PLCA DATA-RATE FAIRNESS

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

MARCH 2018

SECURE CONNECTIONS
FOR A SMARTER WORLD

Supporters

- Olaf Krieger, VW
- Alexander Meier, VW
- Gergely Huszak, Kone
- Helge Zinner, Continental
- Steffen Graber, Pepperl+Fuchs
- Mick McCarthy, Analog Devices
- Thomas Müller, Rosenberger

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

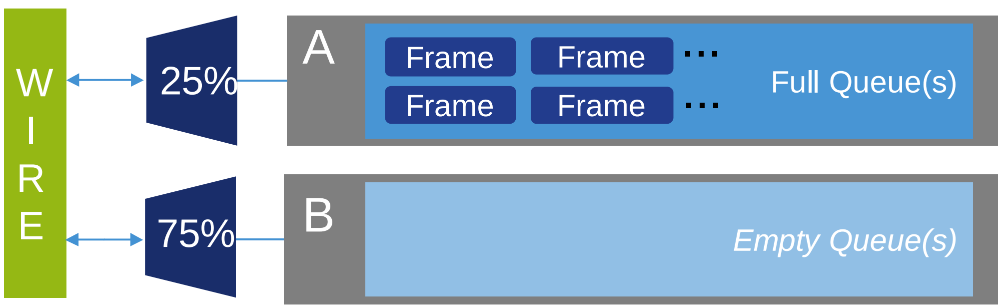
Round-robin scheduling guarantees fairness

– 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)

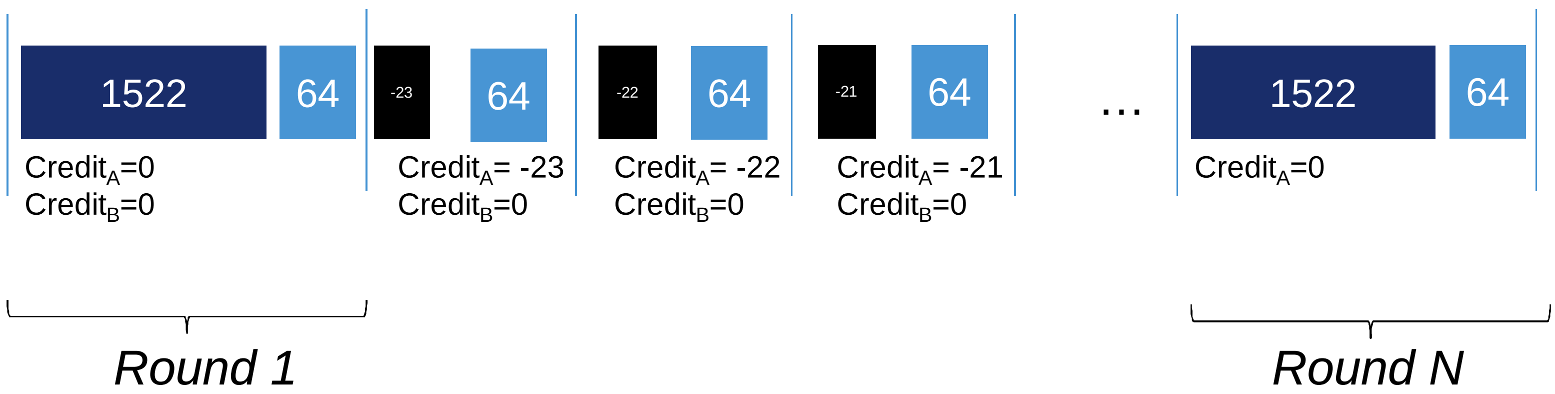


Observation: Node A is limited to 25% link capacity, but node B has no data
→ 75% link capacity is wasted

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

Credit-Based, Round-Robin Fairness - Mechanism

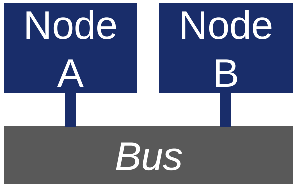
- PLCA „as is“ plus a credit counter per node
- TX of frame consumes credit (*here 1 credit per 64 byte as example*).
- Each Time slot replenishes credit (*here 1 credit per round as example*)
- Each node keeps track of other nodes credit
- Example:



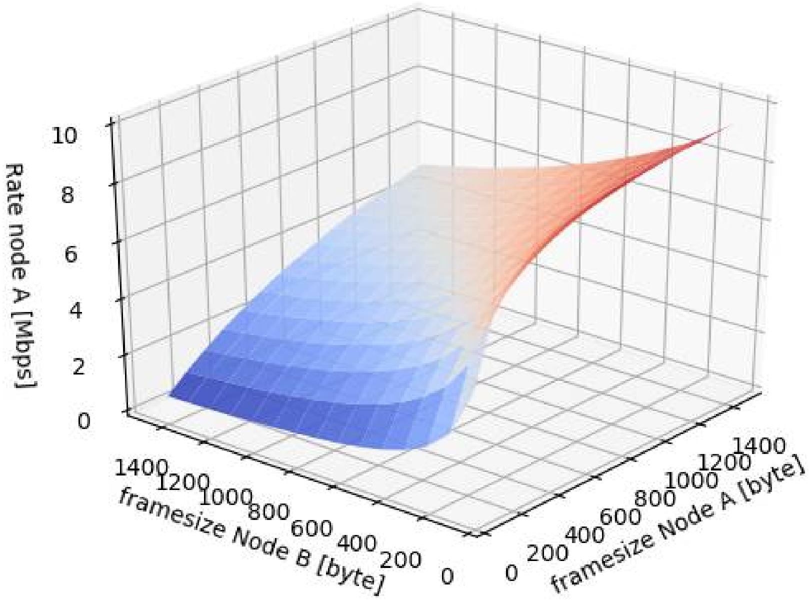
Note: Here, 1 credit = 64 bytes

Credit-Based, Round-Robin Fairness - Results

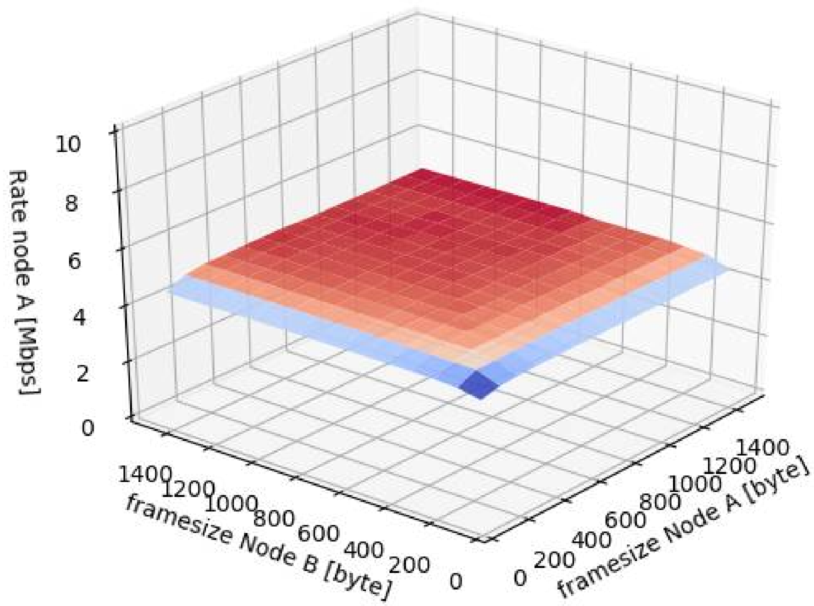
Example: Two node simulation
Each node transmits as much as possible
All frame size combinations are evaluated



Baseline PLCA



Credit-Based Fairness

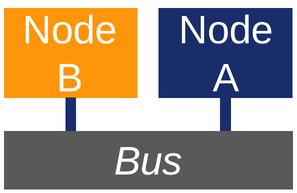


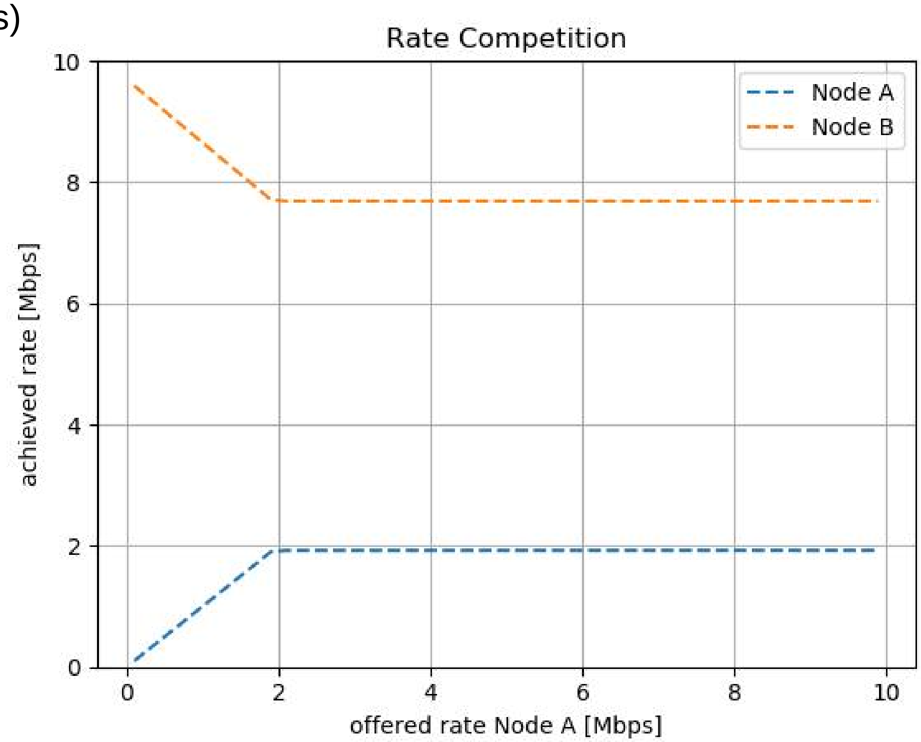
Competing Load – Frames/Sec Fairness (Baseline PLCA)

Experiment:

Offered load Node B: fixed 10Mbps (512 byte frames)

Offered load Node A: variable rate (128 byte frames)



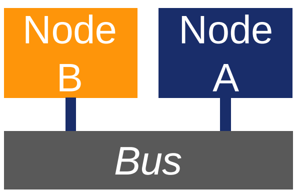


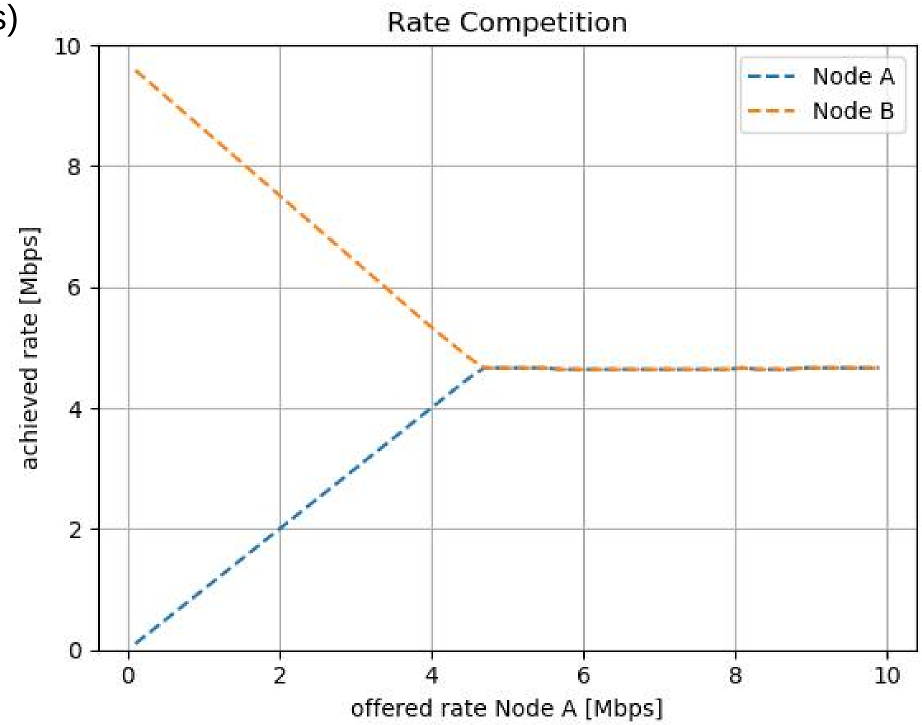
Competing Load – Bits/Sec Fairness (Credit-Based Fairness)

Experiment:

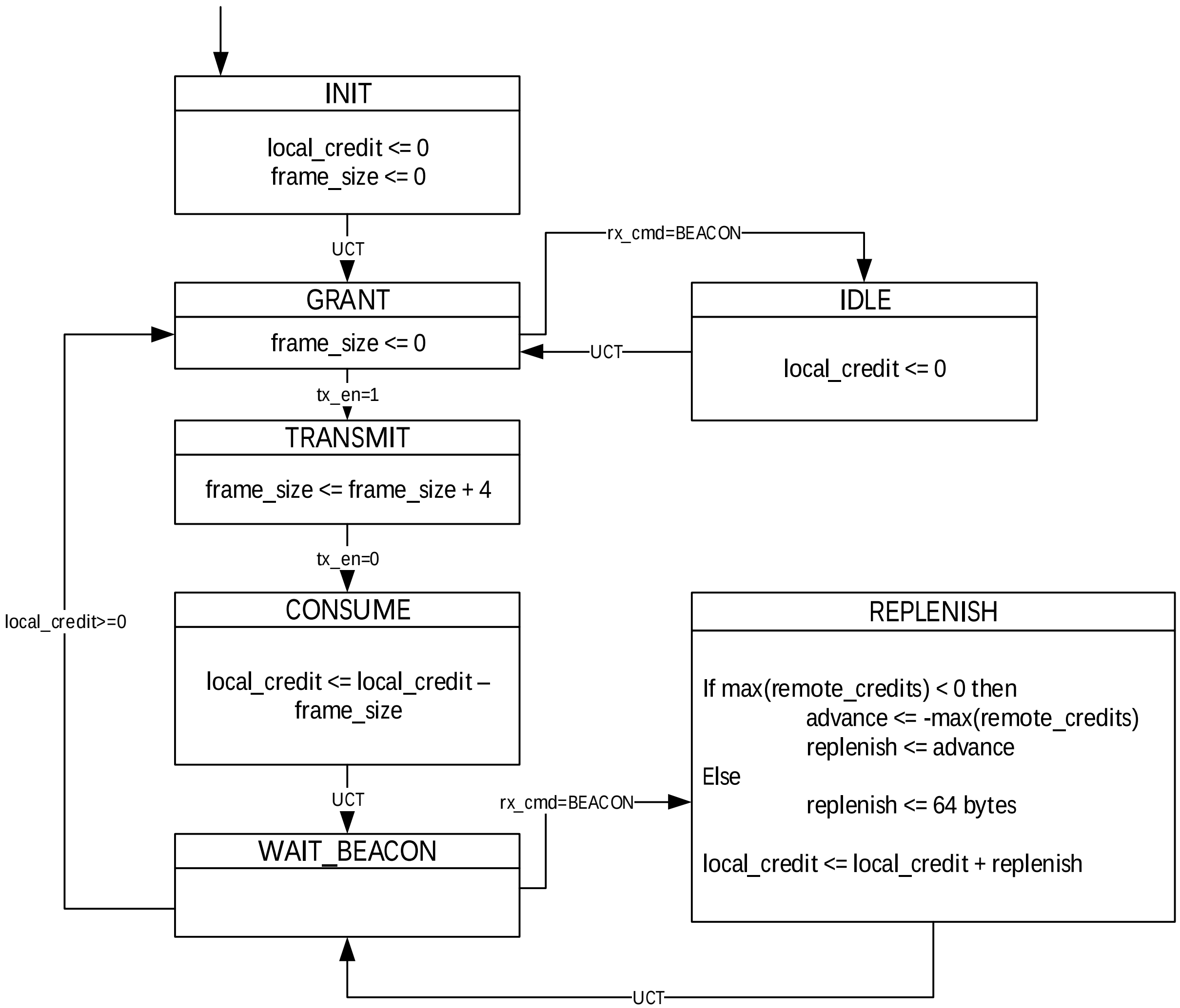
Offered load Node B: fixed 10Mbps (512 byte frames)

Offered load Node A: variable rate (128 byte frames)





Credit-Based, Round-Robin Fairness – State Machine



frame_size = *number of bits of most recently transmitted frame (i.e. derived by the number of cycles tx_en was asserted)*

local_credit = *current credit of local node*

remote_credit = *credit of remote nodes*

Features of this Approach

- No higher layer information is used – only the bits seen on the wire
- The proposed feature is optional and can be specified as disabled by default
- No changes to the PCS and PMA
- The fairness % of a node is configurable at start-up – default is balanced fairness

Conclusion

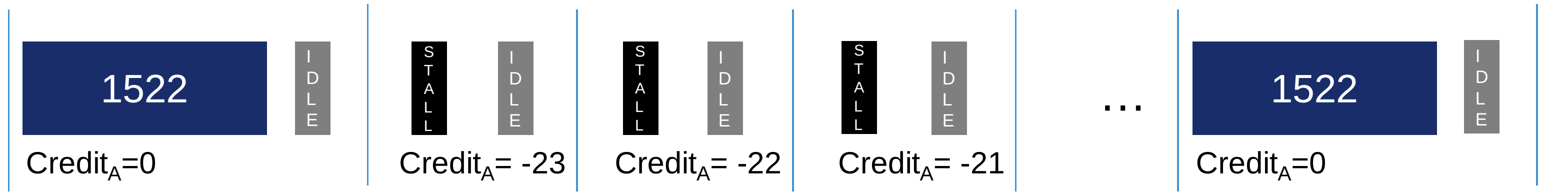
- Data-rate fairness is achievable with PLCA with the addition of:
  - Credit counters
  - Credit state machine
- Data-rate fairness can be extended to generate selectable data-rate fairness (i.e. one node can be configured to get a higher data-rate than the others)
  - This is QoS on a node-by-node basis
- What is needed is needed QoS on a frame-by-frame basis (as discussed in 802.3cg)
- In order to get QoS on a frame-by-frame basis, what needs to be understood first is QoS on a node-by-node basis
  - More work needs to be done to get QoS on a frame-by-frame basis
- Recommendation: Data-rate fairness should be added as an option to PLCA

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

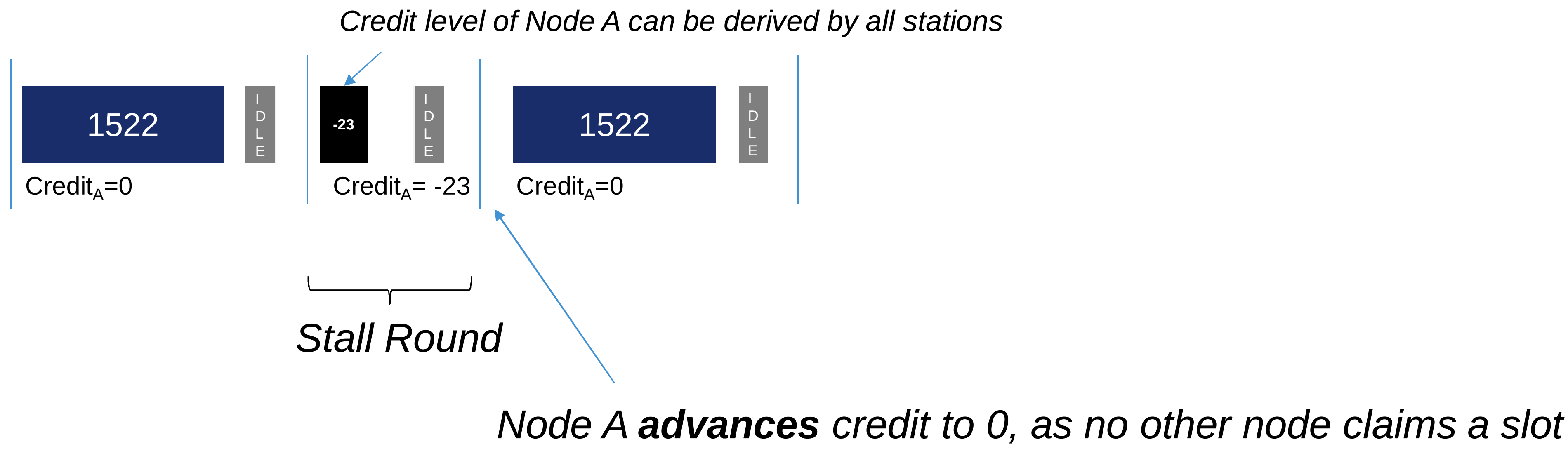
Credit-Based, Round-Robin Fairness - Corner Cases

- One node transmits, others idle (no advancement)
- Example:



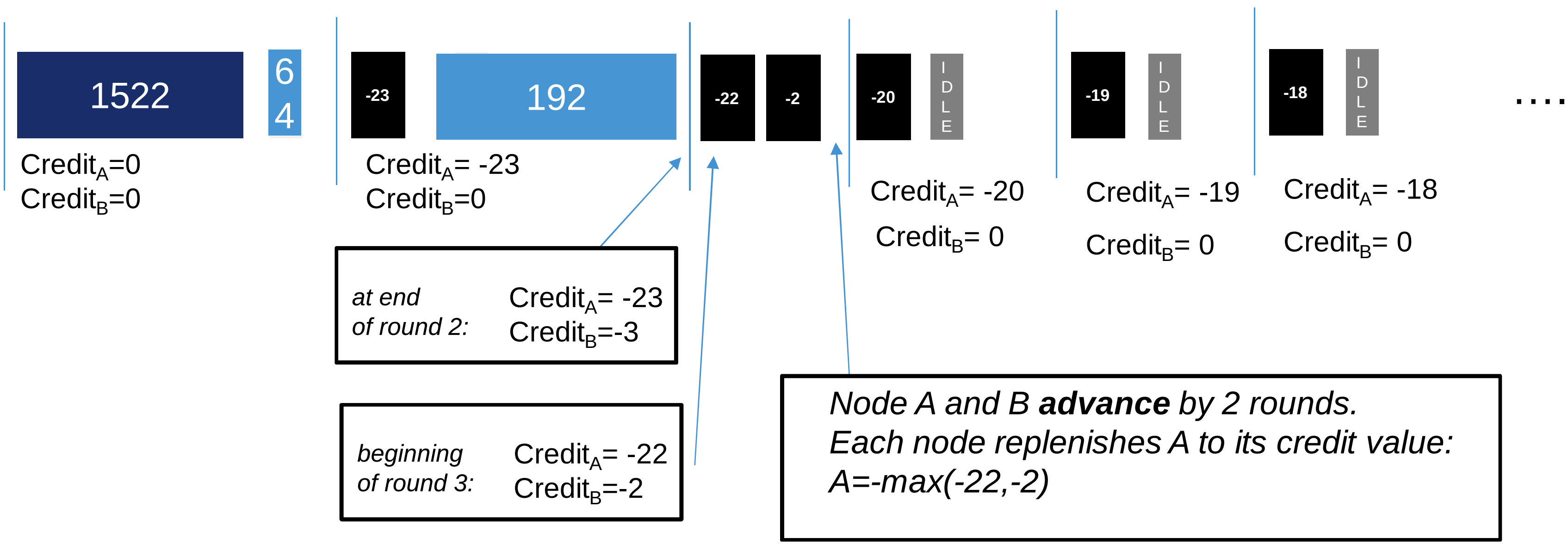
Credit-Based, Round-Robin Fairness - Corner Cases

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



Credit-Based, Round-Robin Fairness - Corner Cases

- Advanced example:



Note: Here, 1 credit = 64 bytes

SECURE CONNECTIONS
FOR A SMARTER WORLD