

【Full Understanding】

TCP Algorithm

Basic edition

《Loss base method》

- Mechanism for reliable data delivery
- Efficient transmission by grasping
network congestion

Overall table of contents

1. Overview of TCP

2 . TCP Algorithm

3 . Summary

1. TCP overview

1-1. What are RFCs?

1-2. What is TCP/IP?

1-3. Protocol stack

1-4. Packet data (capture)

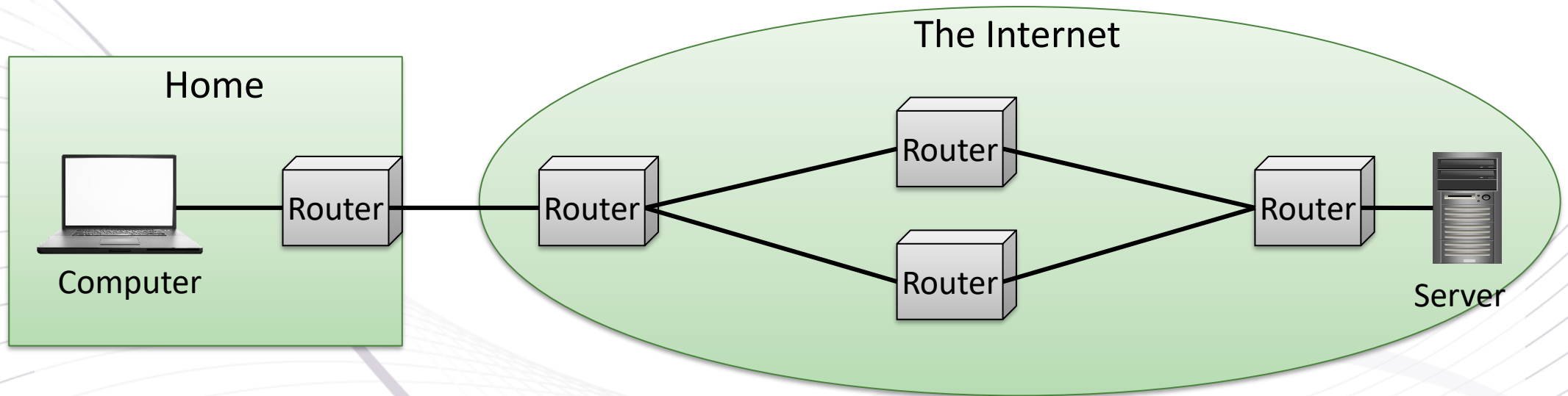
1-5. TCP version

1-6. TCP data format

1-1. What are RFCs?

All devices involved in Internet communication must handle data using pre-defined technology.

Therefore, standardization (arrangement) of Internet technology is being carried out. This standardization is called RFC.

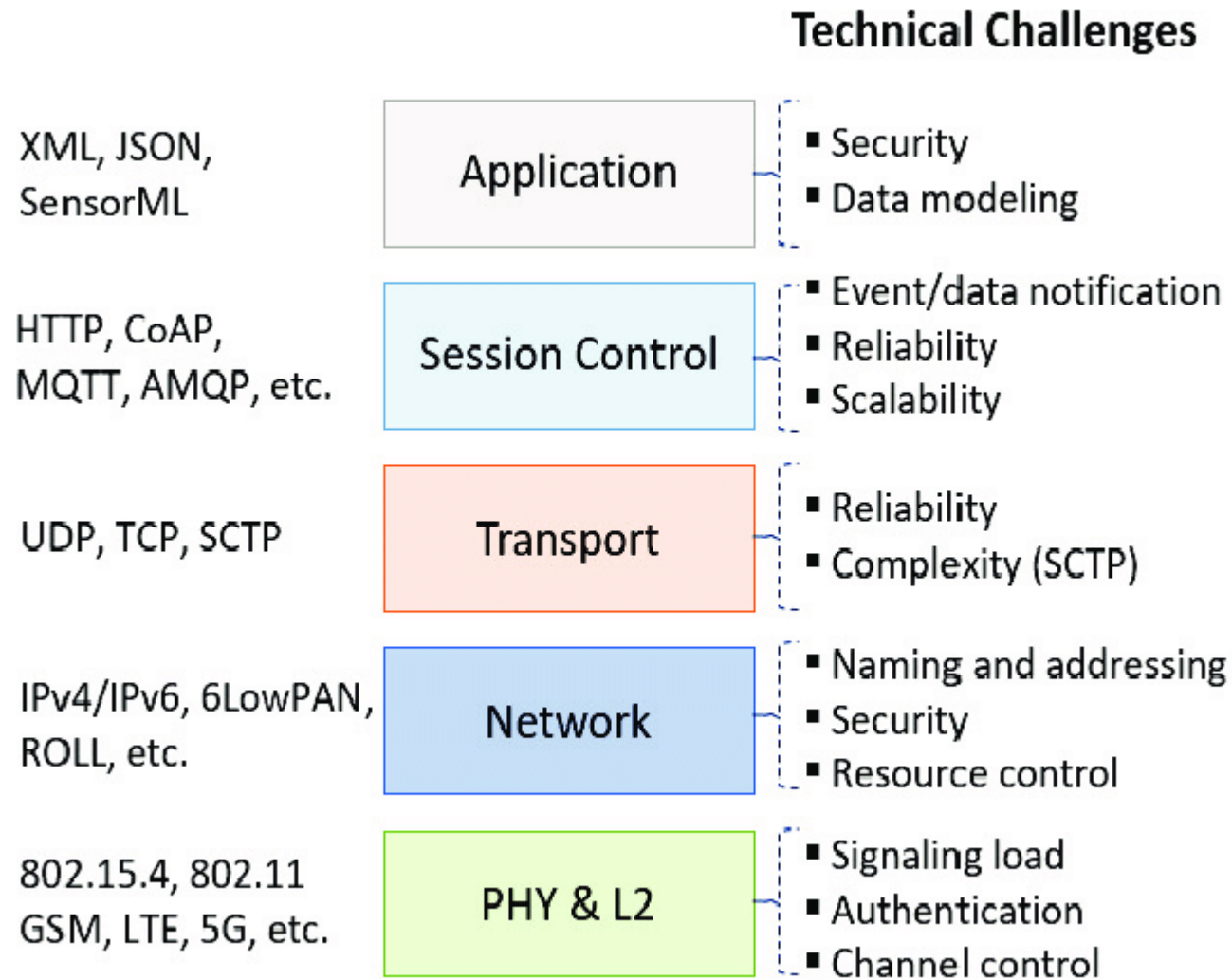


- **Organization : IETF (Internet Engineering Task Force)**
- **Standardized provisions : RFC (Request for Comments)**
 - 1) **IP : RFC791**
 - 2) **TCP : RFC9293 Released in August 2022**
(**Obsoletes: 793, 879, 2873, 6093, 6429, 6528, 6691**)

1-2. What is TCP/IP?

- **TCP/IP** Regulations for communication on the Internet.
 - ① **IP** : The role of representing addresses on the Internet.
 - ② **TCP** : The role of delivering data efficiently and reliably according to the state of the communication path.
- **UDP/IP**
 - ③ **UDP** : It does not perform communication control just to deliver.

1-3. Protocol stack



https://www.researchgate.net/figure/Protocol-Stack-and-Technical-Challenges_fig1_320453832

- **Match the rules used by each layer at the transmitting and receiving terminals**
- **There is no dependency between layers. Determine the rules to be used for each layer**

1-4. Packet capture (Wireshark)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.7	157.7.107.210	TCP	66	53275 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000364	192.168.1.7	157.7.107.210	TCP	66	59662 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.015081	157.7.107.210	192.168.1.7	TCP	66	80 → 53275 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1414 SACK_PERM=1 WS=128
4	0.015193	192.168.1.7	157.7.107.210	TCP	54	53275 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
5	0.016168	192.168.1.7	157.7.107.210	HTTP	614	GET / HTTP/1.1
6	0.016390	157.7.107.210	192.168.1.7	TCP	66	80 → 59662 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1414 SACK_PERM=1 WS=128
7	0.016462	192.168.1.7	157.7.107.210	TCP	54	59662 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
8	0.031443	157.7.107.210	192.168.1.7	TCP	60	80 → 53275 [ACK] Seq=1 Ack=561 Win=30336 Len=0
9	0.542488	157.7.107.210	192.168.1.7	TCP	392	80 → 53275 [PSH, ACK] Seq=1 Ack=561 Win=30336 Len=338 [TCP segment of a reassembled PDU]
10	0.543525	157.7.107.210	192.168.1.7	TCP	2882	80 → 53275 [ACK] Seq=339 Ack=561 Win=30336 Len=2828 [TCP segment of a reassembled PDU]
11	0.543622	192.168.1.7	157.7.107.210	TCP	54	53275 → 80 [ACK] Seq=561 Ack=3167 Win=131328 Len=0
12	0.544684	157.7.107.210	192.168.1.7	TCP	8538	80 → 53275 [ACK] Seq=3167 Ack=561 Win=30336 Len=8484 [TCP segment of a reassembled PDU]

> Frame 5: 614 bytes on wire (4912 bits), 614 bytes captured (4912 bits) on interface \Device\NPF_{EE298557-9F26-440F-870F-072F5825E194}, id 0
> Ethernet II, Src: Tp-LinkT 09:d6:7d (28:ee:52:09:d6:7d), Dst: Mitsubis 86:d6:65 (10:4b:46:86:d6:65)
> Internet Protocol Version 4, Src: 192.168.1.7, Dst: 157.7.107.210
> Transmission Control Protocol, Src Port: 53275, Dst Port: 80, Seq: 1, Ack: 1, Len: 560
> Hypertext Transfer Protocol

Ethernet: 14Byte

IP: 20Byte

TCP: 20Byte

0000	10 4b 46 86 d6 65 28 ee 52 09 d6 7d 08 00 45 00	·KF·e(· R·}··E·
0010	02 58 15 30 40 00 80 06 18 e7 c0 a8 01 07 9d 07	·X·0@·· ······
0020	6b d2 d0 1b 00 50 38 57 b9 9e 16 b5 52 a1 50 18	k· ·P8W ····R·P·
0030	02 01 e5 e0 00 00 47 45 54 20 2f 20 48 54 54 50	· ·····GE T / HTTP
0040	2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6d 61 6e 61	/1.1·Ho st: mana
0050	6b 61 6e 2e 6e 65 74 0d 0a 43 6f 6e 6e 65 63 74	kan.net· ·Connect
0060	69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d	ion: kee p-alive·
0070	0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72	·Upgrade -Insecur
0080	65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 55	e-Request s: 1·U
0090	73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c	ser-Agen t: Mozil

1-5. TCP Versions

Contents of this time

- **Initial standard, most basic TCP**
 - 「Reno」 「NewReno」 *1
- **For high delay/wideband (long fat pipe)**
 - 「BIC」 「CUBIC」 「H-TCP」 *1、 「Fast TCP」 *2、 「Illinois」 *3
- **For data centers with low latency and wide bandwidth**
 - 「DCTCP」 *3
- **For wireless environments such as mobile lines and Wi-Fi**
 - 「Veno」 「Westwood」 *2

《Reference 1: Method》

*1: Loss base method

*2: Delay-based method

*3: Hybrid method of the above two

《Reference 2: Implementation of each OS》

Linux、 Android : CUBIC

OS X : NewReno

Windows : CTCP/DCTCP

1-6. TCP Data Format

- Defined by RFC 9293

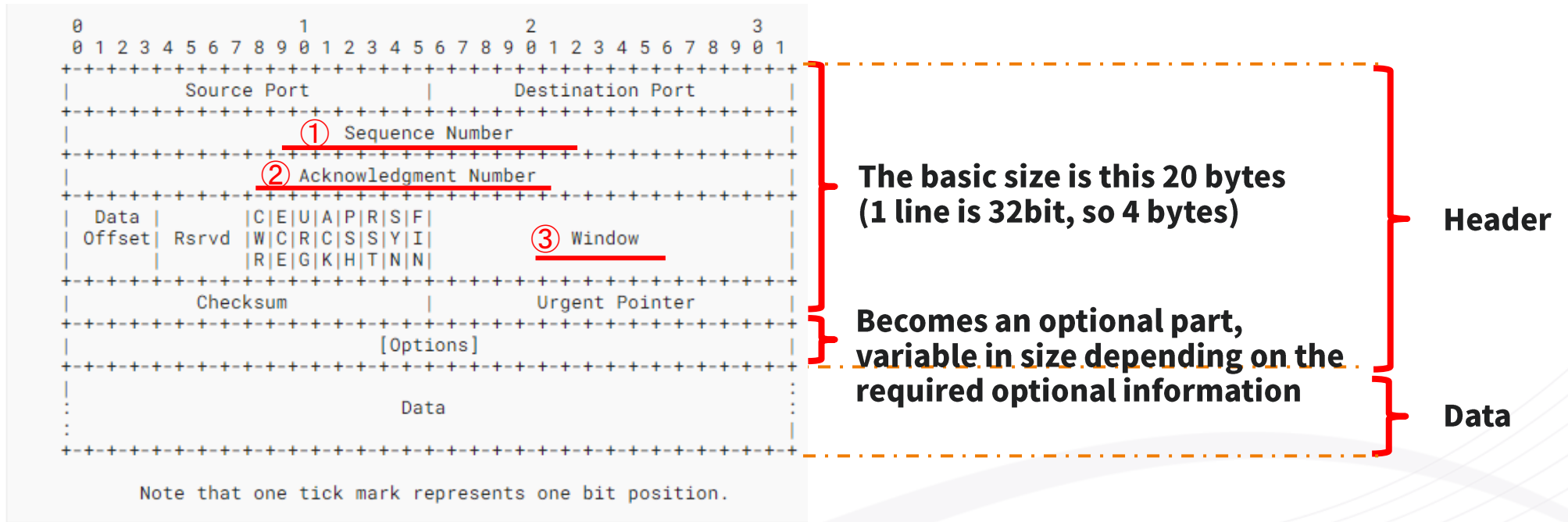


Figure 1: TCP Header Format

①	Sequence Number	The sending terminal attaches a number (Sequence Number) to the sending data and sends it.
②	Acknowledgment Number	Based on the "Sequence Number" received by the receiving terminal, the "Sequence Number" expected to be received next is notified to the sender as an acknowledgment (Acknowledgment Number).
③	Window	Notifies the sender of the amount of data that the receiving terminal can receive (receive buffer size).

2. TCP Algorithm

2-1. Establishing a TCP connection

2-2. Arrival confirmation

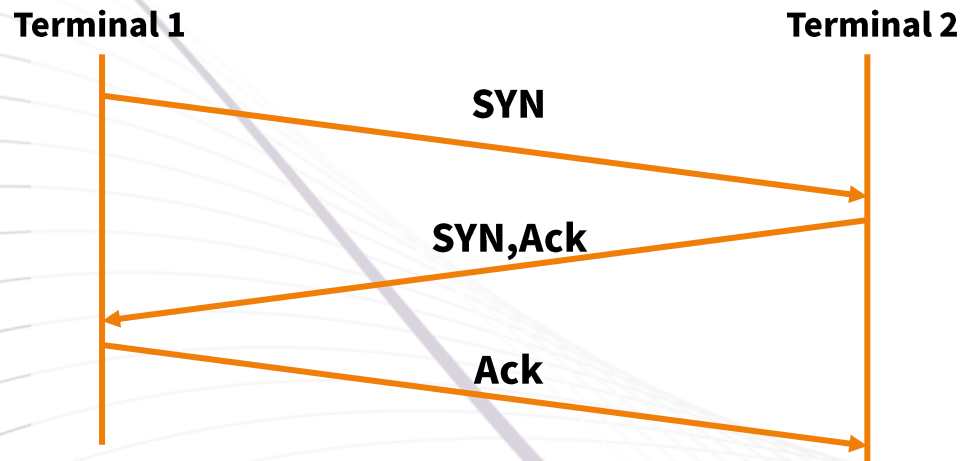
2-3. Window control

2-4. Determining the window size

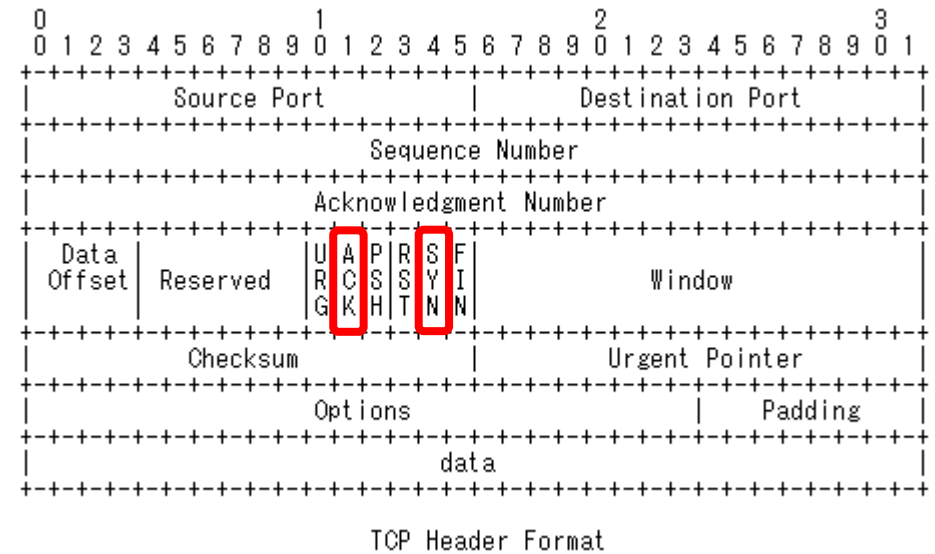
2-5. Retransmission control

2-1. Establishing a TCP connection

3-Way Handshake



Data is sent from here onwards



The availability of optional features such as SelectiveAck and window size options are also determined at this establishment.

Packet capture (Wireshark)

http_cap.pcapng

ファイル(F) 編集(E) 表示(V) 移動(G) キャプチャ(C) 分析(A) 統計(S) 電話(y) 無線(W) ツール(T) ヘルプ(H)

表示フィルタ... <Ctrl-/> を適用

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.7	157.7.107.210	TCP	66	53275 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000364	192.168.1.7	157.7.107.210	TCP	66	59662 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.015081	157.7.107.210	192.168.1.7	TCP	66	80 → 53275 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1414 SACK_PERM=1 WS=128
4	0.015193	192.168.1.7	157.7.107.210	TCP	54	53275 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
5	0.016168	192.168.1.7	157.7.107.210	HTTP	614	GET / HTTP/1.1
6	0.016390	157.7.107.210	192.168.1.7	TCP	66	80 → 59662 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1414 SACK_PERM=1 WS=128
7	0.016462	192.168.1.7	157.7.107.210	TCP	54	59662 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0
8	0.031443	157.7.107.210	192.168.1.7	TCP	60	80 → 53275 [ACK] Seq=1 Ack=561 Win=30336 Len=0
9	0.542488	157.7.107.210	192.168.1.7	TCP	392	80 → 53275 [PSH, ACK] Seq=1 Ack=561 Win=30336 Len=338 [TCP segment of a reassembled PDU]
10	0.543525	157.7.107.210	192.168.1.7	TCP	2882	80 → 53275 [ACK] Seq=339 Ack=561 Win=30336 Len=2828 [TCP segment of a reassembled PDU]
11	0.543622	192.168.1.7	157.7.107.210	TCP	54	53275 → 80 [ACK] Seq=561 Ack=3167 Win=131328 Len=0
12	0.544684	157.7.107.210	192.168.1.7	TCP	8538	80 → 53275 [ACK] Seq=3167 Ack=561 Win=30336 Len=8484 [TCP segment of a reassembled PDU]

> Frame 5: 614 bytes on wire (4912 bits), 614 bytes captured (4912 bits) on interface \Device\NPF_{EE298557-9F26-440F-870F-072F5825E194}, id 0

> Ethernet II, Src: Tp-LinkT_09:d6:7d (28:ee:52:09:d6:7d), Dst: Mitsubis_86:d6:65 (10:4b:46:86:d6:65)

> Internet Protocol Version 4, Src: 192.168.1.7, Dst: 157.7.107.210

> Transmission Control Protocol, Src Port: 53275, Dst Port: 80, Seq: 1, Ack: 1, Len: 560

> Hypertext Transfer Protocol

```
0000  10 4b 46 86 d6 65 28 ee 52 09 d6 7d 08 00 45 00  ·KF··e(· R·}··E·
0010  02 58 15 30 40 00 80 06 18 e7 c0 a8 01 07 9d 07  ·X·0@··· ······
0020  6b d2 d0 1b 00 50 38 57 b9 9e 16 b5 52 a1 50 18  k····P8W ····R·P·
0030  02 01 e5 e0 00 00 47 45 54 20 2f 20 48 54 54 50  ······GE T / HTTP
0040  2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6d 61 6e 61  /1.1··Ho st: mana
0050  6b 61 6e 2e 6e 65 74 0d 0a 43 6f 6e 6e 65 63 74  kan.net· ·Connect
0060  69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 65 0d  ion: kee p-alive·
0070  0a 55 70 67 72 61 64 65 2d 49 6e 73 65 63 75 72  ·Upgrade -Insecur
0080  65 2d 52 65 71 75 65 73 74 73 3a 20 31 0d 0a 55  e-Request: 1·U
0090  73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c  ser-Agen t: Mozil
```

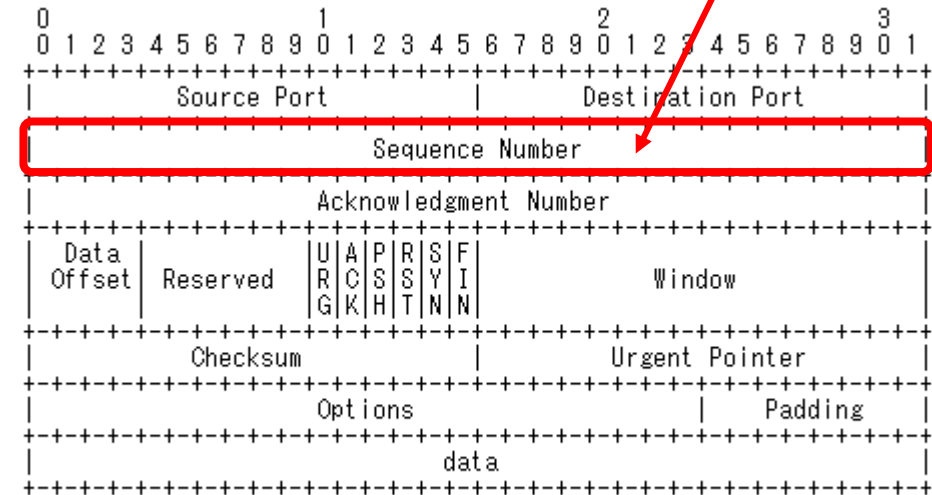
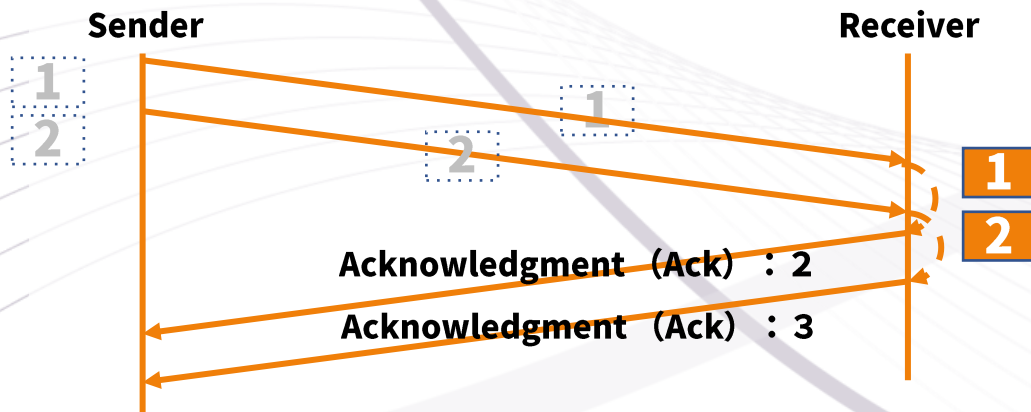
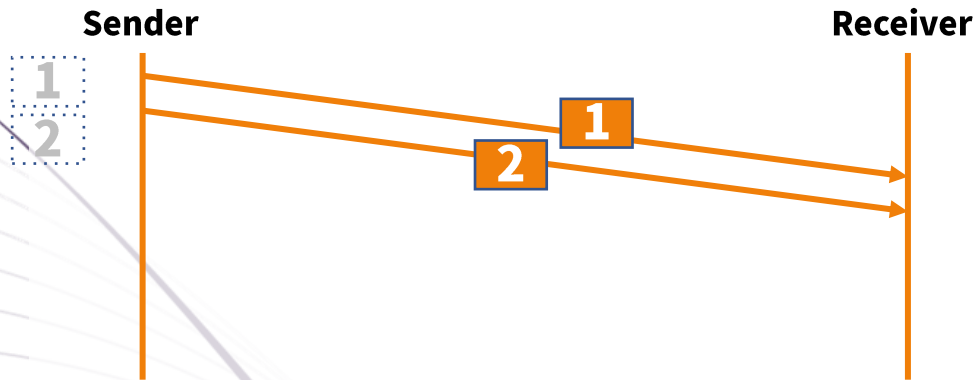
Transmission Control Protocol (tcp), 20 バイト

パケット数: 20 · 表示: 20 (100.0%)

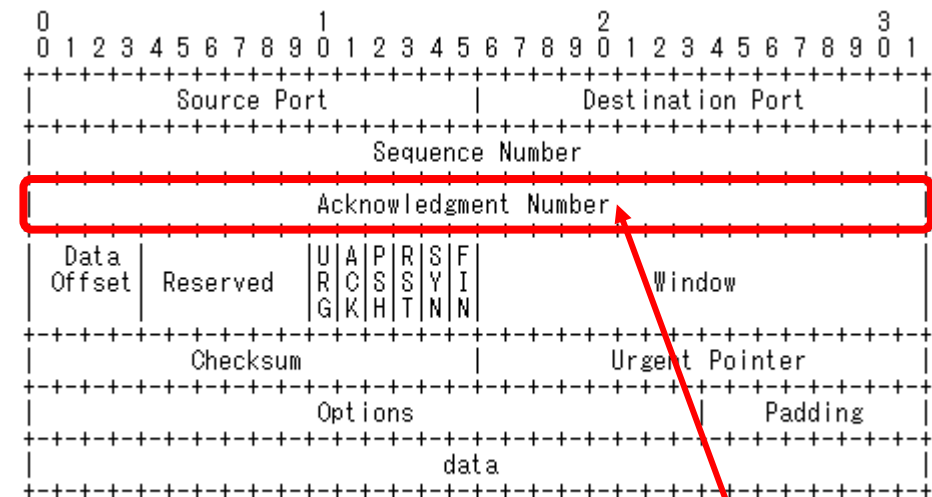
プロファイル: Default

2-2. Arrival confirmation

Send data number



TCP Header Format



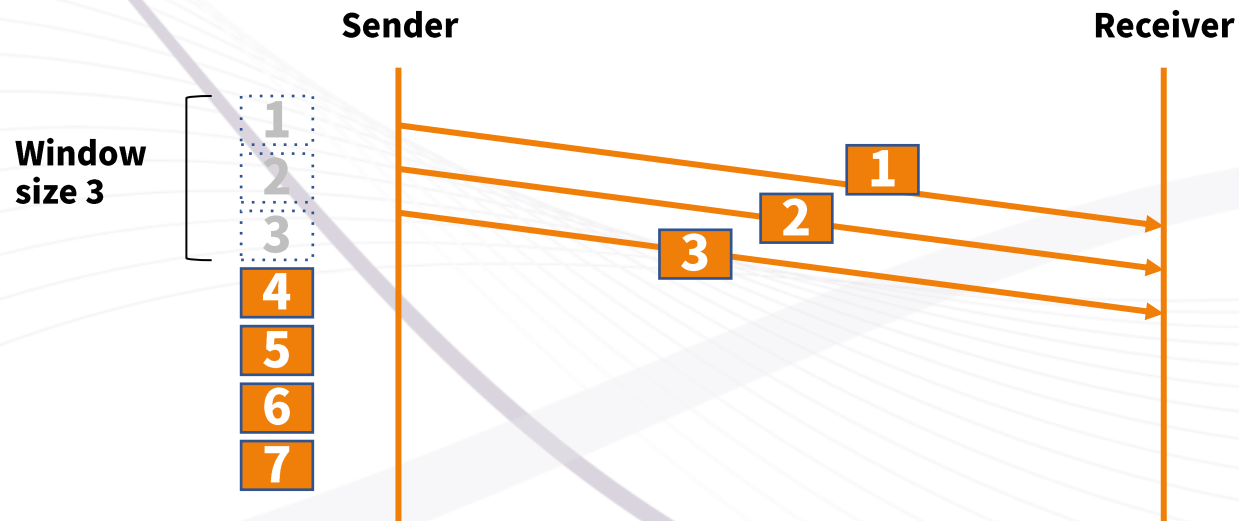
TCP Header Format

Arrival confirmation number
(Next number to receive)

2-3. Window Control

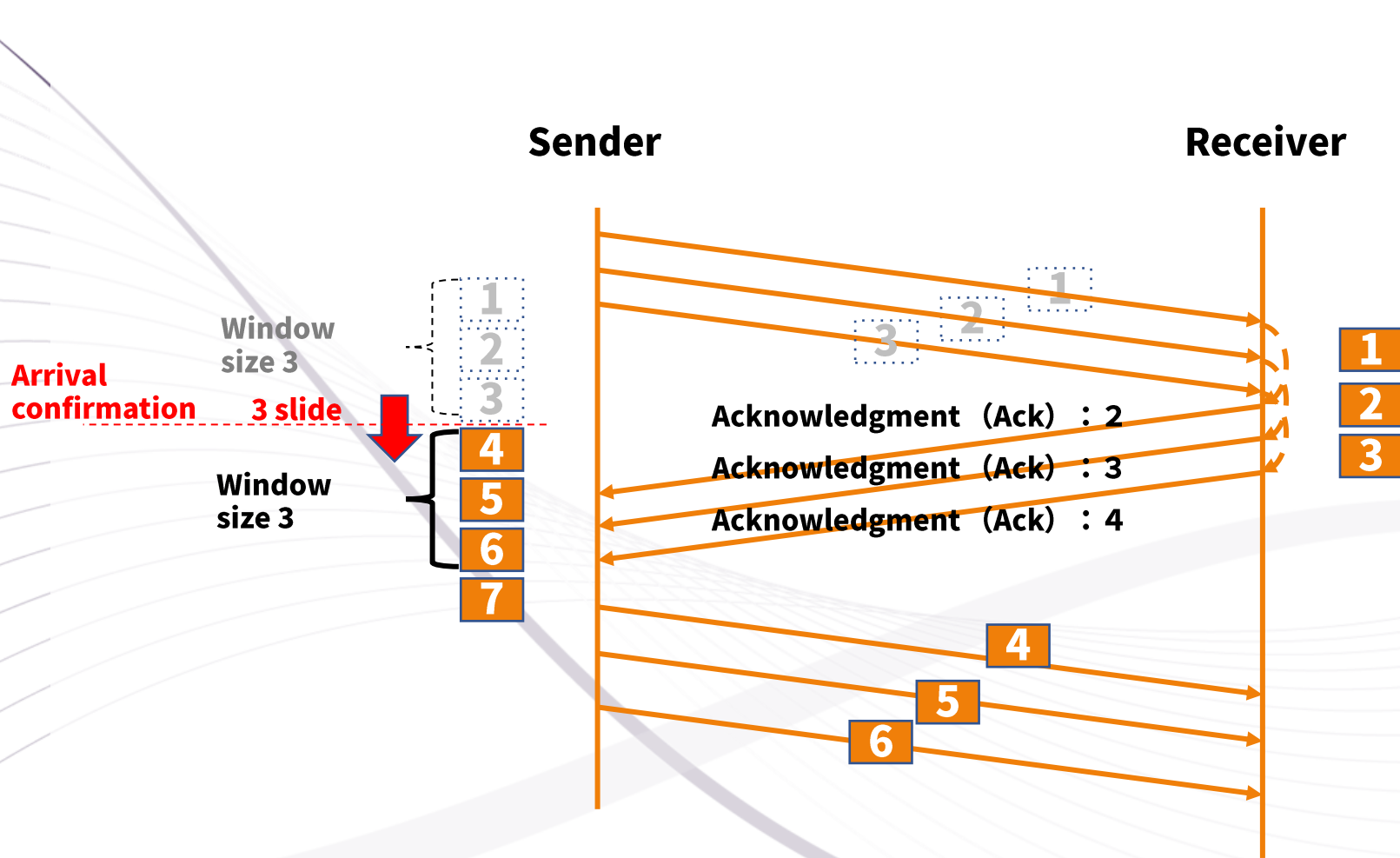
What is a “window”

Amount of data (bytes) that can be sent even though arrival has not been confirmed.



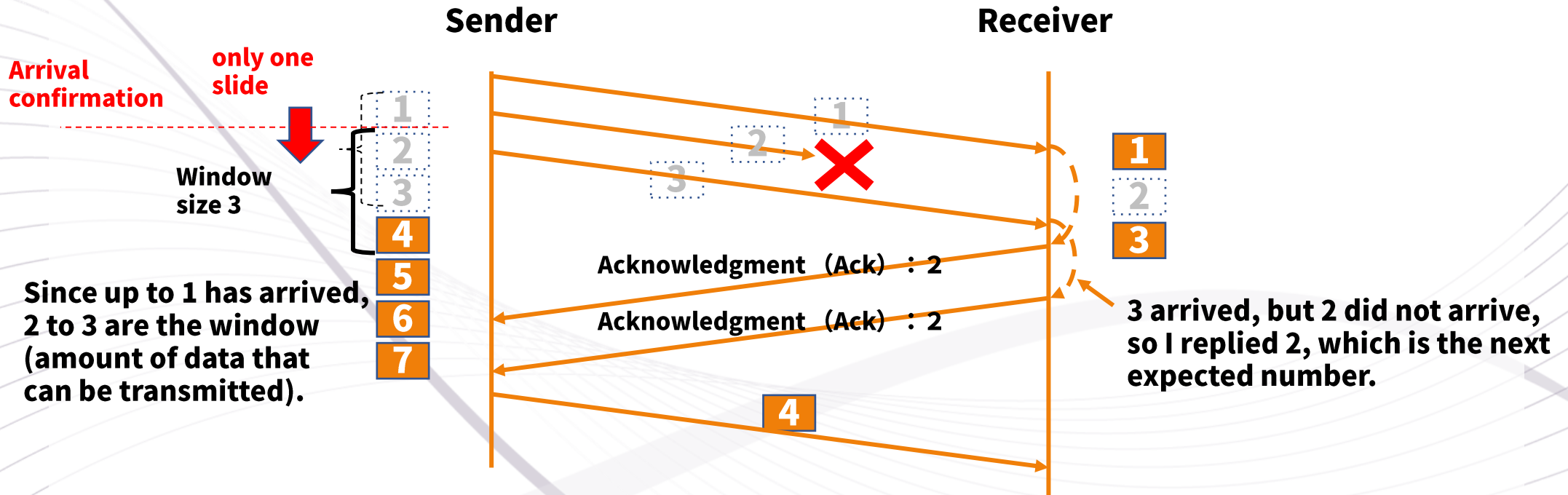
2-3. Window Control

[When everything arrives normally]



2-3. Window Control

[When packet 2 is lost]



2-4. Determining window size

The window size is the size that can be sent, so it is determined by the sending terminal. The decision is made by the smaller value of the following two contents.

1) Flow control

Controlled by the buffer size (window size) notified from the receiving terminal.

2) Congestion control

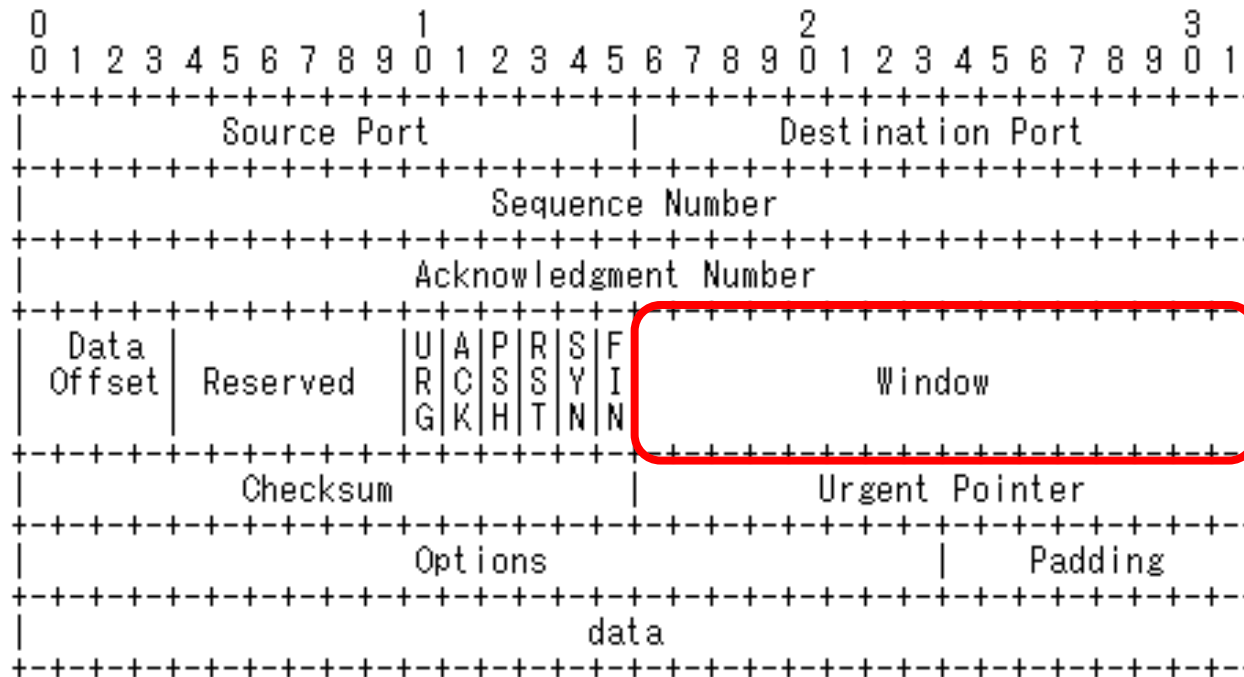
Controlled by the window size calculated by the sending terminal

Nowadays, the performance of personal computers has improved, so the buffer size of the receiving terminal has little effect, and it is safe to assume that the congestion control calculated on the sending side will become the window size.

2-4-1. Flow control

Receiving terminal notifies sending terminal of current buffer size

- Notify with TCP packet header "Window (16bit)"
- 65535 bytes (64 Kbytes) that can be displayed with a maximum size of 16 bits
- In recent years, it is common to use window size options that can expand the maximum size



TCP Header Format

2-4-2. Congestion control (congestion window)

- **Congestion control is controlled by a congestion window.**
- **Congestion window is a value that is stored and calculated inside the sender terminal**
- **No information appears in the packet header.**

Due to the high performance of personal computers, the congestion window size calculated at the sender terminal for congestion control is more dominant than the buffer size of the receiving terminal for flow control.

(However, since 64Kbyte of 16Bit is the maximum value without options and is dominant, it is generally expanded by the window size option.)

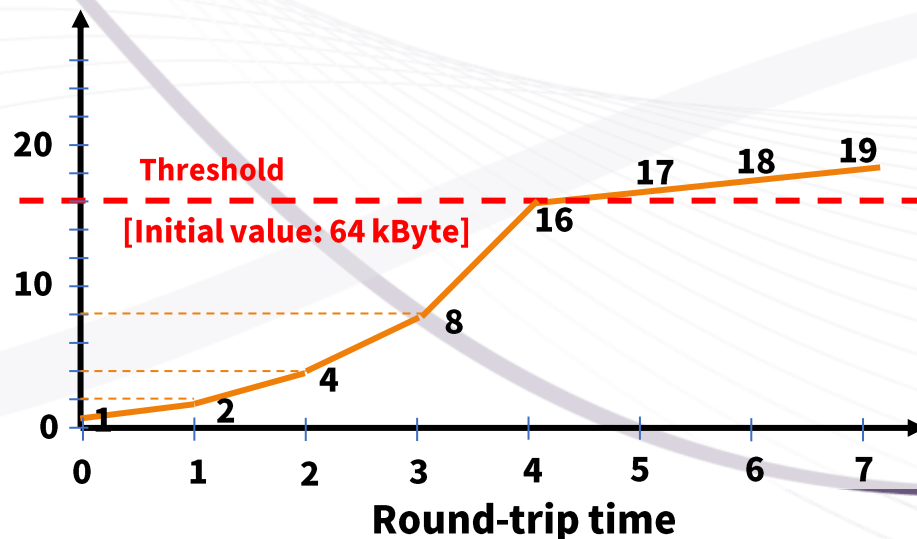
2-4-2. Congestion control (congestion window)

Congestion window (cwnd): Byte size (description is the number of packets)
The congestion window size is calculated in units of MSS (MaximumSegmentSize).

$$\begin{aligned} \text{MSS} &= \text{Ethernet Max Packet Length} - \text{IP Header Length} - \text{TCP Header Length} \\ &= 1500 \text{ Byte} - 20 \text{ Byte} - 20 \text{ Byte (No Option)} \\ &= 1460 \text{ Byte} \end{aligned}$$

There are two phases, [slow start] and [congestion avoidance], and the method of calculating the congestion window size is different.

Congestion window size (cwnd)



[Congestion avoidance]

1 MSS size (linear) increase per round-trip time

[Calculation]

$$\text{cwnd} = \text{cwnd} + \text{MSS}/\text{cwnd}$$

[Slow start]

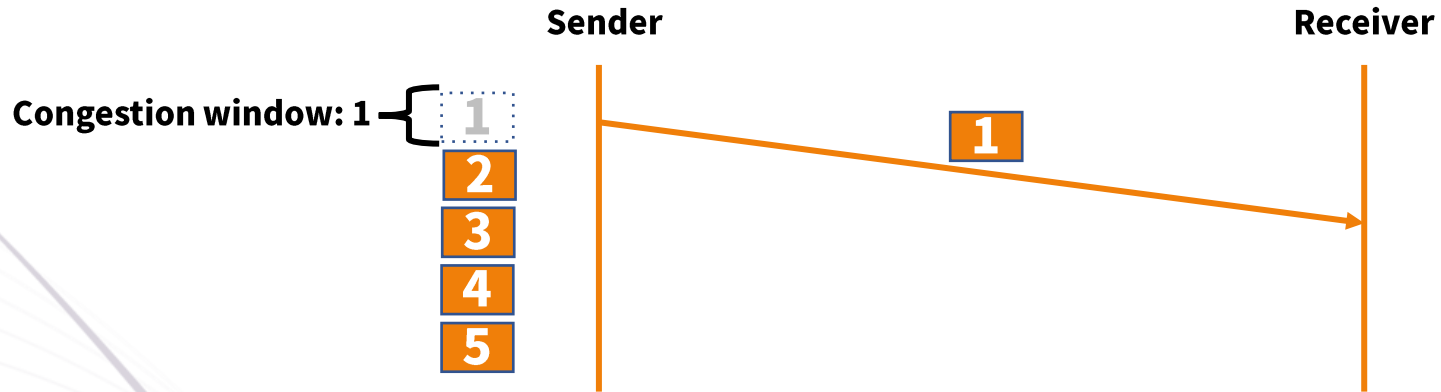
Exponential increase per round trip time

[Calculation]

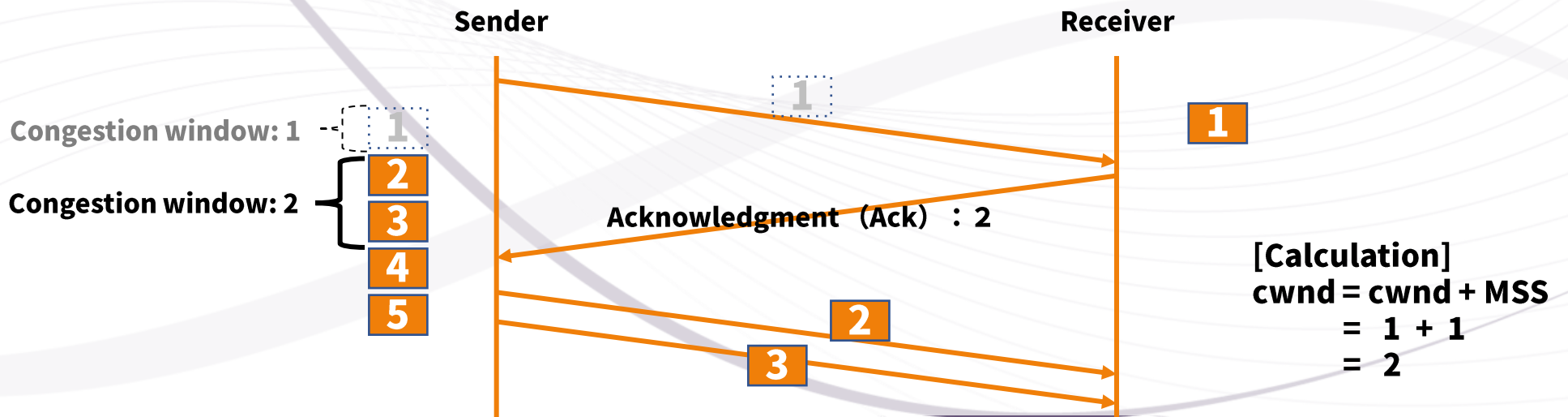
$$\text{cwnd} = \text{cwnd} + \text{MSS}$$

2-4-2-1. Slow start

[When establishing a connection]

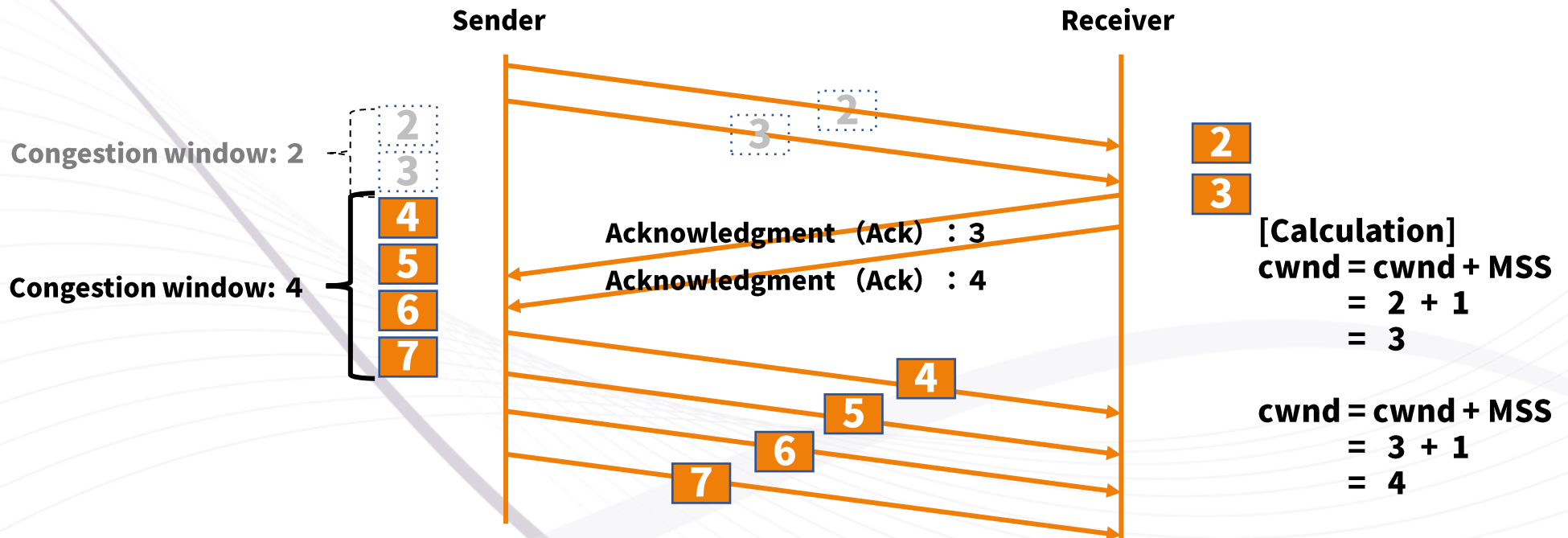


[When acknowledgment 1 is received]



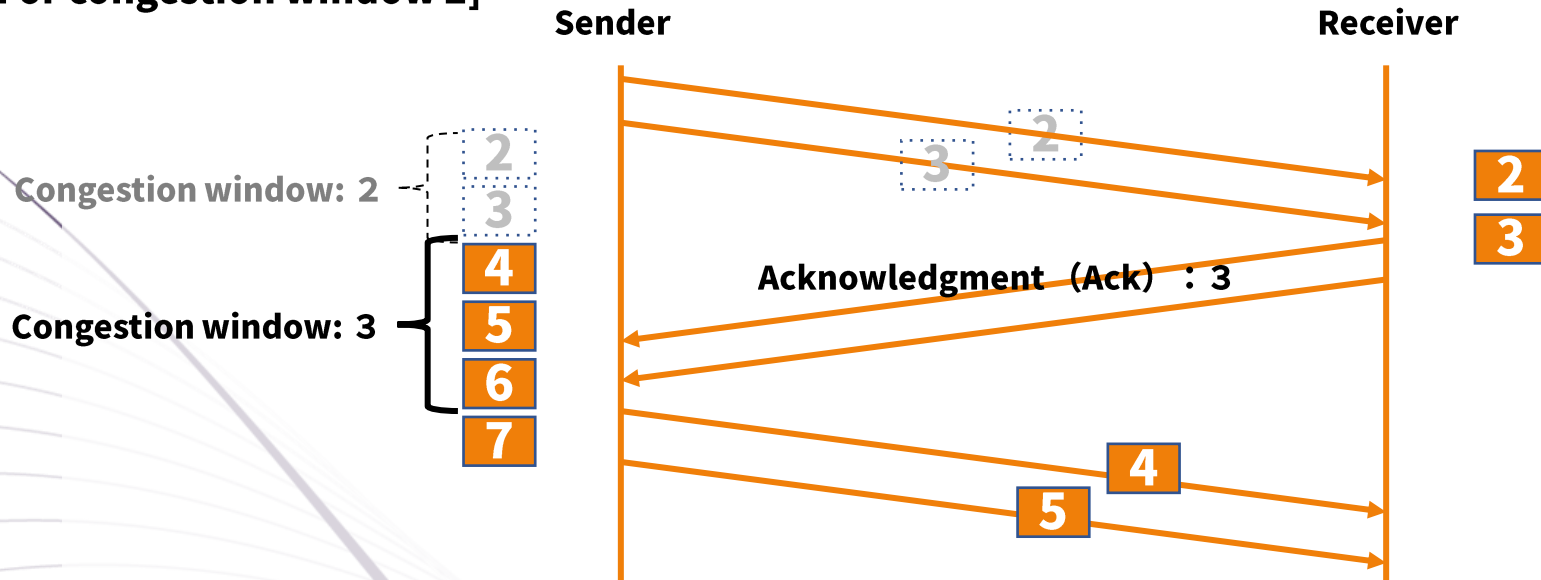
2-4-2-1. Slow start

[When receiving up to acknowledgment 3]



2-4-2-2. Congestion avoidance

[For congestion window 2]

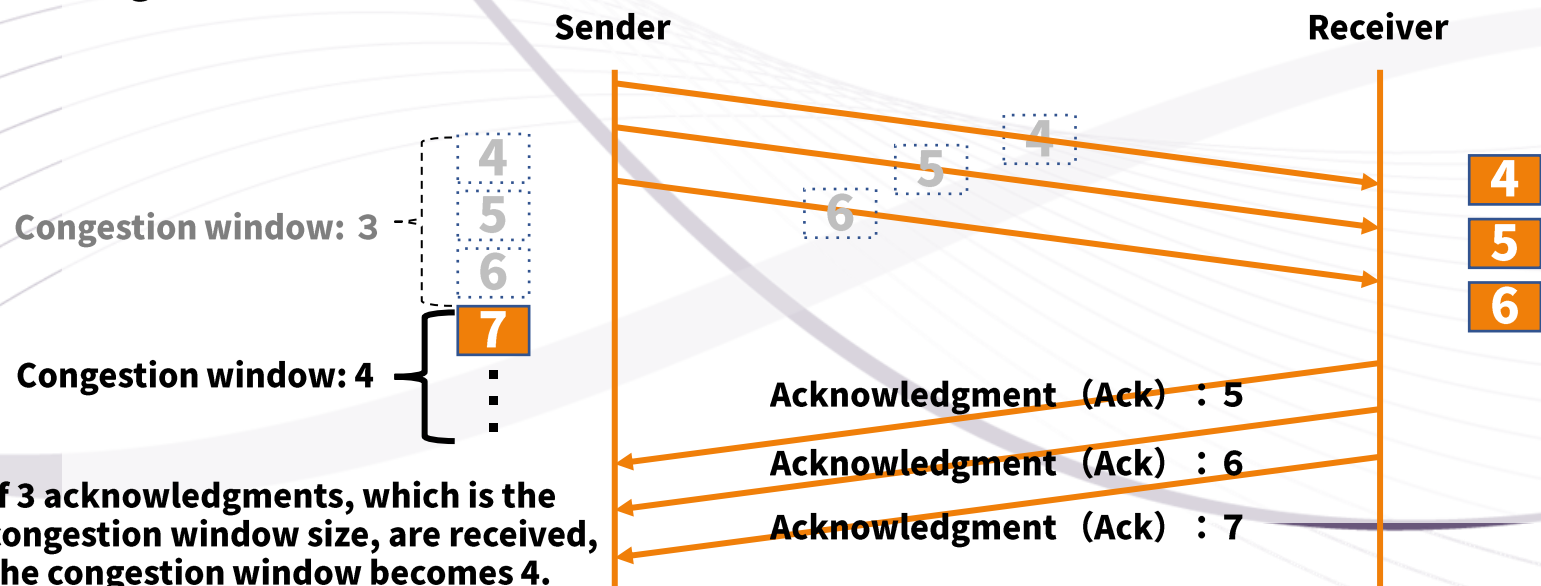


[Calculation]

$$\begin{aligned} \text{cwnd} &= \text{cwnd} + \text{MSS}/\text{cwnd} \\ &= 2 + 1/2 \\ &= 2 \frac{1}{2} \end{aligned}$$

$$\begin{aligned} \text{cwnd} &= \text{cwnd} + \text{MSS}/\text{cwnd} \\ &= 2 \frac{1}{2} + 1/2 \\ &= 3 \end{aligned}$$

[For congestion window 3]



[Calculation]

$$\begin{aligned} \text{cwnd} &= \text{cwnd} + \text{MSS}/\text{cwnd} \\ &= 3 + 1/3 \\ &= 3 \frac{1}{3} \end{aligned}$$

$$\begin{aligned} \text{cwnd} &= \text{cwnd} + \text{MSS}/\text{cwnd} \\ &= 3 \frac{1}{3} + 1/3 \\ &= 3 \frac{2}{3} \end{aligned}$$

$$\begin{aligned} \text{cwnd} &= \text{cwnd} + \text{MSS}/\text{cwnd} \\ &= 3 \frac{2}{3} + 1/3 \\ &= 4 \end{aligned}$$

If 3 acknowledgments, which is the congestion window size, are received, the congestion window becomes 4.

2-5. Retransmission control

The Internet is built on the assumption that packets are lost within the network.

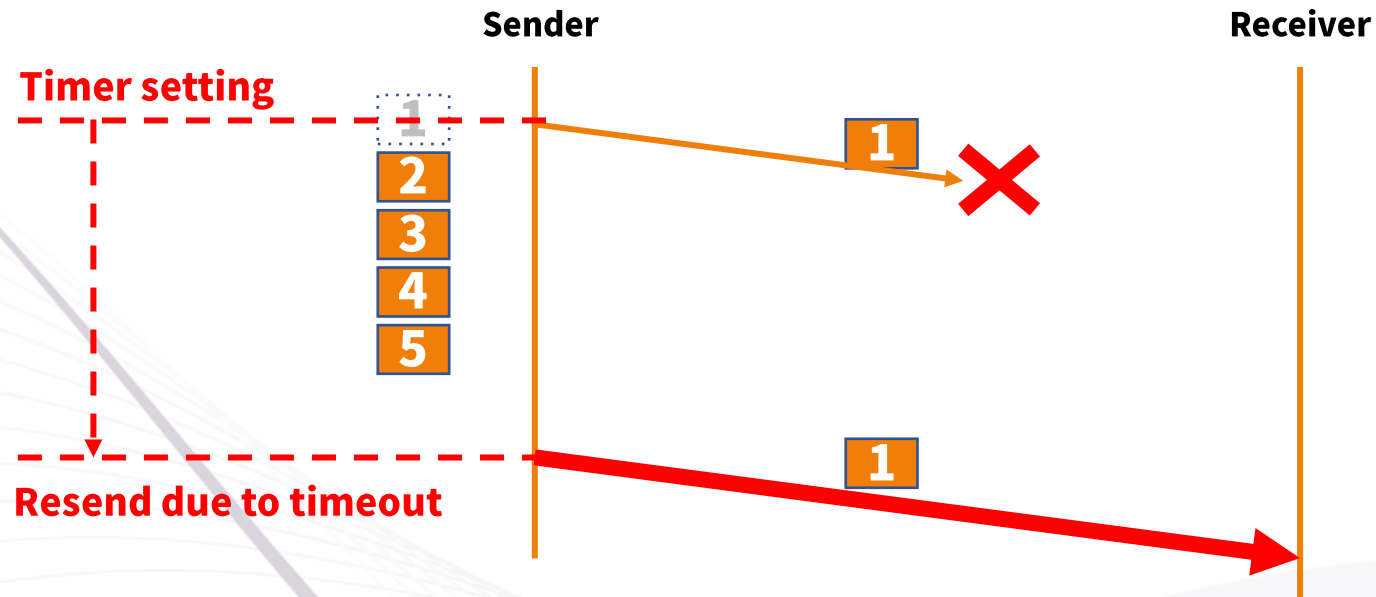
Therefore, TCP has a retransmission control function to ensure data delivery.

Regarding the resend function, the following two functions are implemented.

1) Retransmission due to timeout

2) Retransmission due to duplicate Ack

2-5-1. Retransmission due to timeout

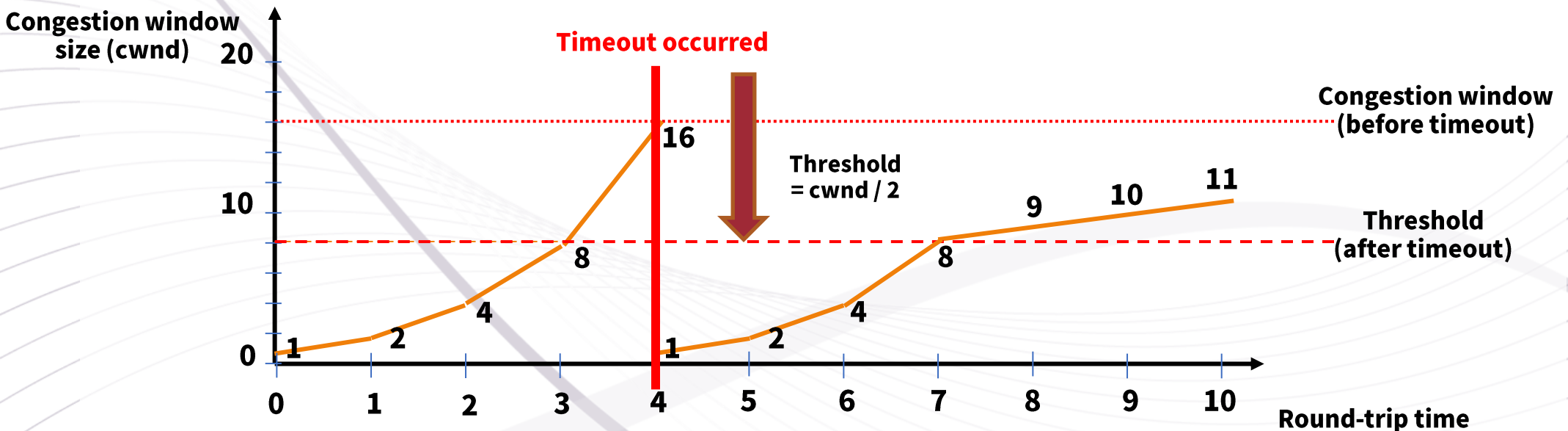


The initial value of the timer time is fixed, but it is updated from time to time based on the actual packet response time.

2-5-1. Retransmission due to timeout

When retransmission occurs due to timeout, the following window control is performed, and the fluctuation of the congestion window size is shown below.

- (1) Congestion window (cwnd) = 1 MSS size
- (2) Slow start threshold = $\text{cwnd} / 2$



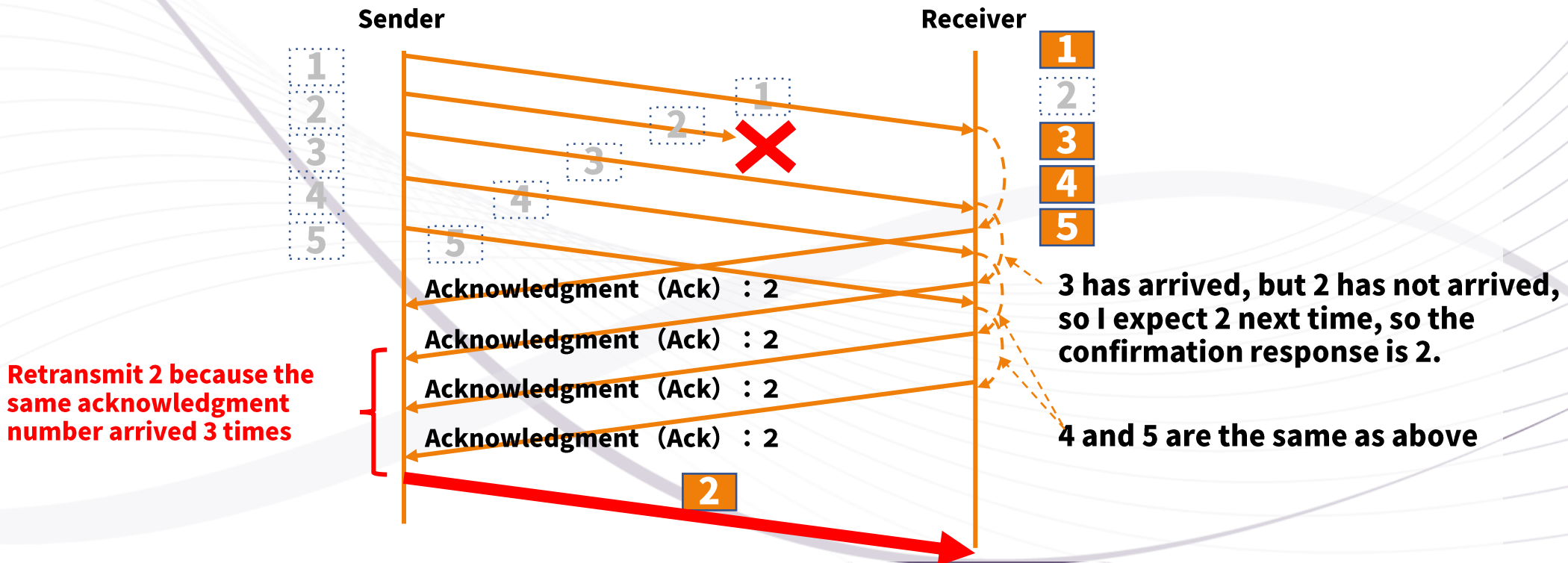
A timeout indicates that the network is very busy. Therefore, it is necessary to carefully transmit and widen the window little by little so as not to cause unnecessary packet loss.

2-5-2. Retransmission due to duplicate Ack

"Fast Recovery Algorithm"

Retransmission by duplicate Ack is retransmitted when the same sequence number as before arrives three times.

[If packet 2 is lost]



2-5-2. Retransmission due to duplicate Ack

Fast Recovery function: The congestion window is not the first 1 MSS size, but the following calculation is implemented to efficiently transmit data by retransmission.

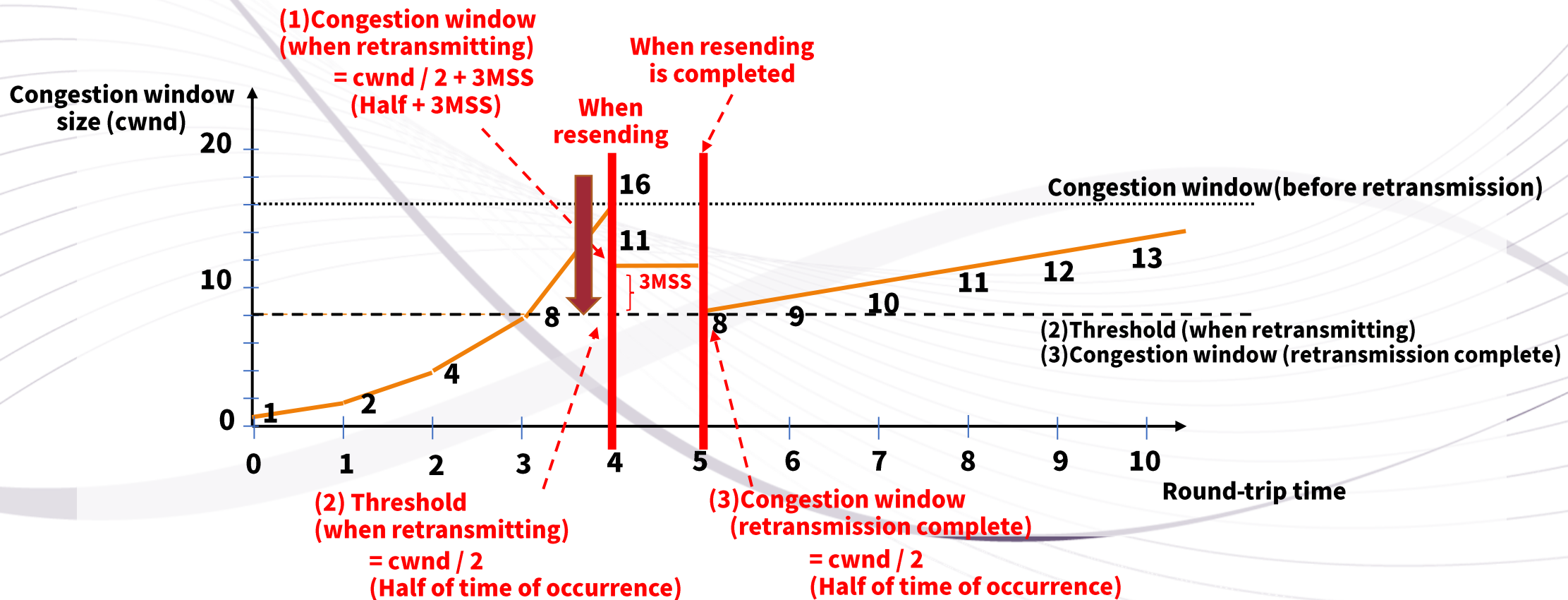
[When retransmission occurs]

(1) Congestion window (cwnd) = $cwnd / 2 + 3 \text{ MSS}$

(2) Slow start threshold = $cwnd / 2$

[When resend is completed]

(3) congestion window (cwnd) = slow start threshold



3. Summary [TCP Algorithm]

● Arrival confirmation

● Window control

- Flow control
- Congestion control
 - Slow start
 - Congestion avoidance

● Resend control

- Retransmission due to timeout ← [Large congestion]
- Retransmission by duplicate Ack ← [Less congestion]