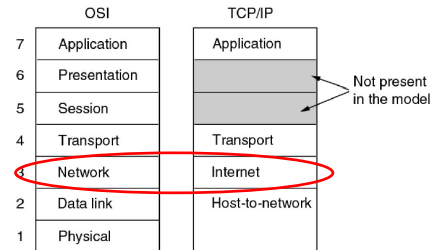


Networking Layer

ECE 650
Systems Programming & Engineering
Duke University, Spring 2016

TCP/IP Model



ECE 650 – Fall 2016

2

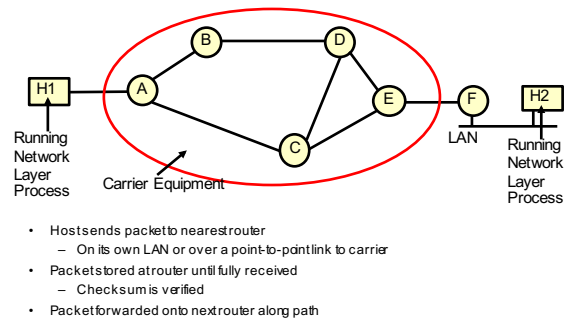
Networking Layer

- Networking Layer's job:
 - Get packets from source all the way to destination
 - In contrast to data link layer:
 - Move frames from one end of wire to the other
- Network Layer must:
 - Be aware of topology of communication subnet
 - i.e. the set of routers
 - Choose appropriate path through subnet
 - Find good path for communication
 - Avoid overloading some communication paths and routers
 - Deal with problems when src and dest are in different networks
- Example:
 - Internet and its IP network protocol layer

ECE 650 – Fall 2016

3

Store-and-Forward Packet Switching



ECE 650 – Fall 2016

4

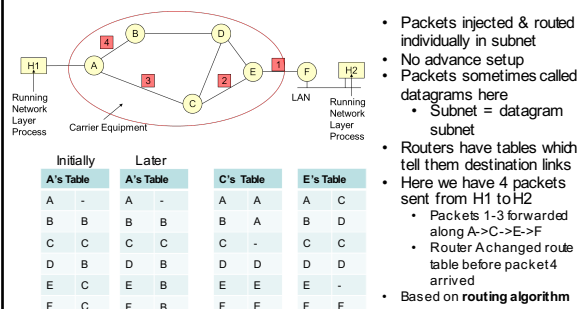
Network Layer Service

- Provides services to transport layer with these goals:
 - Services independent of router technology
 - Hides number, type, topology of routers
 - Network addresses provided to transport layer use uniform numbering scheme, even across LANs, WANs
- Historically, 2 ideas for network layer
 - Connectionless service
 - E.g. Internet
 - Makes sense with subnet is inherently unreliable
 - Higher layers should do error control and flow control
 - Connection-oriented service
 - E.g. ATM, derived from reliable telephone network system
 - Quality of service is a dominant factor

ECE 650 – Fall 2016

5

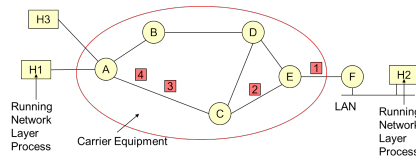
Connectionless Service



ECE 650 – Fall 2016

6

Connection-Oriented Service



A's Table	C's Table	E's Table
H1 1 C 1	A 1 E 1	C 1 F 1
H3 1 C 2	A 2 E 2	C 2 F 2

- Each connection between hosts (virtual circuit) stores an entry in router tables
 - Same route used for all traffic flowing along that connection
- Example, H1 and then H3 establish connection to H2
- Each packet carries an ID of the connection it belongs to
 - In example above, allows router C to distinguish packets in H3→H2 channel from H1→H2

ECE 650 – Fall 2016

7

Qualitative Comparison

Issue	Datagram Subnet	Virtual-Circuit Subnet
Circuit Setup	Not needed	Required
Addressing	Each packet contains the full src and dest address	Each packet contains a short VC number
State information	Routers do not need state info about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is setup; all packets follow it
Effect of router failure	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

ECE 650 – Fall 2016

8

Routing

- Routing algorithm:
 - Determine which output line an incoming packet goes out on
 - May be a new decision for every incoming packet
- Routing vs. Forwarding
 - Forwarding: lookup outgoing line for a packet in routing table
 - Routing: Initialize and update routing tables
- Desirable routing algorithm properties:
 - Correctness, simplicity, robustness, stability, fairness, optimality
 - Some properties trade-off, e.g. fairness vs. optimality
- Adaptive vs. Non-adaptive routing:
 - Adaptive routing changes to reflect topology, current traffic, etc.
 - Also referred to as dynamic vs. static

ECE 650 – Fall 2016

9

Inter vs. Intra-domain routing

- Routing organized in two levels
- Intra-domain routing
 - Complete knowledge, strive for *optimal* paths
 - Scale to ~100 networks
- Inter-domain routing
 - Aggregated knowledge, scale to Internet
 - Dominated by *policy*
 - E.g., route through X, unless X is unavailable, then route through Y
 - Never route traffic from X to Y.
 - Policies reflect business agreements, can get complex

ECE 650 – Fall 2016

10

Optimality Principle

- General statement about optimal routes
- If router J is on the optimal path from router J to K, then:
 - Optimal path from J to K also falls along the same route
- Set of all optimal routes from all sources to a given destination forms a **sink tree**
 - With a root at the destination
- No loops, so each packet delivery is bounded in # hops
- Useful for comparison against other routing algorithms

ECE 650 – Fall 2016

11

Static Routing

- Shortest path routing
 - Build graph of subnet; routers are nodes, links are edges
 - Find shortest path between each pair of routers
 - What is "shortest"?
 - # hops vs. distance vs. queueing delay vs. transmission delay
 - Or some function that takes several factors into account
 - Use Dijkstra's shortest path algorithm
- Flooding
 - Forward every incoming packet out on every other outgoing line
 - Some measures to reduce # of duplicate packets
 - Hop counter per packet; packet discarded on 0
 - Or selective flooding – only send out on lines going in right direction
 - Evaluates every path; and thus guaranteed to find shortest delay

ECE 650 – Fall 2016

12

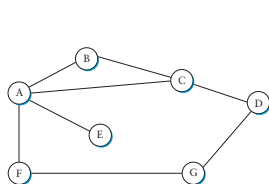
Dynamic Routing

- Two classes of intra-domain routing algorithms
- Distance Vector (Bellman-Ford SP Algorithm)
 - Requires only local state
 - Harder to debug
 - Can suffer from loops
- Link State (Dijkstra-Prim SP Algorithm)
 - Each node has global view of the network
 - Simpler to debug
 - Requires global state

Distance Vector

- Local routing algorithm
 - Also called RIP
- Each node maintains a set of triples
 - $\langle \text{Destination}, \text{Cost}, \text{NextHop} \rangle$
- Exchange updates **with neighbors**
 - Periodically (seconds to minutes)
 - Whenever table changes (*triggered* update)

Distance Vector Example



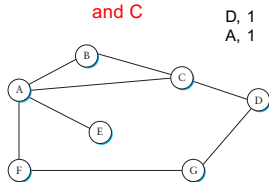
B only exchanges
information with **A**
and **C**

Distance Vector

- Local routing algorithm
- Each node maintains a set of triples
 - $\langle \text{Destination}, \text{Cost}, \text{NextHop} \rangle$
- Exchange updates **with neighbors**
 - Periodically (seconds to minutes)
 - Whenever table changes (*triggered* update)
- Each update is a list of pairs
 - $\langle \text{Destination}, \text{Cost} \rangle$
- Update local table if receive a “better” route
 - Smaller cost

Distance Vector Example

B only exchanges
information with **A**
and **C**



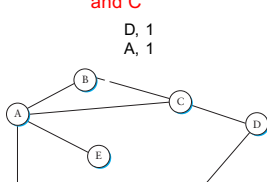
D, 1
A, 1

B's routing table
@ time = 0

Destination	Cost	Next Hop
A	1	A
C	1	C
D	infinity	--
E	infinity	--
F	infinity	--
G	infinity	--

Distance Vector Example

B only exchanges
information with **A**
and **C**



D, 1
A, 1

B's routing table
@ time = 0

Destination	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	infinity	--
F	infinity	--
G	infinity	--

Distance Vector

- Local routing algorithm
- Each node maintains a set of triples
 - $\langle \text{Destination}, \text{Cost}, \text{NextHop} \rangle$
- Exchange updates with neighbors
 - Periodically (seconds to minutes)
 - Whenever table changes (triggered update)
- Each update is a list of pairs
 - $\langle \text{Destination}, \text{Cost} \rangle$
- Update local table if receive a "better" route
 - Smaller cost
- Refresh existing routes, delete if time out

ECE 650 – Fall 2016

19

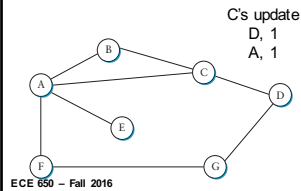
Calculating Best Path

- Bellman-Ford equation

$$D_b(d) = \min_d(\text{infinity}, 1 + 1)$$

$$D_b(A) = \min_A(1, 1 + 1)$$

- Let:
 - $D_b(d)$ denote the current best distance from b to d
 - $C(b,c)$ denote the cost of a link from a to b
- Then $D_b(d) = \min_d(D_b(d), c(b,c) + D_c(d))$



Destination	Cost	Next Hop
A	1	A
C	1	C
D	infinity	--
E	infinity	--
F	infinity	--
G	infinity	--

ECE 650 – Fall 2016

20

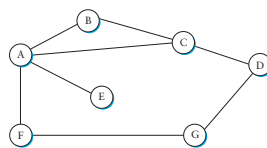
Calculating Best Path

- Bellman-Ford equation
- Let:
 - $D_b(d)$ denote the current best distance from b to d
 - $C(b,c)$ denote the cost of a link from a to b
- Then $D_b(d) = \min_d(D_b(d), c(b,c) + D_c(d))$
- Routing messages contain D
- D is any additive metric
 - e.g. number of hops, queue length, delay

ECE 650 – Fall 2016

21

Distance Vector Example



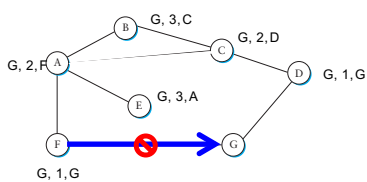
B's routing table

Destination	Cost	Next Hop
A	1	A
C	1	C
D	2	C
E	2	A
F	2	A
G	3	A

ECE 650 – Fall 2016

22

Adapting to Failures

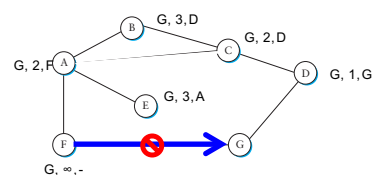


- F-G fails

ECE 650 – Fall 2016

23

Adapting to Failures

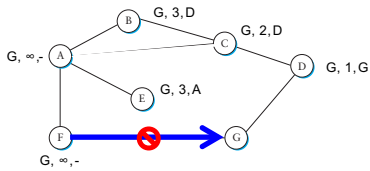


- F-G fails
- F sets distance to G to infinity, propagates

ECE 650 – Fall 2016

24

Adapting to Failures

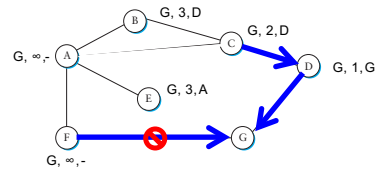


- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity

ECE 650 – Fall 2016

25

Adapting to Failures

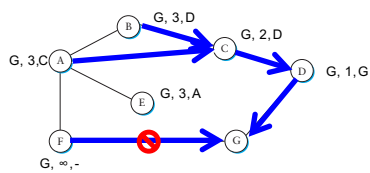


- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity
- A receives periodic update from C with 2-hop path to G

ECE 650 – Fall 2016

26

Adapting to Failures

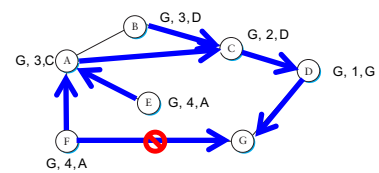


- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and propagates

ECE 650 – Fall 2016

27

Adapting to Failures

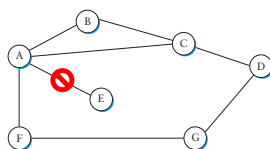


- F-G fails
- F sets distance to G to infinity, propagates
- A sets distance to G to infinity
- A receives periodic update from C with 2-hop path to G
- A sets distance to G to 3 and propagates
- F sets distance to G to 4, through A

ECE 650 – Fall 2016

28

Count-to-Infinity Problem



- Link from A to E fails
- A advertises distance of infinity to E
- B and C advertise a distance of 2 to E
- B decides it can reach E in 3 hops through C
- A decides it can reach E in 4 hops through B
- C decides it can reach E in 5 hops through A, ...
- When does this stop?

ECE 650 – Fall 2016

29

Count-to-Infinity Problem

A	B	C	D	E	
-	-	-	-	-	Initially
-	1	-	-	-	After 1 exchange
1	2	-	-	-	After 2 exchanges
1	2	3	-	-	After 3 exchanges
1	2	3	4	-	After 4 exchanges

- Good news travels fast
- Assume A is down initially, comes up after some time
- Good news travels at rate of 1 hop per exchange

ECE 650 – Fall 2016

30

Count-to-Infinity Problem

A	B	C	D	E	
-	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
					...

- Bad news travels slowly
- Assume A is up initially, goes down after some time
- No router ever has a value more than 1 higher than the minimum of all its neighbors
 - Gradually all work their way up to infinity; Need to bound infinity!

ECE 650 – Fall 2016

31

Avoiding Loops

- IP packet field prevents a packet from living forever
 - Does not *repair* a loop
- Simple approach: consider a small cost n (e.g., 16) to be infinity
 - After n rounds decide node is unavailable
 - But rounds can be long, this takes time
- Problem: distance vector based only on local information

ECE 650 – Fall 2016

32

Better Loop Avoidance

- Split Horizon
 - When sending updates to node A, don't include routes you learned from A
 - Prevents B and C from sending cost 2 to A
- Split Horizon with Poison Reverse
 - Rather than not advertising routes learned from A, explicitly include cost of ∞ .
 - Faster to break out of loops, but increases advertisement sizes
- But still...
 - Split horizon/split horizon with poison reverse only help between two nodes
 - Can still get loop with three nodes involved
 - Might need to delay advertising routes after changes, but affects convergence time

ECE 650 – Fall 2016

33