Access Control Algorithm for RPR MAC

Kanaiya Vasani
Anoop Ghanwani

Lantern Communications

IEEE 802.17 Interim Meeting

Orlando, FL

Outline

- Overview
- MAC architecture
- Components of the bandwidth management entity
- Complexity analysis
- Conclusions

The Need for Access Control

Control access to shared medium in order to

- Prevent collisions and congestion on the shared medium
  - e.g. On-ramp controls on freeways
- Make efficient use of the medium

RPR is a Closed Loop Control System

- Detect level of activity on the medium
- Regulate access to the medium based on this feedback
- All proposals under consideration are closed loop systems

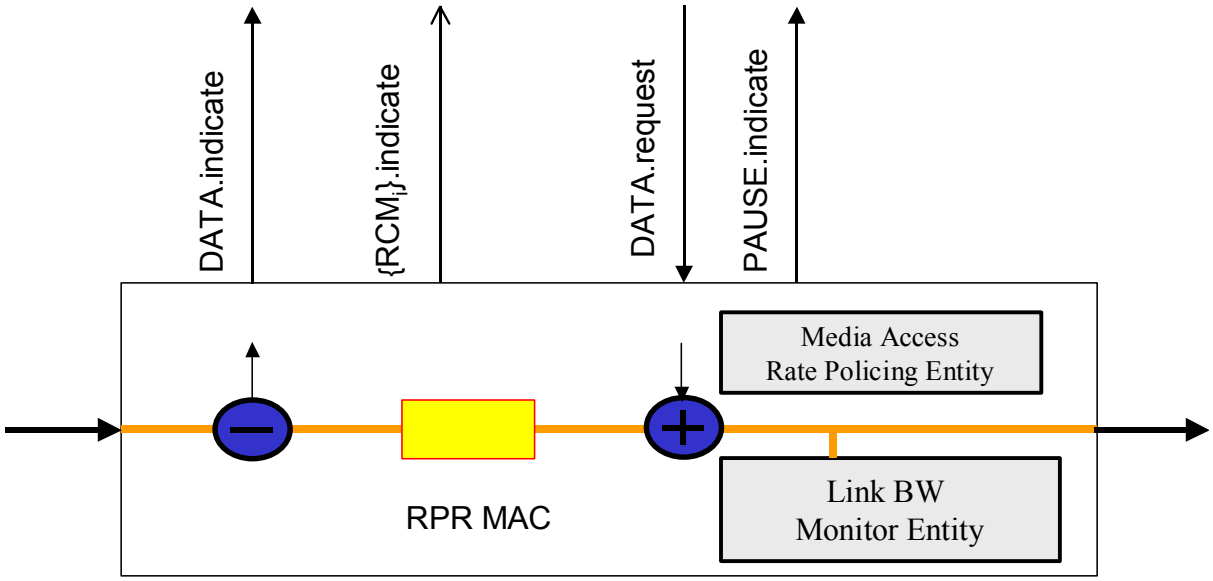
System Performance

- Performance of a closed loop system depends on
  - Precise feedback
  - Timely feedback
  - Rapid response

Feedback Provided to Each Node

- Provides each node its fair share of available bandwidth on a ring segment
- For this, one needs 2 pieces of information
  - The number of sources transmitting
  - The available bandwidth (this is a constant number)

MAC Model


DATA.indicate
{RCM$_i$}.indicate
DATA.request
PAUSE.indicate
Media Access
Rate Policing Entity
RPR MAC
Link BW
Monitor Entity

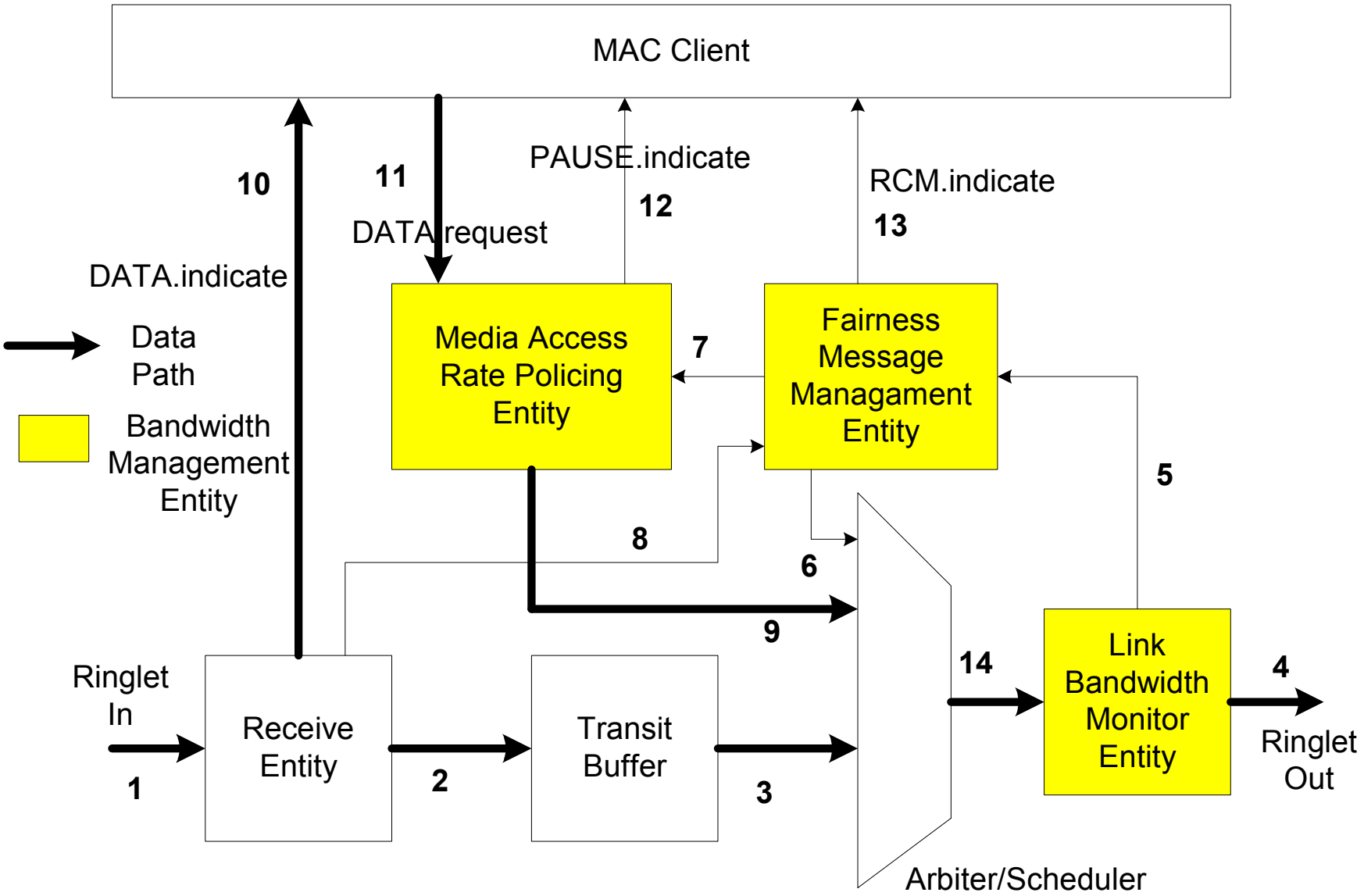
Objectives for a MAC Fairness Algorithm

- Ability to support multiple traffic types
- Provide source-based weighted fairness
- Maximize network utilization
  - Support VoQ-capable and non-VoQ capable clients
- Fast response time
- Stable
- Lossless
- Simplicity
- Scalable (size, speed, multiple ringlets)

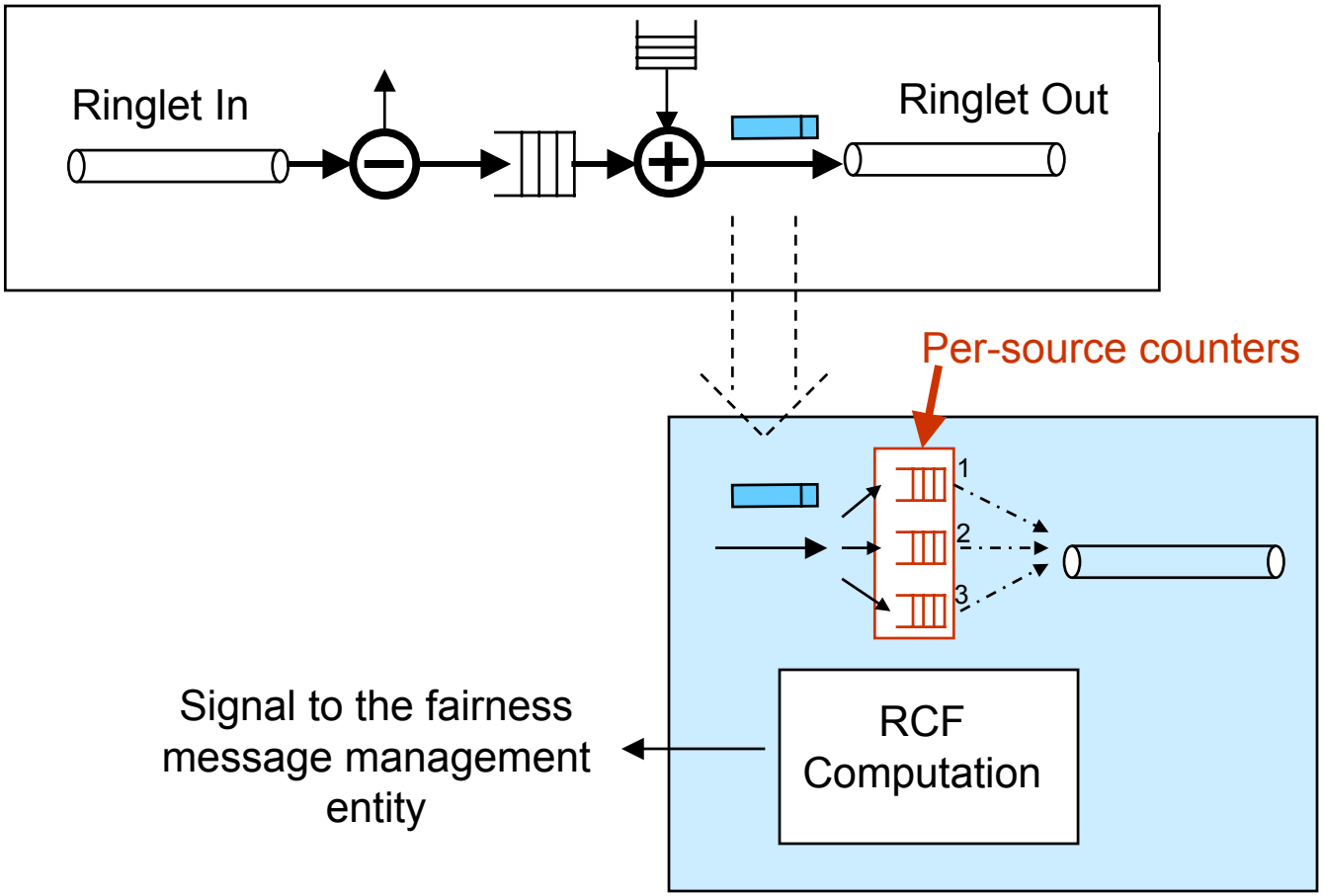
The MAC Fairness Algorithm

- The bandwidth management entity implements a 3-step closed-loop process:
  - Link bandwidth monitor entity
    - Monitor traffic and calculate rate information
  - Fairness message management entity
    - Distribute the rate information to all nodes
  - Media access rate policing entity
    - Use the rate information for access control

MAC Architecture



Link Bandwidth Monitor Entity



What is the Rate Control Factor (RCF)?

- A factor sent to all nodes to convey rate information

  RCF = (AvailableBW – Sum Ri) / Sum Wi

  where the sums are computed over *active* flows only

  AvailableBW = Link speed – guaranteed BW

  Ri is the committed rate for source i

  Wi is the weight for source i

- Used by each node to compute its weighted fair share of BW

  Fair share of source i on a segment

  Fi = Ri + Wi * RCF of that segment

- RCF allows us to "compress" information sent
  - We could have done the computation and sent 255 Fi's!

Link Bandwidth Monitor Entity (Concept)

- Maintain a per-source bucket that increments by the number of bytes in each packet from that source

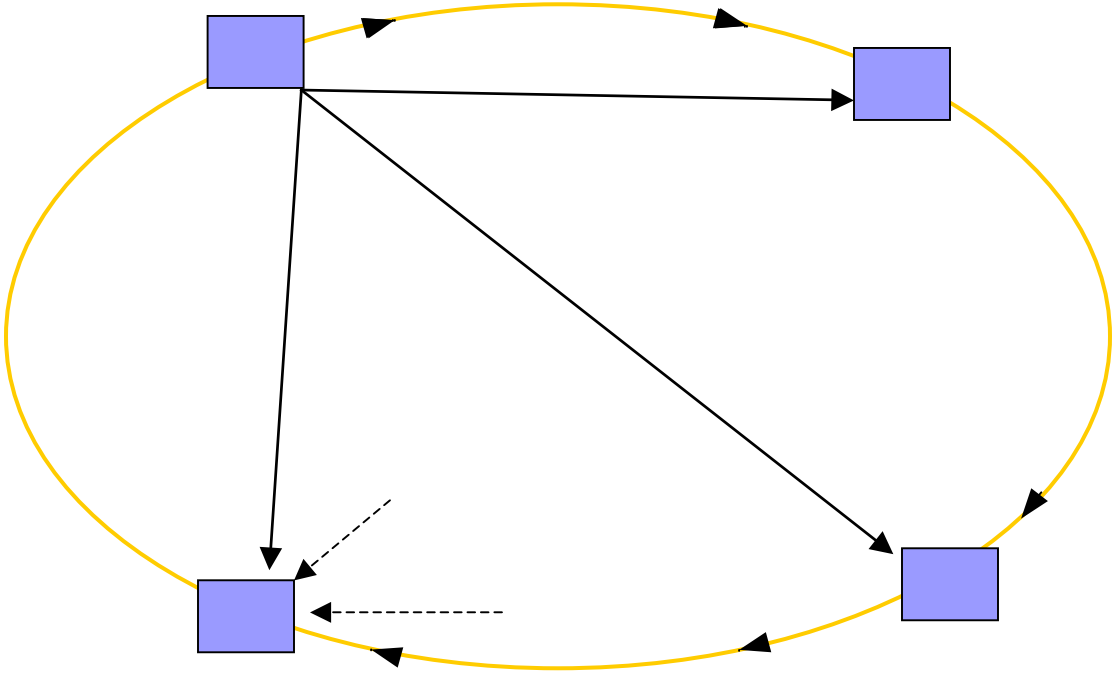
- At each rate measurement interval
  - Drain the bucket based on the fair rate for the source as computed using the previous RCF
  - Compute the new RCF

Link Bandwidth Monitor Entity – Pseudo-code

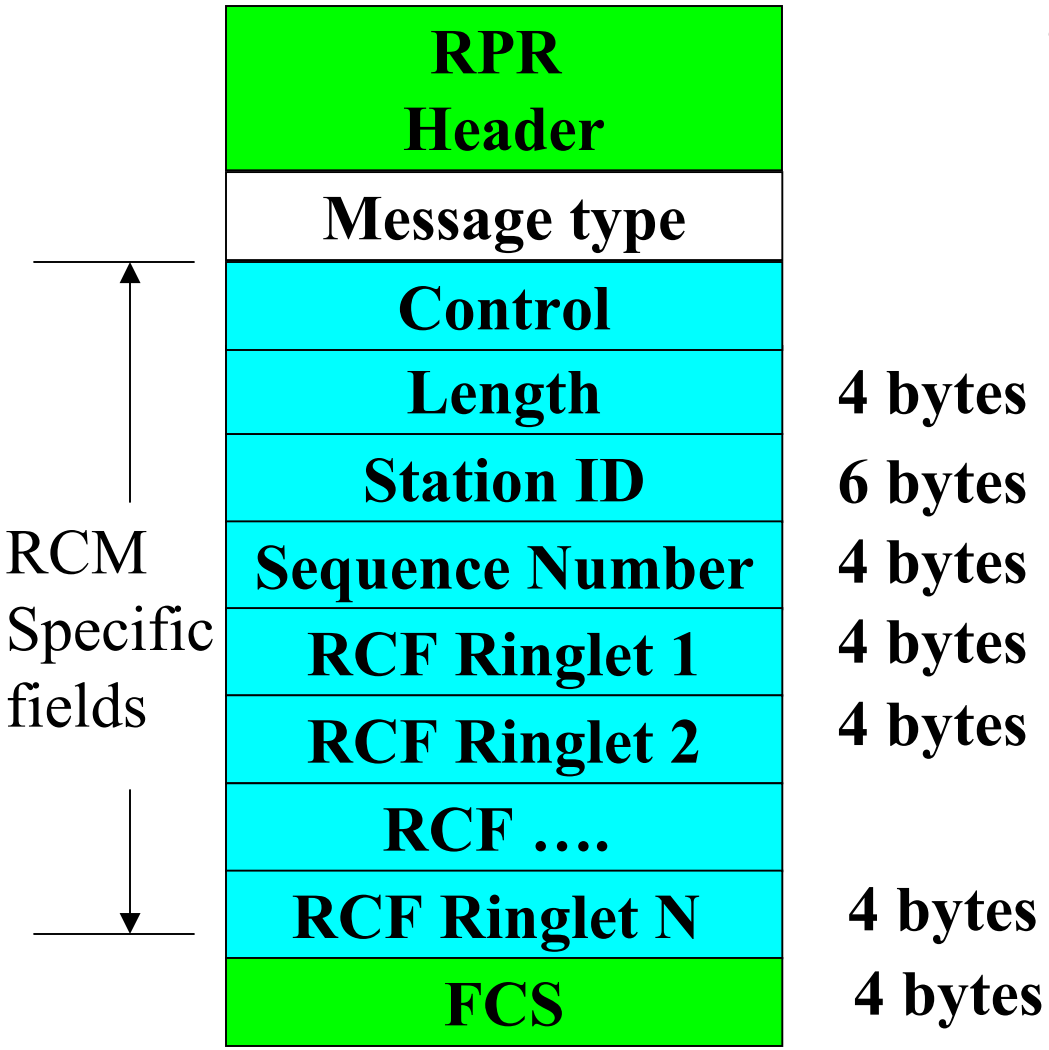
```
@packet arrival  /* Traffic monitoring */
  bucket[source_node] += packet_length;

@calcInterval (Recommended 1 usec at 10G)  /* calculate RCF */
  sum_R = 0;
  sum_W = 0;

  for (i = 0; i < NumNodes; i++) { /* for each source node */
     if (bucket[i] > 0) { /* do only for active flows */
        sum_R += R[i];
        sum_W += W[i];
     }
     bucket[i] -= calcInterval * (R[i] + W[i] * RCF); /* drain bucket */
     if (bucket[i] <= 0) bucket[i] = 0;
  }
  RCF_sampled = min(((AvailableRingBW – sum_R)/sum_W),AvailableRingBW);
  RCF_old = RCF;
  RCF = a * RCF_old + (1 - a) * RCF_sampled; /* low pass filter */
```

Fairness Message Management Entity

- Responsible for sending and receiving rate control messages (RCMs)
- RCMs are periodically broadcast on both rings
  - Fast and robust to message loss
- Sends the RCM information to the MAC client if the client is VoQ-capable



Rate Control Message Format



- **Common control frame and message**
  - Message type = RCM
  - Control: Specific control bits – Version, etc.
  - Length: length of RCM packet
  - Station ID: Packet's source station address
  - Sequence Number: Detect packet loss
  - RCF: Rate Control Factor, one for each ringlet
  - FCS: Error detection for RCM

Media Access Rate Policing Entity

- Uses the RCFs from all nodes for access control (policing)
- Generates PAUSE.indicate to the client if the client exceeds its capacity
  - The PAUSE.indicate is generated per-segment in nodes that support VoQ
  - The node is blocked from transmitting only if the data would need to traverse a paused segment

Media Access Rate Policing Entity (Concept)

- Each node implements a leaky bucket policer for every segment on the ring based on the received RCFs
- If the bucket for any segment goes empty, that segment is paused
  - Client can no longer transmit data that needs to traverse that segment

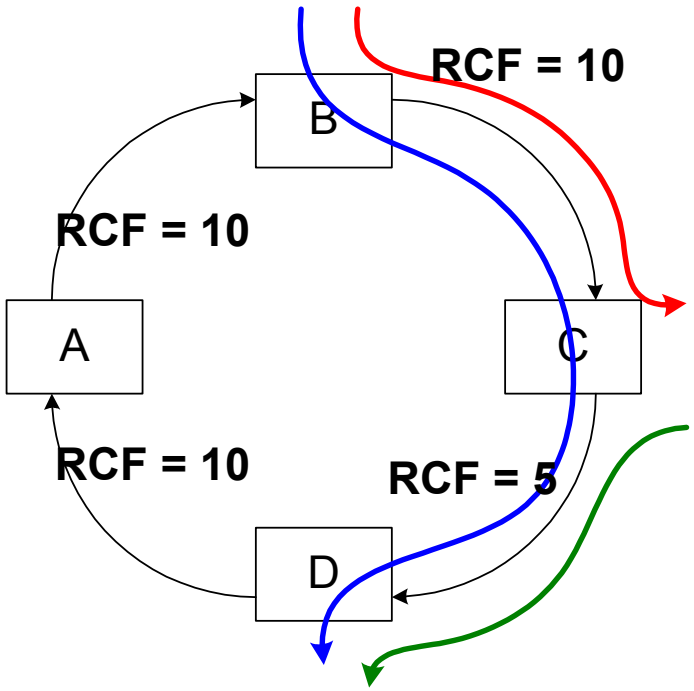
Media Access Rate Policing Entity – Pseudo-code (1)

```
@Tupdate  /* recommend 100 usec at 10G */
   for each (link segment i on the ring) {
      /* calculate the allowable bandwidth for this node */
      F = R[curr_node] + W[curr_node] * RCF[i];

      /* accumulate credits for each segment */
      segment_credit[i] += F * Tupdate;
      segment_credit[i] = MIN(segment_credit[i], MAX_CREDIT);

      /* has client has exceeded credit for segment? */
      if (segment_credit[i] < 0) {
         segment_paused[i] = TRUE;
         assert PAUSE.indicate for segment i;
      } else {
         segment_paused[i] = FALSE;
         clear PAUSE.indicate for segment i;
      }
   }
```

Media Access Rate Policing Entity – Pseudo-code (2)

```
@DATA.request from client /* at every transmission request */
  for each (link segment i between source and destination) {
      if (segment_paused[i] == TRUE) {
          reject DATA.request;
          return;
      }
  }
  accept DATA.request;
  for each (link segment i between source and destination) {
      /* deduct segment credit */
      segment_credit[i] -= packet_length;
  }
```

Tying it all Together: An Example



- Assume weights are all 1 and reserved/committed rates are 0
- One source active on segment 1
- Two sources active on segment 2
- RCFs for each link will be as shown
- At node B, for policing interval

  credits(seg1) = interval * 10

  credits(seg2) = interval * 5
- When sending a packet from B to C, decrement credits(seg1)
- When sending a packet from B to D, decrement both credit(seg1) & credit(seg2)
- If credit(seg1) becomes zero, no traffic can be sent
- If credit(seg2) becomes zero, only traffic from B to C can be sent

Simple and Intuitive

How do you measure complexity?

- Implementation complexity
  - Number of gates and registers
  - Amount of memory
- Logic complexity
  - Number of decision points in a flowchart

Implementation Cost and Complexity

- Memory
  - W[]: 256 nodes x 8 bits = 1 kbits (provisioned)
  - R[]: 256 nodes x 32 bits = 8 kbits (provisioned)
  - bucket[]: 256 nodes x 32 bits = 8 kbits
  - RCF[]: 256 nodes x 32 bits = 8 kbits
  - segment_credit[]: 256 nodes x 16 bits = 4 kbits
- Logic
  - ~40K gates for the link bandwidth monitor entity (monitoring and fair rate computation)
  - ~30K gates for the media access rate policing entity (access control policing)

Summary

- Simple and intuitive MAC algorithm
  - Monitor traffic and compute rate information
  - Broadcast the rate information to all nodes
  - Control access using rate information received
- Very fast response time and convergence
  - Provides accurate and timely feedback
- Easy to implement
  - ~70,000 gates