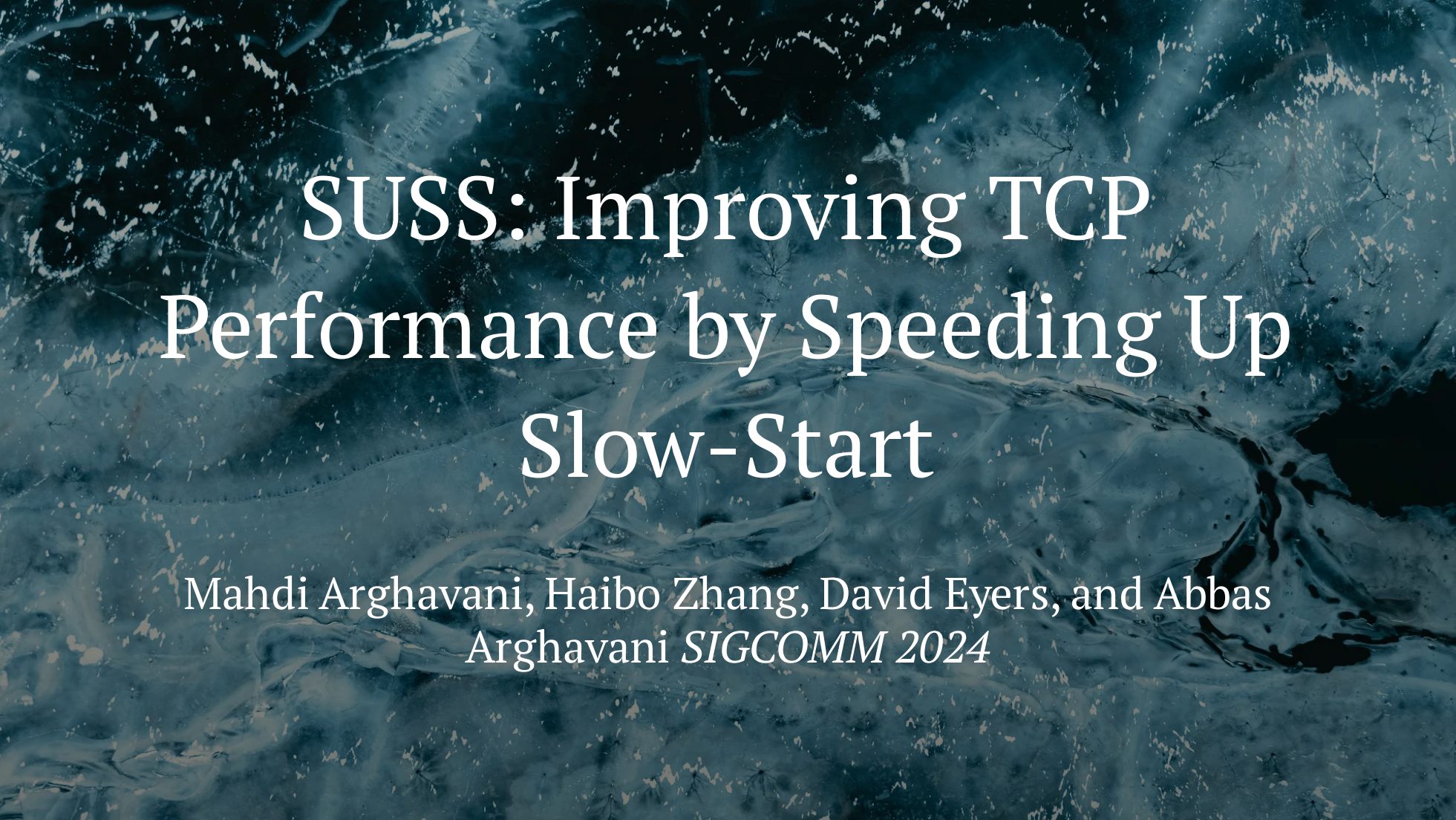
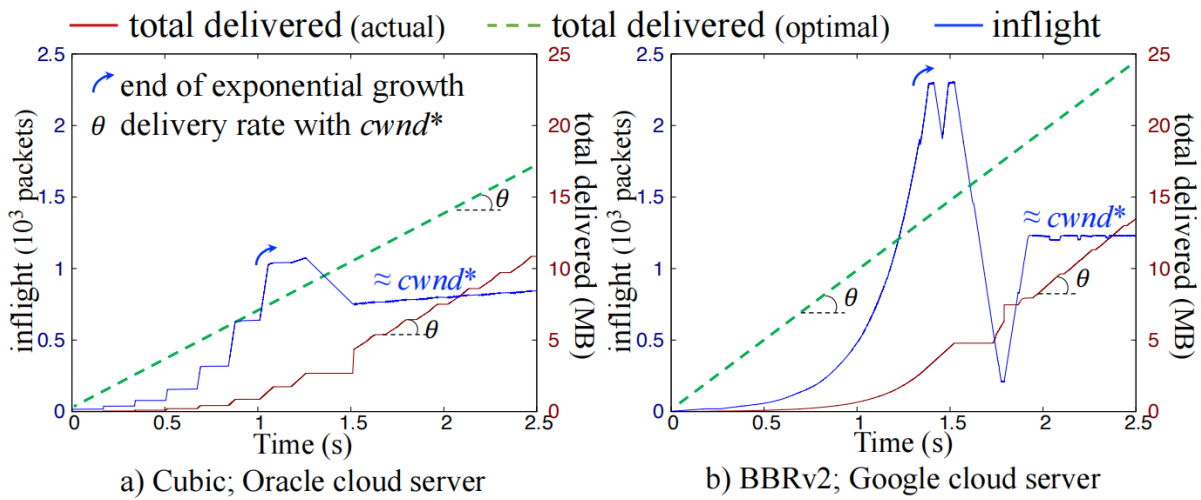


SUSS: Improving TCP Performance by Speeding Up Slow-Start

Mahdi Arghavani, Haibo Zhang, David Eyers, and Abbas Arghavani *SIGCOMM 2024*

TCP slow-start drawbacks

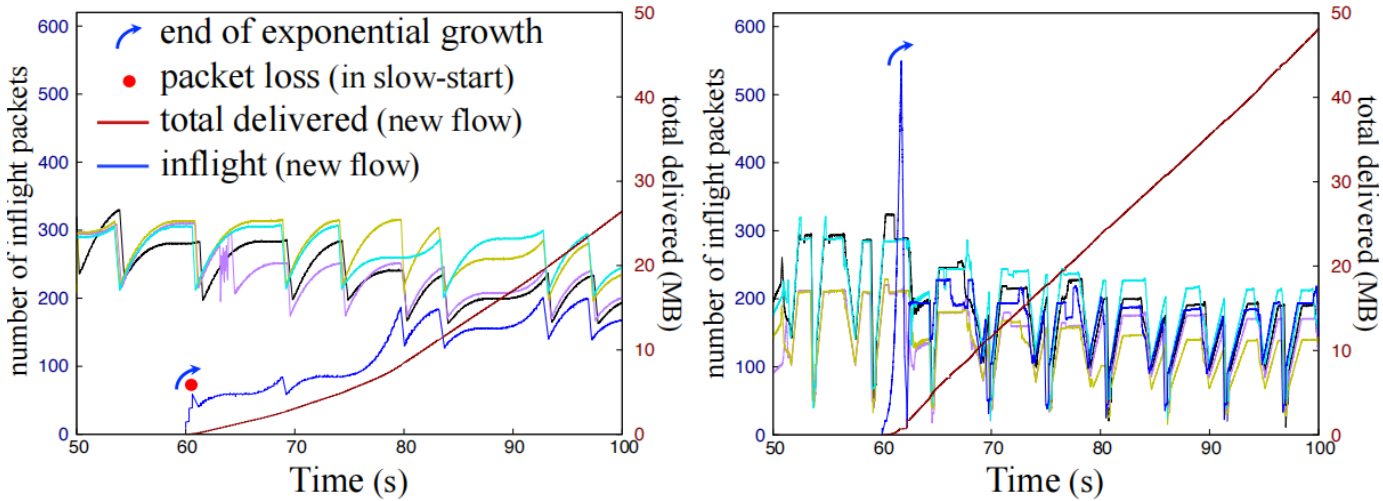
- slow ramping-up of the data delivery rate
  - inefficient bandwidth utilization
- prolonged completion time for small-size flows (flows less than a few megabytes in size)
  - significantly diminish the Quality of Experience (QoE)



- **Two plots highlight the capability for transferring more data during the initial phase**

TCP slow-start drawbacks

- Inefficiency of CUBIC's slow-start in a congested network path



a) Cubic (loss-sensitive slow-start)

b) BBRv2 (loss-tolerant slow-start)

- CUBIC flow cannot gain a fair share of network resources rapidly, resulting in prolonged unfairness
  - reason: sensitivity in packet loss → exit slow-start before reaching $cwnd^*$

Related Works

- network-assisted approaches
  - require feedback from routers and switches → challenging for deployment and testing
- End-to-end solutions
  - designed to estimate $cwnd^*$, based on bandwidth measurement and optimal rate estimation
  - accurate bandwidth measurement and rate estimation are very challenging, especially in early rounds
- intuitive thoughts: use a larger initial window size (*iw*) → now 15KB (10 segments, RFC6928)
  - unaware of the flow length, lack prior knowledge of the network path conditions at the beginning
    - cannot select *iw* appropriately
  - the consequences of releasing a large burst of data packets in early rounds

SUSS

A lightweight, sender-side add-on to CUBIC's slow-start mechanism

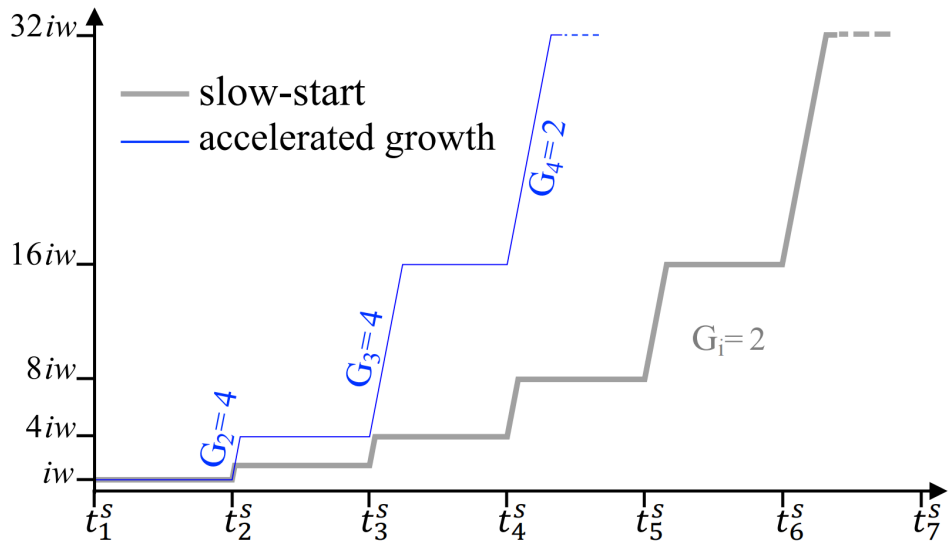
- Aims
  - safely accelerate cwnd growth in large-BDP networks
  - enhance the delivery rate during the early RTTs of slow-start
- Key idea: predict the continuation of exponential growth of cwnd in the subsequent RTT
  - if *cwnd* is expected to grow in the next RTT, then quadruple *cwnd*
  - allow *cwnd* to quickly ramp up in the early RTTs of slow-start
- primary challenges
  - How to predict the likelihood of exponential *cwnd* growth in the next RTT
    - based on HyStart: $cwnd \approx cwnd^*$
  - How to control additional data packet transmission to avoid bursts and packet loss
    - propose a novel combination of ACK clocking and packet pacing
      - ACK clocking: protect the functionality of HyStart and its estimation of the growth factor
      - packet pasing: avoid the bursts of traffic that might be caused by surges in *cwnd*

Key parameters

- $round(i)$: the time interval $[t_i^s, t_{i+1}^s)$, where t_i^s is the time when the ACK of the first data packet sent in $round(i-1)$ is received for $i > 1$. For $i = 1$, $round(1)$ initiates at the beginning of the data transfer.
- $cwnd_i$: maximum congestion window size in $round(i)$.
- iw: volume of data sent in $round(1)$, i.e., $cwnd_1 = iw$.
- *Data train*: sequence of data packets sent in a round.
- *ACK train*: sequence of ACKs for a data train.
- Δt_i^{at}: time taken to receive the ACK train in $round(i)$.
- R_i: the transmission rate of the data train in $round(i)$.
- $minRTT$: measured minimum round-trip time since the initiation of the TCP connection.
- $moRTT_i$: the minimum observed RTT in $round(i)$.
- $BtlBw$: bandwidth of the bottleneck link.
- G_i: growth factor for $cwnd$ in $round(i)$.
- $cwnd^*$: the optimal congestion window size when a fair share of $BtlBw$ is achieved.

Theory Behind SUSS

- After receive the ACK train:
  - evaluate whether the exponential growth of *cwnd* is extrapolated to persist for the next round
  - if true, then send another $2 * cwnd_(i-1)$ this round → $G_i = 4$
  - Why not consider more rounds? the network condition may vary over several upcoming rounds



Key Challenge 1

How to determine whether the exponential growth of cwnd is expected to continue in the next round?

→ implement the approach from HyStart

- HyStart
  - Upon receiving an ACK, the amount of time elapsed from the start of the current round does not exceed half of the minRTT
  - The minimum observed RTT in the current round ($moRTT_i$) must not be greater than $1.125 *$ minRTT.
- Condition 1 for SUSS
  - $\Delta t_{i+1}^{at} \leq minRTT/2$
  - need to estimate Δt_{i+1}^{at}
- Condition 2 for SUSS
  - $moRTT_{i+1} <= 1.125 * minRTT$
  - need to estimate $moRTT_{i+1}$

Formulating Condition 1 for SUSS

$\Delta t_{i+1}^{at} \leq minRTT/2$

- Δt_{i+1}^{at} is not known in round(i) → estimate it by its relationship with Δt_{i}^{at}
  - $\Delta t_{i}^{at} = \frac{cwnd_{i-1}}{BtlBw}$
- As the same as slow-start, $2 * cwnd_{i-1}$ is sent during Δt_{i}^{at}
  - $\Delta t_{i+1}^{at} = \frac{2*cwnd_{i-1}}{BtlBw} = 2 * \Delta t_{i}^{at}$
- **Condition 1**: $\Delta t_{i}^{at} \leq minRTT/4$

Formulating Condition 2 for SUSS

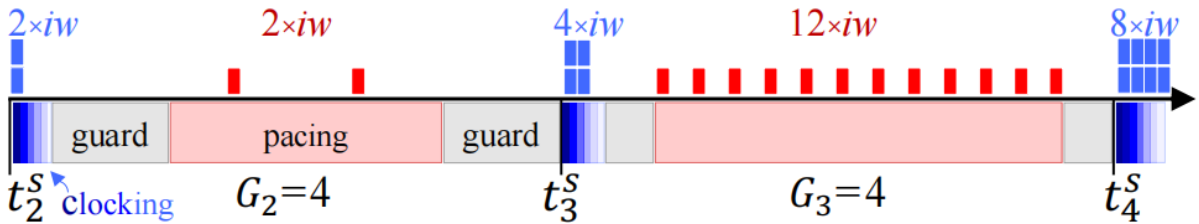
$moRTT_{i+1} \leq 1.125 * minRTT$

- $1.125 * minRTT$ → initial signs of queueing delay → beginning of congestion
- suppose that $minRTT$ was last updated r rounds ago
  - the average increase in queuing delay since then is approximately $\frac{moRTT_i - minRTT}{r}$
  - $moRTT_{i+1} \approx moRTT_i + \frac{moRTT_i - minRTT}{r}$
- **Condition 2**: $moRTT_i + \frac{moRTT_i - minRTT}{r} \leq 1.125 * minRTT$

Key Challenge 2

How to control additional data packet transmission to avoid bursts and packet loss?

- Combination of ACK clocking and packet pacing when $G_i > 2$
  - only ACK clocking → a burst of packets upon receiving each ACK train
  - only packet pacing → the measurement of Δt_i^{at} will no longer be accurate



- In *clocking period*, SUSS always sends twice the amount of data acknowledged by that ACK
  - facilitate the measurement of Δt_i^{at}
- In *pacing period*, the remaining data will be sent at a predetermined pace to mitigate burstiness
- *guard period* → avoid interference with the clocking period in both the current and the next round

3.1 Clocking Period

- $S_i^{Bdt} = iw * 2^{i-1}$: the amount of data sent in the clocking period for round(i)
- Δt_i^{Bat}: the amount of time taken to receive the ACKs for the blue packets send in round(i-1)
- $\Delta t_i^{at} = cwnd_{i-1} * \frac{\Delta t_i^{Bat}}{S_{i-1}^{Bdt}}$

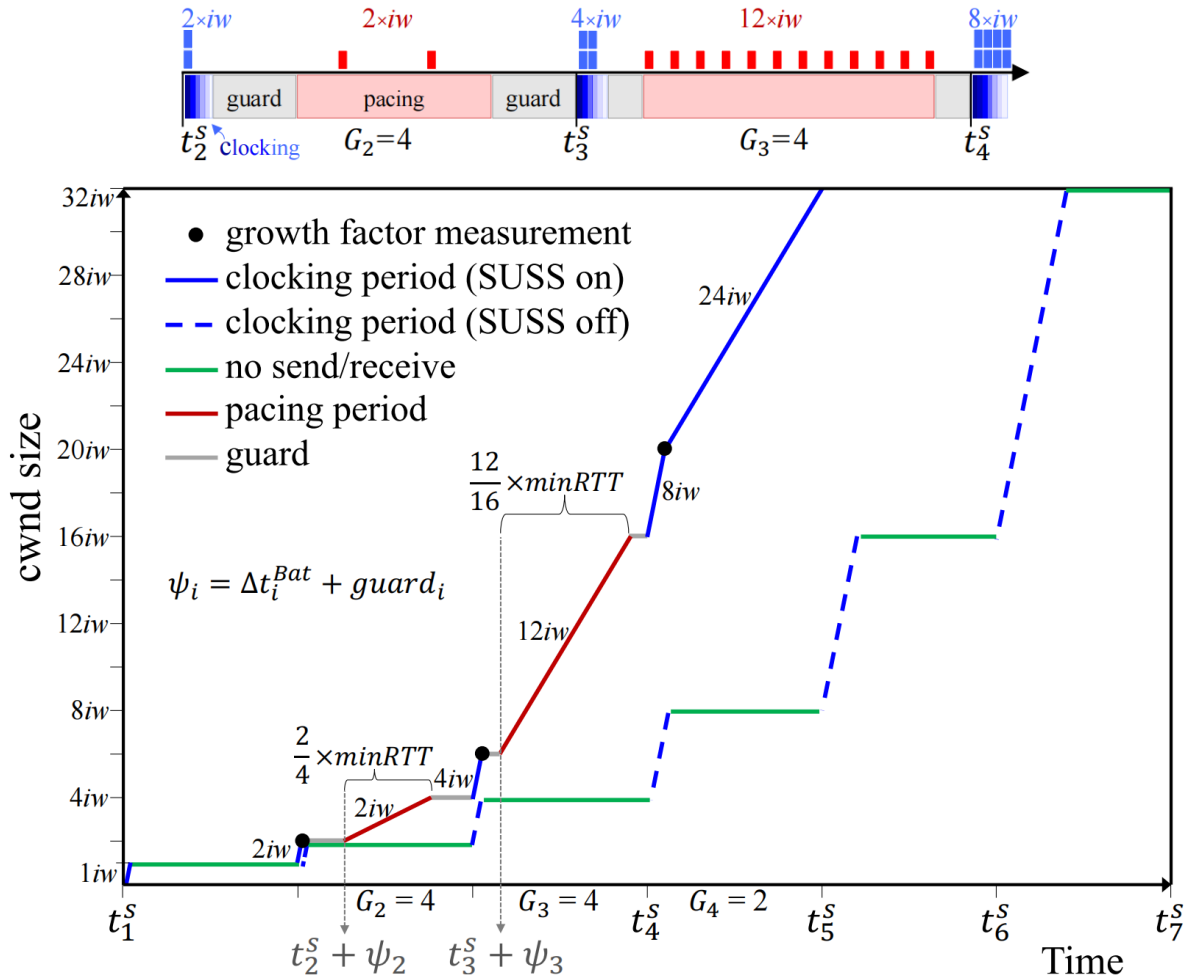
3.2 Pacing Period

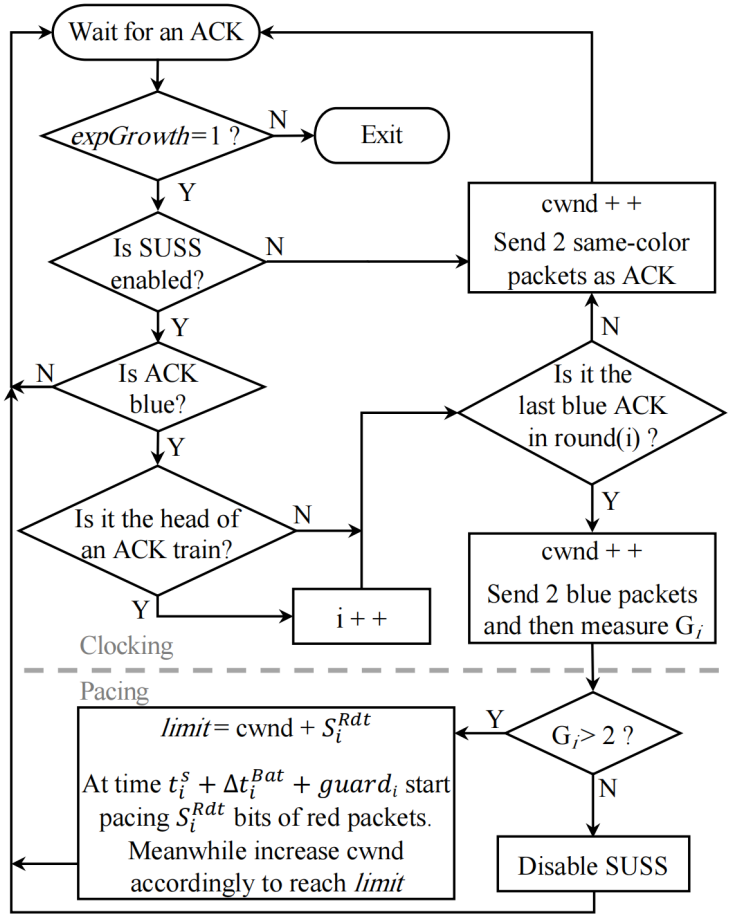
- S_i^{Rdt}: the amount of data to be sent during the pacing period of round(i)
  - $cwnd_i = S_i^{Bdt} + S_i^{Rdt}$
  - $S_i^{Rdt} = G_i * \left(S_{i-1}^{Bdt} + S_{i-1}^{Rdt}\right) - S_i^{Bdt} = G_i * S_{i-1}^{Rdt} + (G_i - 2) * 2^{i-2} * iw$
  - the growth of S_i^{Rdt} is higher than S_i^{Bdt} when $G_i > 2$
    - the duration and starting time of the pacing period in each round need to be dynamic
- time of pacing period: $\frac{S_i^{Rdt}}{cwnd_i} * minRTT \longrightarrow$ sending rate: $rate_i = \frac{S_i^{Rdt}}{\frac{S_i^{Rdt}}{cwnd_i} * minRTT} = \frac{cwnd_i}{minRTT}$

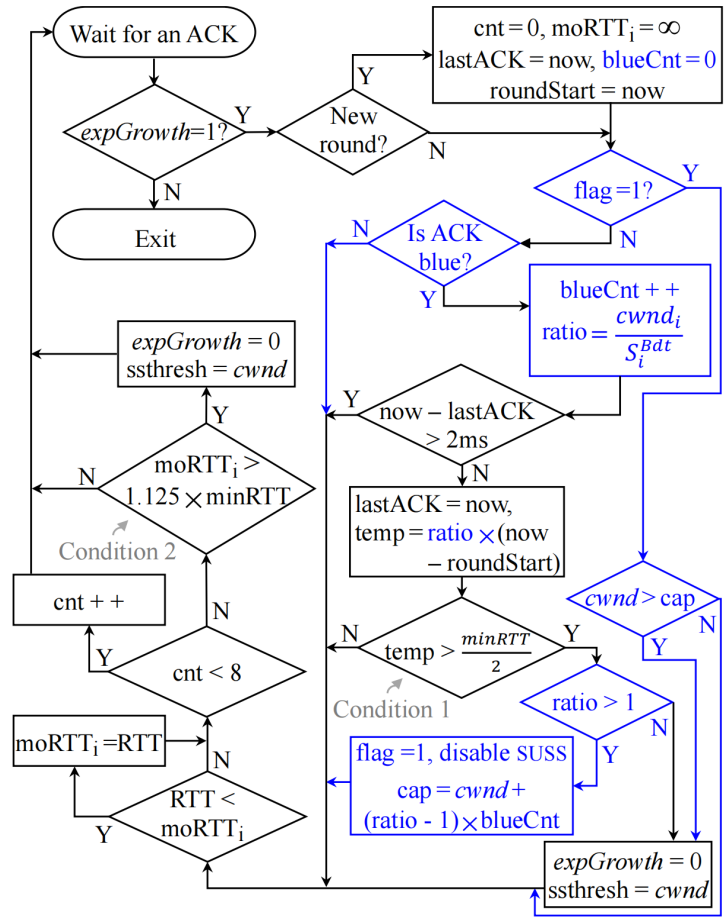
3.3 Guard Period

$$guard_i = \frac{minRTT - \left(\Delta t_i^{Bat} + \frac{S_i^{Rdt}}{cwnd_i} * minRTT\right)}{2}$$

$$= \frac{S_i^{Bdt}}{2 * cwnd_i} * minRTT - \frac{\Delta t_i^{Bat}}{2}$$


2×iw
2×iw
4×iw
12×iw
8×iw
guard
pacing
guard
clocking
t_2^s
G_2=4
t_3^s
G_3=4
t_4^s
32iw
28iw
24iw
20iw
16iw
12iw
8iw
4iw
1iw
cwnd size
growth factor measurement
clocking period (SUSS on)
clocking period (SUSS off)
no send/receive
pacing period
guard
$\psi_i = \Delta t_i^{Bat} + guard_i$
$\frac{12}{16} \times minRTT$
$\frac{2}{4} \times minRTT$
24iw
12iw
8iw
4iw
2iw
2iw
$G_2 = 4$
$G_3 = 4$
$G_4 = 2$
t_1^s
$t_2^s + \psi_2$
$t_3^s + \psi_3$
t_4^s
t_5^s
t_6^s
t_7^s
Time

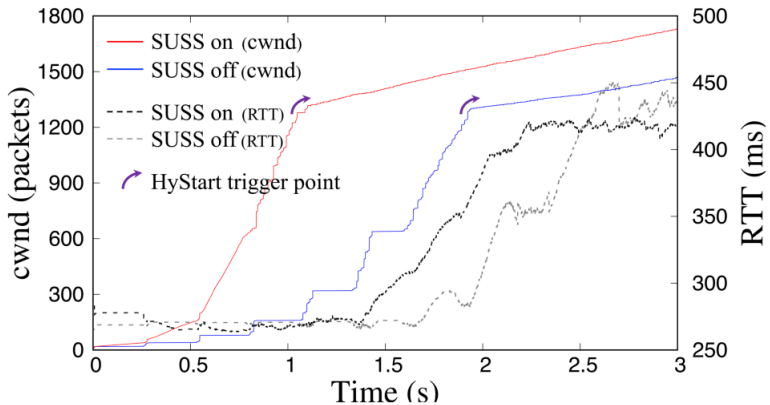

Wait for an ACK
expGrowth=1 ?
N
Exit
Y
Is SUSS enabled?
N
cwnd + +
Send 2 same-color packets as ACK
Y
N
Is ACK blue?
Is it the last blue ACK in round(i) ?
Y
Y
N
Is it the head of an ACK train?
N
cwnd + +
Send 2 blue packets and then measure G_i
Y
i + +
Clocking
Pacing
$limit$ = cwnd + S_i^{Rdt}
At time $t_i^s + \Delta t_i^{Bat} + guard_i$ start pacing S_i^{Rdt} bits of red packets. Meanwhile increase cwnd accordingly to reach $limit$
Y
G_i > 2 ?
N
Disable SUSS

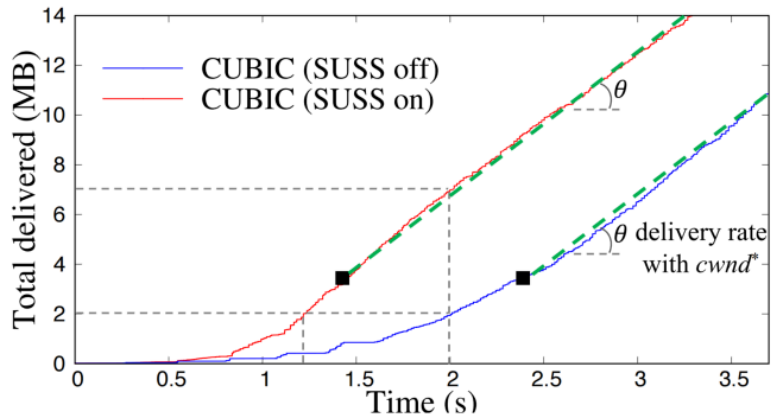

Wait for an ACK
cnt = 0, moRTT$_i$ = ∞
lastACK = now, blueCnt = 0
roundStart = now
Y
expGrowth=1?
Y
New round?
N
N
Exit
flag =1?
Y
N
N
Is ACK blue?
blueCnt + +
ratio = $\frac{cwnd_i}{S_i^{Bdt}}$
Y
Y
now – lastACK > 2ms
N
expGrowth = 0
ssthresh = cwnd
Y
moRTT$_i$ > 1.125 × minRTT
N
Condition 2
lastACK = now,
temp = ratio × (now – roundStart)
cnt + +
N
cwnd > cap
N
Y
Y
N
temp > $\frac{minRTT}{2}$
Y
cnt < 8
Condition 1
ratio > 1
N
Y
moRTT$_i$ =RTT
N
flag =1, disable SUSS
cap = cwnd + (ratio - 1) × blueCnt
RTT < moRTT$_i$
Y
expGrowth = 0
ssthresh = cwnd

Performance Evaluation

- server deployment: Google server(BBR), Oracle server(CUBIC), both 3 servers located in different places
- client deployment: SUSS on server, different kinds of clients in NZ or Sweden

Throughput



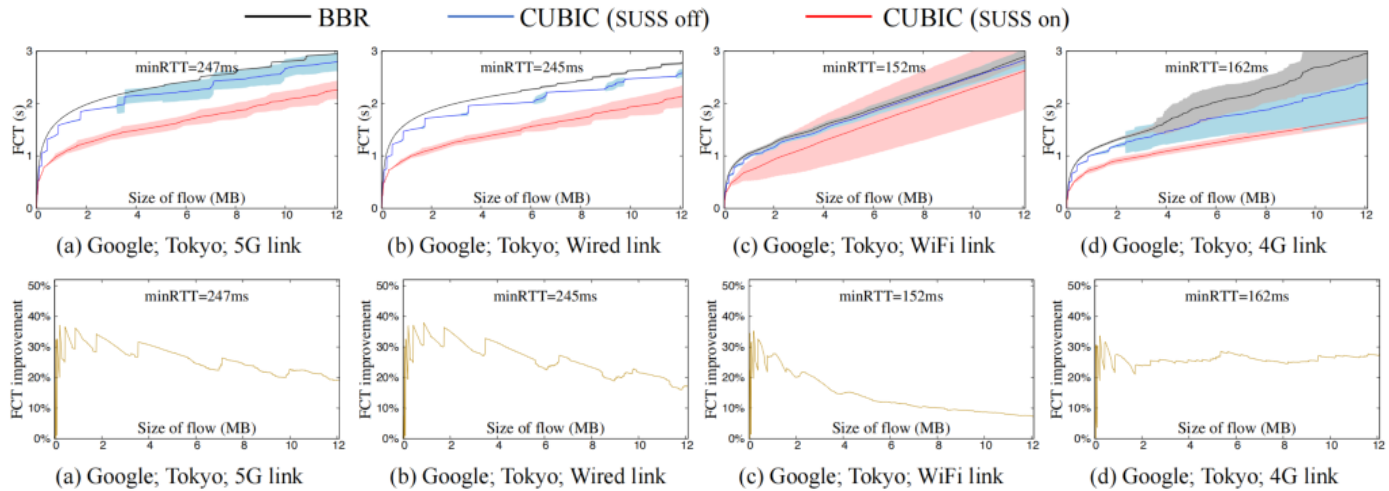


- CUBIC with SUSS exhibits a faster and smoother increase in cwnd and does not incur extra end-to-end delay
- SUSS has brought substantial improvements
  - SUSS can reach $cwnd^*$ more quickly than the traditional slow-start

Performance Evaluation

FCT (Flow Complete Time)

Small Flow

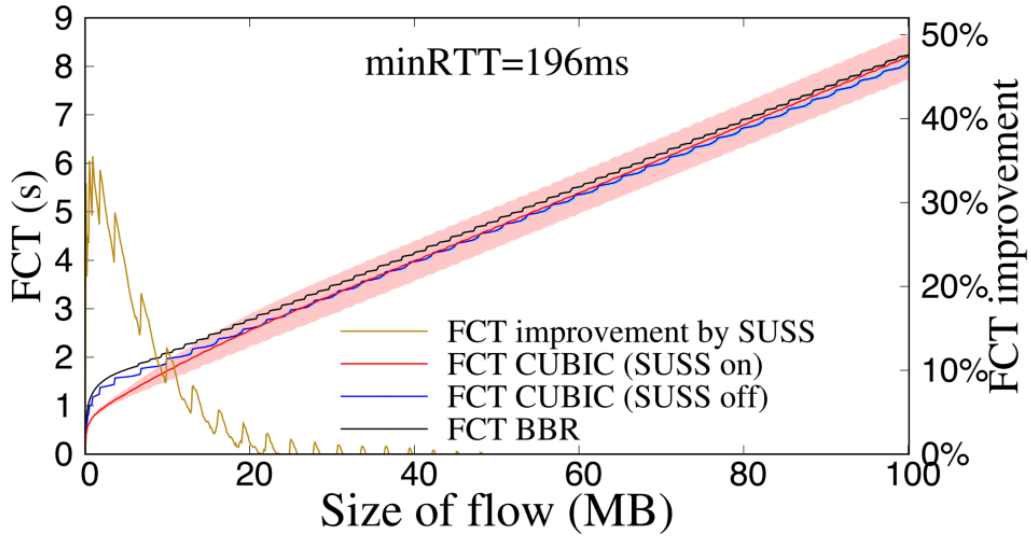


- accelerating slow-start can yield significant gain for small flows
  - as many small flows predominantly reside within the slow-start phase

Performance Evaluation

FCT (Flow Complete Time)

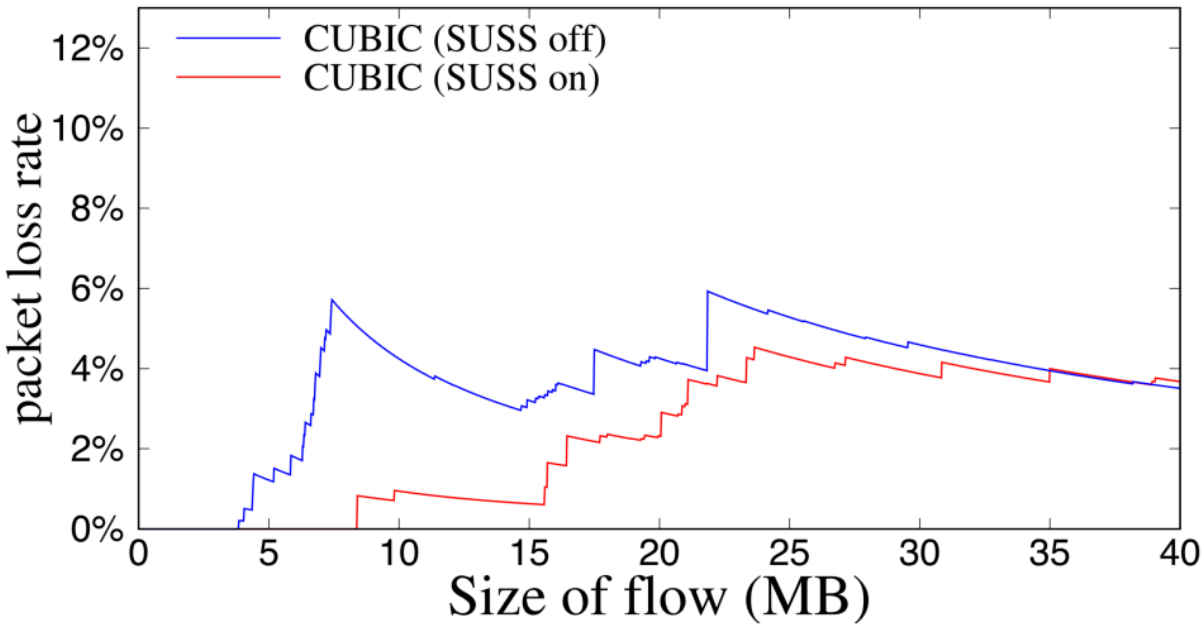
Large Flow



- SUSS enhances the performance of small-sized flows, it does not increase cwnd beyond its optimal value, nor does it impact the FCT of larger flows

Performance Evaluation

Packet Loss

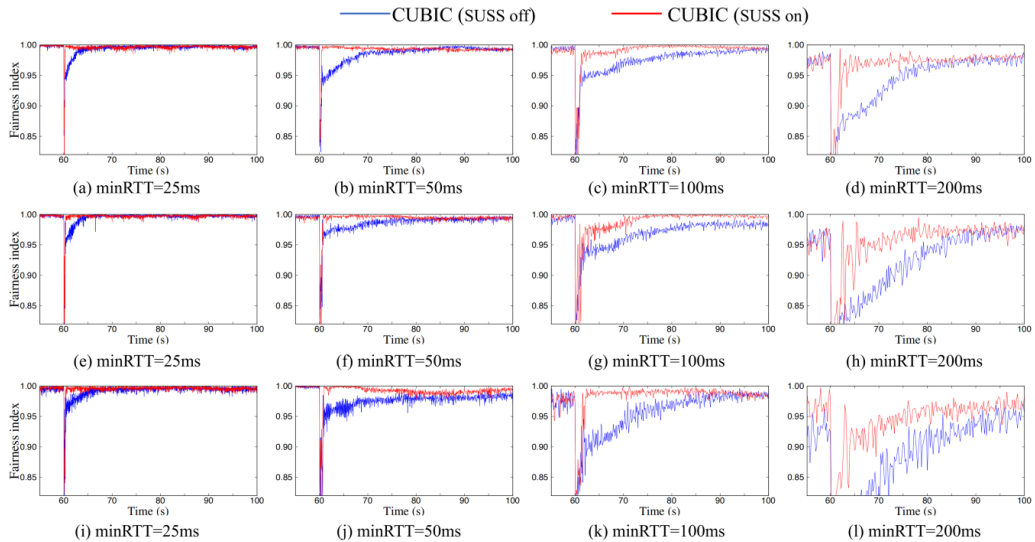


- Pacing can significantly reduce packet density and thereby reduce the packet loss rate
- It will converge when the size of flow increase

Performance Evaluation

Fairness

- $F = \frac{\sum(x_i)^2}{n*\sum(x_i^2)}$, n is the number of flows and x_i is the goodput of the i-th flow → $F \rightarrow 1$, fairness ↑

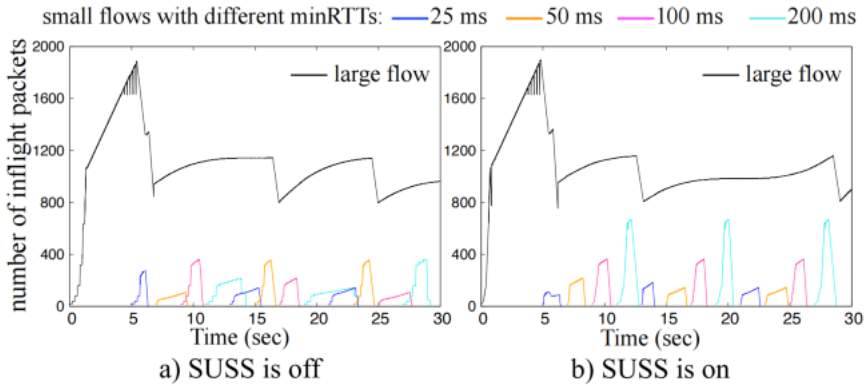


- when a level of fairness is achieved among the four flows, the fifth flow starts downloading
- the initiation of the fifth flow results in an immediate decrease in the value of F , exhibiting a prolonged delay in approaching 1 when SUSS is off

Performance Evaluation

Stability

- A large flow facing the initiation of multiple concurrent small flows



Buffer size	minRTT of large flow	SUSS off: FCT of large flow	SUSS off: FCT of small flows	SUSS on: FCT of large flow	SUSS on: FCT of small flows	Improvement for small flows
1 BDP	25 ms	30.4	1.33	31.2	0.94	29%
	50 ms	37.9	1.18	35.6	0.98	17%
	100 ms	29.0	1.42	28.5	1.22	14%
	200 ms	27.2	2.69	26.9	2.03	25%
2 BDP	25 ms	27.7	1.17	25.7	1.05	10%
	50 ms	26.3	1.54	25.7	1.30	16%
	100 ms	25.8	3.30	25.7	2.09	37%
	200 ms	26.2	5.63	26.1	2.82	50%
Average		28.8	2.28	28.2	1.55	**32%**

(a) The large flow employs CUBIC

Buffer size	minRTT of large flow	SUSS off: FCT of large flow	SUSS off: FCT of small flows	SUSS on: FCT of large flow	SUSS on: FCT of small flows	Improvement for small flows
1 BDP	25 ms	24.1	20.07	24.3	17.54	29%
	50 ms	24.6	13.64	25.2	8.28	17%
	100 ms	25.7	8.11	26.2	4.30	14%
	200 ms	27.6	4.43	28.2	2.52	25%
2 BDP	25 ms	25.5	2.24	25.7	1.45	10%
	50 ms	25.6	2.29	25.8	1.66	16%
	100 ms	25.9	2.75	26.1	1.90	37%
	200 ms	26.3	5.70	26.5	4.92	50%
Average		25.7	7.40	26.0	5.32	**28%**

(b) The large flow employs BBRv1

Buffer size	minRTT of large flow	SUSS off: FCT of large flow	SUSS off: FCT of small flows	SUSS on: FCT of large flow	SUSS on: FCT of small flows	Improvement for small flows
1 BDP	25 ms	28.5	1.44	29.2	1.04	29%
	50 ms	27.3	1.49	28.4	1.16	17%
	100 ms	27.0	1.69	27.2	1.34	14%
	200 ms	27.5	3.13	28.0	1.83	25%
2 BDP	25 ms	27.0	1.81	28.0	1.23	10%
	50 ms	26.1	1.90	26.6	1.38	16%
	100 ms	26.2	2.29	26.4	1.72	37%
	200 ms	26.4	5.01	26.6	4.17	50%
Average		27.0	2.35	27.6	1.73	**26%**

(c) The large flow employs BBRv2

- SUSS improves FCT of small CUBIC flows without compromising the stability of the large flow

Thanks for Listening

Paper · GitHub · Conference video