RACK: a time-based fast loss recovery
draft-cheng-tcpm-rack-01

Yuchung Cheng
Neal Cardwell
Google

IETF96 Berlin, July 2015

Motivation: simplification

Linux TCP loss recovery: RFC5681, RFC6675, RFC5827, RFC4653, RFC5682, FACK, thin-dupack, tail loss probe (TLP), ...

Does it need to be that complicated?

Do they even work well?

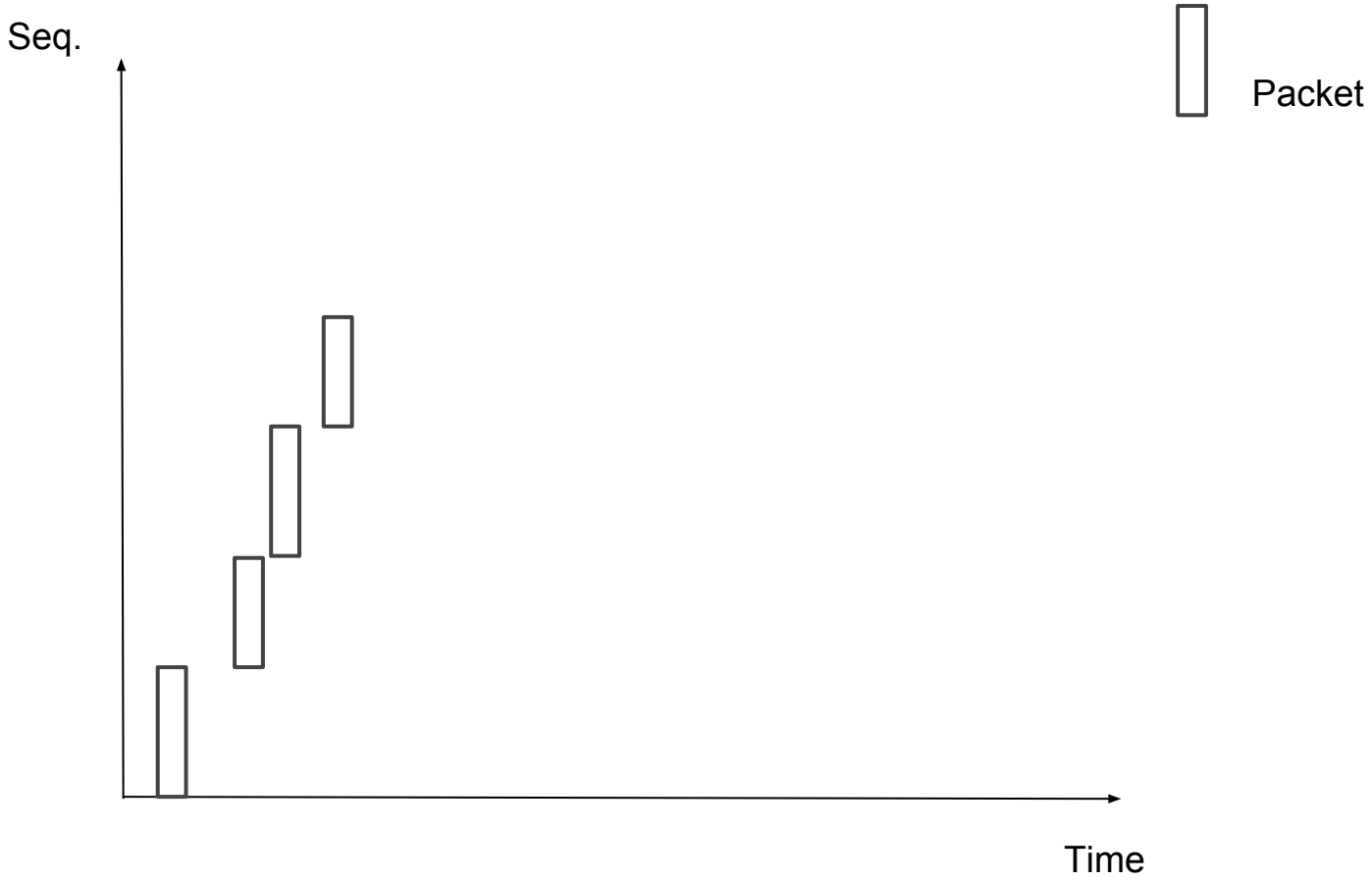
But most of them share a common rationale ...

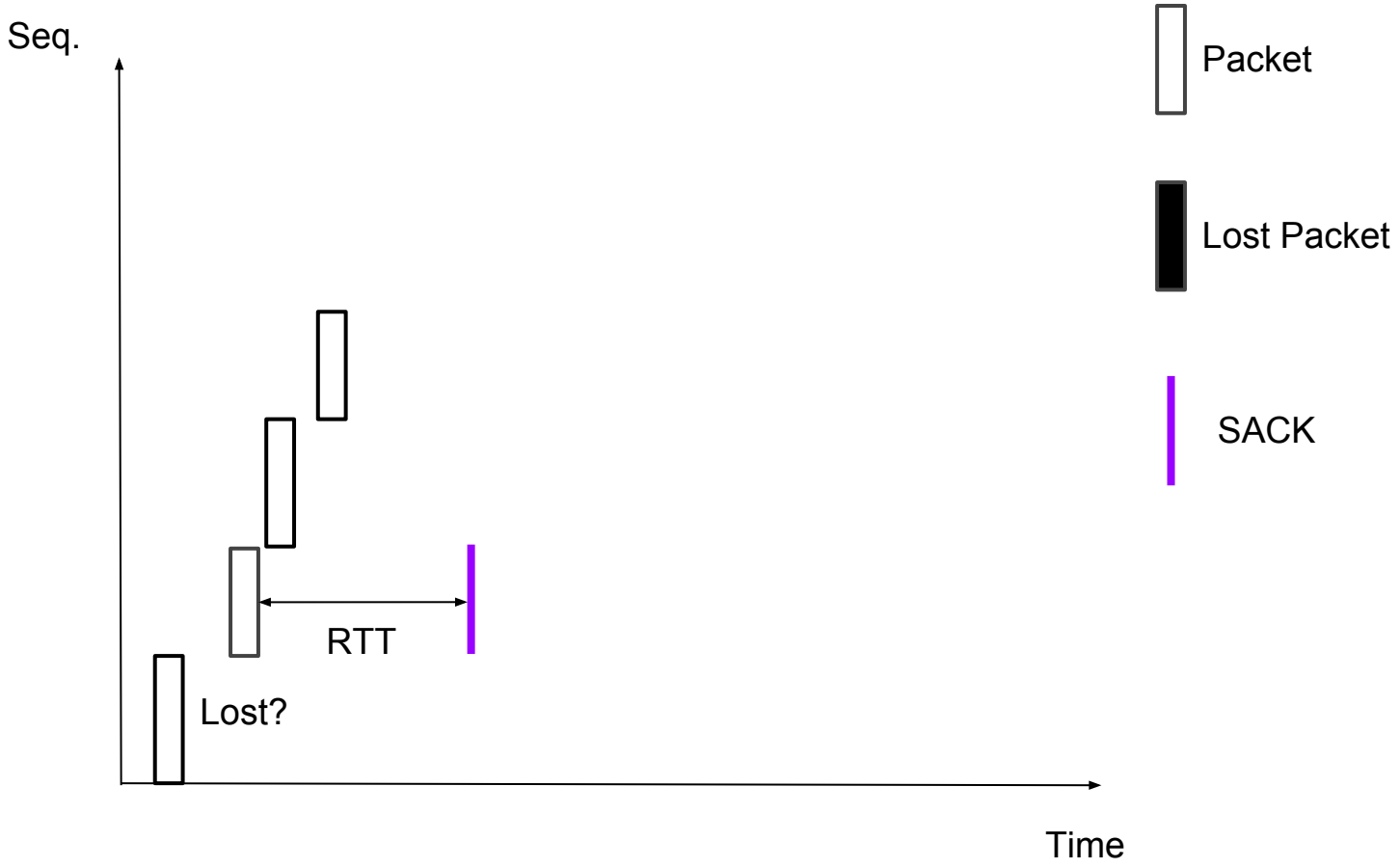
What is RACK (Recent Ack)?

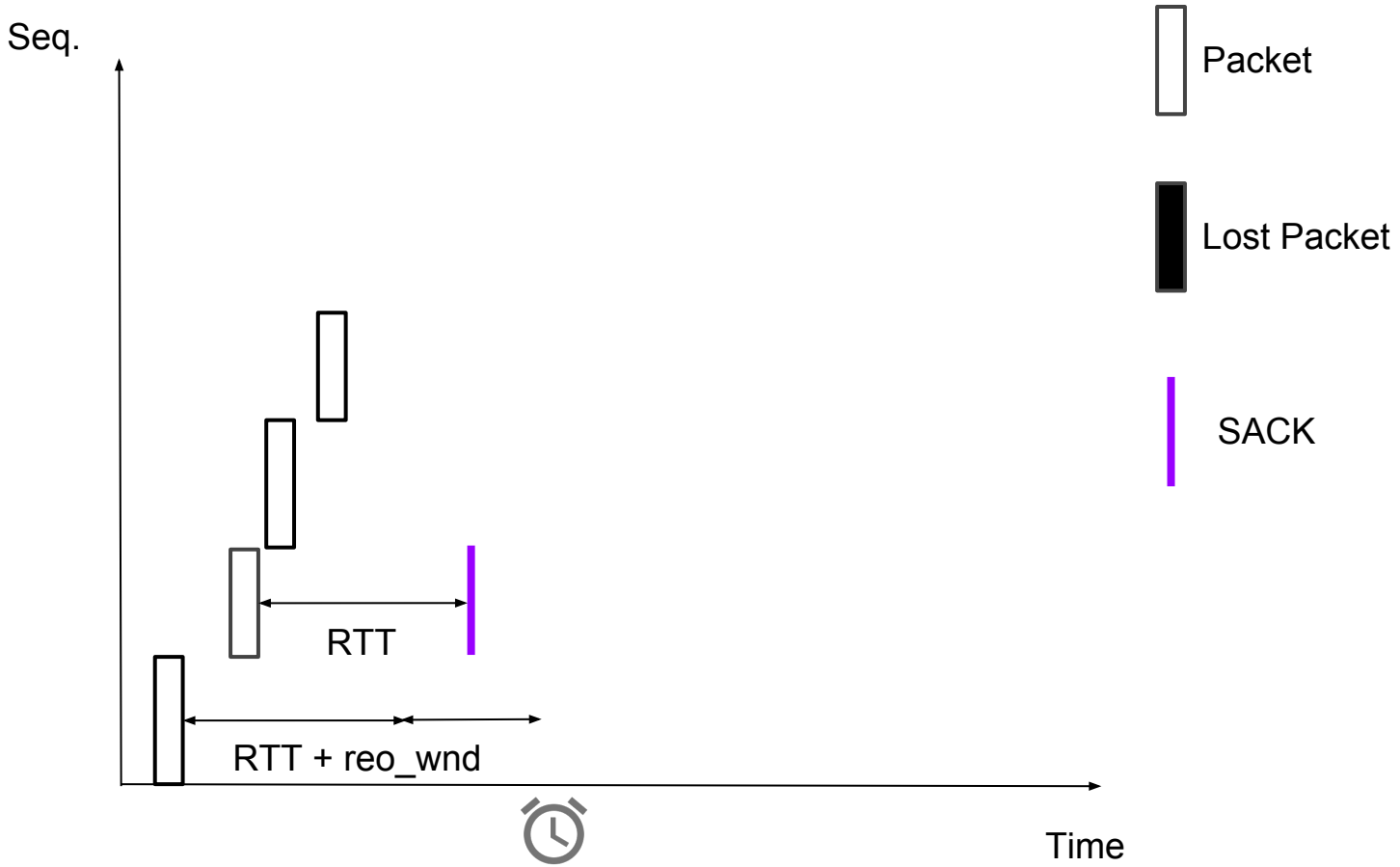
Monitors the delivery process of every packet (incl. rtx)

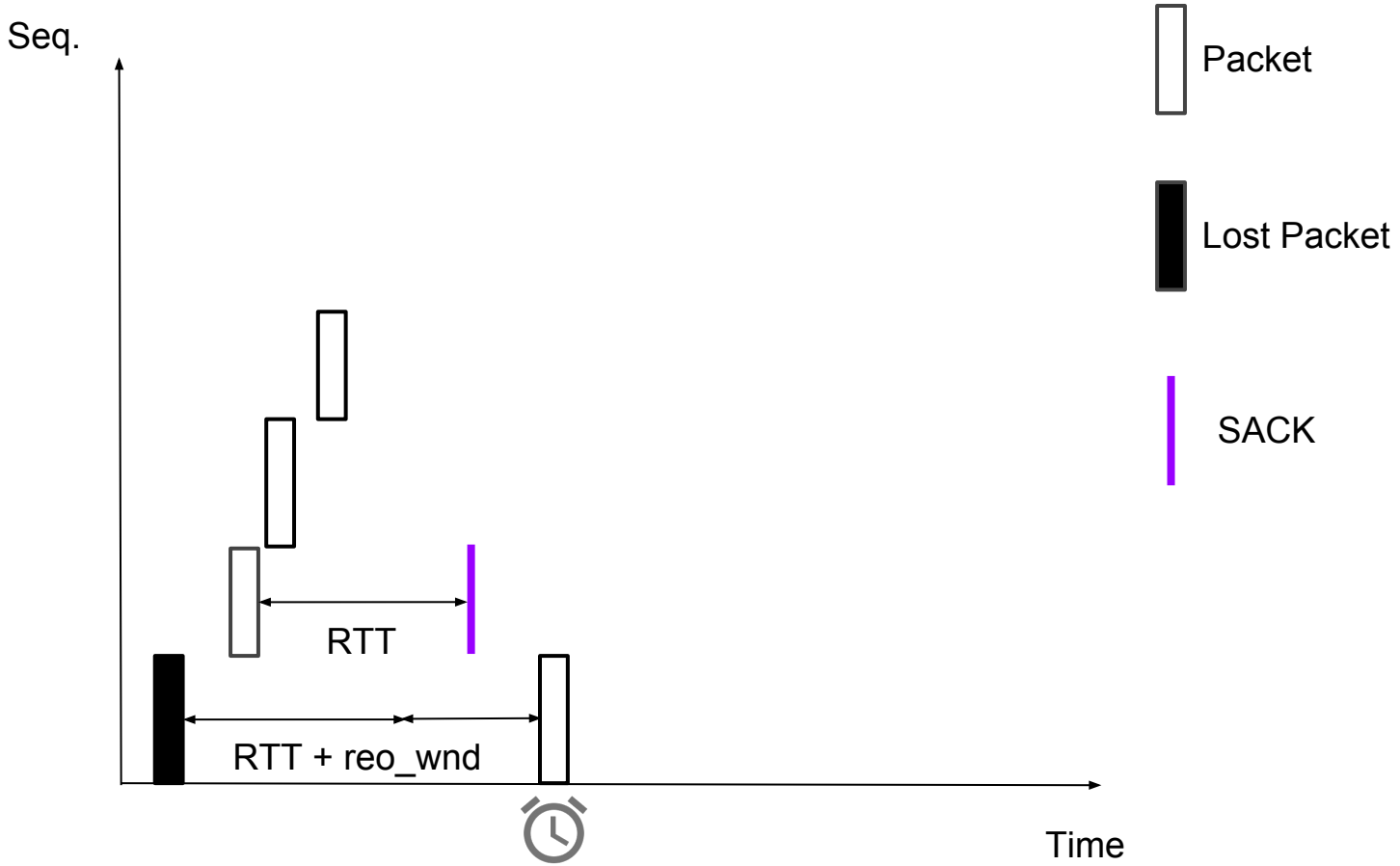
A sender sends two packets P1 and P2:

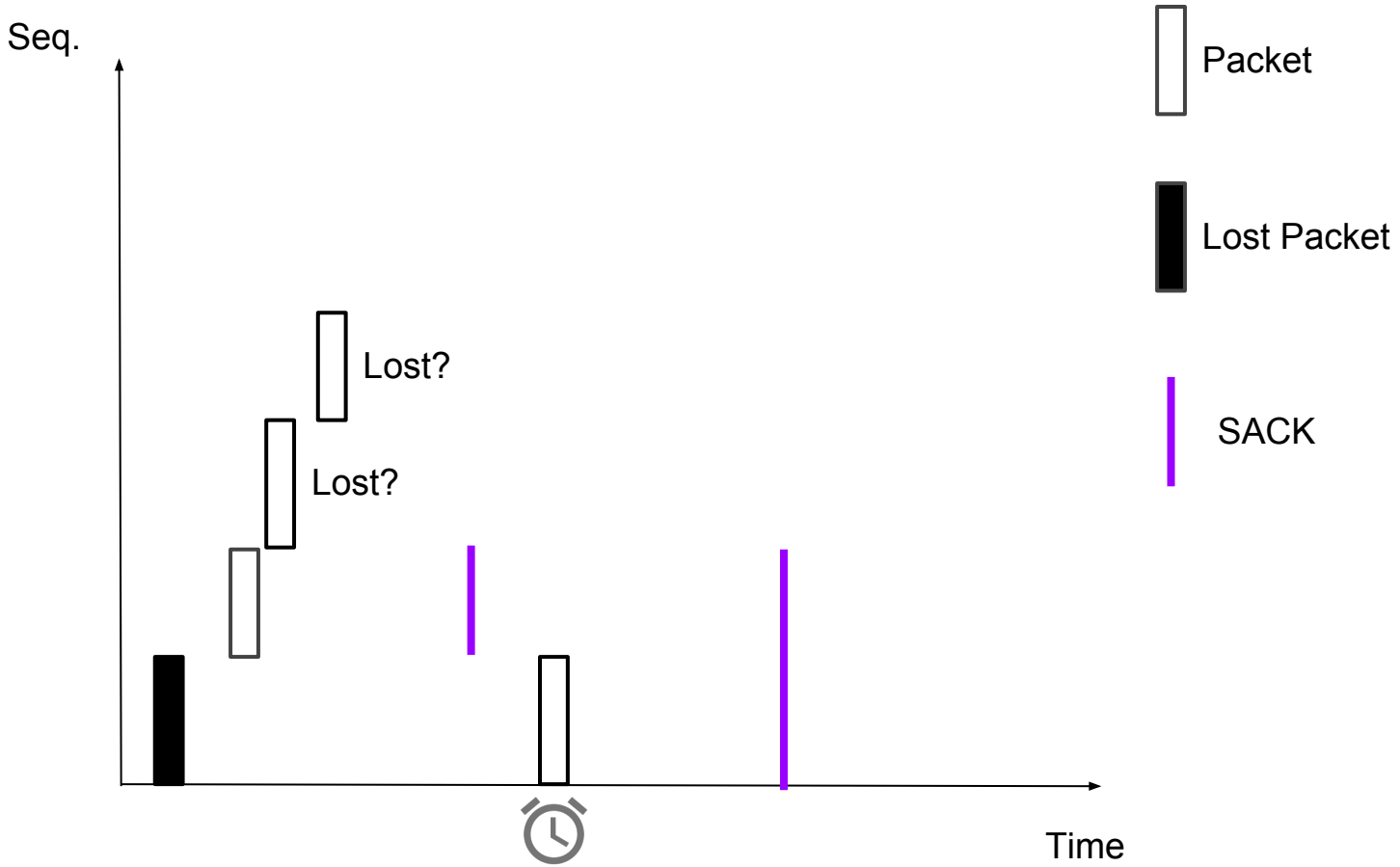
If P2 is delivered, P1 is lost if it was sent more than $RTT + $reo_wnd ago

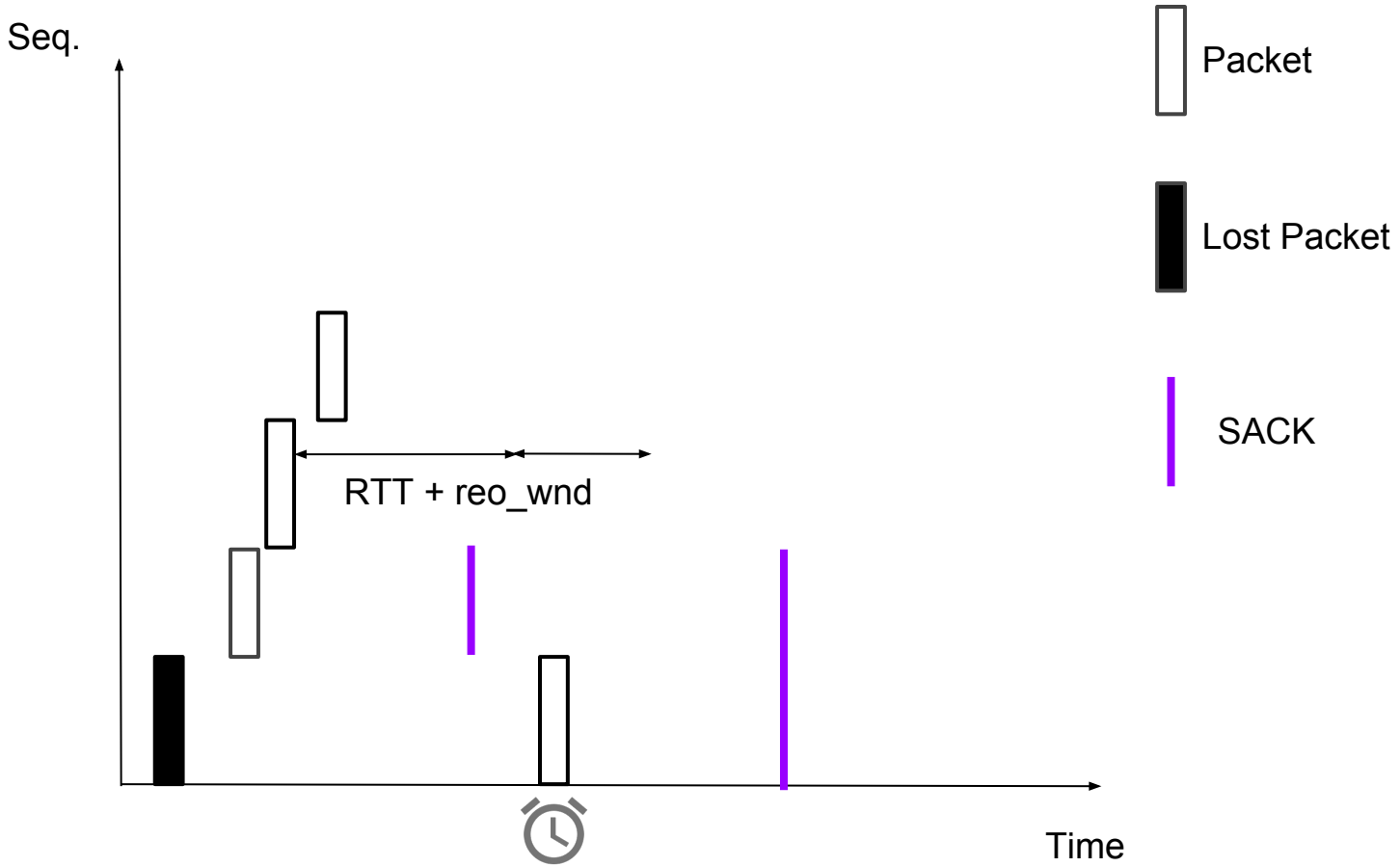

Seq.
Packet
Time

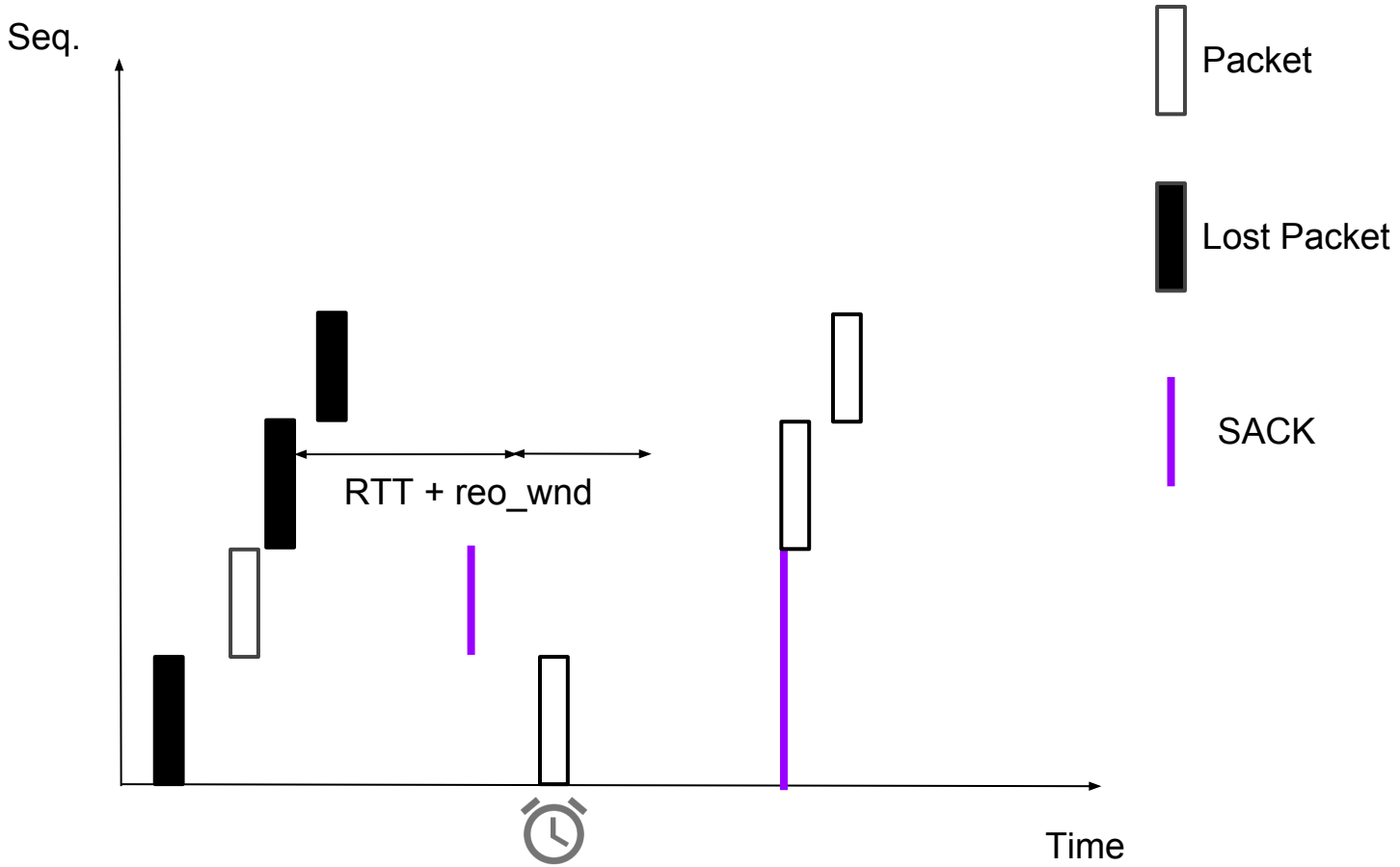

Seq.
Packet
Lost Packet
SACK
RTT
Lost?
Time

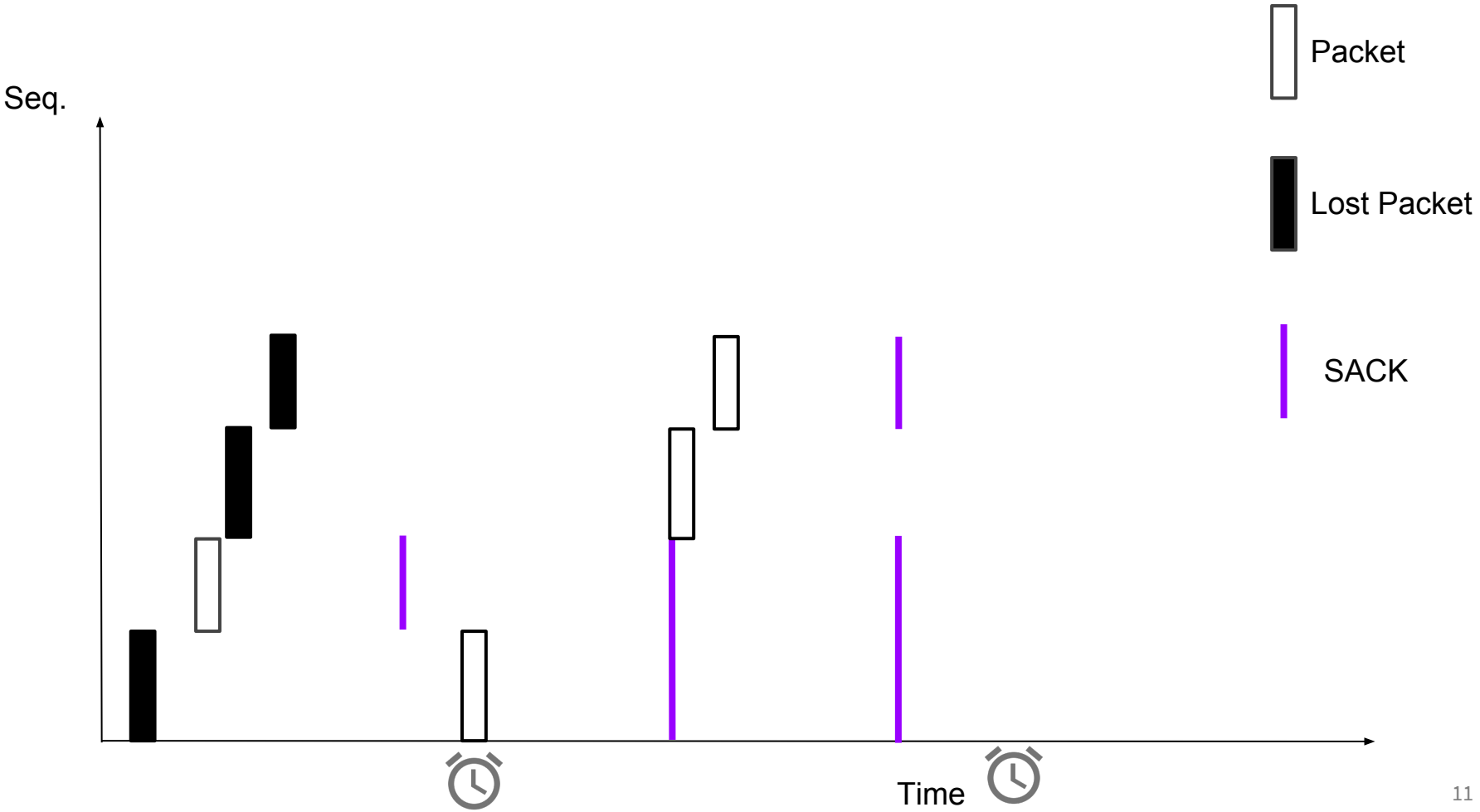

Seq.
Packet
Lost Packet
SACK
RTT
RTT + reo_wnd
Time

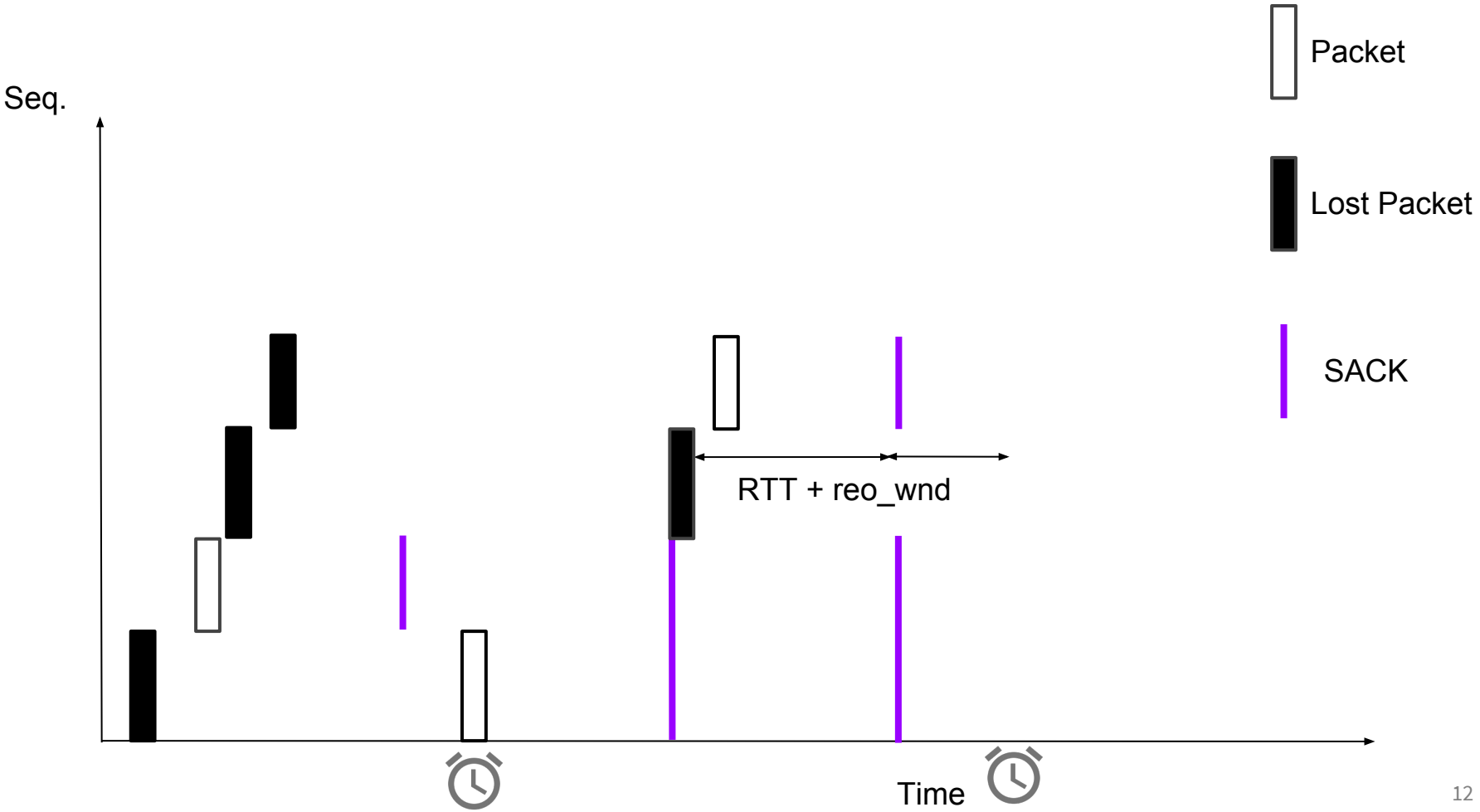

Seq.
Packet
Lost Packet
SACK
RTT
RTT + reo_wnd
Time

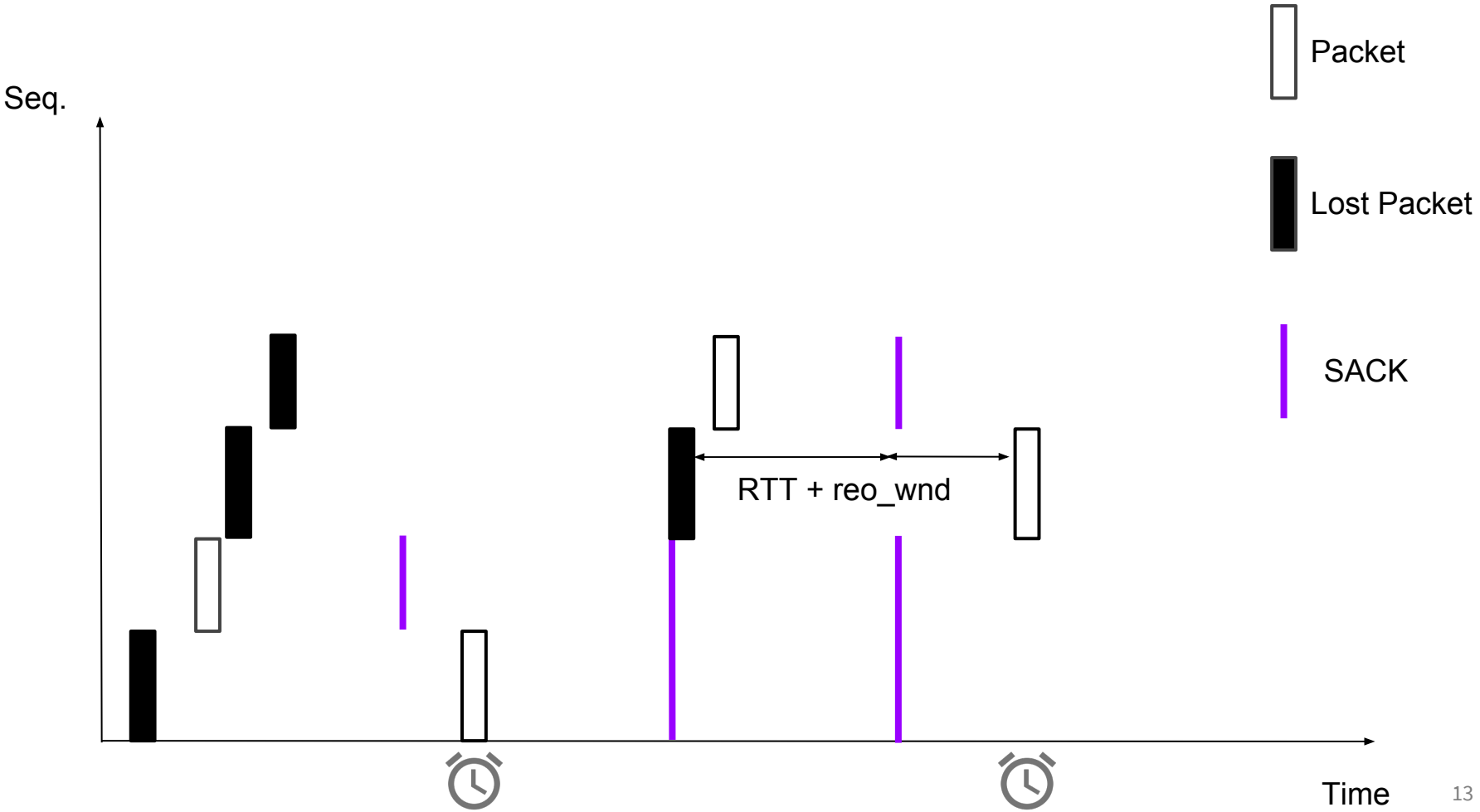

Seq.
Lost?
Lost?
Time
Packet
Lost Packet
SACK


Seq.
Packet
Lost Packet
SACK
RTT + reo_wnd
Time


Seq.
Packet
Lost Packet
SACK
RTT + reo_wnd
Time


Packet
Lost Packet
SACK
Seq.
Time


Packet
Lost Packet
SACK
Seq.
RTT + reo_wnd
Time


Packet
Lost Packet
SACK
Seq.
RTT + reo_wnd
Time

Why RACK makes more sense

Tail drops and lost retransmission are common

1. Structured traffic
2. Traffic policing [1]

Need to use every packet’s info

[1] An Internet-wide analysis of traffic policing. SIGCOMM 2016

Why RACK makes more sense

Common reorderings:

1. Last (runt) packet of a burst gets delivered first: P4, P1, P2, P3
2. Small out-of-order burst: P[4-6], P[1-3], P[9-11], P7,P8
3. Route becomes shorter: P[21-40], P[1-20]

RACK helps 1,2 but not 3

Back to motivation: simplification

Linux TCP loss recovery: ~~RFC5681, RFC6675, RFC5827, RFC4653, FACK, thin-dupack~~
RACK, tail loss probe (TLP), …

RACK works naturally with TLP (no change)

RACK vs FACK exp

FACK [1]

- A packet is lost if end_seq + 3 * mss < highest_sack
- Default on Linux since 2005 (off on after reordering)
- FACK inspired RACK

Multi-days experiment on Google servers near Berlin

1. FACK (control)
2. RACK but disable FACK (exp)

[1] Mathis, M. and M. Jamshid, "Forward acknowledgement: refining TCP congestion control", CCR, 1996

RACK vs FACK exp result

RACK significantly reduces stalls in the recovery process

- Disorder state: 82%
- Loss state: 44%
- Recovery state: 7%

Recovery latency reduction: total 4%, mean 11% **

Surprise benefit on TSO: 15% less SACK merge events

** We expect better results with two recent (last friday!) optimizations

Status

Code deployed at Google since 2014

- First upstreamed to Linux in 2015
- Timer enhancement soon be upstreamed

Other implementations coming? FreeBSD, Windows

Next steps in IETF

1. WG interest?
2. Merge TLP draft to have **ONE** loss recovery RFC