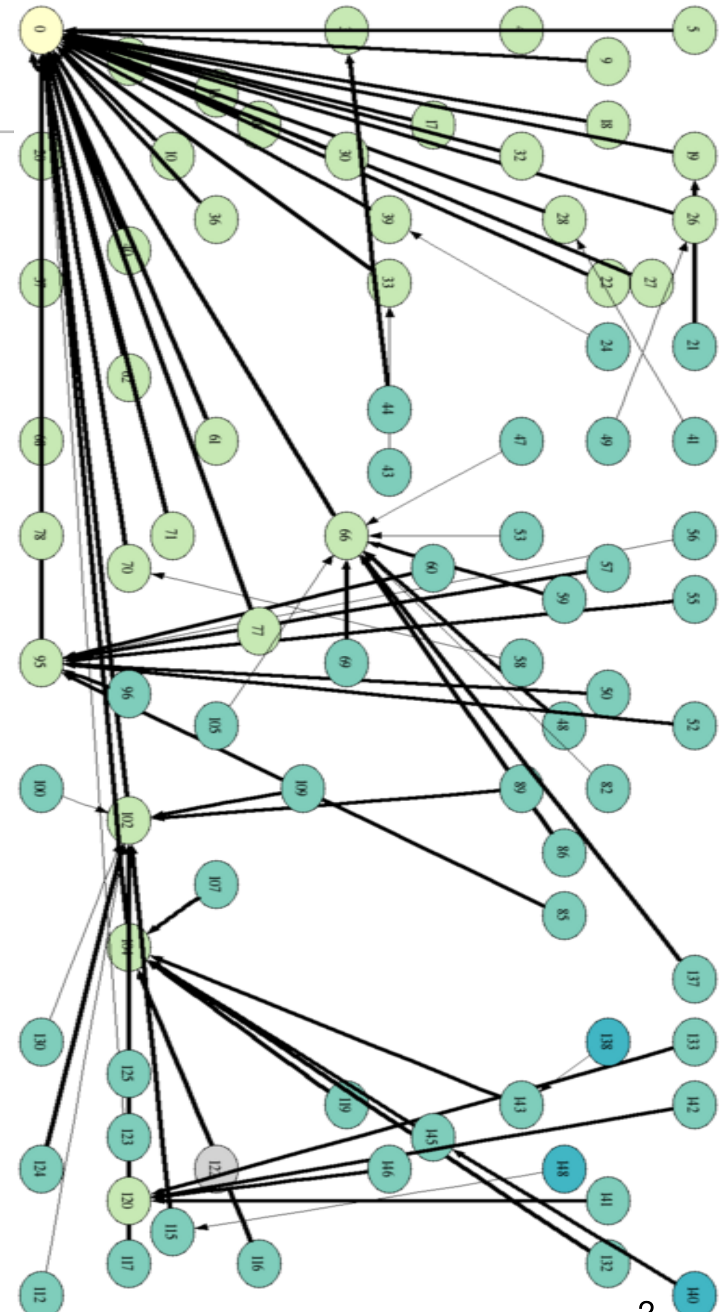
Routing:
Collection Tree Protocol

Original slides by Omprakash Gnawali

Collection

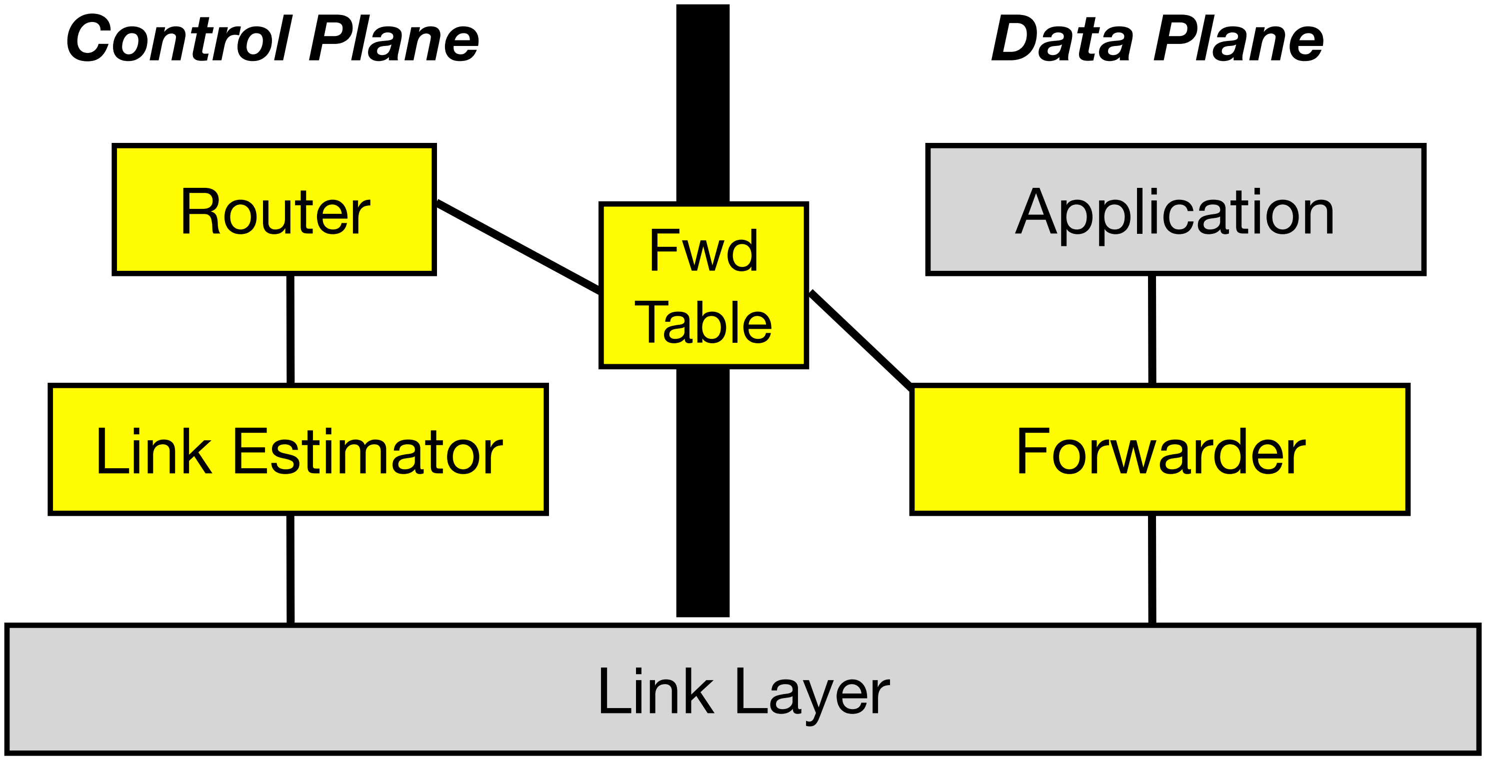
- **Anycast route to the sink(s)**
  - collects data from the network to a small number of sinks
  - network primitive for other protocols
- **A distance vector protocol**



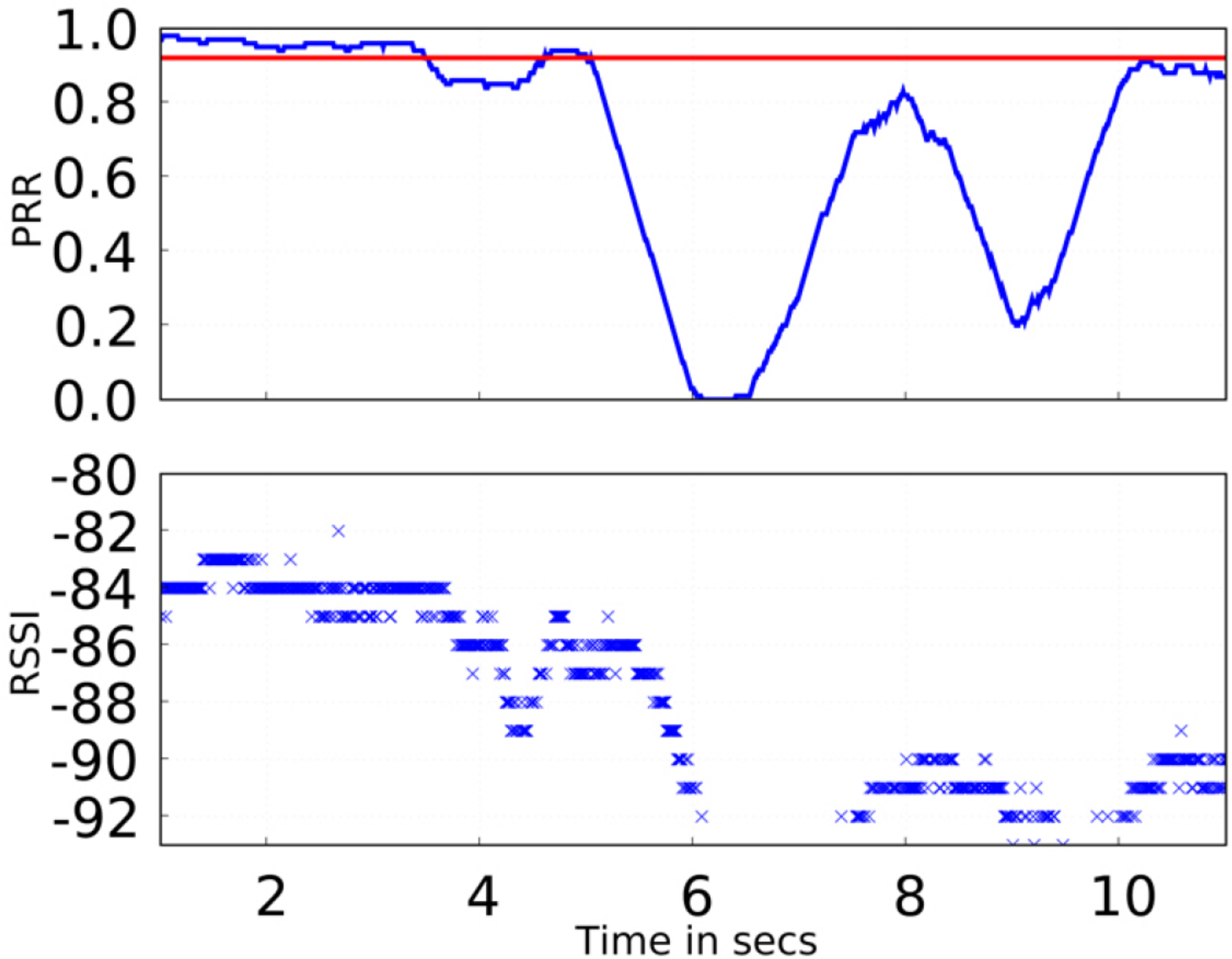
Why focus on a few sinks?

distance vector vs link state

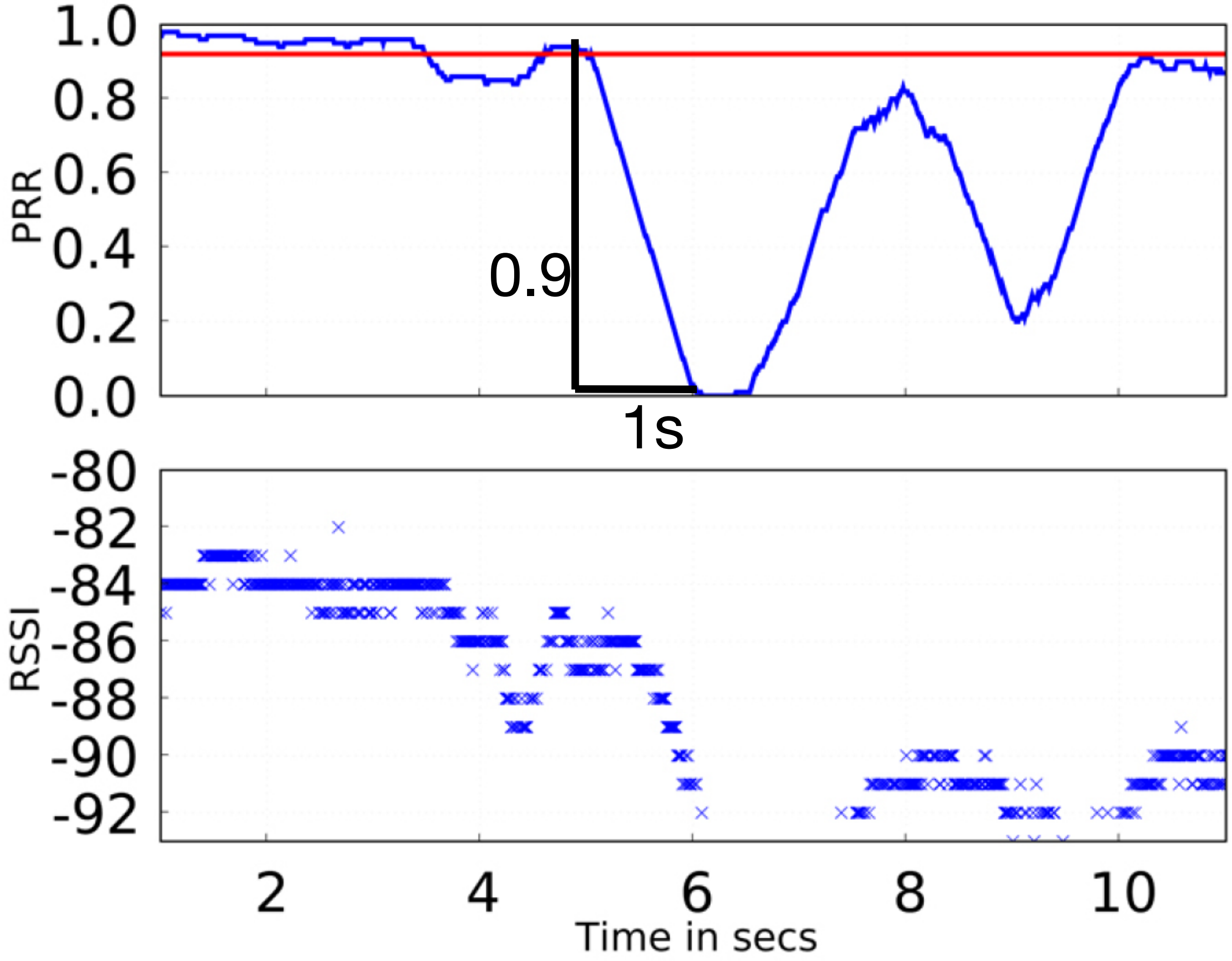
Common Architecture



Wireless Link Dynamics

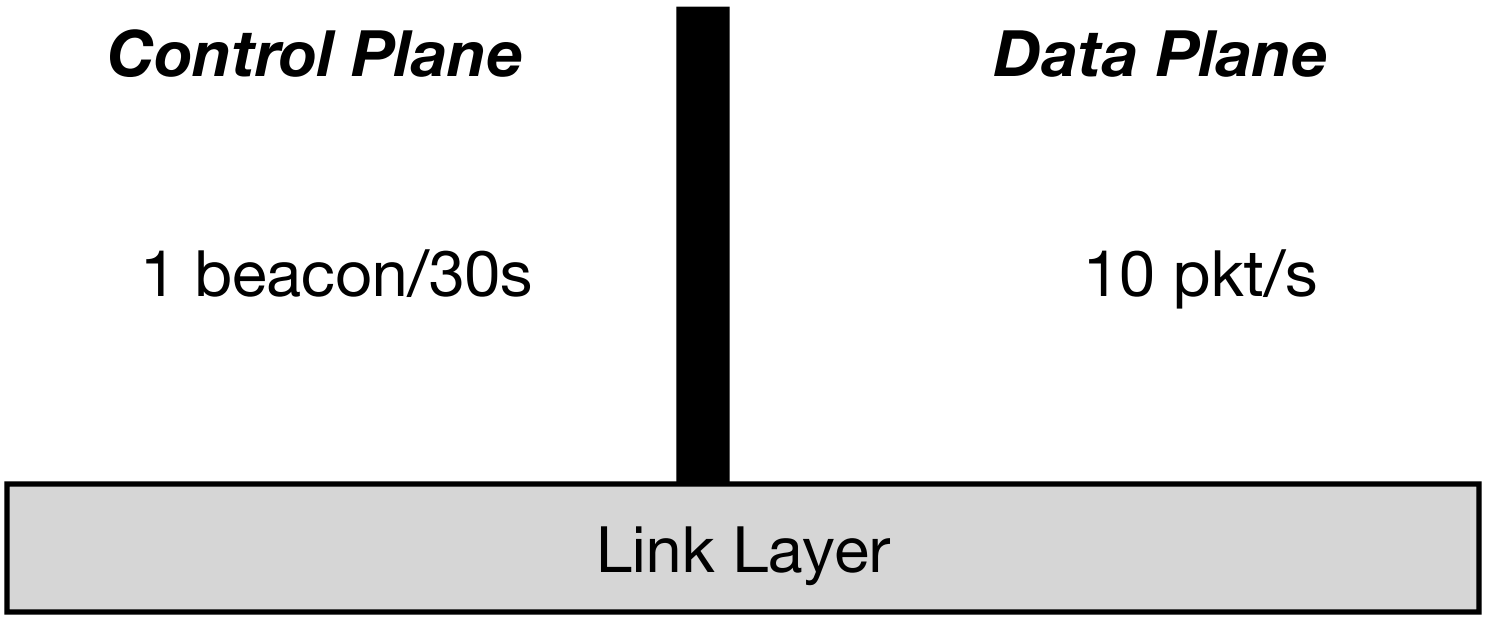


Wireless Link Dynamics



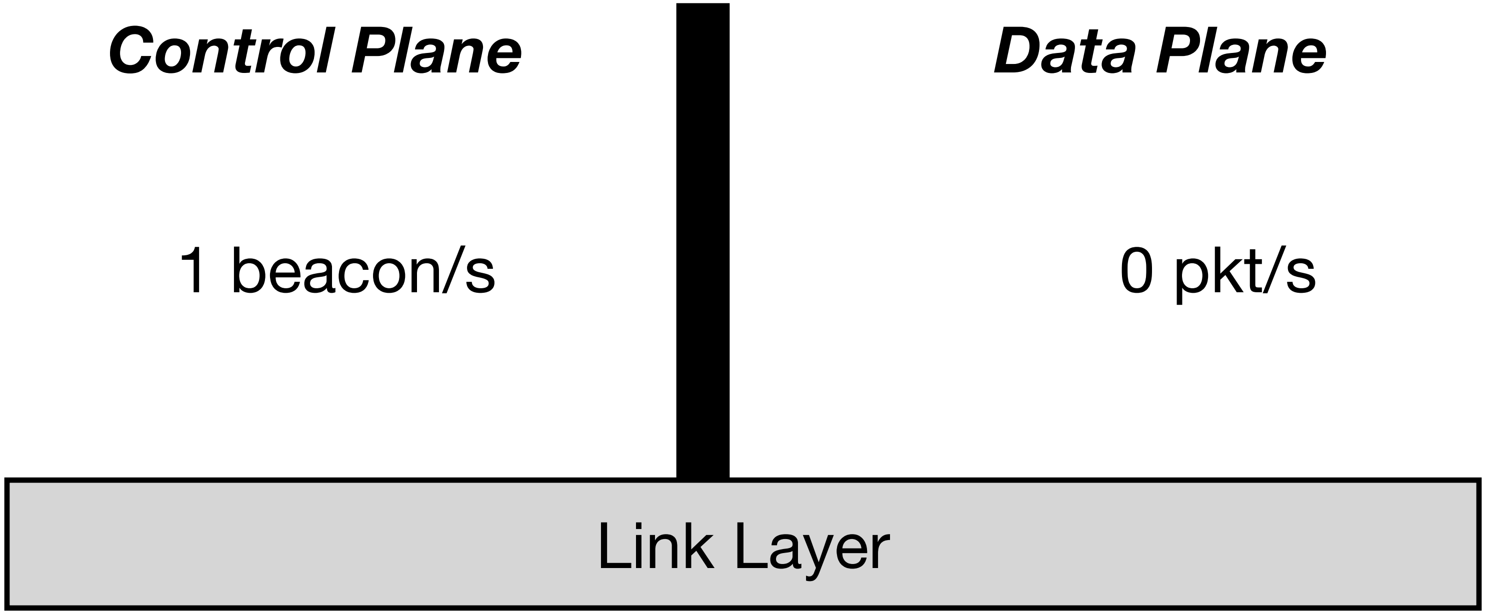
Control and Data Rate Mismatch

- **Can lead to poor performance**

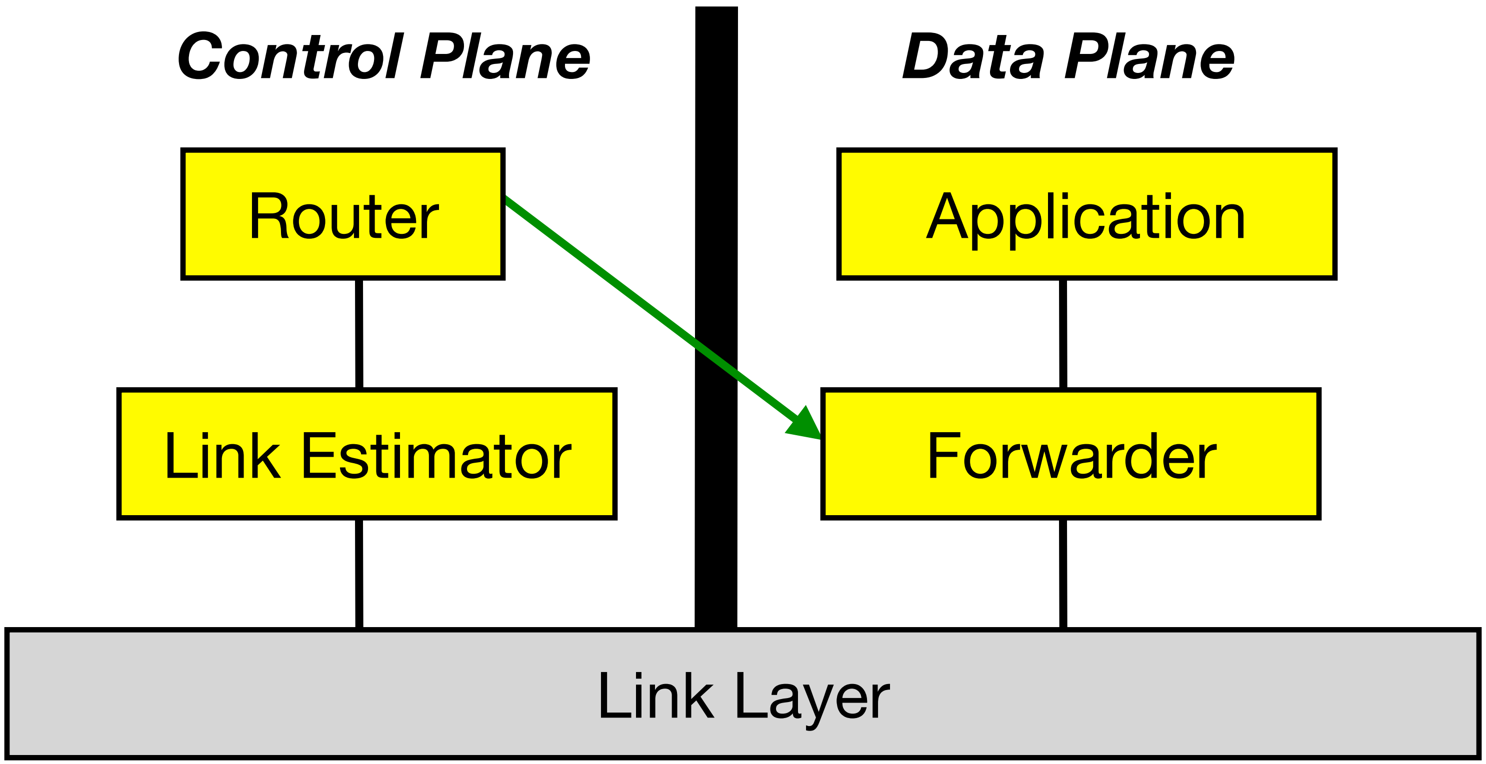


Control and Data Rate Mismatch

- Can lead to poor performance

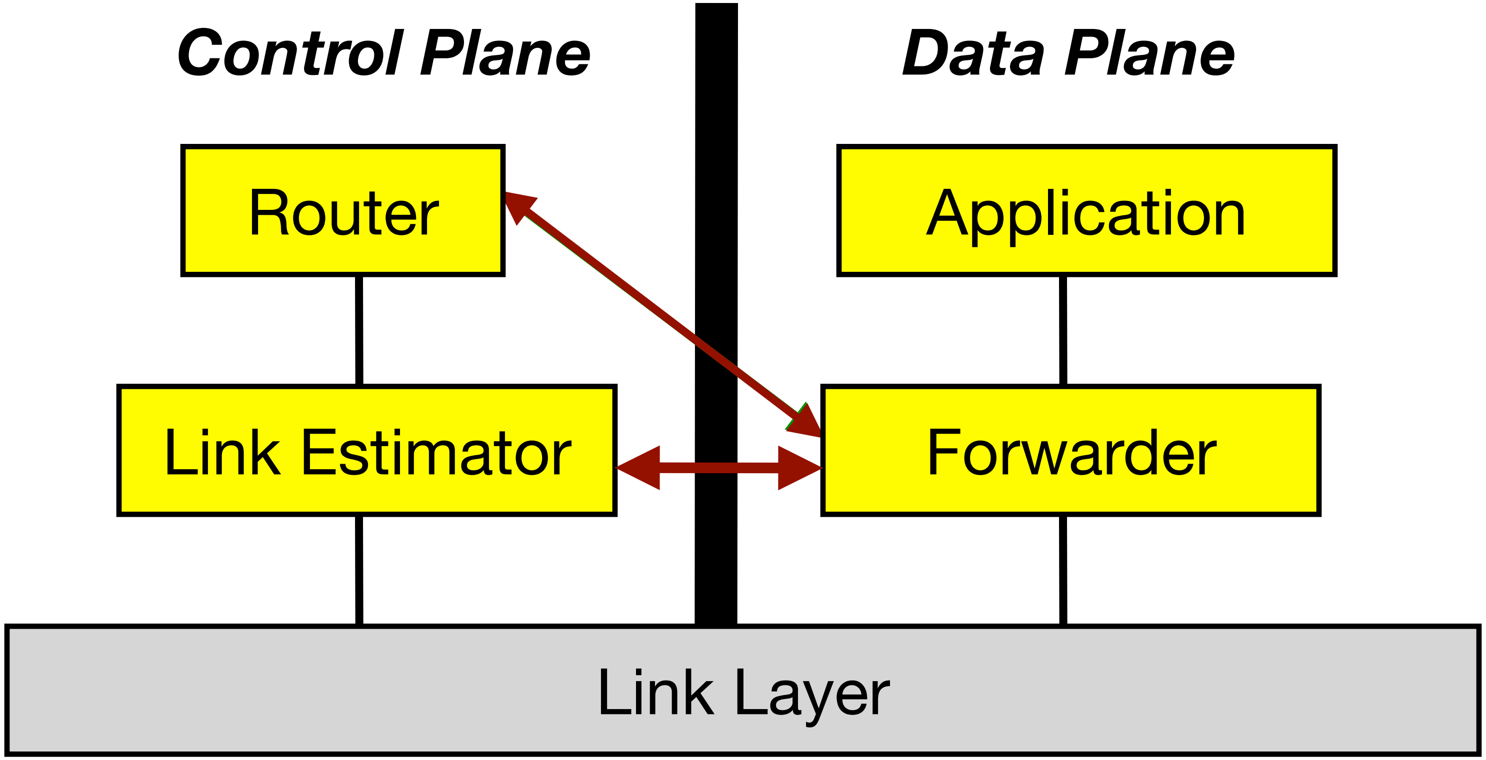


CTP Noe



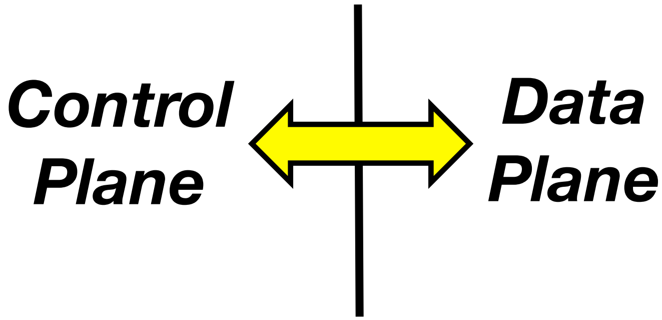
Control Plane
Data Plane
Router
Application
Link Estimator
Forwarder
Link Layer

CTP Noe


Control Plane
Data Plane
Router
Application
Link Estimator
Forwarder
Link Layer

CTP Noe's Approach

- **Enable control and data plane interaction**
- **Two mechanisms for efficient and agile topology maintenance**
  - datapath validation
  - adaptive beaconing



Outline

- **Control plane**
  - datapath validation
  - adaptive beacons
- **Data plane**
  - queuing
  - transmit time
  - cache
- **Evaluation**
- **Conclusion**

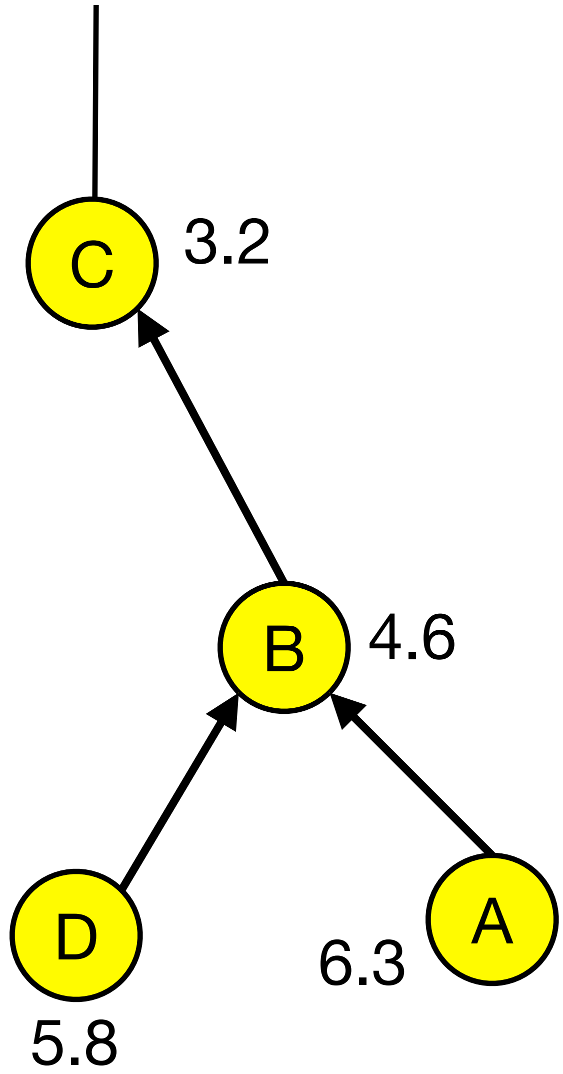
Data path validation

Datapath validation

- **Use data packets to validate the topology**
  - inconsistencies
  - loops
- **Receiver checks for consistency on each hop**
  - transmitter's cost is in the header
- **Same time-scale as data packets**
  - validate only when necessary

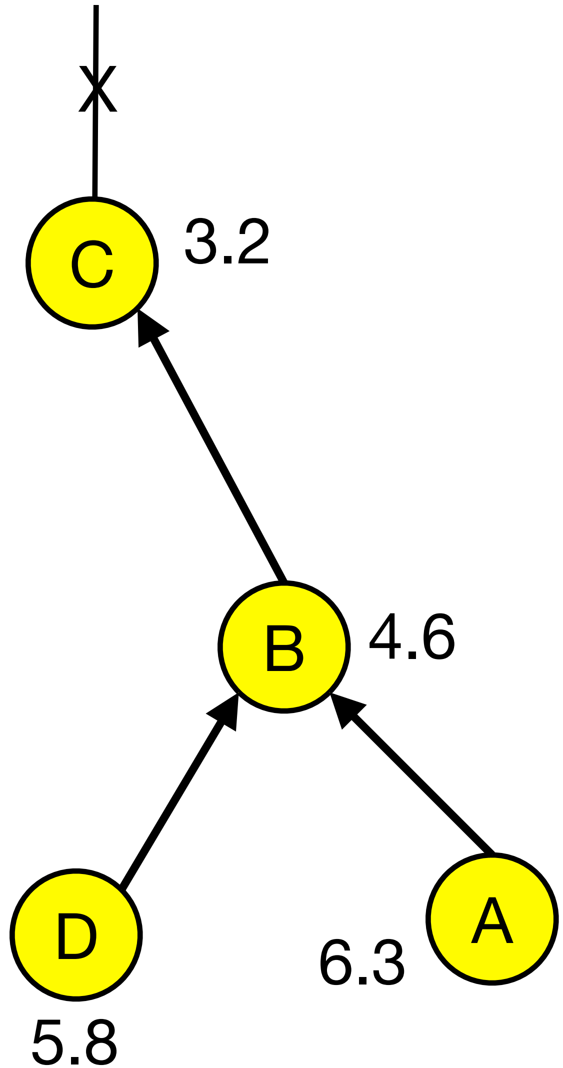
Routing Loops

- **Cost does not decrease**



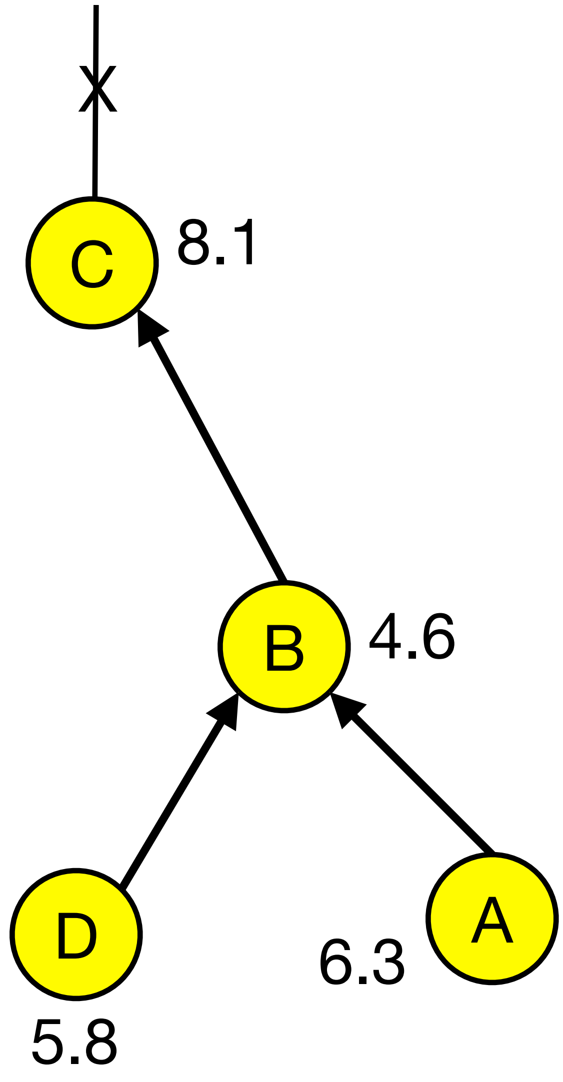
Routing Loops

- **Cost does not decrease**



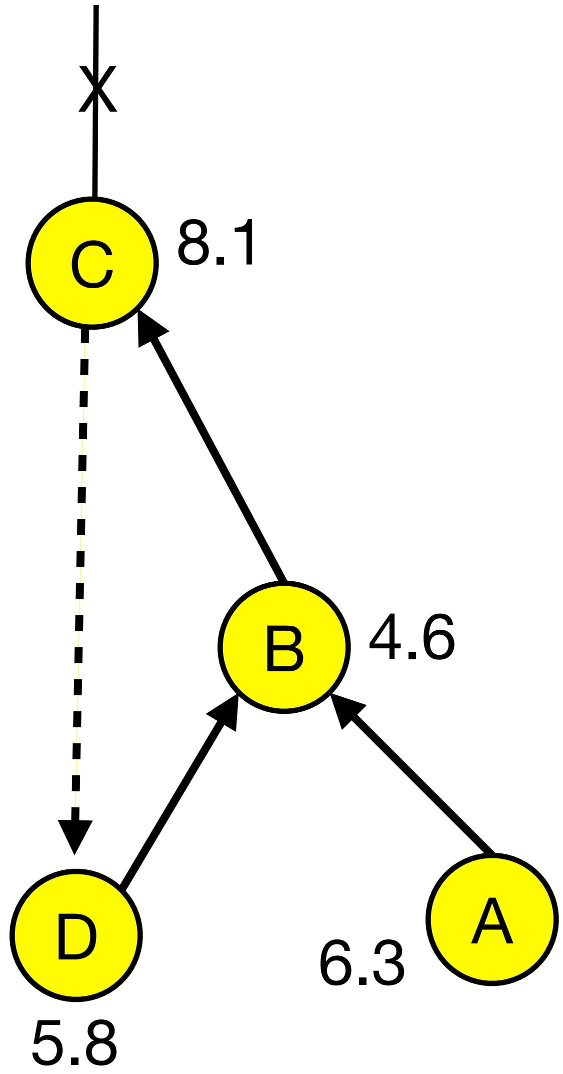
Routing Loops

- **Cost does not decrease**



Routing Loops

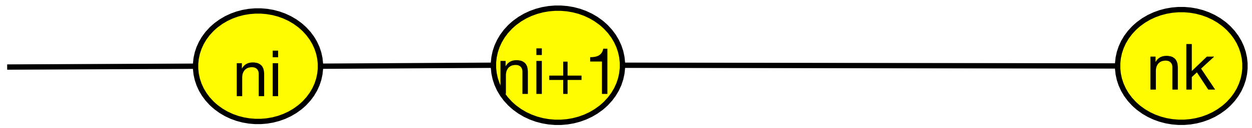
- Cost does not decrease



Routing Consistency

- **Next hop should be closer to the destination**
- **Maintain this consistency criteria on a path**

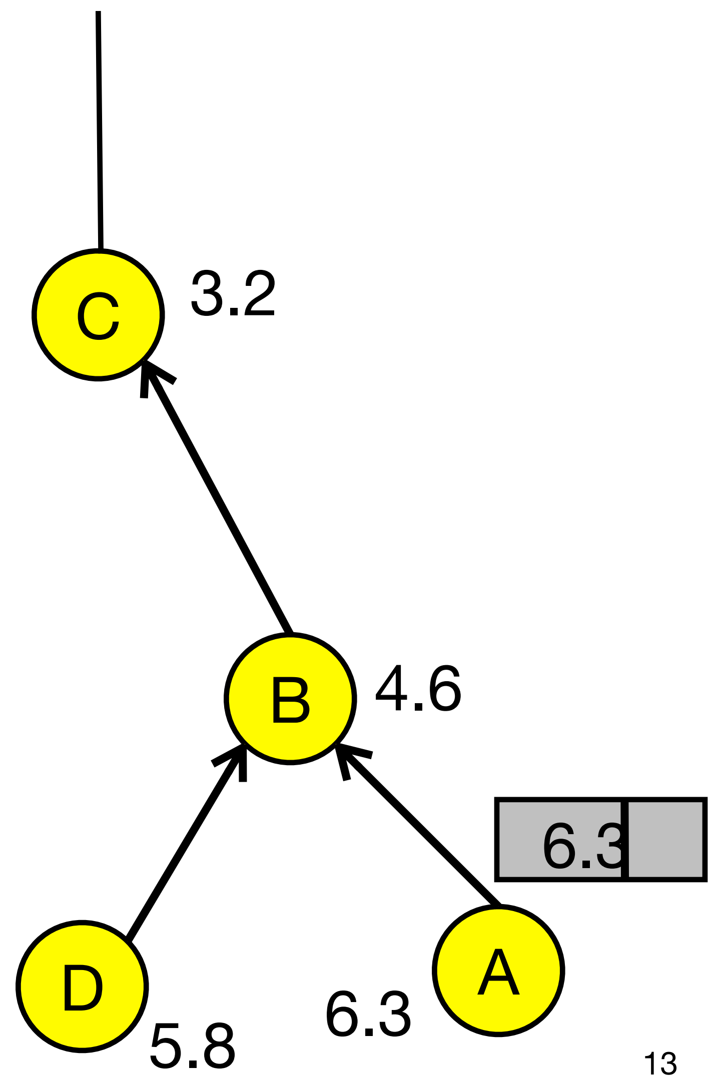
$$\forall i \in \{0, k-1\},\ ETX(n_i) > ETX(n_{i+1})$$



- **Inconsistency due to stale state**

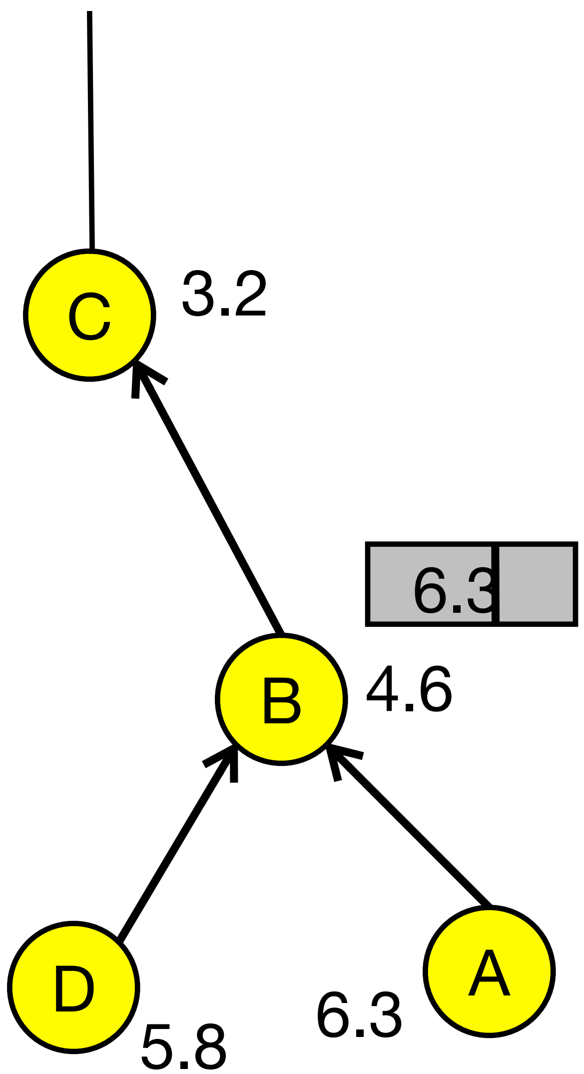
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



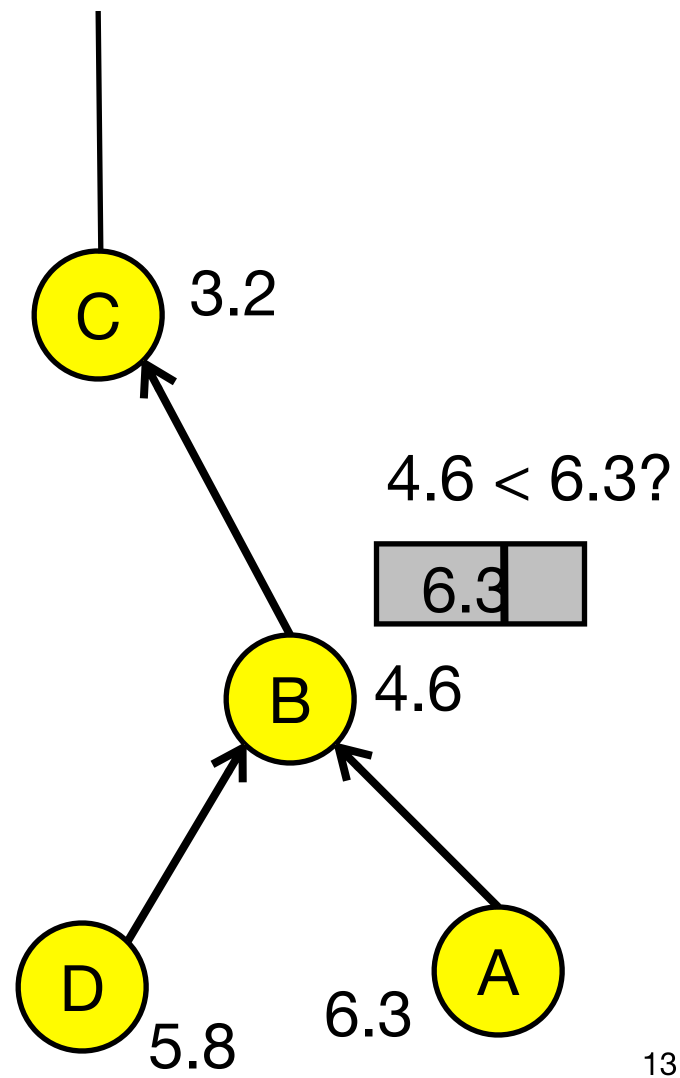
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



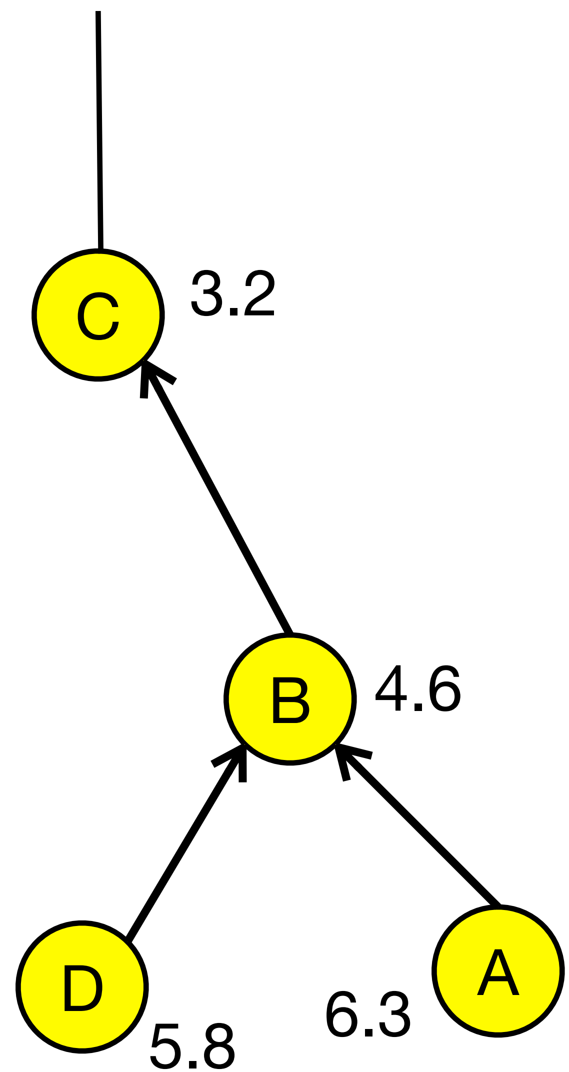
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



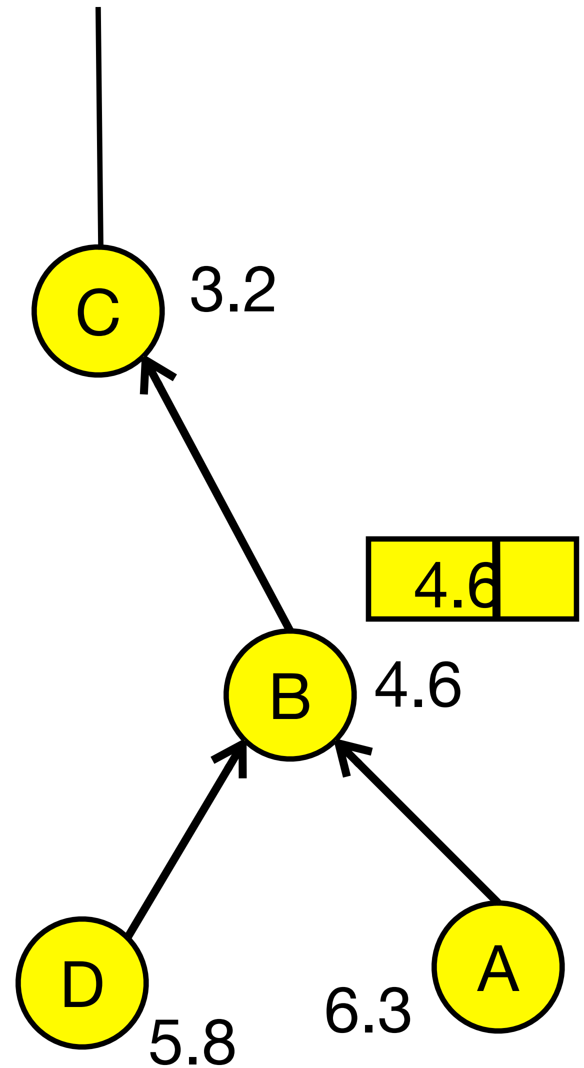
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



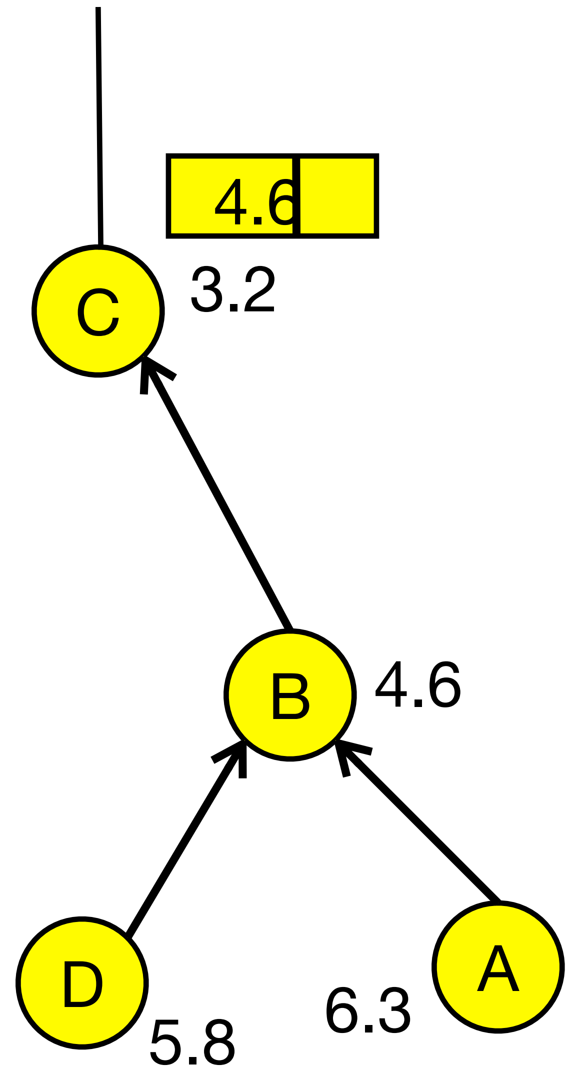
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



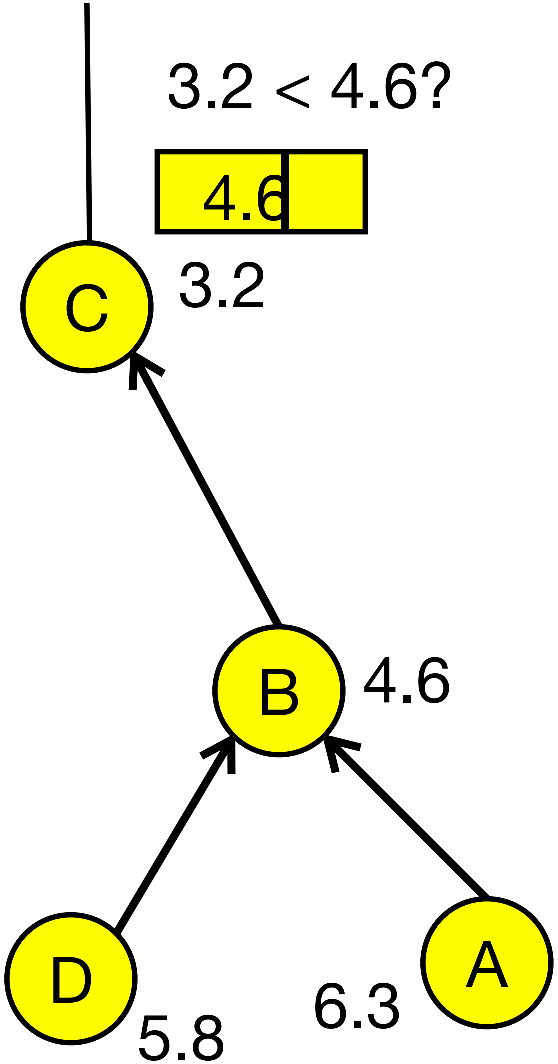
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don’t drop the packets
  - signal the control plane



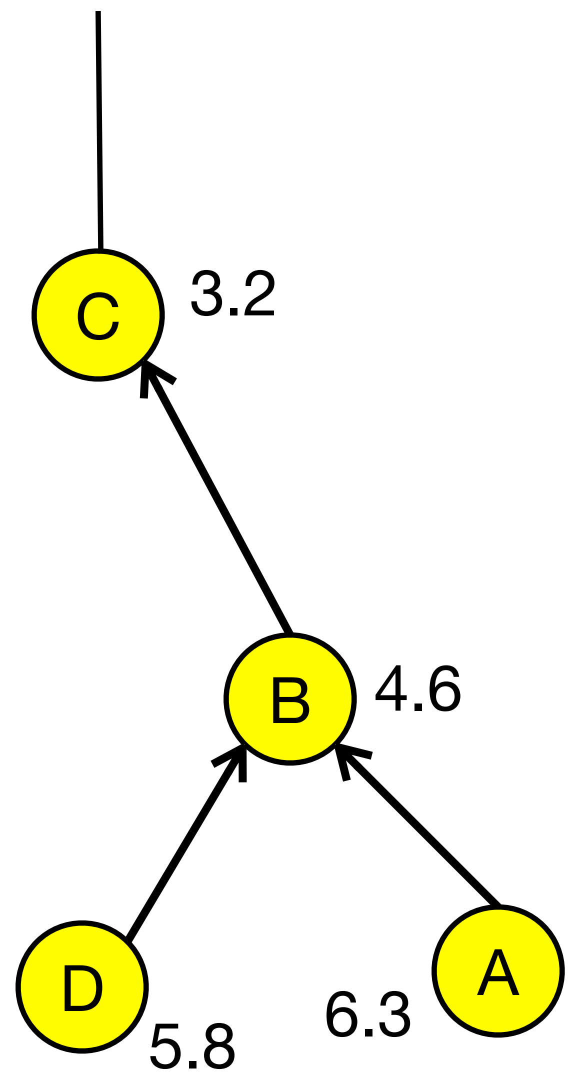
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



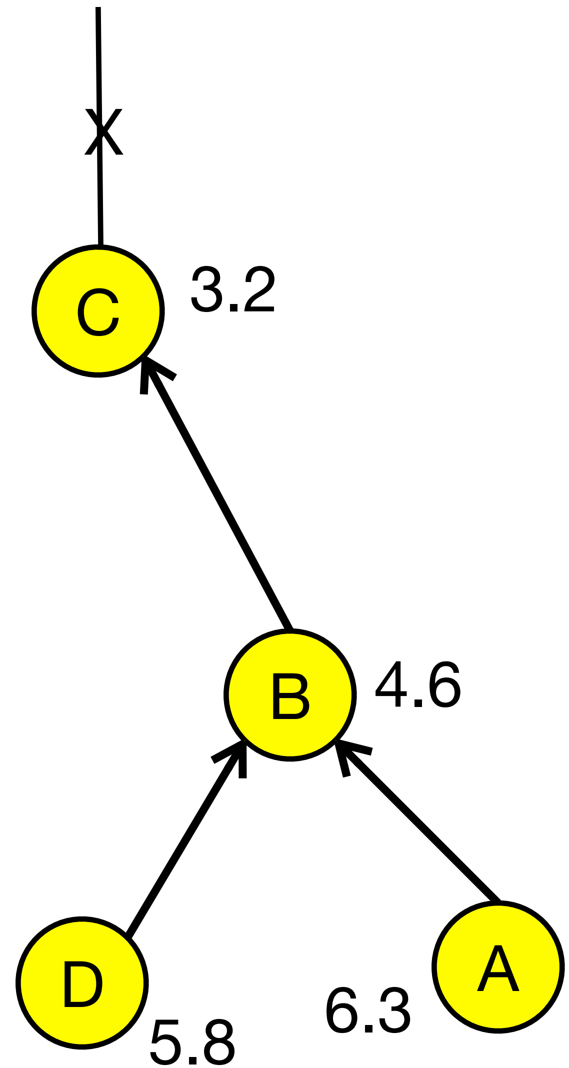
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don’t drop the packets
  - signal the control plane



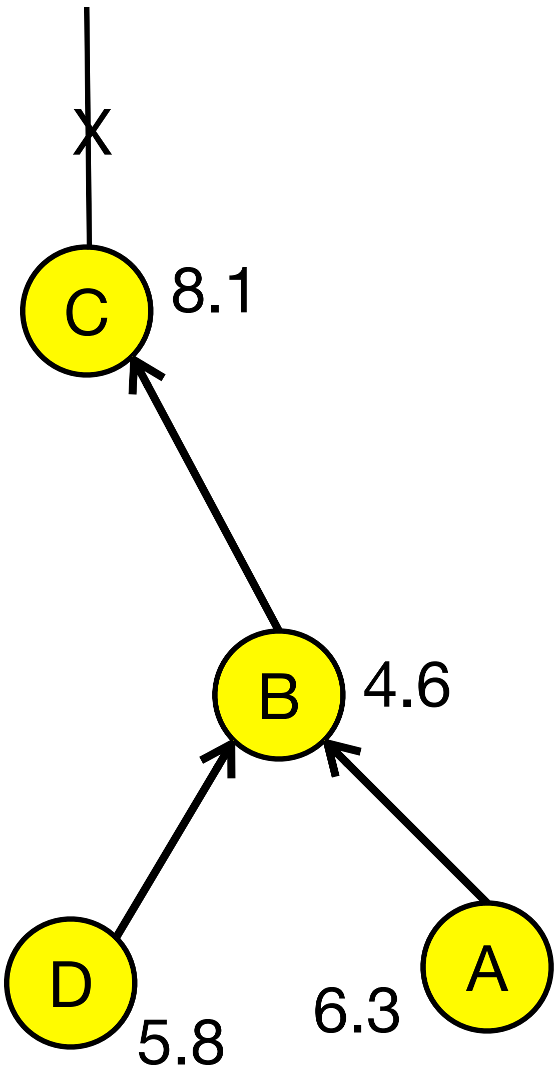
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



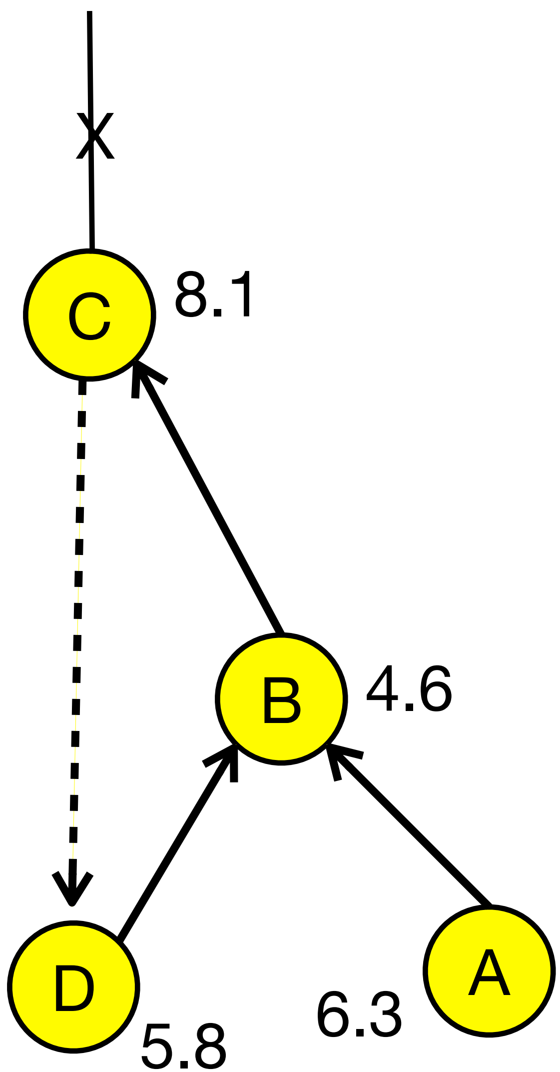
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



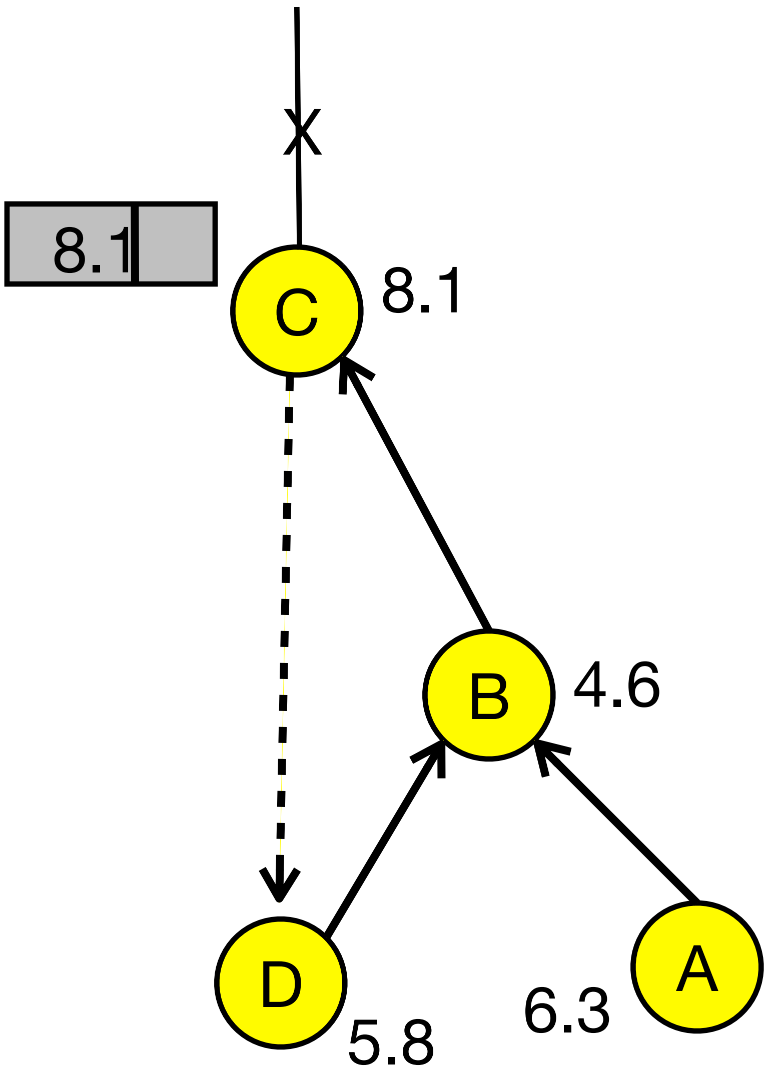
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



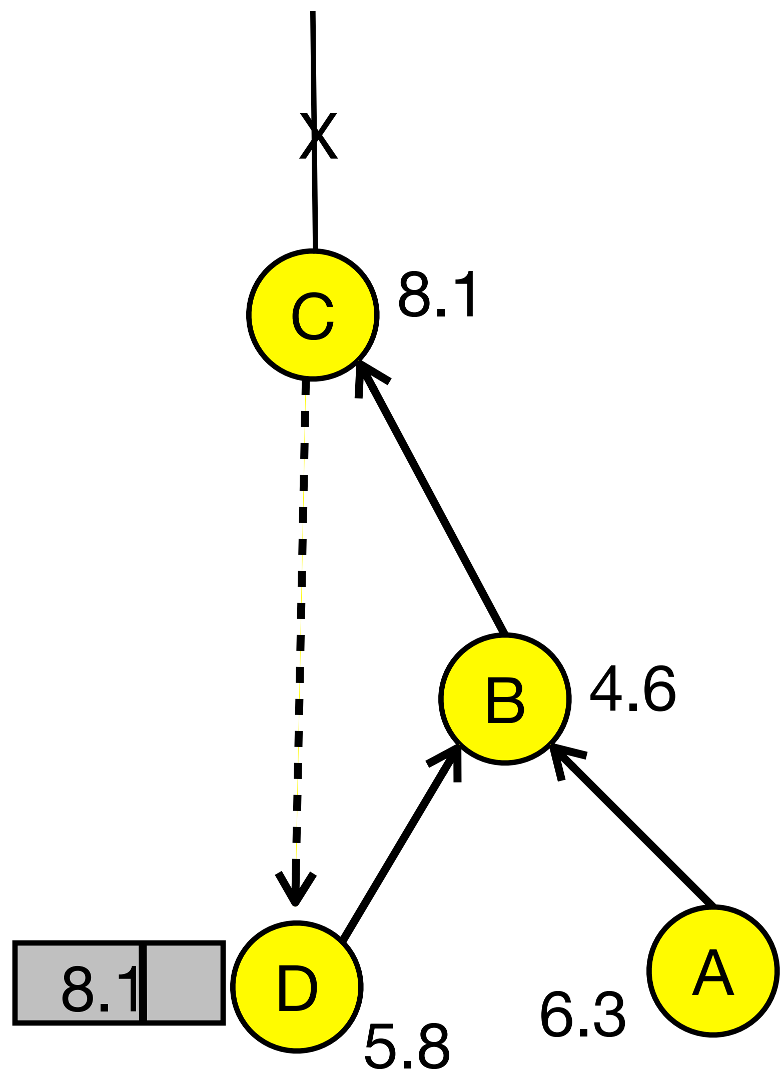
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don’t drop the packets
  - signal the control plane



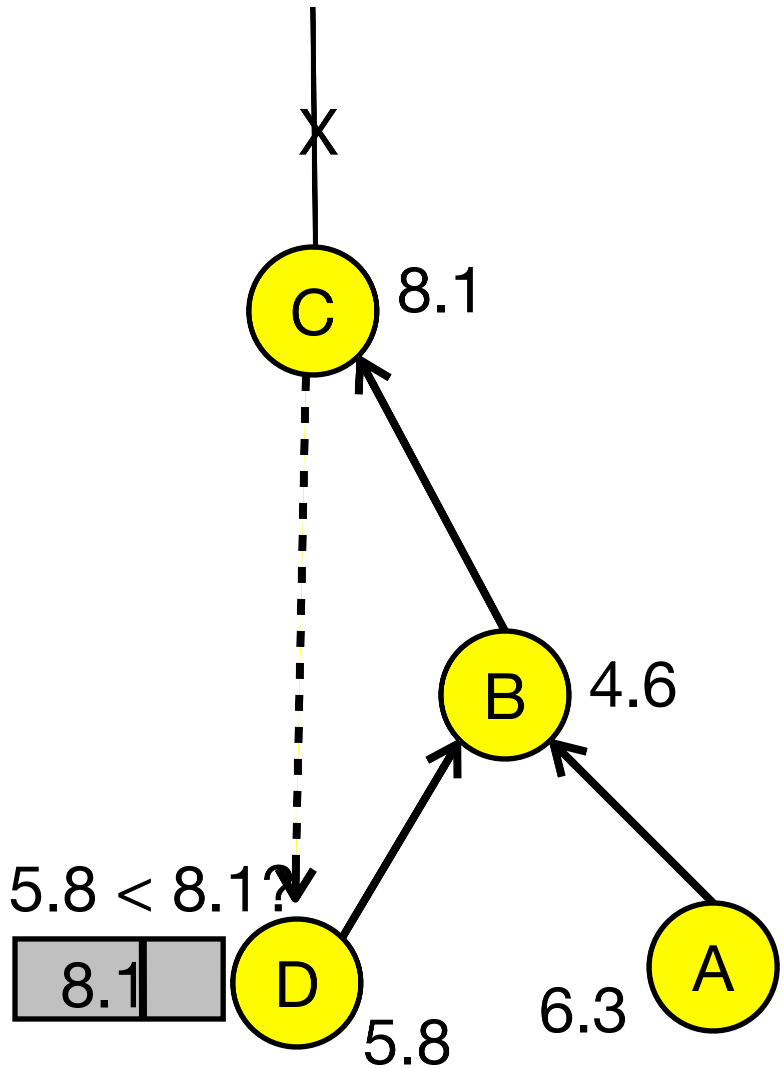
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



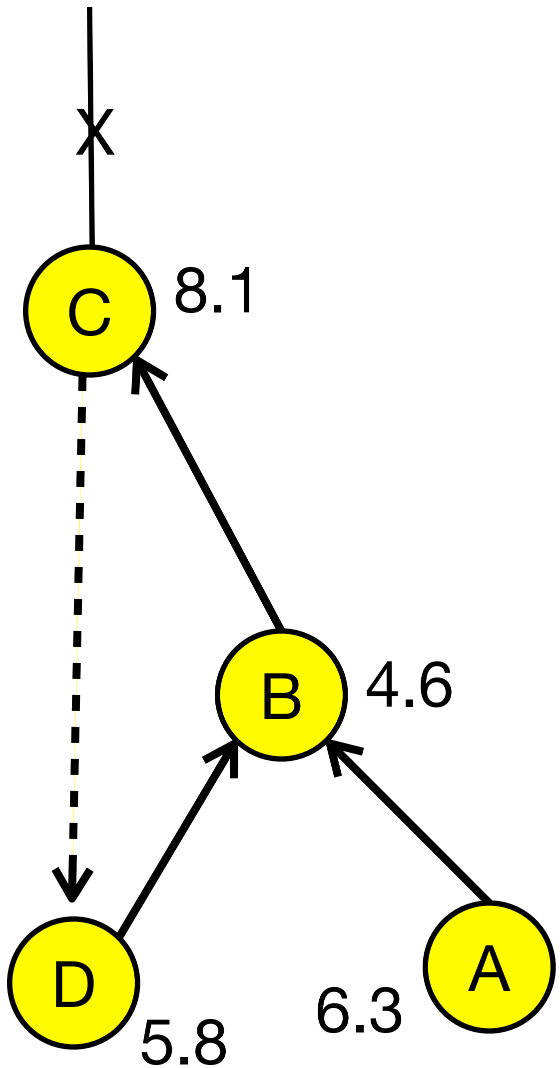
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don’t drop the packets
  - signal the control plane



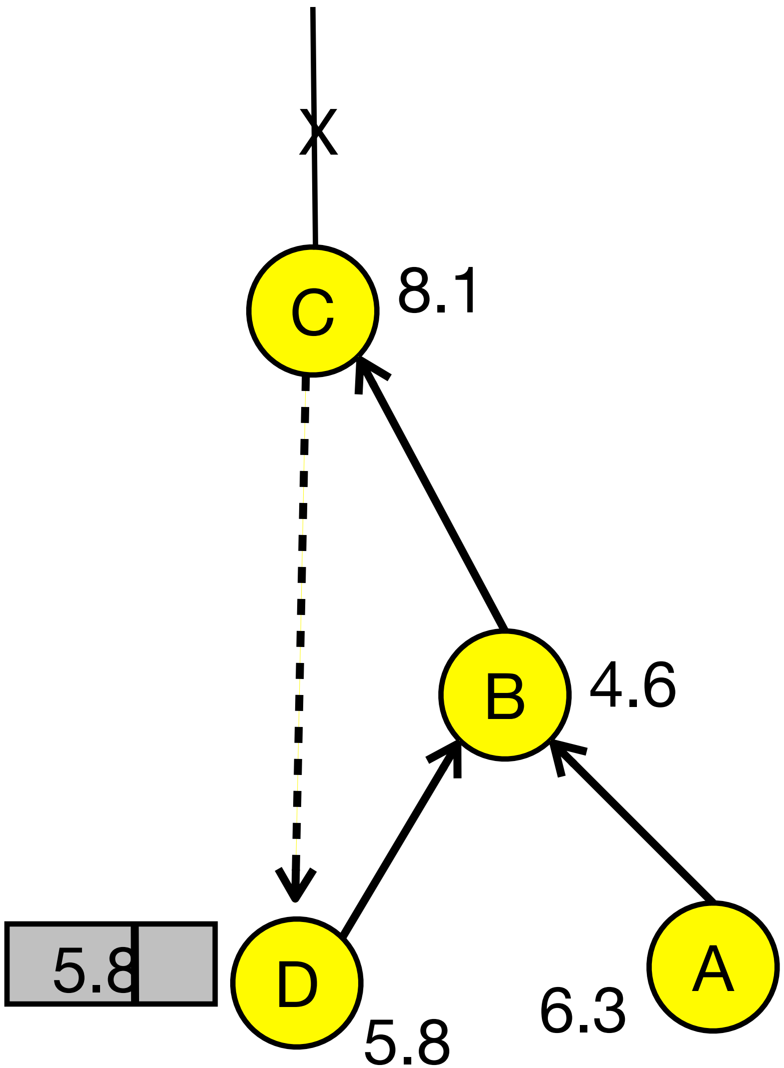
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



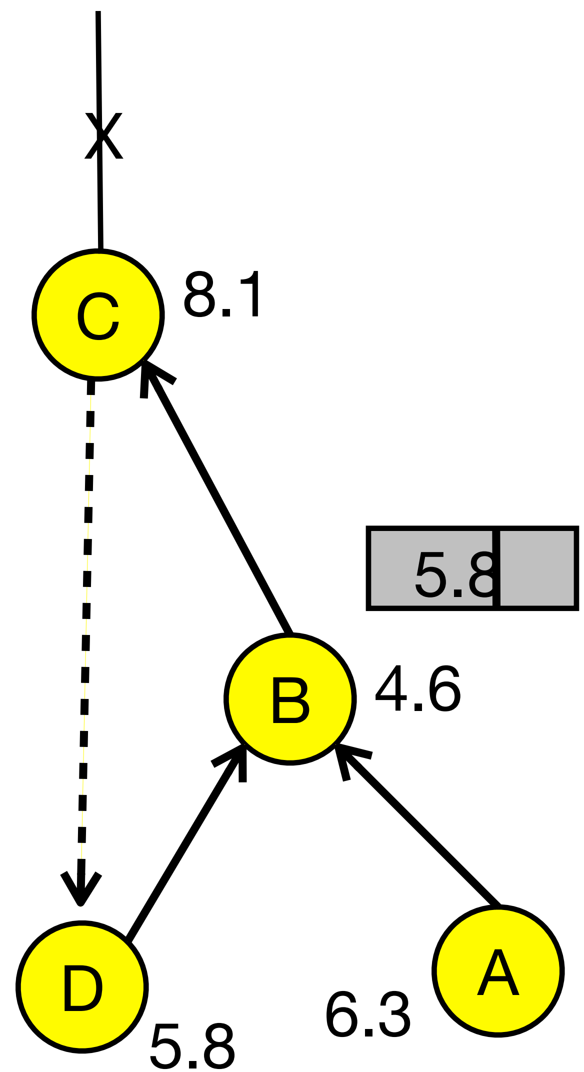
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



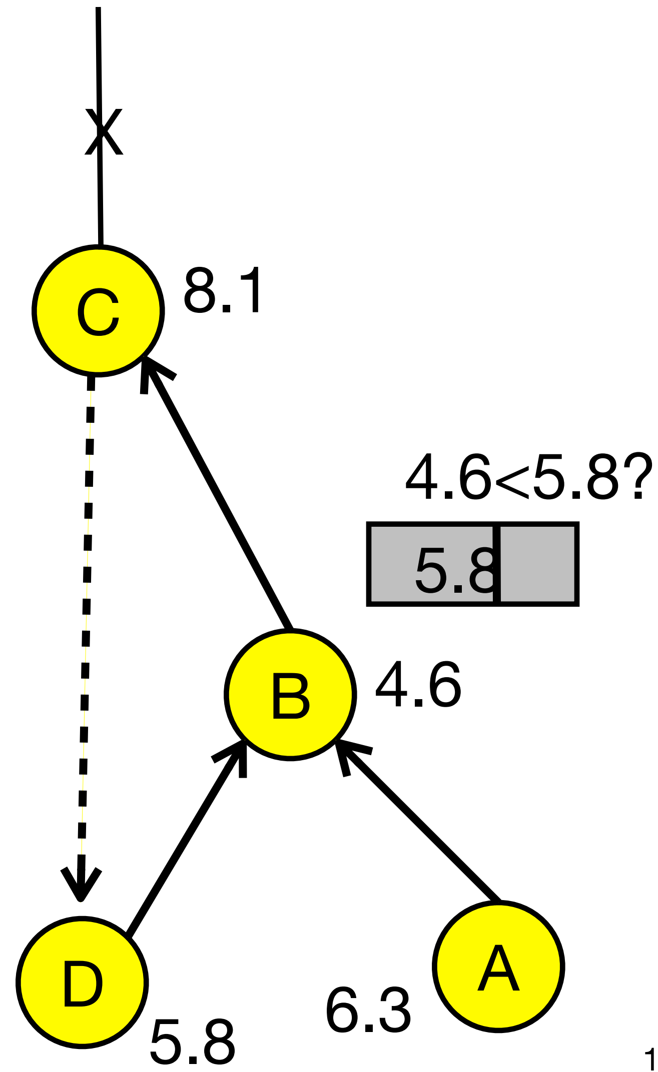
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



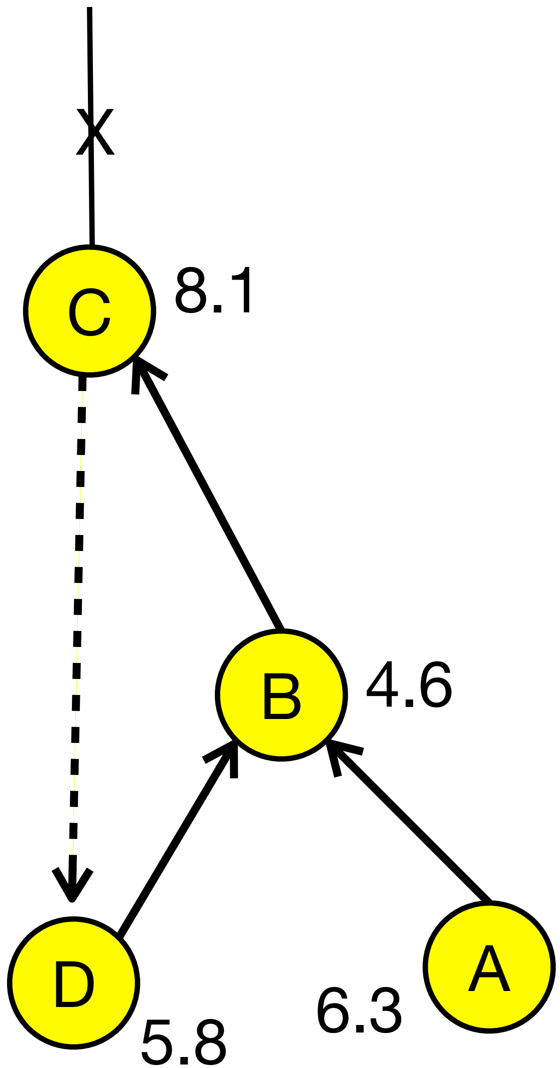
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don’t drop the packets
  - signal the control plane



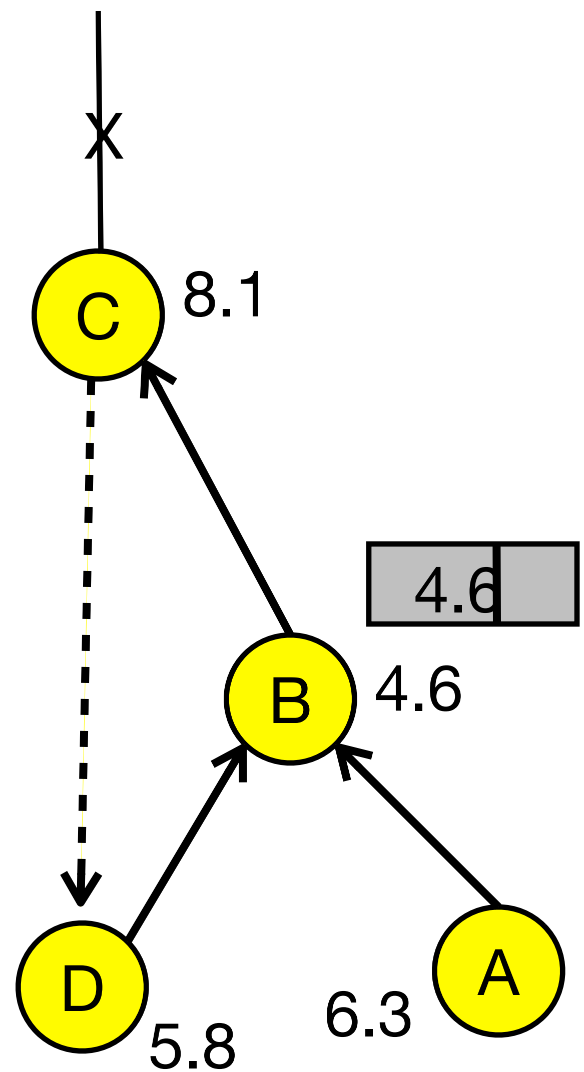
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



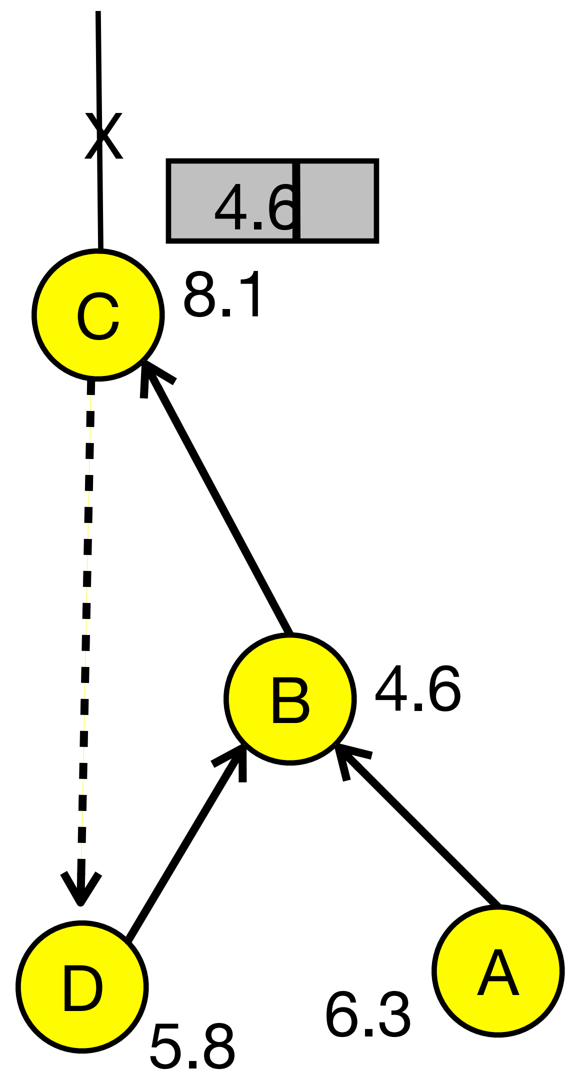
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



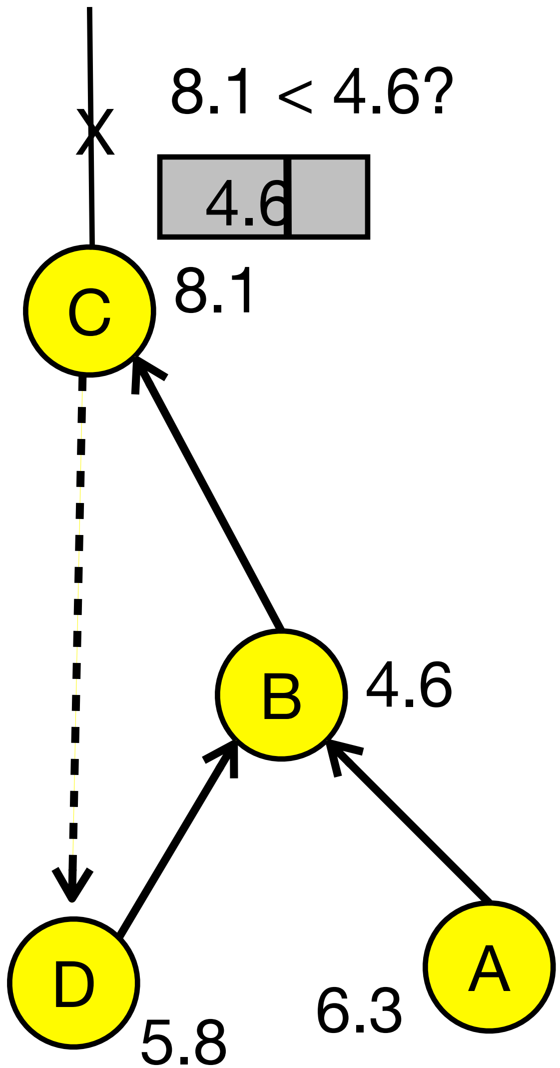
Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



Detecting Routing Loops

- **Datapath validation**
  - cost in the packet
  - receiver checks
- **Inconsistency**
  - larger cost than on the packet
- **On Inconsistency**
  - don't drop the packets
  - signal the control plane



Routing Consistency

How Fast to Send Beacons?

- **Using a fixed rate beacon interval**
  - Can be too fast
  - Can be too slow
  - Agility-efficiency tradeoff
- **Agile+Efficient possible?**

Routing as Consistency

- **Routing as a consistency problem**
  - costs along a path must be consistent
- **Use consistency protocol in routing**
  - leverage research on consistency protocols
  - trickle

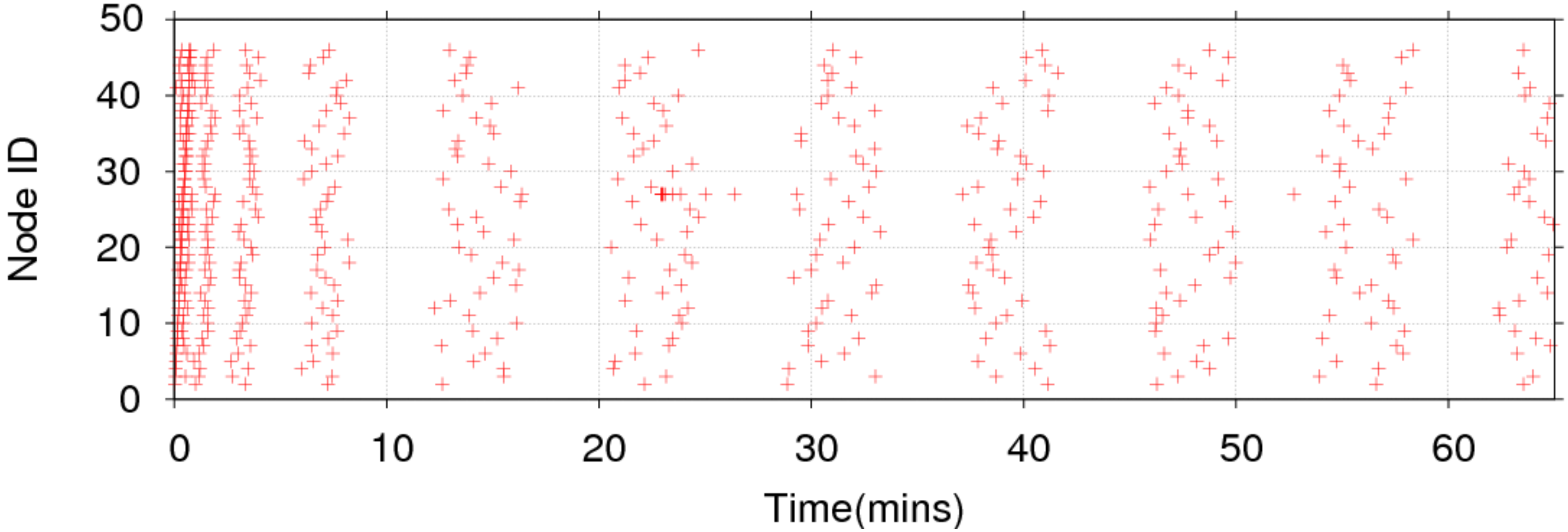
Trickle

- **Detecting inconsistency**
  - code propagation: version number mismatch
  - does not work for routing: use path consistency
- **Control propagation rate**
  - start with a small interval
  - double the interval up to some max
  - reset to the small interval when inconsistent

Control Traffic Timing

- **Extend Trickle to time routing beacons**
- **Reset the interval**
  - ETX(receiver) >= ETX(sender)
  - significant decrease in gradient [found better link]
  - “Pull” bit - no valid route

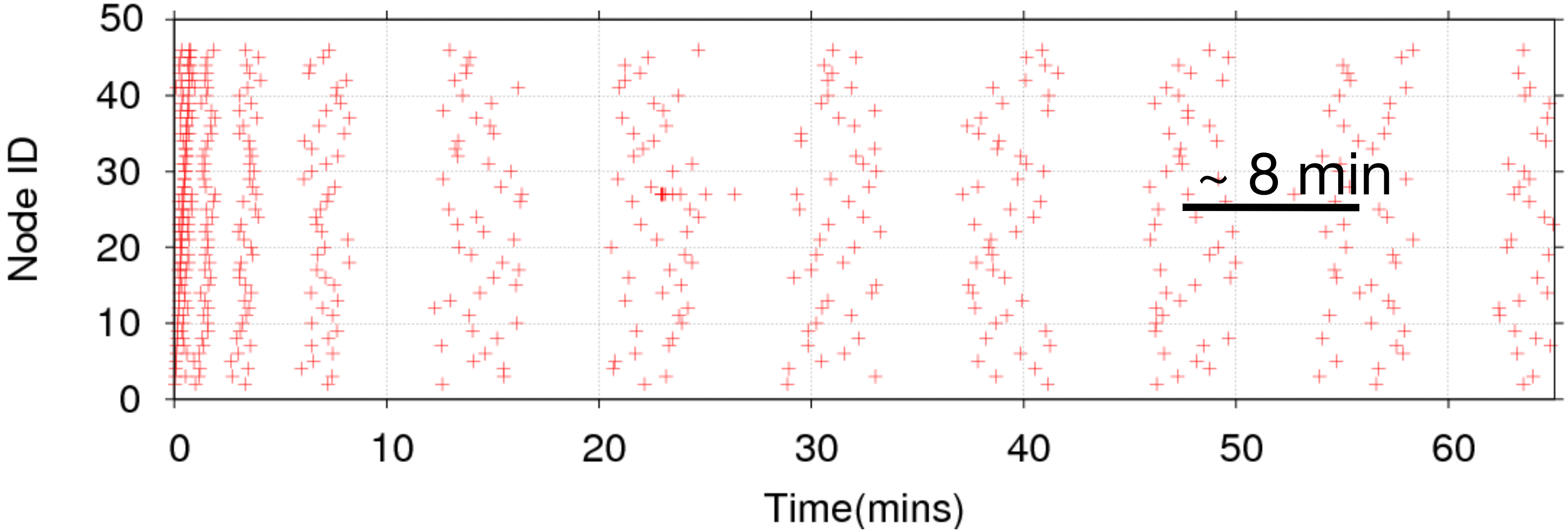
Adaptive Beacon Timing



Tutornet

Infrequent beacons in the long run

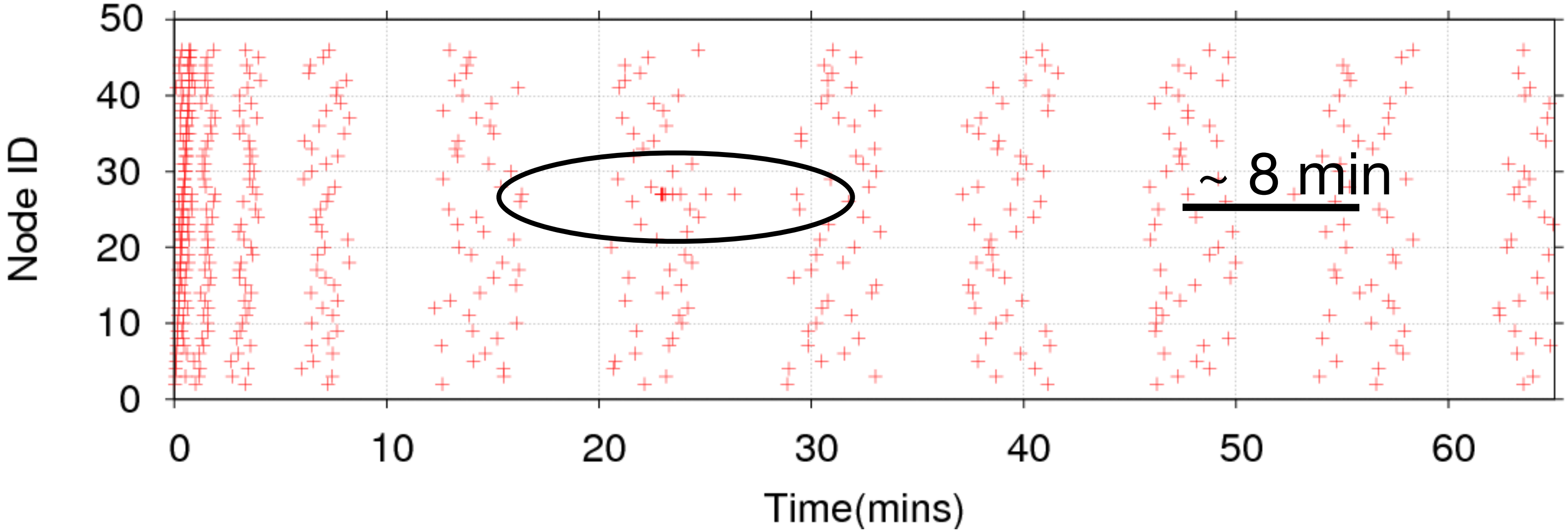
Adaptive Beacon Timing



Tutornet

Infrequent beacons in the long run

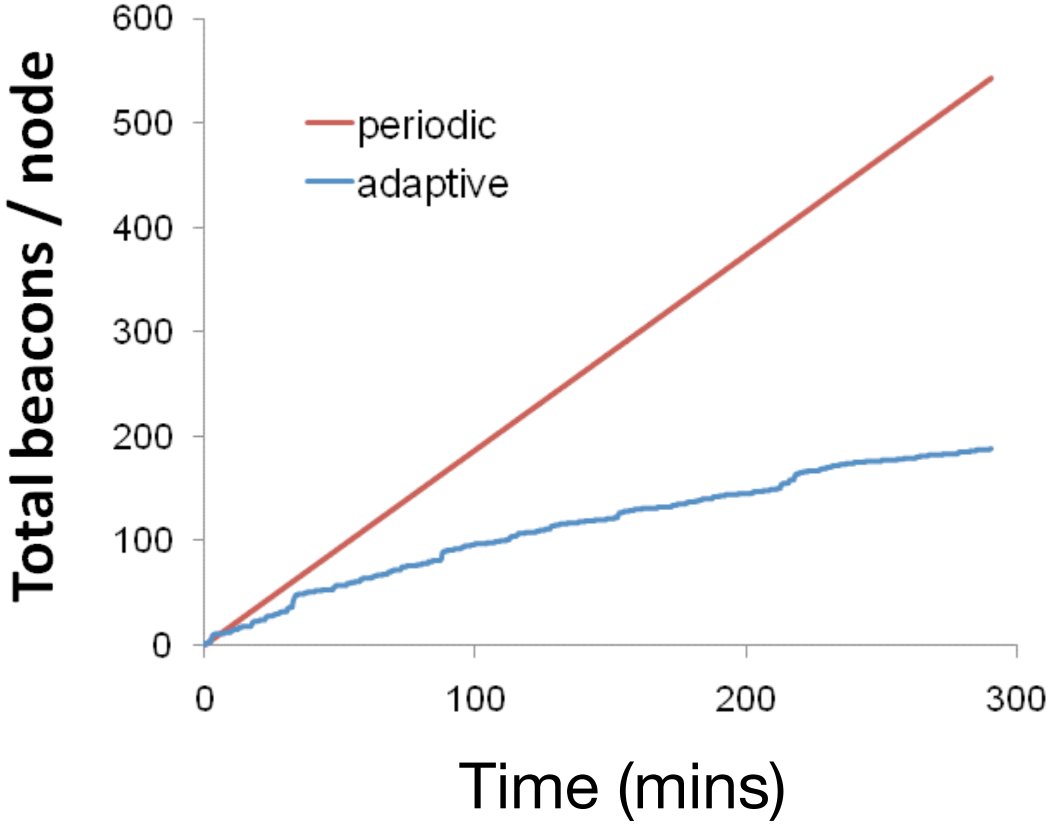
Adaptive Beacon Timing



Tutornet

Infrequent beacons in the long run

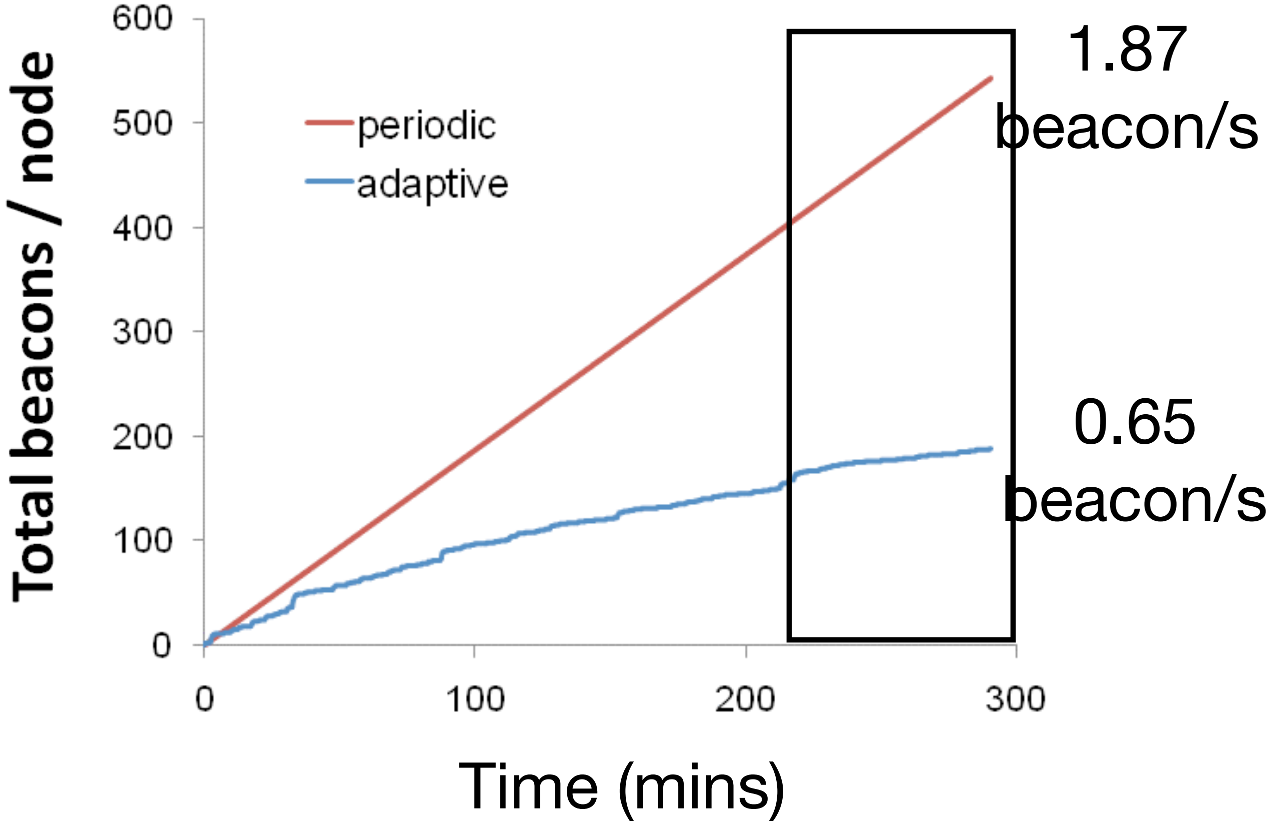
Adaptive vs Periodic Beacons



Tutornet

Less overhead compared to 30s-periodic

Adaptive vs Periodic Beacons

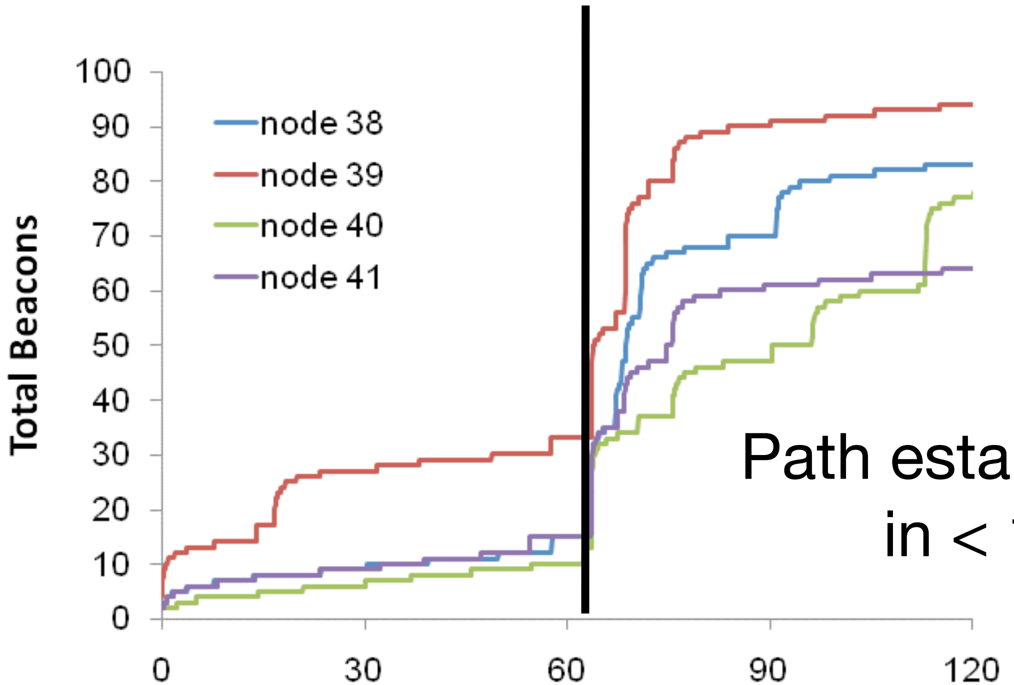


Tutornet

Less overhead compared to 30s-periodic

Node Discovery

A new node introduced



Path established in < 1s

Tutornet

Time (mins)

Efficient and agile at the same time

Data Plane

Data plane

- **Goals: efficient, robust, and reliable forwarding**
- **Mechanisms**
  - per client queueing
  - hybrid send queue
  - transmit timer
  - transmit cache

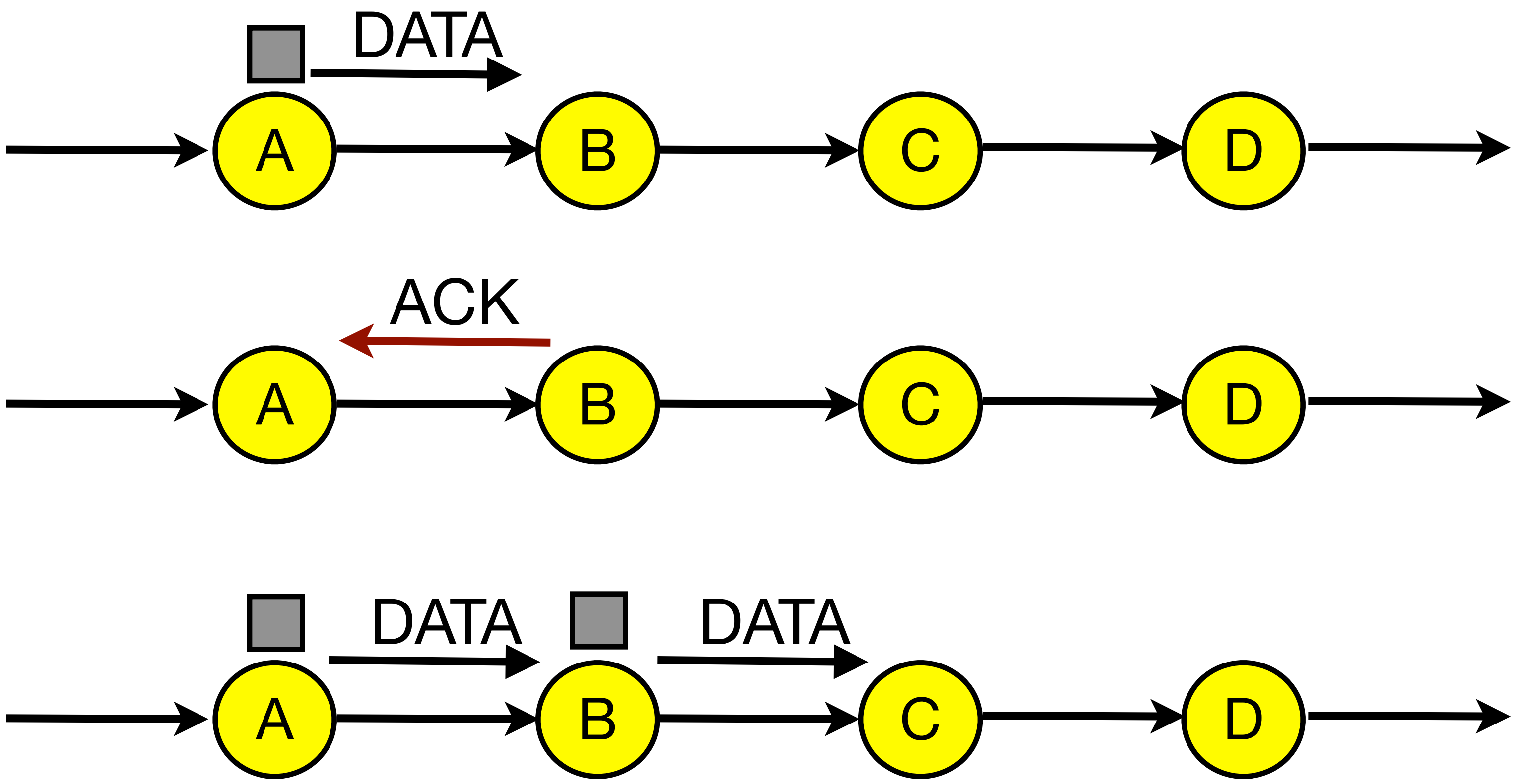
Data plane mechanisms

- **Queueing discipline**
  - Per-client queueing [top-level]
    - each client may have one outstanding packet
    - achieves better fairness than a shared queue
  - Hybrid send queue [lower-level]
    - contains both route-through and locally-generated traffic
    - duplicate packets are dropped [i.e., not inserted in the queue]

- **Transmission Cache**
  - for each transmitted packet insert (src, seq, THL)
  - determine if a packet is duplicate

Transmit Timer

- **Self-interference between packets may be a problem**



Increased likelihood of collisions

Transmit Timer

- **Rate control: delay the transmission of packets**
  - the transmission of consecutive packets is randomized between (1.5, 2.5) packet times
  - Is this good enough?

Evaluation

Experiments

- 12 testbeds
- 20-310 nodes
- 7 hardware platforms
- 4 radio technologies
- 6 link layers

Testbed	Platform	Nodes	Physical size m^2 or m^3
Tutornet (16)	Tmote	91	$50\times25\times10$
Wymanpark	Tmote	47	80×10
Motelab	Tmote	131	$40\times20\times15$
Kansei[a]	TelosB	310	40×20
Mirage	Mica2dot	35	50×20
NetEye	Tmote	125	6×4
Mirage	MicaZ	86	50×20
Quanto	Epic-Quanto	49	35×30
Twist	Tmote	100	$30\times13\times17$
Twist	eyesIFXv2	102	$30\times13\times17$
Vinelab	Tmote	48	60×30
Blaze[b]	Blaze	20	30×30

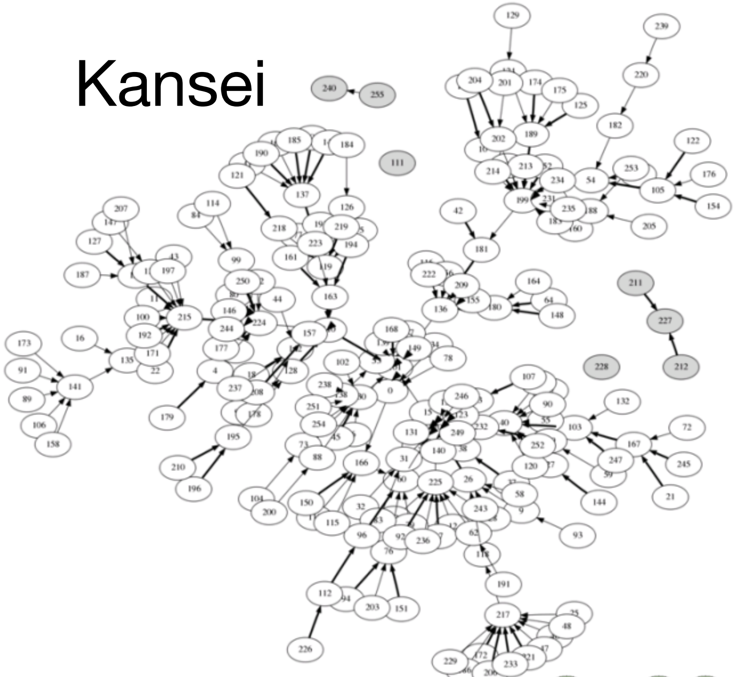
Variations in hardware, software, RF environment, and topology

Evaluation Goals

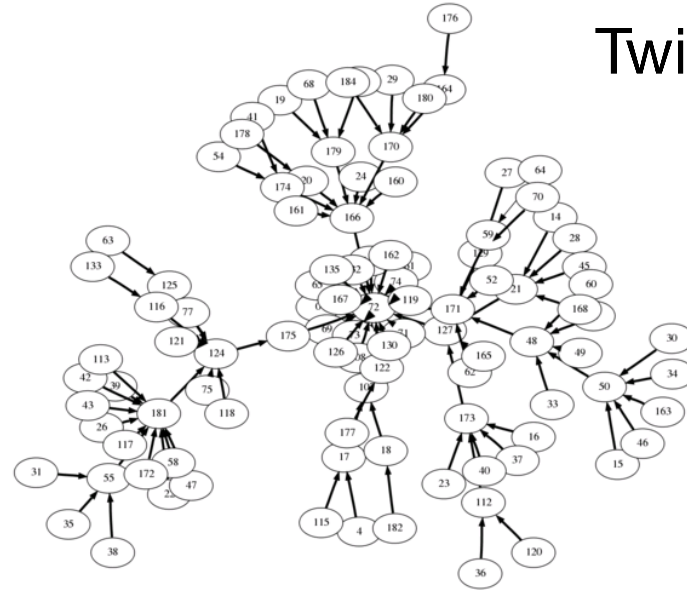
- **Reliable?**
  - Packets delivered to the sink
- **Efficient?**
  - TX required per packet delivery
- **Robust?**
  - Performance with disruption

CTP Noe Trees

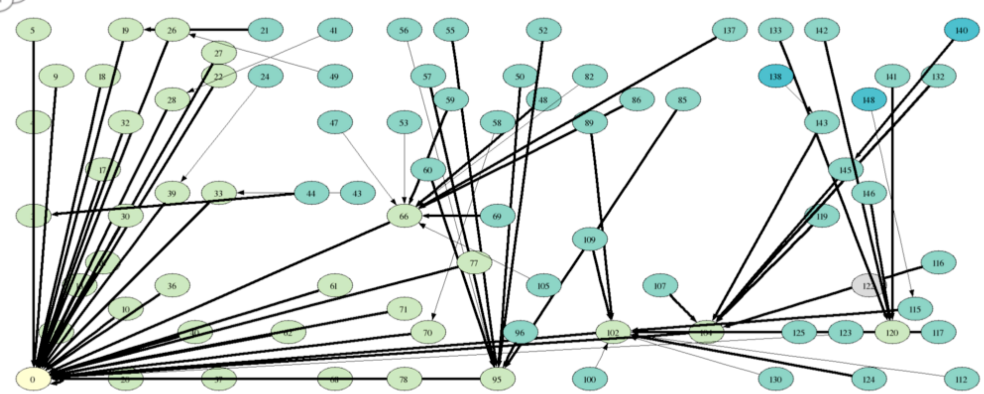
Kansei



Twist



Mirage



Reliable, Efficient, and Robust

Testbed	Delivery Ratio
Wymanpark	0.9999
Vinelab	0.9999
Tutornet	0.9999
NetEye	0.9999
Kansei	0.9998
Mirage-MicaZ	0.9998
Quanto	0.9995
Blaze	0.9990
Twist-Tmote	0.9929
Mirage-Mica2dot	0.9895
Twist-eyesIFXv2	0.9836
Motelab	0.9607

Reliable, Efficient, and Robust

Testbed	Delivery Ratio
Wymanpark	0.9999
Vinelab	0.9999
Tutornet	0.9999
NetEye	0.9999
Kansei	0.9998
Mirage-MicaZ	0.9998
Quanto	0.9995
Blaze	0.9990
Twist-Tmote	0.9929
Mirage-Mica2dot	0.9895
Twist-eyesIFXv2	0.9836
Motelab	0.9607

Retransmit

Reliable, Efficient, and Robust

Testbed	Delivery Ratio
Wymanpark	0.9999
Vinelab	0.9999
Tutornet	0.9999
NetEye	0.9999
Kansei	0.9998
Mirage-MicaZ	0.9998
Quanto	0.9995
Blaze	0.9990
Twist-Tmote	0.9929
Mirage-Mica2dot	0.9895
Twist-eyesIFXv2	0.9836
Motelab	0.9607

False ack

Retransmit

Reliable, Efficient, and Robust

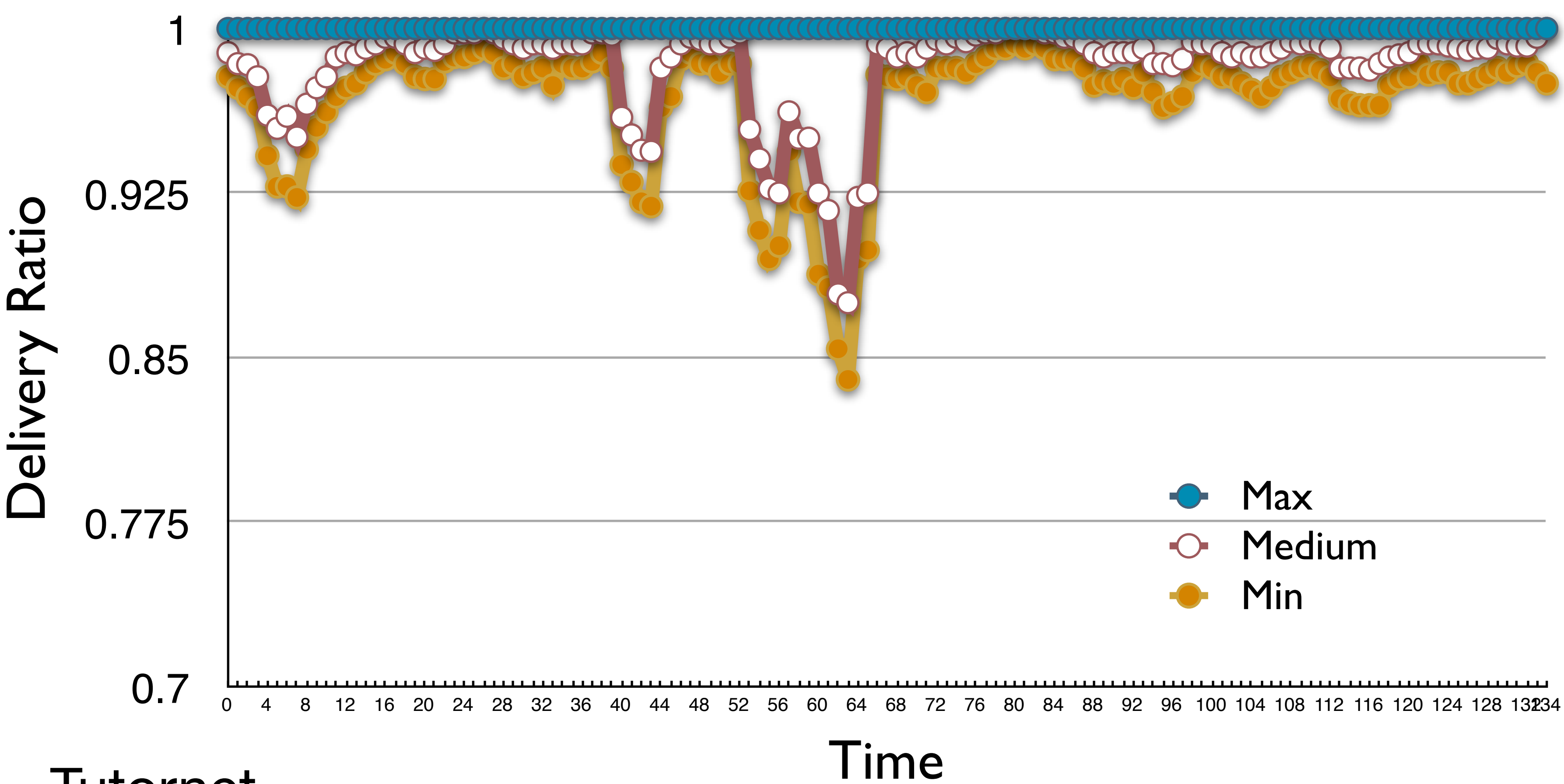
Testbed	Delivery Ratio
Wymanpark	0.9999
Vinelab	0.9999
Tutornet	0.9999
NetEye	0.9999
Kansei	0.9998
Mirage-MicaZ	0.9998
Quanto	0.9995
Blaze	0.9990
Twist-Tmote	0.9929
Mirage-Mica2dot	0.9895
Twist-eyesIFXv2	0.9836
Motelab	0.9607

False ack

Retransmit

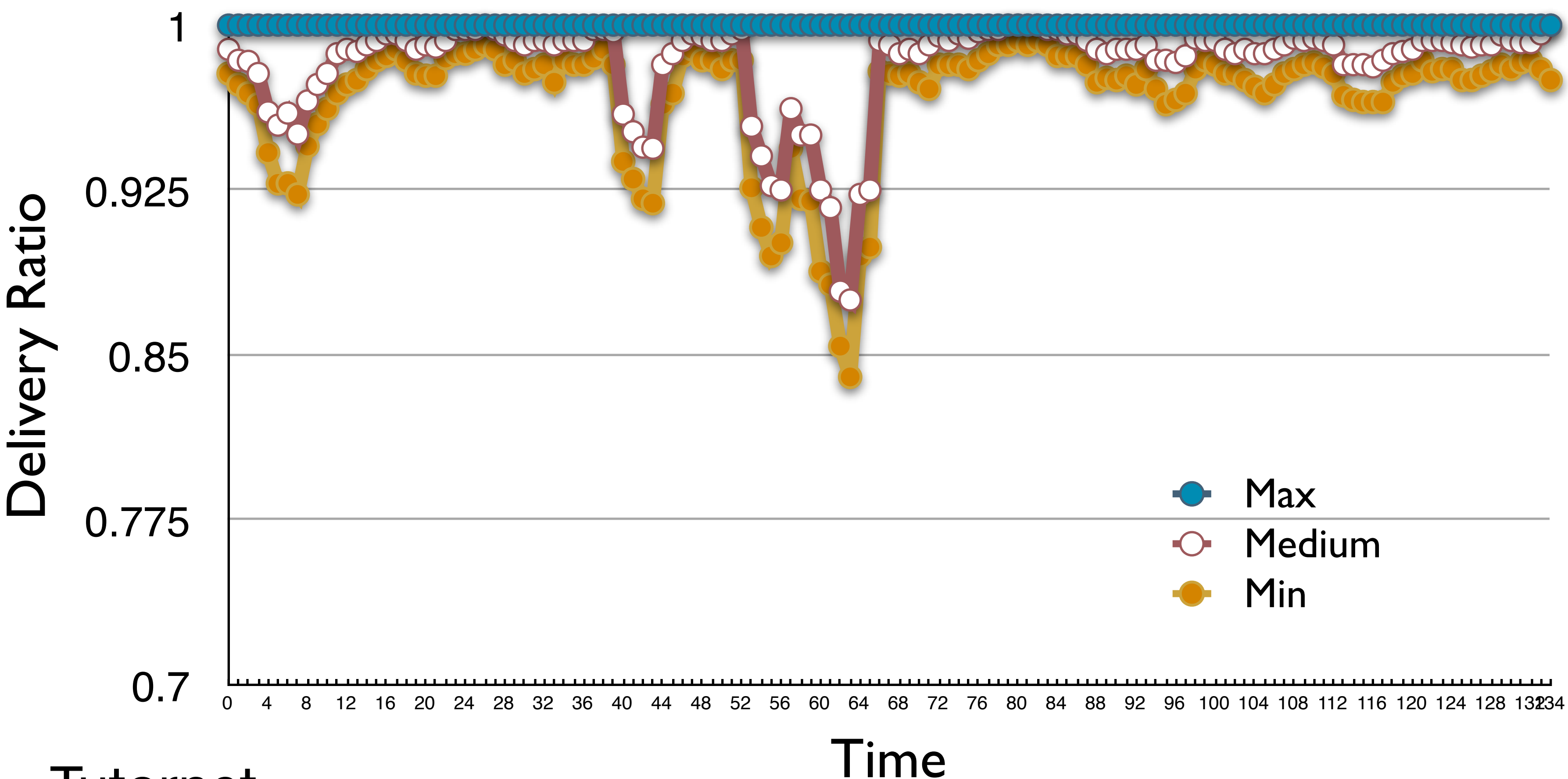
High end-to-end delivery ratio
(but not on all the testbeds!)

Reliable, Efficient, and Robust



Tutornet

Reliable, Efficient, and Robust

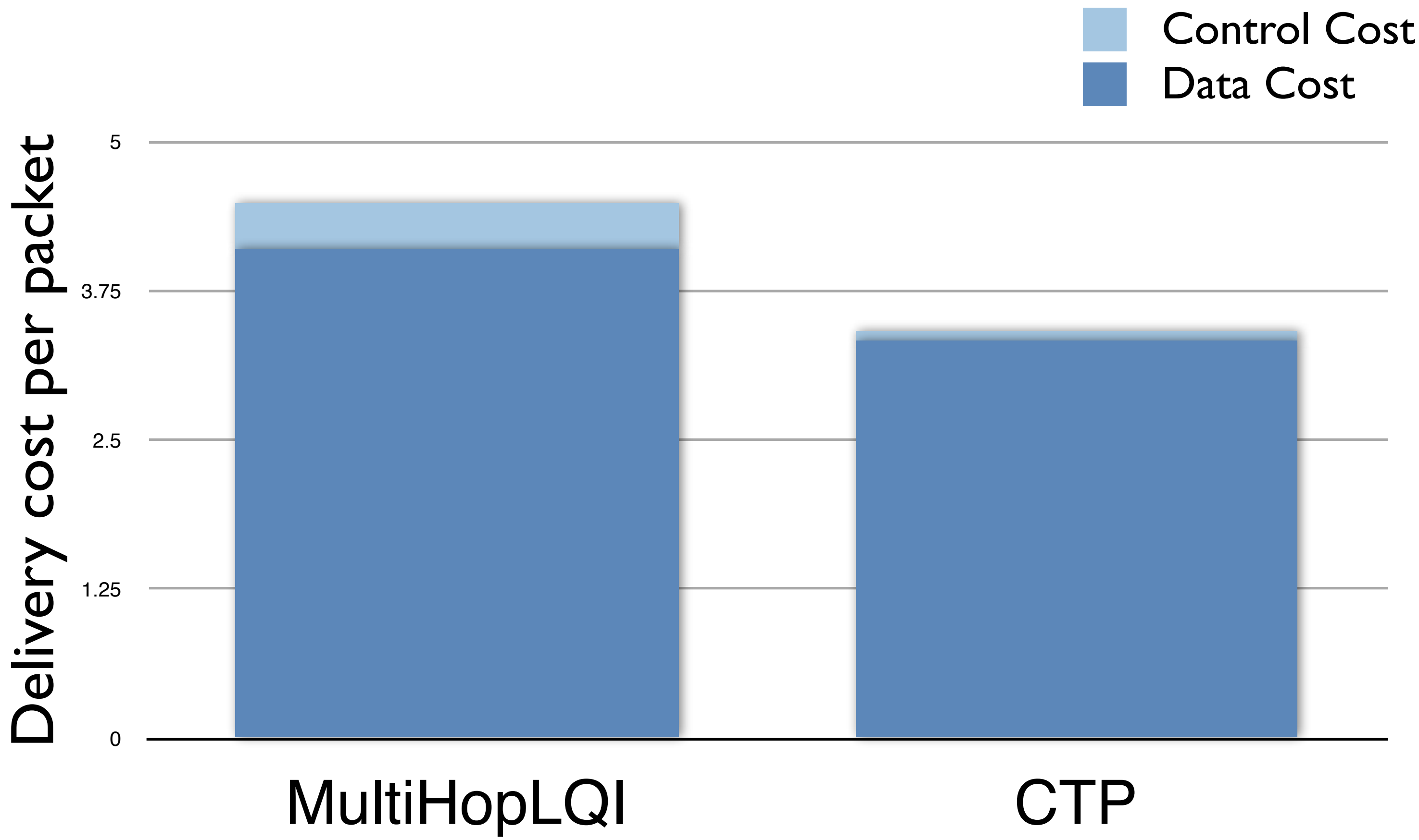


Tutornet

High delivery ratio across time
(short experiments can be misleading!)

28

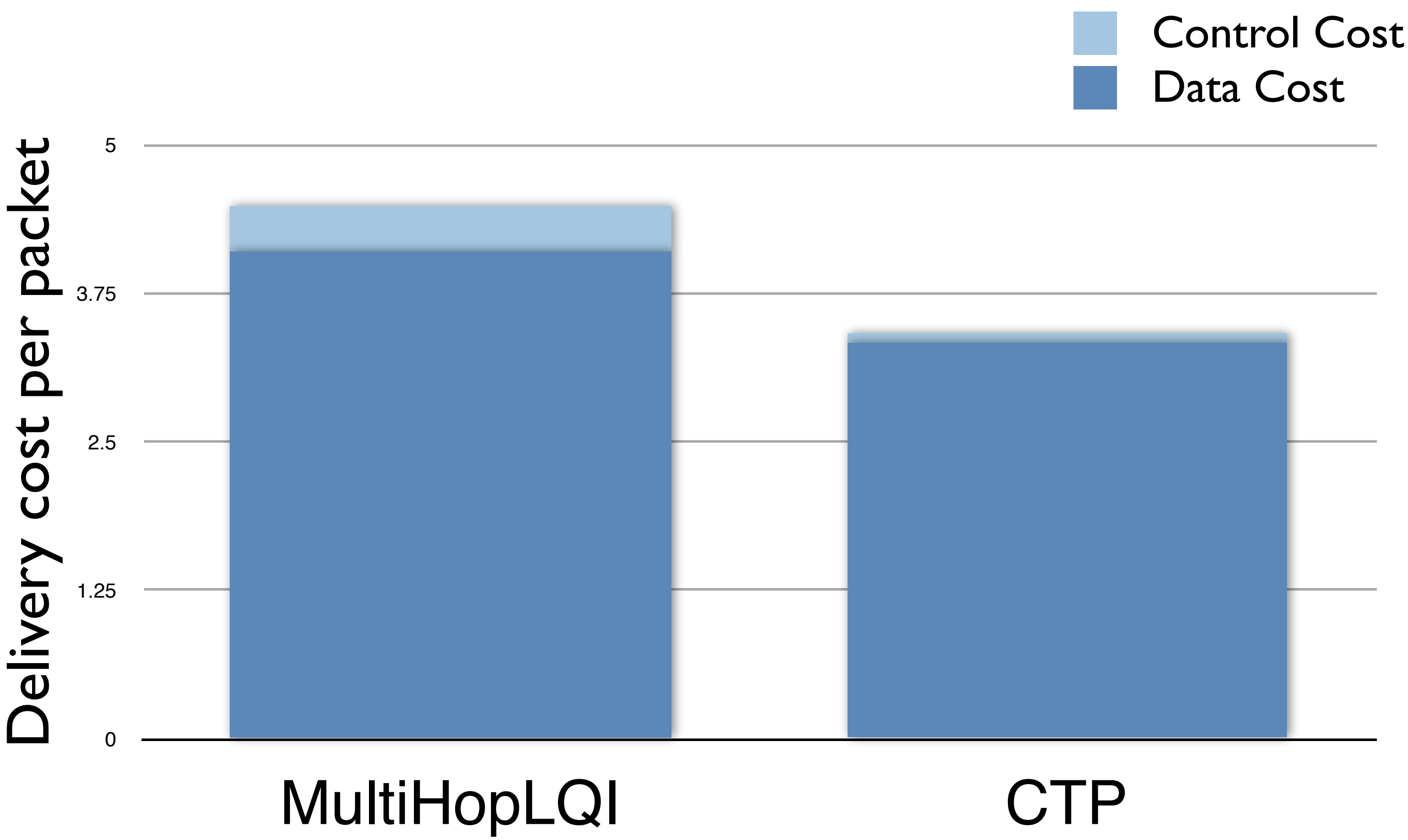
Reliable, Efficient, and Robust



Tutornet

29

Reliable, Efficient, and Robust

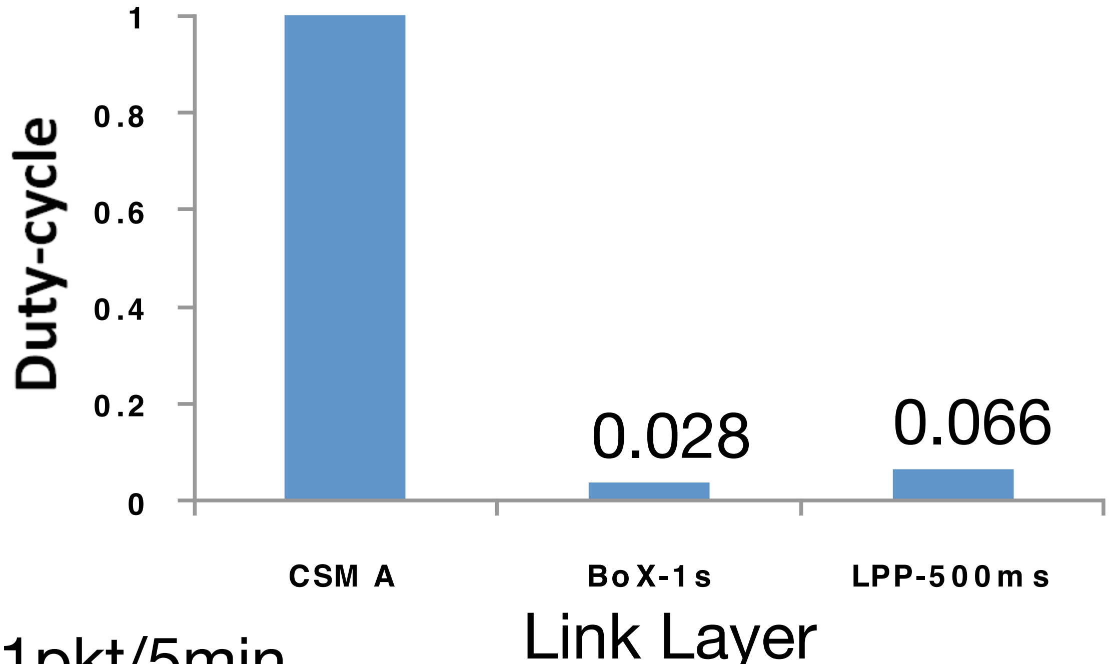


Tutornet

Low data and control cost

29

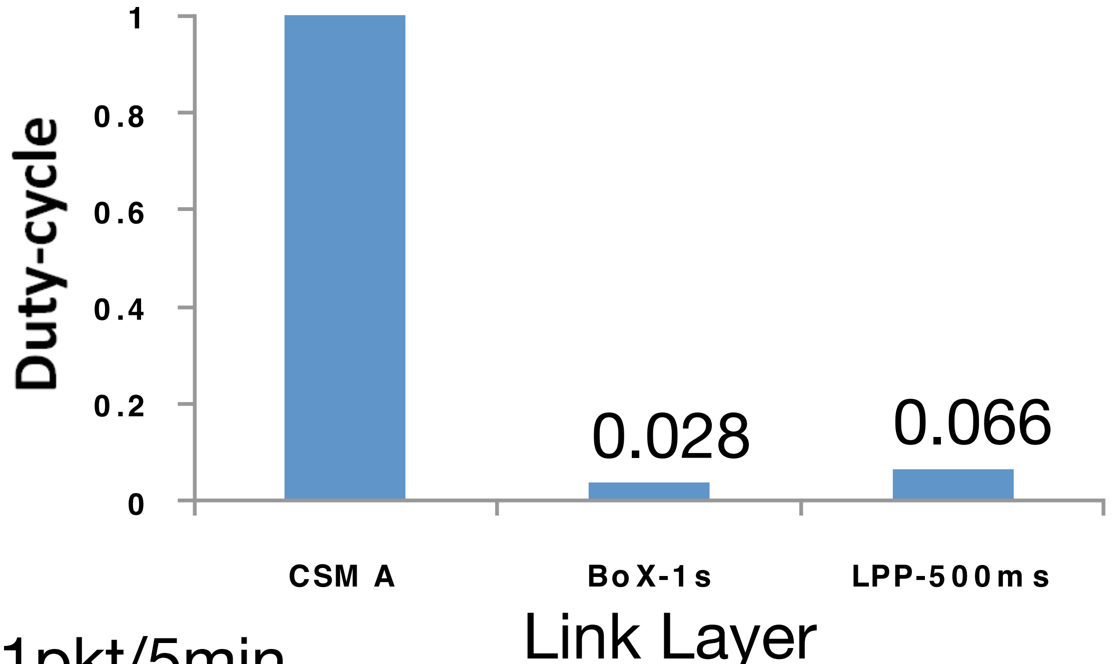
Reliable, Efficient, and Robust



Motelab, 1pkt/5min

30

Reliable, Efficient, and Robust

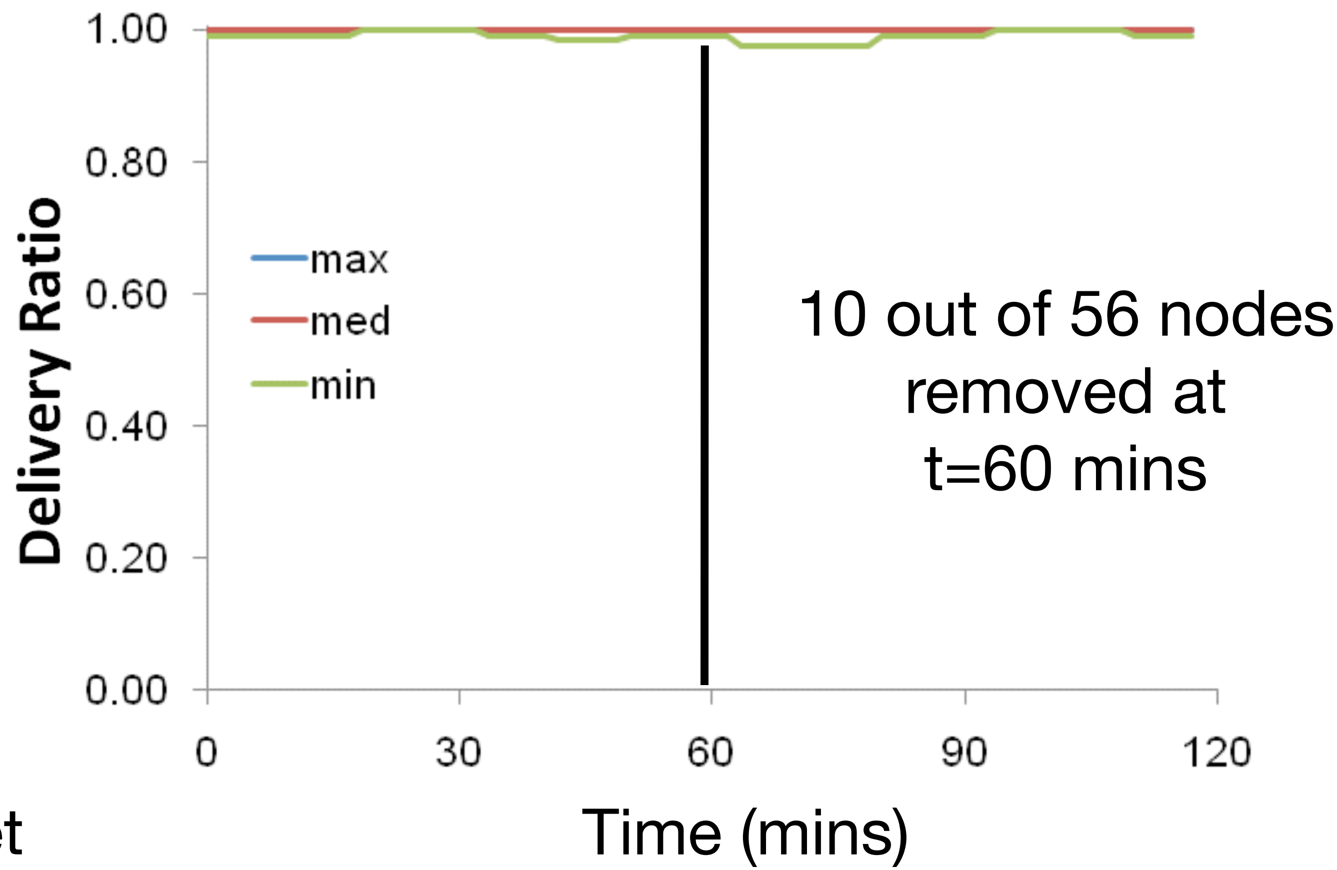


Motelab, 1pkt/5min

Low duty-cycle with low-power MACs

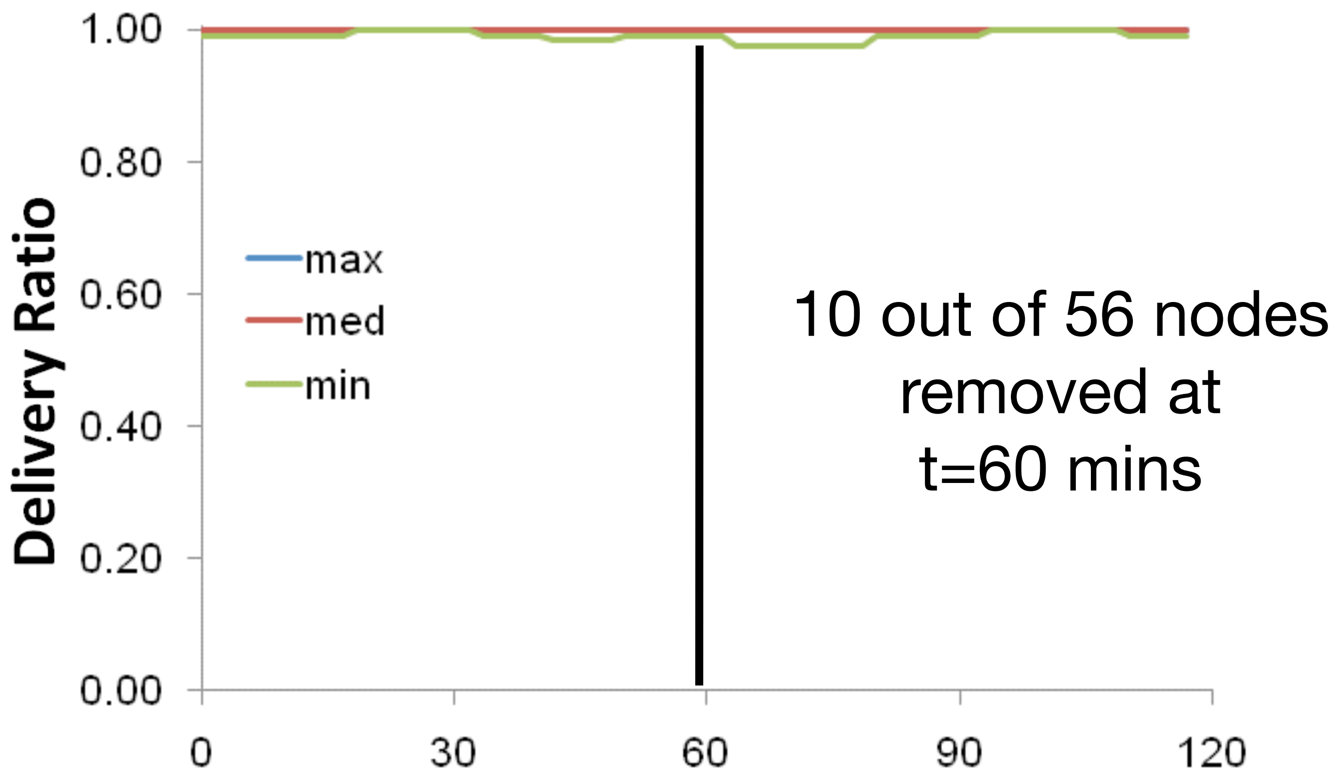
30

Reliable, Efficient, and Robust



Tutornet

Reliable, Efficient, and Robust



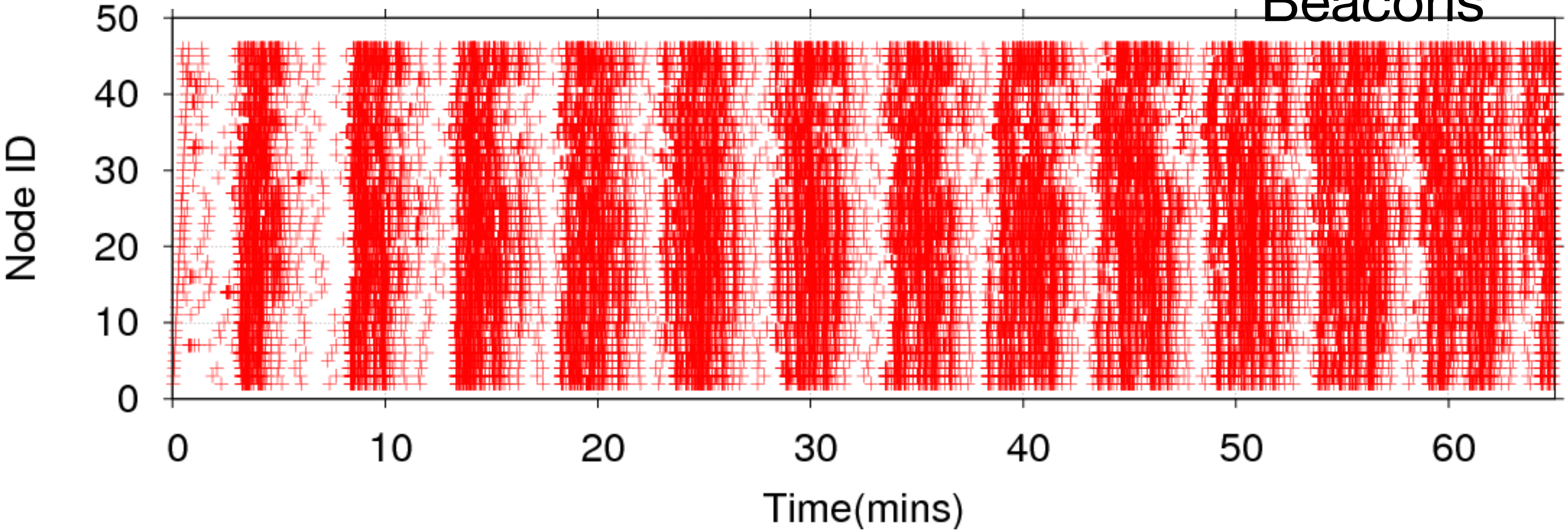
Tutornet

Time (mins)

No disruption in packet delivery

31

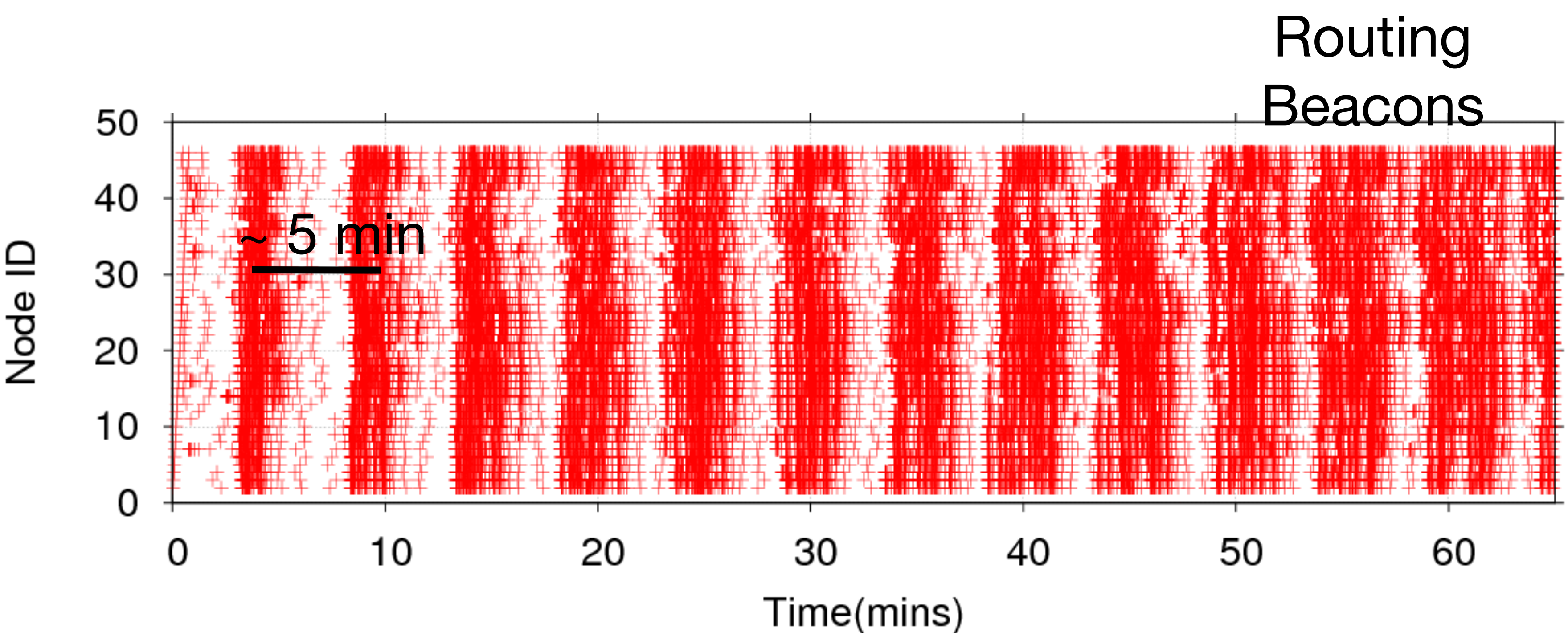
Reliable, Efficient, and Robust



Tutornet

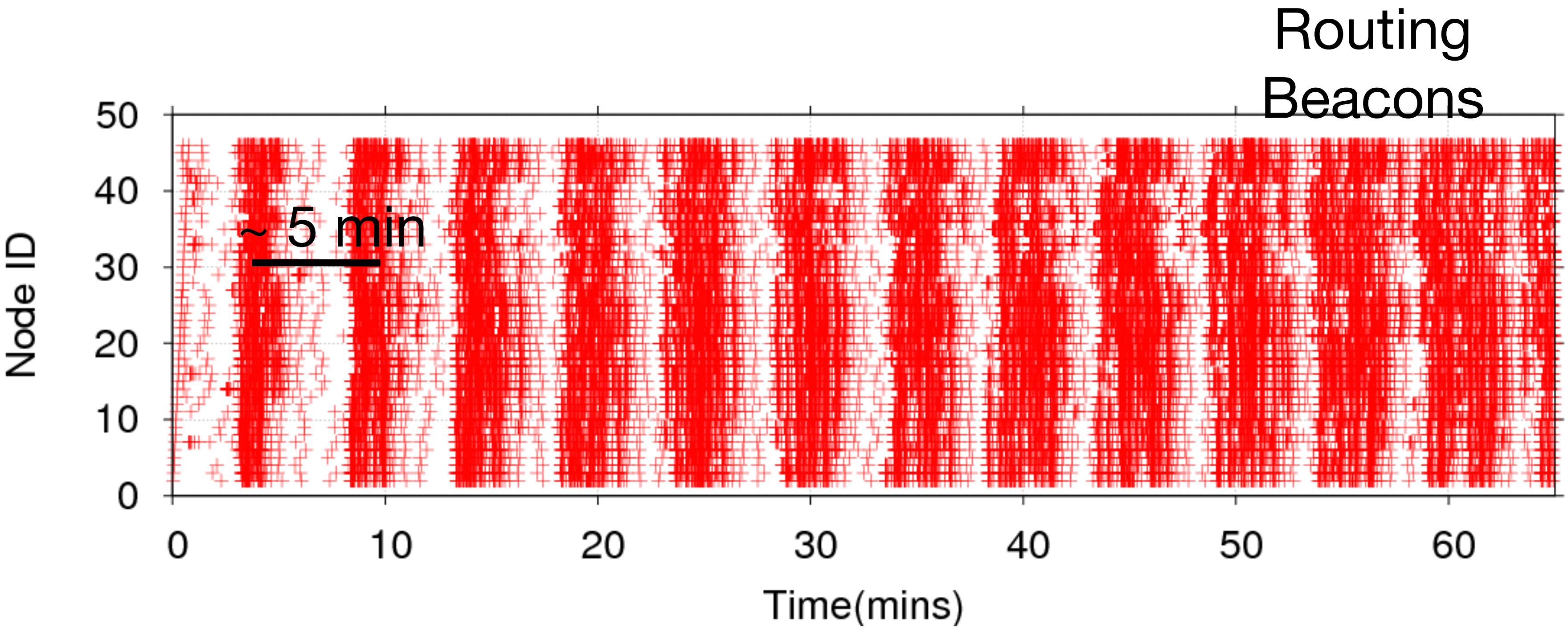
32

Reliable, Efficient, and Robust



Tutornet

Reliable, Efficient, and Robust

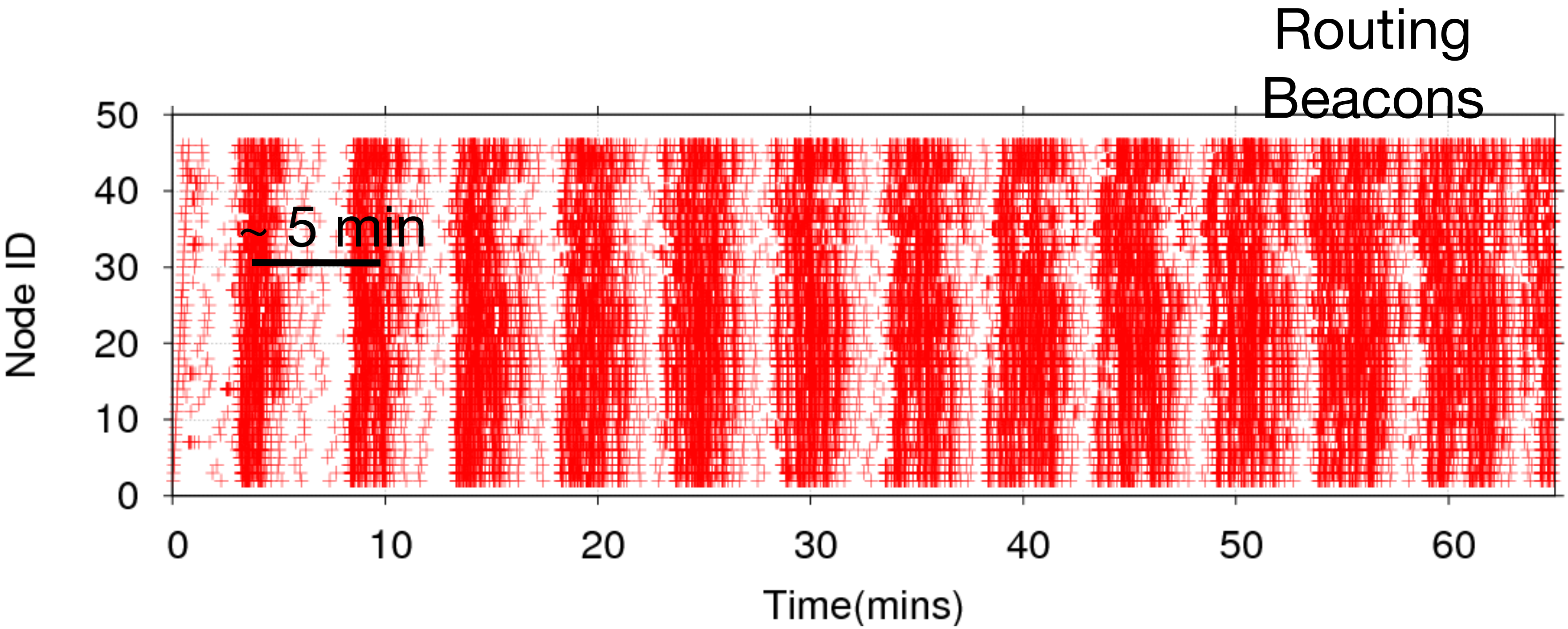


Tutornet

Delivery Ratio > 0.99

32

Reliable, Efficient, and Robust



Tutornet

Delivery Ratio > 0.99

High delivery ratio despite serious network-wide disruption
(most loss due to reboot while buffering packet)

CTP Noe Performance Summary

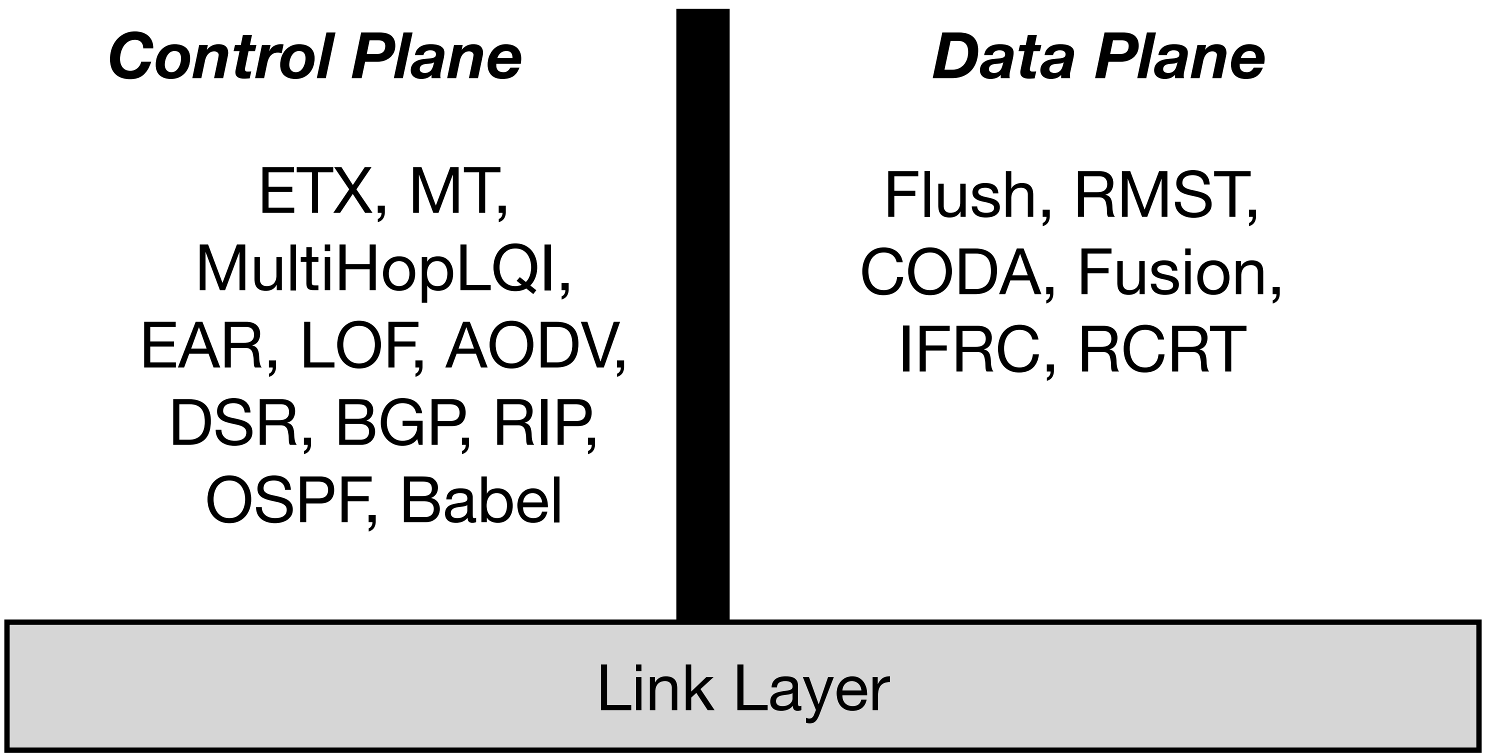
- **Reliability**
  - Delivery ratio > 90% in all cases
- **Efficiency**
  - Low cost and 5% duty cycle
- **Robustness**
  - Functional despite network disruptions

Conclusion

- **“Hard” networks → good protocols**
    - Tutornet & Motelab
- **Wireless routing benefits from data and control plane interaction**
- **Lessons applicable to distance vector routing**
    - Datapath validation & adaptive beaconing

- **Data trace from all the testbeds available at**
- **http://sing.stanford.edu/gnawali/ctp/**

Prior Work


Control Plane
ETX, MT,
MultiHopLQI,
EAR, LOF, AODV,
DSR, BGP, RIP,
OSPF, Babel
Data Plane
Flush, RMST,
CODA, Fusion,
IFRC, RCRT
Link Layer

4