TCP EX MACHINA: COMPUTER-GENERATED CONGESTION CONTROL

KEITH WINSTEIN AND HARI BALAKRISHNAN

Presented by: Angela Jiang

Network congestion

- Cause: Sources trying to send data faster than the network can process
- Result: QoS deterioration of network
  - Queuing delays
  - Packet losses
  - Congestion collapse

Congestion control

- Prevent congestion collapse
- Allocates network resources
  - Link bandwidth
  - Queue space
- End-to-end or network assisted

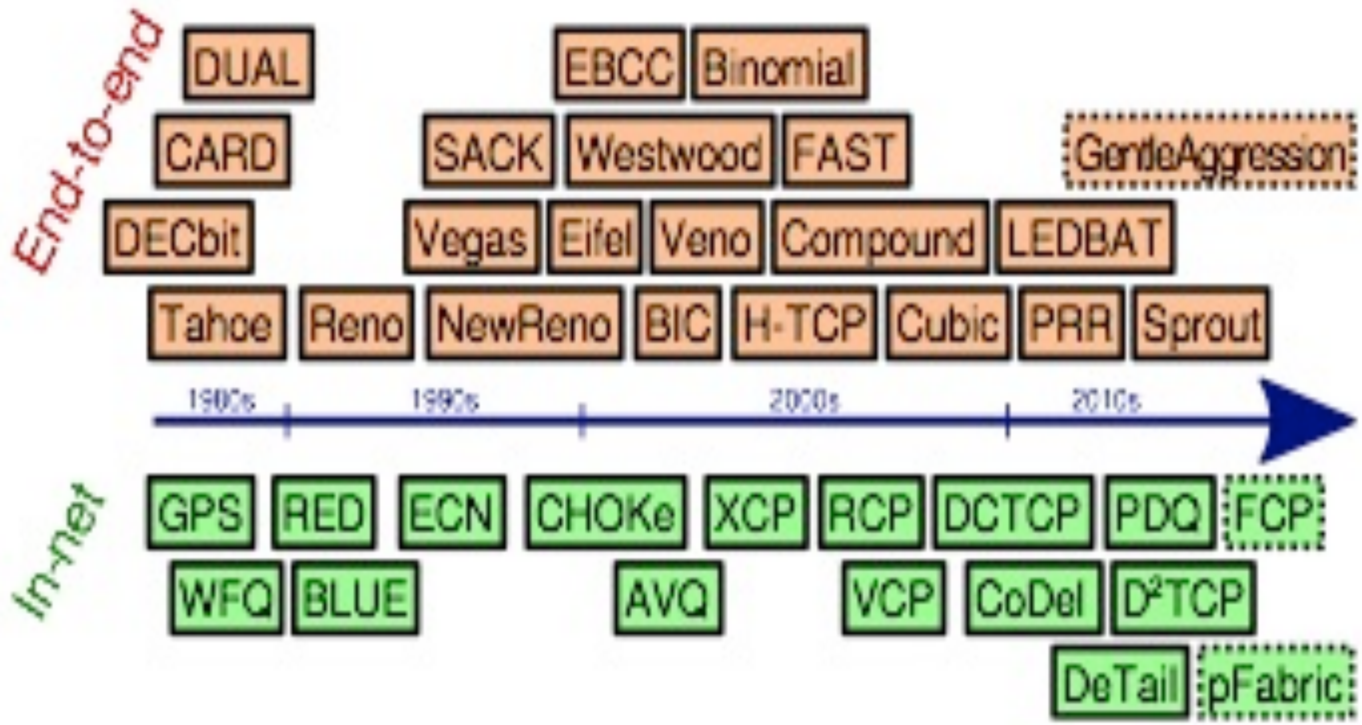
End-to-end congestion control

- No support from network layer
- Hosts must infer presence of congestion
  - Packet loss
  - Queueing delay
  - ECN marks
- Individually control transmission rate

Challenge: evolving networks

- Wireless
- Short connections
- Bursty traffic
- Datacenters

Evolution of congestion control



Challenge: rational choice of scheme

- Different goals?
- Different assumptions about network?
- One scheme just plain better?

Current congestion control

- Inflexible; doesn't allow network evolution
- Unclear what an algorithm is optimized for

Remy

- *Program that generates end-to-end congestion control schemes offline*
- Given
  - Network representation
  - Objective of app (e.g. high throughput)
- Generates
  - RemyCC; a congestion control algorithm

Objective function

- Fairness vs. efficiency

$$U_\alpha(x) = \frac{x^{1-\alpha}}{1-\alpha}$$

- Delay vs. throughput

$$U_\alpha(\mathrm{x}) - \delta \cdot U_\beta(\mathrm{y})$$

- Objectives used

$$U = \log(\mathrm{throughput}) - \delta \cdot \log(\mathrm{delay})$$

$$U = -\frac{1}{\mathrm{throughput}}$$

Prior assumptions of network

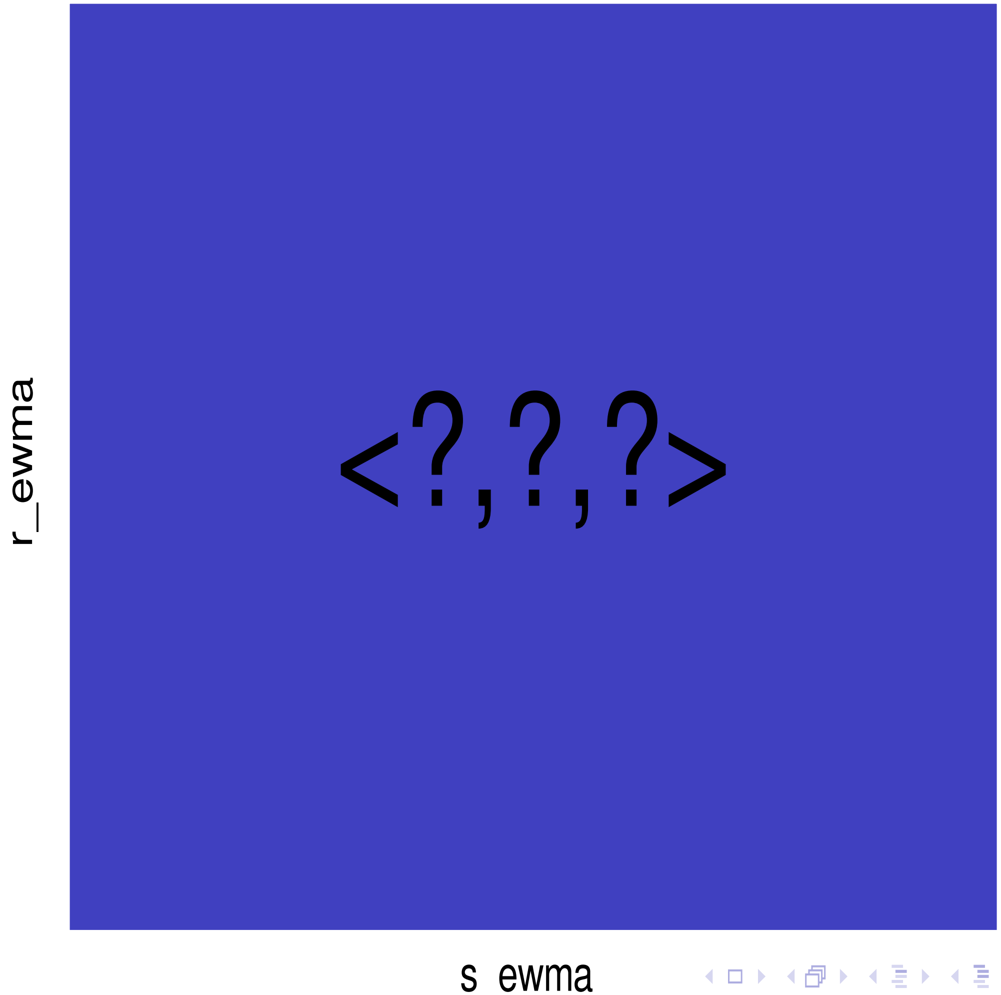
- Model of network uncertainty
  - Link speed distribution
  - Delay distribution
  - Degree of multiplexing
- Traffic model
  - Off-to-on model
  - Web browsing, MapReduce, VoIP

RemyCC maps state to an action

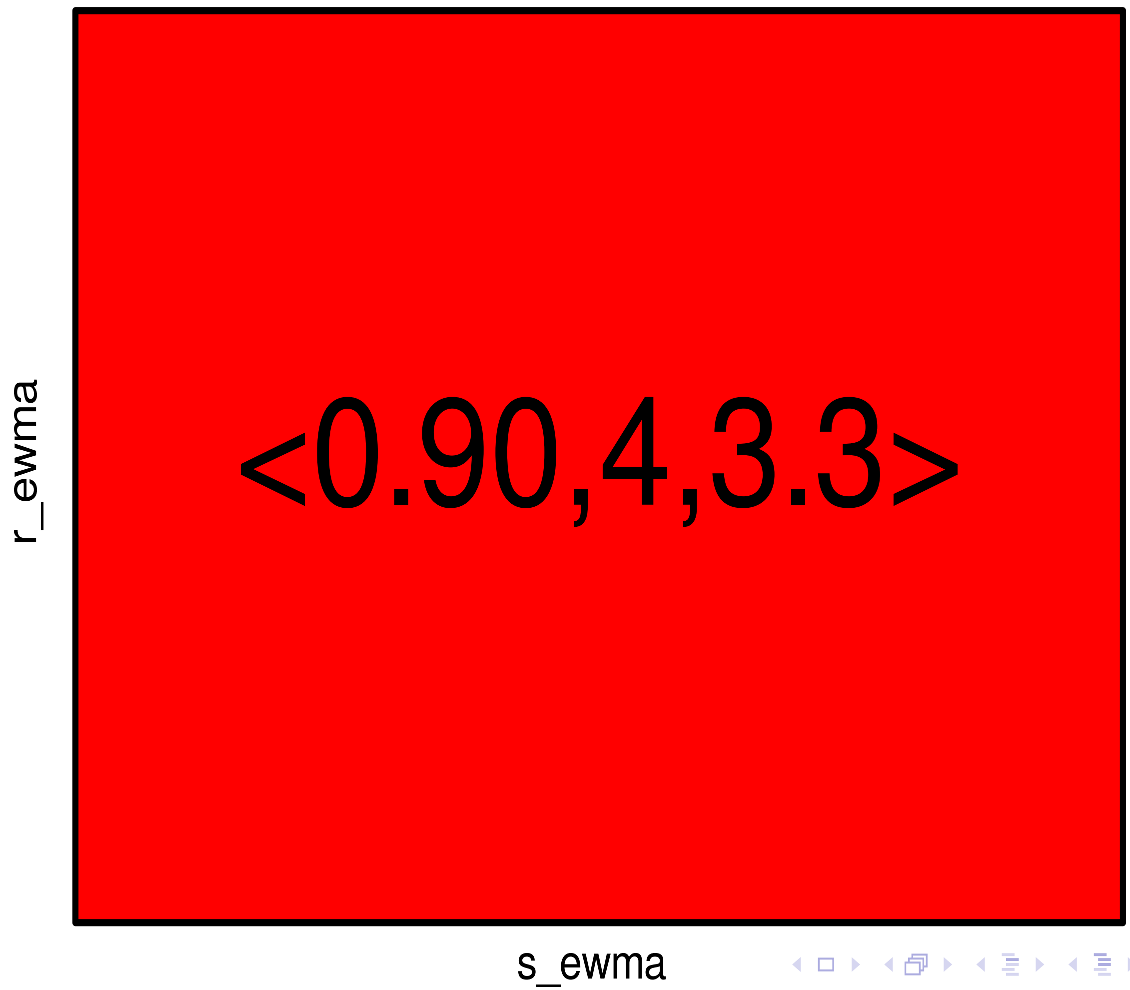
$$\textsc{Rule}(\mathit{r\_ewma}, \mathit{s\_ewma}, \mathit{rtt\_ratio}) \rightarrow \langle m, b, \tau \rangle$$

m	Multiple to congestion window
b	Increment to congestion window
τ	Minimum interval between two outgoing packets

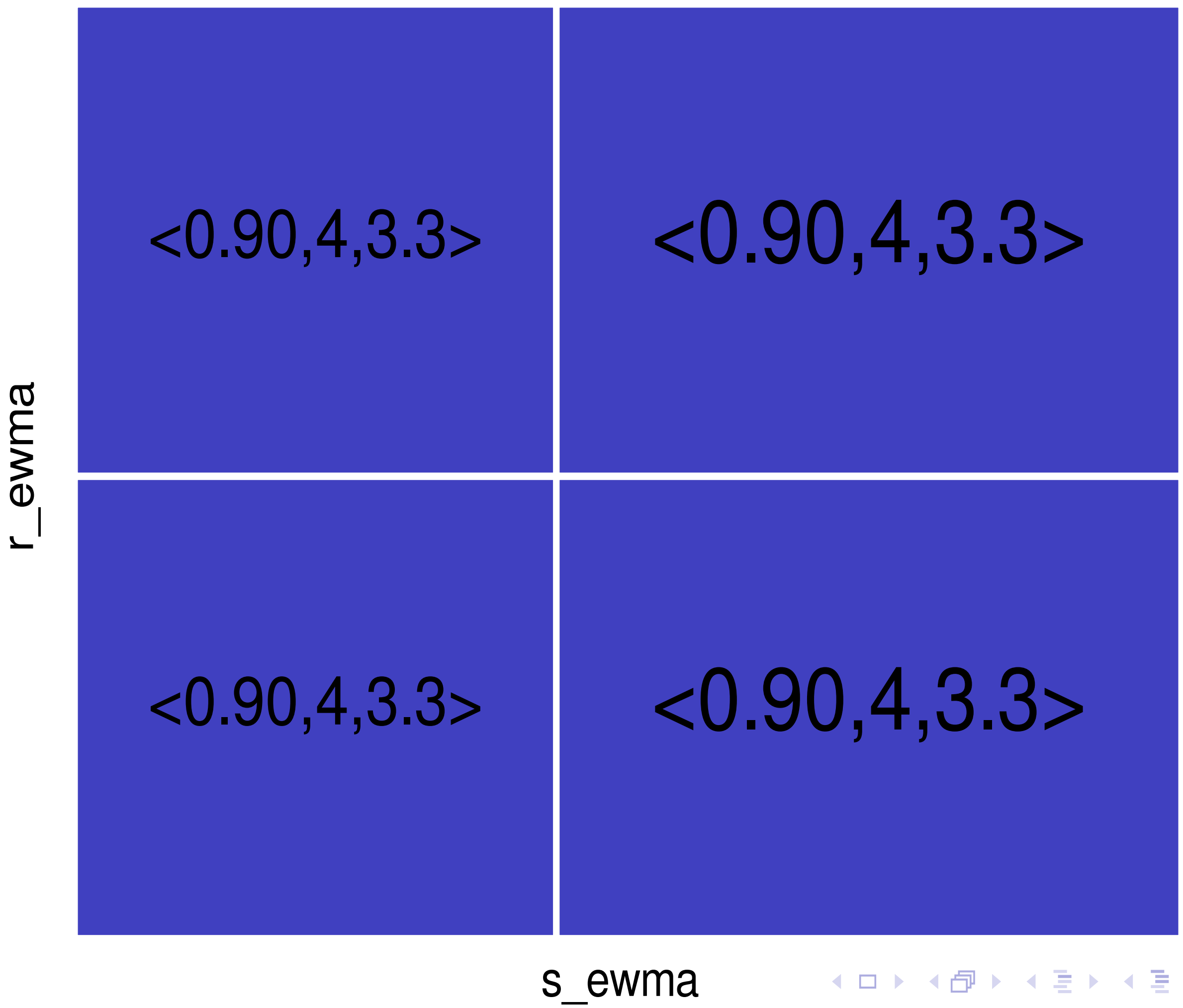
Find the best value



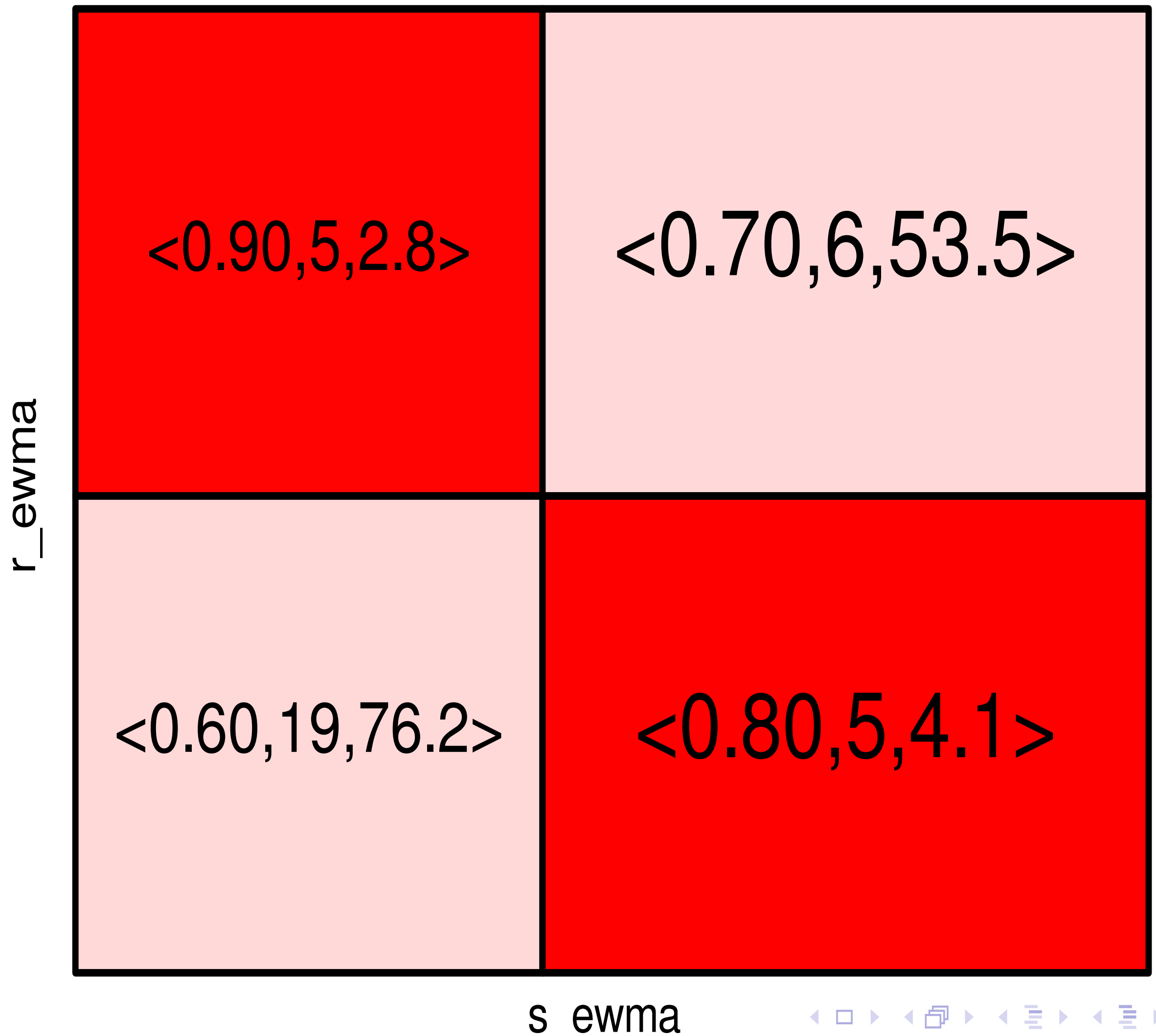
Best single action



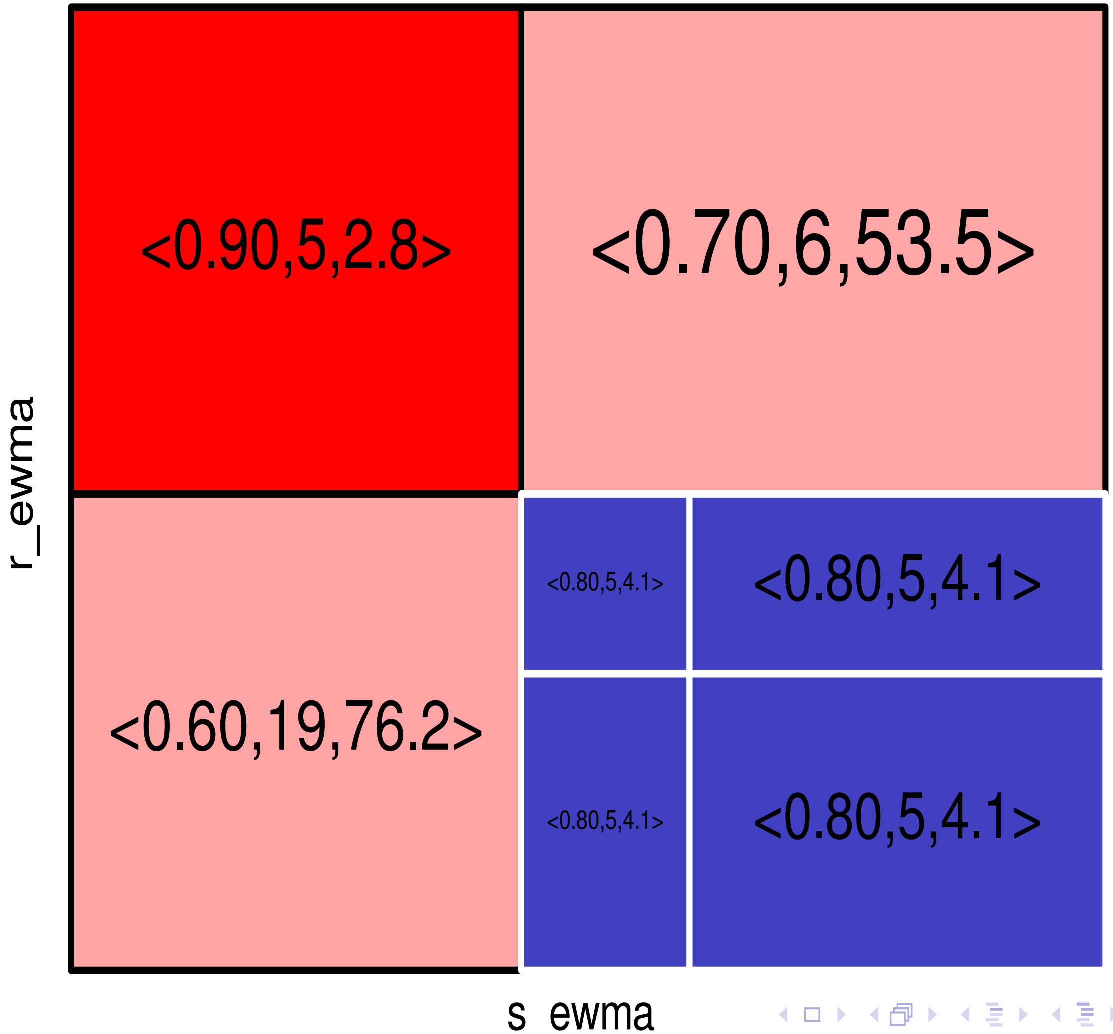
Subdivide most used rule



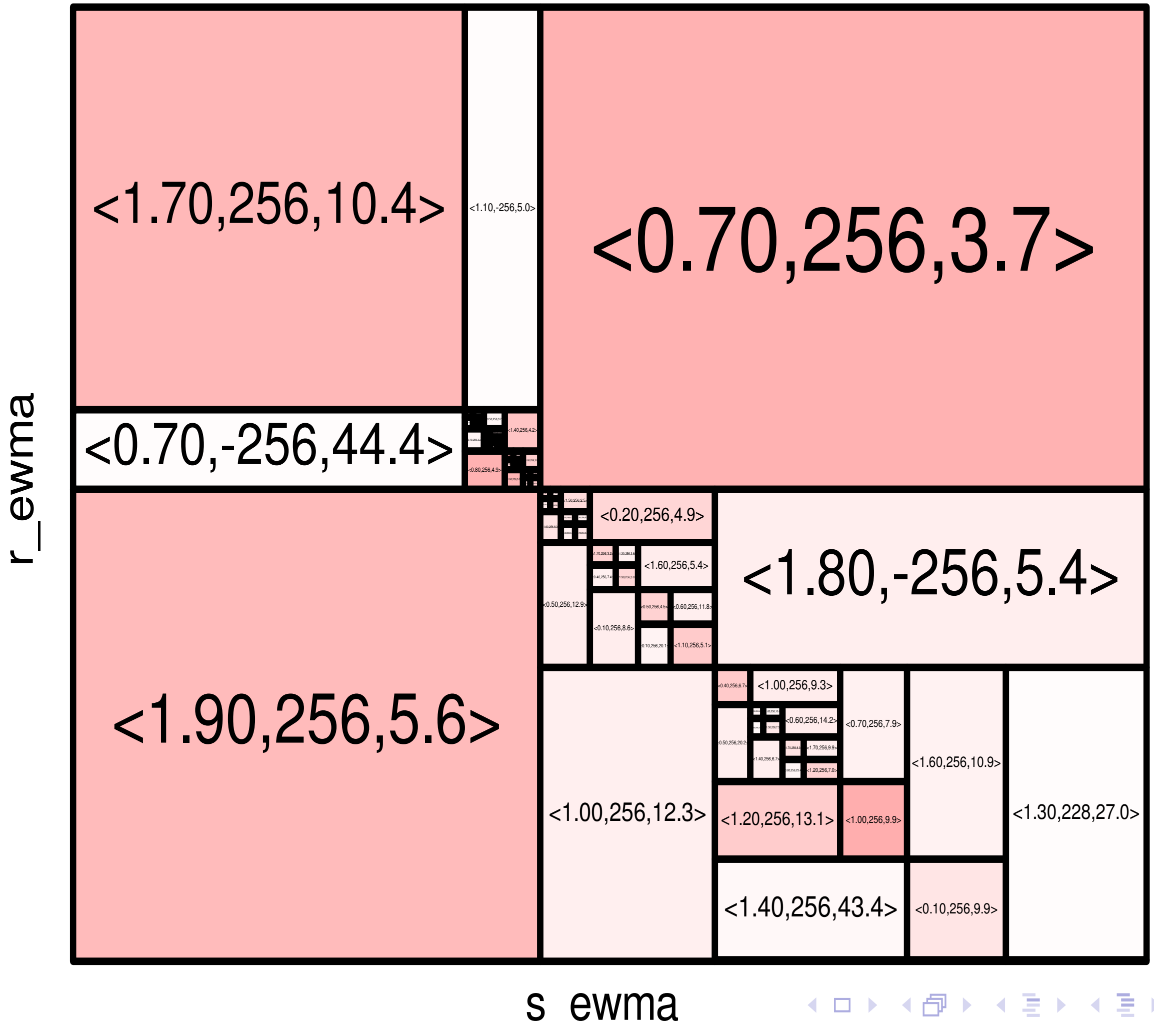
Optimize each new action



Split most used rule



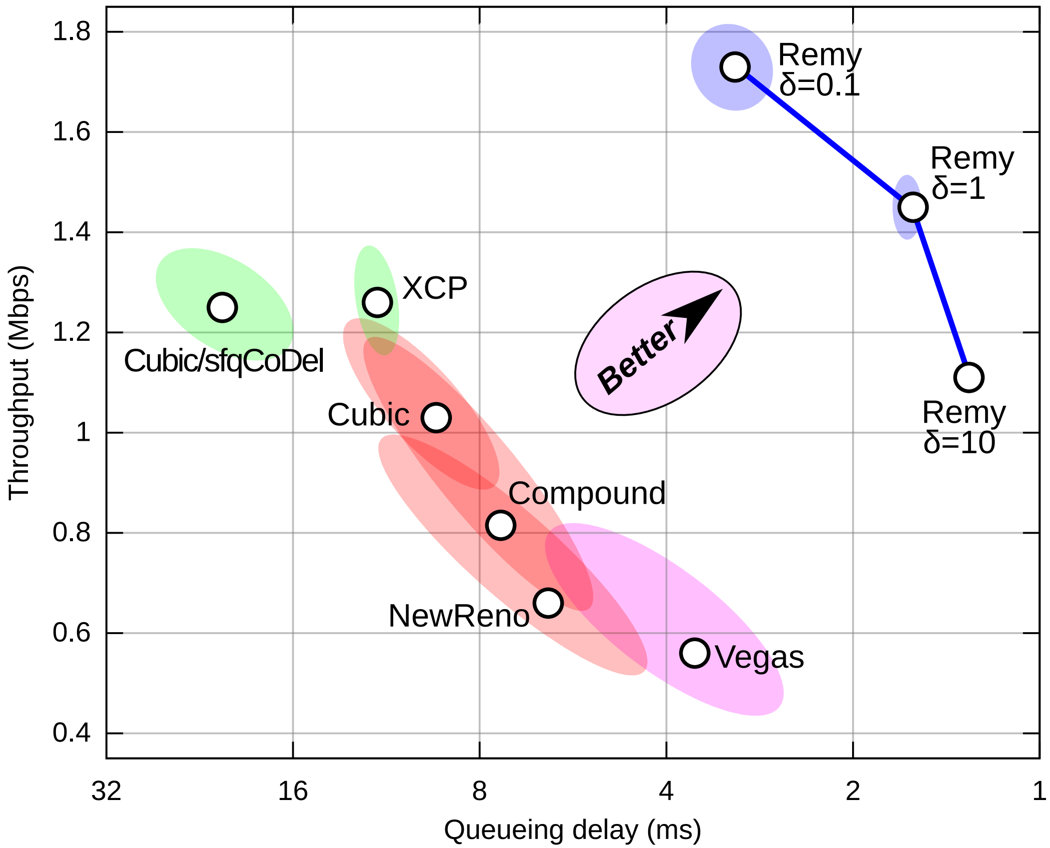
Final RemyCC rule table



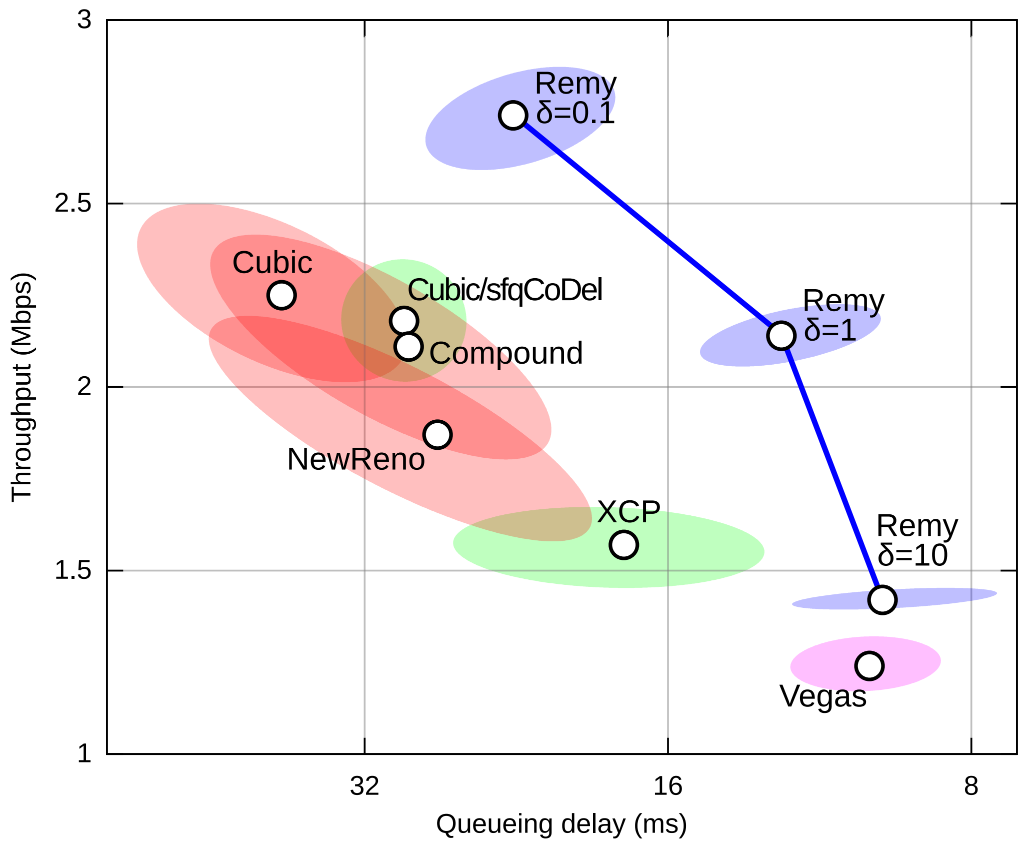
Evaluation

- 3 RemyCCs with $\delta = .1, 1, 10$
- Generating one RemyCC
  - Takes a few hours
  - $5-$10 on EC2
- Compared against end-to-end and network assisted congestion control schemes

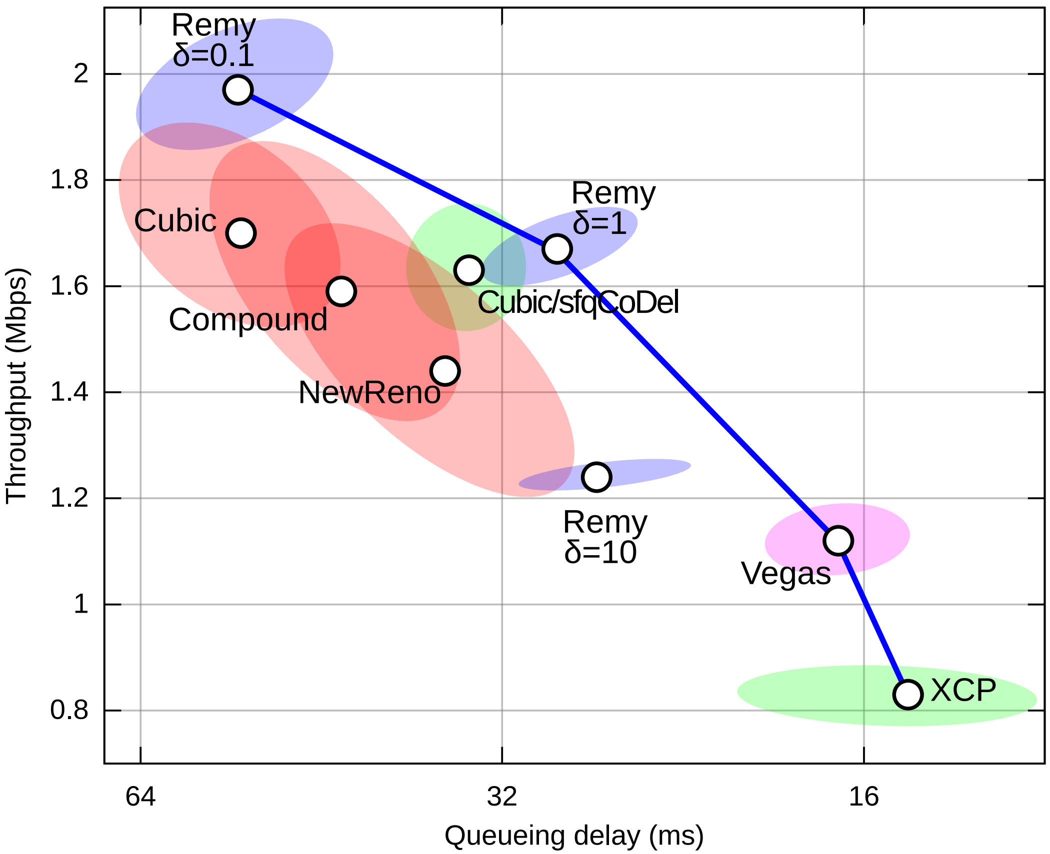
Single bottleneck ("dumbbell")



Varying throughput ("cellular")

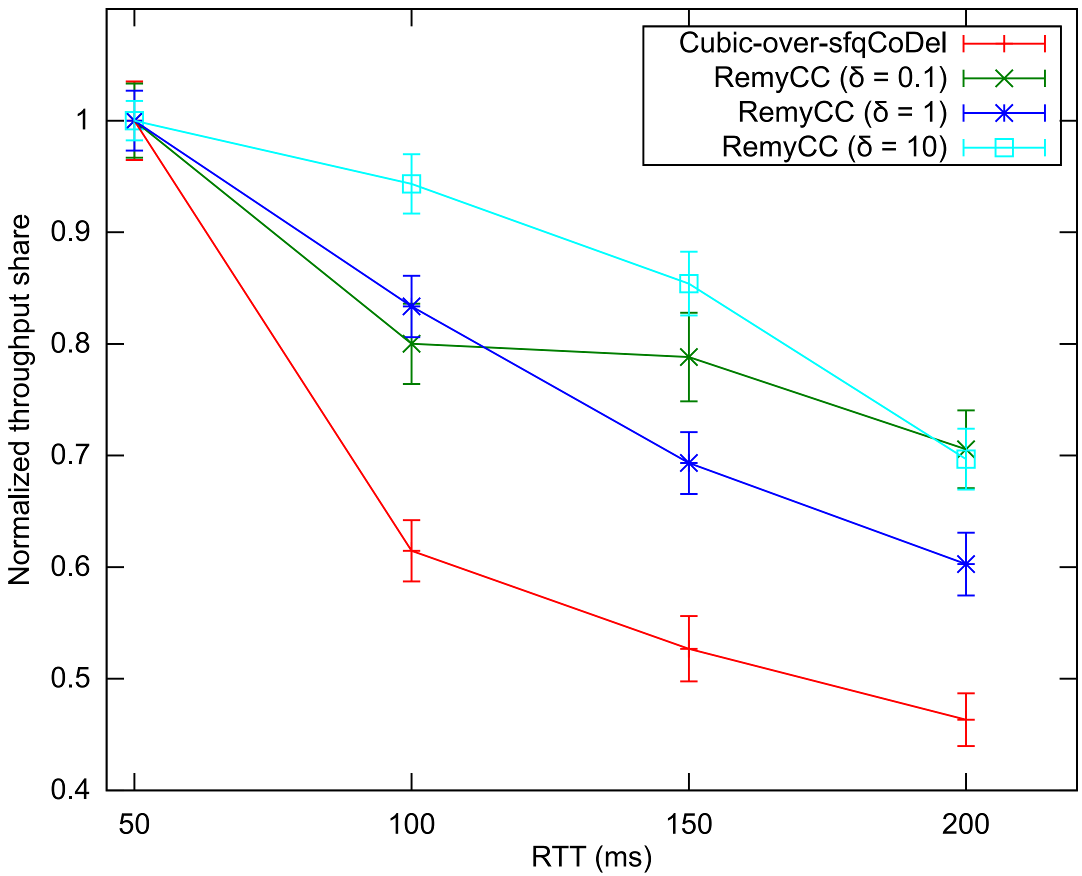


Cellular: n=8



Remy
δ=0.1
Remy
δ=1
Cubic
Cubic/sfqCoDel
Compound
NewReno
Remy
δ=10
Vegas
XCP
2
1.8
1.6
1.4
1.2
1
0.8
Throughput (Mbps)
64
32
16
Queueing delay (ms)

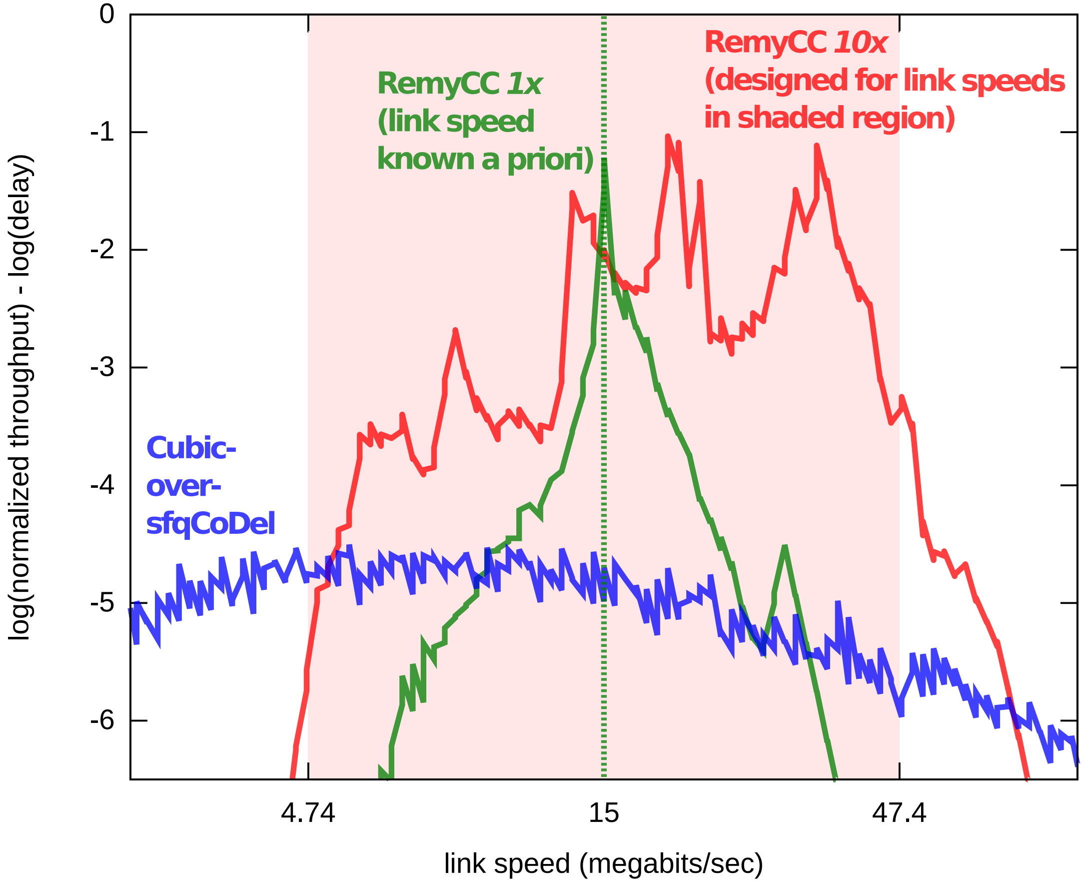
Fairness for varying RTTs



Datacenters

- Issues
  - Many synchronous requests -> incast
  - Diverse mix of short and long flows
- Simulation
  - 64 connections, 10 Gpbs link
  - Objective function maximizes throughput
  - Comparable throughputs to DCTCP
  - DCTCP has shorter RTTs

Sensitivity of design range



Discussion: practicality

- Providing network assumptions
- Scalability
- Using CC algorithms we don't understand
- Coexisting with other protocols

Approaching congestion control differently

- Complex rules but consistent behavior
- Objective and environment driven
- Able to evolve