UC SANTA CRUZ

Policy-Aware Connectionless Routing

Bradley R. Smith

J.J. Garcia-Luna-Aceves

Baskin School of Engineering

University of California, Santa Cruz

Santa Cruz Baskin School of Engineering
Engineering

UC SANTA CRUZ

Outline

- Motivation
- Internet Routing Architecture
- Proposed policy enhancements
- PACeR
  - QoS
  - Traffic Engineering
  - Retains strengths of the Internet Architecture

UC SANTA CRUZ

Internet Routing Architecture

- Distributed routing
  - Topology-driven route computation
  - Routers autonomously compute
    - "Best" path
    - to every destination
  - Manages local next-hop forwarding state
- Forwarding decisions made
  - Hop-by-hop
  - Based on destination address

UC SANTA CRUZ

Strengths

- Robust
  - Co-locates controlling process and state
  - "Fate sharing"
  - Localizes affects of topology changes
- Efficient and responsive
  - "Simplex" control communication
  - "Partial-topology" routing

UC SANTA CRUZ

Limitations

- Single traffic class
  - Only compute one (“best”) route
  - Forwarding state for only one route
- Homogeneous assumption
  - Performance requirements
  - Network resource usage policies
  - This is a bad assumption!
- **Need support for multiple paths per destination**
  - Routing
  - Forwarding

UC SANTA CRUZ

Quality-of-Service

- E.g. Video-on-Demand vs. IP telephony
  - VoD → high b/w, delay insensitive
  - VoIP → low b/w, low delay
- 2 paths (b/w, delay)
  - 200kbps, 10msec
  - 2Mbps, 1sec
- “Best” depends on application
- Homogeneous performance requirements is bad assumption!

UC SANTA CRUZ

Traffic Engineering

- Routing with resource usage constraints
  - Service differentiation (bronze, silver, gold)
  - Resource allocation (sales, engineering, etc.)
  - Security (unclassified, secret, top-secret)
- Route over traffic-class specific paths
- Homogeneous network usage policies is bad assumption!

UC SANTA CRUZ

Previous Solutions

- Traffic classification by ingress router
  - Performance requirements
  - Resource-usage policies
- On-demand routing by ingress router
  - Traffic-driven route computation
  - Centrally computes
    - Path to requested destination
    - That satisfies traffic-specific policies
  - Installs label-swap forwarding state over full path
- Virtual-circuit forwarding using labels

UC SANTA CRUZ

Problems with Previous Solutions

- Compromise robustness
  - Centralized control of distributed state
  - Paths are “brittle”
  - Magnifies affects of topology changes
- Compromise efficiency and responsiveness
  - “Duplex” control communication
  - Requires “full topology” routing protocol
  - Path setup delay
- **Need distributed, policy-based routing model**

UC SANTA CRUZ

PACeR

- "**P**olicy **A**ware **C**onnectionl**e**ss **R**outing"
- Distributed routing architecture
  - Topology-driven route computation
  - Autonomously compute
    - Best *set* of paths
    - to every destination
- Manages local label-swap forwarding state
  - "Distributed Label-Swap Forwarding"
- **Goal is default policy-based routing.**

UC SANTA CRUZ

PACeR Design

- Specify policies using
  - **Path algebra** – formalizes notion of best set of paths to a destination (QoS)
  - **Traffic algebra** – formalizes notion of resource usage policies (Traffic Engineering)
- PACeR path-selection algorithms
- Efficient forwarding over multiple-paths
  - **Distributed Label-Swap Forwarding**

UC SANTA CRUZ

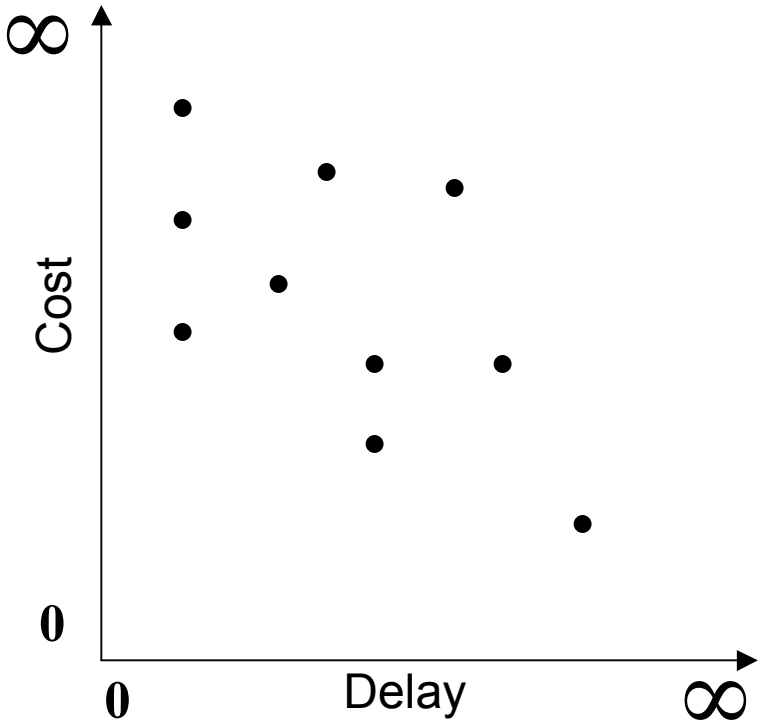
Path Algebra

- Extension of Sobrinho's path algebra
- Set of multi-component weights
  - E.g. (delay, cost)

UC SANTA CRUZ

Path Weights

- Weights can be visualized as points in a space



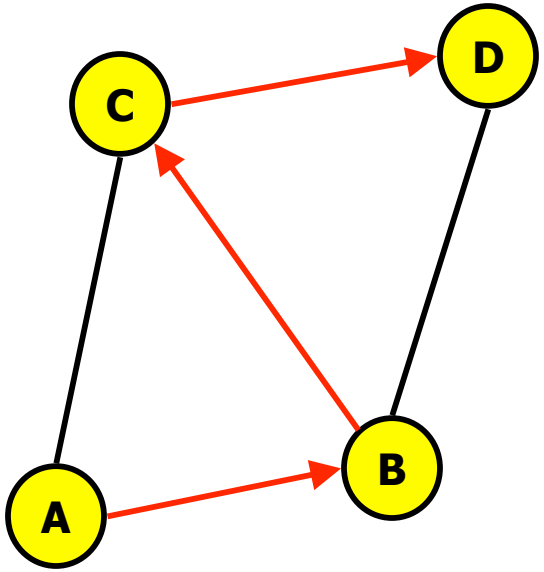
Santa Cruz Baskin School of Engineering
Engineering

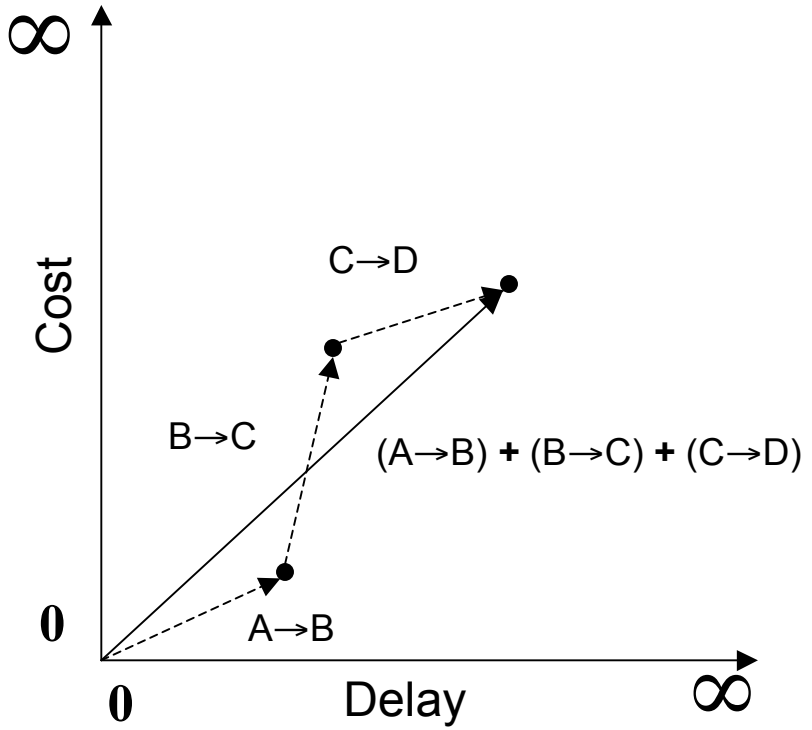
UC SANTA CRUZ

Path Algebra

- Extension of Sobrinho's path algebra
- Set of multi-component weights
  - E.g. (cost, delay)
- "+" operator defines vector algebra
  - E.g. $(d_1, c_1) + (d_2, c_2) \equiv (d_1+d_2, c_1+c_2)$

Vector Algebra of Weights





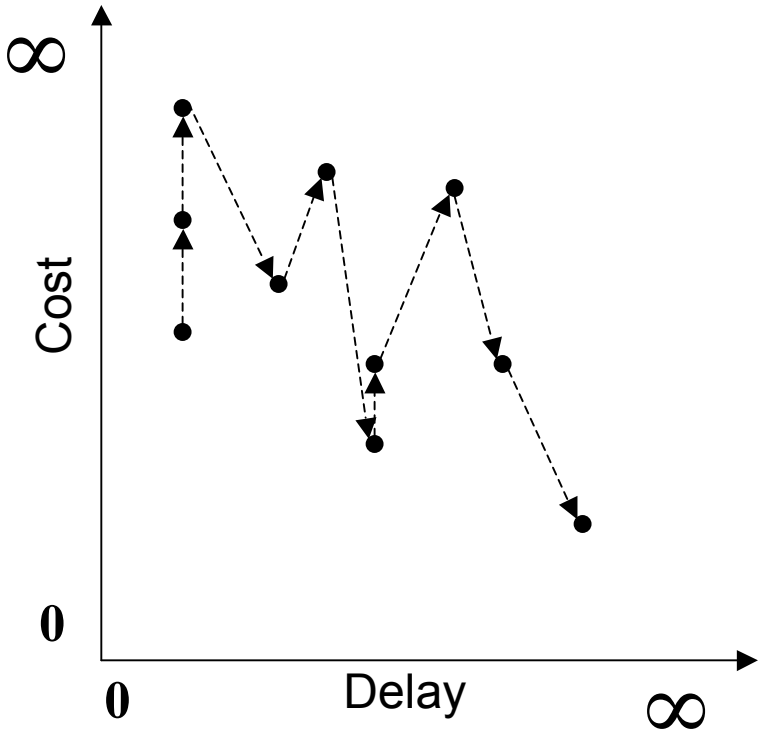
UC SANTA CRUZ

Path Algebra

- Extension of Sobrinho's path algebra
- Set of multi-component weights
  - E.g. (cost, delay)
- "+" operator defines vector algebra
  - E.g. $(d_1, c_1) + (d_2, c_2) \equiv (d_1+d_2, c_1+c_2)$
- "≤" total ordering on weights
  - E.g. $(d_i,c_i) \leq (d_k,c_k) \equiv (d_i < d_k) \vee ((d_i = d_k) \wedge (c_i \leq c_k))$

≤ Relation on Weights

- Used to search paths
- Needs to provide convenient ordering for efficiently finding best *set* of paths



UC SANTA CRUZ

Path Algebra

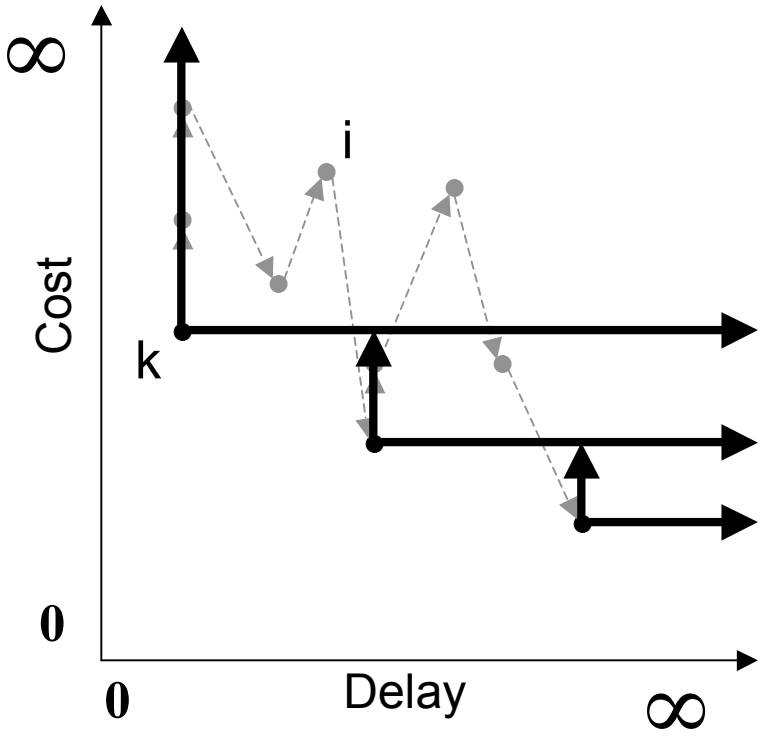
- Extension of Sobrinho's path algebra
- Set of multi-component weights
  - E.g. (cost, delay)
- "+" operator defines vector algebra
  - E.g. $(d_1, c_1) + (d_2, c_2) \equiv (d_1+d_2, c_1+c_2)$
- "≤" total ordering on weights
  - E.g. $(d_i,c_i) \leq (d_k,c_k) \equiv (d_i < d_k) \vee ((d_i = d_k) \wedge (c_i \leq c_k))$
- "$\subseteq$" partial ordering on weights where

$$(d_i,c_i) \subseteq (d_k,c_k) \rightarrow (d_i,c_i) \leq (d_k,c_k)$$

  - E.g. $(d_i,c_i) \subseteq (d_k,c_k) \equiv ((d_i \leq d_k) \wedge (c_i \leq c_k))$

⊆ Relation on Weights

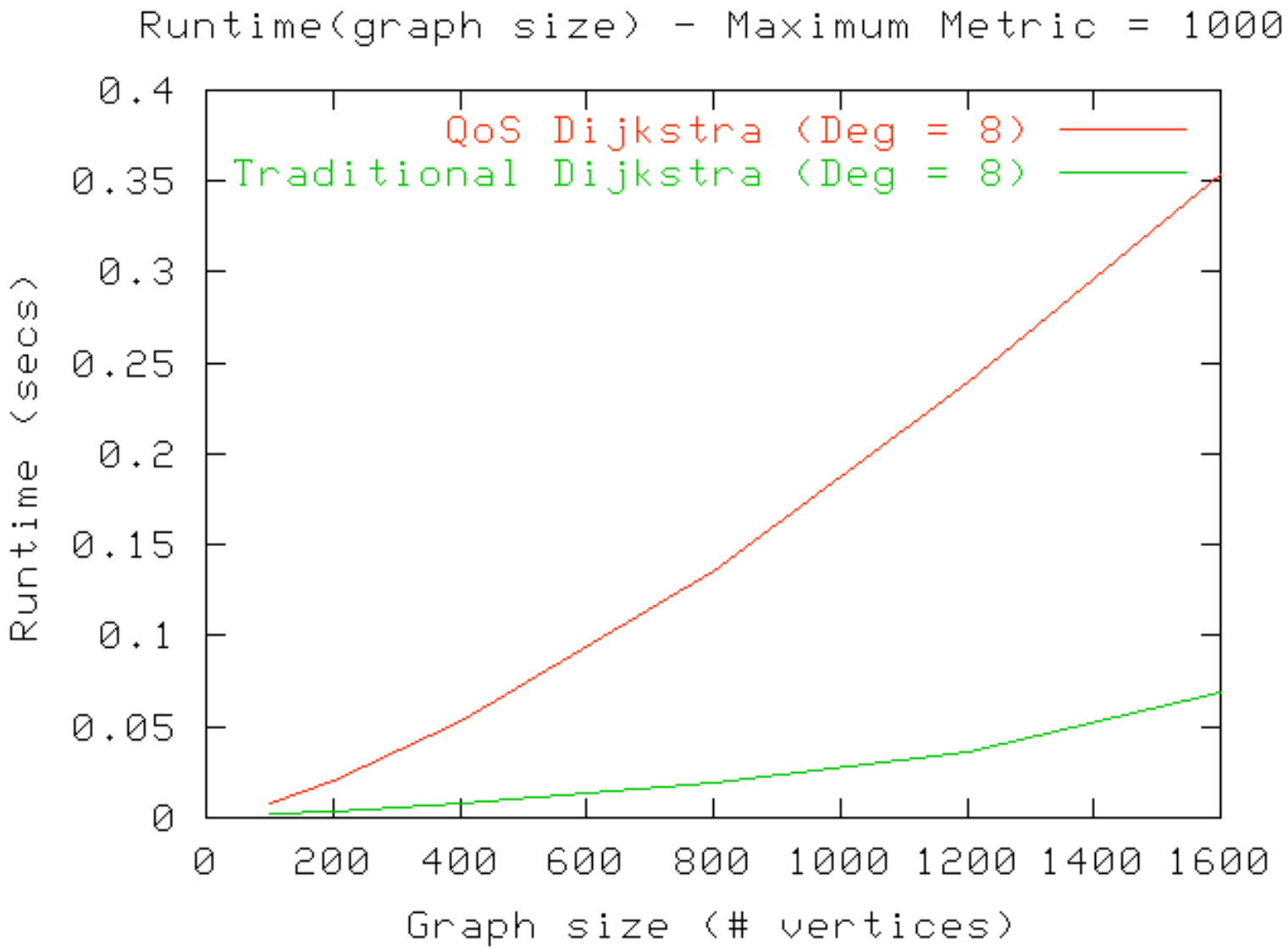
- i ⊆ k *iff* all performance requirements satisfied by i are also satisfied by k
- i ⊆ k *iff* k is down and to the left of i
- Want the smallest set of weights that provide the full range of performance available in an internet.
- This is the set of weights with no other weights down and to their left
- This is the set of all *maximal elements* of the partial order

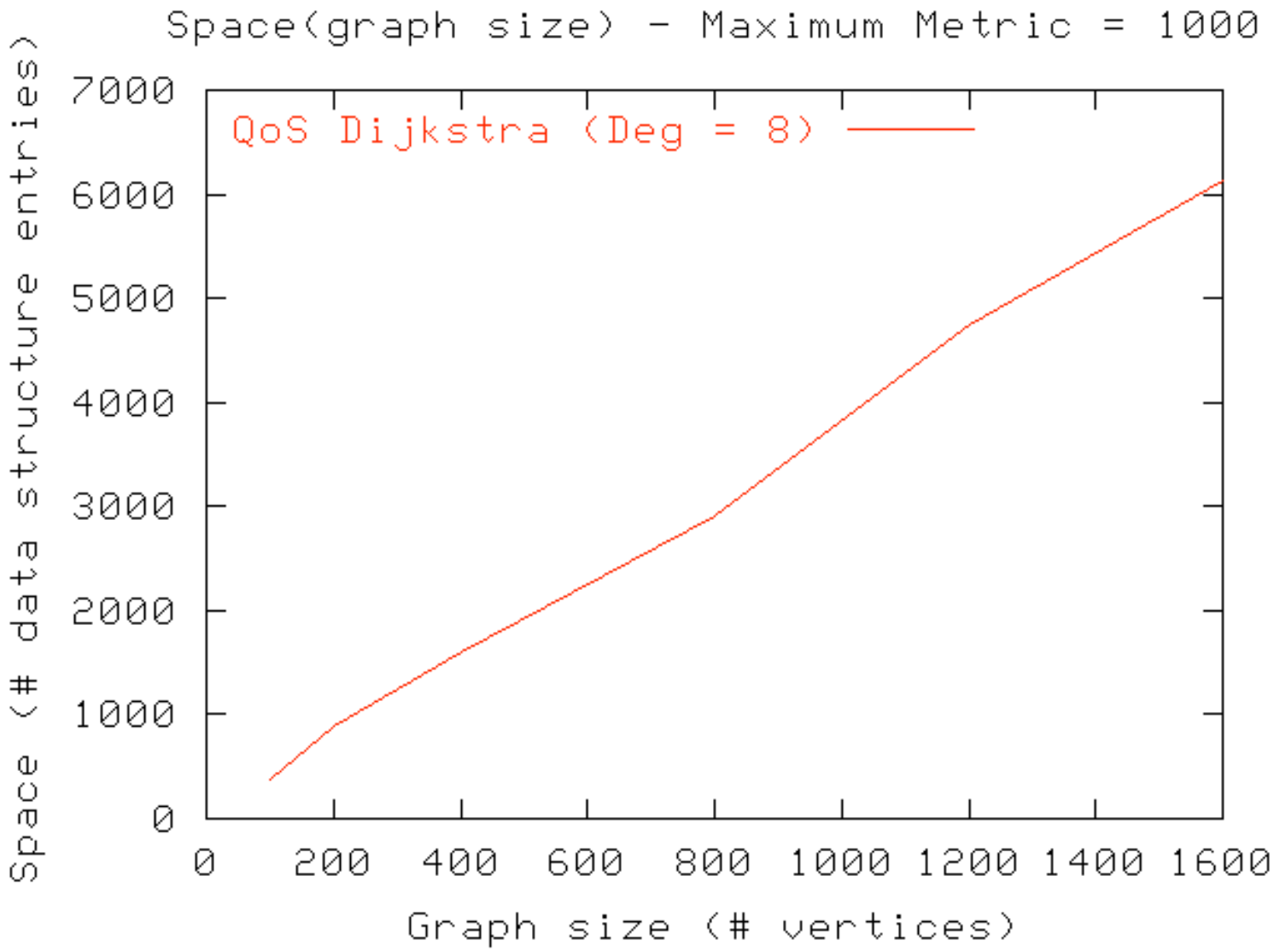


UC SANTA CRUZ

PACeR QoS Performance

- Environment
  - 1GHz Intel Pentium 3
  - C++ Standard Template Library
  - Boost Graph Library
- Each sample is worst case of 100 tests
  - 10 random weight assignments
  - 10 randomly generated graphs


Runtime(graph size) - Maximum Metric = 1000
QoS Dijkstra (Deg = 8)
Traditional Dijkstra (Deg = 8)
Runtime (secs)
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
200
400
600
800
1000
1200
1400
1600
Graph size (# vertices)


Space(graph size) - Maximum Metric = 1000
QoS Dijkstra (Deg = 8)
Space (# data structure entries)
0
1000
2000
3000
4000
5000
6000
7000
0
200
400
600
800
1000
1200
1400
1600
Graph size (# vertices)

UC SANTA CRUZ

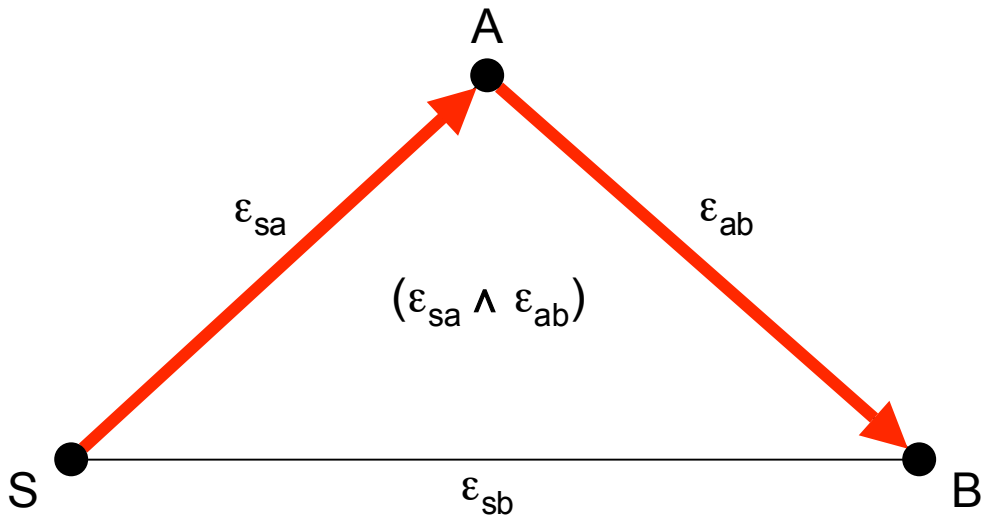
Traffic Algebra

- Boolean algebra
- Set of primitive propositions reflect characteristics of network traffic
- Truth assignment to primitive propositions defines a ***traffic class***
- Example:

```
(Gold ∧ Web) ∨
((Gold ∨ Silver) ∧ VideoOnDemand)
```

Use of Traffic Algebra

- Assign **link predicates**
- **Path predicate** is conjunction of link predicates
- Find shortest path for all possible traffic classes



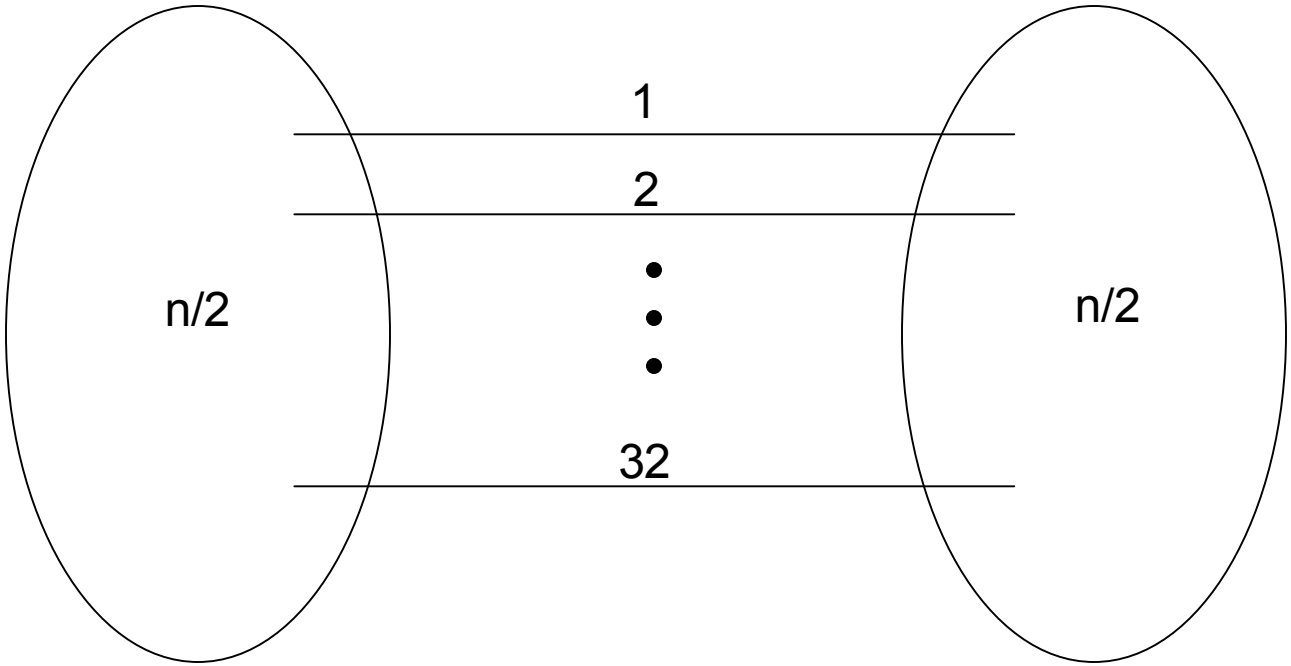
UC SANTA CRUZ

PACeR TE Performance

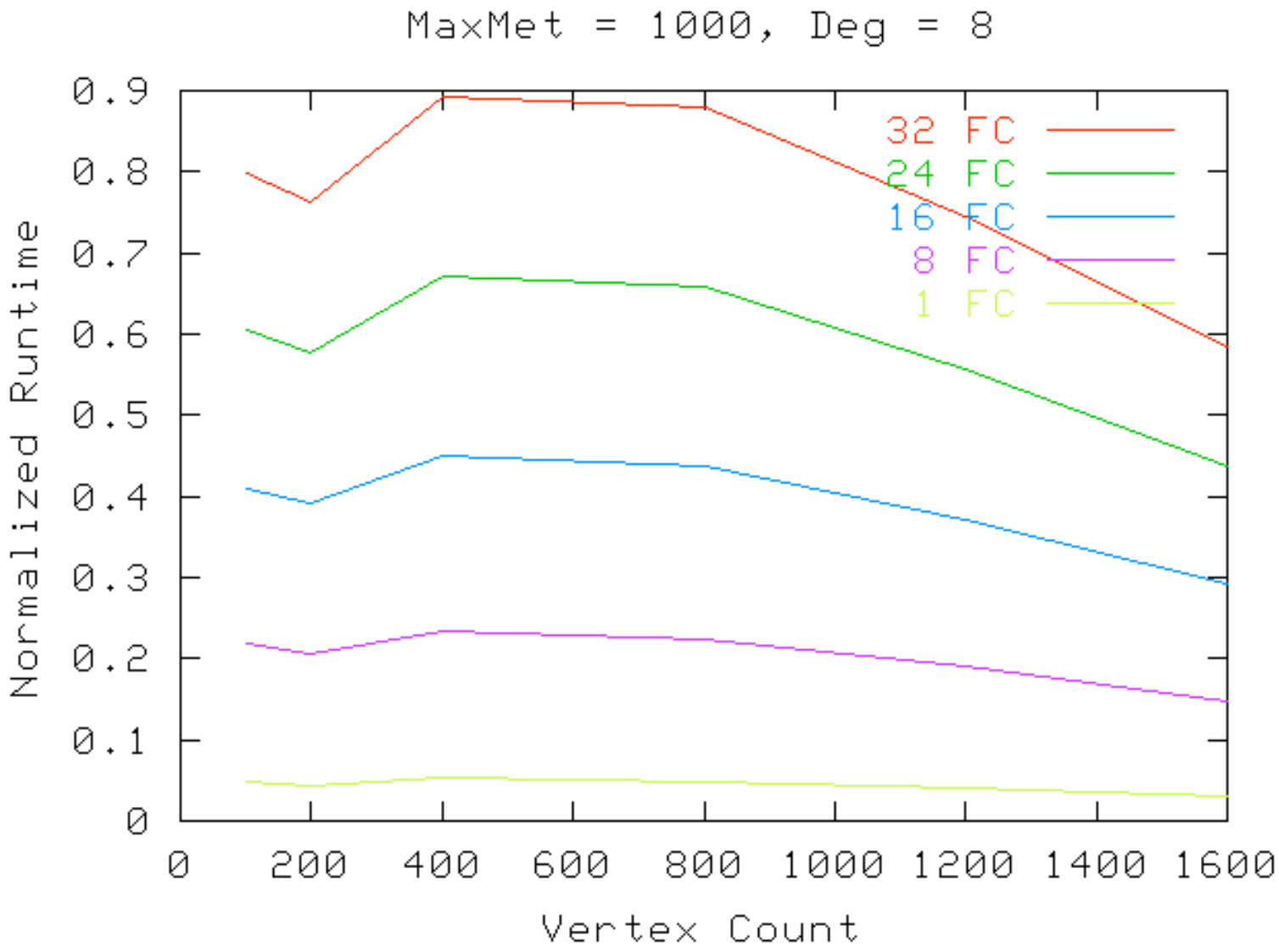
- Efficient bitmap implementation
- Each sample is worst case of 9 runs
  - 3 random weight assignments
  - 3 randomly generated graphs
  - for 1-32 forwarding classes
- Results normalized to brute-force computation (run Dijkstra 32 times)
- Control number of forwarding classes through the topology

UC SANTA CRUZ

Test Topology



Santa Cruz Baskin School of Engineering
Engineering


MaxMet = 1000, Deg = 8
Normalized Runtime
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
200
400
600
800
1000
1200
1400
1600
Vertex Count
32 FC
24 FC
16 FC
8 FC
1 FC

UC SANTA CRUZ

PACeR P.S. Algorithms

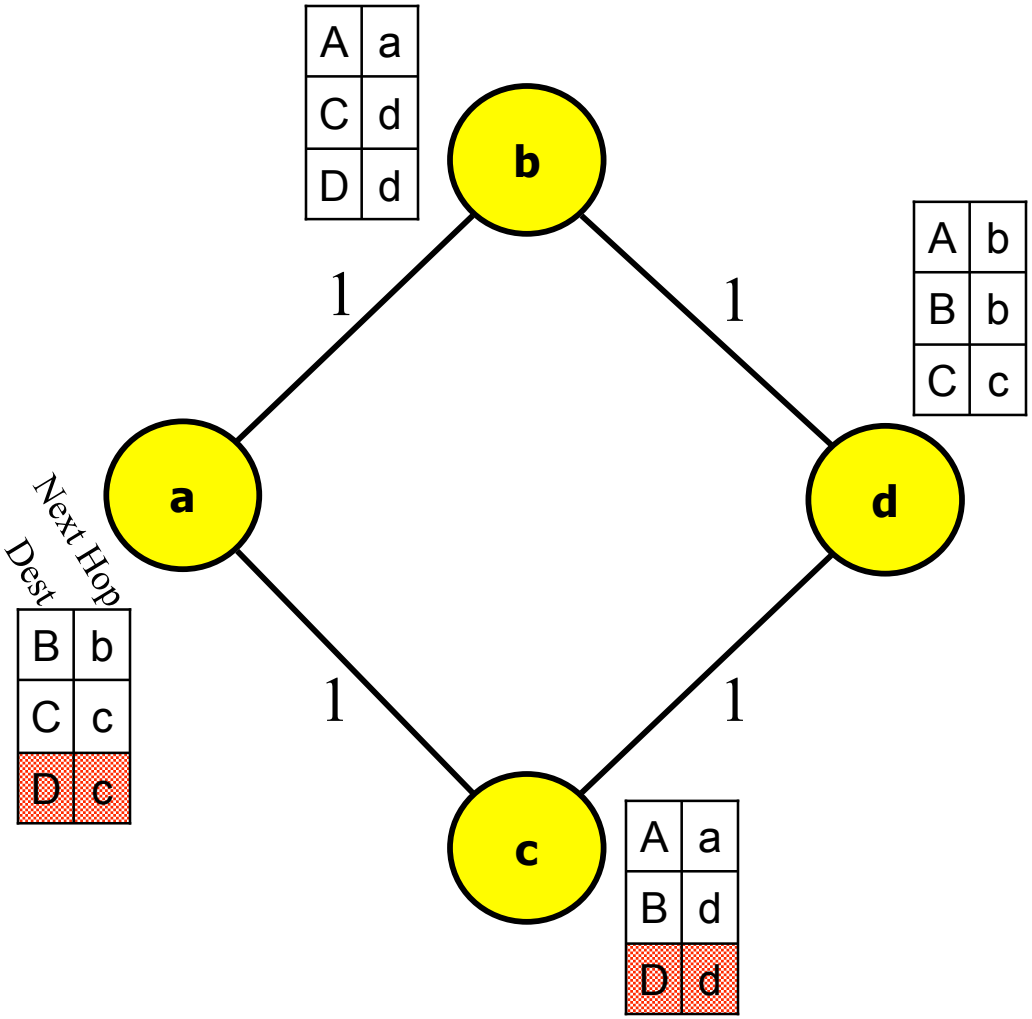
- Find best *set* of routes that
  - Provide range of performance available in internet…
  - …for each class of traffic
- Enhance Dijkstra SPF
  - **Path algebra** – formalizes notion of best set of paths to a destination
  - **Traffic algebra** – formalizes notion of resource requirements
- Efficient, *generalized* SPF algorithms
  - Outline of algorithms presented in paper

UC SANTA CRUZ

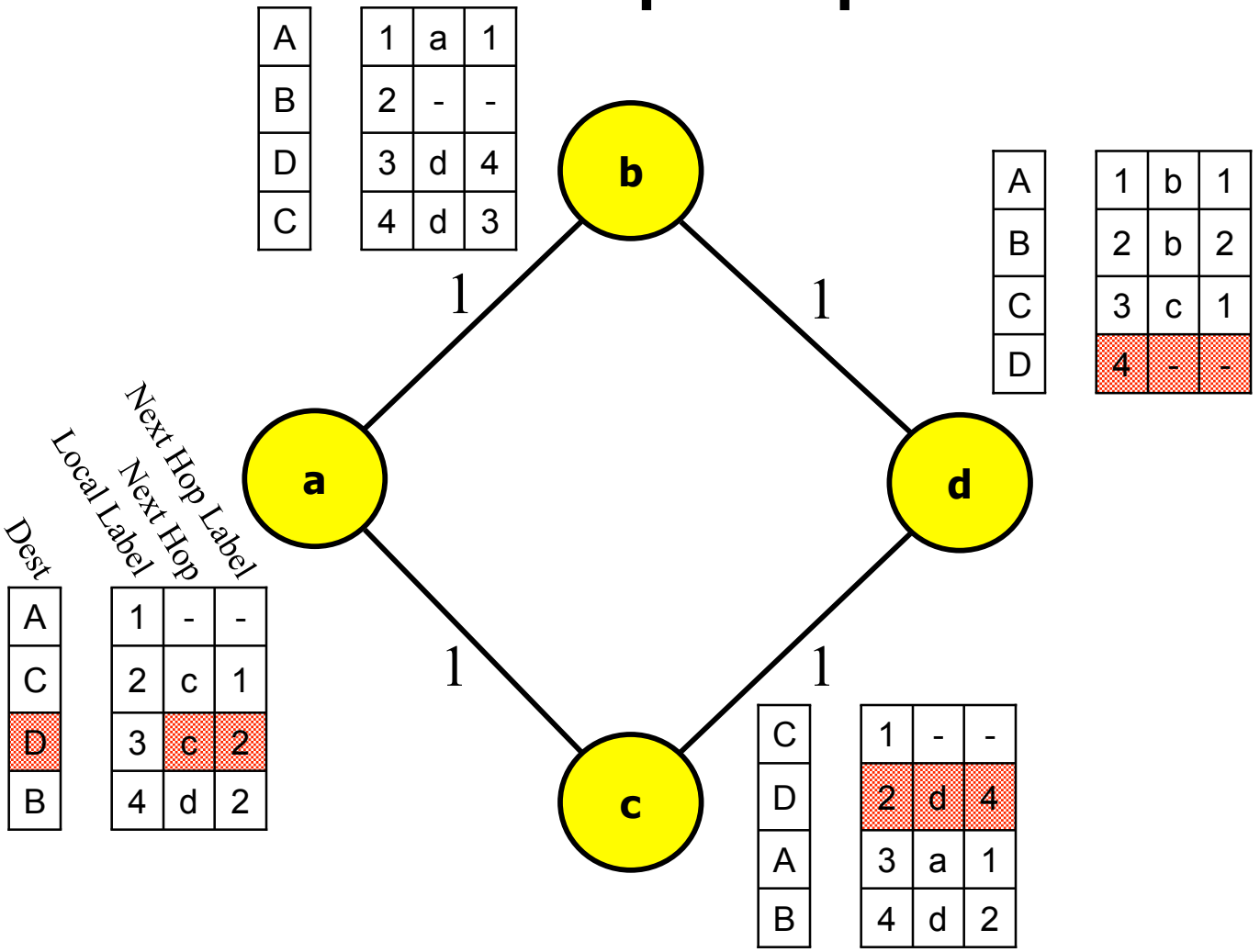
Distributed Label-Swap Forwarding

- Label-swap forwarding is generalization of address-based forwarding
- Semantically neutral mechanism
  - Policies defined at routing layer
  - Efficiently implemented in forwarding layer
  - Multiple paths per destination
- Routers autonomously compute
  - Routes for all policies supported by an internet
  - Install local label-swap state for each route

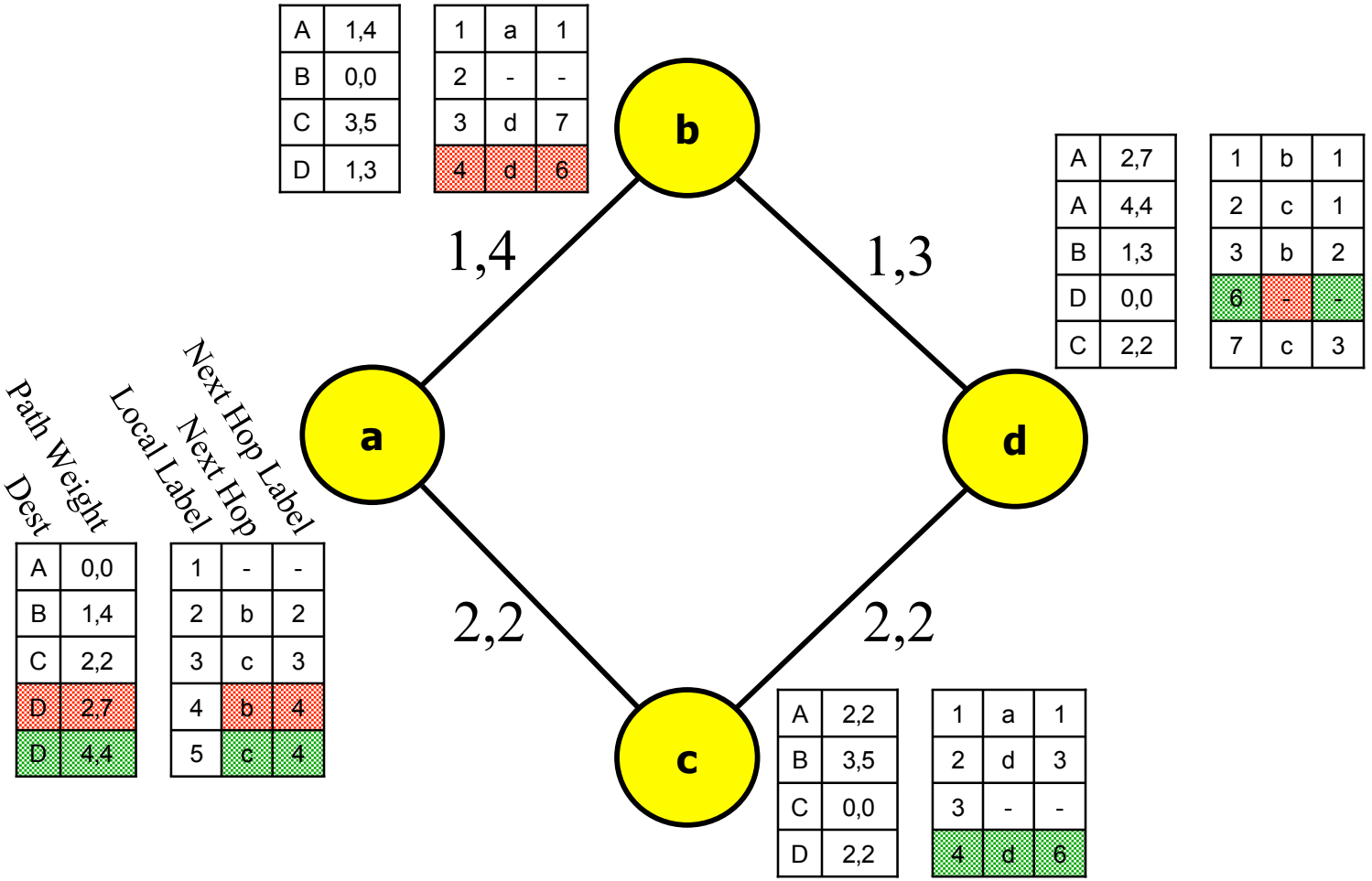
Address-Based Forwarding



Label-Swap Equivalent



Distributed QoS Forwarding



UC SANTA CRUZ

Conclusions

- Policy-based path selection
  - Is NP-complete, but…
  - is tractable for topologies supportable by routing protocols for the foreseeable future
- Label-swap forwarding
  - Generalized forwarding mechanism
  - Provides clean separation of data and signaling
  - Supports multiple paths per destination
  - Can be managed in a distributed fashion

UC SANTA CRUZ

Conclusions (cont.)

- PACeR implements
  - A distributed routing model…
  - that supports policy-based routing…
  - while retaining the robustness, efficiency, and responsiveness of the original Internet Routing Architecture