Mobile Ad Hoc Networking

Network Layer Issues

Network Layer Issues and Applications

Contents
1. Network Service Models
2. Routing Principles
   - Link state routing
   - Distance vector routing
   - Hierarchical routing
3. Multicast Routing
4. Peer-to-Peer
5. Internet QoS
Network layer functions

- deliver packets from sending to receiving hosts
- network layer protocols in every host, router

Three important functions:

- path determination: route taken by packets from source to dest. Routing algorithms
- forwarding: move packets from router’s input to appropriate router output
- call setup: some network architectures require router call setup along path before data flows

Network service model

Q: What service model for “channel” transporting packets from sender to receiver?

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

The most important abstraction provided by network layer: virtual circuit or datagram?
Virtual circuits

"source-to-dest path behaves much like telephone circuit"
- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call before data can flow
- each packet carries VC identifier (not destination host ID)
- every router on source-dest path maintains "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be allocated to VC
  - to get circuit-like performance

- used to setup, maintain, teardown VC
- used in ATM, frame-relay, X.25
- not used in today’s Internet
Datagram networks: Internet’s model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of “connection”
- Forwarded: using destination host address
  - packets between same source-dest pair may take different paths

Datagram or VC network: why?

**Asynchronous Transfer Mode**
- ATM (VC)
  - evolved from telephony
  - human conversation:
    - strict timing, reliability requirements
    - need for guaranteed service
  - "dumb" end systems
    - telephones
    - complexity inside network

**Internet ( Datagram)**
- data exchange among computers
  - "elastic" service, no strict timing req.
- "smart" end systems (computers)
  - can adapt, perform control, error recovery
  - simple inside network, complexity at "edge"
- heterogeneous link types
  - different characteristics
  - uniform service difficult

Hard state vs Soft state
Outline

1. Network Service Models
2. Routing Principles
   - Link state routing
   - Distance vector routing
   - Hierarchical routing
3. Multicast Routing
4. Peer-to-Peer
5. Internet QoS

Routing

Routing protocol

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

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

"good" path:
- typically means minimum cost path
- other def's possible
Routing Algorithm classification

Global or decentralized information?

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

Decentralized:
- router knows physically-connected neighbors, link costs to neighbors
- iterative process of computation, exchange of info with neighbors
- "distance vector" algorithms

Static or dynamic?

Static:
- routes change slowly over time

Dynamic:
- routes change more quickly
  - periodic update
  - in response to link cost changes

Link-State Routing Algorithm

Dijkstra’s algorithm
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node (source”) to all other nodes
  - gives routing table for that node

Notation:
- $c(i,j)$: link cost from node $i$ to $j$, cost infinite if not direct neighbors
- $D(v)$: current value of cost of path from source to dest $V$
- $p(v)$: predecessor node along path from source to $v$, that is next $v$
- $N$: set of nodes whose least cost path definitively known
Distance Vector Routing Algorithm

Key Idea
- Given my distance to a neighboring node
- Given the distances from the neighboring nodes to remote nodes
  ➔ My distances to remote nodes

iterative:
- continues until no nodes exchange info.
- self-terminating: no "signal" to stop

asynchronous:
- nodes need not exchange info/iterate in lock step!

distributed:
- each node communicates only with directly-attached neighbors

Distance Vector
- Each node x has its own
- Each element of the vector contains the current estimate of and the next hop to a destination
- Each node in the network corresponds to an element of the vector

DV of node x

<table>
<thead>
<tr>
<th>Dest</th>
<th>Dist</th>
<th>Via</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>w</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>z</td>
<td>3</td>
<td>y</td>
</tr>
</tbody>
</table>

\[ D_X(Y) = \text{Estimated distance from X to Y} \]
\[ = \text{Min}_w \{c(X,w) + D_w(Y)\} \]
**Distance Vector Routing: overview**

- **Iterative, asynchronous:** each local iteration caused by:
  - message from neighbor: its least cost path change from neighbor
- **Distributed:**
  - each node notifies neighbors only when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

**Each node:**

- wait for (msg from neighbor)
- recompute distance table
- if least cost path to any dest has changed, notify neighbors

---

**Distance Vector: Count-to-Infinity Problem**

[Diagram showing network flow and count-to-infinity issue]
Comparison of LS and DV algorithms

Message complexity
- **LS**: with \( n \) nodes, \( E \) links, \( O(nE) \) msgs sent each
- **DV**: exchange between neighbors only
  - convergence time varies

Robustness: what happens if router malfunctions?
- **LS**:
  - node can advertise incorrect link cost
  - each node computes only its own table
- **DV**:
  - DV node can advertise incorrect path cost
  - each node’s table used by others
    - error propagate thru network

Speed of Convergence
- **LS**: \( O(n^2) \) algorithm requires \( O(nE) \) msgs
  - may have oscillations
- **DV**: convergence time varies
  - may be routing loops
  - count-to-infinity problem

IP Addresses

“classful” addressing:

<table>
<thead>
<tr>
<th>Class</th>
<th>Network Mask</th>
<th>Host Bits</th>
<th>Example Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>32</td>
<td>1.0.0.0 to 127.255.255.255</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
<td>24</td>
<td>128.0.0.0 to 191.255.255.255</td>
</tr>
<tr>
<td>C</td>
<td>110</td>
<td>18</td>
<td>192.0.0.0 to 223.255.255.255</td>
</tr>
<tr>
<td>D</td>
<td>1110</td>
<td>32</td>
<td>224.0.0.0 to 239.255.255.255</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Host Bits</th>
<th>32 bits</th>
</tr>
</thead>
</table>

Review: Network Layer 17

Review: Network Layer 18
**IP addressing: CIDR**

- **Classful addressing:**
  - inefficient use of address space, address space exhaustion
  - e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network

- **CIDR: Classless InterDomain Routing**
  - network portion of address of arbitrary length
  - address format (1): `a.b.c.d/x`, where `x` is # bits in network portion of address

```
11001000  00010111 00010000  00000000
```

```
200.23.16.0/23
```

**Routing in the Internet**

- The Global Internet consists of **Autonomous Systems (AS)** interconnected with each other:
  - **Stub AS**: small corporation: one connection to other AS's
  - **Multihomed AS**: large corporation (no transit): multiple connections to other AS's
  - **Transit AS**: provider, hooking many AS's together

- Two-level routing:
  - **Intra-AS**: administrator responsible for choice of routing algorithm within network
  - **Inter-AS**: unique standard for inter-AS routing: BGP
**Internet AS Hierarchy**

- Inter-AS border (exterior gateway) routers
- Intra-AS interior (gateway) routers

**Topology of Tier-1 ISP**
Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

Internet inter-AS routing: BGP

- BGP (Border Gateway Protocol): the de facto standard
- Path Vector protocol:
  - similar to Distance Vector protocol
  - each Border Gateway broadcast to neighbors (peers) entire path (i.e., sequence of AS's) to destination
  - BGP routes to networks (ASs), not individual hosts
  - E.g., Gateway X may send its path to dest. Z:

\[ \text{Path (X,Z)} = X,Y_1,Y_2,Y_3,...,Z \]
**BGP operation**

Q: What does a BGP router (gateway) do?
- Receiving and filtering route advertisements from directly attached neighbor(s).
- Route selection.
  - To route to destination X, which path (of several advertised) will be taken?
- Sending route advertisements to neighbors.

---

**Why different Intra-/Inter-AS routing?**
BGP: controlling who routes to you

- A, B, C are provider networks
- X, W, Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C

A advertises to B the path AW
B advertises to X the path BAW
Should B advertise to C the path BAW?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C is B's customer
  - B wants to force C to route to W via A
  - B wants to route only to/from its customers!

Outline

1. Network Service Models
2. Routing Principles
   - Link state routing
   - Distance vector routing
   - Hierarchical routing
3. Multicast Routing
4. Peer-to-Peer
5. Internet QoS

Multicast: one sender to many receivers

- **Multicast**: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- **Question**: how to achieve multicast

**Multicast via unicast**
- source sends N unicast datagrams, one addressed to each of N receivers

routers forward unicast datagrams
multicast receiver (red)
not a multicast receiver (red)
Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast

Multicast via unicast:
Problems
1. Inefficient
2. Addressing

Multicast: one sender to many receivers

- Multicast: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- Question: how to achieve multicast

Network multicast
- Router actively participate in multicast, making copies of packets as needed and forwarding towards multicast receivers
**Multicast: one sender to many receivers**

- **Multicast**: act of sending datagram to multiple receivers with single "transmit" operation
  - analogy: one teacher to many students
- **Question**: how to achieve multicast

**Application-layer multicast**
- end systems involved in multicast copy and forward unicast datagrams among themselves

---

**Internet Multicast Service Model**

- **multicast group concept**: use of *indirection*
  - hosts addresses IP datagram to multicast group
  - routers forward multicast datagrams to hosts that have "joined" that multicast group
  - Many-to-many communications
**Multicast groups**

- Class D Internet addresses reserved for multicast:
  
<table>
<thead>
<tr>
<th>1110</th>
<th>Multicast Group ID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>28 bits</td>
</tr>
</tbody>
</table>

- Host group semantics:
  - Anyone can "join" (receive) multicast group
  - Anyone can send to multicast group
  - No network-layer identification to hosts of members

- Needed: infrastructure to deliver mcast-addressed datagrams to all hosts that have joined that multicast group

**Joining a mcast group: two-step process**

- **Local:** Host informs local mcast router of desire to join group: IGMP (Internet Group Management Protocol)
- **Wide area:** Local router interacts with other routers to receive mcast datagram flow
  - Many protocols (e.g., DVMRP, MOSPF, PIM)
**IGMP: Internet Group Management Protocol**

- **host**: sends IGMP report when application joins mcast group
  - IP_ADD_MEMBERSHIP socket option
  - host need not explicitly "unjoin" group when leaving
- **router**: sends IGMP query at regular intervals
  - host belonging to a mcast group must reply to query

---

**IGMP**

**IGMP version 1**

- **router**: Host Membership Query msg broadcast on LAN to all hosts
- **host**: Host Membership Report msg to indicate group membership
  - randomized delay before responding
  - implicit leave via no reply to Query
- RFC 1112

**IGMP v2**: additions include

- Leave Group msg
  - last host replying to Query can send explicit Leave Group msg
  - router performs group-specific query to see if any hosts left in group
- RFC 2236

**IGMP v3**: under development as Internet draft

---

Multicast Routing: Problem Statement

- **Goal**: find a tree (or trees) connecting routers having local mcast group members
  - **tree**: not all paths between routers used
  - **source-based**: different tree from each sender to rcvrs
  - **shared-tree**: same tree used by all group members (senders)

![Shared tree](image1)

![Source-based trees](image2)

Approaches for building mcast trees

**Approaches**:
- **source-based tree**: one tree per source
  - shortest path trees
  - reverse path forwarding
- **group-shared tree**: group uses one tree
  - minimal spanning (Steiner)
  - center-based trees

...we first look at basic approaches, then specific protocols adopting these approaches
Source based Tree:
**Shortest Path Tree**

- Mcast forwarding tree: tree of shortest path routes from source to all receivers
  - Dijkstra’s algorithm

**LEGEND**
- Router with attached group member
- Router with no attached group member
- Link used for forwarding, \( i \) indicates order link added by algorithm

Source based Tree:
**Flooding**

- Problem: Broadcast storm
Source based Tree:
Reverse Path Forwarding

- rely on router’s knowledge of unicast shortest path from it to sender
- each router has simple forwarding behavior:

```c
if (mcast datagram received on incoming link on shortest path back to center)
    then flood datagram onto all outgoing links
else ignore datagram
```

Reverse Path Forwarding: example

- result is a source-specific reverse spanning tree
  - may be a bad choice with asymmetric links
Reverse Path Forwarding: pruning

- forwarding tree contains subtrees with no mcast group members
  - no need to forward datagrams down subtree
  - “prune” msgs sent upstream by router with no downstream group members

Legend:
- router with attached group member
- router with no attached group member
- prune message
- links with multicast forwarding

Reverse Path Forwarding: Multiple trees for multi-sender
Shared-Tree: General Problem

- **Minimum Spanning Tree**: minimum cost tree connecting all routers with attached group members
  - Algorithms?
- **Steiner Tree**: minimum cost tree connecting a set of routers, which includes all that with attached group members
**Spanning Tree vs Steiner Tree**

- **Minimum Spanning Tree**: minimum cost tree connecting all routers with attached group members
  - Prim, Kruskal algorithms
- **Minimum Steiner Tree**: minimum cost tree connecting a set of routers, which includes all that with attached group members
  - Problem is NP-complete
  - Excellent heuristics exist

Internet Multicasting Routing: DVMRP

- **DVMRP**: distance vector multicast routing protocol, RFC1075
- **flood and prune**: reverse path forwarding, source-based tree
  - RPF tree based on DVMRP’s own routing tables constructed by communicating DVMRP routers
  - no assumptions about underlying unicast
  - initial datagram to mcast group flooded everywhere via RPF
  - routers not wanting group: send upstream prune msgs

**Facts**
- commonly implemented in commercial routers
- Mbone routing done using DVMRP

PIM: Protocol Independent Multicast

- not dependent on any specific underlying unicast routing algorithm (works with all)
- two different multicast distribution scenarios:

**Dense:**
- group members densely packed, in “close” proximity.
- bandwidth more plentiful

**Sparse:**
- # networks with group members small wrt # interconnected networks
- group members “widely dispersed”
- bandwidth not plentiful
### Tunneling

**Q:** How to connect “islands” of multicast routers in a “sea” of unicast routers?

- mcast datagram encapsulated inside “normal” (non-multicast-addressed) datagram
- normal IP datagram sent thru “tunnel” via regular IP unicast to receiving mcast router
- receiving mcast router decapsulates to get mcast datagram

### Other issues

- Inter-AS multicast routing?
  - No standard, but DVMRP often used
- Any link-state based multicast protocol?
  - Yes, MOSPF
Outline

1. Network Layer Service Models
2. Routing Principles
   - Link state routing
   - Distance vector routing
   - Hierarchical routing
3. Multicast Routing
4. Peer-to-Peer
5. Internet QoS

Peer-to-Peer Paradigm

A peer is both client and server

Recall Client/Server paradigm

Client:
- initiates contact with server ("speaks first")
- typically requests service from server,
- Web: client implemented in browser; e-mail: in mail reader

Server:
- provides requested service to client
- e.g., Web server sends requested Web page, mail server delivers e-mail
Peer-to-Peer Communications

**Example**
- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for “Network love.mp3”
- Application displays other peers that have copy of “Network love.mp3”.
- Alice chooses one of the peers, Bob.
- File is copied from Bob’s PC to Alice’s notebook: HTTP
- While Alice downloads, other users uploading from Alice.
- Alice’s peer is both a Web client and a transient Web server.
  
  All peers are servers = highly scalable!

---

P2P: centralized directory

For file sharing

**Key issue:** File searching

- which peer has the file?

original “Napster” design

1) when peer connects, it informs central server:
   - IP address
   - content

2) Alice queries for “Network love.mp3”

3) Alice requests file from Bob

---

Review: Network Layer 57

Review: Network Layer 58
P2P: problems with centralized directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

File transfer is decentralized, but locating content is highly decentralized.

P2P: decentralized directory

- Each peer is either a group leader or assigned to a group leader.
- Group leader tracks the content in all its children.
- Peer queries group leader; group leader may query other group leaders.
More about decentralized directory

**advantages of approach**
- no centralized directory server
  - location service distributed over peers
  - more difficult to shut down

**disadvantages of approach**
- bootstrap node needed
- group leaders can get overloaded

---

**P2P: Query flooding**

- Gnutella
- no hierarchy
- use bootstrap node to learn about others
- join message

- Send query to neighbors
- Neighbors forward query
- If queried peer has object, it sends message back to querying peer

![Diagram of P2P network with join message]
P2P: more on query flooding

Pros
- peers have similar responsibilities: no group leaders
- highly decentralized
- no peer maintains directory info

Cons
- excessive query traffic
- query radius: may not have content in scope
- bootstrap node
- maintenance of overlay network

Aside: Search Time?
Aside: All Peers Equal?

1.5Mbps DSL

1.5Mbps DSL

10Mbps LAN

DHT: A New Story...

- In 2000-2001, academic researchers said “we want to play too!”
- Motivation:
  - Frustrated by popularity of all these “half-baked” P2P apps
  - We can do better!
  - Guaranteed lookup success for files in system
  - Provable bounds on search time
  - Provable scalability to millions of node
- Hot Topic in networking ever since
  - No practical use so far (?)
P2P: Content Addressing (Hash Routing)

Hash routing

- Given an object identifier I, calculate its hash value \( H=\text{hash}(I) \), and (hopefully) find it (or its location info) in peer \( H \)
- Not a new idea:
  - Load balancing - hash IP address, re-direct to different servers

```
application
put(key, data)  get (key)  data
hash table
node  node  ....  node
```

Distributed Hash Table (DHT)

Challenges

- For each object, node(s) whose range(s) cover that object must be reachable via a "short" path
- # neighbors for each node should scale well (e.g., should not be \( O(N) \))
- Fully distributed (no centralized bottleneck/single point of failure)
- DHT mechanism should gracefully handle nodes joining/leaving
  - need to repartition the range space over existing nodes
  - need to reorganize neighbor set
  - need bootstrap mechanism to connect new nodes into the existing DHT infrastructure
Case Studies

- Structured overlay (p2p) systems
  - Chord
  - CAN (Content Addressable Network)
- Key Questions
  - Q1: How is hash space divided "evenly" among existing nodes?
  - Q2: How is routing implemented that connects an arbitrary node to the node responsible for a given object?
  - Q3: How is the hash space repartitioned when nodes join/leave?
- Let \( N \) be the number of nodes in the overlay
- Let \( H \) be the size of the range of the hash function (when applicable)

Chord

- Associate to each node and file a unique id in an unidimensional space (a Ring)
  - E.g., pick from the range \([0...2^m]\)
  - Usually the hash of the file or IP address
- Properties:
  - Routing table size is \( O(\log N) \), where \( N \) is the total number of nodes
  - Guarantees that a file is found in \( O(\log N) \) hops

from MIT in 2001
Consistent Hashing

A key is stored at its successor: node with next higher ID

Chord Basic Lookup

"Where is key 80?"

"N90 has K80"
Entry $i$ in the finger table of node $n$ is the first node that succeeds or equals $n + 2^i$.

In other words, the $i$th finger points $1/2^{n-i}$ way around the ring.

**Chord Summary**

- Routing table size?
  - $\log N$ fingers

- Routing time?
  - Each hop expects to $1/2$ the distance to the desired id $\Rightarrow$ expect $O(\log N)$ hops.
**CAN (Content Addressable Network)**

Hyper-cube space
- hash value is viewed as a point in a D-dimensional Cartesian space
- each node responsible for a D-dimensional “cube” in the space
- The space is covered by all the nodes
- Example:
  - Initial node n1:(1, 2)

**CAN Illustration: 2-D**

- node n2:(4, 2) joins
node n3:(3, 5) joins?

node n4:(5, 5) and n5:(6,6) join
CAN Illustration: 2-D

- nodes: n1:(1, 2); n2:(4, 2); n3:(3, 5); n4:(5, 5); n5:(6, 6)
- Data (key): f1:(2, 3); f2:(5, 0); f3:(2, 1); f4:(7, 5)

CAN Illustration: 2-D

- Association
**CAN Illustration: 2-D**

- A node knows its neighbors
- Forward query to the neighbor that is nearest to key
- Example: n1 want to find f4
- Multiple paths → reliable

![CAN Illustration](image1)

---

**DHT: Discussion**

- **Pros:**
  - Guaranteed Lookup
  - $O(\log N)$ per node state and search scope
- **Cons:**
  - No one uses them?
  - Supporting non-exact match search is hard
  - Topology-awareness?
Outline

1. Network Layer Service Models
2. Routing Principles
   - Link state routing
   - Distance vector routing
   - Hierarchical routing
3. Multicast Routing
4. Peer-to-Peer
5. Internet QoS
   - Scheduling and policing
   - IntServ/DiffServ

Beyond Best Effort

Internet service model

Best-effort (or least-effort)
   - Guarantee only one thing (sometimes even cannot)
     -- delivery a packet to destination
Thus far: "making the best of best effort"
   - TCP
   - Multicast
   - Proxy filtering/caching
   - Content distribution (replication)
   - P2P ...

But, many things cannot be guaranteed in transport/application layer if network layer does not guarantee them
   - Delay, bandwidth - important to media applications
Improving QOS in IP Networks

Future: next generation Internet with QoS guarantees
- RSVP: signaling for resource reservations
- Differentiated Services: differential guarantees
- Integrated Services: firm guarantees

- simple model for sharing and congestion studies:

![Network Diagram](image)

Principles for QOS Guarantees

- Example: 1Mbps IP phone, FTP share 1.5 Mbps link.
  - bursts of FTP can congest router, cause audio loss
  - want to give priority to audio over FTP

![Network Diagram](image)

Principle 1: packet marking needed for router to distinguish between different classes; and new router policy to treat packets accordingly
Principles for QoS Guarantees (more)

- what if applications misbehave (audio sends higher than declared rate)
  - policing: force source adherence to bandwidth allocations
- marking and policing at network edge:
  - similar to ATM UNI (User Network Interface)

Principle 2
provide protection (isolation) for one class from others

Principles for QoS Guarantees (more)

- Allocating fixed (non-sharable) bandwidth to flow: inefficient use of bandwidth if flows doesn’t use its allocation

Principle 3
While providing isolation, it is desirable to use resources as efficiently as possible
**Principles for QoS Guarantees (more)**

- Basic fact of life: can not support traffic demands beyond link capacity

**Principle 4**

Call Admission: flow declares its needs, network may block call (e.g., busy signal) if it cannot meet needs

---

**Summary of QoS Principles**

QoS for networked applications

- Packet classification
- Scheduling and policing
- High resource utilization
- Call admission

Let's next look at mechanisms for achieving this ....
**Scheduling Policies (1)**

- **scheduling**: choose next packet to send on link
- **FIFO (first in first out) scheduling**: send in order of arrival to queue
  - **discard policy**: if packet arrives to full queue: who to discard?
    - Tail drop: drop arriving packet
    - Priority: drop/remove on priority basis
    - Random: drop/remove randomly

![Scheduling Diagram](image1)

**Scheduling Policies (2)**

- **Priority scheduling**: transmit highest priority queued packet
- **multiple classes, with different priorities**
  - Class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc..
Scheduling Policies (3)

round robin scheduling:
- multiple classes
- cyclically scan class queues, serving one from each class (if available)

 Weighted round robin

- Normalize the weights so that they become integer

Normalize

\[ W_A = 1.4 \quad W_B = 0.2 \quad W_C = 0.8 \]
\[ W_A = 7 \quad W_B = 1 \quad W_C = 4 \]

round length = 13
**Scheduling Policies (4)**

**Generalized Processor Sharing**
- **Round-robin, sounds good**
  - Is it fair? Only if packets are of same size
  - Does it guarantee bandwidth (in a small time scale)?
- **Fairness, protection.**
  - GPS can provide fairness and protection
  - Visit each non-empty queue in turn, serve infinitesimal data (1 bit)
  - GPS is not implementable; we can serve only packets

![Diagram](image.png)

**Weighted Fair Queueing (WFQ)**
- Deals better with variable size packets and weights
- Also known as packet-by-packet GPS (PGPS)
- Find finish time of a packet, had we been doing GPS; serve packets in order of their finish times
Policing Mechanisms

**Token Bucket**: limit input to specified Burst Size and Average Rate.

- bucket can hold $b$ tokens
- tokens generated at rate $r$ token/sec unless bucket full
- over interval of length $t$: number of packets admitted less than or equal to $(r \cdot t + b)$.

Policing Mechanisms (more)

- token bucket, WFQ combine to provide guaranteed upper bound on delay, i.e., QoS guarantee!

$$D_{\text{max}} = \frac{b}{R}$$
**IETF Integrated Services**

- architecture for providing QoS guarantees in IP networks for individual application sessions
- resource reservation: routers maintain state info of allocated resources, QoS req’s
- admit/deny new call setup requests:

**Question:** can newly arriving flow be admitted with performance guarantees while not violated QoS guarantees made to already admitted flows?

---

**Intserv: QoS guarantee scenario**

- **Resource reservation**
  - call setup, signaling (RSVP)
  - traffic, QoS declaration
  - per-element admission control
  - QoS-sensitive scheduling (e.g., WFQ)
Call Admission

Arriving session must:
- declare its QoS requirement
  - R-spec: defines the QoS being requested
- characterize traffic it will send into network
  - T-spec: defines traffic characteristics
- signaling protocol: needed to carry R-spec and T-spec to routers (where reservation is required)
  - RSVP

Intserv QoS: Service models [rfc2211, rfc 2212]

Guaranteed service:
- worst case traffic arrival: leaky-bucket-policed source
- simple (mathematically provable) bound on delay [Parekh 1992, Cruz 1988]

Controlled load service:
- "a quality of service closely approximating the QoS that same flow would receive from an unloaded network element."

\[ D_{max} = \frac{b}{R} \]
**Signaling in the Internet**

connectionless (stateless) forwarding by IP routers + best effort service = no network signaling protocols in initial IP design

- New requirement: reserve resources along end-to-end path (end system, routers) for QoS for multimedia applications
- RSVP: Resource Reservation Protocol [RFC 2205]
  - "...allow users to communicate requirements to network in robust and efficient way." i.e., signaling!
- earlier Internet Signaling protocol: ST-II [RFC 1819]

---

**RSVP Design Goals**

1. accommodate heterogeneous receivers (different bandwidth along paths)
2. accommodate different applications with different resource requirements
3. make multicast a first class service, with adaptation to multicast group membership
4. leverage existing multicast/unicast routing, with adaptation to changes in underlying unicast, multicast routes
5. control protocol overhead to grow (at worst) linear in # receivers
6. modular design for heterogeneous underlying technologies
RSVP: does not...

- specify how resources are to be reserved
  - rather: a mechanism for communicating needs
- determine routes packets will take
  - that's the job of routing protocols
  - signaling decoupled from routing
- interact with forwarding of packets
  - separation of control (signaling) and data (forwarding) planes

IETF Differentiated Services

Concerns with Intserv:
- Scalability: signaling, maintaining per-flow router state
difficult with large number of flows
- Flexible Service Models: Intserv has only two classes. Also want "qualitative" service classes
  - "behaves like a wire"
  - relative service distinction: Platinum, Gold, Silver

DiffServ approach:
- simple functions in network core, relatively complex
  functions at edge routers (or hosts)
- define service classes, provide functional components to build service classes
**Diffserv Architecture**

**Edge router:**
- per-flow traffic management
- marks packets as in-profile and out-profile

**Core router:**
- per class traffic management
- buffering and scheduling based on marking at edge
- preference given to in-profile packets
- Assured Forwarding

---

**Edge-router Packet Marking**

- **profile:** pre-negotiated rate $A$, bucket size $B$
- **packet marking at edge based on per-flow profile**

Possible usage of marking:
- class-based marking: packets of different classes marked differently
- intra-class marking: conforming portion of flow marked differently than non-conforming one
Classification and Conditioning

- Packet is marked in the Type of Service (TOS) in IPv4, and Traffic Class in IPv6
- 6 bits used for Differentiated Service Code Point (DSCP) and determine PHB that the packet will receive
- 2 bits are currently unused

![Diagram of packet classification and conditioning]

Classification and Conditioning

may be desirable to limit traffic injection rate of some class:
- user declares traffic profile (e.g., rate, burst size)
- traffic metered, shaped if non-conforming

![Diagram of network layer classification and conditioning]
Randomization in Router Queue Management

- normally, packets dropped only when queue overflows
  - "Drop-tail" queueing

The case against drop-tail queue management

- large queues in routers are "a bad thing"
  - End-to-end latency dominated by length of queues at switches in network

- allowing queues to overflow is "a bad thing"
  - connections transmitting at high rates can starve connections transmitting at low rates
  - connections can synchronize their response to congestion
Idea: early random packet drop

- When queue length exceeds threshold, packets dropped with fixed probability
  - probabilistic packet drop: flows see same loss rate
  - problem: bursty traffic (burst arrives when queue is near full) can be over-penalized

Random early detection (RED) packet drop

- use exponential average of queue length to determine when to drop
  - avoid overly penalizing short-term bursts
  - React to longer term trends
- tie drop prob. to weighted avg. queue length
  - avoids over-reaction to mild overload conditions
Random early detection (RED) packet drop

RED: why probabilistic drop?

- provide gentle transition from no-drop to all-drop
  - provide "gentle" early warning
- provide same loss rate to all sessions:
  - with tail-drop, low-sending-rate sessions can be completely starved
- avoid synchronized loss bursts among sources
  - avoid cycles of large-loss followed by no-transmission
- WRED (Cisco)