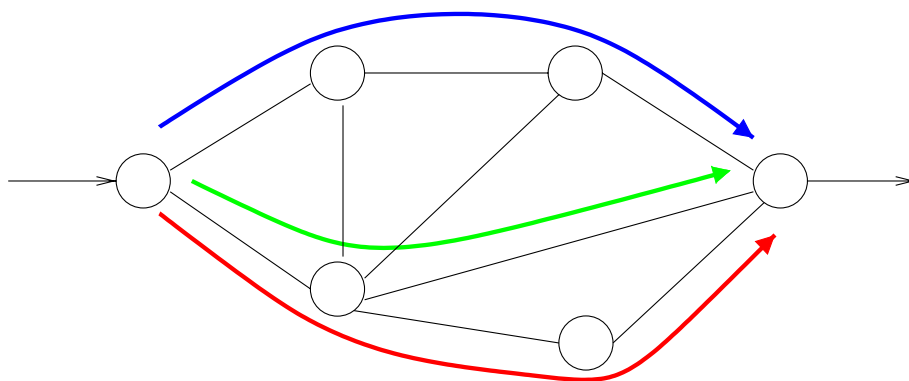


## ROUTING

Problem: Given more than one path from source to destination, which one to take?



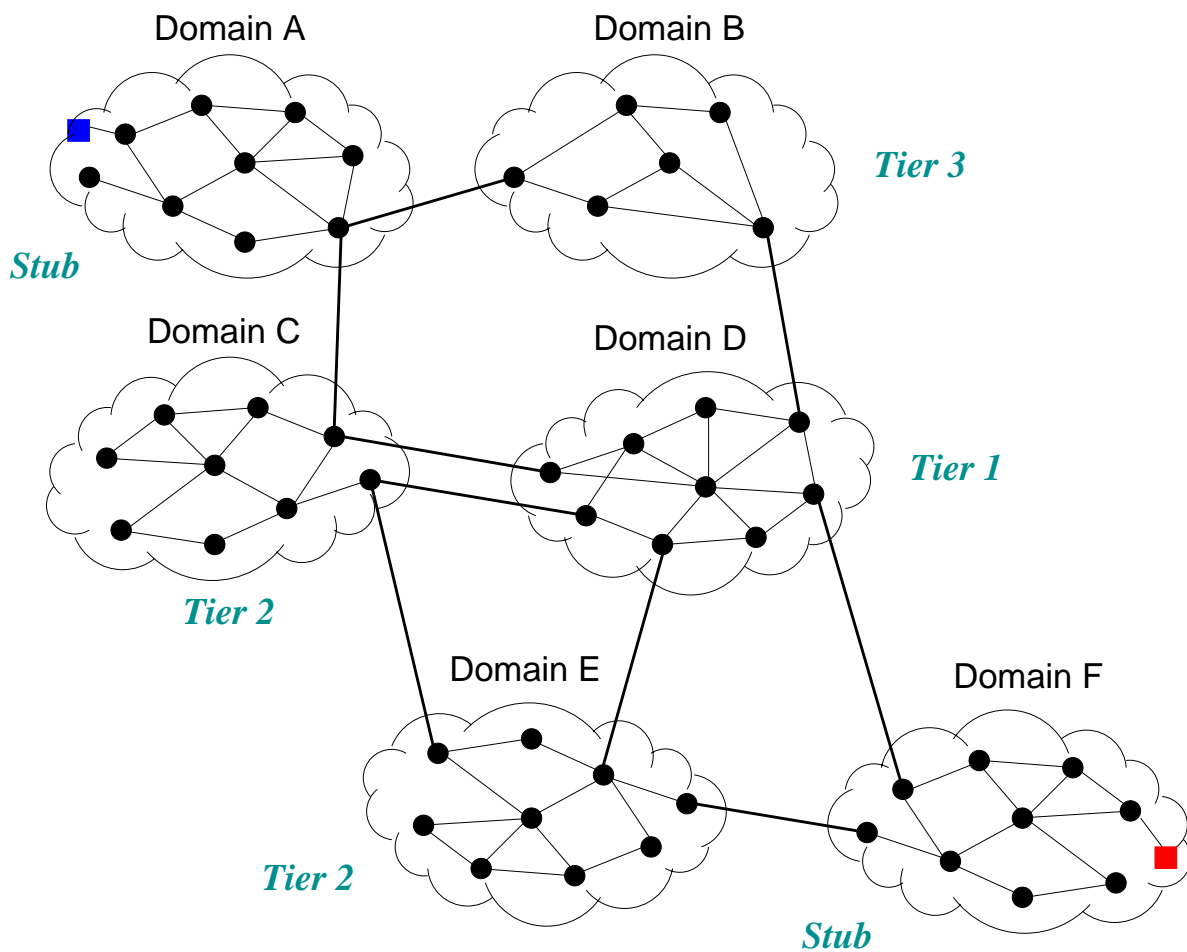
Features:

- Architecture
- Algorithms
- Implementation
- Performance

## Architecture

Hierarchical routing:

- Internet: intra-domain vs. inter-domain routing
- separate decision making



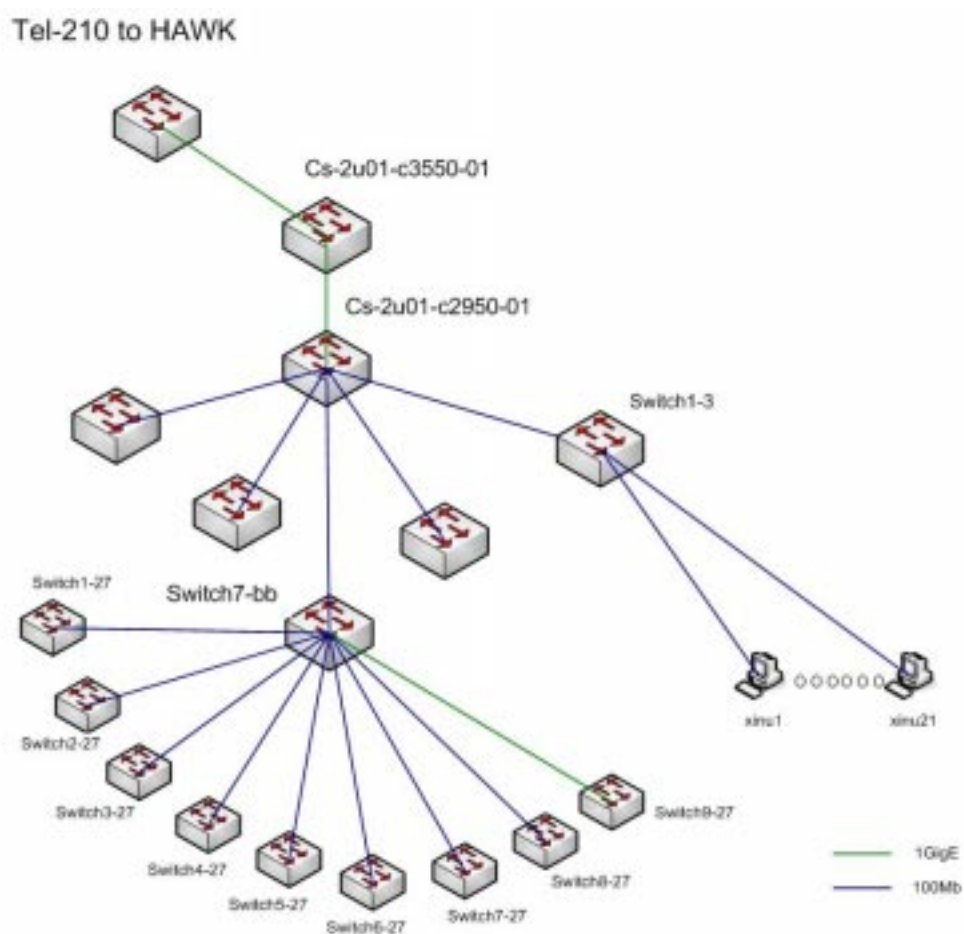
## Ex.: Purdue to east coast (BU)

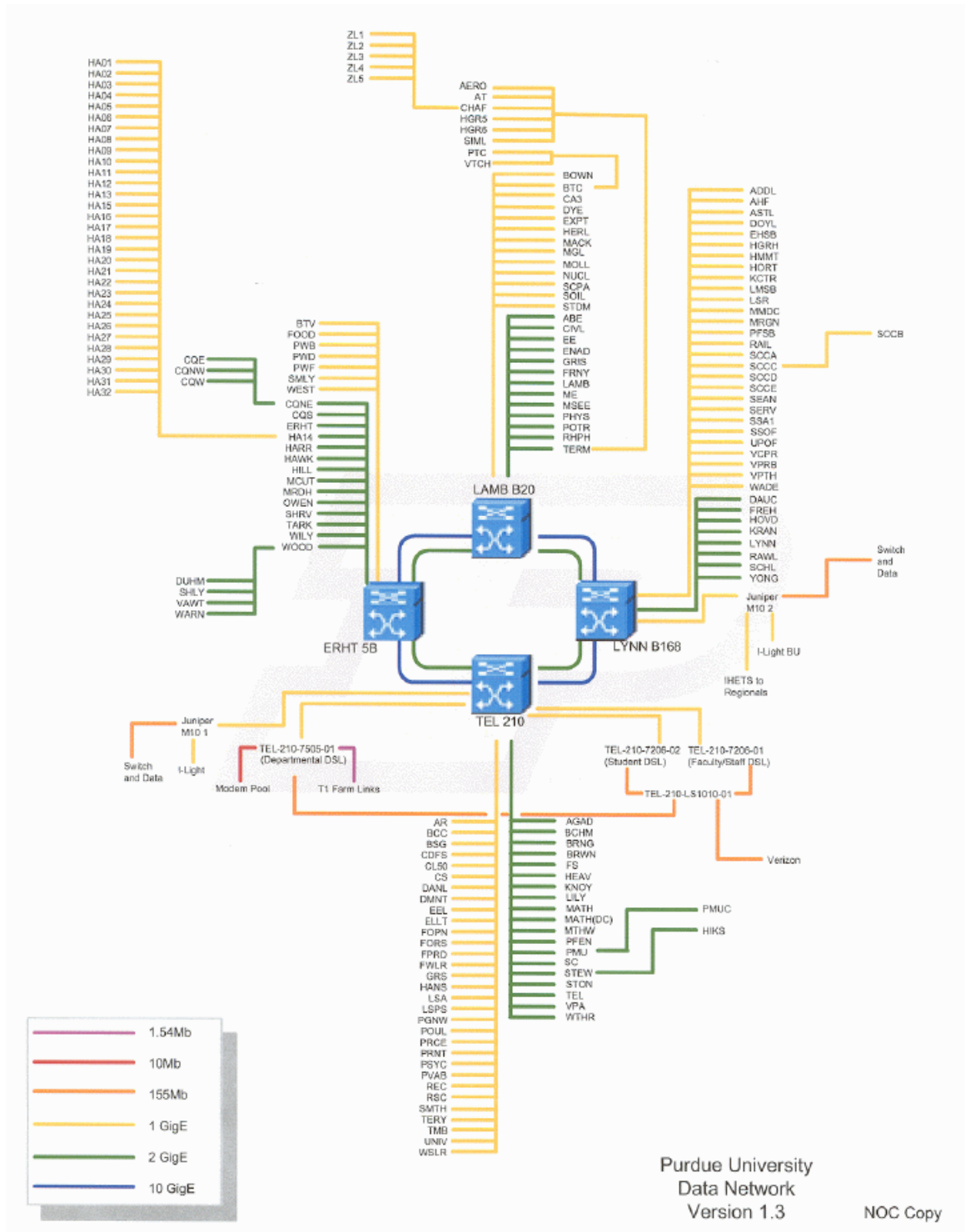
```
[109] infobahn:Routing % traceroute csa.bu.edu
traceroute to csa.bu.edu (128.197.12.3), 30 hops max, 40 byte packets
 1 cisco5 (128.10.27.250)  3.707 ms  0.616 ms  0.590 ms
 2 172.19.60.1 (172.19.60.1)  0.406 ms  0.431 ms  0.520 ms
 3 tel-210-m10-01-campus.tcom.purdue.edu (192.5.40.54)  0.491 ms  0.600 ms  0.510 ms
 4 gigapop.tcom.purdue.edu (192.5.40.134)  9.658 ms  1.966 ms  1.725 ms
 5 192.12.206.249 (192.12.206.249)  1.715 ms  3.381 ms  1.749 ms
 6 chinng-iplsng.abilene.ucaid.edu (198.32.8.76)  5.669 ms  8.319 ms  5.601 ms
 7 nycmng-chinng.abilene.ucaid.edu (198.32.8.83)  25.626 ms  25.664 ms  25.621 ms
 8 noxgs1-P0-6-0-NoX-NOX.nox.org (192.5.89.9)  30.634 ms  30.768 ms  30.722 ms
 9 192.5.89.202 (192.5.89.202)  31.128 ms  31.045 ms  31.082 ms
10 cumm111-cgw-extgw.bu.edu (128.197.254.121)  31.287 ms  31.152 ms  31.146 ms
11 cumm111-dgw-cumm111.bu.edu (128.197.254.162)  31.224 ms  31.192 ms  31.308 ms
12 csa.bu.edu (128.197.12.3)  31.529 ms  31.243 ms  31.367 ms
```

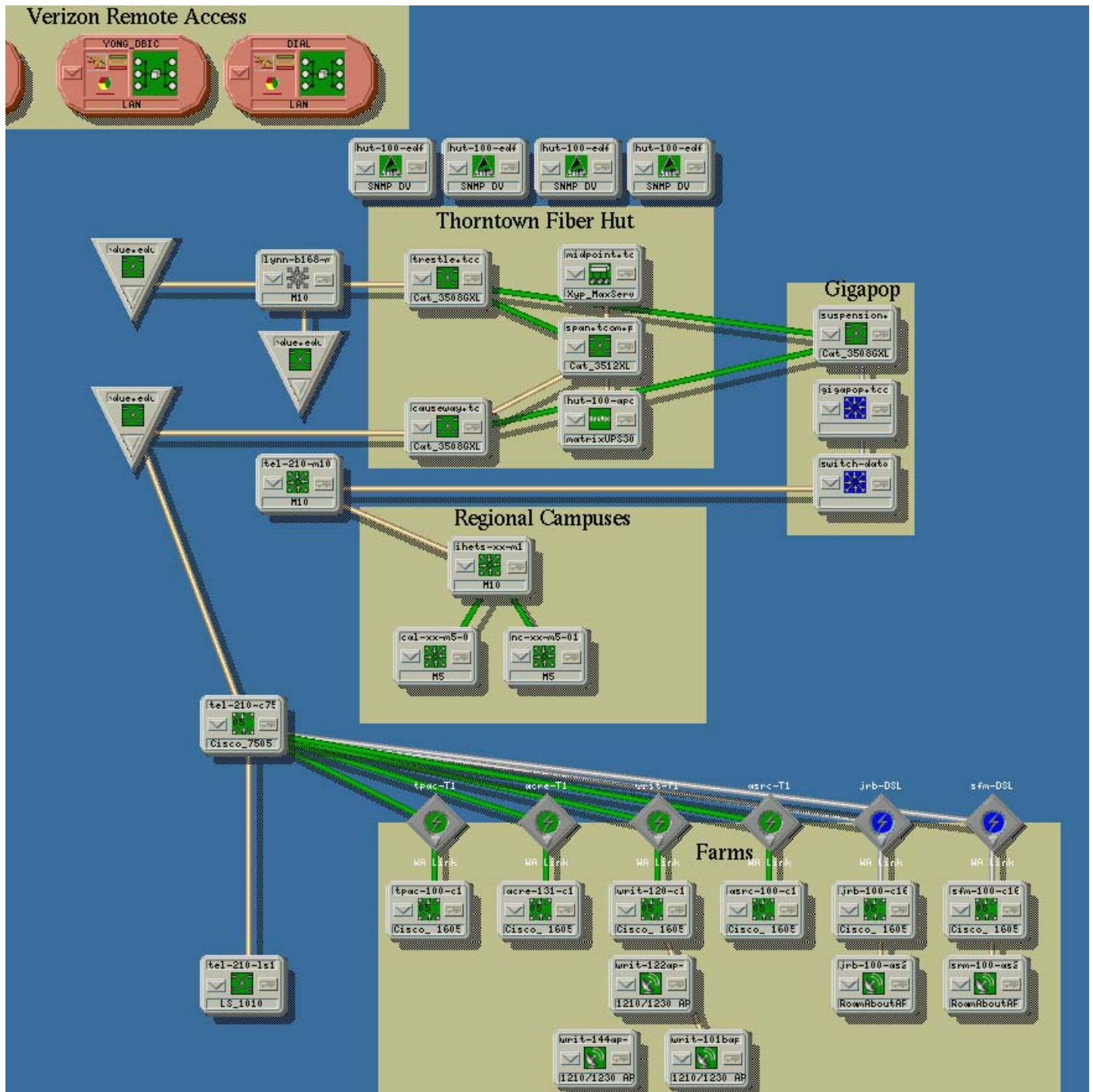
## Ex.: Purdue to west coast (Cisco)

```
[112] infobahn:Routing % traceroute www.cisco.com
traceroute to www.cisco.com (198.133.219.25), 30 hops max, 40 byte packets
 1 cisco5 (128.10.27.250)  0.865 ms  0.598 ms  1.282 ms
 2 172.19.60.1 (172.19.60.1)  0.518 ms  0.379 ms  0.405 ms
 3 tel-210-m10-01-campus.tcom.purdue.edu (192.5.40.54)  0.687 ms  0.551 ms  0.551 ms
 4 switch-data.tcom.purdue.edu (192.5.40.34)  3.496 ms  3.523 ms  2.750 ms
 5 so-2-3-0-0.gar2.Chicago1.Level3.net (67.72.124.9)  8.114 ms  20.181 ms  8.512 ms
 6 so-3-3-0.bbr1.Chicago1.Level3.net (4.68.96.41)  11.543 ms  9.079 ms  8.239 ms
 7 ae-0-0.bbr1.SanJose1.Level3.net (64.159.1.129)  62.319 ms  as-1-0.bbr2.SanJose1.Level3.net
 8 ge-11-0.ipcolo1.SanJose1.Level3.net (4.68.123.41)  68.180 ms  ge-7-1.ipcolo1.SanJose1.Level
 9 p1-0.cisco.bbnplanet.net (4.0.26.14)  75.006 ms  72.557 ms  70.377 ms
10 sjce-dmzbb-gw1.cisco.com (128.107.239.53)  66.075 ms  69.223 ms  68.350 ms
11 sjck-dmzdc-gw1.cisco.com (128.107.224.69)  65.650 ms  74.358 ms  69.952 ms
12 ^C
```

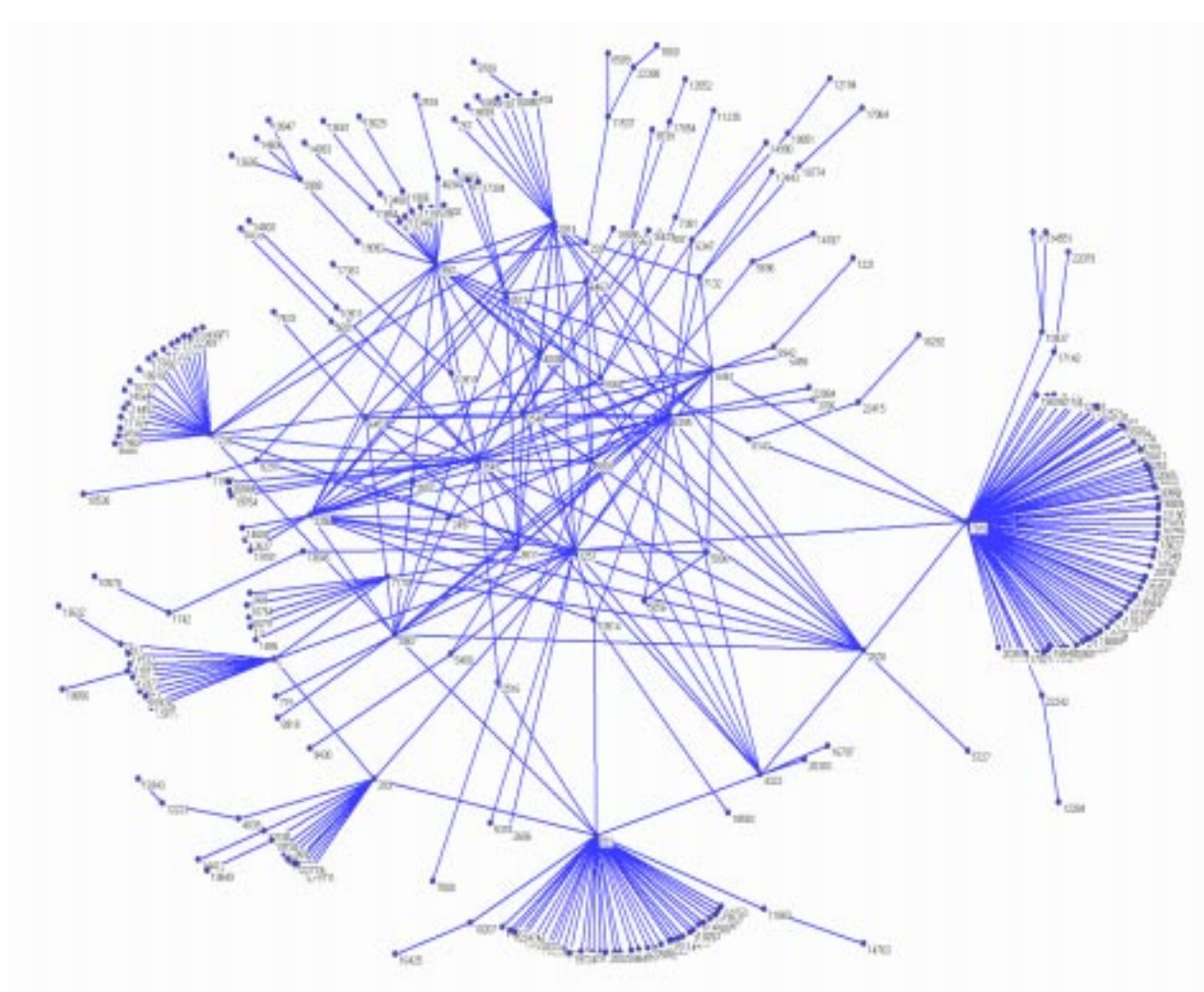
Three levels: LAN, intra-domain, and inter-domain







Inter-domain topology:



- domain called autonomous system (AS)
- 16 bit identifier

Inter-domain connectivity of Purdue:

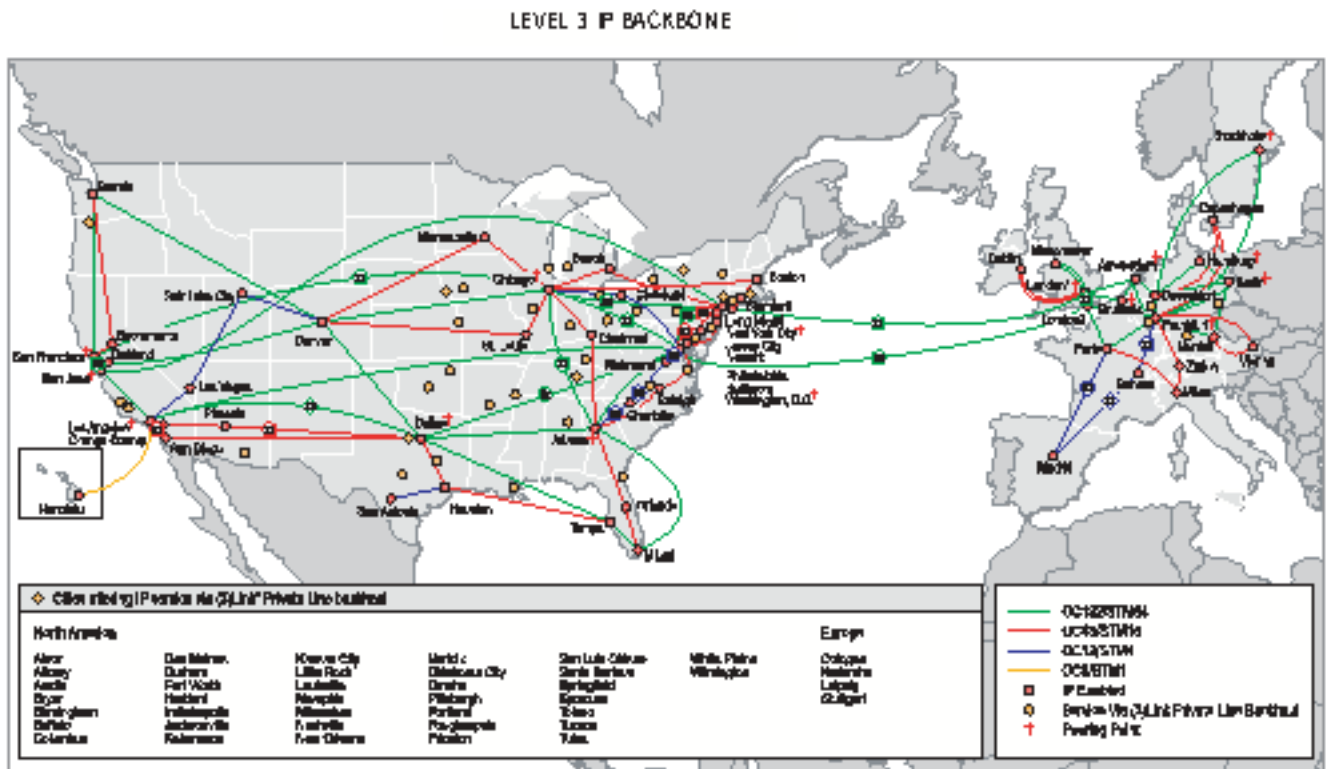
- Level3 (AS 3356) → INDIANAGIGAPOP (AS 19782)  
→ Purdue (AS 17)
- Internet2/Abilene (AS 11537) → INDIANAGIGAPOP  
(AS 19782) → Purdue (AS 17)

The Indy GigaPoP has its own AS number (19782).

- part of I-Light (Indiana state-wide project)
- located at IUPUI, connects Purdue & IU



Level3 backbone network: [www.level3.com](http://www.level3.com)



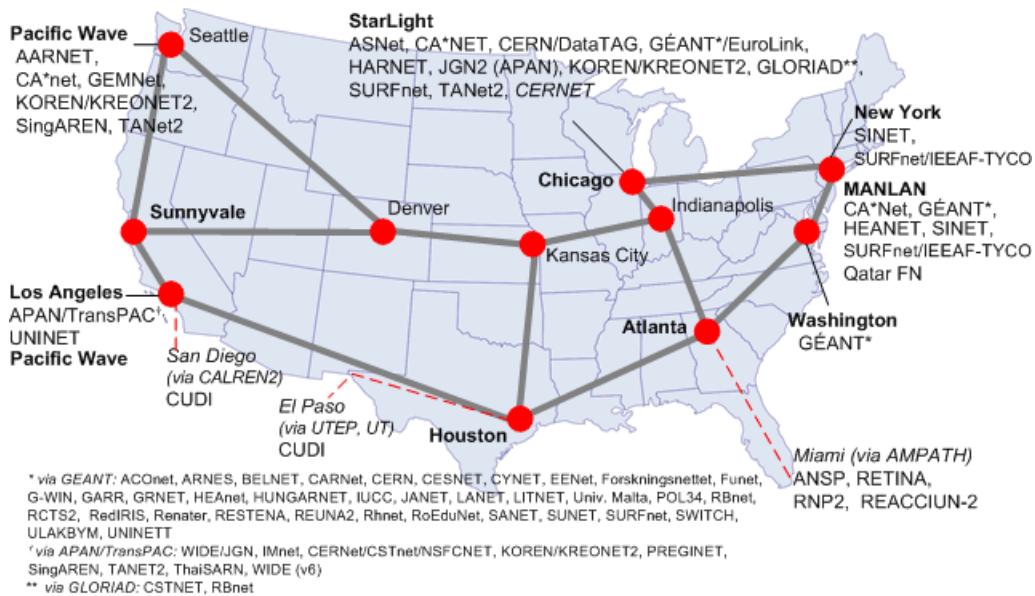
→ 10 Gbps backbone (same as Purdue)

→ part of backbone: OC-48 (2.488 Gbps)

Abilene/Internet2 backbone: [www.internet2.edu](http://www.internet2.edu)



### Abilene International Network Peers



Granularity of routing network:

- Router
- Domain: autonomous system
  - 16 bit identifier ASN
  - assigned by IANA along with IP prefix block (CIDR)

Network topology (i.e., map/connectivity):

- Router graph
  - node: router
  - edge: physical link between two routers
- AS graph
  - node: AS
  - edge: physical link between 2 or more border routers
  - sometimes at exchange point or network

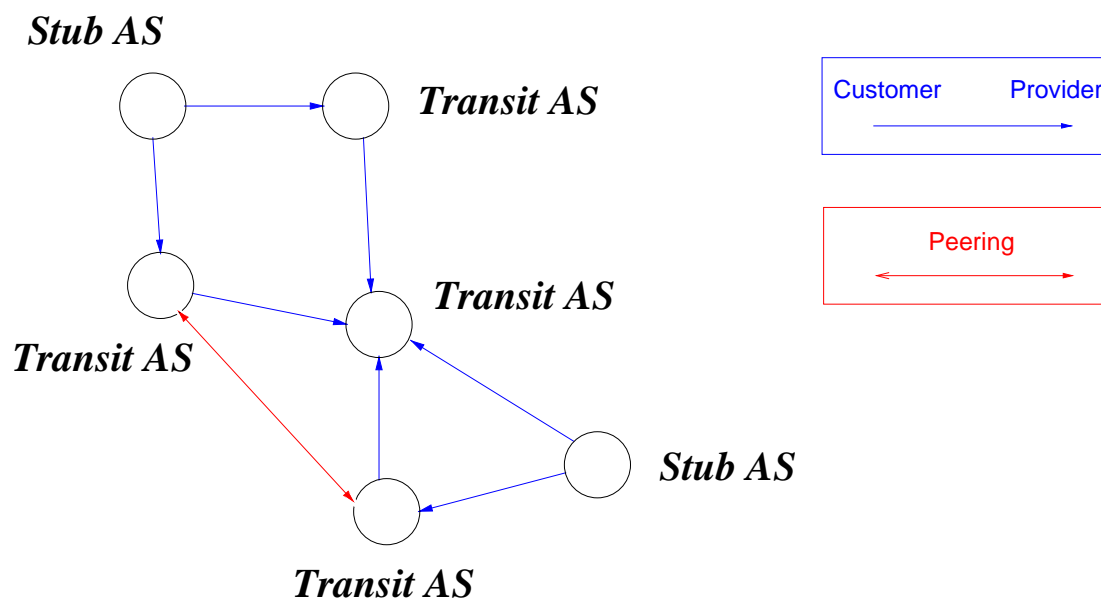
Router type:

- access router
- border router
- backbone router

AS type:

- stub AS
  - no forwarding
  - may be multi-homed (more than one provider)
- transit AS
  - tier-1: global reachability & no provider above
  - tier-2 or tier-3: providers above

AS graph:



Inter-AS relationship: bilateral

- customer-provider: customer subscribes BW from provider
  - most common
  - customer can reach provider's reachable IP space
- peering:
  - only the peer's IP address and below
  - the peer's provider's address space: invisible

Common peering:

- among tier-1 providers
  - ensures global reachability
- among tier-2 providers
  - economic factors
- among stubs
  - economic factors
  - e.g., content provider & access (“eyeball”) provider
  - e.g., Time Warner & AOL

Route or path: criteria of goodness

- Hop count
- Delay
- Bandwidth
- Loss rate

Composition of goodness metric:

→ quality of end-to-end path

- Additive: hop count, delay
- Min: bandwidth
- Multiplicative: loss rate

Goodness of routing:

→ assume  $N$  users or sessions

→ suppose path metric is delay

- System optimal routing

→ choose paths to minimize  $\sum_{i=1}^N D_i$

- User optimal routing

→ each user  $i$  chooses path to minimize  $D_i$

→ selfish actions



Pros/cons:

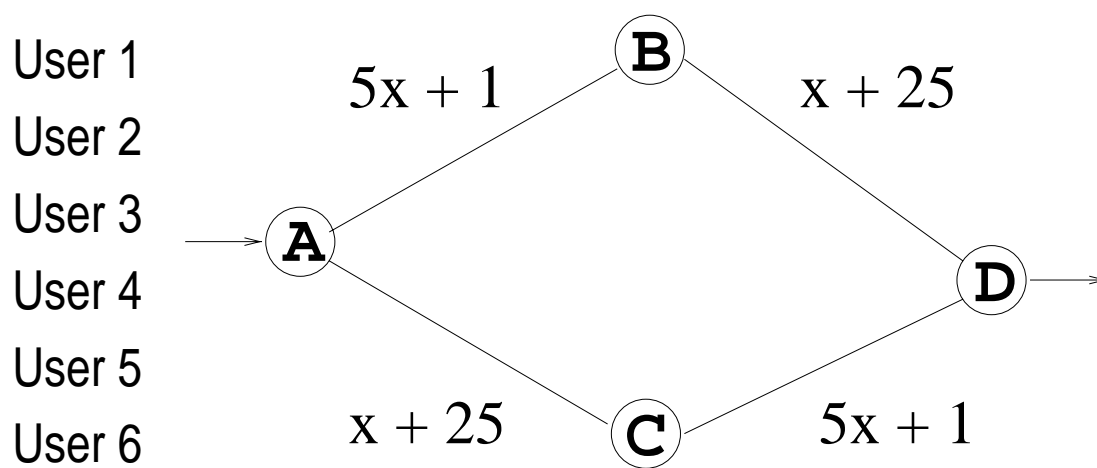
- System optimal routing:
  - Good: minimizes delay for the system as a whole
  - Bad: complex and difficult to scale up
- User optimal routing:
  - Good: simple
  - Bad: may not make efficient use of resources
    - utilization

Some pitfalls of user optimal routing:

- stemming from selfishness
- Fluttering or ping pong effect
- Braess paradox

Braess paradox example:

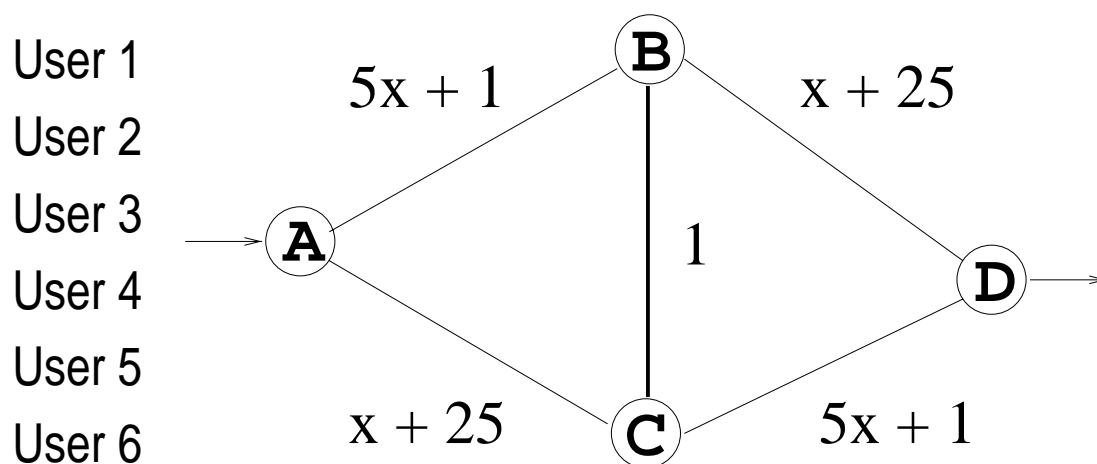
- 6 users sending 1 Mbps traffic
- Delay on shared link increases with traffic volume  $x$
- Users make routing decisions one after the other



- 3 users will take  $A \rightarrow B \rightarrow D$
- 3 users will take  $A \rightarrow C \rightarrow D$
- total delay per user:  $(5 \cdot 3 + 1) + (3 + 25) = 44$

Resource provisioning:

→ high bandwidth link is added between  $B$  and  $C$



- User 1:  $A \rightarrow B \rightarrow C \rightarrow D$  (13)
- User 2:  $A \rightarrow B \rightarrow C \rightarrow D$  (23)
- User 3:  $A \rightarrow B \rightarrow C \rightarrow D$  (33)
- User 4:  $A \rightarrow B \rightarrow C \rightarrow D$  (43)
- User 5:  $A \rightarrow B \rightarrow D$  (52)
- User 6:  $A \rightarrow C \rightarrow D$  (52)

Adding extra link should improve things, but has the opposite effect

- high-speed link induces load imbalance
- paradox possible due to selfishness
- D. Braess (1969)
- cannot arise in system optimal routing
- i.e., cooperative routing

Modus operandi of the Internet: user optimal routing

- simplicity wins the day

## Algorithms

Find short, in particular, shortest paths from source to destination.

Key observation on shortest paths:

- Assume  $p$  is a shortest path from  $S$  to  $D$

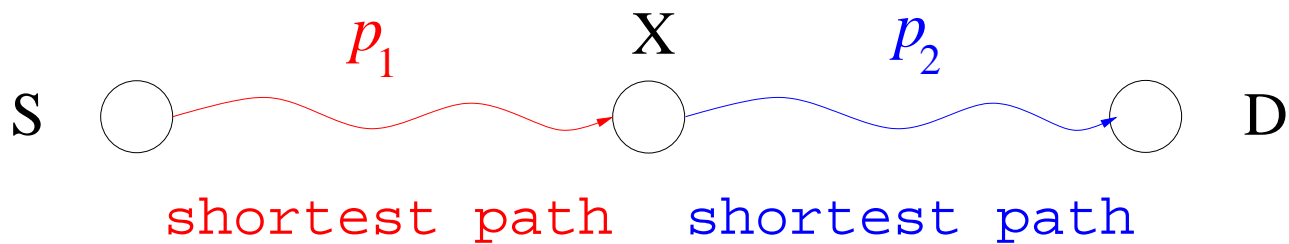
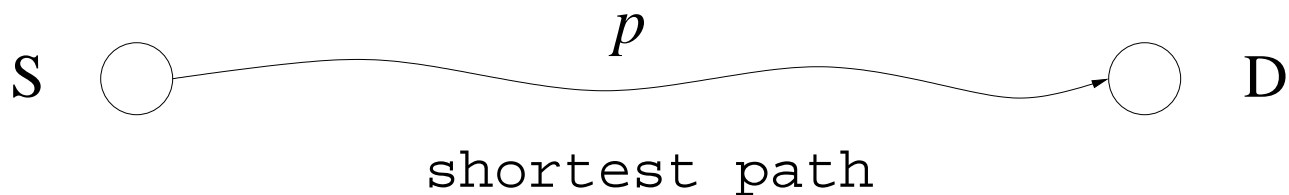
$$\rightarrow S \overset{p}{\rightsquigarrow} D$$

- Pick any intermediate node  $X$  on the path
- Consider the two segments  $p_1$  and  $p_2$

$$\rightarrow S \overset{p_1}{\rightsquigarrow} X \overset{p_2}{\rightsquigarrow} D$$

- The path  $p_1$  from  $S$  to  $X$  is a shortest path, and so is the path  $p_2$  from  $X$  to  $D$

Illustration:



→ reverse implication need not hold

→ suggests algorithm for finding shortest path

Procedure: Grow a routing tree  $\mathcal{T}$  rooted at source  $S$

→ initially  $\mathcal{T}$  only contains  $S$

1. Find a node  $X$  with shortest path from  $S$

→ there may be more than one such node

→ add  $X$  (and path  $S \overset{p}{\rightsquigarrow} X$ ) to routing tree  $\mathcal{T}$

2. Find node  $Y \notin \mathcal{T}$  with shortest path from  $S$

→ update existing paths if going through  $Y$  is shorter

→ i.e.,  $\min\{d(S, Z), d(S, Y) + \ell(Y, Z)\}$

→ need only check for  $Z \notin \mathcal{T}$

3. Repeat step two until no more nodes left to add

Observations:

→ once node is added, it's final (no backtracking)

→ builds minimum spanning tree routed at  $S$

→ Dijkstra's algorithm

Remarks:

- Running time:  $O(n^2)$  time complexity
  - $n$ : number of nodes
- If heap is used:  $O(|E| \log |V|)$ 
  - good for sparse graphs:  $|E| \ll n^2$
  - e.g., if linear:  $O(n \log n)$
- Can also be run “backwards”
  - start from destination  $D$  and go to all sources
  - a variant used in inter-domain routing
  - forward version: used in intra-domain routing
- Source  $S$  requires global link distance knowledge
  - centralized algorithm (center: source  $S$ )
  - every router runs Dijkstra with itself as source



- Internet protocol implementation
  - OSPF (Open Shortest Path First)
  - link state algorithm
  - broadcast protocol
- Minimum spanning tree rooted at  $S$ :
  - multicasting: multicast tree
  - standardized but not implemented on Internet

Distributed/decentralized shortest path algorithm:

- Bellman-Ford algorithm
- based on shortest path decomposition property

Key procedure:

- Each node  $X$  maintains current shortest distance to all other nodes
  - a distance vector
- Each node advertises to neighbors its current best distance estimates
  - i.e., neighbors exchange distance vectors
- Node  $X$ , upon receiving an update from neighbor  $Y$ , performs update: for all  $Z$

$$d(X, Z) \leftarrow \min\{ d(X, Z), d(Y, Z) + \ell(X, Y) \}$$

... same criterion as Dijkstra's algorithm

Remarks:

- Running time:  $O(n^3)$
- Each source or router only talks to neighbors
  - local interaction
  - no need to send update if no change
  - if change, entire distance vector must be sent
- Knows shortest distance, but not path
  - just the next hop is known
- Elegant but additional issues compared to Dijkstra's algorithm
  - e.g., stability
- Internet protocol implementation
  - RIP (Routing Information Protocol)

QoS routing:

Given two or more performance metrics—e.g., delay and bandwidth—find path with delay less than target delay  $D$  (e.g., 100 ms) and bandwidth greater than target bandwidth  $B$  (e.g., 1.5 Mbps)

- from shortest path to best QoS path
- multi-dimensional QoS metric
- other: jitter, hop count, etc.

How to find best QoS path that satisfies all requirements?

Brute-force

- Enumerate all possible paths
- Rank them

How many paths are there:

- If there are  $n$  nodes, there can be up to

$$\frac{n(n-1)}{2}$$

undirected links

- Hence, from source  $S$  there can be up to

$$(n-1)(n-2)\cdots 321 = (n-1)!$$

paths

- By Stirling's formula

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

→ superexponential

→ too many for brute-force

Is there a more clever or better algorithm?

- as of Apr. 21, 2005: unknown
- specifically: QoS routing is NP-complete
- strong evidence there may not exist good algorithm

In networking: several problems turn out to be NP-complete

- e.g., scheduling, control, ...
- “P = NP” problem
- one of the hardest problems in science ever

Doesn't matter too much for QoS routing

- little demand for very good algorithm
- roughly ok is fine
- intra-domain: short paths
- inter-domain: other factors (“policy”)

Policy routing:

- policy is not precisely defined
- almost anything goes

Routing criteria include

- Performance
  - e.g., short paths
- Trust
  - what in the world is “trust”?
- Economics
  - pricing
  - flexibility through multiple providers
- Politics, social issues, etc.
  - no good understanding of “policy” to date
  - anecdotal

## Implementation

Major Internet routing protocols:

- RIP (v1 and v2): intra-domain, Bellman-Ford
  - also called “distance vector”
  - metric: hop count
  - UDP
  - nearest neighbor advertisement
  - popular in small intra-domain networks
- OSPF (v1 and v2): intra-domain, Dijkstra
  - also called “link state”
  - metric: average delay
  - directly over IP: protocol number 89
  - broadcasting via flooding
  - popular in larger intra-domain networks



- IS-IS: intra-domain, Dijkstra
  - “link state”
  - directly over link layer (e.g., Ethernet)
  - more recently: also available over IP
  - flooding
  - popular in larger intra-domain networks
- Source routing: packet specifies path
  - implemented in various link layer protocols
  - ATM call set-up: circuit-switching
  - IPv4/v6: option field
  - mostly disabled
  - large ISPs: sometimes used internally for diagnosis