TCP Protocol Specification

CPSC 441 - TUTORIAL 6

WINTER 2020

UNIVERSITY OF CALGARY

*Modified from slides by Reza Gholizadeh

What is TCP?

Transmission **C**ontrol **P**rotocol

Connection-Oriented and Reliable transport layer protocol

Features

Connection-Oriented

Byte Order Preservation

Flow and Congestion control

Reliability

TCP Header

10 requiring fields totaling 160 bits in size.

Options field is optional!

<table>
<tr><th>offset (bits)</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td><b>0</b></td><td colspan="16">Source Port Number</td><td colspan="16">Destination Port Number</td></tr>
<tr><td><b>32</b></td><td colspan="32">Sequence number</td></tr>
<tr><td><b>64</b></td><td colspan="32">Acknowledgment number (if ACK set)</td></tr>
<tr><td><b>96</b></td><td colspan="4">Data offset</td><td colspan="4">Reserved</td><td>C
W
R</td><td>E
C
E</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="16">Window Size</td></tr>
<tr><td><b>128</b></td><td colspan="16">Checksum</td><td colspan="16">Urgent pointer (if URG set)</td></tr>
<tr><td><b>160
…</b></td><td colspan="32">Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)</td></tr>
</table>

Source and **Destination Ports**

- Communication endpoints for sending and receiving data

Sequence number

- The accumulated sequence number of the first data byte of this segment
- Initial sequence number is random (When SYN is set)

Acknowledgment number

- Contains the next sequence number that the sender of ACK expects to receive. It acts as a receipt of all bytes in the previous segment (Works when ACK is set)

Data offset

- Represents the number 32-bit words in the TCP header. It should be 5 words at least and 15 words at most

TCP Header

10 requiring fields totaling 160 bits in size.

Options field is optional!

<table>
<tr><th>offset (bits)</th><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td><b>0</b></td><td colspan="16">Source Port Number</td><td colspan="16">Destination Port Number</td></tr>
<tr><td><b>32</b></td><td colspan="32">Sequence number</td></tr>
<tr><td><b>64</b></td><td colspan="32">Acknowledgment number (if ACK set)</td></tr>
<tr><td><b>96</b></td><td colspan="4">Data offset</td><td colspan="4">Reserved</td><td>C
W
R</td><td>E
C
E</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="16">Window Size</td></tr>
<tr><td><b>128</b></td><td colspan="16">Checksum</td><td colspan="16">Urgent pointer (if URG set)</td></tr>
<tr><td><b>160 …</b></td><td colspan="32">Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)</td></tr>
</table>

Reserved

- Should be set to zero

Flags

- CWR/ECE: Congestion Window Control
- URG: Shows that **Urgent pointer** field is active
- ACK: Indicates that **Acknowledgment** field is significant
- PSH: Activates **Push** function. Asks to push the buffered data to the receiving application
- RST: **Reset** the connection
- SYN: **Synchronize** sequence numbers (Only first packets of each end should have this flag set)
- FIN: Data from the sender is **Finished**

TCP Header

10 requiring fields totaling 160 bits in size.

Options field is optional!

offset (bits)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source Port Number																Destination Port Number															
32	Sequence number																															
64	Acknowledgment number (if ACK set)																															
96	Data offset				Reserved				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
128	Checksum																Urgent pointer (if URG set)															
160 …	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.)																															

Window Size

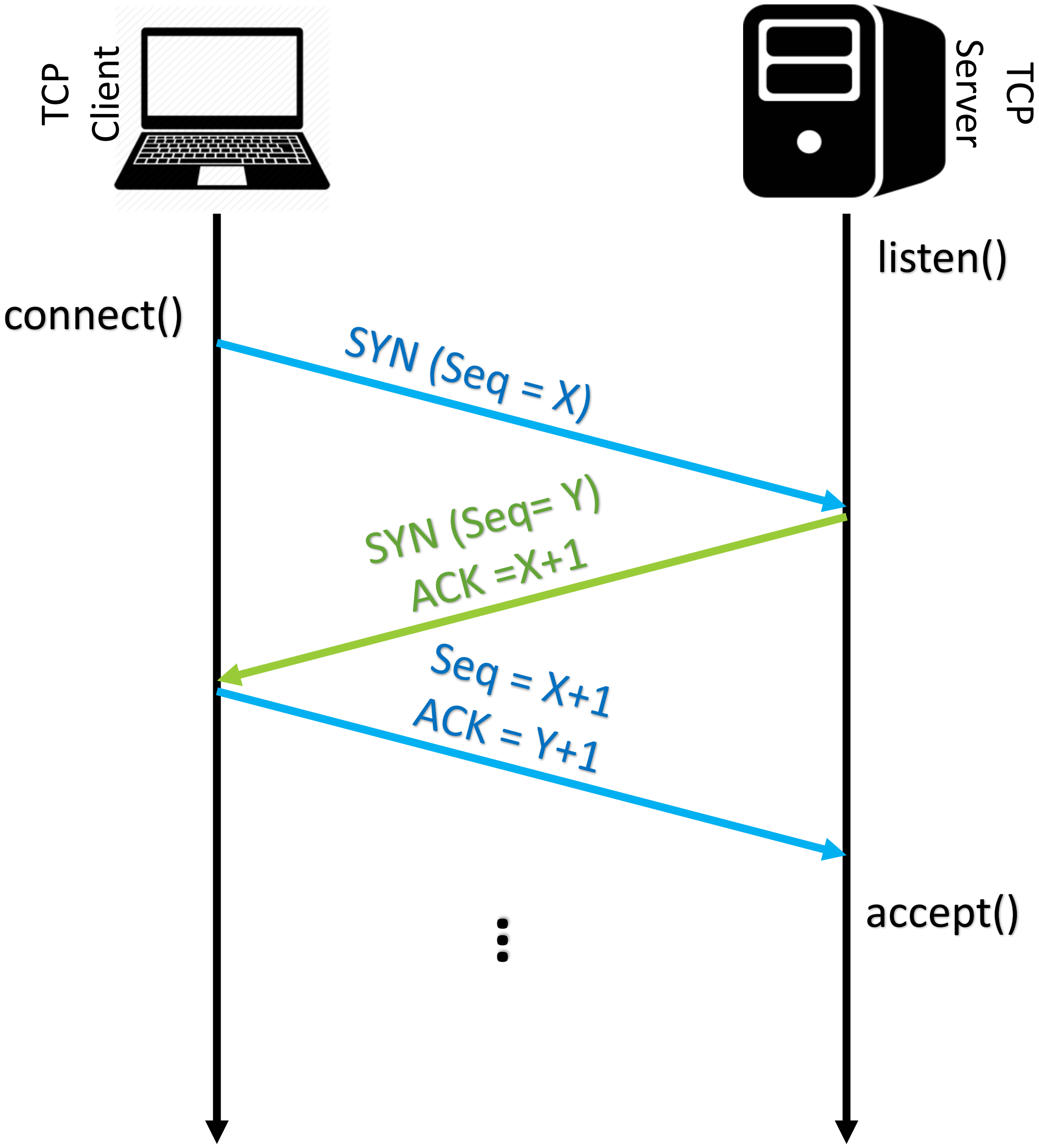
- The number of bytes that the receiver is currently willing to receive

Checksum

- Used for error checking of the header and data

Urgent pointer

- Is an offset from the sequence number indicating the last urgent data byte

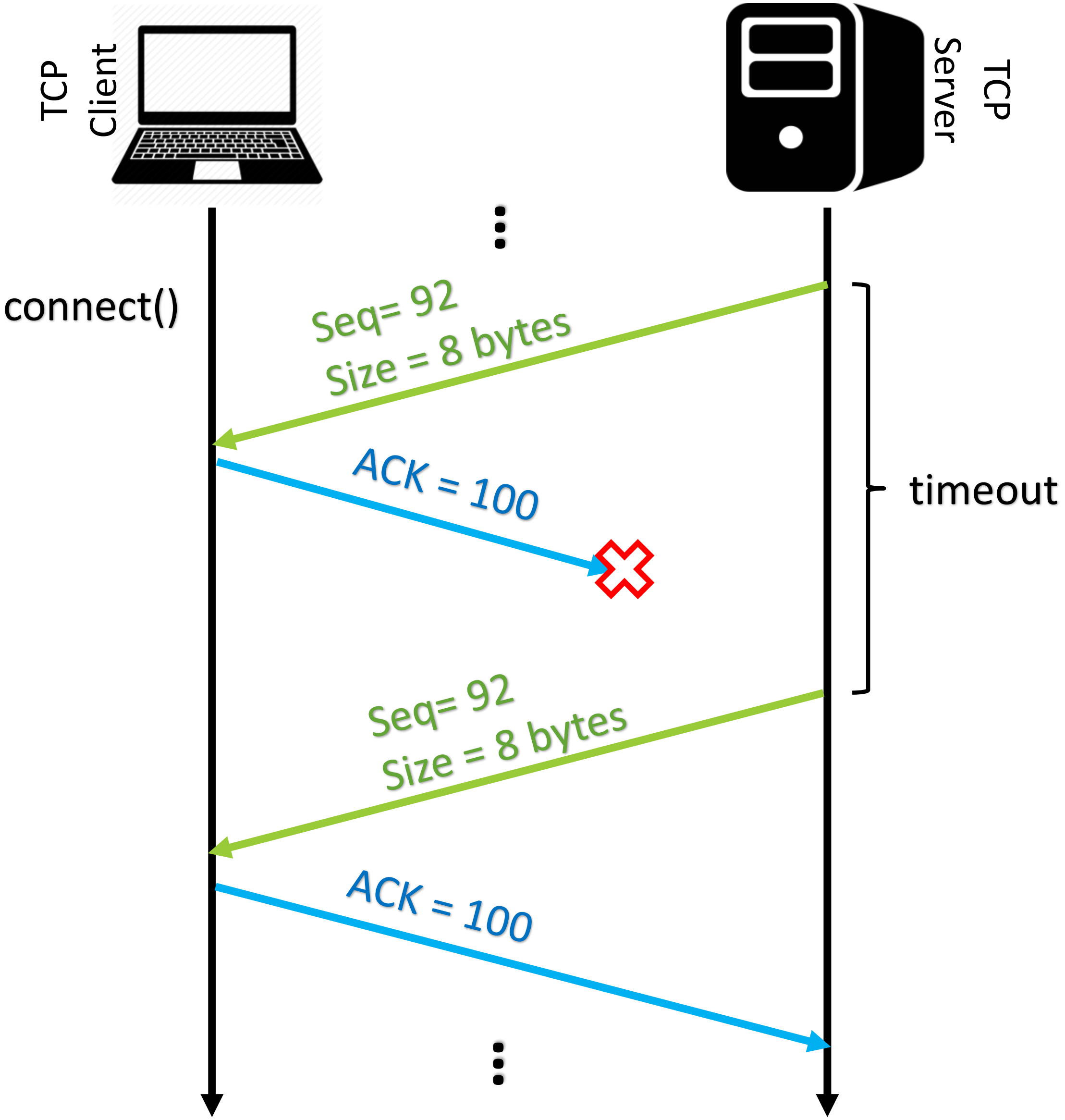


3-Way Handshake

Client sends SYN packet with initial sequence number of X

Server responds with its own SYN packet with initial sequence number of Y and acknowledgment number of X+1 (which is next expected byte)

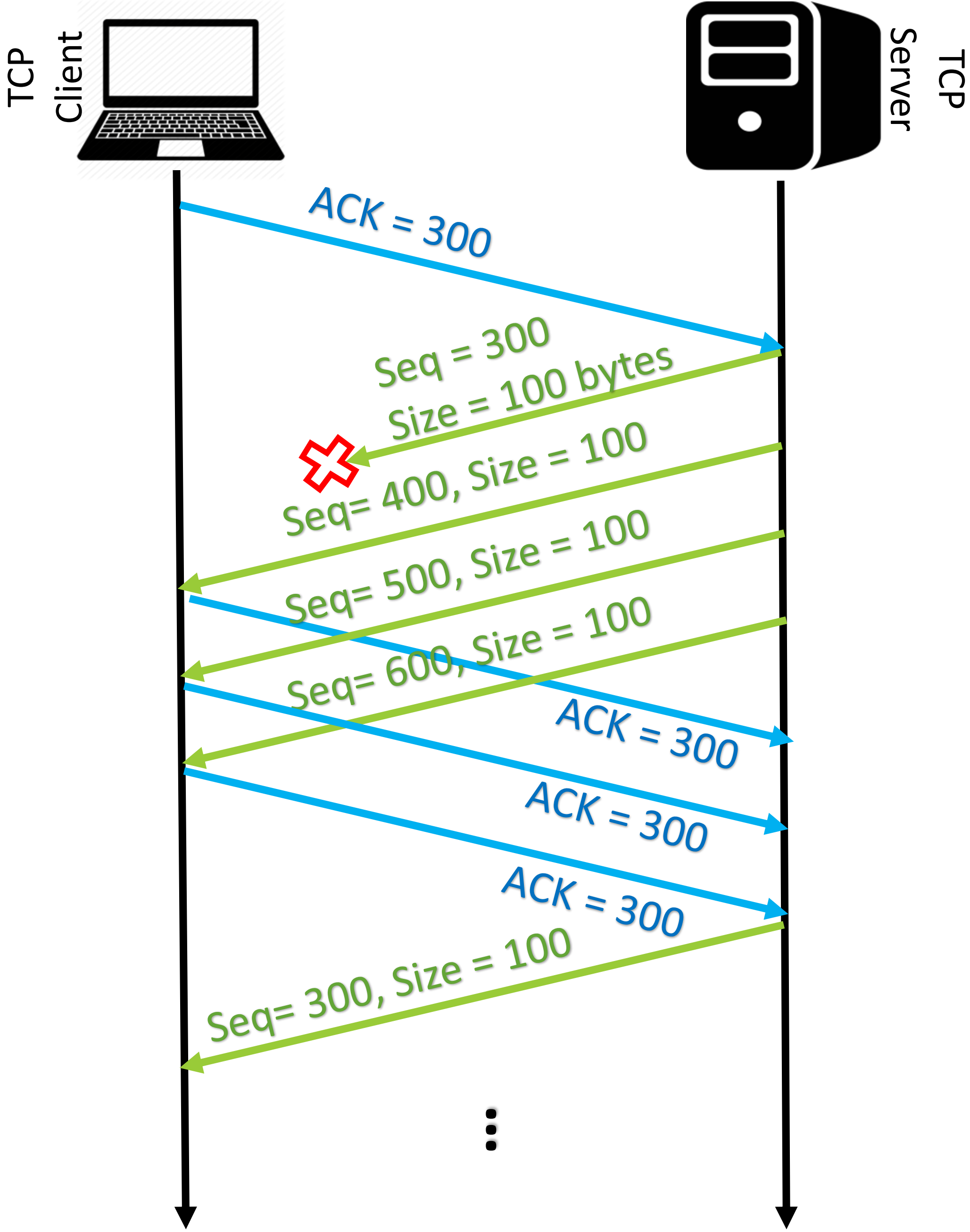
Client send a packet with sequence number of X+1 and acknowledgment number of Y+1



Retransmission Time-Out

Lost ACK Scenario:

- Sender sends a packet with data and waits for the receiver’s ACK
- Receiver send the ACK but somehow the packet is lost
- After waiting for a specific **amount of time** and not getting the ACK, sender retransmit the same packet of data. The event is called a **Retransmission Time-Out** (RTO)
- Receiver sends the ACK again



Fast Retransmission

Receiver expects N, gets N+1:

- Immediately sends ACK(N)
- This is called a duplicate ACK
- Does NOT delay ACKs here!
- Continue sending dup ACKs for each subsequent packet (not N)

Sender gets 3 duplicate ACKs:

- Infers N is lost and resends
- 3 is chosen so out-of-order packets don’t trigger Fast Retransmit accidentally
- Called “fast” since we don’t need to wait for a full RTT

Connection Termination

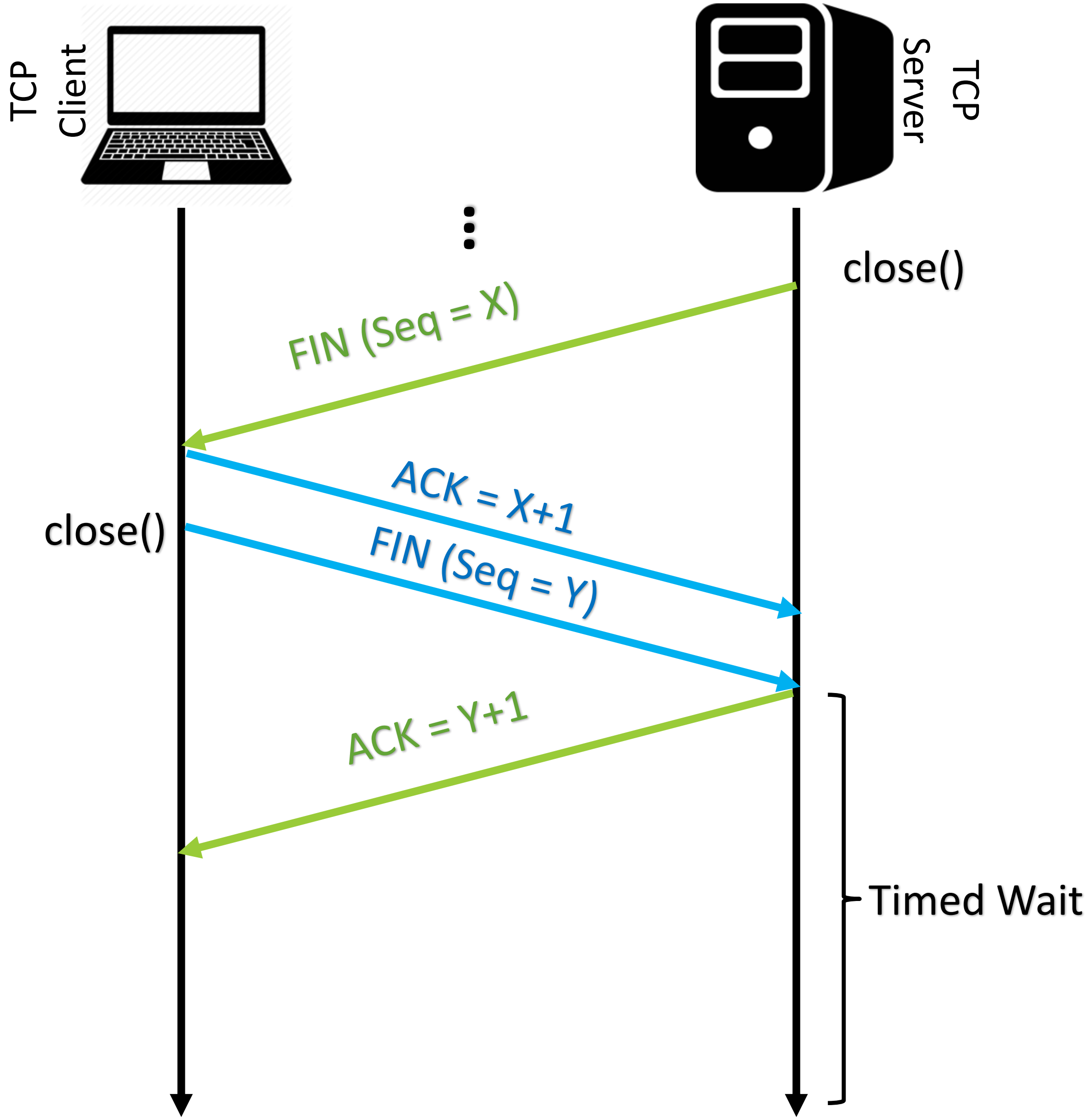
Either side may terminate a connection. (In fact, connection can stay half-closed.) Let's say the server closes (typical in WWW)

Server sends FIN with seq Number (SN+1) (i.e., FIN is a byte in sequence)

Client ACK's the FIN with SN+2 ("next expected")

Client sends it's own FIN when ready

Server ACK's client FIN as well with SN+1



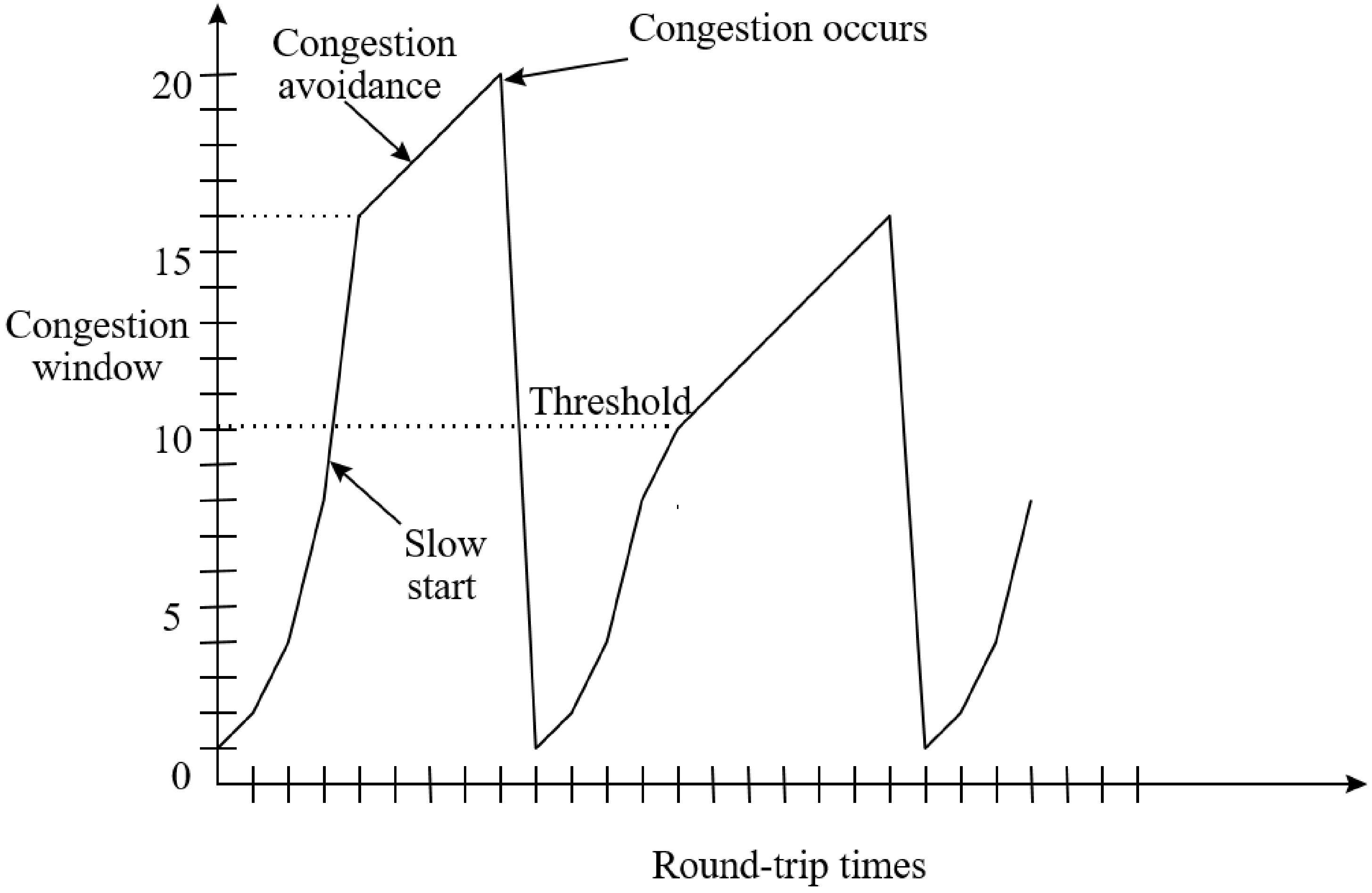
Congestion Control

Server perceives that there is congestion if:

- Timeout happens or
- The receipt of three duplicate ACKs

So then:

- It decreases the rate
- When it gets ACKs, starts increasing rate again



Congestion Control Algorithm

Three major components:

Slow start (mandatory)

Congestion avoidance (mandatory)

Fast recovery

We go to this state, when 3 duplicate ACKS happens

Timer in C

Library:
`#include <time.h>`

Function:
`clock()`

`CLOCKS_PER_SEC`

```
void setTimeout(int milliseconds)
{
     if (milliseconds <= 0)
     {
          printf("Count milliseconds for timeout is less or equal to
          0\n");
          return;
     }
     // a current time of milliseconds
     int milliseconds_since = clock() * 1000 / CLOCKS_PER_SEC; //
     needed count milliseconds of return from this timeout
     int end = milliseconds_since + milliseconds;
     // wait while until needed time comes
      do {
          milliseconds_since = clock() * 1000 / CLOCKS_PER_SEC;
      } while (milliseconds_since <= end);
 }
```

```
//c
void wait(long seconds)
{
 sleep(seconds);
}
```

Timer in C++

Library:
`#include <time.h>`

Function:
`clock()`

Type:
`clock_t`

`CLOCKS_PER_SEC`

```
void setTimeout(int milliseconds)
{
     clock_t start, end;

     // a current time of milliseconds
     clock_t start = clock() * 1000 / CLOCKS_PER_SEC;
     int gap;
      do {
          end = clock() * 1000 / CLOCKS_PER_SEC;
          gap = (int) end-start;
     } while (gap < milliseconds);
 }
```

```
//waits for 1 second or 1000 milliseconds
//c++
void wait(long seconds)
{
 seconds = seconds * 1000;
 Sleep(seconds);
}
```

Multi-process/multi-thread

C Programming:

`fork`

C++ Programming:

`thread`

```
#include <stdio.h>
#include <string.h>
#include <sys/types.h>

void main(void)
{
  pid_t pid;
  int i;
  fork();
  pid = getpid();
  if (pid>0)
  {
     for (i = 1; i <= 10; i++)
     {
          printf("This line is from pid %d\n", pid);
     }
  }
  else
  {
          printf("This line is from the parent process\n");
  }
}
```

References

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

https://www.tutorialspoint.com/c_standard_library/time_h.htm

http://www.cplusplus.com/forum/general/8255/

http://www.cplusplus.com/reference/thread