Proposed VOQ-aware MAC

Outline

- MAC - Client Service Interface
- Client Implementation Options
- MAC Implementation
- ASIC Implementation and Complexity
- Simulation Model

Goals

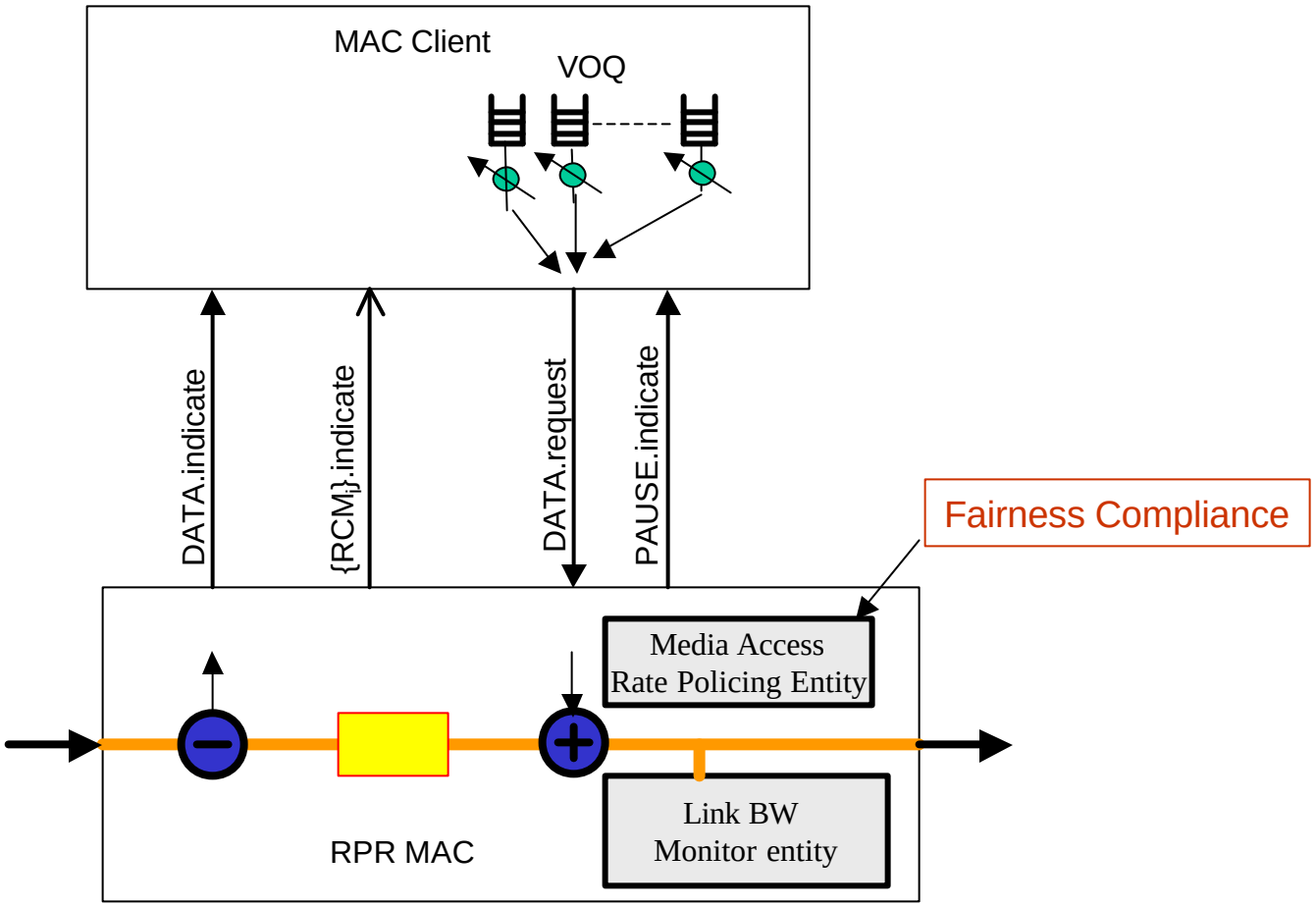
- Solve HoL blocking and poor spatial reuse problems
- Keep MAC simple
  - Complex queueing and scheduling are done outside the MAC by the client
- Support various queueing disciplines.
  - Single queue
  - Priority queues
  - CoS queues
  - VOQ
- Enable systems to provide variety of services
  - Best-effort, diff-serv, guaranteed services, etc.

MAC-Client
Service Interface

MAC Highlights

- **Congestion avoidance**
  - Pro-active BW management to prevent congestion
- **Per-node fairness**
  - MAC monitors transit traffic from **each source** and allocates available BW to each competing upstream source.
- **Access rate policing**
  - MAC meters and enforces its client insert rate for compliance with BW allocations received from **all segments** on the path.
- **No rate-shaping or scheduling**
  - Clients may perform rate-shaping on their own so to conform to the MAC policing.

MAC Service Model



MAC Operation

- **Two modes of operation**
  - Mode 1: Client uses RCM to rate-shape its traffic to pass the policer in the MAC.
  - Mode 2: Client relies on the MAC's policer to tell it when to stop sending traffic.

MAC Behavior

- Packet Admission
  - Accepts **DATA.request** from the client as long as the insert rate does not violate BW allocation on any segment
  - Asserts **PAUSE.indicate** when violation occurs to enforce fairness.
- Packet Drop
  - Asserts **DATA.indicate** when receiving a packet from the ring destined for the client
- Rate Update Messaging
  - Asserts **{RCMi}.indicate** when receiving a new RCM.

Client Behavior

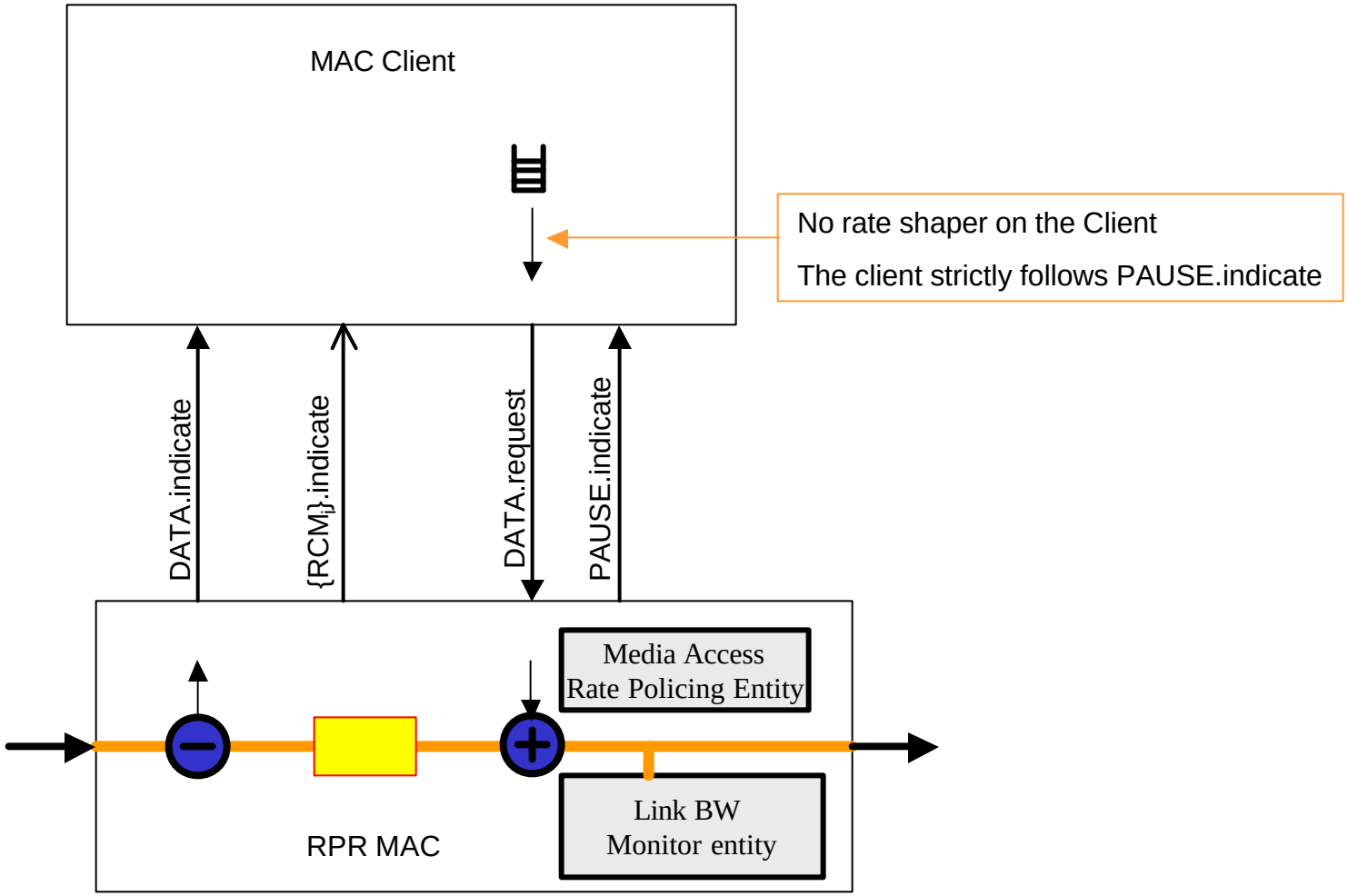
- Transmit
  - Presents **DATA.request** containing destination node and packet length to the RPR MAC.
  - Stops issuing **DATA.request** to the MAC after **PAUSE.indicate** is asserted
- Receive
  - Accepts a packet from the MAC when **DATA.indicate** is asserted
- Rate Update
  - May update per destination rate when **{RCMi}.indicate** is asserted

Client Implementations

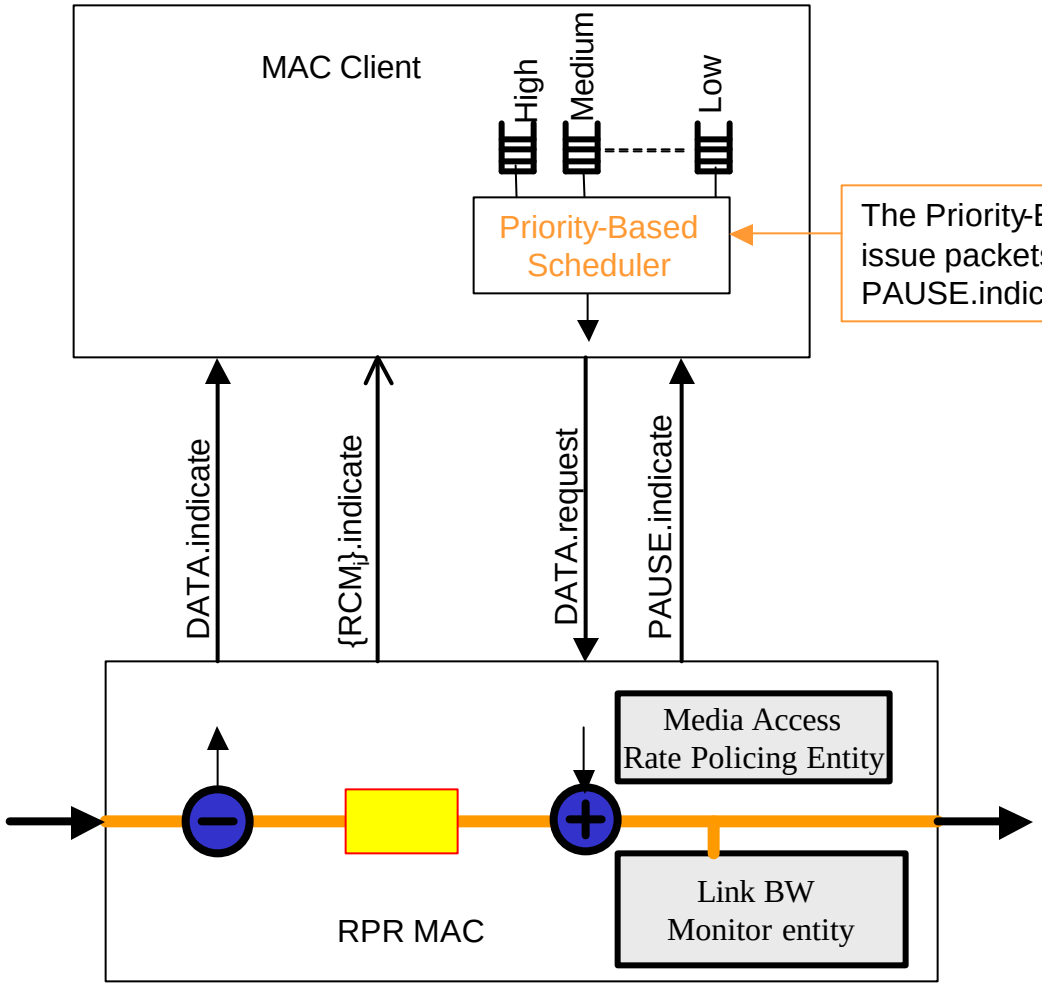
Supported Client Implementations

- Single queue
- Priority queues
- CoS queues
- VOQ

Single Queue

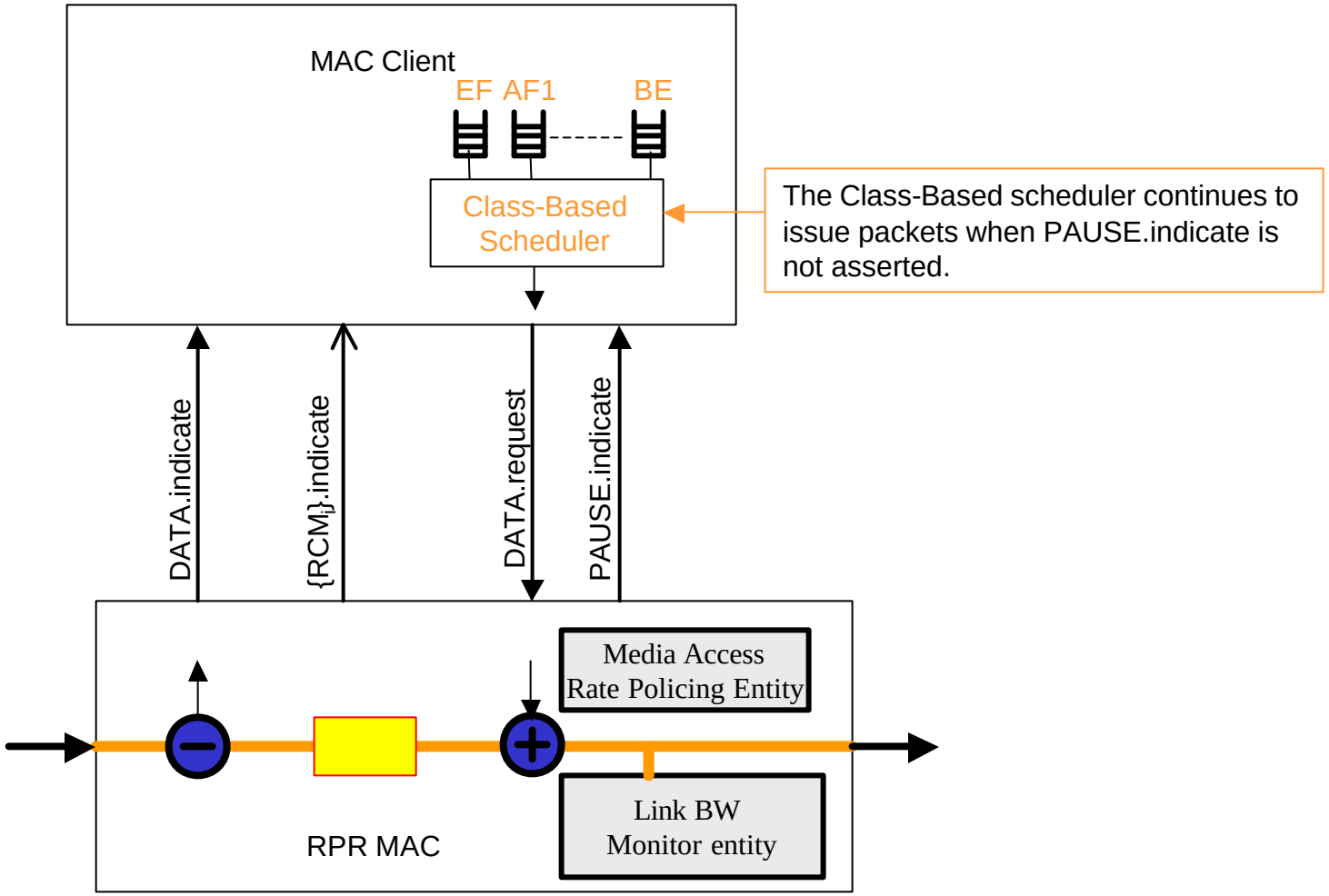


Priority Queues

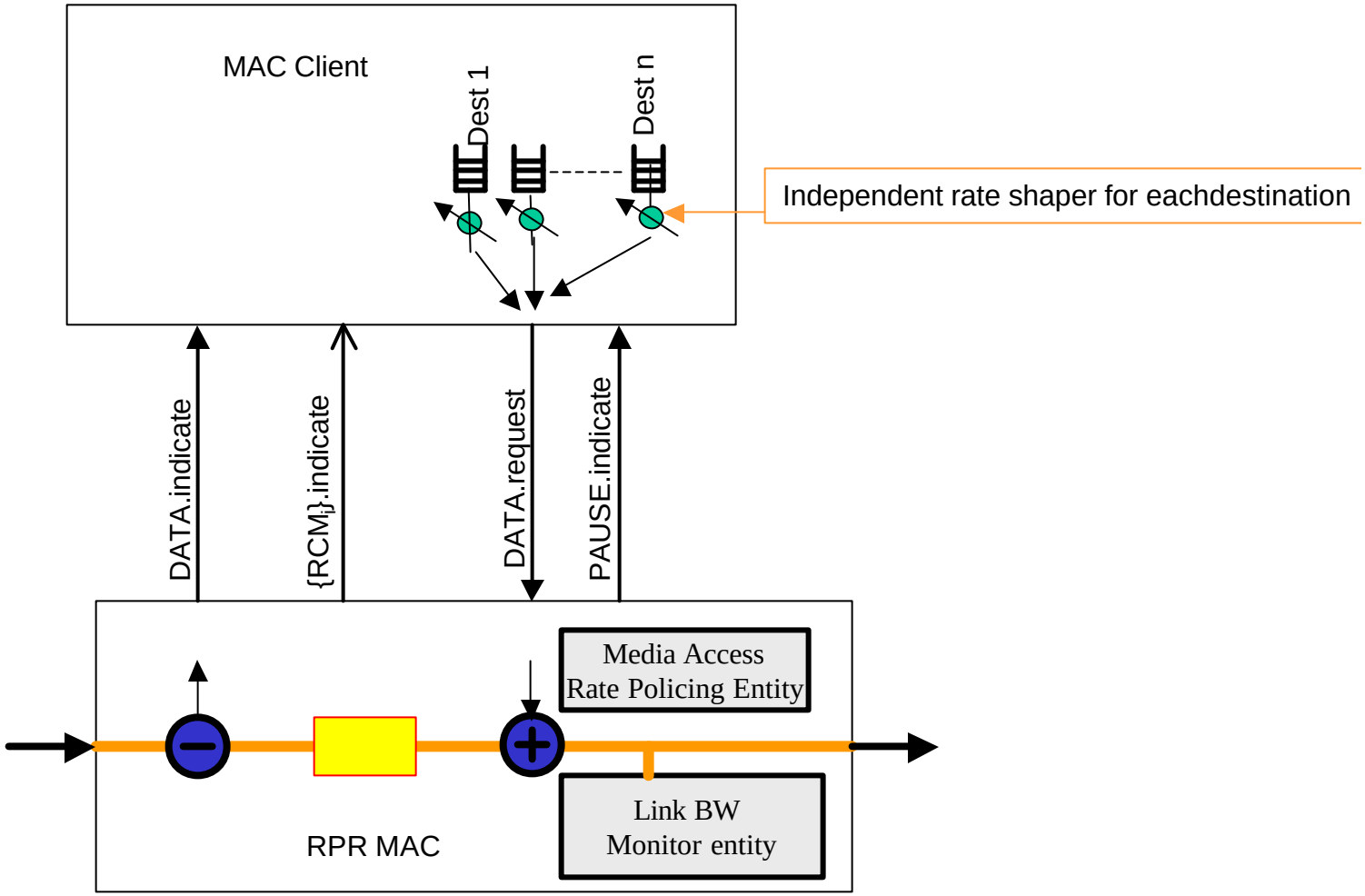


The Priority-Based scheduler continues to issue packets based on priority as long as PAUSE.indicate is not asserted.

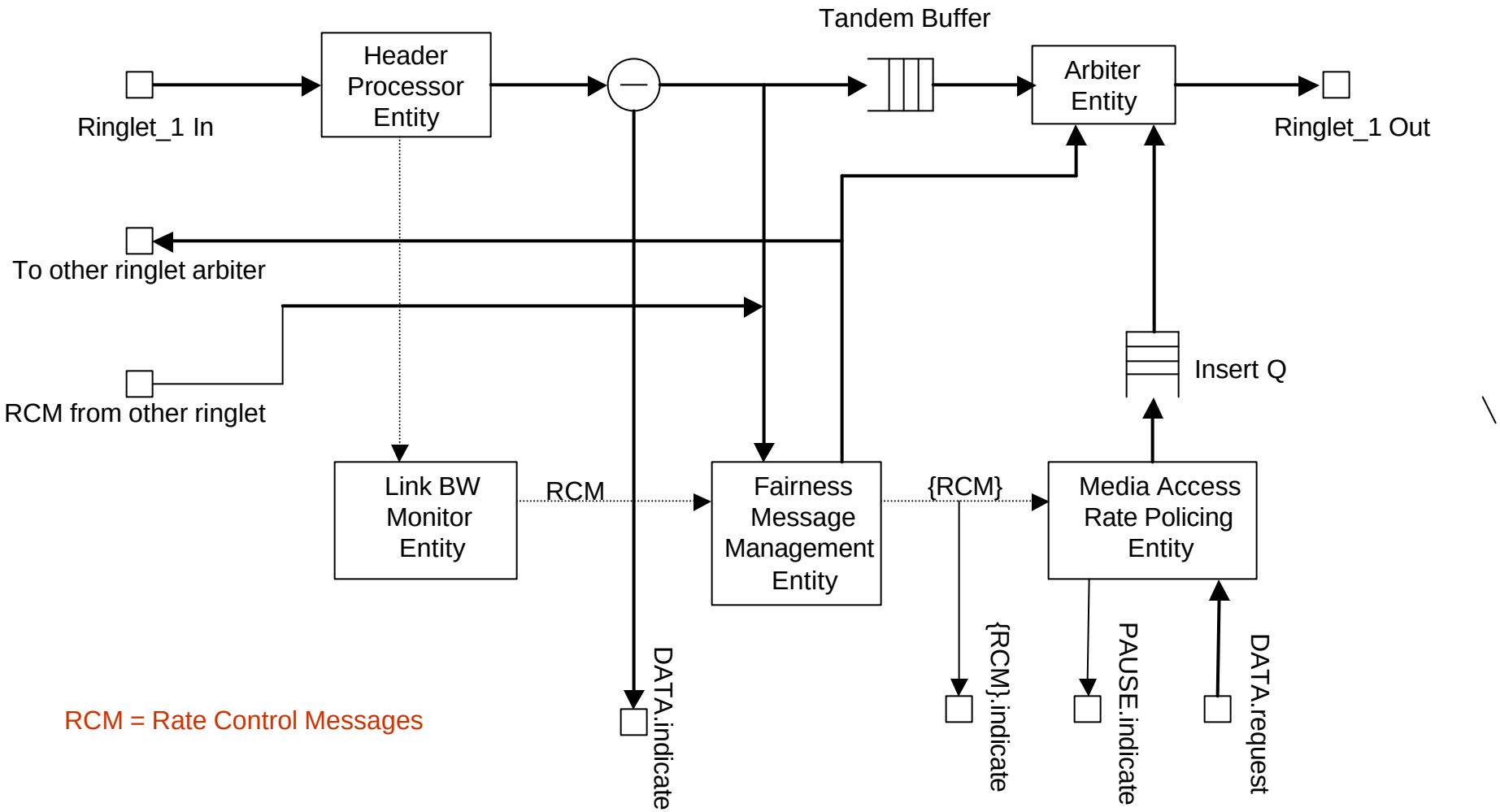
CoS (e.g. diffserv) Queues



VOQ (per-destination)



Fairness Functional Block



MAC Implementation

Weighted Fair Congestion Avoidance (WFCA)

- Per Node BW allocation calculation

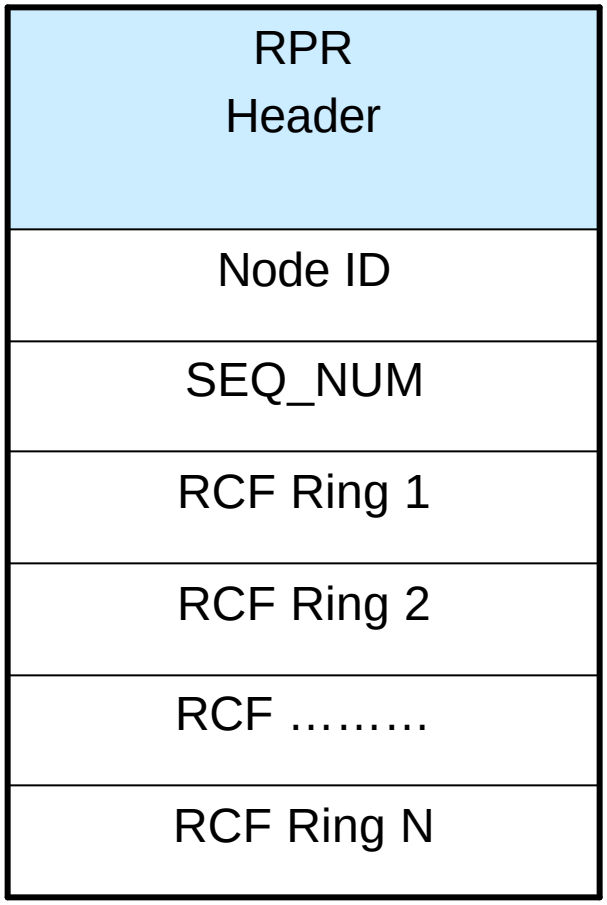
$$f_i = r_i + w_i \frac{(C - \sum_{active} r_i)}{\sum_{active} w_i}$$

- Rate Control Factor (RCF)

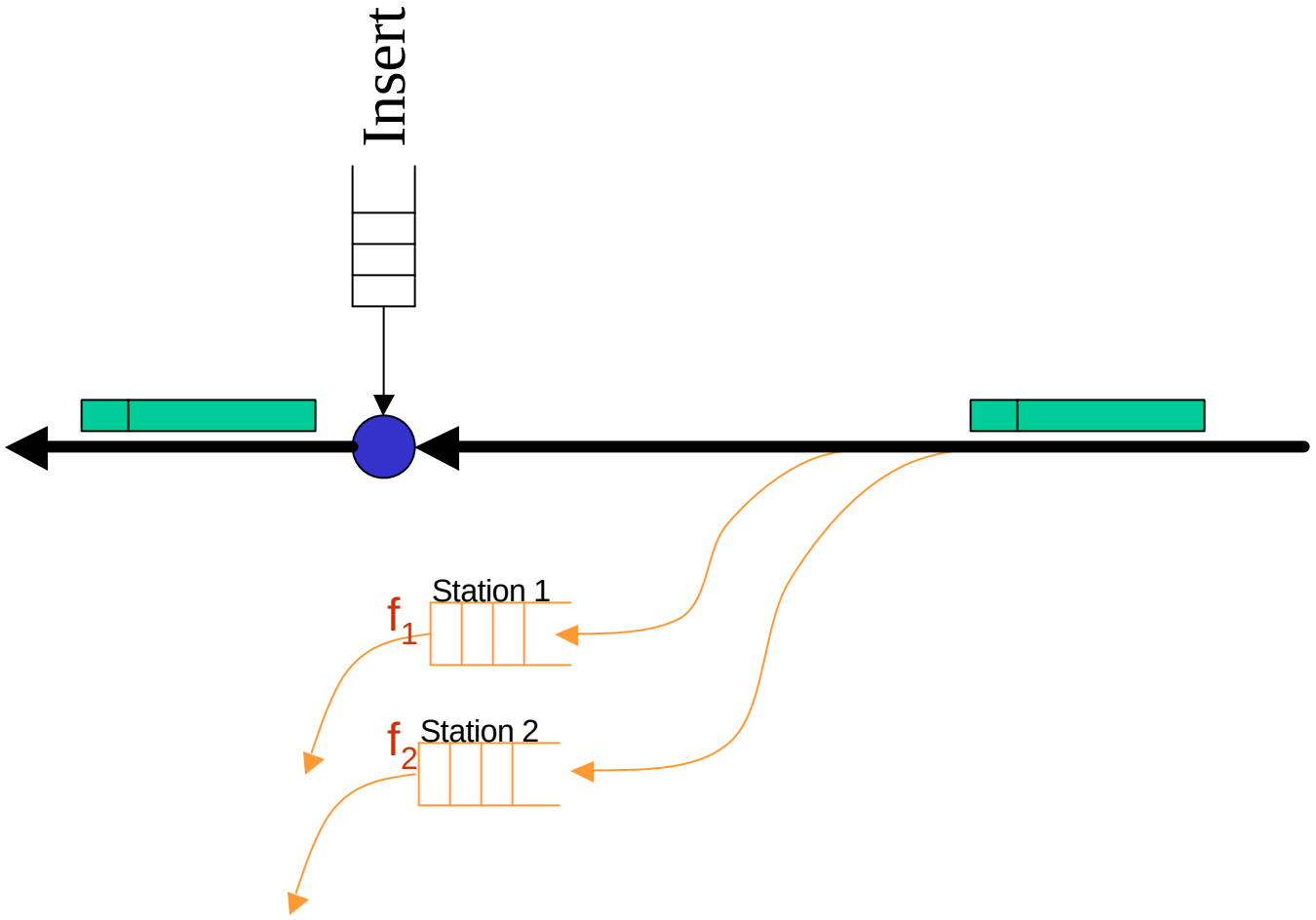
$$RCF = \frac{(C - \sum_{active} r_i)}{\sum_{active} w_i}$$

f_i = BW node i received
r_i = committed BW for node i
w_i = node i weight
C = link capacity

Rate Control Message Format



Leaky Bucket



WFCA Implementation

```
@packet arrival
    bucket(source node) += packet length
```

```
@calc_interval (1us)
    for each source node (I=1 to 256)
        drain bucket(i)
        bucket(I) -= calc_interval*(Ri+Wi*ABF)
        if (bucket(I) <0), bucket(I) = 0
        if (bucket(I) >0)
          SUM Ri += Ri
          SUM Wi += Wi
   end FOR
  RCF = (link capacity – SUM Ri) / SUM Wi
  SUM Ri = 0
  SUM Wi = 0
```

Register File

Source node →

Ri 16 bits	wi 16 bits	Leaky Bucket 32 bits
1		
2		
.		
.		
.		
.		
256		

SUM Ri
SUM Wi
RCF
Calc Interval
Link Capacity

Media Access Rate Policing Entity

```
@ each pacing_interval (256 clks)
  for each link segment
     calculate the node (for this MAC) allowed BW, fj.
       fj = rj + wj*RCF
     give credit for each segment
       if ( segment_credit) < 1 MTU
          segment_credit += fj * interval_value
      if ( segment_credit) < 0   // client BW exceeds limit
          assert PAUSE.indicate
 end FOR
```

Register File

segment →

RCF 32 bits	Segment Credit 16 bits
1	
2	
. . . .	
256	

```
@ each DATA.request
   if no PAUSE.indicate asserted, accept DATA.request
        for each segment between this and the dest nodes
          deduct segment credit
             segment_credit -= packet_length
        end FOR
```

Rj
Wj
Interval_value

ASIC Implementation

ASIC Implementation

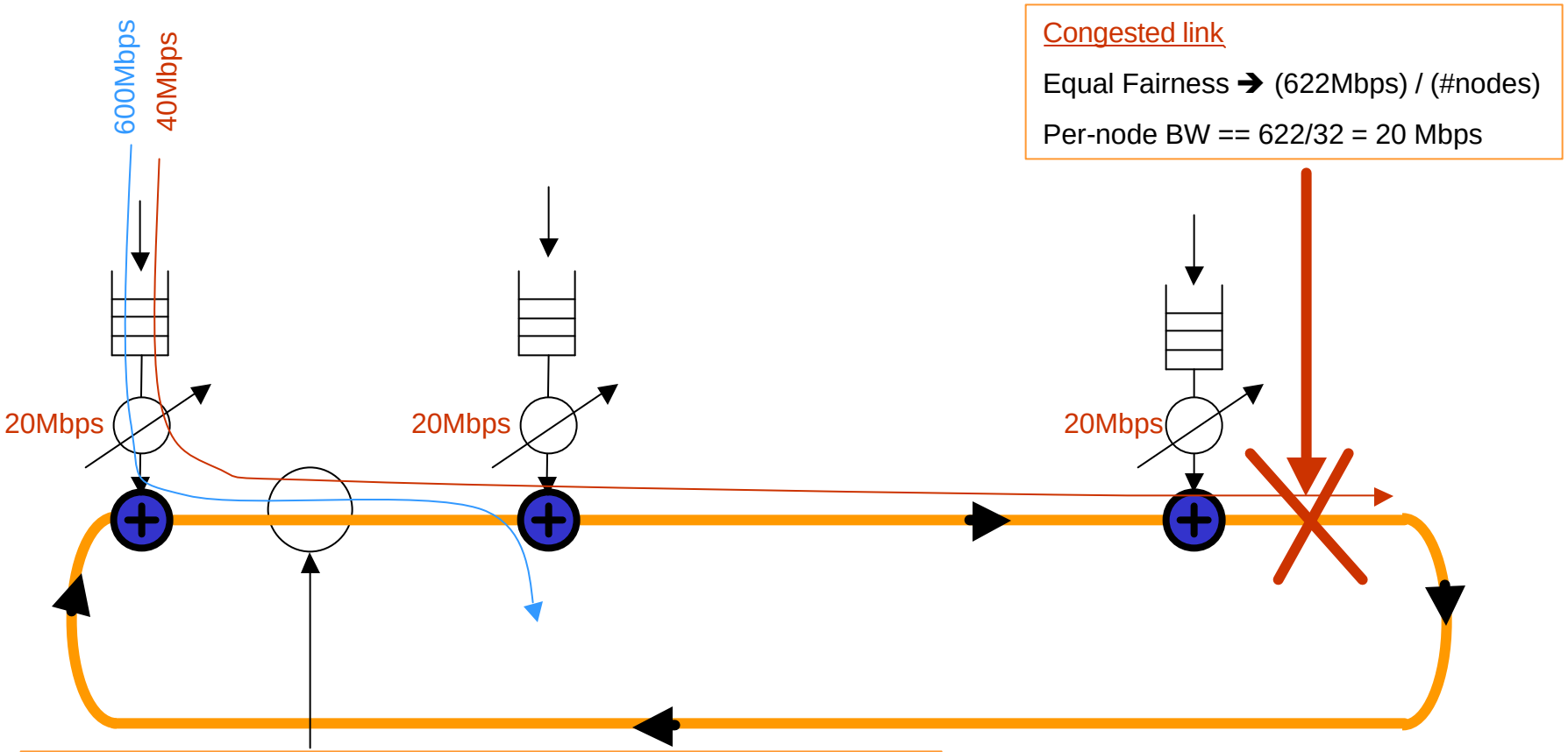
- Implemented in less than 1000 lines of Verilog code
- Less than 250k gates (not optimized) per ring for WFCA and rate metering (0.18 um)
- For 10G Ring's speed
- 156 MHz clock speed

Gate Count

Block Name	Gate Count	Memory Size
WFCA	25K	16Kbit
Rate Policing	70K	12Kbit
Data Path	?	?

Quiz!

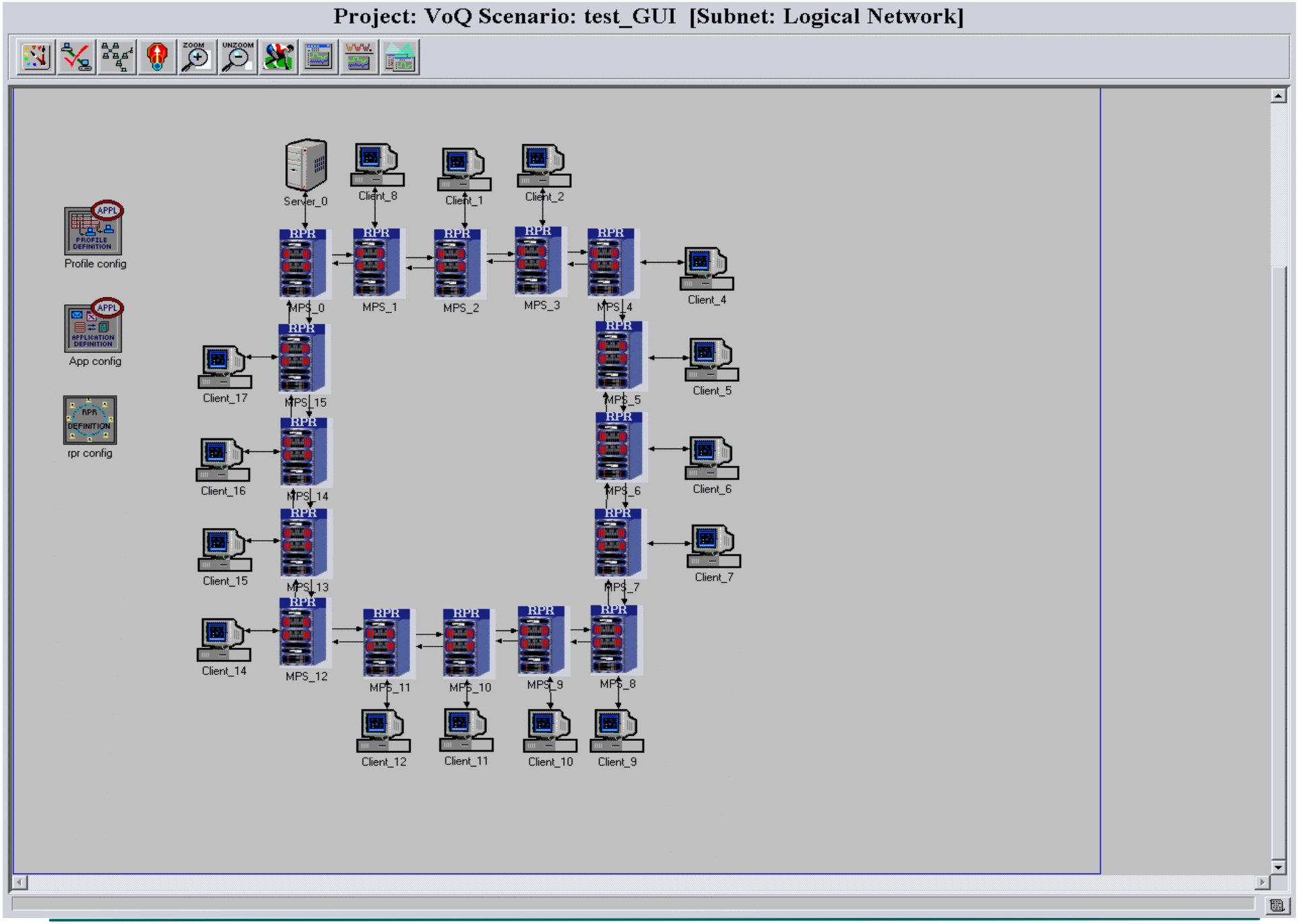
What to do?

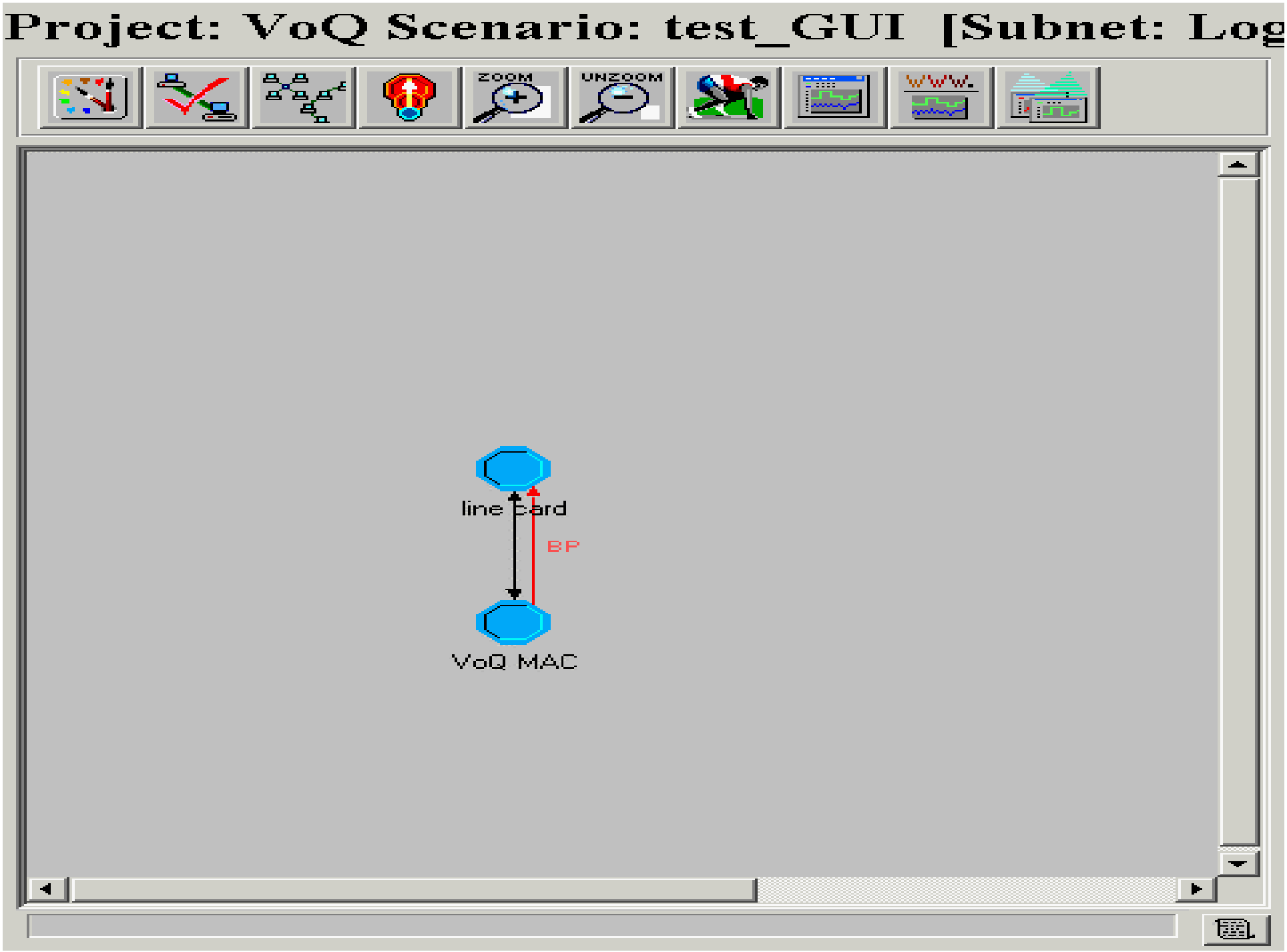

600Mbps
40Mbps
20Mbps
20Mbps
20Mbps
Congested link
Equal Fairness ➔ (622Mbps) / (#nodes)
Per-node BW == 622/32 = 20 Mbps
PROBLEM: The rate shaper limits the total insert rate to 20 Mbps blocking the 600Mbps traffic

Summary

- Low Complexity
- Universal client support (FIFO, CoS, VOQ, etc.)
- No HoL blocking and 100% spatial reuse
- Simulation model available

Simulation Model


Project: VoQ Scenario: test_GUI [Subnet: Logical Network]
ZOOM
UNZOOM
APPL
PROFILE DEFINITION
Profile config
APPL
APPLICATION DEFINITION
App config
RPR
DEFINITION
rpr config
Server_0
Client_8
Client_1
Client_2
RPR
MPS_0
MPS_1
MPS_2
MPS_3
MPS_4
Client_4
Client_5
Client_6
Client_7
MPS_5
MPS_6
MPS_7
MPS_8
MPS_9
MPS_10
MPS_11
MPS_12
MPS_13
MPS_14
MPS_15
Client_17
Client_16
Client_15
Client_14
Client_12
Client_11
Client_10
Client_9


Project: VoQ Scenario: test_GUI [Subnet: Log
ZOOM
UNZOOM
WWW.
line card
BP
VoQ MAC

Node Model: VoQ_MAC