

Taming Acknowledgements

... aka, ACK thinning, Ack scaling, Ack reduction

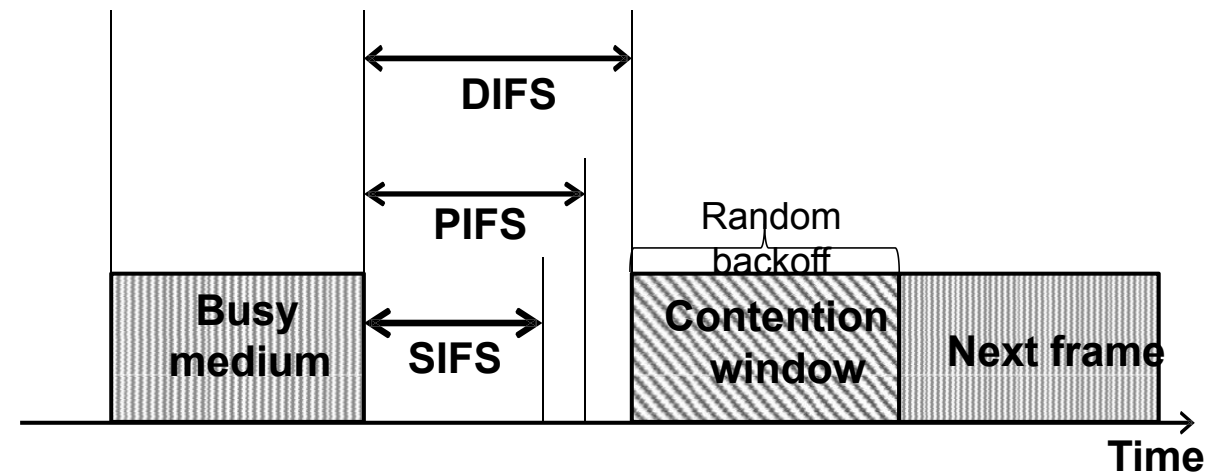
Tong Li, Kai Zheng, **Rahul Jadhav**, Jiao Kang

Huawei

IETF 107

Why?

- In wireless scenarios, cost of frame transmission, regardless of its size, is significant
 - Inter-frame spacing coordinates access to common medium
 - Faster PHY rates accentuates the problem
- Savings in ACK traffic translates to improvements in goodput rates
- Some mitigation strategies used in WLAN
 - Frame aggregation
 - Impact?



Our use-Case: Wi-Fi Direct transfers

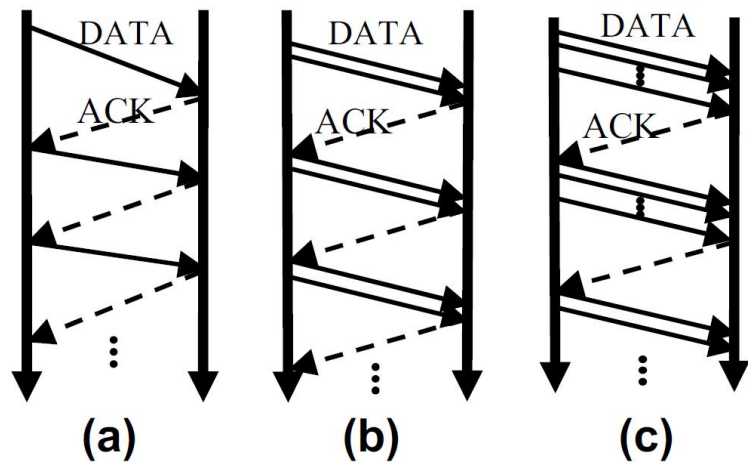


- Single hop
 - Local file transfer
 - Wireless Projection
 - Wi-Fi Direct scenarios
- High throughput scenario
 - High throughput more ACKs
- Similar delay in both direction
- Problem increases with 4K wireless streaming
- *TCP, not very well suited*

ACK intensity

Byte-counting ACK

$$f = \frac{bw}{L \cdot MSS}$$

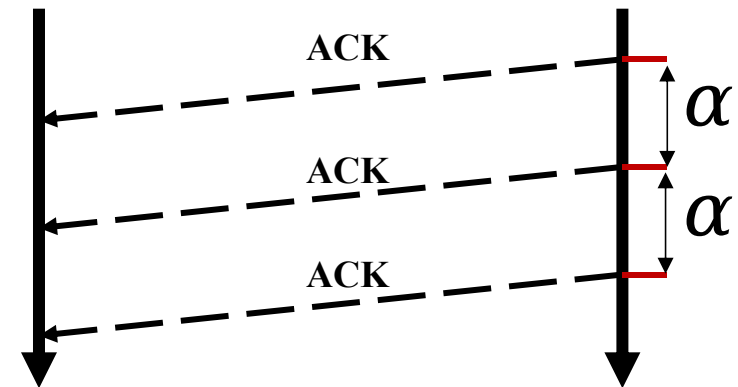


- L : sending an ACK for every L ($L=1, 2, 3, 4\dots$) incoming full-sized packets

$$bw \rightarrow \infty \quad f \rightarrow \infty$$

Periodic ACK

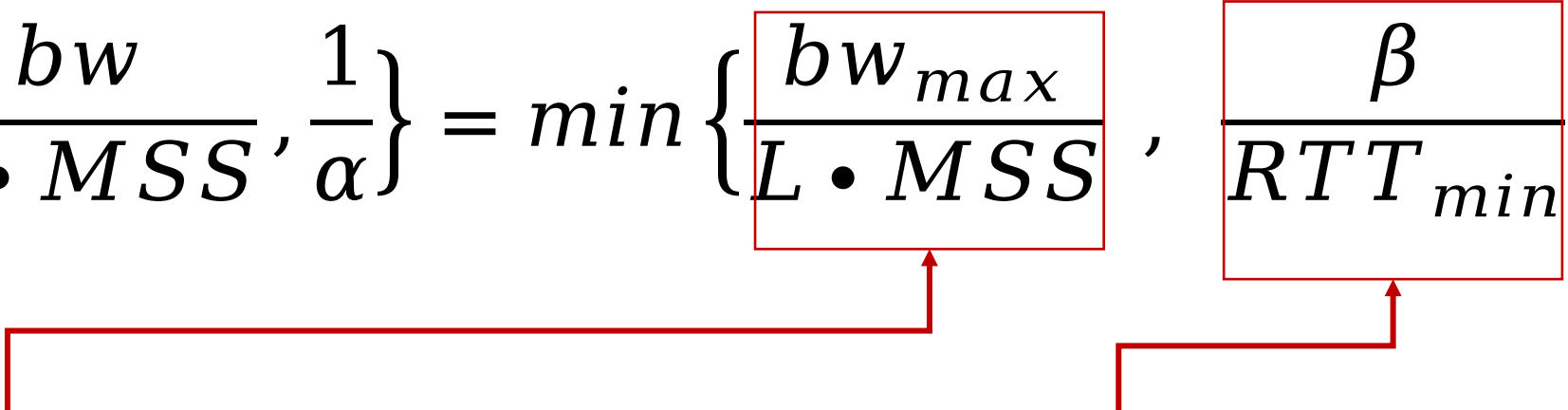
$$f = \frac{1}{\alpha}$$



- α : sending an ACK for each time interval (α)
... where α is a function of RTT.

$$bw \rightarrow \infty \quad f \rightarrow \text{Constant}$$

Tamed ACKs

$$f_{tack} = \min \left\{ \frac{bw}{L \cdot MSS}, \frac{1}{\alpha} \right\} = \min \left\{ \frac{bw_{max}}{L \cdot MSS}, \frac{\beta}{RTT_{min}} \right\}$$


if bdp < L · β · MSS
(Byte-counting ACK)

if bdp ≥ L · β · MSS
(Periodical ACK)

If BDP is small, byte-counting ACK is used

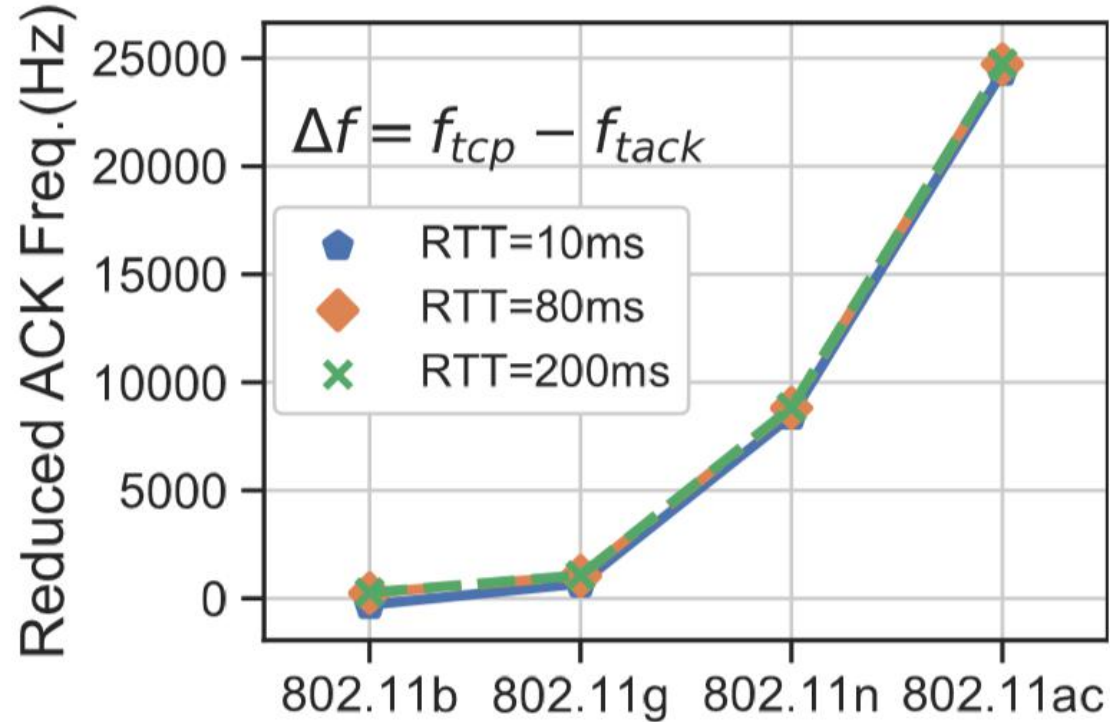
If BDP is large, periodical ACK is used

With newer wireless standards, the BDP is significantly increasing

Triggering ACK on out-of-order data reception takes care of responding to loss events.

Data: ACK reduction with TACK

Considering $L=2$, $\beta=4$

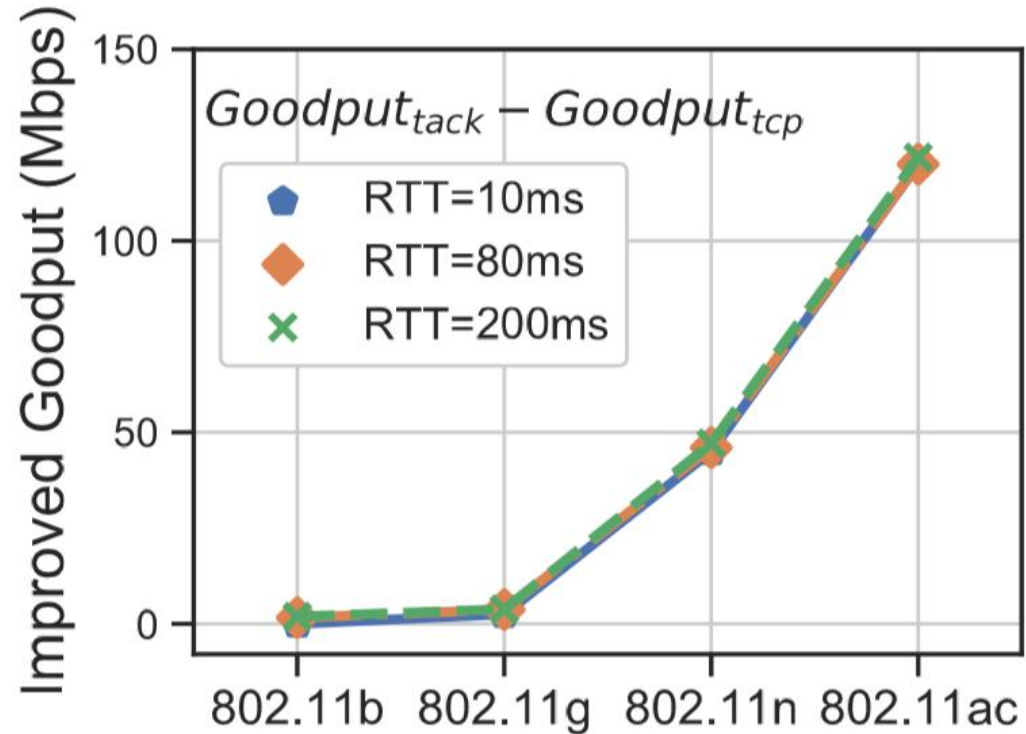


(a) ACK frequency reduction

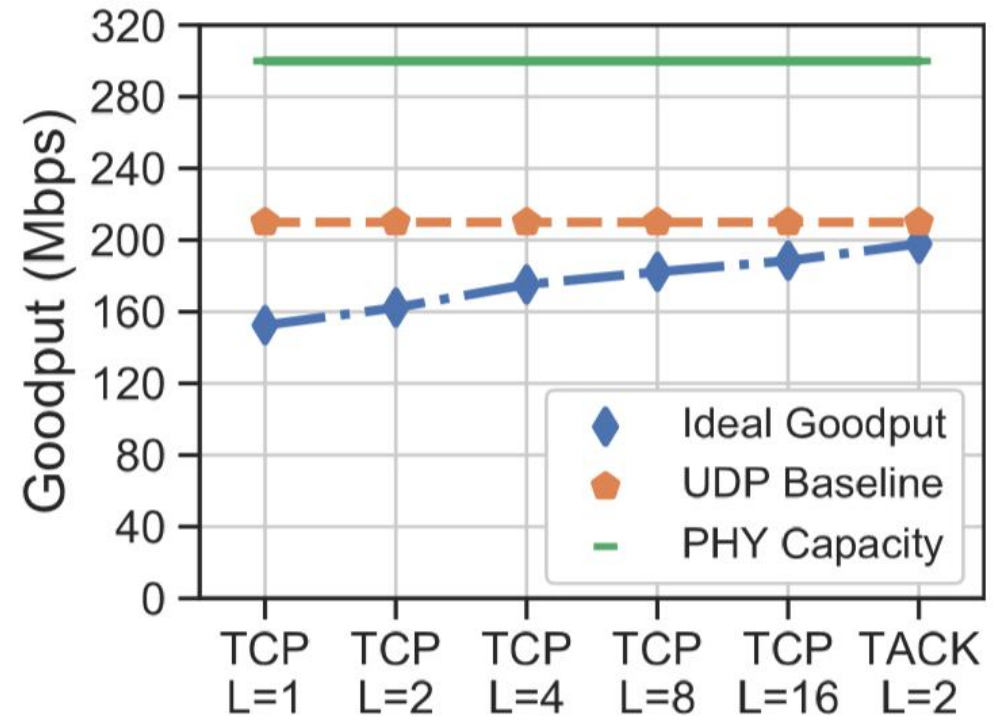
	802.11b		802.11ac	
RTT_{min}	TCP (L=2)	TACK (L=2)	TCP (L=2)	TACK (L=2)
10 ms	294	294	24777	400
80 ms	294	50	24777	50
200 ms	294	20	24777	20

(b) ACK frequency (Hz)

Data: Goodput improvement



(a) Goodput improvement



(b) Ideal goodput trend

TCP's Acknowledgement Scheme

- Scheme is tailored
 - to reduce traffic bursts
 - thus low buffering requirements
 - L cannot be greater than 2
 - provides fine-grained RTT estimation
 - Loss Recovery following TCP's design constraints
 - Tolerate renegeing
- TCP enforces these conditions for all connections
 - Even if certain connections may be tolerant to some of the above conditions
- TCP does not consider app's latency tolerance
 - Tries minimizing latency given the network conditions and not app's requirements

Lower bound for ACKs?

- Max ACK traffic reduction before impacting performance?
- Ideal Conditions: No loss, Less RTT variation
 - Landstrom et.al. theoretically analyze that 2 ACKs per send window is good enough
 - Practically, 4 ACKs per send window suffices to maintain performance
 - Having said that it has implications on traffic burstiness etc.

Side-effects of ACK thinning

- Widely documented [RFC2525]
 - Impact on Loss Recovery
 - Mitigation: Immediately ACK out-of-order segment
 - QUIC already allows it
 - Traffic burstiness impacting buffering
 - Certain scenarios do not have problems with this (our case for e.g.)
 - RTT estimation
 - One way delay i.e., from sender to receiver serves as good input
 - Our case: Near-symmetry of paths (in terms of propagation delay)
 - QUIC allows to measure receiver induced delay

Existing work in IETF

- draft-gomez-tcpm-ack-pull-01
 - Proposing quite the opposite of what we trying
- draft-iyengar-quic-delayed-ack-00, submitted in Jan 2020
 - Proposes QUIC extensions for sender-controlled ACKing
- draft-fairhurst-quic-ack-scaling-00, submitted in Jan 2020
 - ACK scaling for asymmetric links, however, can be generalized

What do we bring to the WG?

- Our use-cases, observations, data-set
- Techniques, how we ended up reducing ACK frequency?
 - Theoretical analysis
 - Experiments with Linux kernel TCP
 - Experiments with our proprietary protocol FILLP
- Hoping to generalize some of the techniques
 - across WAN scenarios
 - and across asymmetric links
- Regardless, different use-cases will demand different behaviour
 - The transport scheme should take this into account
 - i.e. be flexible on **per connection basis**