Traffic Engineering with re-feedback

João Taveira Araújo j.araujo@ee.ucl.ac.uk

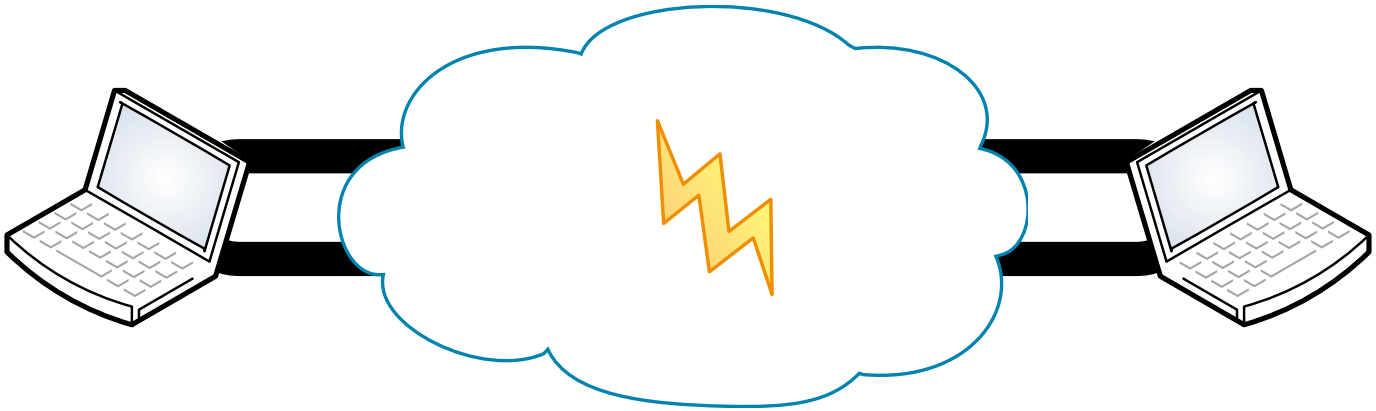
UCL

Pub Crawl [t-1], 4th March 2010

Flag day

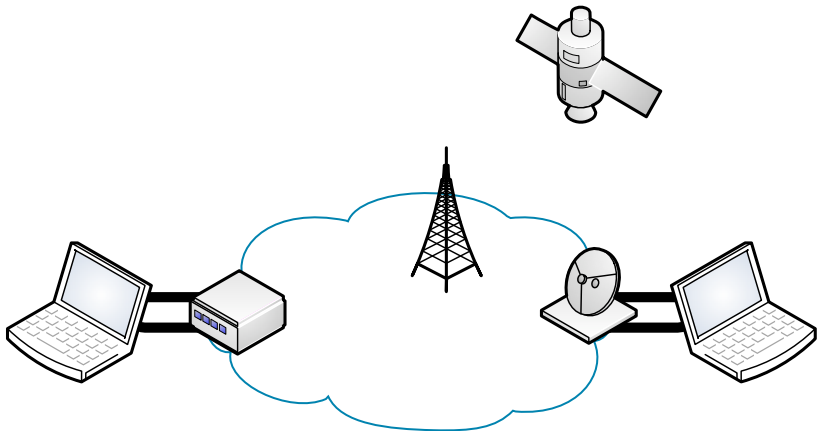
1st January 1983

CC in theory..



- end hosts send traffic
- network signals congestion with packet drop
- end hosts adapt

.. but in practice

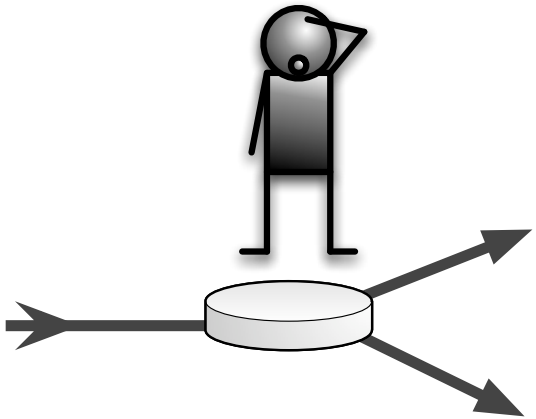


- deep packet inspection
- traffic engineering / dynamic re-routing
- performance enhancing proxies

Result

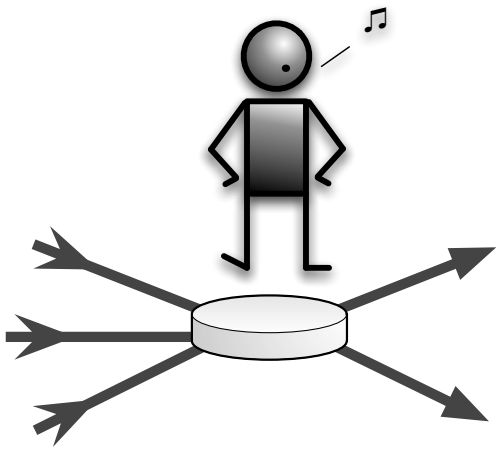
- new transport protocols can't get deployed
- TCP ossified as waist of hourglass
- no major innovation at transport since 1987

Network layer in theory ..



- dumb network
- forward packets

.. but in practice

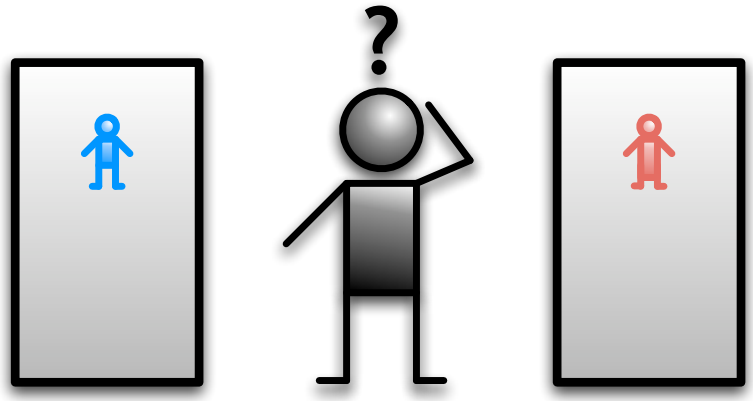


- capable of more than best-effort
- highly parallelized

Result

- transport makes assumptions about network
- forward flows, not packets
- network can't improve significantly without breaking transport

Layer identity disorder



- networks attempt to manipulate resource sharing
  - traffic engineering
  - traffic shaping
- end hosts attempt to manipulate routing
  - overlay routing, p2p
  - user centric networking
  - multipath TCP

Traffic engineering

- attempt to load balance traffic and provision for demand
- per-domain, minimize local costs (utilization)
  - sometimes at expense of global performance
- traditionally offline
  - but traffic is fluctuating more and more
  - at odds with app-layer load balancing, e.g. bittorrent

Multipath TCP

- attempt to load balance ~~traffic~~ congestion
- relies on multi-homing
  - no other way of obtaining path diversity
  - end user becomes router
- only benefits subset of flows (bulk transfer, mobility)

Crossing the divide

- network layer might improve with some transport information
  - to evaluate path quality
  - to set up state
- transport layer could use more network information
  - indication of imminent congestion (ECN) already specified
  - indication of paths

Need for transport information

- first proposed by Bob Briscoe / BT
  - provides network means to police sources
  - provides metric on which to compare usage
- lots of use cases
  - mediating between p2p users and light users
  - getting iPhone users in New York to share limited bandwidth?

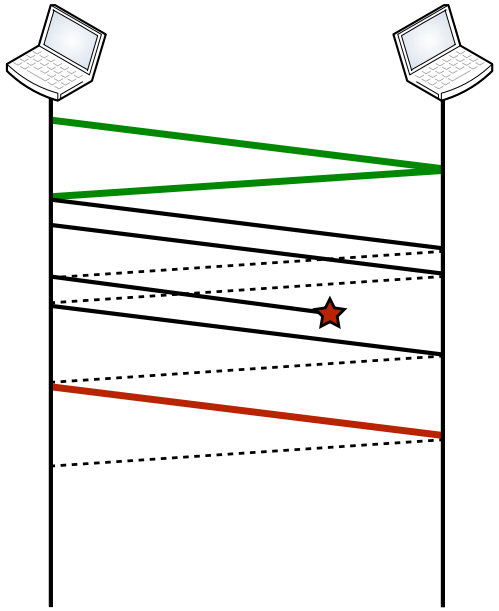
Basic idea

- take what the receiver feeds back ...
- ... re-insert it into the network.
- re-ECN does this with ECN markings
- REFLEX does this relative to loss.

REFLEX: re-feedback for loss exposure

Mark IP header:

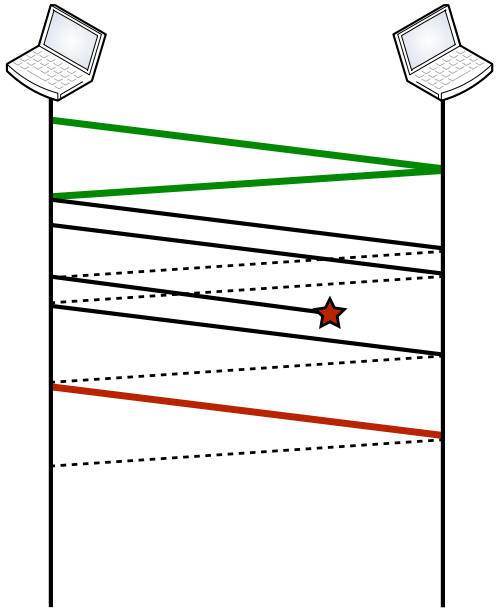
- green if no feedback loop established,
- red for each packet lost,
- grey otherwise.



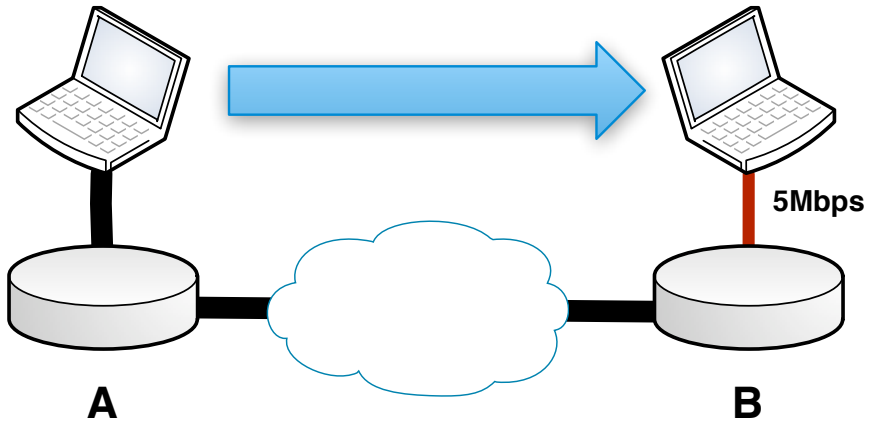
REFLEX: re-feedback for loss exposure

... in terms of TCP:

- SYNs are green,
- retransmits are red,
- everything else is grey.

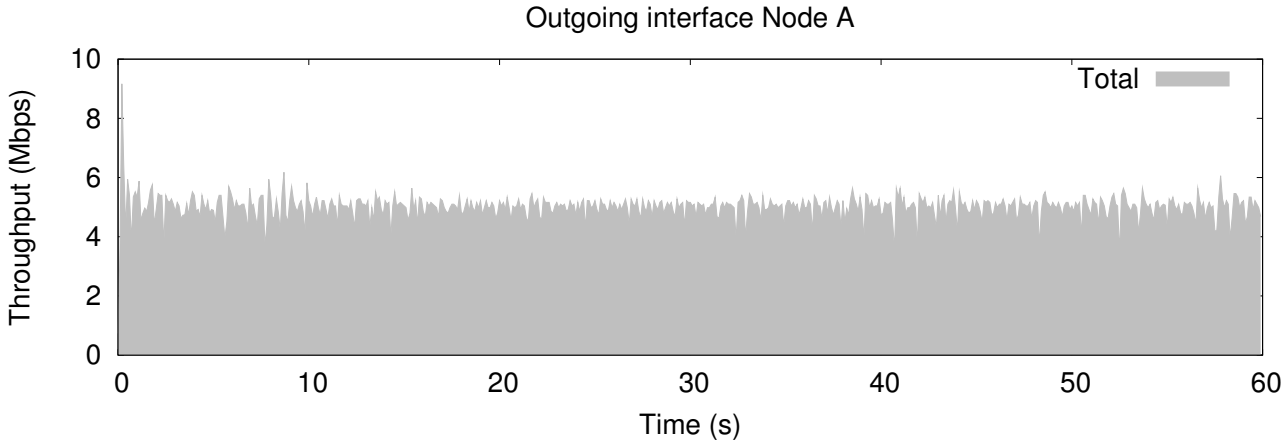


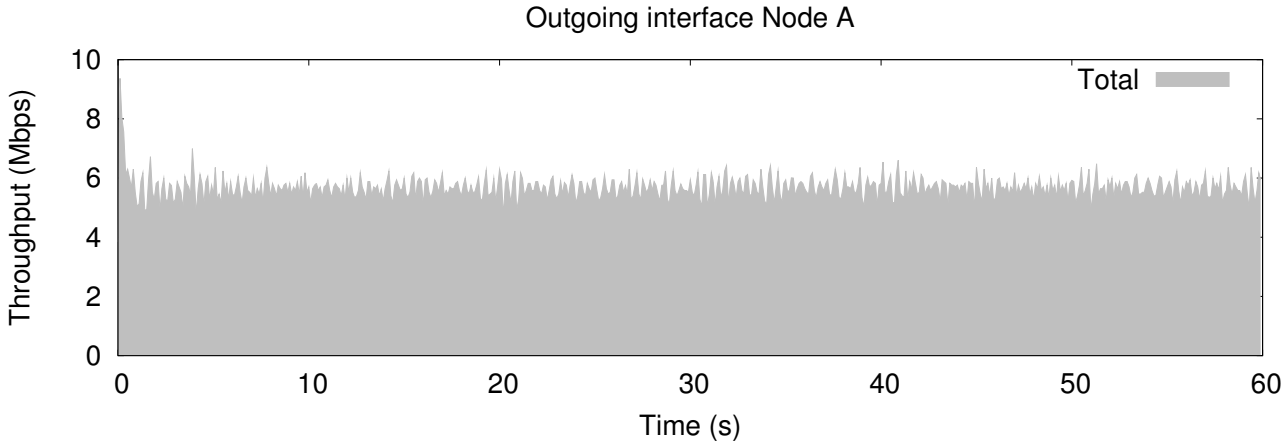
REFLEX



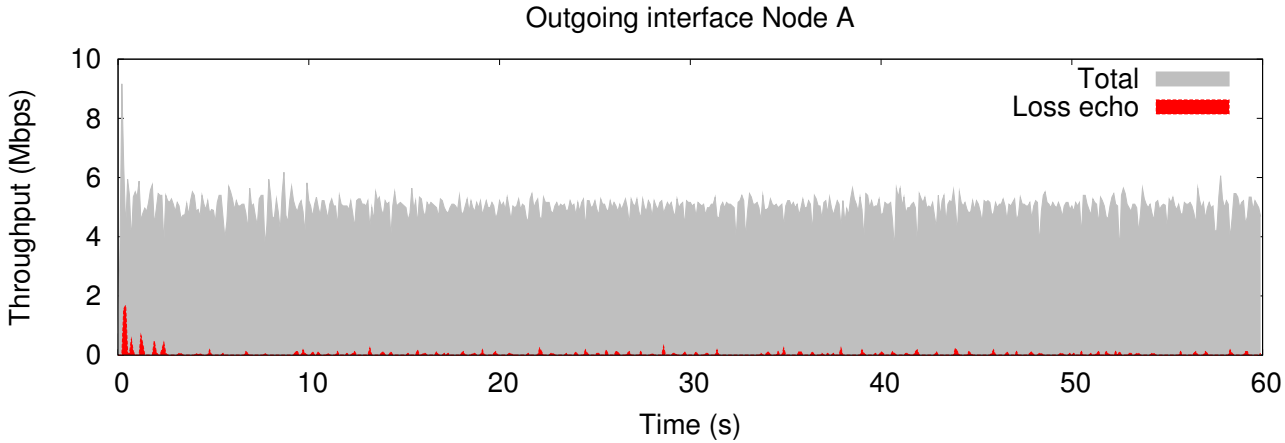
- RTT = 50ms
- run 5 flows through bottleneck
- .. then run 50

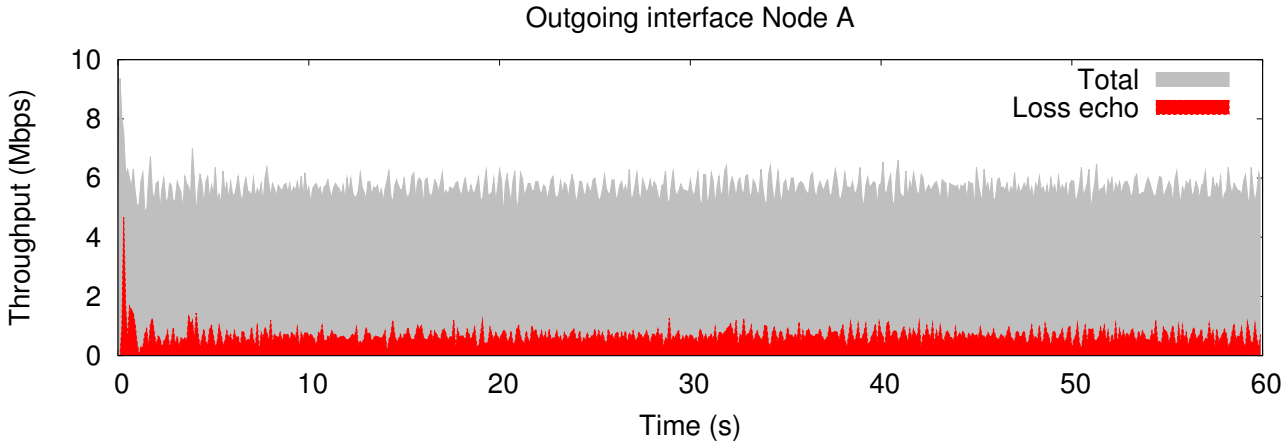
5 flows, or 50?


Outgoing interface Node A
Throughput (Mbps)
Time (s)
Total
0
2
4
6
8
10
0
10
20
30
40
50
60

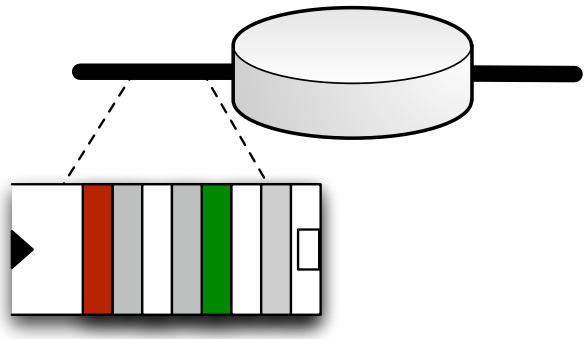

Outgoing interface Node A
Throughput (Mbps)
Time (s)
Total
0
2
4
6
8
10
0
10
20
30
40
50
60

REFLEX


Outgoing interface Node A
Throughput (Mbps)
Time (s)
Total
Loss echo
0
2
4
6
8
10
0
10
20
30
40
50
60


Outgoing interface Node A
Throughput (Mbps)
Time (s)
Total
Loss echo
0
2
4
6
8
10
0
10
20
30
40
50
60

REFLEX



Network can calculate loss aggregates simply from inspecting IP headers.

- removes need for probes - traffic is probe.
- information must be aggregated by destination address
  - trade-off between state and precision.

REFLEX vs re-ECN

- works with or without re-ECN
- can be used as incremental deployment path for re-ECN

REFLEX vs re-ECN

- works with or without re-ECN
- can be used as incremental deployment path for re-ECN
- good++
  - congestion exposed
  - tiny flow state transport agnostic
  - small no network changes (ECN)

REFLEX vs re-ECN

- works with or without re-ECN
- can be used as incremental deployment path for re-ECN
- good++
  - congestion exposed
  - tiny flow state transport agnostic
  - small no network changes (ECN)
- ungood++
  - end-to-end congestion only
  - no accountability across domains
  - harder to police sources

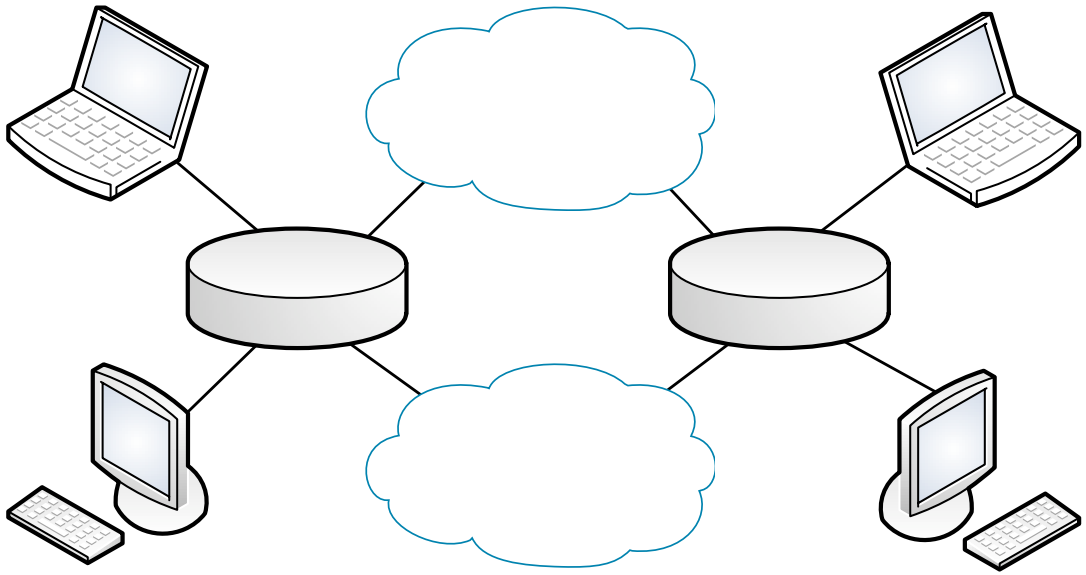
What next?

What to do with information given by REFLEX?

TE in stub domains

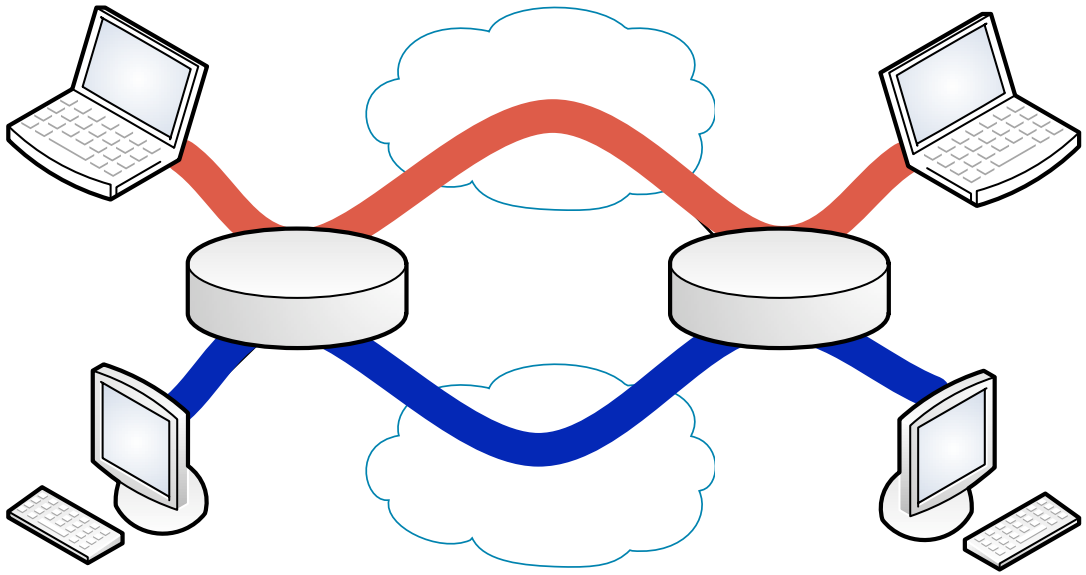
- path loss information only useful at edges
- stub domains typically multihomed
- MPLS rules

TE in stub domains



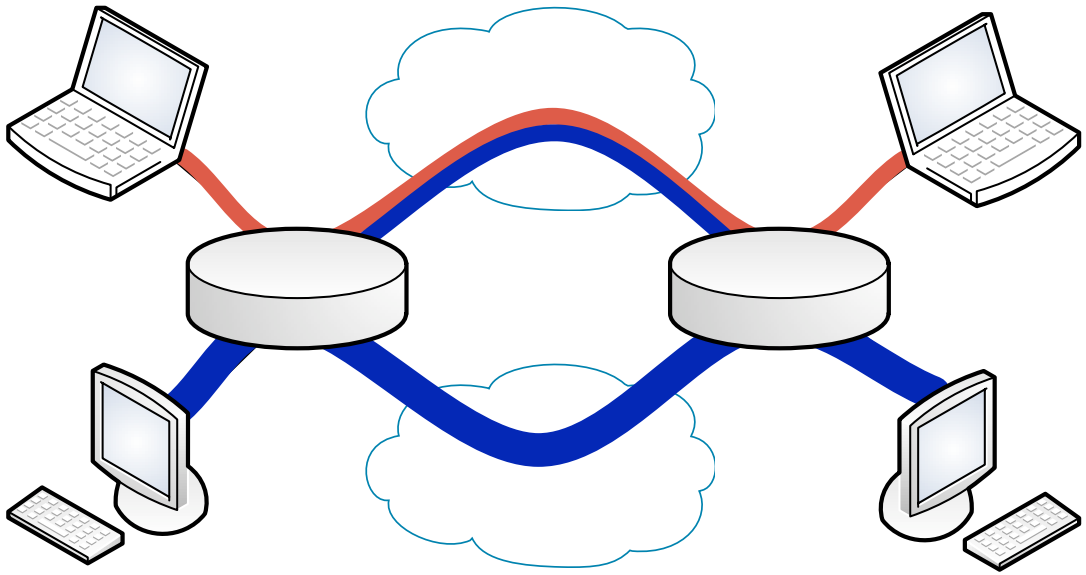
- gather demand matrix offline

TE in stub domains



- calculate routing to balance traffic

TE in stub domains

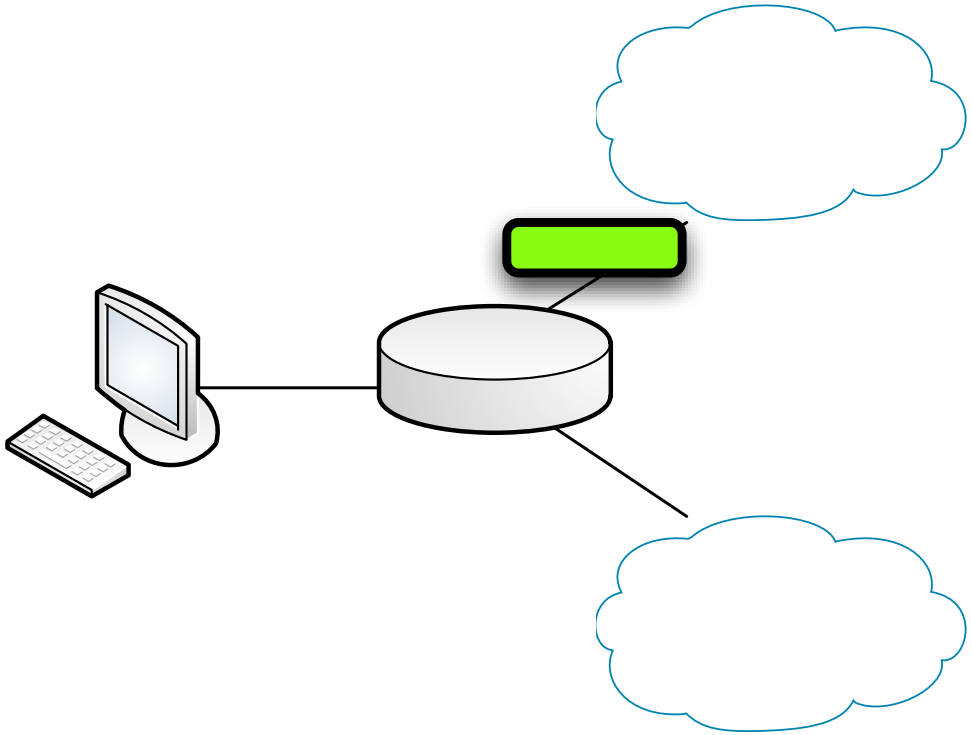


- pray demand doesn't shift

REACT: REFLEX Action

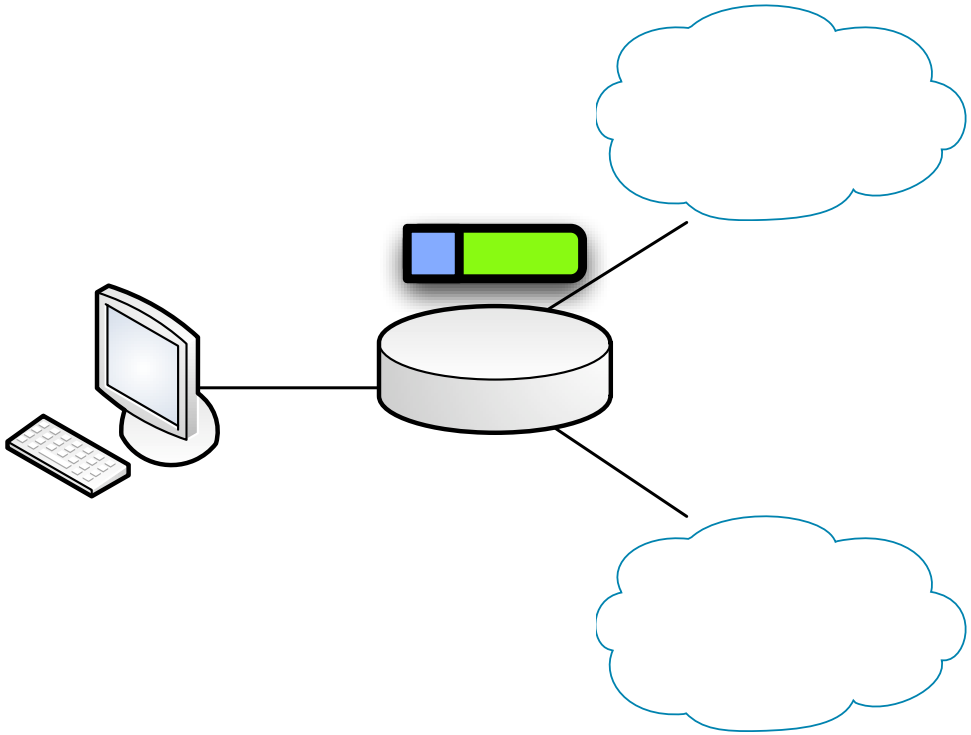
- don’t keep state in network
- react dynamically
- not intra-domain or inter-domain, optimize e2e

REACT: Flow tagging



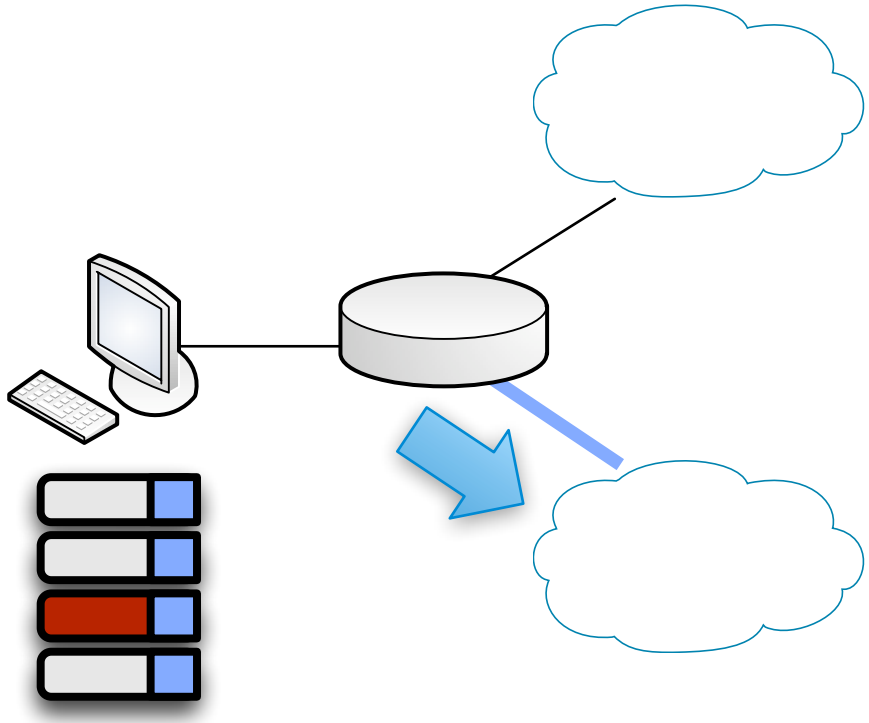
Edge router intercepts ’green’ packets

REACT: Flow tagging



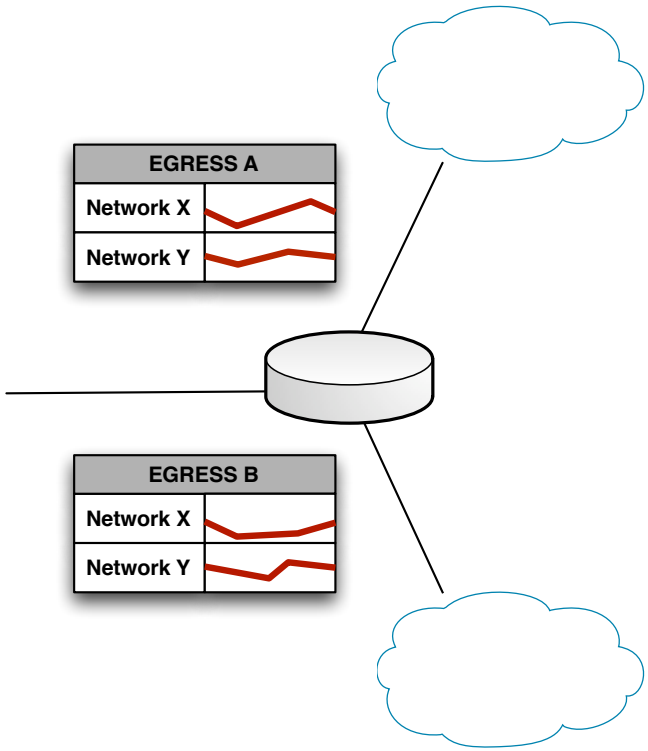
Reverse lookup and tag packet with path

REACT: Flow tagging



Host tags all subsequent packets with same tag

REACT: Keeping track



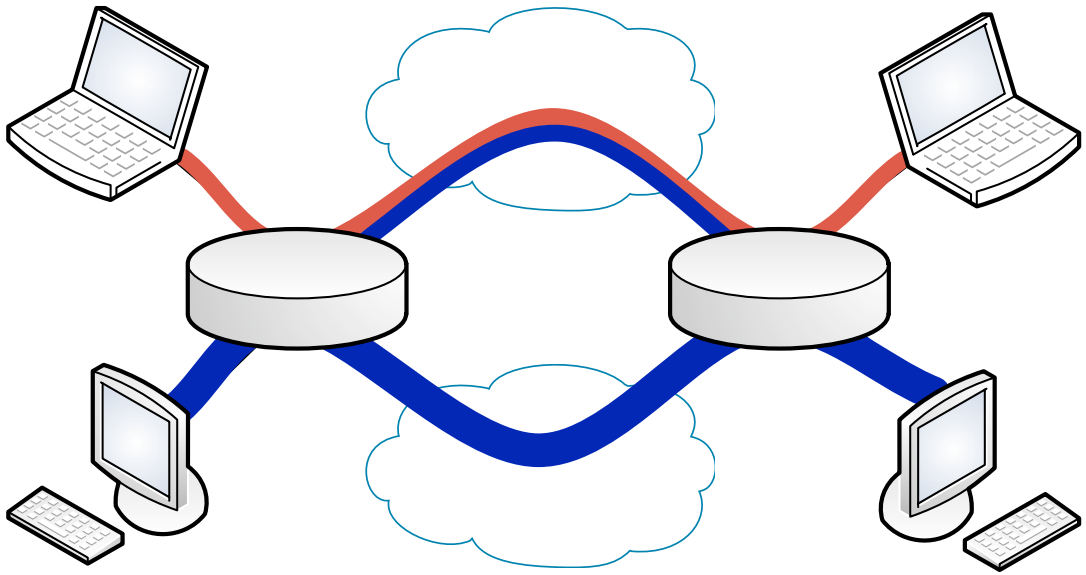
- per interface / per prefix stats
- keep weighted exp. average of loss as metric

REACT: Balancing loss

For every green packet coming in:

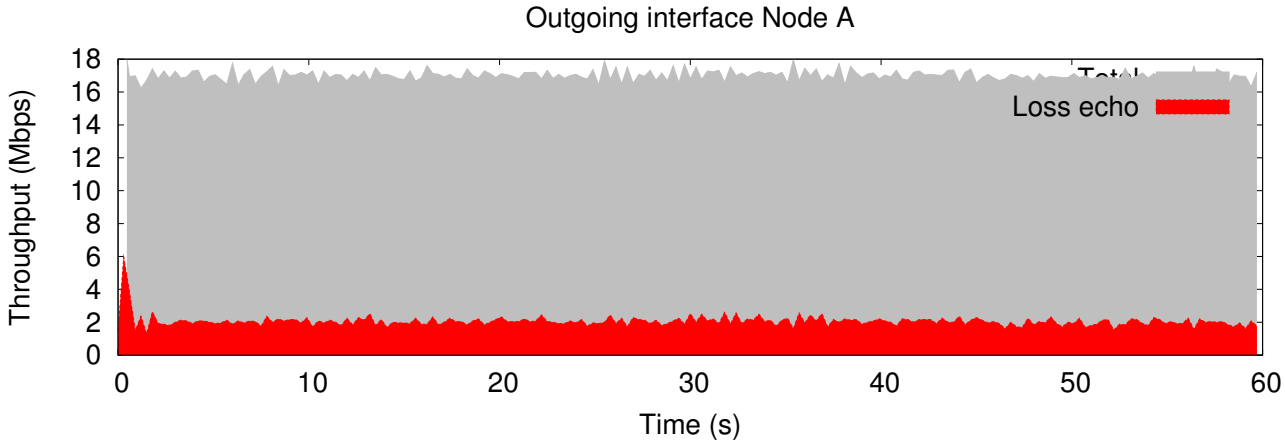
- check all interfaces for matching prefix
- pick randomly, proportinally to inverse of loss

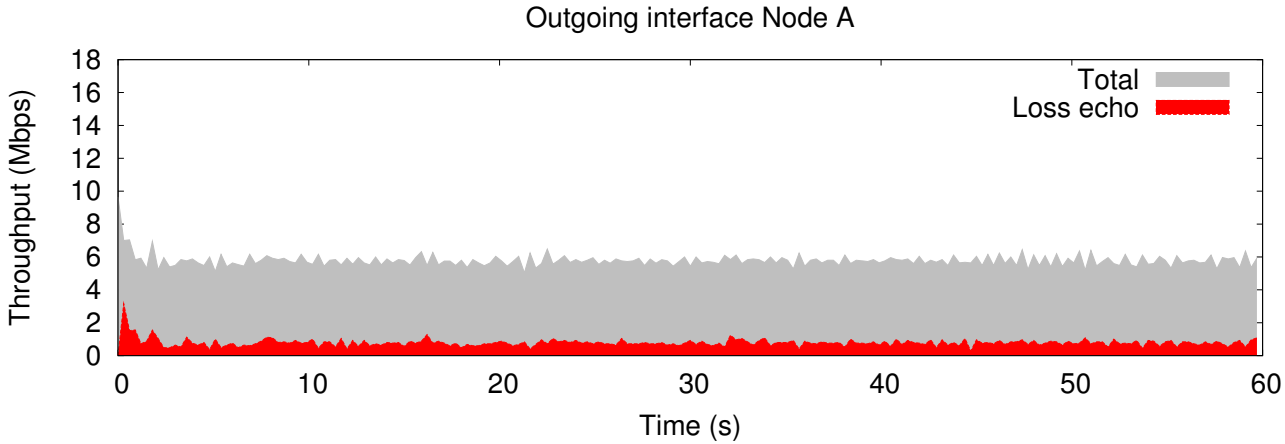
REACT: Balancing loss



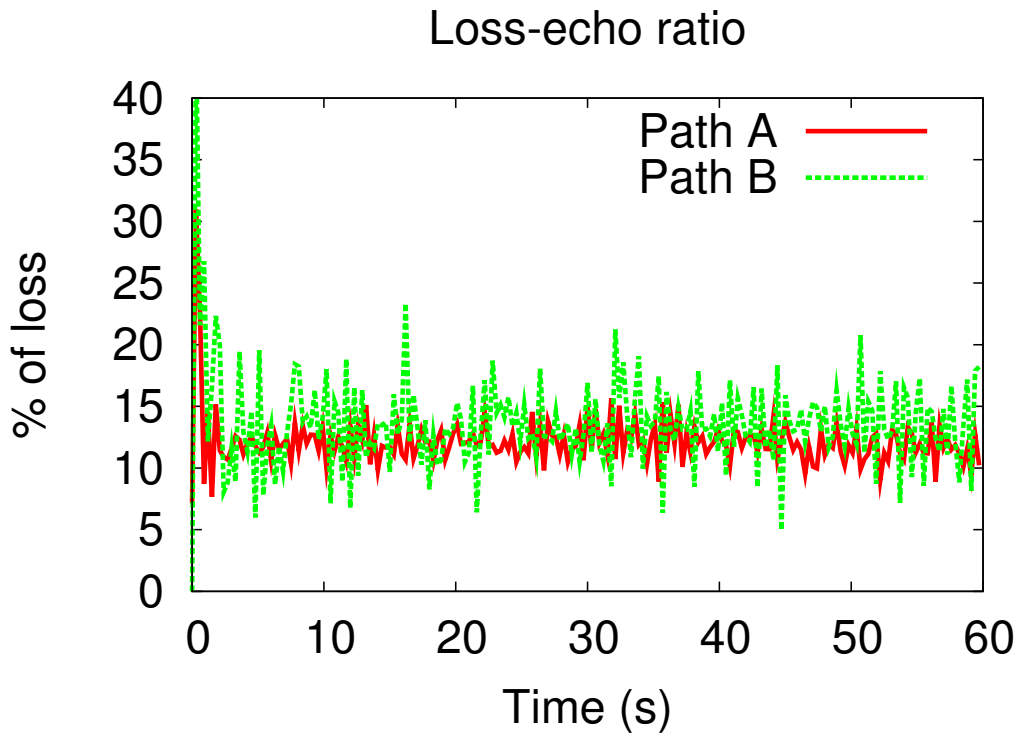
- Bottleneck set to 15Mbps and 5Mbps
- 50 concurrent flows to each network

REACT: Balancing loss





REACT: Balancing loss



Conclusions

- Thinking strictly in layers gets in the way
- Network can take some part in load balancing
  - but can't get in transport's way
  - should not remain opaque for path changes
- Transport needs to allow itself to be policed
  - information allows transport agnostic throttling
  - can be used to evolve form single path

End?

Questions