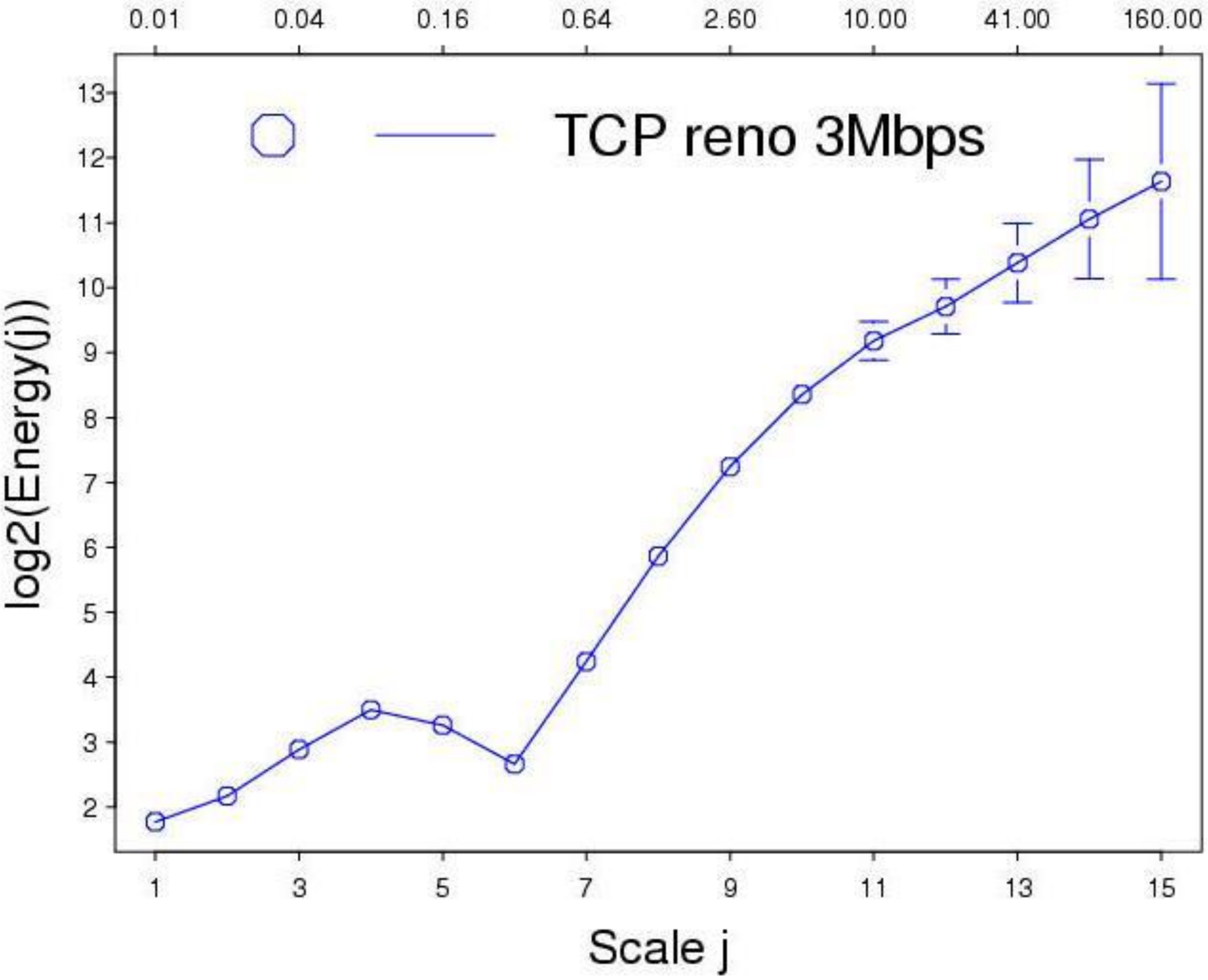


Global Scaling - Trace

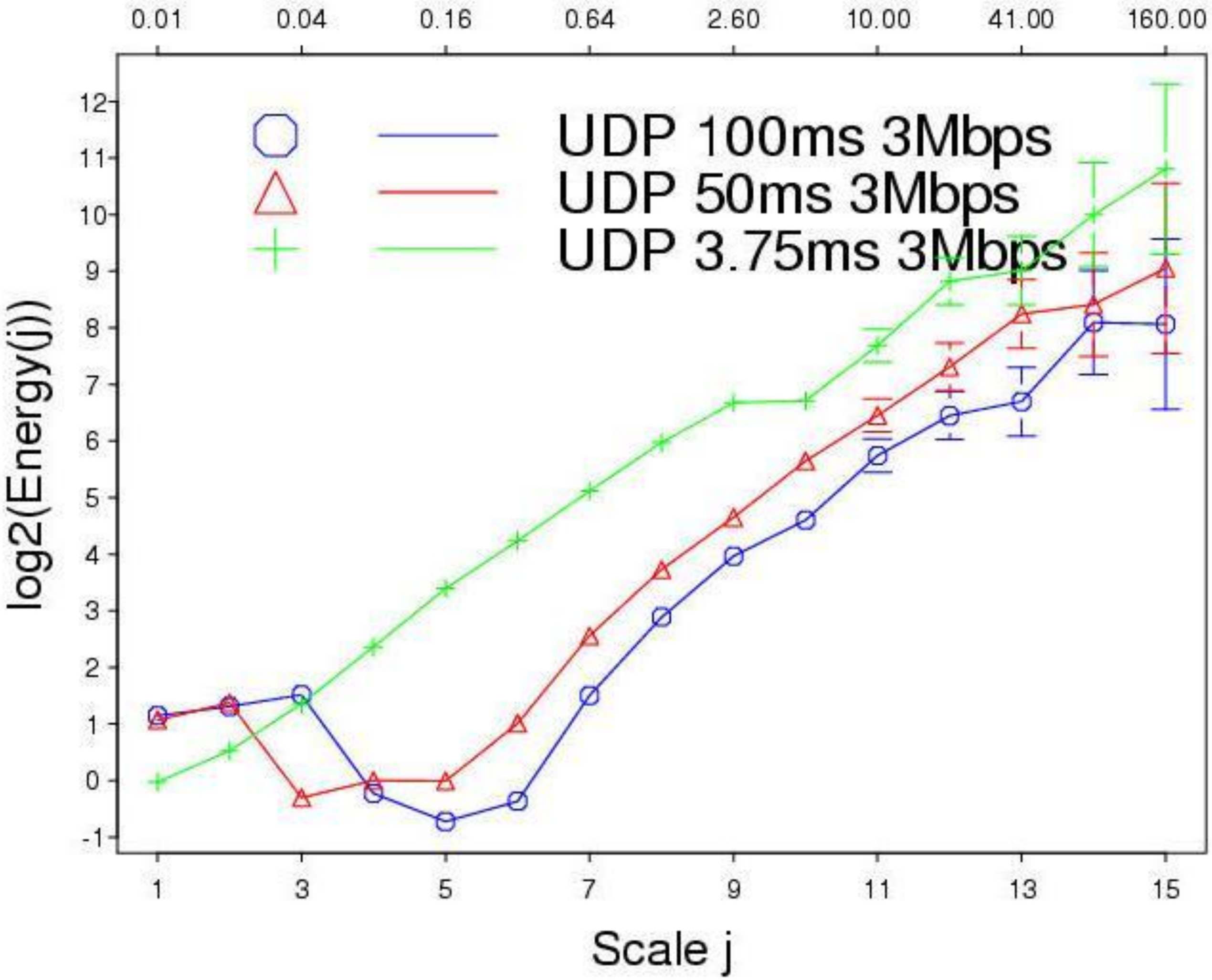


- Round Trip Time

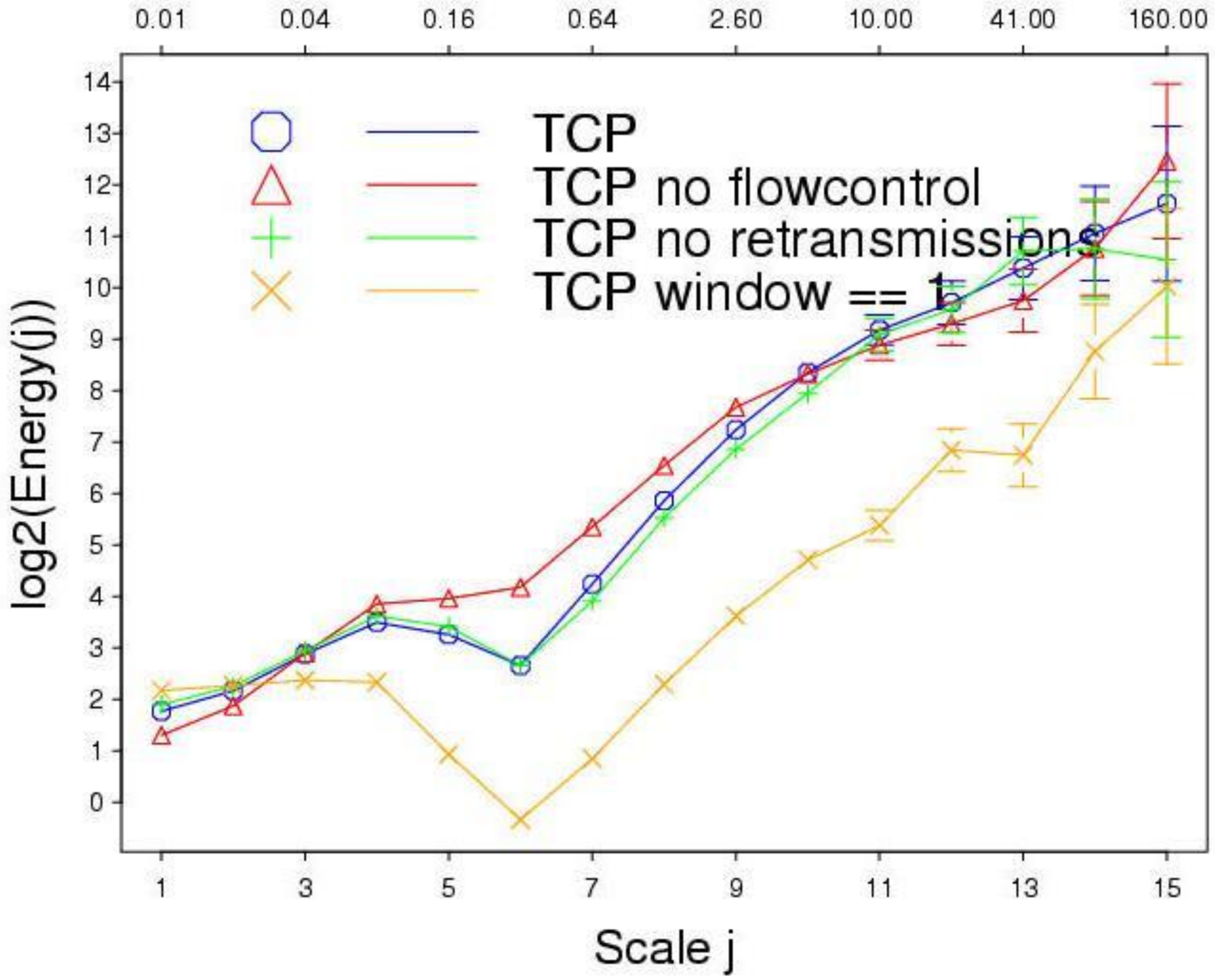
Global Scaling - Simulation



UDP


0.01
0.04
0.16
0.64
2.60
10.00
41.00
160.00
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
log2(Energy(j))
UDP 100ms 3Mbps
UDP 50ms 3Mbps
UDP 3.75ms 3Mbps
1
3
5
7
9
11
13
15
Scale j

TCP


0.01
0.04
0.16
0.64
2.60
10.00
41.00
160.00
log2(Energy(j))
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
○ TCP
△ TCP no flowcontrol
+ TCP no retransmissions
× TCP window == 1
1
3
5
7
9
11
13
15
Scale j

Findings

- Periodicity emerges at round-trip time scales
- That periodicity dominates the traffic behavior at those scales
- TCP ack clocking plays a critical role
- Need to be cautious when to use or not use mathematical self-similar models

The Real Message

- Proposed (traffic) models need to be validated in the context of the Internet.
- Mechanisms can influence Internet characteristics in a surprising way
- Simulation tools like ns can help us track the implicit complexity
- Ns en-powers such studies
  - A rich library base
  - A large community contributing to the base

Concluding remarks

- Learning ns
  - video recording (huang@tik.ee.ethz.ch)
  - on-line tutorials (audio and slides)
  - tons of info from the ns web site
- Research with ns
  - promote sharing and confidence