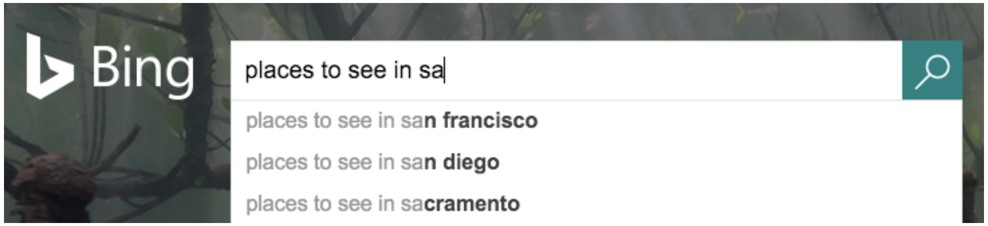
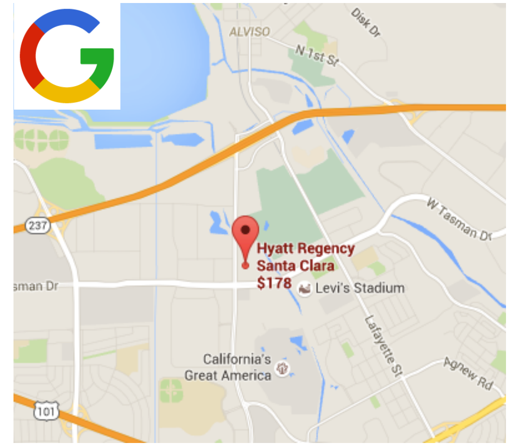
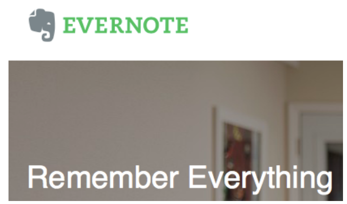
Efficiently Delivering Online Services over Integrated Infrastructure

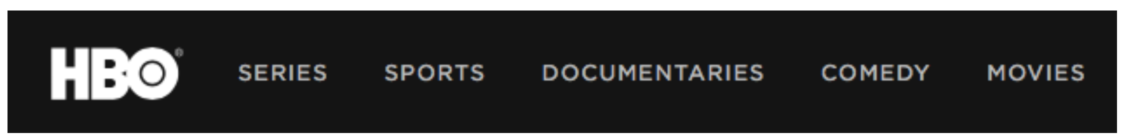
Hongqiang Harry Liu, Raajay Viswanathan, Matt Calder
Aditya Akella, Ratul Mahajan, Jitendra Padhye, Ming Zhang

Online Services

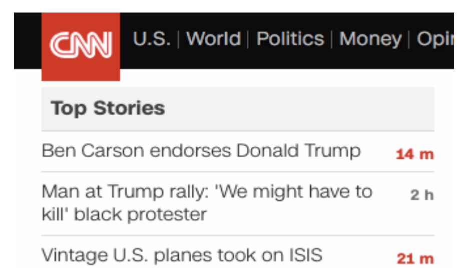


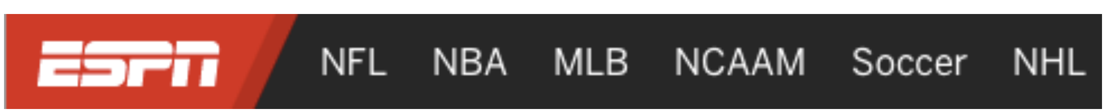




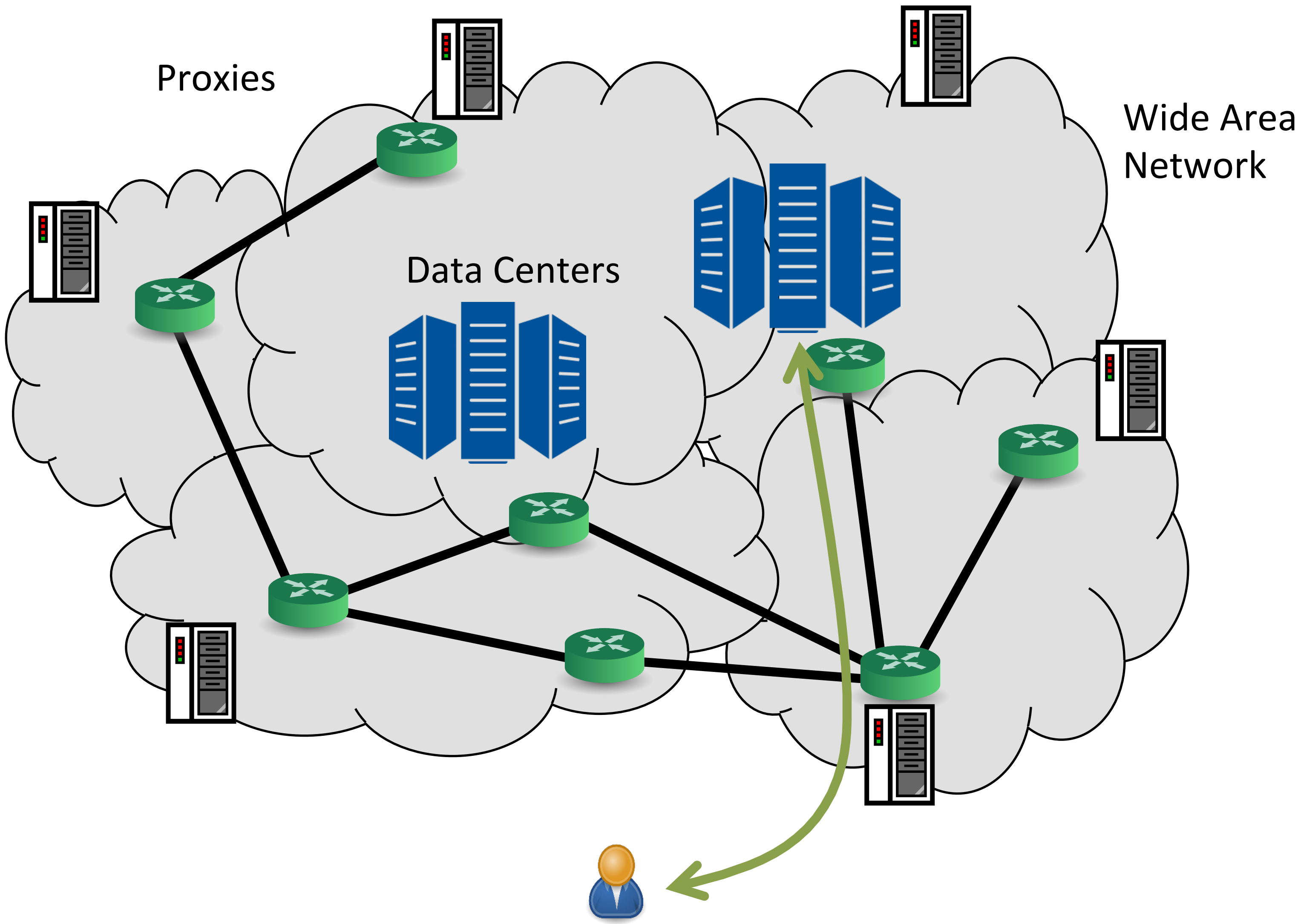


stackoverflow

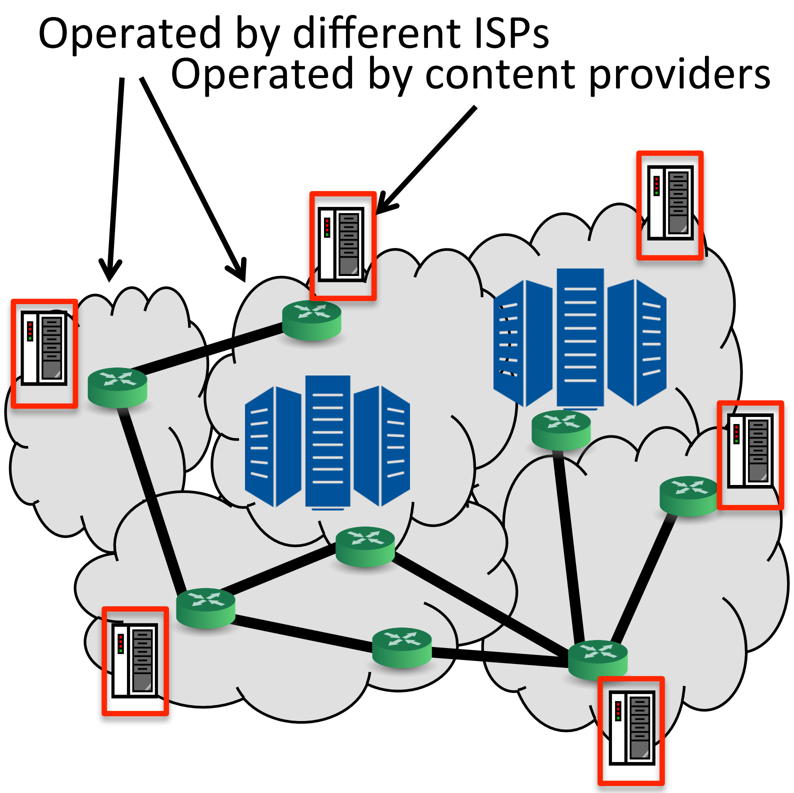




Online Service Delivery Infrastructure

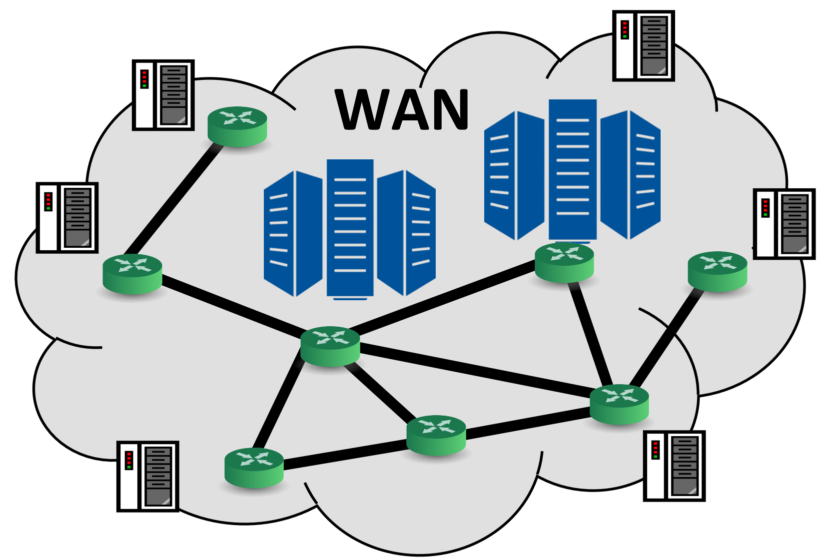


Online Service Delivery is Evolving



Multiple owners

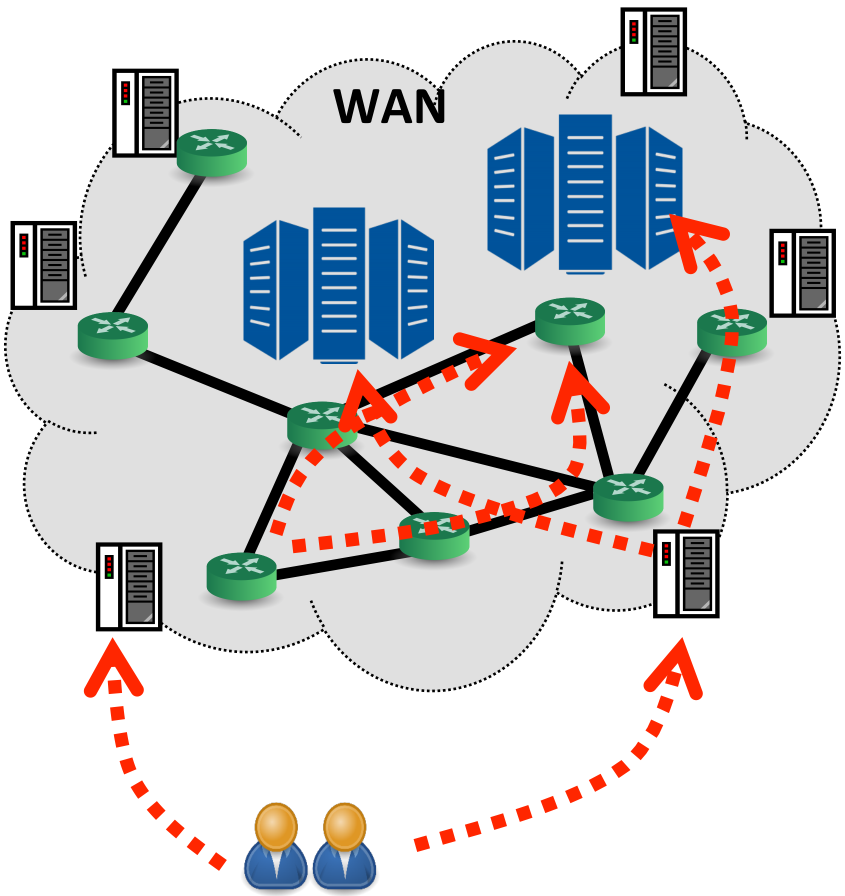
Traditional infrastructure



Owned by a single entity

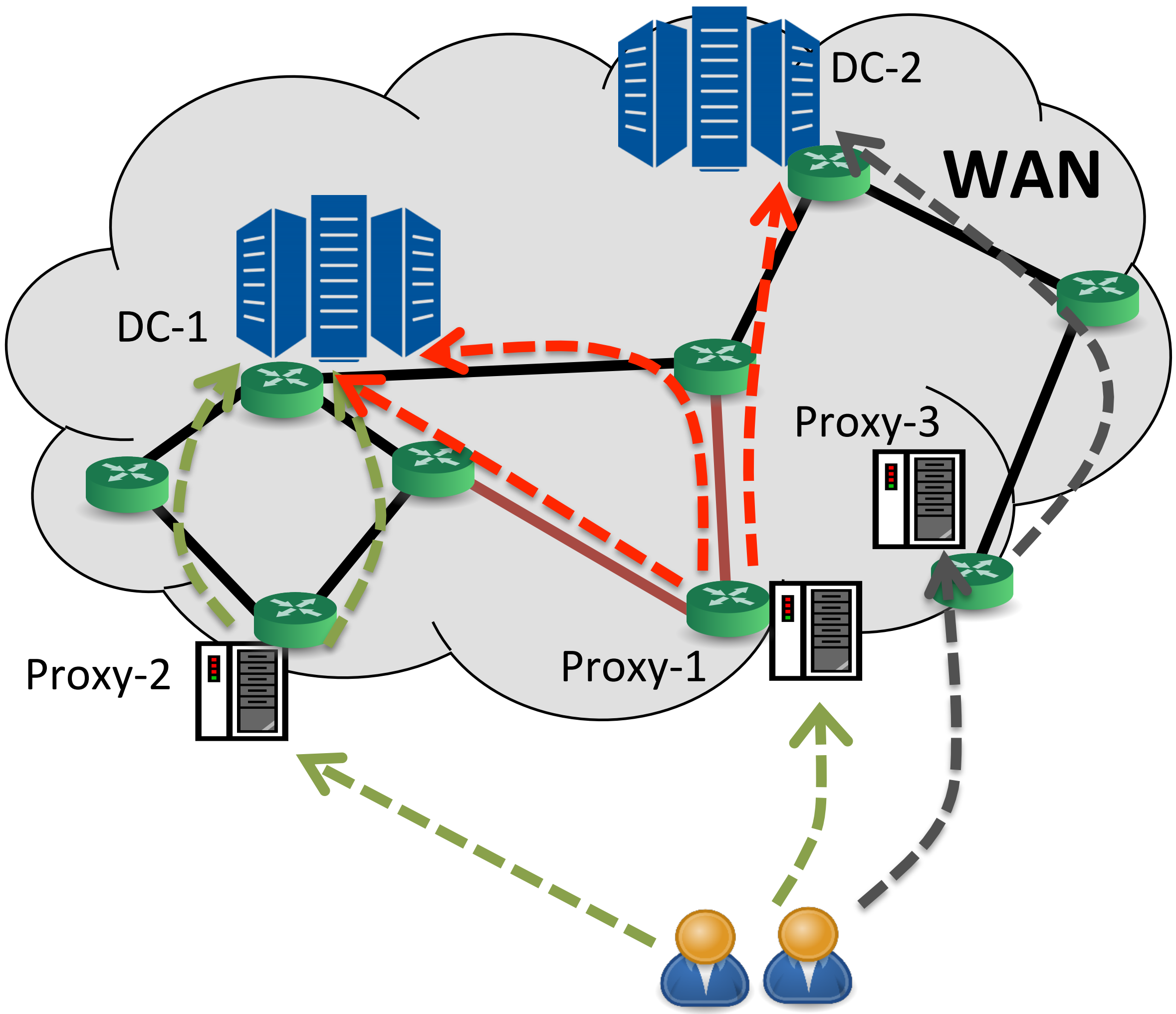
Integrated infrastructure

Integrated Infrastructure Enables Joint Control of All Decisions



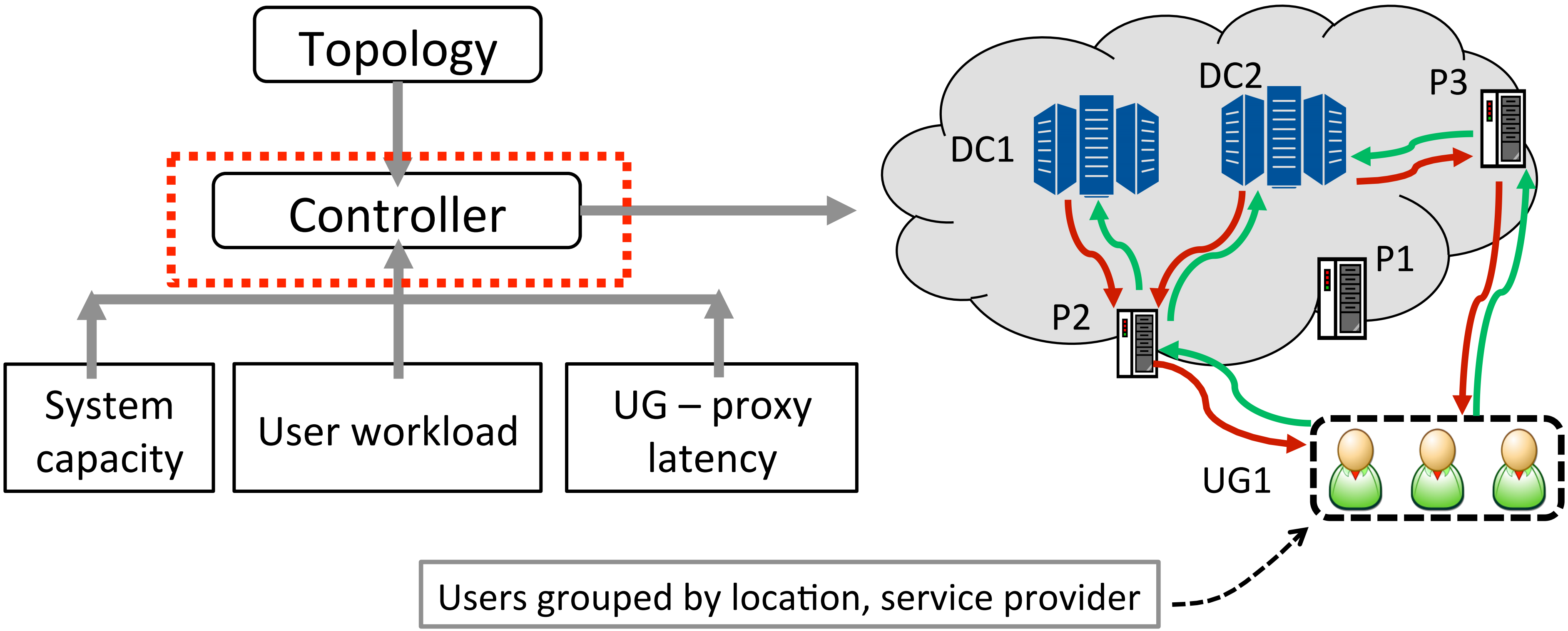
1. User – Proxy mapping
2. Proxy – DC mapping
3. Paths in the wide area network

Advantages of Joint Control



- Increase **efficiency**: total traffic without congestion
- Improve **performance**: aggregate end-to-end latency

Footprint: Jointly Controls the Integrated Infrastructure



Control decisions for a user group:

- UG—proxy mapping
- proxy—DC mapping
- network paths

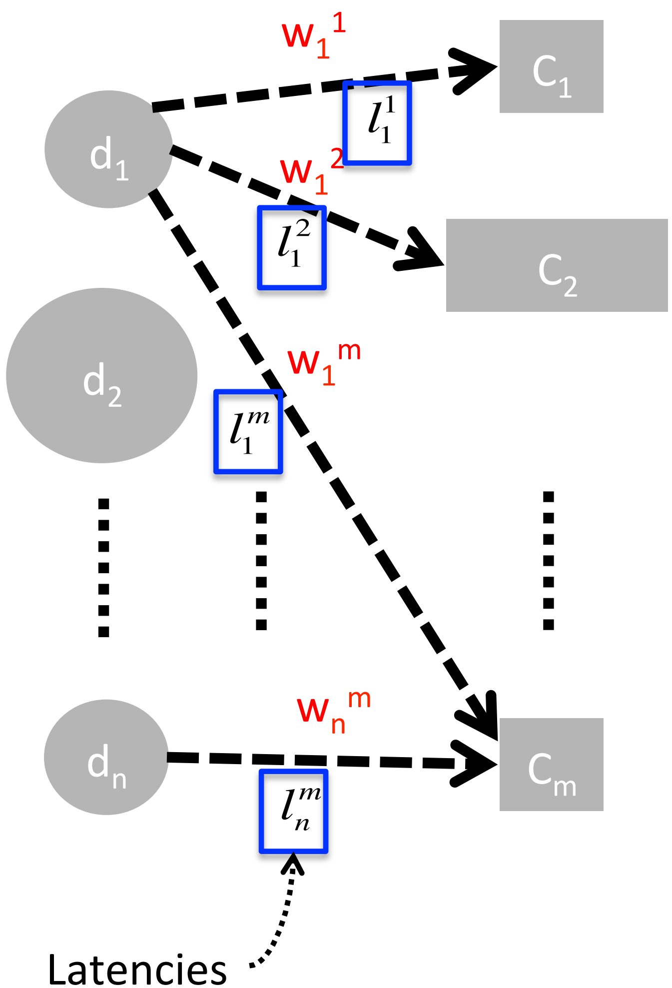
Goals:

- Maximize congestion free traffic
- Minimize end-to-end latency

Outline

- Challenges in computing forwarding configuration
- Other challenges in realizing Footprint
- Evaluation

Computing Configuration: Basic Approach



Estimated user demands for an epoch

$$\{d_1, d_2, \ldots, d_n\}$$

Resource capacities

$$\{C_1, C_2, \ldots, C_m\}$$

Estimated load from user ‘u’ on resource ‘r’

$$n_u^r = d_u . w_u^r$$

Capacity constraint for resource ‘r’

$$n^r = \sum_u n_u^r \leq C_r$$

Linear Program

Objective

maximize $$\sum_u \sum_r n_u^r - \sum_u \sum_r l_u^r n_u^r$$

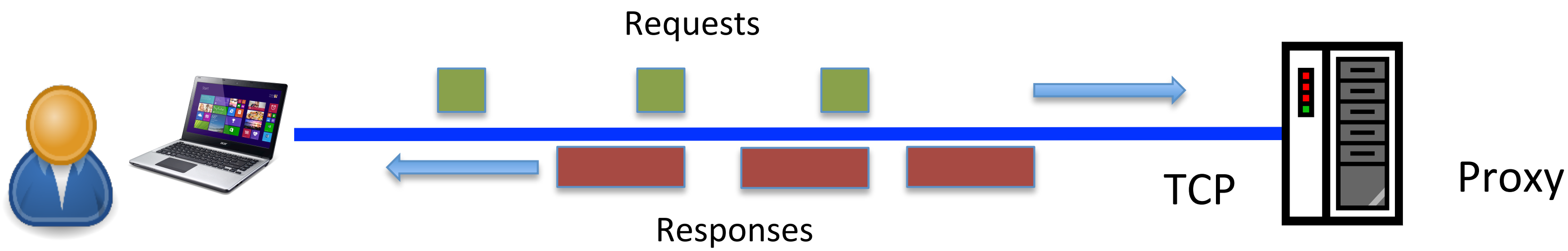
Does such a simple model suffice ?

No

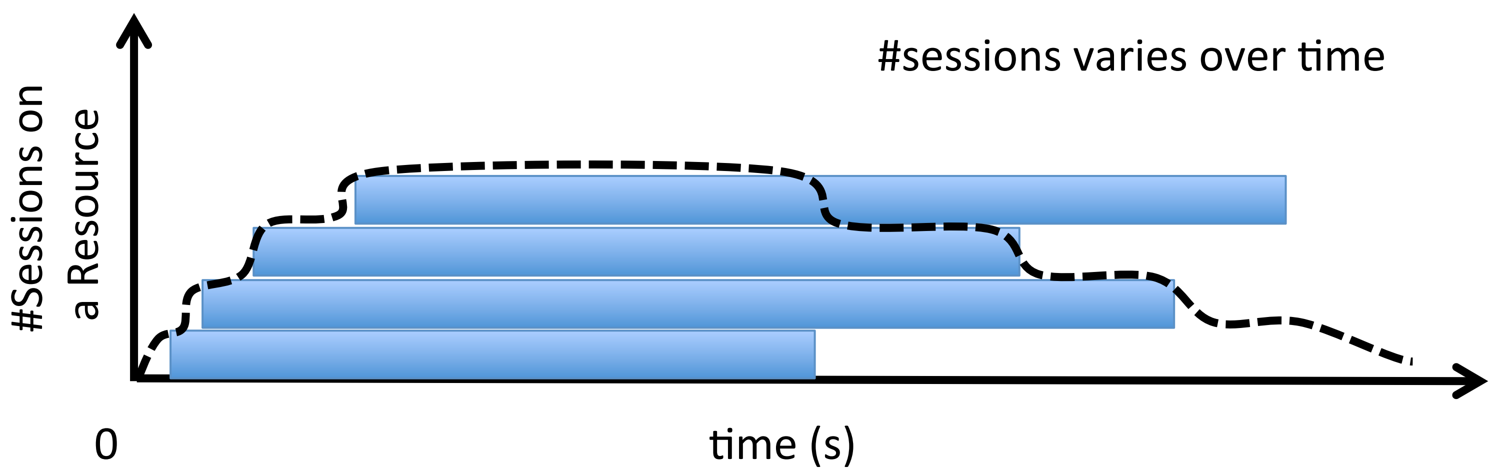
Because of the nature of traffic from different online applications

User Traffic Arrives over Sessions

- Multiple requests and responses over a single session

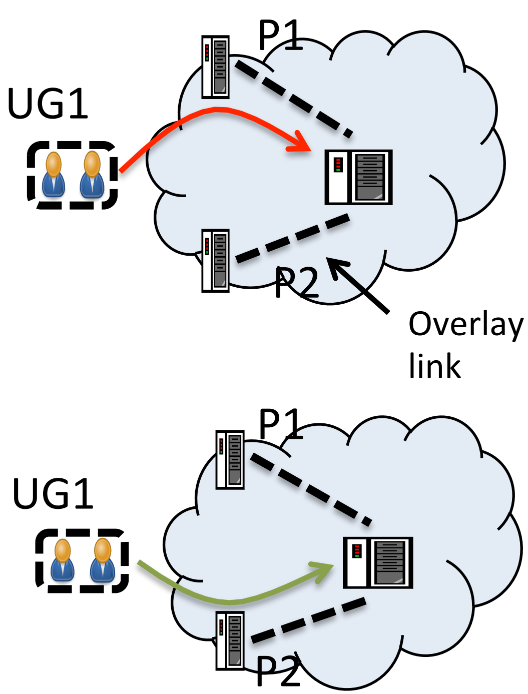


- Sessions are long-lived and arrive all through the duration of an epoch

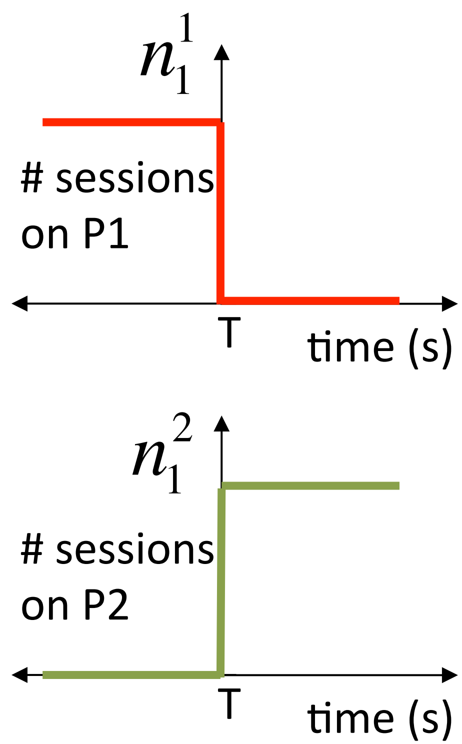


Session Stickiness

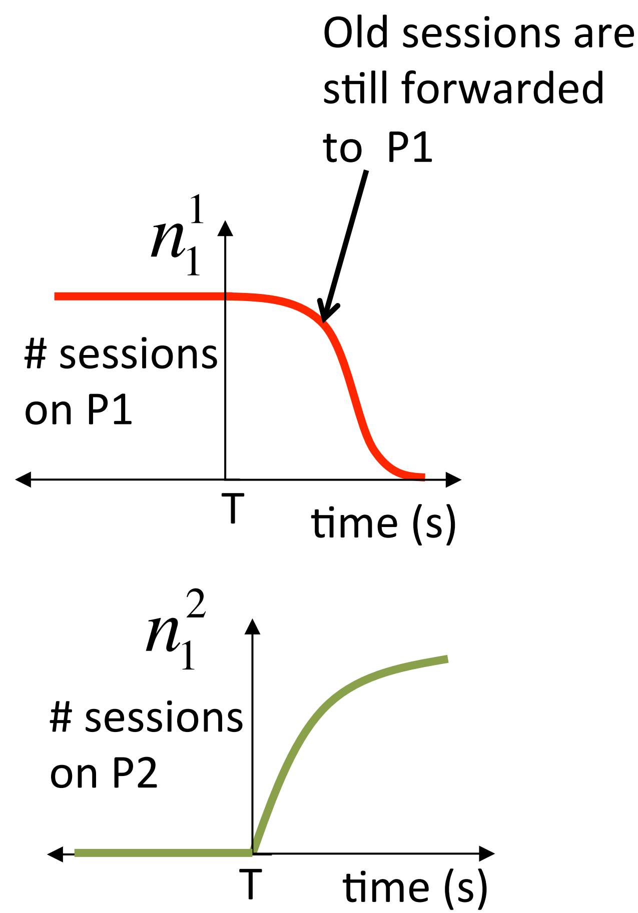
Sessions **stick** to proxy and DC
- Long lived TCP sessions
- No fresh DNS query in the middle of a session



Switch traffic at t = T



Non-sticky sessions



Sticky sessions

Challenge: Temporal Variation of Load

1. Non-zero session lifetime
2. Session stickiness

$\Big\}$ **Gradually varying** load from a user group to a resource

- Resource capacity constraints should be satisfied during entire epoch

$$\sum_u n_u^r \le C_r \longrightarrow \sum_u n_u^r \boxed{(t)} \le C_r$$

- Computationally infeasible if $n_u^r(t)$ does not have a closed form
  - Applications have arbitrary session life distributions

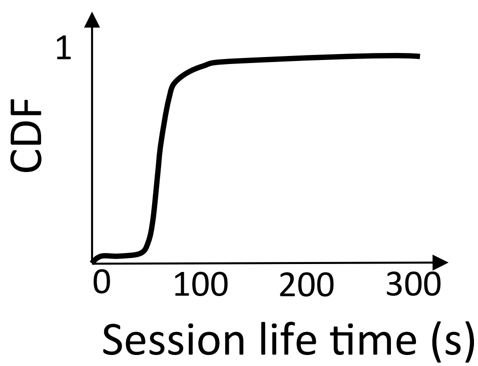
How to guarantee congestion free delivery for traffic on sessions?

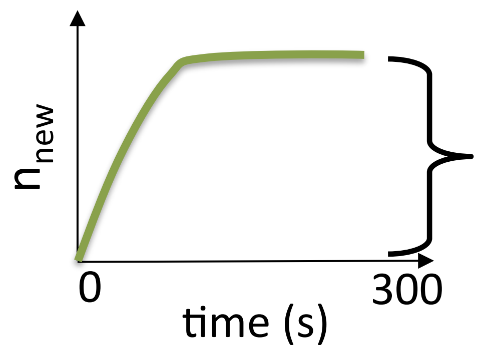
High Fidelity Modeling of Load

$$n^r(t) = n^r_{new}(t) + n^r_{old}(t)$$

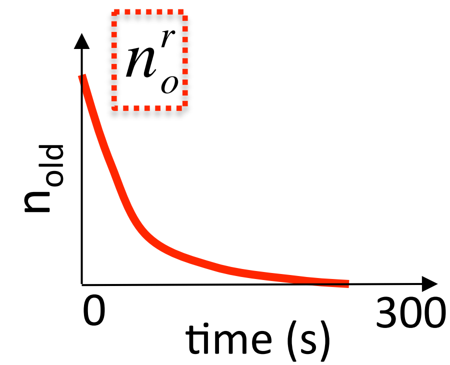
current previous

Always holds this pattern





Proportional to arrival rate of new sessions



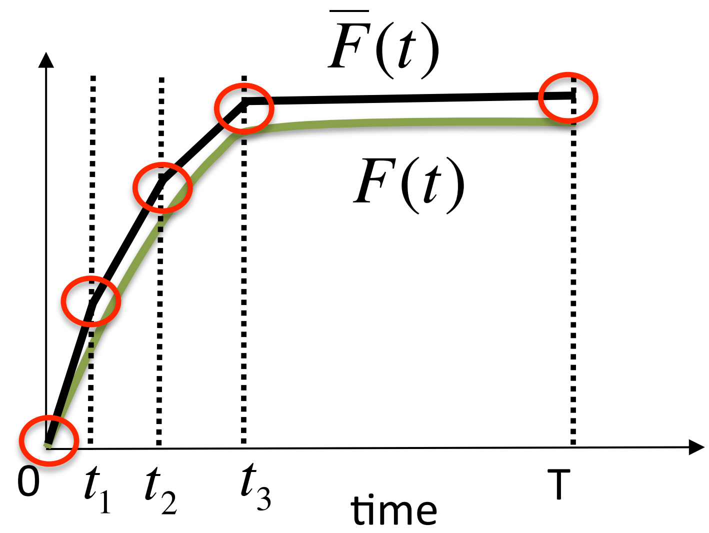
$$n^r(t) = \boxed{\lambda^r} F(t) + n^r_o G(t)$$

Arrival rate of sessions (decision variable)

Pattern Functions ← Session length distribution

Discretizing the Temporal Model

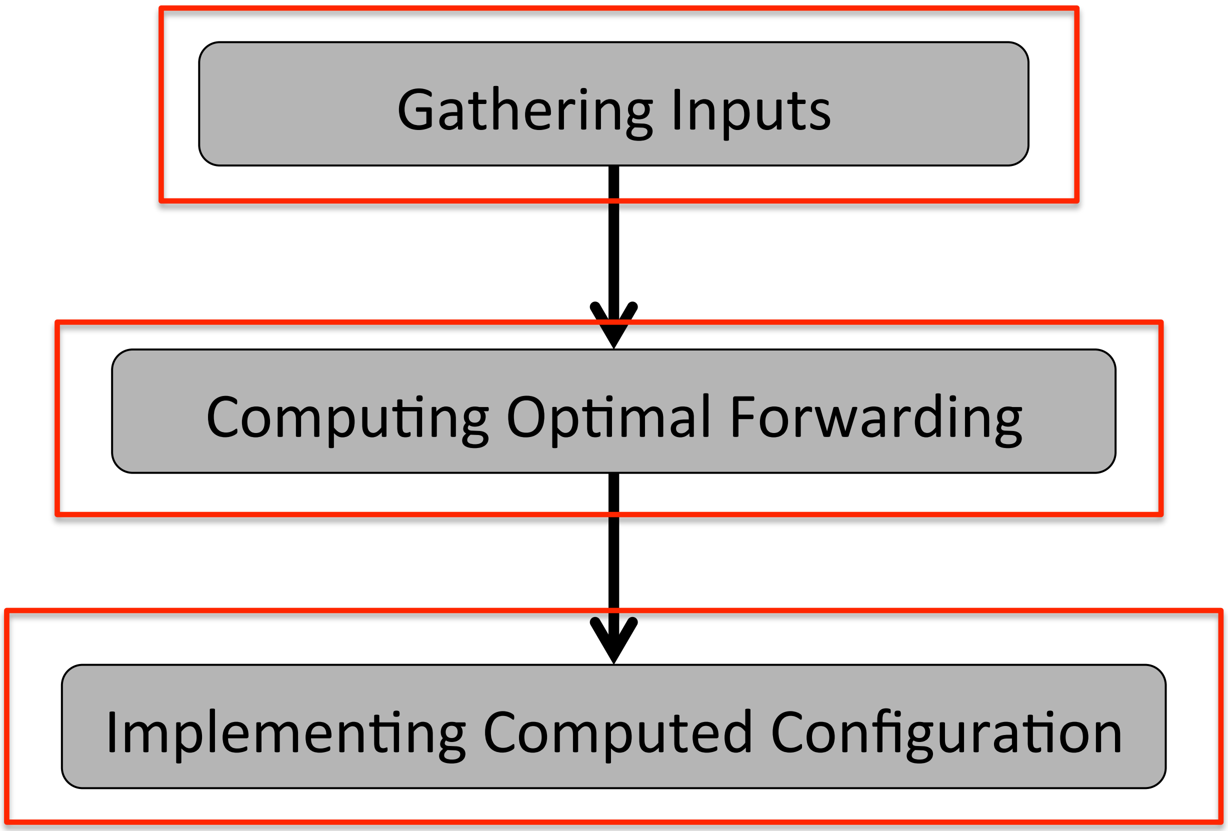
Approximate $F(t)$ by a tight piecewise linear **upper bound,** $\overline{F}(t)$



$$n^r(t) \le \overline{n}^r(t) = \lambda^r \overline{F}(t) + n_0^r G(t)$$

- $\overline{n}^r(t)$ has maximum at one of the corners
- Capacity constraints have to be checked only at fixed set of points
- **Optimal** λ^r's obtained by solving a **linear program**

Footprint: System Implementation



Footprint: Inputs to the controller

- Input data collected every 5 minutes
- Inputs:
  - User group – proxy latency measurements
    - Piggy-back on end-host applications
    - Instrumented JavaScript on bing.com webpage [Calder *et al.,* IMC 2015]
  - User workload
    - Estimated using observed workload in prior epochs
  - System health status
    - From Microsoft internal system monitoring pipelines
- Deployed in production

Implementing Computed Configuration

- UG—proxy mapping: DNS (BIND)
- Proxy—DC mapping: Custom software to change configuration
- WAN path selection: OpenFlow
- Prototyped on a modest-sized testbed

Evaluation

1. Joint Decisions

2. Temporal Modeling

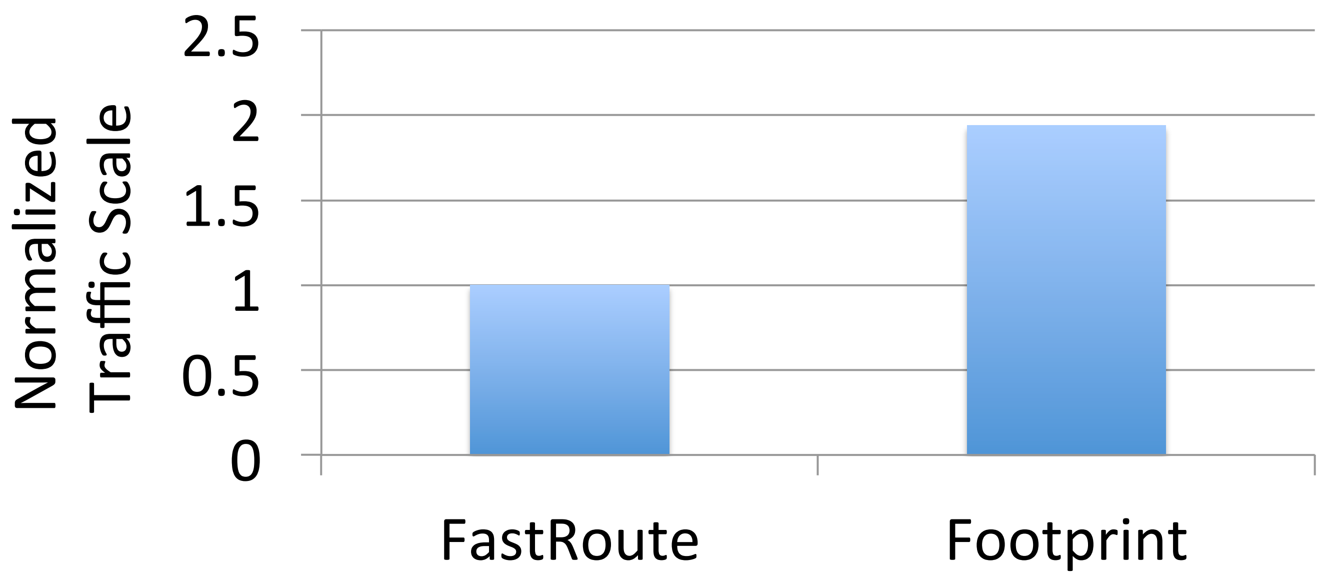
Evaluation Setup

- Trace driven simulations
- Data
  - Taken from production deployment of Footprint
  - One week worth of data
  - Multiple topologies (North America, Europe)
- Scale
  - O(10k) user groups
  - O(100) routers and links
  - O(100) proxies
  - O(10) data centers
- Metric
  - Efficiency: Maximum traffic with no congestion
  - Performance: Aggregated end-to-end latency

Evaluation: Efficiency of Joint Control

FastRoute [Flavel *et al.*, NSDI 2015]

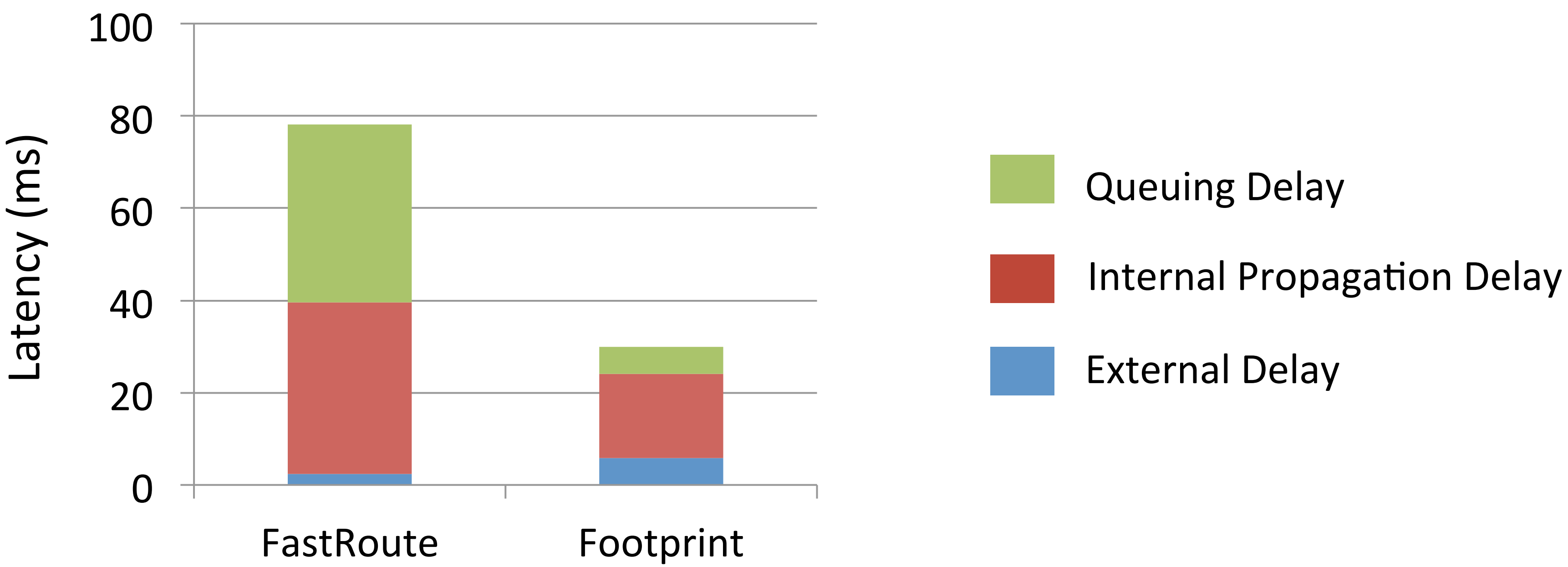
- UG—proxy: Closest proxy decided by Anycast routing
- Proxy—DC: Closest proxy based on active measurements
- WAN path selection: Independent traffic engineering module



- **Footprint can carry 2x more load** because user traffic is diverted to resources with unused capacity

Evaluation: Latency Improvement

Compare end-to-end latency at 70% capacity of FastRoute



Footprint decreases overall latency by ~60%

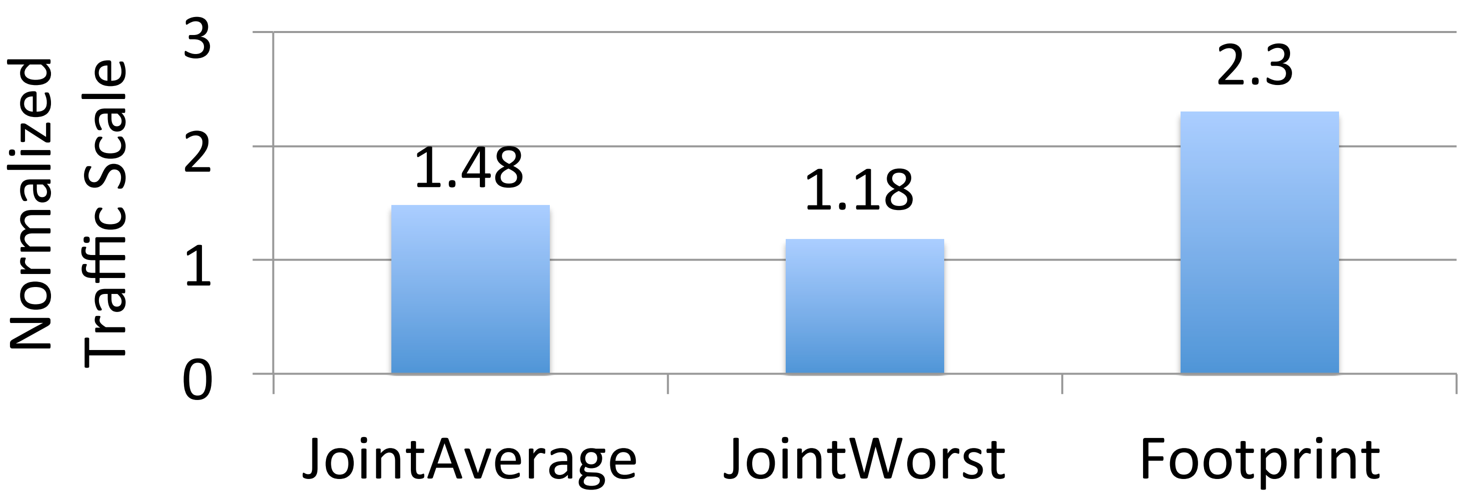
Evaluation: Efficiency of Temporal Modeling

- Compare with non-temporal models
  - **JointAverage**:

$$n^r(t) = \lambda \text{ x Average session length}$$

  - **JointWorst:**

$$n^r(t) = \max_t (\#\text{old sessions}) + \max_t (\#\text{new sessions})$$



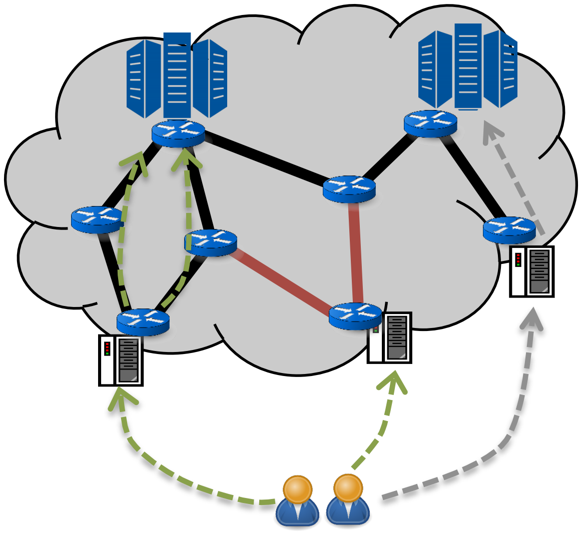
More than 50% gains with respect to non-temporal models.

Related Work

- To coordinate or not to coordinate? [Narayana *et. al*, SIGMETRICS 2012]
- Cooperative world *vs* Single entity world
- Show importance of temporal load modeling

Summary

- ***Joint decision*** for proxy, DC and WAN path selection
  - 100% increase in supported users, and,
  - 60% reduction in end-to-end latency



- High fidelity temporal models 50% efficient than non-temporal models