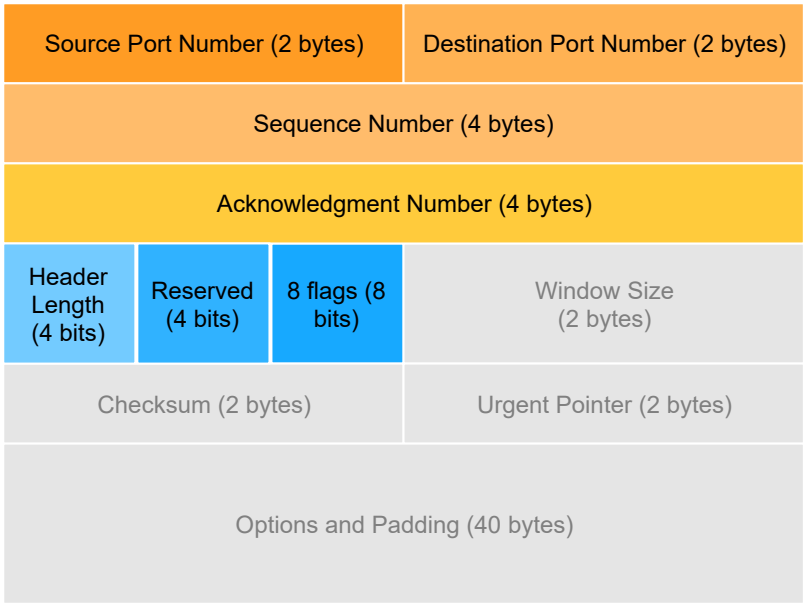
TCP Header Flags

In the last lesson, we discussed eight fields of the TCP header. Let's now discuss the last few!

WE'LL COVER THE FOLLOWING

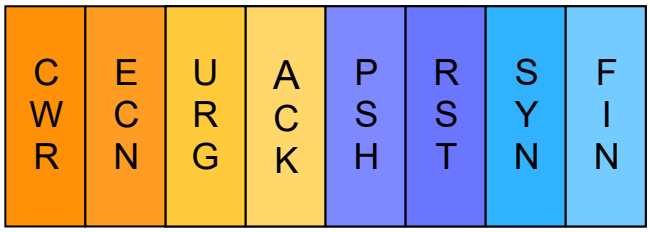
TCP headers have eight 1-bit flags that are imperative to signaling in the protocol.



Each 1-bit flag is crucial to make the protocol work.

Flags

Let's have a quick look at what each flag is meant for.



The eight flags used in TCP headers.

The first four discussed below, namely **ACK, RST, SYN,** and **FIN** are used in the establishment, maintenance, and tear-down of a TCP connection.

ACK

This flag is set to 1 in a segment to **acknowledge** a segment that was received previously. This is an important part of the protocol. In other words, when a receiver wants to acknowledge some received data, it sends a TCP segment with the ACK flag and the acknowledgment number field appropriately set. This flag is also used in connection establishment and termination as we will see in more detail later.

> **Note** that these acknowledgment messages can and often do contain data as well!

RST

The **reset** flag immediately terminates a connection. This is sent due to the result of some confusion, such as if the host doesn't recognize the connection, if the host has crashed, or if the host refuses an attempt to open a connection.

SYN

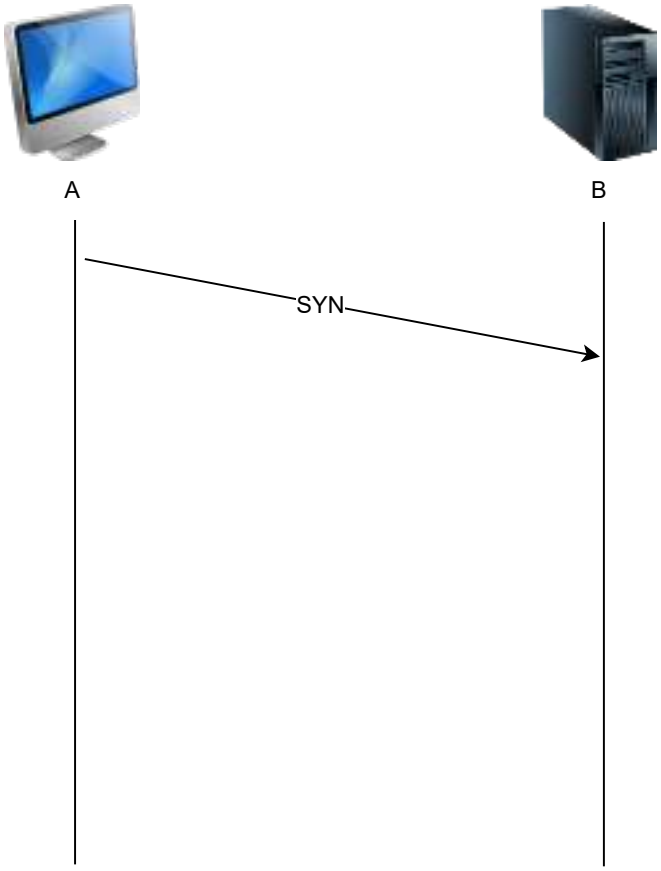
The **synchronization** flag initiates a connection establishment with a new host. The details will be covered later in the lesson on connection establishment.

FIN

This flag is used to terminate or **finish** a connection with a host.

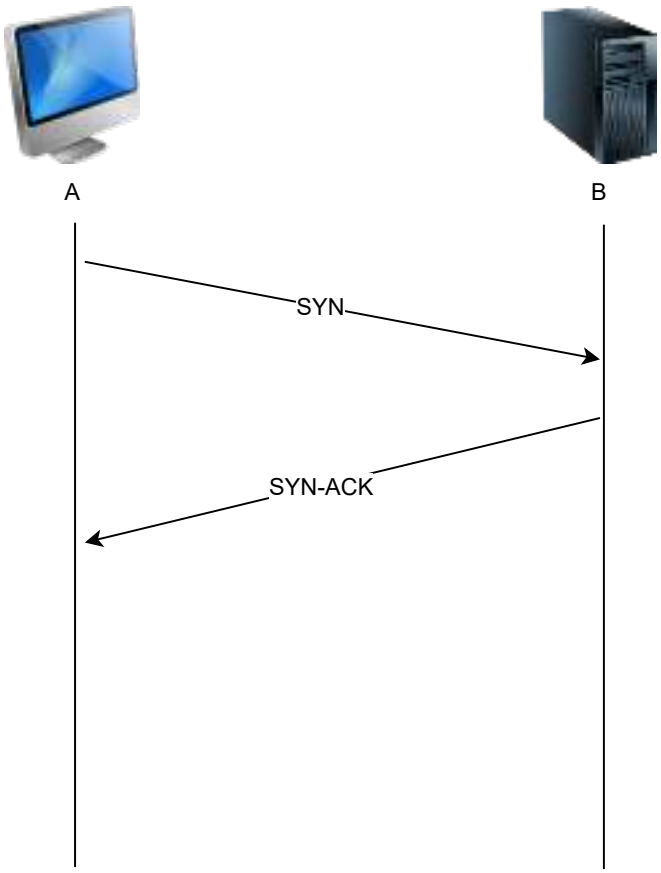
TCP Connection Establishment & Termination

The slides below give a very high level overview of how these flags are used to establish and terminate a TCP connection.

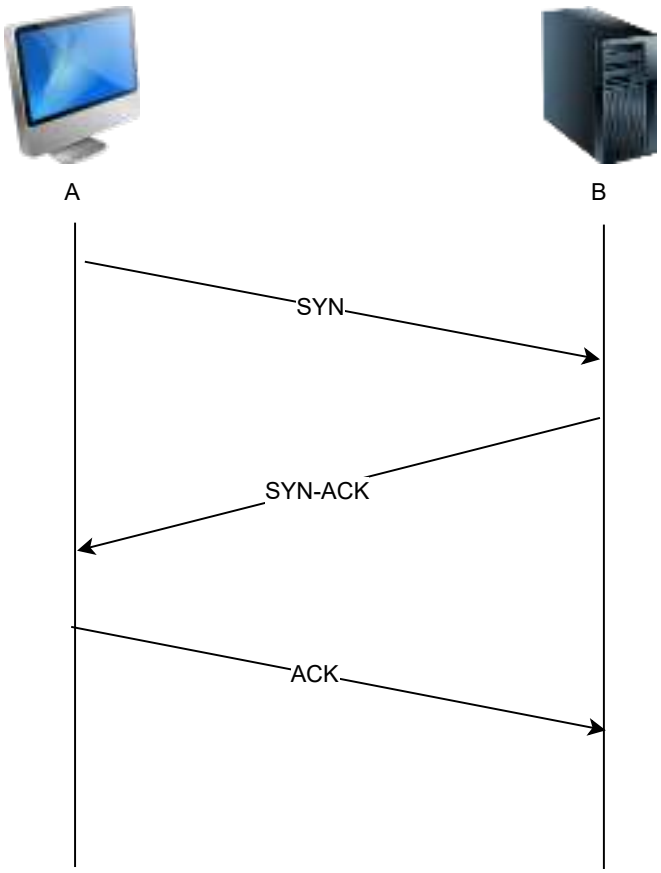


Host A wants to establish a connection with host B and so sends a segment with the SYN flag set

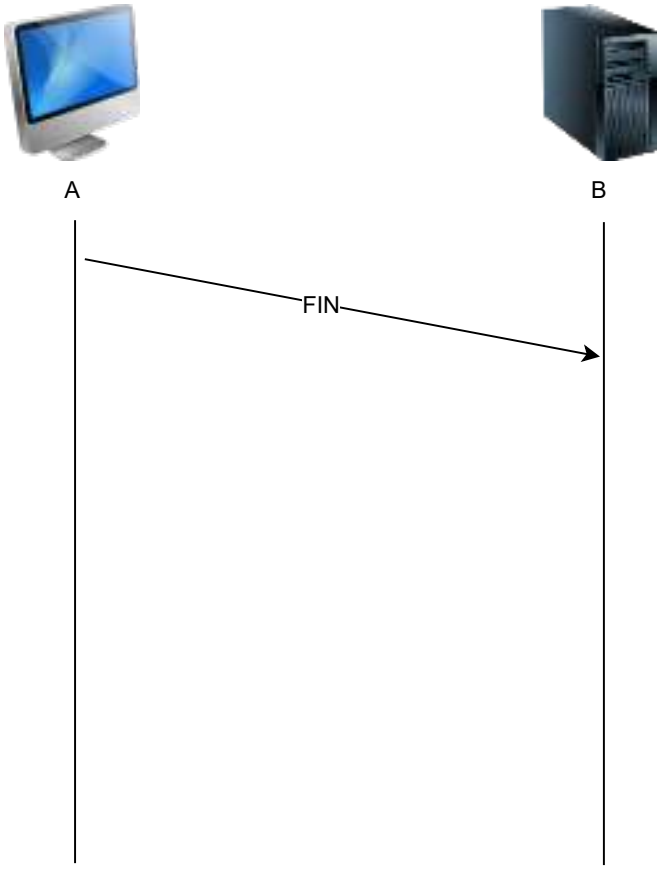
1 of 8



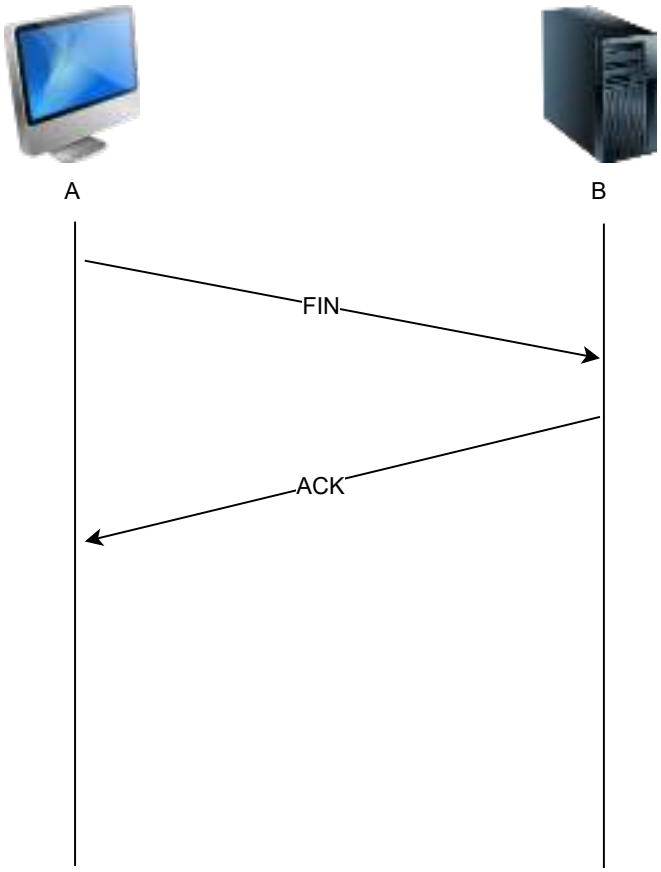
Host B acknowledges the SYN with a segment with an ACK flag along with a SYN flag of its own set in the same message



Host A acknowledges the SYN with a segment with an ACK flag

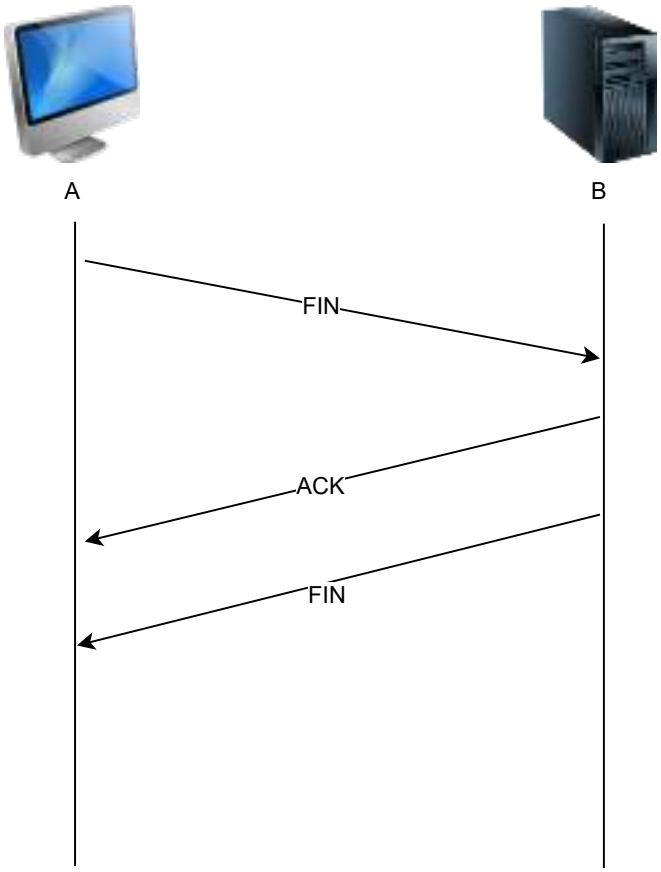


Host A wants to TERMINATE a connection with host B and so sends a segment with the FIN flag set to 1

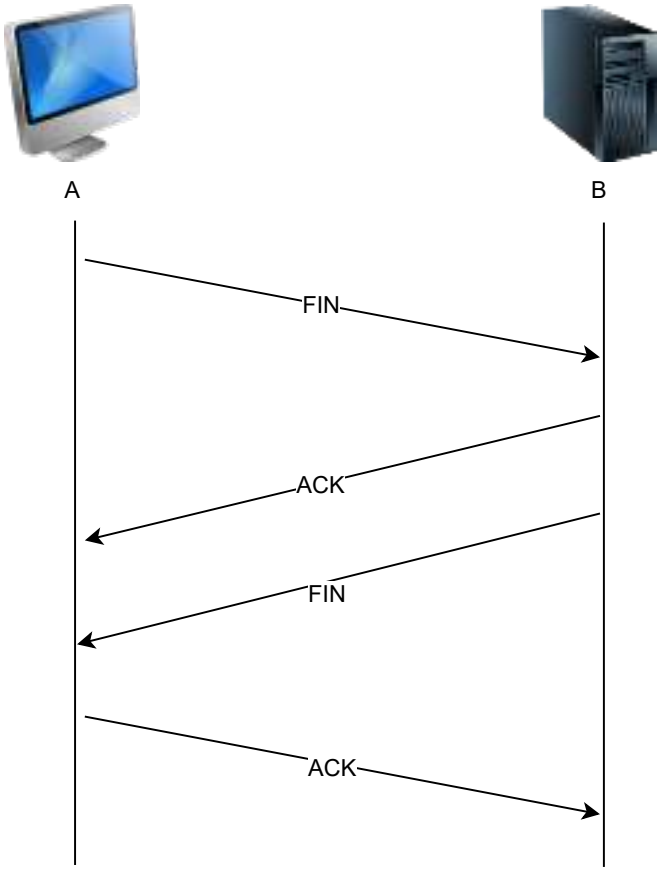


Host B acknowledges the FIN with a segment with an ACK flag

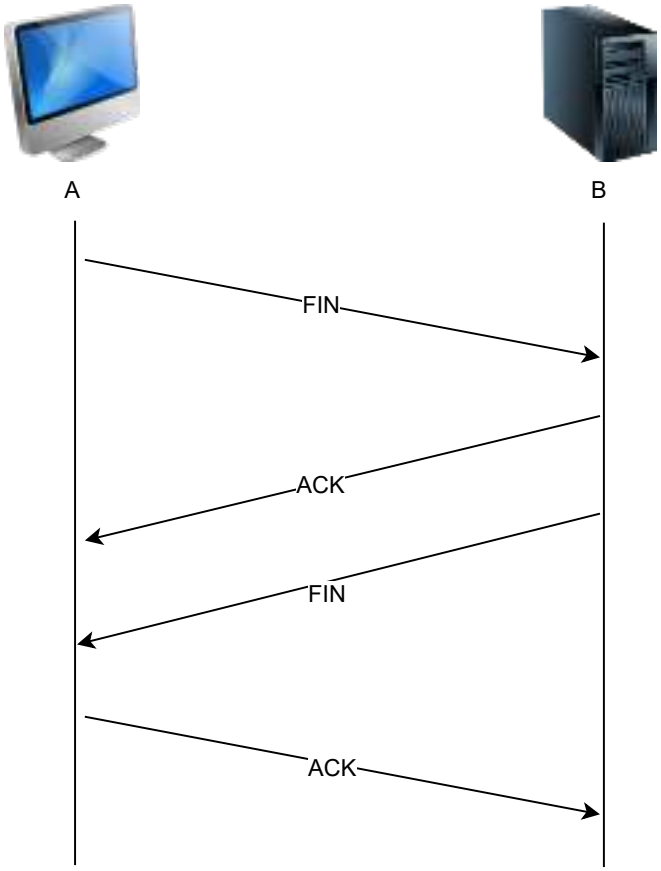
Host B acknowledges the FIN with a segment with an ACK flag



Host B sends a FIN flag



Host A acknowledges host B's FIN segment with an ACK



The connection is terminated

The rest of the flags, given below, are not very well-known. However, it doesn't hurt to know about them.

CWR & ECN

These flags, **Congestion Window Reduced** and **Explicit Congestion Notification** are used to handle congestion. To put it very simply, the ECN flag is set by the receiver, so that the sender knows that congestion is occurring. The sender sets the CWR flag in response to this so that the receiver knows that the receiver has reduced its congestion window to compensate for congestion and the sender is sending data at a slower rate.

PSH

The default behavior of TCP is in the interest of efficiency; if multiple small TCP segments were received, the receiving TCP will combine them before

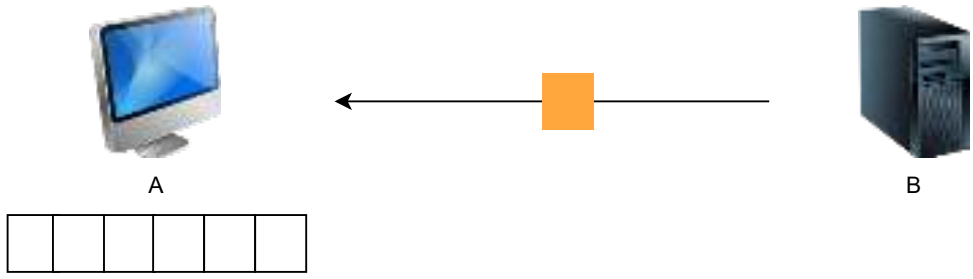
handing them over to the application layer. However, when the Push (PSH)

flag is set, the receiving end immediately flushes the data from its buffer to the application instead of waiting for the rest of it to arrive.

This is usually used for applications like Telnet, where every keystroke is a command. It would not make sense to say, buffer 50 keystrokes and send them to the application layer at once, so, every keystroke is pushed.

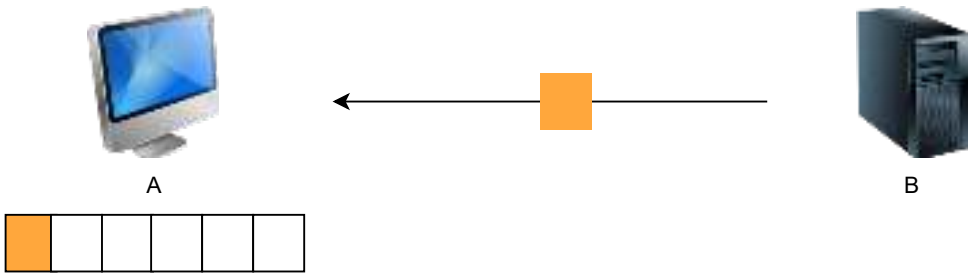
Host A has a buffer for the data it receives from host B

1 of 9

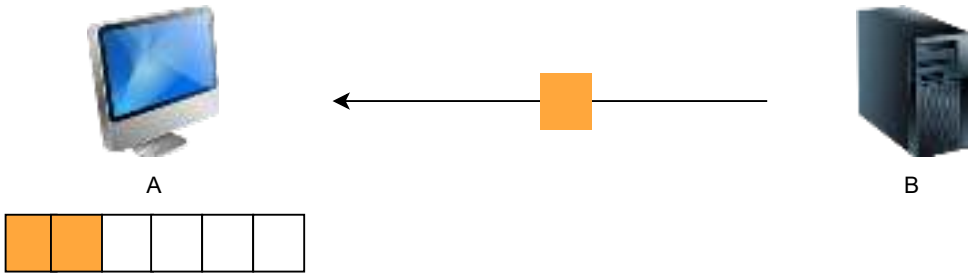


Host A will 'flush' that data out to the application when it is full

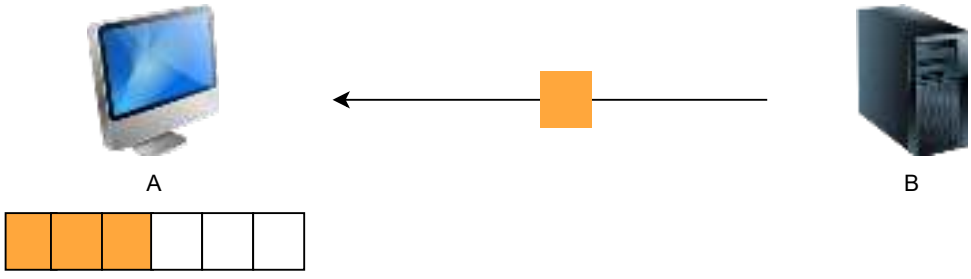
2 of 9



Host A receives a TCP segment from host B

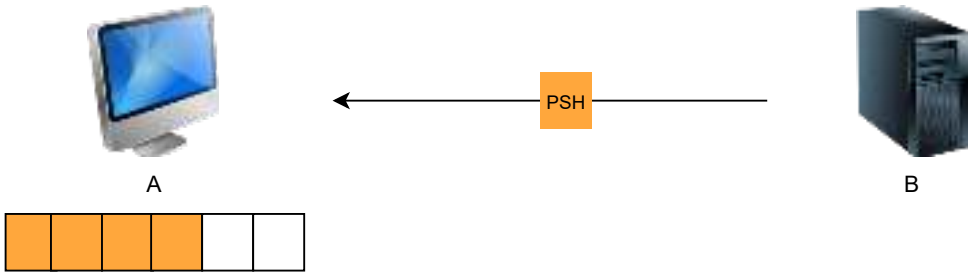


Host A receives a TCP segment from host B



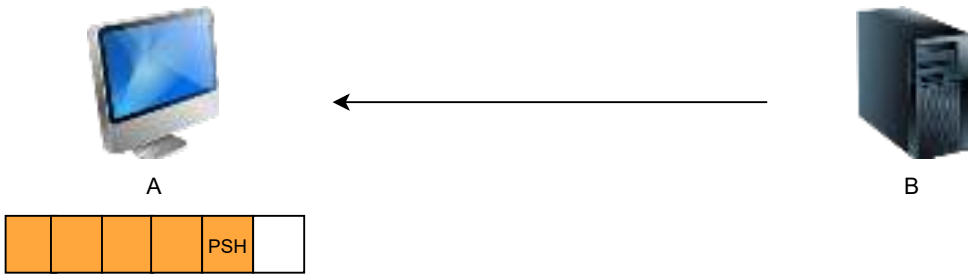
Host A receives a TCP segment from host B

5 of 9



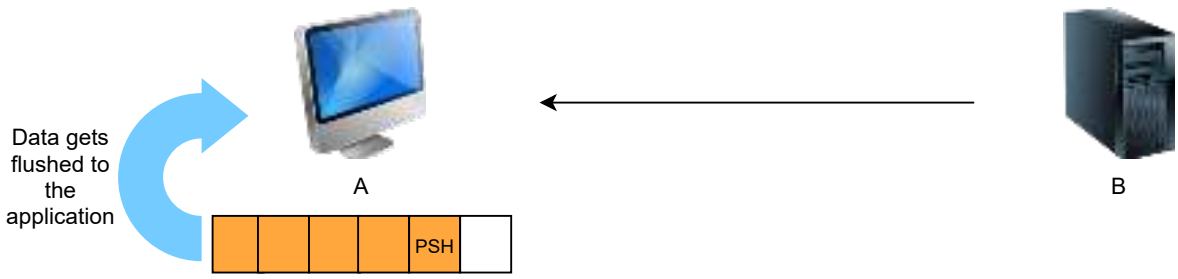
Host B sends a segment with the PSH flag on

6 of 9



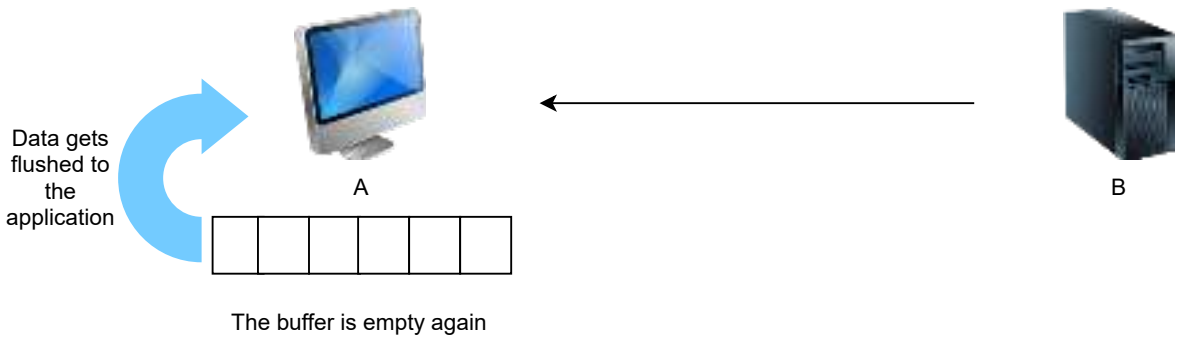
Host A receives said segment

7 of 9



and flushes the buffer out to the application even though the buffer is not full yet

8 of 9



The buffer gets flushed, is empty and is ready to receive data again

9 of 9

URG

The **Urgent** flag marks some data within a message as urgent. Upon receipt of an urgent segment, the receiving host forwards the urgent data to the application with an indication that the data is marked as urgent by the sender. The rest of the data in the segment is processed normally.

This would be used when suppose a large file is being transferred but the

This would be used when suppose a large file is being transferred but the sender realizes that it's the wrong file and sends a command to stop transfer.

It wouldn't make sense to have the file finish transferring first, hence the command to stop transfer is marked as **urgent** and is executed before the file is done transferring.

Quick Quiz!

1 What functionality does the urgent flag allow that the push flag does not?

COMPLETED 0%

1 of 3

Let's finish off looking at the rest of the headers in the next lesson!