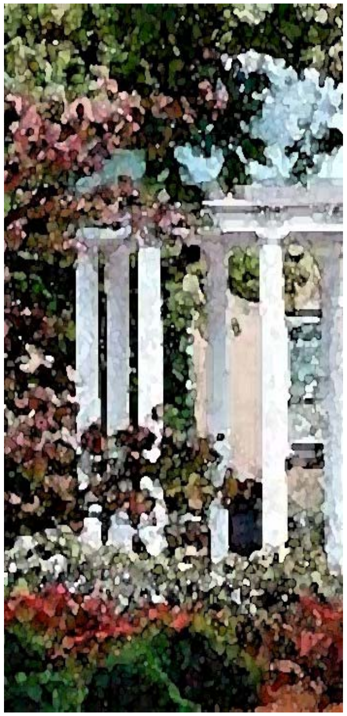


Generating Realistic TCP Workloads

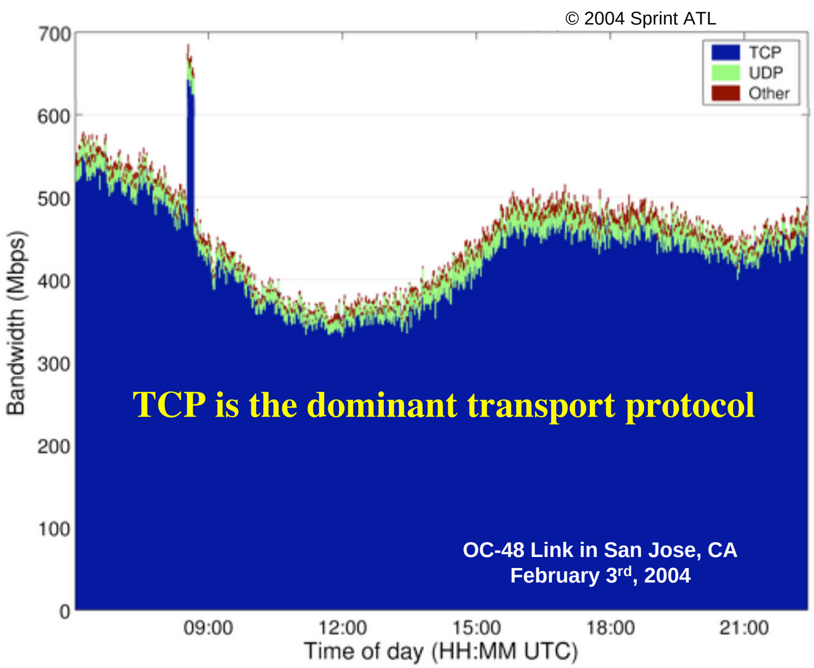
Felix Hernandez-Campos
Ph. D. Candidate
Dept. of Computer Science
Univ. of North Carolina at Chapel Hill

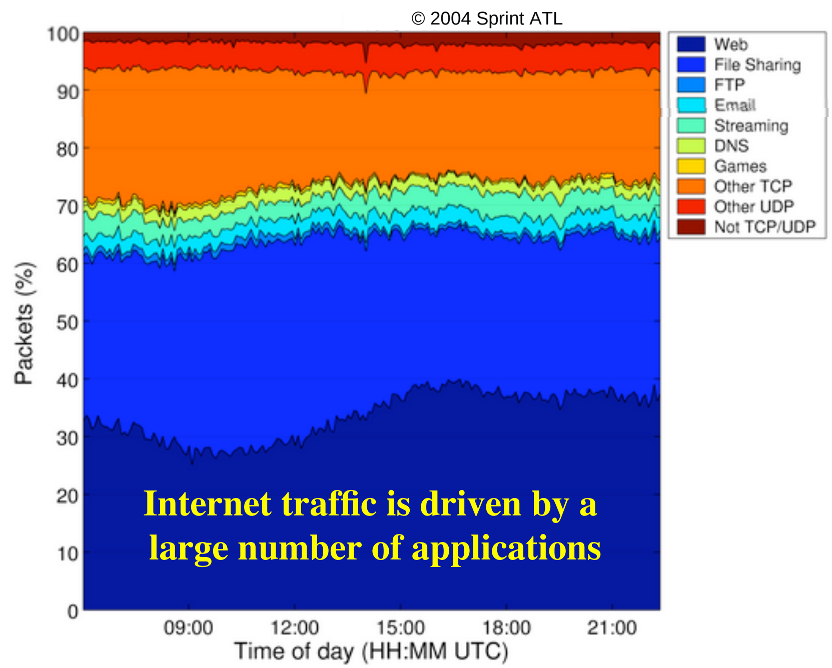
Recipient of the 2001 CMG Fellowship

Joint work with
F. Donelson Smith and Kevin Jeffay

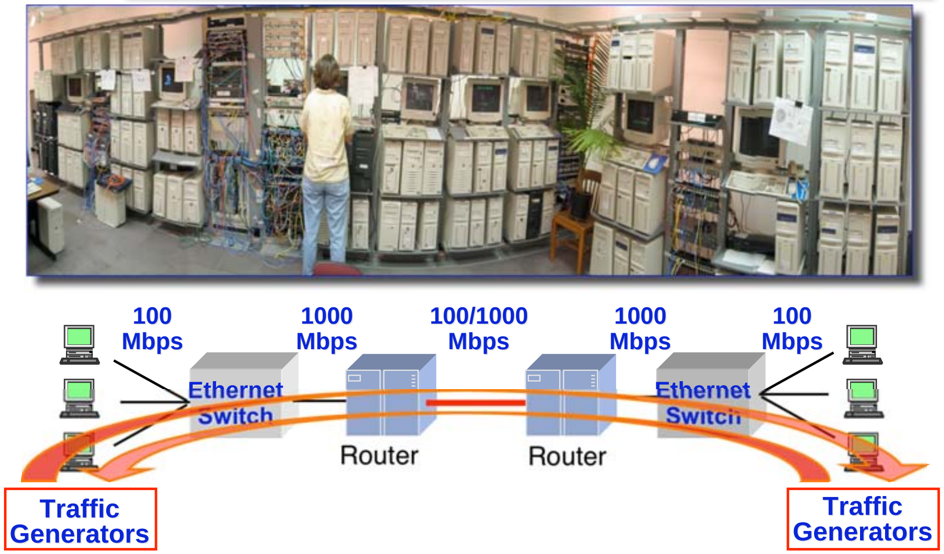
Experimental Networking Research and Performance Evaluation

- Evaluating network protocols and mechanisms requires careful experimentation
  - Network simulation (NS, Opnet, *etc*.)
  - Network testbeds
- A critical element of these experiments is the **traffic workload**
  - Are current workloads *realistic*?
- Let's look at some measurements
  - Sprint's tier-1 backbone network



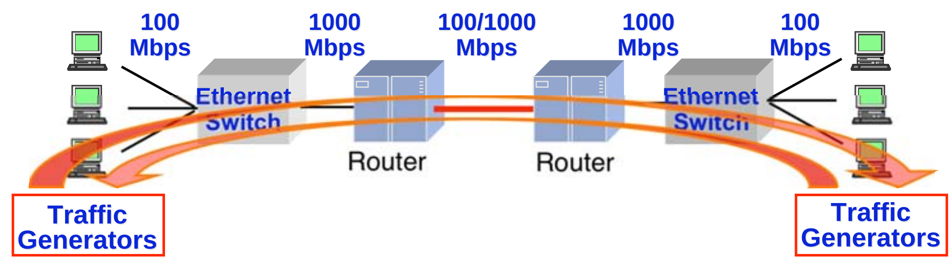


Internet Traffic Generation
Testbed example



Internet Traffic Generation
Testbed example

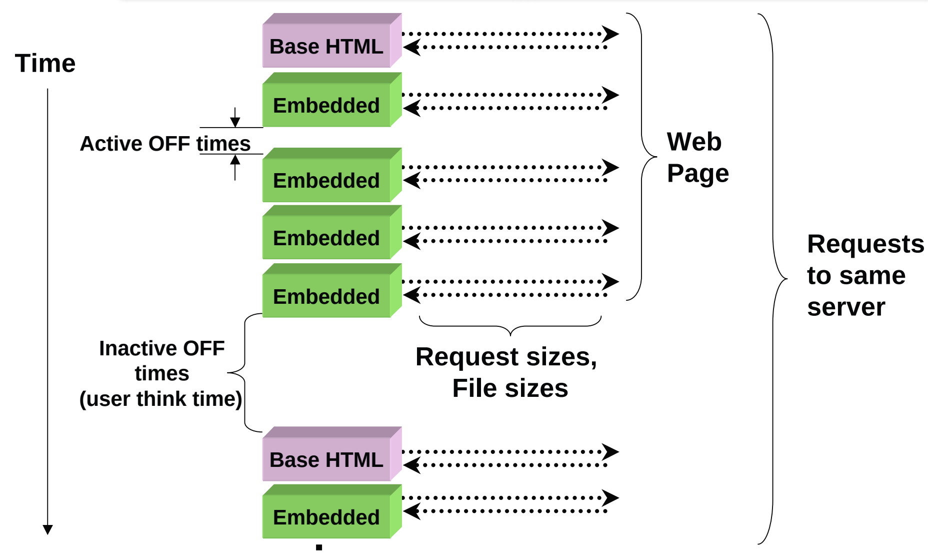
- Evaluate queuing mechanisms in the routers
- Evaluate transport protocols
- Evaluate intrusion/anomaly detection mechanisms
- Evaluate traffic monitoring techniques
- …



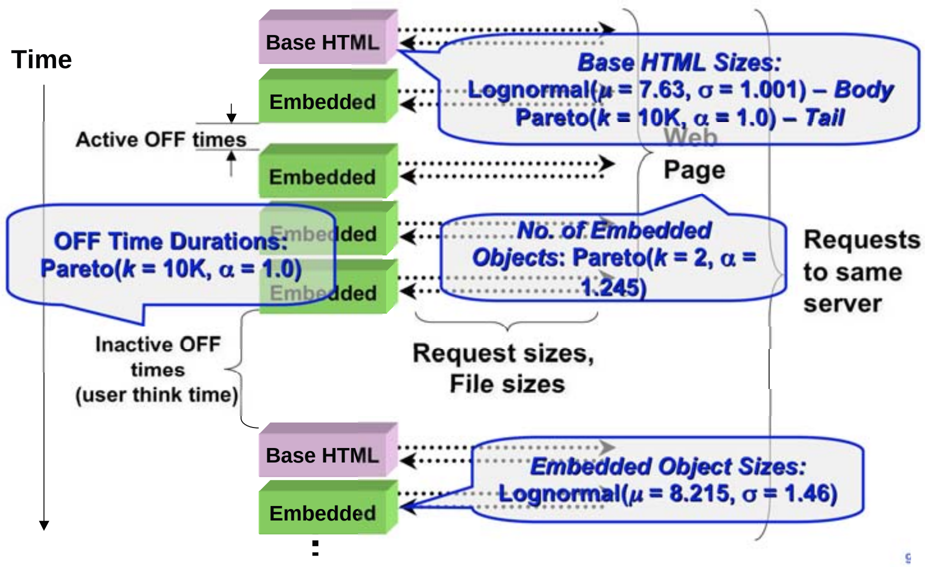
Traffic Generation
State-of-the-Art

- **Open-loop**
  - Large number of sophisticated models
    - » ***Packet-level modeling***
  - But TCP is a closed-loop protocol
    - » Open-loop traffic generation breaks reliability, flow control, and congestion control
- **Closed-loop**
  - The idea is to simulate the behavior of users/applications
    - » ***Source-level modeling of specific application***

Application-Specific Modeling
HTTP Model Example



SURGE HTTP Model
[Barford and Crovella, 1998]



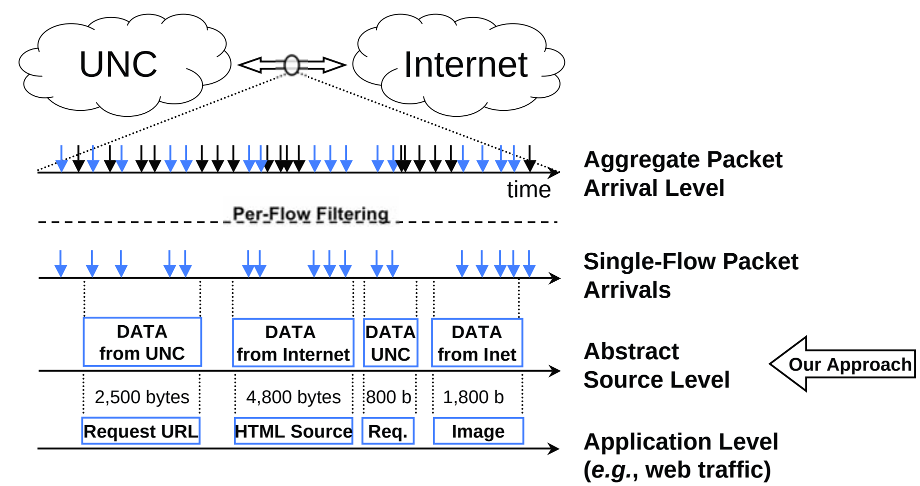
Application-Specific Models
Shortcomings

- Creating application models is a challenging, time-consuming task
  - Traffic mixes are driven by a large number of applications
- Set of dominant applications evolves quickly
  - Applications themselves also evolve
  - We cannot even identify a significant fraction of the traffic
- Modeling closed application protocols requires reverse-engineering
- Privacy considerations complicate data acquisition
  - *TCP header traces* are ok, *application data* is not

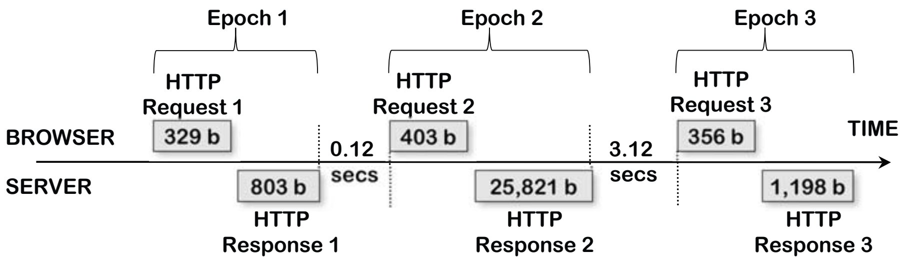
Our Approach

- **Abstract source-level modeling**
  - Application-neutral technique to describe the source-level behavior of any TCP connection
  - Efficient analysis applicable any arbitrary trace of TCP headers (no analysis of payloads)
  - We can also measure network-level parameters, such as round-trip times, receiver window sizes, *etc*.
- **Source-level trace replay**
  - Replaying abstract source-level behavior
- **Validation in network testbed**
  - We wrote a distributed, scalable traffic generator (tmix)
  - Comparison of original and synthetic traces

Different Views of Internet Traffic
Abstract Source-level Modeling



Client-Server Applications
Persistent HTTP Example

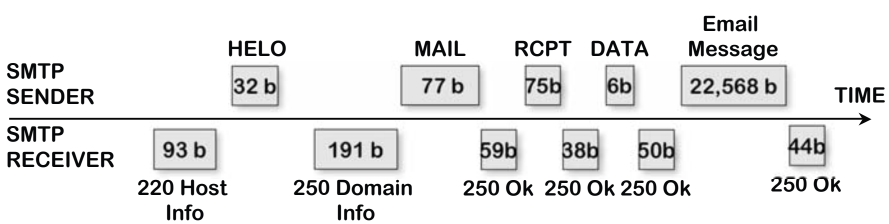


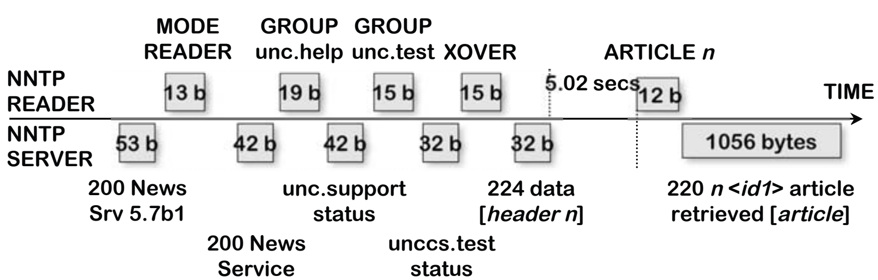
- We call a pair of application data units (ADUs) that carry a request/response exchange an *epoch*
- *Quiet times* are also part of the workload of TCP

Sequential A-b-t Model

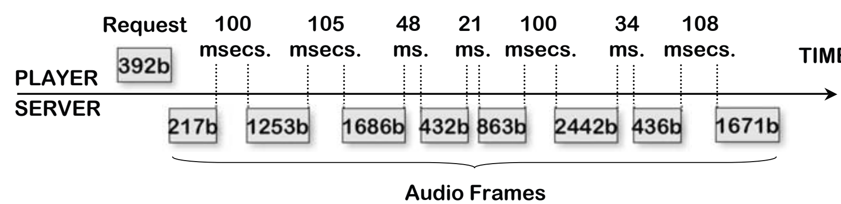
- Abstract source-level model for describing the workload of TCP connections
- Each connection is summarized using a ***connection vector*** of the form $C_i = (e_1, e_2, \ldots, e_n)$ with $n \geq 1$ epochs
  - Each epoch has the form $e_j = (a_j, ta_j, b_j, tb_j)$
- Connection vectors can be extracted from trace of TCP segment headers
  - Sequence number directionality, timing analysis, write size and packet size interactions
  - $O(n \log n) + O(n*W)$

Client-Server Applications
SMTP and NNTP Examples





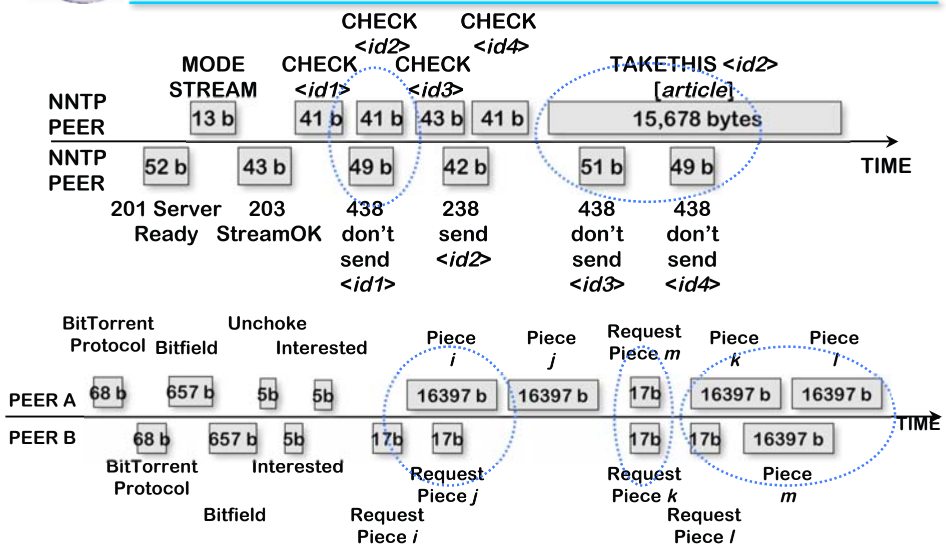
Beyond the Client-Server Model
Icecast – Internet Radio



- Server PUSH applications do not follow the traditional client-server model
- The sequential a-b-t model is still applicable
  - Make a_i and tb_i zero

Beyond the Client-Server Model

NNTP in Stream-Mode



Concurrent A-b-t Model

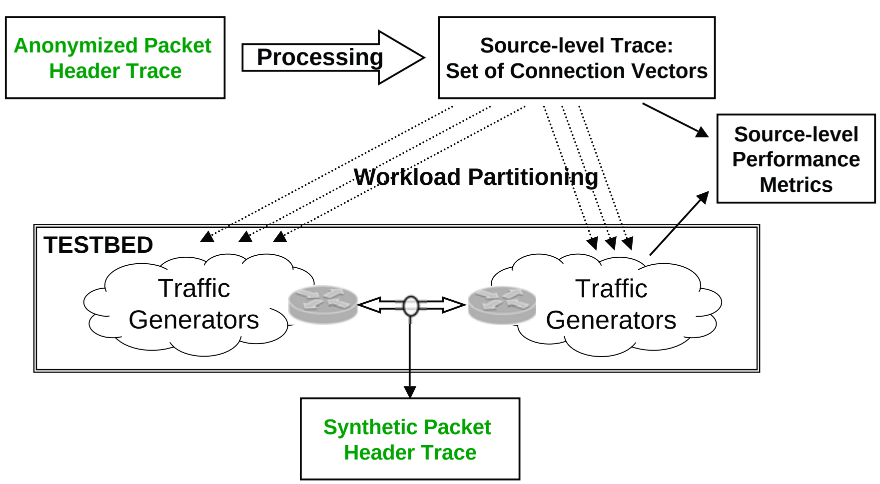
- Some connections are said to exhibit ***data exchange concurrency***
- Two reasons:
  - Increasing performance
  - Enabling natural concurrency
- Concurrent a-b-t model describes each side of the connection separately

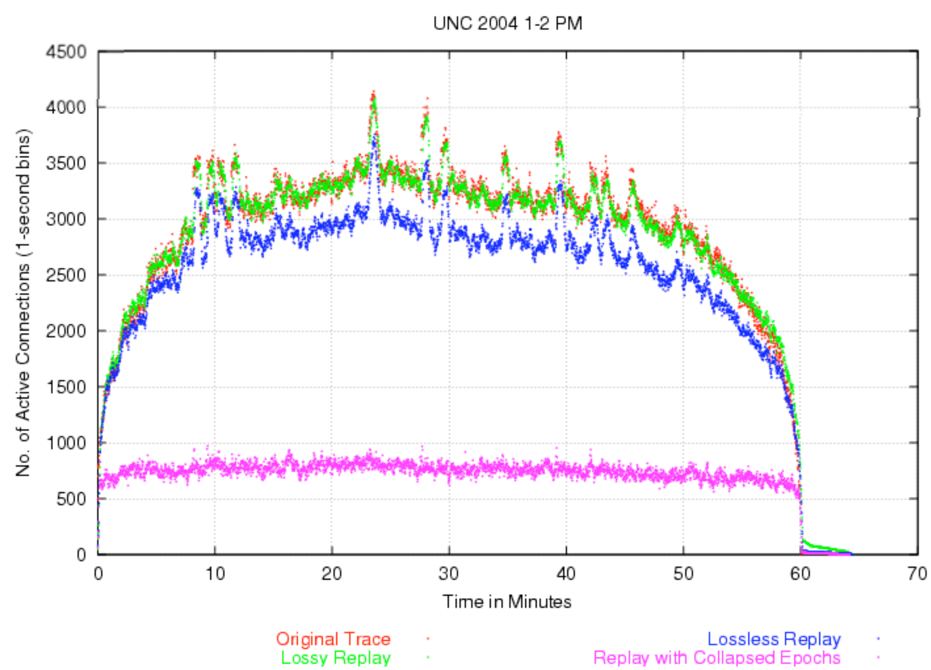
  $((a_1, ta_1), (a_2, ta_2), \ldots, (a_n, ta_n))$

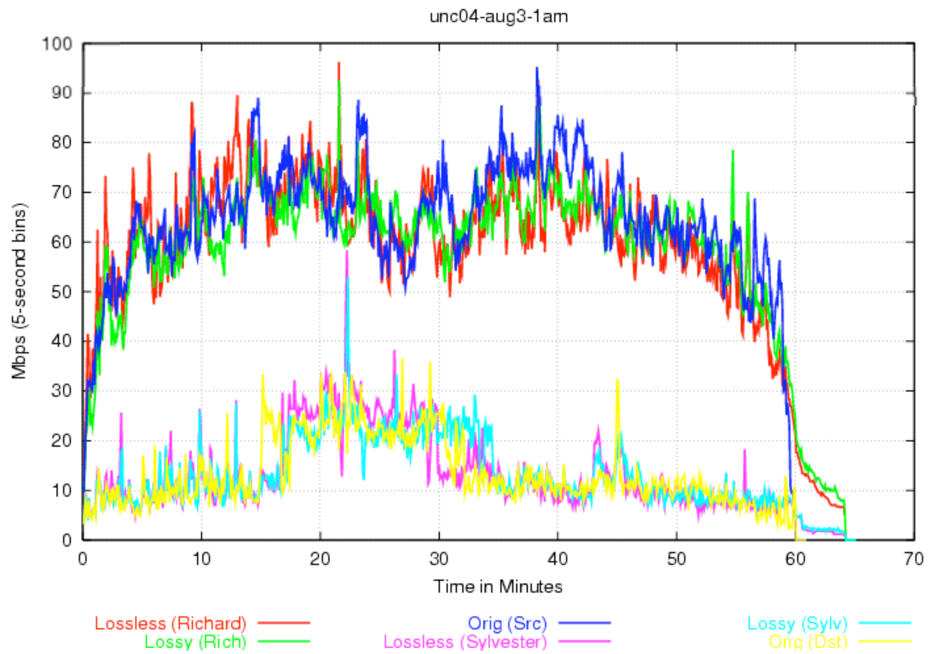
  $((b_1, tb_1), (b_2, tb_2), \ldots, (b_m, tb_m))$
- Concurrency can be detected with high probability
  - $p.seqno > q.ackno$ and $q.seqno > p.ackno$
  - $O(n*W)$

Source-Level Trace Replay

Traffic Generation in Lab Testbed







Conclusion

- New method for modeling traffic mixes
  - Empirically-derived connection vectors
  - Studied sequential vs. concurrent dichotomy
  - Fully automated, efficient analysis
- New traffic generation approach
  - Enables comparison of real and synthetic traffic
  - Implemented a distributed traffic generator
  - Techniques for scaling traffic load