Investigating Efficient Real-time Performance Guarantees on Storage Networks

Andrew Shewmaker
shewa@soe.ucsc.edu

Department of Computer Science
University of California Santa Cruz

October 20, 2009

Motivation

Goals of datacenters

- serve many users
- process petabytes of data

Design of datacenters

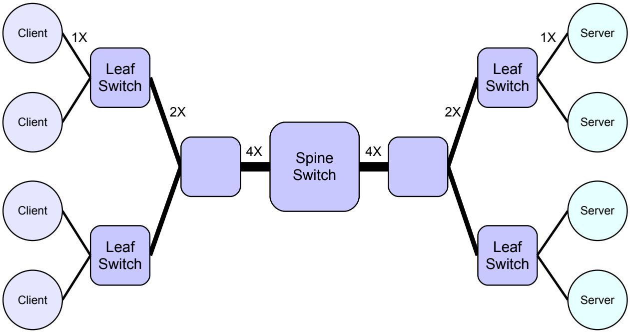
- use rules of thumb
- over-provision

An ad hoc approach creates marginal storage systems that cost more than necessary. A better system would be able to guarantee each user the performance they need from the CPUs, memory, disks, and network.

1. Introduction
2. RAD on Networks (Radon)
3. Evaluation of Radon
4. Conclusion

A Canonical Storage Network

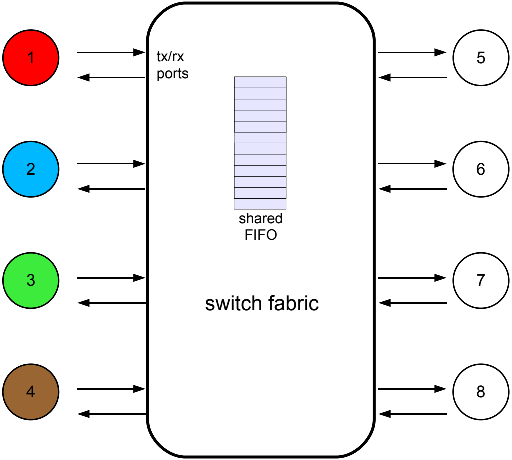
Fat-tree with full bisection bandwidth
trunk capacity matches the sum of the outer branches



This research investigates standard Gigabit Ethernet

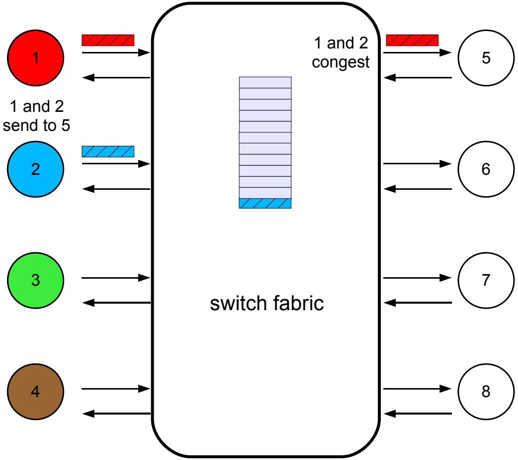
Congestion in a simple switch model

Each transmit port on the switch is a collision domain



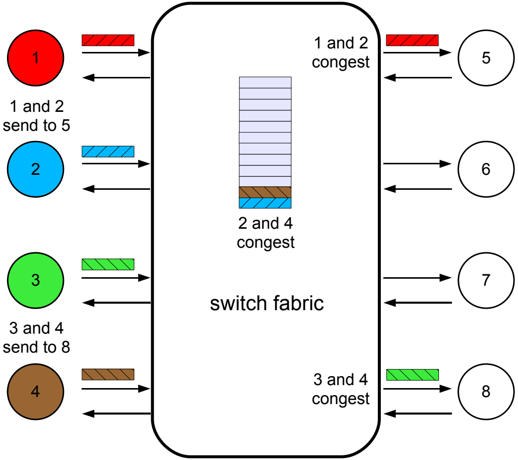
Congestion in a simple switch model

One of the packets destined for the same switch transmit port is delayed on the queue

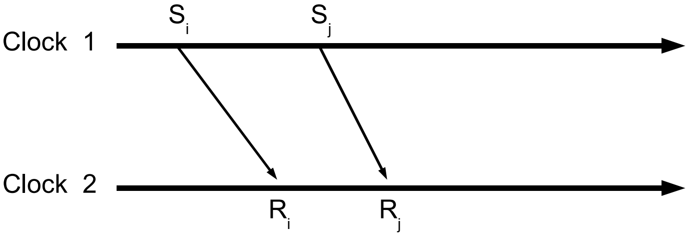


Congestion in a simple switch model

Delayed packets from unrelated streams affect each other on the queue



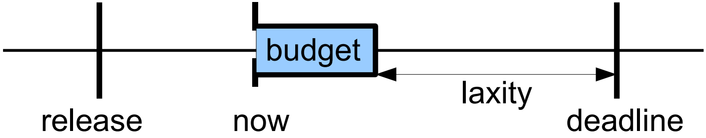
Network Resource Measurements



Relative Forward Delay $RFD_{i,j} = (R_j - R_i) - (S_j - S_i)$

While the clocks requires no synchronization, they should be stable and not reset between timestamps

Real-time Information



- Deadline is absolute
- Laxity is relative
- Budget gives global information

Rate-based Percent Budget scheduling

Flow Control Budget (in packets) $m_i = e_i / pktS$, where $pktS$ (s/packet) is the worst case packet service time

Congestion Control Adjust wait time between packets

- **Percent Budget** $\%budget = (1 - \%laxity) = \frac{e_i}{d-t}$
- **Packet Wait Time Target** $w_{op} = \frac{w_{min}}{\%budget}$
- **New Wait Time** $w_{k+1} = \min\left(w_{max},\ \max\left(w_{min},\ w_k - \frac{w_k - w_{op}}{2}\right)\right)$

Jump to window-based Radon

Radon Userspace Proof of Concept

Detection of Congestion and its Severity

- Relative Forward Delay
- Five element median filter
- TCP Santa Cruz queue model

Response to Congestion

- Network time reservation
- Inter-packet wait time varied according to $\%budget$

Experimental Setup

- Seven cluster nodes with Gigabit Ethernet
- Gigabit switch capable of Jumbo Frames
- Modified UDPmon network analysis tool
- Compare constant rate and adaptive streams

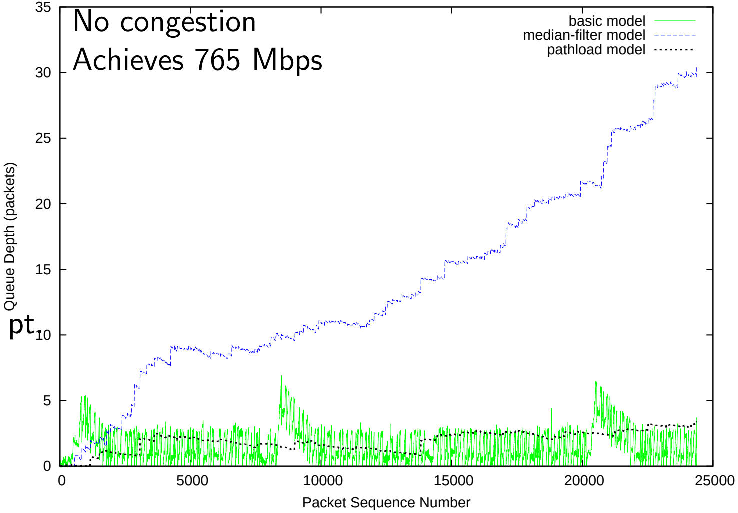
Single max rate baseline with no congestion

Punctuated primary stream interrupted by five short streams

Fairshare six equal streams

Unfair concurrent unequal streams

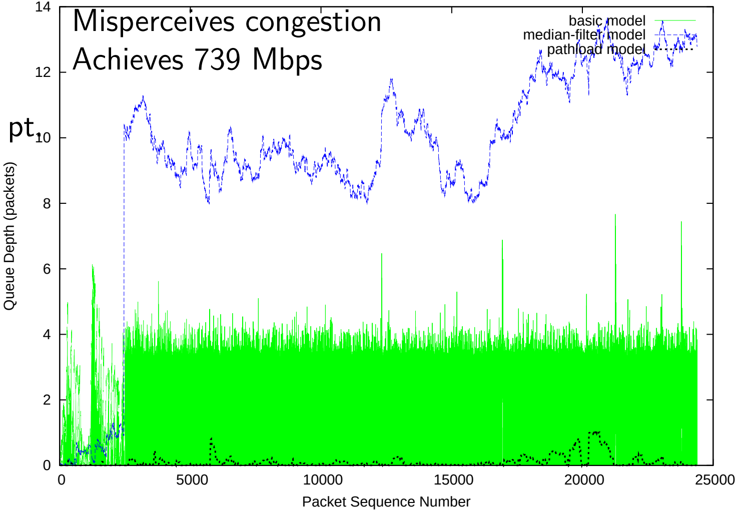
Queue model for a single network stream



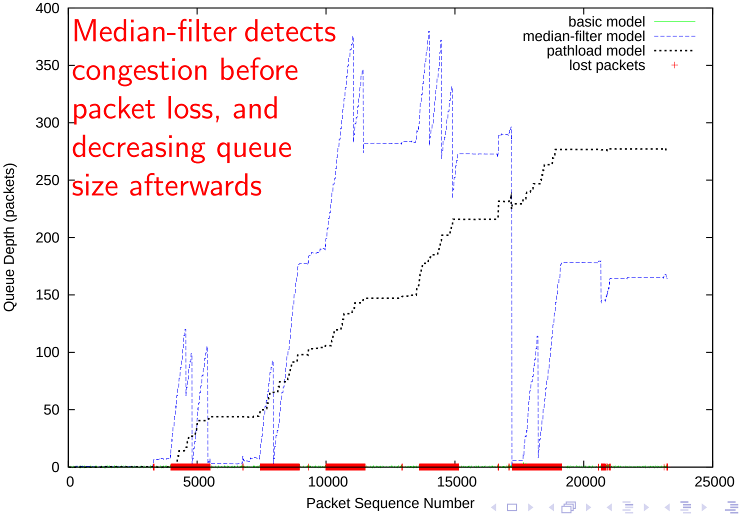
operating pt.

Queue model for a single adaptive network stream

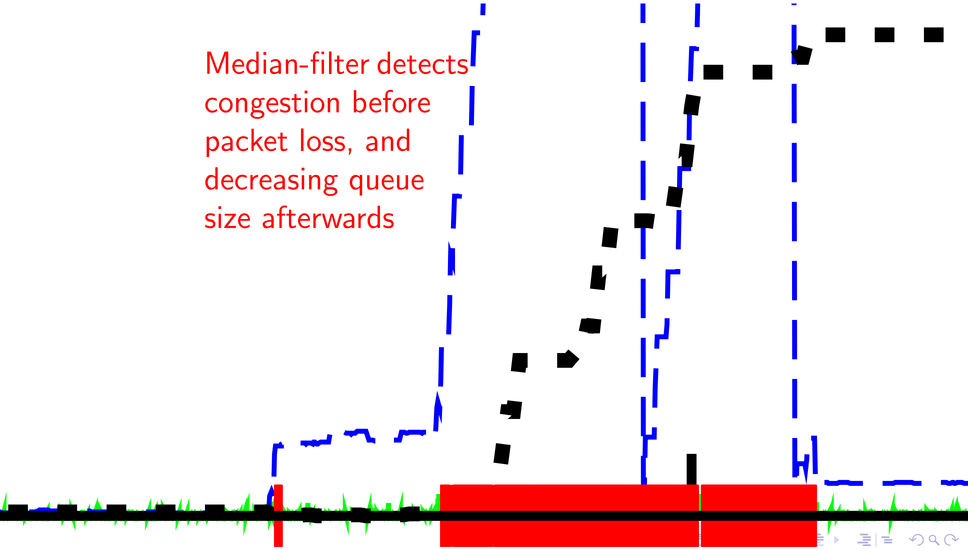
operating pt.



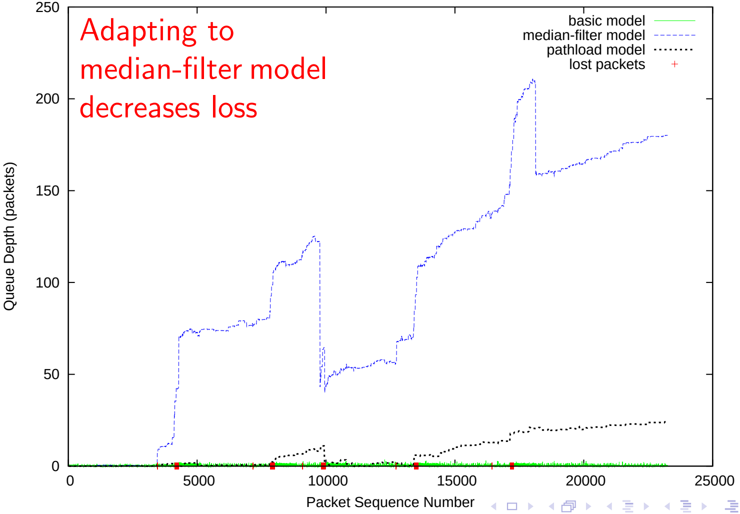
Queue model for a punctuated stream



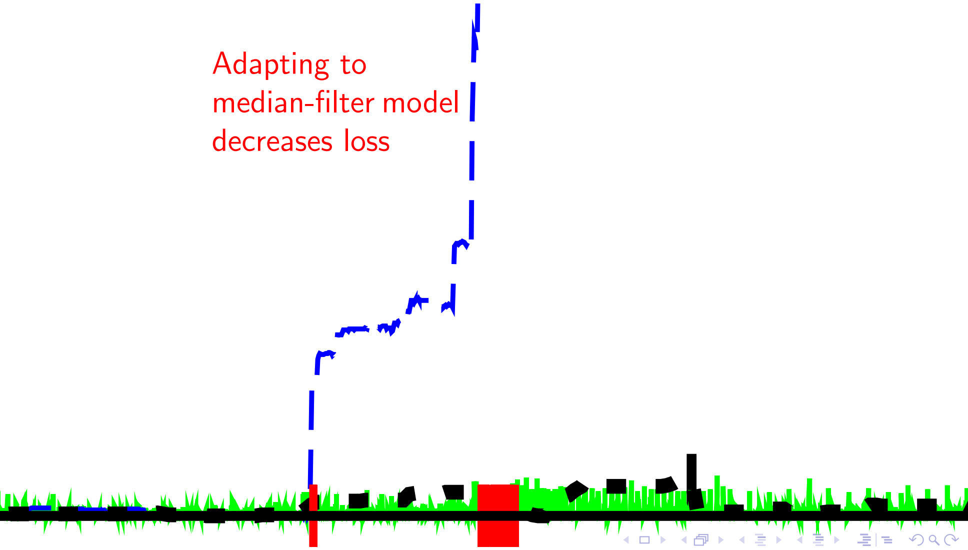
Queue model for a punctuated stream



Queue model for a punctuated adaptive stream



Queue model for a punctuated adaptive stream



Effectiveness of Radon for a punctuated stream

Greater goodput for primary stream

Stream ID	target rate (Mbps)	%lost packets constant	%lost packets adaptive	recv rate (Mbps) constant	recv rate (Mbps) adaptive
1	749	24.0	2.5	565.5	725.2
2	251	3.8	0.2	245.5	1.5
3	251	4.4	0.2	244.2	1.5
4	251	4.6	0.2	244.2	1.5
5	251	4.4	0.2	240.8	1.5
6	251	3.8	0.2	238.9	1.5

All had period of 1 s, but 2-6 consisted of 500 packets

Jump to queue graphs

Effectiveness of Radon for six fairshare streams

Greater aggregate goodput and fairer

Stream ID	target rate (Mbps)	%lost packets constant	%lost packets adaptive	recv rate (Mbps) constant	recv rate (Mbps) adaptive
1	166	3.00	0.81	158.40	161.84
2	166	0.39	0.35	162.57	162.71
3	166	0.10	0.04	163.08	163.02
4	166	0.06	0.08	163.16	163.06
5	166	0.08	0.06	163.06	163.14
6	166	0.04	0.06	163.13	163.02

All had period of 1 s

Jump to graphs

Effectiveness of Radon for six unfair streams

Greater goodput, but unable to deliver > 80%

Stream ID	target rate (Mbps)	%lost packets constant	%lost packets adaptive	recv rate (Mbps) constant	recv rate (Mbps) adaptive
1	500.00	38.0	35.0	305.62	318.66
2	250.00	2.3	2.0	239.65	240.28
3	125.00	0.1	0.0	122.68	122.82
4	62.50	0.1	0.0	61.36	61.40
5	31.25	0.0	0.0	30.73	30.73
6	31.25	0.2	0.0	30.64	30.71

All had period of 1 s

Jump to graphs

Conclusion

The userspace prototype of Radon:

- Detects congestion using Relative Forward Delay
- Responds to congestion using RAD real-time theory
- Prevents packet loss to some degree
- Improves goodput
- And does not require global knowledge or synchronization

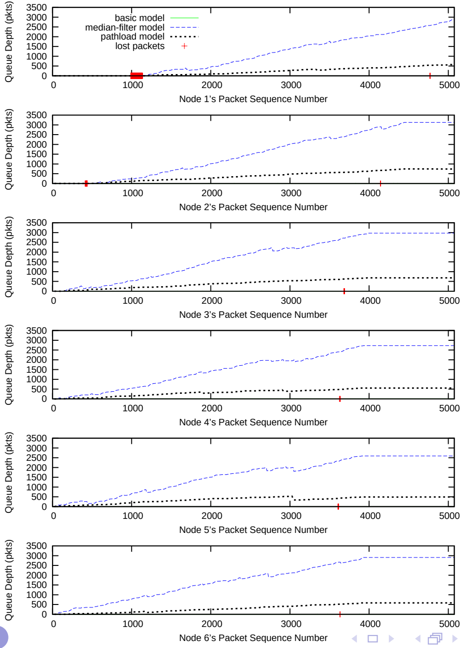
Future Work

- Implement kernel qdisc of window-based Radon
- Compare to global scheduler
- Evaluate using 10 Gigabit Ethernet and Infiniband
- Analyze interaction with TCP
- Combine with other RAD-based resource schedulers

5 Appendix

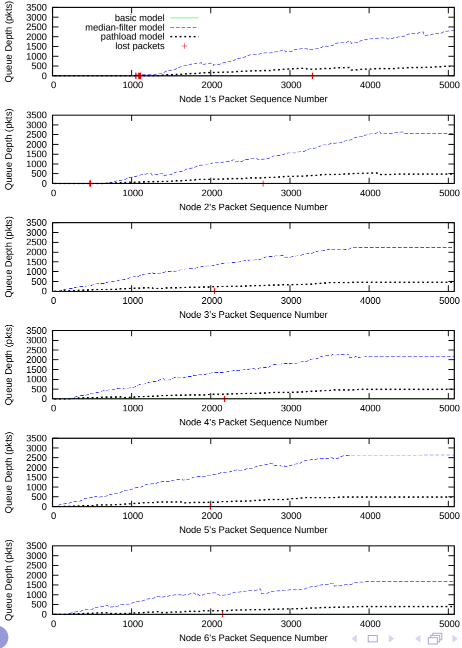
- Experiment 3 Graphs
- Experiment 4 Graphs
- Window-based Radon
- Related Work

Queue model for six fairshare streams



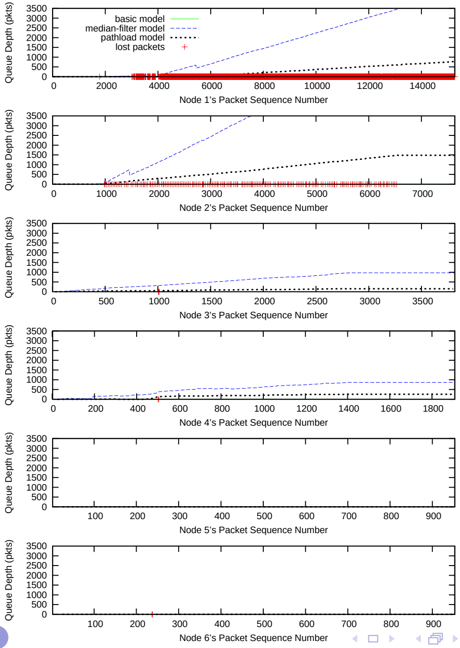
▸ Jump to experiment

Queue model for six fairshare adaptive streams



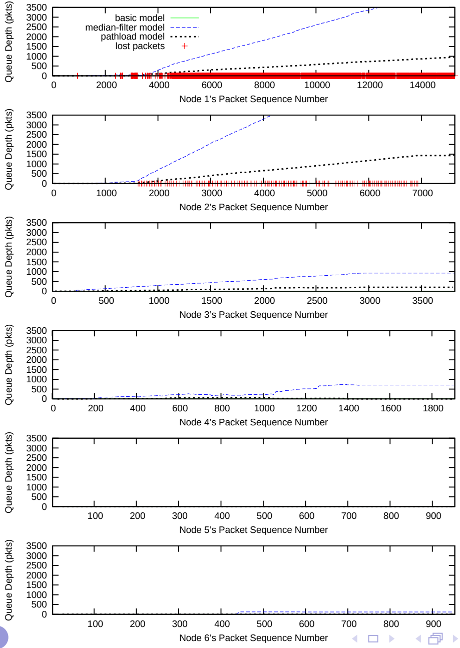
Queue model for six unfair streams

The X axis shows the streams send a different number of packets over the same two second interval



Jump to experiment

Queue model for six unfair adaptive streams



Jump to experiment

Window-based Percent Budget scheduling

Flow Control Budget (in packets) $m_i = e_i / pktS$, where $pktS$ (s/packet) is the worst case packet service time

Congestion Control Adjust window size and offset

- **Percent Budget** $\%budget = (1 - \%laxity) = \frac{e_i}{d-t}$
- **Window Target** $w_{op} = (1 - \%laxity) \cdot w_{max}$
- **Size Change** $w_\Delta = \frac{-|w_k - w_{op}|}{2}$
- **Dispatch Offset** $w_{offset} = \frac{N_{obs}}{pktS} \cdot \text{rand}$

Where w_k is the current window size and N_{obs} is the depth of the bottleneck switch's queue modeled using observations of relative forward delay.

Jump to rate-based Radon

Less Laxity More scheduling

Flow Control Budget (in packets) $m_i = e_i/pktS$, where $pktS$ (s/packet) is the worst case packet service time

Congestion Control Windows adjusted in size and dispatch time

Percent Budget $\%budget = (1 - \%laxity) = \frac{e_i}{d-t}$

Less Laxity More Window Target

$$w_{op} = \min\left(m_i, \max\left(\frac{w_{max}}{l_{i,j}+1}\right), 2\right)$$

Size Change $w_\Delta = \frac{-|w_k - w_{op}|}{2}$

Dispatch Offset $w_{offset} = \frac{N_{obs}}{pktS} \cdot \text{rand}$

Where w_k is the current window size and N_{obs} is the depth of the bottleneck switch's queue modeled using observations of relative forward delay.

Jump to rate-based Radon

Related Work

- Traffic shaping
- FAST TCP
- Probe Control Protocol
- VRE-NET
- Netnice