**Network Layer: Control Plane (Routing)**

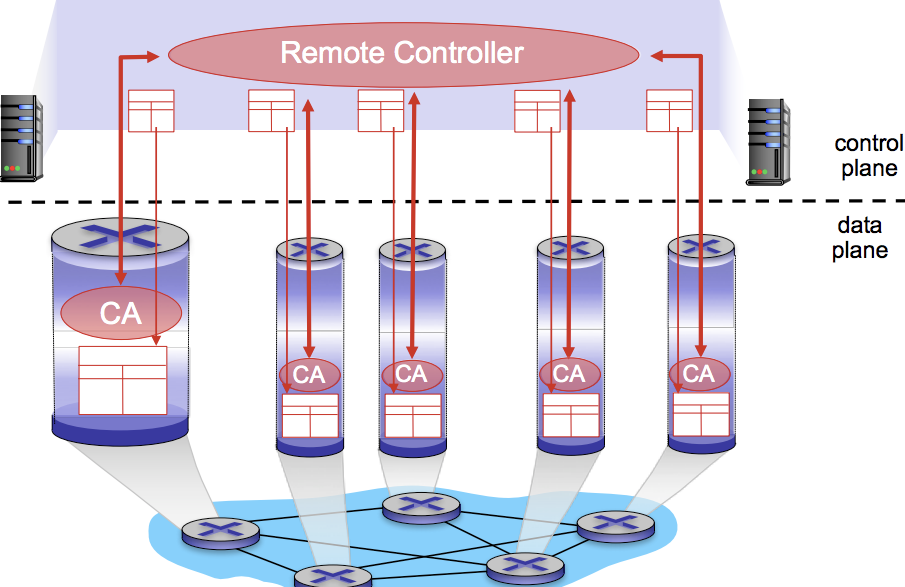
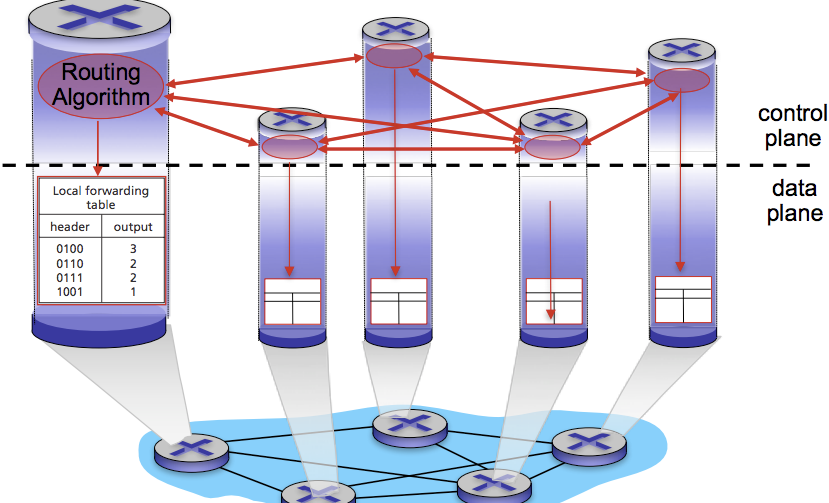
**Two Network Layer Functions**

**Forwarding (DATA PLANE)**: Move packets from router’s input to appropriate router output

**Routing (CONTROL PLANE)**: Determine route taken by packets from source to dest

Two approaches to structuring a network control plane:

* Per-router control (traditional) VS. Logically centralised control (software defined networking)



**Per-Router Control Plane**

Individual routing algo components in EVERY ROUTER interact w/ each other in the control plane to compute forwarding tables.

**Logically Centralised Control Plane**

A distinct (typically remote) controller interacts with local control agents (CA’s) in routers to compute forwarding tables.

**Routing Protocols: Link State / Distance Vector / Hierarchical Routing**

**Interplay between Routing and Forwarding functions in the control / data plane**

* A routing algorithm determines the end-to-end path through the network.
* The forwarding table determines the local forwarding at this router.

**Autonomous System (AS)** or **Domain** is a region of a network under a single administrative authority.

The internet is partitioned into AS’s such as Internet Service Providers (ISPs), each of which controls routes involving its network

* Internet Routing works at two levels:

1. **Intra-Domain Routing Protocol**: Each AS runs this protocol that establishes routes within its domain:
   1. Link State 🡪 **Open Shortest Path First (OSPF)**
   2. Distance Vector 🡪 **Routing Information Protocol (RIP)**
2. **Inter-Domain Routing Protocol**: AS’s participate in this protocol that establishes routes between domains:
   1. Path Vector 🡪 **Border Gateway Protocol (BGP)**

**Graphs Abstraction: Link Costs / Routing Algorithm Classification**

**Key Question**: What is the least-cost path between node X to node Y?

**Routing Algorithm**: Algorithm that finds the least-cost path between Router A to Router Z

**Link Cost**: Assuming that all links are equal, then LEAST COST PATH = SHORTEST PATH (HOP COUNT)

**Network operators add policy exceptions:** Lower operational costs / Peering agreements / Security concerns

Routing algorithm classification:

|  |  |
| --- | --- |
| 1. **Link State (Global)** | 1. **Distance Vector (Decentralised)** |
| * All routers have the complete topology and they maintain / know the cost of each link in the network. * Connectivity/cost changes are flooded to all routers * Converges quickly (less inconsistency, looping etc.) * Limited network sizes | * Routers only know its physically connected neighbours, and the link cost to neighbours. * Connectivity/cost changes iteratively, exchanges info from neighbour to neighbour * Requires multiple rounds to converge * Scales to large networks |

**Link State Routing (Global)**

Each node maintains its local “Link State” i.e. a list of its directly attached links and their costs.

* i.e. in a topology A 🡪 B 🡪 C , the list for Router B would be [ (B, A) , (B, C) ]

Each node floods its local link state.

* On receiving a new Link State message, a router forwards the message to all neighbours other than the one it received the message from.

Flooding **Link State Advertisements (LSA)**

* Routers transmit a Link State Advertisement on links
  + A neighbouring router forwards out on all links except incoming links
  + Keeps a copy locally, don’t forward previously seen LSA’s.
* Challenges: (1) Packet loss (2) Out of order arrival.
* Solutions: (1) Acknowledge and retransmissions (2) Sequence numbers (3) Time-to-live for each packet

Eventually, each node learns the entire network topology: Use Djikstra’s to compute shortest path between nodes.

* Djikstra’s will compute the least cost path from one node (source) to all other nodes (destinations)
* It provides a **forwarding table** to the source node.

**Issue #1: Scalability**

* How many messages needed to flood link state message?
  + **O(N \* E)** , where N = #nodes | E = #edgess
* Processing complexity for Djikstra’s algorithm?
  + **O(N2)** , because we check all nodes W in !visited S at each iteration and we have O(N) iterations.
  + **O(nLog(n))** such as min-heap
* How many entries in the Link State topology database? **O(E)**
* How many entries in the forwarding table? **O(N)**

**Issue #2: Transient Disruptions**

* Inconsistent link-state database
  + Some routers know about failure before others.
  + The shortest paths are no longer consistent.
  + Can cause transient **forward loops**

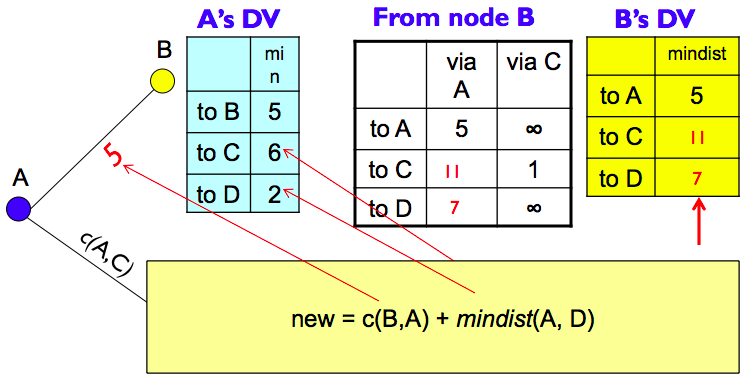
**Oscillations** are possible. Suppose that link cost = amount of carried traffic. Oscillations are possible as costs will start from:

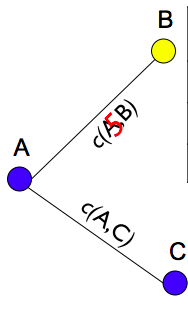
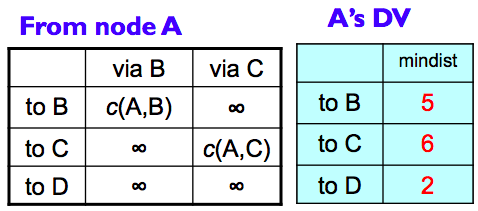
INITIAL COSTS 🡪 GIVEN COSTS, FIND NEW ROUTING = NEW COSTS 🡪 GIVEN NEW COSTS, FIND NEW ROUTING = and so on.

**Distance Vector Routing (Decentralised)**

Each router maintains its shortest distance to every destination via. each of its neighbours.

Each router computes its shortest distance to every destination via. any of its neighbours. **MIN { distB(A,B) , distC(A,B) }**

1. Source router A initialises its **dist()** table based on its immediate neighbours and link costs, then creates DV table
2. A then sends it DV to its immediate neighbours B and C.
3. Router B and C then process received DV’s
4. New cost is calculated, router B and C sends its DV to neighbours and so on



Each router:

* Knows the links to its neighbours
* Has a provisional “shortest path” to every other router – its **Distance Vector (DV)**
* Exchanges this DV with their neighbours
* Looks over the set of options offered by their neighbours and select the best one.

**This** **iterative process** **converges to a set of shortest paths**.

Distance Vector Routing

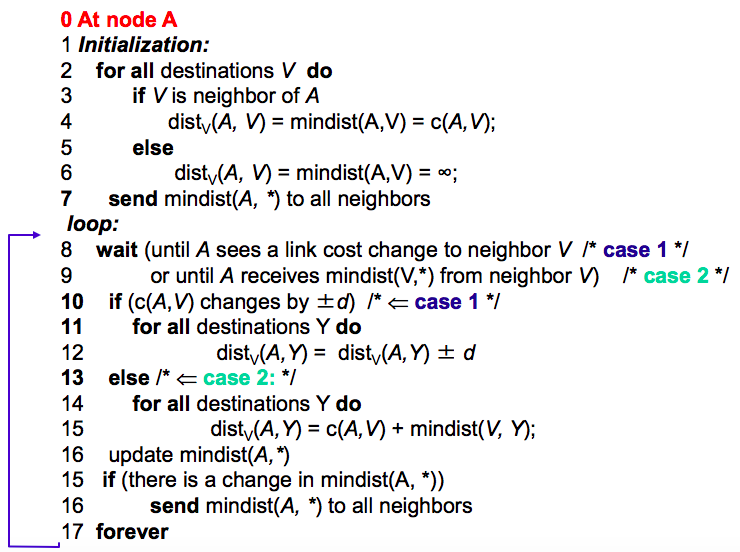
* **Iterative, asynchronous =** each local iteration is caused by:
  + Local link cost change
  + DV update message from neighbour
* **Distributed** = each nod notifies neighbours ONLY when its DV changes
  + Neighbours then notify their neighbours if necessary
* Each node:
  + 🡪 **Waits** for a change in local link cost or msg from neighbour
  + 🡪 **Recomputes** estimates
  + 🡪 **Notify neighbours** if DV to any dest has changed

Pseudocode for Distance Vector:

**c(i,j)** : link cost from node i to j

**distz(A,V)** : shortest dist from A🡪V via. Z

**mindist(A,V)** : shortest dist from A🡪V



Distance-Vector routing protocols have **less computational complexity** and **less** **message overhead** as opposed to Link State protocols.

* DV protocols require a router to inform **neighbours of topological changes periodically**.
* Link State protocols requires a router to inform **all nodes in a network of topology changes**.

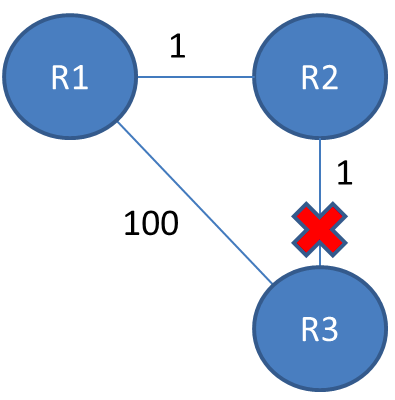
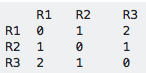
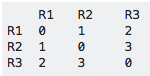
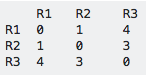
Distance-Vector iteration:

* Initial state: best ONE-HOP paths
* One simultaneous round: best TWO-HOP paths
* Two simultaneous rounds: best THREE-HOP paths

. . .

* Kth simultaneous rounds: best (K + 1) HOP PATHS
* Eventually they converge as soon as it reaches the longest best path.

**Counting to Infinity Problem ( “Bad news travels slowly” )**



**Forwarding Table #1**

* This is the initial routing table.

**Forwarding Table #2**

* Assume the connection between R2 <-> R3 is broken.  
  After one iteration of sending info, you will arrive at the following routing table.
* Because R2 <-> R3 is broken, R2 thinks it can redirect packets to R3 through R1, which has cost R1 <-> R3 = 2, so the total path = 2 + 1 = 3.

**Forwarding Table #3**

* After another iteration, R1 sees that R2 is more expensive than it used to be, so it modifies its routing table.
* And so on until they converge on the correct value, which could take a long time especially if R1 <-> R3 is expensive.

**Poisoned Reverse Rule** is a heuristic to avoid the Count-To-Infinity problem.

* If B routes via. C to get to A, B will tell C that B🡪A distance is infinite. (so C won’t route to A via. B)
* When the protocol detects an invalid route, all of the routers in the network are informed that the bad route has an infinite cost, preventing any of the routers from sending packets over the invalid route.

**Comparison of LS and DV algorithms**

**Message Complexity**

* LS: with n nodes, e links, **O(n \* e)** messages sent
* DV: exchange between neighbours only, convergence time varies

**Speed of Convergence**

* LS: O(n2) algorithm requires O(n \* e) messages, may have oscillations
* DV: convergence time varies, may be routing loops

**Robustness**: what happens if the router malfunctions?

* LS: Node can advertise incorrect link cost. Each node computes only its own table.
* DV: DV node can advertise incorrect path cost. Each node’s table is used by others, errors propagate through network

**Hierarchical Routing**

Our assumptions so far with routing is that ALL ROUTERS ARE IDENTICAL / NETWORK IS FLAT which is not true in practise.

* **Scale** = With 600 million destinations:
  + We can’t store all destinations in routing tables
  + Routing table exchange would swamp links
* **Administrative Autonomy**
  + Internet = network of networks / each network admin may want to control routing in its own network
  + Aggregate routers into regions = **Autonomous Systems (AS)**.
  + Routers in the same AS run the same routing protocol.
* **Gateway Router**
  + At “edge” of its own AS
  + Has link to router in another AS.

Difference between INTRA-AS / INTER-AS routing

* **Policy differences:**
  + Inter-AS: admin wants control over how its traffic is routed, and who routes through its network.
  + Intra-AS: single admin, so no policy decisions needed
* **Scale:**
  + Hierarchical routing saves table size, reduce update traffic
* **Performance differences**:
  + Inter-AS: Policy may dominate over performance
  + Intra-AS: can focus on improving performance

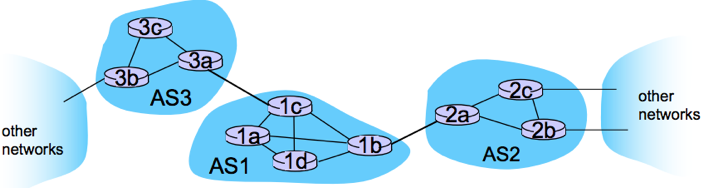
**Interconnected Autonomous Systems**

Forwarding table configured by both Intra-AS and Inter-AS routing algorithms.

* Intra-AS sets entries for internal destinations
* Inter-AS & Intra-AS sets entries for external destinations

**Inter-AS tasks**

* Suppose router in AS1 receives datagram outside of AS1. Router should forward the packet to the gateway router, but which one?
* AS1 must learn which dests are reachable through AS2, AS3 etc.
* Propagate this reachability info to all routers in AS1 = **This is the job of Inter-AS routing ☺**



**INTER-AS**

**=**

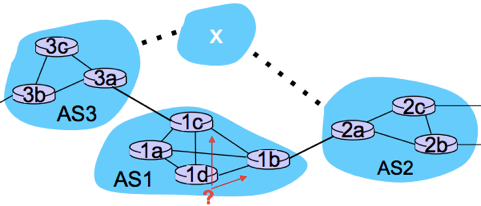
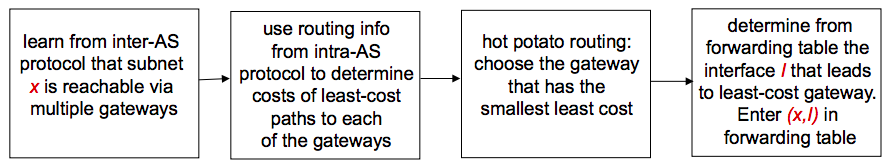
**INTERNAL ROUTERS**

Example: setting a forwarding table in router **1d**.

* Suppose AS1 learns (via. inter-AS protocol) that subnet **X** is reachable via. AS3 (gateway **1c**) but not via. AS2.
* Inter-AS protocol propagates reachability info to all internal routers.
* Router **1d** determines from intra-AS routing info that its interface **1** is on the least cost path to to **1c**, so it installs the forwarding table entry **(X, 1) [ i.e. 1d 🡪 1c 🡪 Subnet X on AS3 ]**

Choosing between multiple AS’s

* AS1 learns from inter-AS protocol that subnet X is reachable from BOTH AS3 and AS2:



**ICMP: Internet Control Message Protocol**

ICMP is a supporting protocol in the IP suite, used to send error messages and operational information.

ICMP is used by hosts & routers to communicate network-level information.

* Error reporting: unreachable host, network, port, protocol
* Echo request / reply (used by ping)

ICMP messages are carried in IP datagrams (in the network-layer “above” IP)

ICMP message:

* Type
* Code + first 8 bytes of IP datagram causing error.

ICMP codes:

