3/11/09 -- Lecture 11

Sam Madden

Network Layer

Today: Network layer.

Job of network layer is to find and forward packets along a path between sender and receiver in a multi-hop network.

Show example network:

"routers"

Network layer interface

Two main jobs:

Forwarding -- sending data over links according to a *routing table*
Routing -- process whereby routing tables are built

Forwarding:

Show routing table in network

Point out that many routing tables are possible (example)

Show stack -- annotate with "net_send" and "link_send" and "net_handle" and "e2e_handle" calls, encapsulation

Show link selection in stack

Forwarding -- mechanical. Just perform a lookup in a table.

Pseudocode.

```
forwarding_table t

net_send(payload, dest, e2eprot):
    pkt = new packet(payload,dest,e2eprot)
    net_handle(pkt)

net_handle(pkt):
    if (pkt.dest == LOCAL_ADDR):
        e2e_handle(pkt.payload, pkt.e2eprot)
    else:
        link_send(t[p.dest].link, pkt)
```

Routing -- compute the forwarding table

How to compute forwarding table? Manually -- not scalable.

Centrally -- not a good idea (why?)

- need a routing algorithm to collect
- collection requires many messages
- hard to adapt to changes

Path Vector Algorithm -- Distributed

Each node maintains a forwarding table T , with:

"e2e_handle" calls, encapsulation

Dest	Link	Path

Two steps:

advertise (periodic)

send T to neighbors

```
integrate(N,neighbor, link) -- on receipt of advertisement from neighbor
    merge neighbor table N heard from neighbor on link into T
```

```
Merging:
    for each dest d w/ path r in N:
        if d not in T, add (d, link, neighbor ++ r) to T
        if d is in T, replace if (neighbor ++ r) is shorter than old path
```

Example:

(If everybody picks best path to every dest, you can see that for a network with most distant nodes separated by N hops, in N rounds everyone will know how to reach everyone else in N steps.)

Q: what is the purpose of keeping the path in the table?

Problems:

- permanent loops?
    - won't arise if we add a rule that we don't pick paths with ourselves in them; this is what we need the path for!

- temporary loops -- arise because two nodes may be slightly out of date
    example
    - soln: add send count -- "TTL" -- to packet

- failures / changes -- repeat advertisements periodically,
    remove paths in your table that aren't re-advertised
        (e.g., a path P that begins with router R should be in the

next advertisement from R.)

- graph changes -- same as failures

How does this work on the Internet:

At first, internet was a small network like this

Show evolution slides

What is the problem with using path vector here?

Network is huge

> 1 B nodes on network

Even if we assume most of those are compters that connect to only their local router (so don't really need to run the path vector protocol), there are still many millions of routers in the Internet

Each router needs to know how to reach of these billions of computers

With pure path vector, each node has a multi-billion entry table (requiring gigabytes of storage)

Each router has to send these gigabyte tables to each of its neighbors; millions of advertisements propagating around. Disaster.

Solution: hierarchical routing

Subdivide net into areas; with multiple levels of routing

One node representative of each area; perform path vector at area level. Within each area, free to do whatever. (For example, use more hierarchy.)

(e.g., a path P that begins with router R should be in the

How to name nodes:

    area.name

On Internet, this is IP address

    E.g., 18.7.22.69 -- this is mit.edu

Internet routers running -- BGP -- advertise prefixes of these address

Show advertisements (e.g., "18.*.*.*"...)
17.1*.*.*