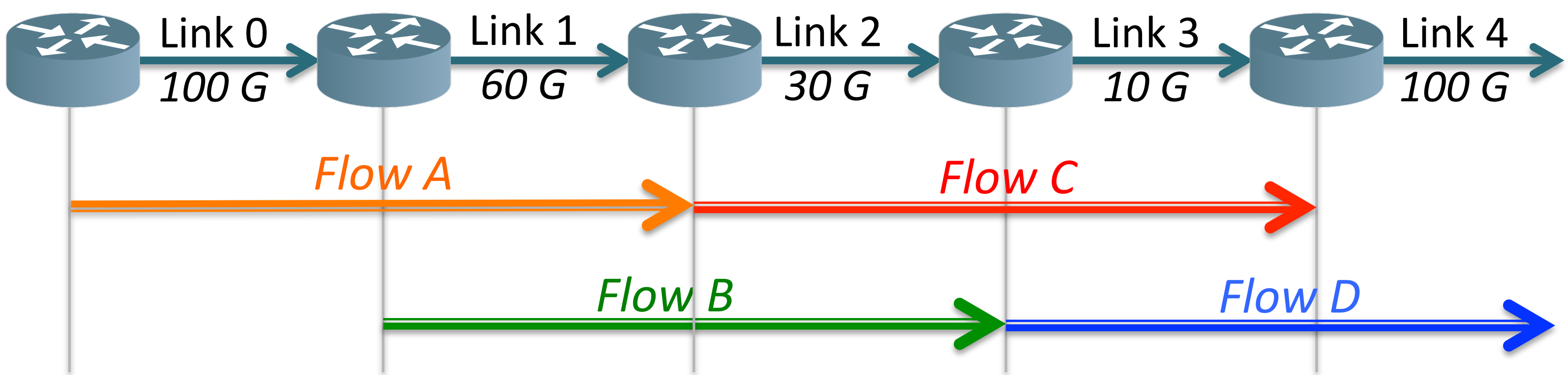
High Speed Networks Need Proactive Congestion Control

Lavanya Jose, Lisa Yan, Nick McKeown, Sachin Katti
Stanford University

Mohammad Alizadeh
MIT

George Varghese
Microsoft Research

The Congestion Control Problem

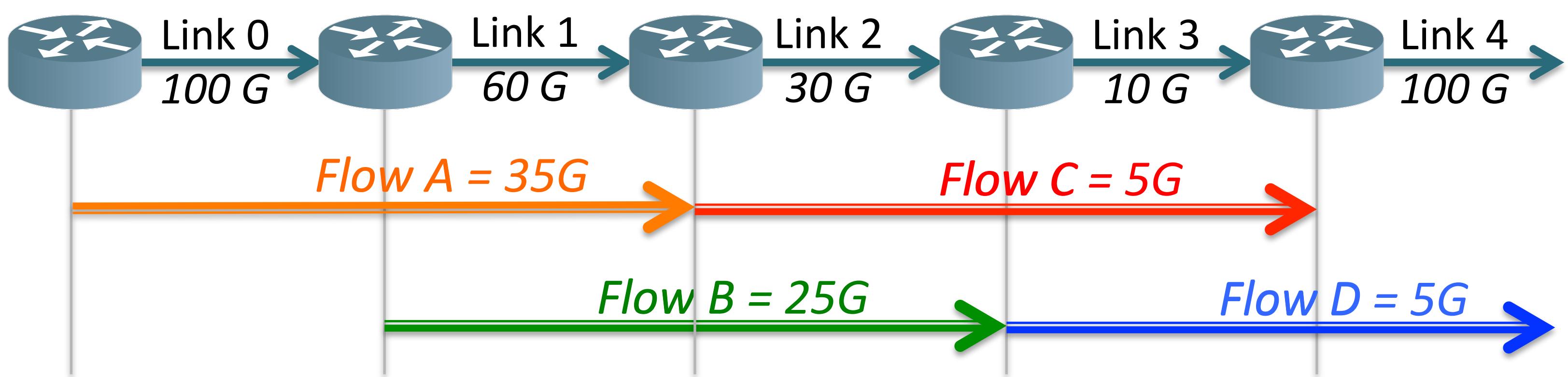


Ask an oracle.

	Link 0
Flow A	√
Flow B	
Flow C	
Flow D	

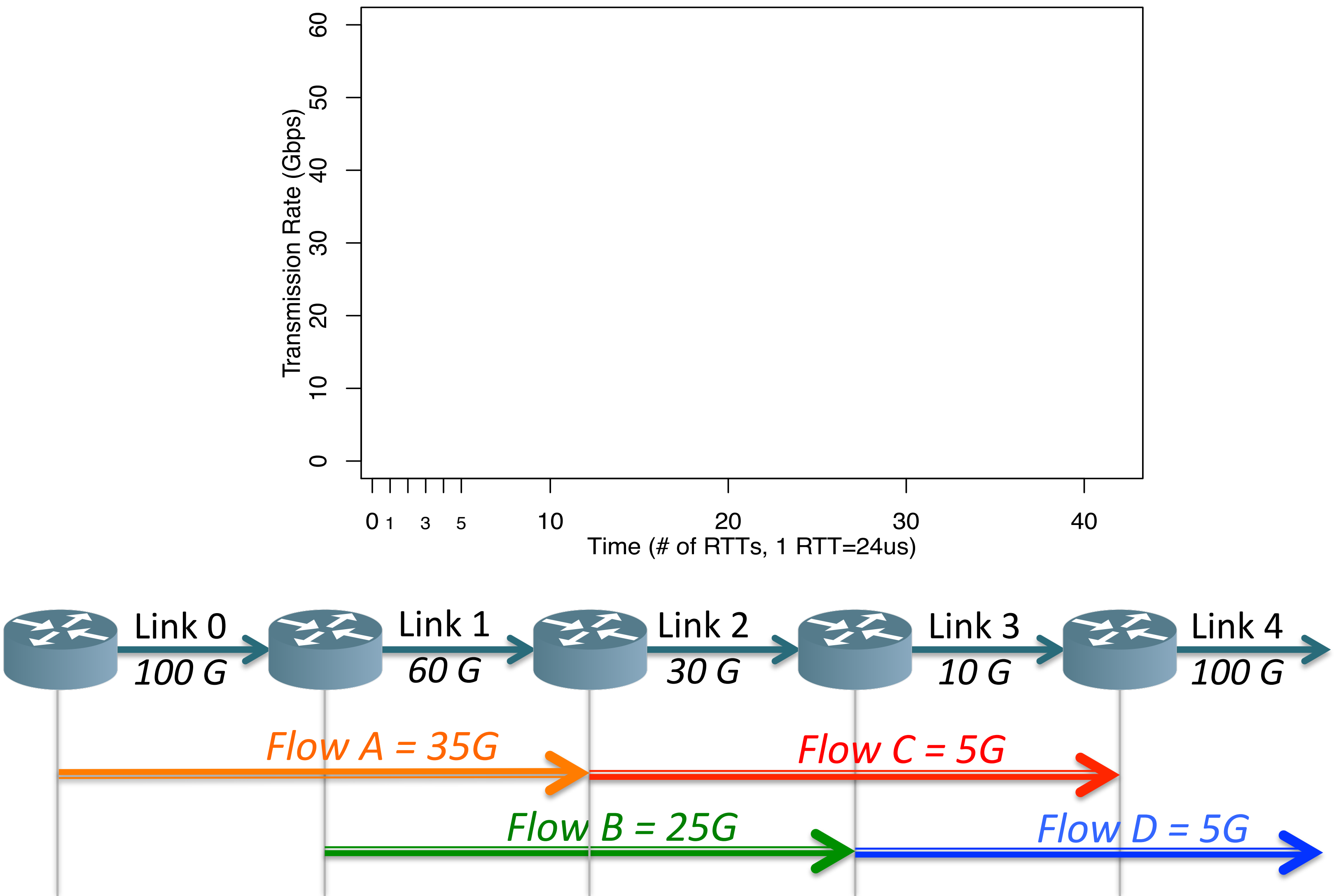
Link	Capacity
0	100
1	60
2	30
3	10
4	100

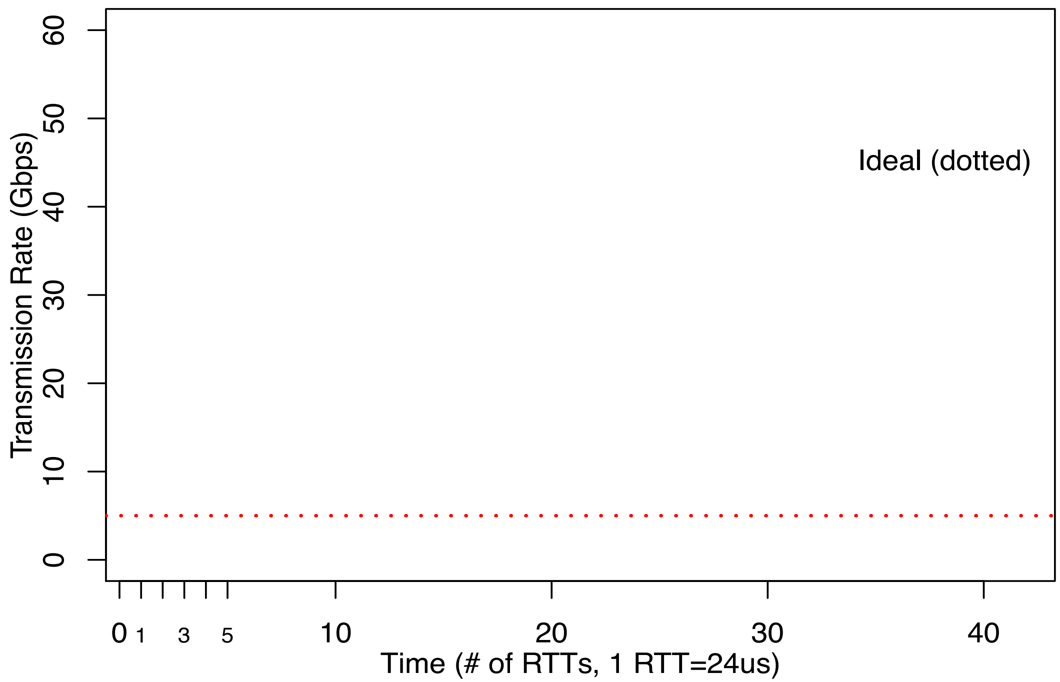
Flow	Rate
Flow A	35
Flow B	25
Flow C	5
Flow D	5

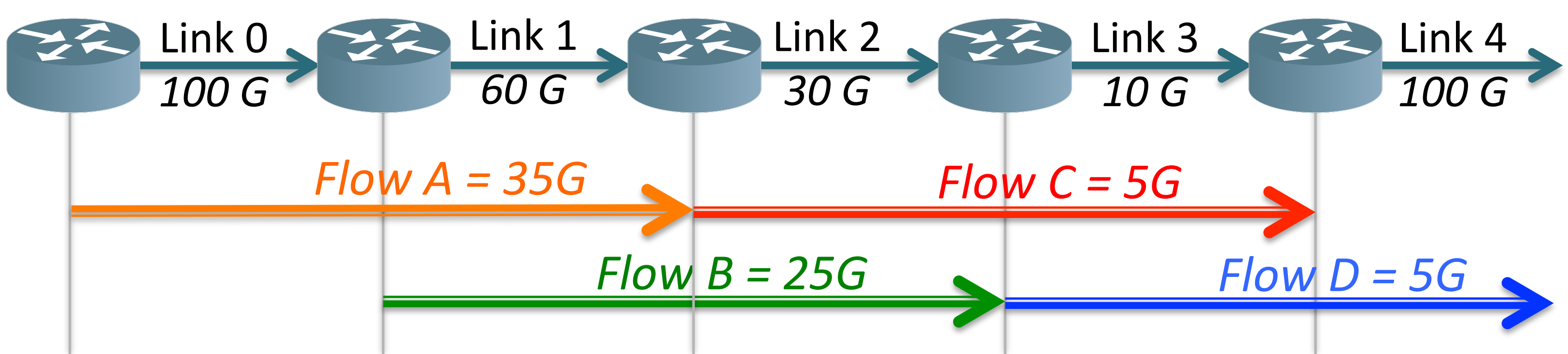


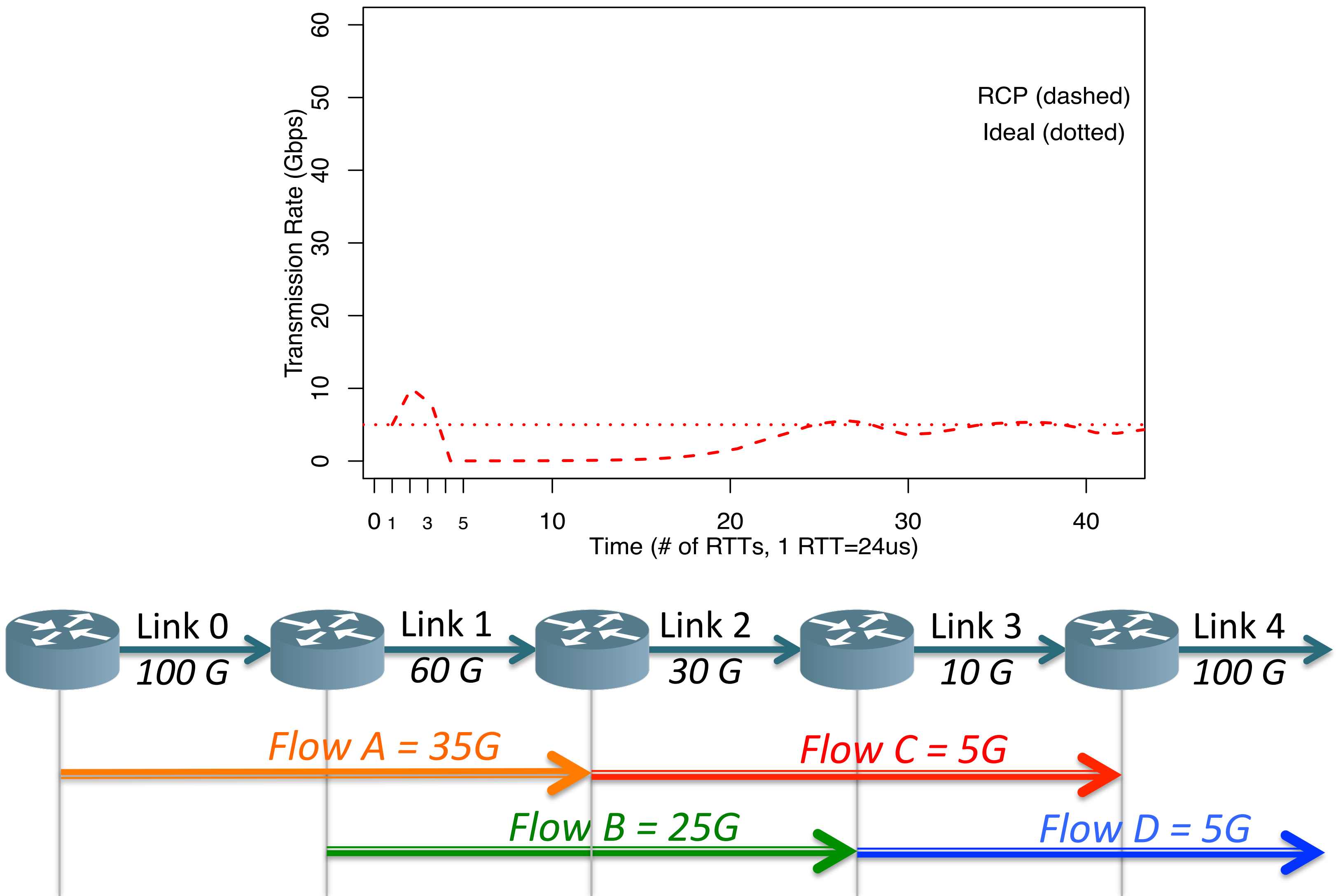
Traditional Congestion Control

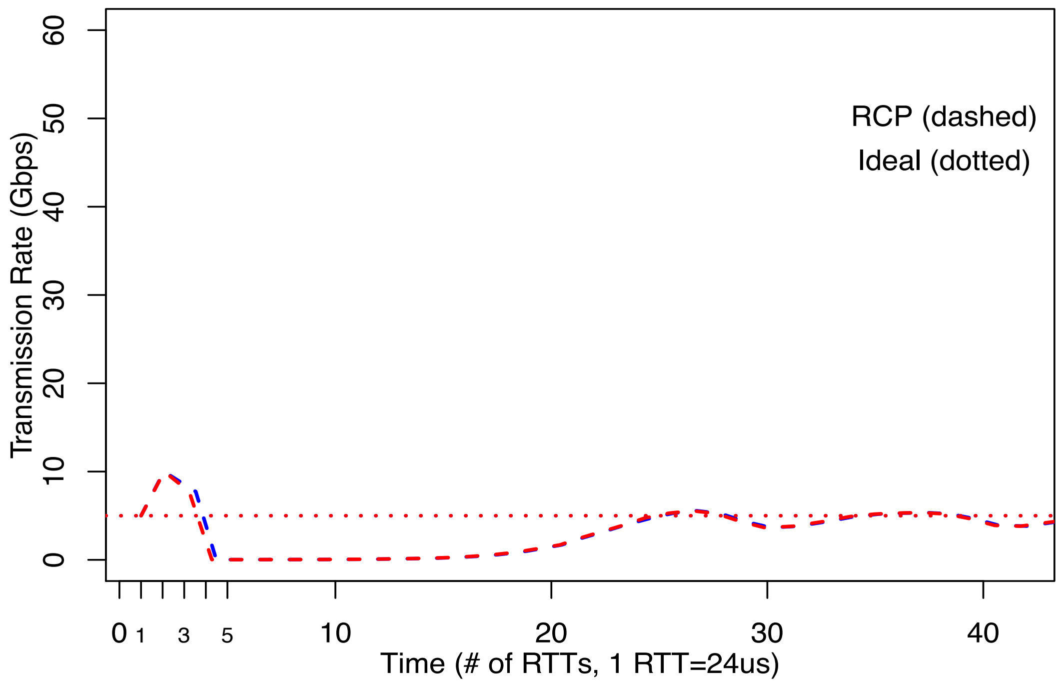
- No explicit information about traffic matrix
- Measure congestion signals, then react by adjusting rate after measurement delay
- Gradual, can't jump to right rates, know direction
- "Reactive Algorithms"


Transmission Rate (Gbps)
0
10
20
30
40
50
60
0
1
3
5
10
20
30
40
Time (# of RTTs, 1 RTT=24us)
Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G

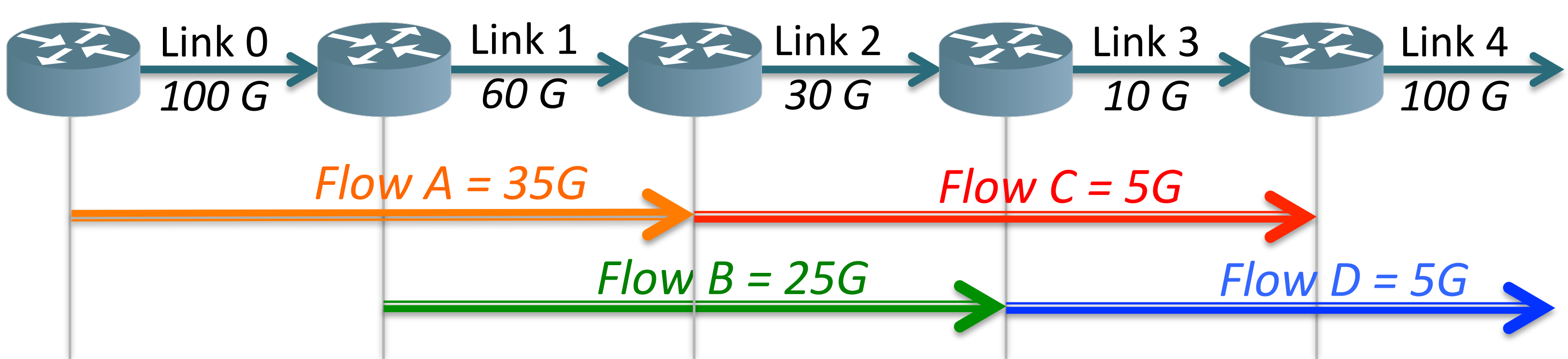

Transmission Rate (Gbps)
0
10
20
30
40
50
60
Ideal (dotted)
0 1 3 5 10 20 30 40
Time (# of RTTs, 1 RTT=24us)


Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G

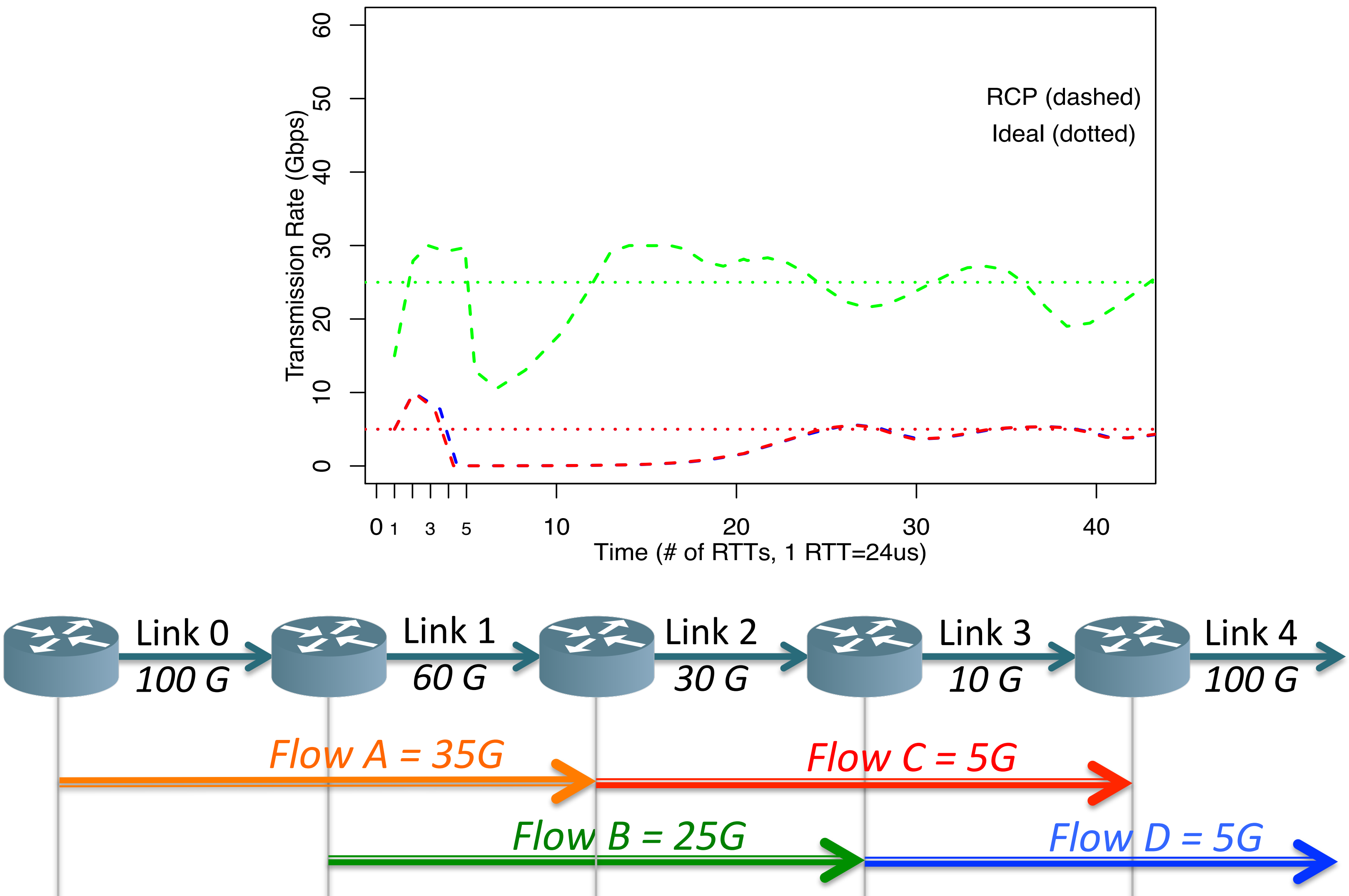

RCP (dashed)
Ideal (dotted)
Transmission Rate (Gbps)
0
10
20
30
40
50
60
0 1 3 5 10 20 30 40
Time (# of RTTs, 1 RTT=24us)
Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G



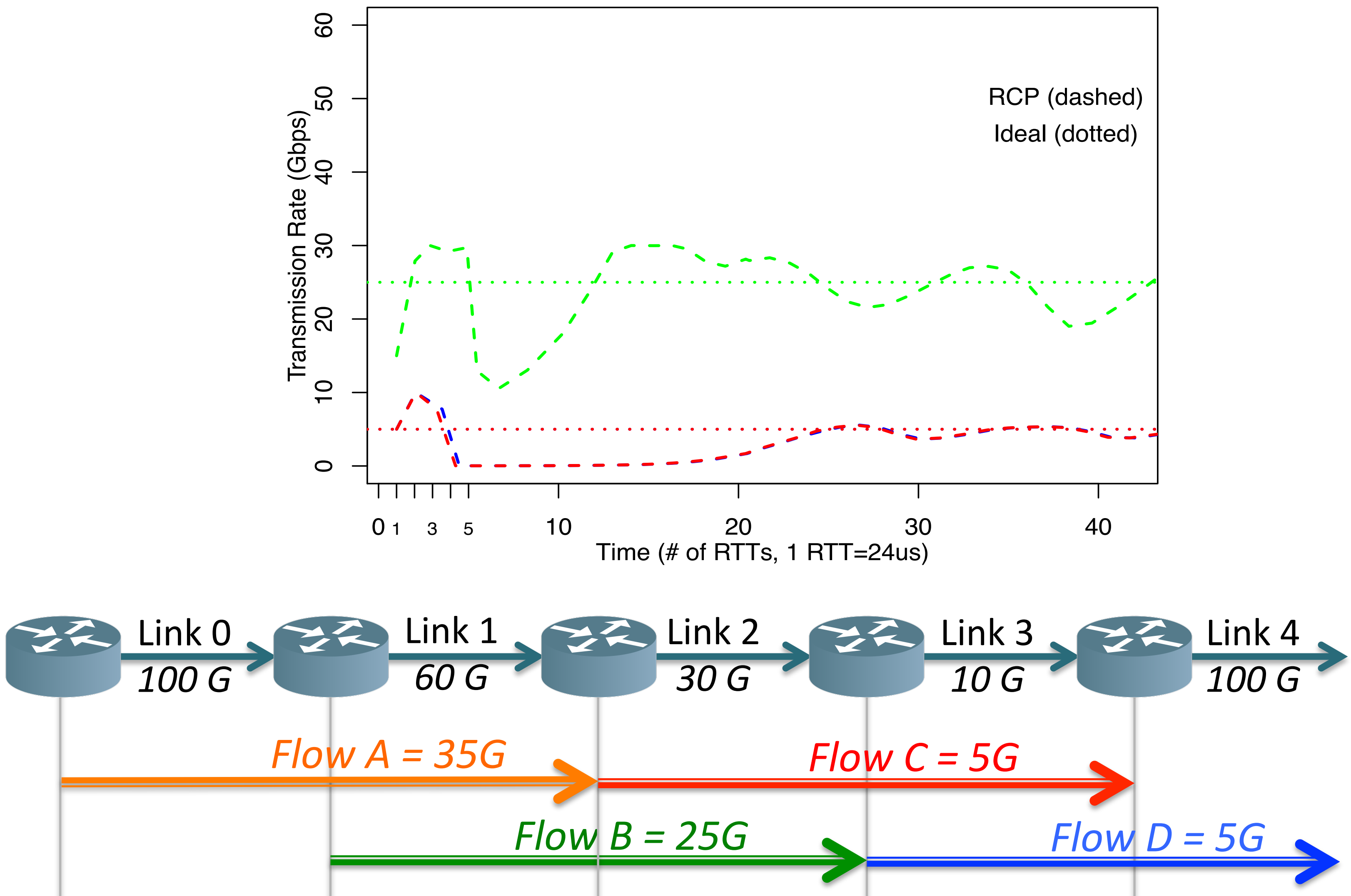
RCP (dashed)
Ideal (dotted)
Transmission Rate (Gbps)
0
10
20
30
40
50
60
0
1
3
5
10
20
30
40
Time (# of RTTs, 1 RTT=24us)



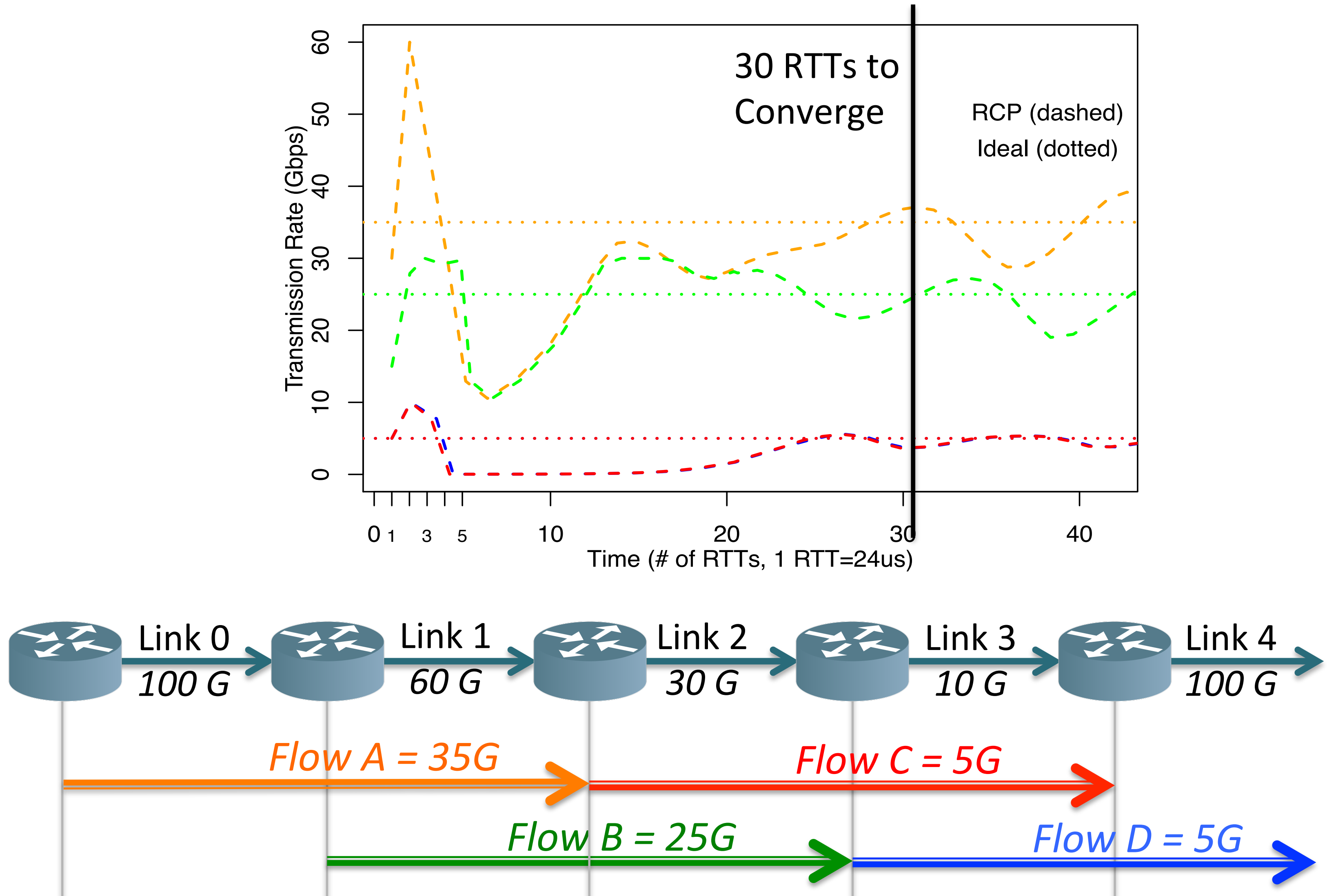
Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G



RCP (dashed)
Ideal (dotted)
Transmission Rate (Gbps)
0
10
20
30
40
50
60
0
1
3
5
10
20
30
40
Time (# of RTTs, 1 RTT=24us)
Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G

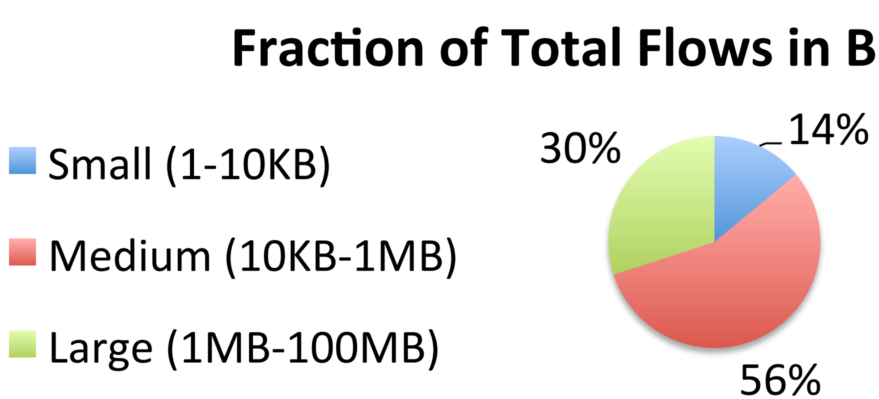


RCP (dashed)
Ideal (dotted)
Transmission Rate (Gbps)
0
10
20
30
40
50
60
0 1 3 5 10 20 30 40
Time (# of RTTs, 1 RTT=24us)
Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G


30 RTTs to Converge
RCP (dashed)
Ideal (dotted)
Transmission Rate (Gbps)
0
10
20
30
40
50
60
0 1 3 5 10 20 30 40
Time (# of RTTs, 1 RTT=24us)
Link 0
100 G
Link 1
60 G
Link 2
30 G
Link 3
10 G
Link 4
100 G
Flow A = 35G
Flow C = 5G
Flow B = 25G
Flow D = 5G

Convergence Times Are Long

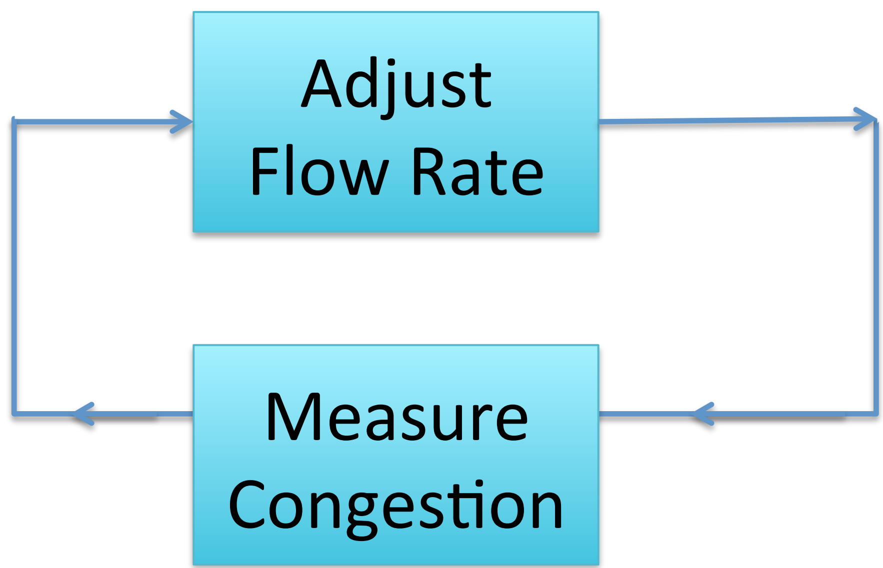
- If flows only last a few RTTs, then we can't wait 30 RTTs to converge.
- At 100G, a typical flow in a search workload is < 7 RTTs long.

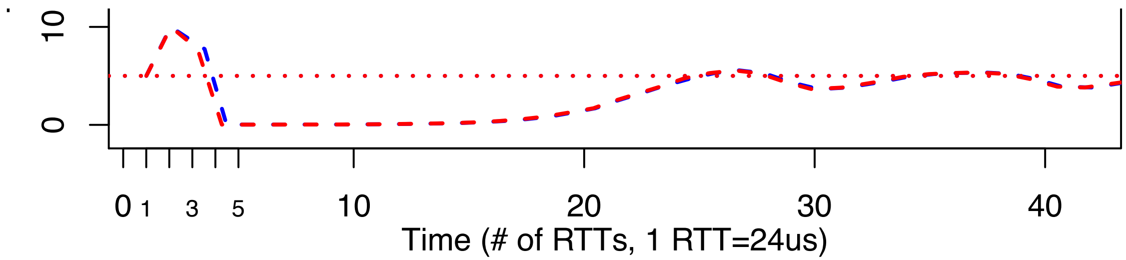


1MB / 100 Gb/s = 80 µs

Why "Reactive" Schemes Take Long

1. No explicit information
2. Therefore measure congestion signals, react
3. Can't leap to correct values but know direction
4. Reaction is fed back into network
5. Take cautious steps

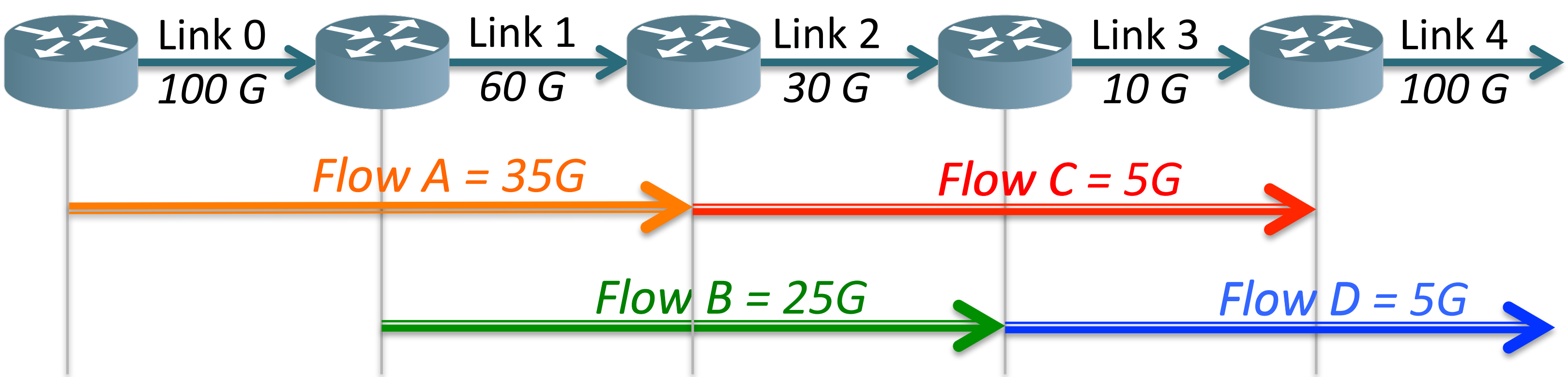




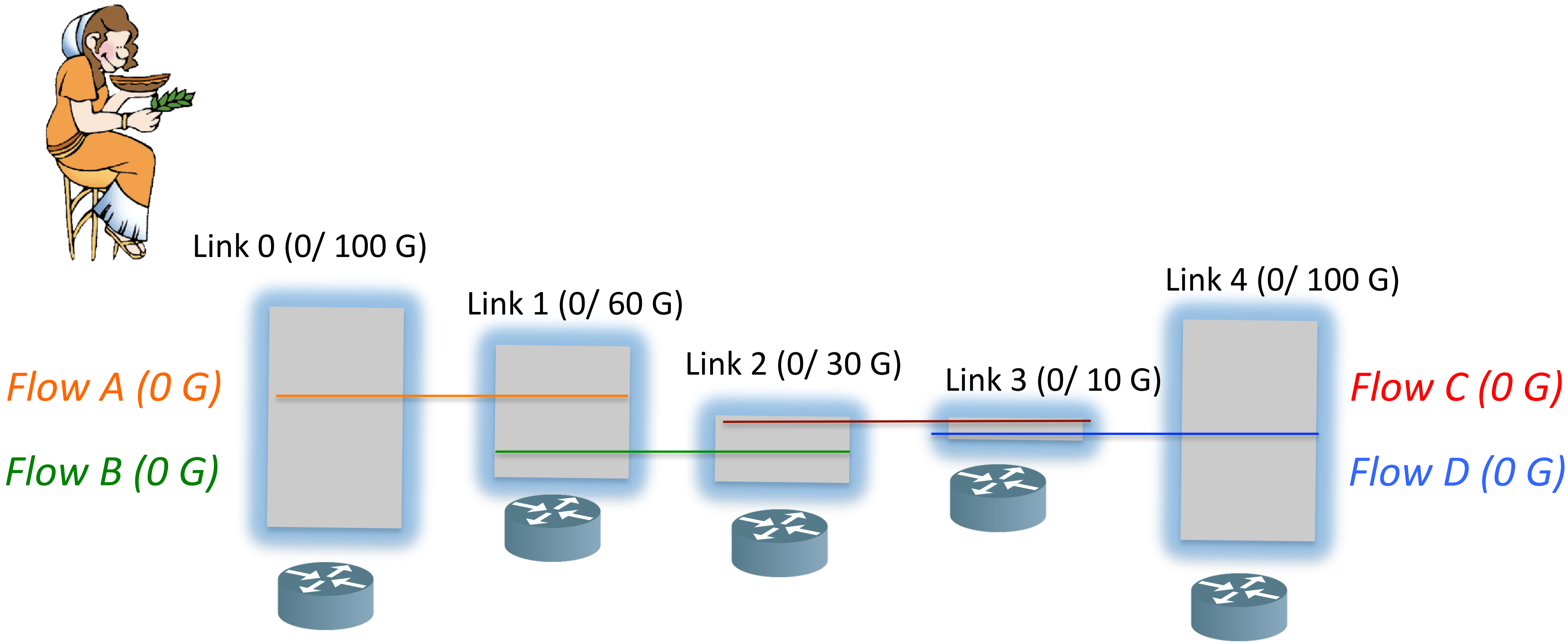
Reactive algorithms trade off *explicit flow information* for *long convergence times*

Can we use explicit flow information and get shorter convergence times?

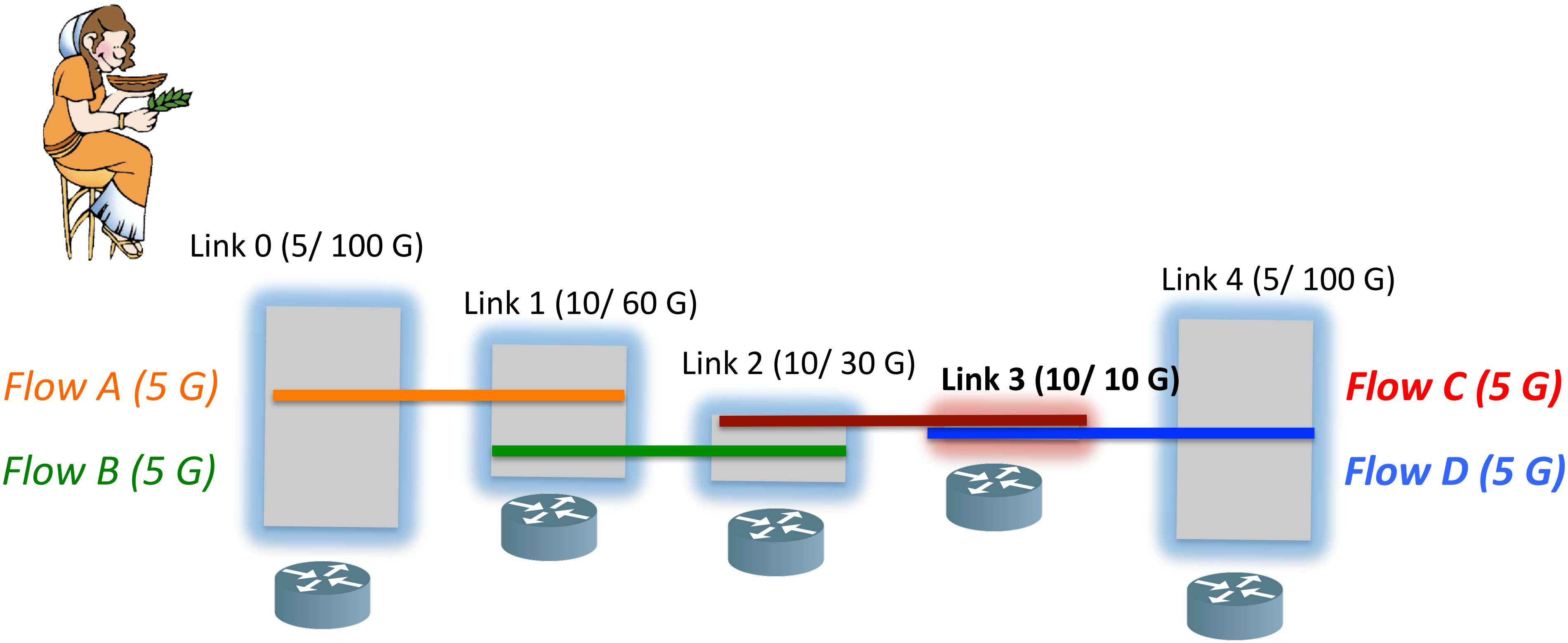
Back to the oracle, how did she use traffic matrix to compute rates?



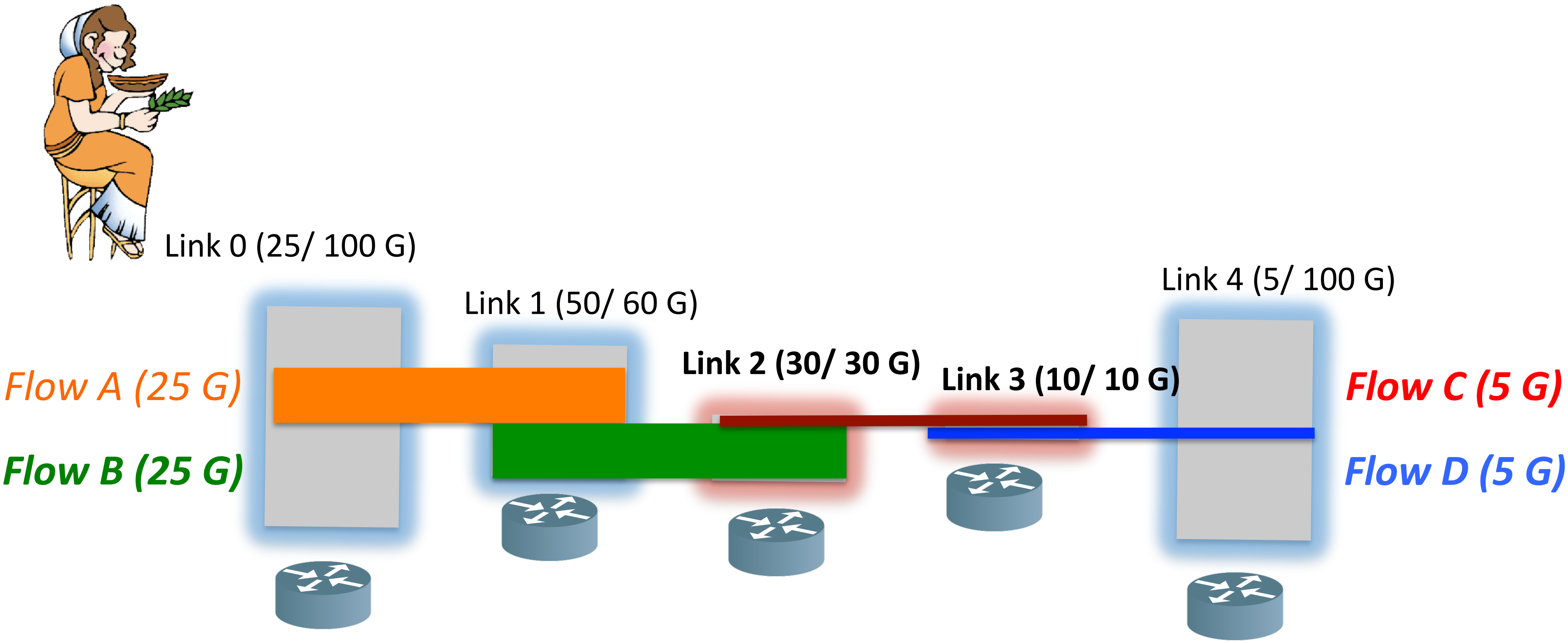
Waterfilling Algorithm



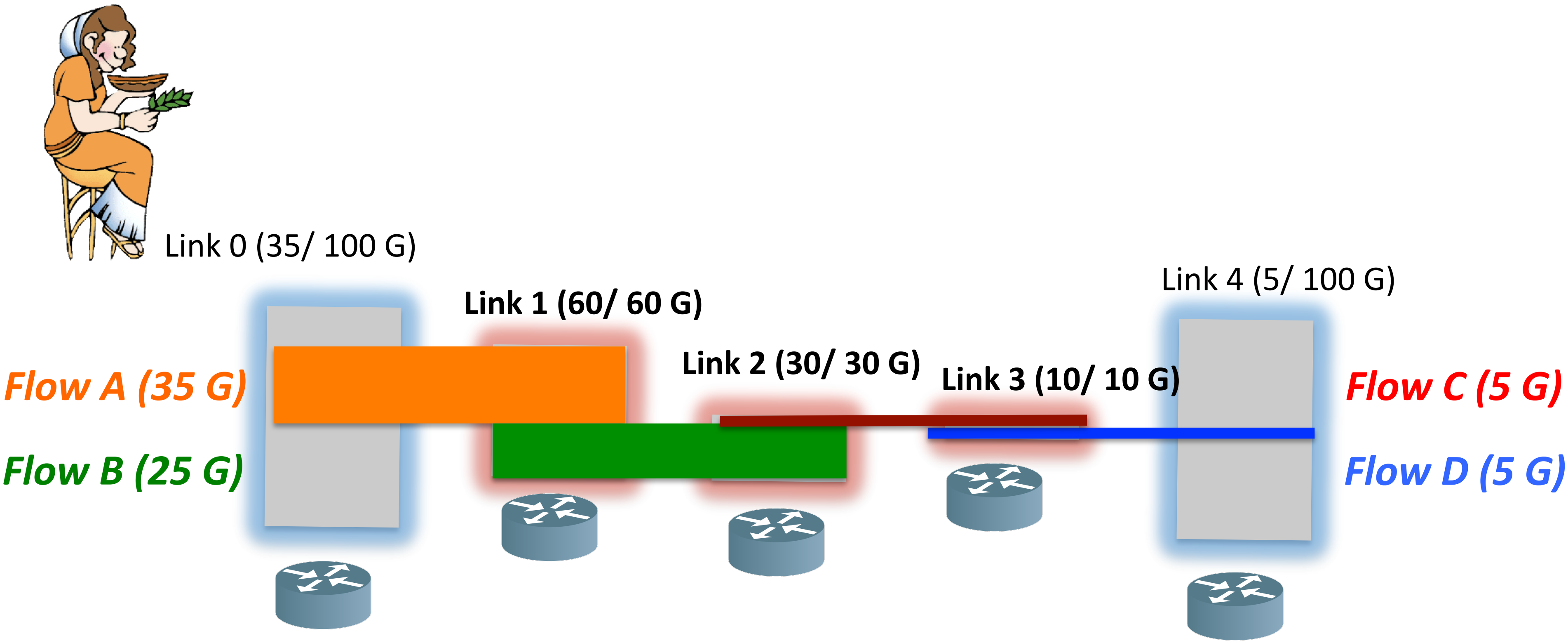
Waterfilling- 10 G link is fully used



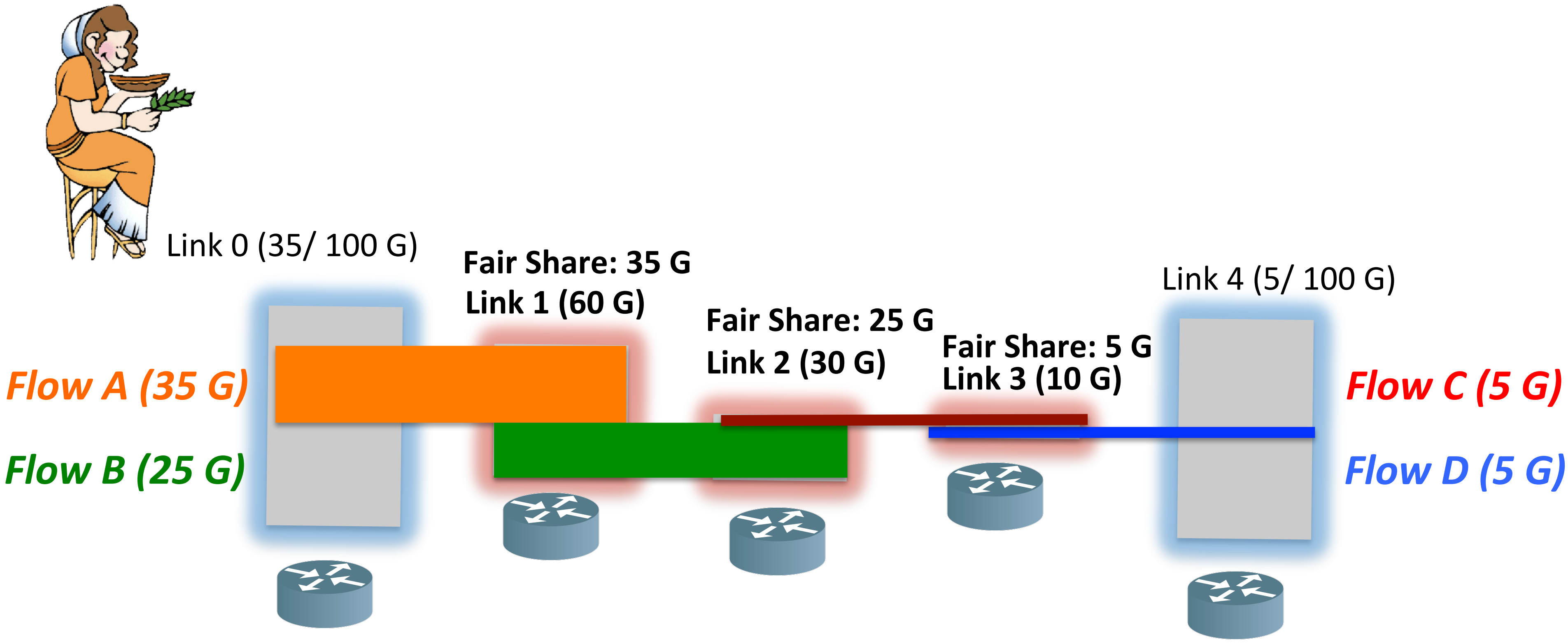
Waterfilling- 30 G link is fully used



Waterfilling- 60 G link is fully used



Fair Share of Bottlenecked Links



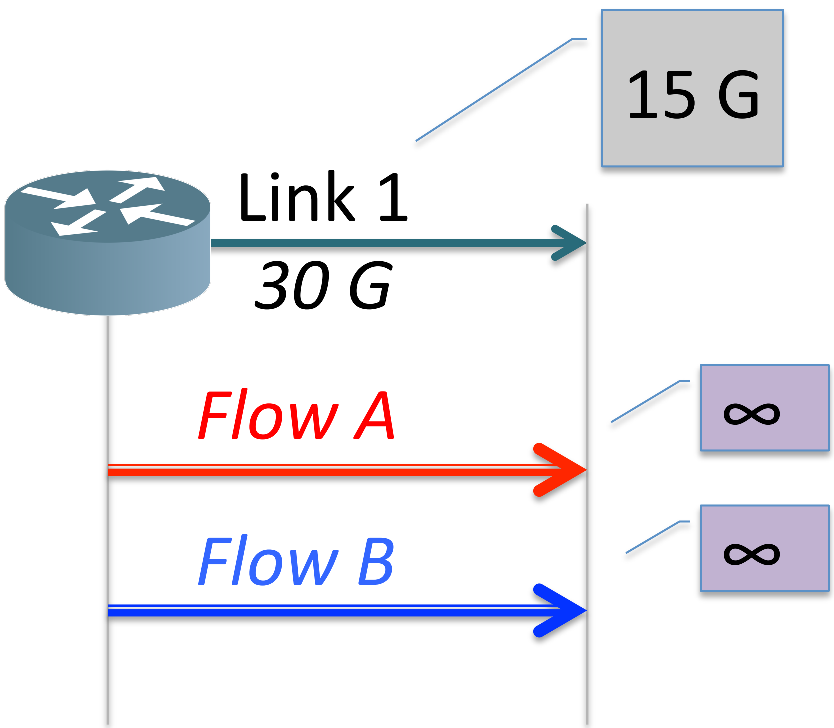
A *centralized* water-filling scheme may not scale.

Can we let the network figure out rates in a distributed fashion?

Fair Share for a Single Link

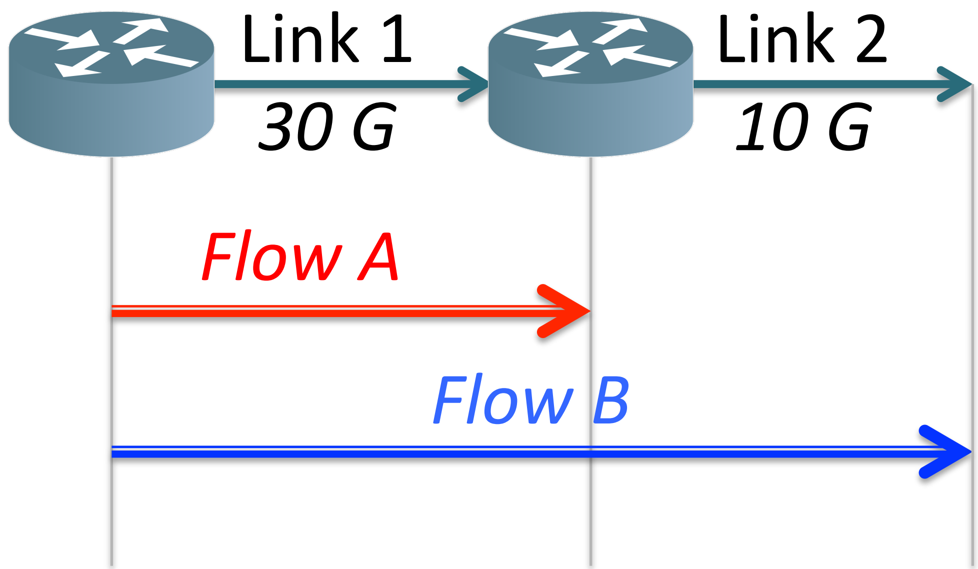
flow	demand
A	∞
B	∞

Capacity at Link 1: 30G
So Fair Share Rate: 30G/2 = 15G

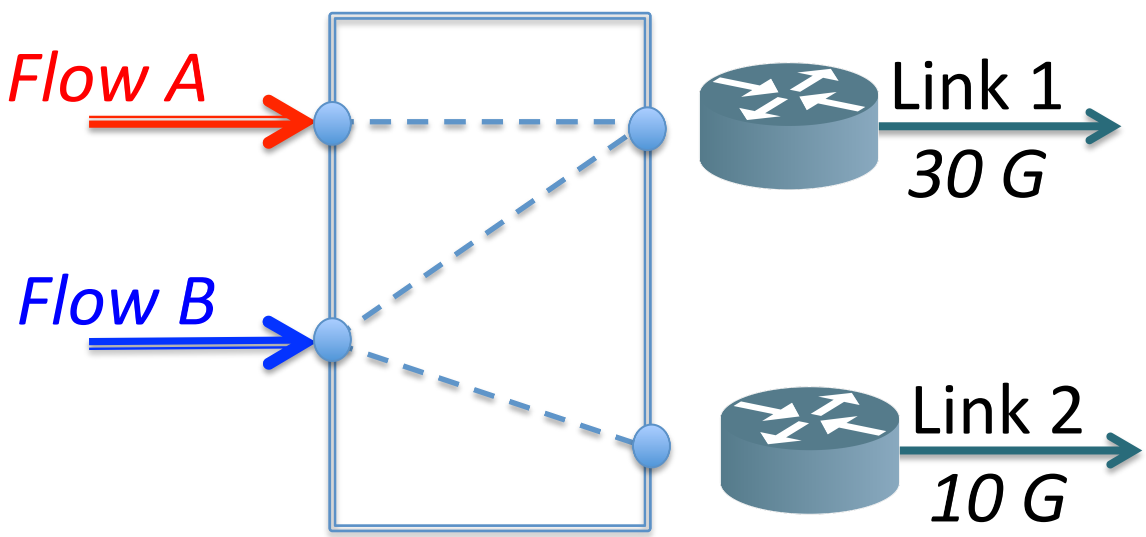


A second link introduces a *dependency*

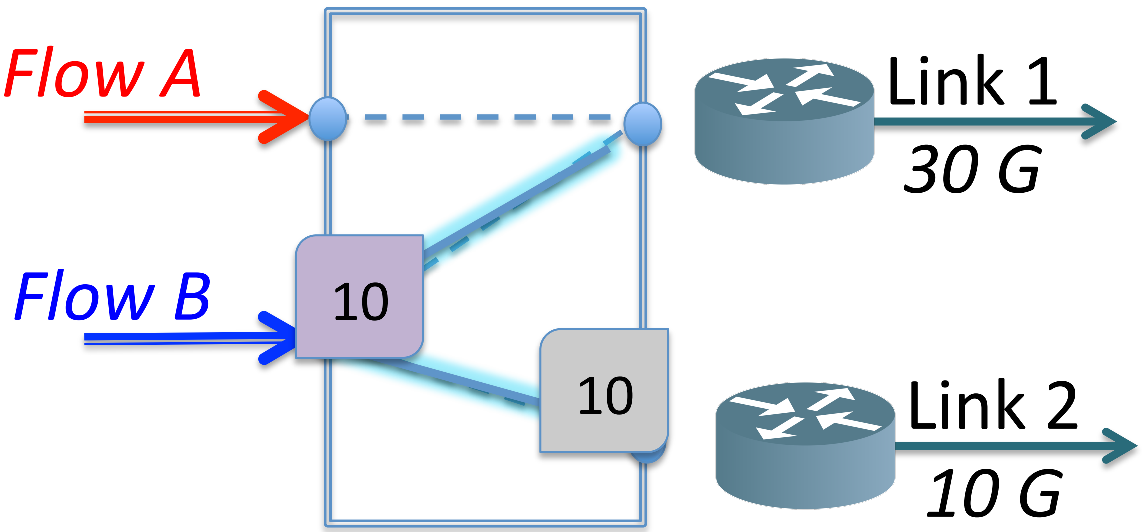
flow	demand
A	∞
B	10 G



Dependency Graph



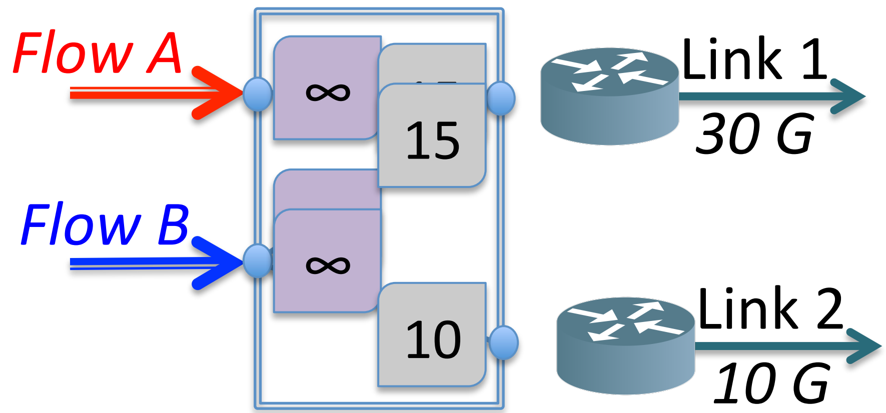
Dependency Graph



Proactive Explicit Rate Control (PERC) Overview

Round 1 (Flows → Links)

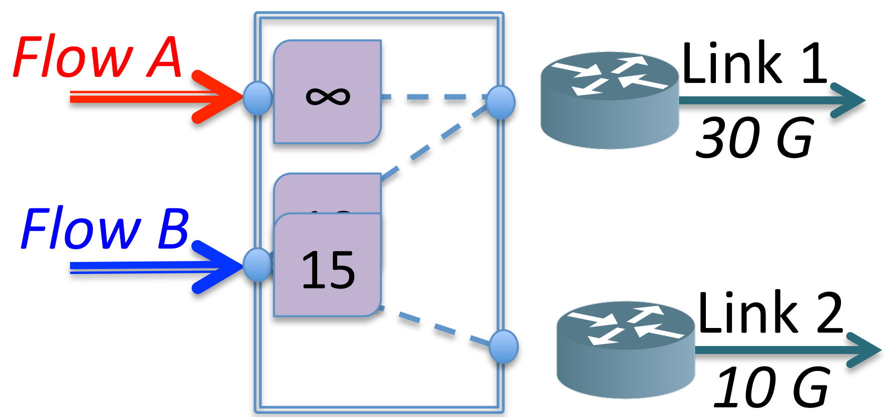
- Flows and links alternately exchange messages.
- A flow sends a "demand"
  - **∞ when no other fair share**
  - min. fair share of other links
- A link sends a "fair share"
  - **C/N when demands are ∞**
  - otherwise use water-filling



Proactive Explicit Rate Control (PERC) Overview

Round 2 (Flows → Links)

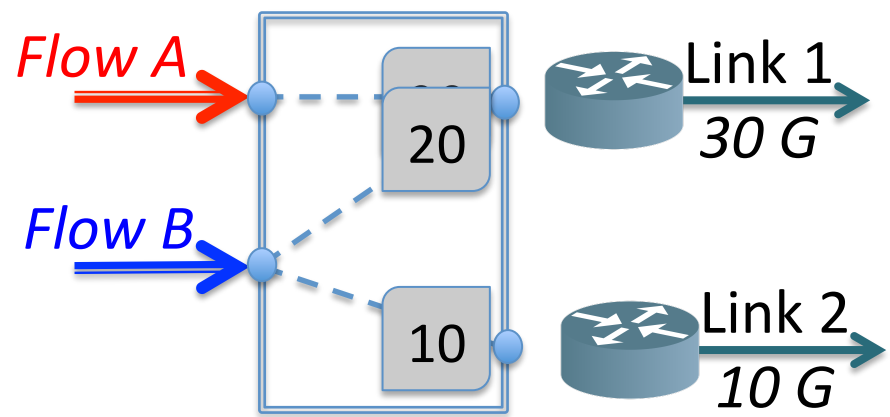
- Flows and links alternately exchange messages.
- A flow sends a "demand"
  - ∞ when no other fair share
  - **min. fair share of other links**
- A link sends a "fair share"
  - C/N when demands are ∞
  - otherwise use water-filling
- Messages are approximate, jump to right values quickly with more rounds



Proactive Explicit Rate Control (PERC) Overview

- Flows and links alternately exchange messages.
- A flow sends a "demand"
  - ∞ when no other fair share
  - min. fair share of other links
- A link sends a "fair share"
  - C/N when demands are ∞
  - **otherwise use water-filling**
- Messages are approximate, jump to right values quickly with more rounds

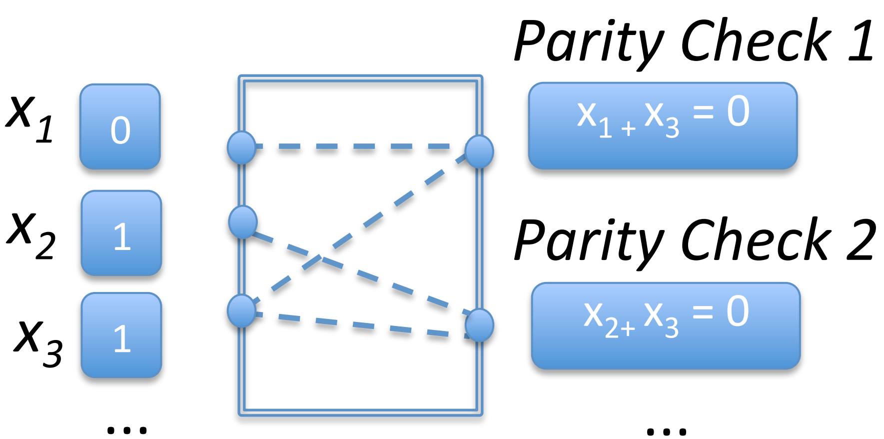
Round 2 (Links → Flows)

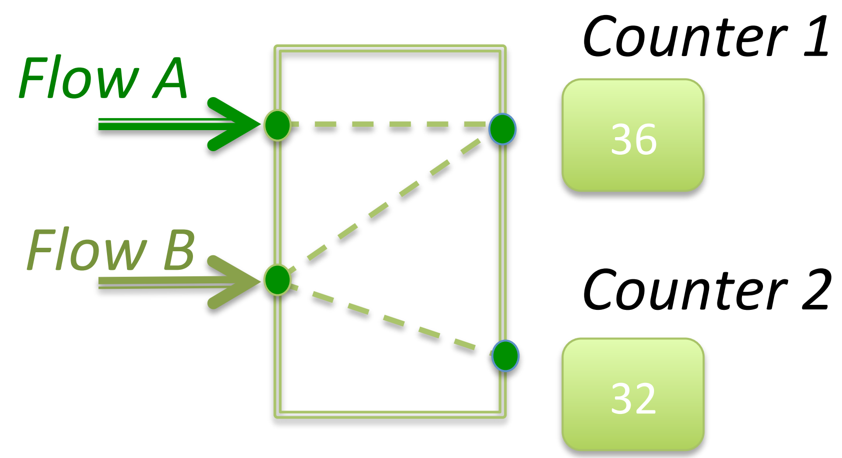


Message Passing Algorithms

Decoding error correcting codes
(LDPC- Gallager, 1963)

Flow counts using shared counters
(Counter Braids- Lu et al, 2008)



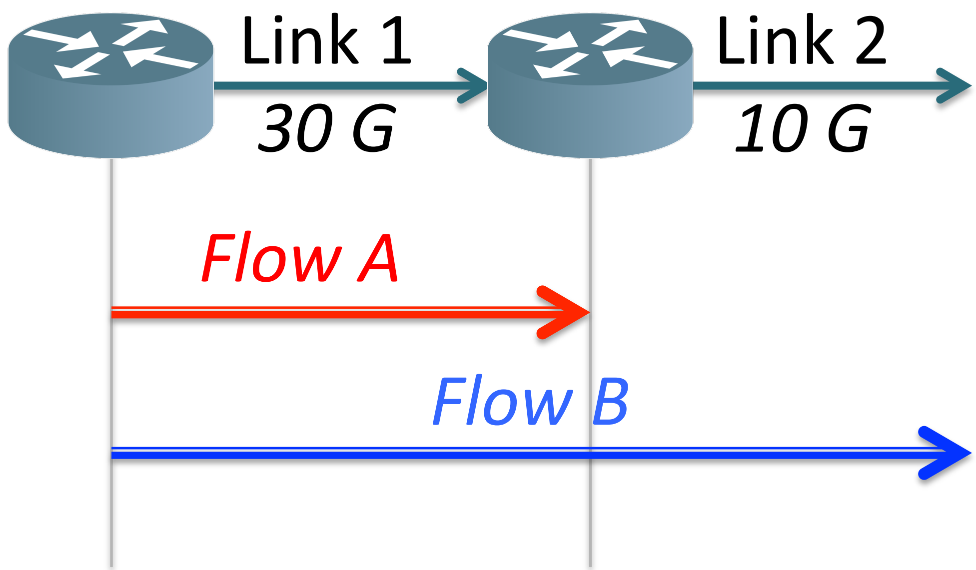


Making PERC concrete

PERC Implementation

Control Packet For Flow B

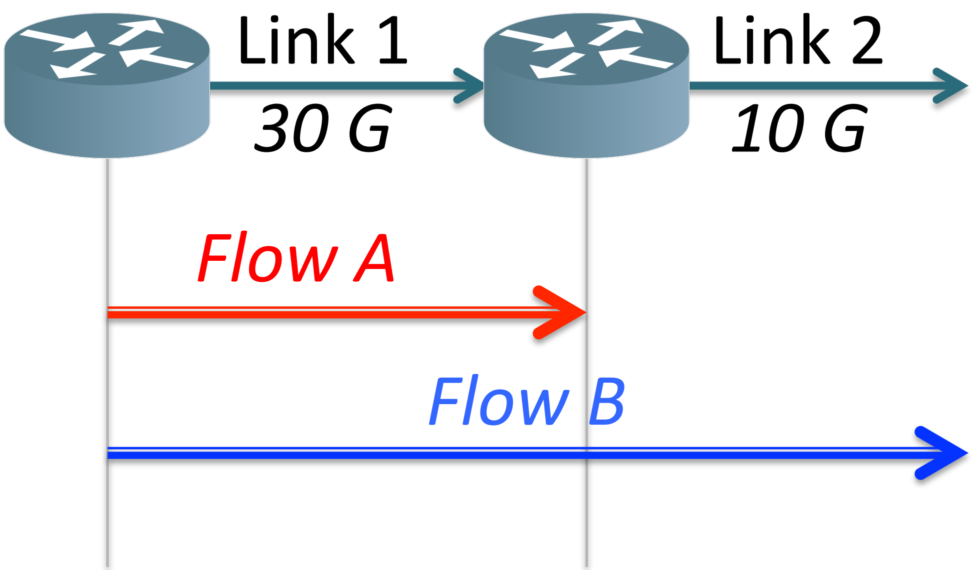
d	∞	∞
f	?	?



PERC Implementation

Control Packet For Flow B

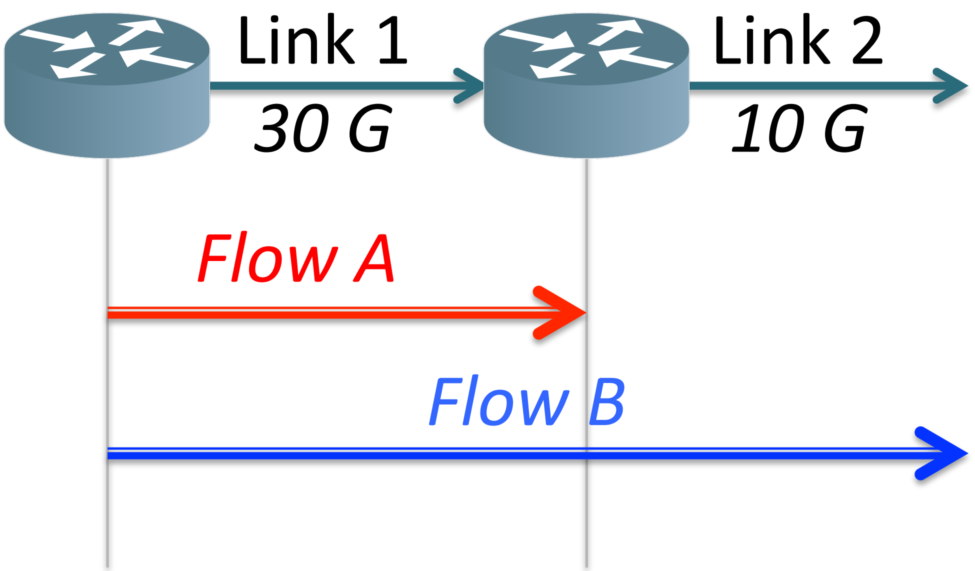
d	∞	∞
f	15	?



PERC Implementation

Control Packet For Flow B

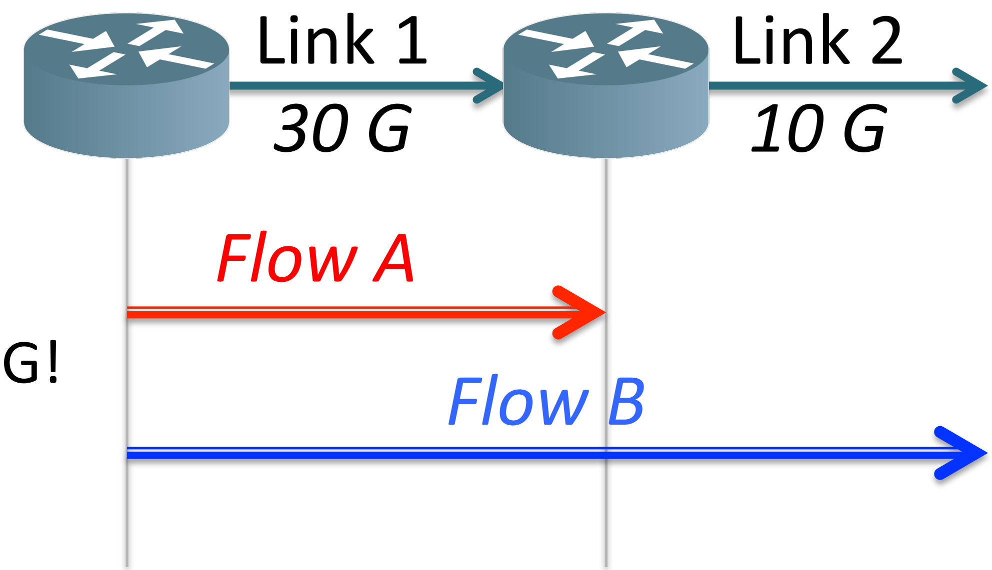
d	∞	∞
f	15	10



PERC Implementation

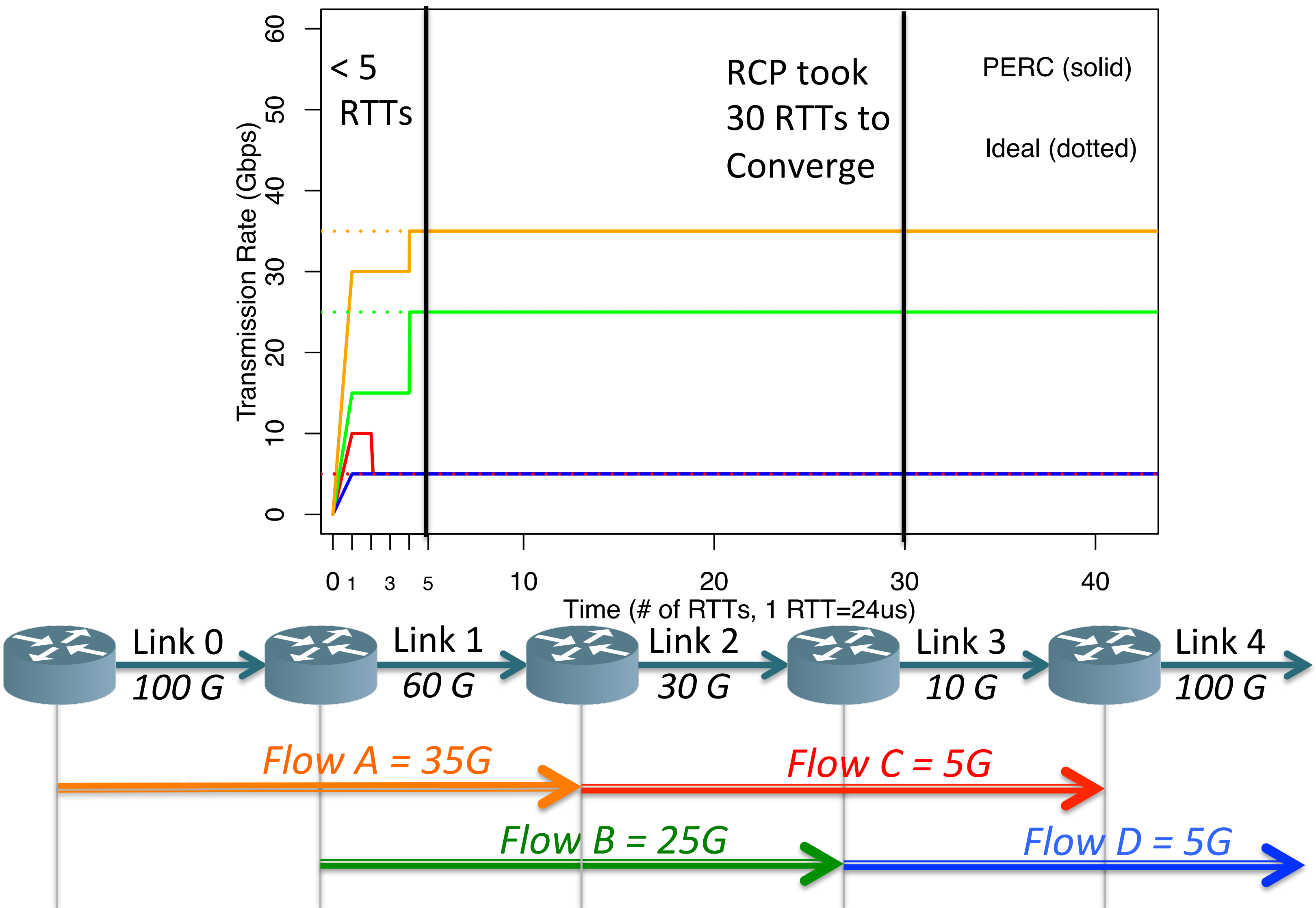
Control Packet For Flow B

d	***10***	***15***
f	?	?

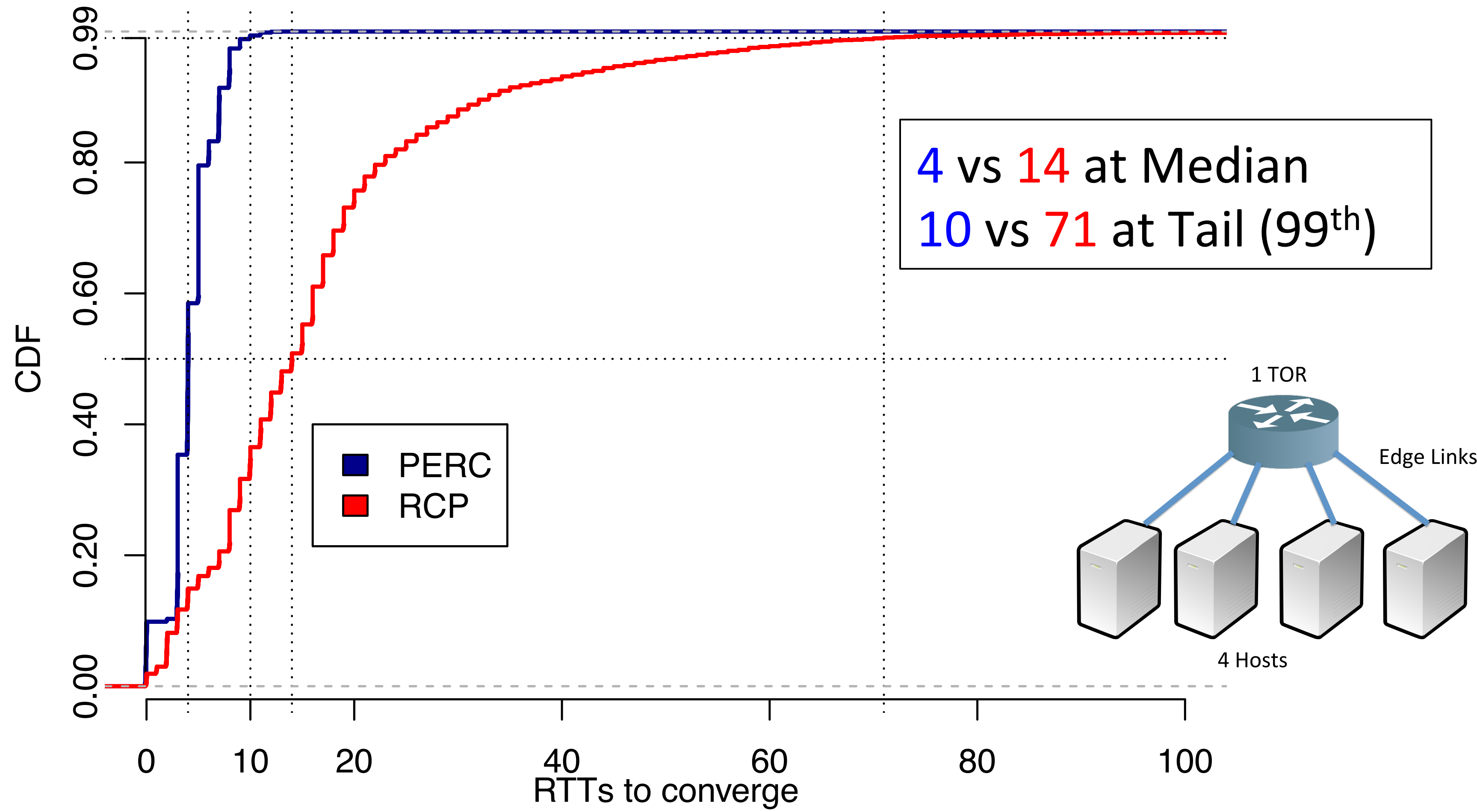


send at 15G!

PERC converges fast



PERC Converges Fast



Some unanswered questions

- How to calculate fair shares in PERC switches?
- How to bound convergences times in theory?
- What about other policies?

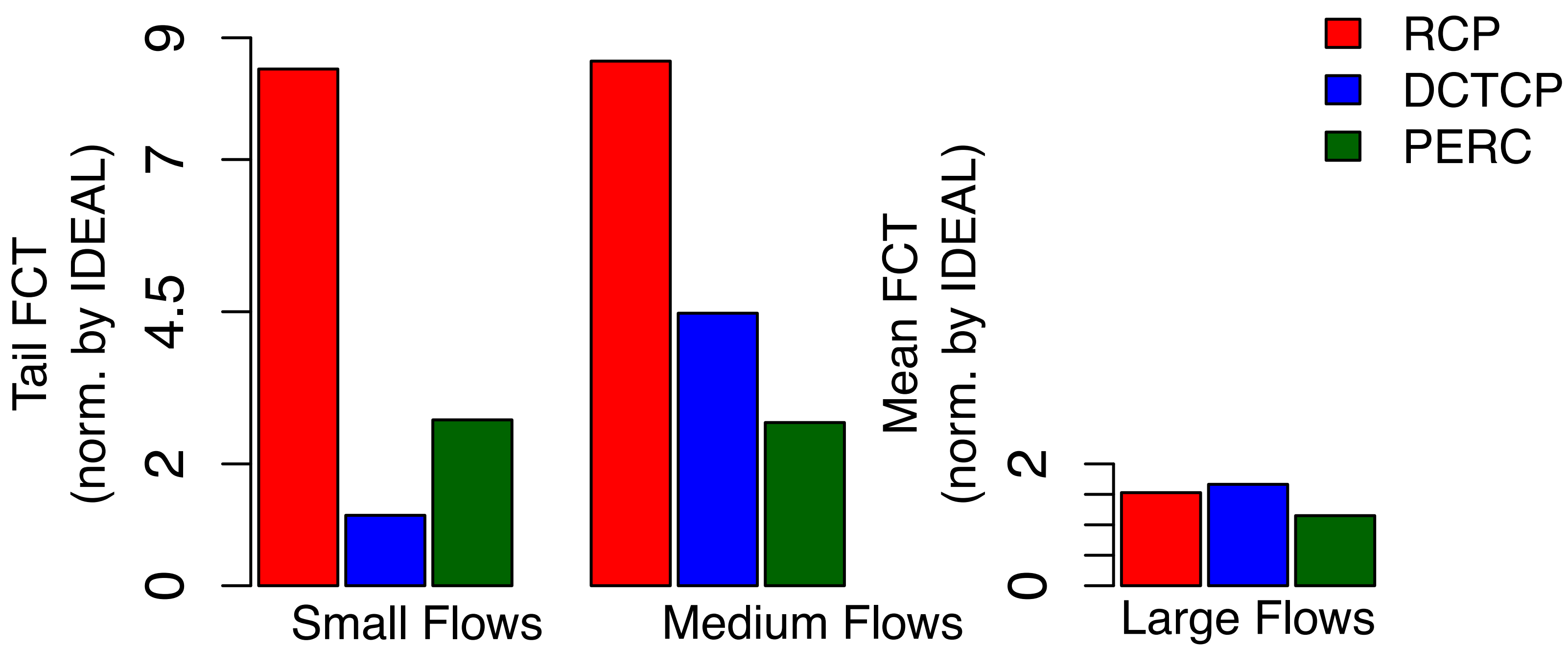
Takeways

- Reactive schemes are slow for short flows (majority) at 100G
- Proactive schemes like PERC are fundamentally different and can converge quickly because they calculate explicit rates based on out of band information about set of active flows.
- Message passing promising proactive approach- could be practical, need further analysis to understand good convergence times in practice.

Thanks!

Shorter FCTs For Flows That Last A Few RTTs ("Medium")

100G, 12us



XCP

As XCP is window-based, the EC computes a desired increase or decrease in the number of bytes that the aggregate traffic transmits in a control interval (i.e., an average RTT). This aggregate feedback ϕ is computed each control interval:

$$\phi = \alpha \cdot d \cdot S - \beta \cdot Q, \qquad (1)$$

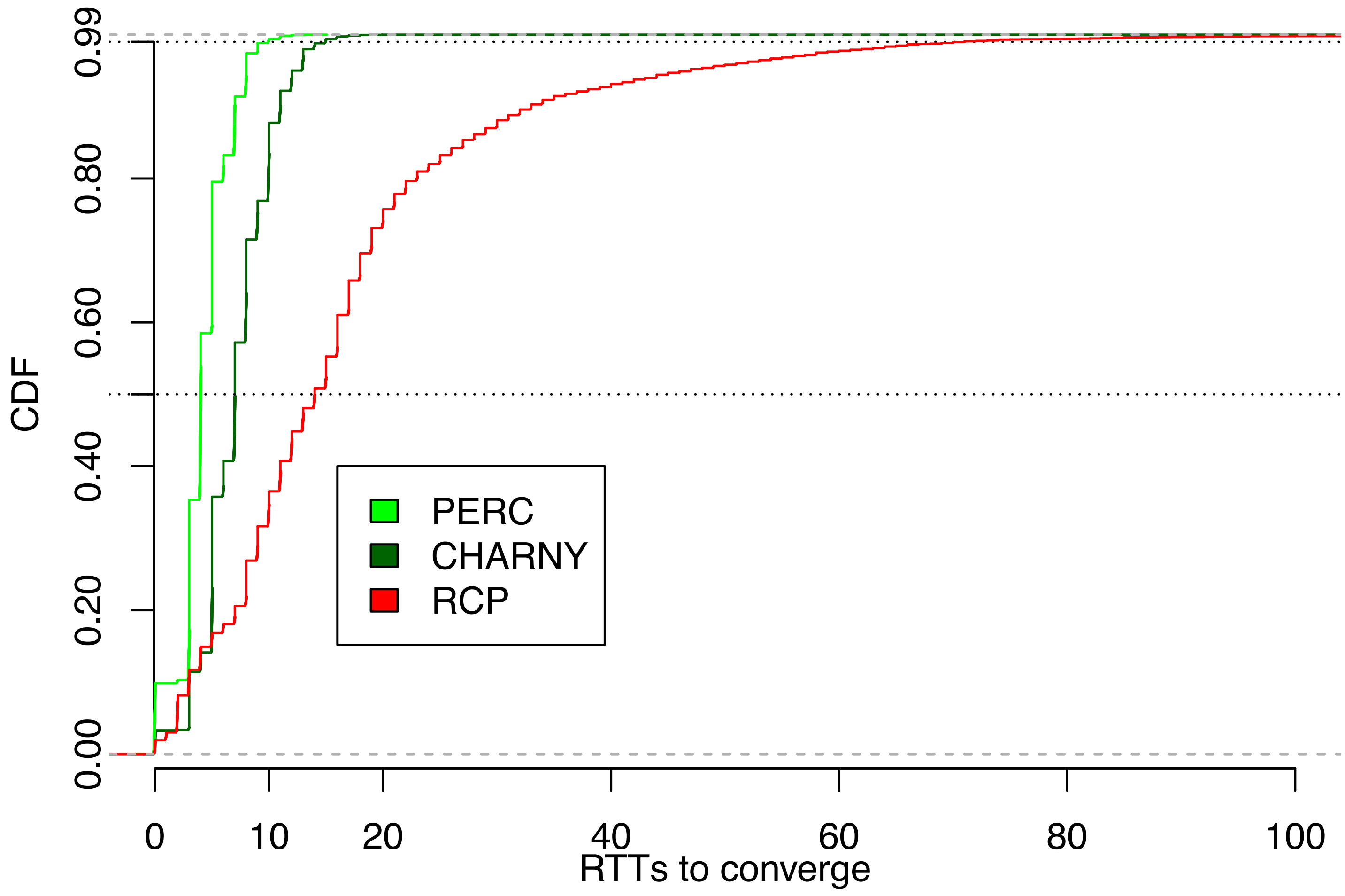
α and β are constant parameters, whose values are set based on our stability analysis (§ 4) to 0.4 and 0.226, respectively. The term d is the average RTT, and S is the spare bandwidth defined as the difference between the input traffic rate and link capacity. (Note that S can be negative.) Finally, Q is the persistent queue size (i.e., the

RCP

R_l based on congestion signals, such as the input traffic rate $y_l(t)$ and queue length $q_l(t)$ at the link at time t. Every RTT, d, RCP updates R_l at each link as follows:

$$R_l(t) = R_l(t-d)\left(1 + \frac{\alpha(C_l - y_l(t)) - \beta\frac{q_l(t)}{d}}{C_l}\right),$$

ATM/ Charny etc.



Discussion

- Fundamentally any limit on how fast we can get max-min rates? Explicit or implicit whatever