# *Fast and Cautious:* Leveraging Multi-path Diversity for Transport Loss Recovery in Data Centers

**Guo Chen**

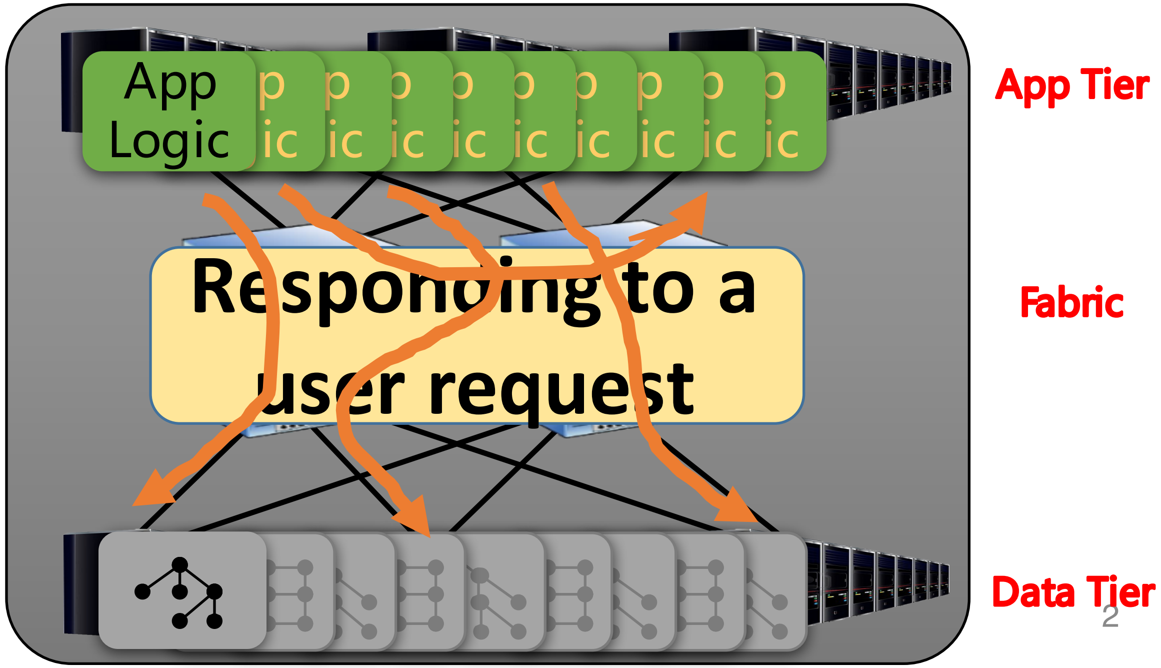
Yuanwei Lu, Yuan Meng, Bojie Li, Kun Tan, Dan Pei, Peng Cheng, Layong (Larry) Luo, Yongqiang Xiong, Xiaoliang Wang, and Youjian Zhao

Tsinghua University 清華大學 · Microsoft Research · 中国科学技术大学 University of Science and Technology of China · 南京大學 NANJING UNIVERSITY

# Motivation

- Services care about the **tail flow completion time** (tail FCT)
  - ***Large number of flows*** generated in each operation
  - Overall performance governed by the ***last completed flows***

Large-scale web application
hosted in
**Data Center Network (DCN)**



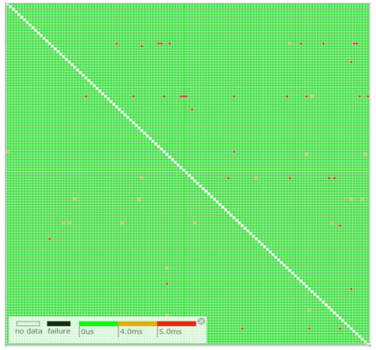
# Motivation

- Services care about the **tail flow completion time** (tail FCT)
  - ***Large number of flows*** generated in each operation
  - Overall performance governed by the ***last completed flows***

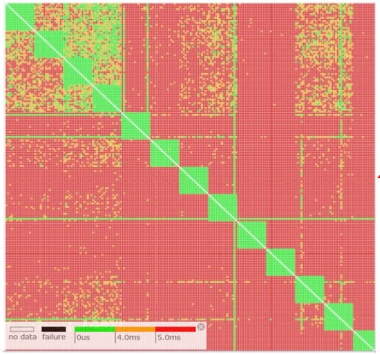
- But packet loss **hurts tail FCT**
  - ***Real case*** in a Microsoft Azure's DCN

DCN tail latency visualization

[Pingmesh (SIGCOMM'15)]



(a) Normal



(b) Spine failure

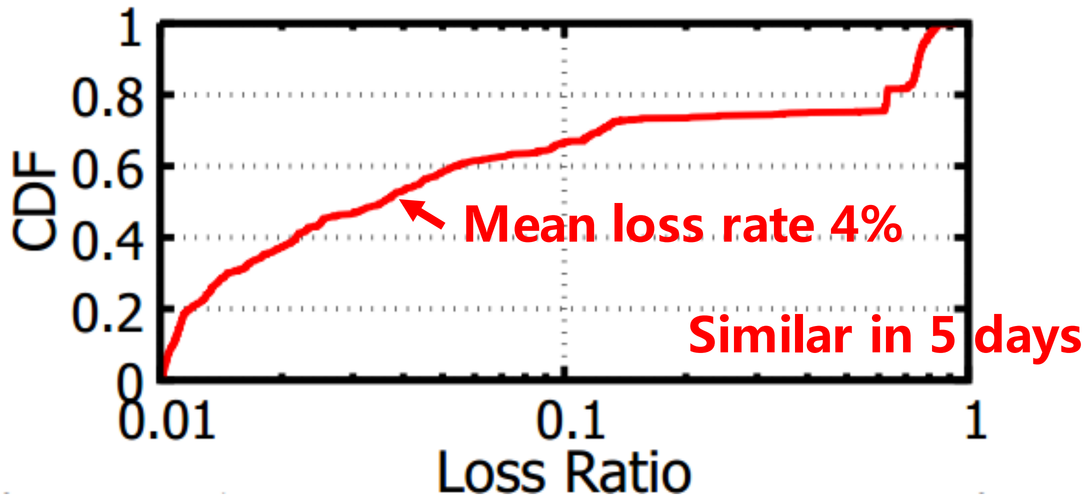
Spine switch **2%** random drop rate --> increase of 99th percentile latency of **all users**

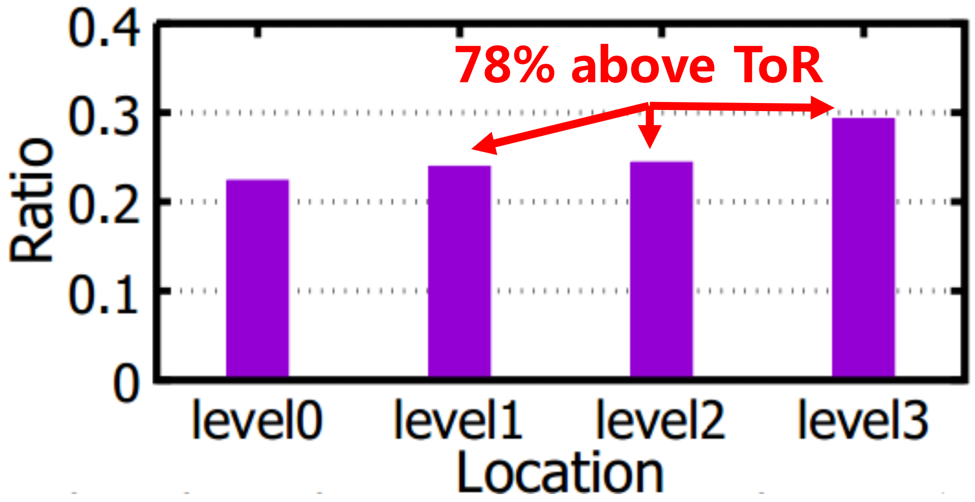
# Outline

- Motivation
- **Packet Loss in DCN**
- Impact of Packet Loss
- Challenge for Loss Recovery
- FUSO Design
- Evaluation
- Summary

# Packet Loss in DCN

- Loss characteristics
  - Measured in a Microsoft production DCN during Dec. 1st-5th, 2015




*Loss rate and location distribution of lossy links (loss rate > 1%)*

1) Loss **frequently** happens (the overall loss rate is low)
2) Most losses happen **in the network** instead of the edge

# Packet Loss in DCN

- Reasons causing loss
  - Congestion loss — **Bursty; Transient**
    - Uneven load-balance
    - Incast

    → Greatly mitigated (e.g., 1%->0.01%) [Jupiter Rising SIGCOMM'15]
  - Failure loss — **Complex; Hard to detect**
    - Silent random drop
    - Packet black-hole

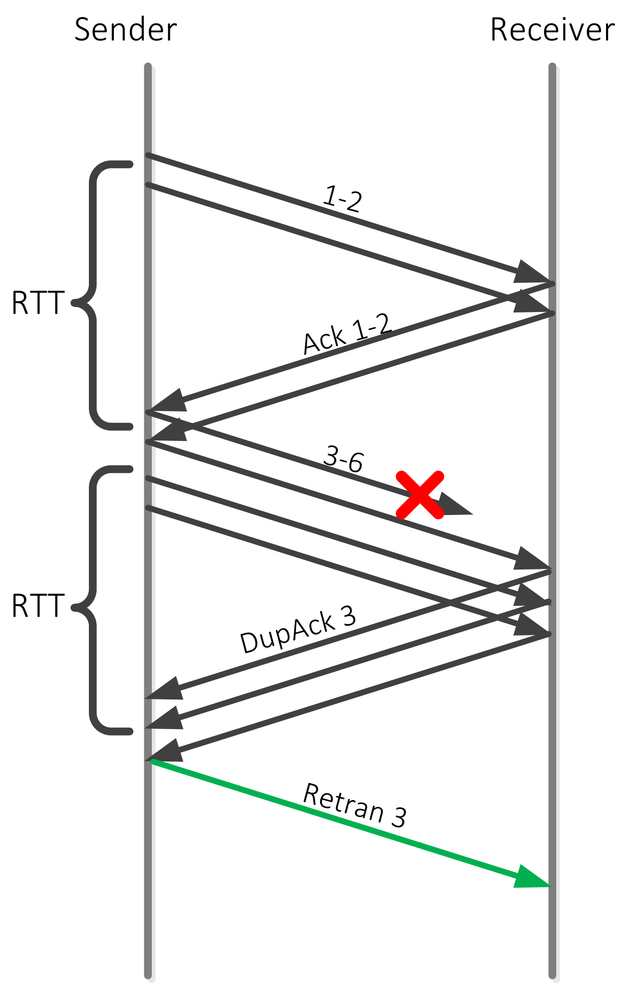
    → Common & Huge impact on performance [Pingmesh SIGCOMM'15]

# Outline

- Motivation
- Packet Loss in DCN
- **Impact of Packet Loss**
  - **Why loss hurts the tail?**
  - **How hard loss hurts?**
- Challenge for Loss Recovery
- FUSO Design
- Evaluation
- Summary

# How TCP Handles Loss?

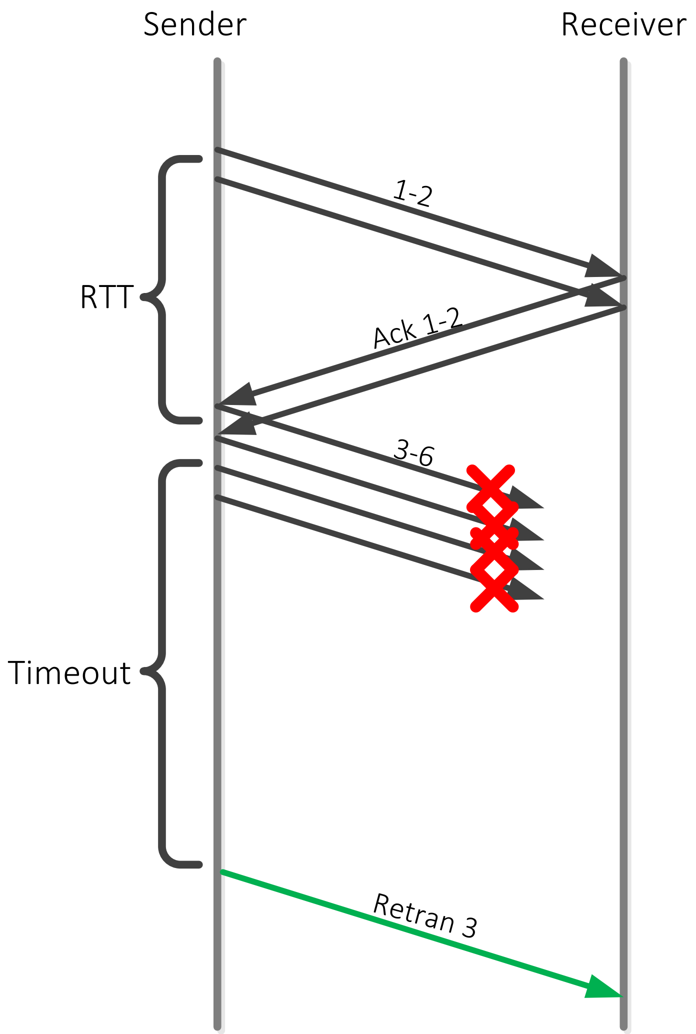
- Fast recovery
  - Wait for certain number of DACKs to detect the loss and retransmit



# How TCP Handles Loss?

- Fast recovery
  - Wait for certain number of DACKs to detect the loss and retransmit
- Timeout (RTO)
  - If not enough DACKs return, retransmit after a timeout

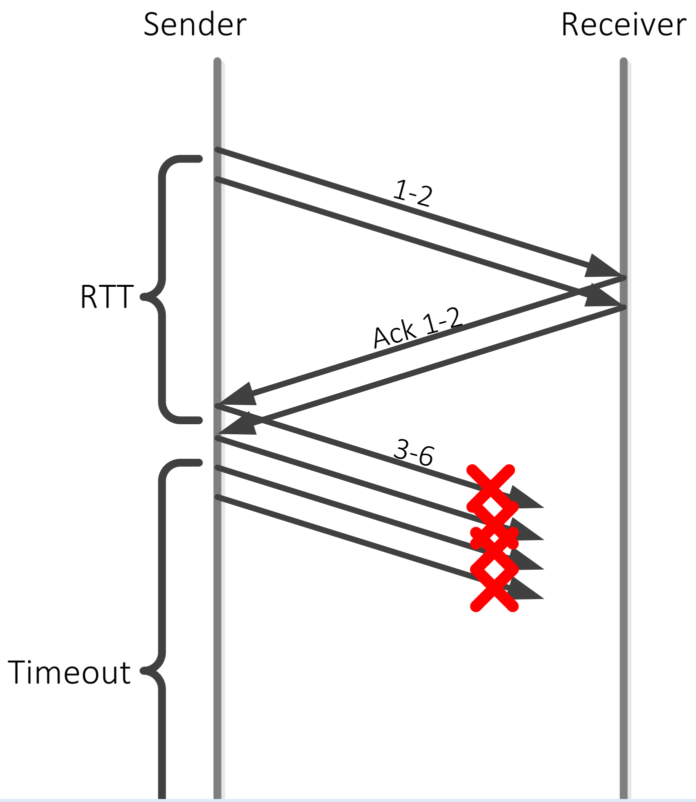
**RTO >> RTT** e.g. RTO=5ms, RTT<100us
[Pingmesh (SIGCOMM'15), DCTCP (SIGCOMM'10)]



# How TCP Handles Loss?

- Fast recovery
  - Wait for certain number of DACKs to detect the loss and retransmit
- Timeout (RTO)
  - If not enough DACKs return, retransmit after a timeout

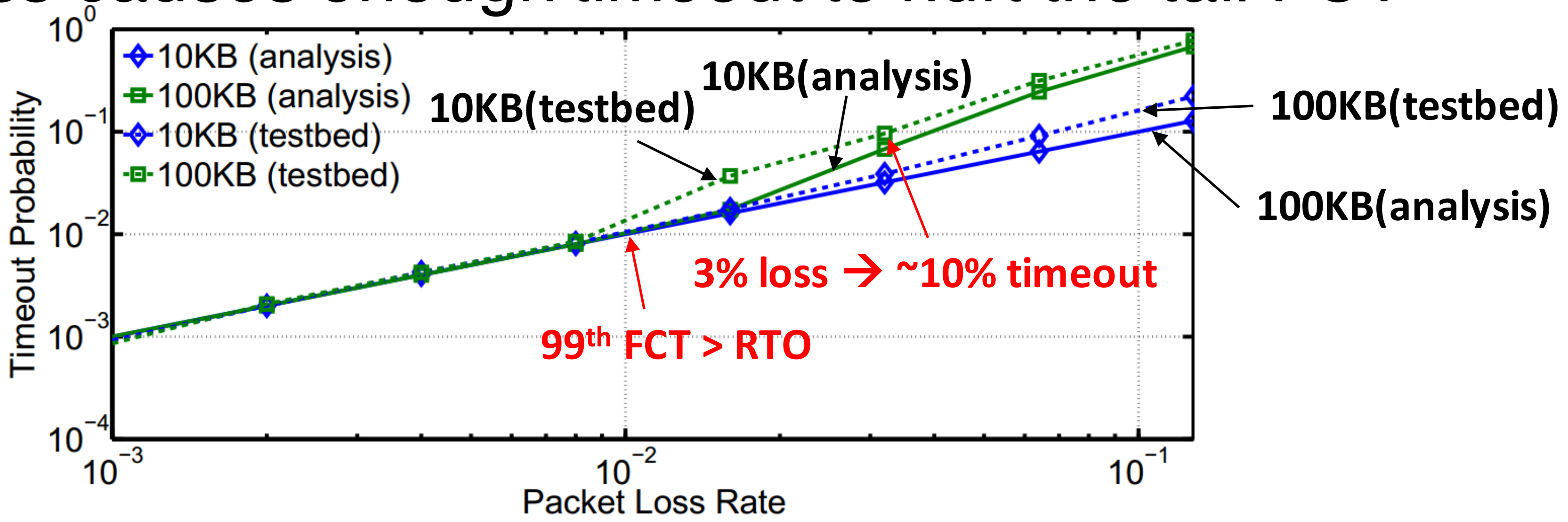
**RTO >> RTT** e.g. RTO=5ms, RTT<100us
[Pingmesh (SIGCOMM'15), DCTCP (SIGCOMM'10)]




**Encountering one RTO → dramatically increase the FCT**

# Loss Incurs Timeout

- A little loss causes enough timeout to hurt the tail FCT

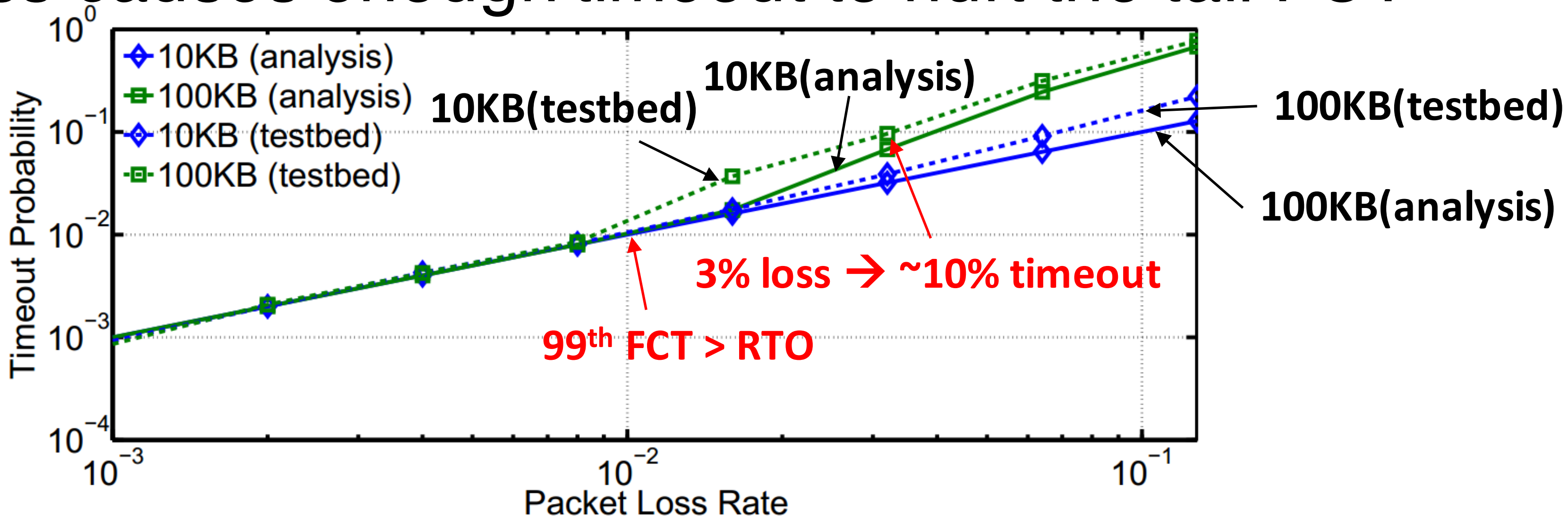



Timeout probability of flows with different sizes passing a path with different packet loss rate

1. 1% loss → more than 1% flows timeout
2. Larger flows (e.g. 100KB)
   a. timeout ratio sharply grows when loss rate > 1%

# Loss Incurs Timeout

- A little loss causes enough timeout to hurt the tail FCT




Timeout probability of flows with different sizes passing a path with different packet loss rate

**To avoid RTO**

# Outline

- Motivation
- Packet Loss in DCN
- Impact of Packet Loss
- **Challenge for Loss Recovery**
- FUSO Design
- Evaluation
- Summary

# Challenge for TCP Loss Recovery

- Prior works **add aggressiveness** to congestion control to do loss recovery **before** timeout (RTO)
  - Tail Loss Probe (TLP) [SIGCOMM'13, RFC 5827]
    - transmit one prober **after 2RTT**
  - Instant Recovery (TCP-IR) [SIGCOMM'13, RFC 5827]
    - generate **an FEC packet** for every group of packets (up to 16)
    - FEC packets also act as probers, **delayed 1/4RTT** before sent
  - Proactive/RepFlow [SIGCOMM'13, INFOCOM'14]
    - **Duplicate** every packet/flow

# Challenge for TCP Loss Recovery

- **How long to wait** before sending recovery packets?
  - For congestion loss
    - Should **delay enough** in case of worsening congestion

**Bursty:**
**Lead to multiple consecutive losses**
[Incast (WREN'09), DCTCP (SIGCOMM'10)]

# Challenge for TCP Loss Recovery

- **How long to wait** before sending recovery packets?
  - For congestion loss
    - Should **delay enough** in case of worsening congestion
  - For failure loss such as random drop
    - Should recover as **fast** as possible, otherwise already increase the FCT

- **Wait 2RTT is too costly** [TLP SIGCOMM'13, RFC 5827]
- **Accurate & high-precision RTT measurement is challenging**

# Brief Summary

- Loss easily incurs **timeout** to hurt the tail
- To prevent timeout, prior works **add fixed aggressiveness** to recover loss before timeout
- Hard to adapt to various loss conditions
  - Should be **fast** for failure loss
  - Should be **cautious** for congestion loss

**How to accelerate loss recovery as soon as possible, under various loss conditions without causing congestion?**

# Outline

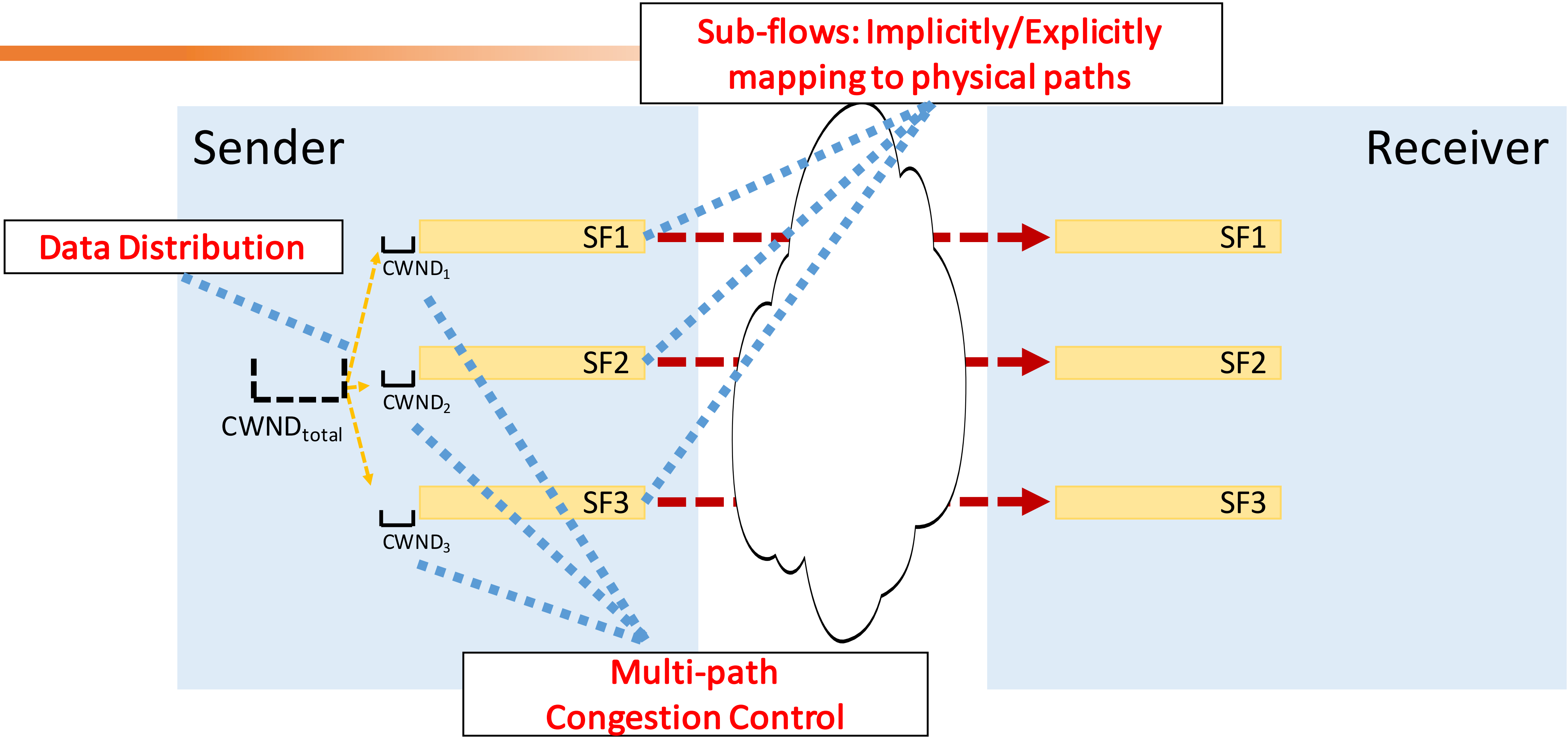
- Motivation
- Packet Loss in DCN
- Impact of Packet Loss
- Challenge for Loss Recovery
- **FUSO Design**
- Evaluation
- Summary

# FUSO: ***F***ast M***u***lti-path Lo***s***s Rec***o***very

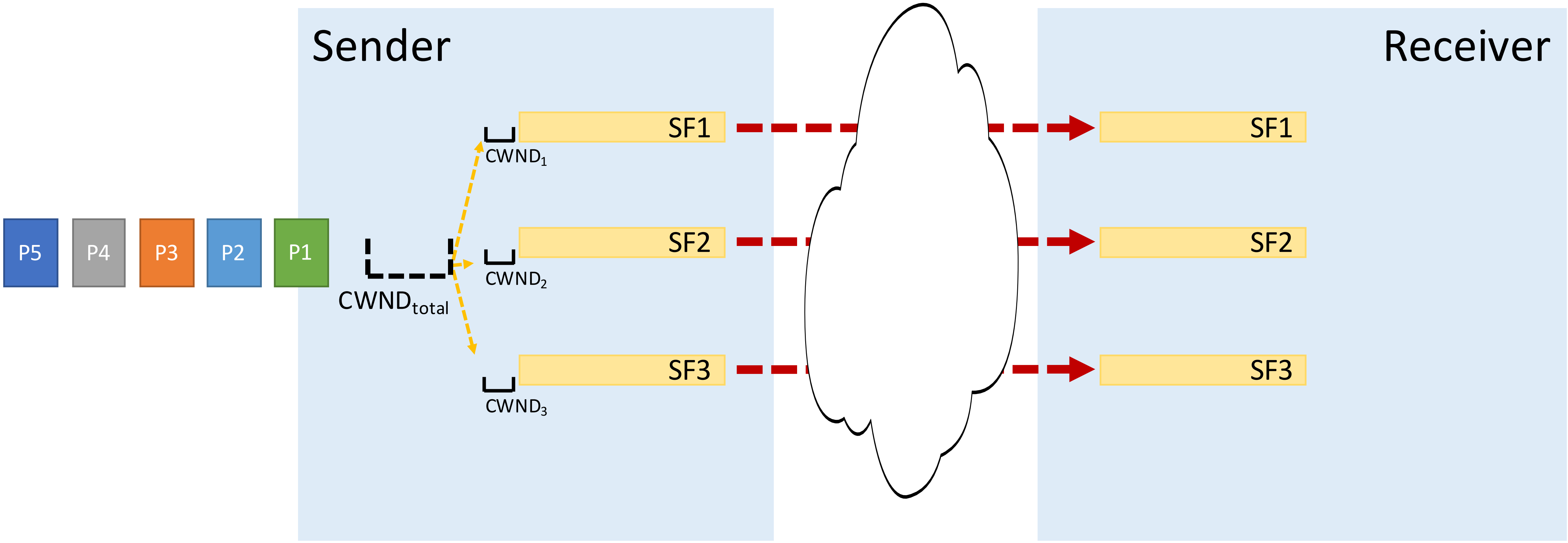
- Utilize the **"good" paths** to proactively conduct loss recovery for **"bad" paths**
  - Leveraging path diversity (multiple paths; a few encounter loss)

- Fast and Cautious
  - **Fast**
    - Proactive (immediate) recovery for potential packet loss utilizing spare transmission opportunity
  - **Cautious**
    - Strictly follow congestion control without adding aggressiveness

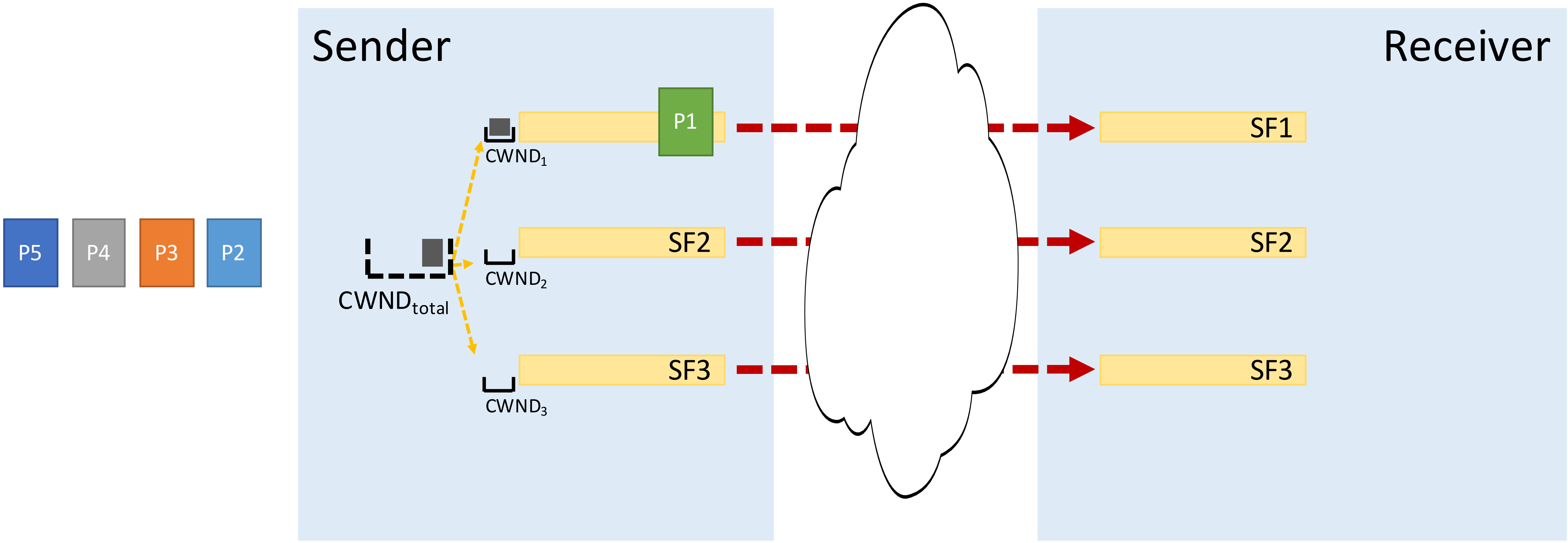
# Multi-path Transport Background



# FUSO

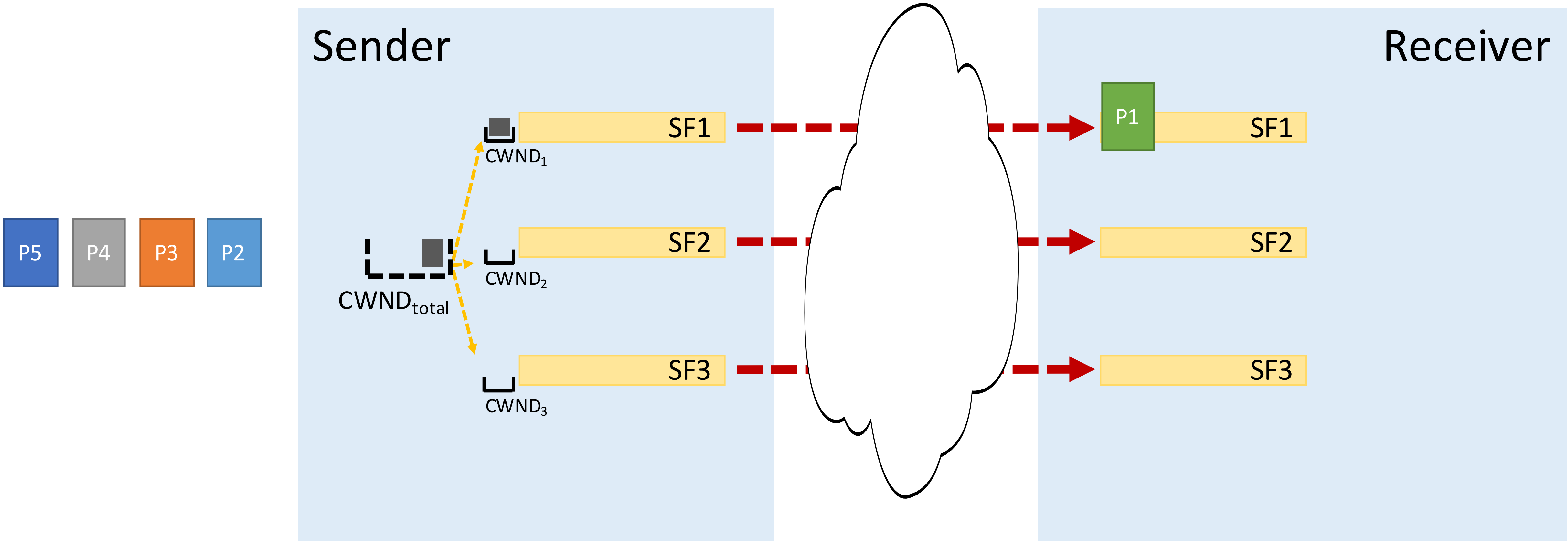

Sender
Receiver
P5
P4
P3
P2
P1
CWND_total
CWND_1
CWND_2
CWND_3
SF1
SF2
SF3
SF1
SF2
SF3

# FUSO

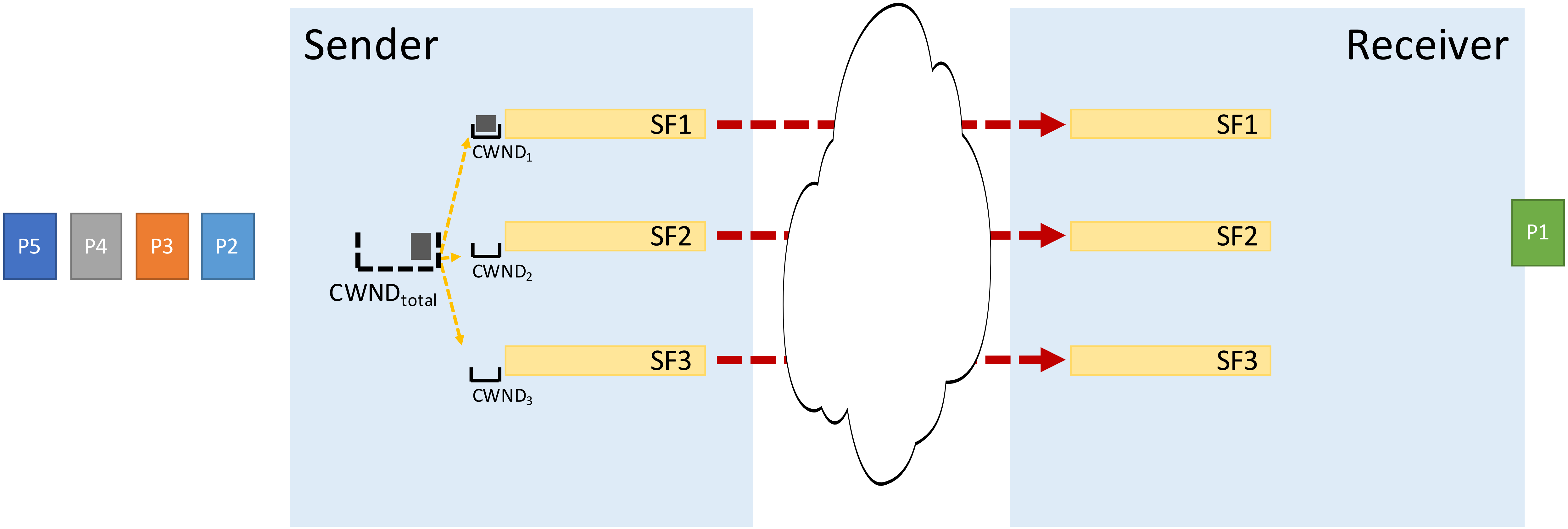


Sender
Receiver
P5
P4
P3
P2
P1
CWND1
CWND2
CWND3
CWNDtotal
SF2
SF3
SF1
SF2
SF3

# FUSO

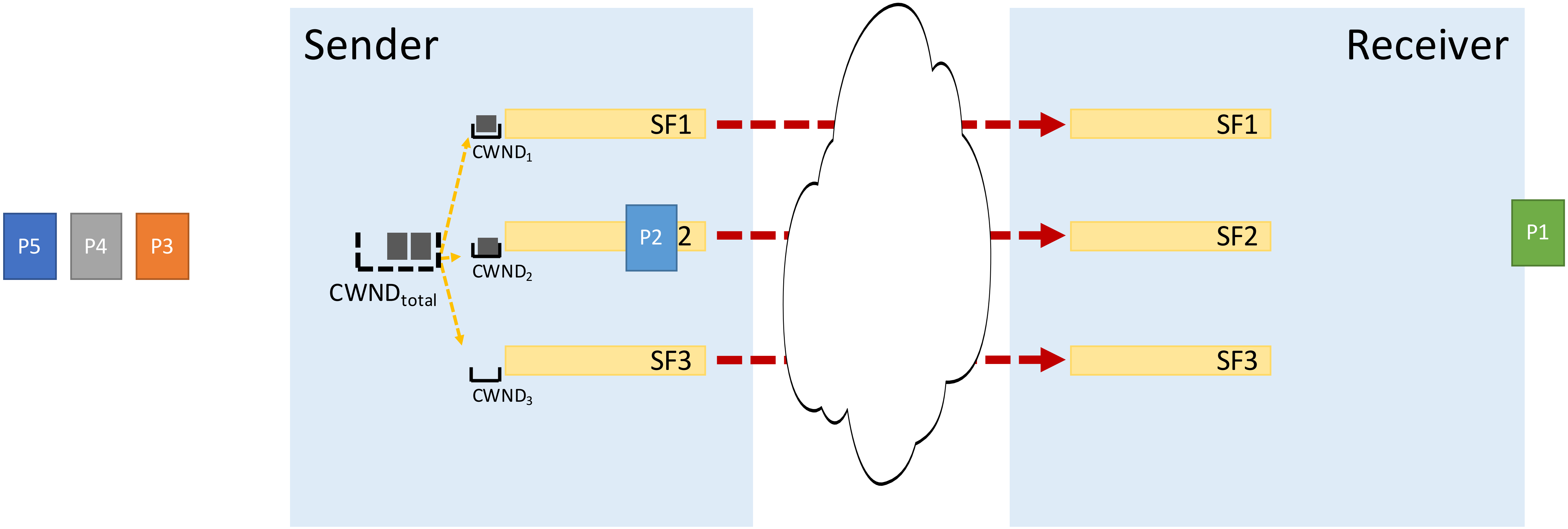

P5
P4
P3
P2
Sender
CWND_total
CWND_1
CWND_2
CWND_3
SF1
SF2
SF3
Receiver
P1
SF1
SF2
SF3

# FUSO

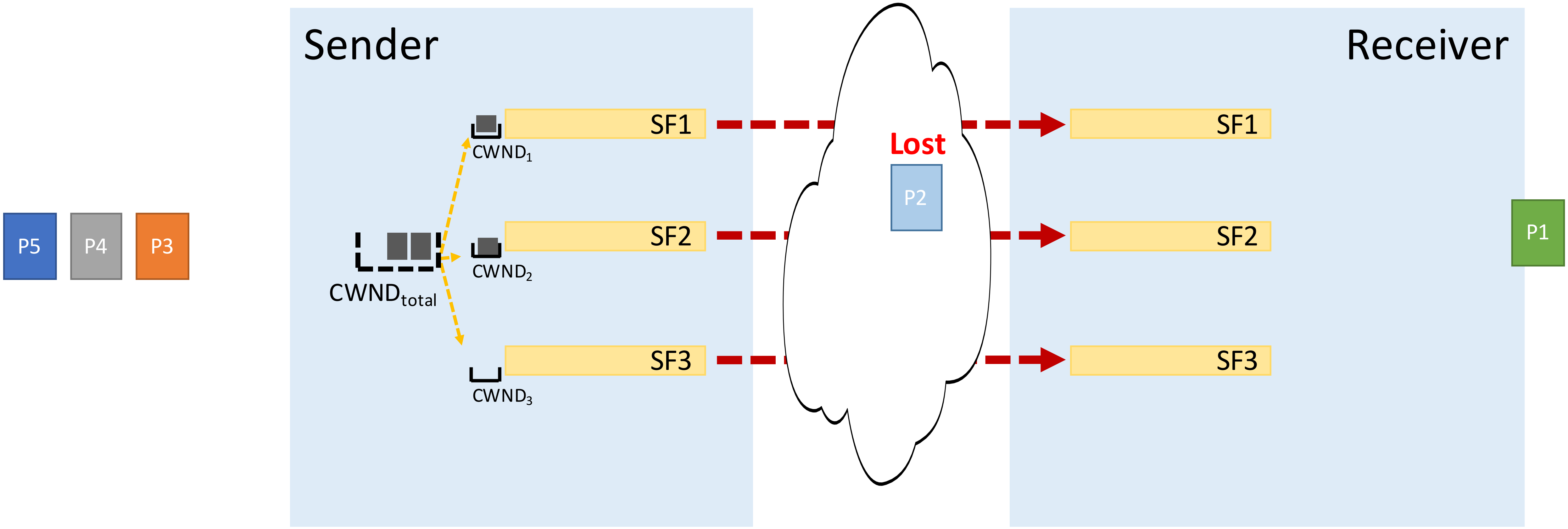


P5
P4
P3
P2
Sender
CWND$_{total}$
CWND$_1$
CWND$_2$
CWND$_3$
SF1
SF2
SF3
Receiver
SF1
SF2
SF3
P1

# FUSO

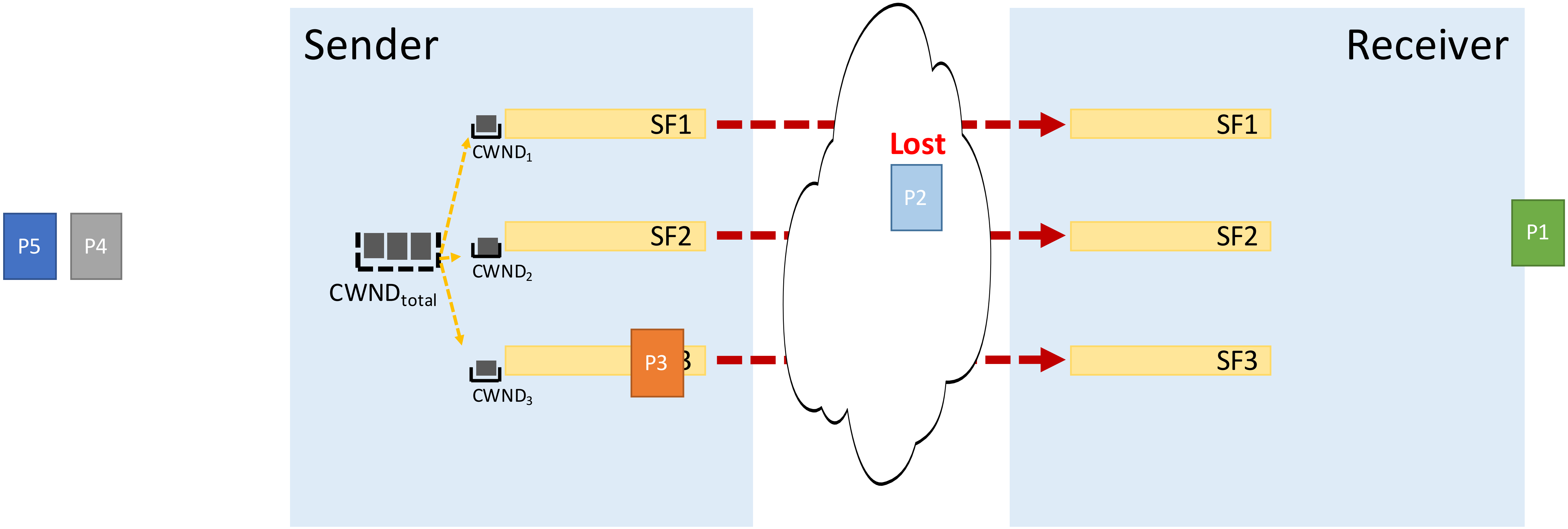

Sender
Receiver
P5
P4
P3
SF1
CWND$_1$
P2
2
CWND$_2$
CWND$_{total}$
SF3
CWND$_3$
SF1
SF2
SF3
P1

# FUSO



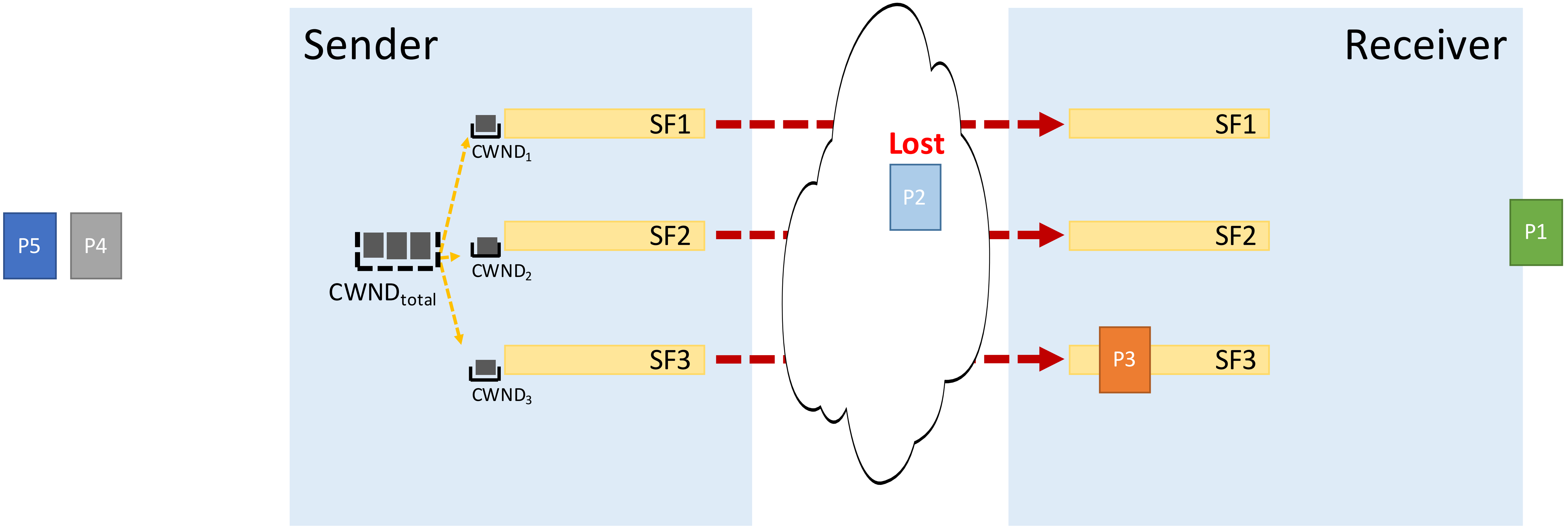
Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
Lost
P2
P1
P3
P4
P5
SF1
SF2
SF3

# FUSO

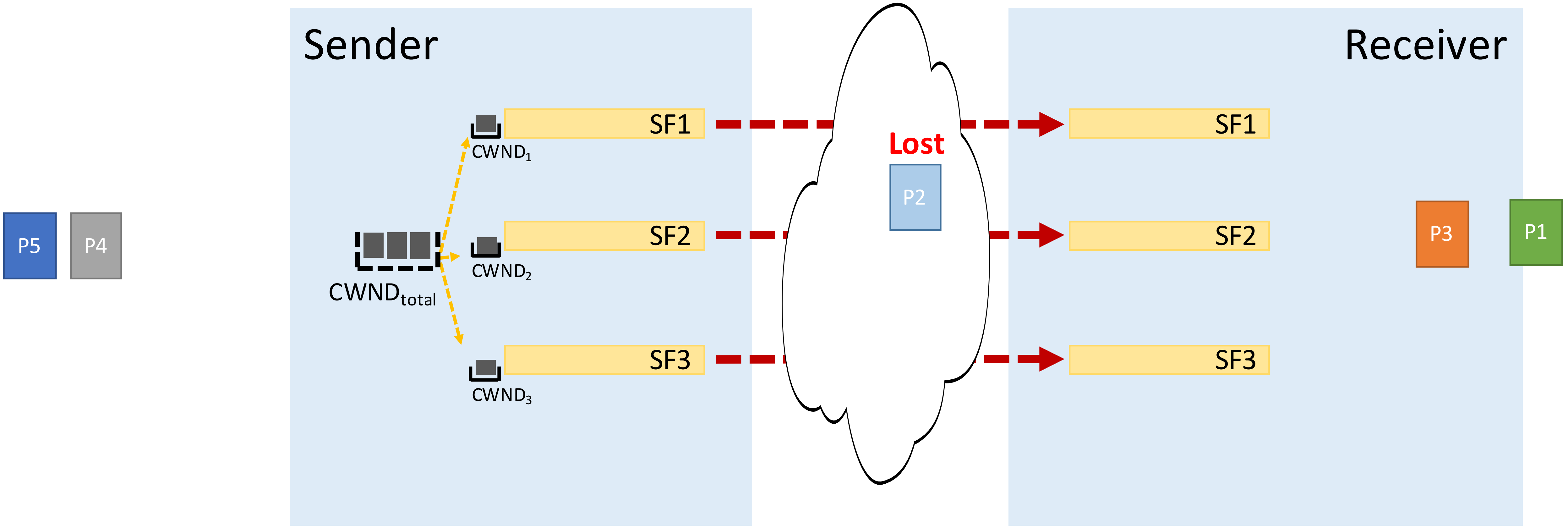


P5
P4
Sender
SF1
CWND$_1$
SF2
CWND$_2$
CWND$_{total}$
P3
CWND$_3$
Lost
P2
Receiver
SF1
SF2
SF3
P1

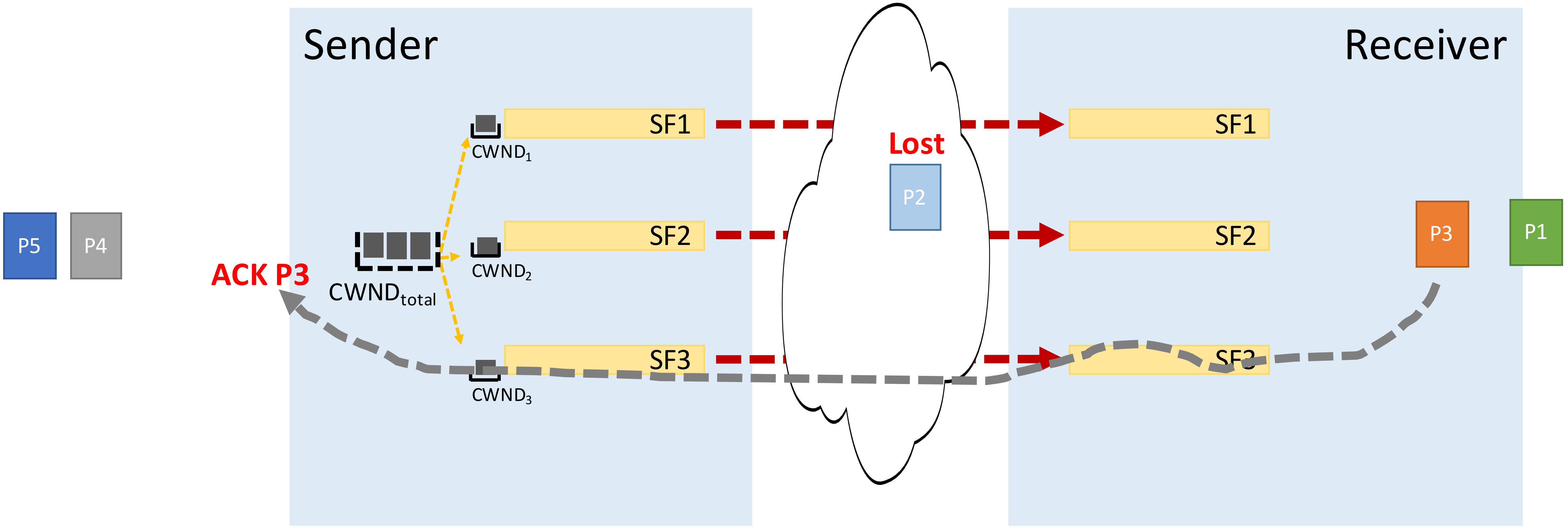
# FUSO


P5
P4
Sender
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
Lost
P2
Receiver
SF1
SF2
P3
SF3
P1

# FUSO

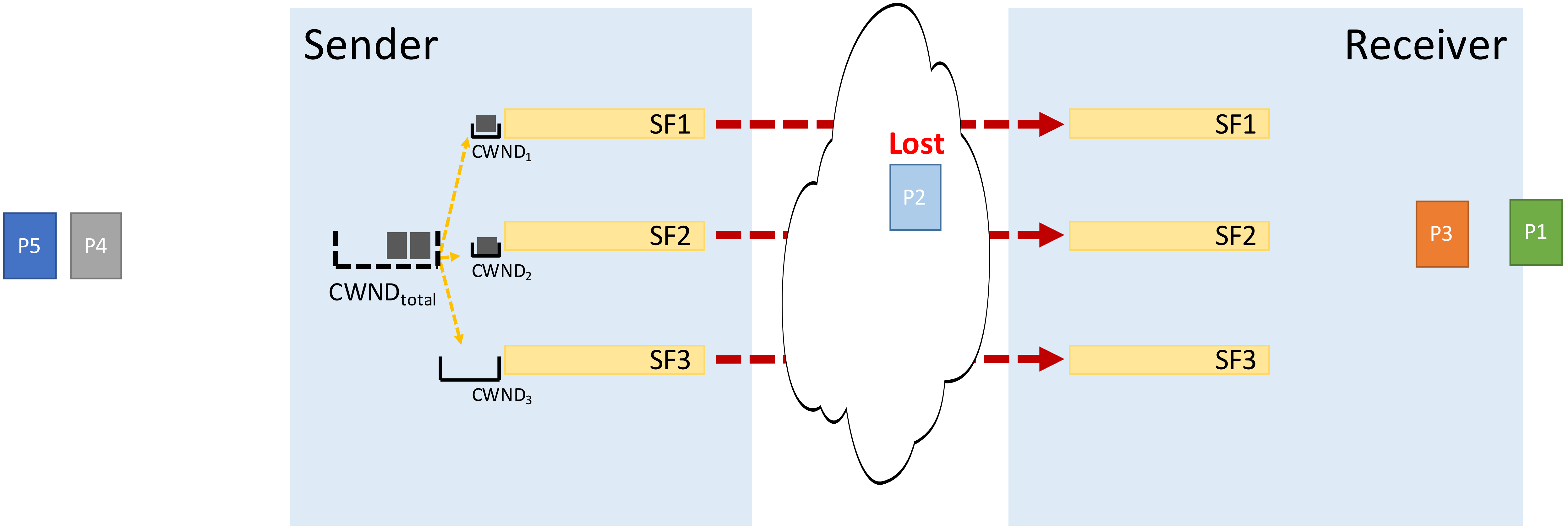

P5
P4
Sender
CWND_total
CWND1
CWND2
CWND3
SF1
SF2
SF3
Lost
P2
Receiver
SF1
SF2
SF3
P3
P1

# FUSO



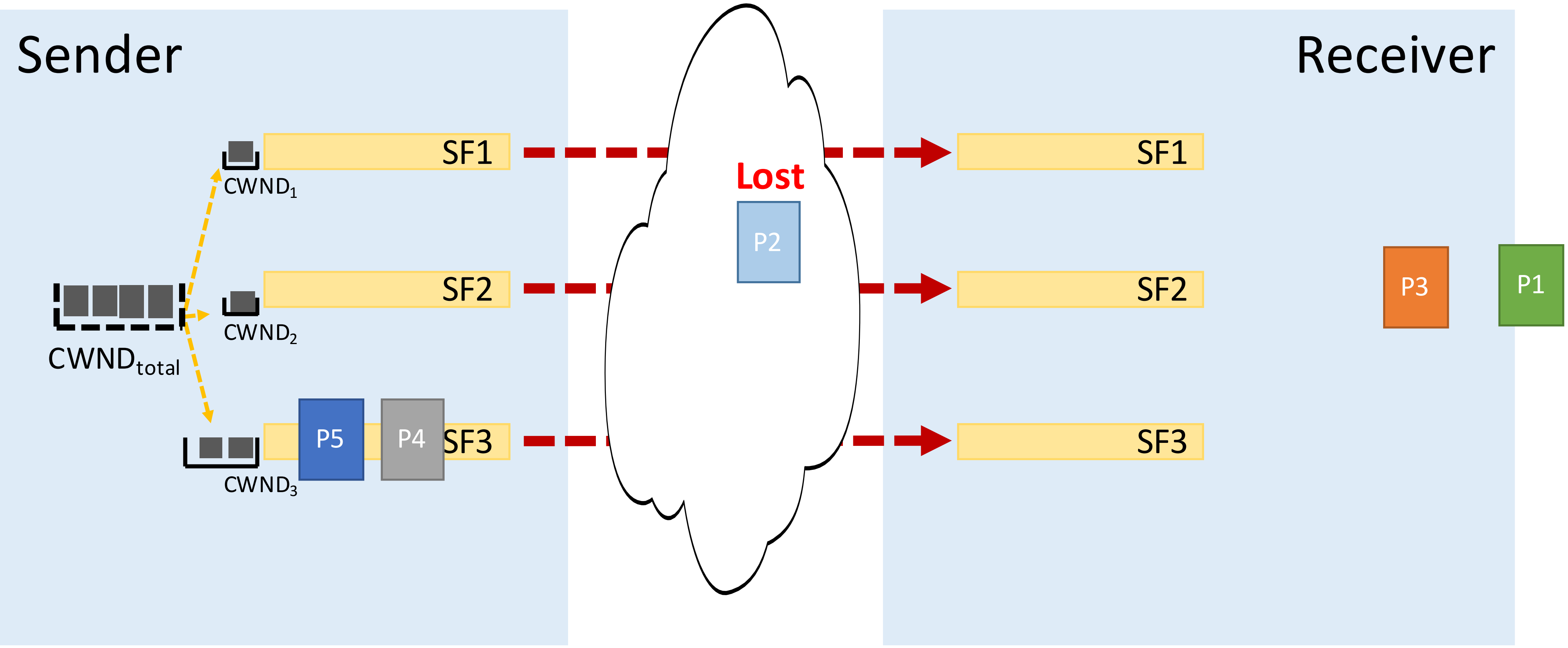
Sender
Receiver
SF1
SF2
SF3
SF1
SF2
CWND1
CWND2
CWND3
CWNDtotal
Lost
P2
P5
P4
ACK P3
P3
P1

# FUSO

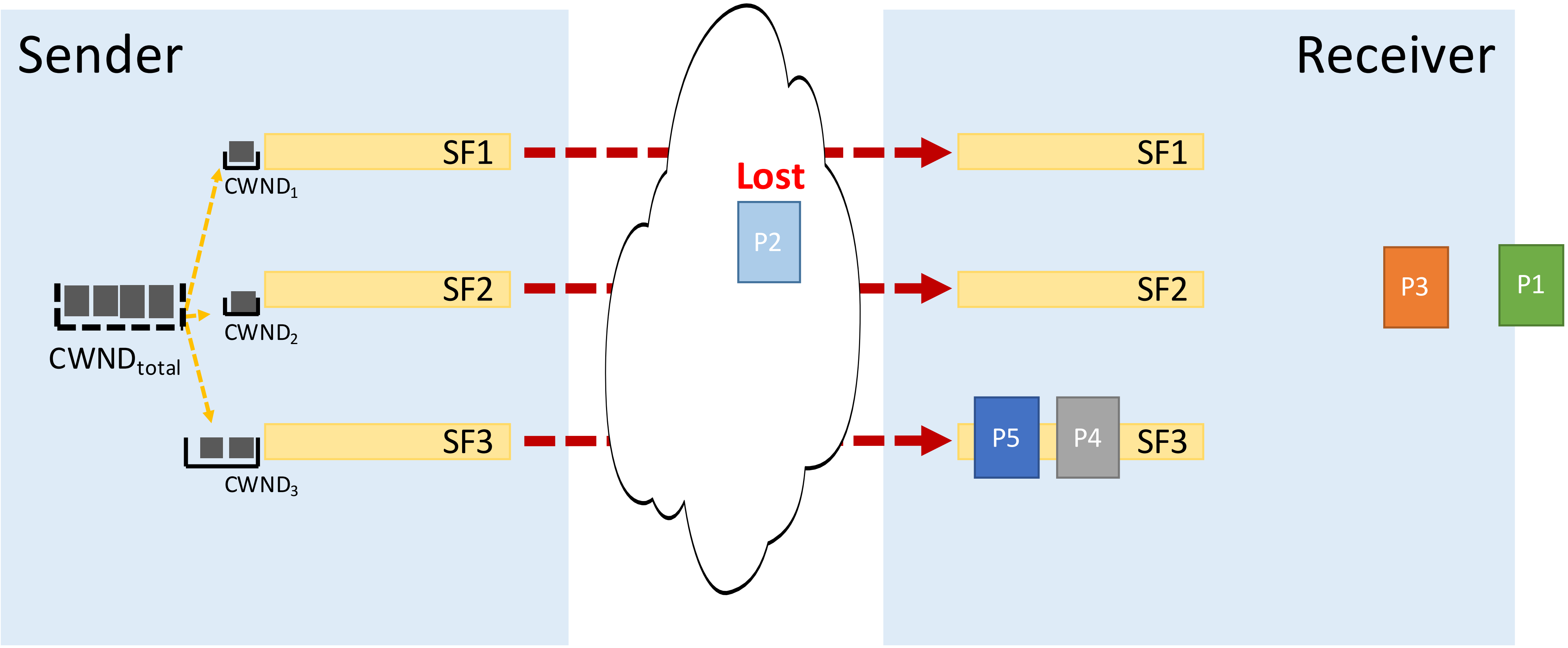


Sender
Receiver
P5
P4
CWND$_{total}$
CWND$_1$
CWND$_2$
CWND$_3$
SF1
SF2
SF3
Lost
P2
SF1
SF2
SF3
P3
P1

# FUSO

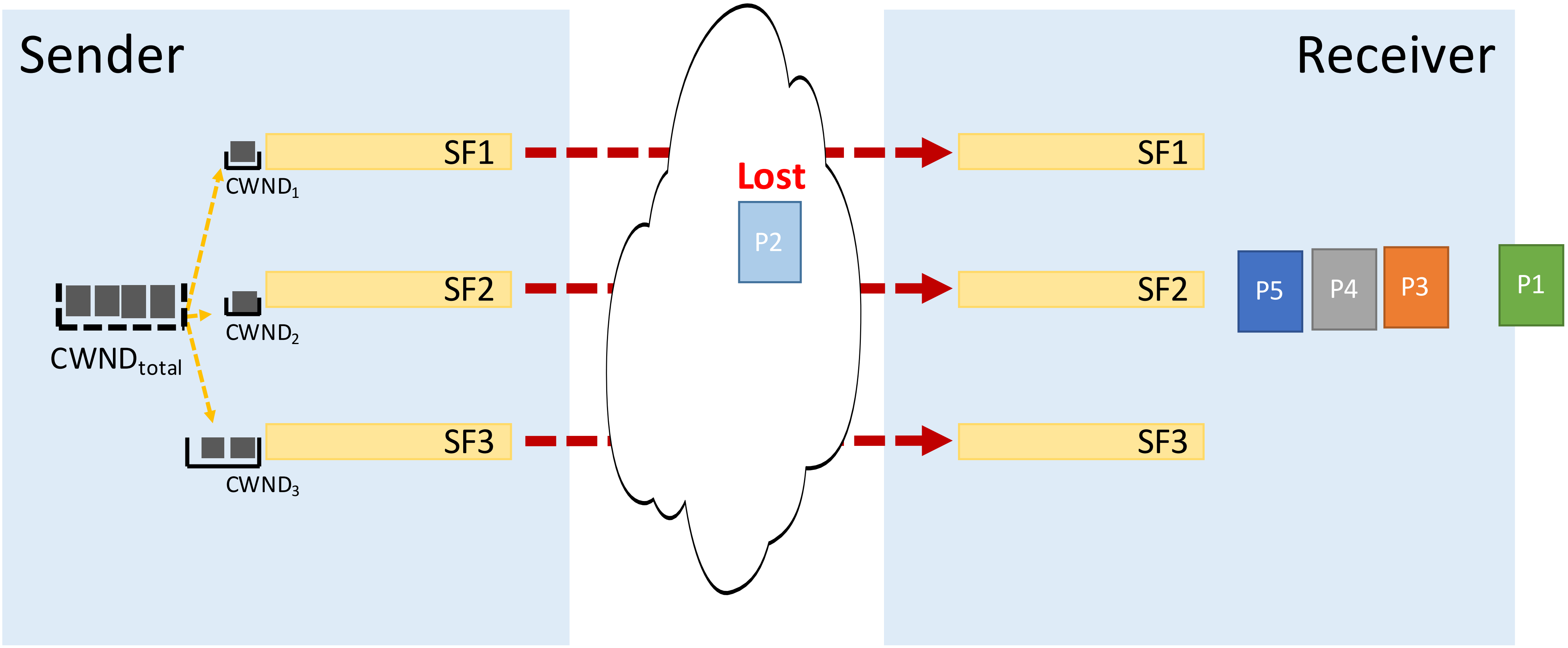

Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
P5
P4
Lost
P2
SF1
SF2
SF3
P3
P1

# FUSO



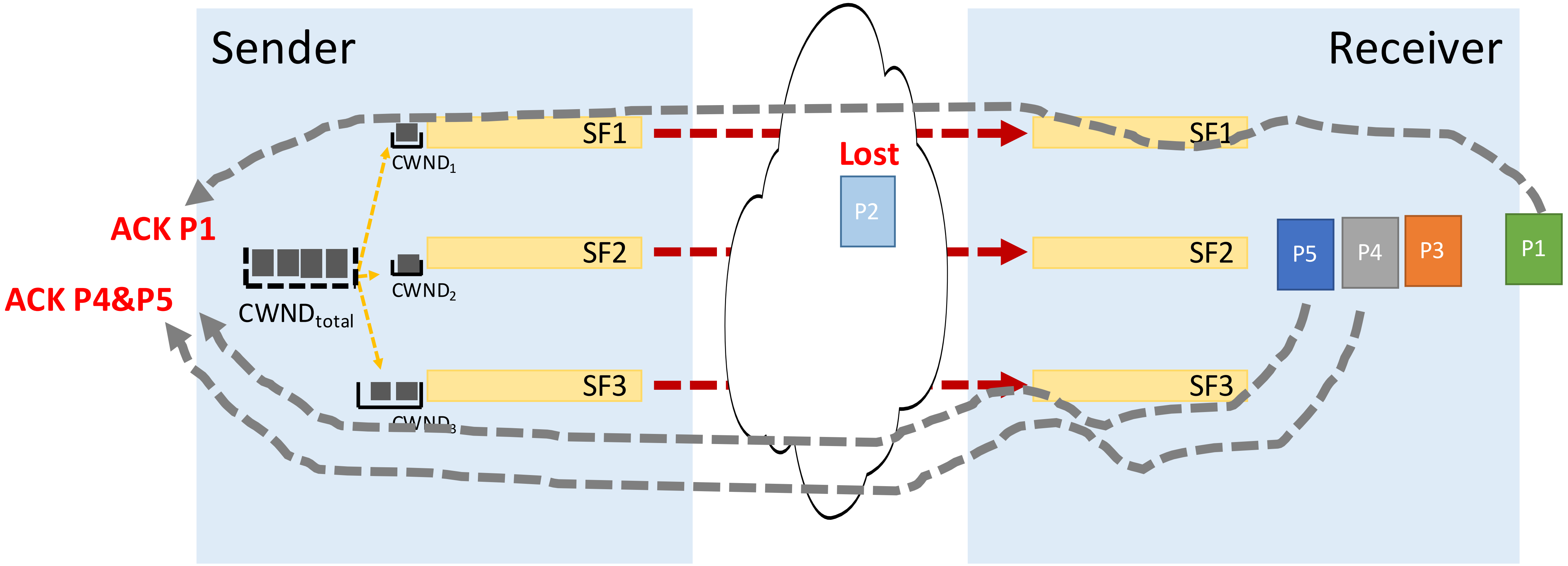
Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
Lost
P2
SF1
SF2
SF3
P3
P1
P5
P4

# FUSO

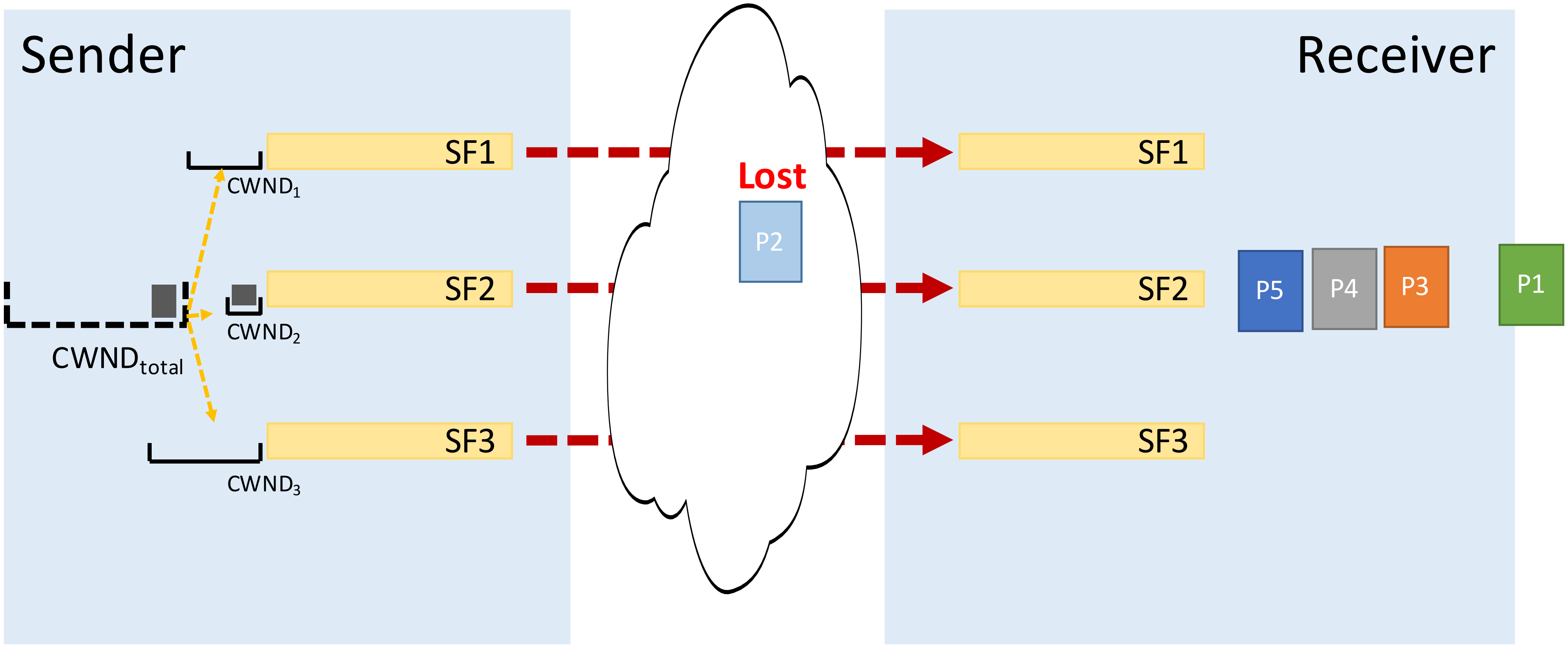


Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
Lost
P2
SF1
SF2
SF3
P5
P4
P3
P1

# FUSO

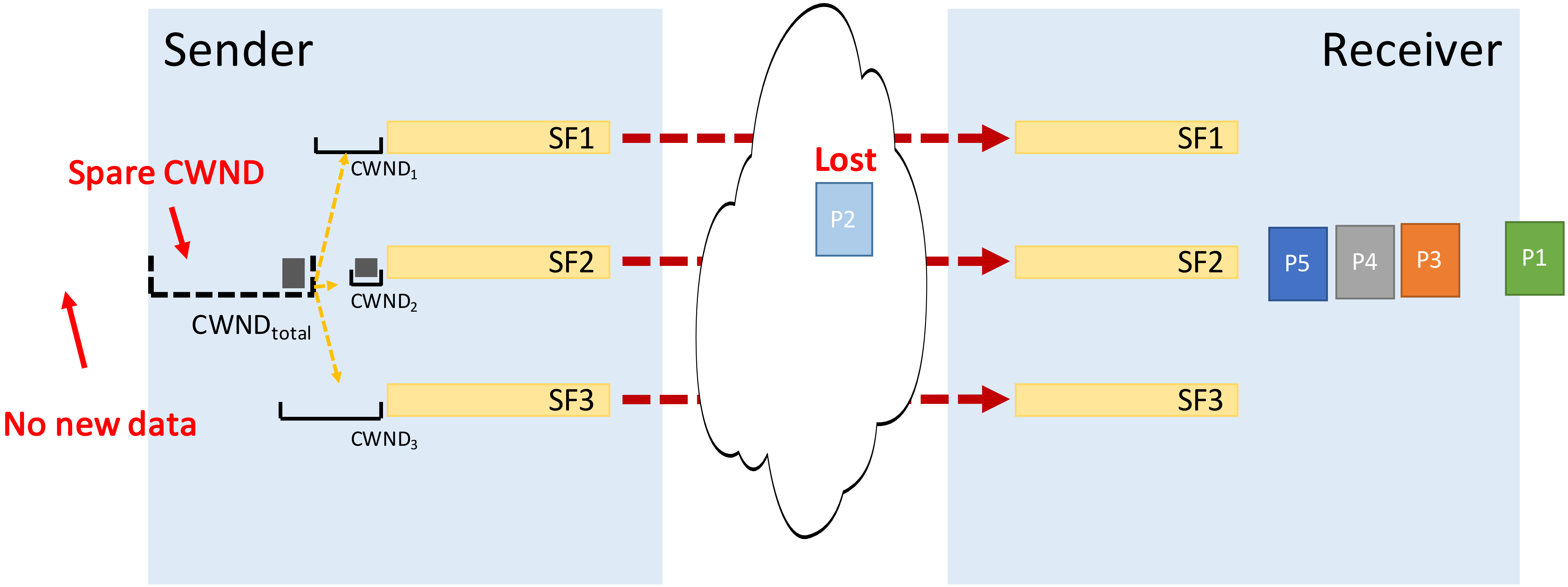

Sender
Receiver
SF1
SF2
SF3
SF1
SF2
SF3
CWND$_1$
CWND$_2$
CWND$_3$
CWND$_{total}$
Lost
P2
P5
P4
P3
P1
ACK P1
ACK P4&P5

# FUSO



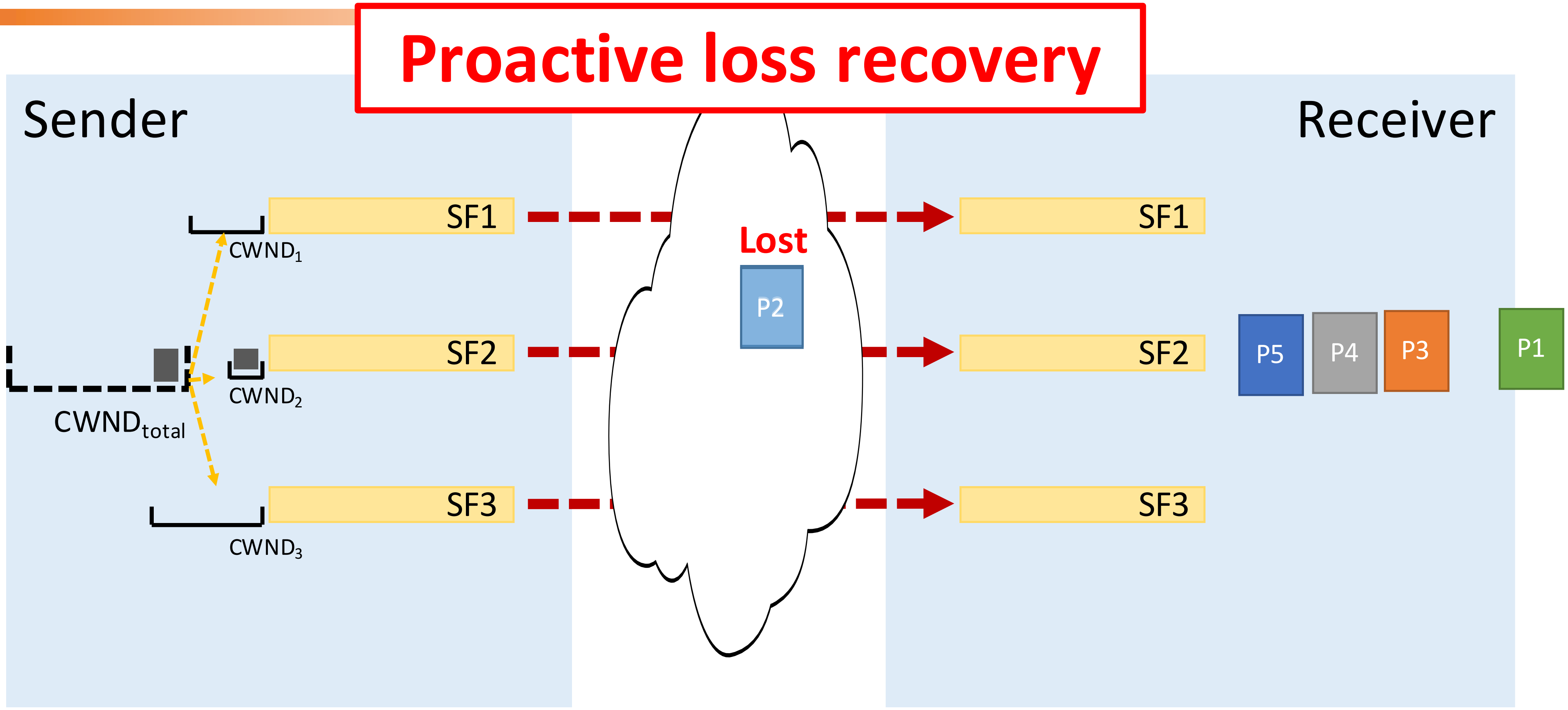
Sender
SF1
CWND1
SF2
CWND2
CWNDtotal
SF3
CWND3
Lost
P2
Receiver
SF1
SF2
P5
P4
P3
P1
SF3

# FUSO



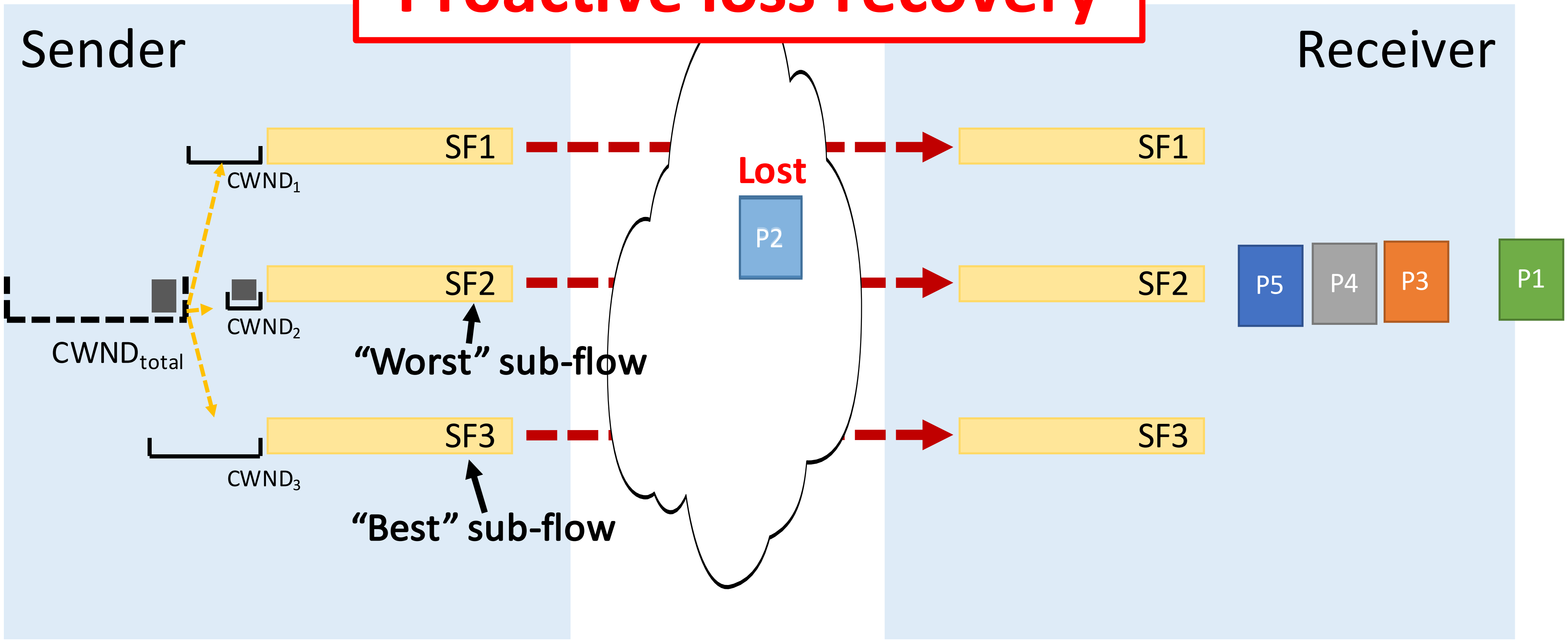
Sender
Receiver
Spare CWND
No new data
CWND1
CWND2
CWND3
CWNDtotal
SF1
SF2
SF3
Lost
P2
SF1
SF2
SF3
P5
P4
P3
P1

# FUSO

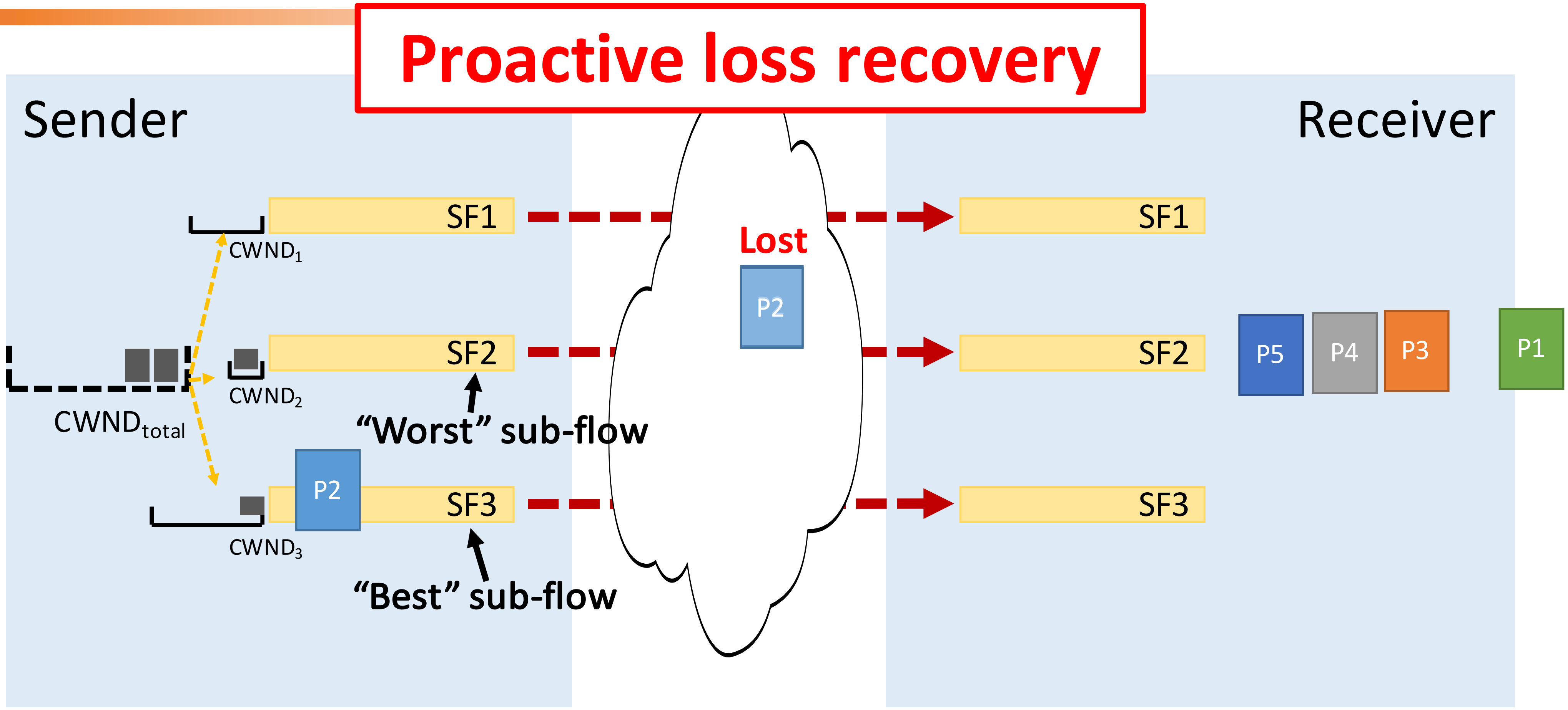

Proactive loss recovery
Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
Lost
P2
SF1
SF2
SF3
P5
P4
P3
P1

# FUSO

Proactive loss recovery

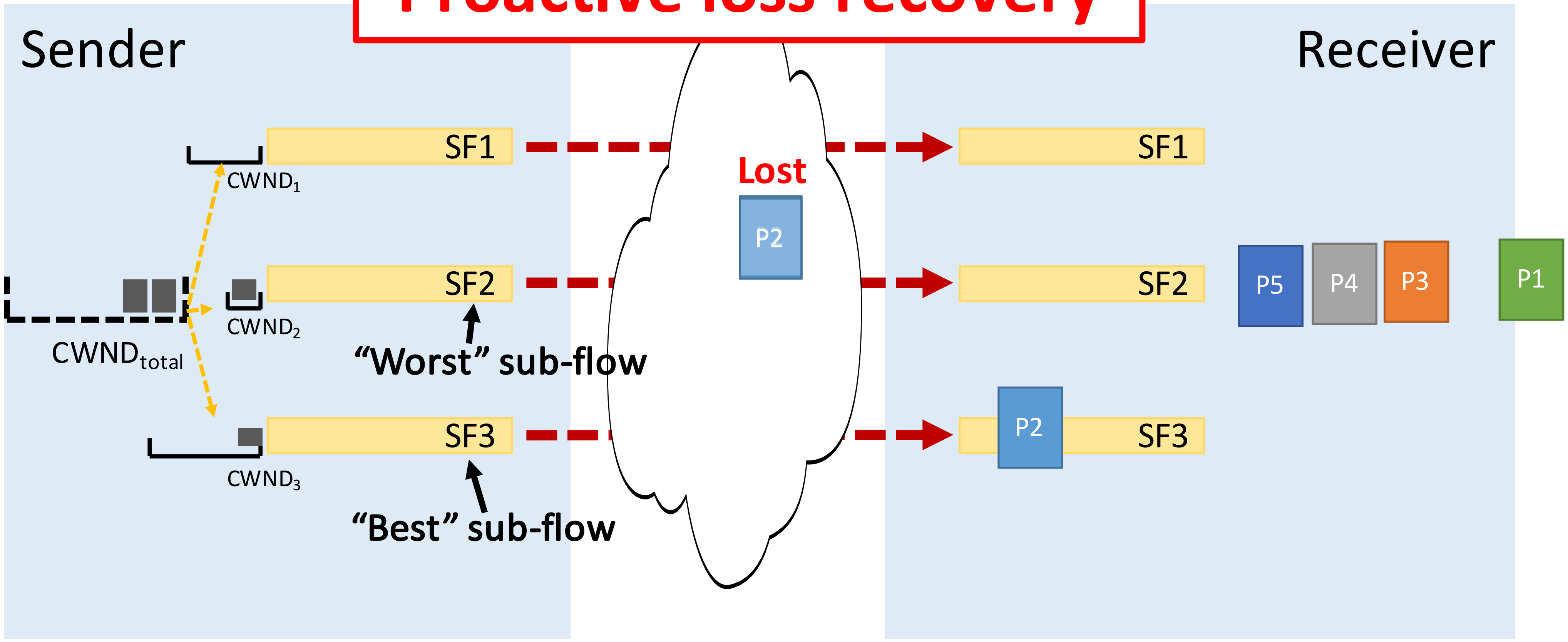


# FUSO

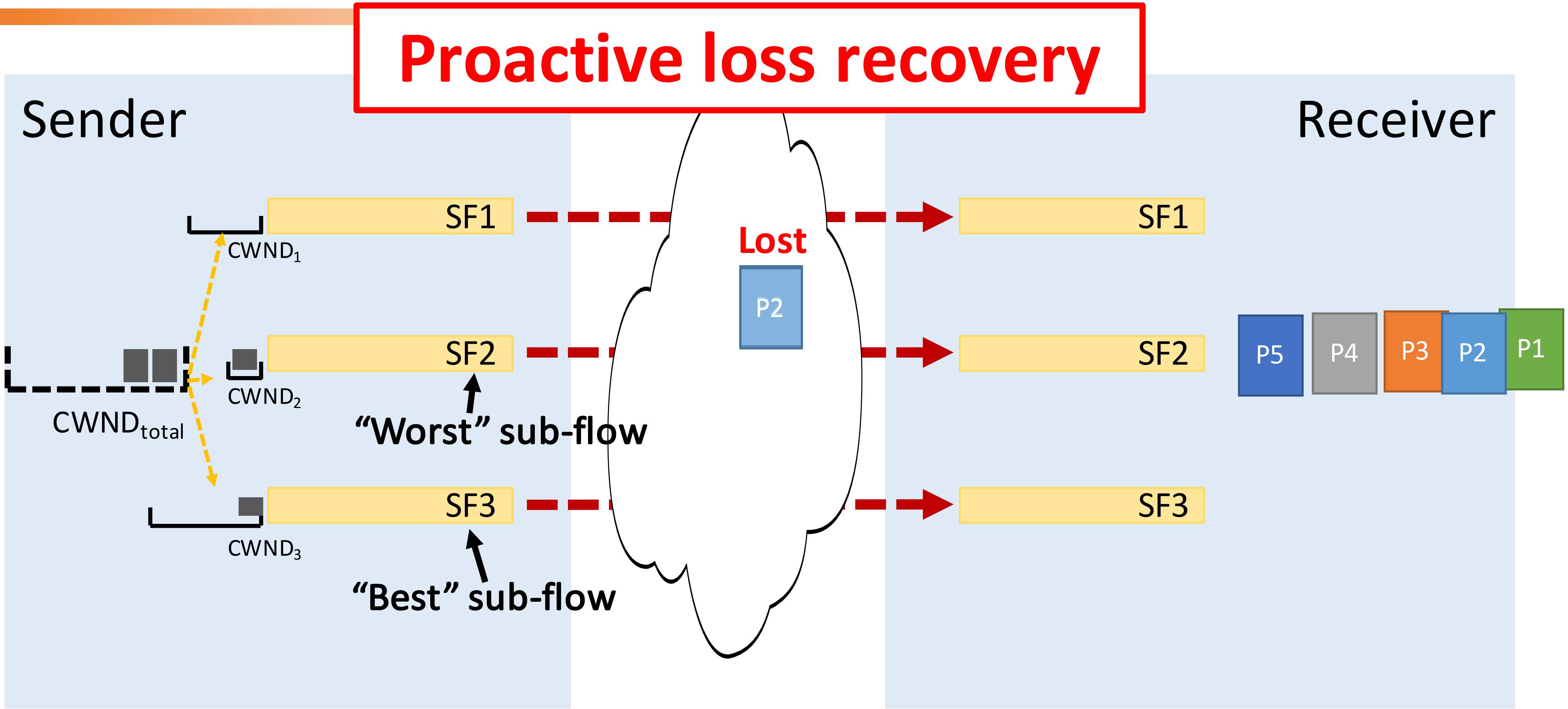

Proactive loss recovery
Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
"Worst" sub-flow
"Best" sub-flow
Lost
P2
P5
P4
P3
P1

# FUSO

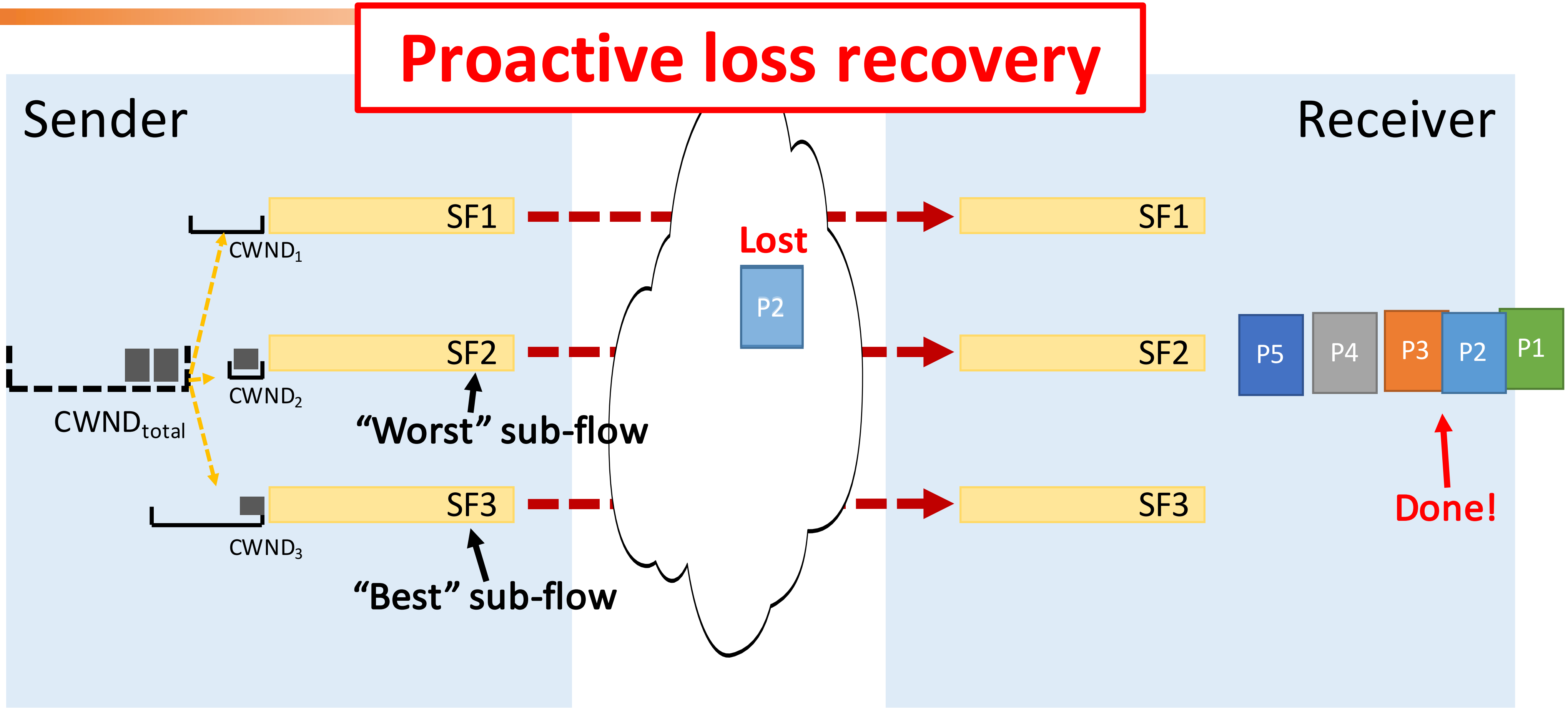
**Proactive loss recovery**



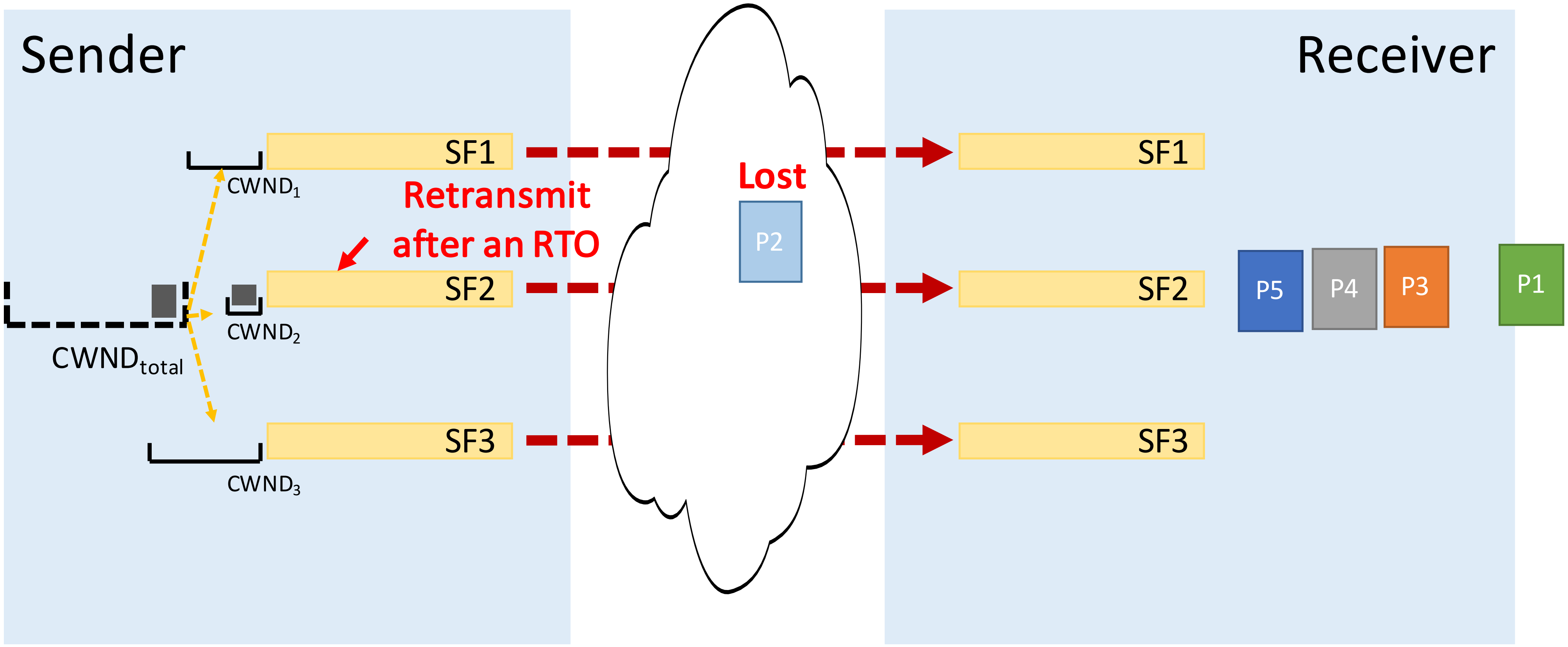
# FUSO


Proactive loss recovery
Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
"Worst" sub-flow
"Best" sub-flow
Lost
P2
SF1
SF2
SF3
P5
P4
P3
P2
P1

# FUSO


Proactive loss recovery
Sender
Receiver
SF1
SF2
SF3
CWND1
CWND2
CWND3
CWNDtotal
"Worst" sub-flow
"Best" sub-flow
Lost
P2
SF1
SF2
SF3
P5
P4
P3
P2
P1
Done!

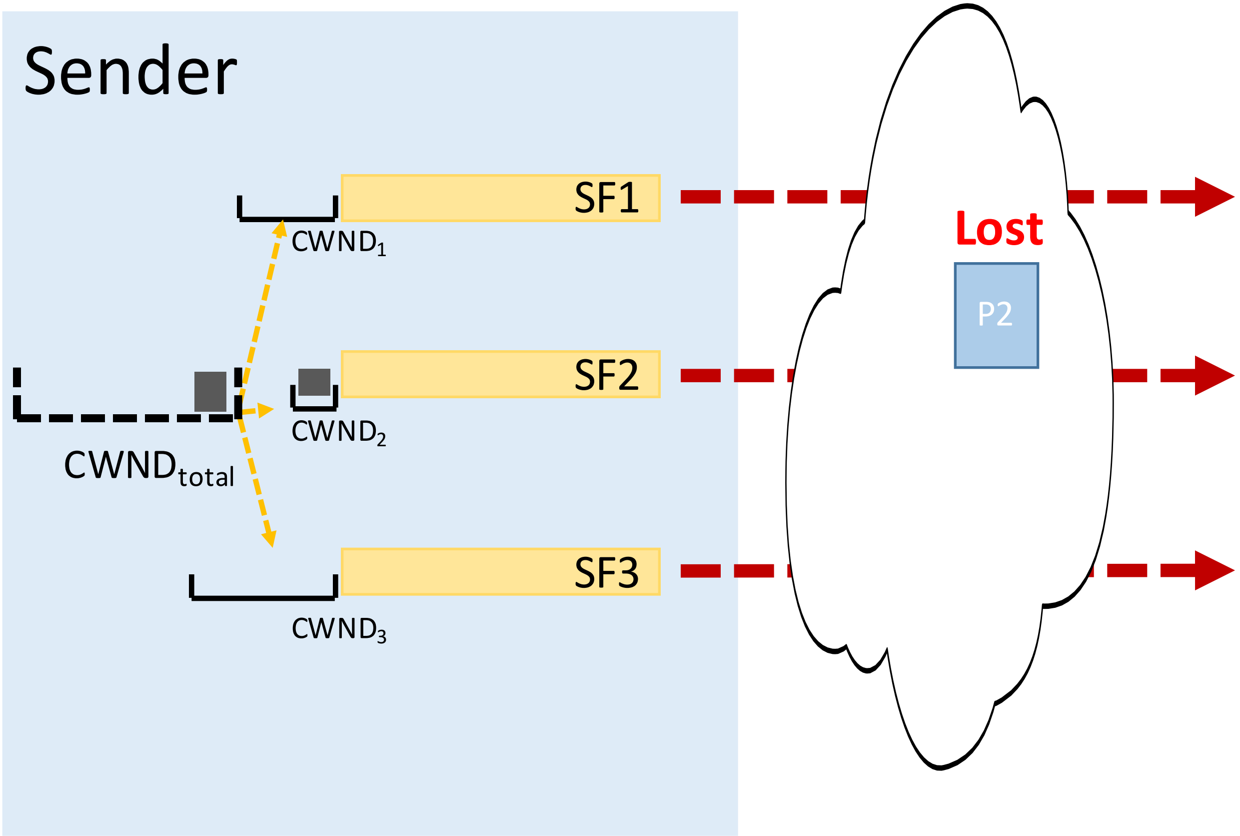
# Standard MPTCP



# FUSO: Path Selection

Possibility of encountering loss

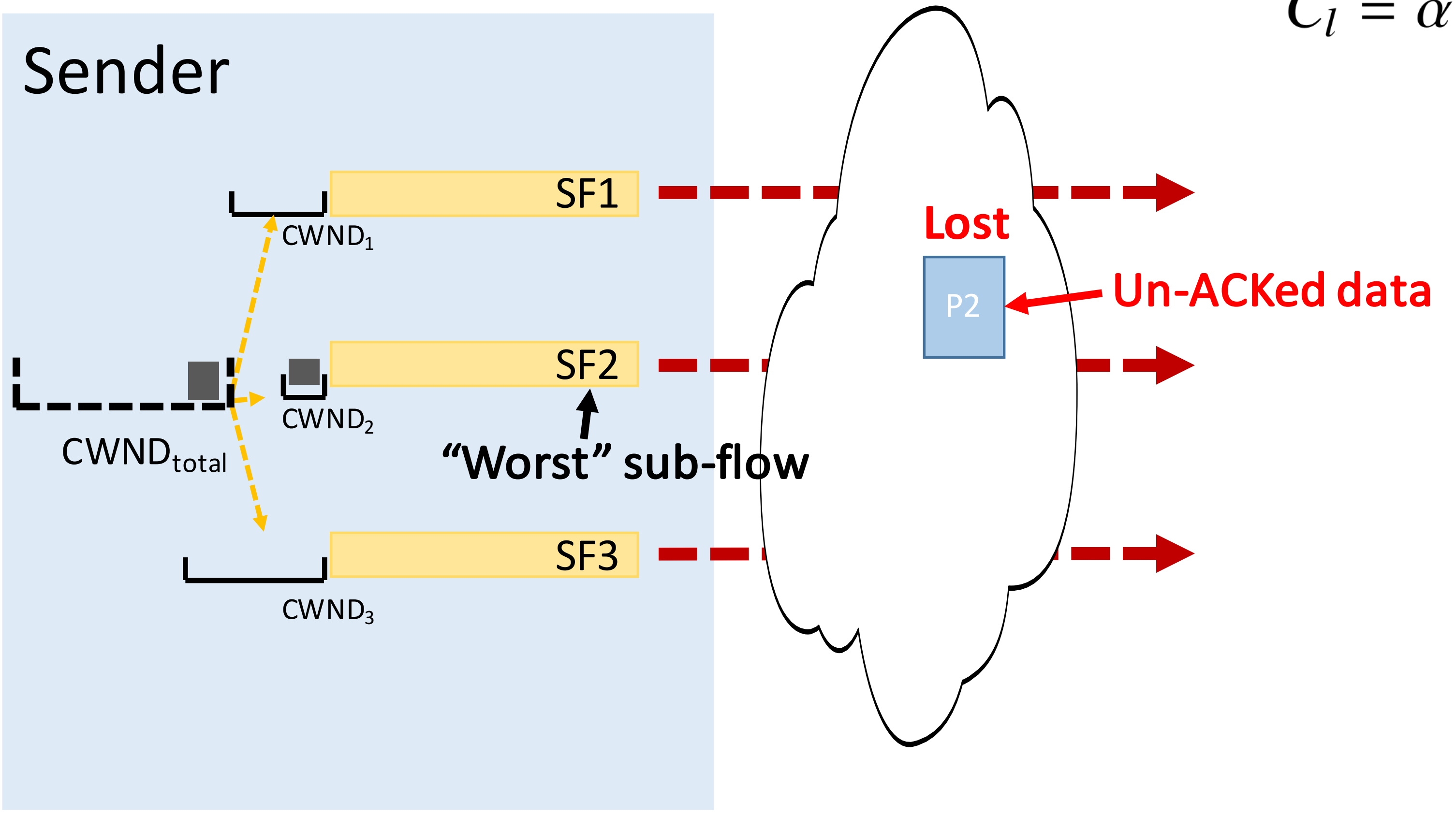
$$C_l = \alpha \cdot \overline{lossrate} + \beta \cdot lossrate_{last}$$



# FUSO: Path Selection

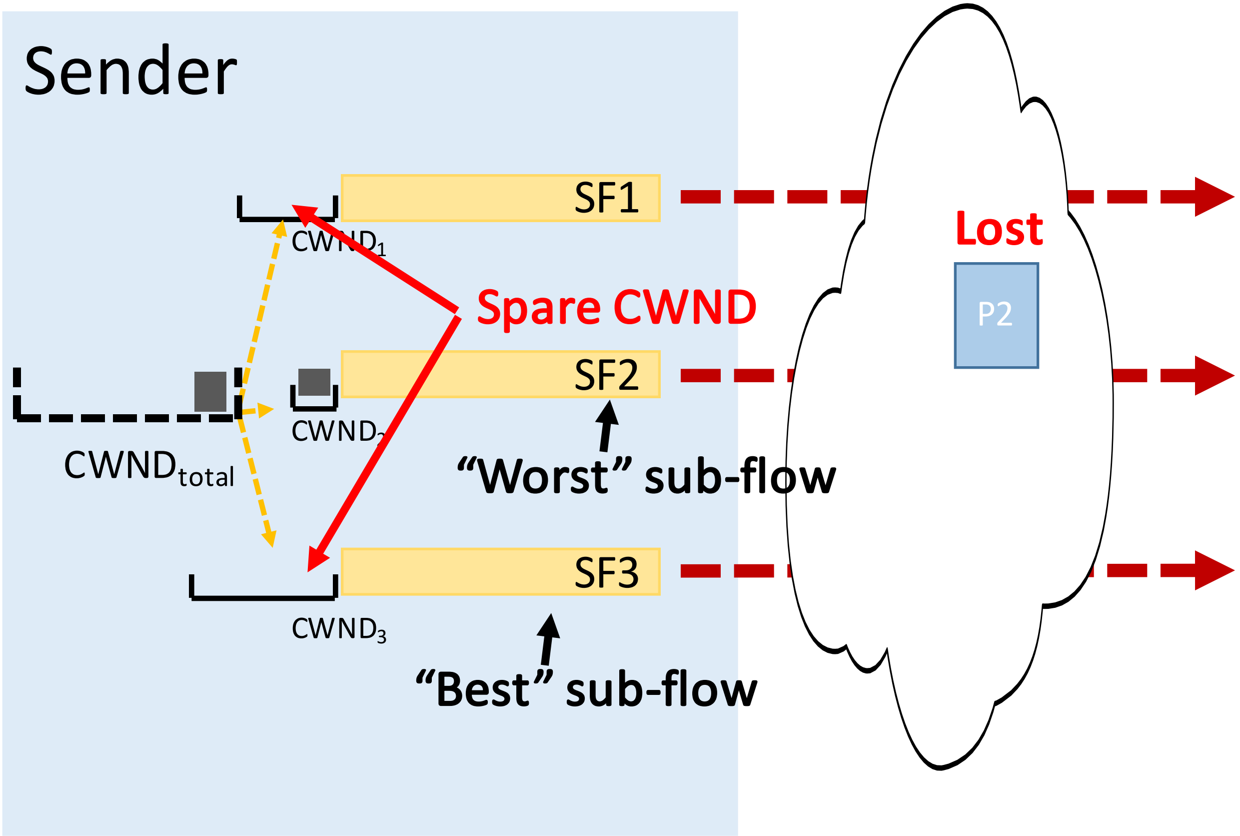
Possibility of encountering loss

$$C_l = \alpha \cdot \overline{lossrate} + \beta \cdot lossrate_{last}$$




- **"Worst"** Sub-flow
  - With un-ACKed data
  - Most likely having loss

# FUSO: Path Selection

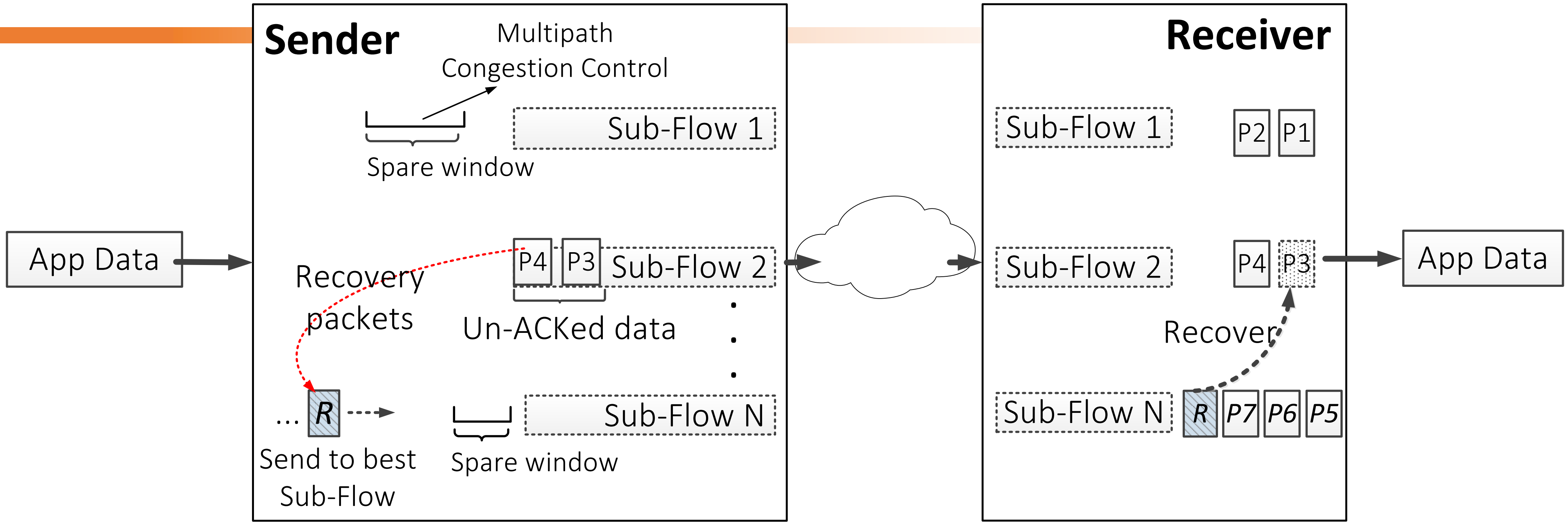



Possibility of encountering loss

$$C_l = \alpha \cdot \overline{lossrate} + \beta \cdot lossrate_{last}$$

- **"Worst"** Sub-flow
  - With un-ACKed data
  - Most likely having loss
- **"Best"** Sub-flow
  - With spare CWND
  - Least likely having loss

# FUSO in 1 Slide




- If (spare CWND) && (no new data)
  - Utilize the transmission opportunity to proactively recover
  - Use “good” paths to help “bad” paths
- Multi-path diversity offers many transmission opportunities
  - “Good” paths have spare window

# FUSO Implementation

https://github.com/1989chenguo/FUSO

- Implemented in Linux kernel; ~900 lines of code

**function** TRY_SEND_RECOVERIES( )
    **while** $BytesInFlight_{Total} < CWND_{Total}$ **and** no new data **do**
        return ← SEND_A_RECOVERY( )
        **if** return == NOT_SEND **then**
            break

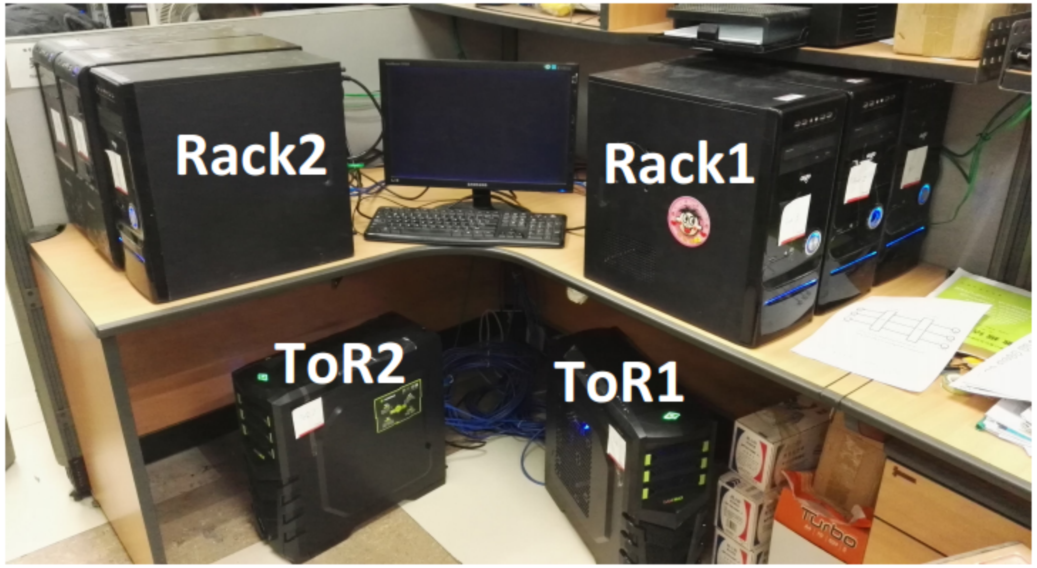
**function** SEND_A_RECOVERY( )
    FIND_WORST_SUB-FLOW( )
    FIND_BEST_SUB-FLOW( )
    **if** no worst found *or* no best sub-flow found **then**
        return NOT_SEND
    recovery_packet←one un-ACKed packet of the worst sub-flow
    Send the recovery_packet through the best sub-flow
    $BytesInFlight_{Total}$ += $Size_{recovery\_packet}$

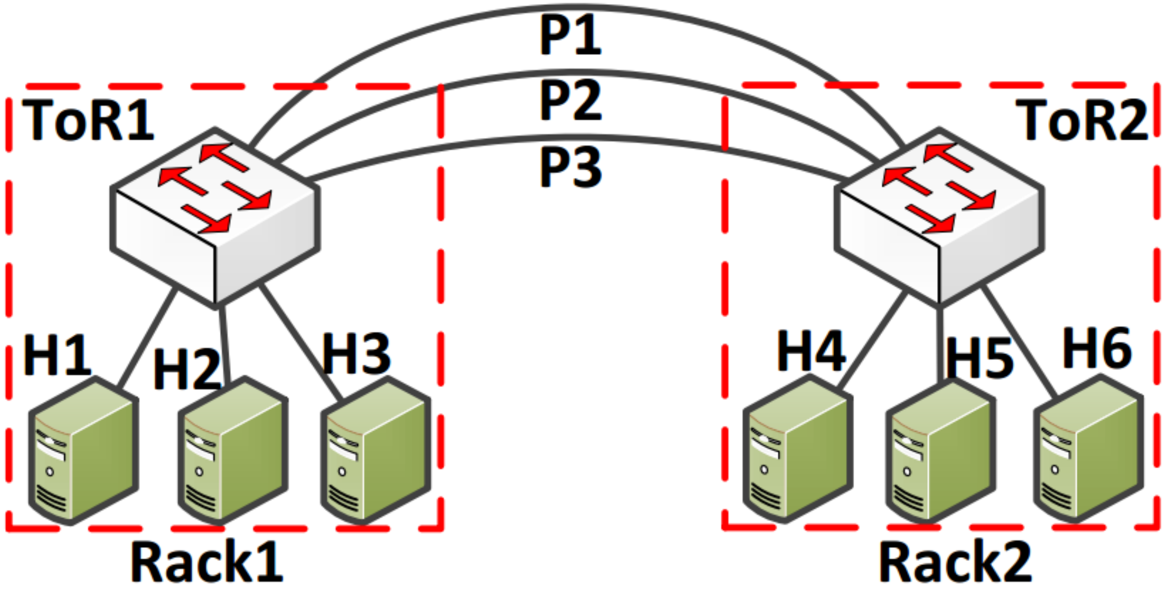
# Outline

- Motivation
- Packet Loss in DCN
- Impact of Packet Loss
- Challenge for Loss Recovery
- FUSO Design
- **Evaluation**
- Summary

# Testbed Settings

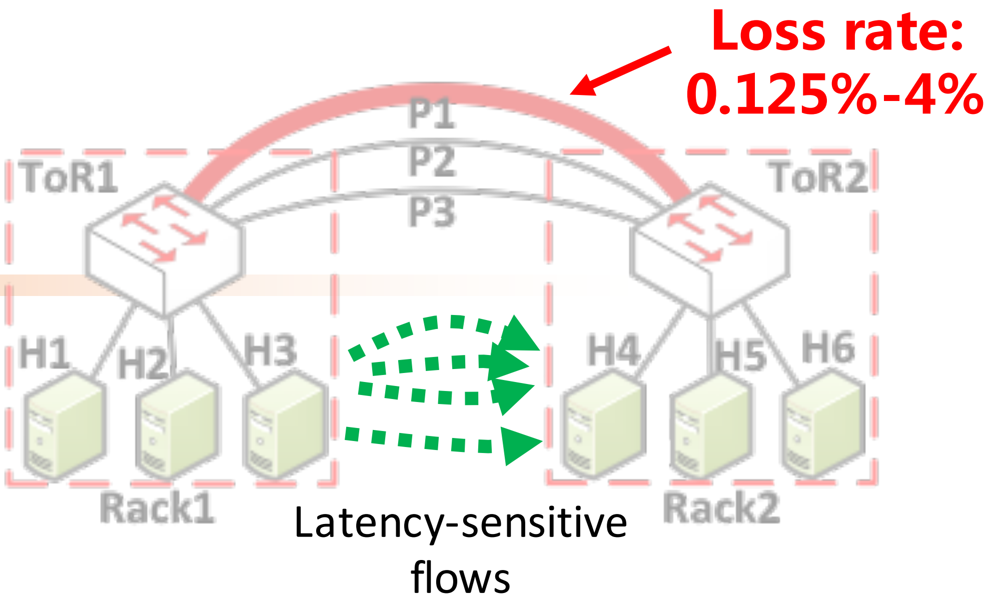
- Network
  - 1Gbps fabric & 1Gbps hosts; ECMP routing; ECN enabled
- TCP
  - Init_cwnd=16; min_RTO=5ms



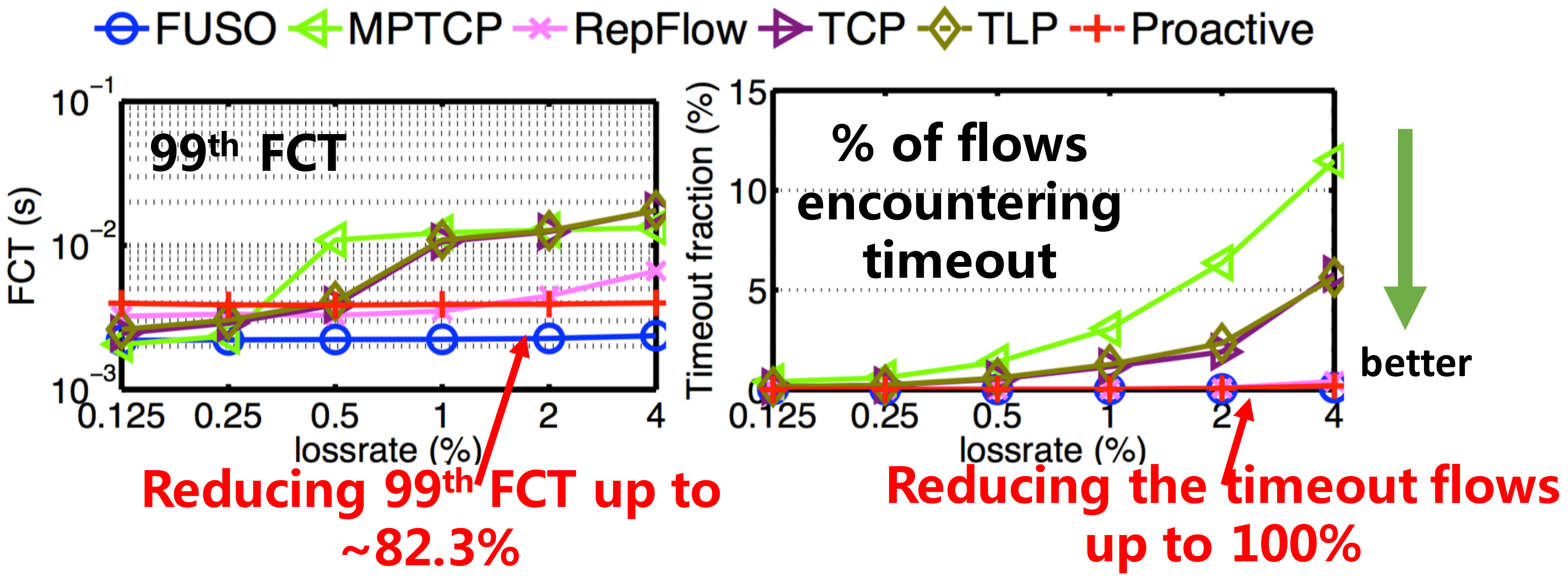



# Testbed Results

- Failure loss
  - Random-drop

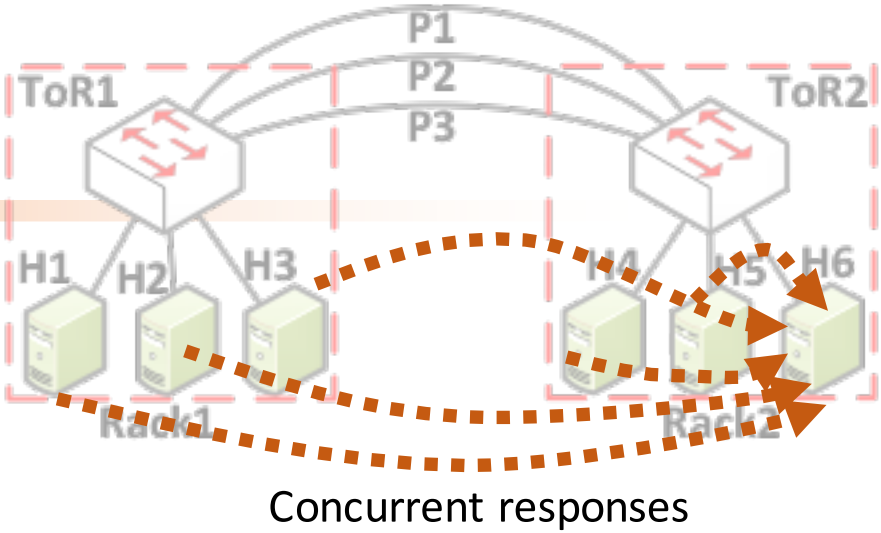



**Fast**

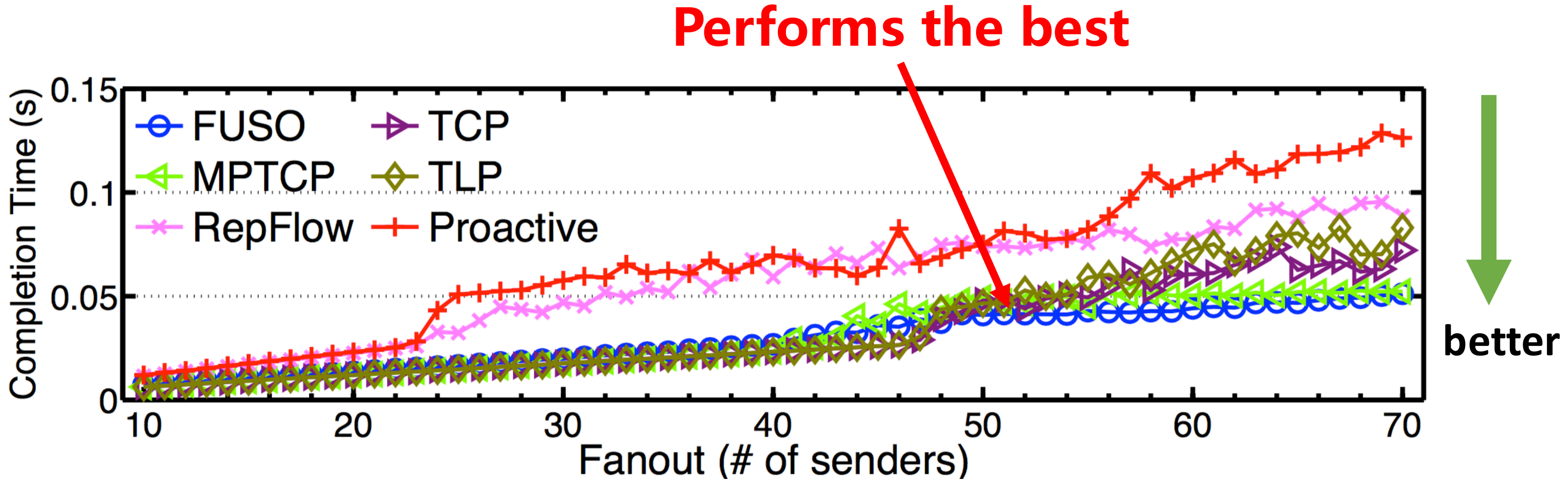


# Testbed Results

- Congestion loss
  - Incast

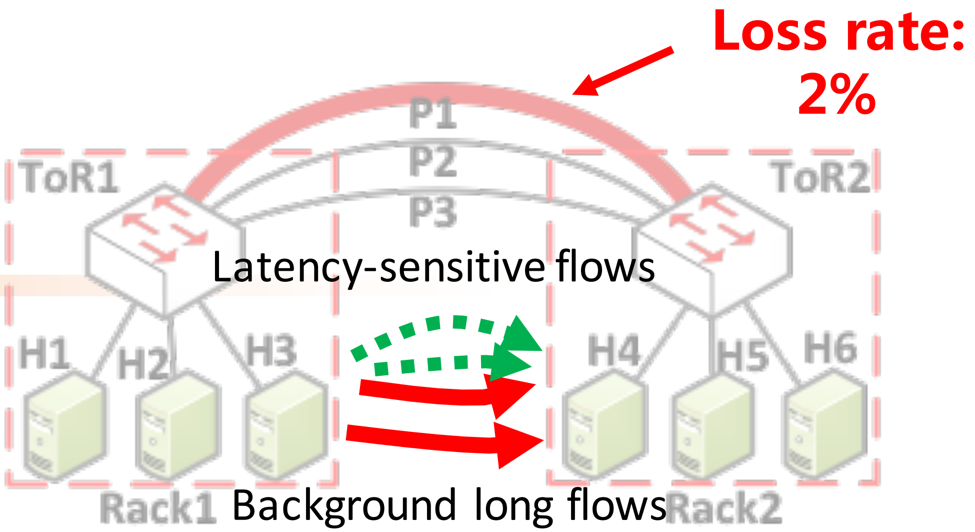


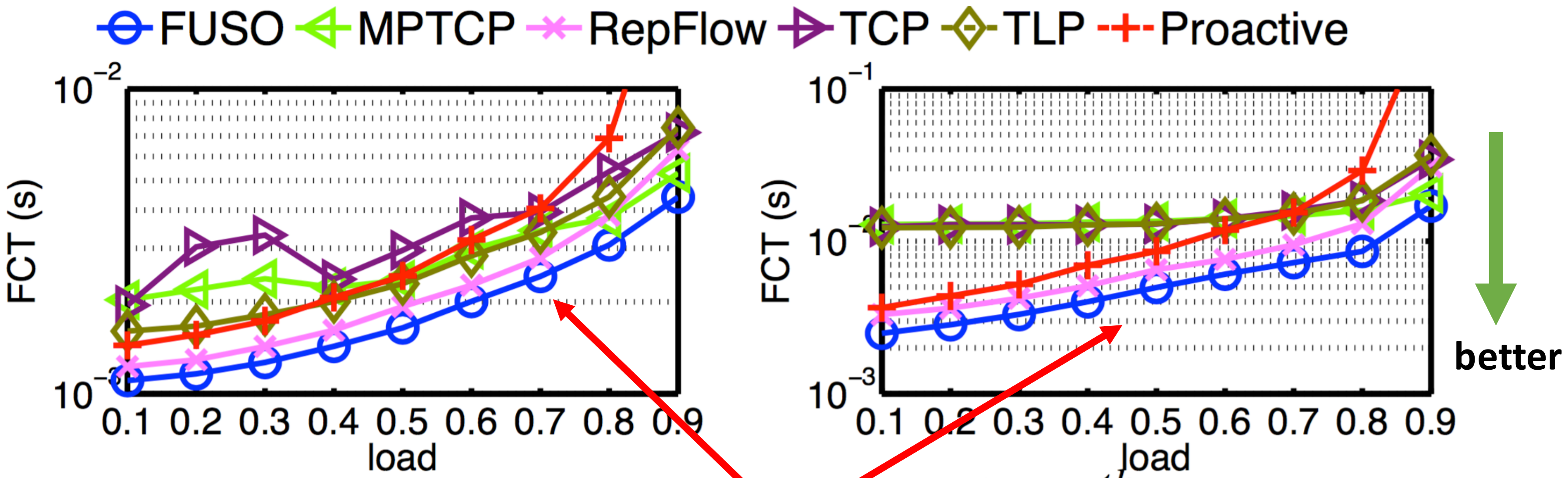

**Cautious**



# Testbed Results

- Failure loss & Congestion loss
  - From failure-loss-dominated to congestion-loss-dominated



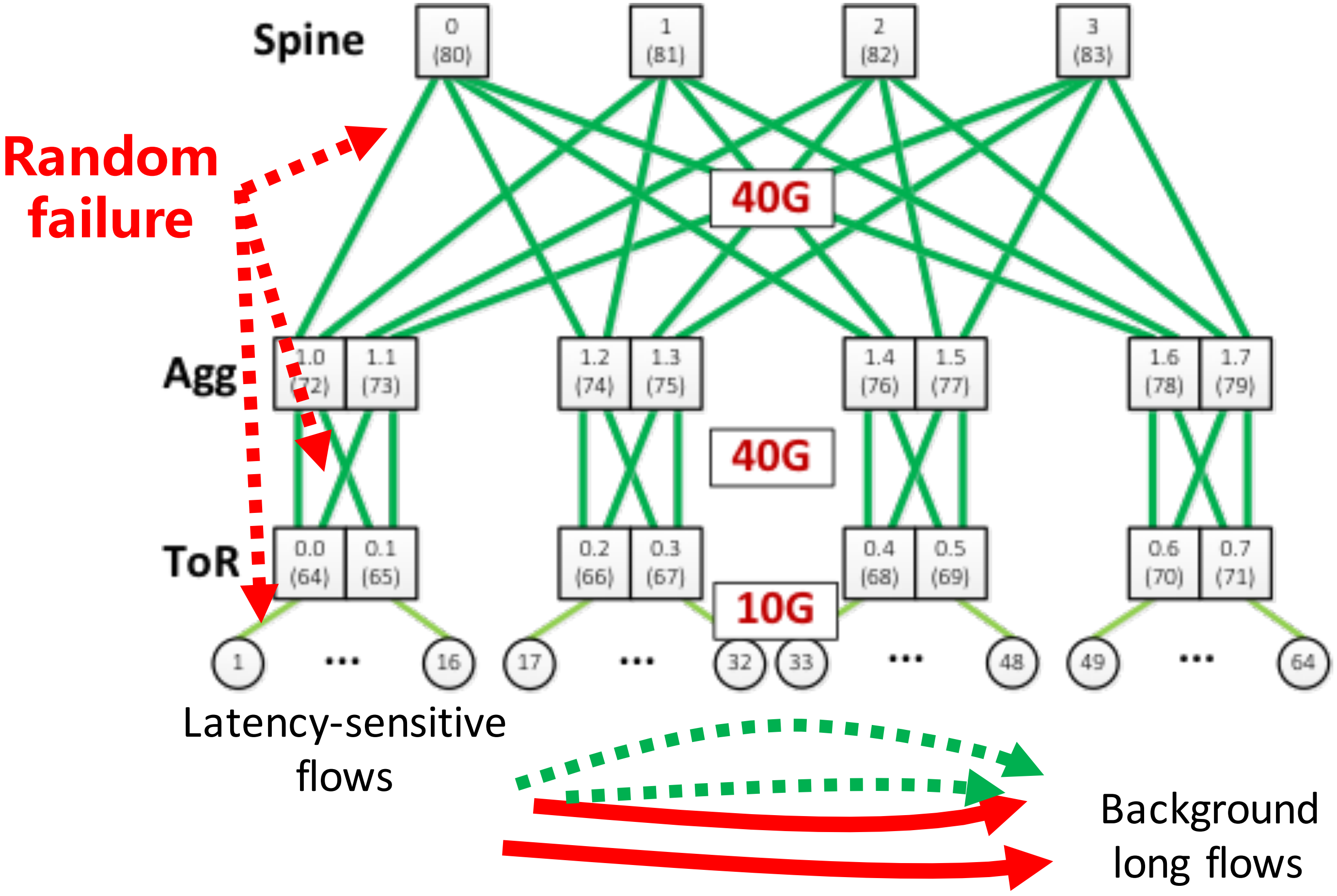




(a) Avg FCT (b) $99^{th}$ FCT

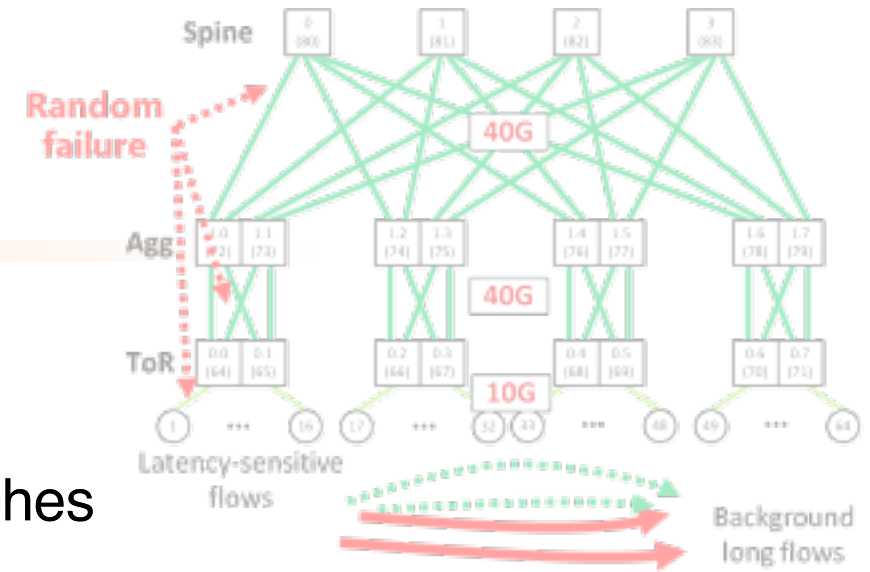
**Adapt to various loss condition**

# Larger-scale Simulations

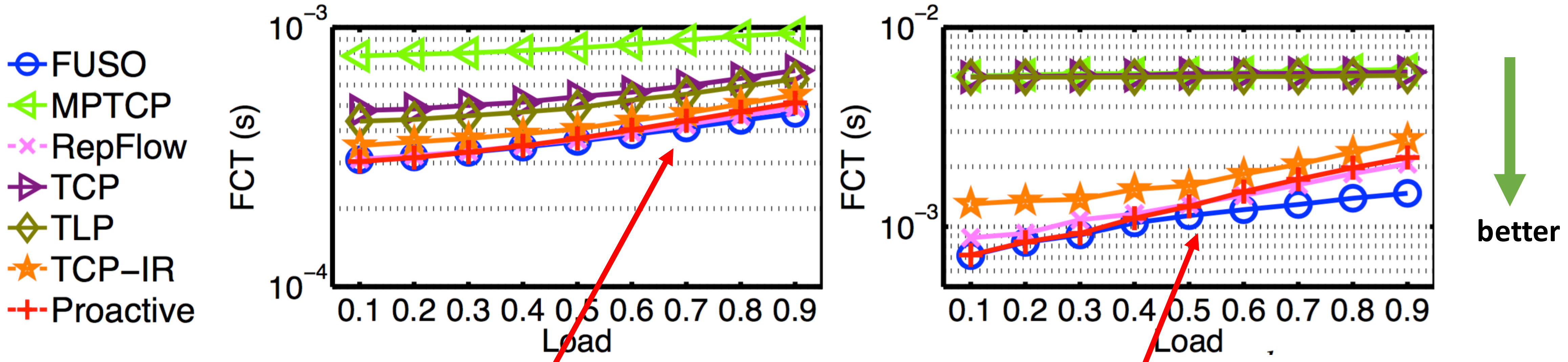
- Simulation settings
  - NS2 simulator; 3-layer, 4-port FatTree
  - 40Gbps fabric, 10Gbps host; 64 hosts, 20 switches
  - Empirical failure generation



# Larger-scale Simulations




- Simulation settings
  - NS2 simulator; 3-layer, 4-port FatTree fabric
  - 40Gbps fabric, 10Gbps host; 64 hosts, 20 switches
  - Empirical failure generation




**Reducing the average FCT up to ~60.3%**

**Reducing the 99th FCT up to ~87.4%**

# Outline

- Motivation
- Packet Loss in DCN
- Impact of Packet Loss
- Challenge for Loss Recovery
- FUSO Design
- Evaluation
- **Summary**

# Summary

- Loss hurts tail latency
  - Loss is not uncommon
  - A little loss leads to enough timeout, hurting the tail
- Challenges for loss recovery
  - How to accelerate loss recovery under various loss conditions without causing congestion?
- Philosophy for FUSO
  - To be fast & cautious are equally important
  - **Fast**: Proactive loss recovery utilizing spare transmission opportunity, leveraging multipath diversity
  - **Cautious**: Strictly follows congestion control without adding aggressiveness

# Thanks

Q&A?