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 $D$

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

- Pick any intermediate node $X$ on the path

- Consider the two segments $p_1$ and $p_2$

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

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

  $\rightarrow$ leads to Dijkstra’s algorithm
Illustration:

- $S$ to $D$ via $p$
- Shortest path

$S$ to $D$ via $p_1$ and $p_2$
- Shortest path
- Shortest path

→ 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:

\[\rightarrow \text{ Bellman-Ford algorithm}\]

Key procedure:

- Each node \(X\) maintains current shortest distance to all other nodes
  \[\rightarrow \text{ a distance vector}\]
- Each node \(X\) advertises to neighbors its current best distance estimates
  \[\rightarrow \text{i.e., neighbors exchange distance vectors}\]
- Each node \(X\) updates shortest paths based on neighbors’ advertised information
  \[d(X, Z) \leftarrow \min\{ d(X, Z), d(Y, Z) + \ell(X, Y) \}\]
  \[\rightarrow \text{same update criterion as Dijkstra’s algorithm}\]
Features:

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

- 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., 10 Mbps)

\[ \rightarrow \text{ from shortest path to best QoS path} \]

\[ \rightarrow \text{ multi-dimensional QoS metric} \]

\[ \rightarrow \text{ other: jitter, hop count, etc.} \]

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
  \[
  (n - 1)(n - 2) \cdots 3 \cdot 2 \cdot 1 = (n - 1)!
  \]
  paths

- By Stirling’s formula
  \[
  n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n
  \]
  \[
  \rightarrow \text{superexponential}
  \]
  \[
  \rightarrow \text{too many for brute-force}
  \]
Is there a poly-time algorithm?

→ as of April 2024: unknown

→ specifically: QoS routing is NP-hard

→ strong belief no fast algorithm

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

→ e.g., scheduling, crypto, …

→ “P = NP” problem

→ one of the hardest problems in science

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

→ no pressing demand for very good algorithm

→ “roughly OK” is fine

→ intra-domain: short paths

→ inter-domain: policy routing
Policy routing:

→ meaning of “policy” is not precisely defined

→ almost anything goes

Criteria include:

• Performance
  → e.g., short paths

• Trust
  → what is “trust”? 

• Economics
  → 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
  → directly over link layer (e.g., Ethernet)
  → also available over IP (more recent)
  → flooding
  → popular in larger intra-domain networks
BGP (Border Gateway Protocol):

→ inter-domain routing

→ “peering” between two domains

→ typical: customer-provider relationship

→ in some cases: $A$ and $B$ are equals (true peers)
- CIDR addressing
  \[ \rightarrow \text{i.e., } a.b.c.d/x \]
  \[ \rightarrow \text{Purdue: 128.10.0.0}/16, 128.210.0.0}/16, 204.52.32.0}/20 \]
  \[ \rightarrow \text{check at www.iana.org (e.g., ARIN for US)} \]

- Metric: policy
  \[ \rightarrow \text{e.g., shortest-path, trust, pricing} \]
  \[ \rightarrow \text{meaning of “shortest”: delay, router hop, AS hop} \]
  \[ \rightarrow \text{mechanism: path vector routing} \]
  \[ \rightarrow \text{BPG update message} \]
BGP route update:

\[ \rightarrow \text{BGP update message propagation} \]

BGP update message format:

\[ \text{ASN}_k \rightarrow \cdots \rightarrow \text{ASN}_2 \rightarrow \text{ASN}_1; a.b.c.d/x \]

Meaning: ASN \( \text{A}_1 \) (with CIDR address a.b.c.d/x) can be reached through indicated path

\[ \rightarrow \text{called path vector} \]

\[ \rightarrow \text{also AS-PATH} \]

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:

```

Example:

- AS F
- AS C
- AS B
- AS A
- AS H
- AS D
- AS G
- AS E

Purdue: ASN 17; 128.10.0.0/16

```

---

```

Purdue: ASN 17; 128.10.0.0/16

```

---

```

Example:

- AS F
- AS C
- AS B
- AS A
- AS H
- AS D
- AS G
- AS E

Purdue: ASN 17; 128.10.0.0/16

```

---

```

Purdue: ASN 17; 128.10.0.0/16

```

---

```

Example:

- AS F
- AS C
- AS B
- AS A
- AS H
- AS D
- AS G
- AS E

Purdue: ASN 17; 128.10.0.0/16

```

---

```

Purdue: ASN 17; 128.10.0.0/16

```

---
Performance:

Route update frequency:

→ routing table stability vs. responsiveness

→ rule: not too frequently

→ 30 seconds

→ stability wins

→ hard lesson learned from the past (sub-second)

→ legacy: TTL

Other factors for route instability:

→ selfishness (e.g., fluttering)

→ BGP’s vector path routing: inherently unstable

→ more common: slow convergence

→ target of denial-of-service (DoS) attack
Route amplification:

\[\rightarrow\] shortest AS path $\neq$ shortest router path

\[\rightarrow\] e.g., may be several router hops longer

\[\rightarrow\] AS graph vs. router graph

\[\rightarrow\] policy: company in Denmark

Route asymmetry:

\[\rightarrow\] routes are not symmetric

\[\rightarrow\] estimate: $>50\%$

\[\rightarrow\] mainly artifact of inter-domain policy routing

\[\rightarrow\] various performance implications

\[\rightarrow\] source traceback
Black holes:

- $\rightarrow$ persistent unreachable destination prefixes
- $\rightarrow$ BGP routing problems
- $\rightarrow$ further aggrevated by DNS