IP QOS
Theory and Practice

eng. Nikolay Milovanov
CCIE SP# 20094

QoS Architectures

QoS Architecture Models

- Best Effort Service
- Integrated Service
- Differentiated Service

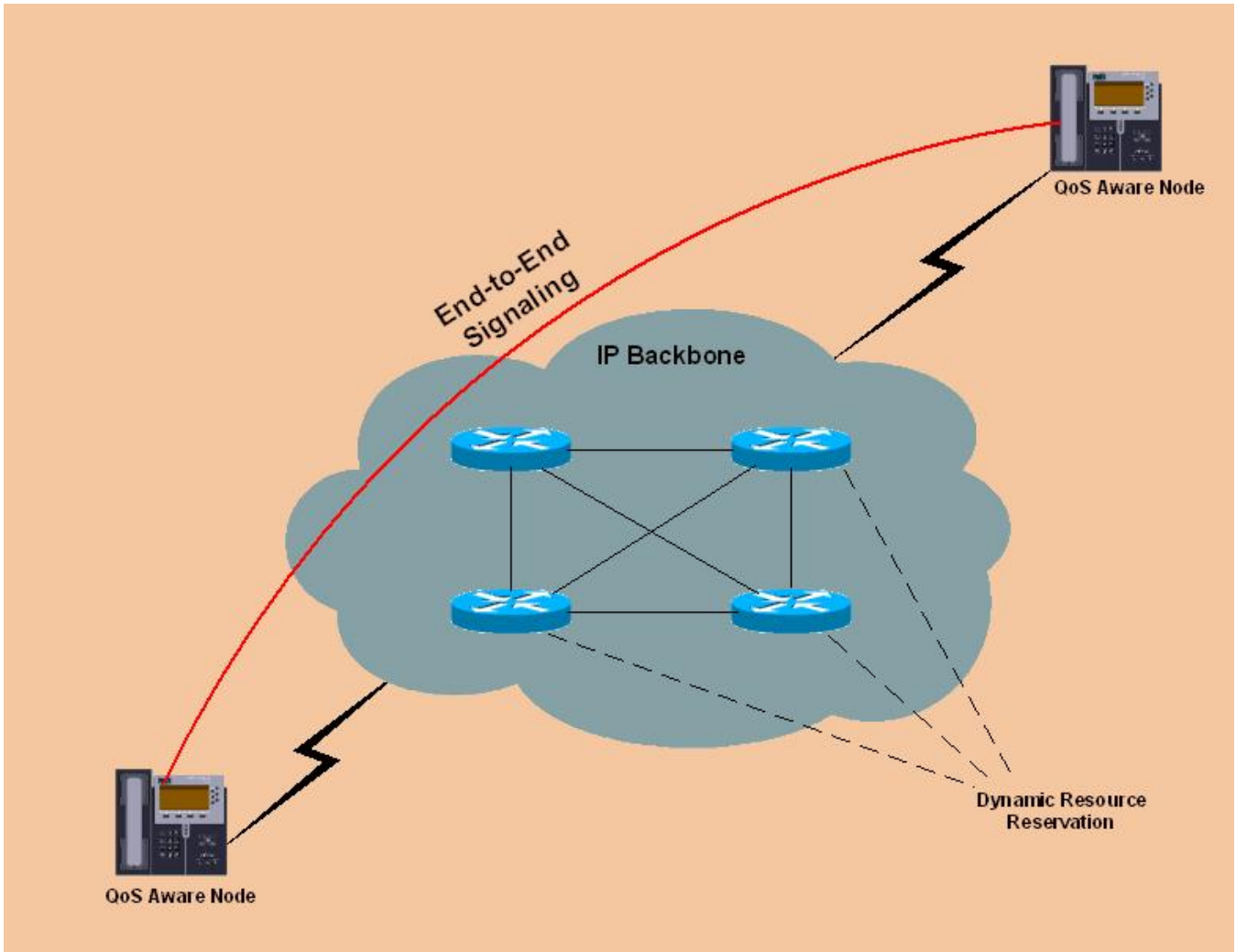
Best Effort Service

What exactly IP does:

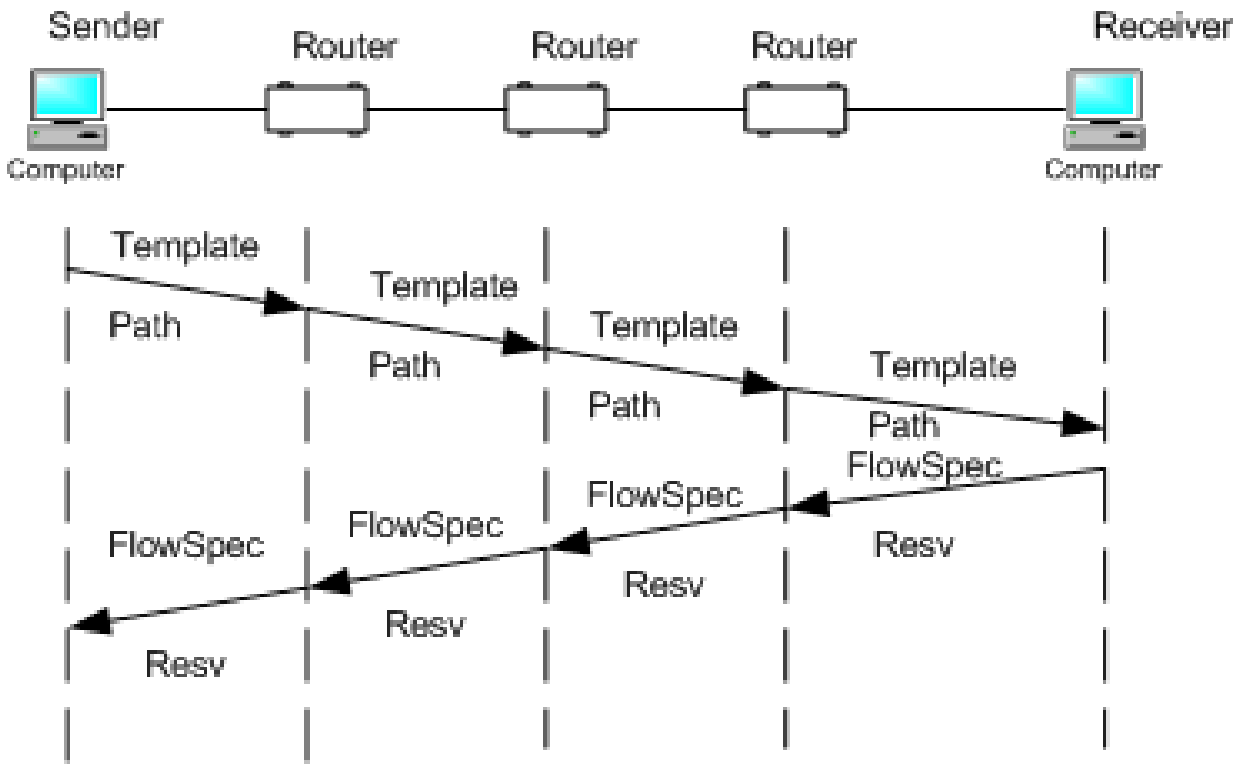
- All packets treated equally
- Unpredictable bandwidth
- Unpredictable delay and jitter

Integrated Services

IntServ (RFC1633)



IntServ – Resource Reservation through RSVP



Traffic Engineering

Without Traffic Engineering

Cars:

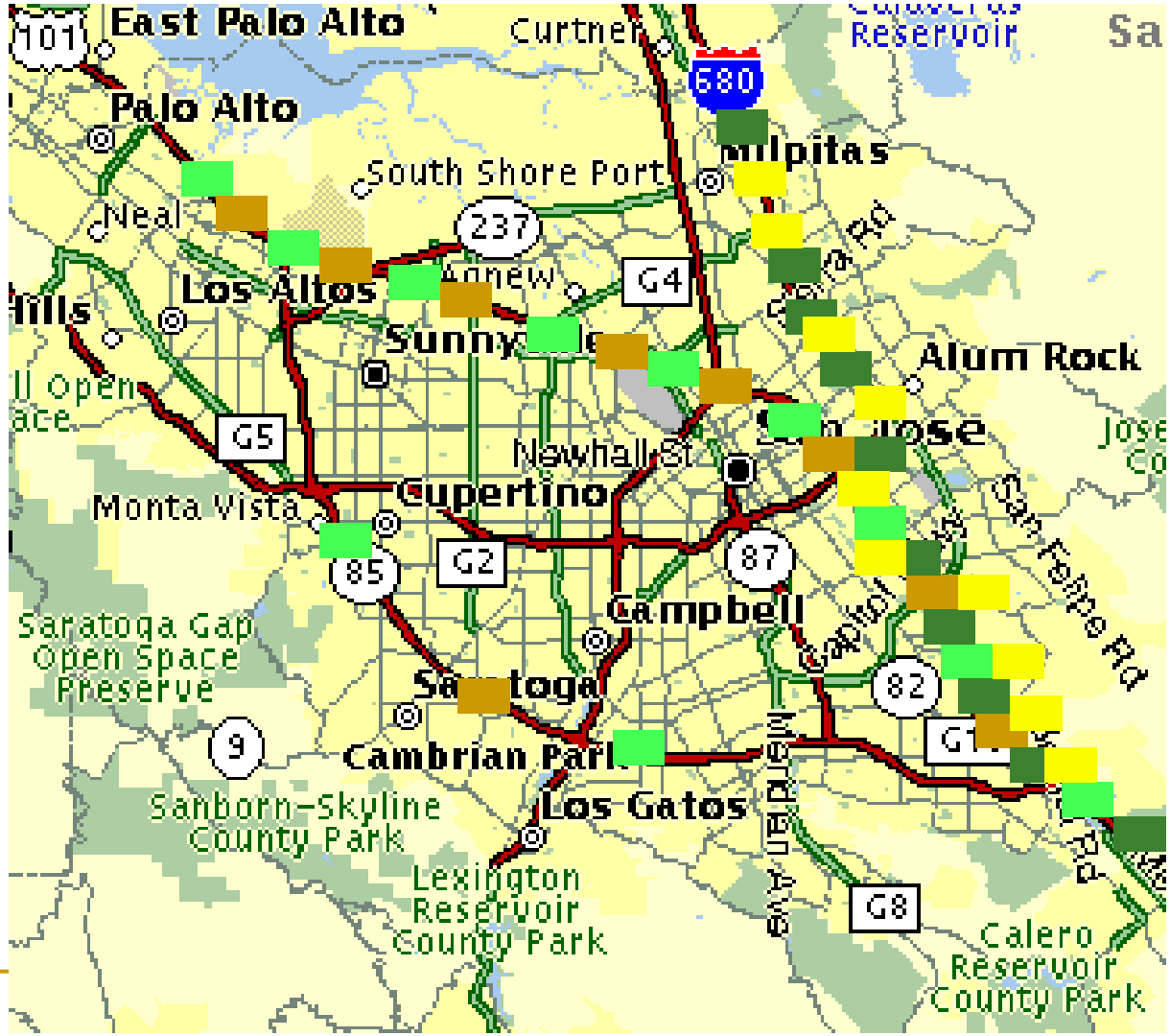
SFO-LAX SAN-SMF

LAX-SFO SMF-SAN

No Traffic Engineering

analogy

to Human Drivers



With Traffic Engineering

Cars:

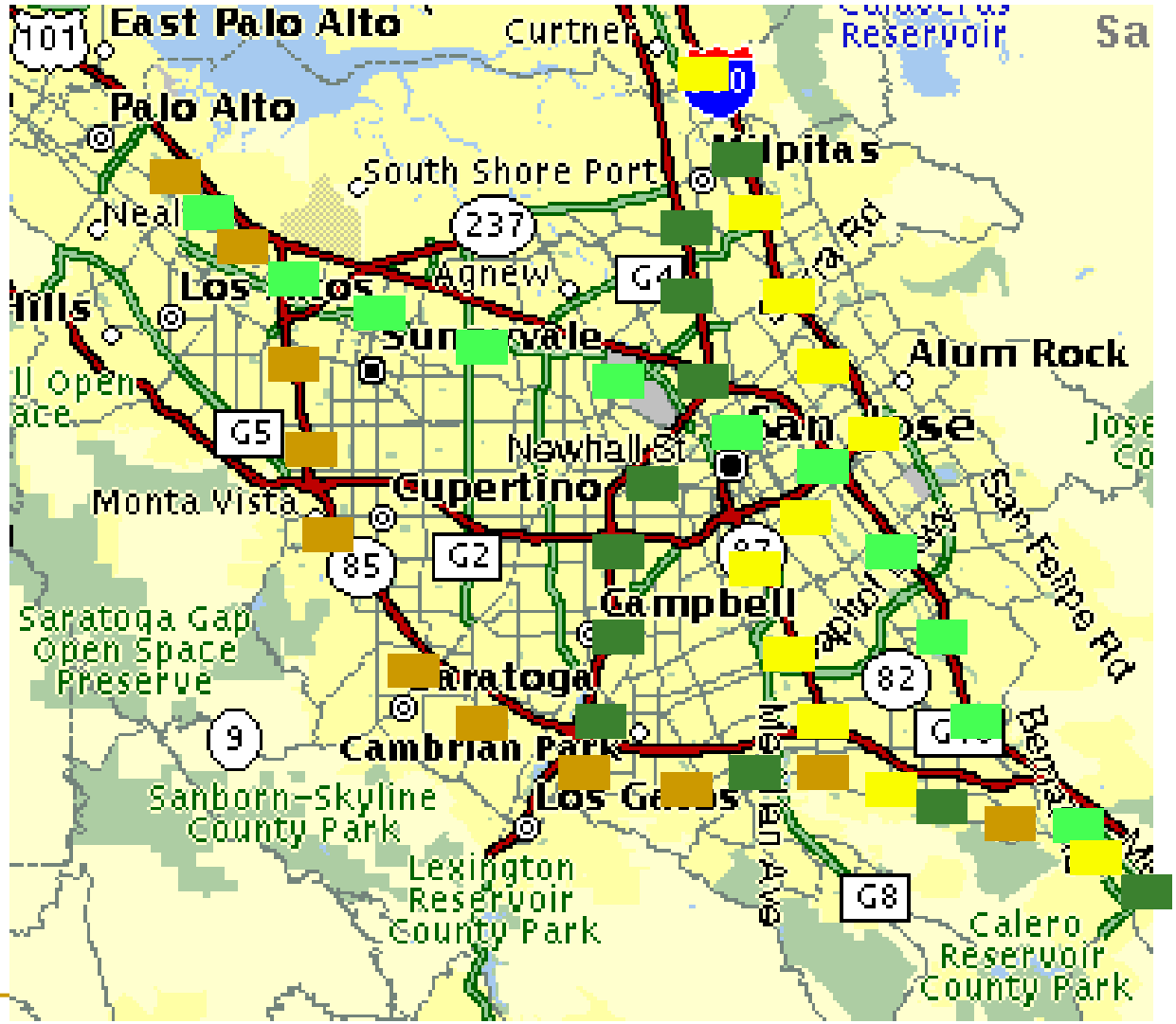
SFO-LAX

SAN-SMF

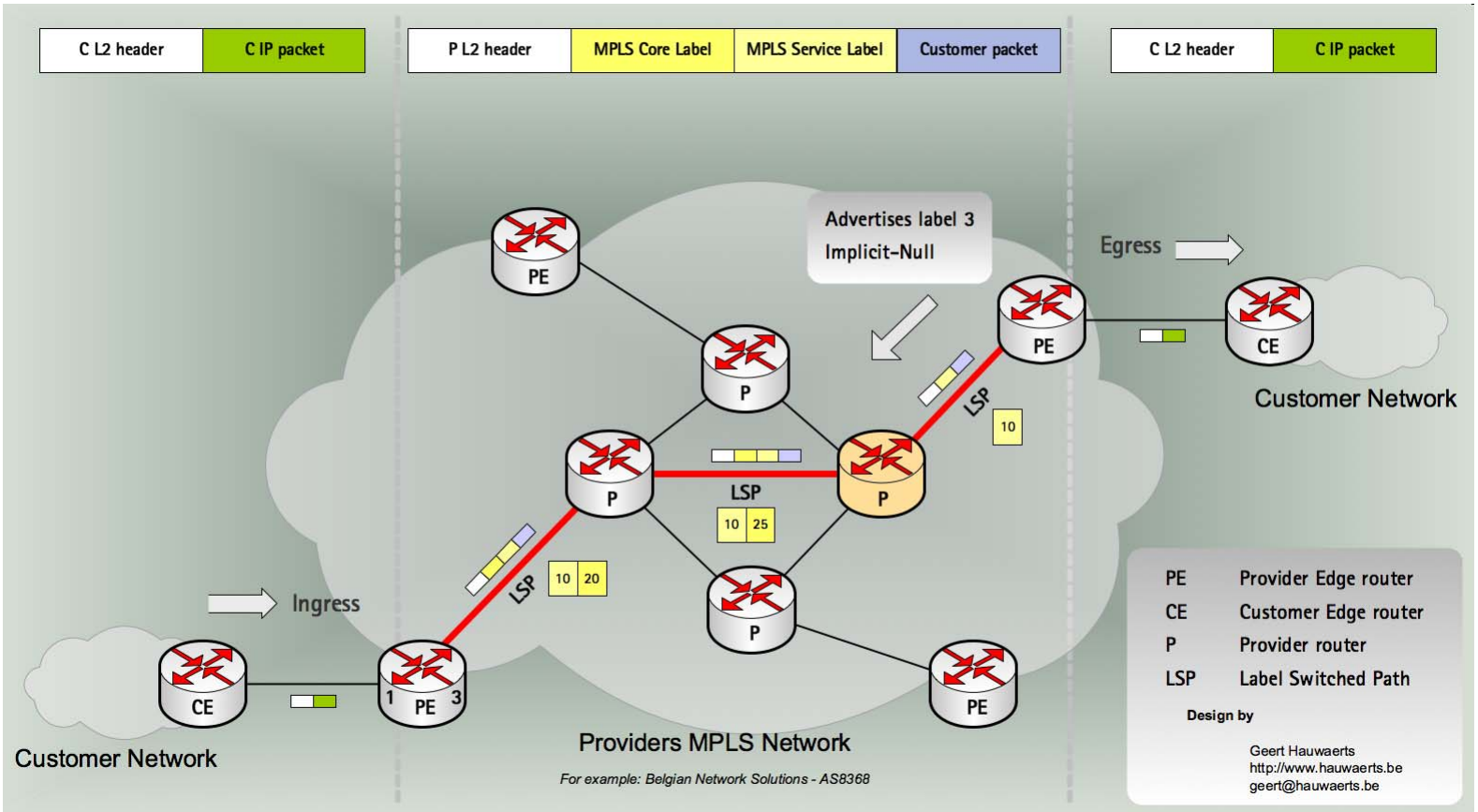
LAX-SFO

SMF-SAN

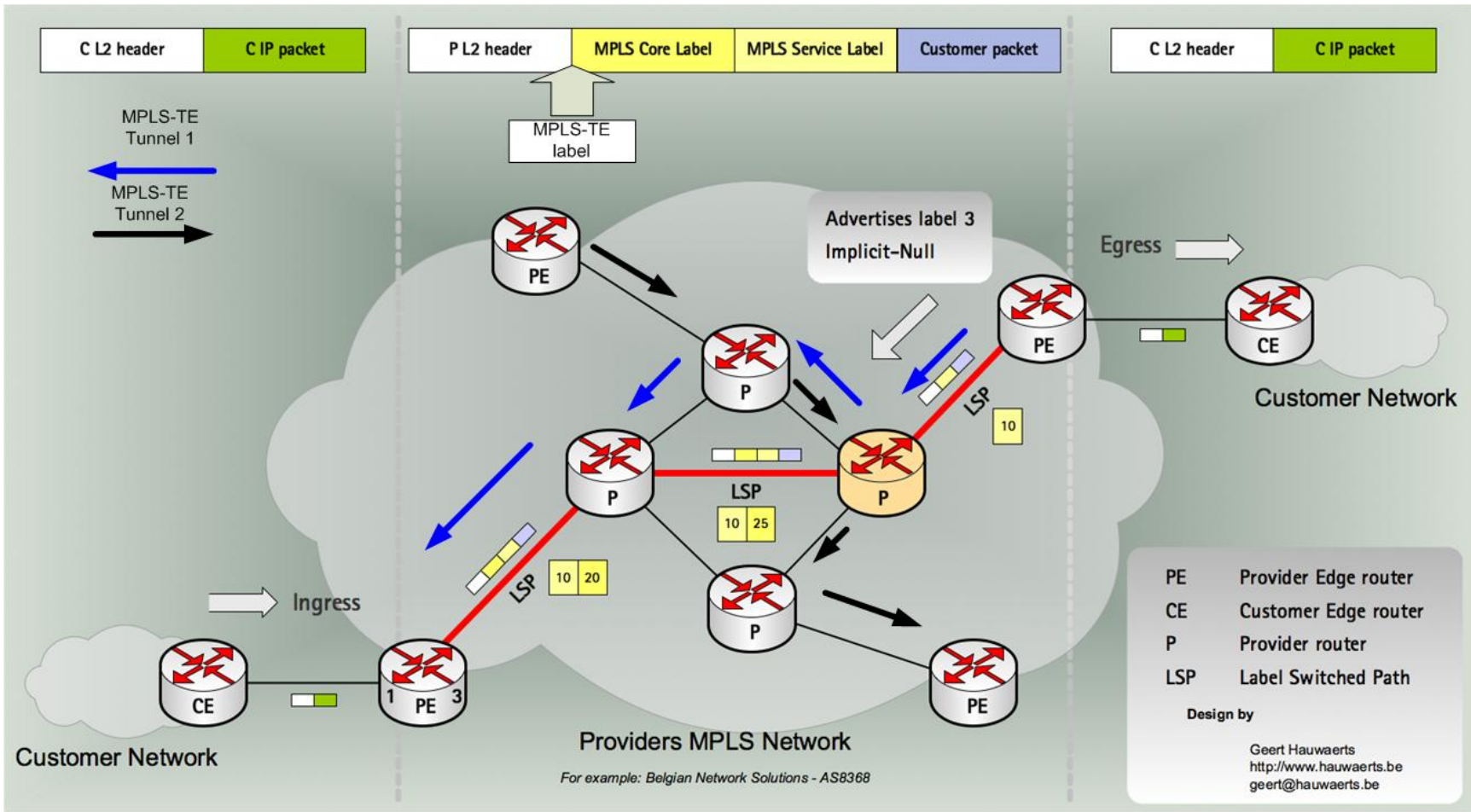
Traffic Engineering analogy to Auto Pilot



MPLS


C L2 header
C IP packet
P L2 header
MPLS Core Label
MPLS Service Label
Customer packet
C L2 header
C IP packet
Advertises label 3
Implicit-Null
Egress
PE
P
PE
CE
Customer Network
LSP
10
P
LSP
10 25
P
Ingress
LSP
10 20
P
CE
1 PE 3
PE
Customer Network
Providers MPLS Network
For example: Belgian Network Solutions - AS8368
PE Provider Edge router
CE Customer Edge router
P Provider router
LSP Label Switched Path
Design by
Geert Hauwaerts
http://www.hauwaerts.be
geert@hauwaerts.be

MPLS TE

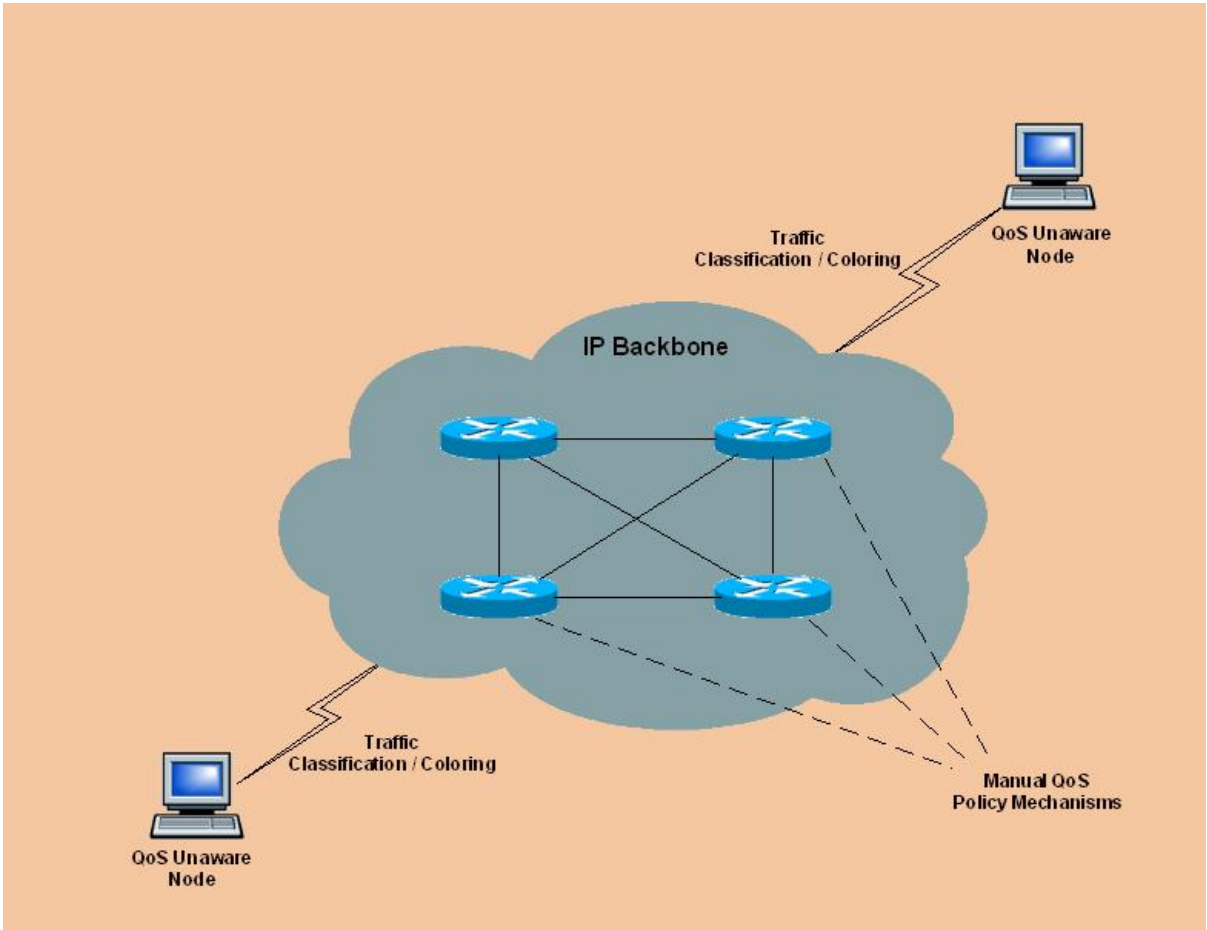

C L2 header
C IP packet
P L2 header
MPLS Core Label
MPLS Service Label
Customer packet
C L2 header
C IP packet
MPLS-TE label
MPLS-TE Tunnel 1
MPLS-TE Tunnel 2
Advertises label 3
Implicit-Null
Egress
PE
P
P
P
PE
CE
Customer Network
LSP
10
LSP
10 25
LSP
10 20
Ingress
CE
1 PE 3
PE
Customer Network
Providers MPLS Network
For example: Belgian Network Solutions - AS8368
PE Provider Edge router
CE Customer Edge router
P Provider router
LSP Label Switched Path
Design by
Geert Hauwaerts
http://www.hauwaerts.be
geert@hauwaerts.be

MPLS + TE

- To build MPLS TE tunnels you need
  - RSVP-TE to signal the tunnel, to reserve the resources and to distribute the labels
  - OSPF/ISIS TE extension to provide a way of describing the traffic engineering topology (including bandwidth and administrative constraints) and distributing this information within a given area.

Differentiated Services

DiffServ (RFC2474/2475)



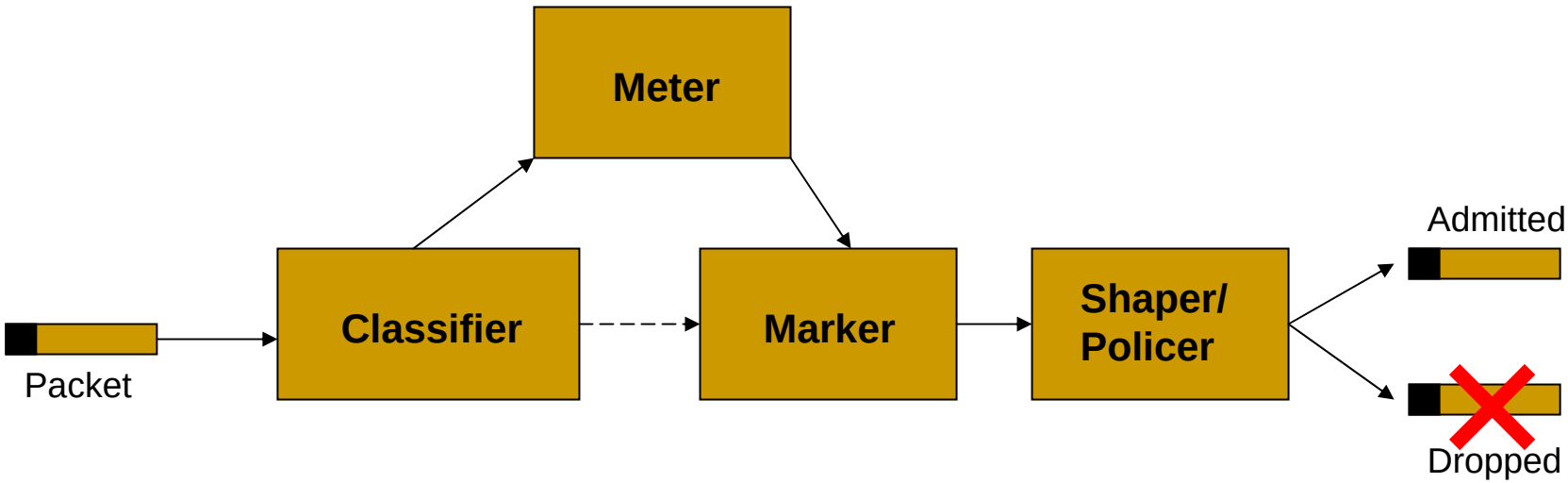
Traffic Classification

- Most fundamental QoS building block
- The component of a QoS feature that recognizes and distinguishes between different traffic streams
- Without classification, all packets are treated the same

Traffic Classification/ Admission Control Issues

- Always performed at the network perimeter
- Makes traffic conform to the internal network policy
- Marks packets with special flags (colors)
- Colors used afterwards inside the network for QoS management

Classification/ Admission Control Scheme



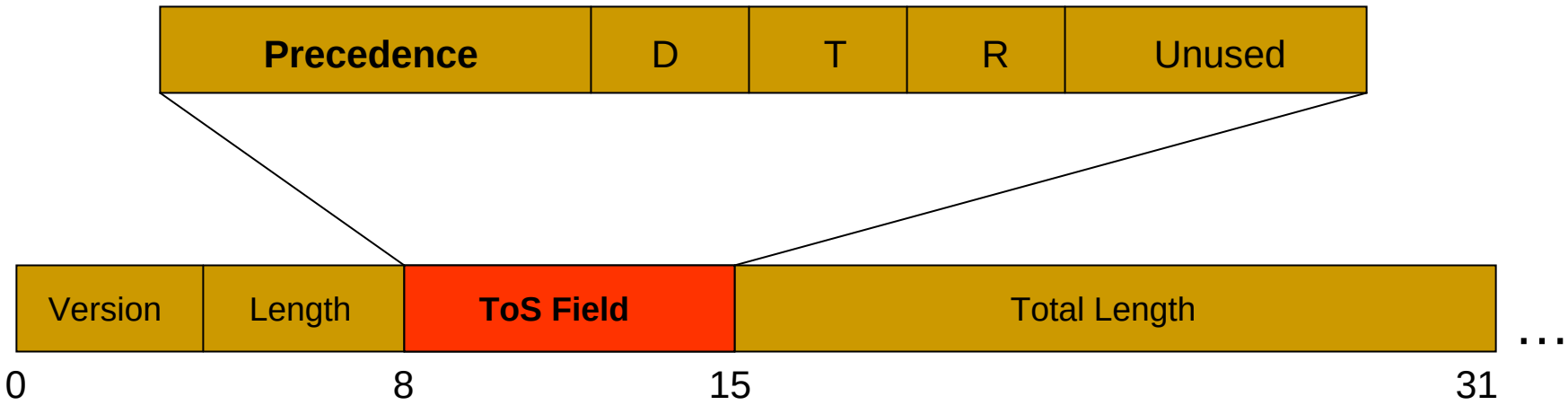
Classification Criteria

- IP header fields
- TCP/UDP header fields
- Routing information
- Packet Content (NBAR)
  i.e. HTTP, HTTPS, FTP, Napster etc.

Traffic Coloring Options

- IP Precedence
- DSCP
- QoS Group
- 802.1p CoS
- ATM CLP
- Frame Relay DE

Type-of-Service (RFC791)



	0	1
D	**Normal Delay**	**Low Delay**
T	**Normal Throughput**	**High Throughput**
R	**Normal Reliability**	**High Reliability**

IP Precedence Values

111	Network Control
110	Internetwork Control
101	Critical
100	Flash Override
011	Flash
010	Immediate
001	Priority
000	Routine

DSCP - Diffserv Code Point

DSCP (6 bits)	Unused

	Class 1	Class 2	Class 3	Class 4
Low Drop Precedence	001010	010010	011010	100010
Medium Drop Precedence	001100	010100	011100	100100
High Drop Precedence	001110	010110	011110	100110

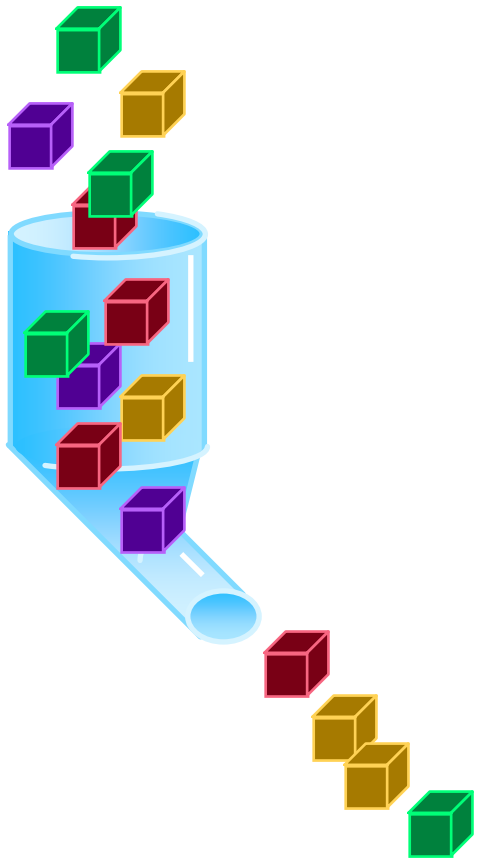
Classification mechanisms

- MQC (Modular Qos Command Line Interface)

- CAR (Commited Access Rate)

Congestion Management

Queuing



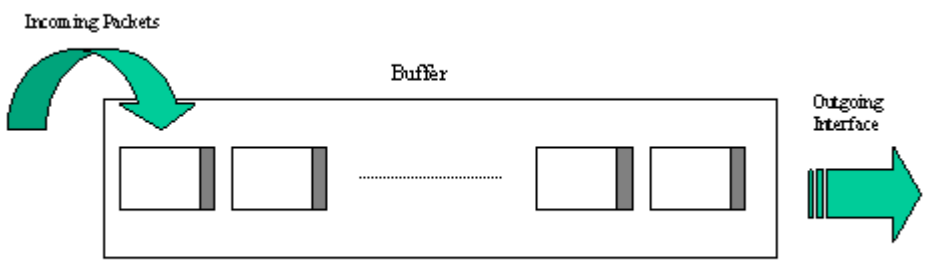
- Traffic burst may temporarily exceed interface capacity
- Without queuing this excess traffic will be lost
- Queuing allows bursty traffic to be transmitted without drops
- Queuing strategy defines order in which packets are transmitted through egress interface
- Queuing introduced additional delay which signals to adaptive flows (like TCP) to back off their throughput

Queuing Algorithms

- FIFO
- Priority (Absolute)
- Weighted Round Robin (WRR)
- Fair

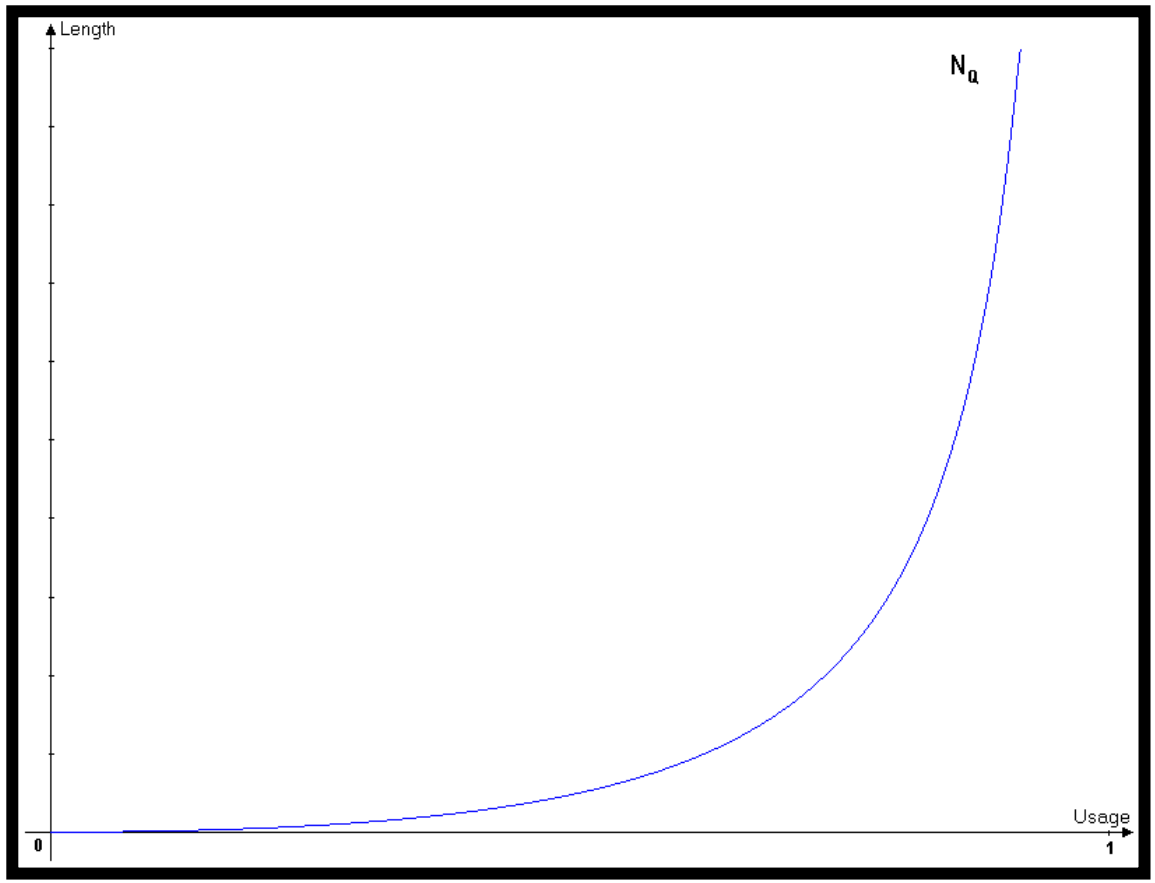
FIFO

- Simplest queuing method with the least CPU overhead
- No congestion control
- Transmits packets in the order of arrival
- High volume traffic can suppress interactive flows
- Default qu .e. Ethernet)



FIFO

FIFO average queue depth dependence on load



Absolute Priority Queuing

- Generic Priority Queuing
- Custom Queuing
- RTP Priority Queuing
- Low Latency Queuing (LLQ)

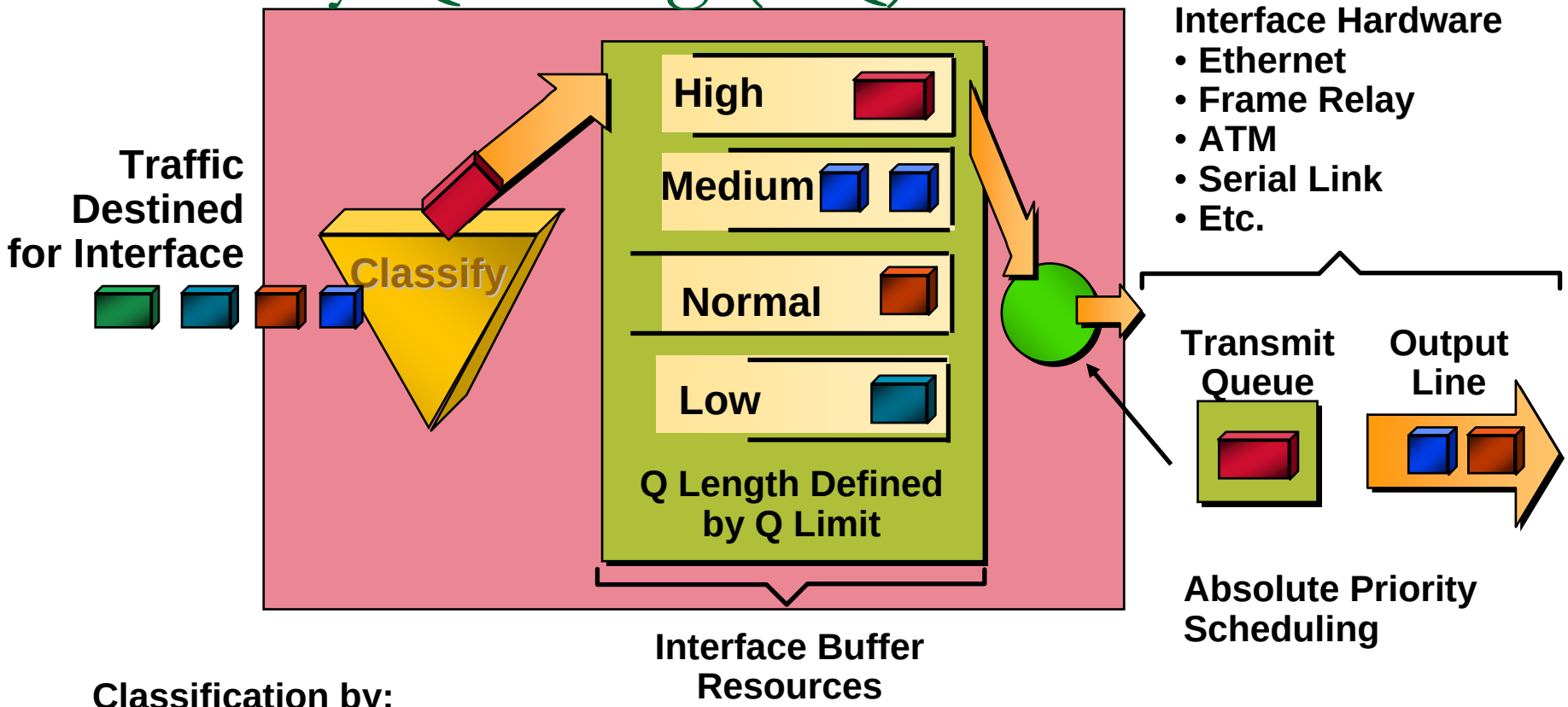
Simplest QoS Algorithm: Priority Queuing

- Stated requirement:
  - "If <application> has traffic waiting, send it next"
- Commonly implemented
  - Defined behavior of IP precedence

Priority Queuing Implementation Approach

- Identify interesting traffic
  - Access lists
- Place traffic in various queues
- Dequeue in order of queue precedence

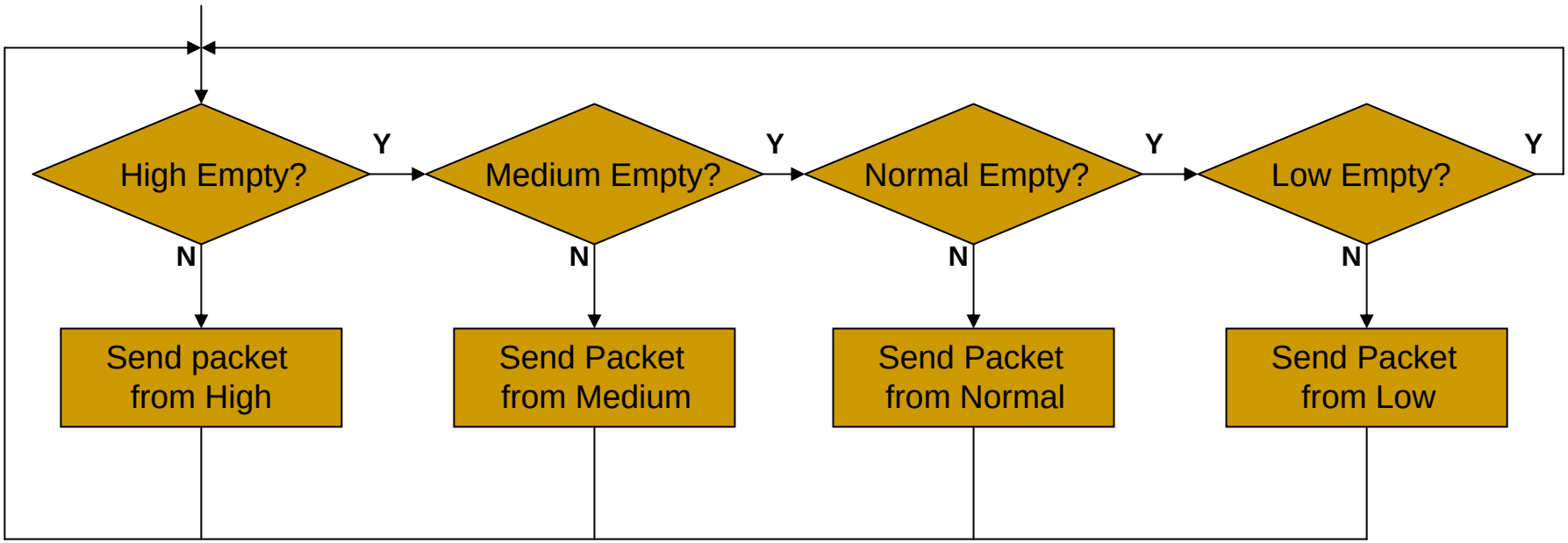
Priority Queuing (PQ)



Classification by:

- **Protocol (IP, IPX, AppleTalk, SNA, DecNet, Bridge, etc.)**
- **Incoming Interface (EO, SO, S1, etc.)**

Priority Queuing Scheme



Generic PQ Drawbacks

- Needs thorough admission control
- No upper limit for each priority level
- High risk of low priority queues` starvation effect

Generic PQ Configuration Sample

```
priority-list 1 protocol ip high tcp telnet
priority-list 1 protocol ip high list 100
priority-list 1 protocol ip medium lt 1000
priority-list 1 interface ethernet 0/0 medium
priority-list 1 default low
!
interface serial 2/1
   ip unnumbered loopback 0
   priority-group 1
!
access-list 100 permit tcp host 10.0.0.1 any eq http
```

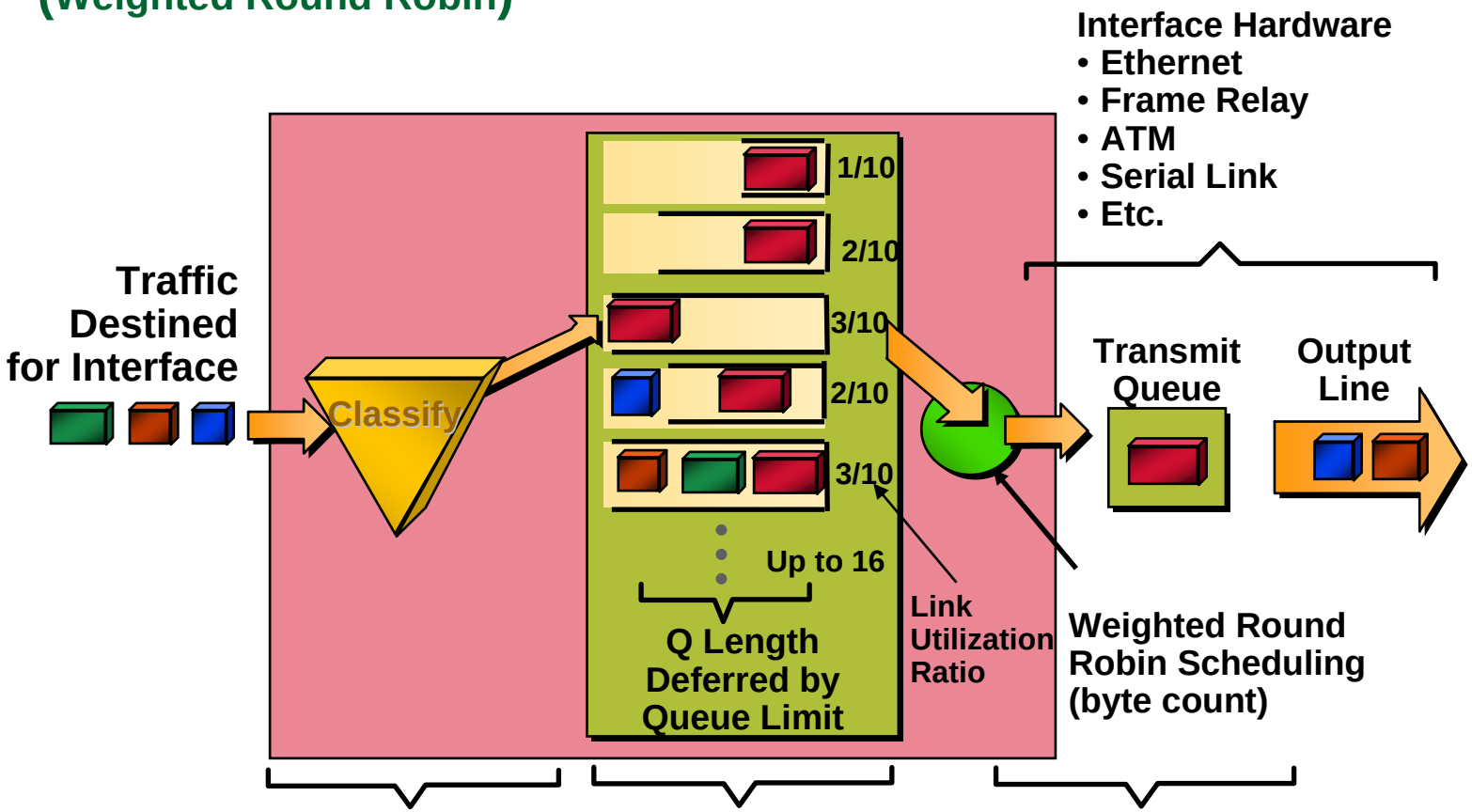
PQ Definition

PQ Attached to Interface

ACL definition

Custom Queuing (CQ)

(Weighted Round Robin)



Classification by:

- Protocol (IP, IPX, AppleTalk, SNA, DecNet, Bridge, etc.)
- Incoming interface (EO, SO, S1, etc.)

Interface Buffer Resources

Allocate Proportion of Link Bandwidth)

WRR Drawbacks

- Unpredictable jitter
- Fairness significantly depends on MTU and TCP window size
- Complex calculations to achieve desired traffic proportions

CQ Byte-count Calculus

Distribute bandwidth to 3 queues with proportion *x:y:z* and packet sizes q_x, q_y, q_z.

1. Calculate $a_x = x/q_x$, $a_y = y/q_y$, $a_z = z/q_z$.
2. Normalize and round a_x, a_y, a_z.
   $a_x' = round(a_x/min(a_x, a_y, a_z))$; $a_y' = round(a_y/min(a_x, a_y, a_z))$; $a_z' = round(a_z/min(a_x, a_y, a_z))$.
3. Convert obtained packet proportion into byte count
   $bc_x = a_x' \cdot q_x$; $bc_y = a_y' \cdot q_y$; $bc_z = a_z' \cdot q_z$.
4. Actual bandwidth share of *i*-th queue can be calculated with the following formula:

$$share_i = \frac{bc_i}{\sum_{j=1}^{n} bc_j} \cdot C$$

5. For better approximation obtained byte-counts can be multiplied by some positive whole number.

CQ Configuration Sample

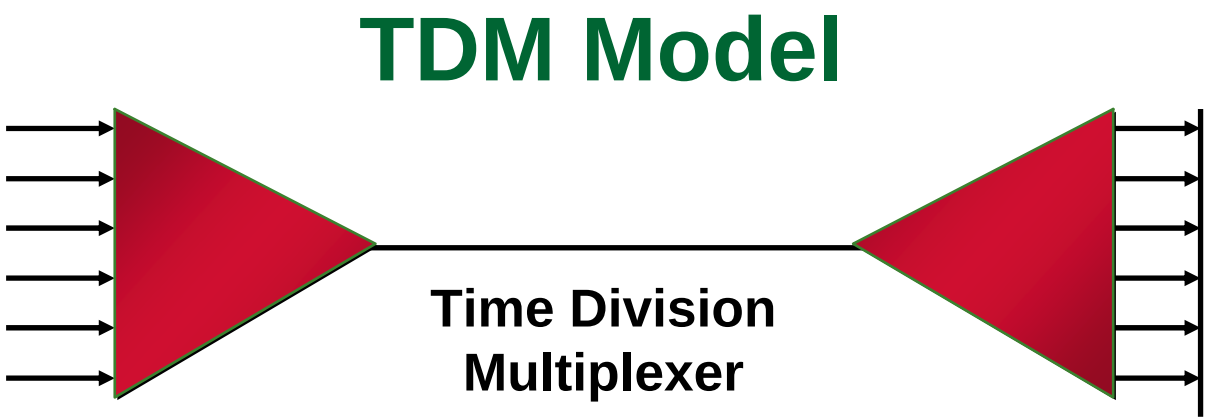
```
queue-list 1 protocol ip 1 tcp telnet
queue-list 1 protocol ip 2 list 100
queue-list 1 protocol ip 3 udp 53
queue-list 1 interface ethernet 0/0 4
queue-list 1 queue 1 byte-count 3000
queue-list 1 queue 2 byte-count 4500
queue-list 1 queue 3 byte-count 3000
queue-list 1 queue 4 byte-count 1500
queue-list 1 default 4
!
interface serial 2/1
   ip unnumbered loopback 0
   custom-queue-list 1
!
access-list 100 permit tcp host 10.0.0.1 any eq http
```

CQ List Definition

CQ Attached to Interface

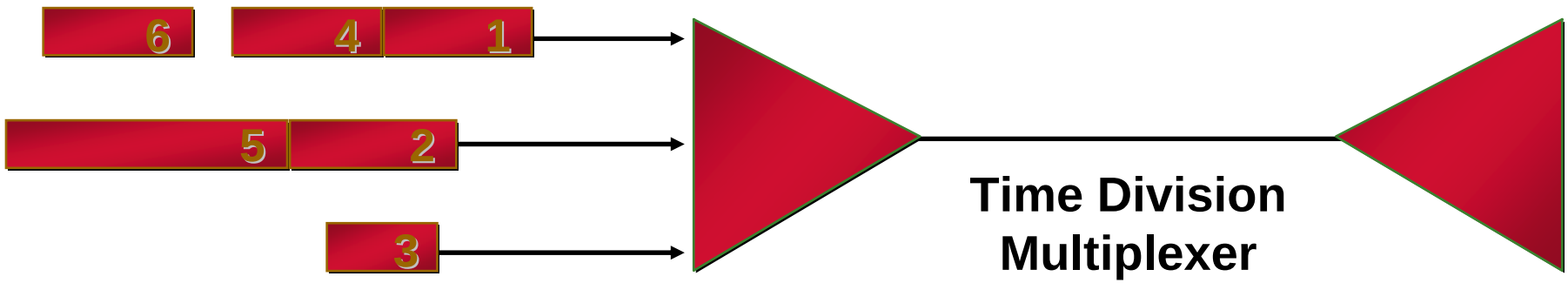
ACL Definition

"Bitwise Round Robin" Fair Queuing

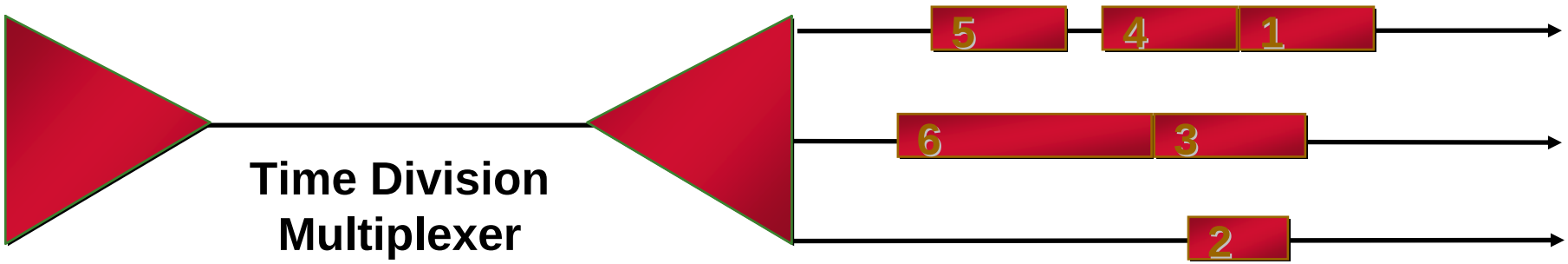


- Keshav, Demers, Shenker, and Zhang
- Simulates a TDM
- One flow per channel

TDM Message Arrival Sequence



TDM Message Delivery Sequence



Fair Queuing Algorithm

Employs virtual bit-by-bit round robin model (BRR)

BRR dynamics are described by the equation:

$$\frac{\partial R}{\partial t} = \frac{\mu}{N_{ac}(t)}$$

***i*-th packet from flow α arriving at time t_0 is services at time *t* :**

$$R(t_i^{\alpha}) = R(t_{i\ 0}^{\alpha}) + P_i^{\alpha}$$

Servicing of *i*-th packet from flow α will start at S_i^{α} and finish at F_i^{α} :

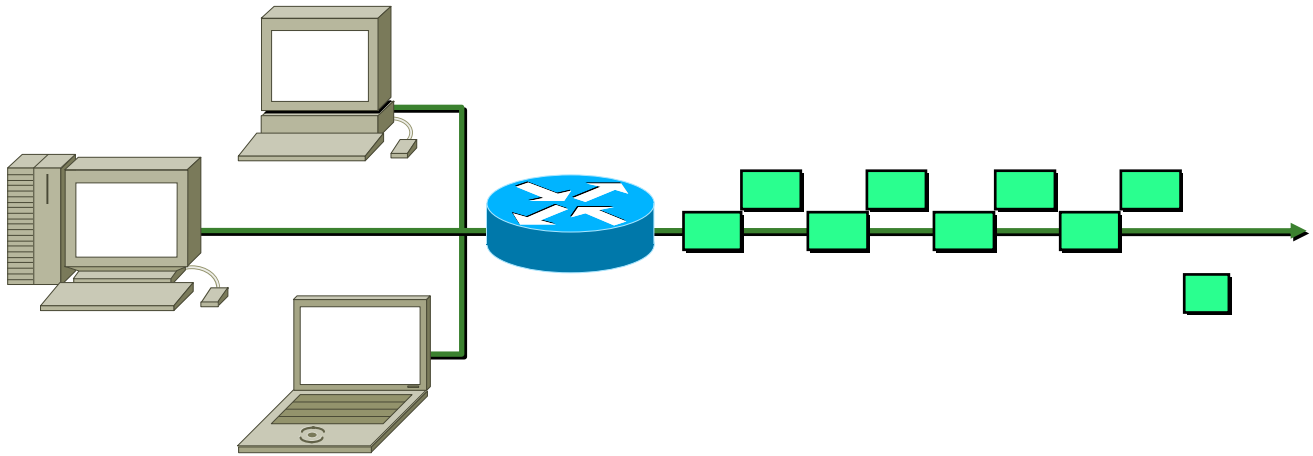
$$S_i^{\alpha} = MAX(F_{i-1}^{\alpha}, R(t_i^{\alpha})) \qquad F_i^{\alpha} = S_i^{\alpha} + P_i^{\alpha}$$

Additional δ parameter is added for priority assignment to inactive flows :

$$B_i^{\alpha} = MAX(F_{i-1}^{\alpha}, R(t_i^{\alpha}) - \delta)$$

Packets are ordered for transmission according to B_i^{α} values.

Fair Queuing Approach



- Enqueue traffic in the sequence the TDM would deliver it
- As a result, be as fair as the TDM

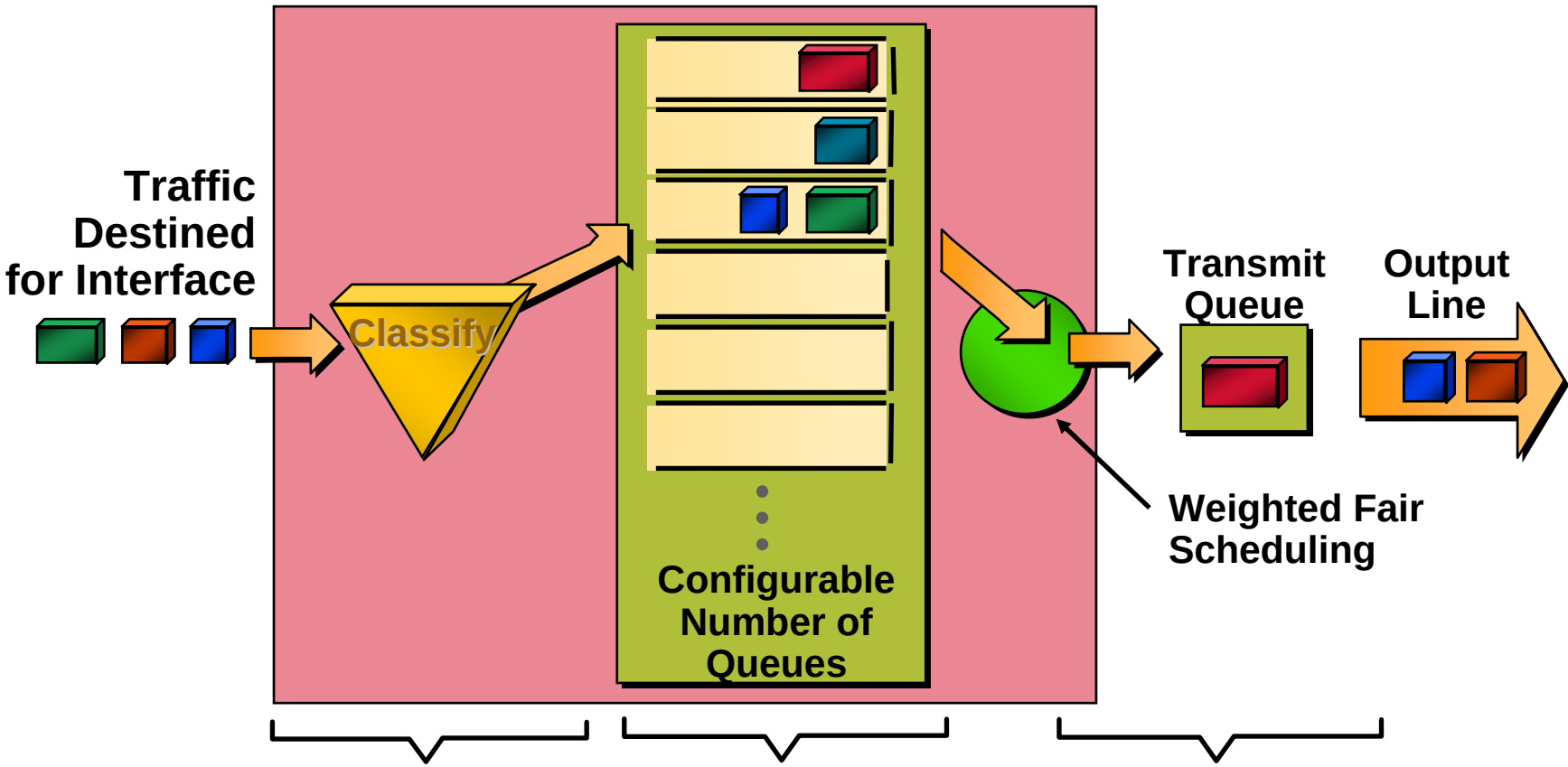
Effects of Fair Queuing

- Low-bandwidth flows get
  - As much bandwidth as they can use
  - Timely service
- High-bandwidth flows
  - Interleave traffic
  - Cooperatively share bandwidth
  - Absorb latency

What Weighting Does

- In TDM
  - Channel speed determines message “duration”
- In WFQ
  - Multiplier on message length changes simulated message “duration”
- Result:
  - Flow’s “fair” share predictably unfair

Weighted Fair Queuing (WFQ)



Flow-Based Classification by:
- Source and destination address
- Protocol
- Session identifier (port/socket)

Interface Buffer Resources

Weight Determined by:
- Requested QoS (IP Procedure, RSVP)
- Frame Relay FECN, BECN, DE (For FR Traffic)
- Flow throughput (weighted-fair)

Weighted Fair Queuing (WFQ)

- Fair bandwidth per flow allocation
- Low delay for interactive applications
- Protection from ill-behaved sources

Weighted Fair Queuing (WFQ)

Flow classified by the following fields:

- Source address
- Source port
- Destination address
- Destination port
- ToS

Weight of each flow (queue) depends on ToS:

weight = 1/(precedence+1)

Bandwidth distributed in 1/weight proportions

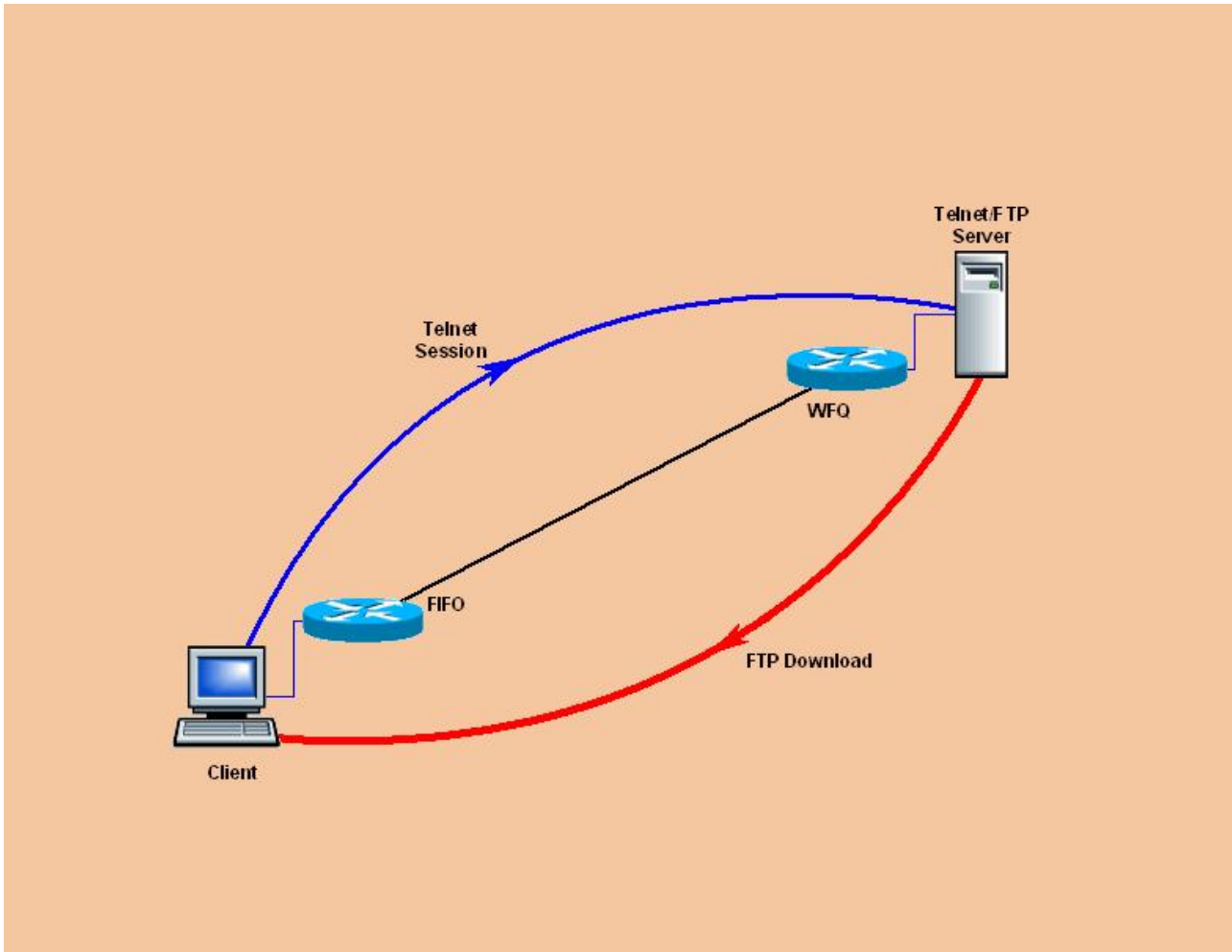
Weighted Fair Queuing (WFQ)

- Packets are ordered according to the expected virtual departure time of their last bit.
- Low volume flows have preference over high volume transfers.
- Low volume flow is identified as using less than its share of bandwidth.
- The special queue length threshold value is established, after which only low volume flows can enqueue. All the packets, that belong to high volume flows are dropped.

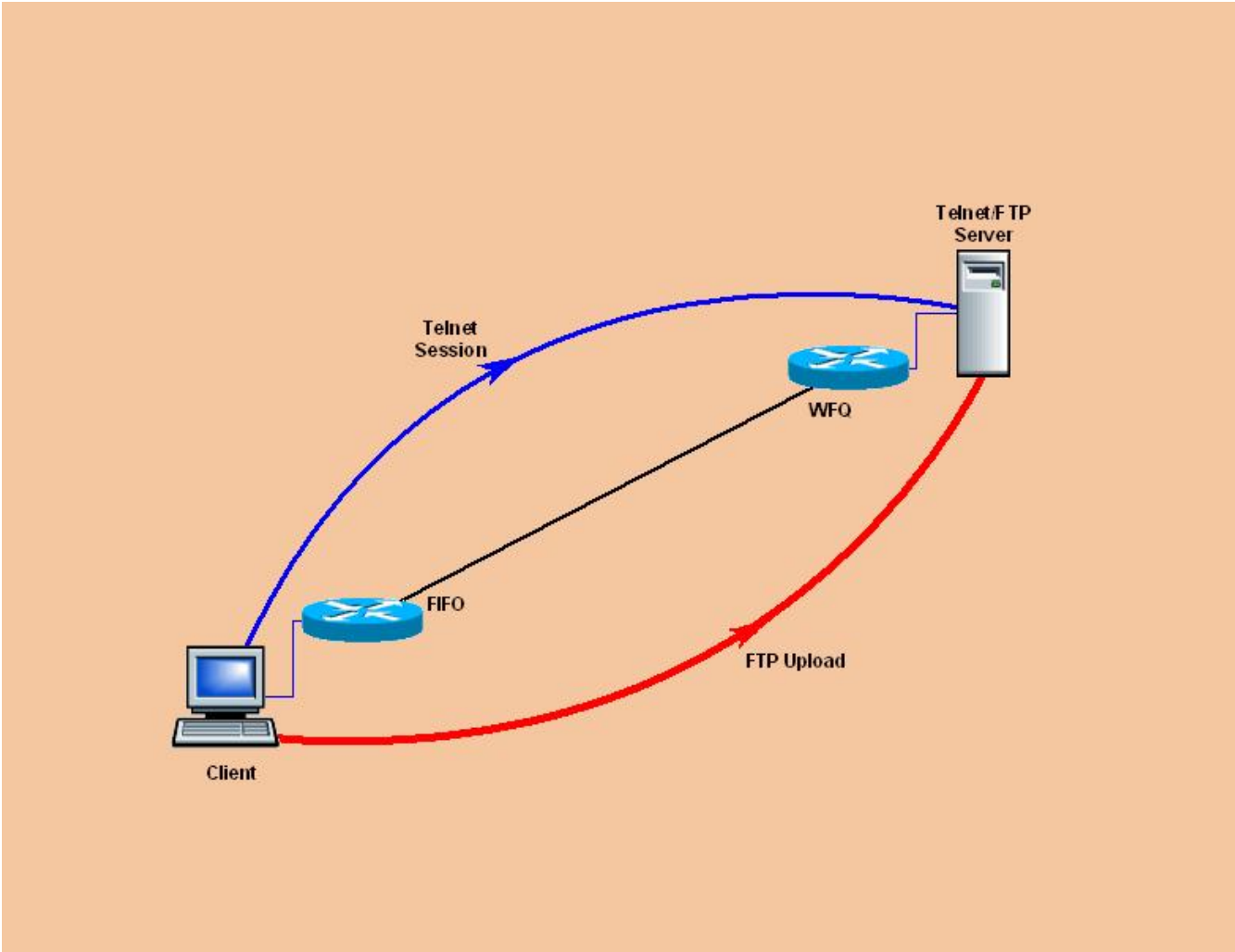
Drawbacks of Weighted Fair Queuing

- Requires more sorting than other approaches

Weighted Fair Queuing (WFQ)

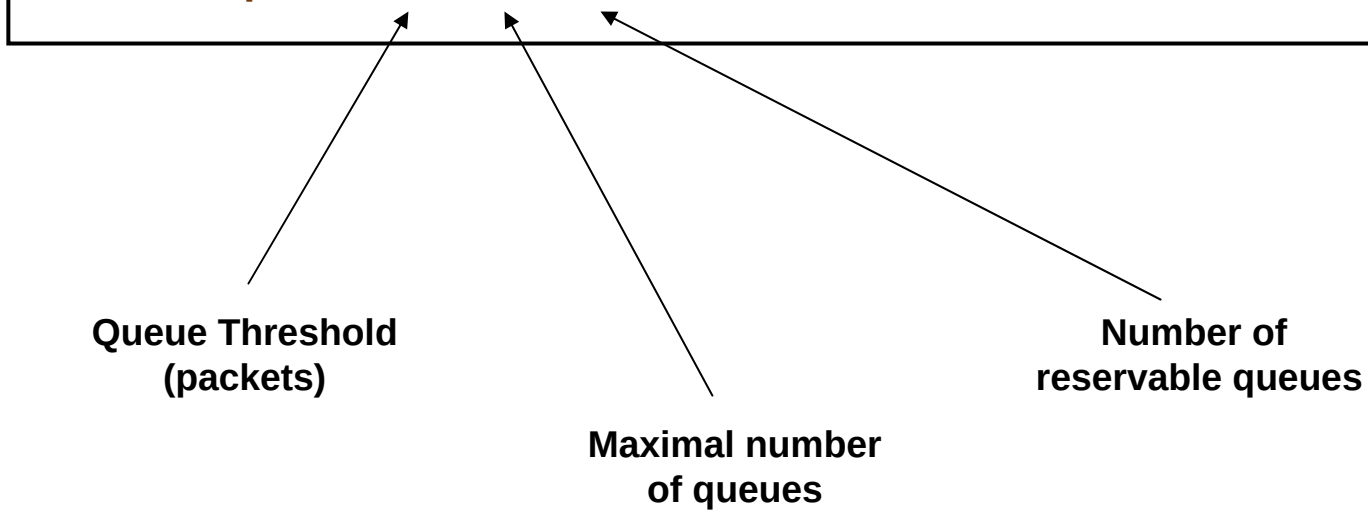


Weighted Fair Queuing (WFQ)



WFQ Configuration Sample

```
interface serial 2/1
   ip unnumbered loopback 0
   fair-queue 32 128 0
```

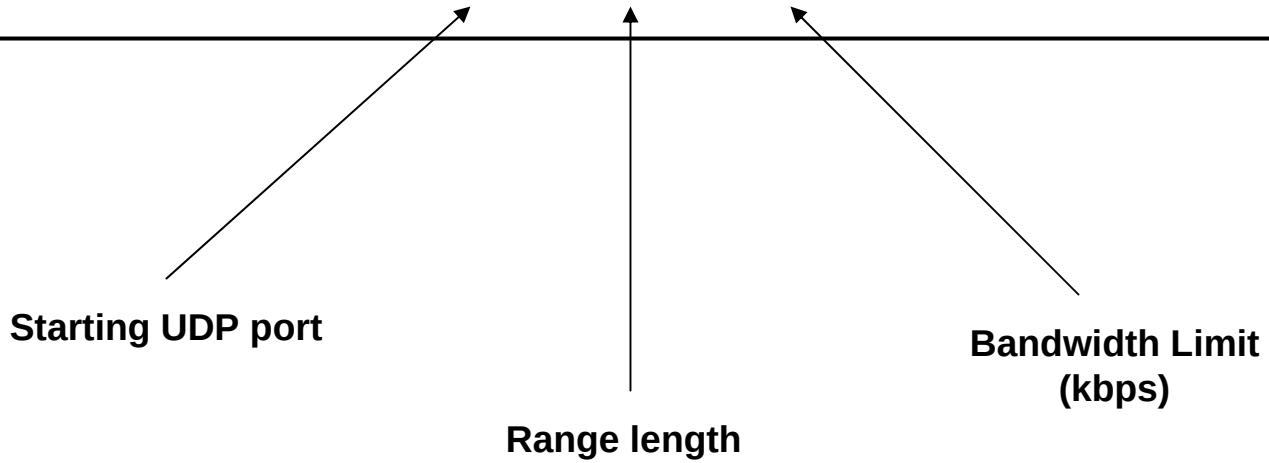


RTP Priority Queuing

- Classifies only by UDP port range
- Only even ports from the range are classified
- Establishes upper limit via integrated policer
- Excess traffic dropped during congestion periods
- RTP PQ has priority over LLQ

RTP PQ Configuration Sample

```
interface serial 2/1
   ip unnumbered loopback 0
   ip rtp priority 16384 16383 256
```



Low Latency Queuing (LLQ)

- Implemented using MQI
- Very rich classification criteria (class-map)
- Establishes upper limit via integrated policer
- Excess traffic dropped during congestion periods

LLQ Configuration Sample

```
class-map match-all voice
   match access-group name voip
!
policy-map llq
   class voip
         priority 30
   class class-default
         fair-queue 64
!
interface serial 2/1
   ip unnumbered loopback 0
   service-policy output llq
!
ip access-list extended voip
   permit ip host 10.0.0.1 any
```

Class definitions

LLQ policy definition

LLQ Policy attached to interface

ACL definition

Class Based WFQ (CBWFQ)

- Based on the same algorithm as WFQ
- Weights can be manually configured
- Allows to easily specify guaranteed bandwidth for a class
- Configuration based on Cisco MQI

CBWFQ Configuration Sample

```
class-map match-all premium
    match access-group name premium-cust
class-map match-all low-priority
    match protocol napster
!
policy-map cbwfq-sample
    class premium
          bandwidth 512
    class low-priority
          shape average 128
          shape peak 512
    class class-default
          fair-queue 64
!
interface serial 2/1
    ip unnumbered loopback 0
    max-reserved-bandwidth 85
    service-policy output cbwfq-sample
!
ip access-list extended premium-cust
    permit ip host 10.0.0.1 any
```

Class definitions

Qos policy definition

QoS Policy attached to interface

ACL definition

CBWFQ Configuration Sample

Hierarchical Design

```
class-map match-all premium
    match access-group name premium-cust
class-map match-all voice
    match ip precedence flash
!
policy-map total-shaper
    class class-default
          shape average 1536
          service-policy class-policy
policy-map class-policy
    class premium
          bandwidth 512
    class voice
          priority 64
    class class-default
          fair-queue 128
```

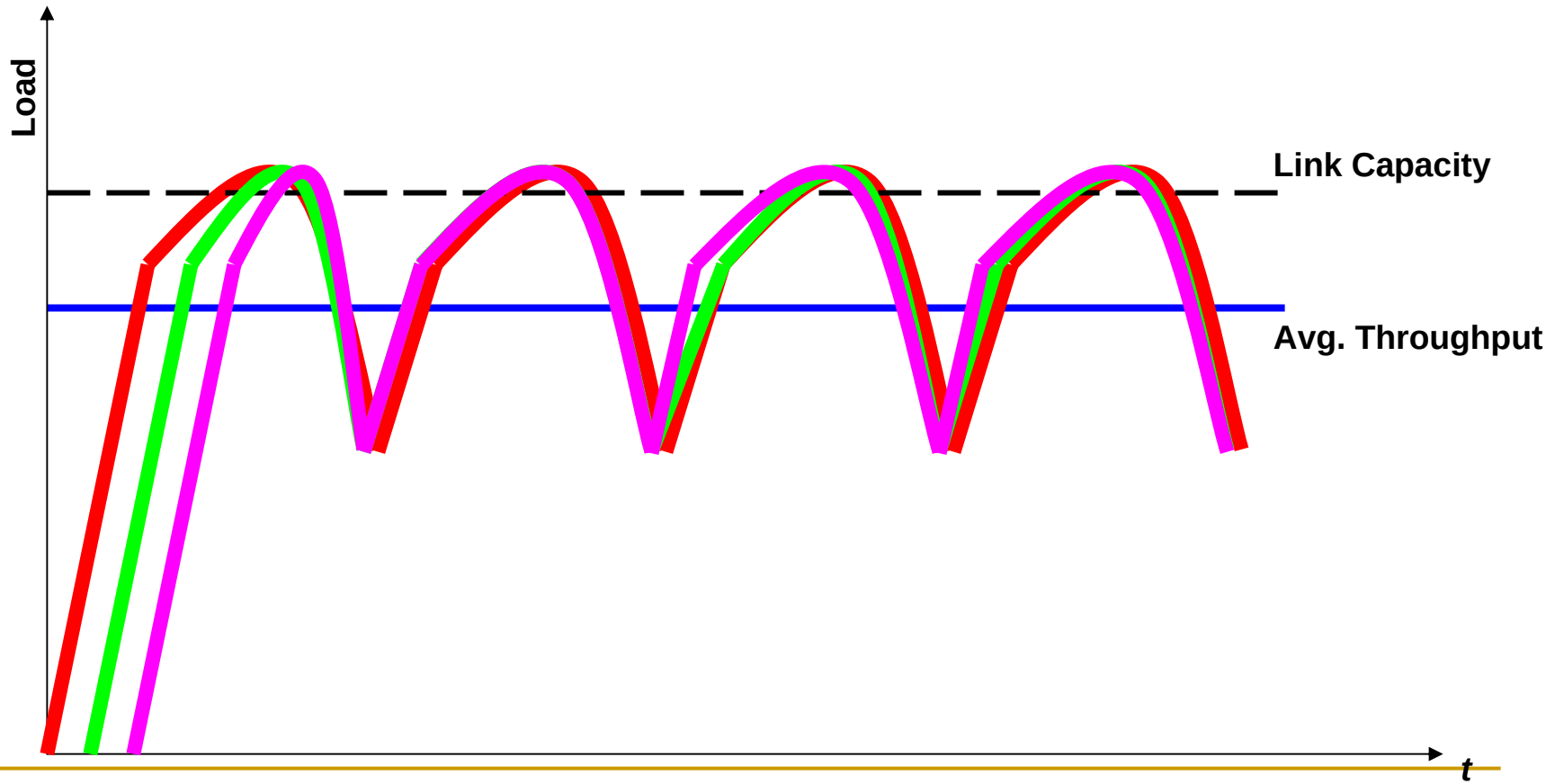
```
interface fastethernet 1/0
    ip unnumbered loopback 0
    max-reserved-bandwidth 85
    service-policy output total-shaper
!
ip access-list extended premium-cust
    permit ip host 10.0.0.1 any
```

Hierarchical CBWFQ Limitations

- Only two levels of hierarchy are supported
- *set* command not supported in child policy
- Shaping allows only in parent policy
- LLQ can be configured only either in child or parent policies but not in both
- FQ allowed only in child policy

Congestion Avoidance

Global Synchronization Effect



Tail Drop and TCP Flow Control

- Packet drops from all TCP sessions simultaneously
- High probability of multiple drops from the same TCP session
- Uniformly distributed drops from high volume and interactive flows

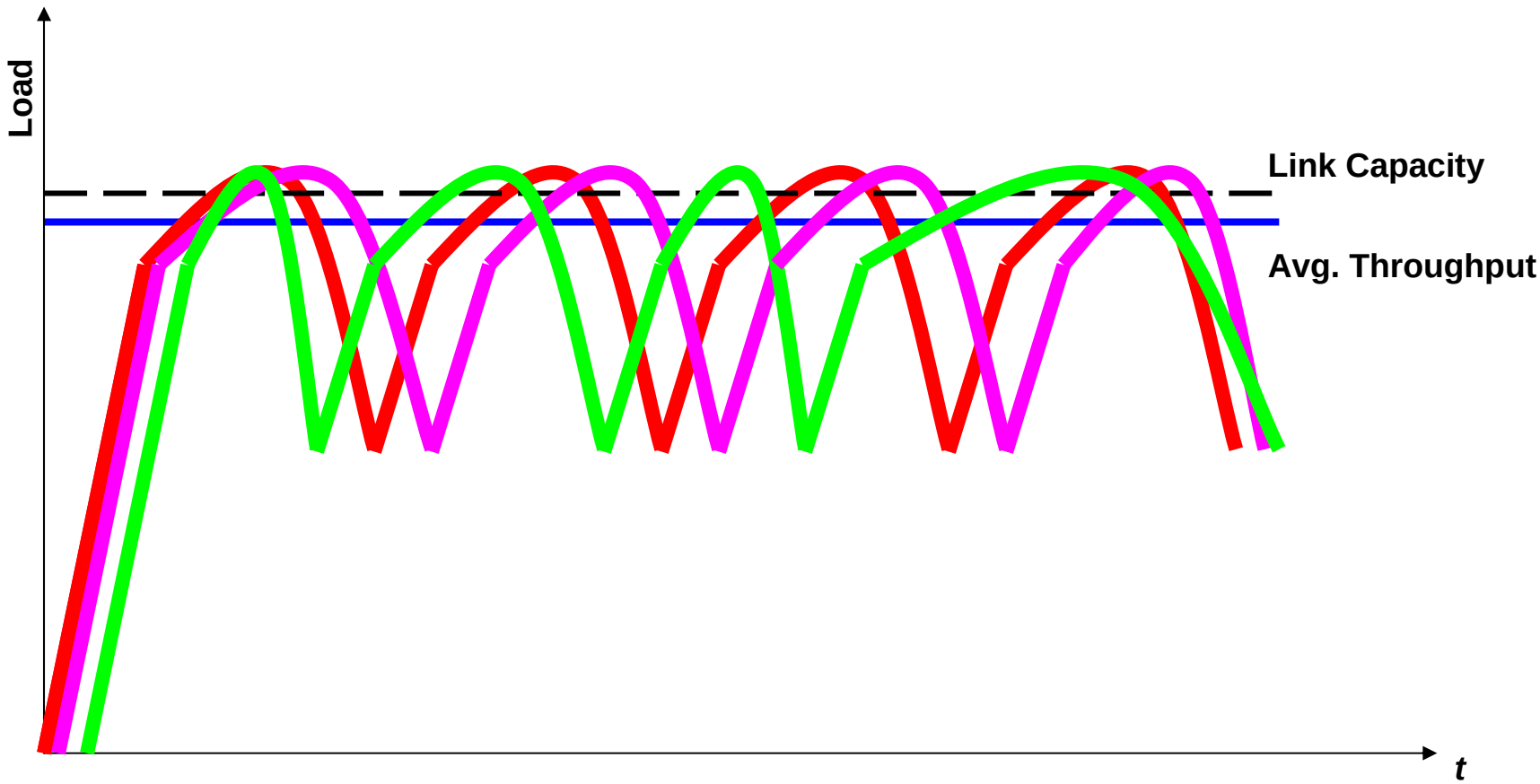
Result: Low average throughput!

Random Early Detection (RED)

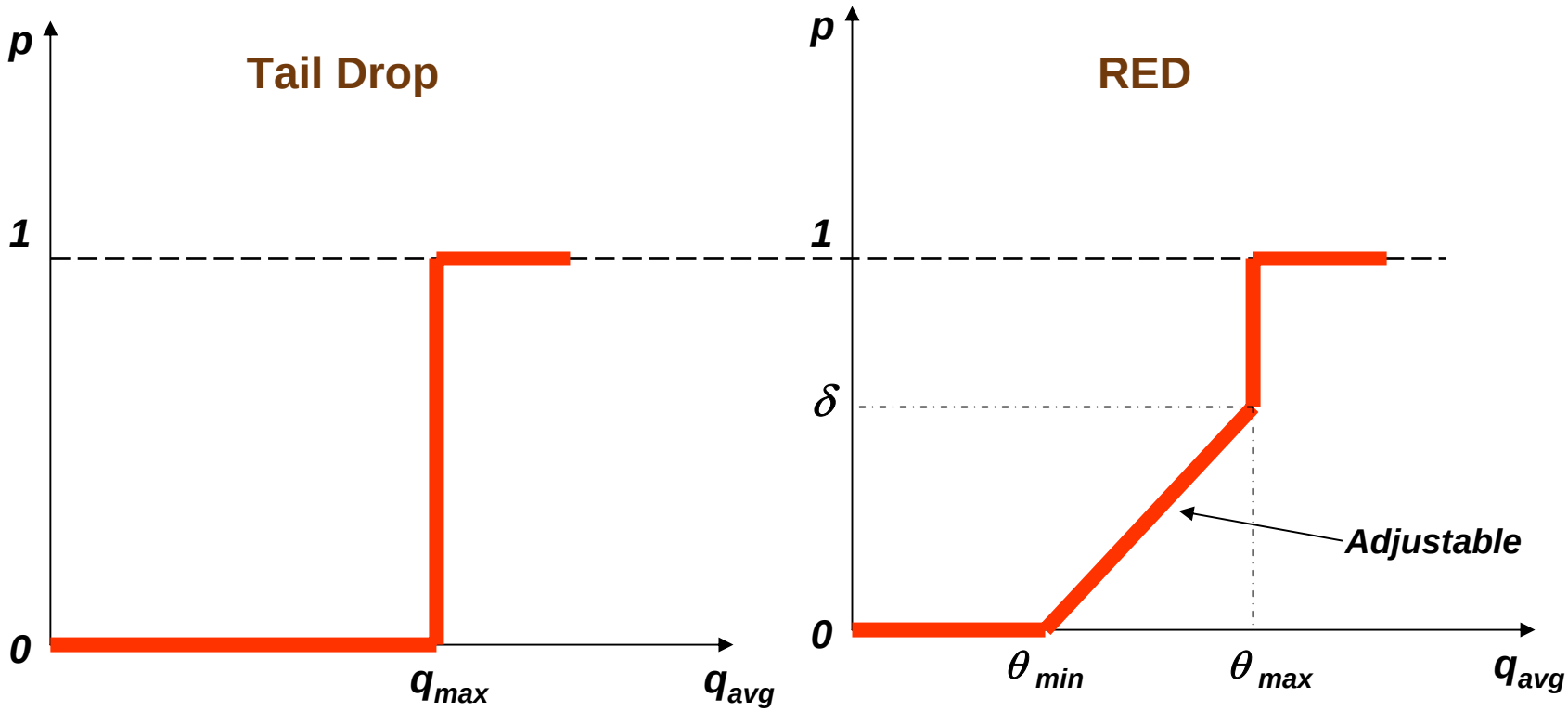
Developed by Van Jacobson in 1993

- Starts randomly dropping packets before actual congestion occurs
- Keeps average queue depth low
- Increases average throughput

Global Synchronization Removed



Random Early Detection (RED)



Random Early Detection (RED)

RED Parameters:

- θ_{min} – Minimal threshold after which RED starts packet drops. Minimal recommended value is 5 packets.
- θ_{max} – Maximal threshold after which all packets are dropped. Recommended value is 2-3 times θ_{min}.
- δ - Mark probability denominator denotes packet drop probability at θ_{max} average queue depth. Optimal value – 0.1 .
- α - Exponential weighting factor determines the level of backward value-dependence in average queue depth calculation:

$$q_{avg} = (q_{old} \cdot (1 - 2^{-\alpha})) + (q_{cur} \cdot 2^{-\alpha})$$

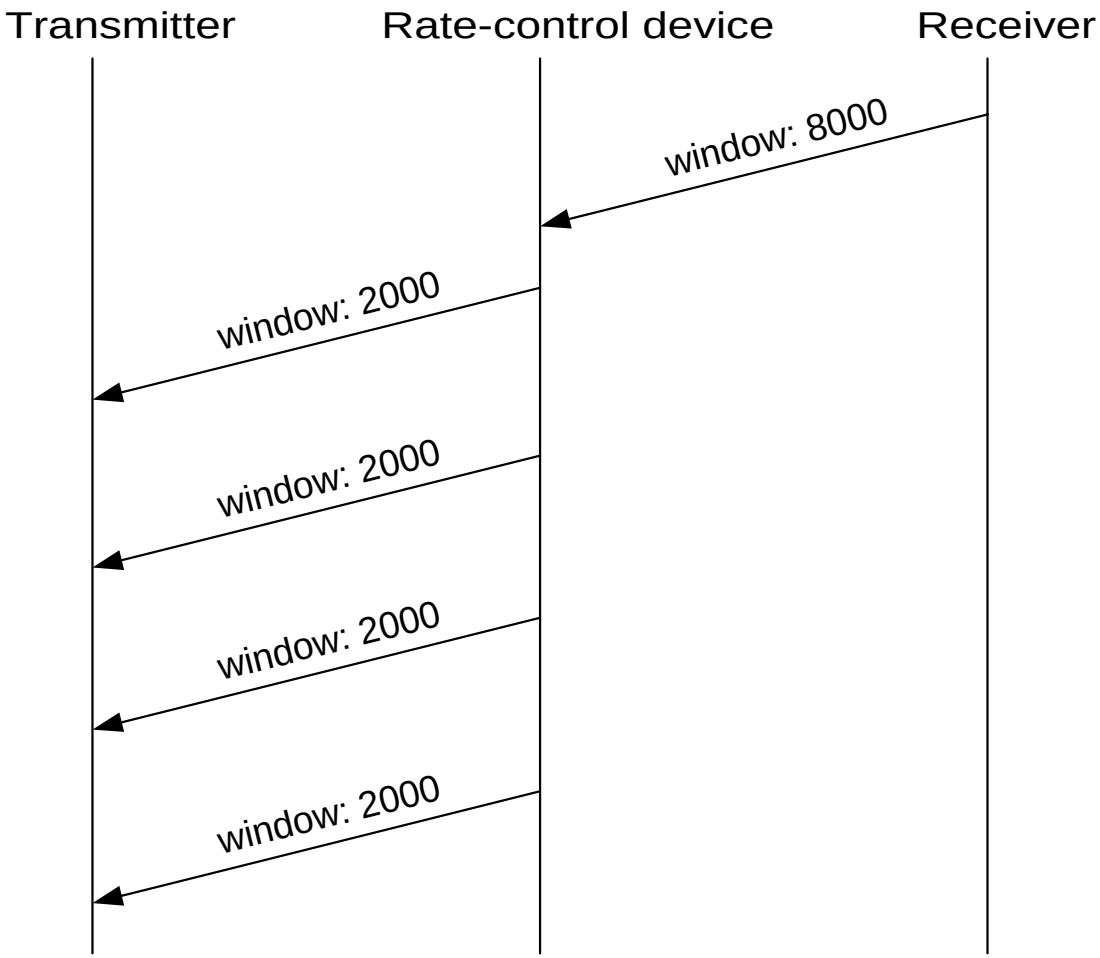
General recommendation $\alpha = 9$.

TCP Rate Control - 1

- In TCP, the spacing of ACKs and the window size in the ACKs controls the transmitter's rate.
- Rate Control manipulates the ACKs as they pass through the rate control device by:
  - Adjusting the size of TCP ACK window
  - Inserting new ACKs
  - Re-spacing existing ACKs
- Rate Control works only with TCP; other methods, such as Token Bucket, must be used with UDP.
- Rate Control violates the protocol layering design, as it allows network devices to manipulate a higher-layer protocol's operation. Nevertheless, it usually functions well and provides fine-grained control.

TCP Rate Control - 2

- Example:



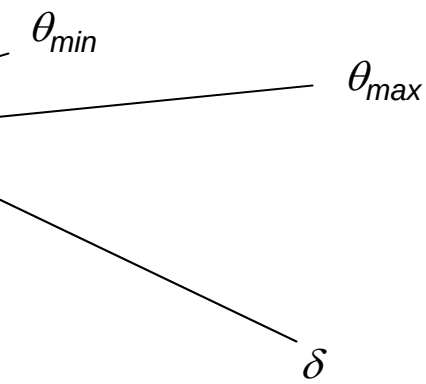
Weighted Random Early Detection (WRED)

- Modified version of RED
- Weights determine the set of parameters: θ_{min} , θ_{max} and δ .
- Weight depends on ToS field value
- Interactive flows are preserved

WRED Configuration Sample

Interface based

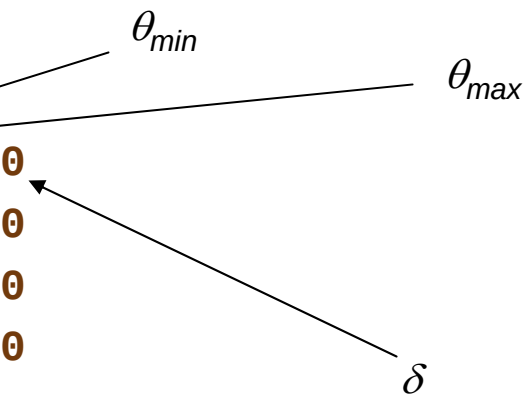
```
interface serial 2/1
   ip unnumbered loopback 0
   random-detect
   random-detect 0 32 64 20
   random-detect 1 32 64 20
   random-detect 2 32 64 20
   random-detect 3 32 64 20
   …
```



WRED Configuration Sample

MQI based

```
policy-map red
   class class-default
        random-detect
        random-detect 0 32 64 20
        random-detect 1 32 64 20
        random-detect 2 32 64 20
        random-detect 3 32 64 20
        …
interface Serial2/1
   ip unnumbered loopback 0
   service-policy output red
```

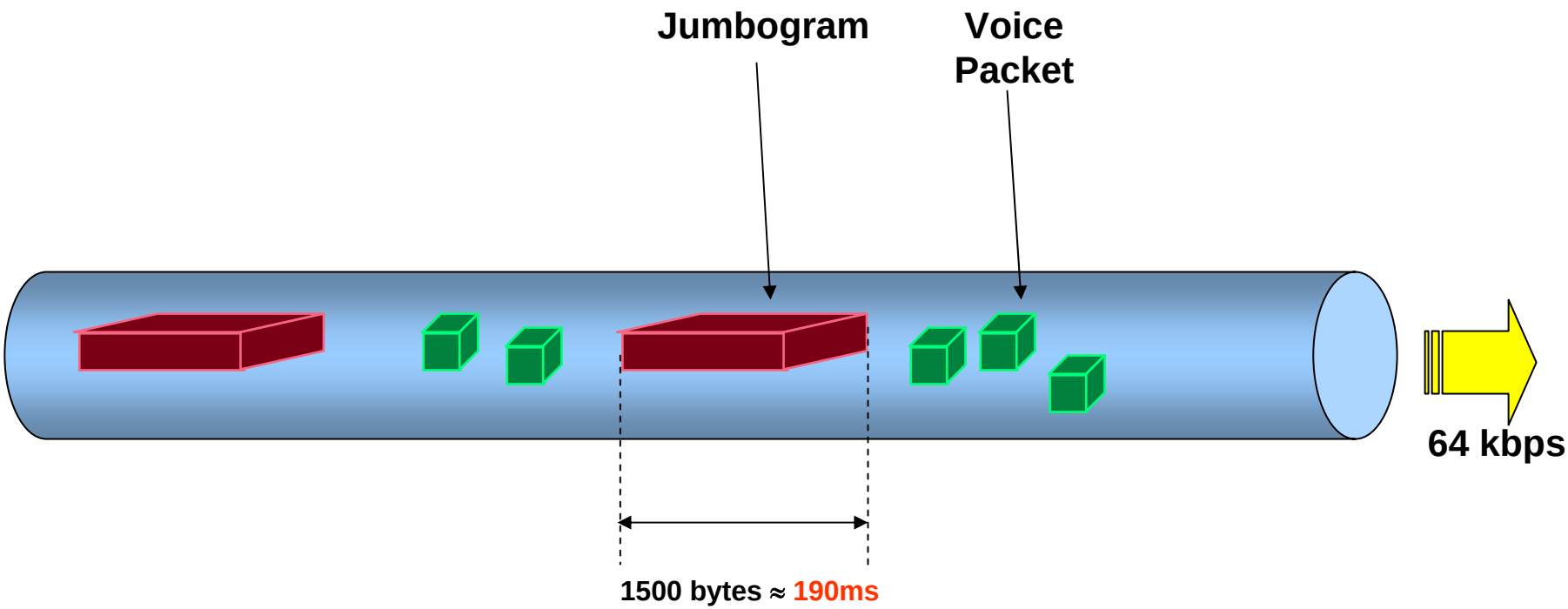


WRED is incompatible with LLQ feature!

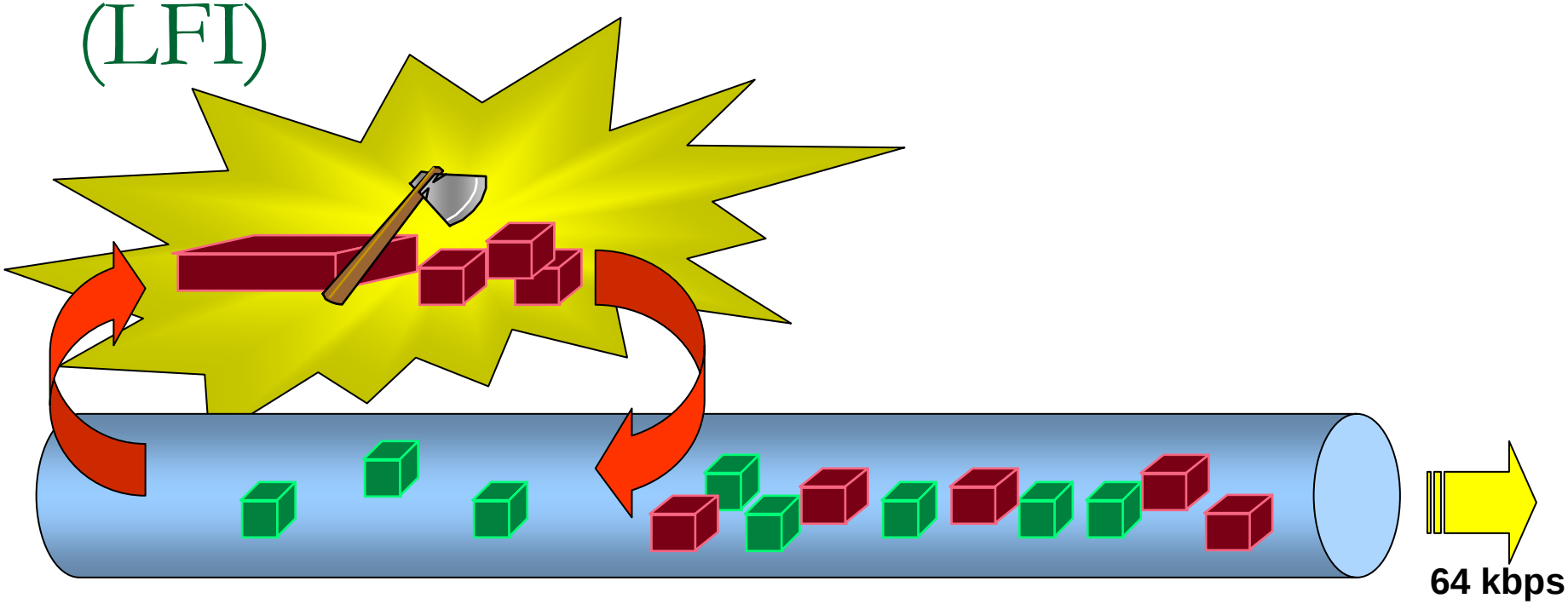
Link Optimization

Link Fragmentation and Interleaving (LFI)

For links < 128kbps



Link Fragmentation and Interleaving (LFI)



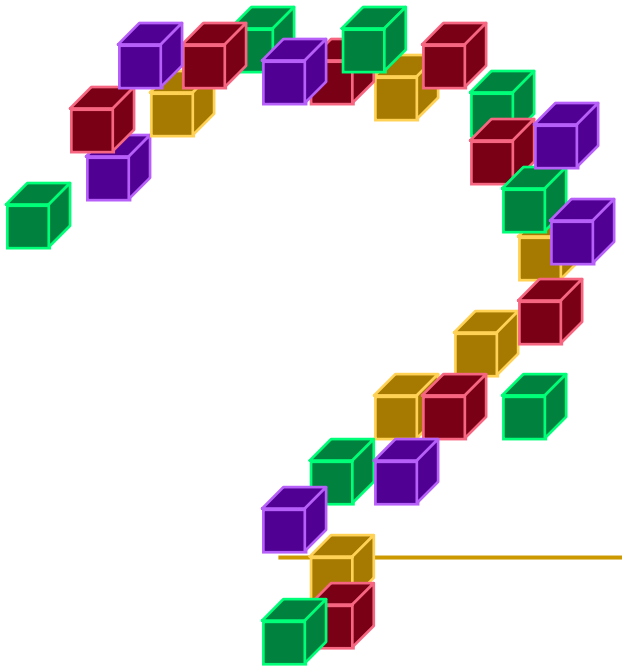
Supported interfaces:

- Multilink PPP
- Frame Relay DLCI
- ATM VC

LFI Configuration Sample

MLP version

```
interface virtual-template 1
   ip unnumbered loopback 0
   ppp multilink
   ppp multilink interleave
   ppp multilink fragment-delay 30
   ip rtp interleave 16384 1024 512
   …
```



Questions???

eng. Nikolay Milovanov

email: *nmil@niau.org*