# RPR MAC Transit Path Design

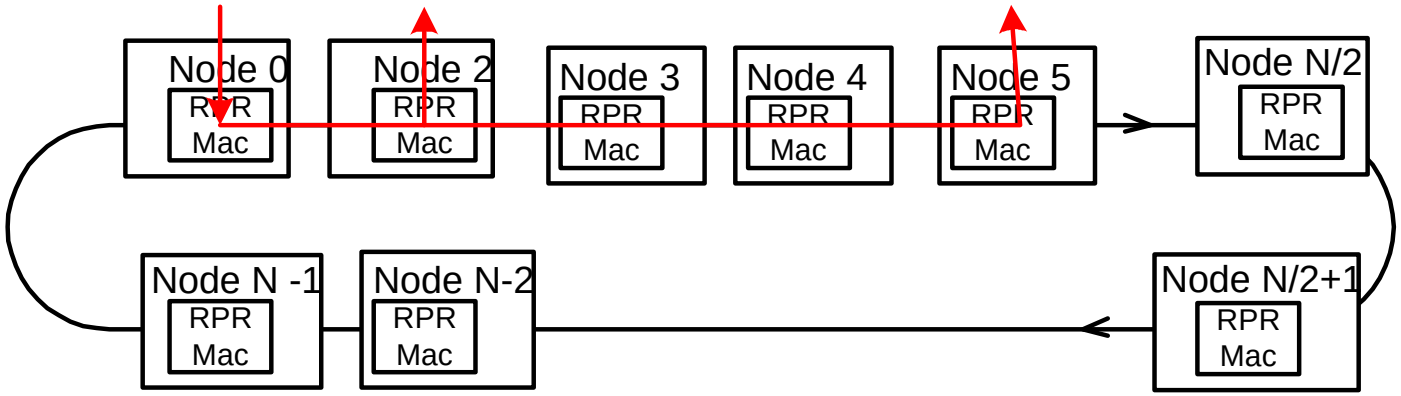
>>

Sanjay K. Agrawal, Luminous
Jean De_Jaegher, Alcatel
Adisak Mekkittikul, Lantern
Harry Peng, Nortel

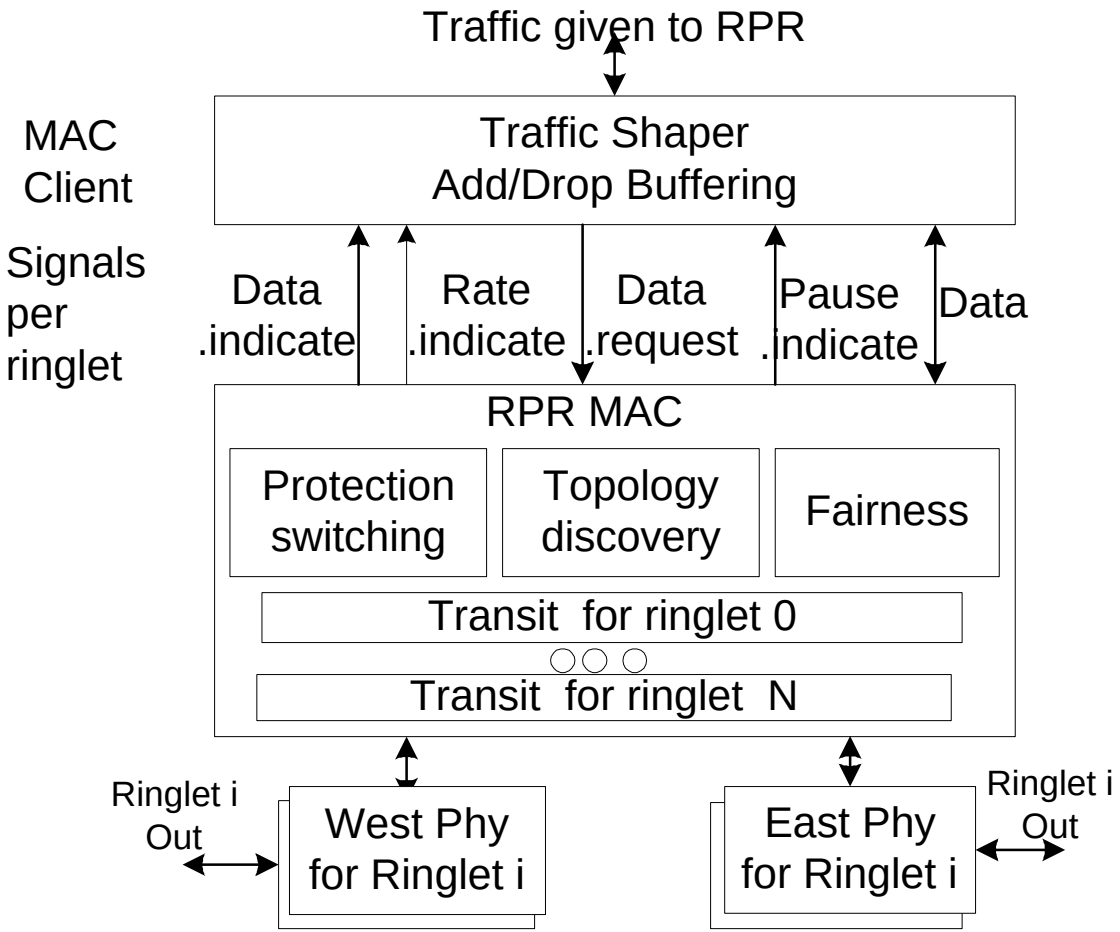
# Outline

- RPR System Architecture
- RPR MAC Requirements and Objectives
- RPR MAC Framework Proposal
- RPR MAC Transit Path Design
- Optional Modes of Operation
- RPR MAC Client Add/Drop Path Design
- Conclusion

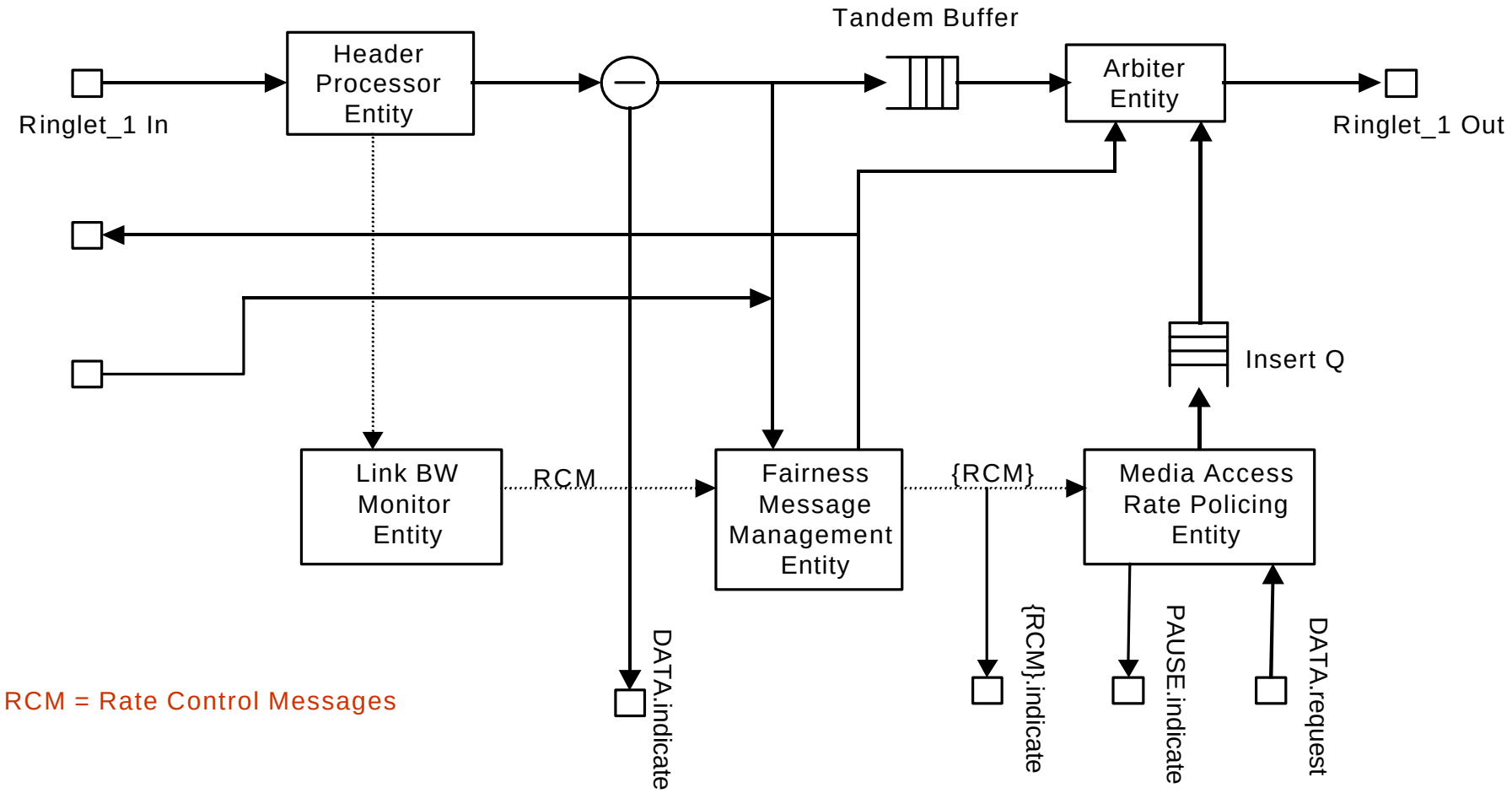
# RPR Network


Node 0
RPR Mac
Node 2
RPR Mac
Node 3
RPR Mac
Node 4
RPR Mac
Node 5
RPR Mac
Node N/2
RPR Mac
Node N -1
RPR Mac
Node N-2
RPR Mac
Node N/2+1
RPR Mac

# RPR System Architecture



# Signaled Mode: MAC Control Path Architecture

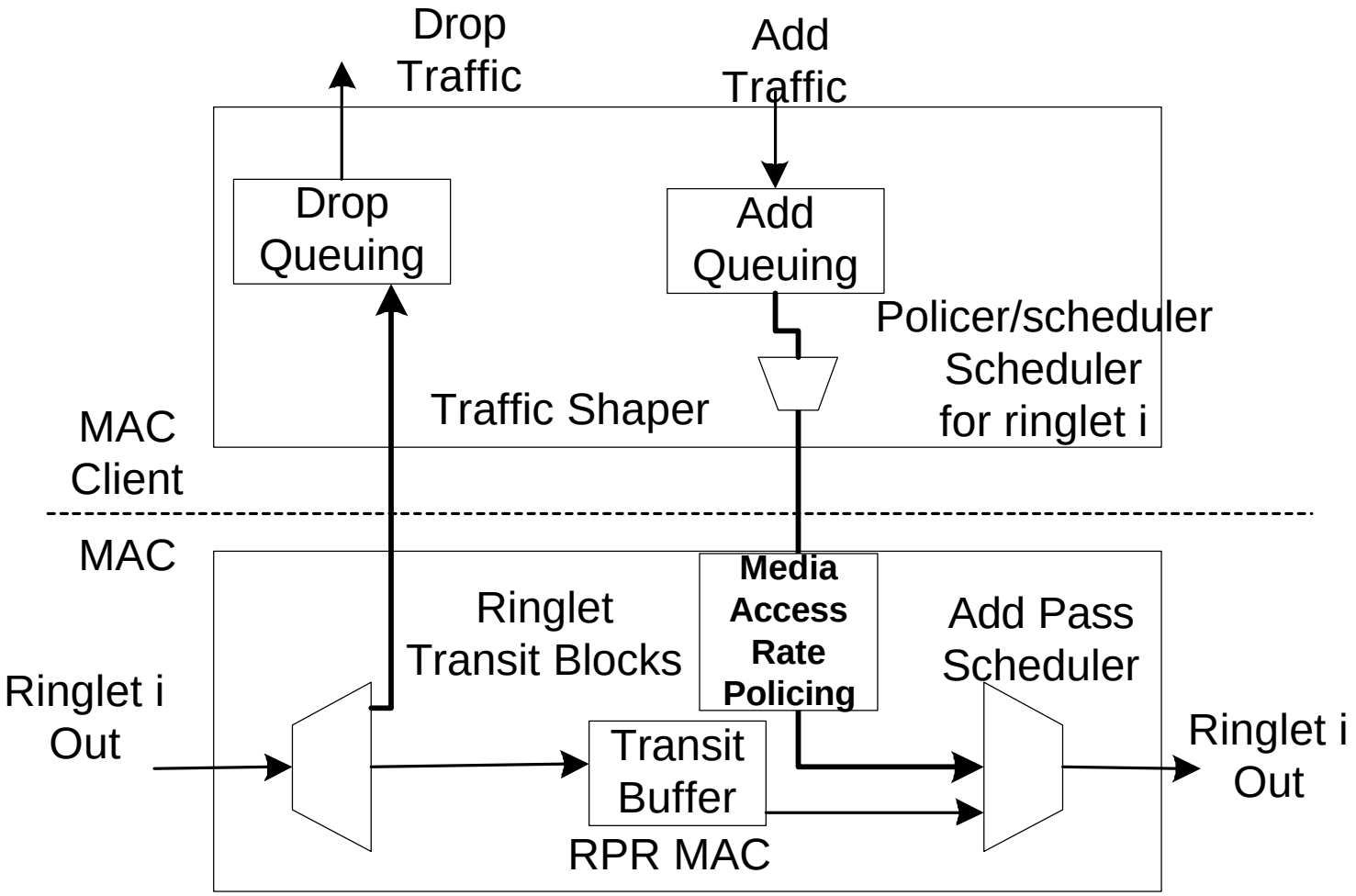



RCM = Rate Control Messages

# Objectives and requirements for the transit path

- The transit path is part of the shared medium
- The transit path is lossless.
- The transit path implements destination and source stripping.
- The transit path implements broadcasting and multicasting: drop and pass mode
- Minimal buffering in the transit path
  - Minimize the cost of the standard RPR MAC chip saving cost of external memory
  - Minimize delay in the transit path
  - Maximize scalability as RPR MAC chip scales at higher speed

# RPR MAC: Transit Data Path



# RPR MAC Reception rules

- When a frame arrives at RPR MAC, the DA MAC address is matched with the RPR database in the header-processing block.
- The decision to strip or bypass the frame:
  - If the frame DA matches in the RPR database
    - Frame is stripped from the ring.
  - If the frame DA is a broadcast,
    - Frame is both stripped and sent out
  - If the frame SA matches the RPR MAC database,
    - Frame is stripped, and discarded.
  - If the frame has a bad CRC on the RPR MAC header,
    - Frame is stripped and discarded. A bad CRC counter is incremented.
  - If the DA MAC address of the incoming frame does not match the RPR database and TTL <=1
    - Frame is stripped and discarded.

# RPR MAC Reception rules (cont…)

- Else, the frame is bypassed.
  - The TTL field in the RPR MAC header is decremented by one.
- Reception of the RPR header only for decision.
- Promiscuous Mode:
  - RPR MAC allows all the transit traffic to be received to the MAC client.
  - Promiscuous mode of operation is necessary for debugging operations.
  - In addition, this mode allows MAC client to arbitrate and re-queue the transit traffic with the add traffic before transmitting back into the MAC.

# RPR MAC Transit Rules

- Transit frames are sent to the transit buffer.

The scheduling algorithm:

- *Step 1: Choose a frame to be transmitted*

  *If a transit buffer has at least one byte*

  *Choose a frame from the transit buffer*

  *Else if an insert buffer has at least one frame*

  *choose a frame from the insert buffer*

  *Step 2: transmit the chosen frame with no pre-emption*

  *Step 3: complete the transmission, repeat step 1*

- Add frames cannot interrupt the transit frame for the transmission (No pre-emption).
- Transit frame cannot interrupt the add frames under transmission.
- The default store and forward mode of operation transit frames are received entirely before they are sent out.
- Minimum buffering needed in the transit buffer for the transit frame is single MTU.

# RPR MAC drop rules

- The FCS for the header, HEC, is checked. If the FCS is incorrect
  - The frame is dropped.
- If the source MAC address matches the RPR MAC database in Header Processing block,
  - The frame is dropped.

# RPR MAC add rules

- Add frames are queued in the MAC client indicates frame to RPR MAC through data.indicate signal.
- RPR MAC accepts the frame in the add path:
  - There is no packet under transmission.
  - Transit buffer is empty.
  - Media access rate control block accepts the packet.
- Else, pause.indicate signal is asserted to prevent MAC clients to insert frames into the RPR MAC
- Media Access Rate Control
  - The PAUSE shall be asserted to prevent the MAC client from exceeding the allocated bandwidth on any segment downstream.

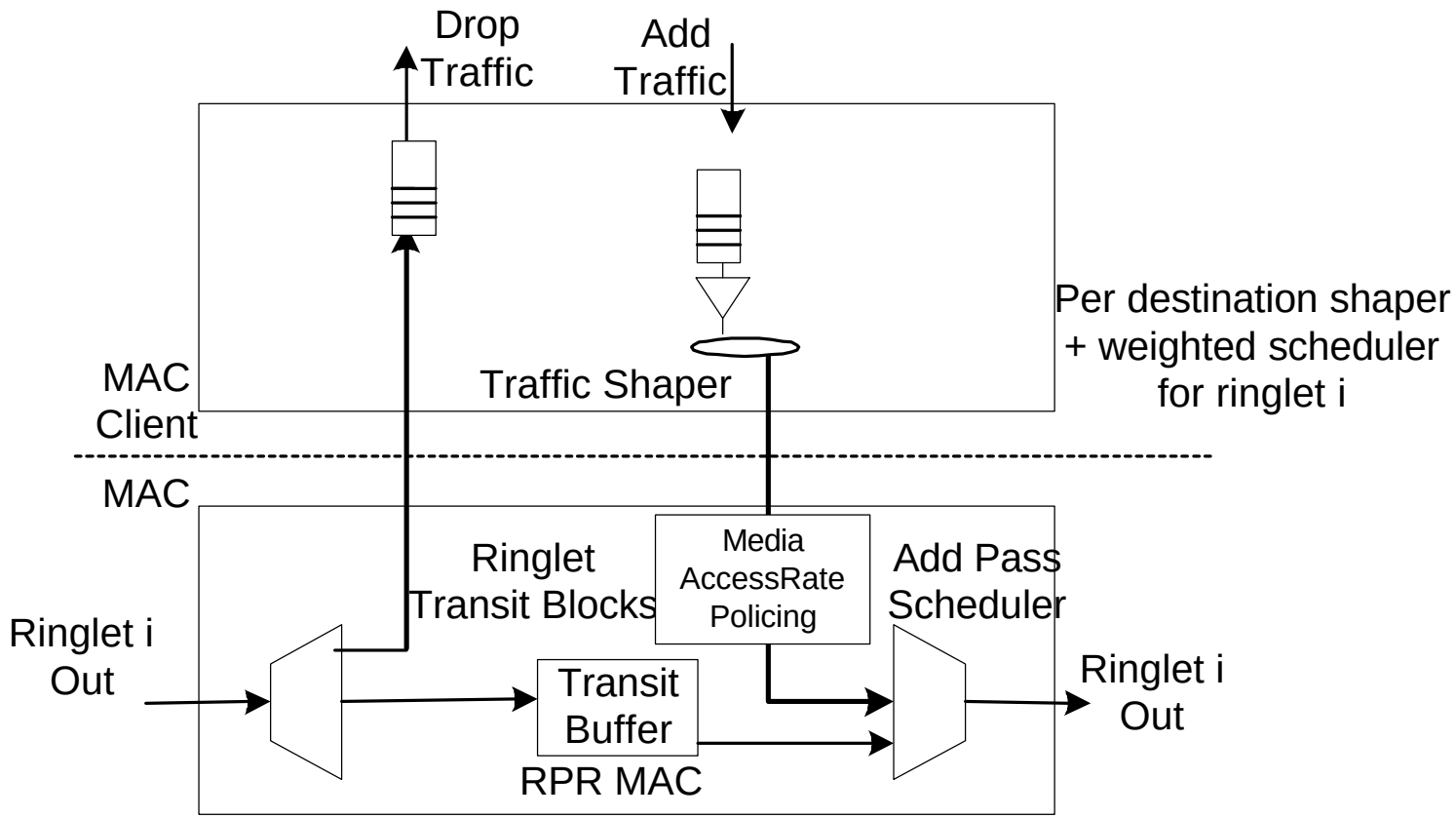
# Support for Virtual Output Queuing

- Objective: maximize the spatial reuse and the link utilization for frame flows with arbitrary (source, destination) pairs.
- Problems
  - RPR MAC sets the access rate low to satisfy the bandwidth allocated by one congested destination
    - Severely limits the access rates to other uncongested destinations.
  - (HoL) blocking problem occurs in a single queue access.
    - Frame destined to uncongested destination waits behind an frame congested destinations.
- Proposed Solution
  - Signaling messages propagates independent media access rate control for each ring segment in the RPR MAC.
  - Virtual Output Queuing (VoQ) in the MAC client
    - Dedicated output queue for each destination.

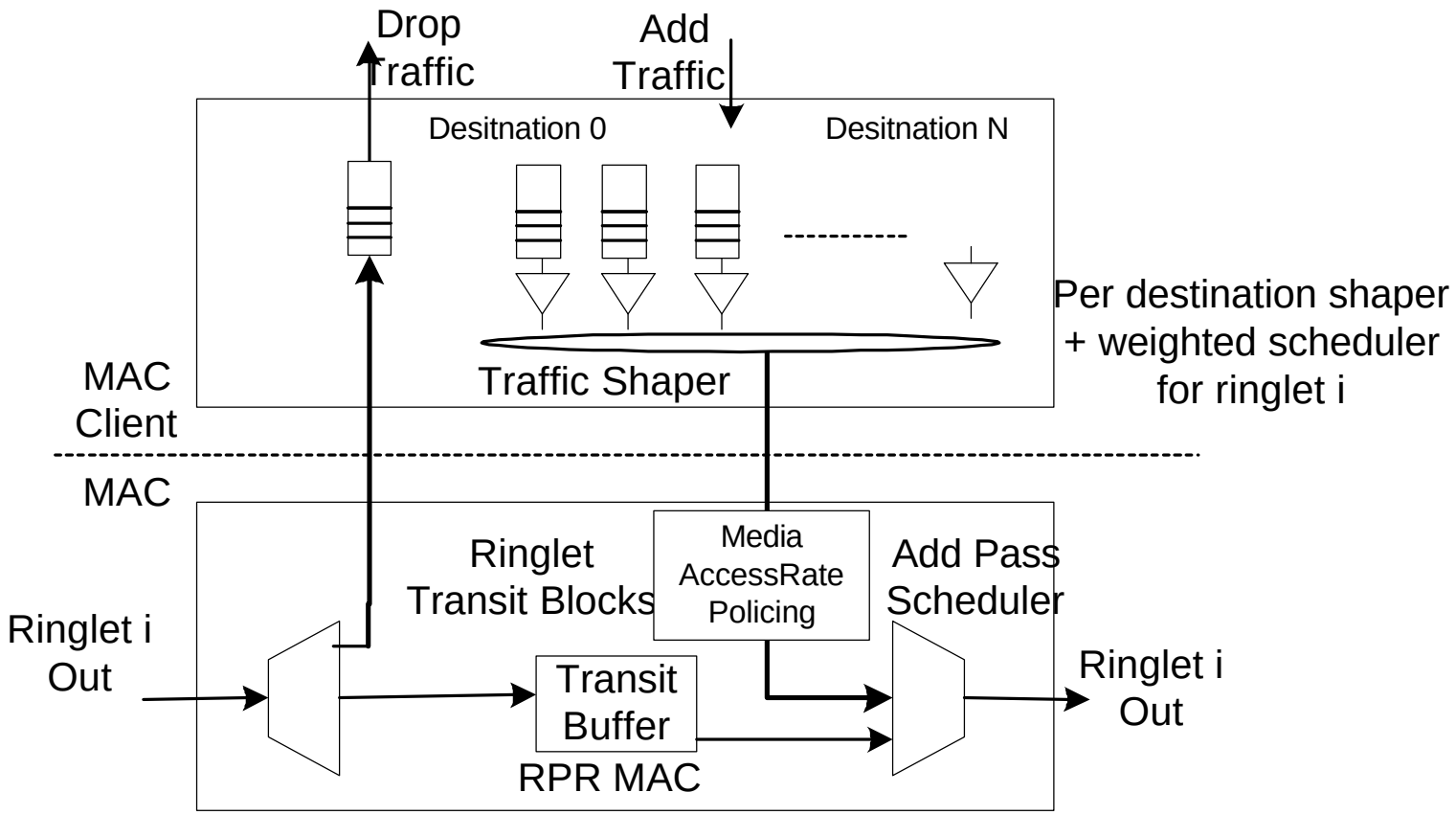
# Optional modes of operation and transit buffer considerations

- Store & forward mode:
  - Frame is entirely received before it is considered for transmission. Thus, minimum buffering needed in the transit buffer for the transit frame is single MTU.
  - This mode of operation allows FCS errored frames to be stripped and transit error counter incremented.
    - Eliminates degraded frames in the transit path rare instance of FCS error.
- **Cut-through Mode:**
  - First bytes of the transit frames are sent out before the last bytes of the frame are received.
  - RPR header should be received entirely before beginning transmission out of the outgoing ringlet, since the header has to be processed.
    - Reduces the delay that frames experience in the transit path.

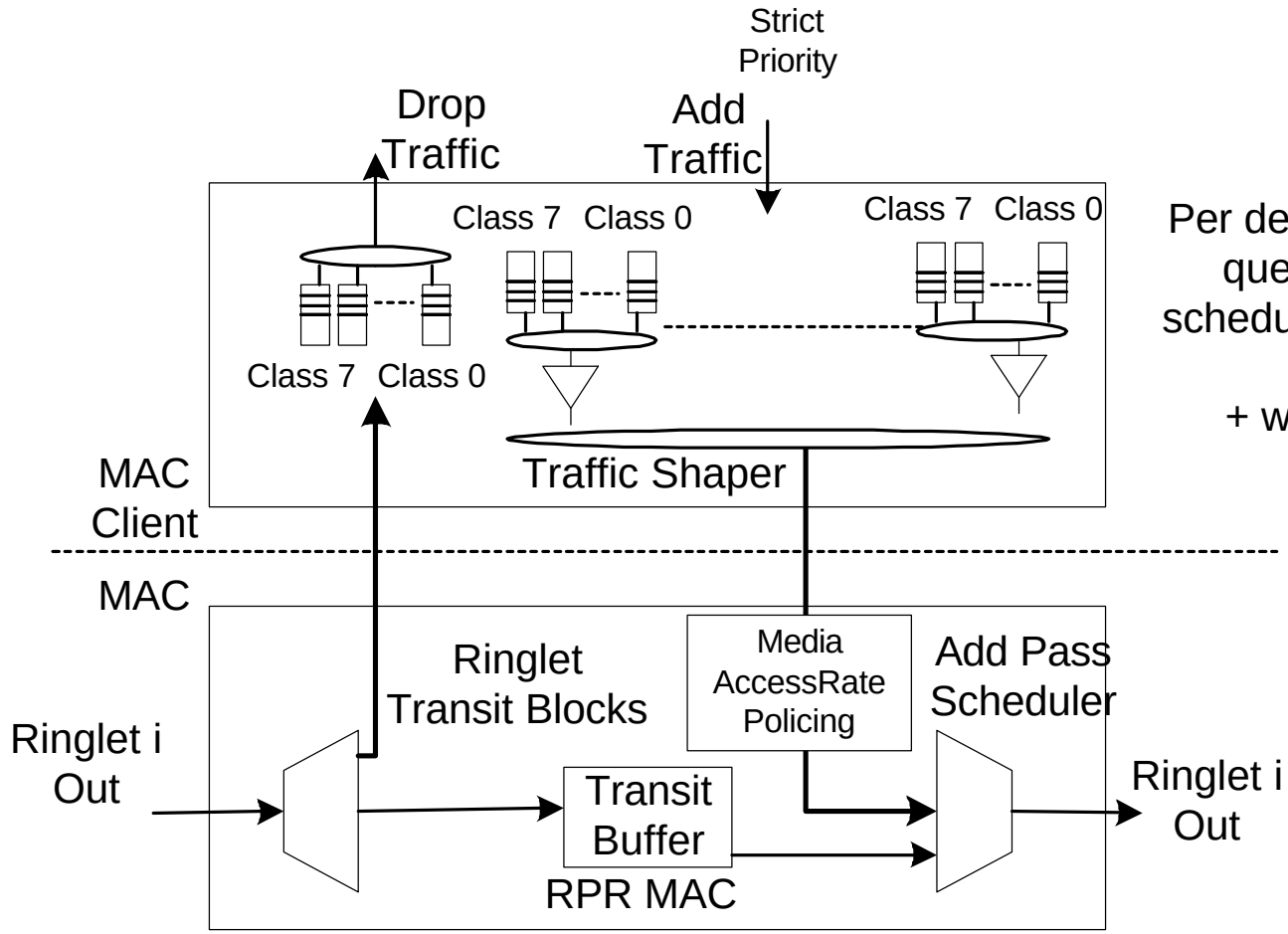
# MAC Client Add/Drop Path



# MAC Client Add/Drop Path: Virtual Output Queuing

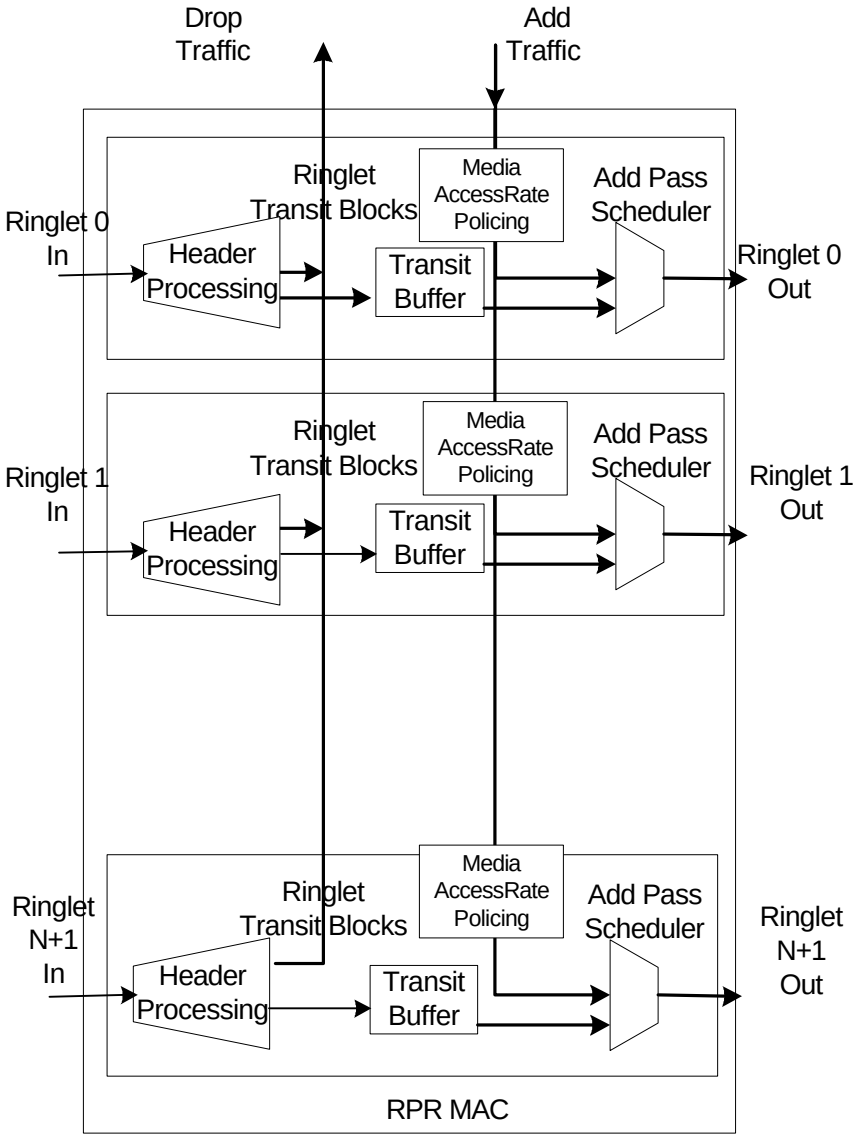


# MAC Client Add/Drop Path: Class of service based queuing per destination




Per destination class based queuing, strict priority scheduling + per destination shaper
+ weighted scheduler
for ringlet i

# Transit Path: Multiple Ringlets Option



# Conclusion

- Conforms to the 802 shared MAC architecture
- Scalable for high-link speeds.
- Cost effective solution that minimizes the cost of silicon implementation.

# Media Access Rate Control

*At each 10usec interval*

*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) < 15,000,000*

*segment_credit += fj*

*if ( segment_credit) < 0 // client BW exceeds limit*

*assert PAUSE.indicate*

*end FOR*

*At 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 -= frame_length*10,000*

*end FOR*