Emulation of Ad Hoc Networks

David A. Maltz, Qifa Ke,
David B. Johnson

CMU Monarch Project

Computer Science Department
Carnegie Mellon University

`http://www.monarch.cs.cmu.edu/`

Carnegie Mellon

Early Emulation Successes

Stress testing of networking code via ***macfilter***

- Used to debug our 8 node ad hoc network testbed
- Description of interesting/problematic movement scenarios created
- Changing network topology extracted into ***trace file***
- Physical machines running actual implementation synchronously read trace-file
- Machines prevented from receiving packets from non-neighbors
- Described at previous IETF and in CMU CS Tech Report 99-116

Found to be critical for achieving code stability, however ***cannot be used for performance evaluation***

- Propagation model grossly simplified
- MAC-layer effects unaccounted for

Goal of Current Emulation Work

Evaluate real systems under ad hoc network conditions:

- ***Performance study*** of network protocols or applications
- Without implementing them in ***ns-2***
- Without deploying and operating physical machines in the field
- Using ***real*** implementation and ***real*** user pattern
- ***Only*** the network environment is simulated

Evaluating real systems with just simulation is not practical:

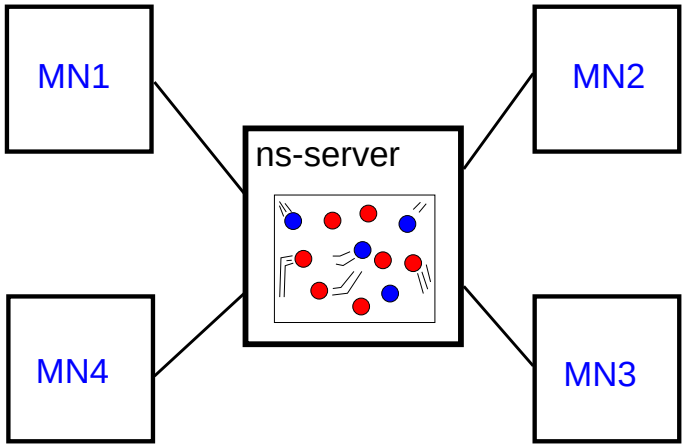
- Difficult to model real applications inside the simulator
- Real systems are significantly performance tuned

Example: Evaluate performance of CMU Coda file system in an ad hoc network:

- Coda is 100+ KLOC implemented and tuned
- Coda researchers have expertise to evaluate their system but can’t create network environments

Method

- Leverage emulation work by Kevin Fall and VINT project
- ns-server is a single central machine running ***ns-2***
- Real application code runs on real machines directing real packets to “default router” (ns-server)
- User creates scenarios to control the environment inside ***ns-2***:
  - Movement pattern of all mobile nodes inside ***ns-2***
  - Background traffic



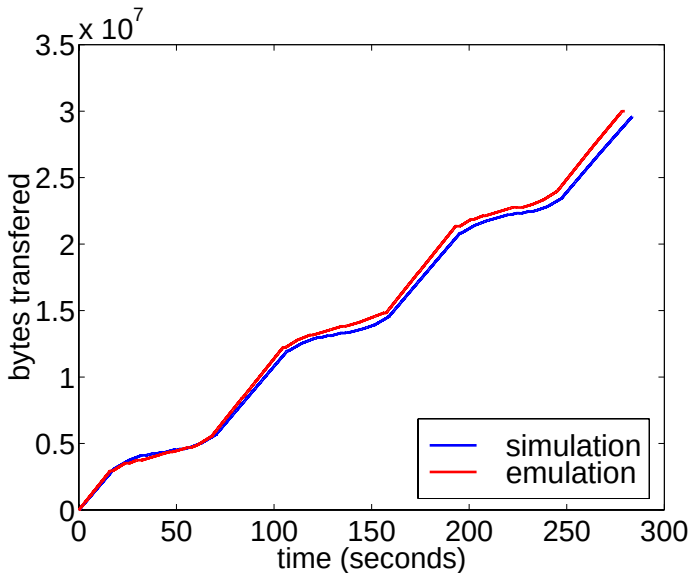
Logical view of emulation setup

Informal Validation of Emulation Method

Example: Compare FTP behavior between simulation and emulation:

- ns-server: 450 MHz Pentium II CPU, running FreeBSD 3.1
- 10 simulated nodes
- Compare 30 MB FTP transfer between simulation and emulation
- 2 additional simulated FTP connections for background traffic

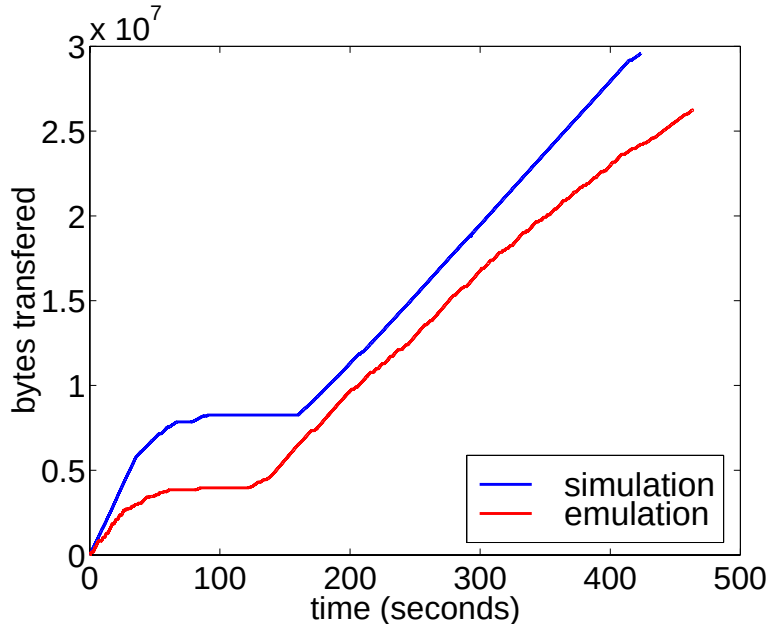
TCP time–sequence # plots
have basically the same shape:



But how well does this scale?

A More Challenging Example

- 16 simulated nodes moving in a complicated pattern with CBR and FTP background traffic
- 30 MB FTP transfer simulated in ***ns-2***
- 30 MB FTP transfer between two physical nodes with emulation



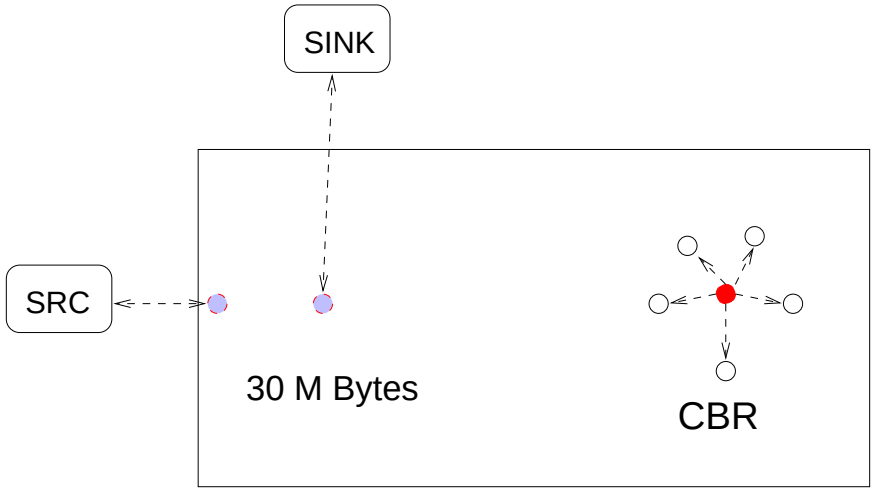
Evaluating Emulation Scalability

Limitations on emulation scalability:

- System effects: scheduling delays, potential buffer overruns
- Additional latency of transferring packets to/from ns-server
- Only so much one CPU can do

Experimental scenario:

- Conduct real FTP transfer between 2 physical nodes
- Vary the number of other simulated nodes in ***ns-2***
- Vary the amount of background traffic

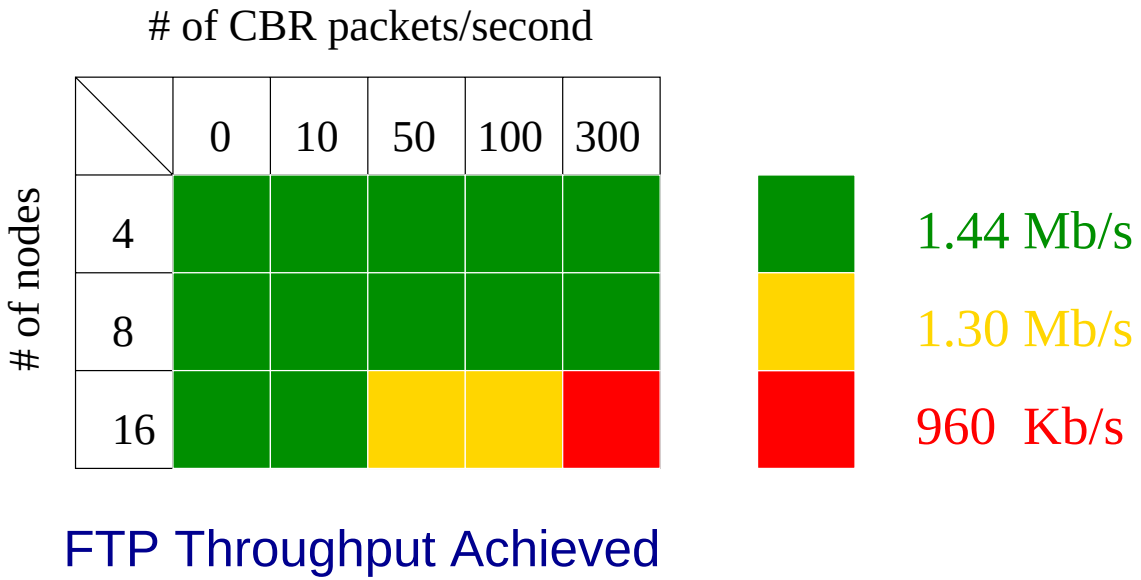


Scalability Test Scenario

Scalability Results Summary

Ideally: emulation throughput = simulation throughput

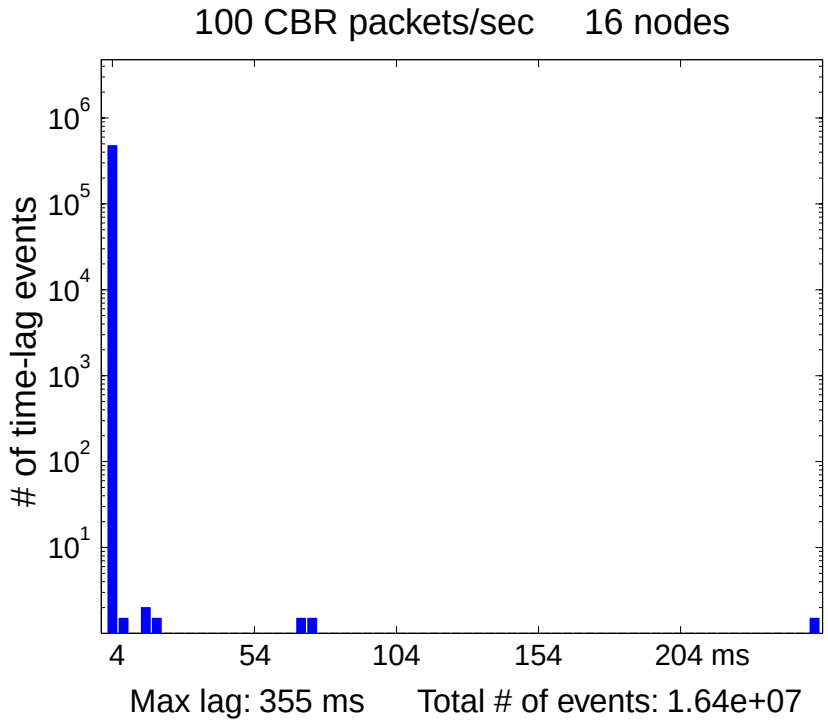
- Pure simulation achieves 1.49 Mb/s on a 2 Mb/s link
- Emulation achieves throughput that depends on overall simulator workload:



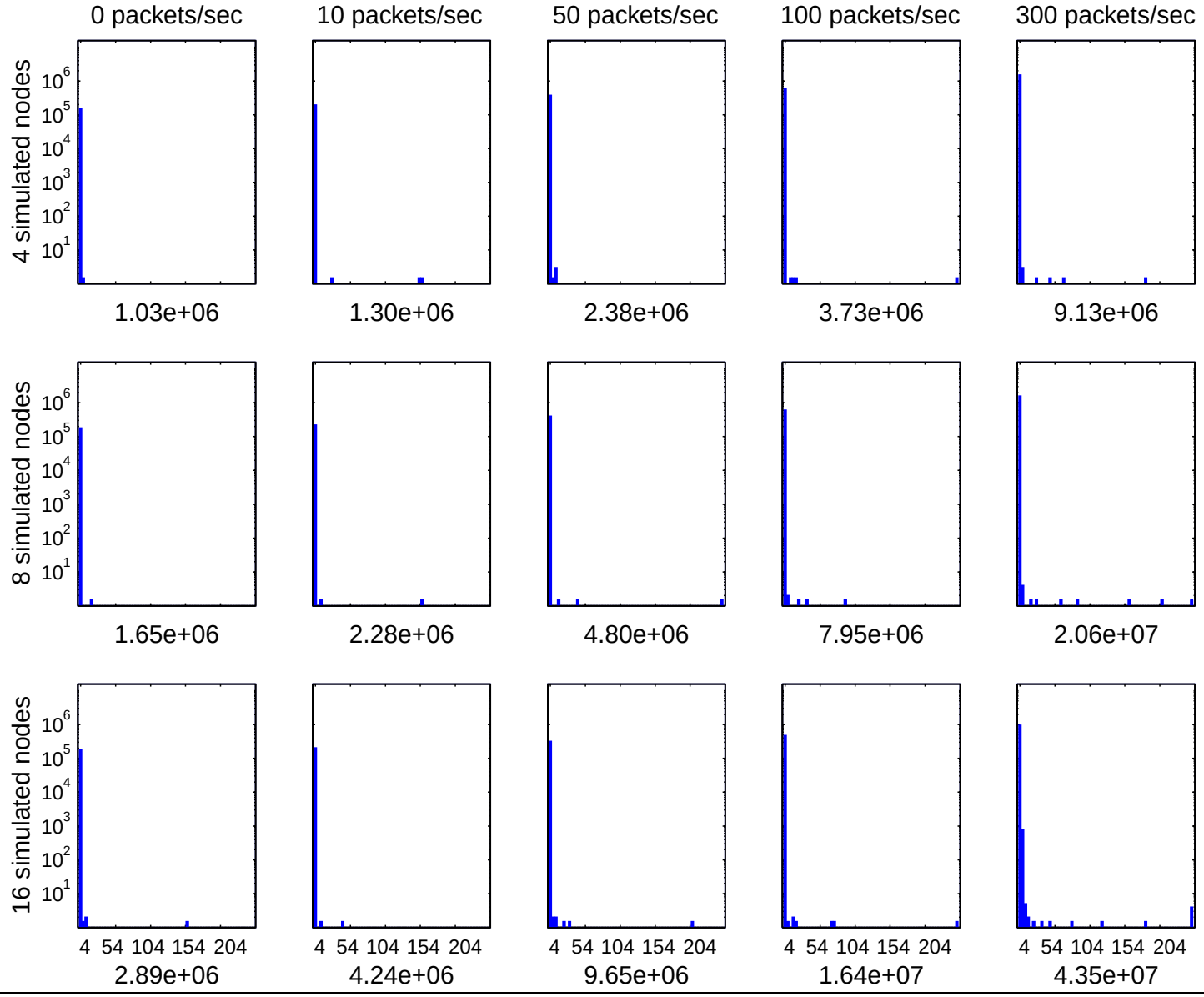
FTP Throughput Achieved

Does the Simulator Keep up with Real Time?

- For event scheduled at t_1 but processed at t_2: time-lag = $t_2 - t_1$
- Record the time-lag for each delayed event
- Histograms show almost all time-lags are less than 4 ms



All of the Histograms



monarch@monarch.cs.cmu.edu

Conclusion

Emulation works for networks of sufficient size and complexity to enable study of interesting application scenarios

- Using direct emulation based on ***ns*** leverages future improvements to propagation models
- The bottleneck is the real-time requirement in ns-server
- Currently using to evaluate the Coda distributed file system

Open questions:

- How far can this technique be generalized?
- How to ensure performance measurements are meaningful?
  - Open loop: evaluate emulation system on many scenarios using reference application with good simulation models
  - Closed loop: after each emulation run, examine emulation trace to determine if emulation system introduced artifacts
  - Validate against real system in ad hoc network testbed