Quality of Service

Acute Communication Corp.
Edward Jin-Ru Chen
jzchen@acutecomm.com.tw

Contents

- Introduction
- QoS Mechanisms
- QoS Architecture

Definitions

- ❒ **What is Quality of Service (QoS)?**
  - ➔describes absolute, quantifiable level of network behavior (e. g. 5Mbps of bandwidth, 100ms delay)
  - ➔per- application, per- connection, per- flow granularity
- ❒ **What is Class of Service (COS)?**
  - ➔sort all traffic (flows) into a few groups
  - ➔treat groups with relative level of service
  - ➔per- group/ class granularity

QoS Examples

- ❒ **Examples**
  - ➔IP network - RSVP/Integrated service (IntServ)
  - ➔ATM network
- ❒ **Characteristics**
  - ➔"Hard " guaranteed at
    - RSVP/IntServ provides per-flow guarantee
    - ATM provides per-VC guarantee

Term Definitions

- ❐ **Flow**
  - ➔ IntServ: distinguishable stream of related datagrams that results from a single user activity and requires the same QoS
  - ➔ RSVP: Destination IP and Protocol
  - ➔ DiffServ: Source IP, Destination IP, Source port, Destination port, Protocol
- ❐ **VC/VP**
  - ➔ Virtual Circuit/ Virtual Path
  - ➔ Decided after the signaling process
  - ➔ Tag for switching
  - ➔ Address is used for signaling and forwarding path establish

COS Examples

- ❐ **Examples**
  - ➔ IP TOS
  - ➔ IP DiffServ
- ❐ **Characteristics**
  - ➔ "Soft" guarantee at per-class base

QoS Mechanisms

- ❒ **Traffic Conditioner**
  - ➔Classifier
  - ➔Dropper/Shaper
- ❒ **Queue Management**
- ❒ **Scheduler**

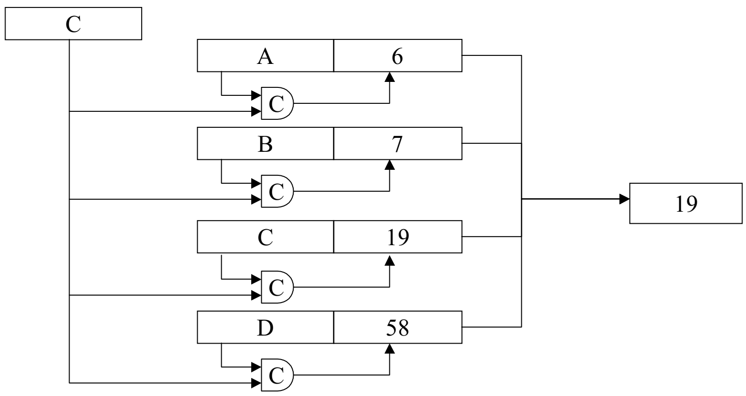
Classifier

- ❒ **Determine the class/flow of packet based on the class/flow definition**
- ❒ **Mechanisms for classifier**
  - ➔Linear search and caching
  - ➔Content addressable memory
  - ➔Tree search
  - ➔Hashing function

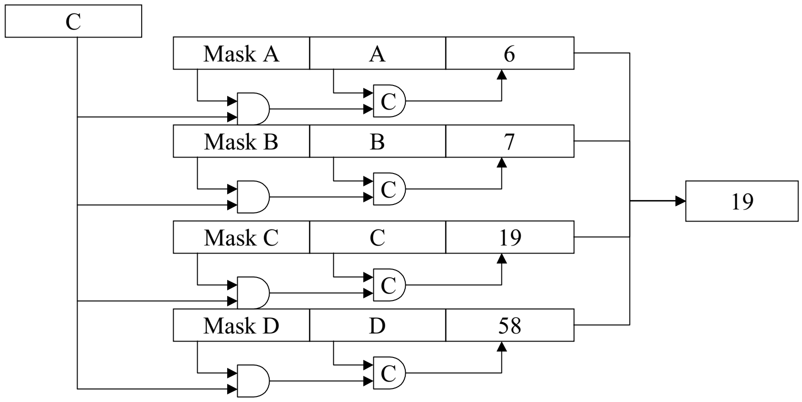
Content Addressable Memory

- ❒ **Pure hardware solution**
- ❒ **Parallel search all stored entries**
- ❒ **Advanced CAM can provide mask for search**
  - ➔Priority setting for conflict resolution

What is CAM?



What is Ternary CAM?



Ternary CAM

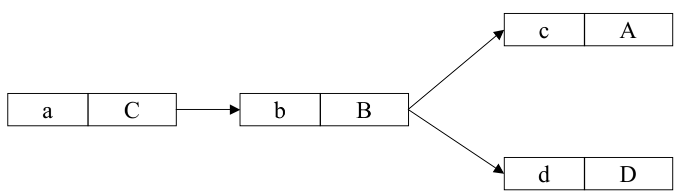
- ❒ **Store words with three-valued digits**
  - ➔0, 1, ‘x’ (wildcard)
- ❒ **Stored position represents the priority**
  - ➔Multiple-match may occurred at ternary CAM
  - ➔Priority to resolve the multiple matching problem

Tree Search

- ❒ **Based on each prefix to construct a searching tree**
- ❒ **Go through the tree and end with the search result**
- ❒ **Improvements**
  - ➔Extend tree to array for speed-up
  - ➔Compress tree to reduce space requirement
- ❒ **Facing multiple path selection problem when apply to layer4 lookup**

Tree Search Example

a.b.c.x	A
a.b.x.x	B
a.x.x.x	C
a.b.d.x	D



Hashing Function

- **Transfer search key into hashing result**
  - ➔ $H(X) = Y$
  - ➔ X may be MAC address, IP address or anything required
  - ➔ X has more bits than Y
- **Using the hashing result to identify the record position**
- **Store the search key for collision checking**
- **Fast but collision may occur**

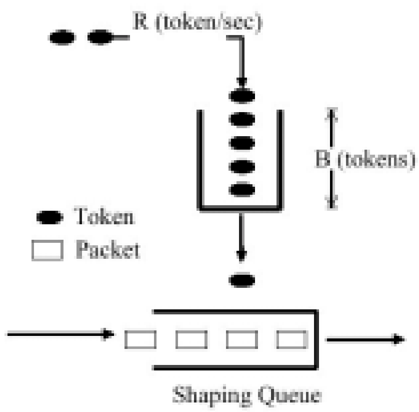
Classes of Hashing Function

- **Assumes: key *j* bits, hashing result *i* bits**
- **Bit extraction hashing functions**
  - ➔ Selecting *j* bits out of the *i* bits
- **Hashing Functions from XOR method**
  - ➔ Such as $h(x) = (x_1 \text{ xor } x_5)(x_2 \text{ xor } x_6)(x_3 \text{ xor } x_7)(x_4 \text{ xor } x_8)$
- **Hashing functions from the class H_3**
  - ➔ Based on Q, which is i x j Boolean matrices
  - ➔ Hashing result = Q X Key
  - ➔ $H_q(x) = x(1)q(1) \text{ xor } x(2)q(2) \text{ xor } \ldots x(i)q(i)$

Policing/Shaping

- ❒ **Based on token-bucket or leaky-bucket**
- ❒ **Meter measures the traffic characteristics**
- ❒ **Policing checks conformance to a configured (or signaled) traffic profile**
  - ➔Data passed when in-profile
  - ➔Dropping or shaping when out-of-profile

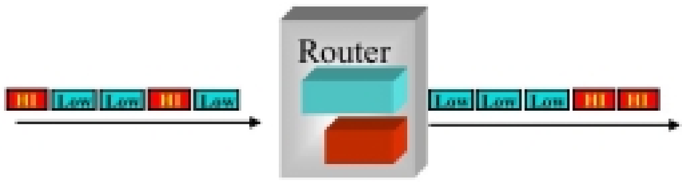
Leaky Bucket



Leaky Bucket Parameters

- ❒ **Parameters**
  - ➔Token arrival rate : Allowed average packet arrival rate
  - ➔Token queue size : Allowed burst size
  - ➔Packet queue size : Packet number allowed for shaping
- ❒ **Dropper**
  - ➔Zero packet queue size
- ❒ **Shaper**
  - ➔Non-zero packet queue size

Queue Scheduling



- ❒ **Traditional FIFO queuing provides no service differentiation and can lead to network performance problems**
- ❒ **IP QoS requires routers to support (some form of) queue scheduling and management to prioritize outbound packets and control queue depth (minimize congestion)**

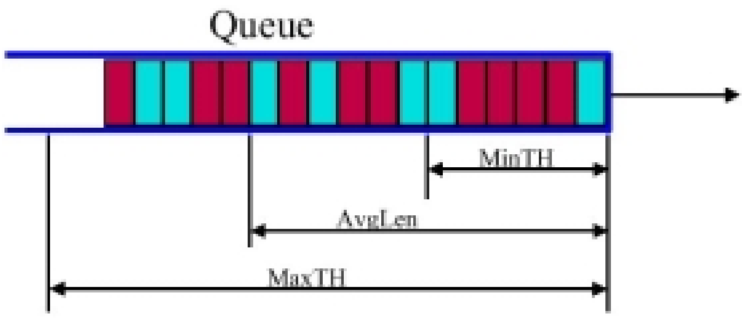
Why Queue Management?

- ❒ **Full Queues are problematic**
  - ➔New connections cannot get thru (called Lock-Out)
  - ➔All packets from existing flows are dropped resulting in across-the-board TCP slow-starts (called Global Synchronization)
  - ➔Can't handle bursts of traffic
- ❒ **RED (random early detection) is a queue management solution**

Queue Management Scheme

- ❒ **RED**
  - ➔Random Early Detection
  - ➔Drop packet before queue is full
  - ➔Derived from TCP flow control characteristic
- ❒ **RIO**
  - ➔RED with in-profile/out-of-profile awareness
- ❒ **TSW**
  - ➔Time-sliding window
  - ➔Provide a smooth estimate of the TCP sending rate

Random Early Detection

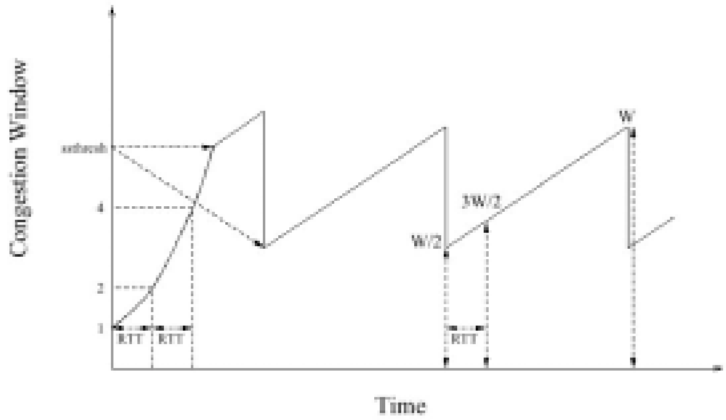


- ❐ Dropping decision based on Average Queue Length

TCP Flow Control

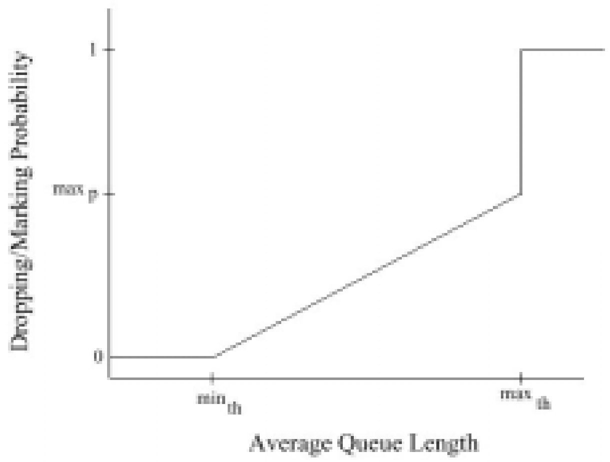
- ❐ Sliding window flow control
- ❐ Accumulated acknowledgement
- ❐ Duplicate ACK to indicate packet loss
- ❐ Packet loss represents the occurrence of congestion
- ❐ Reduce the window size when congestion is detected

TCP Flow Control Figure



Edward Jin-Ru Chen QoS 25

Rough Description



Edward Jin-Ru Chen QoS 26

General Algorithm

For each packet arrival

Calculate the average queue size *avg*

If $min_{th} <= avg < max_{th}$

calculate probability P_a

with probability P_a

mark the arriving packet

Else if $max_{th} <= avg$

mark the arriving packet

Pa Calculation

P_b <- $max_p(avg - min_{th})/(max_{th} - min_{th})$

P_b <- P_b PacketSize/ MaximumPacketSize

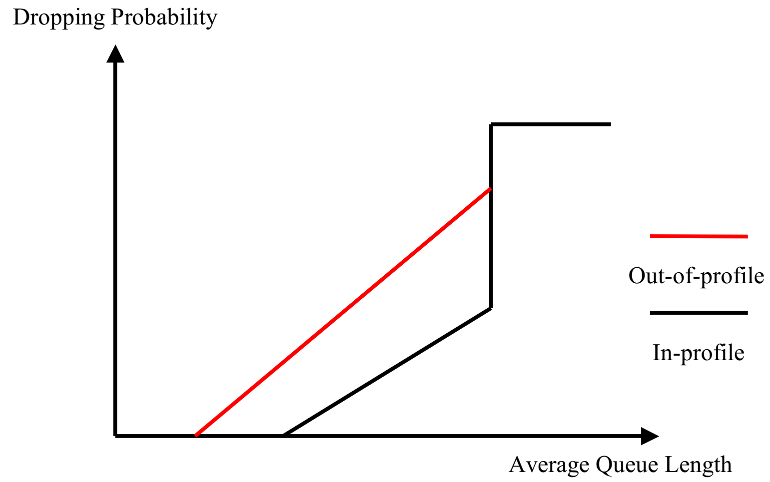
P_a <- $P_b/(1 - count \cdot P_b)$

Queue Length Calculation

- **Low-pass filter is an exponential weighted moving average (EWMA)**
  - ➔ $avg = (1 - w_q)avg + w_q q$
  - ➔ If w_q too large unable to filter out transient congestion
  - ➔ If w_q too small, *avg* is not a reasonable reflection of the current average queue size

Edward Jin-Ru Chen QoS 29

RIO Algorithm



Edward Jin-Ru Chen QoS 30

Time-sliding window

- ❒ Rate estimator and tagging packet as *out* when estimated rate larger than threshold
- ❒ Initially
  - ➔Win_length = a constant
  - ➔Avg_rate = connection's target rate
  - ➔T_front = 0
- ❒ Upon each packet arrival
  - ➔Bytes_in_TSW = Avg_rate * Win_length
  - ➔New_bytes = Bytes_in_TSW + pkt_size
  - ➔Avg_rate = New_bytes/ (now – T_front + Win_length)
  - ➔T_front = now

Scheduler

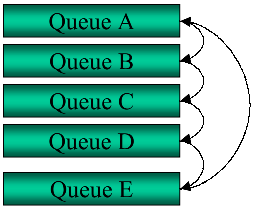
- ❒ Weighted Round Robin
- ❒ Deficit Round Robin
- ❒ Weighted Fair Queuing
- ❒ Class Based Queue

Round Robin Scheduling

❒ **Round Robin**

➔A server process circularly and repeatedly visits a number of clients and performs one job for each of them that has such a need at the time of the visit

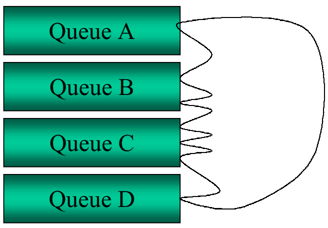
➔Fair to each client with identical packet size



Weighted Round Robin

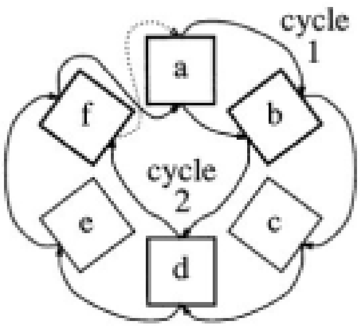
❒ **Provide different service instead of fair service**

❒ **Some queues are serviced more than one time in a cycle**



Weighted Round Robin

- ❐ **Make the visit more smooth among clients**



Fair Queuing

- ❐ **Objectives**
  - ➔fair access to bandwidth and resources in routers
  - ➔no one flow shall receive more than its fair share
- ❐ **Assume queues are serviced in a bit-by-bit round robin (BR) fashion**
  - ➔transmit one bit from each queue
  - ➔But one cannot interleave bits from different queues

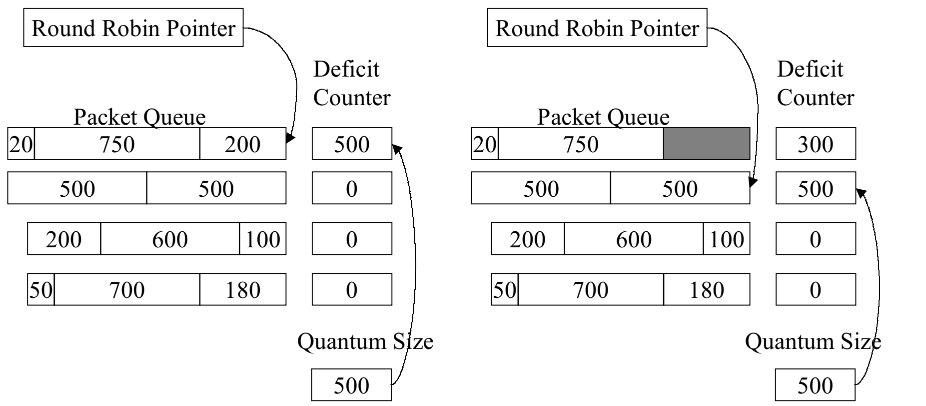
Weighted Fair Queue

- ❒ **Provide more than fair-share of the bandwidth**
- ❒ **WFQ is work-conserving**
  - ➔ Router will always transmit packets if they are present in the queue – link is never idle
- ❒ **What is non-work-conserving?**
  - ➔ Even the queue has packet to send when the output port is idle, packets may stay at the queue instead of transmission
  - ➔ For delay and jitter requirement

WFQ Algorithms

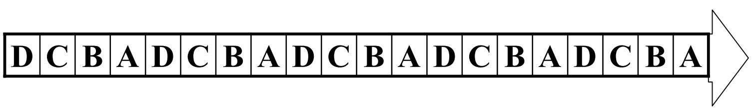
- ❒ **Deficit Round Robin (DRR)**
- ❒ **Virtual Clock**
- ❒ **Packet-by-packet Generalized Processor Sharing (PGPS)**
- ❒ **Self-Clocked Fair Queueing (SCFQ)**

Deficit Round Robin



Time Division Multiplexing

- Guaranteed service rate and service delay
- Service time of each packet is known at the packet arrival time
- Bandwidth waste when packet not arrived on time

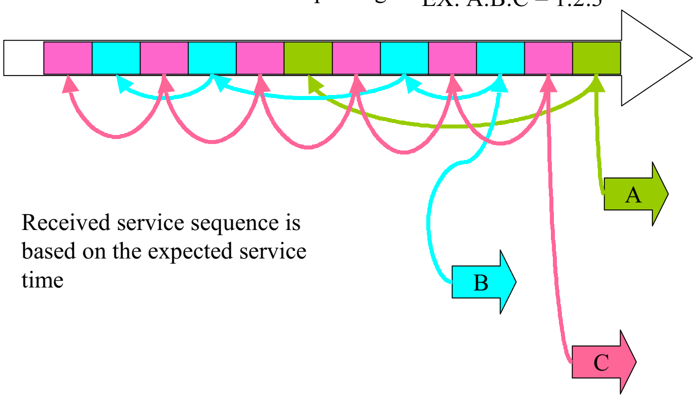


Virtual Clock

$$F_{i,s} = \max\{t_{i,s}, F_{i-1,s}\} + \frac{L_{i,s}}{r_s}, \qquad i \geq 1$$
$$F_{0,s} = 0.$$

Virtual Clock Concept

Emulate Time Division Multiplexing EX. A:B:C = 1:2:3



Received service sequence is based on the expected service time

Virtual Clock Characteristics

- ❒ **Achieve fairness and bounded delay service when all links are full of data traffic**
  - ➔All packet received the same service time as TDM does
- ❒ **Unused bandwidth will be accumulated for latter use**
- ❒ **Not fair for other link**
- ❒ **Sorting is required for selecting the link to be serviced next**

PGPS

$$F_i^0 = 0$$

$$S_i^k = \max\{F_i^{k-1}, V(a_i^k)\}$$

$$F_i^k = S_i^i + \frac{L_i^k}{\phi_i}$$

$V(a_i^k)$ is update every time packet arrived

PGPS Characteristics

- ❒ **Avoid the bandwidth accumulation weakness of Virtual Clock Algorithm**
  - ➔The no-used bandwidth will be used by other link
  - ➔The silence link will get its fair-share when become active
- ❒ **Centralized virtual clock is required**
- ❒ **Sorting is still required**

SCFQ

$$\hat{F}_k^0 = 0$$

$$\hat{F}_k^i = \frac{1}{r_k} L_k^i + \max(\hat{F}_k^{i-1}, \hat{v}(a_k^i))$$

$$\hat{v}(t) \cong \hat{F}_l^j, \hat{s}_l^j < t \leq \hat{d}_l^j$$

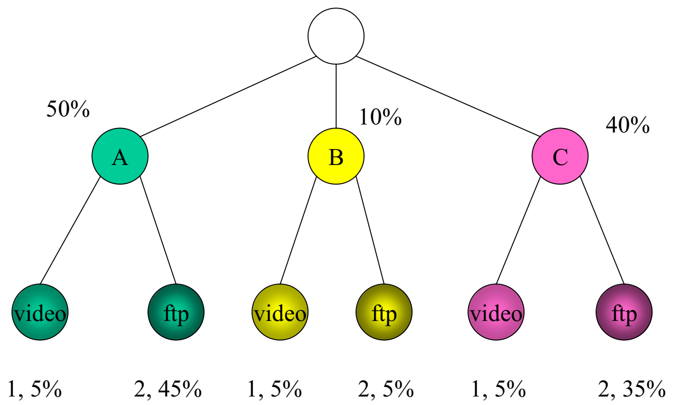
SCFQ Characteristics

- ❒ **Avoids the centralized virtual clock maintenance required by PGPS**
  - ➔Simpler the implementation
  - ➔Reduce the level of delay guarantee
- ❒ **Sorting is still required**

Link Sharing

- ❒ **Requirement for link-sharing**
  - ➔Share bandwidth on a link between multiple organizations
  - ➔Share bandwidth on a link between different protocol families
  - ➔Share bandwidth on a link between different traffic types
- ❒ **Link-sharing goal**
  - ➔Rough quantitative bandwidth commitment by the network
  - ➔When some class is not using its allocated bandwidth, the distribution of the 'excess' bandwidth among the other classes should follow some appropriate guideline

Class-Based Queueing



QoS Architectures (I)

- **Integrated Services (IntServ)**
  - ➔Internet service model that includes best-effort service, real-time service, and controlled link sharing
  - ➔RFC1633
- **Resource ReSerVation Protocol (RSVP)**
  - ➔Provide receiver-initiated setup of resource reservations for multicast or unicast data flows, with good scaling and robustness properties
  - ➔RFC2205, RFC2208, RFC2209

QoS Architectures (II)

- ❒ **Differentiated Services (DiffServ)**
  - ➔ Enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop
  - ➔ A variety of services may be built from a small, well-defined set of building blocks which are deployed in network nodes
  - ➔ RFC2474, RFC2475, RFC2597, RFC2598

Traditional Architecture

- ❒ **Best-effort architecture**
- ❒ **No service guarantee**
- ❒ **No signaling requirement**
- ❒ **Users send as much as they want**
- ❒ **Networks service as much as they can**
- ❒ **Simple FIFO, drop tail**
- ❒ **Drawback**
  - ➔ Hard to achieve real-time requirement
  - ➔ Hard to give different service quality based on different billing level

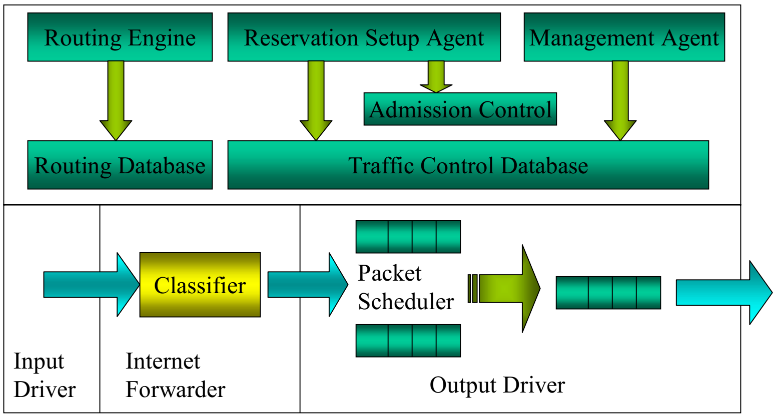
Integrated Services

- ❒ **Extend the best-effort architecture**
  - ➔An extended service model, IS model
  - ➔A reference implementation framework
- ❒ **Assumptions**
  - ➔Resources (e.g., bandwidth) must be explicitly managed in order to meet application requirements
    - Resource reservation
    - Admission control
  - ➔Use the Internet as a common infrastructure

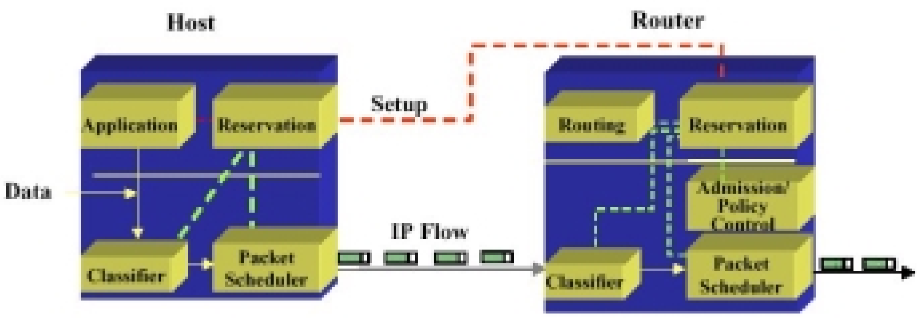
IntServ Framework

- ❒ **Components**
  - ➔Packet scheduler
  - ➔Admission control routine
  - ➔Classifier
  - ➔Reservation setup protocol
- ❒ **Traffic control components to provide QoS**
  - ➔Packet scheduler
  - ➔Admission control routine
  - ➔Classifier

IntServ Reference Model



IntServ Model



QoS Requirements

- ❒ **The only quantity are bounds on the maximum and minimum delays**
- ❒ **Classes of applications**
  - ➔ Real-time applications
    - Applications needs the data in each packet by a certain time and, if the data has not arrived by then, the data is essentially worthless
  - ➔ Elastic applications
    - Applications will always wait for data to arrive

Link-Sharing Examples

- ❒ **Multi-entity link-sharing**
  - ➔ Shared among organizations
- ❒ **Multi-protocol link-sharing**
  - ➔ Shared among defined protocol groups
- ❒ **Multi-service sharing**
  - ➔ Shared among defined applications

Admission Control

- The decision about resource availability
- Require the router understand the demands that are currently being made on its assets
- Traditional way
  - ➔Remember the service parameters of past requests and make a computation based on the worst-case bounds on each service
- Recent proposal for better link utilization
  - ➔Measure the actual usage by existing packet flows and use this measured information for admission control

IntServ Components

- Reservation Protocol – RSVP
- Traffic Control:
  - ➔Admission/Policy Control – Determines if QoS request can/should be granted
  - ➔Classifier – maps packets to a service class by looking at contents of IP header
  - ➔Packet Scheduler – forwards packets based on service class using sophisticated queuing mechanisms (e. g. WFQ)

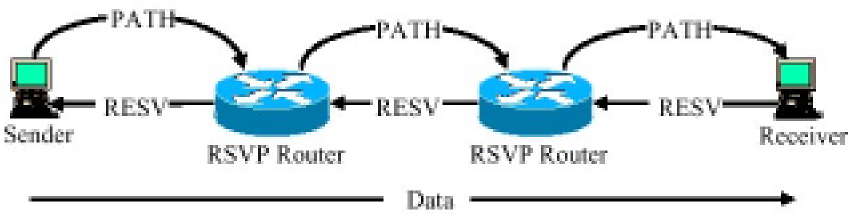
Resource Reservation Protocol (RSVP)

- ❒ **RSVP Flow:**
  - ➔PATH messages mark path and deliver sender and path QoS information to receiver
  - ➔RESV messages flow upstream towards sender and lay down QoS state
  - ➔soft state so PATH and RESV refresh messages flow periodically
- ❒ **RFC2205**

RSVP Attributes

- ❒ **Make resource reservation for both unicast and many-to-many multicast**
- ❒ **Make reservations for unidirectional data flows**
- ❒ **Receiver-oriented**
- ❒ **Maintains "soft" state in routers and hosts**
- ❒ **Not routing protocol but depends on routing protocol**
- ❒ **Provides several reservation models or styles**

RSVP Flow



Reservation Model

- ❒ **Flowspec is a reservation request include**
  - ➔Rspec: defines the desired QoS
  - ➔Tspec: defines the data flow
- ❒ **Reserve request actions**
  - ➔Make a reservation on a link
  - ➔Forward the request upstream

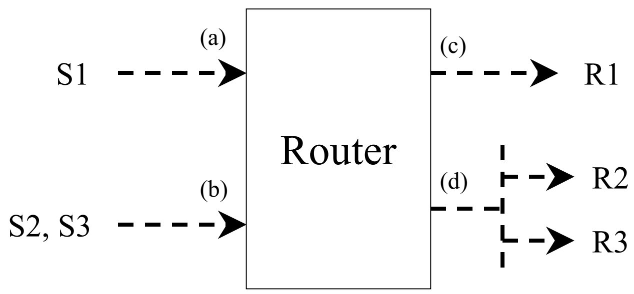
Reservation Styles

- ❒ **Session is a data flow with a particular destination and transport-layer protocol**
  - ➔ Transport-layer contains protocol and destination port
- ❒ **Concerns the treatment of reservations for different senders with the same session**
- ❒ **Three styles are defined**
  - ➔ Wildcard-filter style (WF)
  - ➔ Fixed-filter style (FF)
  - ➔ Shared explicit style (SE)

RSVP Style Definition

Sender Selection	Reservation	
	Distinct	**Shared**
Explicit	**Fixed-Filter (FF) Style**	**Shared-Explicit (SE) Style**
Wildcard	**None defined**	**Wildcard-Filter (WF) Styles**

RSVP Example Configure



WF Reservation Example

Sends		Reserves		Receives	
WF(*{4B})	**a**	**c**	***{4B}**	**c**	**WF(*{4B})**
WF(*{4B})	**b**	**d**	***{3B}**	**d**	**WF(*{3B})** **WF(*{2B})**

FF Reservation Example

Sends		Reserves		Receives	
FF(S1{4B})	a	c	S1{4B} S2{5B}	c	FF(S1{4B}, S2{5B})
FF(S2 {5B}, S3{B})	b	d	S1{3B} S3{B}	d	FF(S1{3B}, S3{B}) FF(S1{B})

SE Reservation Example

Sends		Reserves		Receives	
SE(S1{3B})	a	c	(S1, S2){B}	c	SE((S1,S2) {B})
SE((S2, S3) {3B})	b	d	(S1, S2, S3) {3B}	d	SE((S1,S3){ 3B}) SE(S2{2B})

Differentiated Services

- ❐ **Uses edge-based packet-marking, local per-class forwarding behaviors and network provisioning to support multiple service levels over an IP-based network**
- ❐ **DiffServ concepts:**
  - ➔DiffServ codepoints (DSCP) in packet header indicate how packet should be serviced at each hop
  - ➔DSCP marked at ingress based on analysis of packet
  - ➔Intermediate routers/switches service the packets based on the codepoints

Why DiffServ?

- ❐ **Simpler than RSVP/IntServ or ATM**
  - ➔no signaling or per-flow state to maintain in network
  - ➔Scalable!!
- ❐ **No change to applications**
- ❐ **Efficient for Core Routers**
  - ➔couple of bits indicate forwarding treatment
  - ➔difficult “high touch” work done at the network edge
- ❐ **Interior Network Independence**
  - ➔can be IP, ATM, Frame Relay, MPLS, mixture, etc.
- ❐ **Different packet handling services and mappings possible:**
  - ➔Service class indicator (e.g. premium and best-effort)
  - ➔Congestion Control – low priority packets are discarded first

DiffServ Philosophy

- ❒ **Push complex/ heavy tasks to edges**
  - ➔Traffic conditioning
    - Marking, metering, policing, shaping
- ❒ **Make the core case fast**
- ❒ **Assume core is secure**
  - ➔Marking is trusted
- ❒ **Use PHBs as building blocks for E2E service**
  - ➔Myth alert!

DiffServ Definitions

- ❒ **Service – what is sold and delivered to customer**
- ❒ **DiffServ Codepoints (DSCP) – bits in packet header that define the PHB the packet will receive**
- ❒ **Per-Hop Behavior (PHB) – Externally observable forwarding treatment that packets with the same DSCP receive from a network node**
- ❒ **Behavior Aggregate (BA)– group of packets with the same DSCP crossing a link in a particular direction**

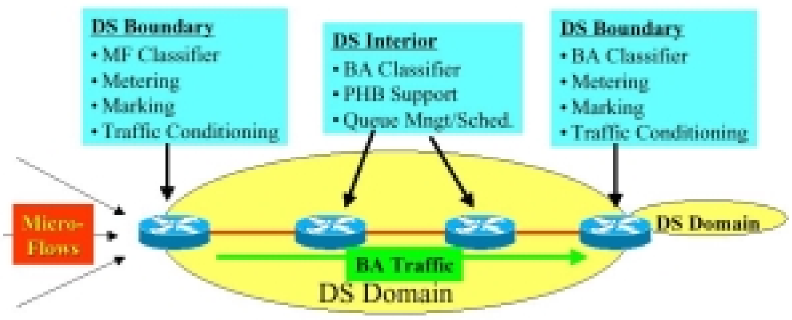
DiffServ Definitions

- ❒ MF Classifier – examines multiple fields in packet header
- ❒ BA Classifier – examines just the DSCP in packet header
- ❒ Traffic Conditioning – functions such as policing, marking, metering, shaping performed at network edge
- ❒ DS Boundary Device – positioned at ingress or egress of DS domain
- ❒ DS Interior Device – operates in the core of a DS domain

DiffServ SLA/TCA

- ❒ Traffic Conditioning Agreement (TCA): an agreement specifying classifier rules and any corresponding traffic profiles and metering, marking, discarding and/ or shaping rules which are to apply to the traffic streams selected by the classifier.
- ❒ Service Level Agreement (SLA): a service contract between a customer and a service provider that specifies the forwarding service a customer should receive. A customer may be a user organization (source domain) or another DS domain (upstream domain). A SLA may include traffic conditioning rules which constitute a TCA in whole or in part.

DiffServ Network



DS-field Format

- **Differentiated Services Codepoint (DSCP)**
  - ➔used to select the service (PHB) the packet will receive at each DS-capable node
- **formerly the IPv4 TOS and IPv6 Traffic Class fields**
- **6 bits for DSCP, 2 bits are currently undefined (ECN??)**
- **RFC2474**

DSCP Marking

- ❒ **Client may mark DiffServe Code Point (DSCP) and may also mark 802.1P**
- ❒ **Layer 2 switch may mark 802.1P**
- ❒ **1st DiffServ router may mark DSCP using**
  - ➔802.1P marking, VLAN, physical port, IP source/destination addresses, IP protocol, TCP/UDP source/destination ports
- ❒ **Boarder router may mark/ re-mark DSCP(using above criteria)**

DiffServ PHBs

- ❒ **Default - best effort**
- ❒ **Expedited Forwarding (EF)**
  - ➔low delay, latency, jitter service
  - ➔RFC2598
- ❒ **Assured Forwarding (AF)**
  - ➔4 “relative” classes of service
  - ➔3 level of dropping precedence in each class
  - ➔RFC2597

EF PHB Definition

- ❒ **EF PHB ensures a minimum departure rate**
  - ➔Independently of any other traffic attempting to transit across the node
  - ➔DSCP: "101110"
- ❒ **EF PHB can be used to build a low loss, low latency, low jitter, assured bandwidth, e2e service through DS domains**
  - ➔Because it ensures that BA sees no (very small) queues. The aggregate's max arrival rate is less than that aggregate's min departure rate
  - ➔Strict policing at DS domain boundary (excess of contract must be dropped!!!)
  - ➔Drops on the ingress policer and in the EF-PHB within the DS domain indicate either denial of service attack or wrong configuration.

EF PHB Behavior

- ❒ **The departure rate of the aggregate's packets from any diffserv node must equal or exceed a configurable rate**
- ❒ **The configured minimum rate MUST be settable by a network administrator**
- ❒ **If EF received unlimited preemption, rate limiter must be implemented**
  - ➔Priority queue is the example of unlimited preemption

AF DSCP Definition

- ❒ **AF Class**
  - ➔Class1: DSCP 001dd0
  - ➔Class2: DSCP 010dd0
  - ➔Class3: DSCP 011dd0
  - ➔Class4: DSCP 100dd0
- ❒ **Dropping precedence (dd part)**
  - ➔01: Low drop priority
  - ➔10: Medium drop priority
  - ➔11: High drop priority

AF PHB Definition

- ❒ **Offer different levels of forwarding assurances for IP packets received from a customer DS domain**
- ❒ **4 independently forwarded AF classes**
- ❒ **Within each AF class, 3 levels of drop precedence**
  - ➔Within each AF class, RED-like buffer management is used to implement drop precedence

Assured Level

- ❒ How much forwarding resources has been allocated to the AF class that the packet belongs to
- ❒ What is the current load of the AF class and in case of congestion within the class
- ❒ What is the drop precedence of the packet

AF Service Requirement

- ❒ Implement all four general use AF classes
  - ➔Different class packets are serviced independently
- ❒ Allocate a configurable, minimum amount of forwarding resources to each AF class
  - ➔Resources contain buffer space and bandwidth
- ❒ At least two different levels of loss probability
- ❒ Must not reorder packets belong to the same microflow

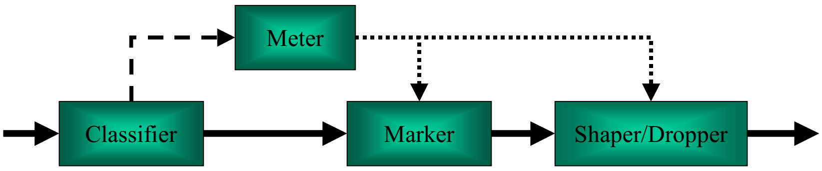
Other Codepints

- **Best-effort**
  - DSCP: 000000
- **Class Selector**
  - DSCP: xxx000
- **Network control traffic**
  - DSCP: 11x000

DiffServ Routers

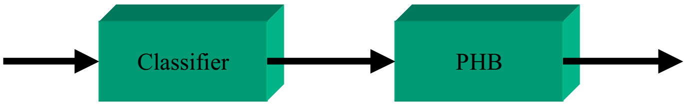
- **Edge router**
  - At the edge of the DS domain
  - Contains meter, classifier, DSCP marker and shaper/dropper
- **Interior router**
  - Inside of the DS domain
  - Contains DS classifier and PHB provider

DiffServ Edge Router



- ❒ Classifier – MF or BA
- ❒ Meter – Measures traffic against profile
- ❒ Packet Marker – Marks DSCPs
- ❒ Shaper/ Dropper – Traffic Conditioners

DiffServ Interior Router

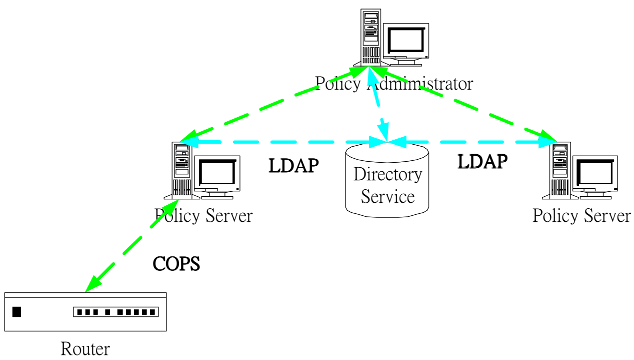


- ❒ Classifier – BA only
- ❒ PHB – supported by queue management/scheduling techniques

DiffServ Configuration

- ❒ **Manual setting**
- ❒ **Provided by policy server**
  - ➔ **Policy Application defines the rules, filters, etc. of DiffServ services Client**
  - ➔ **LDAP – Lightweight Directory Access Protocol**
  - ➔ **COPS – Common Open Policy Service Protocol**

Configuration Model



LDAP Overview

- ❒ Protocol elements are carried directly over TCP or other transport protocol
- ❒ Many protocol data elements are encoding as ordinary strings
- ❒ Lightweight Basic Encoding Rules (BER) encoding is used

LDAP Protocol Model

- ❒ One of clients performing protocol operations against servers
  - ➔ Client transmits a protocol request describing the operation to be performed to a server
  - ➔ Server performs necessary operation on Directory
- ❒ No synchronous behavior are required
  - ➔ Requests and responses for multiple operations may be exchanged by client and servers in any order

LDAP Operations

- ❒ **Bind/Unbind**
  - ➔Protocol session initial related operation
- ❒ **Search**
- ❒ **Modify/Add/Delete**
- ❒ **Modify RDN**
  - ➔Change the last component of the name of an entry
- ❒ **Compare**
- ❒ **Abandon**

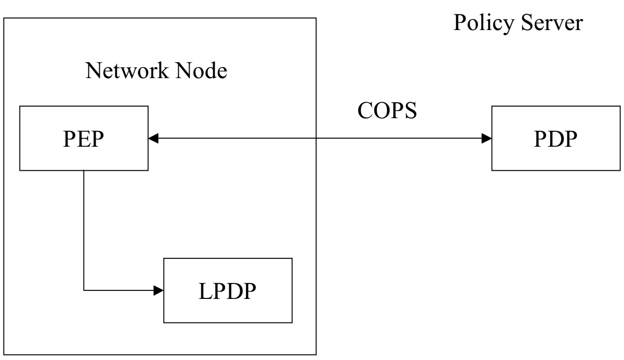
COPS Protocol

- ❒ **Common Open Policy Service**
  - ➔Client-server protocol
  - ➔Outsource policy decisions to an external server
- ❒ **Exchange policy information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs)**
- ❒ **Begin with a simple but extensible design**

COPS Characteristics

- ❒ **Employ a client/server model**
- ❒ **Use TCP as its transport protocol**
- ❒ **Created for the general administration, configuration, and enforcement of policies**
- ❒ **Provide message level security**
- ❒ **Stateful in two main aspects**
  - ➔ Request/Decision state is shared between client and server
  - ➔ State from various events may be inter-associated
- ❒ **Allow the server to push/remove configuration information to the client**

COPS Basic Model



Why Would DiffServ Succeed?

- ❒ **Market Maturity**
  - ➔Different era, customers, demands
  - ➔More sophisticated equipment
- ❒ **Manageability & Provisioning**
  - ➔ISP Peering (vs. exchange point)
  - ➔Policy based network management
  - ➔Automation of provisioning