Intra-Domain Networking

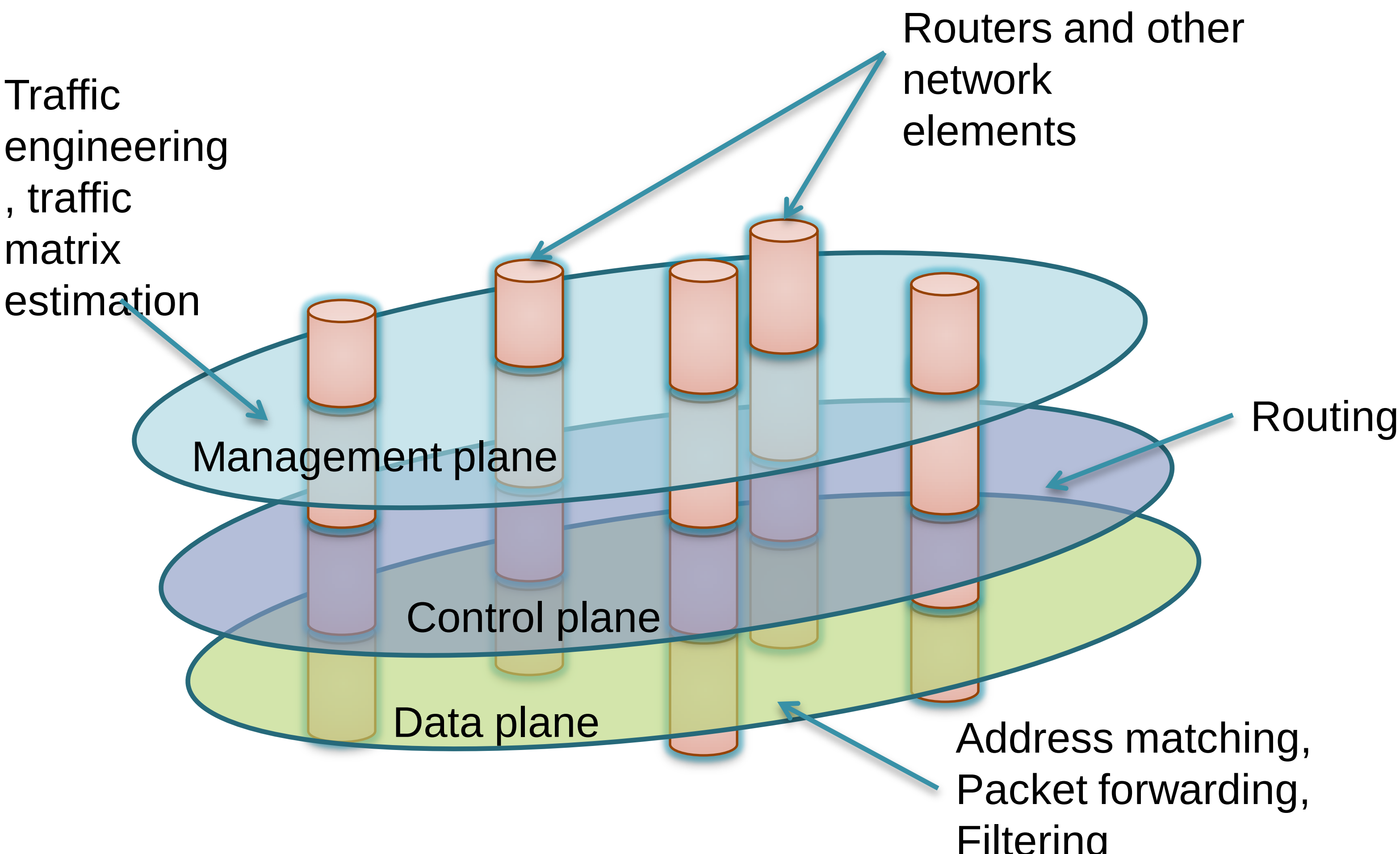
• Planes of the network • Dissemination versus decision • Dissemination algorithms • Routing decisions • Distance vector algorithm • RIP

• Link state algorithm • OSPF • Stability issues

Planes of the Network

- "Planes" is one way of categorizing the functions that go into an operating network
  - ***Data plane***: functions at a router that manipulate the actual packets (e.g., forwarding, matching, filtering)
  - ***Control plane***: network-wide functions that compute the state that goes into data plane (e.g., routing)
  - ***Management plane***: analyze measurement data to create policies, configuration (e.g., traffic engineering, detect and react to DoS)

Planes of the Network..



More on Control Plane

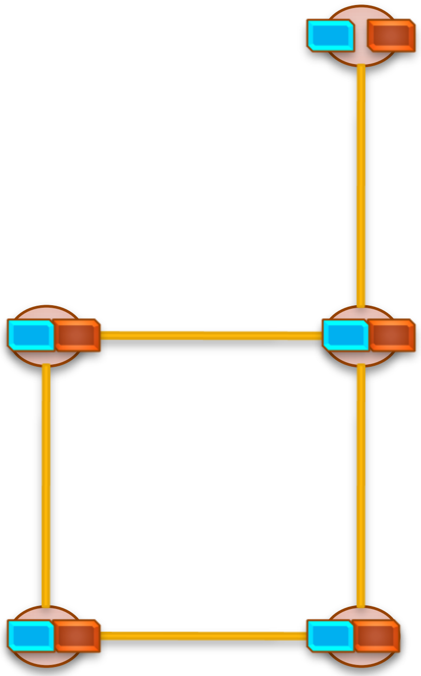
- Routing is one of the key functions of the control plane
- Divide into two activities
  - ***Dissemination*** – state information is spread across the network (e.g., link state information)
  - ***Decision*** – select a path to reach from point X to point Y on the network (e.g., route computation)

Dissemination Algorithms

- What is dissemination?
  - Spread the some information to the whole network
- What are the concerns?
  - ***Spreading time*** – how long does it take for the message to reach everyone?
  - ***Number of messages*** – total overhead as well as the amount of message processed by a router
  - ***Hotspot creation*** – can the algorithm create bottlenecks (can lead to slowdowns or failures)

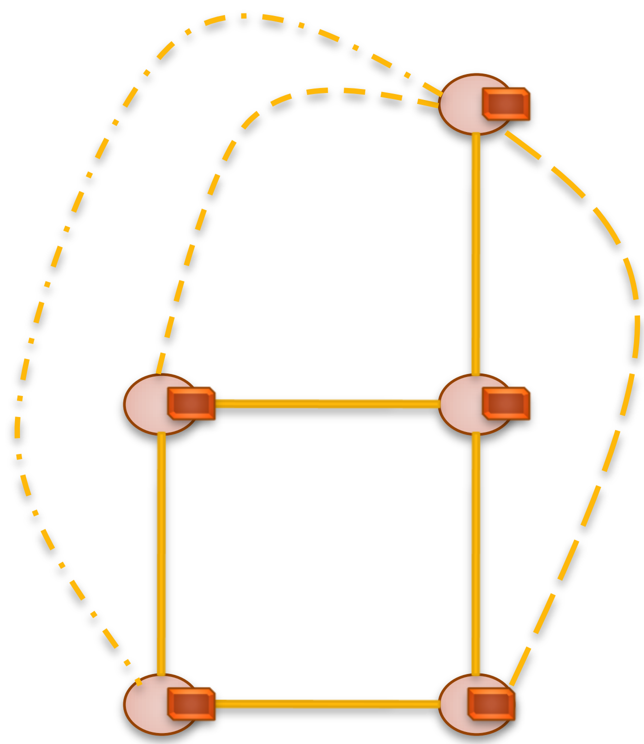
Flooding Algorithms

- Routers organized in a fixed topology
- Communicate along “direct” links only
- The “initial” neighbors don’t change
- Dissemination time is ***diameter*** of the network
- Message complexity is $\Theta(nm)$; n – nodes, m – edges; assuming each node has a message



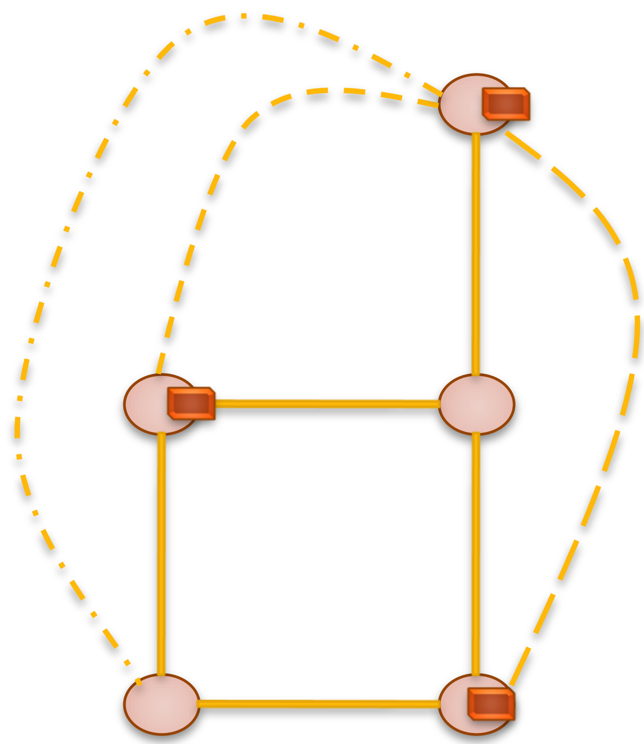
Swamping Algorithms

- Flooding algorithms stick to the "initial" set of neighbors
- Swamping – neighbors grow as messages arrive from distant nodes
- Last iteration – graph is fully connected
- Dissemination time – how long it takes to create a fully connected graph - $\Theta(\log n)$
- Message complexity – more than n^3
- Wasted messages – tell what machines already know



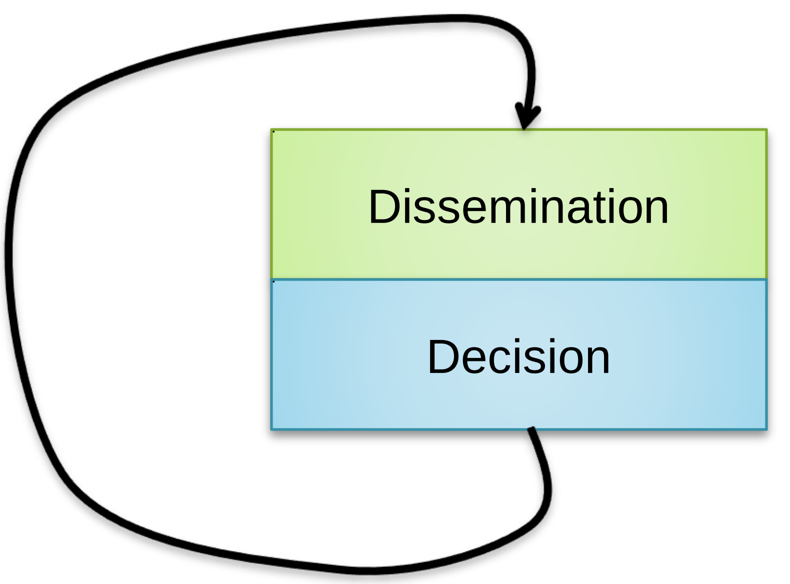
Gossiping Algorithms

- Probabilistic algorithm
- Each node randomly select k neighbors and exchanges messages
- With high probability algorithm converges relatively fast



Decision Algorithms

- Decision algorithms operate on the "state" information aggregated by dissemination algorithms
- One approach is to interleave (OSPF)
- Another is to do both simultaneously (RIP)

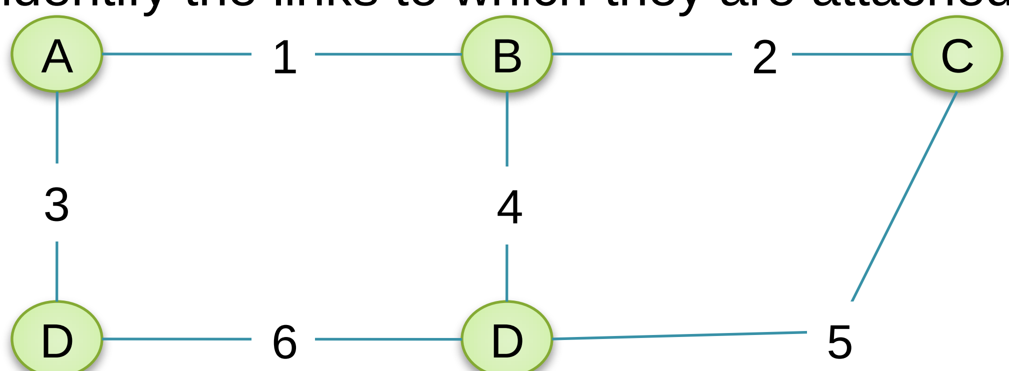


Routing Information Protocol

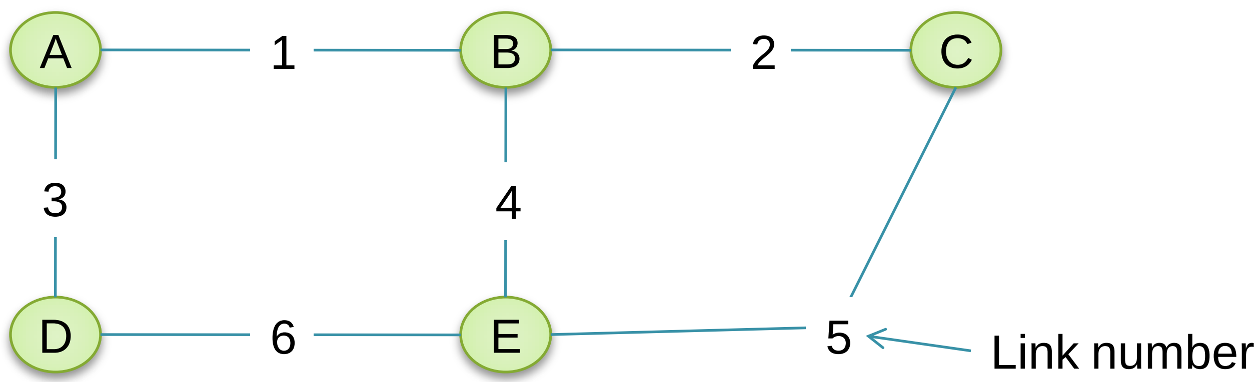
- Simple protocol using “Bellman-Ford” algorithm
- Distributed message-passing type implementation

Distance Vector Routing

- Consider a simple network
- Each node is identified by its address – e.g., A, B, ...
- Suppose the network is powered up simultaneously -- “cold start”
  - the nodes need to remember their addresses
  - identify the links to which they are attached



Distance Vector Routing…

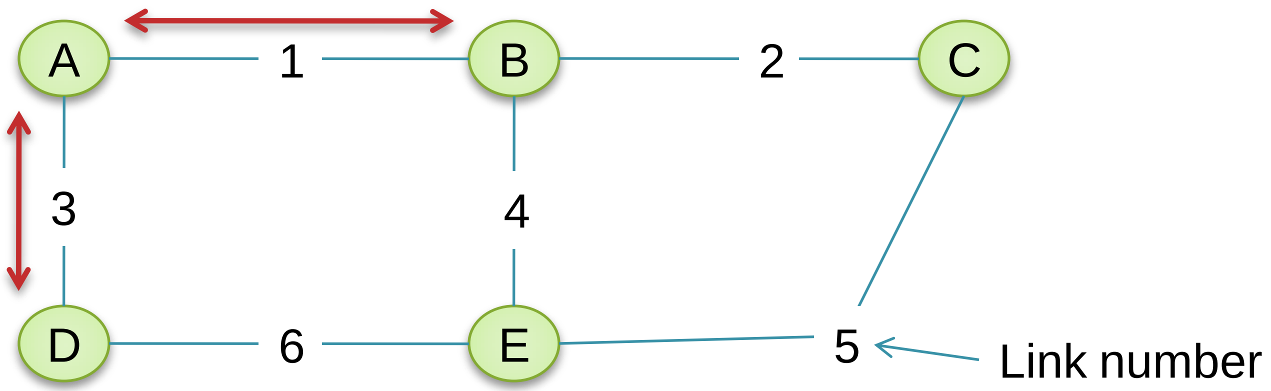


Src	Dst	Link	Cost
A	A	local	0
B	B	local	0
C	C	local	0
D	D	local	0
E	E	local	0

From A to	Link	Cost
A	local	0

Table at the initial (power on state) – shown separately for clarity

Distance Vector Routing…

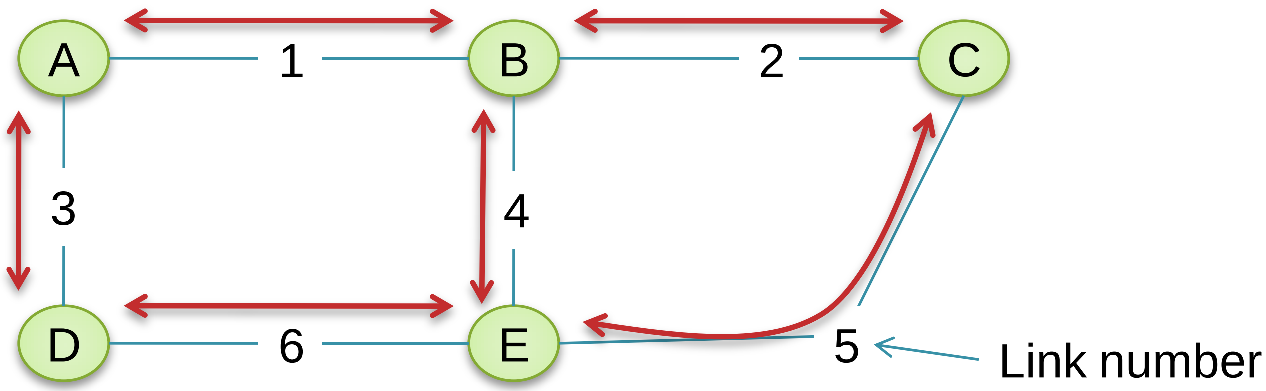


Src	Dst	Link	Cost
A	A	local	0
B	B	local	0
C	C	local	0
D	D	local	0
E	E	local	0

From A to	Link	Cost
A	local	0
B	1	1
D	3	1

Node (e.g., A) broadcasts its info.
to all others on its links
Others use the info. to enlarge their knowledge

Distance Vector Routing…



Src	Dst	Link	Cost
A	A	local	0
B	B	local	0
C	C	local	0
D	D	local	0
E	E	local	0

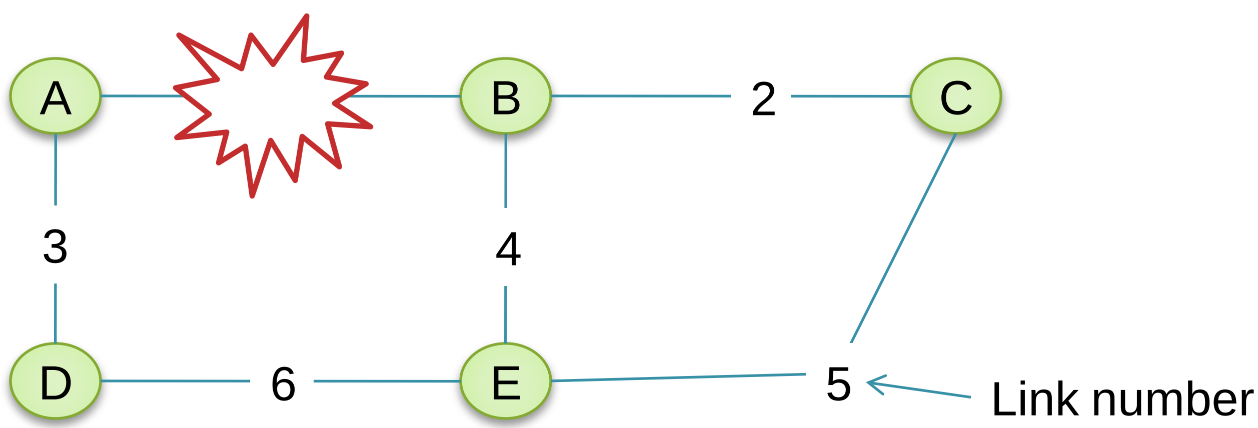
From A to	Link	Cost
A	local	0
B	1	1
D	3	1
C	1	2
E	1	2

Table at A after convergence

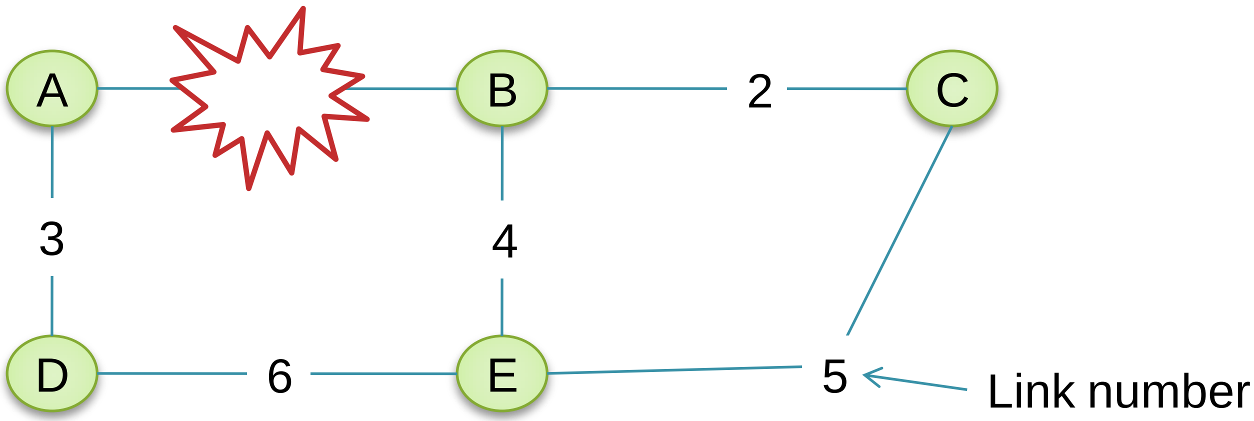
Distance Vector Routing...

What if a link breaks?

- Assume that after the routing tables have been computed, link 1 suddenly breaks



Distance Vector Routing...



From A to	Link	Cost
A	local	0
B	1	inf
D	3	1
C	1	inf
E	1	inf

From B to	Link	Cost
B	local	0
A	1	inf
D	1	inf
C	2	1
E	4	1

Distance Vector Routing…

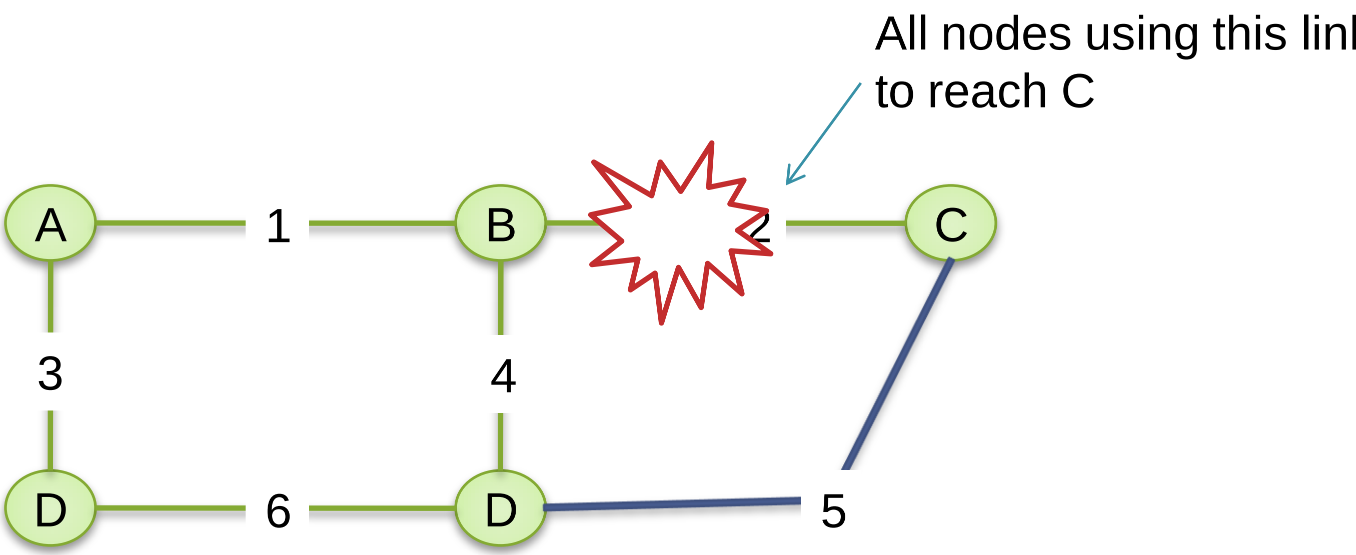
- The distance vector algorithm will update the routing tables -- A's routing table after convergence:

From A to	Link	Cost
A	local	0
B	3	3
D	3	1
C	3	3
E	3	2

Distance Vector Routing…

The Bouncing Effect:

- Assume that link costs are not uniform and (e.g., link 5 has cost 10, while others 1)



- Assume that link #2 breaks -- this is immediately noticed by B -- updating the distance to C to infinity

Distance Vector Routing…

From	Link	Cost
A to C	1	2
B to C	2	1
C to C	local	0
D to C	3	3
E to C	4	2

From	Link	Cost
A to C	1	2
B to C	2	inf
C to C	local	0
D to C	3	3
E to C	4	2

- Suppose before B sends its distance vectors to neighbors, A sends its own distance vectors to B and D
- What will happen if B sends its distance vectors to neighbors ***first***?

Distance Vector Routing…

- The message will have no effect on D
- B will add 1 to A's advertised cost of 2 to reach C and update its value to 3 which is lower than INF
- B advertises this value to A and E
- Creates a loop: packets bound to C will reach B and then bounce back and forth between B and A until TTL expires

From	Link	Cost
A to C	1	4
B to C	1	3
C to C	local	0
D to C	3	3
E to C	4	4

After another iteration

From	Link	Cost
A to C	1	4
B to C	1	5
C to C	local	0
D to C	3	5
E to C	4	4

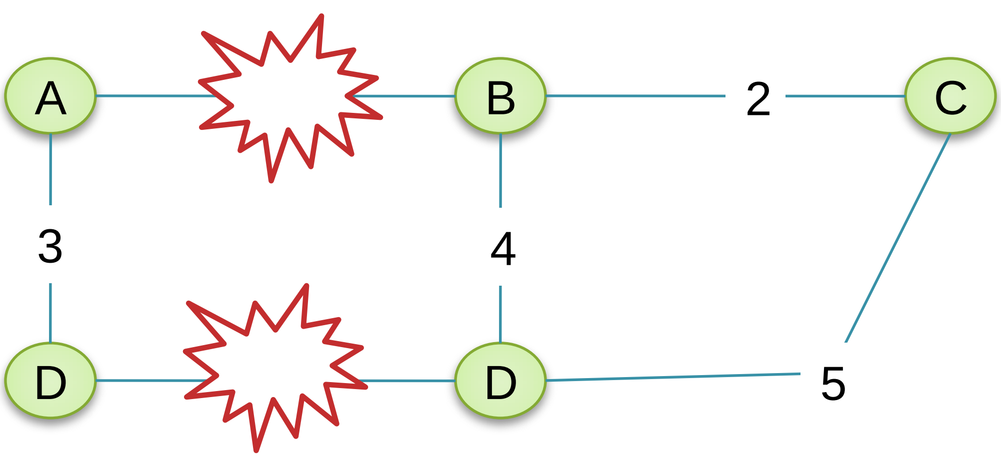
Distance Vector Routing…

- A “round” of update messages increase the cost by 2 units when there is a loop involved
- In this case, the loop will be broken when the distance between E and C as given by the routing table exceeds 10
- During the intermediate states -- when loops occur -- packets will accumulate and “congest” the corresponding links

Distance Vector Routing…

<u>Counting to Infinity:</u>

- Consider following situation:
  - link #1 fails -- ***routing tables updated***
  - link #6 fails -- A & D are isolated from the other nodes



Distance Vector Routing…

- D noticed the link failure and updates its routing table

From D to	Link	Cost
D	local	0
A	3	1
B	6	inf
E	6	inf
C	6	inf

- If D transmits its new value to A the algorithm will converge immediately -- no problem

Distance Vector Routing…

- If A transmits first its last distance vector given by
- A: A =0, B = 3, D = 3, C = 3, E = 3 (link #s shown)
- Table @ D will be updated to:

From D to	Link	Cost
D	local	0
A	3	1
B	3	4
E	3	4
C	3	4

Distance Vector Routing…

- A loop is created
- Because B, C, and E are isolated -- no chance to converge to naturally to a stable state
- At each exchange, distances to B, C, and E increases by 2
- This process is called -- counting to infinity -- can be stopped by a convention that represents a large distance as infinity

Distance Vector Routing…

Split horizon:

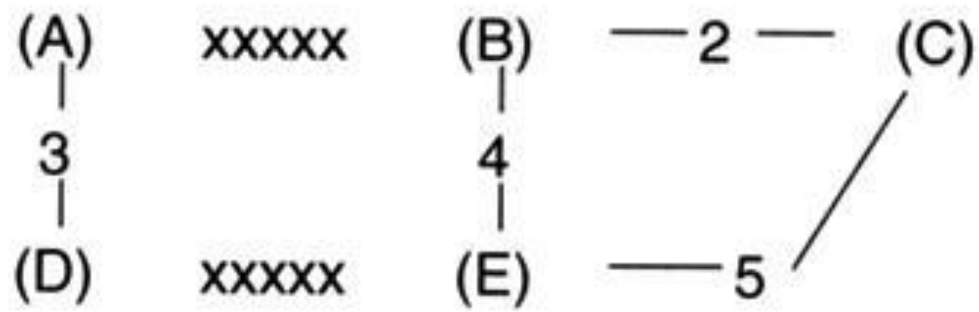
- Bouncing effect and the long time taken for “counting to infinity” are undesirable features of distance vector protocols
- Split horizon is one of the technique to address this problem
- Idea: if A is routing packets to X via B, B should not try to reach X through A
  - A should not announce to B that X is a short distance away from A

Distance Vector Routing…

- Nodes send different versions updates on different outgoing links
- Simple form:
  - nodes omit from messages information about destination routed on the link
- Split horizon with poisonous reverse:
  - will include all destinations in the distance message but will set the corresponding distance to INF
- Kills loops with two hops but three or more can exist!

Distance Vector Routing…

- After link failure between D & E, routing tables at B, C, and E include following entries:



From	Link	Cost
B to D	4	2
C to D	5	2
E to D	6	inf

Distance Vector Routing…

- E notices the failure of the link and sends an advertisement message on links 4 and 5 -- distance to D is INF
- Message reaches B but not C (lost)

From	Link	Cost
B to D	4	inf
C to D	5	2
E to D	6	inf

Distance Vector Routing…

- If C advertises with poisonous reverse
  - advertise INF distance to E on link 5 and a distance of 2 on link 2
- B will update its table and advertise
  - INF on link 2 and distance of 3 on link 4

From	Link	Cost
B to D	2	3
C to D	5	2
E to D	4	4

Distance Vector Routing…

Triggered updates:

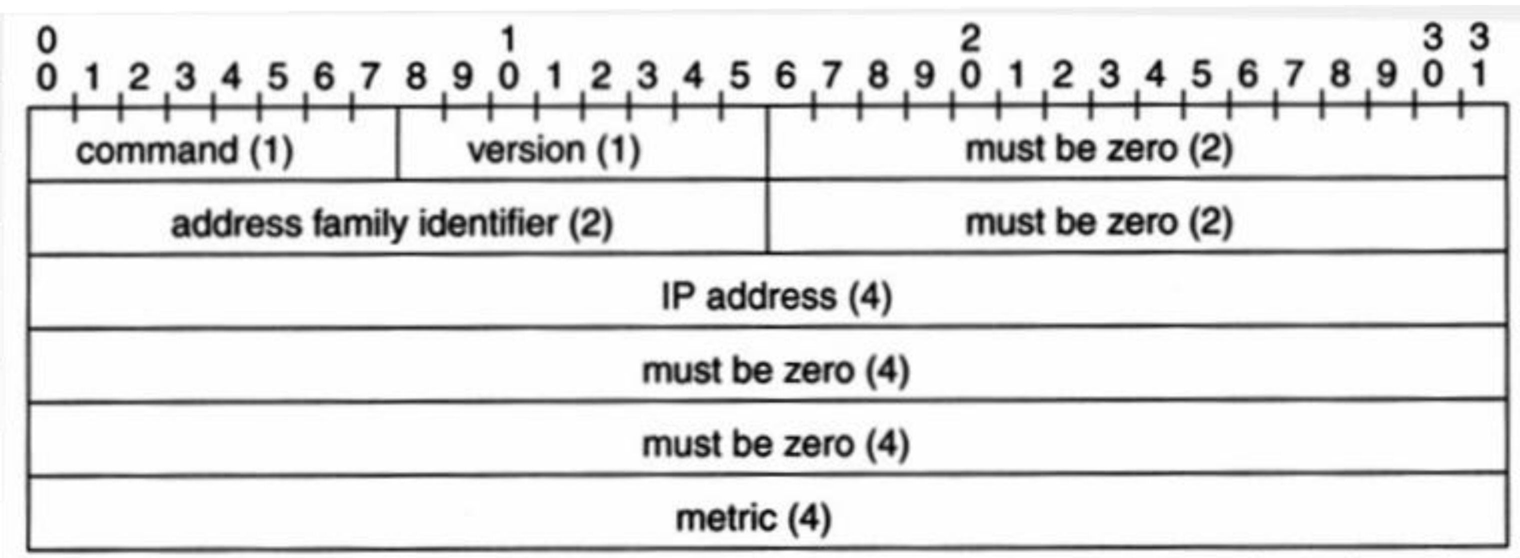
- Issue: when to send update to neighbors?
- Implementations of DV rely on regular sending of distance vectors
- Triggered updates -- nodes should send messages as soon as they notice a change in their routing tables
- Triggered updates can speed up the loop resolution even when counting to

RIP Version 1

- RIP is one distance vector protocol
- RIP is an “internal gateway protocol” (IGP)
  - used within an autonomous system (AS)
- By default, RIP uses hop count as the distance between 1 to 15, 16 is INF
- RIP packets are carried over IP/UDP -- uses UDP port 520 for emission and reception

RIP Version 1

- Packets are normally sent as broadcasts
- Packets sent every 30 seconds or faster incase of triggered updates
- If route not refreshed for 180s set to INF
- **Message format:**



RIP Version 1

<u>RIP processing:</u>

- RIP process on reception of a response -- updates its routing table
  - if entry is not present and if received message is not INF, add it, init the metric to received value, set next router to message sender, start timer

RIP Version 1

- if entry is present with a larger metric, update the metric to received value, set next router to message sender, start timer
- if entry is present and next router is message sender, update metric if it differs from stored value, restart the timer

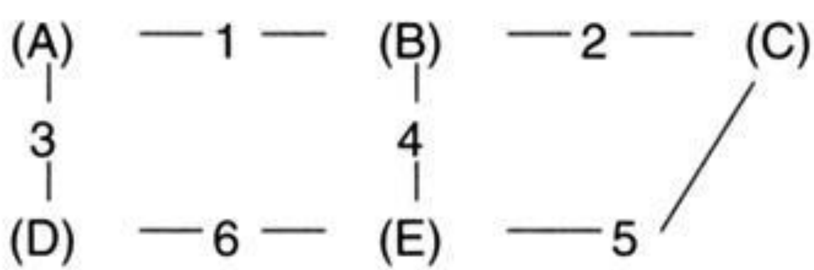
Open Shortest Path First (OSPF)

- Link state protocols are based on a "distributed map" concept
  - all nodes have a network map – regularly updated
- Issues:
  - how the maps are represented
  - how updates are "flooded" to the network nodes
  - why the map updates must be secured
  - how networks can split and then rejoin
  - why "shortest path first?"

OSPF…

Principle:

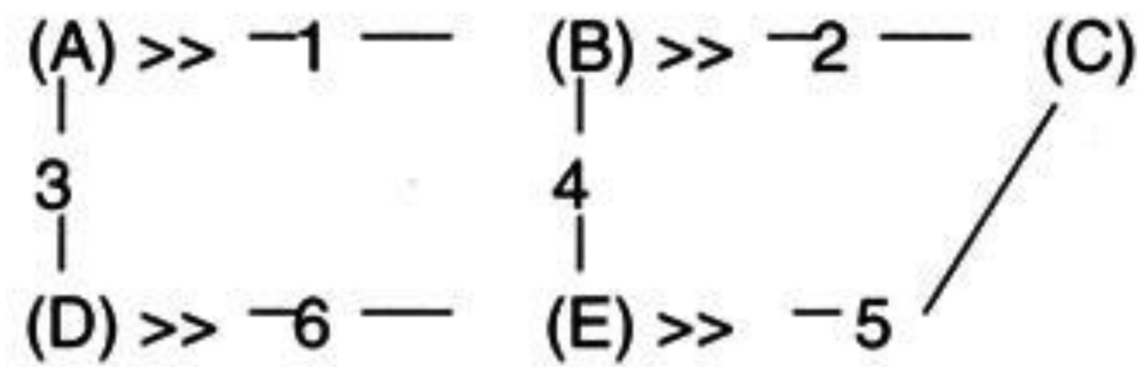
- each node maintains a complete copy of the network map
- performs a complete computation of the best routes from this local map



From	To	Link	Distance
A	B	1	1
A	D	3	1
B	A	1	1
B	C	2	1
B	E	4	1
C	B	2	1
C	E	5	1
D	A	3	1
D	E	6	1
E	B	4	1
E	C	5	1
E	D	6	1

OSPF…

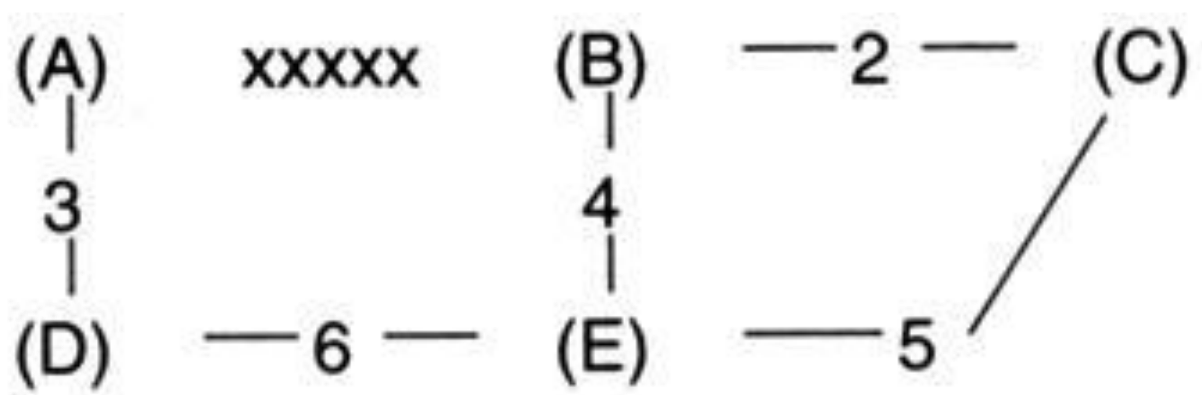
- Each record has been inserted by one station that is responsible for it
- If we send a packet from A to C, we rely on computations by A and B
  - A send on link #1 to B; B sends on link #2 to C



OSPF…

Flooding Protocol:

- A routing protocol should adapt the routes according to network changes
- Database should be updated after each change



OSPF…

Flooding algorithm:

- receive the message; look record in database
- if record not present – add it to database – broadcast the message
- else if record found & database record # is lower, replace record with new value – broadcast msg.
- else if record found & database record # is higher, transmit the database value in a new message through the incoming interface
- else if both record #s are equal – do nothing

OSPF…

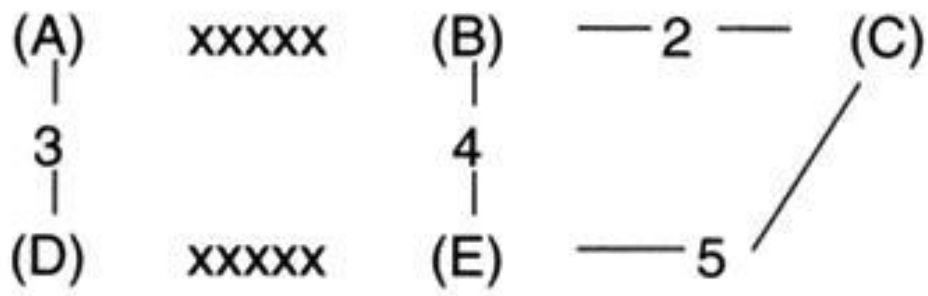
- After the flooding process following is the database:

From	To	Link	Distance	Number
A	B	1	inf	2
A	D	3	1	1
B	A	1	inf	2
B	C	2	1	1
B	E	4	1	1
C	B	2	1	1
C	E	5	1	1
D	A	3	1	1
D	E	6	1	1
E	B	4	1	1
E	C	5	1	1
E	D	6	1	1

The database after flooding

OSPF…

- Bringing up adjacencies:
- Consider the example where we had two failures:



- Failure of link #6 will be detected by D and E
- They can transmit this new information to their “connected” neighbors only
- After executing the flooding, we have two versions of the database

OSPF…

- Two databases will evolve differently – flooding cannot across
- Suppose link #2 fails – one version of the database (I.e., on in A & D) will not detect
  - for routing this is not important – it will be done correctly
- Suppose link #1 becomes operational:
  - records describing link #1 will be corrected
  - records describing links #2 and #6 may be incoherent

OSPF…

- It is necessary to ensure that both sides end up having "aligned" databases
  - known as "bringing up adjacencies" in OSPF
  - two parties should synchronize and keep only the most up-to-date version of each record
- Most records may have similar copies – inefficient to send the records
  - data description packets are sent – link identifiers and version numbers
  - routers compare their version numbers and build a "interesting records" packet –

OSPF…

- In OSPF, we need to protect distributed routing database against corruption
- OSPF includes a number of protections:
  - flooding procedures include hop-by-hop ACKs
  - link state record protected by timers – removed if not refreshed on time
  - records are protected by checksum
  - messages can be authenticated by password

OSPF…

Why is link state protocol better?

- fast, loopless convergence
- support for precise, if needed multiple metrics
- support of multiple paths to a destination
- separate representation of external routes

OSPF…

Fast loopless convergence:

- "Triggered updates" may not require more messages than flooding protocol – but multiple updates may be needed to correct the routing tables
- Most important is the loopless property of OSPF
- Loops can cause congestion and prolong the loop duration → makes OSPF better

OSPF…

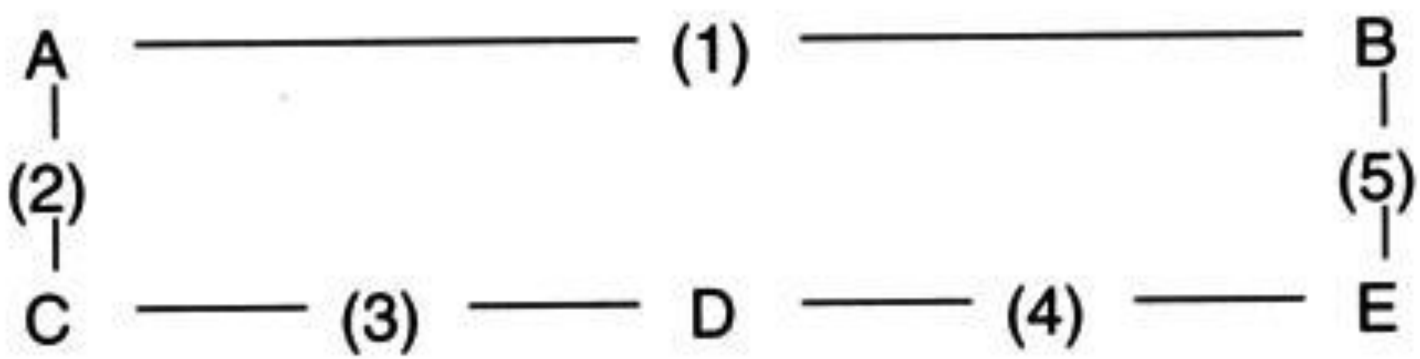
Support for multiple metrics:

- It is difficult for distance vector protocol to support fine-grained metrics – it is not impossible!
- In OSPF, it is possible to have fine-grained variation and also support several metrics in parallel
- “best route” definition is arbitrary:
  - largest throughput
  - lowest delay
  - lowest cost
  - best reliability

OSPF…

- Handling different metrics with link state algorithm requires:
  - documenting several metrics for each link
  - computing different routing tables for each metric
  - presenting the selected metric in packets

OSPF…



- link #1 – T1 satellite link
- link #2 & #3 – T1 terrestrial links
- link #4 & #5 – 64 kbps terrestrial links
- satellite links have long delays (275 ms) and terrestrial links have a short delays (10 ms)

OSPF…

- Path D, C, A, B – throughput 1.5 Mbps and delay 295 ms
- Path D, E, B – throughput 64 Kbps and delay 20 ms

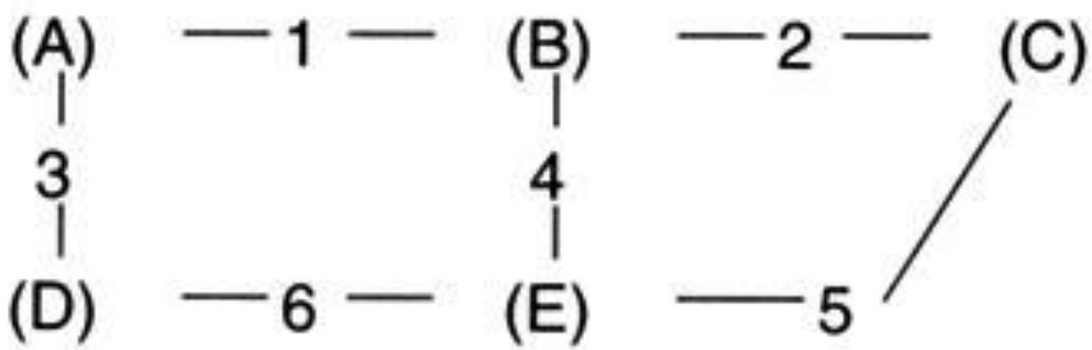
OSPF…

- When throughput metric is used, D, C, A, B path is chosen
- When delay metric is used D, E, B path is chosen
- It is necessary to make consistent decisions
  - if D routes a packet to B to C based on throughput
  - C should use “throughput” for routing this packet otherwise it may route it back to D! – routing loop
  - solution: packet should indicate what metric should be used

OSPF…

<u>Multiple paths:</u>

- "Almost equivalent" paths exists for a given source and destination pair



- Two paths from A to E: one via B and via D
- RIP chooses one arbitrarily because there is only one next hop entry in the routing table

OSPF…

- Splitting traffic over two paths is more efficient
- Simple improvement → give us a list of "shortest paths" to a destination
- Splitting traffic between several paths has downsides too – e.g., with TCP flows
  - packets routed along different paths
  - can arrive out-of-order at the destination
  - can trigger retransmissions

OSPF…

External Routes:

- So far only the “internal routes” problem was considered
- “network” is generally connected through one or several “external gateways” to other “networks”
- When there is only one gateway to the external world – the situation is simple – have default route

OSPF..

- When there are multiple gateways – default route solution is very inefficient
  - it usually picks the nearest external gateway even though another gateway would have been quicker to destination
  - OSPF has “gateway link state records” to support this