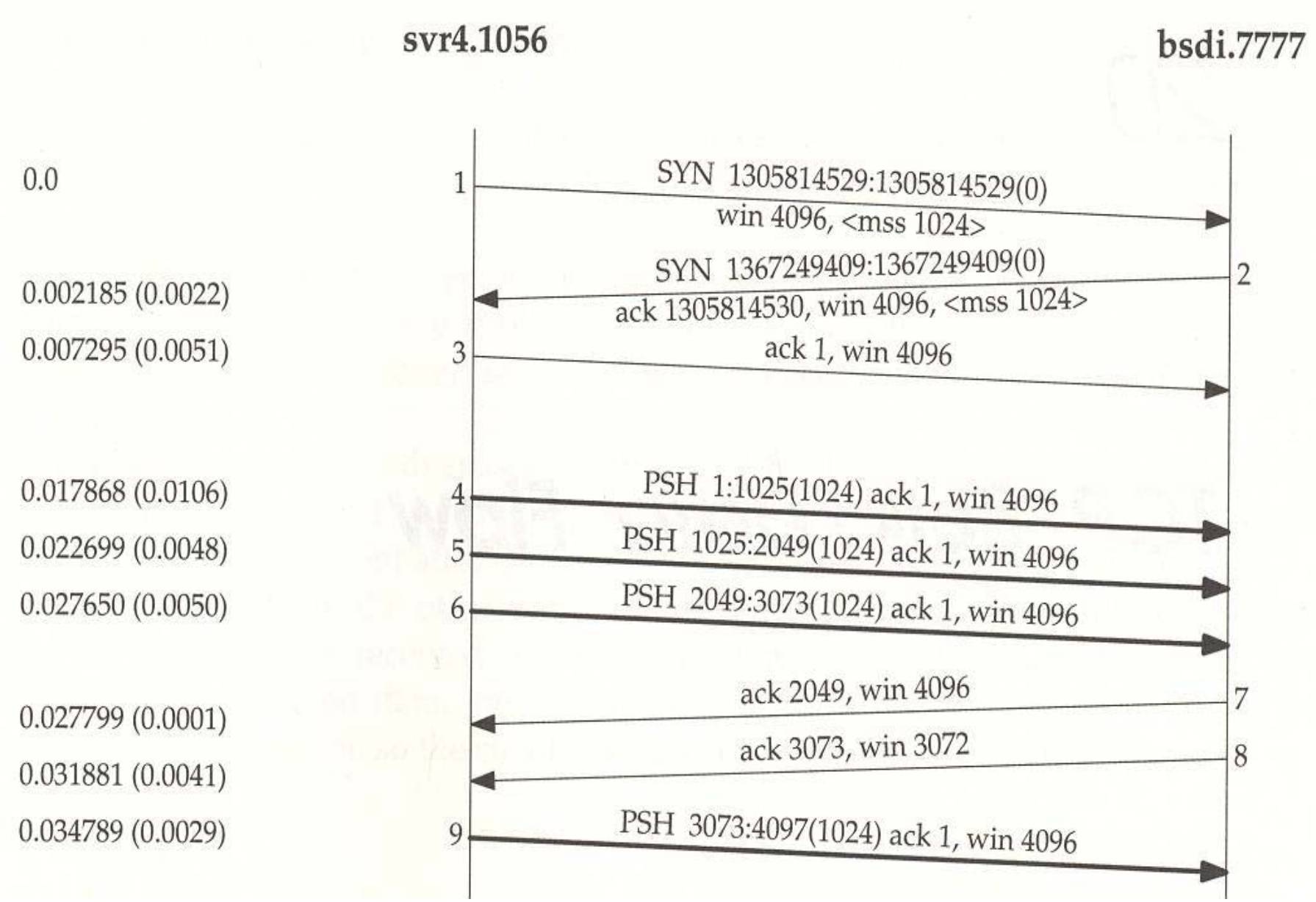
Topics

- TCP sliding window protocol
- TCP PUSH flag
- TCP slow start
- Bulk data throughput

Introduction

- In this chapter we will discuss TCP's form of flow control called a *sliding window* protocol
- It allows the sender to transmit multiple packets before it stops and waits for an ACK
- The leads to faster data transfer – sender does not have to wait for an ACK after each packet
- We will also look at the TCP PUSH flag and slow start
- Slow start is the technique used by TCP for getting the flow of data established
- Finally, we will examine bulk data throughput

Normal Data Flow



Normal Data Flow (Continued)

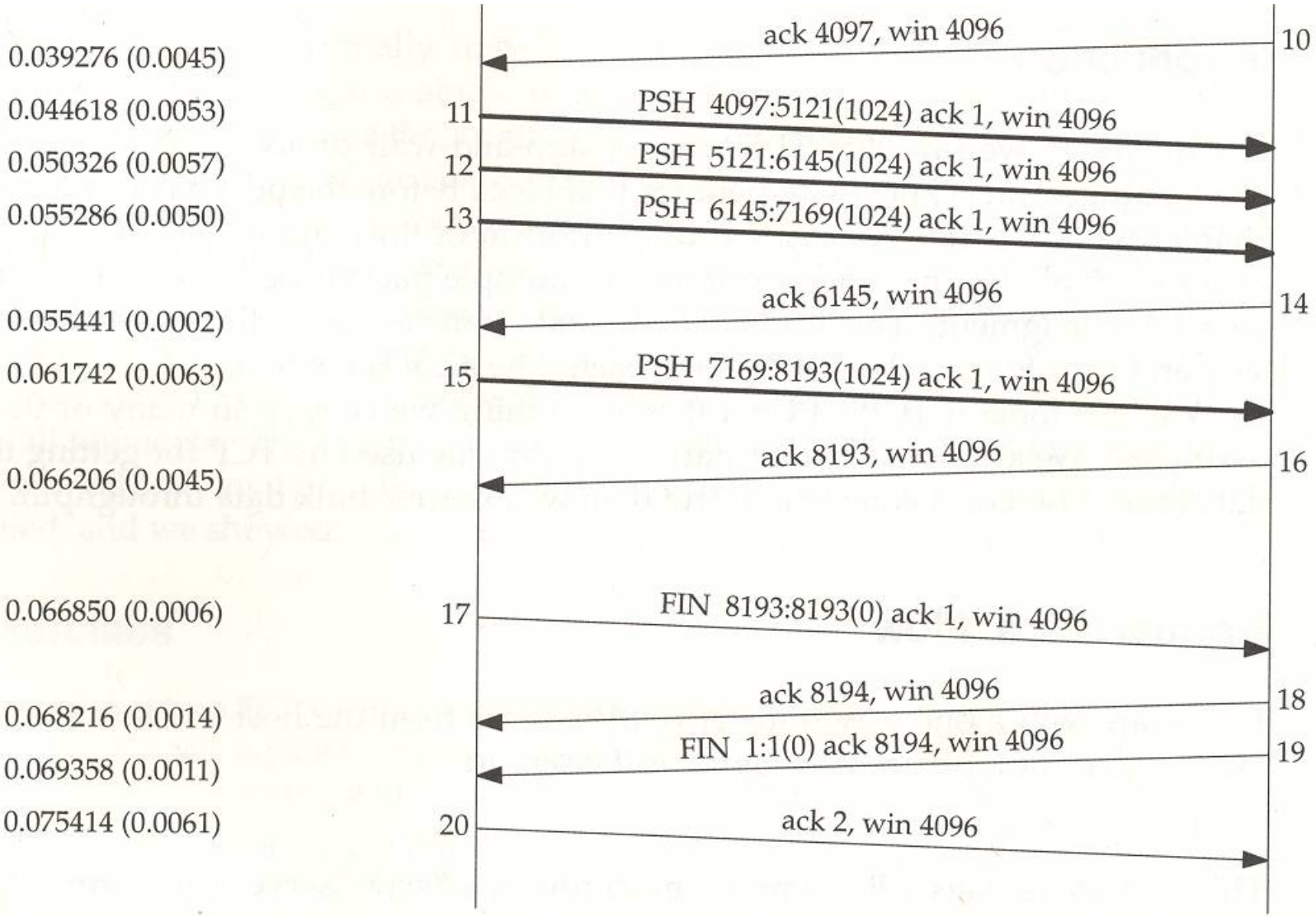


Figure 20.1 Transfer of 8192 bytes from `svr4` to `bsdi`.

Normal Data Flow (Continued)

- Normal Data Flow Example Transfer of 8192 bytes of data from srv4 to bsdi

  *Warning: This example appears to not use slow start (discussed later)

- Sender transmits 3 data segments (4-6)
- Next segment (7) acknowledges the first two data segments only. This is because of the following:
  - ❑ When TCP processes segment 4 the connection is marked to generate a delayed ACK
  - ❑ When segment 5 arrives TCP has two outstanding segments and immediately acknowledges
- TCP Immediately ACK's two outstanding segments

Normal Data Flow (Continued)

- Next segment (8) ACK's the third data segment due to ACK acknowledgement timer reaching a “ 200 ms Interval”
- Window of only 3072 advertised since 1024 bytes of data still in TCP receive buffer
- In TCP ACK's are cumulative. They acknowledge receipt of up through ACK sequence number minus one
- Note in Fig. 20.1 that the FIN segment has no data and is 8193:8193(0). The ACK for this is an ACK8194 which is one more than the last data. This is always true. A FIN uses up one ACK number

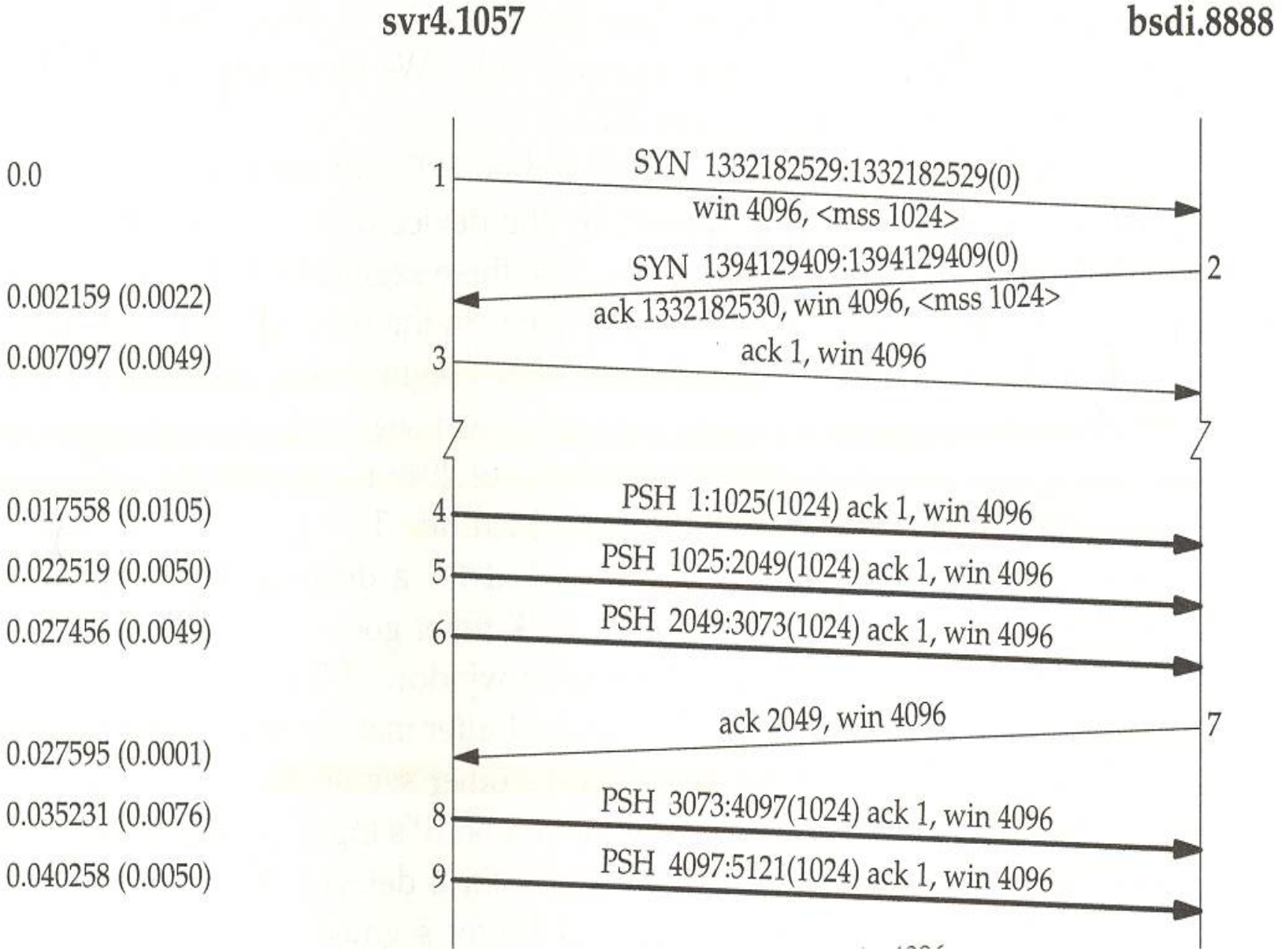
Normal Data Flow (Continued)

Another Normal Data Flow Example:

[Fig 20.2] Same as before but data sent a bit different

Warning: This example appears to not use slow start (discussed later)

Normal Data Flow (Continued)



Normal Data Flow (Continued)

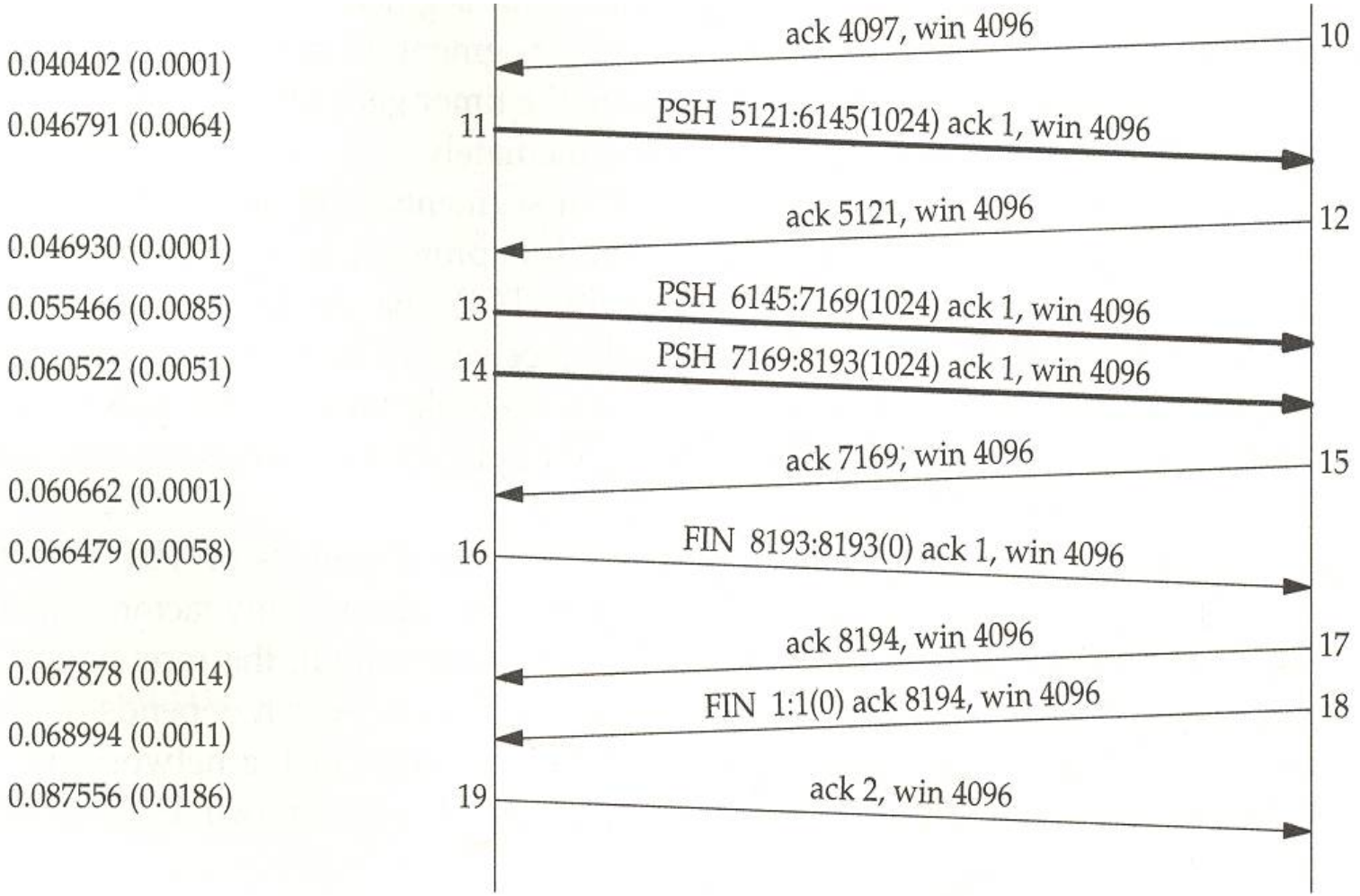


Figure 20.2 Another transfer of 8192 bytes from `svr4` to `bsdi`.

Normal Data Flow (Continued)

Fast Sender, Slow Receiver Example:

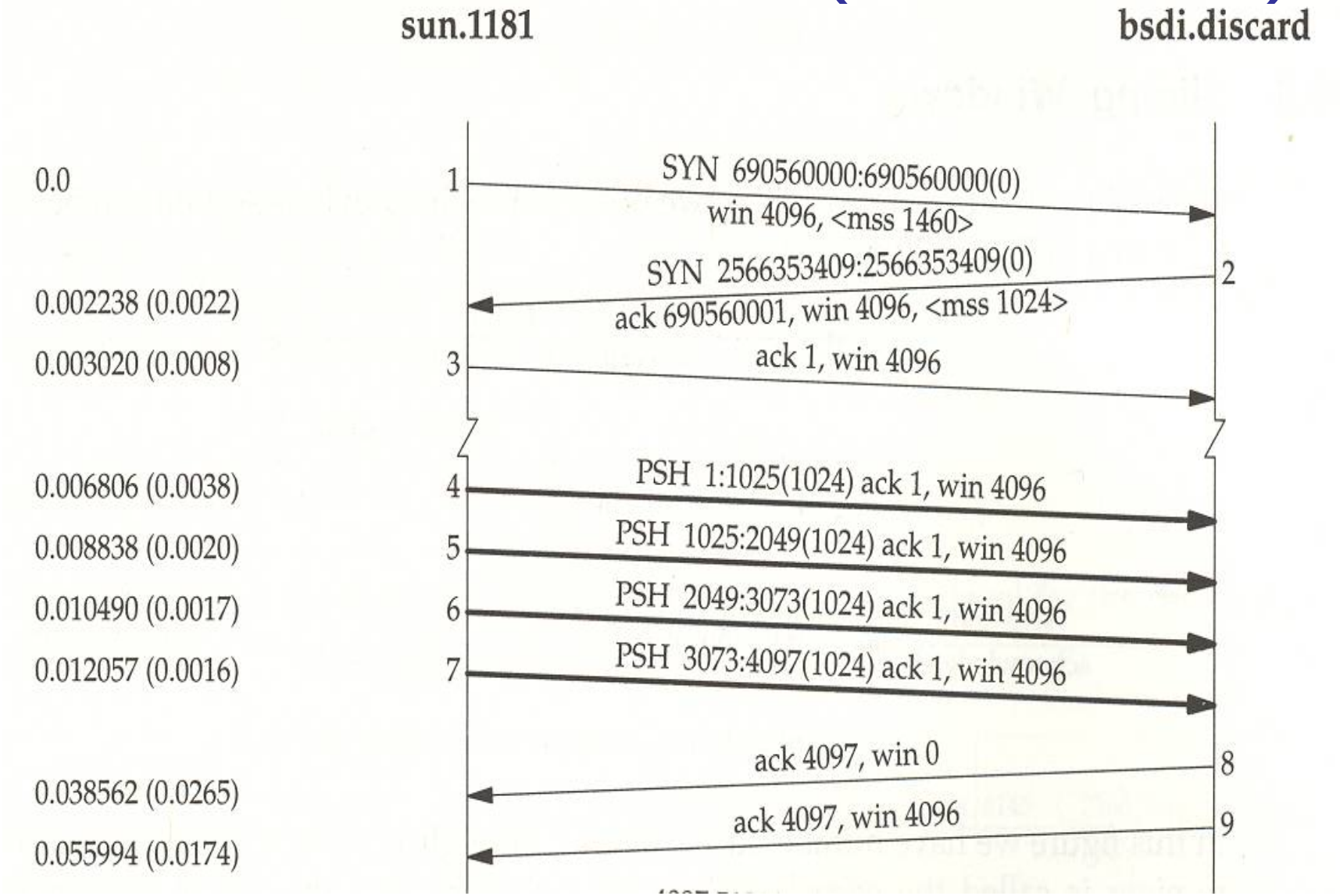
[Fig 20.3] Warning: This example appears to not use slow start (discussed later)

- Sender transmits four back-to-back data segments (4-7) to fill receiver's window

- Receiver sends ACK but with advertised window 0. Application has not yet read data

- Another ACK called a window update is sent later announcing that it can now receive another 4096 bytes

Normal Data Flow (Continued)



Normal Data Flow (Continued)

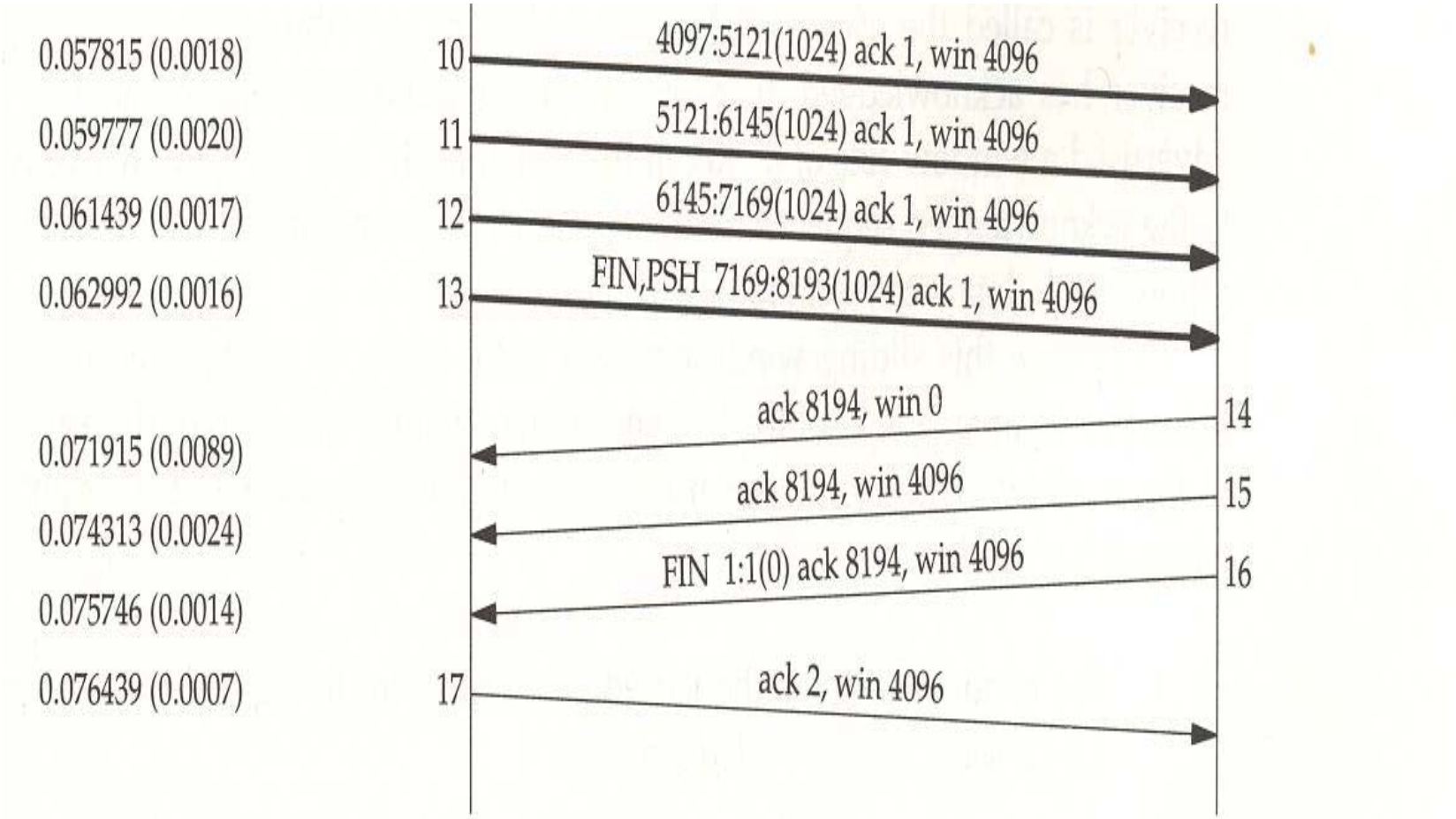


Figure 20.3 Sending 8192 bytes from a fast sender to a slow receiver.

Sliding Windows

The sliding window protocol that we observed in the previous section can be visualized as shown in Figure 20.4.

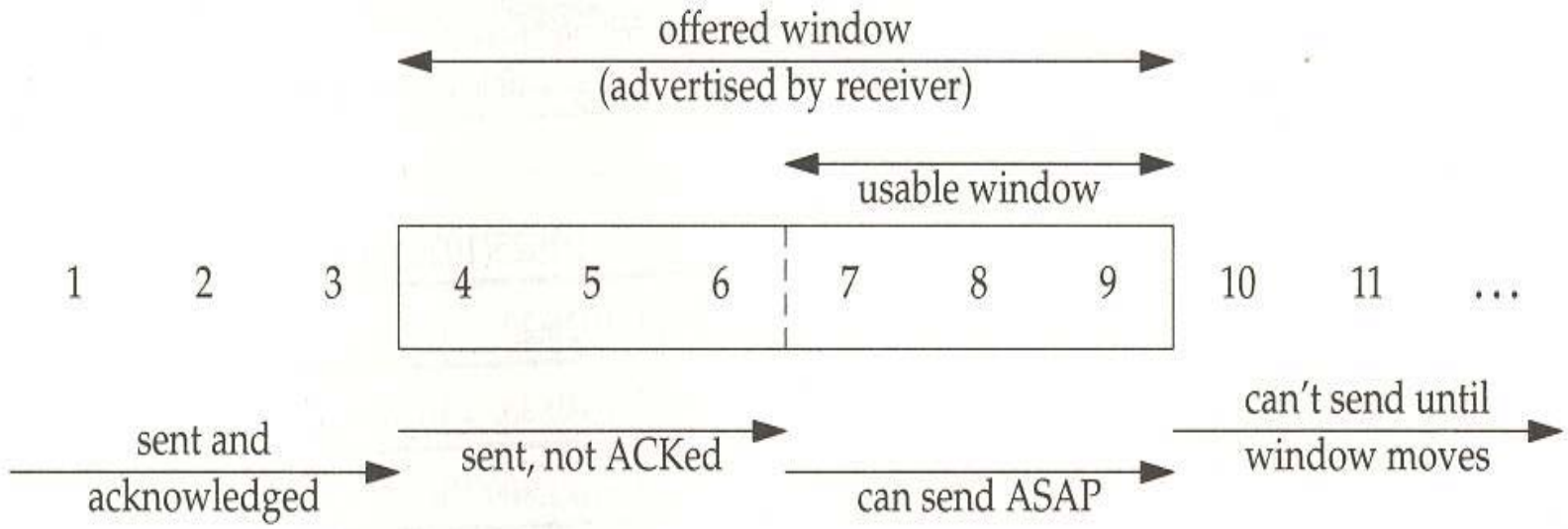


Figure 20.4 Visualization of TCP sliding window.

Sliding Windows (Continued)

Figure 20.6 shows the dynamics of TCP's sliding window protocol for the data transfer in Figure 20.1.

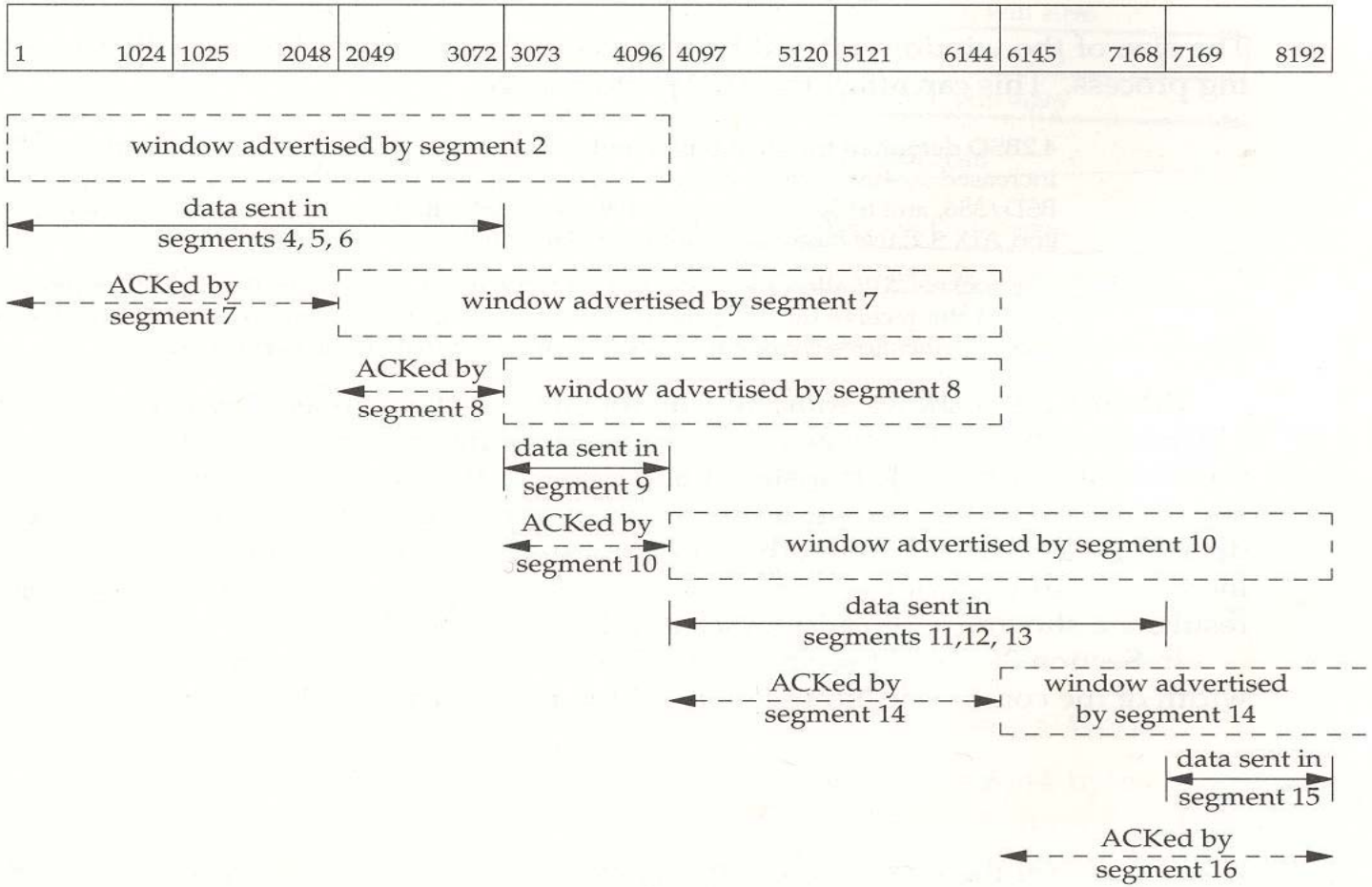
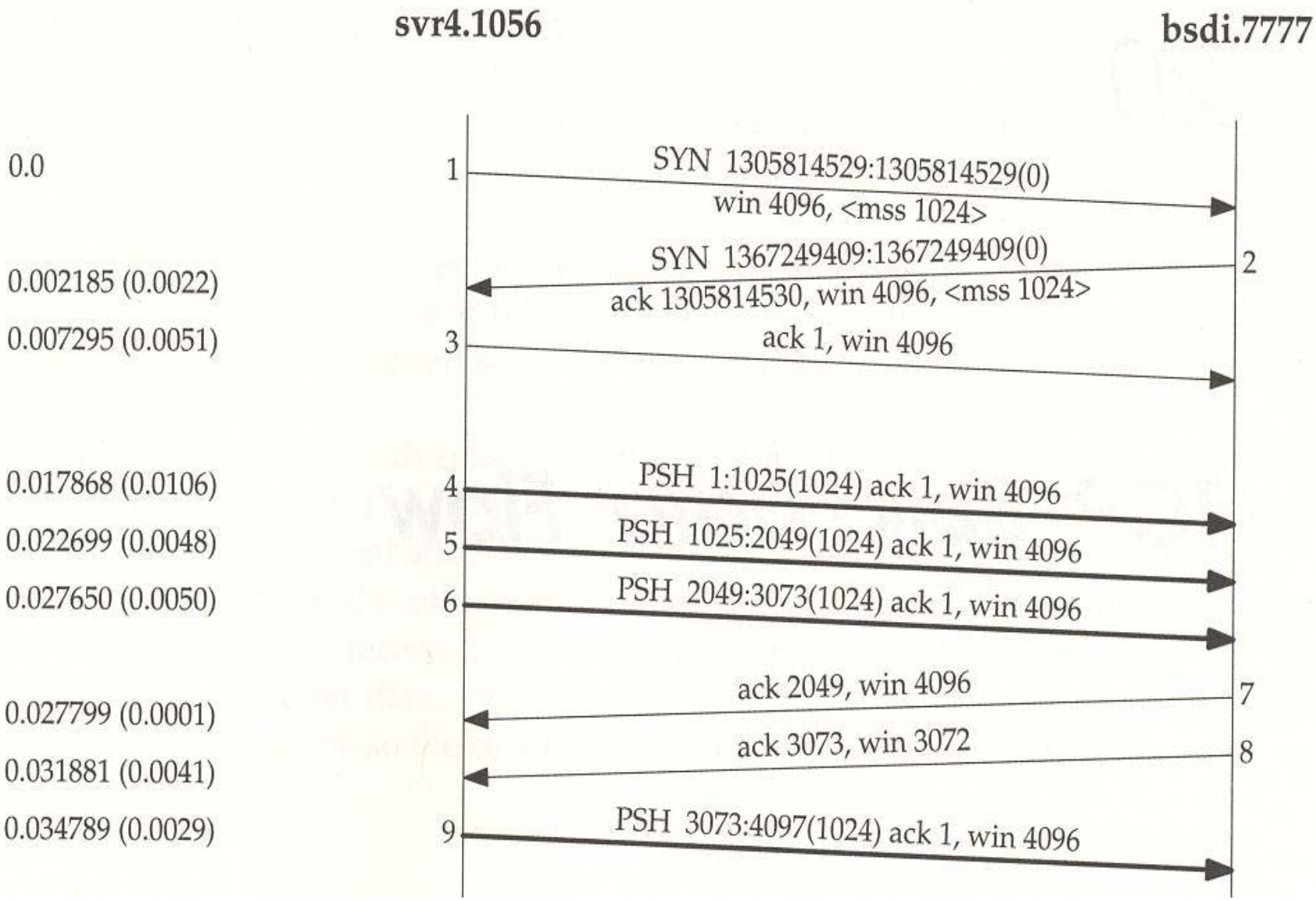


Figure 20.6 Sliding window protocol for Figure 20.1.

Sliding Windows (Continued)



Sliding Windows (Continued)

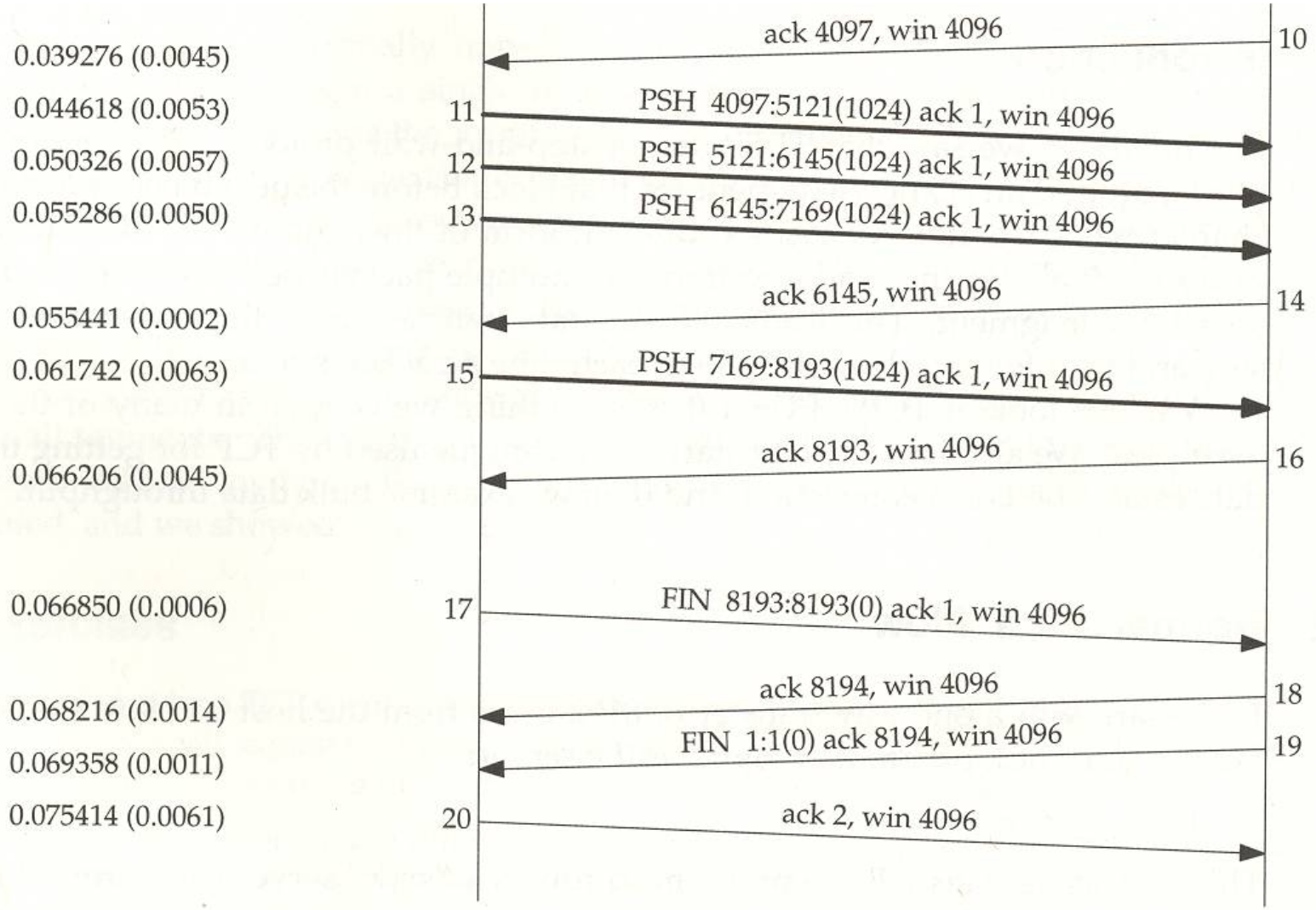


Figure 20.1 Transfer of 8192 bytes from `svr4` to `bsdi`.

PUSH Flag

- A notification from the sender to the receiver to pass all the data the receiver has to the receiving application
- Some implementations of TCP automatically set the push flag if the data in the segment being sent empties the send buffer
- These same implementations ignore a received PUSH flag because they normally deliver the received data to the application as soon as possible
- In Fig 20.3 missing 3 PSH because sends did not empty the send buffer. Application has already sent all of its data to the send buffer so only last write empties the send buffer
- We cannot manually set this flag through the Sockets Application Programming Interface

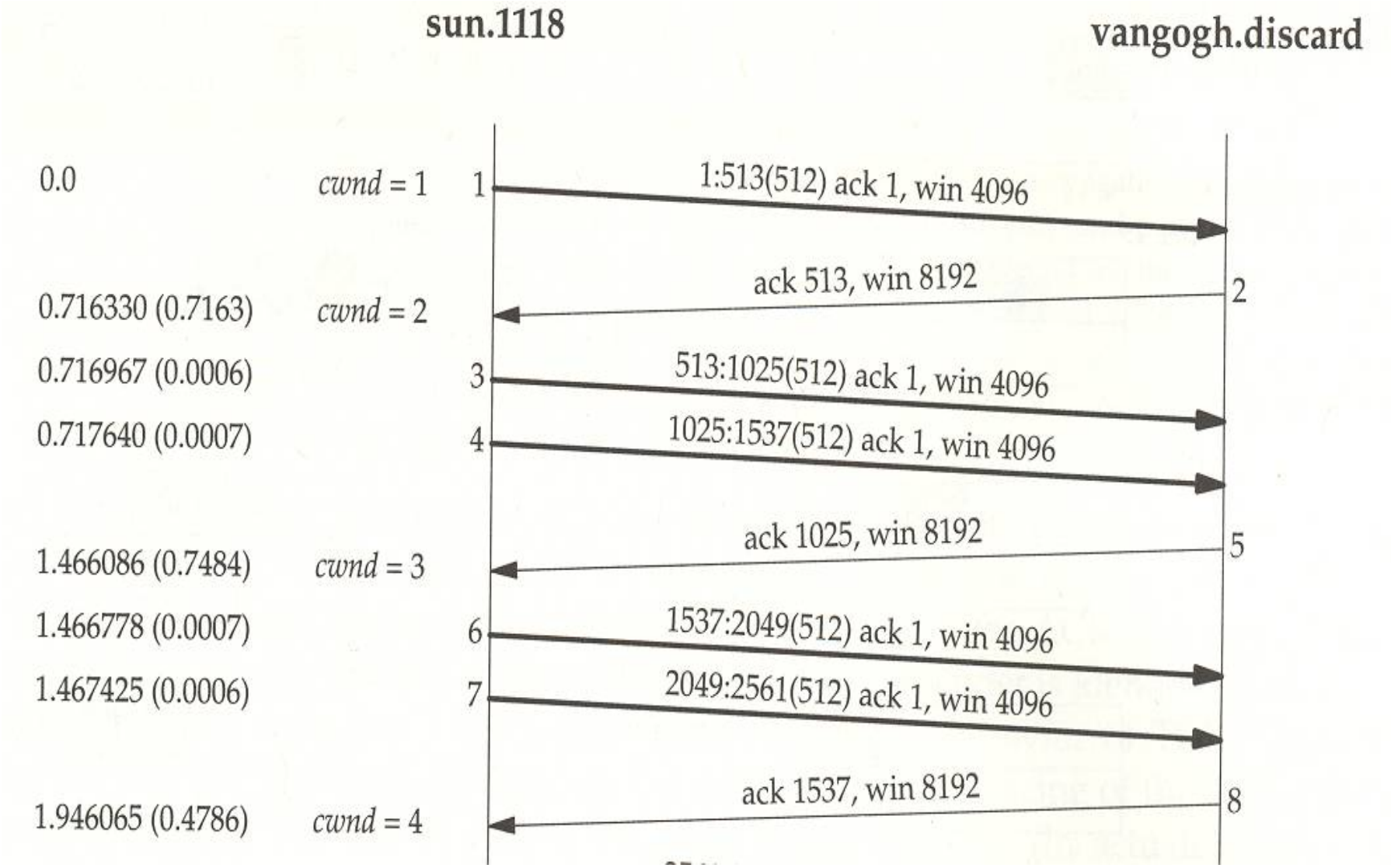
Slow Start

- Steven's examples did not use slow start up until now
- All TCP implementations must now use the slow start algorithm!
- An Intermediate router queue may run out of space
- Slow start algorithm notes that the rate it should inject new packets into the network is the rate at which acknowledgements are returned by the other end
- Add a congestion window " cwnd " on the SENDER side
- In a new connection, the congestion window is initialized to one MSS announced by other end
- cwnd is maintained in bytes however cwnd is incremented by segment size
- For each ACK received cwnd is increased by one segment

Slow Start (Continued)

- Sender can transmit up to minimum of (the congestion window and the advertised window)
(Congestion window “cwnd” is set by sender while advertised window is set by receiver)
- Sender starts by transmitting one segment and waiting for its ACK. When that ACK is received, congestion window is incremented from one to two and two segments can be sent
- When each of those two segments acknowledged, congestion window set to four
- This provides an exponential increase
- At some point an intermediate router will discard packets. Congestion window “cwnd” is too large

Slow Start (Continued)



Slow Start (Continued)

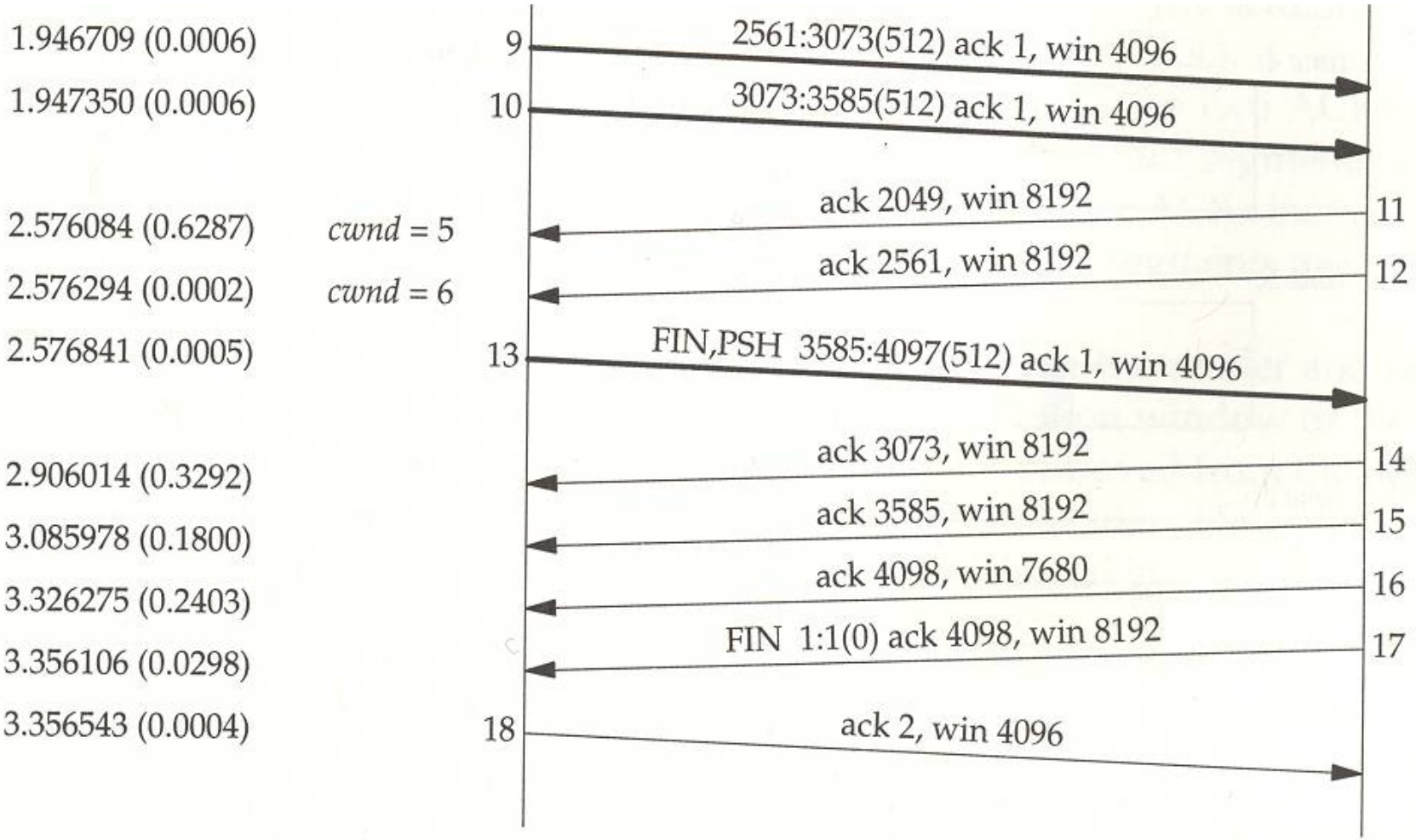


Figure 20.8 Example of slow start.

Slow Start (Continued)

Slow Start Example

[Fig 20.8] MSS=512

- Bulk data throughput - interaction of window size, windowed flow control, and slow start on the throughput of a TCP connection

[Fig 20.9& 20.10]
- At time 0 sender transmits one segment cwnd = 1, must wait for ACK
- At times 1, 2, 3, segment moves one time unit right
- At time 4 ACK generated
- At time 7 ACK received at sender
- At time 8 sender can transmit two segments, cwnd = 2 (we have round trip time = 8)
- At times 12, 13 ACK 2 and ACK 3 generated
- At time 15, 16 ACK's RCVD and with cwnd = 4 sender transmits four segments
- At time 24 and on can always transmit

Bulk Data Throughput

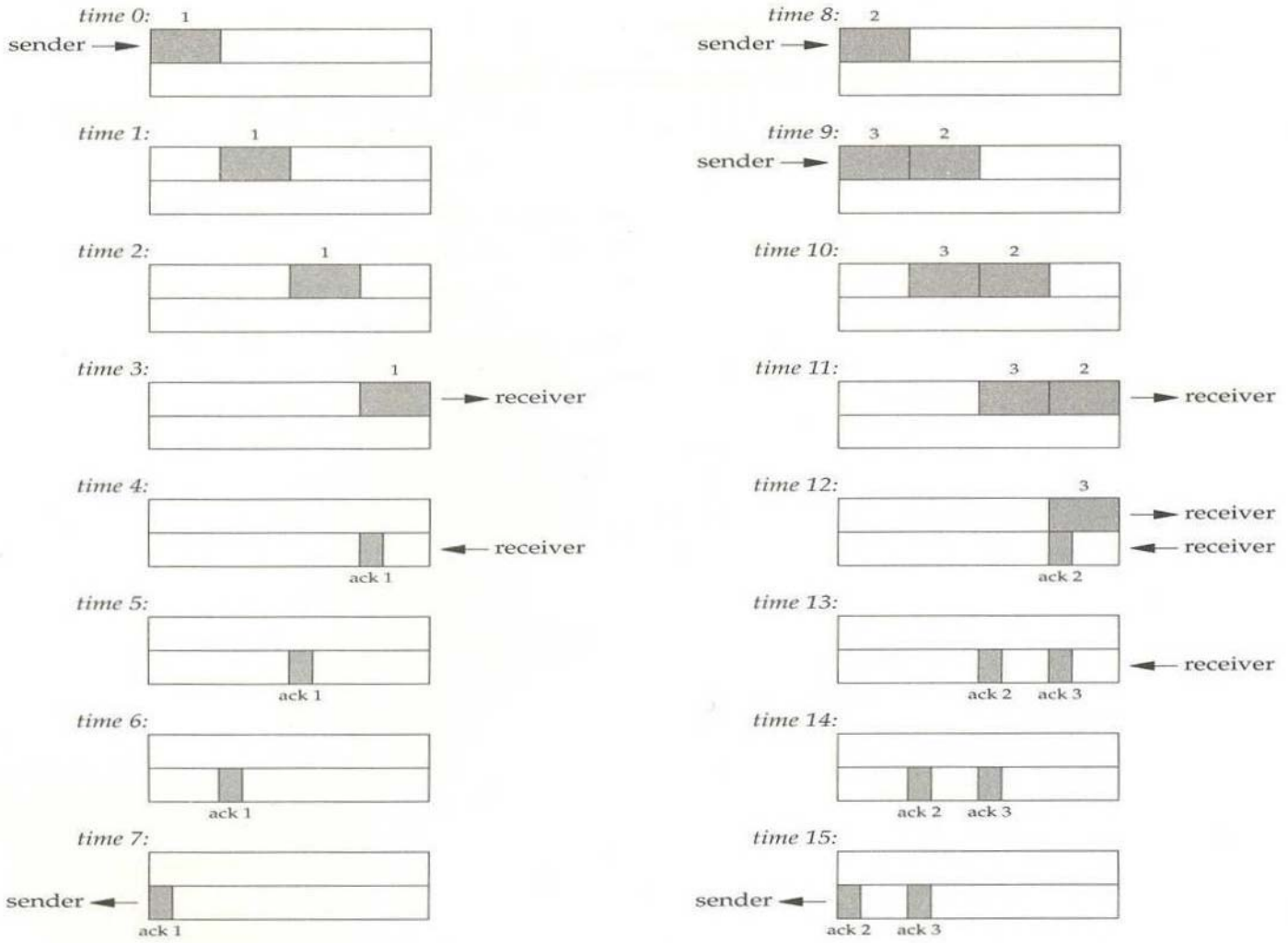


Figure 20.9 Times 0–15 for bulk data throughput example.

Bulk Data Throughput (Continued)

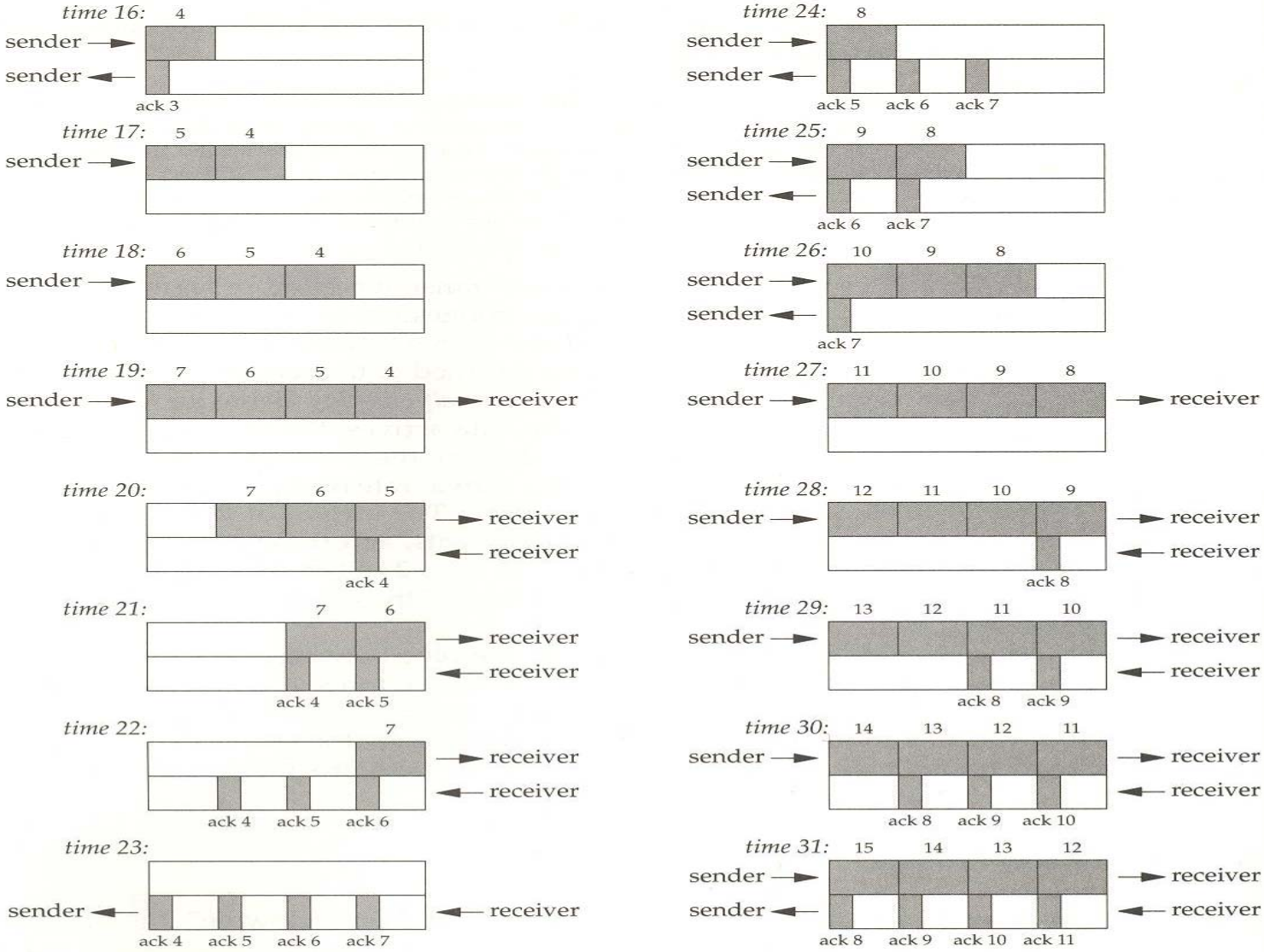


Figure 20.10 Times 16–31 for bulk data throughput example.

Bulk Data Throughput (Continued)

Bandwidth - Delay Product

• In previous example sender needs to have 8 segments outstanding and unacknowledged for max throughput. Thus receivers advertised window must be at least that large so as not to limit throughput

• Capacity of pipe = bandwidth (bits/ sec) x round trip time (sec) (also know as bandwidth-delay product)

-How does increasing the RTT affect the connection?

Example:

What size should receivers advertised window be for a T1 Cross USA country phone line?

= 1, 544, 000 Bits / Sec x 0.060 Sec round trip time

= 11,580 byte window

Bulk Data Throughput (Continued)

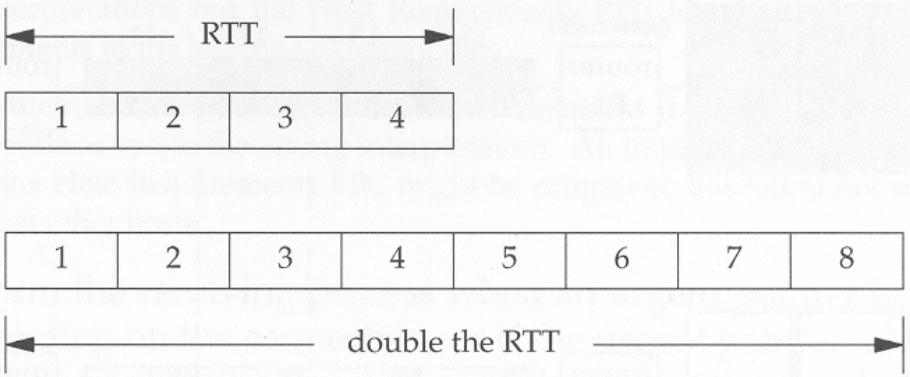


Figure 20.11 Doubling the RTT doubles the capacity of the pipe.

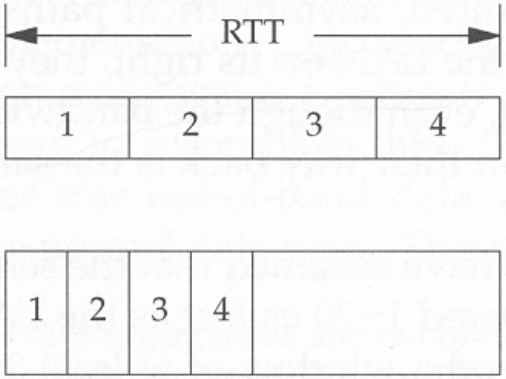


Figure 20.12 Doubling the bandwidth doubles the capacity of the pipe.

Bulk Data Throughput (Continued)

Congestion

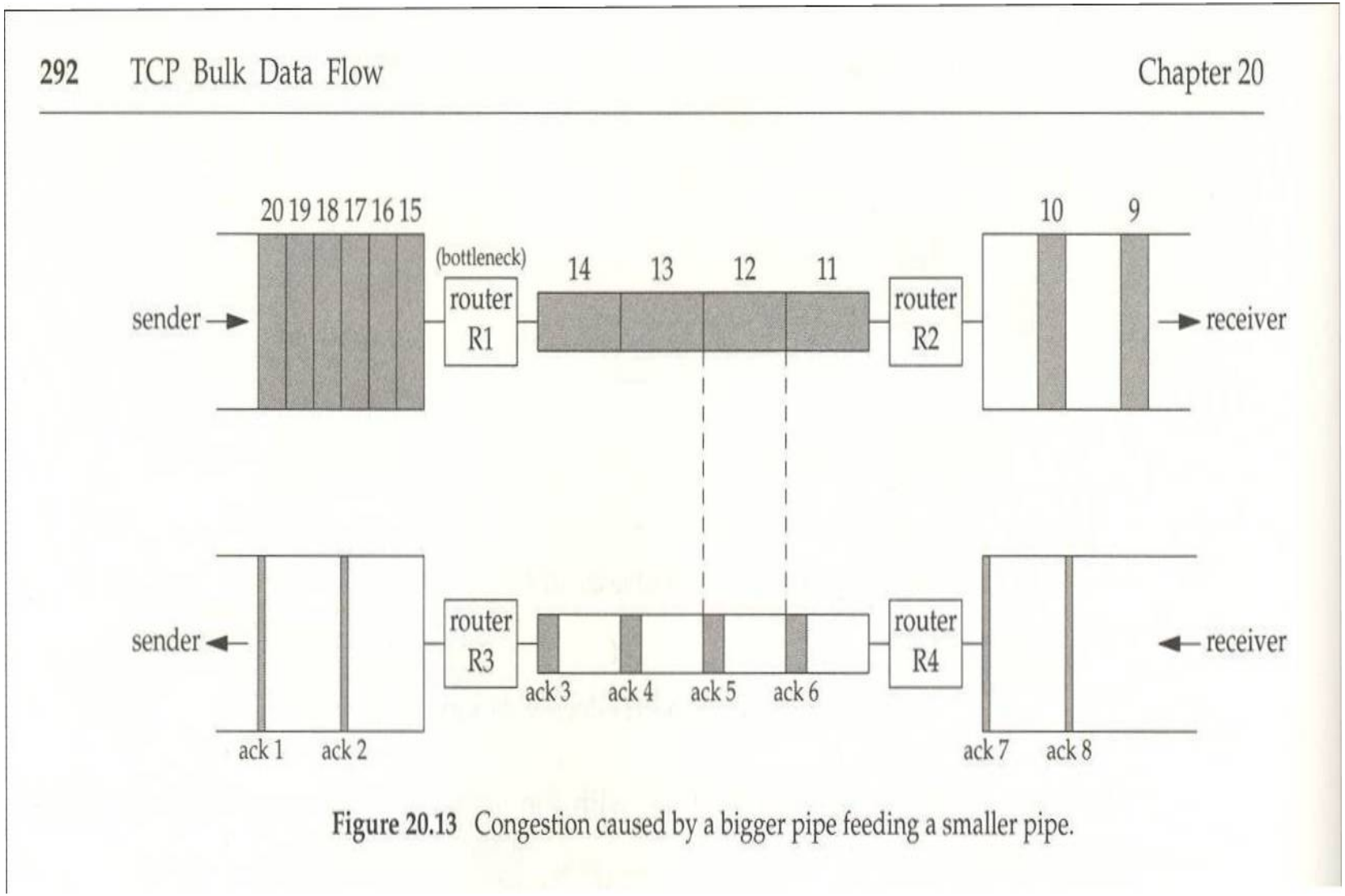


Figure 20.13 Congestion caused by a bigger pipe feeding a smaller pipe.

Bulk Data Throughput (Continued)

Congestion

• Can occur when data arrives on a big pipe (i.e., fast LAN) and gets sent out a smaller pipe (a slower WAN)

• Can also occur when multiple input streams arrive at a router whose output capacity is less than the sum of the inputs

• The spacing of the ACKs will correspond to the bandwidth of the slowest links