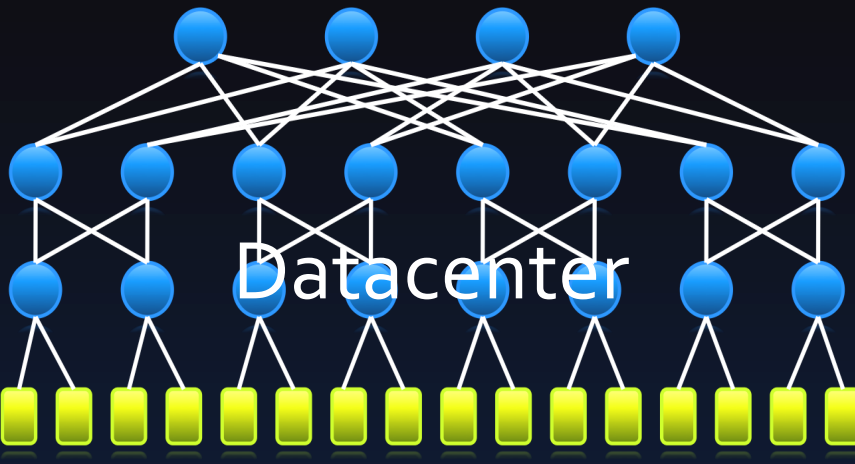
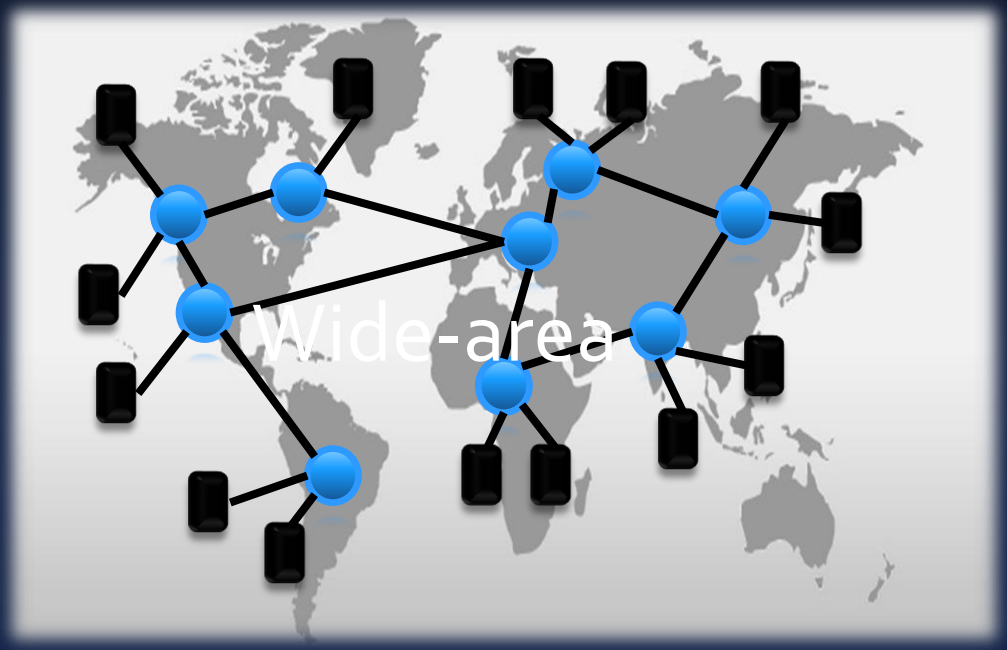
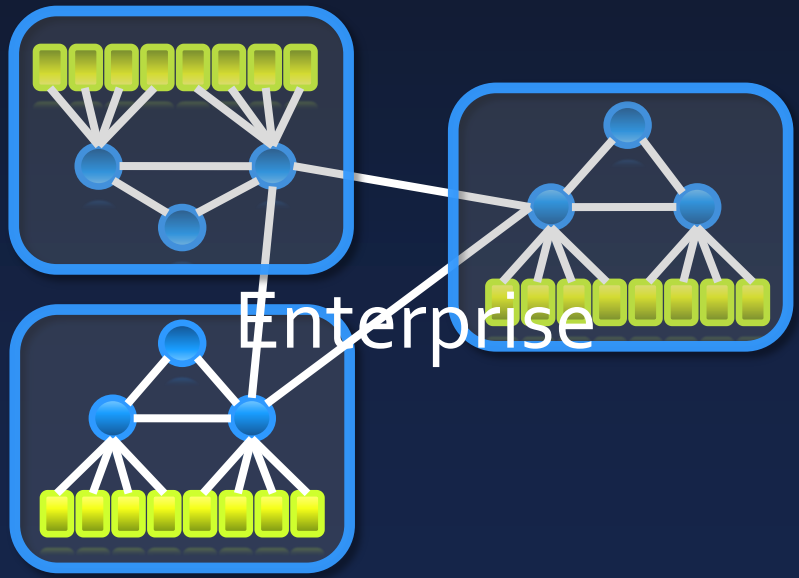
Should a load-balancer choose the path

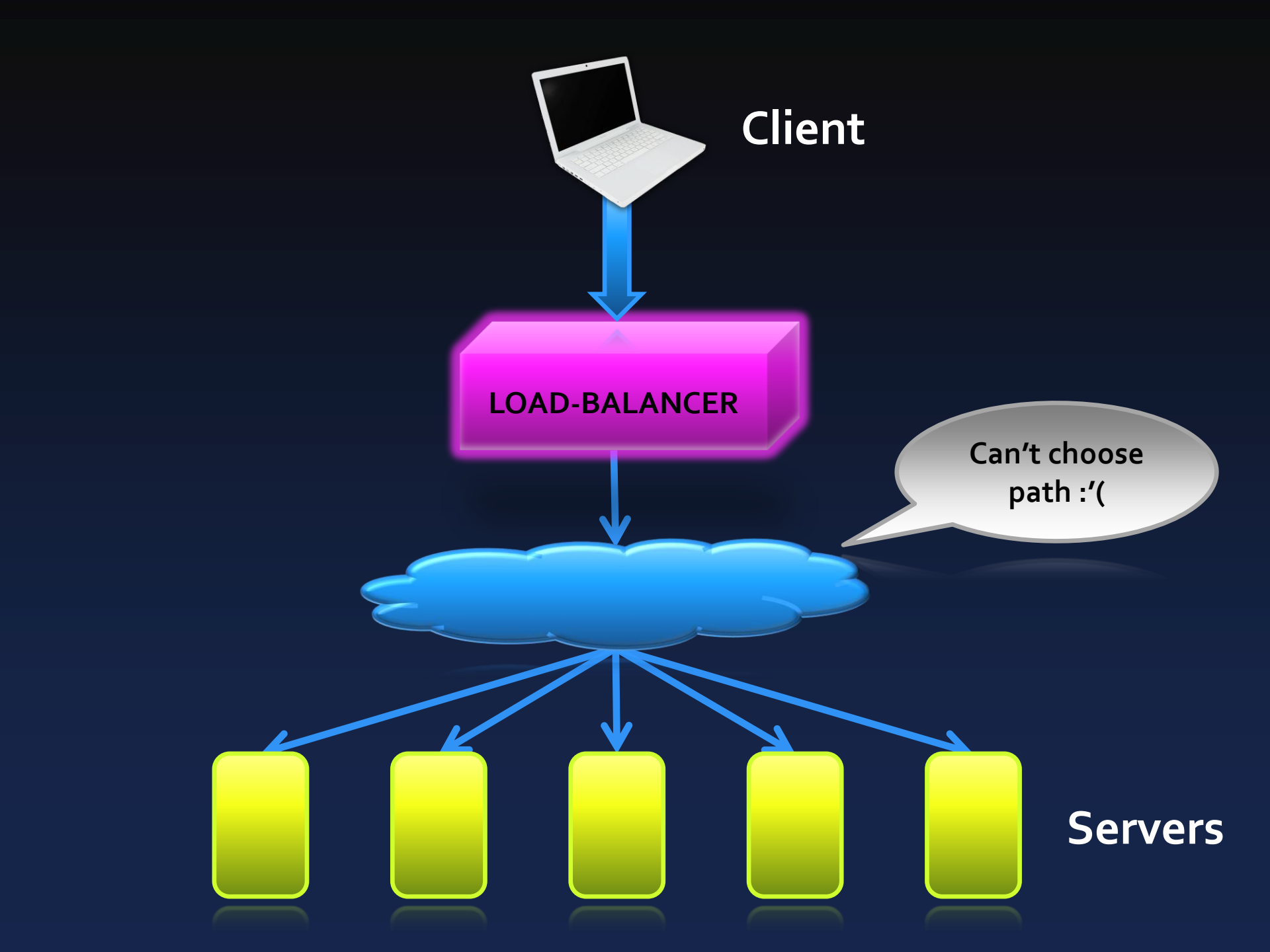
as well as the server?

Nikhil Handigol
Stanford University
Joint work with Nick McKeown and Ramesh Johari


Datacenter


Wide-area

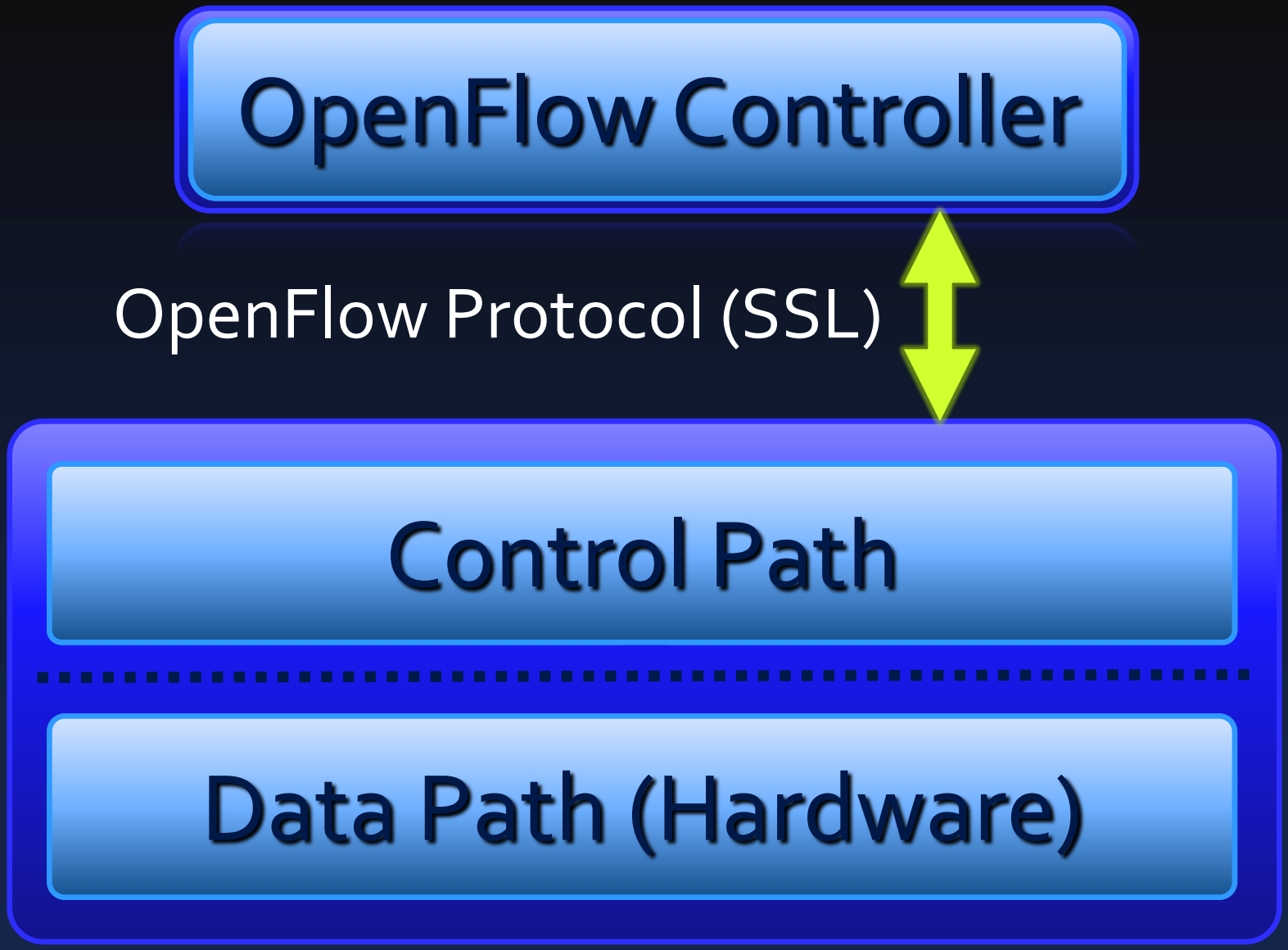

Enterprise


Client
LOAD-BALANCER
Can’t choose path :’(
Servers

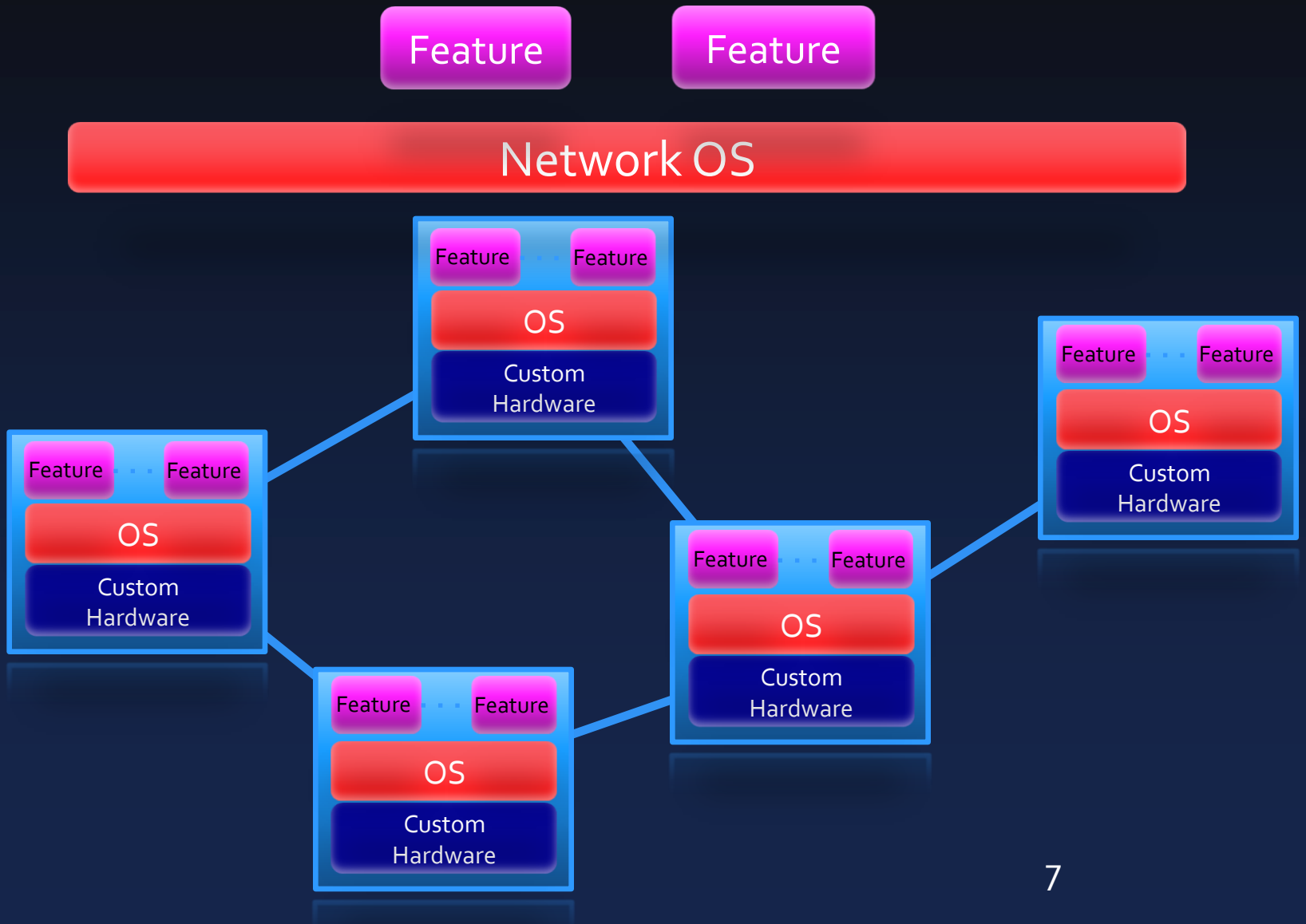
Outline and goals

- A new architecture for distributed load-balancing
  - joint (server, path) selection
- Demonstrate a nation-wide prototype
- Interesting preliminary results

I’m here to ask for your help!

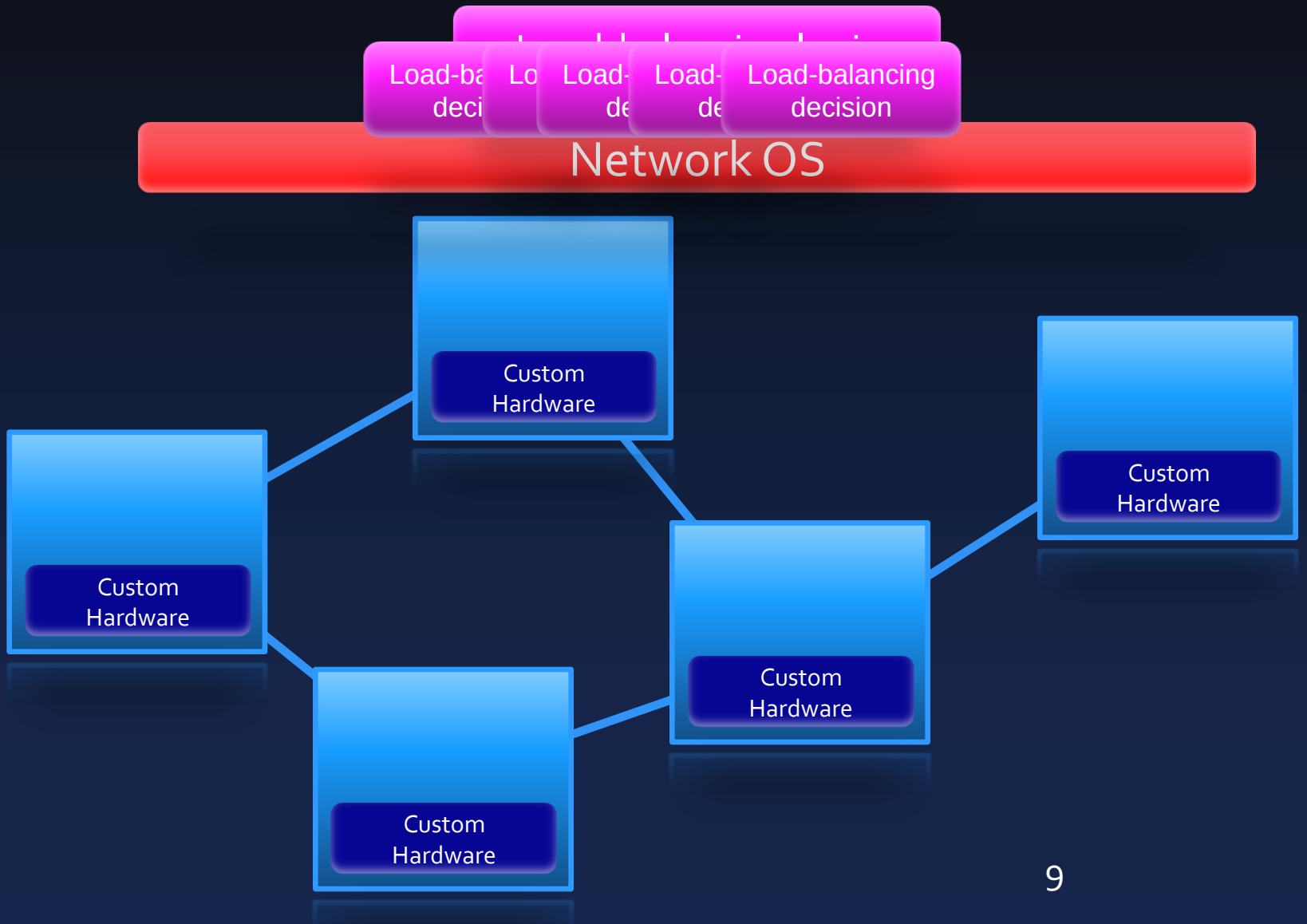

OpenFlow Controller
OpenFlow Protocol (SSL)
Control Path
Data Path (Hardware)

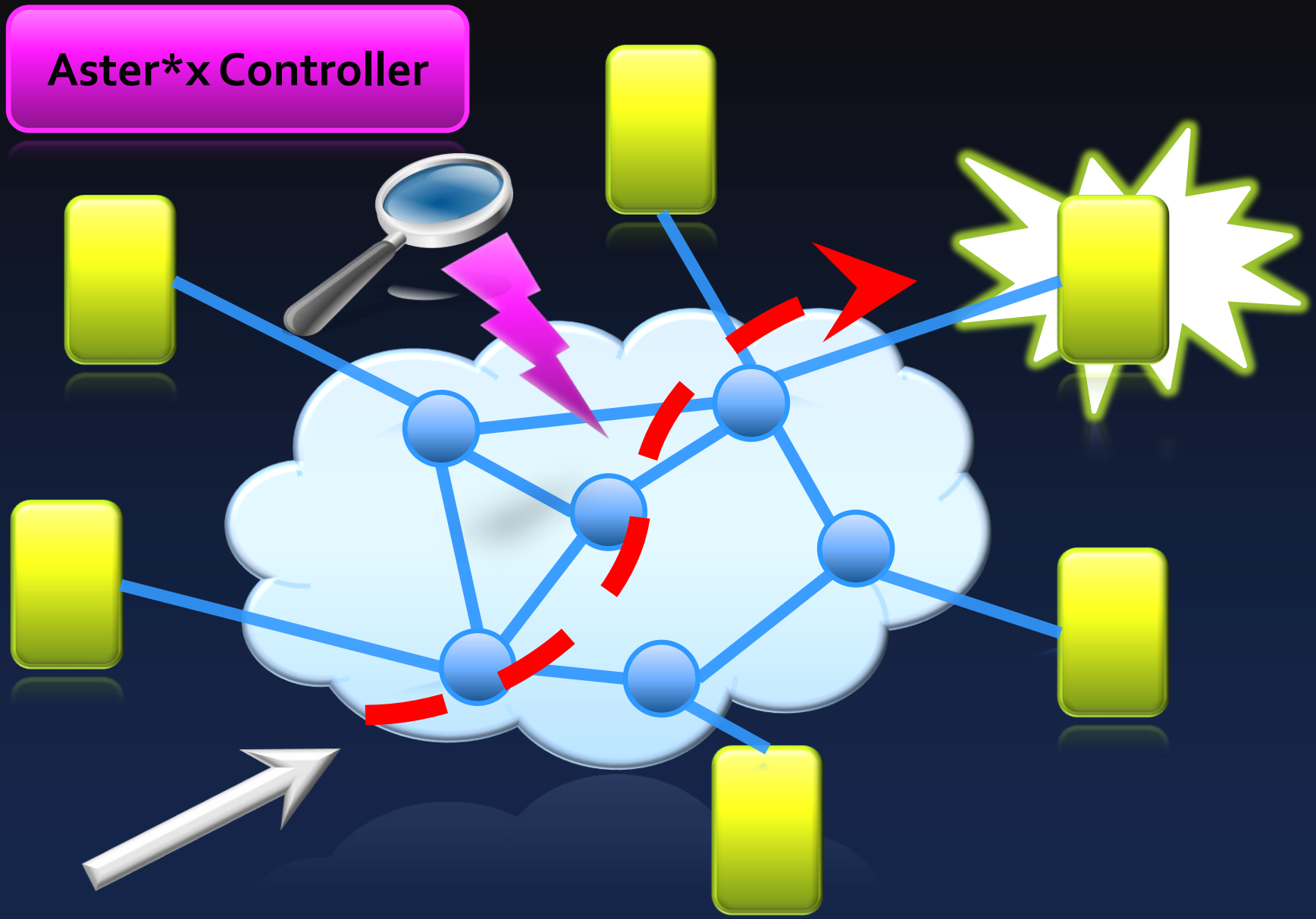
Software Defined Networking



Load Balancing is just Smart Routing

Load-balancing as a network primitive




Aster*x Controller


NLR

Aster*x Demo Video

http://www.youtube.com/watch?v=Sfqofxdk1gE

http://www.openflow.org/videos

So far…

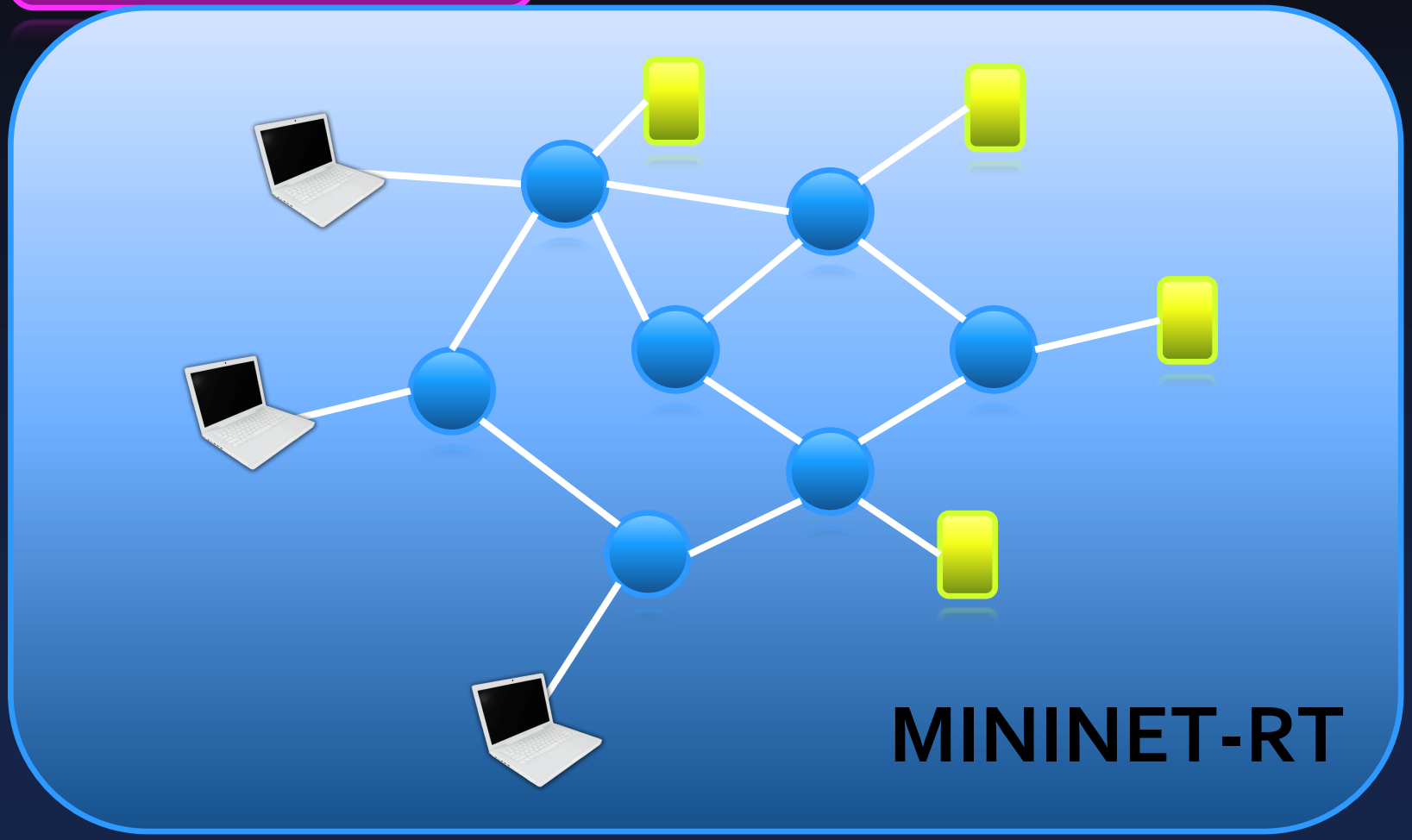
- A new architecture for distributed load-balancing
  - joint (server, path) selection
- Aster*x – a nation-wide prototype
- Promising results that joint (server, path) selection might have great benefits

What next?

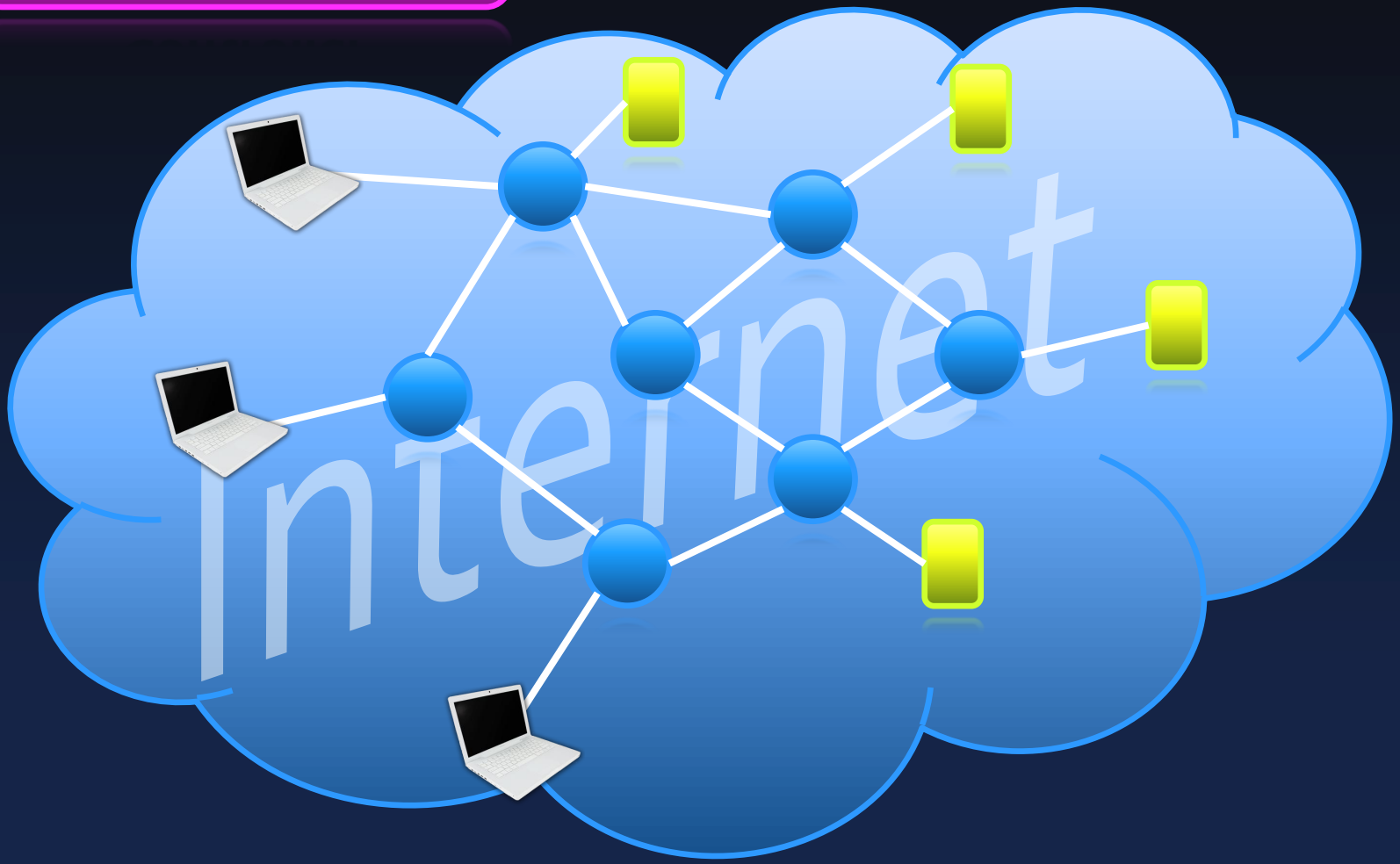
How big is the pie?

Characterizing and quantifying the performance of joint (server, path) selection

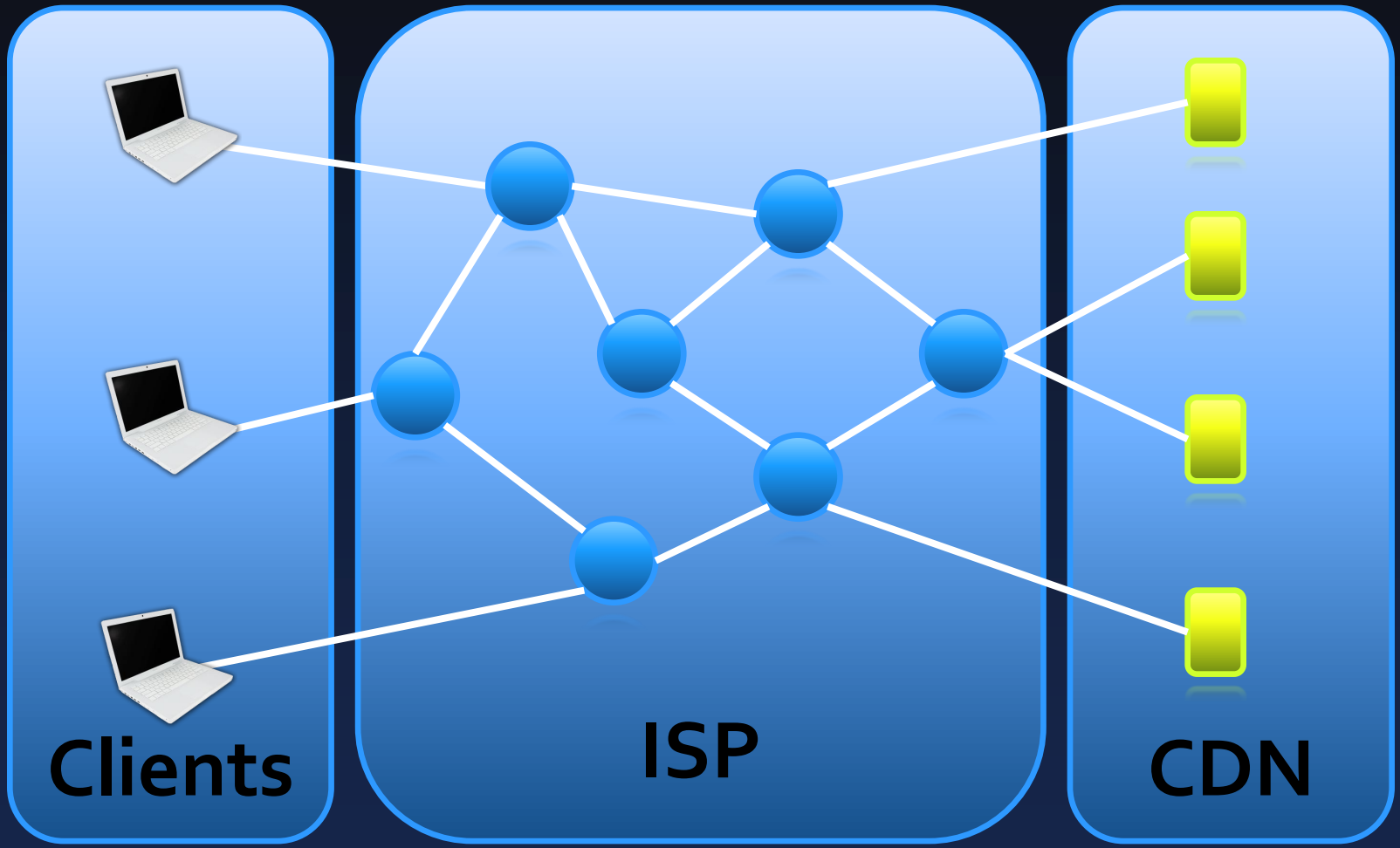
Load-balancing Controller



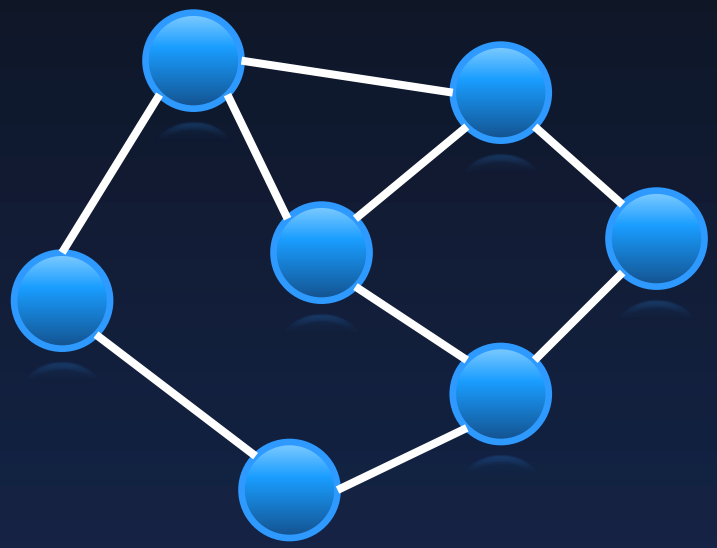
Load-balancing Controller



Model



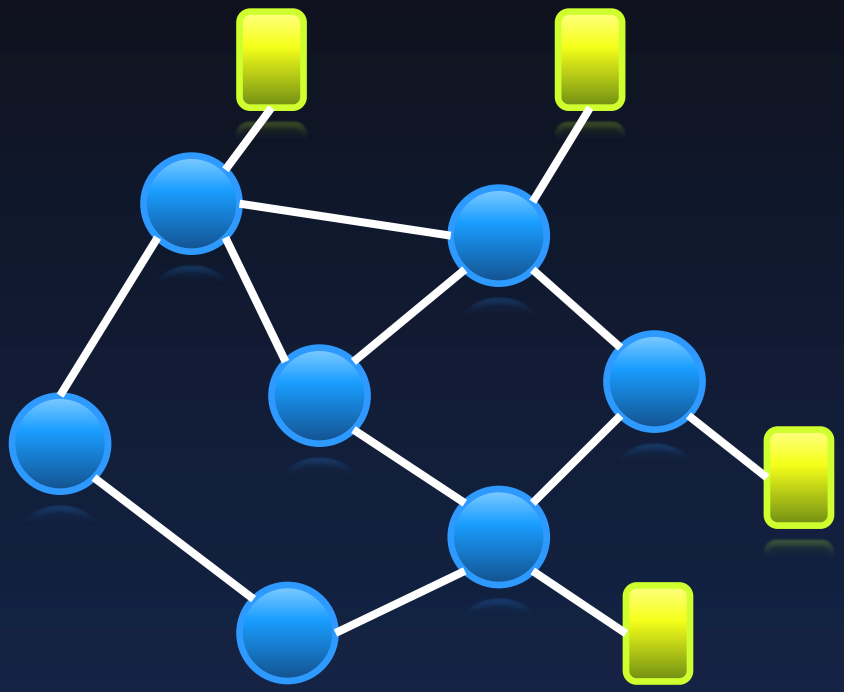
Parameters



Topology

- Intra-AS topologies
  - BRITE (2000 topologies)
  - CAIDA (1000 topologies)
  - Rocketfuel (~100 topos.)
- 20-50 nodes
- Uniform link capacity

Parameters



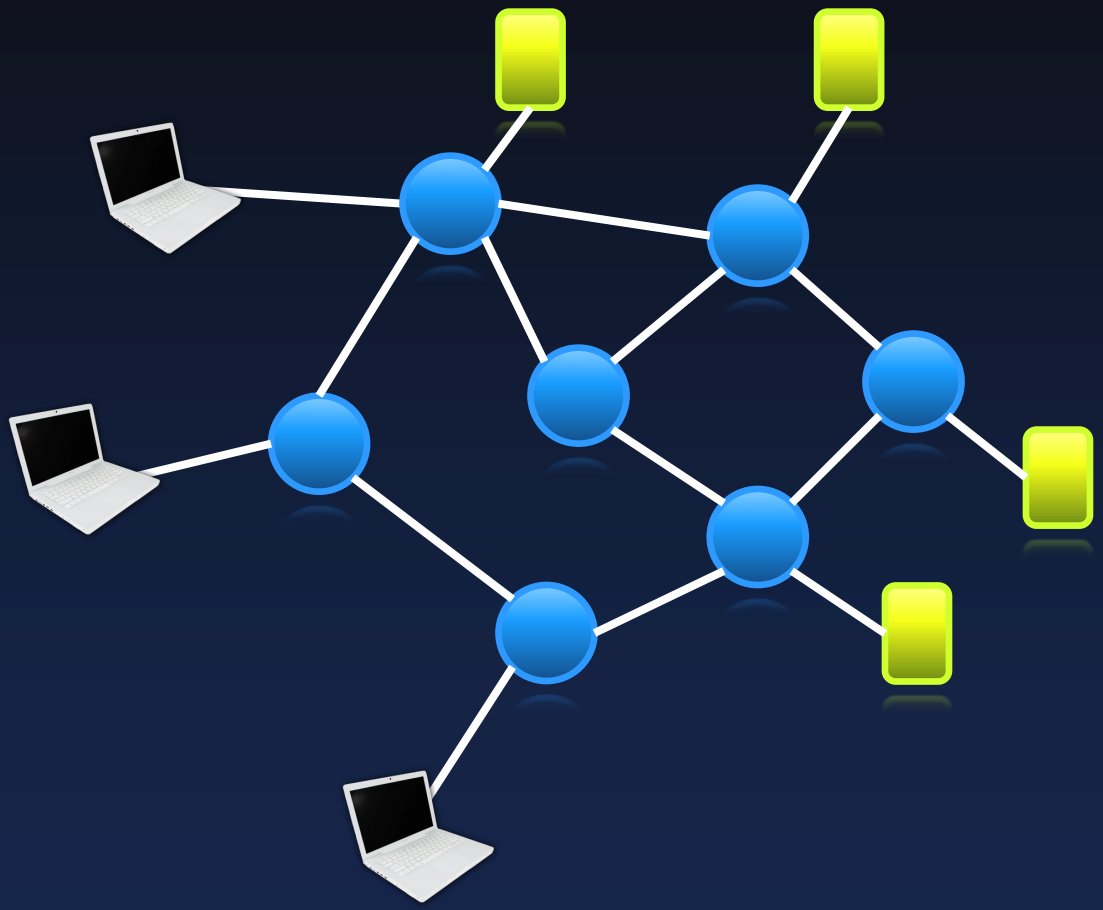
Servers

- 5-10 servers
- Random placement

Service

- Simple HTTP service
- Serving 1 MB file
- Additional server-side computation

Parameters



Clients

- 3-5 client locations
- Random placement

Request pattern

- Poisson process
- Mean rate: 5-10 req/sec

Load-balancing strategies?

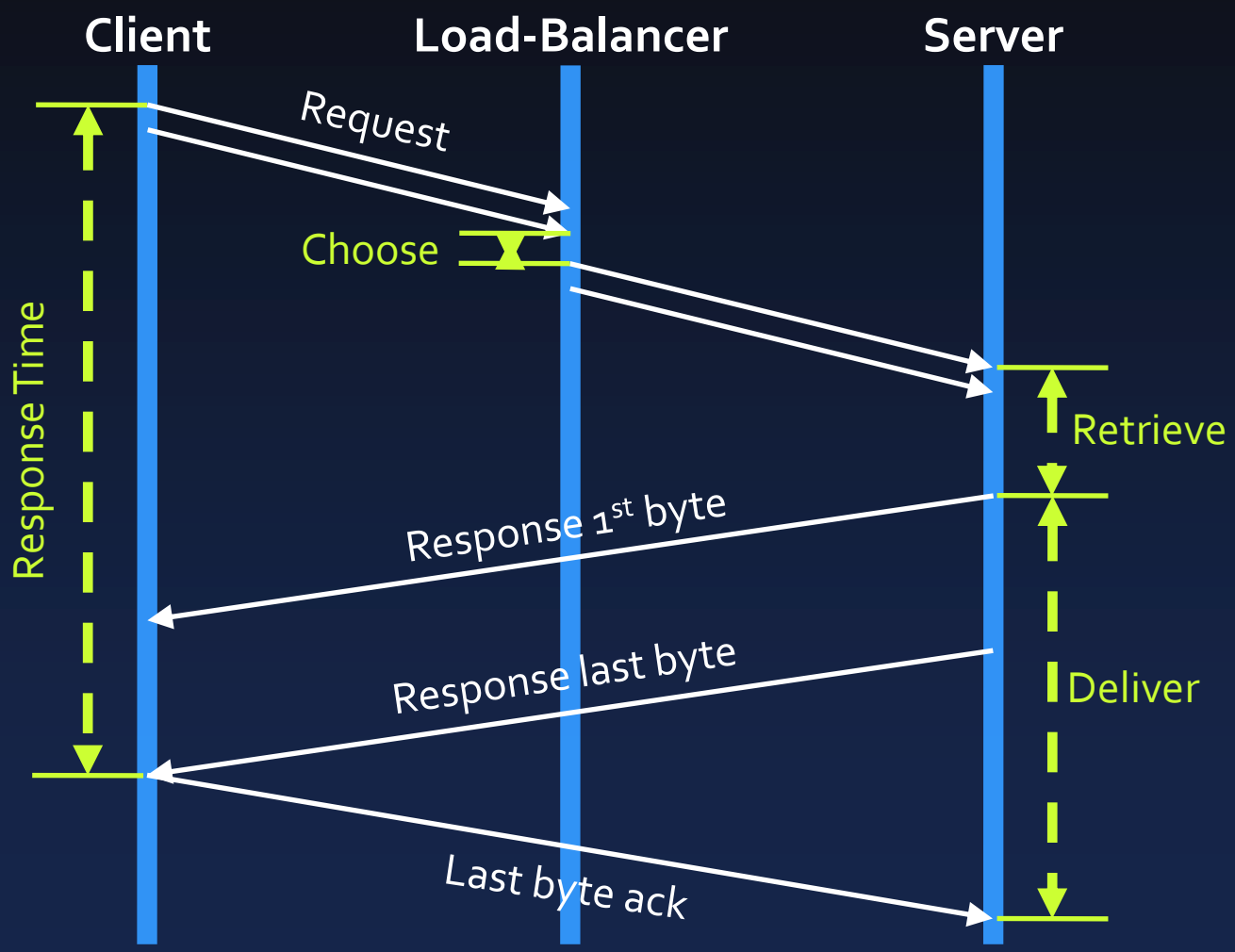
Design space

Disjoint-Shortest-Path

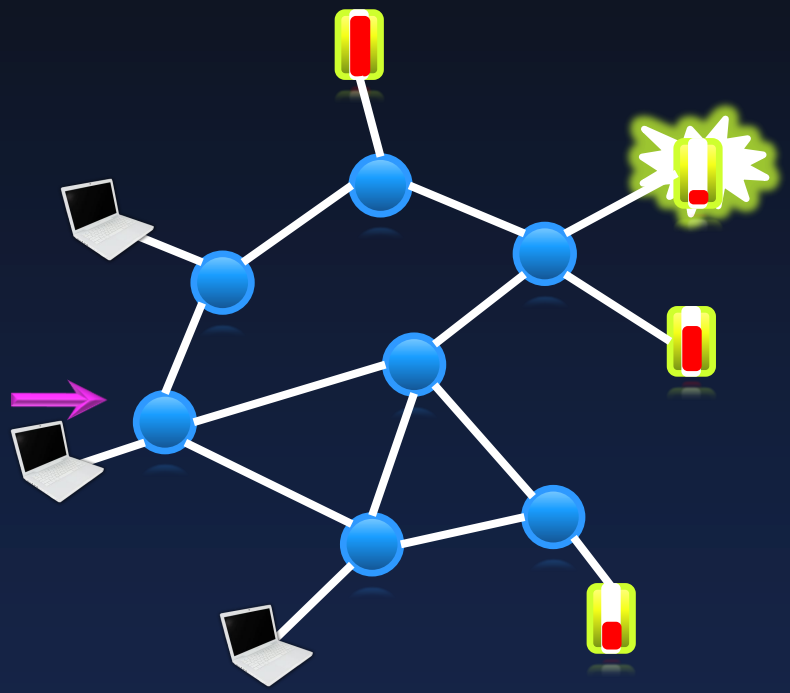
Disjoint-Traffic-Engineering

Joint

Anatomy of a request-response

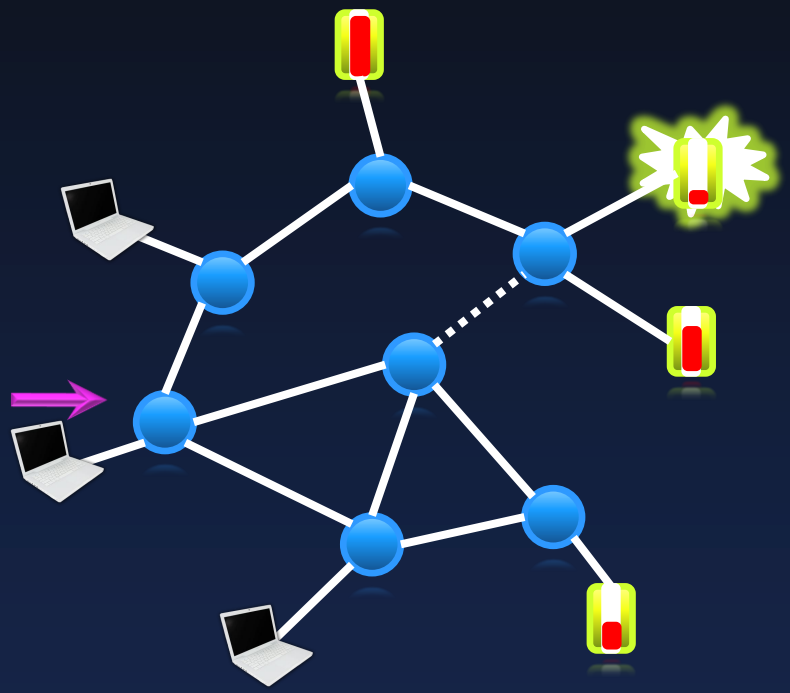


Disjoint-Shortest-Path



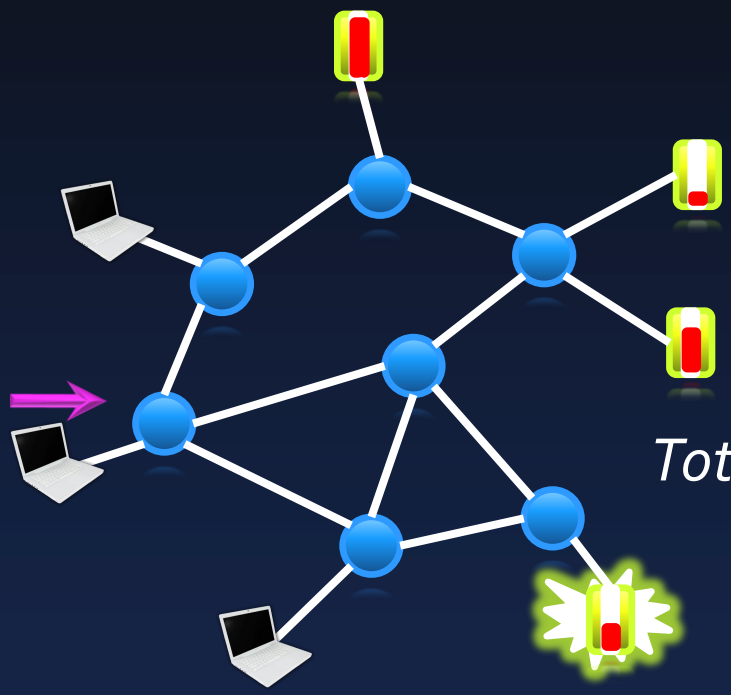
- CDN selects the least loaded server
  - *Load = retrieve + deliver*
- ISP independently selects the shortest path

Disjoint-Traffic-Engineering



- CDN selects the least loaded server
  - *Load = retrieve + deliver*
- ISP independently selects path to minimize max load
  - Max bandwidth headroom

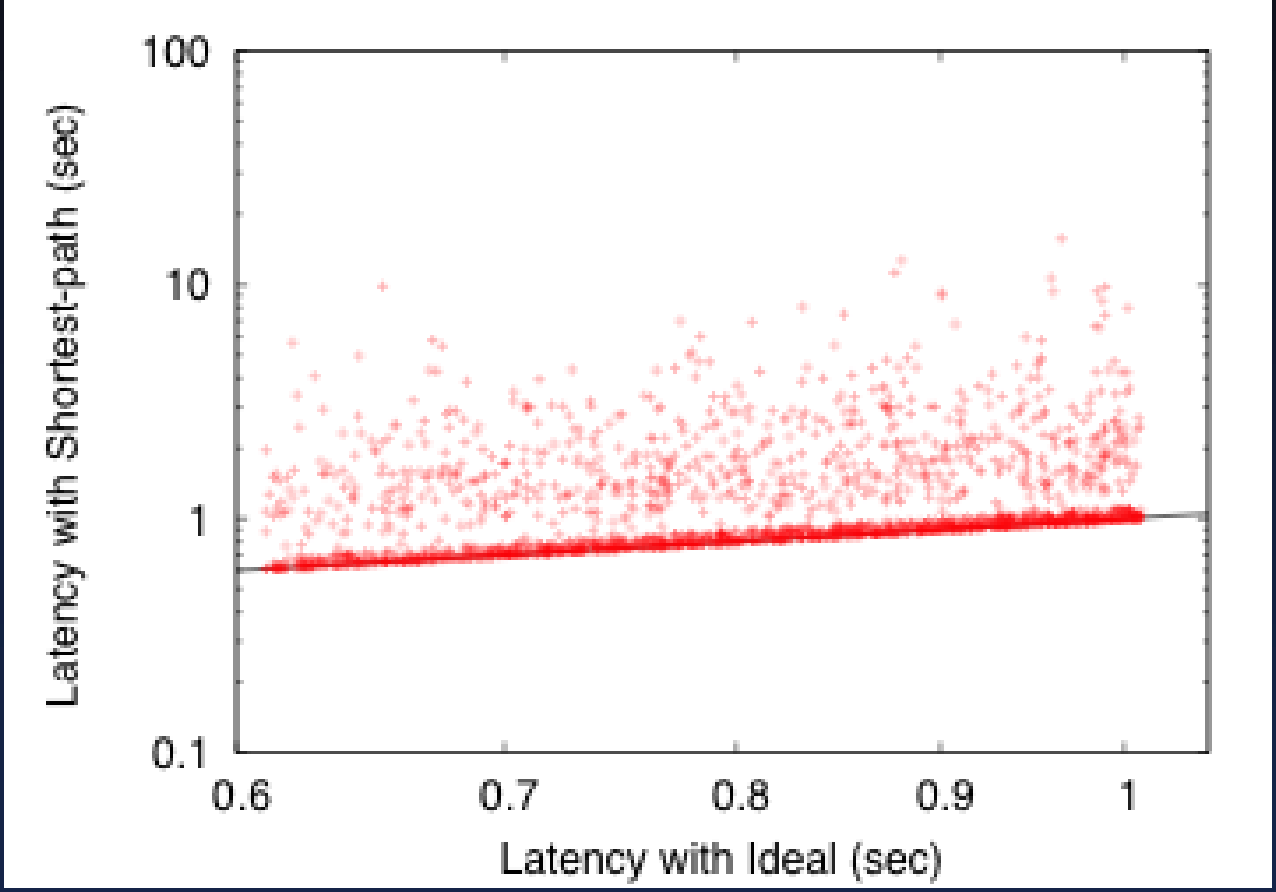
Joint



- Single controller jointly selects the best (server, path) pair

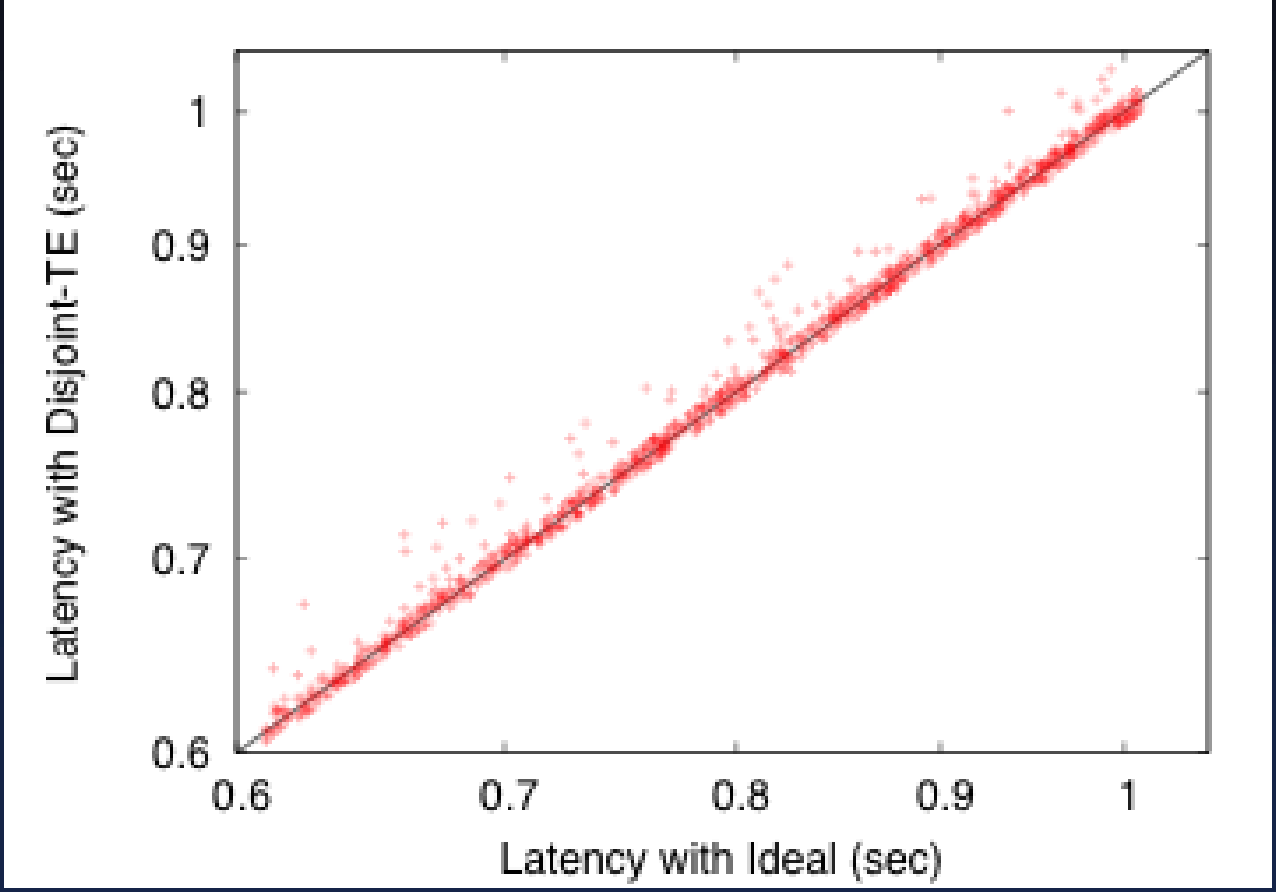
Total latency = retrieve + estimated deliver

Disjoint-Shortest-Path vs Joint



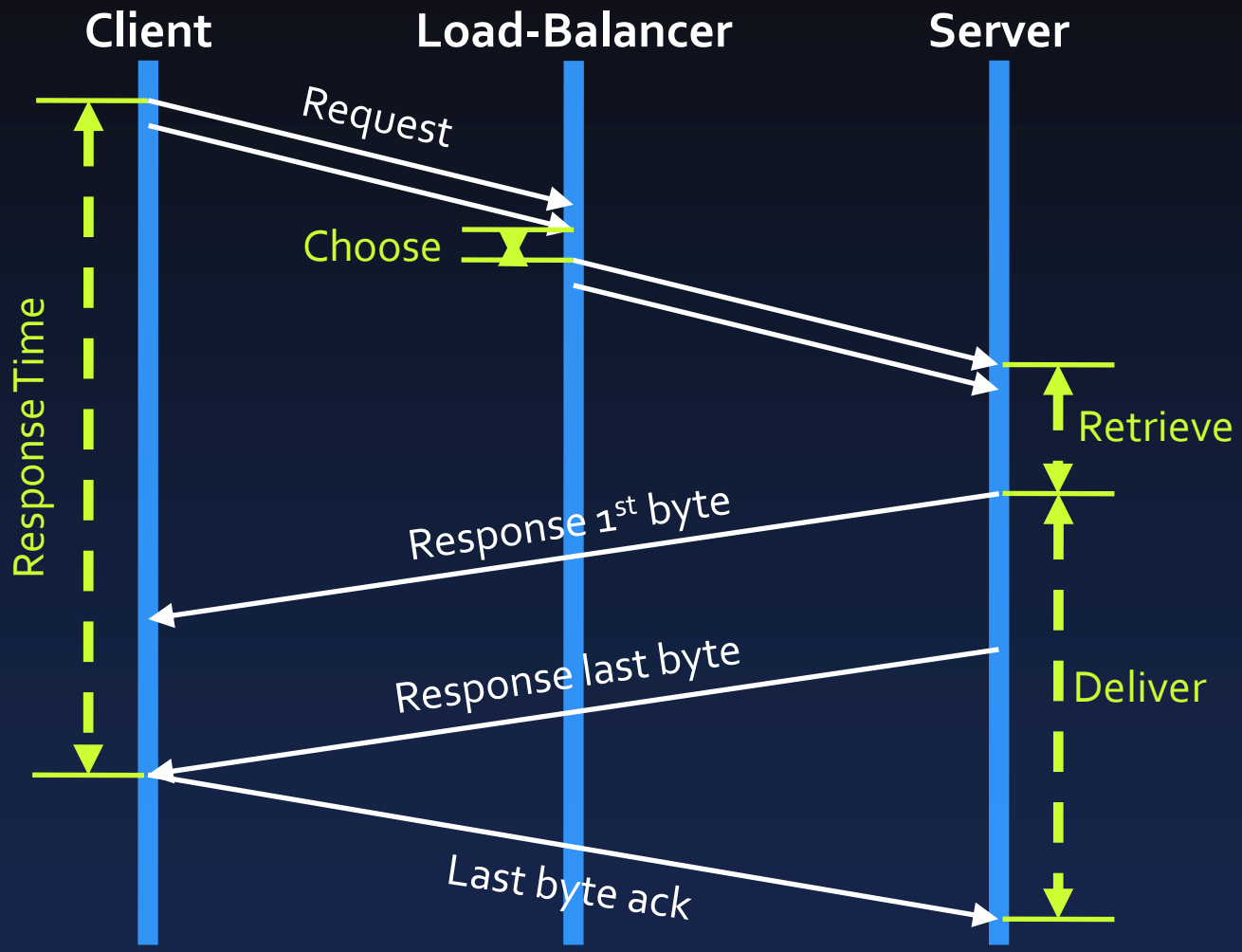
Disjoint-Shortest-Path performs ~2x worse than *Joint*

Disjoint-Traffic-Engg. vs Joint



Disjoint-Traffic-Engineering performs almost as well as *Joint*

Is *Disjoint* truly disjoint?

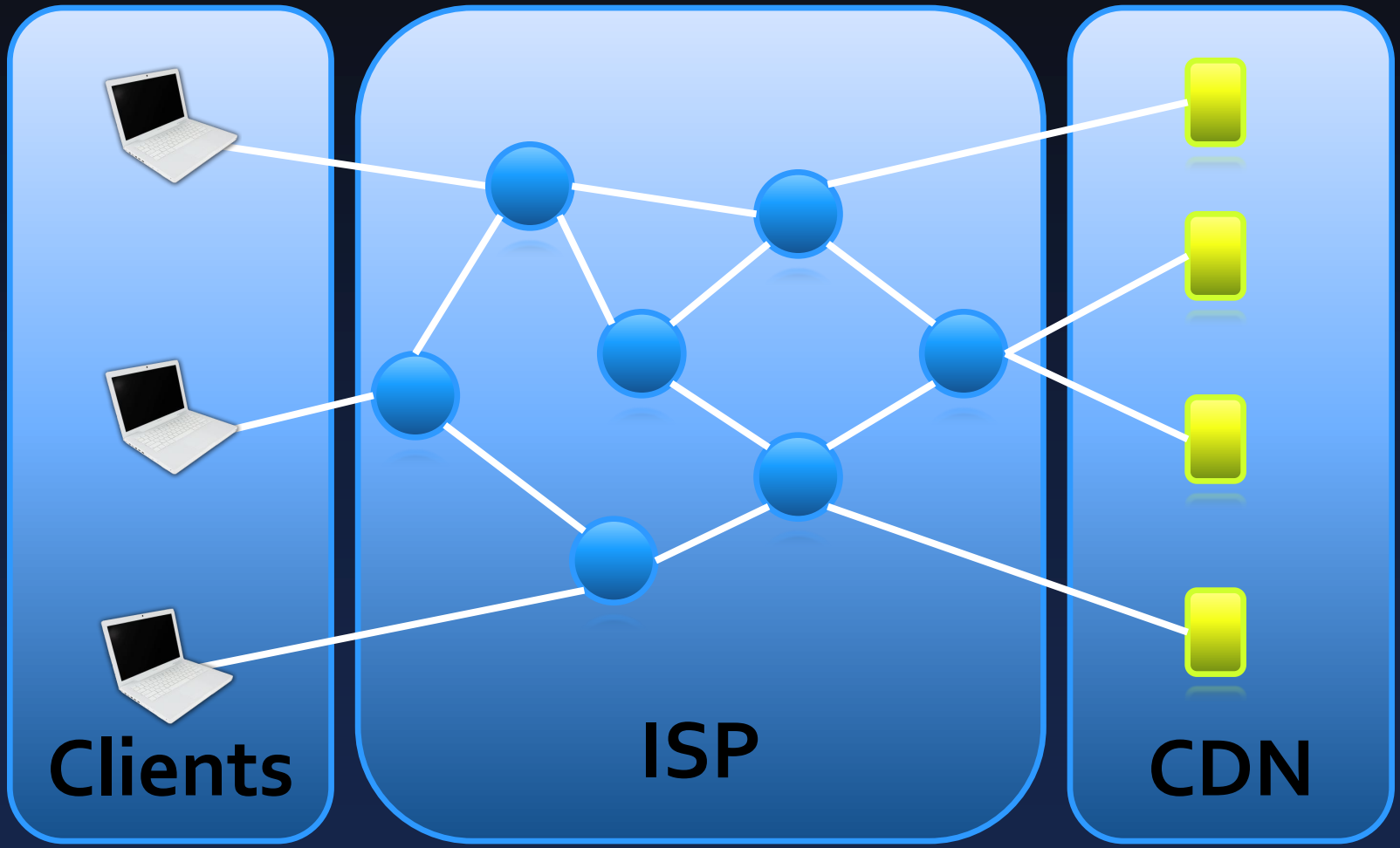


Server response time contains network information

The bottleneck effect

A single bottleneck resource along the path determines the performance.

The CDN-ISP game



The CDN-ISP game

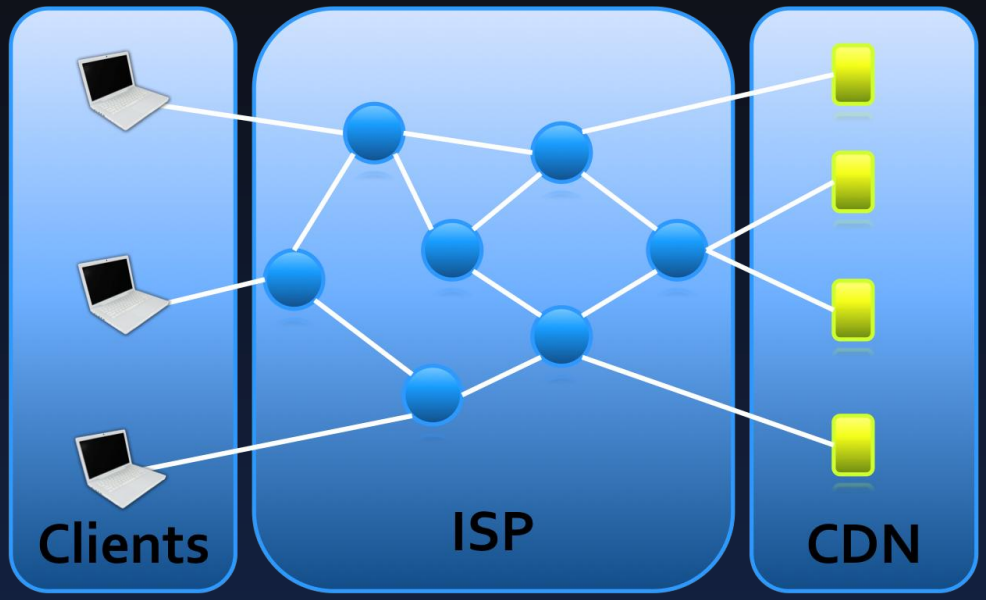
- System load monotonically decreases
- Both push system in the same direction

Summary of observations

- Disjoint-SP is ~2x worse than Joint
- Disjoint-TE performs almost as well as Joint

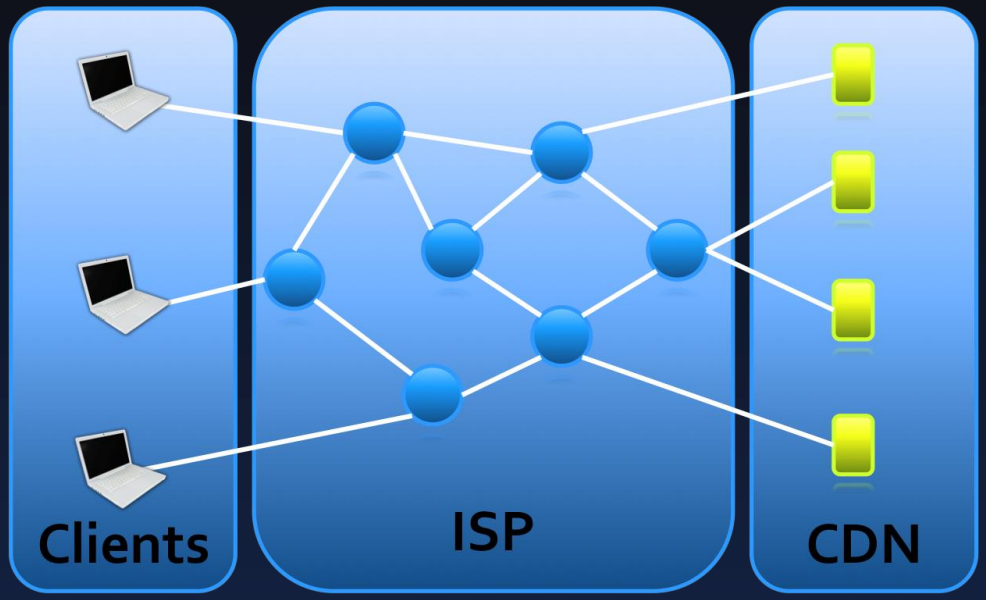
  (despite decoupling of server selection and traffic engineering)
- Game theoretic analysis supports the empirical observation

Questions for you!



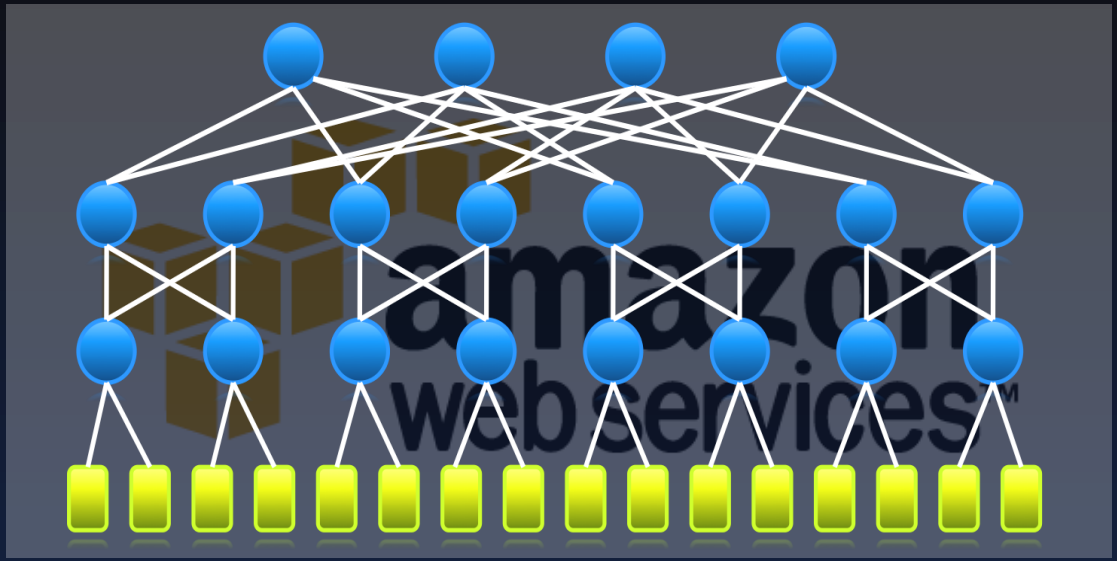
- How should I change the model to mimic a real CDN?

Questions for you!



- How can I get real data?
  - What network topologies should I use?
  - How should I decide the no. of servers and their location?
  - How should I decide the client request pattern?

Questions for you!



- How can I try it out in your network?
  - Elastic Load Balancing in EC2
  - Amazon CloudFront

Conclusion

- A new architecture for distributed load-balancing
  - joint (server, path) selection
- Aster*x - a nation-wide prototype
- Interesting preliminary results
- Future – Evaluation with real data

Let’s chat more!