## L11: Link and Network layer

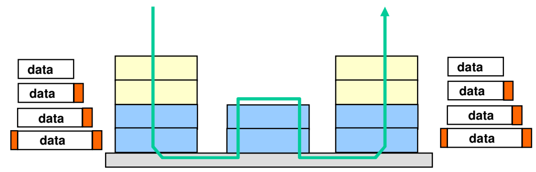
6.033 Spring 2007

http://web.mit.edu/6.033

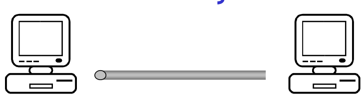
Slides from many folks

## Last lecture: layering of protocols

- Each layer adds/strips off its own header
- Each layer may split up higher-level data
- Each layer multiplexes multiple higher layers
- Each layer is (mostly) transparent to higher layers



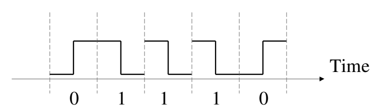

## Link Layer



Problem:
Deliver data from one end of the link to the other

Need to address:
- Bitsà Analog à Bits
- Framing
- Errors
- Medium Access Control (The Ethernet Paper)

## Manchester encoding




- Each bit is a transition
- Allows the receiver to sync to the sender's clock

## Framing

- Receiver needs to detect the beginning and the end of a frame
- Use special bit-pattern to separate frames
  - E.g., pattern could be 1111111 (7 ones)
- *Bit stuffing* is used to ensure that a special pattern does not occur in the data
  - If pattern is 1111111 à Whenever the sender sees a sequence of 6 ones in the data, it inserts a zero (reverse this operation at receiver)

## Error Handling

- Detection:
  - Use error detection codes, which add some redundancy to allow detecting errors
- When errors are detected
  - Correction:
    - Some codes allow for correction
  - Retransmition:
    - Can have the link layer retransmit the frame (rare)
  - Discard:
    - Most link layers just discard the frame and rely on higher layers to retransmit

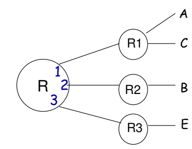
Network Layer:

finds a path to the destination and forwards packets along that path

- Difference between routing and forwarding
  - Routing is finding the path
  - Forwarding is the action of sending the packet to the next-hop toward its destination

## Forwarding

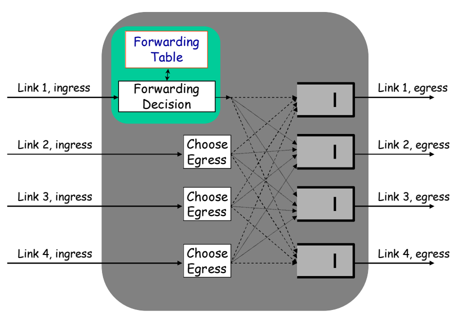
- Each router has a forwarding table
- Forwarding tables are created by a routing protocol



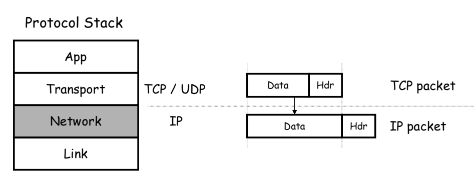

Forwarding table at R

| Dst. Addr | Link |
|---|---|
| A | 1 |
| B | 2 |
| C | 1 |
| E | 3 |

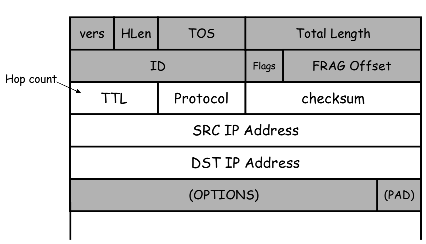
## Inside a router




## The Internet Protocol (IP)



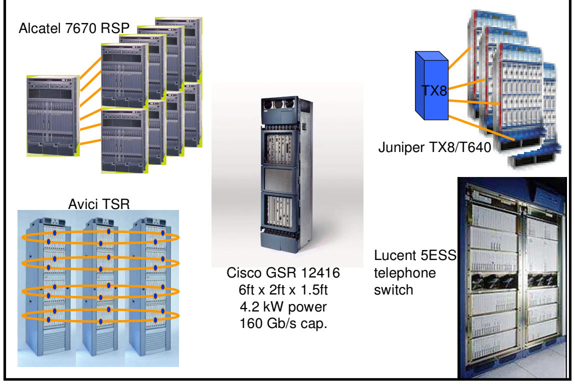

## The IP Header




## Forwarding an IP Packet

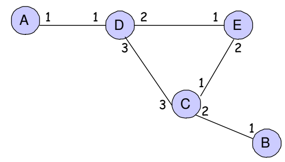
- Lookup packet's DST in forwarding table
  - If known, find the corresponding outgoing link
  - If unknown, drop packet
- Decrement TTL and drop packet if TTL is zero; update header Checksum
- Forward packet to outgoing port
- Transmit packet onto link

## And switches today…




## The Routing Problem:

- Generate forwarding tables




Goals: No loops, short paths, etc.

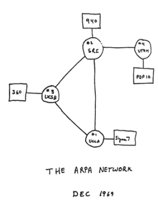
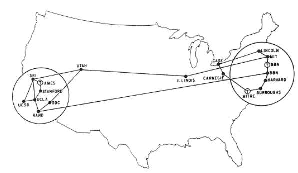
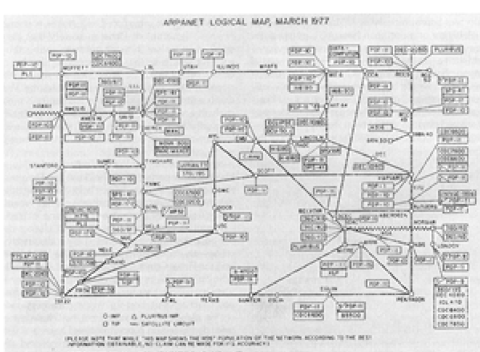
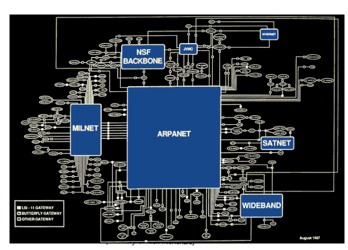


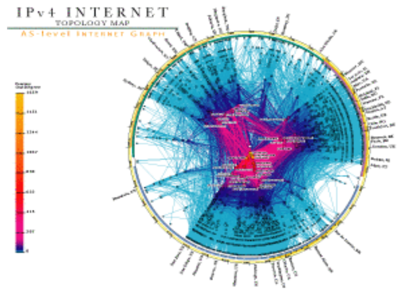

FIGURE 6.2 Drawing of 4 Node Network
(Courtesy of Alex McKenzie)



MAP 4 September 1971



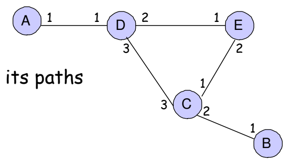








## Path Vector Routing Protocol

- Initialization
  - Each node knows the path to itself

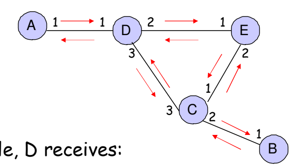



For example, D initializes its paths

| DST | Link | Path |
|---|---|---|
| D | End layer | null |

## Path Vector

- Step 1: Advertisement
  - Each node tells its neighbors its path to each node in the graph




For example, D receives:

From A:

| To | Path |
|---|---|
| A | null |

From C:

| To | Path |
|---|---|
| C | null |

From E:

| To | Path |
|---|---|
| E | null |

## Path Vector

- Step 2: Update Route Info
  - Each node use the advertisements to update its paths

D received: From A:

| To | Path |
|---|---|
| A | null |

From C:

| To | Path |
|---|---|
| C | null |

From E:

| To | Path |
|---|---|
| E | null |

D updates its paths:

| DST | Link | Path |
|---|---|---|
| D | End layer | null |

| DST | Link | Path |
|---|---|---|
| D | End layer | null |
| A | 1 | <A> |
| C | 3 | <C> |
| E | 2 | <E> |

Note: At the end of first round, each node has learned all one-hop paths

## Path Vector

- Periodically repeat Steps 1 & 2

In round 2, D receives:

From A:

| To | Path |
|---|---|
| A | null |
| D | <D> |

From C:

| To | Path |
|---|---|
| C | null |
| D | <D> |
| E | <E> |
| B | <B> |

From E:

| To | Path |
|---|---|
| E | null |
| D | <D> |
| C | <C> |

D updates its paths:

| DST | Link | Path |
|---|---|---|
| D | End layer | null |
| A | 1 | <A> |
| C | 3 | <C> |
| E | 2 | <E> |

| DST | Link | Path |
|---|---|---|
| D | End layer | null |
| A | 1 | <A> |
| C | 3 | <C> |
| E | 2 | <E> |
| B | 3 | <C, B> |

Note: At the end of round 2, each node has learned all two-hop paths

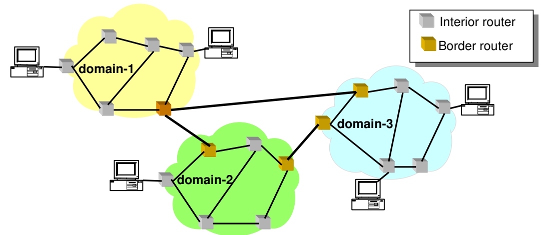
## Questions About Path Vector

- How do we avoid permanent loops?
- What happens when a node hears multiple paths to the same destination?
- What happens if the graph changes?

## Questions About Path Vector

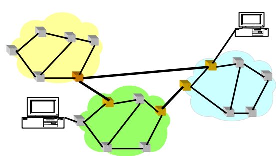
- How do we ensure no loops?
  - When a node updates its paths, it never accepts a path that has itself
- What happens when a node hears multiple paths to the same destination?
  - It picks the better path (e.g., the shorter number of hops)
- What happens if the graph changes?
  - Algorithm deals well with new links
  - To deal with links that go down, each router should discard any path that a neighbor stops advertising

## Hierarchical Routing




- Internet: collection of domains/networks
- Inside a domain: Route over a graph of routers
- Between domains: Route over a graph of domains
- Address consists of "Domain Id", "Node Id"

## Hierarchical Routing



Advantage
- Scalable
  - Smaller tables
  - Smaller messages
- Delegation
  - Each domain can run its own routing protocol

Disadvantage
- Mobility is difficult
  - Address depends on geographic location
- Sup-optimal paths
  - E.g., in the figure, the shortest path between the two machines should traverse the yellow domain.

## Routing: many open issues

- Flat addresses and scalable?
- Routing in multihop WiFi networks?
- Routing in peer-to-peer networks?
- Misconfigurations between domains?