Modeling and Analysis to Estimate the End-System Performance Bottleneck for High-Speed Data Transfer

Amitabha Banerjee,
Biswanath Mukherjee,
and Dipak Ghosal
Email: abanerjee@ucdavis.edu

PFLDNet, Marina Del Rey, CA, February 2007.

Outline

- Motivation
- Modeling the End-System
- Analytical & Experimental Results
- Conclusions & Future Work

Key Development in Network Technologies

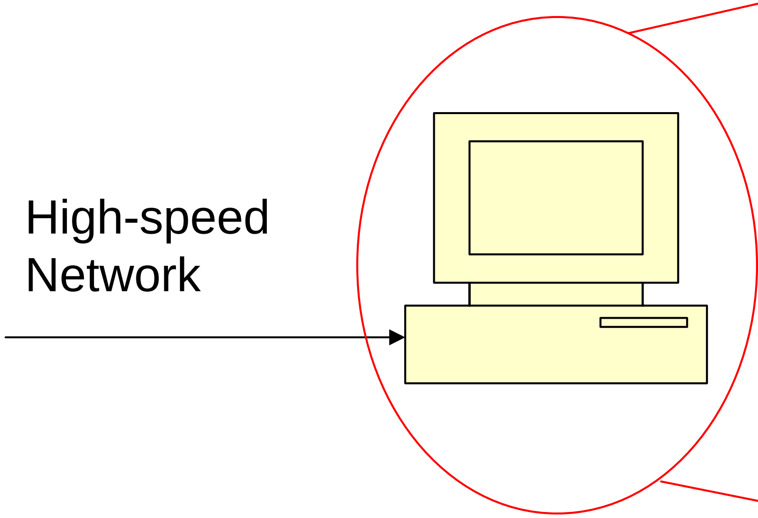
- *Backbone*:
  - Lambda-Grids: Up to 10 Gbps (OC-192) circuits.
    e.g., National Lambda Rail, DoE UltraScienceNet
- *Access:*
  - Passive Optical Networks: 1/10 Gig EPON.
- *Adapters:*
  - 1/10 Gig Network Adapters.
  - Standardization of 100 Gig Ethernet. (IEEE study group)
- *With these we have the ability to establish*:
  - High-capacity end-to-end connections.
  - End-to-end dedicated circuits.

Limited End-System Capacity

- *Disk Speeds*:
  - SATA: 2.4 Gbps (3.0 Gbps reduced by 8/10 coding.)
- *Bus Speeds*:
  - 133 MHz 64-bit PCI-X: 8.5 Gbps
  - PCI-E is much faster (8 GBps)
- *Memory/ Cache contentions*.
- *Overloaded CPU* (Particularly in single processor environments)

End-system not keeping pace with the network

End-System Bottleneck



Operations on Received Packet*

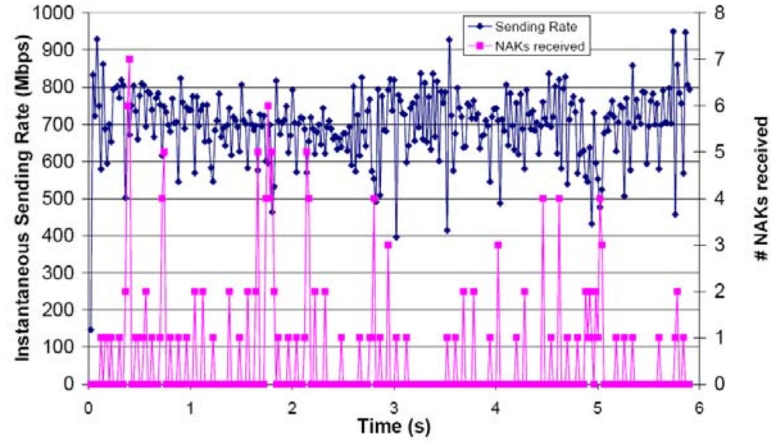
1. DMA'ed to Memory.
2. Processed in Network Protocol Stack.
3. Copied from Kernel-space to User-space, Delivered to Socket Buffer.
4. Read by Application, Processed, Written to Disk, etc.

* Assuming no Zero-copy, RDMA, Offload Engine Optimizations

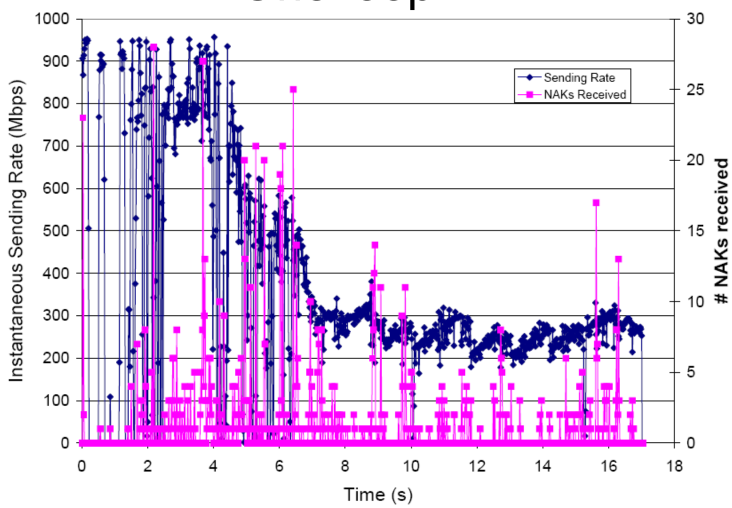
Experiments with UDT

Pentium IV 3 GHz, 2 GB RAM
1 Gig E network, 1500 MTU
RAMDisk-RAMDisk Transfer

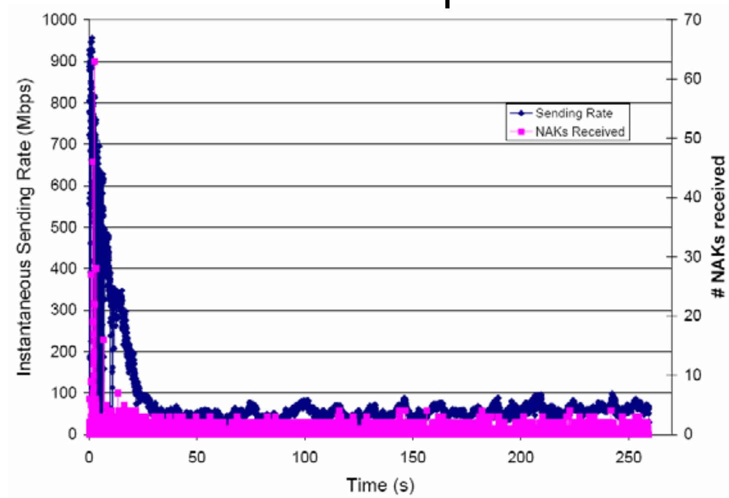
Idle System



One loop



Two loops



Throughput decreases as CPU becomes overloaded with computational load.

Review of Flow Control Mechanisms

- TCP
  - Receiver Advertises Empty Socket Buffer (Flow Window).
  - Sender limits Un-Acked packets to Flow Window.
- LambdaStream
  - Measures packet inter-arrival time. Compares with sending inter-arrival time.
  - Sends feedback whether to increase/ reduce sending rate.

Limitations of Existing Flow Control Mechanisms

- Operates only at Socket – Application Interface.
  - OS and NIC semantics not captured.
- Bursty and transient metrics.
  - Application reads data in bursts.
- When RTT is high, information is stale for sender, particularly when it is very transient.

Our Goal

- Achieve End-System Performance Aware Flow Control
  - Model all possible bottlenecks at end-system.
  - Estimate best data transfer rate considering entire end-system performance.
  - This rate, the *effective bottleneck rate,* is derived as function of current workload.
  - Match sending rate to *effective bottleneck rate.*
- Merits:
  - Workload: Less transient => More reliable data.
  - Rate Matching across all end-system components.

Outline

- Motivation
- End-System Model
- Analytical & Experimental Results
- Conclusions & Future Work

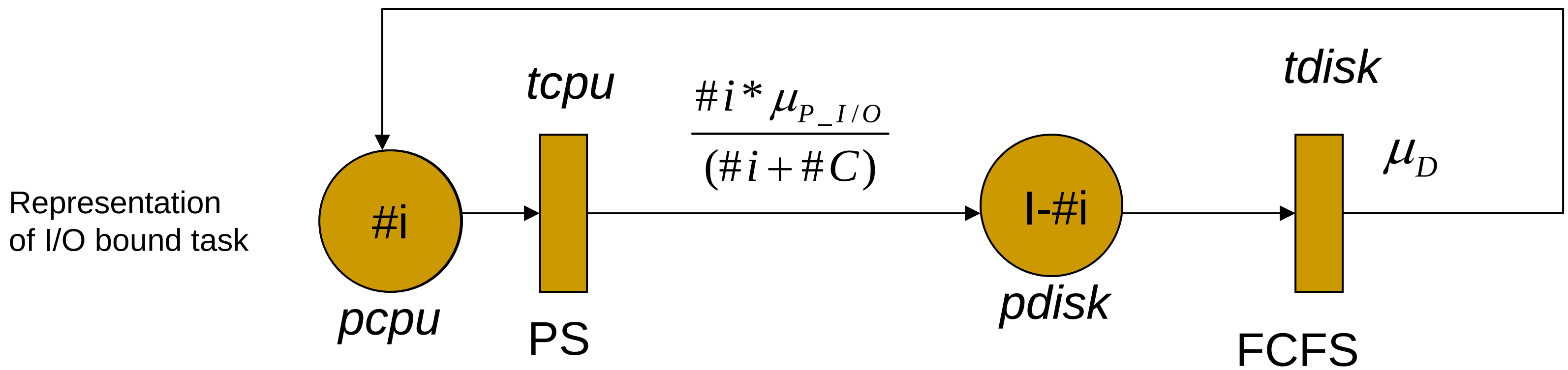
Markov Models

- Markov Models: Stochastic Analysis of System Performance
- Tools to create Markov Models:
  - Petri Nets, introduced in 1962.
  - Stochastic Petri Nets (SPN),
  - Stochastic Reward Nets (SRN)
- Allows for Automatic Generation of Markov chains from any of the above models.
- Tools: SPNP, SHARPE, MOSEL 2, etc.

Categorize Tasks

- CPU-bound tasks
  - Uses CPU cycles constantly.
- I/O-bound tasks
  - Uses CPU and I/O alternately.
- Network tasks
  - Requires processing of ISRs.

SRN Model of End-System (Memory-to-Memory Data Transfer)



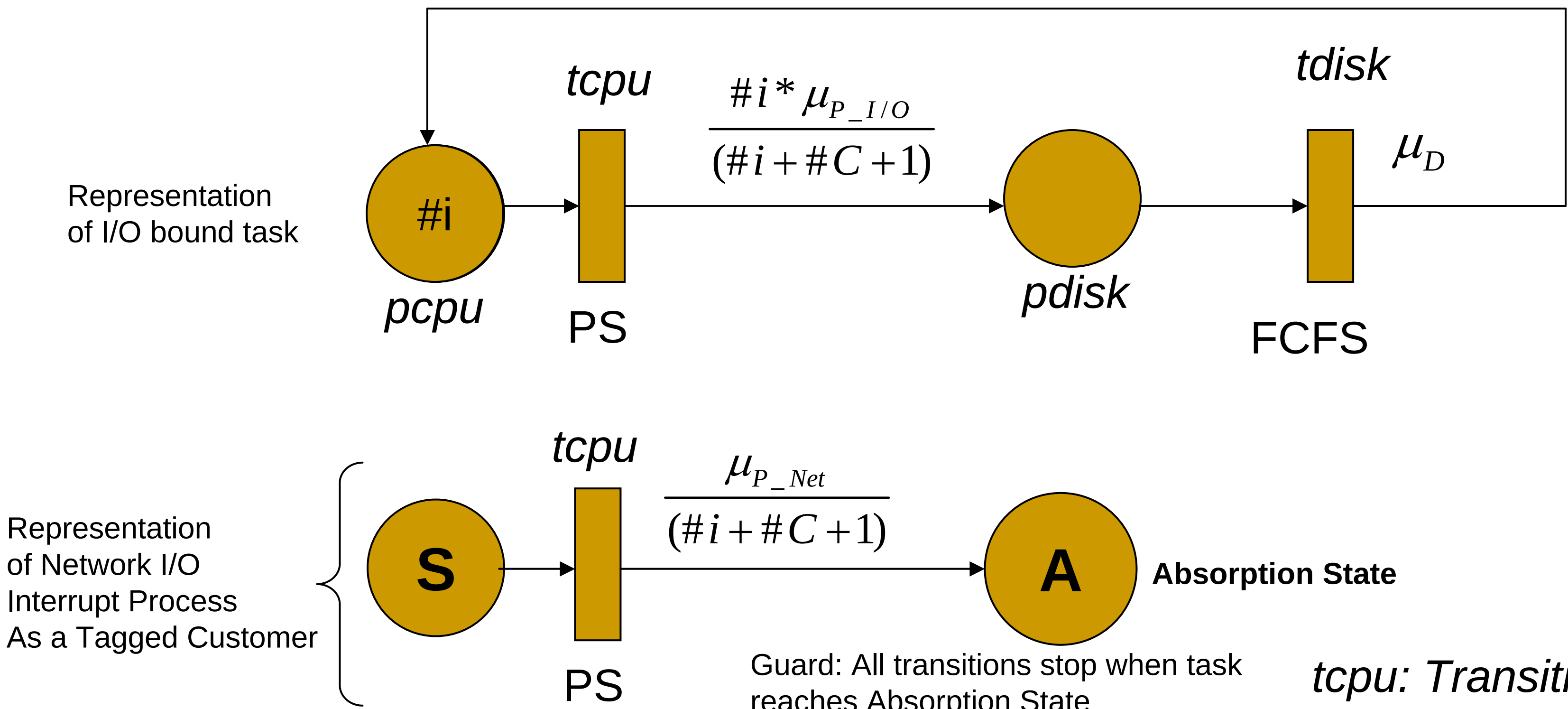
tcpu: Transition for CPU processing
tdisk: Transition for Disk processing

#C: Number of CPU-bound tasks
#I: Number of I/O-bound tasks

Steady State Analysis => Probability of I/O Task Distribution

#C	#i	I - #i	$P_{SteadyState}$
Workload			?

SRN Model of End-System (Memory-to-Memory Data Transfer)



tcpu: Transition for CPU processing
tdisk: Transition for Disk processing

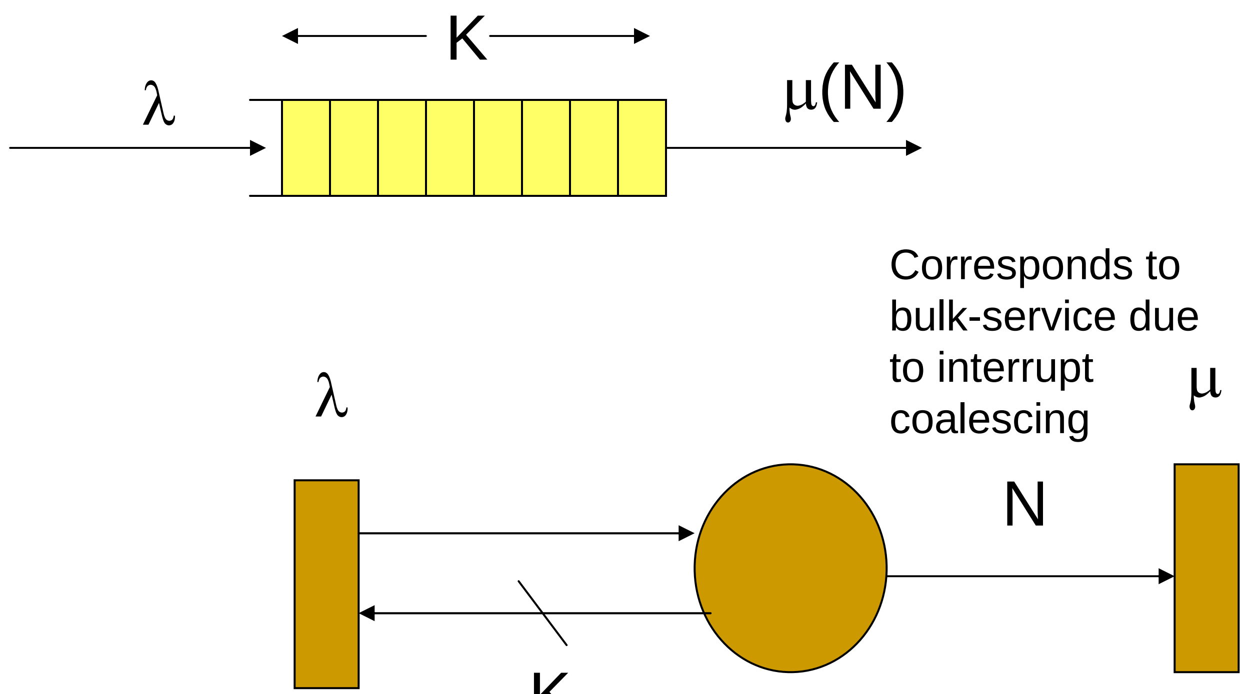
#C=Number of CPU-bound tasks

Transient Analysis of SRN Model

- Yields Response-Time Distribution from states ‘S’ to ‘A’ as function of Workload
- Derive Expected Rate of ISR service

#C	#I	μ_{ISR}
Workload		?

SPN Model of NIC



SPN model is employed to determine packet loss as function of λ

Estimation of *Effective Bottleneck Rate*

$$T = [T_{prop} + \frac{S_{total}}{B_{send}}]$$

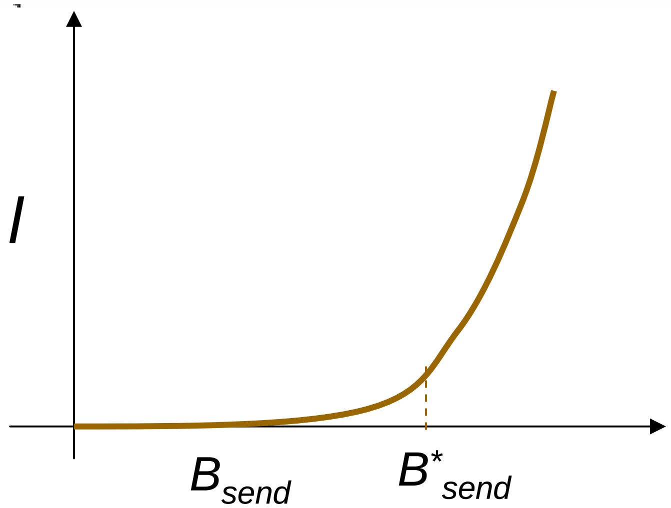
Time required to transmit first burst

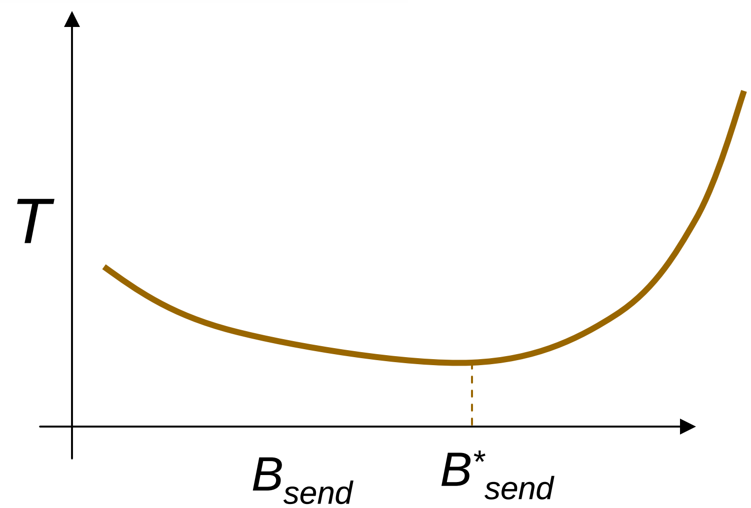
$$+[(N_{resend} * T_{prop}) + \sum_{i=1}^{\lfloor N_{resend} \rfloor} \frac{l * S_{send_{i-1}}}{B_{send}}]$$

Time to send subsequent bursts

$$+[(N_{resend} + 1) * (\frac{S_{total}}{8 * S_{pkt} * B_{send}} + T_{prop}))]$$

Time to send error sequence numbers





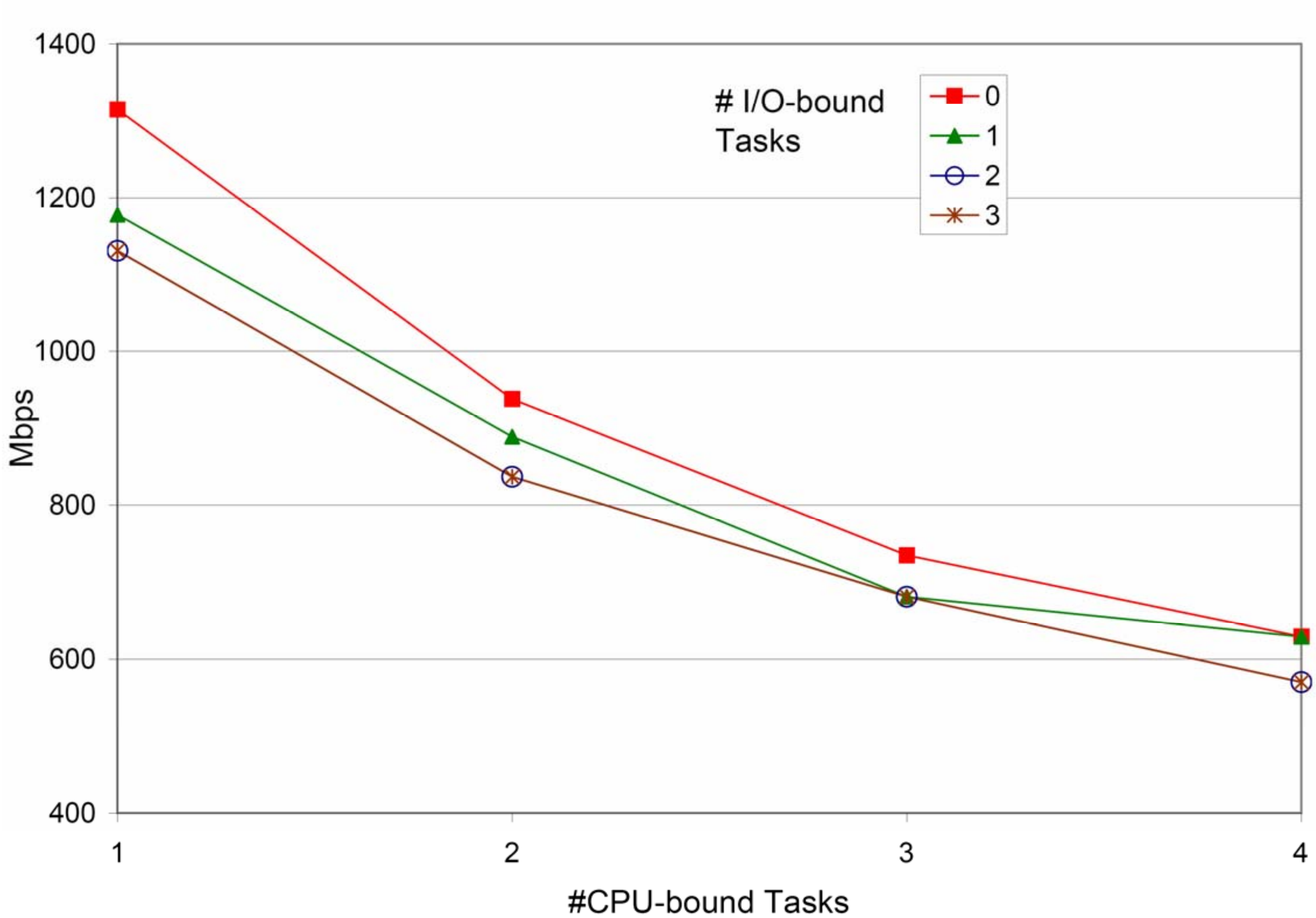
How to Determine Model Parameters

- Representative Workloads
  - I/O-bound Task: Task reading random line from file.
  - Network I/O Task: Task reading data from network.
- Use MAGNET to trace above task
  - Determine service time distributions at CPU & disk
  - Determine Expected Service Rates from these distributions
- Approximations
  - Capture high-level stochastic metrics.
  - Leave out OS & Task specific implementation details.
  - Simple model which can be easily developed and analyzed in software.

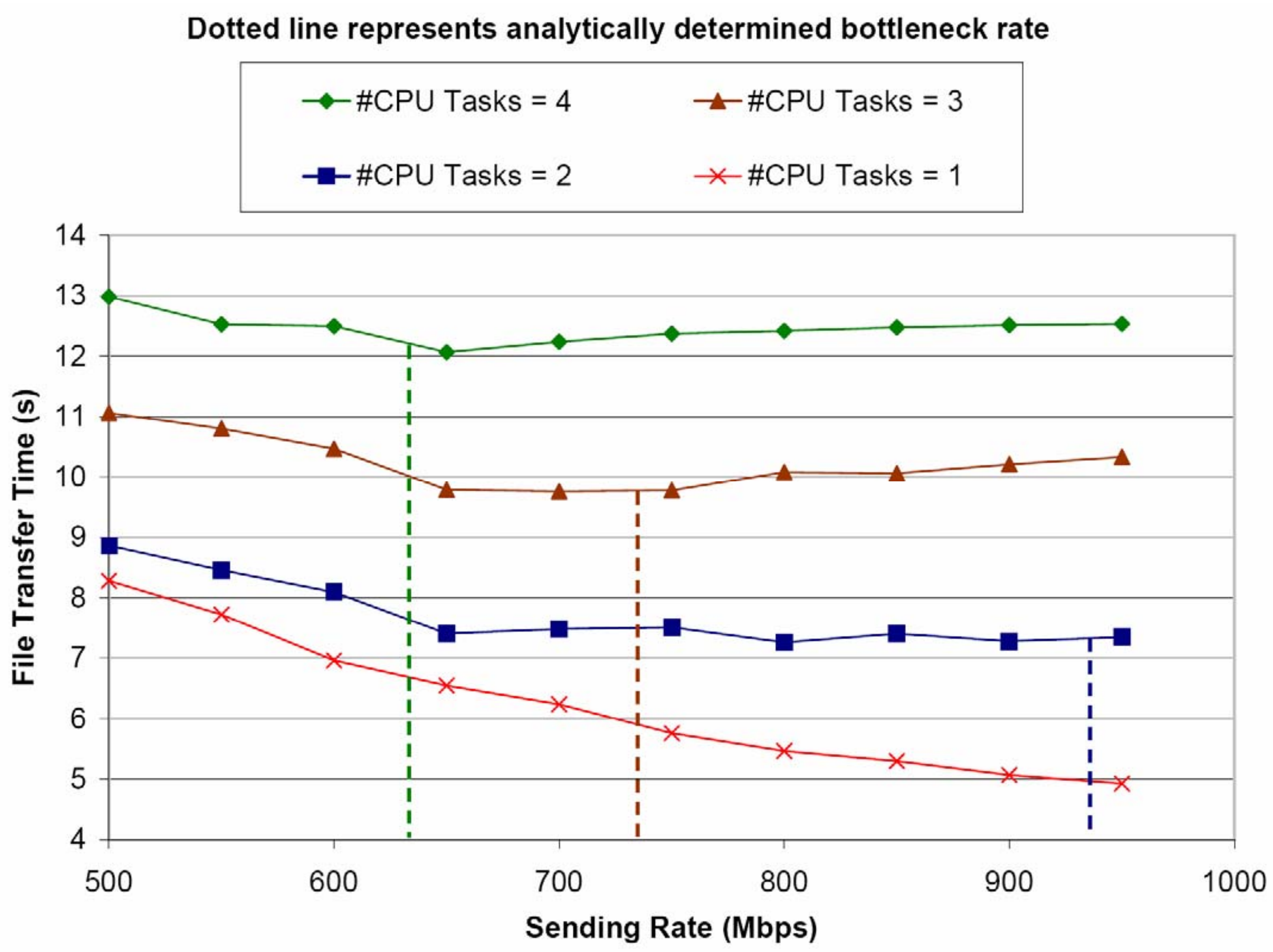
Outline

- Motivation
- End-System Model
- Analytical & Experimental Results
- Conclusions & Future Work

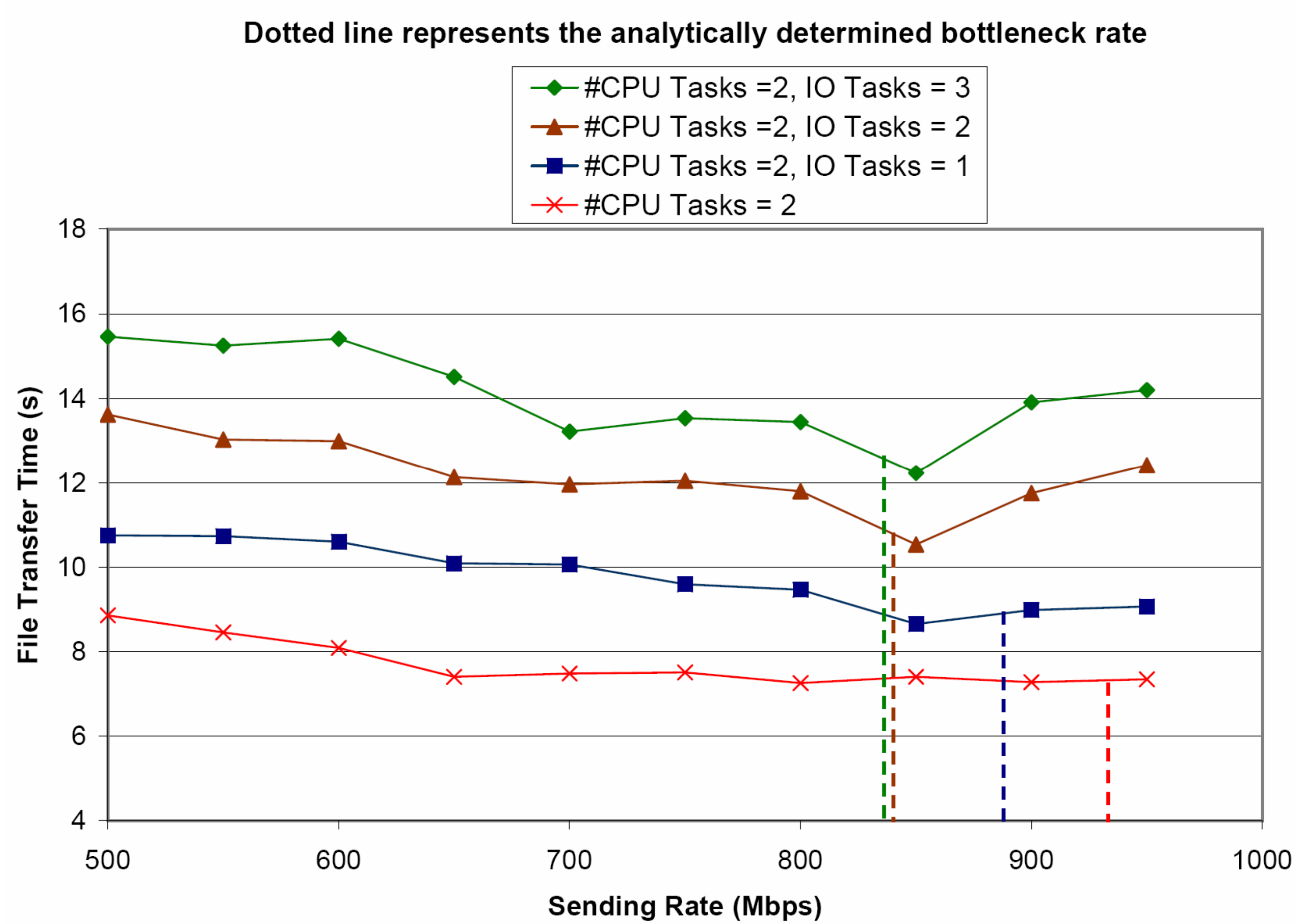
Analytical Results



Experimental Results – CPU-bound tasks



Experimental Results – I/O-bound tasks



Outline

- Motivation
- End-System Model
- Analytical & Experimental Results
- Conclusions & Future Work

Discussion

- Proposed an approach to achieve
  End-System Performance Aware Flow Control
- Illustrated model for memory-to-memory data transfer. Similar models possible for other scenarios.
- Demonstrated that Analytical model yields *effective bottleneck rate* as function of workload.

Challenges

- How to implement in software?
  - Analytical model parameters to be determined only ONCE. Therefore, measure statically (At time of software installation).
  - Construct SRN model at runtime.
  - Workload determined at time of data transfer.
    - Determine tasks, classify them CPU-bound & I/O-bound.
    - Monitor changes in workload.
  - Deliver feedback on *effective bottleneck rate.* (TCP)
  - Match sending rate to receiver bottleneck. (Pacing)

Questions and Comments ?

Thank You

Contact Info: abanerjee@ucdavis.edu