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

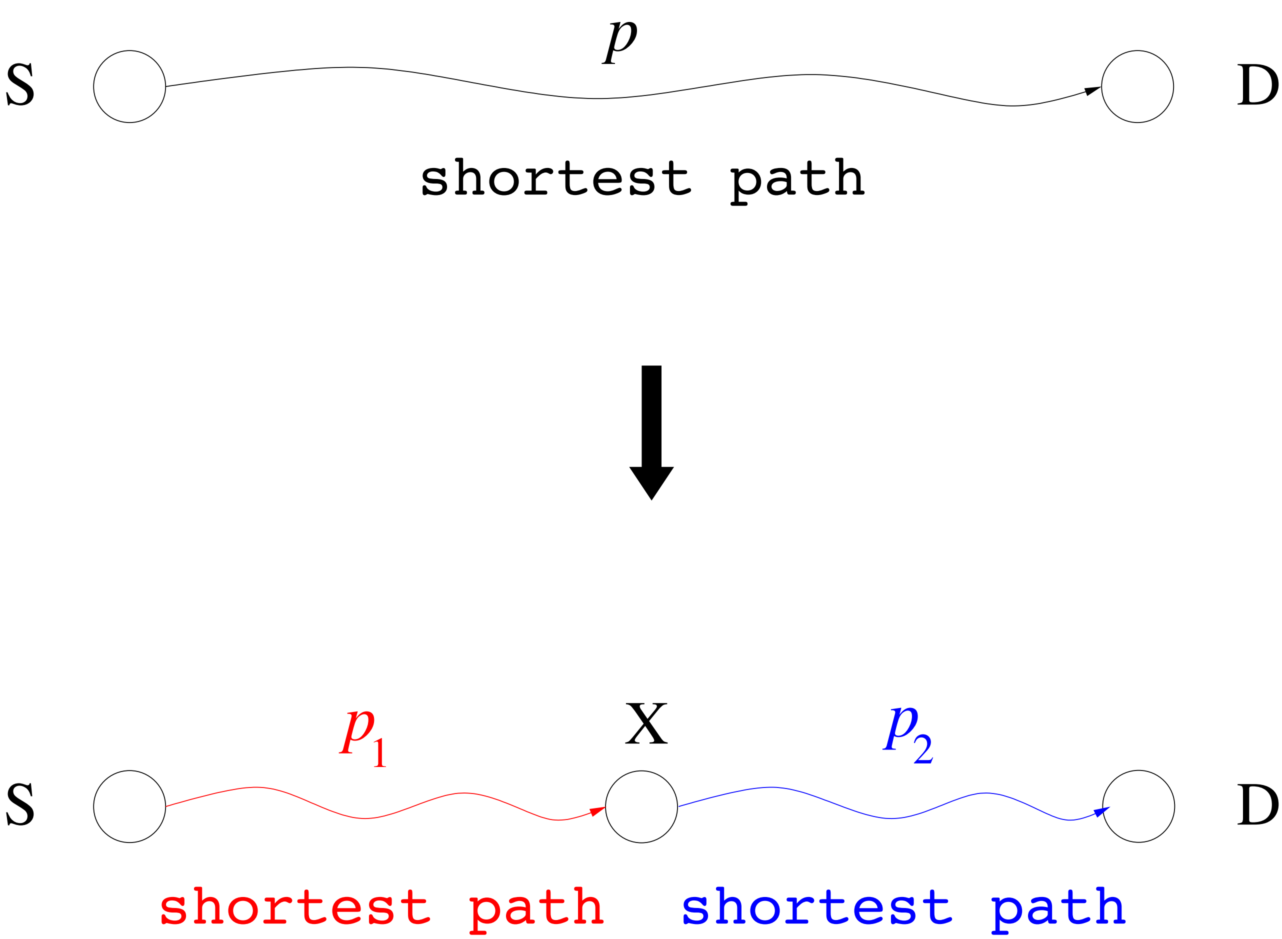
- Consider the two segments p_1 and p_2

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

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

  $\longrightarrow$ leads to Dijkstra's algorithm

Illustration:



$\rightarrow$ suggests algorithm for finding shortest path

Dijkstra: single-source all-destination

Features:

- running time: $O(n^2)$ time complexity

  $\rightarrow$ n: number of nodes

- if heap is used: $O(|E| \log |V|)$

  $\rightarrow O(n \log n)$ if $|E| = O(n)$

- can also be run "backwards"

  $\rightarrow$ start from destination D and go to all sources

  $\rightarrow$ a variant used in inter-domain routing

  $\rightarrow$ forward version: used in intra-domain routing

- source S requires global link distance knowledge

  $\rightarrow$ centralized algorithm (center: source S)

  $\rightarrow$ every router runs Dijkstra with itself as source

  $\rightarrow$ lots of broadcast management packets

- Internet protocol implementation

  → OSPF (Open Shortest Path First)

  → also called link state algorithm

  → broadcast protocol

- builds minimum spanning tree rooted at S:

  → to all destinations

  → if select destination: called multicasting

  → multicast group

  → complexity including group membership management

Distributed/decentralized shortest path algorithm:

$\longrightarrow$ Bellman-Ford algorithm

Key procedure:

- Each node X maintains current shortest distance to all other nodes

  $\rightarrow$ a distance vector

- Each node X advertises to neighbors its current best distance estimates

  $\rightarrow$ i.e., neighbors exchange distance vectors

- Each node X updates shortest paths based on neighbors' advertised information

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

  $\rightarrow$ same update criterion as Dijkstra's algorithm

Features:

- running time: $O(n^3)$, i.e., $O(n|E|)$

  $\rightarrow$ parallel/distributed: $O(|E|)$

- each source or router only talks to neighbors

  $\rightarrow$ local interaction

  $\rightarrow$ no need to send update if no change

  $\rightarrow$ if change, entire distance vector must be sent

- knows shortest distance but not path

  $\rightarrow$ just the next hop is known

- elegant but additional issues compared to Dijkstra's algorithm

  $\rightarrow$ e.g., stability

- Internet protocol implementation

  $\rightarrow$ 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., 10 Mbps)

$\longrightarrow$ from shortest path to best QoS path

$\longrightarrow$ multi-dimensional QoS metric

$\longrightarrow$ 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

$$(n-1)\,(n-2)\cdots 3\,2\,1 \;=\; (n-1)!$$

  paths

- By Stirling's formula

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

  $\rightarrow$ superexponential

  $\rightarrow$ too many for brute-force

Is there a poly-time algorithm?

$\longrightarrow$ as of April 2024: unknown

$\longrightarrow$ specifically: QoS routing is NP-hard

$\longrightarrow$ strong belief no fast algorithm

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

$\longrightarrow$ e.g., scheduling, crypto, . . .

$\longrightarrow$ "P = NP" problem

$\longrightarrow$ one of the hardest problems in science

In practice: doesn't matter too much for QoS routing

$\longrightarrow$ no pressing demand for very good algorithm

$\longrightarrow$ "roughly OK" is fine

$\longrightarrow$ intra-domain: short paths

$\longrightarrow$ inter-domain: policy routing

Policy routing:

$\longrightarrow$ meaning of "policy" is not precisely defined

$\longrightarrow$ almost anything goes

Criteria include:

- Performance

  $\rightarrow$ e.g., short paths

- Trust

  $\rightarrow$ what is "trust"?

- Economics

  $\rightarrow$ pricing

- Geo-politics, etc.

Implementation:

Internet routing protocols:

- RIP: intra-domain, Bellman-Ford

  → also called distance vector routing

  → metric: hop count

  → UDP

  → nearest neighbor advertisement

  → popular in small intra-domain networks

- OSPF: 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

  $\rightarrow$ directly over link layer (e.g., Ethernet)

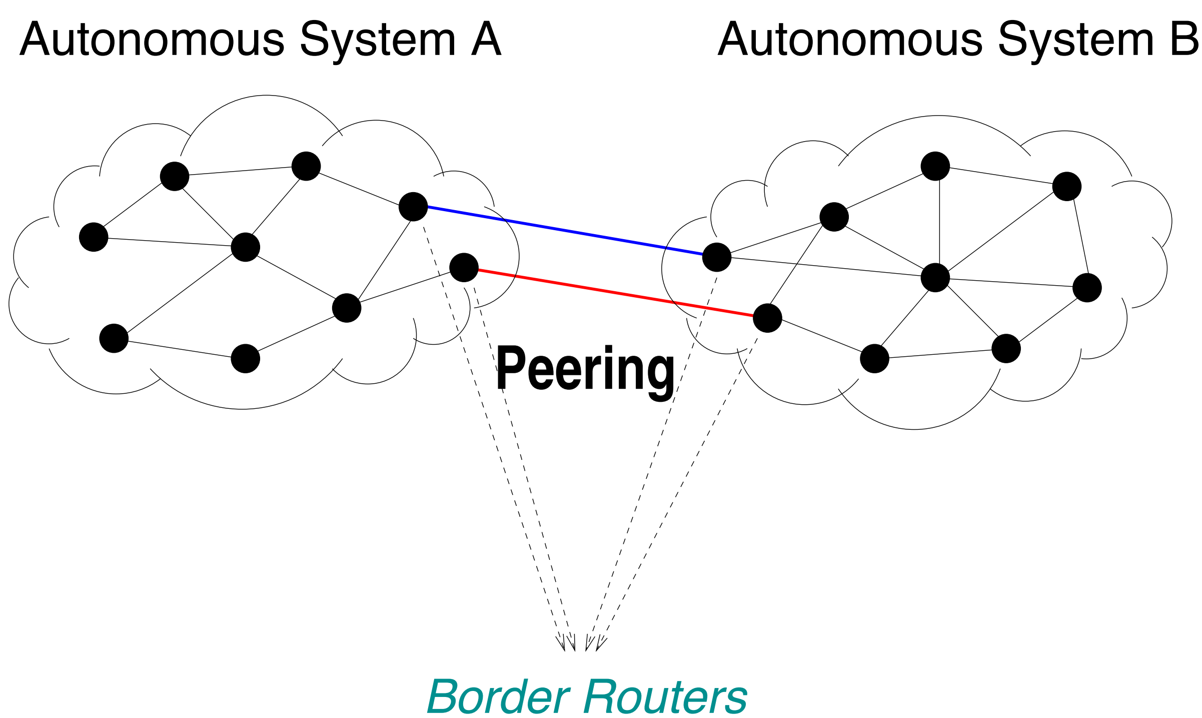
  $\rightarrow$ also available over IP (more recent)

  $\rightarrow$ flooding

  $\rightarrow$ popular in larger intra-domain networks

BGP (Border Gateway Protocol):

$\longrightarrow$ inter-domain routing



$\rightarrow$ "peering" between two domains

$\rightarrow$ typical: customer-provider relationship

$\rightarrow$ in some cases: A and B are equals (true peers)

- CIDR addressing

  → i.e., $a.b.c.d/x$

  → Purdue: 128.10.0.0/16, 128.210.0.0/16, 204.52.32.0/20

  → check at `www.iana.org` (e.g., ARIN for US)

- Metric: policy

  → e.g., shortest-path, trust, pricing

  → meaning of “shortest”: delay, router hop, AS hop

  → mechanism: path vector routing

  → BPG update message

BGP route update:

$\longrightarrow$ BGP update message propagation

BGP update message format:

$$\text{ASN}A_k \to \cdots \to \text{ASN}A_2 \to \text{ASN}A_1; \text{a.b.c.d/x}$$

Meaning: ASN A_1 (with CIDR address a.b.c.d/x) can be reached through indicated path

$\longrightarrow$ called path vector

$\longrightarrow$ also AS-PATH

Some AS numbers:

- BBN: 1
- UUNET: 701
- Level3: 3356
- Abilene (aka "Internet2"): 11537
- AT&T: 7018
- Purdue: 17

Policy:

- if multiple AS-PATHs to target AS are known, choose one based on policy

  → e.g., shortest AS path length, cheapest, least worrisome

- advertise to neighbors target AS's reachability

  → also subject to policy

  → no obligation to advertise!

  → specifics depend on bilateral contract (SLA)

SLA (service level agreement):

⟶ bandwidth (e.g., 1 Gbps)

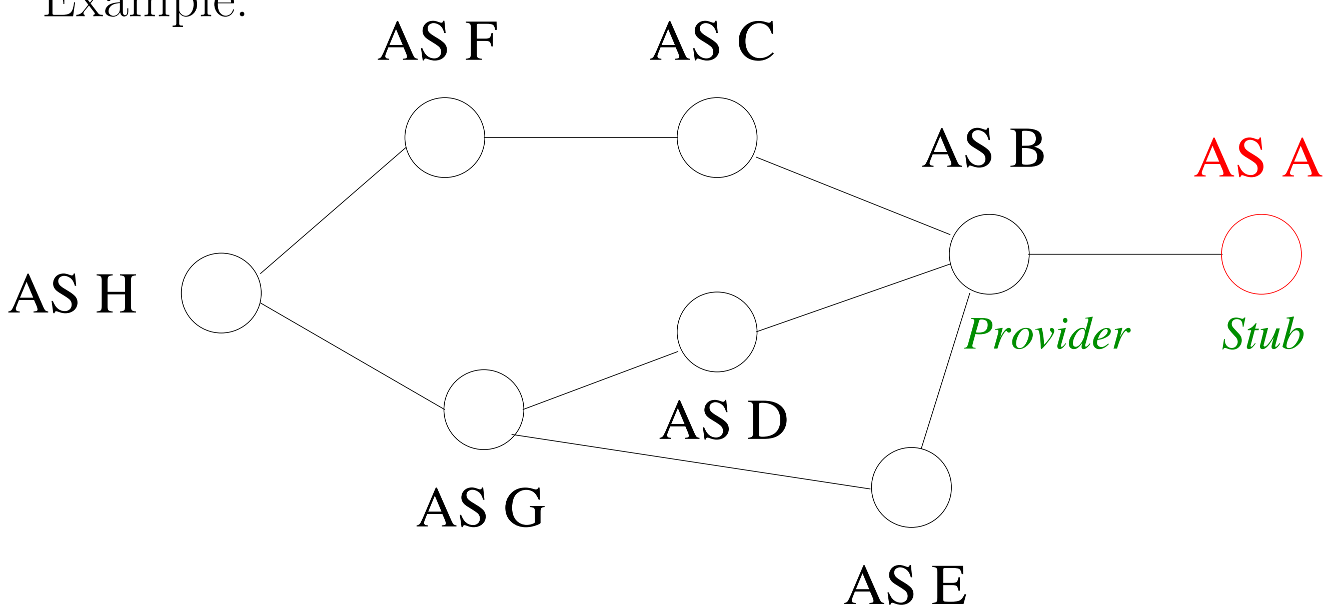
⟶ delay (e.g., avrg. 25ms US), loss (e.g., 0.05%)

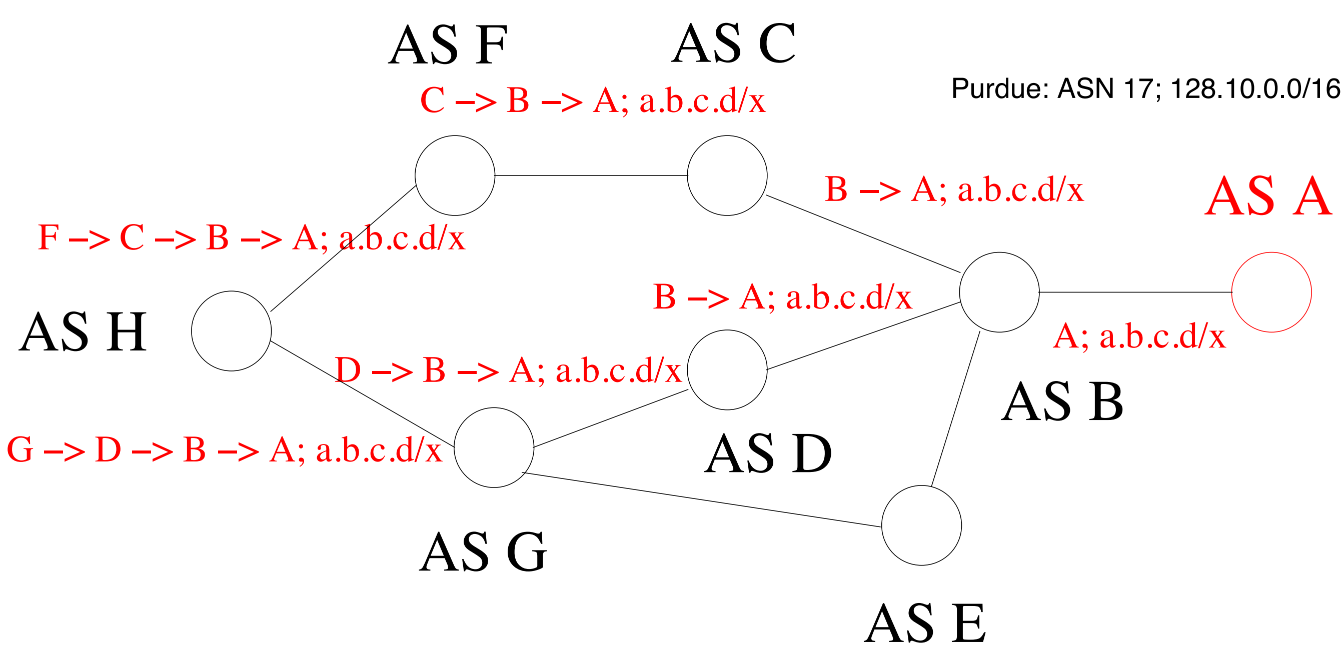
⟶ also peak vs. average

⟶ pricing (e.g., 1 Mbps: below $100)

⟶ availability, etc.

Example:


AS F
AS C
AS B
AS A
AS H
Provider
Stub
AS D
AS G
AS E


AS F
AS C
Purdue: ASN 17; 128.10.0.0/16
C –> B –> A; a.b.c.d/x
B –> A; a.b.c.d/x
AS A
F –> C –> B –> A; a.b.c.d/x
B –> A; a.b.c.d/x
AS H
A; a.b.c.d/x
D –> B –> A; a.b.c.d/x
AS B
G –> D –> B –> A; a.b.c.d/x
AS D
AS G
AS E

Performance:

Route update frequency:

$\longrightarrow$ routing table stability vs. responsiveness

$\longrightarrow$ rule: not too frequently

$\longrightarrow$ 30 seconds

$\longrightarrow$ stability wins

$\longrightarrow$ hard lesson learned from the past (sub-second)

$\longrightarrow$ legacy: TTL

Other factors for route instability:

$\longrightarrow$ selfishness (e.g., fluttering)

$\longrightarrow$ BGP's vector path routing: inherently unstable

$\longrightarrow$ more common: slow convergence

$\longrightarrow$ target of denial-of-service (DoS) attack

Route amplification:

$\longrightarrow$ shortest AS path $\neq$ shortest router path

$\longrightarrow$ e.g., may be several router hops longer

$\longrightarrow$ AS graph vs. router graph

$\longrightarrow$ policy: company in Denmark

Route asymmetry:

$\longrightarrow$ routes are not symmetric

$\longrightarrow$ estimate: $> 50\%$

$\longrightarrow$ mainly artifact of inter-domain policy routing

$\longrightarrow$ various performance implications

$\longrightarrow$ source traceback

Black holes:

$\longrightarrow$ persistent unreachable destination prefixes

$\longrightarrow$ BGP routing problems

$\longrightarrow$ further aggrevated by DNS