PFC-Free
Low Delay Control Protocol (LDCP)

draft-dai-tsvwg-PFC-free-low-delay-control-protocol-00

Huichen Dai, Binzhang Fu, Kun Tan

daihuichen@huawei.com

IETF 108, TSVWG, July 28, 2020

LDCP in Huawei Cloud

- LDCP has been online with RoCEv2 in Huawei Public Cloud, safely running for one year
- Supports Huawei EVS (block storage service) with less than 100us application-level RTT

Outline

- Motivation
- LDCP (Low Delay Control Protocol)
- Use Cases
- Conclusion

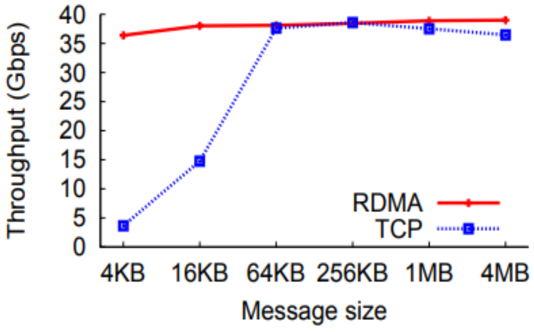
Outline

- Motivation
- LDCP (Low Delay Control Protocol)
- Use Cases
- Conclusion

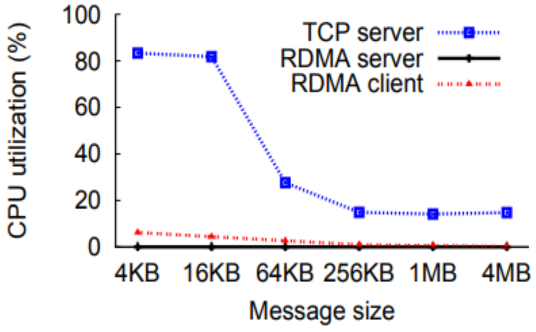
Data center networks (DCN)

- Cloud scale services: IaaS, PaaS, Search, BigData, Storage, Machine Learning, Deep Learning
- Services are latency sensitive or bandwidth hungry or both

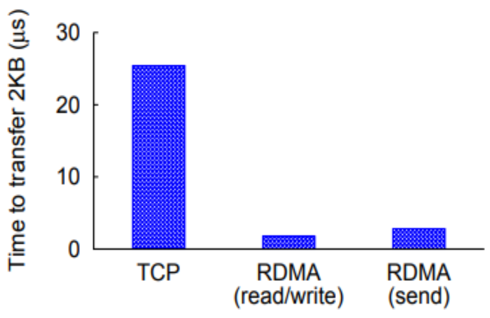
- Solution: RDMA (Remote Direct Memory Access)
  - RDMA bypasses host OS stack → frees host CPU, lowers latency



(a) Mean Throughput



(b) Mean CPU Utilization



(c) Mean Latency

RDMA outperforms TCP in throughput, CPU utilization and latency

RDMA in Modern Datacenters

- In past, RDMA was deployed on special fabrics, i.e., InfiniBand
  - InfiniBand is incompatible with Ethernet + IP, also expensive
  - How to deploy RDMA in data-centers?

- Solution: RoCEv2 (RDMA over Converged Ethernet)
  - A technique that runs RDMA over Ethernet

RoCEv2 Problems

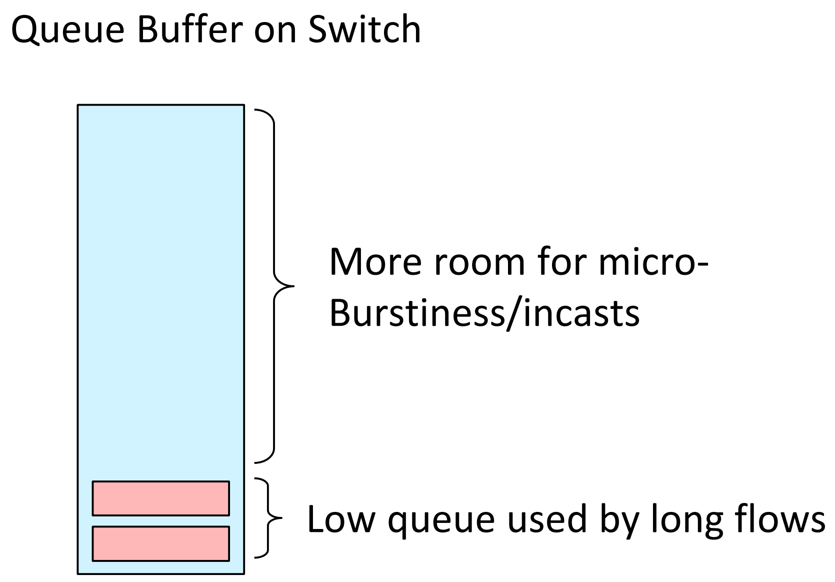
- RoCEv2 performance is sensitive to packet drops
  - Go-back-N: retransmit the lost packet and all subsequent ones
- RoCEv2 uses an *Ethernet extension* “FPC” to achieve losslessness
- PFC signals upstream switch to stop sending when queues build up
- **However, PFC brings adverse effects: performance degradation (HoL blocking) and unreliability (e.g., deadlock)**
  - **significantly harms latency and throughput performance**
  - **limit RoCEv2 deployment to only one pod in data-centers (limit the scale of adverse effects)**

Outline

- Motivation
- LDCP (Low Delay Control Protocol)
- Evaluation Results
- Conclusion

LDCP (Low Delay Control Protocol)

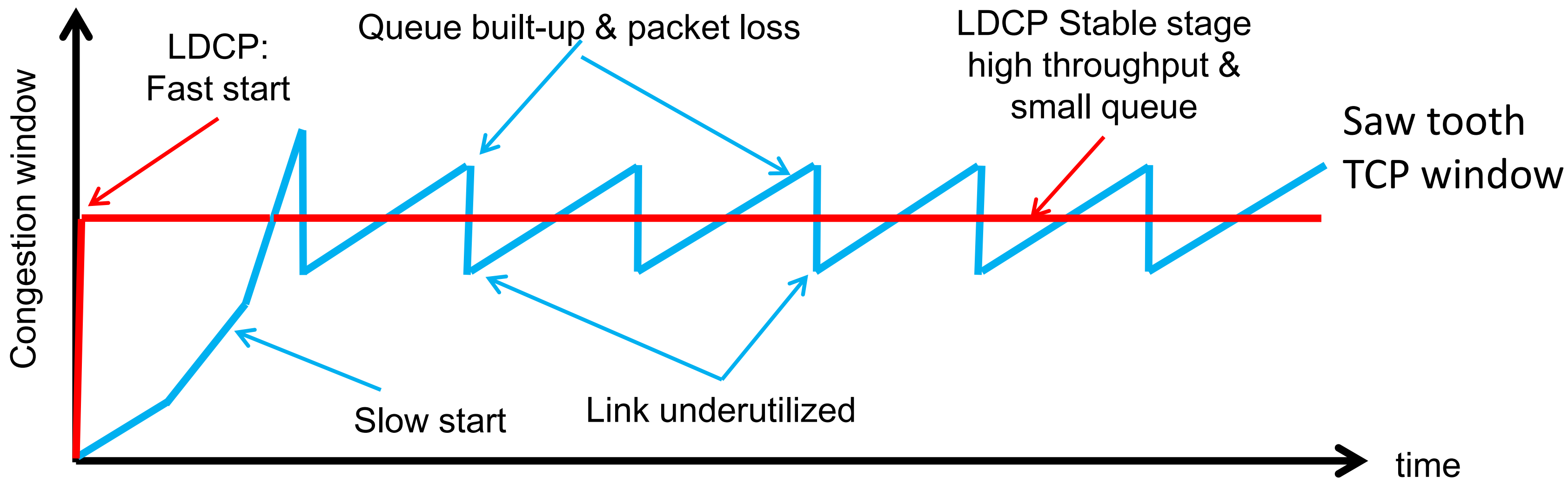
- An end-to-end congestion control that maintains constant low queues
  - Queue usage is much smaller than available buffer size, leading to ***almost no packet loss***, thus
  - PFC-free
  - No throughput degradation



- Small queue size reduces queueing delay and packet loss, so LDCP is not specific to RoCEv2, but is open to all transports with a reliable service

LDCP (Low Delay Control Protocol)

- LDCP consists of two algorithms
  - Fast start algorithm quickly acquires bandwidth in first RTT
  - Stable stage algorithm maintains constant low queue and high throughput

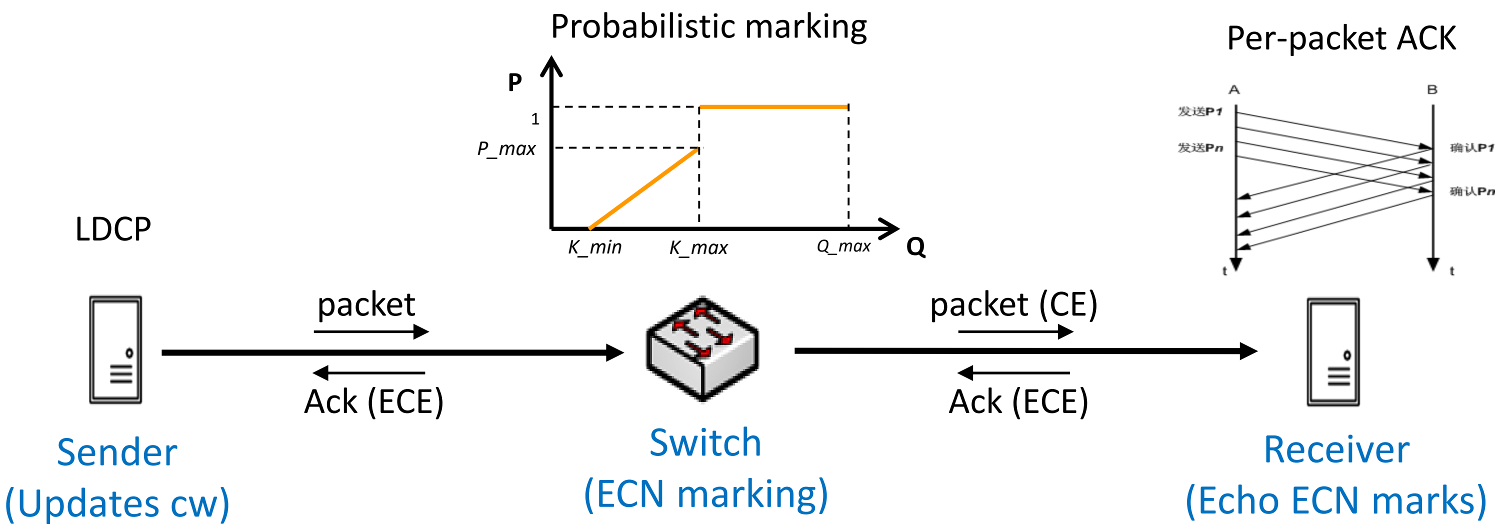


LDCP – Stable stage algorithm

- Window based congestion control
- Switch: standard ECN
- Per-packet ACK: receiver generates an ACK for each received data packet (not mandatory)
- Sender adjusts window on ACK arrival

$$cwnd = \begin{cases} cwnd + \dfrac{\alpha}{cwnd}, & if\ ECE = 0 \\ cwnd - \beta, & if\ ECE = 1 \end{cases}$$

α, β are parameters (0<α, β<=1)



LDCP – Stable stage algorithm

- Fluid Model

$$\frac{\mathrm{d}W}{\mathrm{d}t} = (1 - p(t - R))\frac{\alpha}{R(t)} - p(t - R)\frac{\beta W(t)}{R(t)},$$

$$\frac{\mathrm{d}q}{\mathrm{d}t} = N\frac{W(t)}{R(t)} - C.$$

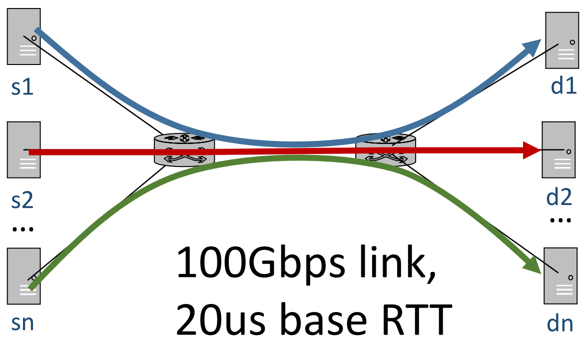
$$R(t) = d + \frac{q(t)}{C}.$$

Table 1: Model Parameters

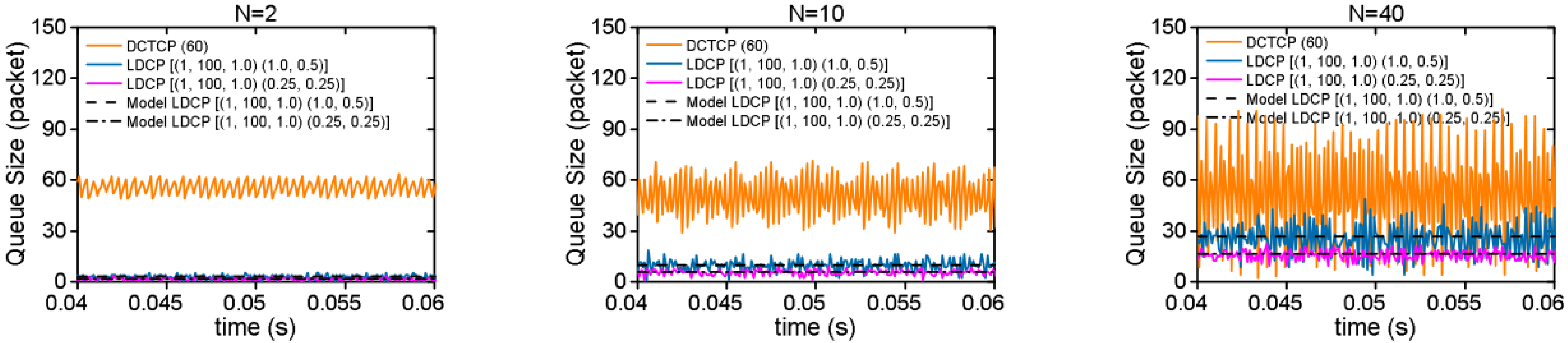
$W(t)$	window size (cw) of a flow at time t
$q(t)$	instant queue size at time t
$p(t)$	ECN marking probability at time t
d	round trip propagation delay
$R(t)$	RTT at time t
N	number of flows on the bottleneck link
C	link capacity
K_{min}	ECN marking threshold lower bound
K_{max}	ECN marking threshold upper bound
P_{max}	largest ECN marking probability

- ***The model reveals that LDCP is able to maintain a stable queue size.***
  - ***Model predictions accurately match the real queue size***

LDCP – Stable stage algorithm



- Simulation and results: stable and small queue size

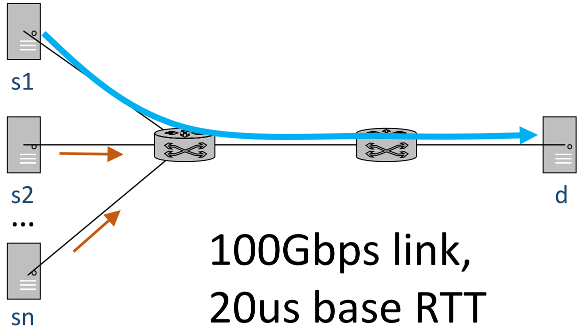


LDCP – fast start algorithm

- Choosing an appropriate initial window (IW) size is challenging when a new flow starts up
  - a too large IW may cause congestion inside network → large queue buildup or even packet drops
  - a too conservative IW may miss the transmission opportunities in the first RTT → longer completion time for small messages

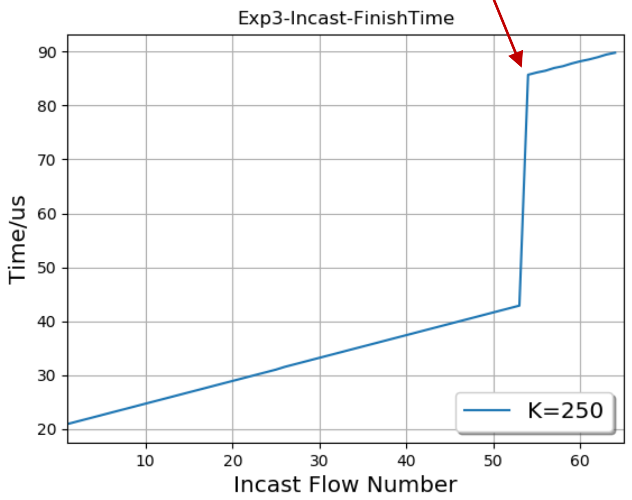
- LDCP fast start algorithm: makes the most of the free bandwidth, but without causing congestion

LDCP – fast start algorithm

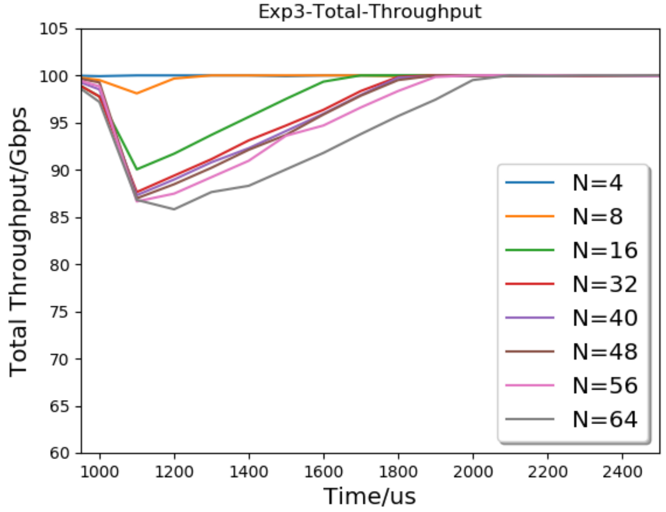


- Incast + long flow share bottleneck link

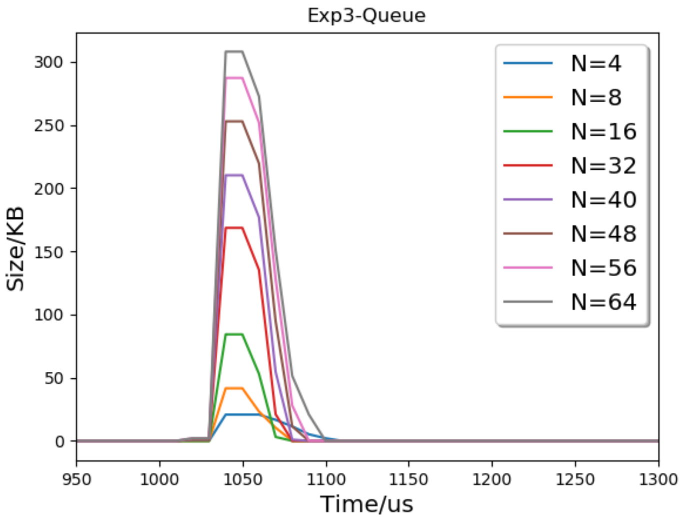
WRED drop begins to happen



Incast Completion Time



Total Throughput



Queue size

***Queue limit: 500KB, no timeout until 245 incast senders, 5KB each sender**

Outline

- Motivation
- LDCP (Low Delay Control Protocol)
- Use Cases
- Conclusion

LDCP with RoCEv2

- Revisions to RoCEv2 standard
  - Add ACK packets for RDMA read responses
  - Add customized headers for sequence numbers, ECN signals, ect.

- RoCEv2 with LDCP outperforms RoCEv2 with DCQCN
  - 32-node testbed, 2-layer Clos topology
  - 1:1 bandwidth subscription: small-msg average FCT reduced by: 28.0%, 48.2%, 34.9%, 59.9% (4 kinds of workloads)
  - 4:1 bandwidth over-subscription: small-msg average FCT reduced by: 35.2%, 30.3%, 29.8%, 50.8% (4 kinds of workloads)

On-going cases

- LDCP with standard RoCEv2

- LDCP with TCP offload Engine (ToE)

Outline

- Motivation
- LDCP (Low Delay Control Protocol)
- Testbed Results
- Conclusion

Conclusion

- LDCP: an end-to-end congestion control protocol, consists of
  - Fast start algorithm
  - Stable stage algorithm
  - Maintains stable and small queue size
- Achieves very small packet loss rate, allows loss-sensitive transports to operate without link-level flow control
  - Also with performance improvement: low latency, high throughput
- Safely running for one year in production environments with RoCEv2

Thanks :-)

Comments are welcome~

LDCP – Stable stage algorithm

- Comparison with existing ECN congestion control

ECN Marks	TCP	DCTCP	LDCP
1 0 1 1 1 1 0 1 1 1	Cut window by 50%	Cut window by 40%	– + – – – – + – – –
0 0 0 0 0 0 0 0 0 1	Cut window by 50%	Cut window by 5%	+ + + + + + + + + –

LDCP – Stable stage algorithm

- Window update rule (Per-packet ACK)

$$cwnd = \begin{cases} cwnd + \frac{\alpha}{cwnd}, & if\ ECE = 0 \\ cwnd - \beta, & if\ ECE = 1 \end{cases}$$

α, β are parameters (0<α, β<=1)

- Window update rule (one ACK confirms *n* packets)

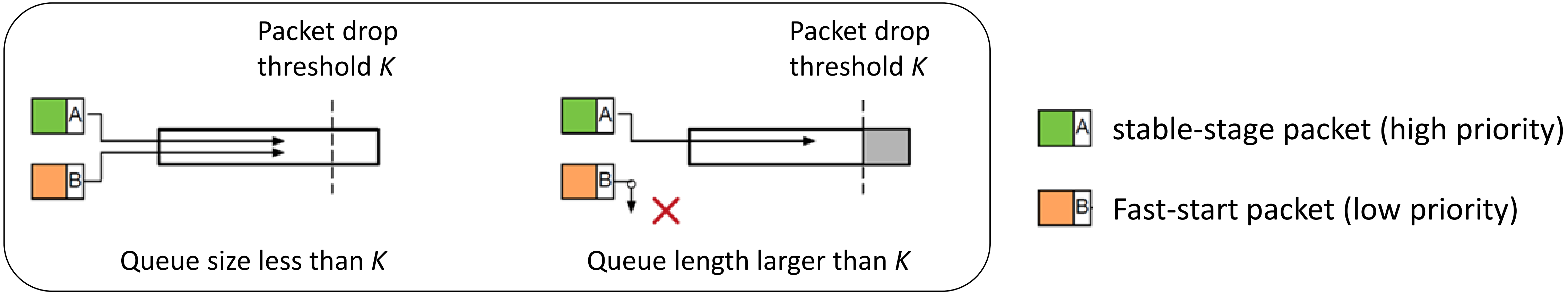
$$cwnd = \begin{cases} cwnd + \frac{n\alpha}{cwnd}, & if\ ECE = 0 \\ cwnd - n\beta, & if\ ECE = 1 \end{cases}$$

α, β are parameters (0<α, β<=1)

- AI-MD algorithm distributed to each ACK
  - $+\frac{\alpha}{cwnd}$: AI factor, increases cwnd by α in one RTT
  - $-\beta$: MD factor, decreases cwnd by $\beta * cwnd$ each RTT

LDCP – fast start algorithm

- Select a large value for IW, e.g., BDP, to probe the network
- Fast-start packets have low priority
  - pass the network if there is enough free bandwidth, but get dropped intentionally by switches if there is congestion
  - Set up a queue size threshold ***K***, low priority packets are dropped if queue size exceeds ***K***, high priority packets are forwarded normally



LDCP – fast start algorithm

- Mark packets and drop packets in the same queue?
- ECN/WRED supports this feature
  - ECN-capable packets are subject to ECN marking
  - ECN-incapable packets comply with WERD dropping
- Fast-start packets are set to ECN-incapable, stable-stage packets are set to ECN-capable

- But if all packets of a message are dropped by WRED, timeout happens
  - Make the last fast-start packet ECN-capable