Institut für
Kommunikationsnetze

Critical Review of all RPR MAC Proposals

Harmen R. van As, Arben Lila, Guenter Remsak, Jon Schuringa
Vienna University of Technology, Austria

RPR MAC Proposals

Gandalf (Cisco, Corrigent Systems, Jedai Broadband Networks)

Darwin (Cisco)

Alladin (Alcatel, Dynarc, Lantern, Luminous, NEC, Nortel, Vitesse)

DVJ (Cypress Semiconductor, University of Oslo)

IKN (Vienna University of Technology)

Required MAC Protocol Properties

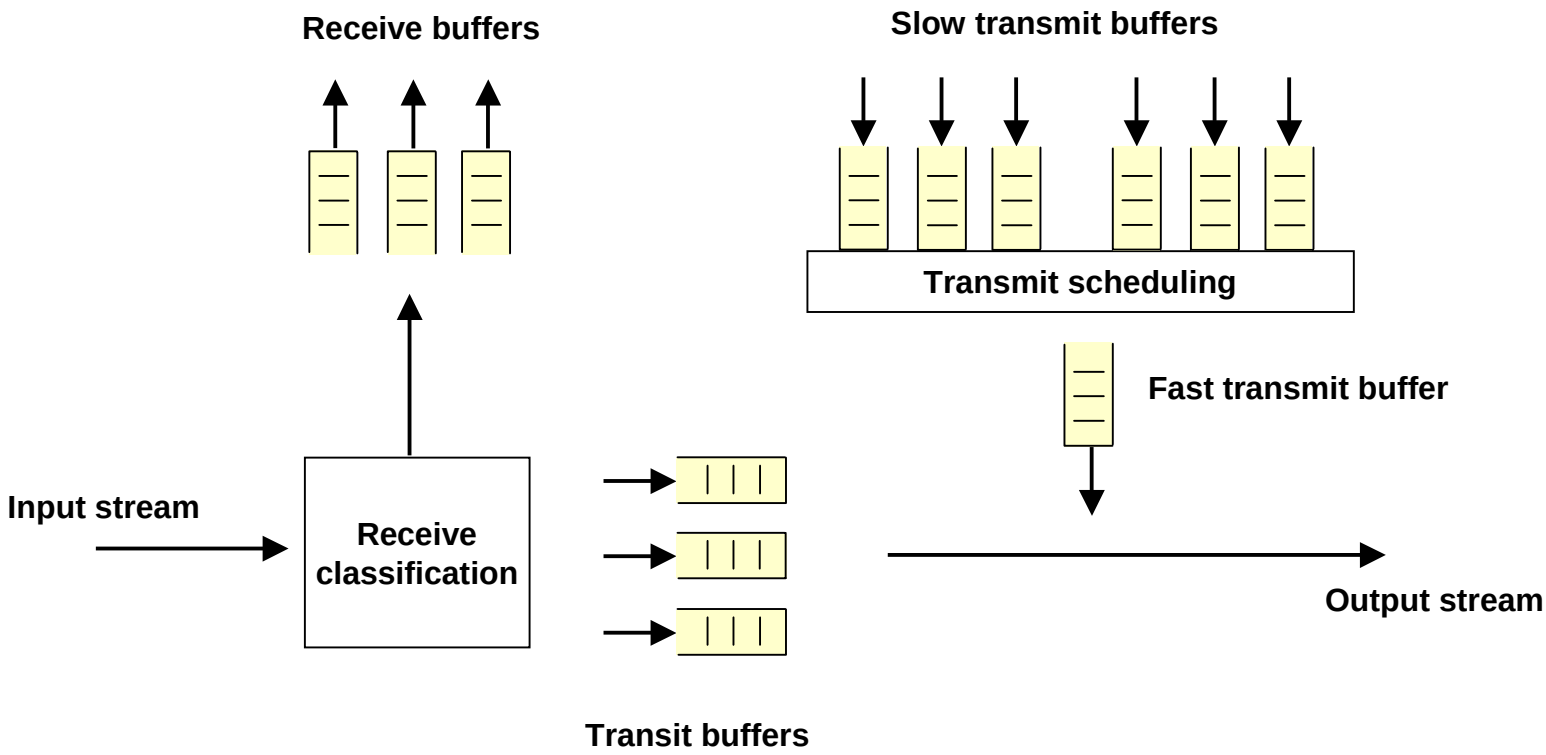
Support of:

- Link-fairness
- Service Level Agreements (SLAs)
- Heterogeneous link speeds

Performance properties:

- Control of flow-based source-destination traffic
- No HOL blocking
- Very high ring throughput
- Low delays
- No losses
- No backpressure required
- Insertion buffer occupancy at most one or two MTU sizes
- Link fairness
- Adaptive to traffic dynamics

Basic Structure of Stations



Gandalf: Support of three Priorities

High Priority

- Guaranteed bandwidth (provisioned)
- Bounded delay and bounded jitter

Medium Priority

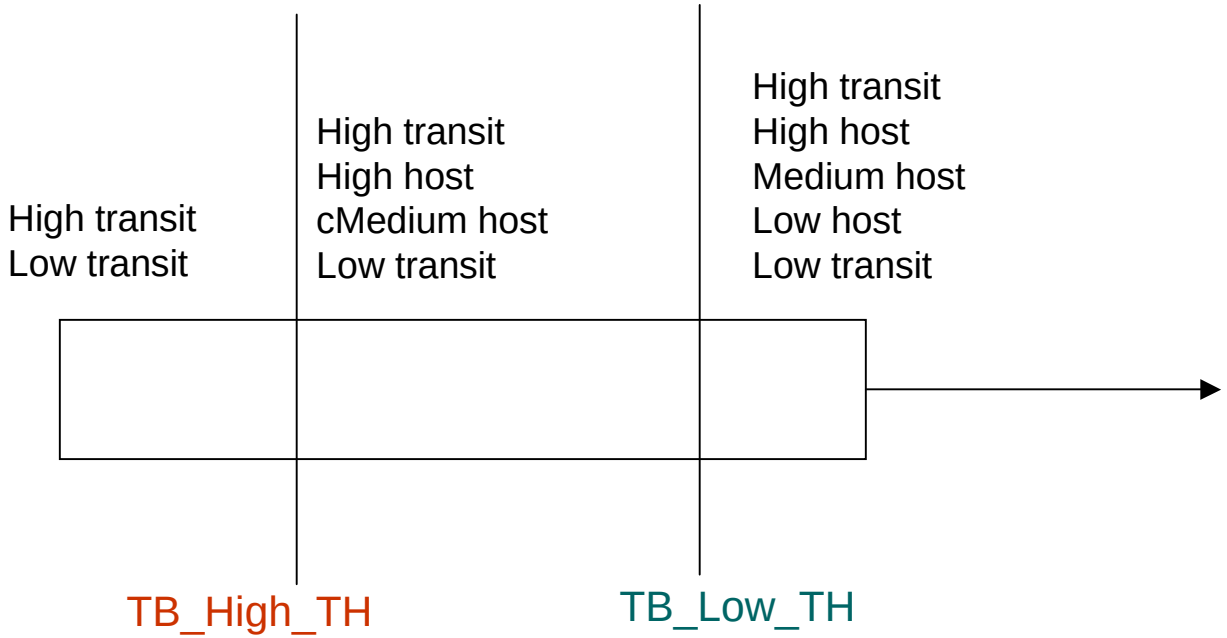
- Committed Access Rate (CAR) for MP (cMP)
- MP Traffic exceeding CAR (eMP) is subject to fairness algorithm control in the transmit path
- Committed bandwidth (provisioned), best effort for excess traffic
- Bounded delay and (loosely) bounded jitter

Low Priority

- No guarantees
- Best effort for bandwidth, delay and jitter

Gandalf: Transmission Scheduling

Transmission order :



Gandalf: Control Packet Generation

- **Low Transit Buffer congestion status**

```
congested = (lo_tb_depth > TB_LO_THRESHOLD/2)
   || ((my_rate + fwd_rate) > (MAX_LRATE - reserved_rate))
```

- **If congested, signal the smallest rate to throttle upstream transmit**

```
if (nlp_my_ rate < rcvd_ rate)
        upstream_ rate = nlp_my_ rate;
else
        upstream_ rate = rcvd_ rate;
```

- **If not congested but some downstream node is congested which is caused by upstream node, pass on received rate to throttle upstream**

```
if ((rcvd_ rate != NULL) && (lp_fwd_rate > allow_ rate_congestion / WEIGHT)
        upstream_ rate = rcvd_ rate;
```

- **Otherwise, signal null rate to upstream nodes**

```
upstream_ rate = NULL
if (upstream_ rate > MAX_LRATE)
        upstream_ rate = NULL
```

Gandalf: Rate Counters

- **Transmit Rate Counter: my_rate**
  - – Incremented when transmitting low and exceed medium priority transmit packets

    my_rate = my_rate + packet_len
  - – decremented by a fixed fraction at decay interval

    my_rate = my_rate - min(allow_rate/AGECOEFF, my_rate/AGECOEFF)
- **Threshold Counter: allow_rate and max_allow**
  - – allow_rate set to feedback rate from downstream neighbors
  - – allow_rate can decay upwards to max_allow if null rate is received

    allow_rate_congestion += (MAX_LRATE - allow_rate_congestion) / (LP_ALLOW)
  - – max_allow is statically pre-configured
- **Transit Rate Counter: fwd_rate**
  - – Incremented when transmitting low and exceed medium priority transit packets ?

    fwd_rate = fwd_rate + packet_len
  - – decremented by a fixed fraction at decay interval

    fwd_rate = fwd_rate - fwd_rate / AGECOEFF

Gandalf: Variables

- lo_tb_depth; //low priority transit buffer depth
- my_rate; //count of LP and eMP octets transmitted by client
- my_rate_congestion; //count of LP and eMP octets transmitted by client and destined beyond the congestion point
- lp_my_rate; //my_rate run through a low pass filter
- nlp_my_rate; //lp_my_rate / WEIGHT
- lp_my_rate_congestion; //my_rate_congestion run through a low pass filter
- nlp_my_rate_congestion; //lp_my_rate_congestion / WEIGHT
- my_rate_ok; // flag indicating that host is allowed to transmit
- allow_rate_congestion; // the fair amount each node is allowed to transmit beyond the congestion point
- forward_rate; //count of LP+eMP octets forwarded from the LP transit buffer
- lp_forward_rate; // forward_rate run through low pass filter
- lp_allow; // allow_rate run through low pass filter
- congested; //node cannot transmit host traffic without the TB buffer filling beyond its congestion threshold point.
- rcvd_rate; //the fair rate received from the downstream neighbor
- rev_rate; //the fair rate passed along to the upstream neighbor
- token_octets; // the number of octets in RPR-fa dynamic shaper
- fairness_pkt_t { char[6] SA; rate; } fairness_pkt_t;

Gandalf: Constants

WEIGHT; //configured weight for this node

BUCKET_SIZE; // provisioned RPR-fa dynamic shaper leaky bucket size in octets

MAX_ALLOWANCE; //configured value for max allowed rate for this node

DECAY_INTERVAL; //8,000 octet times @ OC-12, 32,000 octet times @ OC-48,
128,000 octet times @ OC-192

AGECOEFF = 4; // Aging coeff for my_rate and fwd_rate

LP_FWD = 64; // Low pass filter for fwd_rate

LP_MU = 512; // Low pass filter for my fair rate

LP_ALLOW = 64; // LP filter for allow rate auto increment

LP_ALLOW_COEF = 128; // low-pass filter for lp_allow

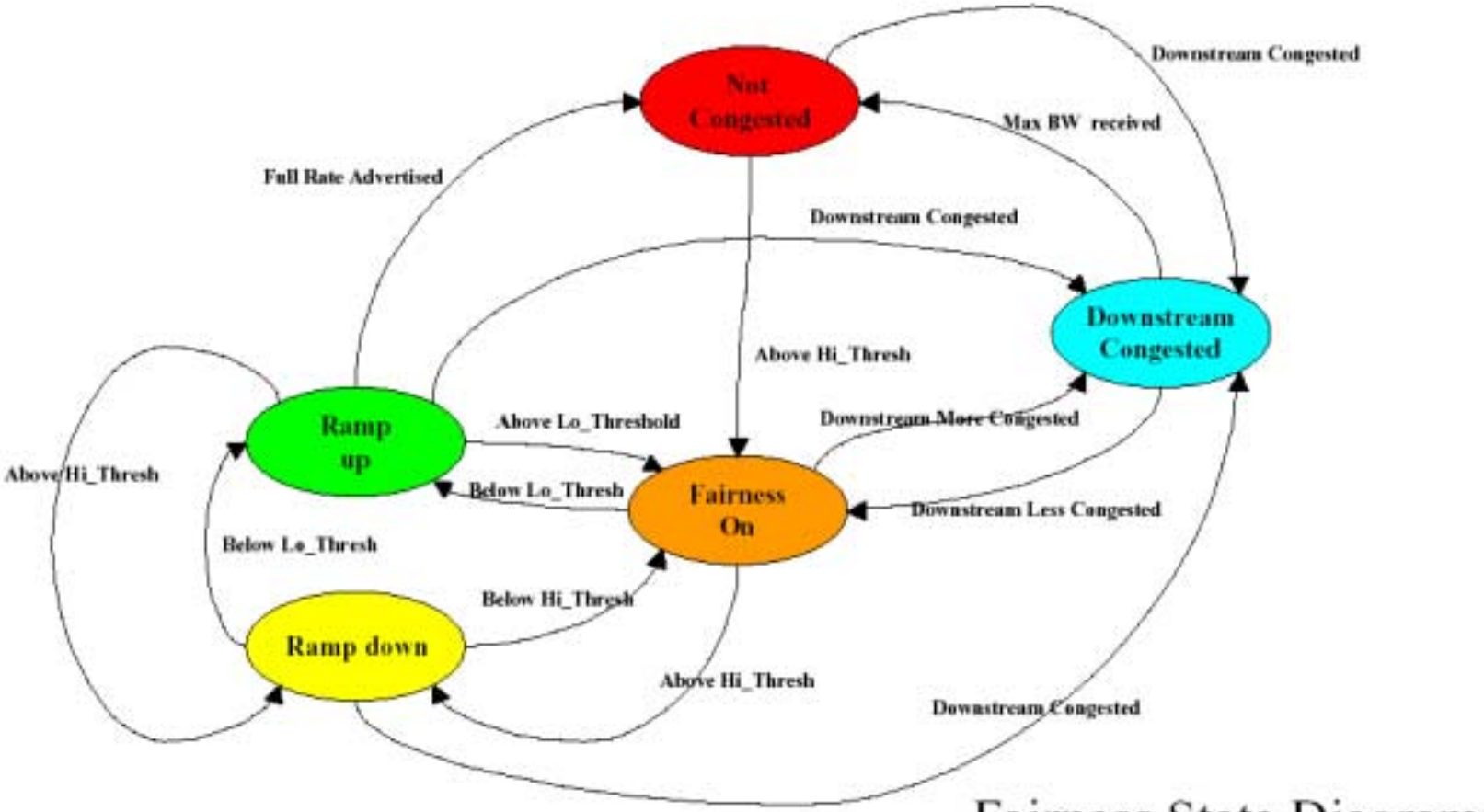
NULL_RCVD_INFO = 65535; //All 1's in rcvd_rate field

TB_LO_THRESHOLD; // TB depth at which no more LP host traffic can be sent

MAX_LRATE; //AGECOEFF * DECAY_INTERVAL = 512,000 for OC-192 128,000 for OC-48
32,000 for OC-12

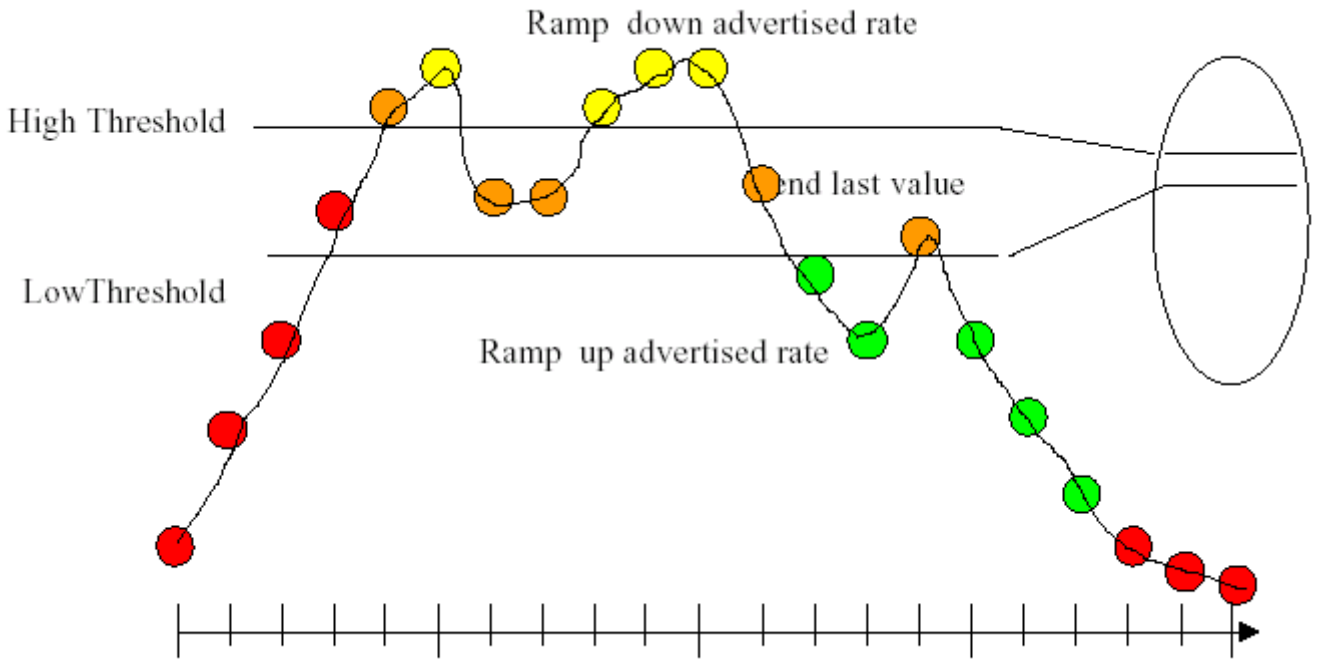
reserved_rate; //high priority reserved rate

Fairness State Machine

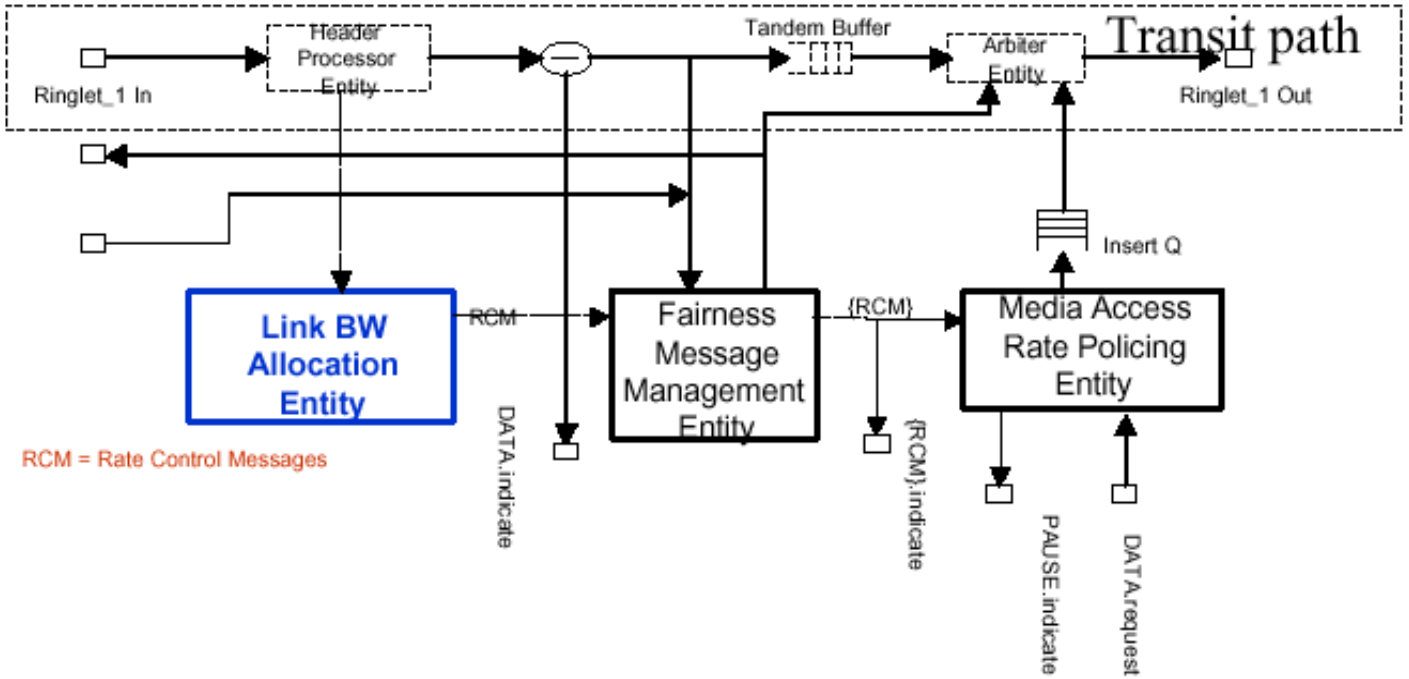


Fairness State Diagram

Hysteresis

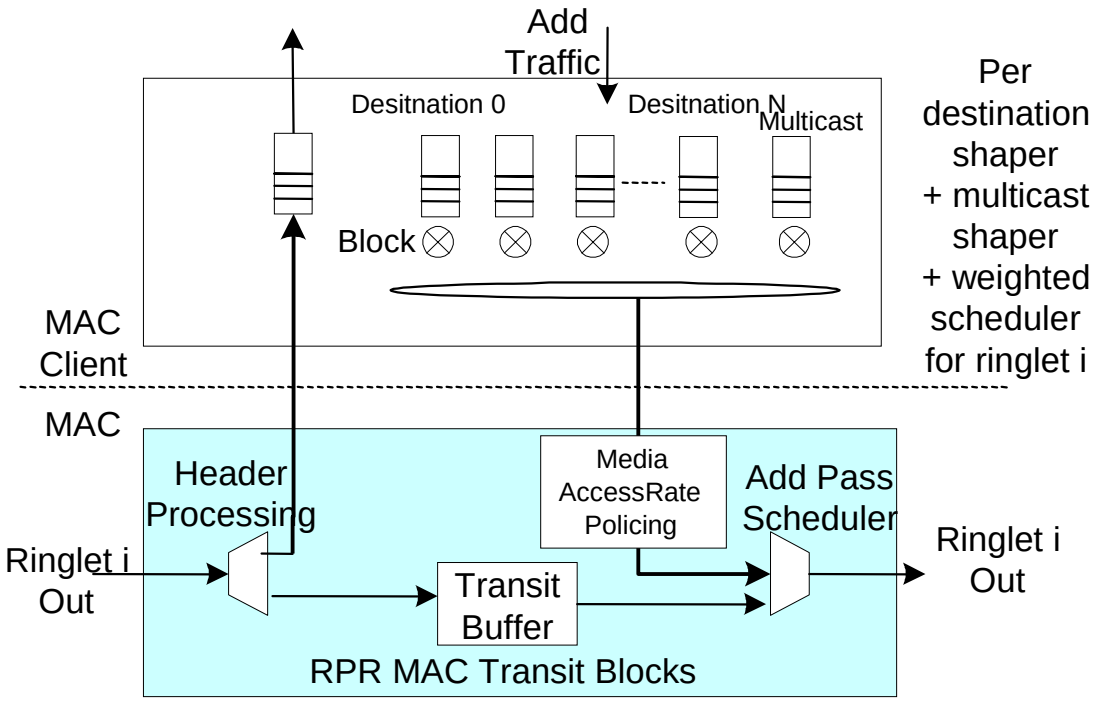

Ramp down advertised rate
High Threshold
nd last value
LowThreshold
Ramp up advertised rate

Alladin: Link BW Allocation Entity



1. Monitors the link utilization per active source to determine the activity and rate at which each source MAC puts traffic on the segment
2. Calculates the weighted fair rate to be advertised to other stations

Alladin: Station



Alladin: BW Monitoring

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

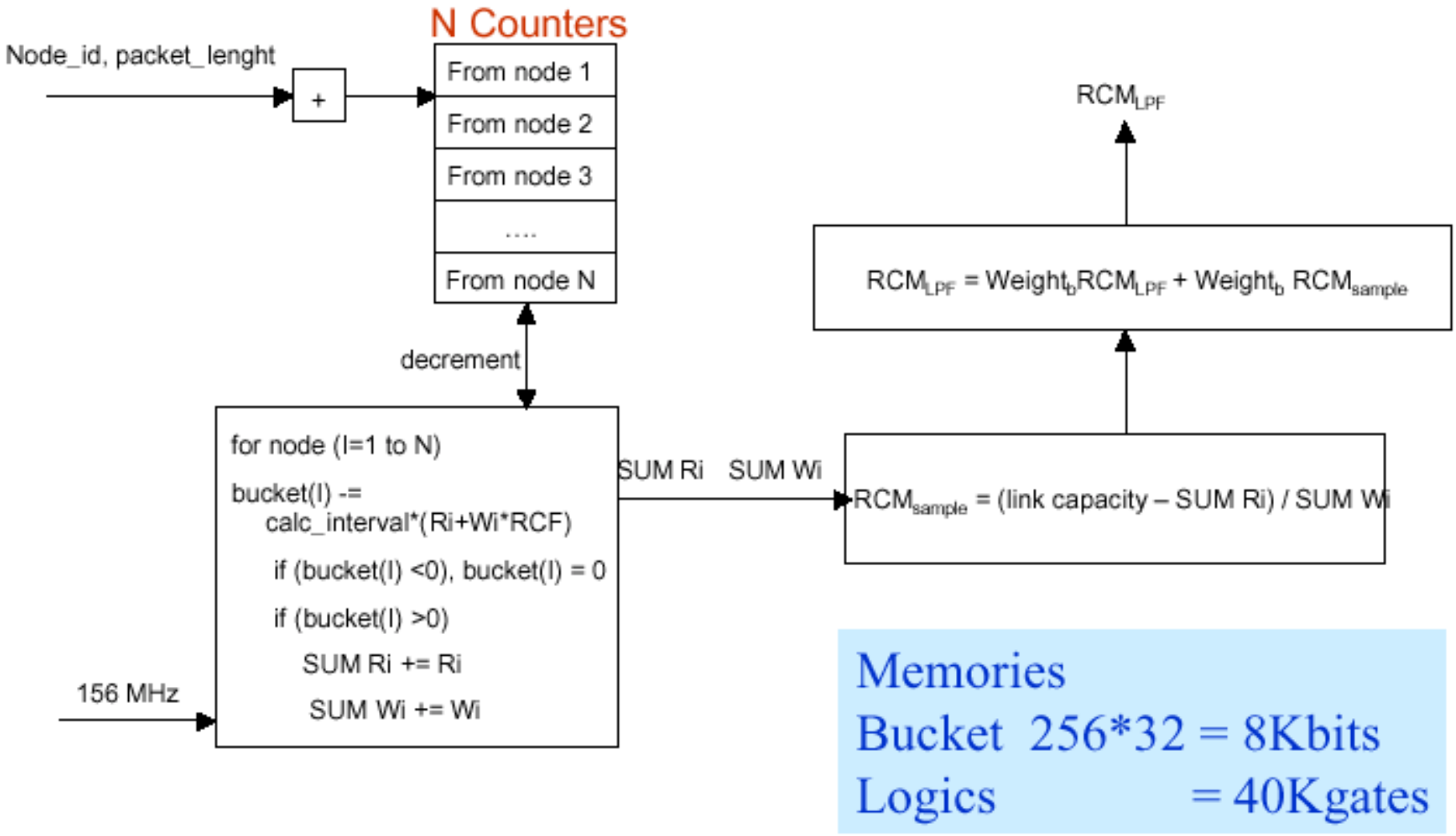
```
@calc_interval (1us)
    for each source node (I=1 to 256)
        if (bucket(I) >0)
            SUM Ri += Ri
            SUM Wi += Wi
        //drain bucket(i)
        bucket(I) -= calc_interval*(Ri+Wi*RCF)
        if (bucket(I) <0), bucket(I) = 0
    end FOR
    RCF = (link capacity – SUM Ri) /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

Register File

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

SUM Ri
SUM Wi
RCF
Calc Interval
Link Capacity

Alladin: Implementation Complexity



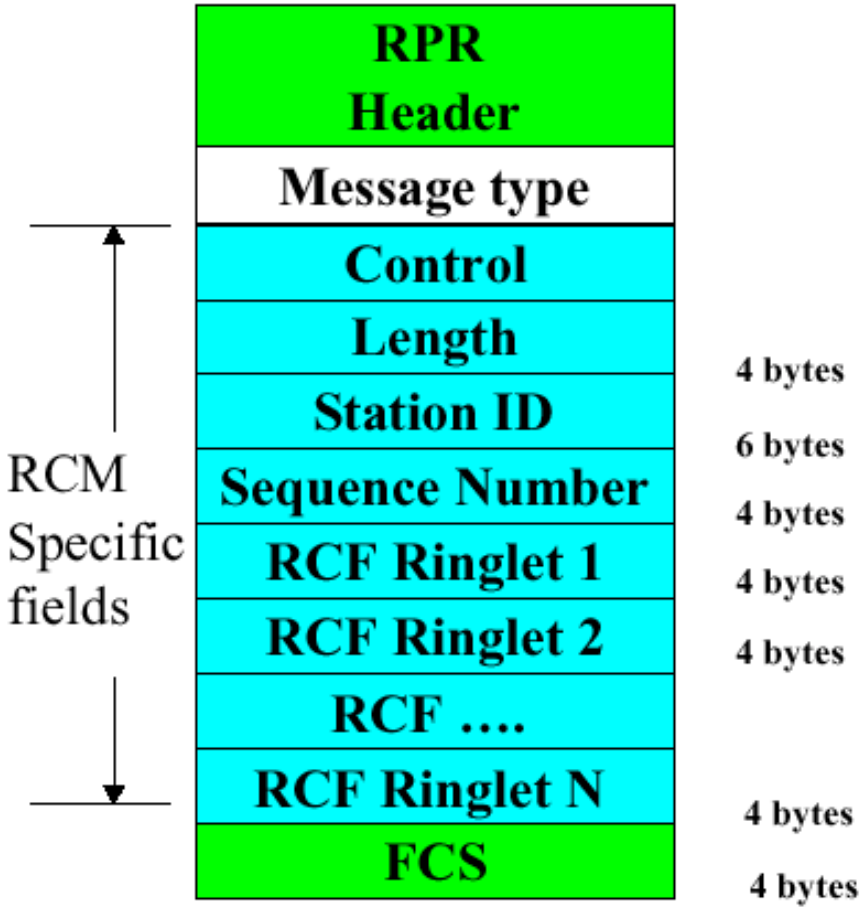
Alladin: Rate Control Factor

- Rate estimator on output link per source
- RCF sample : usage per sample window
- Filtering:

$$RCF_{new} = weight(t\text{-}1) * RCF_{old} + weight(t) * RCF_{sampled}$$

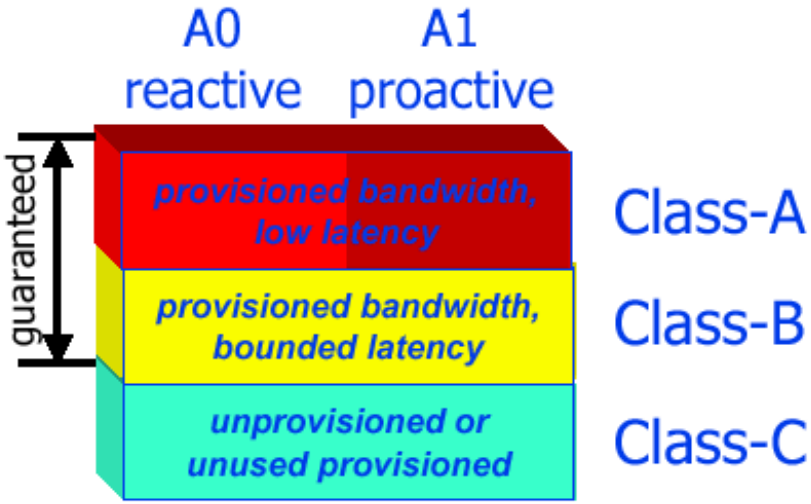
- The sampled RCF is an instantaneous value that may fluctuate greatly depending on traffic dynamic and the fact that the buckets may transit from an active state to inactive state frequently
- Filtering stabilizes and smoothes RCF that will be sent to other stations

Alladin: Rate Control Message Format



- **Common control frame and message type=RCM:**
  - Control: specific control bits-version etc.
  - Length: length of RCM packet
  - Station ID: packet source station address
  - Sequence number: message synchronization
  - RCF: rate control factors. One for each ringlet
  - FCS: error detection for RCM

DVJ Proposal



Traffic Classes

Packet Flow Direction

- Data packets flow in one direction
- Arbitration control flows in the other*

1) Simplicity

General

Gandalf
Darwin

Aladin

DJV

IKN

2) Operability

General

Gandalf
Darwin

Aladin

DJV

IKN

3) Testability

General

Gandalf
Darwin

Aladin

DJV

IKN

4) Extendability

General

Gandalf
Darwin

Aladin

DJV

IKN

5) Scalability

General

Gandalf
Darwin

Aladin

DJV

IKN

6) Traffic Dynamics

General

Gandalf
Darwin

Aladin

DJV

IKN

7) Robustness

General

Gandalf
Darwin

Aladin

DJV

IKN

8) Capabilities

General

Gandalf
Darwin

Aladin

DJV

IKN

9) Throughput Performance

General

Gandalf
Darwin

Aladin

DJV

IKN

10) Delay Performance

General

Gandalf
Darwin

Aladin

DJV

IKN

11) Fairness Performance

General

Gandalf
Darwin

Aladin

DJV

IKN

RPR MAC Comparison

	Gandalf	Darwin	Aladin	DVJ	IKN
simplicity					
operability					
testability					
extendability					
scalability					
traffic dynamics					
robustness					
capabilities					
throughput					
delay					
fairness					

Conclusion