TSS

Time-Step Network Simulation

Andrzej Kochut
Udaya Shankar

University of Maryland, College Park

TSS

Introduction

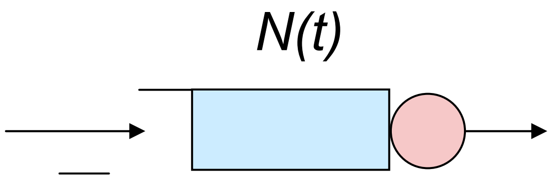
- Goal: Fast accurate performance evaluation tool for computer networks
  - Handles general control schemes (time- and state-dependent)
- Packet-level simulation:
  - Handles general control scheme precisely but prohibitively expensive
- Steady-state exact queuing models
  - Handles only simple models; no transient metrics
- Time-dependent exact queuing model
  - Only very simple systems; no state-dependent control
- Time-dependent stochastic model (fluid and diffusion approximations)
  - Handles time-dependent, but not state-dependent control

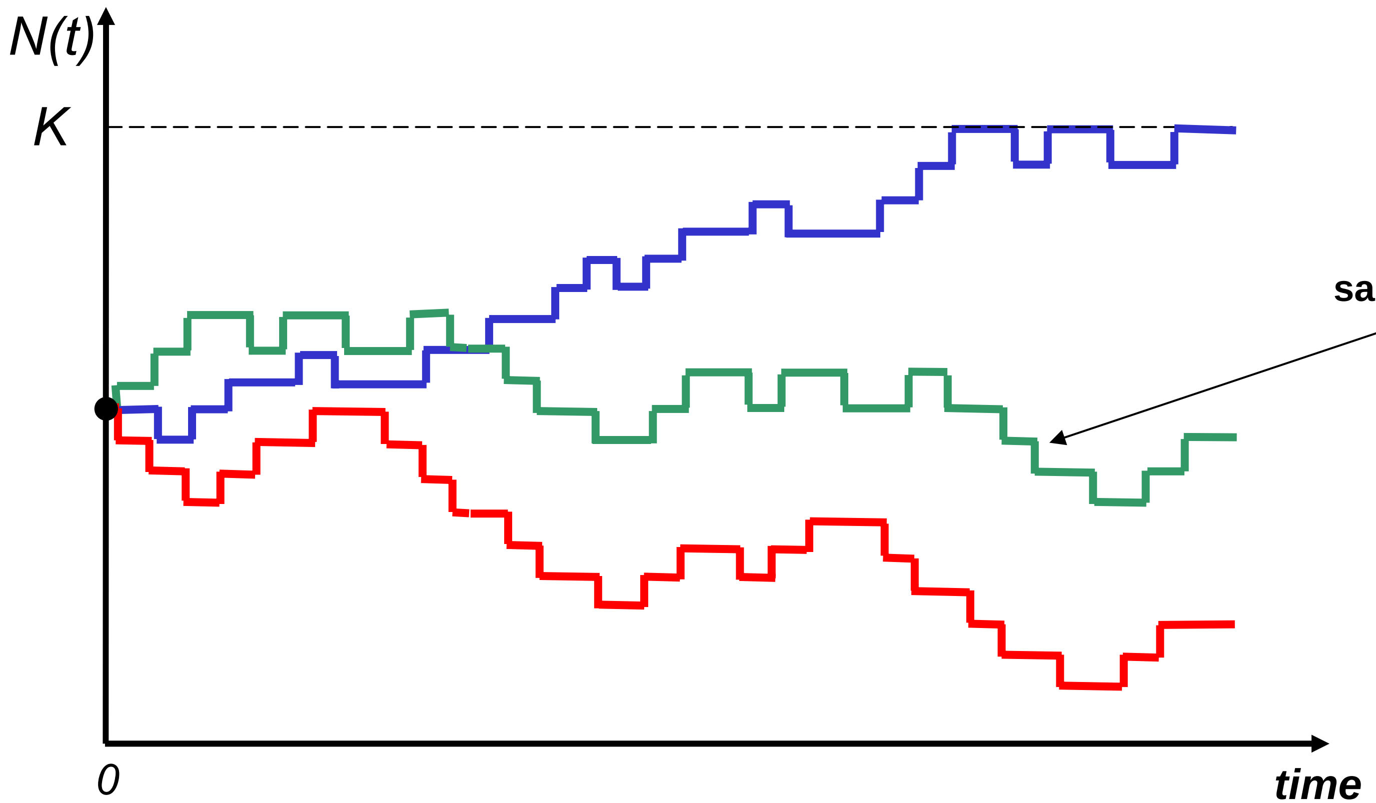
- Approach: Combine discrete-event simulation with diffusion approximation
  - Accurate, inexpensive, handles time- and state-dependent control

TSS

Hybrid time-step simulation

- Consider a single communication link
- Want to generate sample paths efficiently

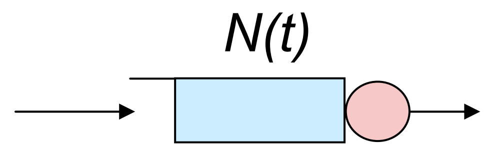


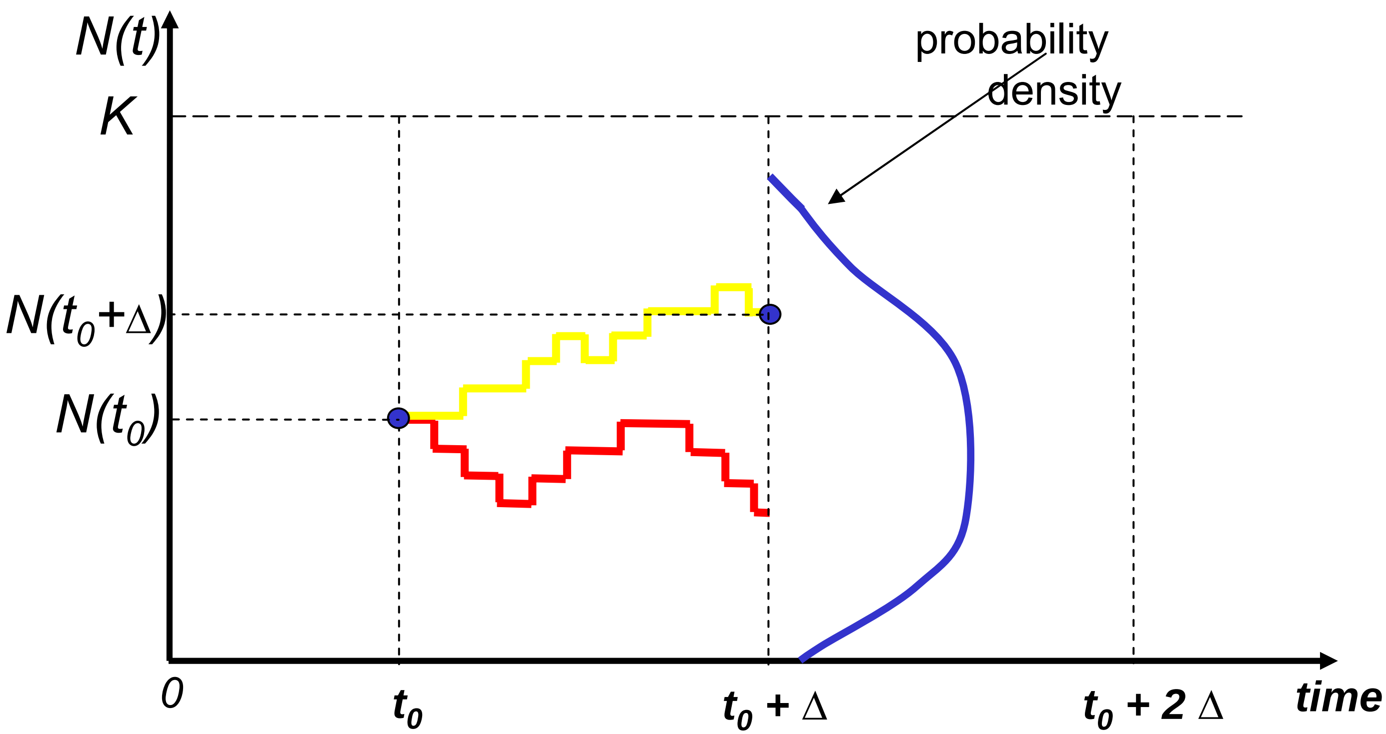


TSS

Hybrid time-step simulation

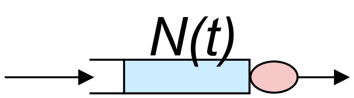
- Divide time axis into small intervals Δ
- For interval $[t_0, t_0 + \Delta]$ choose $N(t_0 + \Delta)$ randomly based on $N(t_0)$ and arrival and service processes
- Repeat for successive time intervals

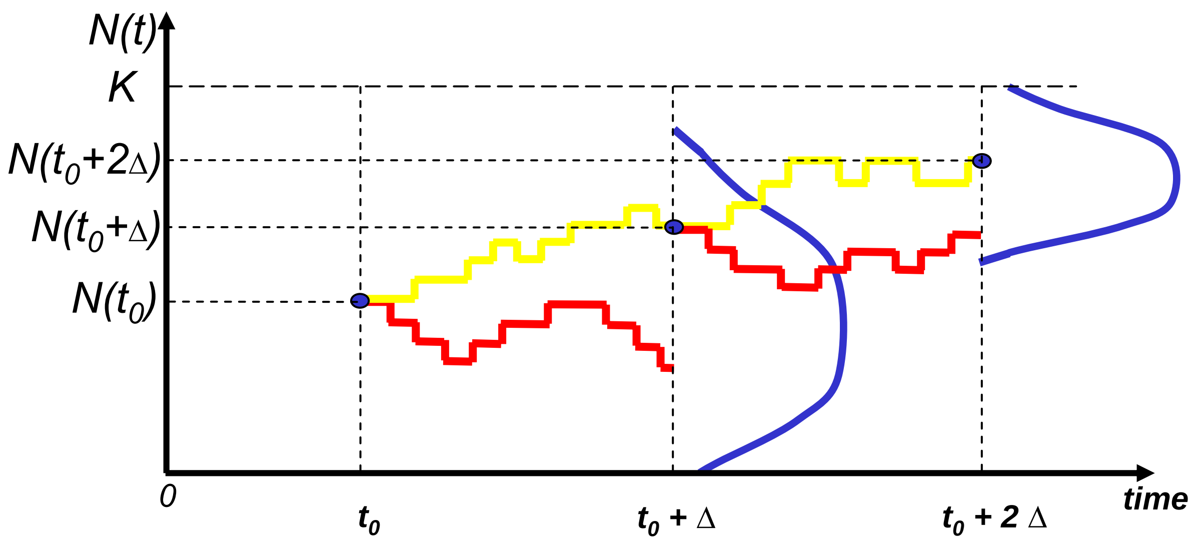




Hybrid time-step simulation

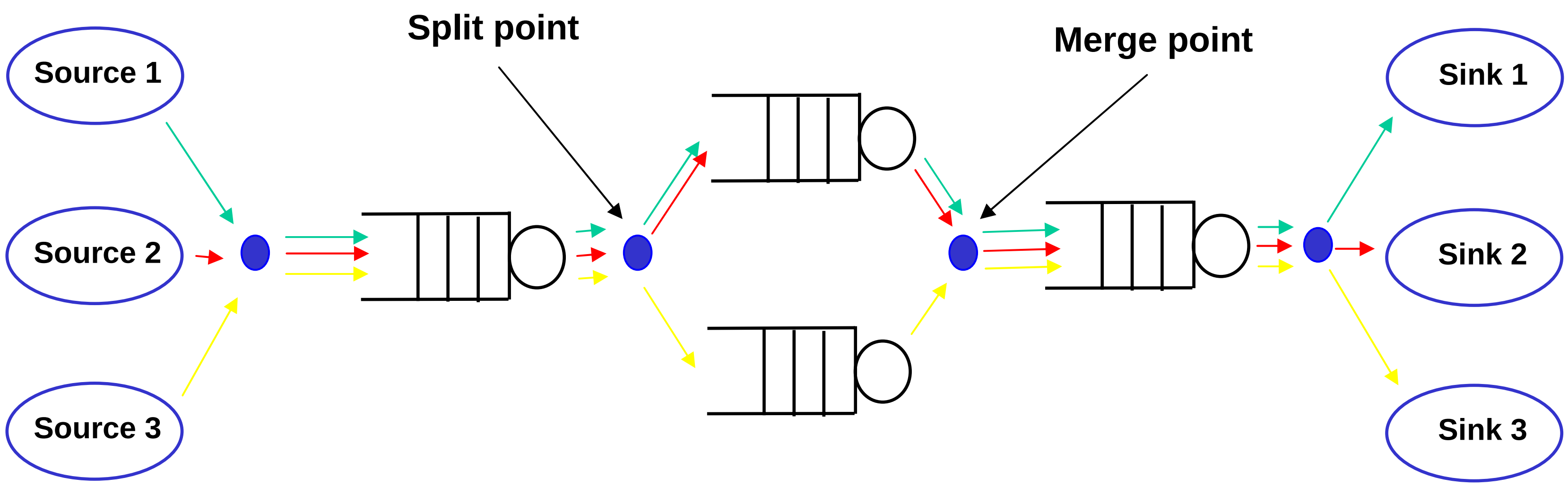
- Time/state dependent sources undergo state changes at every Δ ($\Delta \approx$ time scale of upper-layer control, e.g., RTT for TCP)
- Discrete events are not packet transmissions but time steps
- Captures state-dependent control because sample-path is explicit
- Diffusion approximation [Kolomogorov] to obtain Prob[$N(t+\Delta) \mid N(t)$]
  - Arrival and service processes defined by time-varying mean and variance





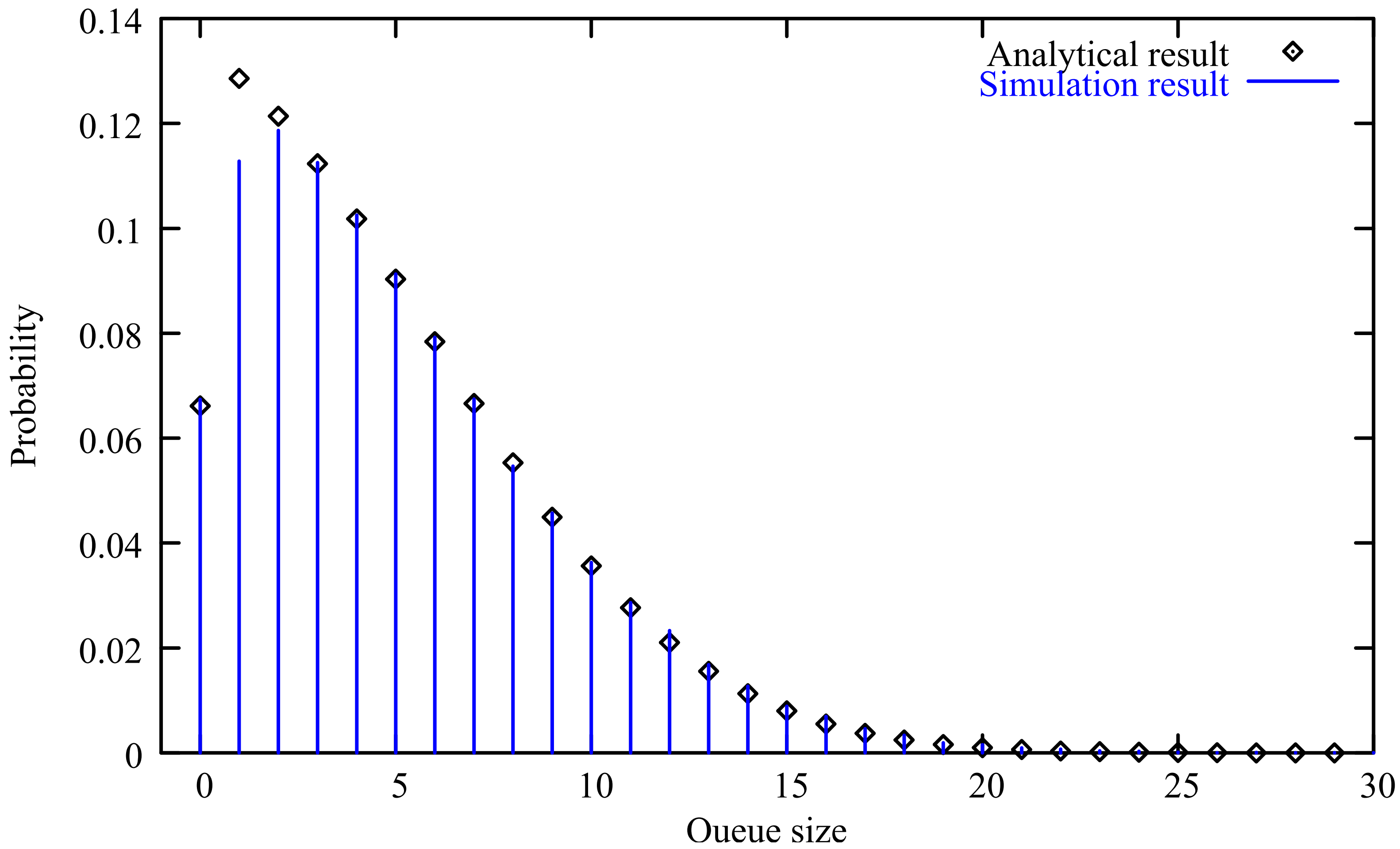
Extension to network of queues

- For each interval [t, $t + \Delta$]
  - Approximate queue departure and internal flows by renewal processes characterized by the first two moments
  - Routing probabilities determined by queue occupancy
- Formulate equations for merging and splitting flows



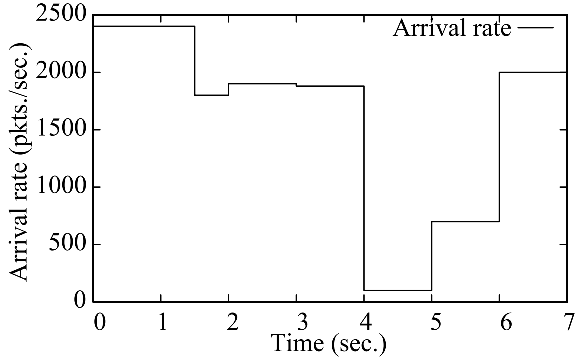
Example: Queue size prob density

- GI/D/1/40 queue, $\lambda=800$, $c_A=1$, and $\mu=810$, $N(t) = 2$, $\Delta=0.05$

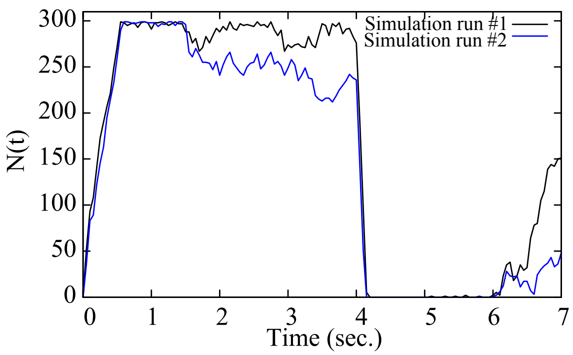


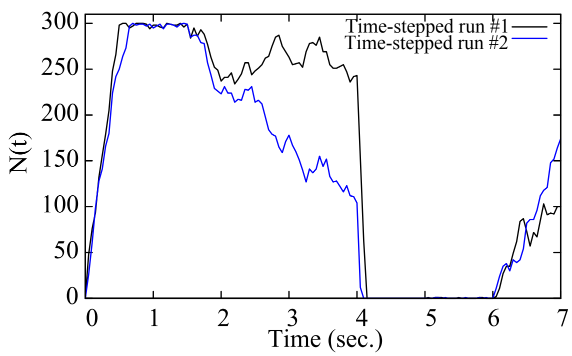
TSS

Example: TSS vs. packet-level simulation



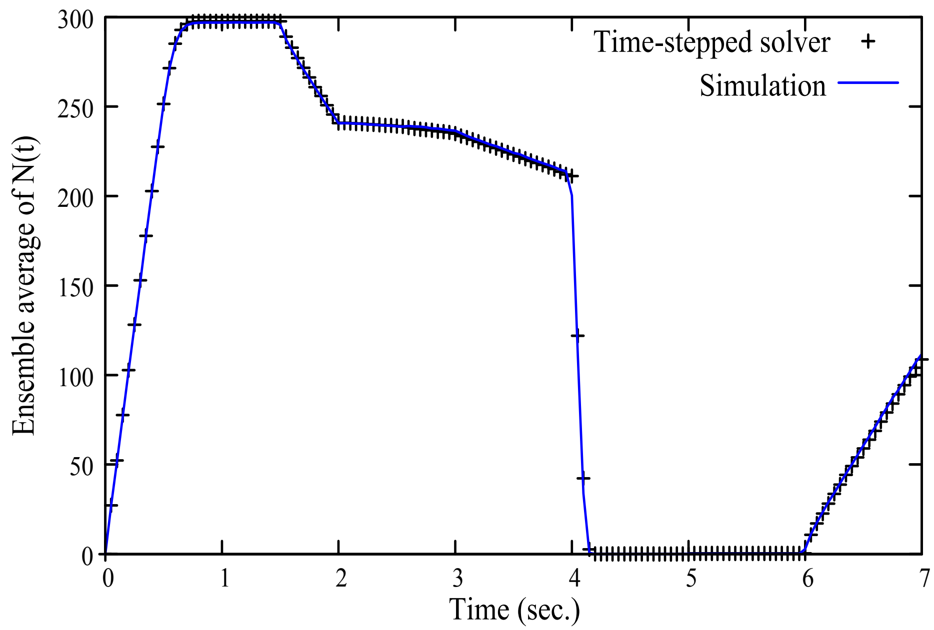
- GI(t)/D/1/300 queue, uniform arrival dist, μ=1900
- Computation time of one run:
  - 10 Mbps link - simulation 1.5 sec., hybrid 0.1 sec.
  - 100 Mbps link - simulation 15 sec., hybrid 0.1 sec.
- Time-step simulator converges faster due to smaller probability space





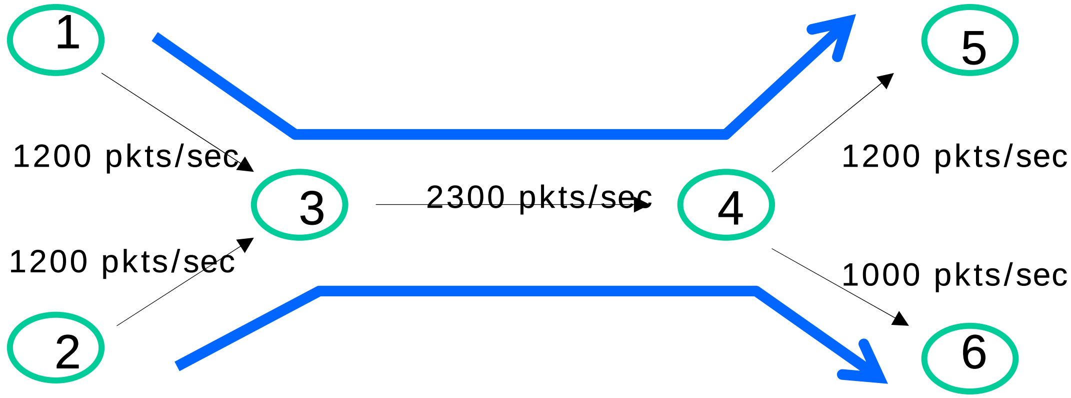
1K runs

100 runs



TSS

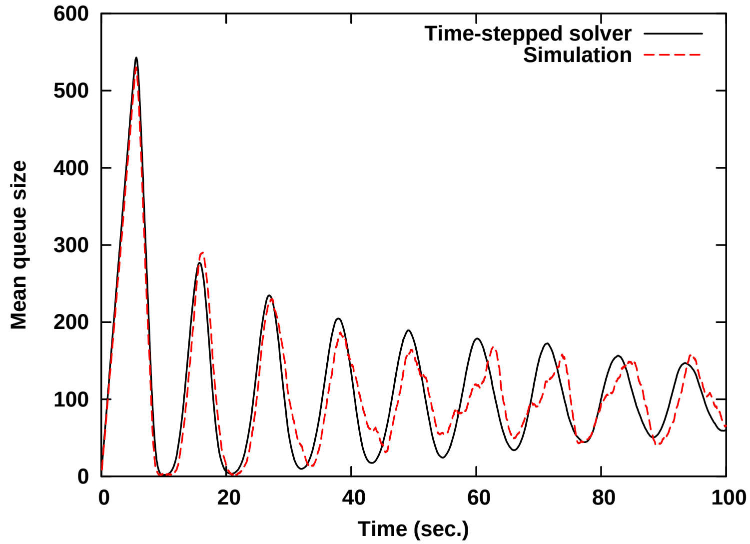
Example: Network with state-dependent traffic sources



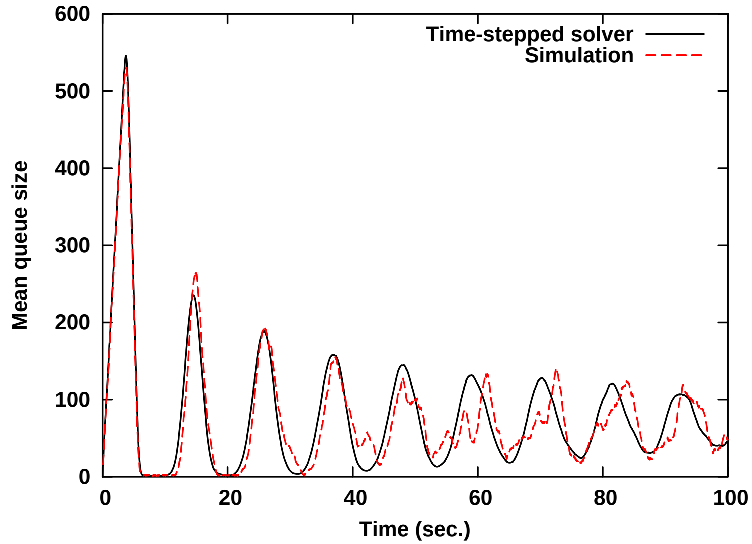
- Traffic flows 1→5 and 2→6 sharing link 3→4.
- Each traffic source:
  - Starts at 1350 pkts/sec
  - 900 pkts/sec when RTT > 1.0 sec
  - 1350 pkts/sec when RTT < 0.5 sec
  - Squared coefficient of variation 1.0
- Service:
  - forward rates as shown above
  - backward rates are all 20000 pkts/sec
  - Squared coefficient of variation of service of all links is 0.0

TSS

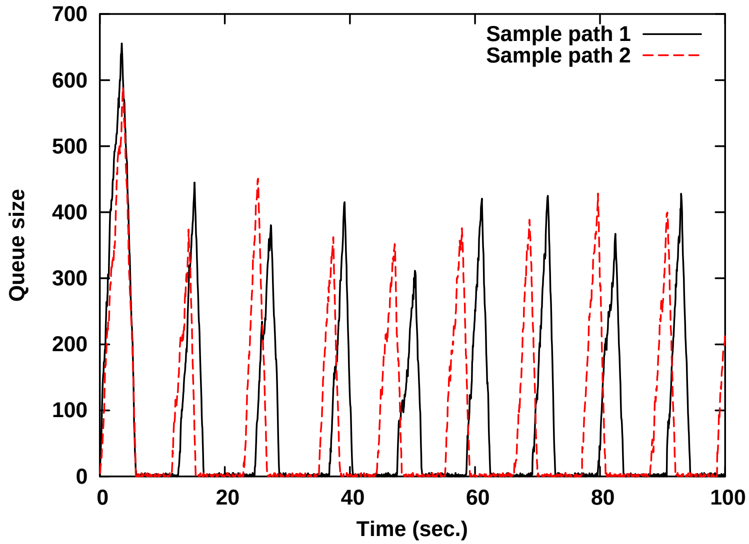
Example: Network with state-dependent traffic sources



Mean queue size of link 3 → 4



Mean queue size of link 2 → 3



Sample paths of queue size of link 2 → 3

Model of the TCP flow (generalized AIMD)

- Congestion window *cwnd(t)* for time interval $[t, t + \Delta]$ based on *RTT* and history

- Send rate of source in interval $[t, t + \Delta]$ is:

$$sndRate(t) = cwnd(t) / rtt$$

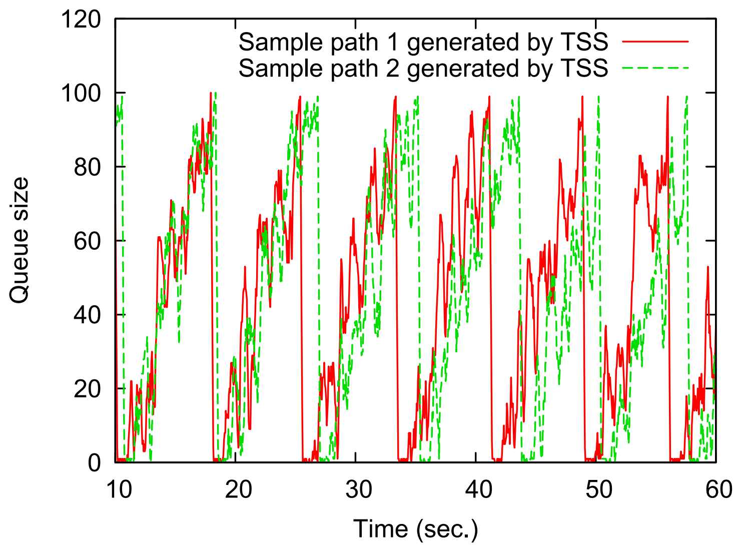
- Loss count in interval $[t, t + \Delta]$ based on probability p of being at the upper boundary:

$$lossCount(t) = arrRate(t) * \Delta * p$$

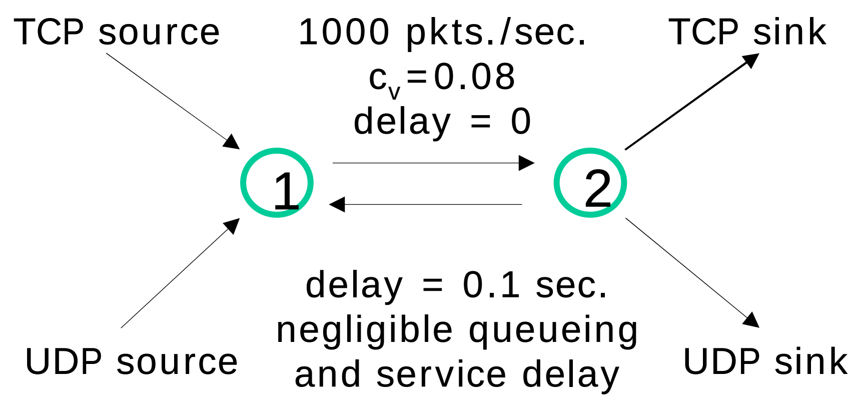
- Losses assigned to flows based on the ratio of arrival rates

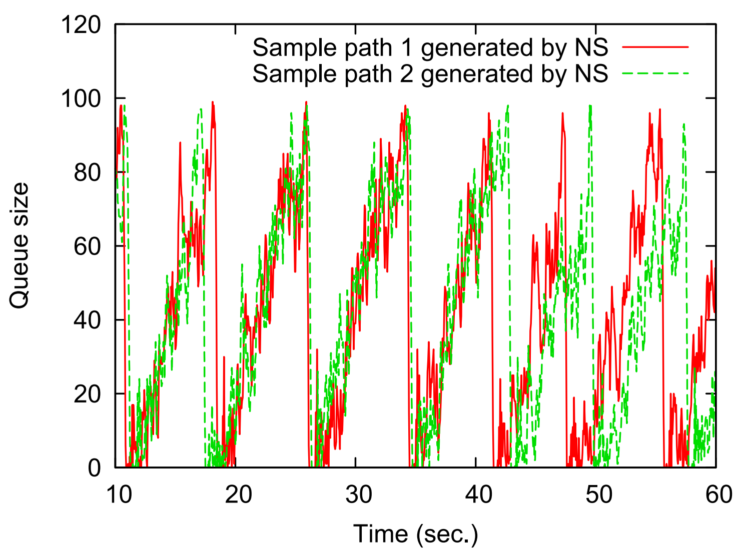
TSS

Example: TCP and UDP sharing a link

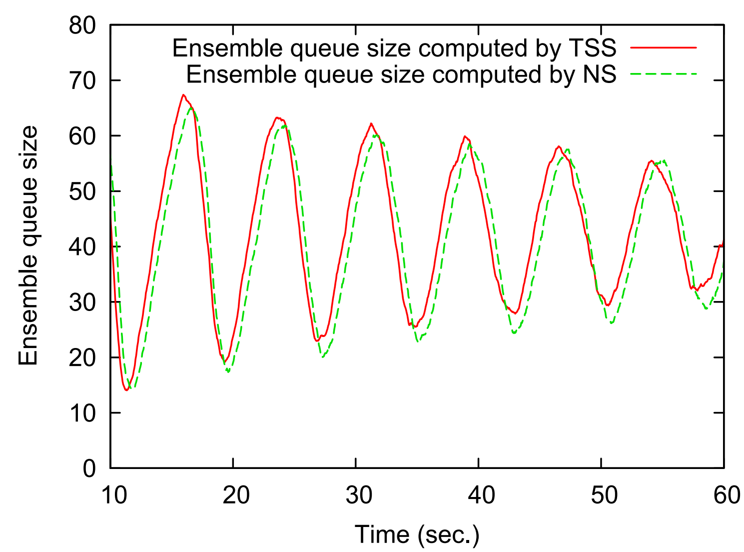


TSS sample paths





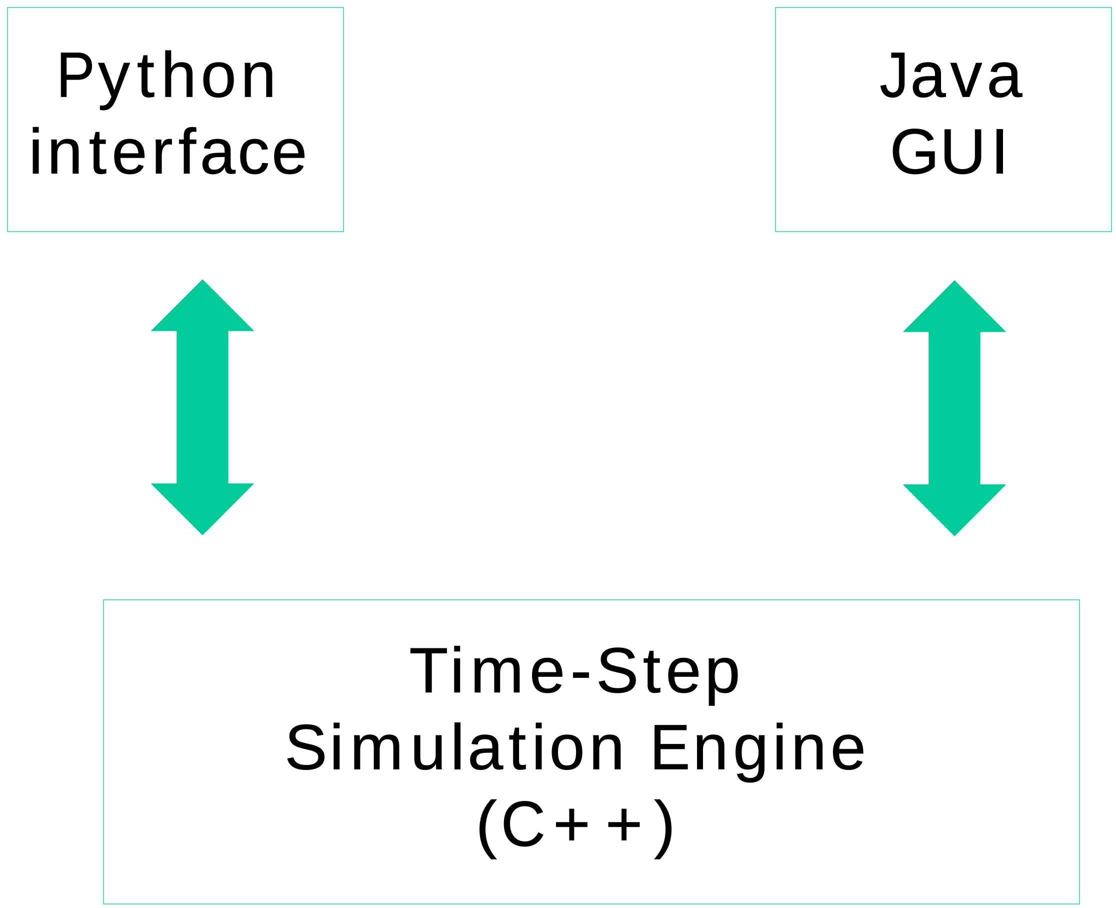
NS sample paths



mean queue size – TSS vs. NS

TSS

Software architecture



Simulation script - example

```
runTime = 400
nrRuns = 2000

N = Network()
N.addNode(1)
N.addNode(2)
N.addLink(0, 1, 2, 2000, 1200, 0.05, 0, 0)
N.addLink(1, 2, 1, 300, 10000, 0.1, 0, 0)

N.addFlow(1, “TCP", "", 1, 2, 0, 10)

N.initSolver()
time = 0
 while time <= runTime:
        log_links(N, time)
        time = N.makeStep()
```

set parameters

define topology

define flows

run simulation

TSS

Time-step simulation - Conclusions

- Time-stepped simulation using diffusion approximation
- Fast and accurate alternative to packet-level (discrete-event) simulation
- Computational complexity not affected by increasing link bandwidth
- Handles state-dependent control schemes
- Yields time-dependent evolution of performance metrics

- Ongoing work
  - Extend queue model to handle wireless links (802.11)
  - Extend to other router disciplines (RED, AQM, CBQ)
  - Optimize numerical computation
  - Detailed comparisons against packet-level simulation for large networks