Network Layer

Chapter 5

Physical and Data Link Layers

Virtual Path

Actual Path

Services Provided to Network Layer

- Transferring data from the network layer on source machine to the network layer of the destination machine
  - Unacknowledged connectionless service
  - Acknowledged connectionless service
  - Acknowledged connection-oriented service

Network Layer

- Responsibility
  - end-to-end delivery of packets to the network
  - selecting routes for the packets to take
  - implies knowledge of the network topology
  - managing utilization of the links
  - provide flow control (across multiple links)
  - spread load among different routes
- Interface Design
  - should be independent of subnet technology
  - hide number, type, and topology of network from upper layers
  - export a common number plan for entire network

Connection vs. Connectionless

- Two possible designs for network layer
  - connection oriented service (ATM)
    - based on experience of telcos
  - connectionless service (IP)
    - based on packet switching (ARPANET)
- Connectionless
  - transport datagrams from source to destination
  - end-point addresses in every datagram
  - less complex network layer, more complex transport
- Connection oriented
  - also called virtual circuits
  - establish an end-to-end connection with network state
  - can use VCI (global or next hop) in each packet
Datagram vs. VC Addresses

- **Datagrams**
  - must include full address in each packet
  - addresses must be unique for entire network
  - don’t re-use too often
  - addresses per src/dest pair
- **Virtual Circuit**
  - globally unique
  - requires allocation scheme to ensure its unique
  - consumes many bits per packet
  - per link
  - requires translation at each switch
  - uses fewer bits (important for small packets like ATM)

Virtual Circuit vs. Datagram

<table>
<thead>
<tr>
<th>Issue</th>
<th>Datagram</th>
<th>Virtual Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit setup</td>
<td>not needed</td>
<td>necessary</td>
</tr>
<tr>
<td>Addresses</td>
<td>full source/dest per packet</td>
<td>next hop vc sufficient</td>
</tr>
<tr>
<td>state</td>
<td>no state in network</td>
<td>per connection data at each router</td>
</tr>
<tr>
<td>routing</td>
<td>each packet individually</td>
<td>once at VC setup</td>
</tr>
<tr>
<td>router/link failure</td>
<td>a few packets may be lost</td>
<td>all VCs through router are terminated</td>
</tr>
<tr>
<td>congestion control</td>
<td>difficult</td>
<td>many pre-allocation and policing policies permitted</td>
</tr>
</tbody>
</table>

Routing

- Find a path from the source node to the destination node through the network
- Required in nearly all networks
- Routing Algorithm
  - responsible for deciding which output line an incoming packet should be transmitted on
  - Decided for each packet in datagram environment
  - Decided for a connection when it is established

Routing: Goals

- Correctness
  - packets get where they are supposed
- Simplicity
  - easy to implement correctly
  - possible to make routing choices fast (or updates easy)
- Robustness
  - failures in network still permit communication
- Stability
  - small changes in link availability results in a small change in the routing information
- Fairness
  - each host, VC, or datagram has the same chance
- Optimality
  - best possible route
  - best utilization of bandwidth

Do Routes Change During Network Operation?

- Non-adaptive routing (static routing)
  - information loaded at boot time
  - never changes during network operation
- adaptive routing
  - changes in network operation after routes
  - issue: where to get this data to make choices
    - from neighbors
    - globally from all routers (or a NIC - network information center)
  - issue: when to change routes
    - only when topology changes (links or routers change)
    - in response to changes in load
    - issue: metric to optimize
      - distance, number of hops, estimated latency
Optimality Principal

- If J is on the optimal route from I to K
  - then the optimal route from I to K shares the optimal route from J to K
- transitive result of this is a sink tree
  - can construct a tree from all nodes to a specific node

Shortest Path Routing

- Graph Representation
  - nodes are routers
  - arcs are links
  - to get between two routers, select the shortest path
  - need to decide metric to use for minimization
- Dijkstra’s Algorithm
  - select source as current node
  - while current node is not destination
    - for each neighbor of current
      - if route via current is better update its tentative route
    - label node with <distance, current Node>
    - find tentative node with shortest route
    - mark a permanent
    - make it current

Shortest Path Example

- Every incoming packet is resent on every outbound link
- generates many duplicate packets
- potentially infinite packets unless they are damped
  - multiple paths to the same destination result in loops
  - can use a lifetime (max hops) to damp traffic
  - can also keep track in routers if the packet has been seen
- good metric to compare algorithms
  - flooding always chooses the shortest path
  - must ignore overhead and congestion due to flooding

Flow-Based Routing

- Compute optimal routes off-line if we know in advance:
  - link capacity
  - topology
  - traffic for each <src, dest> pair
- Testing a routing table:
  - given a tentative routing table
  - for each link we can compute mean delay
  - C is link capacity bps, 1/m is mean packet size, l is actual traffic in packets/sec
- then compute overall utilization (as mean or max of delays)
- possible to exhaustively try all routing tables this way

Flow Based Routing

- Graph Representation
  - nodes are routers
  - arcs are links
  - to get between two routers, select the shortest path
  - need to decide metric to use for minimization
- Dijkstra’s Algorithm
  - select source as current node
  - while current node is not destination
    - for each neighbor of current
      - if route via current is better update its tentative route
    - label node with <distance, current Node>
    - find tentative node with shortest route
    - mark a permanent
    - make it current
Distance Vector Routing

- Also known as Bellman-Ford or Ford-Fulkerson
  - original ARPANET routing algorithm
  - early versions of IPX and DECnet used it too
- Each router keeps a table of tuples about all other routers
  - outbound link to use to that router
  - metric (hops, etc.) to that router
  - routers also must know “distance” to each neighbor
  - each router then updates its table based on the new info
- Problems:
  - fast response to good news
  - slow response to bad news
    » takes max hops rounds to learn of a downed host
    » known as count-to-infinity problem

Distance Vector Routing Tables

<table>
<thead>
<tr>
<th>Router</th>
<th>0</th>
<th>12</th>
<th>25</th>
<th>40</th>
<th>14</th>
<th>23</th>
<th>18</th>
<th>17</th>
<th>21</th>
<th>9</th>
<th>24</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Count to Infinity Problem

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td></td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>E</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>G</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>H</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>I</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>J</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>K</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>L</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>M</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>

Distance Vector Routing

- Environment
  - links and routers unreliable
  - alternative paths scarce
  - traffic patterns can change rapidly
- Two key algorithms
  - distance vector
  - link-state
- Both assume router knows
  - address of each neighbor
  - cost of reaching each neighbor
- Both allow a router to determine global routing information by talking to its neighbors

Basic idea

- Node tells its neighbors its best idea of distance to every other node in the network
- Node receives these distance vectors from its neighbors
- Updates its notion of best path to each destination, and the next hop for this destination
- Features
  - distributed
    - adapts to traffic changes and link failures
  - suitable for networks with multiple administrative entities
Example

Why does it work

- Each node knows its true cost to its neighbors
- This information is spread to its neighbors the first time it sends out its distance vector
- Each subsequent dissemination spreads the truth one hop
- Eventually, it is incorporated into routing table everywhere in the network
- Proof: Bellman and Ford, 1957

Problems with distance vector

- Count to infinity

Dealing with the problem

- Path vector
  - DV carries path to reach each destination
- Split horizon
  - never tell neighbor cost to X if neighbor is next hop to X
  - doesn’t work for 3-way count to infinity
- Triggered updates
  - exchange routes on change, instead of on timer
  - faster count up to infinity
- More complicated
  - source tracing
  - DUAL

Link State Routing

- Used on the ARPANET after 1979
- Each Router:
  - computes metric to neighbors and sends to every other router
  - each router computes the shortest path based on received data
    - Needs to estimate time to neighbor
      - best approach is send an ECHO packet and time response
- Distributing Info to other routers
  - each router may have a different view of the topology
  - simple idea: use flooding
  - refinements
    - use age sequence number to damp old packets
    - use aks to permit reliable delivery of routing info

Link state routing

- In distance vector, router knows only cost to each destination
  - hides information, causing problems
- In link state, router knows entire network topology, and computes shortest path by itself
  - independent computation of routes
  - potentially less robust
- Key elements
  - topology dissemination
  - computing shortest routes
Link state: topology dissemination

- A router describes its neighbors with a link state packet (LSP)
- Use controlled flooding to distribute this everywhere
  - store an LSP in an LSP database
  - if new, forward to every interface other than incoming one
  - a network with E edges will copy at most 2E times

Sequence numbers

- How do we know an LSP is new?
- Use a sequence number in LSP header
- Greater sequence number is newer
- What if sequence number wraps around?
  - smaller sequence number is now newer! (hint: use a large sequence space)
- On boot up, what should be the initial sequence number?
  - have to somehow purge old LSPs
  - two solutions
    - aging
    - lollipop sequence space

Aging

- Creator of LSP puts timeout value in the header
- Router removes LSP when it times out
  - also floods this information to the rest of the network (why?)
- So, on booting, router just has to wait for its old LSPs to be purged
- But what age to choose?
  - if too small
    - purged before fully flooded (why?)
    - needs frequent updates
  - if too large
    - router waits idle for a long time on rebooting

A better solution

- Need a unique start sequence number
- a is older than b if:
  - \( a < 0 \) and \( a < b \)
  - \( a > 0 \), \( a < b \), and \( b-a < N/4 \)
  - \( a > 0 \), \( b > 0 \), \( a > b \), and \( a-b > N/4 \)

More on lollipops

- If a router gets an older LSP, it tells the sender about the newer LSP
- So, newly booted router quickly finds out its most recent sequence number
- It jumps to one more than that
- \(-N/2\) is a trigger to evoke a response from community memory
Connected Subnets

Recovering from a partition

- On partition, LSP databases can get out of synch
- Databases described by database descriptor records
- Routers on each side of a newly restored link talk to each other to update databases (determine missing and out-of-date LSPs)

Router failure

- How to detect?
  - HELLO protocol
  - HELLO packet may be corrupted
    - so age anyway
    - on a timeout, flood the information

Securing LSP databases

- LSP databases must be consistent to avoid routing loops
- Malicious agent may inject spurious LSPs
- Routers must actively protect their databases
  - checksum LSPs
  - ack LSP exchanges
  - passwords

Computing shortest paths

- Basic idea
  - maintain a set of nodes P to whom we know shortest path
  - consider every node one hop away from nodes in \( P \)
  - find every way in which to reach a given node in \( T \), and choose shortest one
  - then add this node to \( P \)

Example
**Link state vs. distance vector**

- **Criteria**
  - stability
  - multiple routing metrics
  - convergence time after a change
  - communication overhead
  - memory overhead
- Both are evenly matched
- Both widely used

**Choosing link costs**

- Shortest path uses link costs
- Can use either static or dynamic costs
- In both cases: cost determines amount of traffic on the link
  - lower the cost, more the expected traffic
  - if dynamic cost depends on load, can have oscillations (why?)

**Static metrics**

- Simplest: set all link costs to 1 => min hop routing
- but 28.8 modem link is not the same as a T3!
- Give links weight proportional to capacity

**Dynamic metrics**

- A first cut (ARPAnet original)
- Cost proportional to length of router queue
  - independent of link capacity
- Many problems when network is loaded
  - queue length averaged over a small time => transient spikes caused major rerouting
  - wide dynamic range => network completely ignored paths with high costs
  - queue length assumed to predict future loads => opposite is true (why?)
  - no restriction on successively reported costs => oscillations
  - all tables computed simultaneously => low cost link flooded

**Modified metrics**

- queue length averaged over a small time
- wide dynamic range queue
- queue length assumed to predict future loads
- no restriction on successively reported costs
- all tables computed simultaneously

**Routing dynamics**

- queue length averaged over a longer time
- dynamic range restricted
- cost also depends on intrinsic link capacity
- restriction on successively reported costs
- attempt to stagger table computation
Hierarchical Routing

- Routing grows more complex with more routers
  - takes more space to store routing tables
  - requires more time to compute routes
  - uses more link bandwidth to update routes
  - more bandwidth wasted on exchanging DVs and LSPs
- Solution:
  - divide the world into several hierarchies
  - Do I really care that router z at foo U just went down?
  - only store info about
    - your local area
    - how to get to higher up routers
  - optimal number of levels for an N router network is ln N
    » requires a total of e ln N entries per router

Key idea

- divide network into a set of domains
- gateways connect domains
- computers within domain unaware of outside computers
- gateways know only about other gateways

Features

- only a few routers in each level
- not a strict hierarchy
- gateways participate in multiple routing protocols
- non-aggregable routers increase core table space

Examples

- Three-level hierarchy in addresses
  - network number
  - subnet number
  - host number
  - Core advertises routes only to networks, not to subnets
    - e.g. 135.104.*, 192.20.225.*
  - Even so, about 80,000 networks in core routers (1996)
  - Gateways talk to backbone to find best next-hop to every other network in the Internet

- If a domain has multiple gateways
  - external records tell hosts in a domain which one to pick to reach a host in an external domain
    - e.g. allows 6.4.0.0 to discover shortest path to 5.* is through 6.0.0.0
  - summary records tell backbone which gateway to use to reach an internal node
    - e.g. allows 5.0.0.0 to discover shortest path to 6.4.0.0 is through 6.0.0.0
  - External and summary records contain distance from gateway to external or internal node
    - unifies distance vector and link state algorithms
### Interior and exterior protocols

- Internet has three levels of routing
  - Highest is at backbone level, connecting autonomous systems (AS)
  - Next level is within AS
  - Lowest is within a LAN
- Protocol between AS gateways: exterior gateway protocol
- Protocol within AS: interior gateway protocol

### Exterior gateway protocol

- Between untrusted routers
  - Mutually suspicious
- Must tell a border gateway who can be trusted and what paths are allowed
- Transit over backdoors is a problem

### Interior protocols

- Much easier to implement
- Typically partition an AS into areas
- Exterior and summary records used between areas

### Issues in interconnection

- May use different schemes (DV vs. LS)
- Cost metrics may differ
- Need to:
  - Convert from one scheme to another (how?)
  - Use the lowest common denominator for costs
  - Manually intervene if necessary

### Routing for Mobility

- Or what happens when computers move?
- Two types of mobility:
  - Migratory: on the net in many locations but not while in motion
  - Roaming: on the net while in motion
- Basic idea:
  - Everyone has a home
  - You spend much of your time near home
  - When not at home, they know where to find you
  - Home agents: know where you are (or that you are missing)
  - Foreign agents: inform home agents of your location
  - Informs users that future communication should be sent via them (this is a huge potential security hole)
Registration

- Foreign Agent broadcasts its existence and address
  - Newly arrived mobile host waits for this broadcast
  - may send “is there a foreign agent listening” message
- Mobile Host registers with foreign agent
- Foreign agent contacts mobile host’s home agent
  - One of your hosts is with me - contains Foreign agent’s network address
- Home agent examines the security information and the timestamp. If satisfied, tells the foreign agent
- When foreign agent gets the ack from home agent it makes an entry in its table and informs mobile host that it is now registered

Packet Routing for Mobile Users

1. Packet is sent to the mobile host's home address
2. Packet is tunneled to the foreign agent
3. Sender is given foreign agent's address
4. Subsequent packets are tunneled to the foreign agent
5. Results in given foreign agent's address

Broadcast Routing

- Sometimes information needs to go to everyone
  - routing updates in link-state
  - stock data, weather data, etc.
- sender iterates over all destinations
  - wastes bandwidth
  - sender must know who is interested
- flooding
  - see routing updates for issues
- multi-destination routing
  - routers support having multiple destinations
  - routers copy output packets to correct link(s)
- spanning tree
  - contains subset of graph with no loops
  - efficient use of bandwidth
  - requires info to be present in routers (but it is for link state)

Routing Broadcast Traffic (cont.)

- Reverse path forwarding
  - check link a packet arrives on
  - if the inbound link is the one the router would use to the source, then
    - forward it out all other links
  - else
    - discard the packet
  - requires no special data sorted in each router

Multicast Routing

- Specify a (relatively) small list of hosts to receive traffic
  - may need to exchange traffic as a group
  - must create/destroy group
- Using spanning trees
  - prune links that have no members of multicast group
  - for distance-vector use a variation on reverse path forwarding
  - when a router gets a message it doesn't need it send a prune message back
  - recursively prunes back un-needed subnets
- core-based trees
  - one tree for group not one per group member
  - hosts send to “core” and it multicasts it out
Congestion

- Too much traffic can destroy performance
  - goal is to permit the network to operate near link capacity
  - can reach a knee in the packets sent vs. delivered curve
- Sources
  - all traffic is destined for a single out link
    - backup in traffic consumes buffers
  - other (cross traffic) will not get through due to lack of buffers
  - slow router CPU
    - can't service all requests at link speed
  - into old traffic
- Often feeds on itself
  - queuing delays can cause packets to timeout
  - introduces more traffic due to re-transmissions

Too Much Traffic

- Maximum carrying capacity of subnet
  - Packets delivered
  - Perfect
  - Desirable
  - Congested
  - Packets sent

Congestion Control

- Two possible approaches
  - open loop: prevent congestion from ever happening
    - tends to be conservative and result in under utilization
  - closed loop: detect and correct
    - some congestion will still occur until it is corrected
- Open loop (Decisions not based on Network State)
  - request resources before using them
  - Discard packets as necessary
  - global (or regional) resource allocation
    - responds yes or no to each request for service
- Closed loop (Decisions based on Network State)
  - monitor network to detect congestion (Packets discarded, Queue Lengths, No of packets which timeout and have to be retransmitted, Avg Delay)
  - pass information back to location where action can be taken
  - adjust system operation to correct the problem

Taxonomy

By Yang and Reddy

- Open Loop
  - At source
  - At Destination
- Closed Loop
  - Explicit Feedback
  - Implicit Feedback

Responding to Congestion

- Add more resources
  - dialup network: start making additional connections
  - SMDS: request additional bandwidth from provider
  - split traffic: use all routes not just optimal
- Decrease load
  - deny service to some users: based on priorities
  - degrade service to some or all users
  - require users to schedule their traffic

Policies that affect congestion

- Transport
  - Retransmission Policy
  - Out-of-order caching Policy
  - Acknowledgement Policy
  - Flow control Policy
  - Timeout Determination
- Network
  - VC versus Datagram in Subnet
  - Packet Queuing and service Policy
  - Packet Discard Policy
  - Routing Algorithm
  - Packet lifetime management
- Data Link
  - Retransmission Policy
  - Out-of-order caching Policy
  - Acknowledgement Policy
  - Flow Control Policy
Traffic Shaping

- Traffic tends to be bursty
  - great variation between min and max bandwidth used
  - this uncertainty leads to inefficient use of the network
- Flow Specification
  - user proposes a specific probability distribution
  - maximum packet size
  - transmission rate (min, max, or mean)
  - maximum delay
  - maximum delay variation (jitter)
  - quality guarantee (how strong is this agreement)
  - network can
    - agree to request
    - refuse it
    - counter offer

Traffic Shaping

- Traffic tends to be bursty
  - great variation between min and max bandwidth used
  - this uncertainty leads to inefficient use of the network

Flow Specification

- user proposes a specific probability distribution
  - maximum packet size
  - transmission rate (min, max, or mean)
  - maximum delay
  - maximum delay variation (jitter)
  - quality guarantee (how strong is this agreement)

Network can

- agree to request
- refuse it
- counter offer

Token Bucket

- Bucket hold tokens (generated one every T seconds)
- Can save up to a fixed limit of n tokens
- When traffic arrives, it must have a token to be sent

Max burst rate
- C: capacity of bucket
- S: burst length in seconds
- M: max output rate
- p: token credit rate
- C + pS = MS

Leaky Bucket

- buffer accepts traffic at link rate
  - buffer has a bounded size (limits burst size that is accepted)
- output is limited to a lower rate
  - traffic is constrained to this rate

Leaky Bucket/Token Bucket Performance

Input

Output

Token Bucket 250KB
Token Bucket 500KB
Token Bucket 750KB
500KB TB into 10MB/sec LB

Congestion Control with Virtual Circuits

- Admission control
  - once traffic reaches a threshold, don’t admit more VCs
  - doesn’t correct current problem, but prevents additional congestion
- Alter routes
  - admit new connections
  - route them around “trouble” areas
- Negotiate traffic
  - establish parameters for volume and shape of traffic

Choke Packets

- Monitor link utilization
  - keep an estimate (u) of average utilization over time
  - \( u_{new} = au_{old} + (1-a)f \)
    - f is a 0/1 sampling of link state
    - a is a parameter to control history
  - can also use queue length or buffer utilization
- When utilization is above a threshold
  - for each new packet to be sent over congested link
    - send “choke” packet back to sender
    - tag forwarded data packet to prevent more choke packets
  - when sender receives choke packet
    - must reduce rate to “choked” destination
- Hop-by-hop choke
  - on path back to sender, each router reduces traffic
  - consumes buffer space along path to sender
  - provides faster relief to congested router/link
Weighted Fair Queuing

- Can give different links different priorities
- Give higher priority length multiple slots per round

Load Shedding

- When all else fails, routers drop (discard) packets
- Policy question: what packets to drop?
  - Oldest ones: they are likely to be useless now
  - Newest ones: helps to close open window in file transfer
  - Less important ones
  - Requires cooperation of application
  - In MPEG frames, are more important than B frames
  - Drop all related packets
  - Fragmentation: loss of one packet renders others useless
  - Requires information from higher levels
- Preemptive shedding
  - When traffic starts to get high, dropping packets can prevent additional congestion

RSVP - Multicast Bandwidth Reservation

- Receivers send request to reserve BW up spanning tree
- Routers propagate request if request up tree
  - Only sent if greater than prev. request for this group
- Dest. can request BW for multiple alternative sources
  - Routers only allocate bandwidth for maximum channel request

Internetworking

- Goal: seamless operation over multiple subnets
  - Could be two similar LANs
  - Link WANs to LANS
- Link two different LANs together
- Issues:
  - Packet size limits (different networks may have different limits)
  - Quality of service (as it is provided, how is it defined)
  - Congestion control
  - Connection vs. connectionless networks
- Possible at many levels
  - Physical layer: repeaters
  - Link layer: bridges - regenerate traffic, some filtering
  - Network: routers - route packets between networks
  - Transport: gateway byte streams
  - Application: gateway email between two different systems

Network Interconnection
**Devices**

- **Repeaters**
  - Low level devices that amplify the signals and repeat them
  - Copy bits as they arrive
- **Bridges**
  - Store and forward devices for frames
- **Multi-protocol Routers**
  - Network layer devices
- **Gateways**
  - Transport
  - Application
- **Half Gateways**

**Gateway Arrangements**

(a) Full gateway
(b) Half gateway
(c) Neutral gateway

**Network Differences**

- **Services Offered**
  - Connection Oriented/connectionless
- **Protocols**
  - IP, IPX, CLNP, Appletalk, DECnet, etc.
- **Addressing**
  - Flat (802), vs. Hierarchical (IP)
- **Multicasting**
  - Present?
- **Packet Size**
  - Maximum?
- **Quality of Service**
  - Present? What kind?
- **Error Handling**
  - Reliable, ordered, or unordered delivery
- **Flow Control**
  - Sliding Window, Rate Controlled, ...
- **Congestion Control**
  - Leaky Bucket, Choke Packets, ...
- **Security**
  - Privacy Rules, encryption, ...
- **Parameters**
  - Timeouts, Flow Specifications, ...
- **Accounting**
  - By connect time, packet, byte, or not at all

**Concatenated Virtual Circuits**

**Connectionless Internetworking**

- Packets travel individually
- Can take different routes

**Tunneling**

- **Problem**
  - Source and Destination are compatible
  - Something in the middle is not compatible
- **Solution**
  - Tunnel through the middle
  - Only multi-protocol routers need to understand conversion
  - Possible to tunnel through almost anything
  - Can tunnel IP through IP (for mobile computing perhaps)
Internetwork Routing

- Autonomous Systems and Gateways
- Interior Gateway Protocol
- Exterior Gateway Protocol

Fragmentation

- Sometimes need to split packets into smaller units
  - limits of the hardware being used
  - operating system buffer constraints
  - protocol limits (max permitted packet is x bytes)
  - reduce channel occupancy (head of line blocking)
- Fragmentation
  - where to split it into smaller packets
  - source (requires end-to-end information on max size)
  - when it reaches boundary
  - how to represent split packets
  - need to encode fragment offset
- Reassembly
  - where to re-combine packets
  - destination (may result in poor performance)
  - at the gateway to the subnet that supports the full size

Fragmentation Example

- Each network may have different maximum packet size
  - Vary from 48 bytes to 65,515 bytes

Internet Structure

Internet Addresses

- Universal Communication Service
  - Any host can communicate with any other host
  - Need to identify each host
- Host Identifiers
  - Name
  - Address
  - Route
  - How to get there
  - Successively lower representations of host identifiers
- Use Binary addresses in TCP/IP
The IP Protocol

- **IP Header**
  - source, destination address, total length
  - version, ihl (header length in 32-bit words), ttl, protocol
  - fragmentation support: identification, df, mf, frag. offset

- **Options**
  - variable length
  - defined options
    - loose source routing
    - timestamp
    - record path

- IP Options
  - Security
  - Strict Source Routing
    - Gives complete path
  - Loose Source Routing
    - Traverse the list of routers in order
  - Record Route
    - Add router address to the header
  - Timestamp
    - Add time stamp as well as router address
Semantics of IP Addresses

- Each address has a network, subnet, and host part
  - for routing only care about network and subnets not hosts
  - what is the network and subnet part varies depending on
    - what the address is (used to be fixed class A, B, C)
    - where the address is viewed from
      - Maryland subnet viewed by world is 128.8.X.X
      - CS Dept. subnet viewed by campus is 128.8.128.X
      - subnets are not visible at higher layers
  - each routing entry is <target, network mask>
  - if multiple matches, one with most 1's in mask wins

- Some special network addresses
  - all 0's -> this host
  - 0's in network address -> host on this network
  - all 1's -> broadcast on this network
  - 127.X.X.X -> loopback (this host)

Classes of IP Addresses

- Each host is assigned 32 bit integer address
- Integers are chosen to make routing efficient
- Encode the network as well as host address
- Five forms of IP Addresses
  - Class A
    - Networks with more than 64K hosts
    - 7 bits to netid and 24 bits to hostid
  - Class B
    - Networks with less than 64K and more than 256 hosts
    - 14 bits for netid and 16 bits for hostid
  - Class C
    - Networks with less than 256 hosts
    - 21 bits for netid and 8 bits for host id
  - Hosts connected to multiple networks require multiple IP addresses

IP Addresses

<table>
<thead>
<tr>
<th>Class</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.0.0.0</td>
<td>All networks</td>
</tr>
<tr>
<td>B</td>
<td>128.0.0.0</td>
<td>All Class B networks</td>
</tr>
<tr>
<td>C</td>
<td>192.0.0.0</td>
<td>All Class C networks</td>
</tr>
<tr>
<td>D</td>
<td>224.0.0.0</td>
<td>Multicast addresses</td>
</tr>
<tr>
<td>E</td>
<td>240.0.0.0</td>
<td>Reserved for future use</td>
</tr>
</tbody>
</table>

0's and 1's

- Field consisting of all 1's
  - means all as in "all hosts"
- Field consisting of all 0's
  - means "this"
  - IP address with host ID 0 means this host
  - IP address with network ID 0 means this network
  - Example
    - Receive a packet with netid 0 and host id - me
    - came from this host

Network Addresses

- IP Addresses can refer to networks as well as hosts
- Host ID 0 is never assigned to a host but is treated as the network address
- Broadcast address
  - Any host ID with all 1's is reserved for broadcast to all hosts in a network
- Limited Broadcast address
  - Local broadcast address
    - 32 1's

Field consisting of all 1's

- Field consisting of all 0's
  - means "this"
  - IP address with host ID 0 means this host
  - IP address with network ID 0 means this network
  - Example
    - Receive a packet with netid 0 and host id - me
    - came from this host
Special Address Conventions

- All 0's: Host
- All 1's: Limited Broadcast
- Net: Directed Broadcast
- Loopback: 127

Internet Addressing

- Advantages
  - Efficient routing
  - Can be used for host or network addresses
  - Contains broadcast addressing scheme

- Weaknesses
  - Addresses refer to network connections and not hosts
    - If a host moves from one network to another, its IP address must change
  - If network grows its class may have to be changed
  - When a host has multiple addresses the path of a packet depends on the address used

IP Addressing

- Written as 128.8.128.179

Dotted Decimal Notation

<table>
<thead>
<tr>
<th>Class</th>
<th>Lowest Address</th>
<th>Highest Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.0.0.0</td>
<td>126.0.0.0</td>
</tr>
<tr>
<td>B</td>
<td>128.0.0.0</td>
<td>191.255.0.0</td>
</tr>
<tr>
<td>C</td>
<td>192.0.0.0</td>
<td>223.255.0.0</td>
</tr>
<tr>
<td>D</td>
<td>224.0.0.0</td>
<td>240.0.0.0</td>
</tr>
<tr>
<td>E</td>
<td>241.0.0.0</td>
<td>247.255.255.255</td>
</tr>
</tbody>
</table>

Network Standard Byte Order

- Integers are sent most significant byte first
  - Big Endian Style

Subnet Addressing

- Permit single network address to span multiple physical networks
**Subnetting**

- Divide the address into
  - network portion
  - local portion
  - local network
  - host
- A form of hierarchical routing
- Subnet addressing is chosen by a network and followed by all host connected to it

**Subnet Mask**

- Site using subnetting uses a subnet mask
  - 1's in mask imply that the network treats the corresponding bits in the IP address as part of the network address
  - 0's imply that those bits are treated as host identifier
- Example
  - 255.255.255.0
  - First three octets identify the network and the last identifies the host

**Goals of New Protocol**

- Support billions of hosts
- Reduce the size of routing tables
- Simplify the protocol - allow faster processing
- Better Security (authentication and privacy)
- Support types of services
- Support Roaming
- Allow the protocol to evolve in the future
- Permit old and new protocol to coexist for years

**IPv6**

- Has longer addresses
  - 16 bytes long
- Simplification of header
  - contains 7 fields (compared to 13 in IPv4)
- Better support for options
  - Easy for routers to skip over the options not relevant for them
- Security
  - Authentication and Privacy features
- Types of service

**IPv6 Fixed Header**

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Priority</td>
<td>4</td>
</tr>
<tr>
<td>Flow Label</td>
<td>20</td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
<tr>
<td>Source addr</td>
<td></td>
</tr>
<tr>
<td>Dest addr</td>
<td></td>
</tr>
</tbody>
</table>

**IPv6 Header**

- Version
  - always 6
- Priority
  - 0 - 7 - can slow down when congestion is detected
  - 1 - new
  - 4 - FTP
  - 6 - Telnet
  - 8 - 15: for real time traffic - send rate is constant even when packets are lost
- Flow Label
  - Supports the source and destination to set up a pseudo-connection with particular properties and requirements
  - Flow designated by (source,destination,flow number)
  - When value is non-zero routers may give required treatment
**IPv6 Header**

- **Payload Length**
  - How many bytes follow the 40-byte header
- **Next Header**
  - Which extension header follows this one
  - If this header is the last one, this field tells which transport protocol handler to pass the packet to
- **Hop Limit**
  - Field decremented for each hop and packet discarded when it becomes zero
- **Source Address**
- **Destination Address**

**IPv6 Address Structure**

- 16 Bytes written as 8 groups of 4 hex digits
- 8000:0000:0000:0000:0123:4567:89AB:CDEF
- Leading zeros can be omitted
- one or more groups of 16 zeros can be replaced by ::
- 8000:123:4567:89AB:CDEF
- IPv4 addresses have 80 leading zeros
- ::128.8.129.75
- Next 16 bits distinguish two variants on how IPv6 packets are tunneled over existing IPv4 network
  - Provider based
  - Geographic based

**IPv6 Addresses**

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000</td>
<td>Reserved (incl IPv4)</td>
</tr>
<tr>
<td>0000 001</td>
<td>OSI NSAP Addresses</td>
</tr>
<tr>
<td>0000 010</td>
<td>Novell Netware IPX addresses</td>
</tr>
<tr>
<td>010</td>
<td>Provider-Based addresses</td>
</tr>
<tr>
<td>100</td>
<td>Geographic Based addresses</td>
</tr>
<tr>
<td>1111 1110 10</td>
<td>Link local use addresses</td>
</tr>
<tr>
<td>1111 1110 11</td>
<td>Site local use addresses</td>
</tr>
<tr>
<td>1111 1111</td>
<td>Multicast</td>
</tr>
</tbody>
</table>

**IPv6 Properties**

- All routers must support 576 byte packets
  - May support large datagrams also
- If a router cannot handle a packet
  - It sends an error message to the sender
    - The sender sends all packets of right size
- **No checksum field**
  - Errors not likely
  - Data link layer has it anyway

**Extension Headers**

<table>
<thead>
<tr>
<th>Extension Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hop-by-hop options</td>
<td>Misc. information for routers</td>
</tr>
<tr>
<td>Routing</td>
<td>Full or partial route to follow</td>
</tr>
<tr>
<td>Fragmentation</td>
<td>Management of fragments</td>
</tr>
<tr>
<td>Authentication</td>
<td>Verification of sender ID</td>
</tr>
<tr>
<td>Encrypted security payload</td>
<td>Info about encrypted contents</td>
</tr>
<tr>
<td>Destination options</td>
<td>Additional info for the destination</td>
</tr>
</tbody>
</table>

**Extension Headers**

- **Format**
  - fixed
  - variable
    - Each item is encoded as (Type, Length, Value)
    - Type is 1 byte field telling which option it is
    - Length is 1 byte field telling how long the value is
    - Value is up to 255 bytes long
**Extension Headers**

- **Hop-by-hop**
  - Every router must examine it
  - One defined so far - for datagrams exceeding 64K

- **Example**
  - Next Header
  - Length of header excluding first 8 bytes
  - Next two bytes (194) indicate that this defines datagram size as a 4 byte number
  - Next 4 bytes give datagram length > 65536

<table>
<thead>
<tr>
<th>Next header</th>
<th>0</th>
<th>194</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jumbo payload length</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Extension Header for Routing**

- List one or more routers which must be visited
  - **Strict**
  - **Loose**

- **Format**
  - Next Header (1 Byte), Routing Type (1 Byte), Number of Addresses (1 Byte, 1 to 24) and the index of next address (1 Byte)
  - Reserved byte and bitmap defining strict or loose for each

<table>
<thead>
<tr>
<th>Next header</th>
<th>0</th>
<th>Number of addresses</th>
<th>Next address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit map</td>
<td>1</td>
<td>2 - 24 Addresses</td>
<td></td>
</tr>
</tbody>
</table>

**Fragment Extension Header**

- **Fragmentation handling**
  - Datagram identifier
  - Fragment number
  - A bit telling if more fragments are to follow

- **Only source host can fragment a packet**
  - Routers along the way can not fragment
  - May send ICMP packet back to source if packet too large

**Other Header**

- **Authentication Header**
  - Mechanisms for authenticate the source

- **Encrypted Security Header**
  - Support the encryption of the contents of a packet
  - Key Assignment - outside IPv6 scope
  - Keys are assigned unique 32 bit key numbers
  - First part - 4 bytes -
    - next header number
    - Length of header
    - 16 zeros
    - 32 bit key number
  - Cryptographic Checksum

**Firewalls**

- A way to limit information flow
  - selective forwarding of information based on policy
  - policy: rules about what should be permitted
  - mechanism: way to enforce policy

- Can be implemented at many levels
  - at higher layers have more information
  - at lower layers can share filtering between multiple higher level entities

- **Possible Layers**
  - link layer: filter based on MAC address
  - network layer: filter based on source/destination, transport
  - transport: filter based on service (e.g. port number)
  - application: filter based on user name in email, based on content

**Internet Control Message Protocol (ICMP)**

- Used to configure and run an IP network
- Just a transport protocol (more or less)

- **Message Types**
  - destination unreachable
  - time exceeded (TTL count reached 0)
  - parameter problem (invalid header)
  - redirect (inform router of possibly bad path)
  - echo request/response (AWA ping packets)
  - timestamp request/response (time stamped pings)

- **Address Resolution Protocol**
  - finding out who owns an IP address on the subnet
  - send link level broadcast with a request
  - response is IP address of destination
Internet Routing

- Use two levels of routing
  - local (subnet) level routing
  - Internet routing between multi-protocol gateways
    - multiple protocol gateways are generally fully connected
    - since they hide the underlying network
    - policies (politics) can dictate acceptable routes
    - don’t route IBM packets of the Microsoft network
    - all packets starting and ending in Canada must stay in Canada
  - Can use any of the standard routing algorithms
    - link-state
    - distance vector

Interior Gateway Routing Protocol

- Routes within a single Autonomous System (AS)
  - An AS contains
    - areas (collection of one or more subnets)
    - backbone (to interconnect areas within AS)
  - Also called Open Shortest Path First (OSPF)
- Divides routers into four classes
  - Internal - only within the area
  - Area boarder routers - connect two or more areas
  - Backbone routers - connect to backbone
  - AS boundary routers - talk to other AS
- Exchanges info between adjacent routers
  - not the same as a neighbor since could have many hops in-between
- Uses link-state
  - flooding with sequence numbers
  - supports multiple metrics: throughput, reliability, delay
  - backbone computes inter-area routes

OSPF

Graph representation of an Autonomous system.

Border Gateway Protocol (BGP)

- Used to route between AS’s
  - concerned with politics and turf battles
  - supports specific policies
  - don’t send my packets of network X
  - don’t send packets through me
- Two types of nodes
  - stub networks (one connection to BGP)
  - multi-connected networks (more than one connection)
    - might also be transit networks (carry traffic for others)
- Uses Distance Vector
  - but includes complete path in table and sent to neighbors
  - uses “scoring” function to select among possible routes