CREDIT-BASED RATE FAIRNESS

DR. PHILIP AXER PHILIP.AXER@NXP.COM
DON PANNELL DONALD.PANNELL@NXP.COM
SUJAN PANDEY SUJAN.PANDEY@NXP.COM

FEB 2018

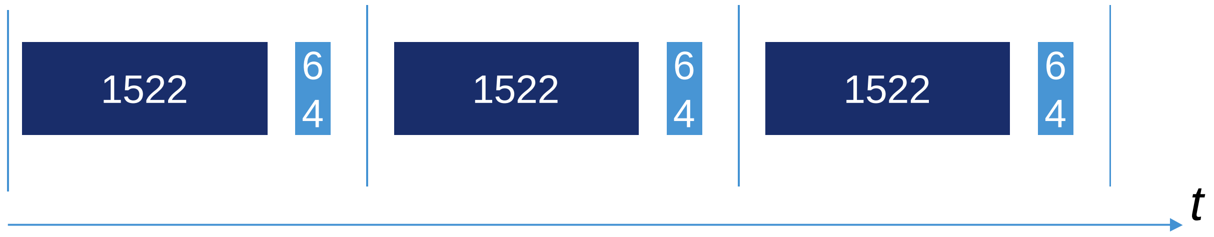
SECURE CONNECTIONS
FOR A SMARTER WORLD

Supporters

Problem Statement

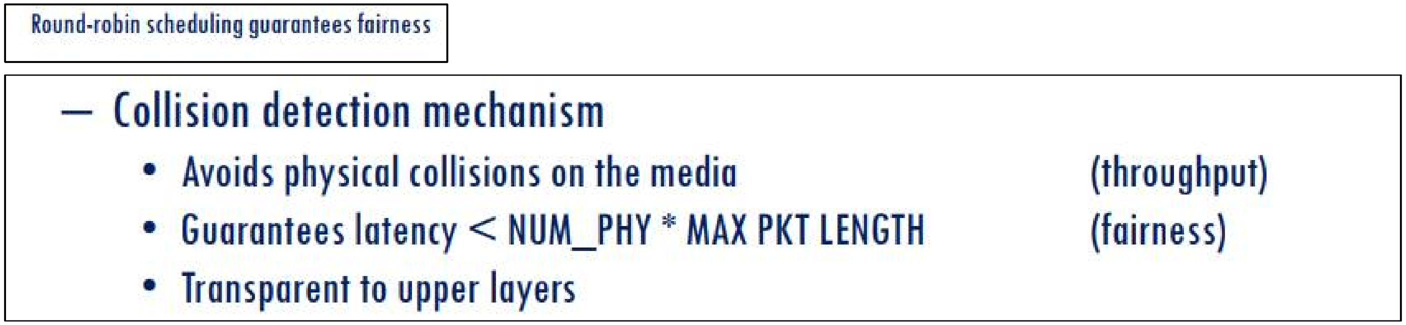
- Current PLCA proposal provides **frame-rate fairness** but not **data-rate fairness**
- **Example:** Assuming two nodes (A, B)
  - Frame sizes of **64 byte (A)** and **1522 byte (B)**
  - Achieved data rates are ~ **A 4%, B: 96%**

→Nodes which send a lot of small (control) frames are **penalized** significantly



- Is this 4% vs 96% data-rate fairness what all applications require?

Fairness in PLCA



Round-robin scheduling guarantees fairness

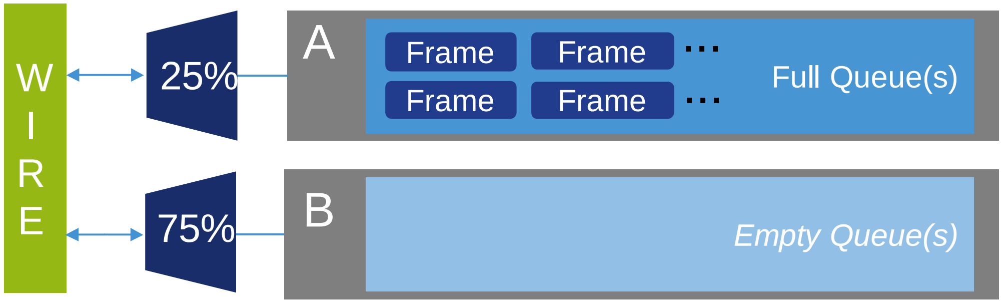
– Collision detection mechanism

- Avoids physical collisions on the media (throughput)
- Guarantees latency < NUM_PHY * MAX PKT LENGTH (fairness)
- Transparent to upper layers

- PLCA provides **bounded latency** due to round-robin scheme
- But bounded latency is **not** data-rate fairness
- PLCA is **starvation free** (assuming bounded MTU - Maximum Transmission Unit)
  - In any bus *exactly one* cycle a slot is guaranteed
- PLCA does not guarantee fair **rate** share

Rate-Limit Shaping is not a Solution

- Shaping (Qav, Leaky-Bucket,…) can address this issue
- But: each note is shaping the traffic in isolation
  → link capacity is not shared
- For example: Two nodes A (25% load), B (75% load)

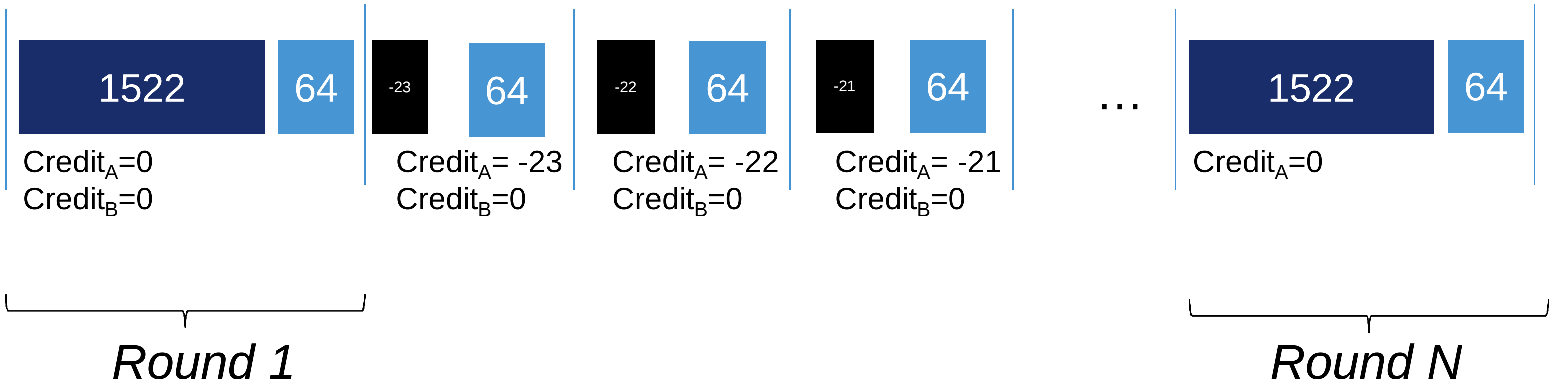


Observation: Node A is limited to 25% link capacity, but node B has no data
→ 75% link capacity is wasted

In a contention-free scenario node A should be able to get full link *capacity*

Credit-Based, Round-Robin Fairness - Mechanism

- PLCA „as is“ plus a credit counter per node
- TX of frame consumes credit (*here 1 credit per 64 byte as example*).
- Each Time slot replenishes credit (*here 1 credit per round as example*)
- Each node keeps track of other nodes credit
- Example:



1522 64
Credit$_A$=0
Credit$_B$=0

-23 64
Credit$_A$= -23
Credit$_B$=0

-22 64
Credit$_A$= -22
Credit$_B$=0

-21 64
Credit$_A$= -21
Credit$_B$=0

…

1522 64
Credit$_A$=0

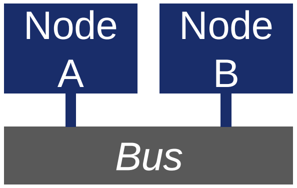
Round 1

Round N

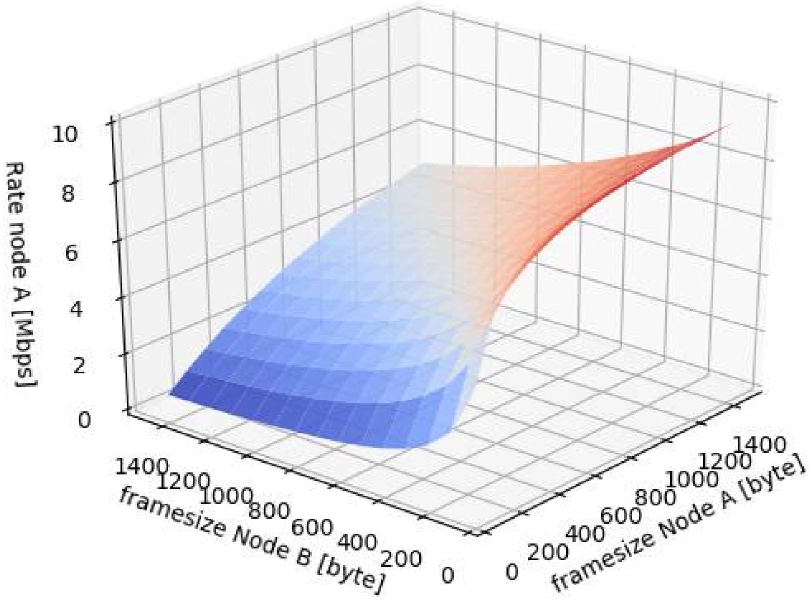
Note: Here, 1 credit = 64 bytes

Proposal: Credit-Based, Round-Robin Fairness - Results

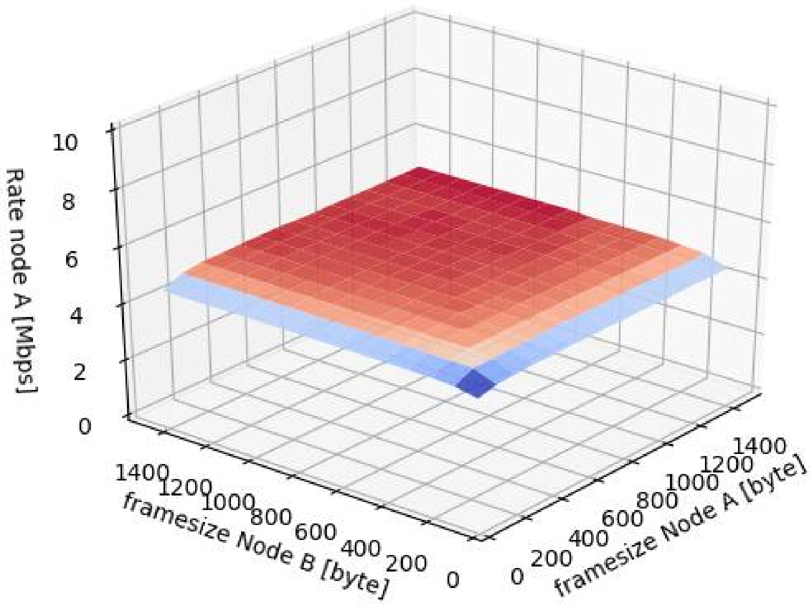
Example: Two node simulation
Each node transmits as much as possible
All frame size combinations are evaluated



Baseline PLCA



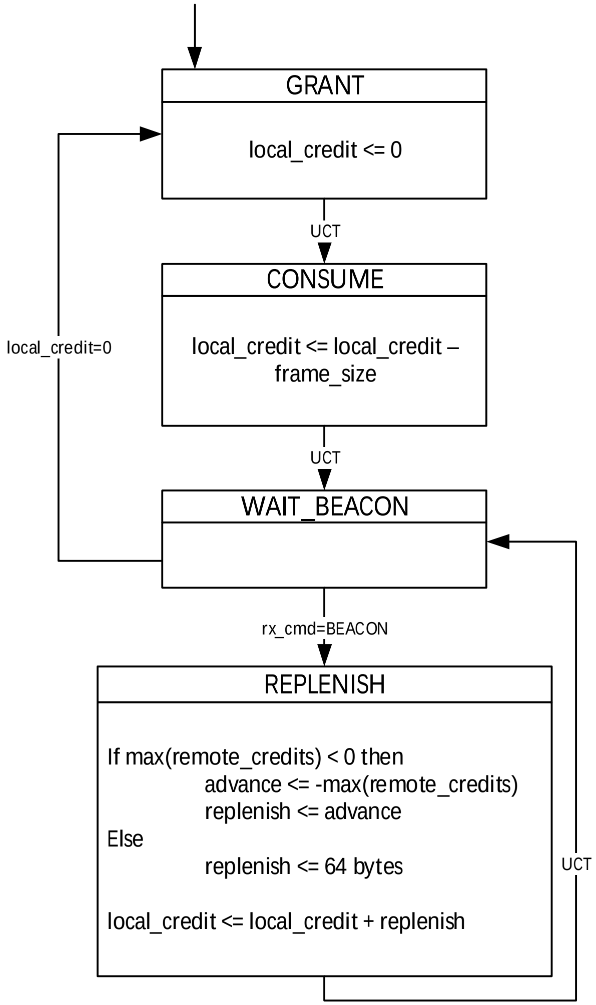
Credit-Based Fairness



Credit-Based, Round-Robin Fairness - Algorithm

- *Parameters and variables:*
  - *replenish_quota (global): credit [bits] replenished each round*
  - $credit_i$ *(per node): Credit level of node i in bits*
- **Transmission**
  - Node is only allowed to send if it's credit level is greater or equal to zero
    - $credit_i >= 0$ → grant
  - Credit is consumed after transmission
    - $credit_i := credit_i$ – framesize [bits]
- **Replenishment**
  - At the beginning of each round → credit is replenished with fixed budget (e.g. 64*8 bit)
    - $credit_i := credit_i$ + replenish_quota
- **Idle Saturation**
  - If a node has an empty queue and no pending transmissions, credit saturated at zero
    - level(queue)==0 && credit >= 0 → $credit_i := 0$
- **Advancement**
  - If *no* node is transmitting in a round and some nodes j are stalled, the credit is advanced to guarantee progress in the next round
    - $credit_i := credit_i - \max_{\{\text{for all stalled nodes } j\}}(credit_j)$ *Note: credits are negative if stalled, hence max operator*
- ***Notes/Observations***
  - Credit level is positively saturated at +MaxFramesize
  - Credit level is negatively saturated at –MaxFramesize
  - There is at most one round with no progress

Credit-Based, Round-Robin Fairness – State Machine



GRANT

local_credit <= 0

CONSUME

local_credit <= local_credit – frame_size

WAIT_BEACON

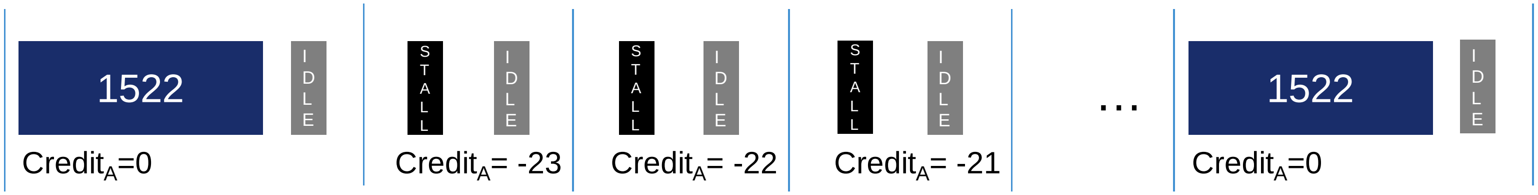
REPLENISH

```
If max(remote_credits) < 0 then
        advance <= -max(remote_credits)
        replenish <= advance
Else
        replenish <= 64 bytes

local_credit <= local_credit + replenish
```

Credit-Based, Round-Robin Fairness - Corner Cases

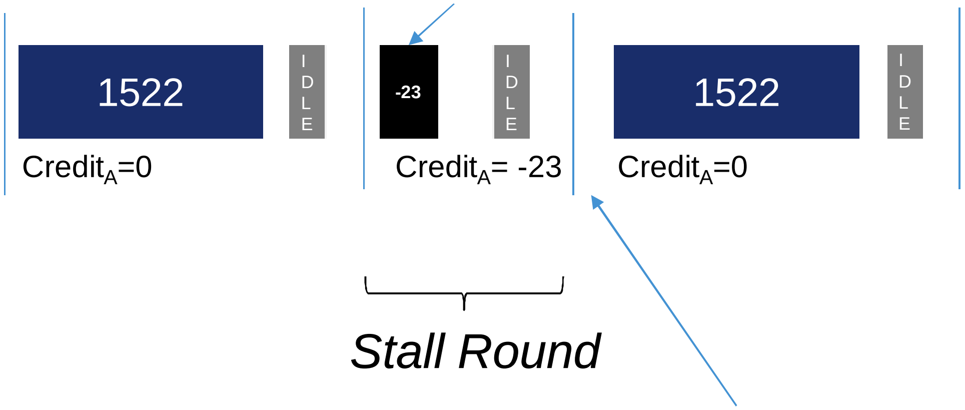
- Worst-case overhead if some nodes stall and others idle
- Example:



Credit-Based, Round-Robin Fairness - Corner Cases

- Efficiency improvement: nodes have a common view on credit levels
- This effectively deactivates shaping in case only one station uses the medium
- Example:

Credit level of Node A can be derived by all stations

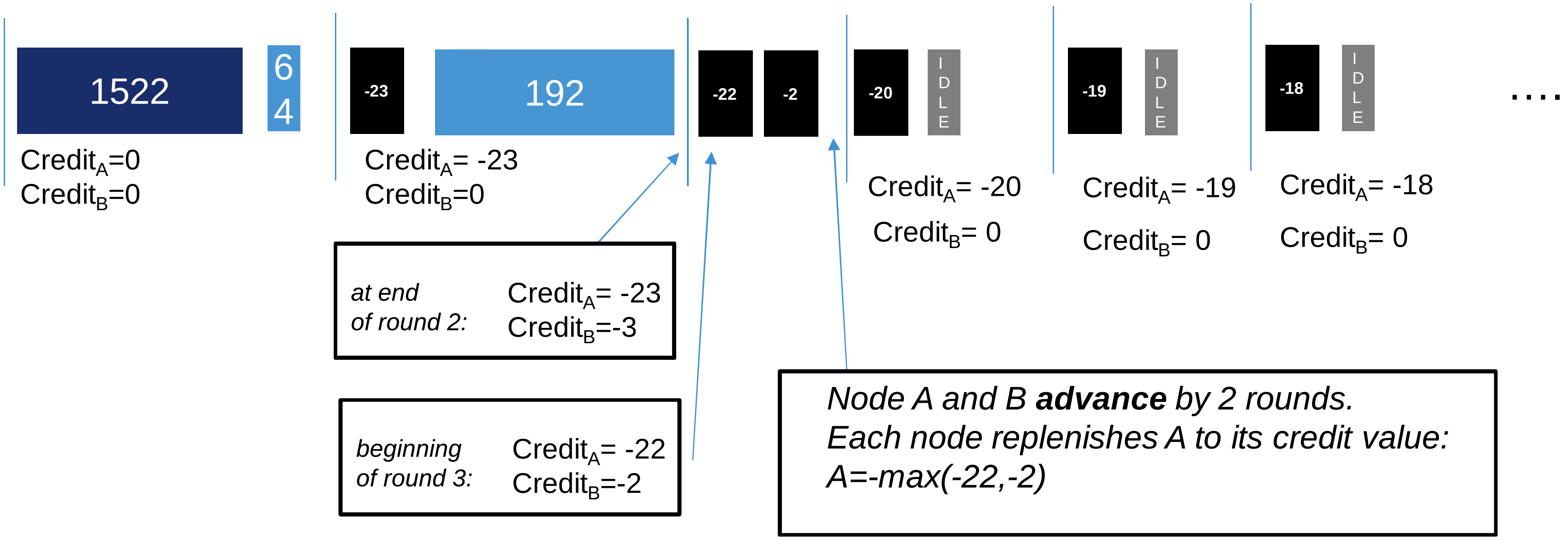


Stall Round

*Node A **advances** credit to 0, as no other node claims a slot*

Credit-Based, Round-Robin Fairness - Corner Cases

- Advanced example:



$\text{Credit}_A=0$
$\text{Credit}_B=0$

$\text{Credit}_A= -23$
$\text{Credit}_B=0$

$\text{Credit}_A= -20$
$\text{Credit}_B= 0$

$\text{Credit}_A= -19$
$\text{Credit}_B= 0$

$\text{Credit}_A= -18$
$\text{Credit}_B= 0$

at end of round 2: $\text{Credit}_A= -23$ $\text{Credit}_B=-3$

beginning of round 3: $\text{Credit}_A= -22$ $\text{Credit}_B=-2$

*Node A and B **advance** by 2 rounds.*
Each node replenishes A to its credit value:
A=-max(-22,-2)

Note: Here, 1 credit = 64 bytes

Conclusion

- Date-rate fairness is achievable with PLCA with the addition of:
  - Credit counters
  - Credit state machine
  - Miscellaneous
- What else should be modelled, simulated?

SECURE CONNECTIONS
FOR A SMARTER WORLD