Differentiated Service Router Architecture - Classification, Metering and Policing

Presenters: Daniel Lin and Frank Akujobi

Carleton University,

Department of Systems and Computer Engineering

94.581 Advanced Topics in Computer Communications

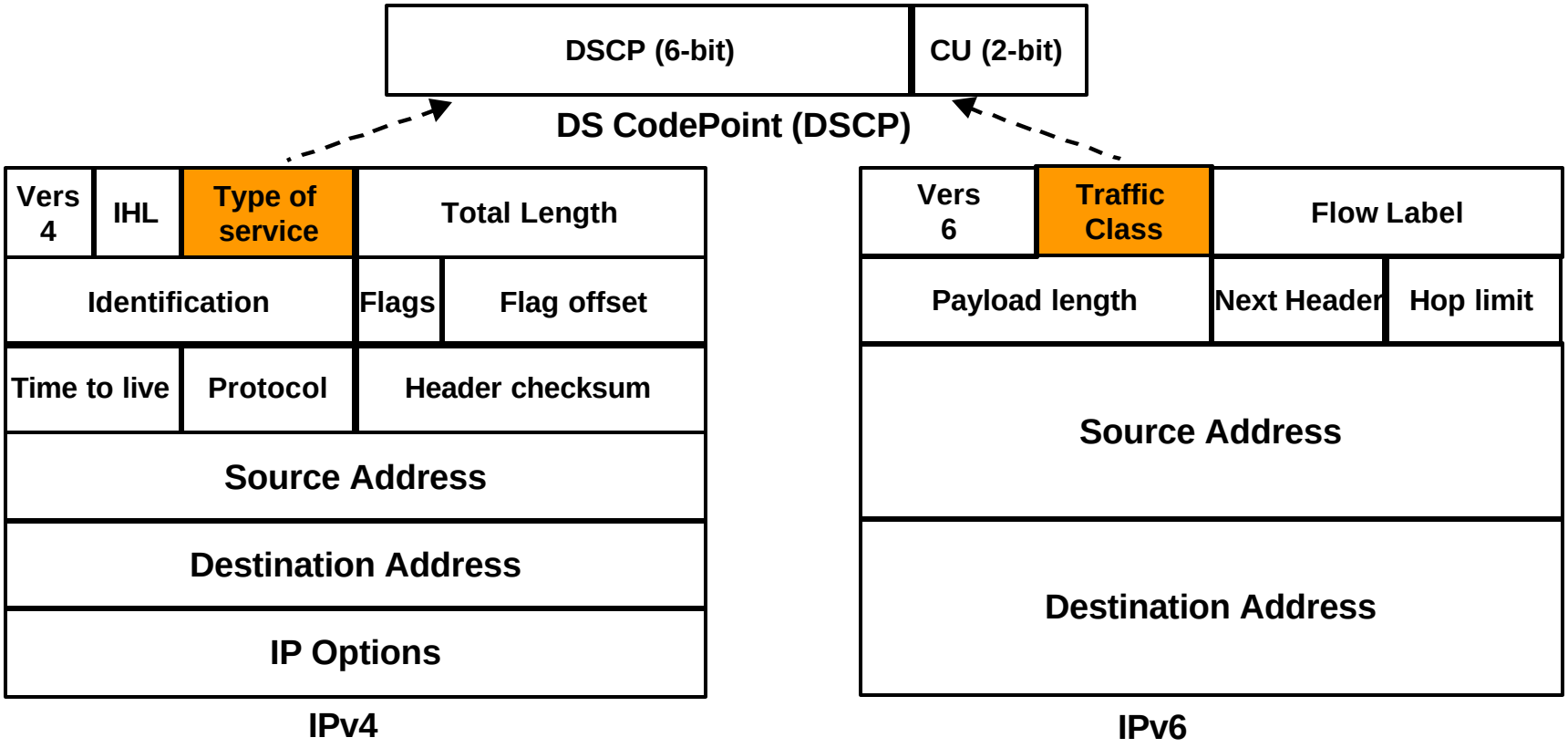
Presentation Outline

- **Introduction**
  - Internet and Differentiate Service (DiffServe)
- **Terminology**
  - DiffServ codepoint (DSCP), traffic profile, per-hop behavior (PHB)
- **DiffServ Network Diagram**
  - General DiffServ network diagram
  - DiffServ router architectures for ingress, interior, and egress nodes
- **DiffServ Router Architecture**
  - fan-out element: classifiers and meters
  - action element: marker, shaper/droppers, multiplexor, counter
  - queueing element: FIFO queue, schedulers, algorithmic dropper
- **Router Architecture Example**
  - Traffic Conditioning Block using BA classifiers for a single customer
  - Cascading BA classifiers for multiple customers
  - Microflow based traffic conditionings for a single customer
- **Summary**
- **Reference**

Introduction

- **Historical background (Internet)**
  - best-effort class in the traditional Internet becomes inefficient and slow
  - QoS is not implemented to provide various services
  - all traffics are treated equally, no service differentiation is implemented
- **Would it be nice if we can**
  - differentiate traffics based on their QoS requirement
  - have different treatments for different types of traffic and different billings
  - scale the network to support the Internet
- **Here comes Differentiated Service (DiffServ)**
  - derived from existing IP network and from Integrated Service (IntServ)
  - differentiate traffics based on their QoS requirement
  - use different per-hop behaviors (PHB) to specify forwarding treatment
- **In this presentation, we will cover**
  - basic framework of DiffServ
  - DiffServ router architecture: classifiers, meters, droppers and queue
  - examples of how a DiffServ router is constructed
  - conclusion of comparison

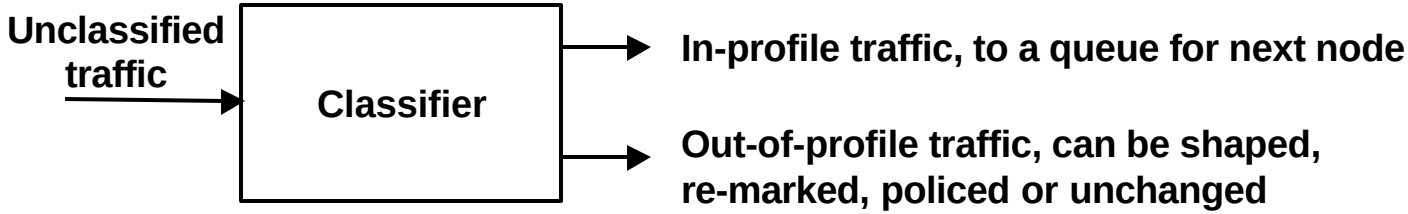
DiffServ codepoint (DSCP) in the IPv4 and IPv6 Headers



- an octet specifying the PHB class
- inserted in the TOS octet of IPv4 or Traffic Class octet in IPv6
- is backward compatible with existing IP packets
- 6-bit codepoint with 2-bit un-used

Traffic Profile

- **Is a temporal property for a stream of packets**
- **Provide rules whether a packet is in-profile or out-of-profile**

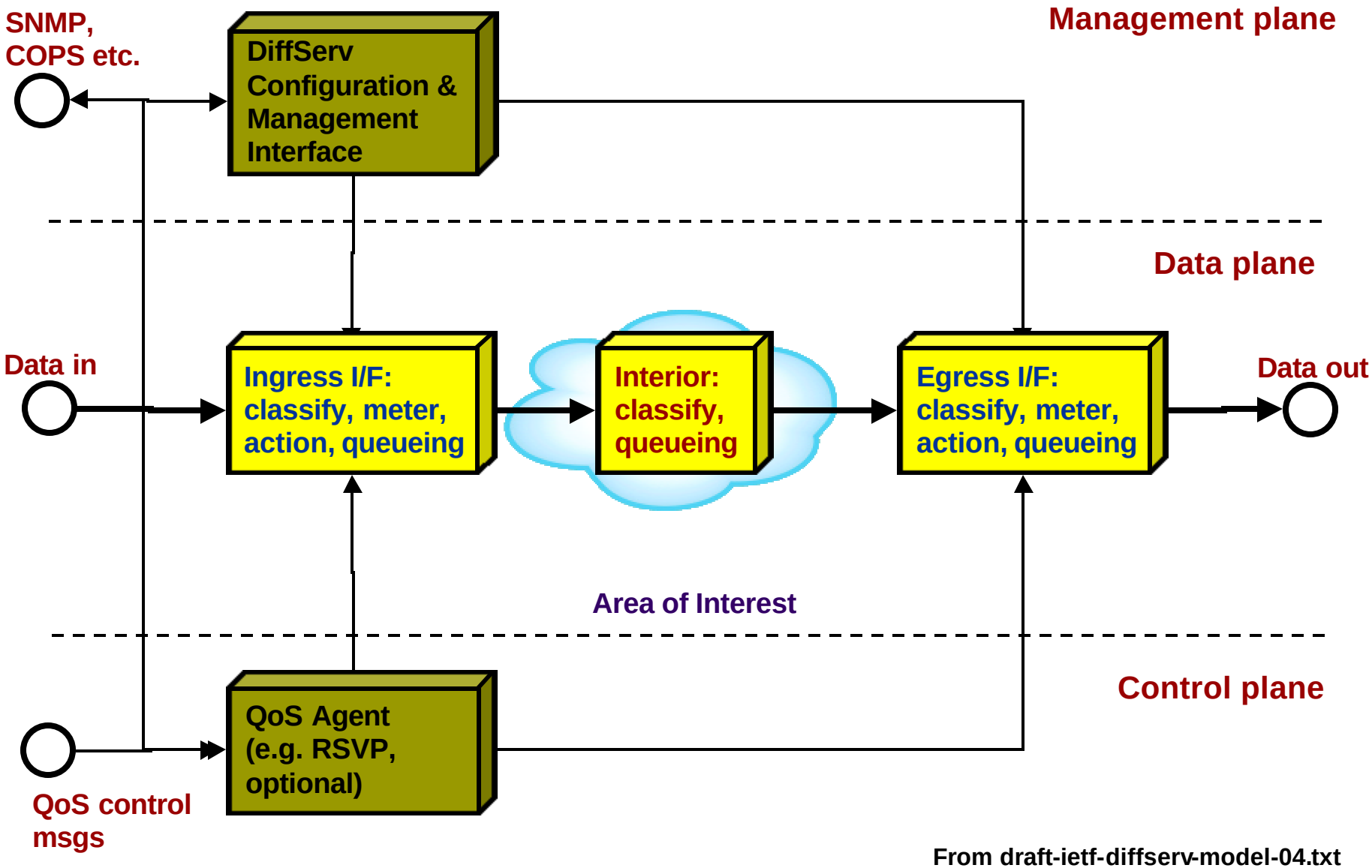


  - in-profile packets
    - **traffic is transmitted within Service Level Agreement (SLA)**
    - **their DSCP will be unchanged or remarked to different value**
  - out-of-profile packets
    - **traffic is transmitted not within SLA**
    - **they can be: queued until they are in-profile (shaped), marked with a new DSCP (remarked), discarded (policed), or forwarded with unchanged DSCP**
- **Can have more than two levels of conformance**
- **For example: a profile of token bucket**
  - DSCP = XXXXXX, token rate r, burst size b
  - This represents that when a packet arrives at the server, as long as there is a token available in the pool, this packet is marked with in-profile DSCP value XXXXXX.

Per-Hop Behavior (PHB)

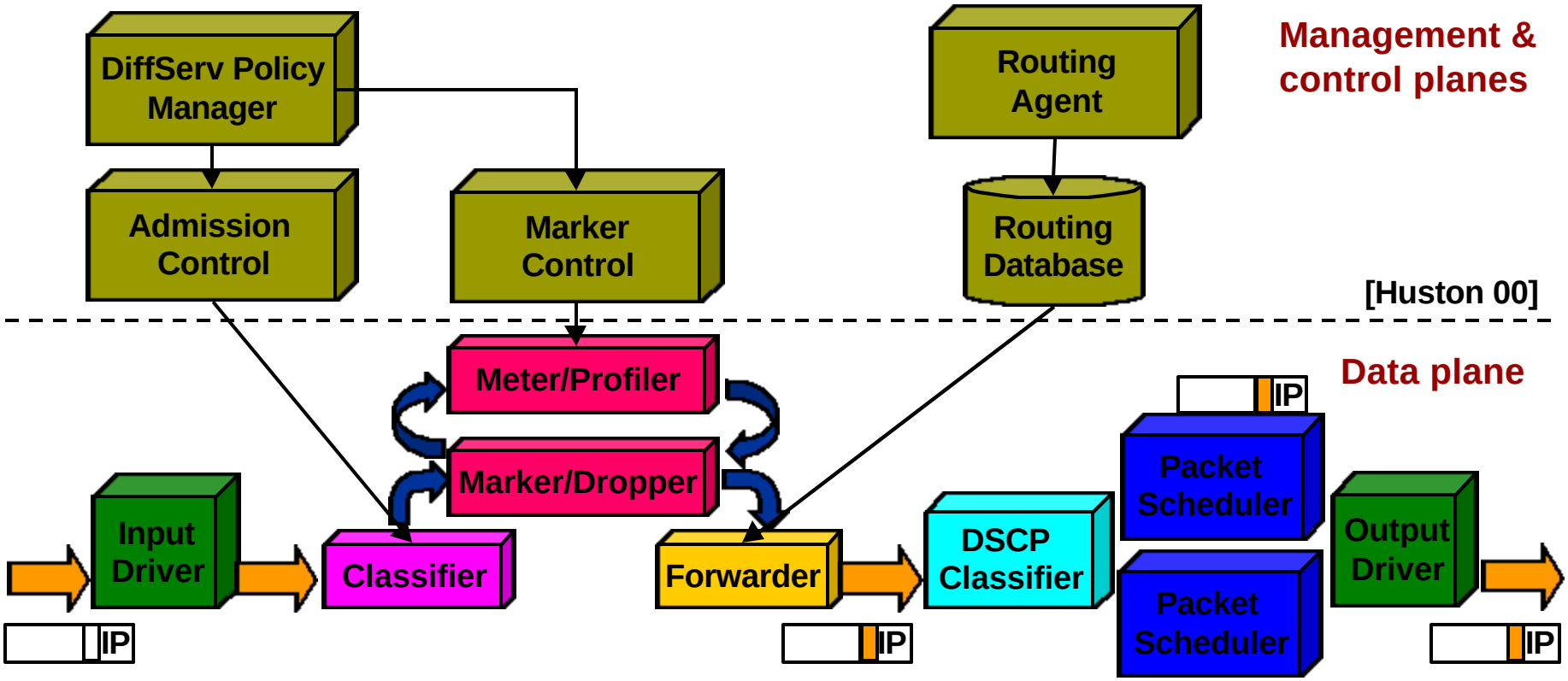
- **Map a forwarding treatment for the following based on DSCP**
  - resource allocation such as buffer and transmission bandwidth
  - traffic characteristics such as delay and loss
- **PHB groups (3 plus 1 default are defined currently)**
  - Default group (DF): DSCP = 000000
    - **the default IP best-effort treatment**
    - **compatible with non-DiffServ-aware nodes**
  - Class Selector (CS): use the first 3 bit XXX000 to specify priorities
    - **code format : 000000 (lowest) to 111000 (highest) priorities**
    - **Provide backward compatibility for IP TOS usage**
  - Expedited Forwarding (EF): [RFC 2598]
    - **support a BA with low loss, low latency, low jitter, assured bandwidth end-to-end service, similar to the lease line in the IP network**
  - Assured Forwarding (AF): 12 DSCPs [RFC 2597]
    - **assure the packets will be forwarded as long they conform**

DiffSerrv Router Major Functional Block



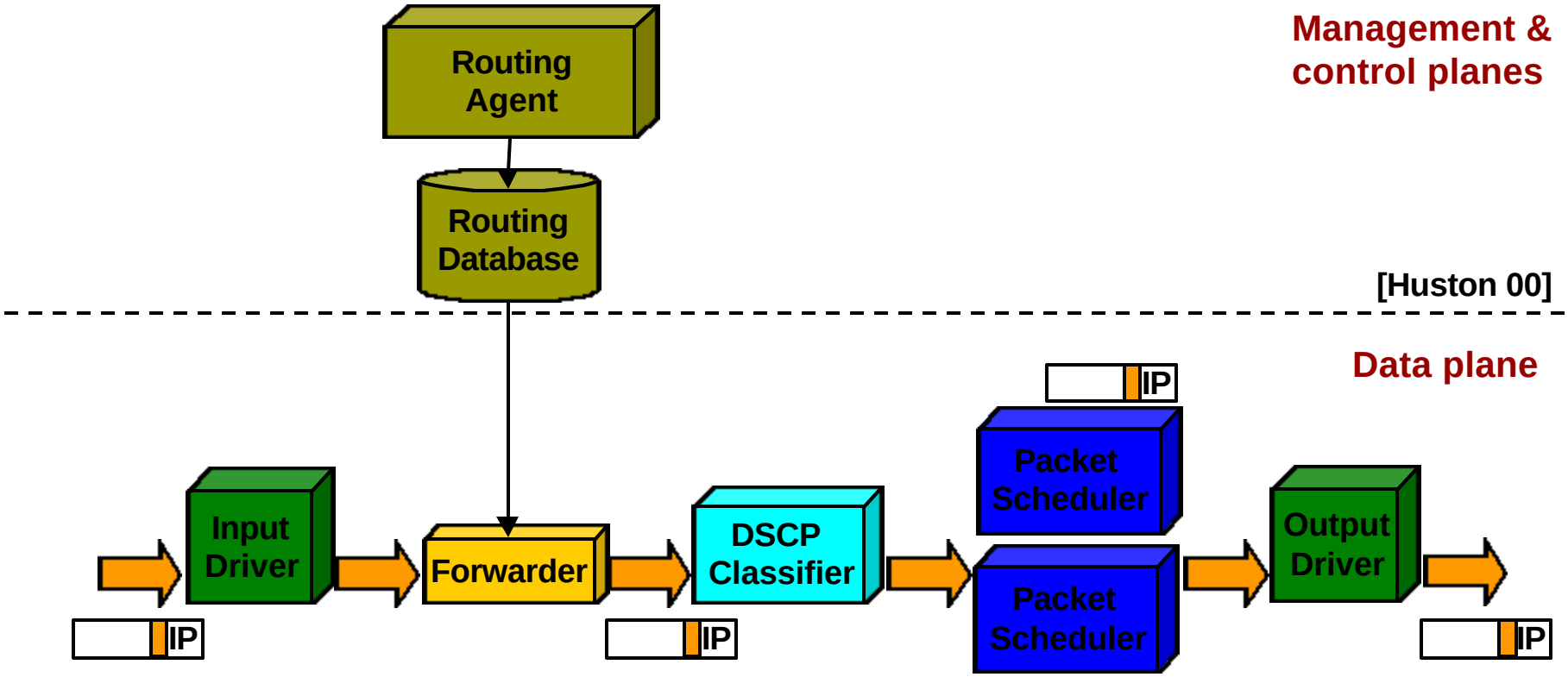
From draft-ietf-diffserv-model-04.txt

Router Architecture - DiffServ Ingress Router



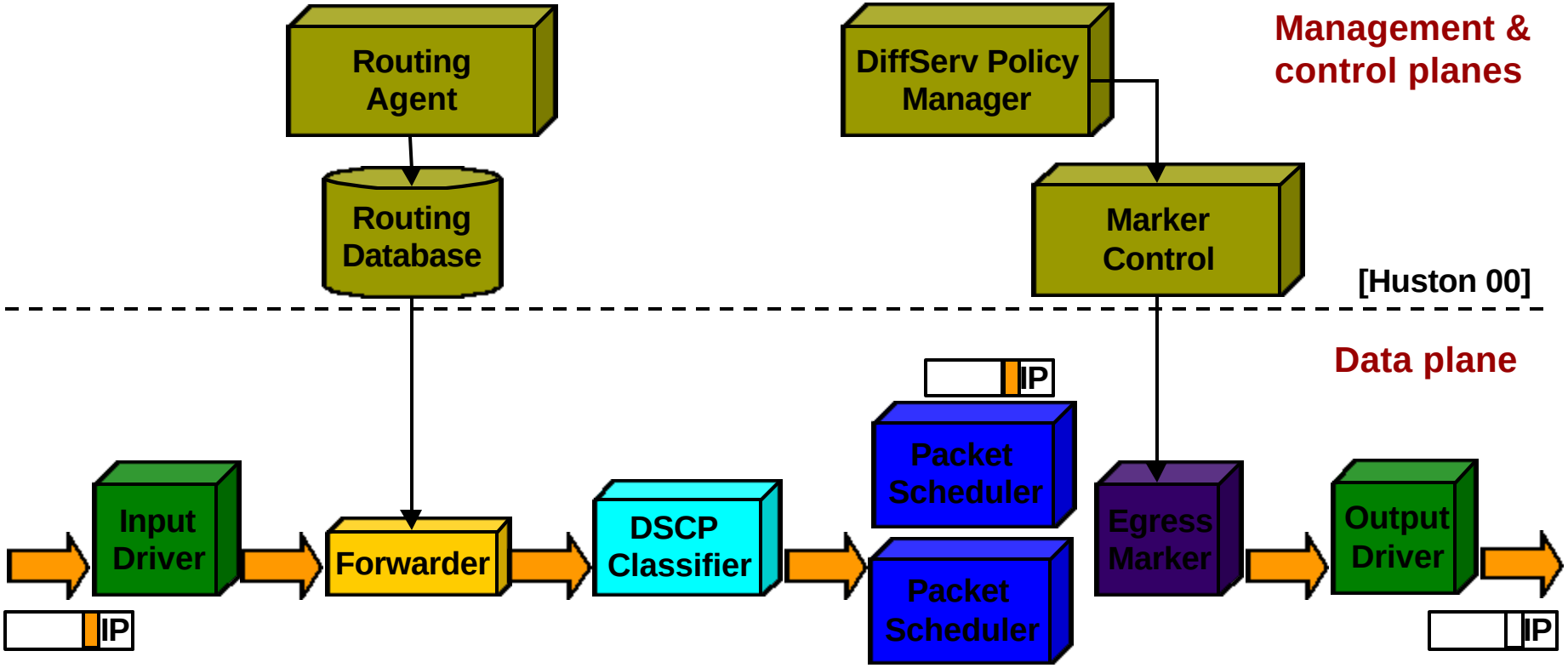
- An un-marked packet enters the Ingress router for classification
- The marker/meter marks and counts the packet
- The forwarder forwards this marked packet to DSCP classifier
- DSCP classifier assigns this packet to different schedulers
- The output driver loads this packet to the next hop

Router Architecture - DiffServ Interior Router



- The marked packet enters and is forwarded based on IP routing
- DSCP classifier assigns this packet to different schedulers
- The output driver loads this packet to the next hop

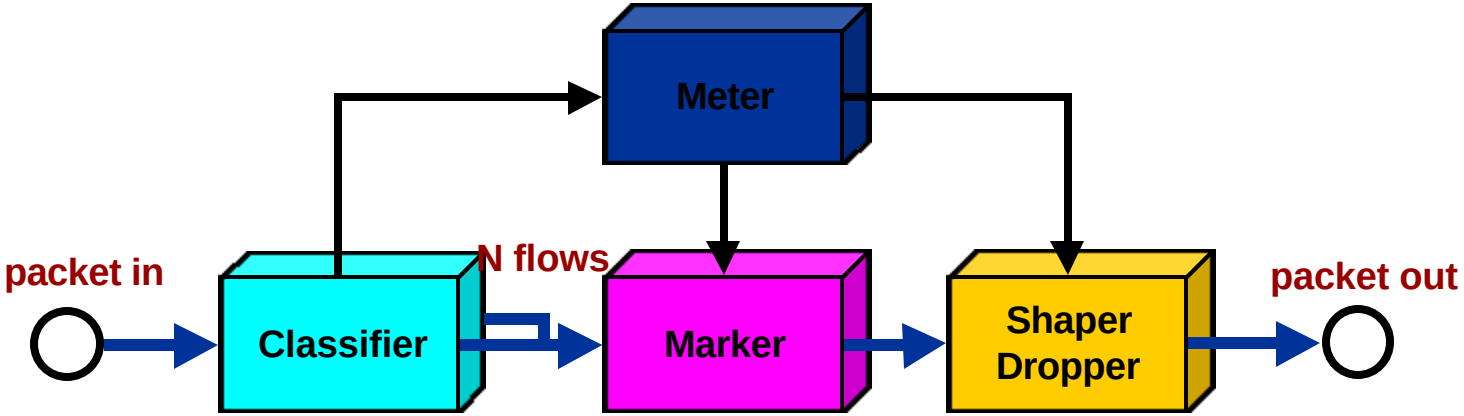
Router Architecture - DiffServ Egress Router



- The marked packet enters and is forwarded based on IP routing
- DSCP classifier assigns this packet to different schedulers
- The egress marker re-mark or reset the DSCP of this packet for the next DS or Non-DS domains
- The output driver loads this packet to the next hop

Router Architecture

Traffic Conditioner Block

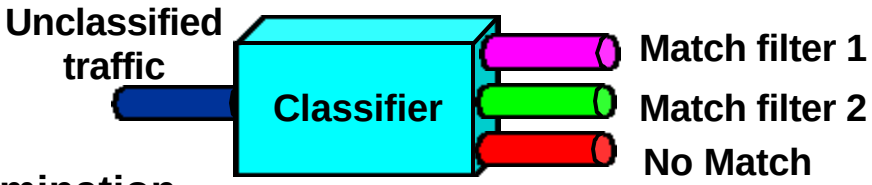


- **The traffic conditioning block (TCB) contains**
  - fan out elements: classifier, meter
  - action elements: marker, shaper/dropper, multiplexor, counter, null action
  - queueing elements: FIFO queues, Weighted Fair Queue (WFQ), schedulers and algorithmic dropper

From rfc2475.txt

Fan-Out Elements: Classifiers (1:N element)

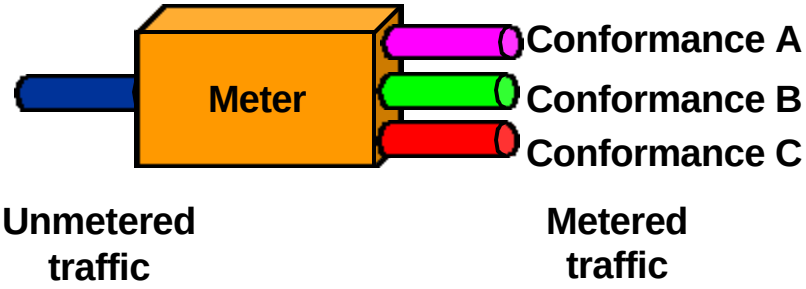
- **Classifier**
  - separate one input traffic to N streams based on the content of packet or other attributes such as input port numbers
- **Types of Classifiers**



  - Behaviour Aggregate (BA)
    - **classify based on DSCP only**
    - **use exact-match condition for determination**
  - Multi-Field (MF)
    - **classify based on various fields in the packet**
    - **example: 6-tuple classifier based on 6 fields (destination address; source address; IP protocol; source port; destination Port; and DSCP) in IP and TCP/UDP headers**
    - **can use other fields such as MAC address, VLAN tags, and other fields**
  - Free-form
    - **classify based on user-definable filters made of (bit-field size, offset (from head of packet), mask)**
  - Others
    - **classify based on data link layer information or ingress or egress IP, logical or physical interface identifier**

Fan-Out Elements: Meters (1:N element)

- **Meters**
  - setup by service provider based on a temporal (i.e., rate) profile
  - collect out-of-band management function with counter
  - parameterized by a temporal profile and by conformance levels
- **Examples of Meters**
  - Average Rate Meter
  - Exponential Weighted Moving Average (EWMA) Meter
  - Two-Parameter Token Bucket Meter
  - Multi-State Token Bucket Meter
  - Null Meter



Average Rate Meter

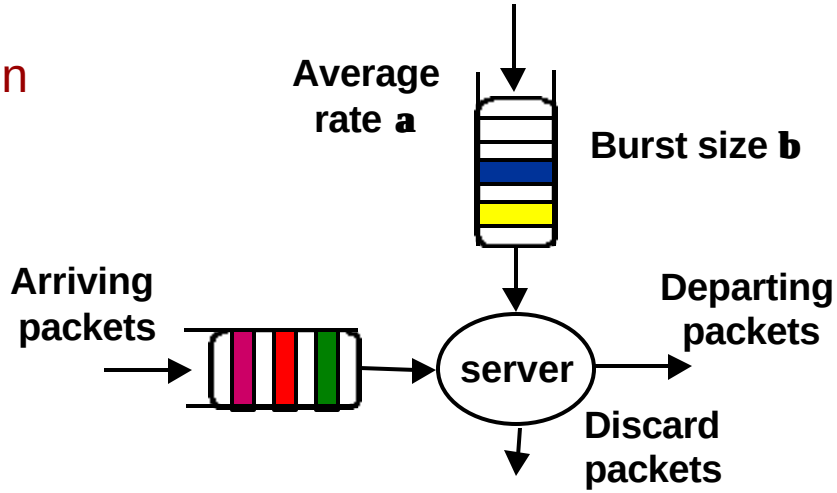
- **Measure average rate the packets are submitted over a time frame**
- **Has one input and two outputs (conforming and non-conforming)**
  - conforming output is connected with a queue for transmission
  - non-conforming output is connected with a counter for out-of-profile treatment
- **Example: average traffic = 120 kbps, delta period = 100 msec**
  - the meter measure overall traffic of the packets between T and T-100 msecs
  - conforming criteria:
    - **traffic in 100 msec does not exceed 12 kbps**
  - non-conforming criteria:
    - **traffic in 100 msec does exceed 12 kbps**
    - **the non-conforming traffics (transmitted packets that exceeds the average rate) will be marked as out-of-profile packets**

Exponential Weighted Moving Average (EWMA) Meter

- **Has one input and two outputs (conforming and non-conforming)**
  - conforming output is connected with a queue for transmission
  - non-conforming output is connected with AbsouteDropper for discarding
- **Parameters:**
  - avg_rate(t) = (1 - gain) * avg_rate(t') + gain * rate(t)
  - t = t' + Delta
  - gain controls the time constant of the response
  - rate(t) measures the number of incoming byte in a small fixed sampling interval delta
  - AverageRate is pre-defined by the SLA
- **Conforming criteria:**
  - if (avg_rate(t) > AverageRate) then
    - **it is non-conforming**
  - else
    - **it is conforming**
- **Example:**
  - AverageRate = 25 kbps
  - Delta = 10 usec
  - Gain = 1/16

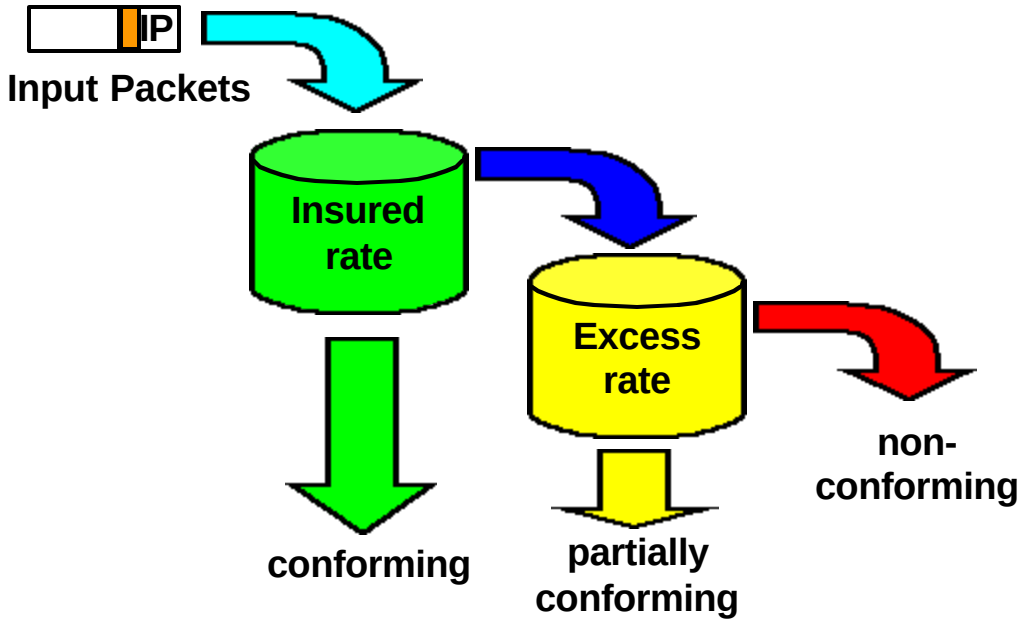
Two-Parameter Token Bucket (TB) Meter

- **Has one input and two outputs (conforming and non-conforming)**
  - conforming output is connected with a queue for transmission
  - non-conforming output is connected with Absolute Dropper for discarding
- **Functions performed**
  - measure conformance with average token rate and a burst size
  - compare the arrival rate of traffic to the average rate specified
  - accumulate tokens in a bucket at the average rate
  - set the bucket size by the burst size specified in TB profile
- **Conforming Criteria**
  - as long as there is token available at the time of packet arrival
  - packets are allowed to exceed the average rate up to the burst size
- **Non-Conforming Criteria**
  - no token is available for transmission
- **Example**
  - AverageRate **(a)** = 200 kbps
  - BurstSize **(b)** = 100 kbytes



Multi-State Token Bucket Meter

- **Provide more choices of conformances using multiple burst sizes**
- **Implemented by cascading many two-parameter TB meters**



- **Conformance Criteria**
  - packets exceeding larger burst size are deemed non-conforming
  - packets exceeding smaller burst size are deemed partially conforming
  - packets exceeding neither are deemed conforming

Action Elements

- **Marker (1:1 element)**
  - set a DSCP to an IP header
  - act on unmarked packets or remark previously marked packets
  - marking based on a preceding classifier match
  - DSCP determines the PHB treatment in downstream or next stage nodes
- **Absolute Dropper (1:0 element)**
  - an action element that simply discard packets
  - not the only element can discard packet (algorithmic dropper)
  - a counter is required to count the dropped packets
- **Multiplexor (N:1 element)**
  - merge multiple traffic streams (datapaths) into a single traffic
- **Counter (1:1 element)**
  - count the arriving/departing packets or packets to be dropped
  - used for customer billing, service verification or network engineering purposes
  - increment by 1 packet with L-byte size
- **Null (1:1 element)**
  - performs no action on the packet
  - useful to define in the event that the configuration or management interface does not have the flexibility to omit an action element in a datapath segment

Traffic Shaping

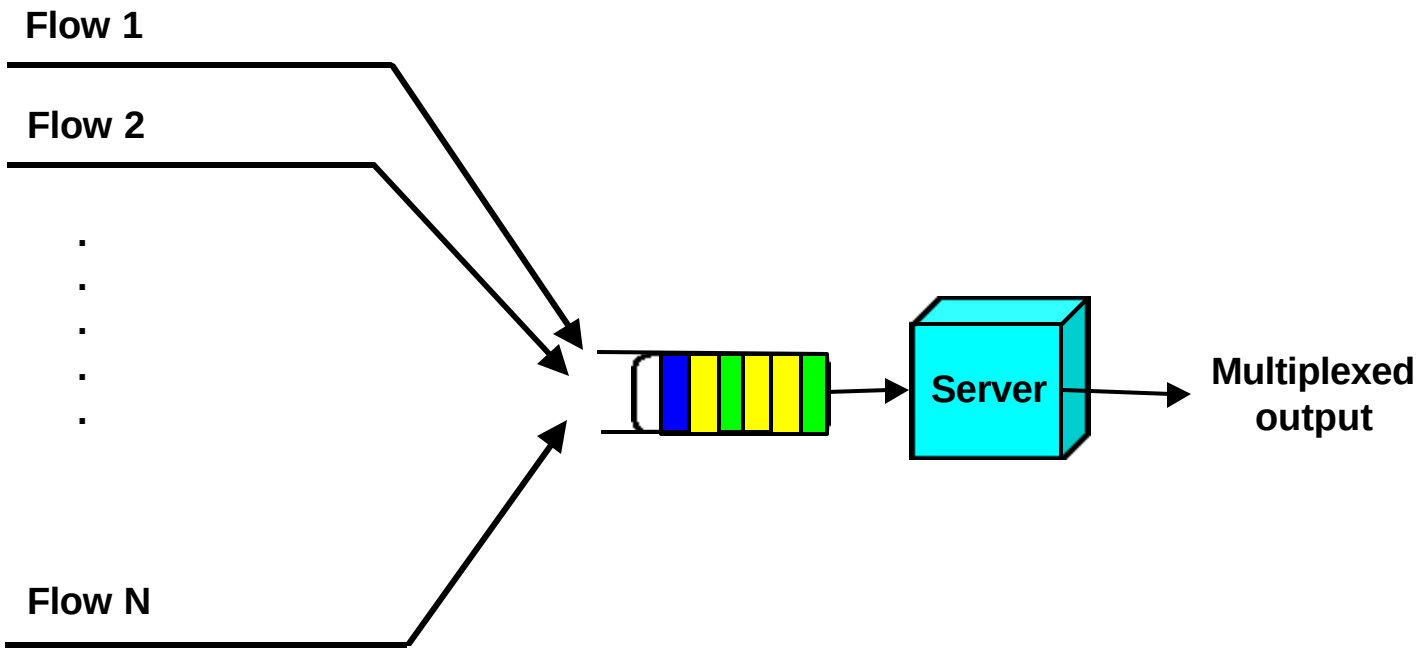
- **About traffic shaping**
  - control the outgoing traffic to match speed at receiving end
  - ensure traffic conforms policy established
- **Why use traffic shaping**
  - control access to available bandwidth
  - ensure traffic conforms to the policies established
  - regulate the flow of traffic to avoid congestion
- **Many forms of traffic shaping**
  - use Token Bucket for traffic shaping (page 406, [Stallings 00])
  - use queues (FIFO, Weighted Fair Queue (WFQ), to perform traffic shaping
  - use droppers (algorithmic droppers like Random Early Detection (RED) and Weighted RED) to regulate excess traffic

Queue Elements

- **Modulate the packets belonging to the different traffic streams**
  - determine their ordering, possibly storing them temporarily or discarding them.
- **Packets are stored/queued**
  - if immediate output port is not available (e.g., bandwidth constraint)
  - to alter the traffic profile like traffic shaping or re-marking
- **Packets are discarded because**
  - limitation of buffer size (e.g., exceeding burst size in TB shaping)
  - traffic profile violation (meter detects exceeding contracted profile)
- **Type of queuing disciplines and dropping algorithms**
  - FIFO Queues, Fair Queues and Weighted Fair Queues (WFQ)
  - Schedulers
  - Algorithmic Droppers using RED and WRED dropping algorithms

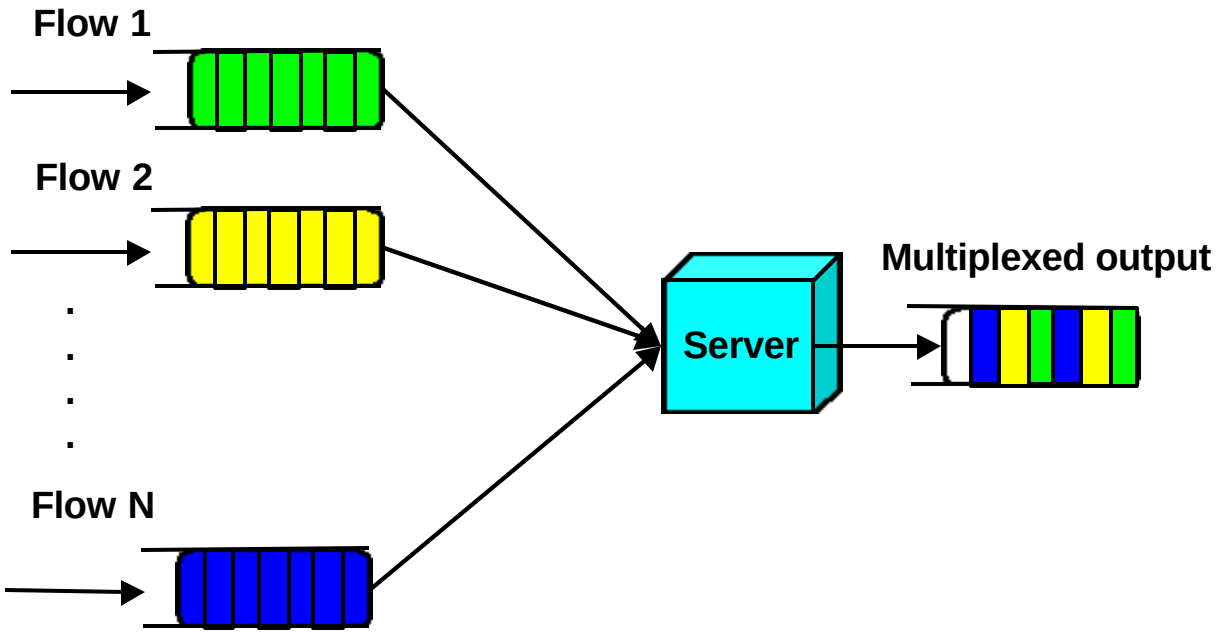
Queue Discipline - FIFO Queue

- **FIFO (first in, first out) Queue**
  - a finite size of queue to store packets
  - have one or more inputs to the queue
  - new arrival packet is enqueued at the tail of the queue
  - new departure packet is dequeued from the head of the queue



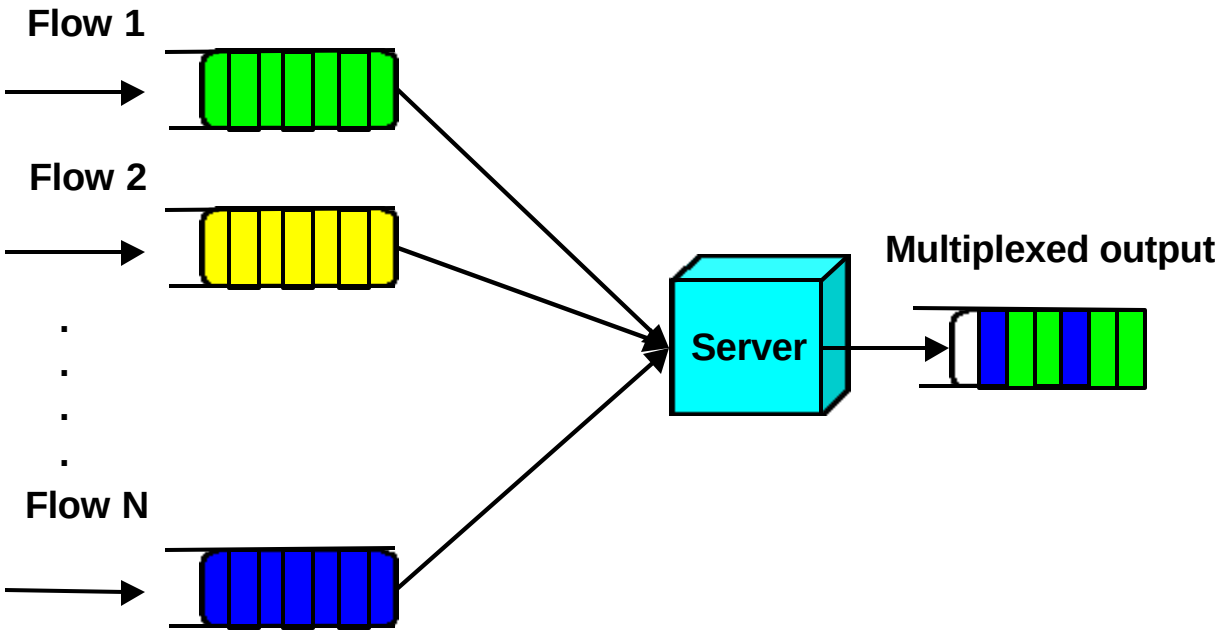
Queue Discipline - Fair Queue

- **Fair Queue**
  - maintain multiple FIFO queues at each input port
  - service each queue in in round robin fashion fairly
  - every input traffic gets equal treatment of service



Queue Discipline - Weighted Fair Queue

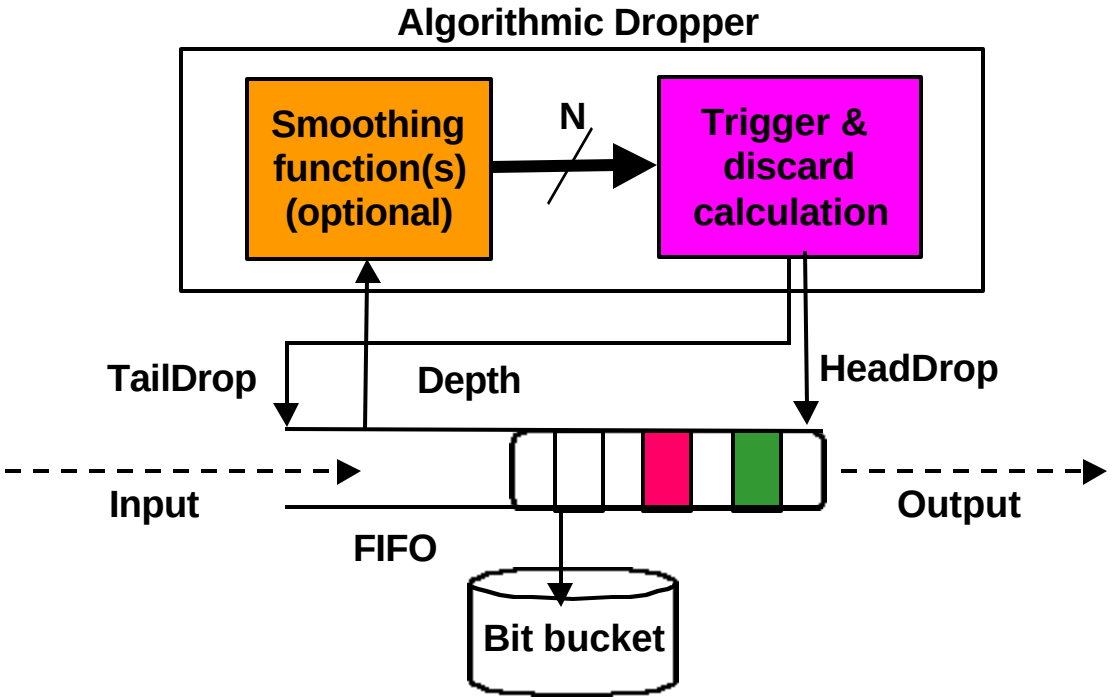
- **Weighted Fair Queue**
  - improved from fair queuing
  - consider the traffic through each queue and gives busier queues more capacity without completely shutting down less busy queue
  - consider the quality of service requested by each traffic flow and adjust queuing discipline accordingly
  - packets from low priority flows may be discarded during periods of congestion



Queue Element - Schedulers

- **Schedulers**
  - gate the departure of each packet based on a service discipline
  - has one or more inputs and exactly one output
  - each input has a set of parameters that affects the scheduling
- **Scheduling algorithm uses**
  - static parameters such as relative priority associated with each inputs
  - absolute token bucket parameters for maximum or minimum rates
  - parameters like DSCP associated with the packet currently present at its input
  - absolute time and/or local state.
- **Possible scheduling algorithms**
  - first come, first served (FCFS)
  - strict priority
  - weighted fair bandwidth sharing (e.g. WFQ)
  - rate-limited strict priority
  - rate-based

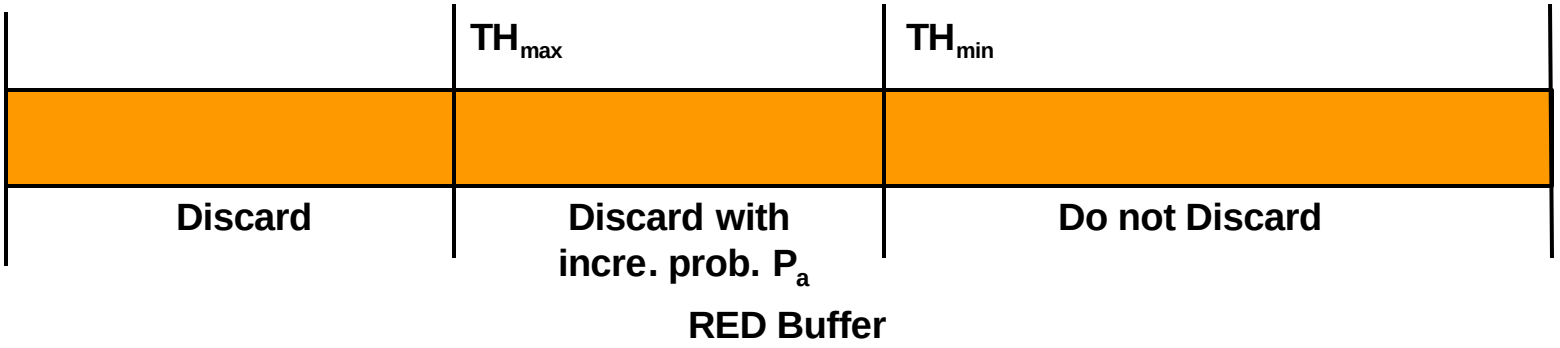
Queue Element - Algorithmic Dropper



- **One input, one output device that discard packets selectively**
- **Discarding is triggered either internally (profile violation) or externally (control signal)**
- **Decide whether to forward or discard a packet**
  - discard from the head, tail or other part of the queue
  - employ different dropping algorithms: head/tail dropping, RED, WRED

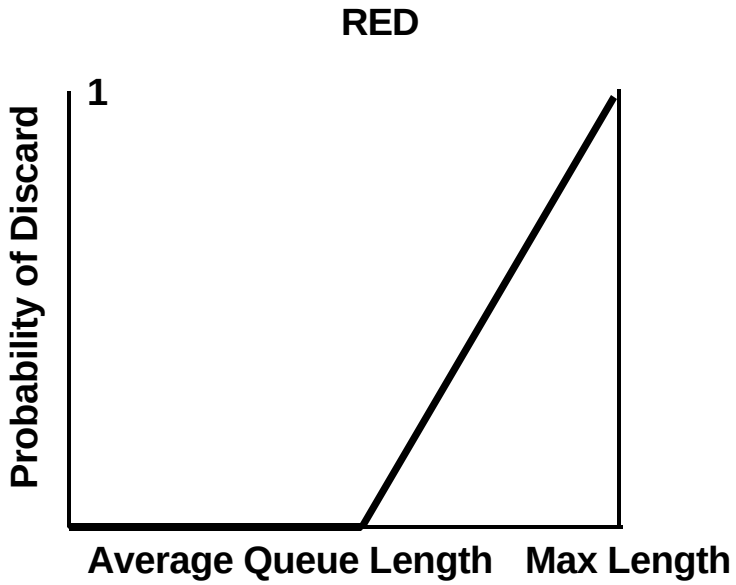
Dropping Algorithm - Random Early Detection (RED)

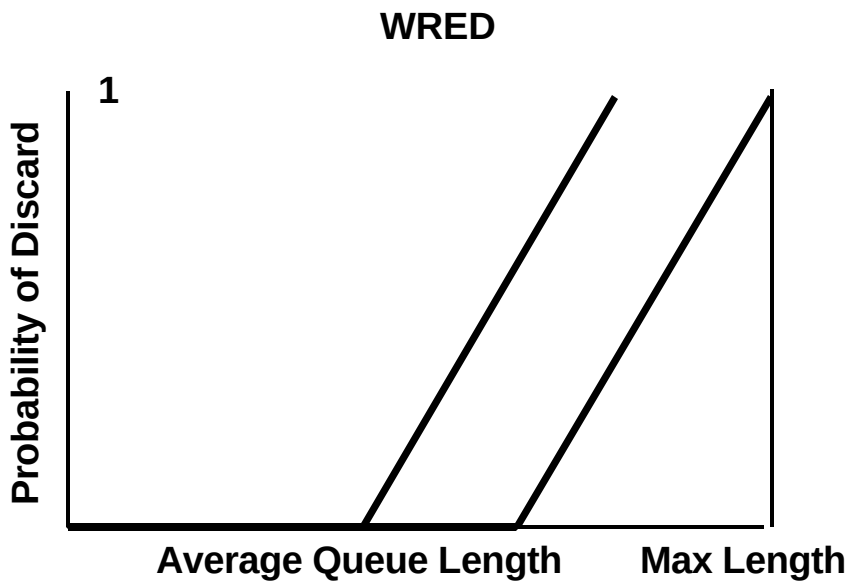
- **Facts**
  - Jacobson and Floyd first documented ([Floyd 93], RFC 2309) in 1993
  - use exponential average of the queue length and a random drop probability based on the linear function of average queue length
- **Design Goals [Stalling 98]**
  - congestion avoidance
  - global synchronization avoidance
  - avoidance of bias against bursty traffic
  - bound on average queue length
- **Results**
  - small bursts can pass through without elevated drop probability
  - large overload condition will trigger higher discard rates



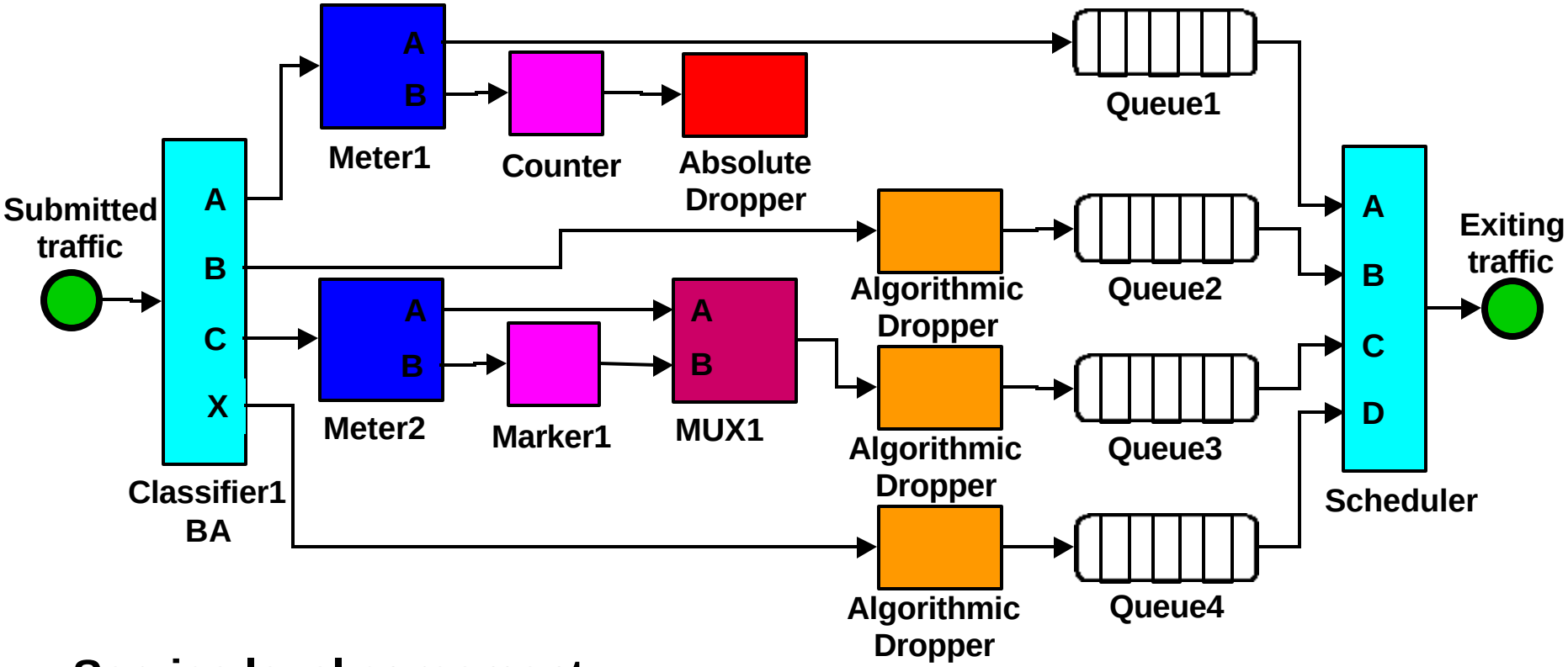
Dropping Algorithm - RED and Weighted RED

- **RED**
  - provide fair treatment for all the input traffic
  - discard packets randomly from the queue from all the traffic
- **WRED**
  - provide unfair or weighted behaviors for all the input traffics
  - packets are labeled with different discard priority



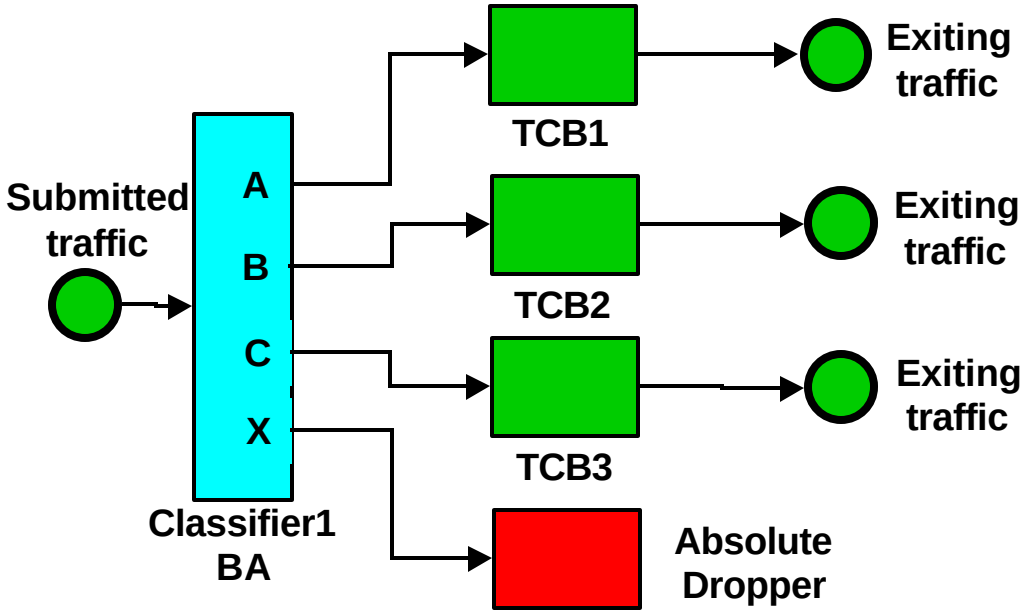


TCB Examples - BA Aggregate



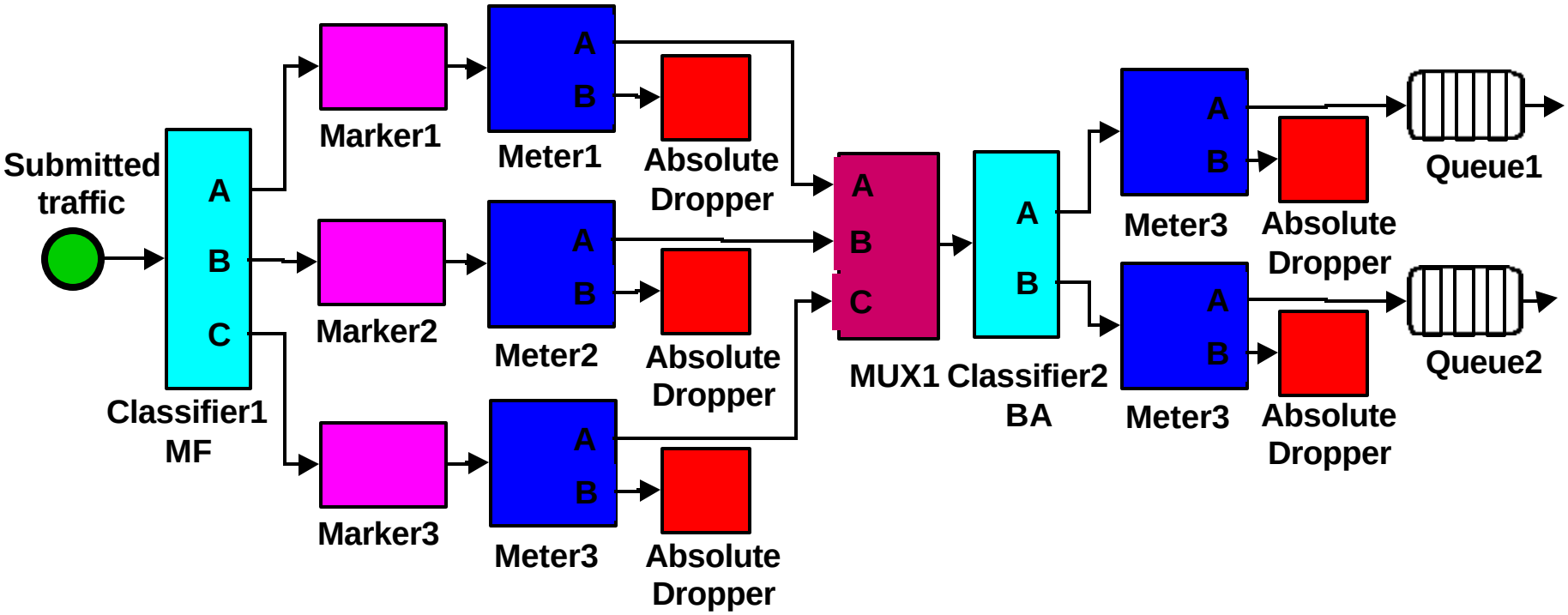
- **Service level agreement:**
  - A. DSCP = 001001, EF , profile 4, discard non-conforming
  - B. DSCP = 001100, AF11, profile 5, shape to profile, tail-drop when full
  - C. DSCP = 001101, AF21, profile 3, re-mark non-conforming to DSCP 001000, tail drop when full
  - X. DSCP = others , BE , no profile, apply RED-like dropping

TCB Example - Multiple Customers using BA TCBs



- **Design goal**
  - support multiple customer traffic over a single interface
  - use the TCB from previous slide as a building block
- **Data flow**
  - the customer traffics for A, B, and C are coming from the same interface
  - the BA classifier first differentiates the customer traffic into A, B and C
  - the BA classifier also discard the non-support customer traffic
  - the differentiated customer traffic is then passed to each TCB blocks

TCB Example - Microflow-based Service



- **Design goal**
  - differentiate 3 microflows to provide 2 service levels from a single customer
  - discard any traffic exceeding the bandwidth specified
- **Data flow**
  - use MF classifier to distinguish microflows from a customer traffic
  - for each microflow, meter the rate and discard any excess traffic
  - use BA classifier to provide 2 service levels and submit to queues for next hop

Summary

- **Differentiated Service (DiffServ)**
  - DiffServ started development in 1997 to provide a simple and scalable solution for QoS transmission on existing IP network and infrastructure
  - DiffServ is backward compatible with IP or non-DiffServ-aware networks
  - traffic should be differentiated for different forwarding behaviors or treatments
- **Router Architecture in Network**
  - ingress node classifies and marks the incoming packets
  - interior node routes and forwards based on PHB specified
  - egress node forwards and re-marks the packet for next DS domain
- **Router Elements**
  - fan out elements: classifier, meter
  - action elements: marker, shaper/dropper, multiplexor, counter, null action
  - queueing elements: FIFO queues, Weighted Fair Queue (WFQ), schedulers and algorithmic dropper
- **Configuration of TCB**
  - the configuration of router elements is based on the service level agreement
  - show three examples: BA traffic for a single customer, BA traffic for multiple customers, and Microflow based traffic for a single customers

References

- **[RFC 2474]** K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, RFC 2474, December 1998
- **[RFC 2475]** S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Services*, RFC 2475, December 1998.
- **[Bernet 00]** Y. Bernet, S. Blake, D. grossman, A. Smith, *Internet Draft - An Informal management Model for Diffserv Routers*, draft-ietf-diffserv-model-05.txt, November 2000
- **[Floyd 93]** S. Floyd and V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking, vol. 1, No. 4, August 1993
- **[Hashem 89]** E. Hashem, *Analysis of Random Drop for Gateway Congestion Control*, Technical Report LCS/TR-465, MIT, MA, 1989
- **[Huston 00]** G. Huston, *Internet Performance Survival Guide*, Wiley & Sons, 2000
- **[Stalling 98]** W. Stallings, *High-Speed Networks - TCP/IP and ATM Design Principles*, Prentice-Hall, 1998
- **[Stalling 00]** W. Stallings, *Data & Computer Communications*, Prentice-Hall, 2000