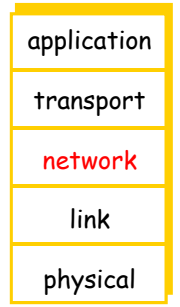
Network Layer Overview

Kai Shen

Internet Architecture

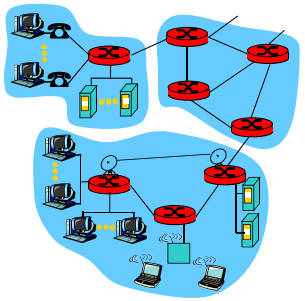
Bottom-up:

- physical: electromagnetic signals "on the wire"
- link: data transfer between neighboring network elements
  - encoding, framing, error correction, access control for shared links
- network: host-to-host connectivity
  - routing, addressing
- transport: host-host data transport
  - reliable data transport, congestion control, flow control
- application: anything you want to do on computer networks

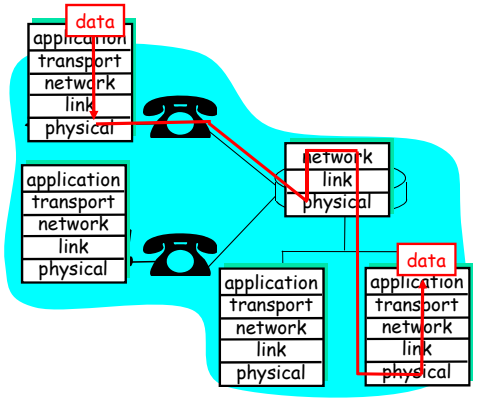


Network Layer Function

- transport packet from sending to receiving hosts
  - routing: determine a path from source to dest and route packets along the path
  - addressing: uniquely identify each node in the network
- network connecting devices
  - called "routers"
  - participate in network protocols
- links
  - connect adjacent hosts, routers

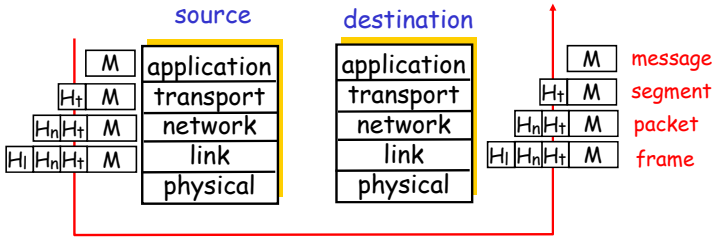


Protocol Layers and Data Flows



Data Formation Across Layers

Each layer takes data from above
- adds header information to create new data unit
- passes new data unit to layer below



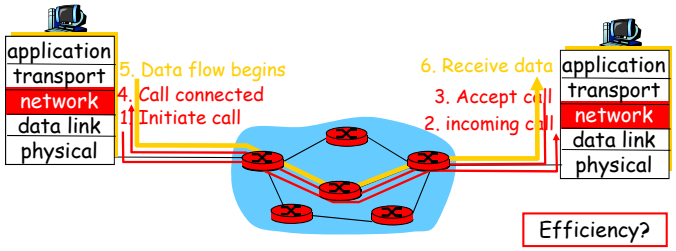
Network Service Model

- loss-free delivery?
- in-order delivery?
- preservation of inter-packet delay (no jitter)?
- guaranteed bandwidth?
- congestion feedback to sender?

connection-based or packet switching?

Connection-based Networks

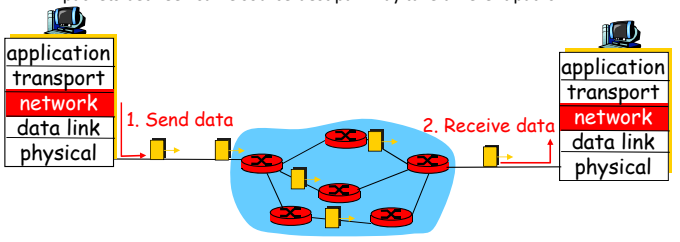
- guaranteed bandwidth, jitter-free, in-order delivery build on the concept of connections, or virtual circuits
- require signaling protocols to setup, maintain, and teardown virtual circuit
- router maintains state about ongoing connections



Efficiency?

Packet Switching

- no connection ⇨ every packet is on its own
- stateless routers ⇨ no need to remember anything about packets
- packets forwarded independently from each other
  - packets between same source-dest pair may take different paths



Packet Switching vs. Connection-based Networks

Packet switching
- poor service guarantee
  - "elastic" service
- efficiency
- scalability
- robustness
- flexibility/customization
  - simple network core, complexity at "edge"

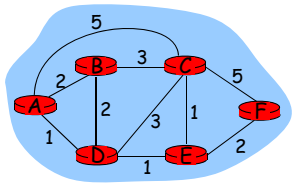
⇒ manage, control, adapt at "smart" end systems (computers)

Connection-based
- evolved from telephony
- guaranteed service
  - e.g., human conversation:
  - strict timing, reliability requirements
- hard to evolve
  - complex network core

Internet employs packet switching; connection-based are more suitable for networks with "dumb" end terminals – telephones.

Routing Principles

Routing protocol
Goal: determine "good" path (sequence of routers) thru network from source to dest.



Graph abstraction for routing algorithms:
- graph nodes are hosts or routers
- graph edges are links
  - link cost: delay, $ cost, or congestion level

- "good" path:
  - typically means minimum cost path

Routing Algorithm Classification

Global information:
- all routers have complete topology, link cost info
- "link state" algorithm

Decentralized:
- router knows connected neighbors, link costs to neighbors
- exchange of info with neighbors to learn remote parts of the network, may take many learning rounds
- "distance vector" algorithm

A Link-State Routing Algorithm

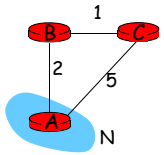
Dijkstra's algorithm
- Network topology, link costs/distances known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- The algorithm calculates the shortest paths from one specific node ("source") to all other nodes
- Basic approach: gradually expands a set "N", containing nodes whose shortest paths from the source are known
  - initially "N" contains only the source itself
  - at every step one more node's shortest path from the source is learned, this node is then added to "N"
  - eventually "N" includes every node and the algorithm terminates

Dijkstra's Algorithm (cont.)

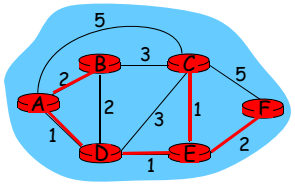
"N", containing nodes with known shortest paths from source

Key step: How to expand "N" to include one more node?

- "1-more-hop paths":
  - paths from source to dest. v (not in N) whose last hop before v is in N
  - D(v) as the distance of the shortest 1-more-hop path to v
- For arbitrary v, is D(v) shortest path distance to v?
- For v* with shortest D(v) among all v's not in N, D(v*) is the shortest path distance to v*.



Dijkstra's Algorithm: An Example



- N: set of nodes whose shortest paths are currently known
- D(v): distance for shortest 1-more-hop path from source to dest. v
- p(v): predecessor node along path from source to dest. v

Step	N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

Dijkstra's Algorithm: Complexity

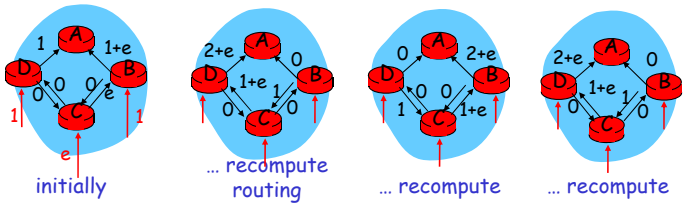
Algorithm complexity: n nodes, e links

- Each iteration: need to check all nodes, v, not in N
  ⇨ $n*(n-1)/2$ checks: $O(n^2)$
- Update 1-more-hop paths: O(e)
- Total: $O(n^2+e)$, or $O(n^2)$
- Using Fibonacci heap to find minimum distance node
  ⇨ O(nlogn + e)

Dijkstra's Algorithm: Stability

Oscillations possible:

- e.g., link cost = amount of carried traffic



Solutions:

- asynchronous (at different time) adjustments across routers
- stable cost metric (independent of routing policy)

Disclaimer

- Parts of the lecture slides contain original work of James Kurose, Larry Peterson, and Keith Ross. The slides are intended for the sole purpose of instruction of computer networks at the University of Rochester. All copyrighted materials belong to their original owner(s).