ECEN 757

Lecture 4: Internet

PCs,routers, switches… =nodes

links= *edges*

Internet = Connecting a huge number of heterogeneous networks

	Example	*Range*	*Bandwidth (Mbps)*	*Latency (ms)*
Wired:				
LAN	Ethernet	1–2 kms	10–10,000	1–10
WAN	IP routing	worldwide	0.010–600	100–500
MAN	ATM	2–50 kms	1–600	10
Internetwork	Internet	worldwide	0.5–600	100–500
Wireless:				
WPAN	Bluetooth (IEEE 802.15.1)	10–30m	0.5–2	5–20
WLAN	WiFi (IEEE 802.11)	0.15–1.5 km	11–108	5–20
WMAN	WiMAX (IEEE 802.16)	5–50 km	1.5–20	5–20
WWAN	3G phone	cell: 1–-5	348–14.4	100–500

km

© Pearson Education 2012

Conceptual layering of protocol software

© Pearson Education 2012

Internet 7-Layer Networking Stack

Packet Encapsulation

- **On the sending side:**
- **Each layer gets a message from the layer above**
- **Each layer adds its own header at the beginning of the message**

Packet Encapsulation

- On the receiving side:
- Each layer gets a message from the layer below
- Each layer decodes its corresponding header

Overview

- **We will talk about three fundamental mechanisms**
- **Routing**
- **TCP**
- **DNS**

Routing

- **How to deliver a packet from A to C?**
  - **A cannot talk to C directly**

Routing Table

Routings from A		
To	*Link*	*Cost*
A	local	0
B	1	1
C	1	2
D	3	1
E	1	2

Routing Algorithms

- Nodes connected in some topology. Routing algorithm runs at network layer in each node.
- Goal of routing algorithm:
  - given the destination IP address in packet, determine the "next hop"
  - thus determine the route for each packet as it travels through the net
  - dynamically update routing information to reflect failures, changes (e.g., router joins and leaves) and congestion (overloaded router)
- Two approaches:
  - Distance-vector (e.g., RIP)
    - Every node knows, for each possible destination LAN, the next-hop
  - Link-state (e.g., OSPF)
    - Every node knows status of every "link" in the network

  In both, information maintained as a table

  Tables updated either
  - Proactively – periodically, or
  - Reactively – when a neighbor/some link status changes

Distance Vector Routing

- **Also termed as distributed Bellman-Ford algorithm or Ford-Fulkerson algorithm, included in RIP (Routing Information Protocol), AppleTalk, and Cisco routers.**
  - Each node/router maintains a table indexed by each destination node. Entry gives (best known distance to destination, best next-hop for destination)
  - Once every T seconds, each router sends to each neighbor its own entire table. Neighbor uses this to update its own table. (proactive)

Pseudo-Code for RIP

Send: Each *t* seconds or when *Tl* changes, send *Tl* on each non-faulty outgoing link.

Receive: Whenever a routing table *Tr* is received on link *n*:

```
for all rows Rr in Tr {
    if (Rr.link not equal n) {
        Rr.cost = Rr.cost + 1;
        Rr.link = n;
        if (Rr.destination is not in Tl) add Rr to Tl;
         // add new destination to Tl
        else for all rows Rl in Tl {
            if (Rr.destination = Rl.destination and
                    (Rr.cost < Rl.cost or Rl.link = n)) Rl = Rr;
            // Rr.cost < Rl.cost : remote node has better route
            // Rl.link = n : remote node is more authoritative
        }
    }
}
```

Distance Vector

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Initial distances stored at each node (global view)

Distance Vector

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Initial routing table at node A

Distance Vector

Destination	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Final routing table at node A

Distance Vector

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Final distances stored at each node (global view)

Distance Vector

- **The distance vector routing algorithm is sometimes called as Bellman-Ford algorithm**
- **Every T seconds each router sends its table to its neighbor each each router then updates its table based on the new information**
- **Problems include fast response to good new and slow response to bad news. Also too many messages to update**

Fault Tolerance

- **If a neighbor fails…**
- **Set the distance to it to be ∞**
- **Can take a long time to recover**
  - **There are various ways to speed up**

Link State Routing

- **Each router must**
  1. **Discover its neighbors and learn their network addresses**
     - **When a router is booted up, it learns who its neighbors are by sending a special Hello packet on each point-to-point link.**
     - **The router on the other end sends back a reply.**
  2. **Measure the delay or cost to each of its neighbors**
     - **A router sends a special Echo packet over the link that the other end sends back immediately. By measuring the round-trip time, the sending router gets an RTT estimate.**
  3. **Construct a packet telling all it has just learned.**
     - **<u>Broadcast</u> this packet**

Link State Routing

- A router broadcasts a link-state-advertisement (LSA) packet after booting, as well as periodically (or upon topology change). Packet forwarded only once, TTL-restricted
- Initial TTL is very high, since need it to get to every router

(a)

Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

(b)

Link State Routing

4. **Broadcast the LSA packet to all other routers.**
   - **Each packet contains a <u>sequence number</u> that is incremented for each new LSA packet sent by that source router.**
   - **Each router keeps track of all the (source router, sequence) pairs it sees. When a new LSA packet comes in, it is checked against these pairs. If the received packet is new, it is forwarded on all the links except the one it arrived on. Old LSA packets are dropped.**
   - **The <u>age</u> of each packet, being stored after reception, is decremented once per time unit. When the age hits zero, the information is discarded. Initial age = high. Such state is often called <u>soft state</u>.**
5. **For routing a packet, since the source knows the entire network graph, it simply computes the shortest path (actual sequence of nodes) locally using the Dijkstra's algorithm. It can include the path in the packet, and intermediate nodes simply follow this route to decide their next hop for the packet.**

Transport Layer =Transmission Control Protocol

- **Function 0 – provide an application with a connection-oriented view of the network (IP is connectionless)**
- **Function 1 (Message decomposition and reassembly): Breaks messages into packets at the transmitting end and reassembles packets into messages at the receiving end.**
  - **E.g., using identification and fragment offset fields in IPv4 header**
- **Function 2 (Multiplexing and demultiplexing): Multiplexes several lower-rate sessions, all from the same source and all going to the same destination, into one session at the network layer.**
- **Function 3 (Reliable communication): Provides reliability to the application by acks + retransmissions in an end to end manner**
- **Function 4 (End-to-end congestion/flow control): Reduces rate at which data is sent when congestion is detected in the network. (TCP-friendliness)**
- **Function 5 (Ordering): Of packets within a stream -- FIFO**
- ***All these functionalities are a part of TCP.***

Functions 3 - 5

- **Transmissions are prone to many non-ideal behaviors**
- **Packets can be arbitrarily delayed**
- **Packets may not arrive in order**
- **Packets may be dropped – Usually due to congestion**

DNS: Domain Name System

People: many identifiers:

- Address: Mailing address, email address, telephone number
- Name: E.g., John Smith

Internet hosts, routers:

- Address: IP address (32/64 bit) - used for addressing datagrams
- Name: URL e.g., ece.tamu.edu – human-readable format

Q: given a resource name, how does a client find out the IP address of the service/server?

Domain Name System:

- *distributed database* implemented in a hierarchy of many *name servers*
- *application-layer protocol that is responsible for resolving* names (address/name translation)

DNS: Domain Name System

People: many identifiers:

- Address: Mailing address, email address, telephone number
- Name: E.g., John Smith

Internet hosts, routers:

- Address: IP address (32/64 bit) - used for addressing datagrams
- Name: URL e.g., ece.tamu.edu – human-readable format

Name: human-readable string

Address: machine-readable string

Name Resolution: Name → Address

E.g.,: given a resource name, how does a client find out the IP address of the service/server

DNS: Domain Name System

People: many identifiers:

- Address: Mailing address, email address, telephone number
- Name: E.g., John Smith

Internet hosts, routers:

- Address: IP address (32/64 bit) - used for addressing datagrams
- Name: URL e.g., ece.tamu.edu – human-readable format

Q: given a resource name, how does a client find out the IP address of the service/server?

Domain Name System:

- *distributed database* implemented in a hierarchy of many *name servers*
- *application-layer protocol that is responsible for resolving* names (address/name translation)

DNS Name Servers

Why not have a central DNS server?

- **single point of failure**
- **traffic volume**
- **may be far**
- **maintenance difficult**

Doesn't *scale!*

(WWW contains several billion pages today)

Alternative

- **no single server has all name-to-IP address mappings**
- **Hierarchy of name servers**

authoritative name servers:

- **for a resource, definitively stores the mapped IP address for that resource**

local name servers:

- **each institution/company/ISP owns a *local (default) name server***
- **Receives DNS queries from host**
- **Caches recently seen name→address mappings, and can answer queries quickly**

DNS: Root Name Servers

- contacted by local name server that cannot resolve query
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server
- ~ 13 root name servers worldwide (as of '12)

Simple DNS Example

host `surf.eurecom.fr` **wants IP address of** `dragon.cs.uiuc.edu`

1. **Contacts its local DNS server,** `dns.eurecom.fr`
2. `dns.eurecom.fr` **contacts root name server, if necessary**
3. **root name server contacts authoritative name server,** `dns.cs.uiuc.edu`, **if necessary**

Answer returned by first server that is caching the mapping tuple

DNS Example

Root name server:

- may not know the authoritative name server
- may know *intermediate name server:* whom to contact to find authoritative name server
- Server Hierarchy parallels URL hierarchy

DNS: Iterated Queries

recursive query:

- puts burden of name resolution on servers along the way
- may fail if a server does not know next server to contact, or if intermediate server fails

iterated query:

- contacted server replies with name of server to querying server
- "I don't know this resource name, but ask this other server"
- takes longer (more longer replies) but gives client more control

DNS: Caching and Updating Records

- **Once (any) name server learns mapping, it *caches* mapping**
  - **cache entries timeout (disappear) after some time**
- **Content Distribution Network (CDN): Use local servers to cache some popular files**
- **Need to redirect requests for these popular files to local servers**
- **Update/notify mechanisms: insert new DNS entries**
  - **RFC 2136**
  - **https://www.rfc-editor.org/info/rfc2136**
  - **Akamai/CDNs used this extensively to redirect HTTP requests to nearby caching servers**