Congestion Modeling for Reconfigurable Inter-Processor Networks

W. Heirman, J. Dambre, J. Van Campenhout
ELIS Department, Ghent University, Belgium

Sponsored by IAP-V 18, Belgian Science Policy Office

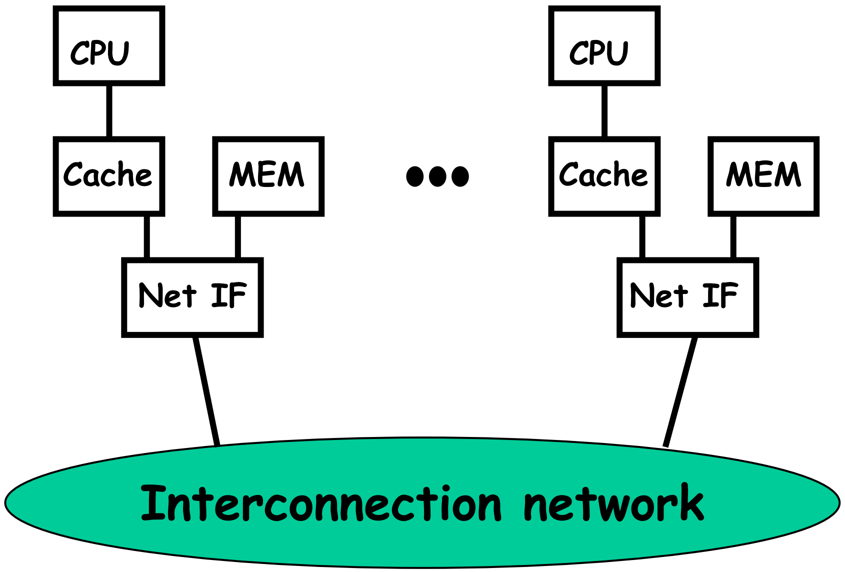
Objective

- Reconfigurable Inter-Processor Network: packet-switched network between CPUs
- Previous work: quick estimate of network performance (end-to-end packet latency), for different network parameters
- Allows fast design-space exploration
- Missing: model of packet congestion

Outline

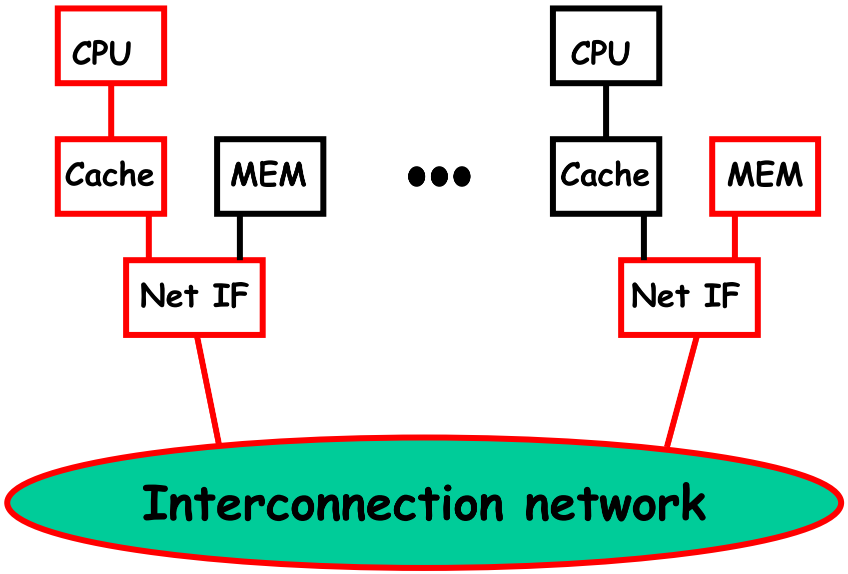
- Introduction
- Prediction Model
- Results
- Conclusions

Architecture of a distributed shared-memory system



- Nodes:
  - Processor
  - Caches
  - Main memory
  - Network interface
- Interconnection network
  - Packet switched

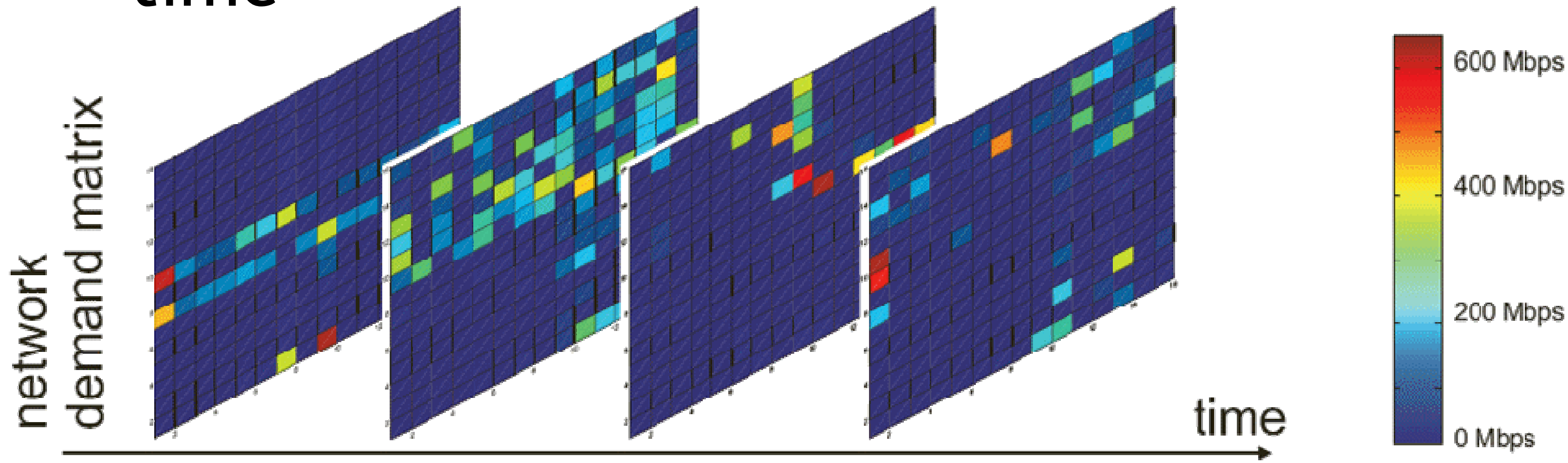
Architecture of a distributed shared-memory system



'Remote' memory access: handled by the network interfaces, requires use of the interconnection network

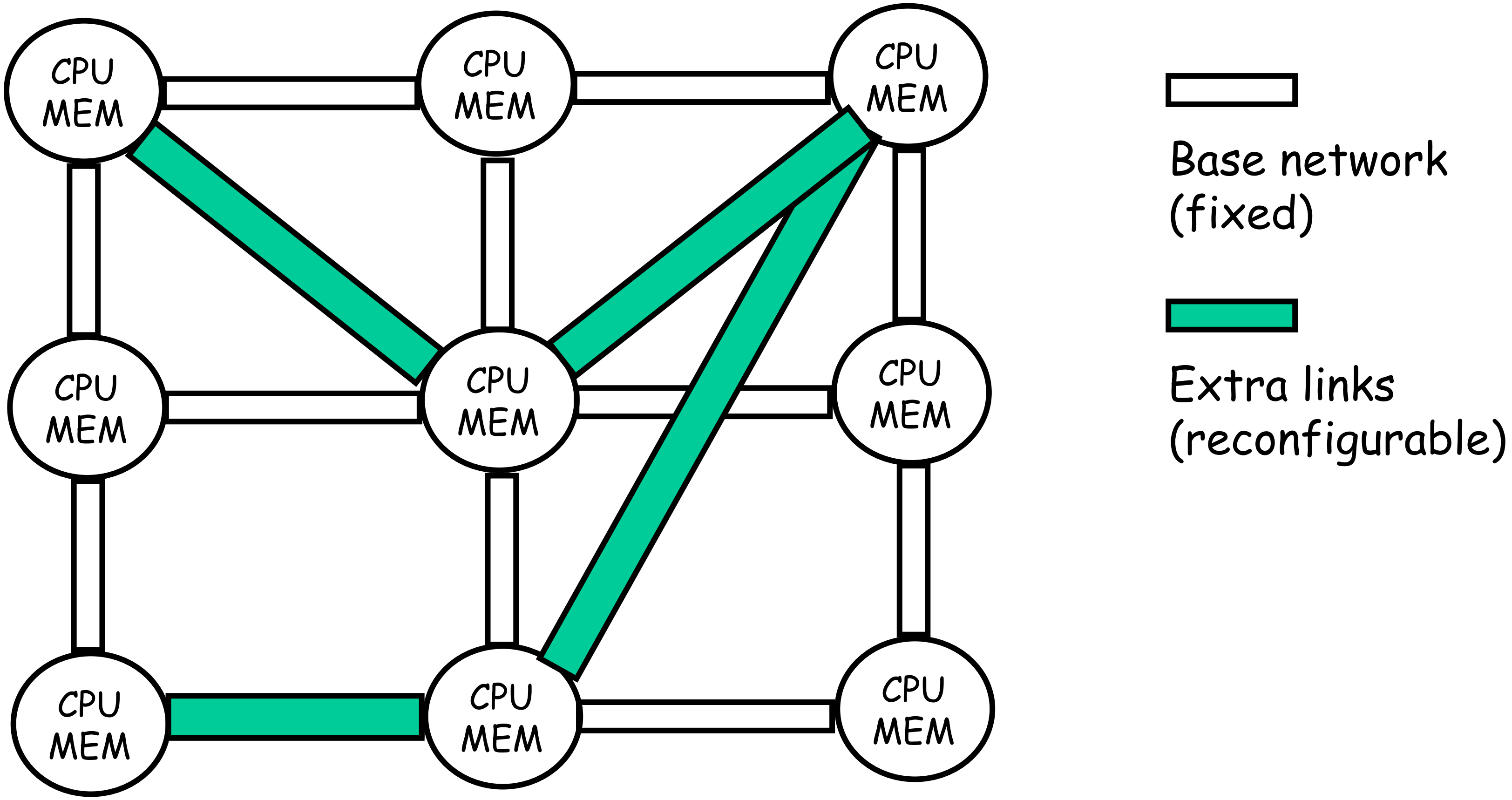
Interconnect requirements

Non-uniform network traffic in space and time



=> Reconfigurable network?

Reconfiguration in shared-memory machines



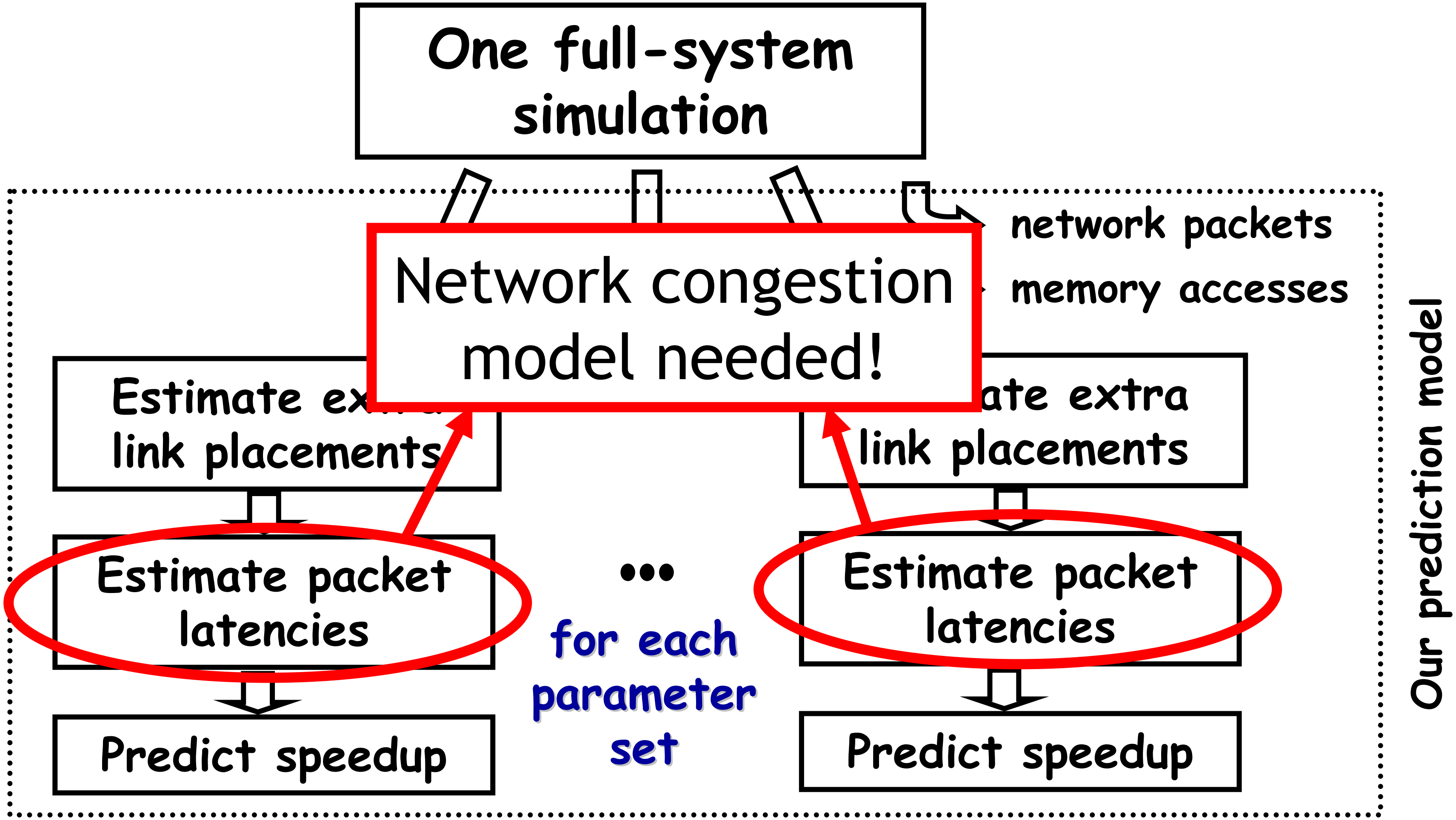
How to design such a network?

- Do design-space explorations
- Lots of parameter combinations to explore
- Full-system simulations are very accurate, but also very slow

Can we do this faster?

- Predict key performance indicators for different network parameters, based on one full-system simulation

Predicting network performance

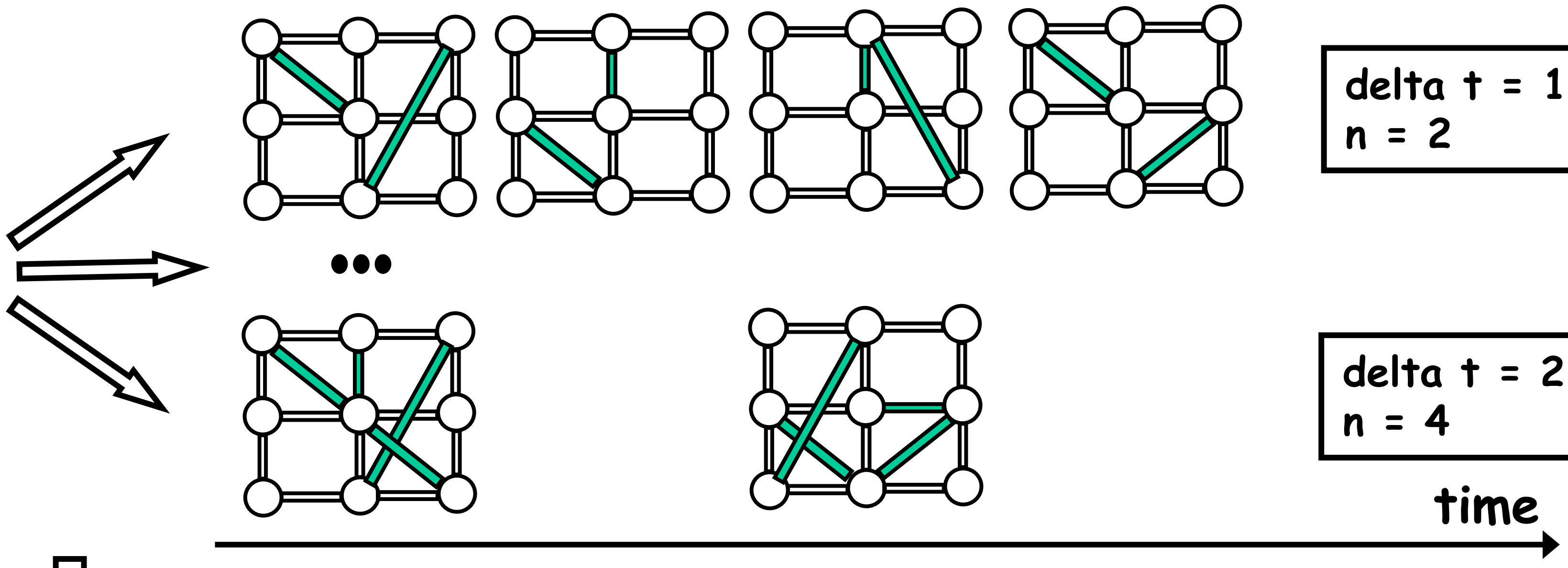


Outline

- Introduction
- Prediction Model
- Results
- Conclusions

Predicting network performance

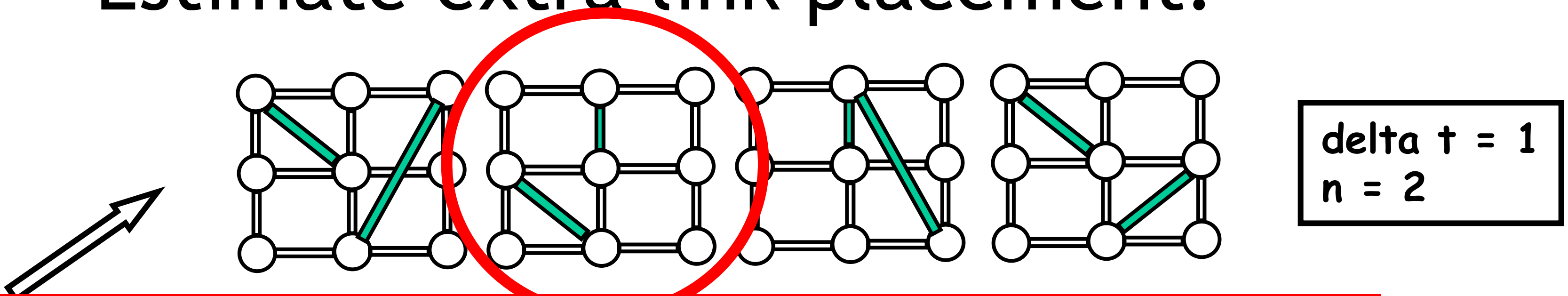
- Estimate extra link placement:



Parameters: reconfiguration interval (delta t), number of extra links (n), link placement algorithm

Predicting network performance

- Estimate extra link placement:



Estimate packet latency

- Given the topology (extra link placement) in this interval
- Given the traffic matrix for this interval

Repeat for each interval

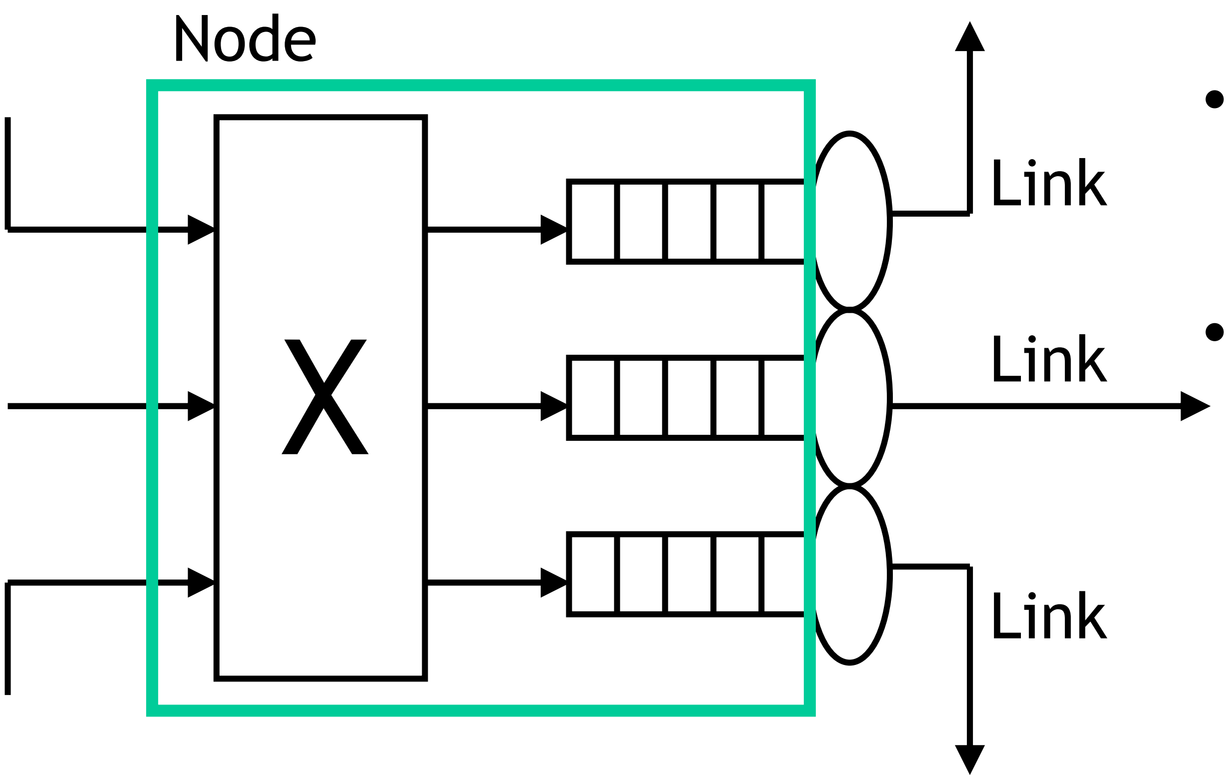
ta t = 2
4

time

elta t),
nent

algorithm

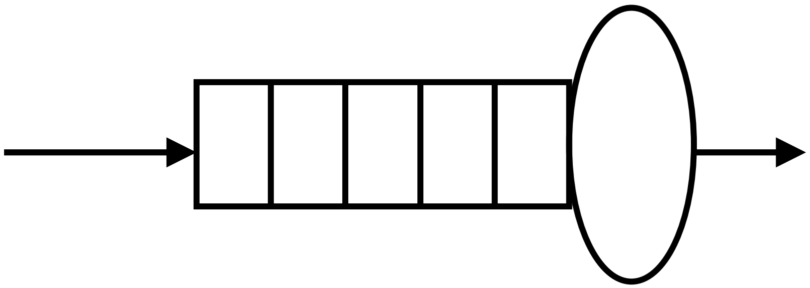
Modeling the network: queues and servers



Model the network as a set of

- Nodes
  - containing buffers (queues)
- Links
  - providing the 'service': transmission over a slow channel

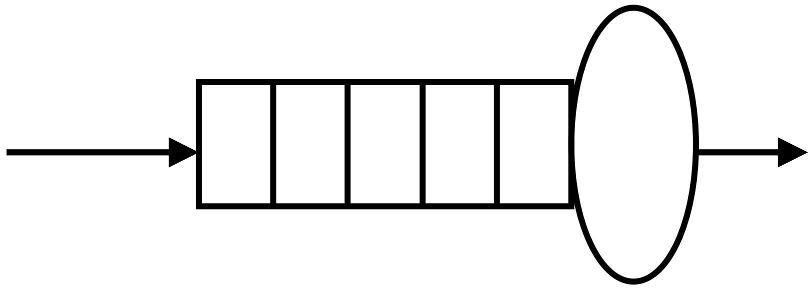
Predict per link waiting time



$$E[W] = \frac{\lambda E[S^2]}{2(1-\rho)}$$

- For each buffer+link system & each time interval: calculate average packet waiting time
- Using “Pollaczek-Khinchin” mean formula
- Link load is known: derived from list of packets from full simulation and routing algorithm
- Assumes network traffic will not change after adding extra links

Predict per link waiting time



$$E[W] = \frac{\lambda E[S^2]}{2(1-\rho)}$$

E[]	Average
W	Waiting time
λ	Arrival intensity
S	Service time distribution
ρ	Server (link) load

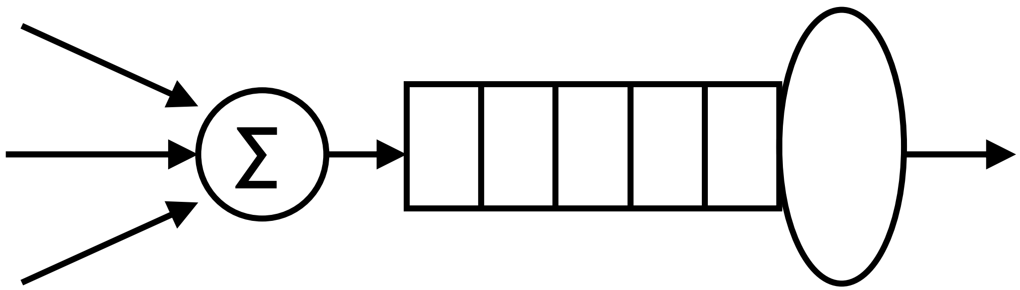
Calculate from traffic pattern measured in full simulation:

λ = # packets / interval length

ρ = total packet size / link bandwidth x interval length

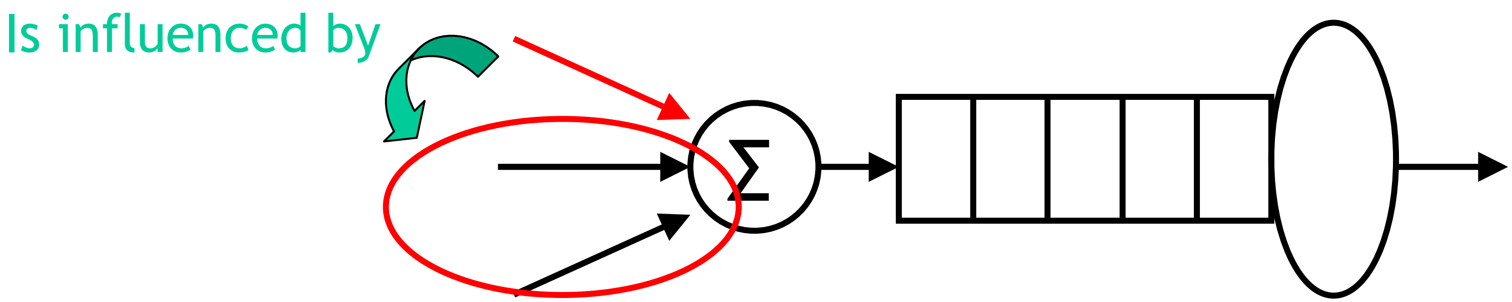
S ~ packet size distribution

Predict per link waiting time



- Poisson process assumes packet arrivals are independent (memory-less)
- Packets enter each buffer+link system through a number of *incoming links*
- Incoming links clearly have memory:
  - Packets take several cycles to transit, this determines the minimum time between 2 packets from one incoming link
  - Link ‘remembers’ when the previous packet passed

Predict per link waiting time

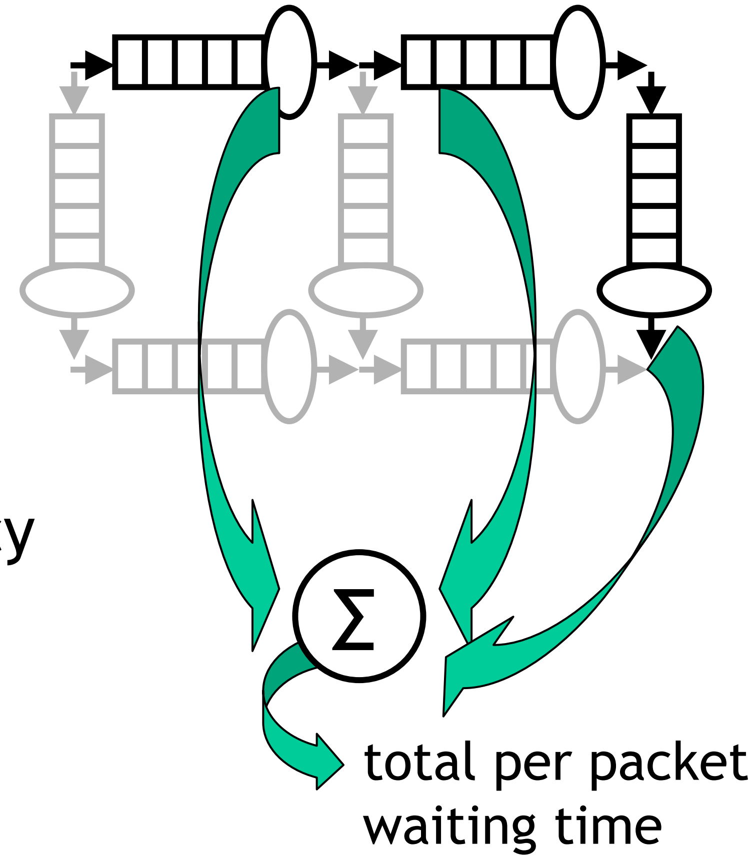


Therefore:

- split packets according to their incoming link
- only use packets from other links to determine waiting time for packets on this link:
  to compute $E[W]_i$ (waiting time for packets entering through link *i*), λ_i, ρ_i and S_i are computed as if link *i* did not exist

Predict total packet latency

- For each packet: determine the path followed through the network
  - Add per link waiting times to obtain total waiting time
  - Add uncongested latency to obtain end-to-end transmission time
- Average over all packets and all intervals…



Outline

- Introduction
- Prediction Model
- **Results**
- Conclusions

Simulations

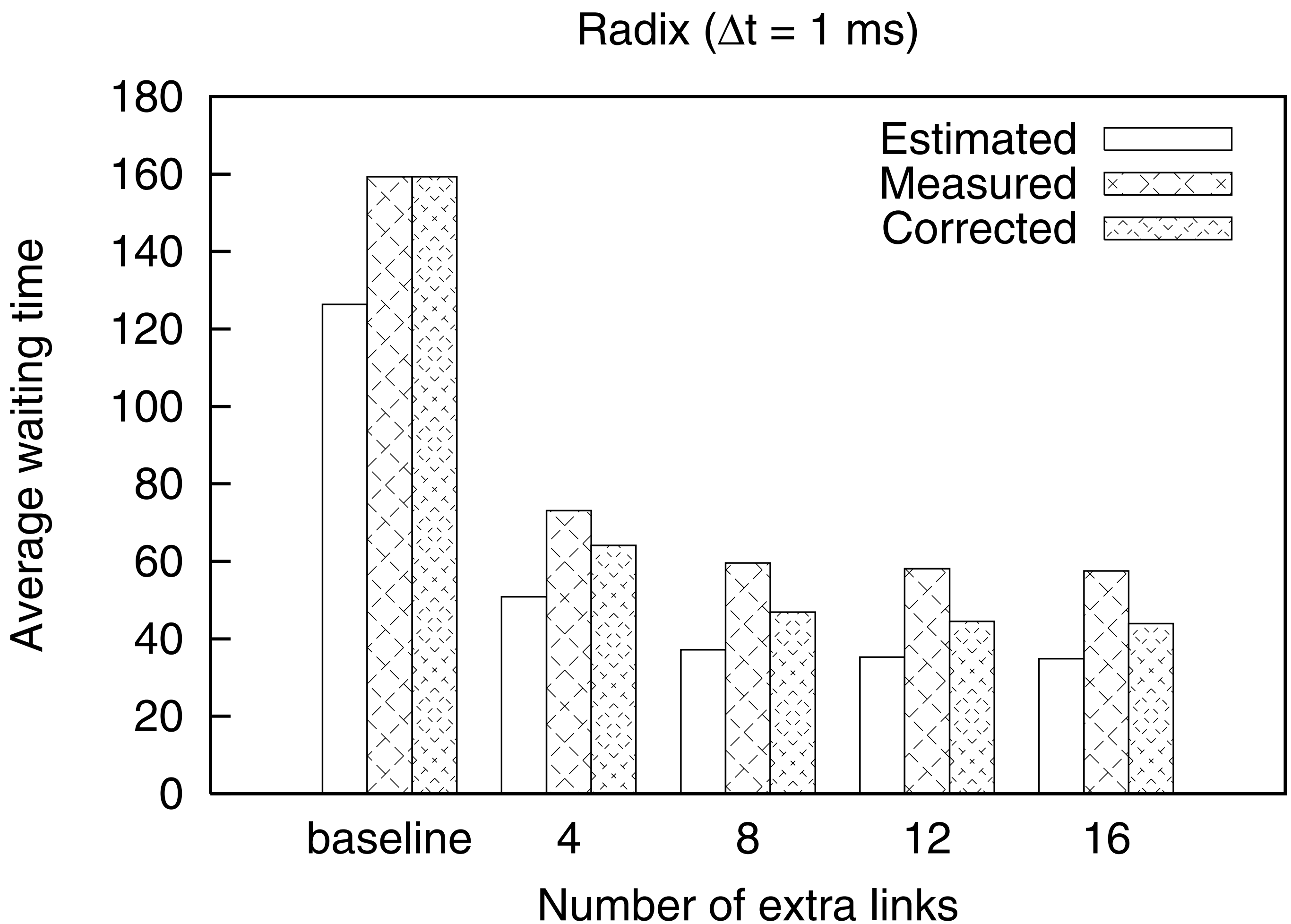
Simulated architecture:

- 16 UltraSPARC CPU's, distributed shared memory
- SPLASH-2 benchmark applications
- Base network: 4x4 torus

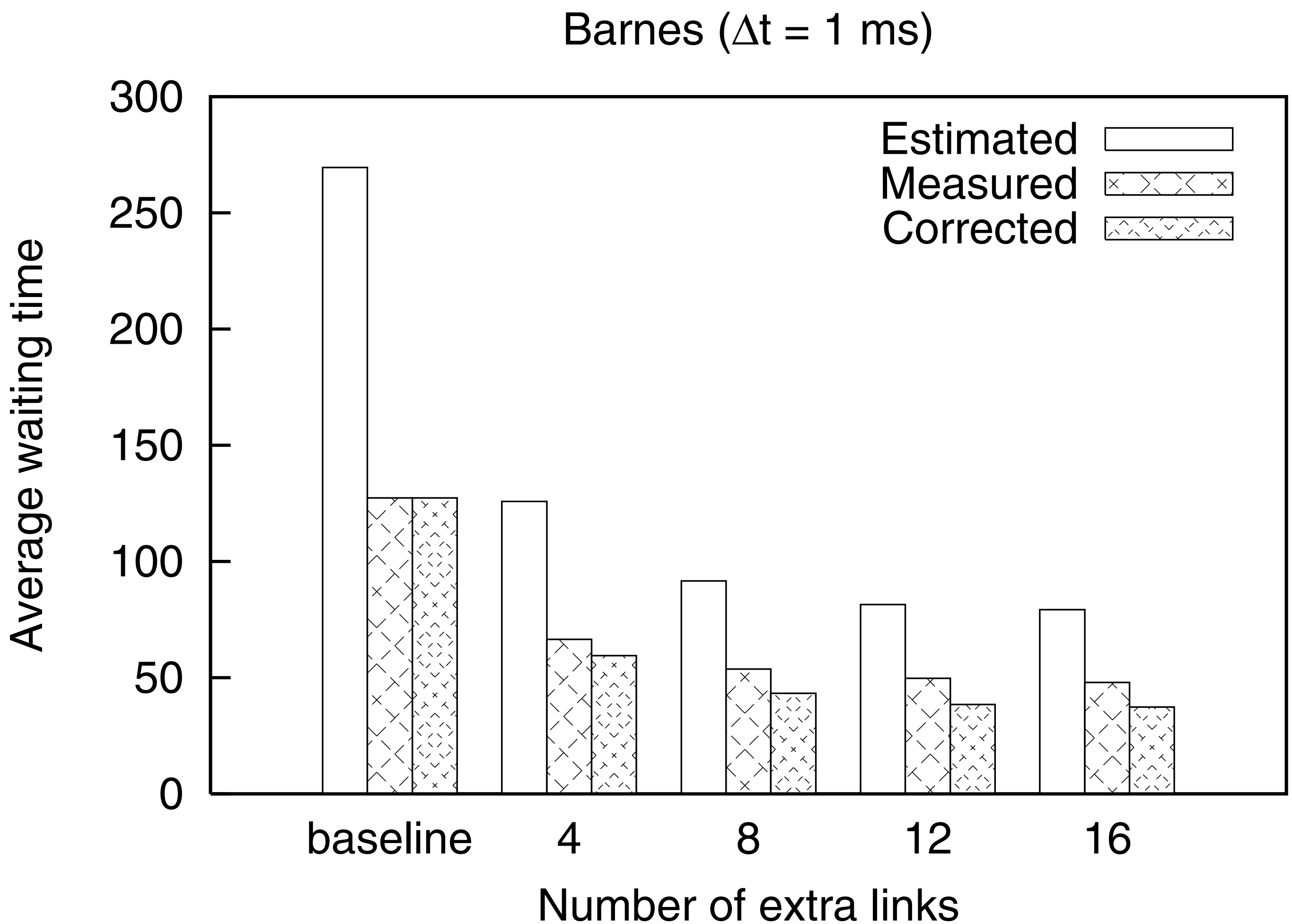
For a given set of extra link parameters:

- Run simulation without extra links and apply our model on the list of packets
  ➔ *estimated* waiting time
- Run simulation with extra links and measure latency directly ➔ *measured* waiting time

Results

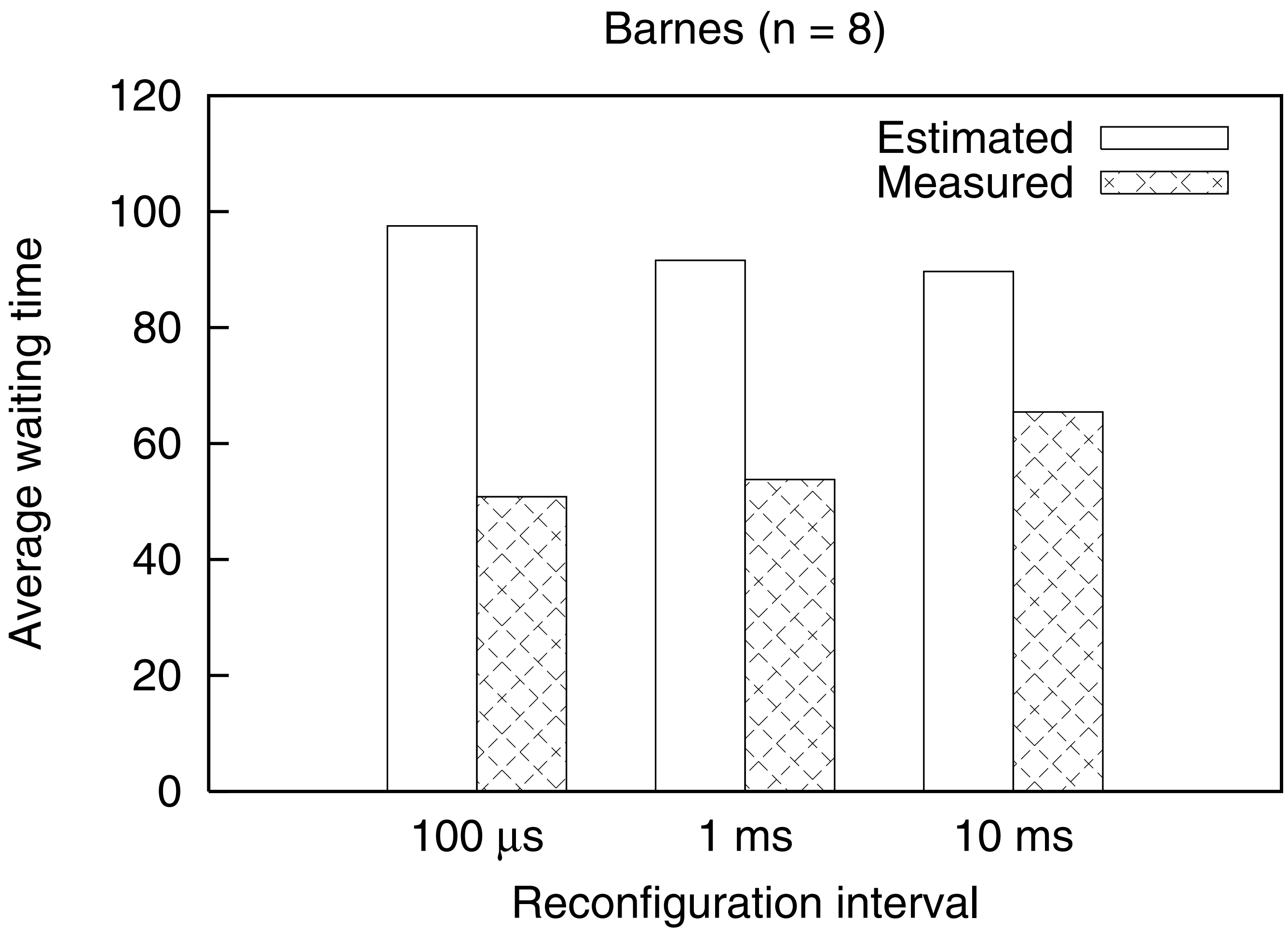

Radix (Δt = 1 ms)
Estimated
Measured
Corrected
180
160
140
120
100
80
60
40
20
0
Average waiting time
baseline
4
8
12
16
Number of extra links

Results



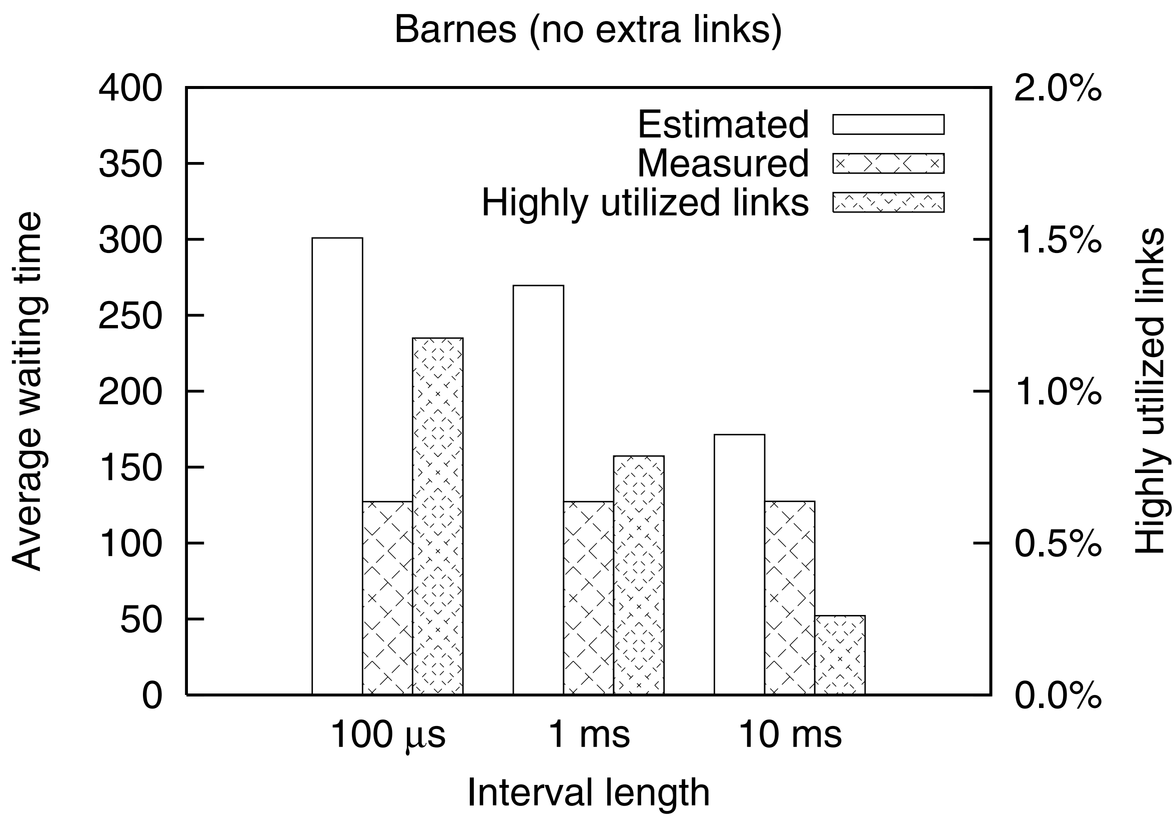
Barnes (Δt = 1 ms)
Estimated
Measured
Corrected
300
250
200
150
100
50
0
Average waiting time
baseline
4
8
12
16
Number of extra links

Results



Barnes (n = 8)
120
100
80
60
40
20
0
Average waiting time
Estimated
Measured
100 µs
1 ms
10 ms
Reconfiguration interval

Results



$$E[W] = \frac{\lambda E[S^2]}{2(1-\rho)}$$

Problem: when ρ → 1 , E[W] → ∞

"Highly utilized links": ρ > .9

Outline

- Introduction
- Prediction Model
- Results
- Conclusions

Conclusions

- Using standard queuing theory, packet congestion can be predicted
- Our model has good relative accuracy for different network topologies
- Changing the time interval length causes too much variation, so comparing interval lengths is not yet possible
- Handling of highly utilized links should be improved

Thank you!

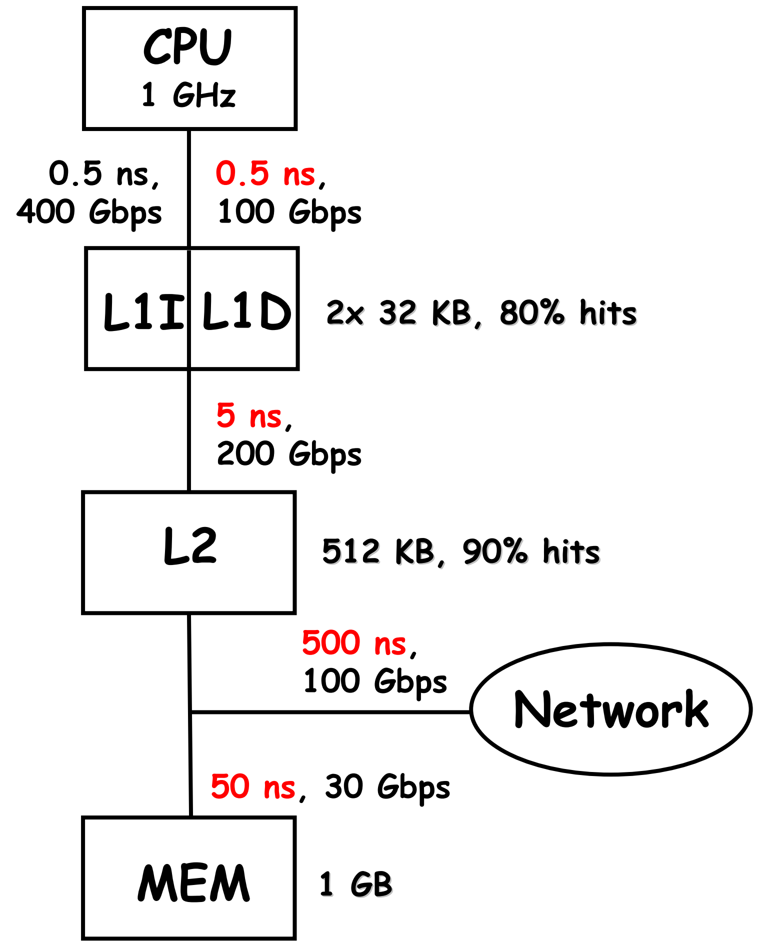
Interconnect requirements

- Network latency is a major bottleneck:

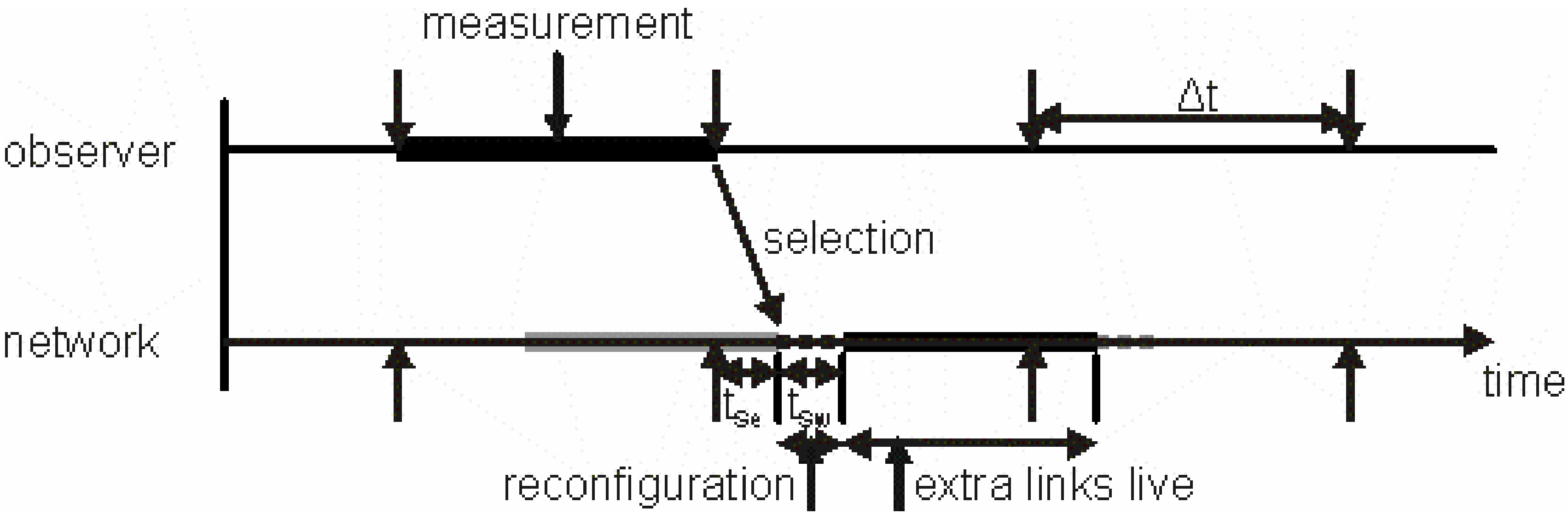
  instruction (.5 ns)

  << local memory access (50 ns)

  << remote memory access (500 ns)



Reconfiguration in shared-memory machines



- Requirement:

  Selection and switching times << reconfiguration interval << traffic locality