



Global Knowledge™

Expert Reference Series of White Papers

# Top 10 Reasons Why TCP Is Reliable

# Top 10 Reasons Why TCP Is Reliable

Ted Rohling, Global Knowledge Instructor, CISSP



## Introduction

The protocol used on the Internet for reliable communication is the Transmission Control Protocol (TCP). In this paper, you will learn some of the essentials of TCP by looking at 10 of the top features that make it reliable.

In-depth knowledge of protocols can help you in many ways. It enables more efficient troubleshooting. It allows you to understand how firewalls and routers limit the flow of dangerous information in and out of your networks. Our discussion in this white paper is a good introduction, but it does not represent a full, detailed analysis of TCP.

## 1. Mature Protocol

Whether or not you agree with the idea, the Transmission Control Protocol (TCP) is reliable because it is a mature protocol. Problems that exist in newer protocols have all been ironed out in the long-standing protocol TCP. TCP is based on Request for Comment (RFC) 793. An RFC is an Internet document that describes processes and procedures recognized by the Internet Engineering Task Force. RFC 793, released as a standard in 1981, is the standard that describes how TCP operates. TCP features the following 9 capabilities that lead to its true reliability.

Protocol captures are being included in the white paper to illustrate the reliability issues discussed. These captures were created with Omnipeek Personal, which is a product from WildPackets, Inc.

## 2. Orderly Session Startup Process

TCP begins a session by going through a three-step startup process. In step 1, the client sends a special request called a Synchronize to the service with which it is attempting to connect. In step 2, the service sends an Acknowledgement back to the client informing the client that the attempt to connect has been received. The service also sends a Synchronize request with the Acknowledgement. The Synchronize sent by the service tells that client that the service is also ready to connect to send information to the client, if necessary. Finally, the third step is an Acknowledgement message sent to the service from the client to let the service know that the client is ready to accept information from the service, if it is sent. Once all three steps have been completed, the session between the client and the service is established.

### Step 1.

In the captured packet below, which shows the start of a FTP session between a client and a service, you will see that the TCP Flags area contains a 1 in the SYN flag indicating the first step of the three-step "handshake."

```

TCP - Transport Control ProtocolSource Port: 46139
Destination Port: 21 ftp
Sequence Number: 3213898066
Ack Number: 0
TCP Offset: 10 (40 bytes)
Reserved: %0000
TCP Flags: %00000010 .....S.
           0... .... (No Congestion Window Reduction)
           .0.. .... (No ECN-Echo)
           ..0. .... (No Urgent pointer)
           ...0 .... (No Ack)
           .... 0... (No Push)
           .... .0.. (No Reset)
           .... ..1. SYN
           .... ...0 (No FIN)

Window: 5840
TCP Checksum: 0x3ABA
Urgent Pointer: 0

```

### Step 2.

In the captured packet below, you will see that the TCP Flags area contains a 1 in the SYN flag and a 1 in the Ack flag indicating the second step of the three-step handshake.

```

TCP - Transport Control Protocol
Source Port: 21 ftp
Destination Port: 46139
Sequence Number: 2062083365
Ack Number: 3213898067
TCP Offset: 11 (44 bytes)
Reserved: %0000
TCP Flags: %00010010 ...A..S.
           0... .... (No Congestion Window Reduction)
           .0.. .... (No ECN-Echo)
           ..0. .... (No Urgent pointer)
           ...1 .... Ack
           .... 0... (No Push)
           .... .0.. (No Reset)
           .... ..1. SYN
           .... ...0 (No FIN)

Window: 65535
TCP Checksum: 0xAD48
Urgent Pointer: 0

```

### Step 3.

In the captured packet below, you will see that the TCP Flags area contains a 1 in the Ack flag indicating the third step of the three-step handshake.

### TCP - Transport Control Protocol

```
Source Port:      46139
Destination Port: 21 ftp
Sequence Number:  3213898067
Ack Number:       2062083366
TCP Offset:       8 (32 bytes)
Reserved:         %0000
TCP Flags:        %00010000 ...A....
                  0... .. (No Congestion Window Reduction)
                  .0.. .. (No ECN-Echo)
                  ..0. .. (No Urgent pointer)
                  ...1 .. Ack
                  .... 0... (No Push)
                  .... .0.. (No Reset)
                  .... ..0. (No SYN)
                  .... ...0 (No FIN)

Window:          1460
TCP Checksum:    0x1A7D
Urgent Pointer:  0
```

The session between the client and the service is now established. Actually, it is really two sessions, one from the client to the service and one from the service to the client. This allows full duplex data transmission to occur.

### 3. Full Duplex Data Transmission.

The fact that TCP allows both the client and the service to send information at the same time was a major departure from protocol in use at the time TCP was created. Full duplex transmission actually speeds up the process of communicating. Rather than waiting for a client or service to acknowledge transmission, senders are at will to acknowledge received data at the same time they transmit information.

Applications submit data to TCP for transmission. Often the data is submitted as large blocks of information as in files, HTML pages or images. As TCP transmits data, it creates segments from the larger blocks. Segments are ideally sized to match the physical network being used to transport the data. More about segments later in the paper.

### 4. Accounting for Information Transfer.

In the examples below, a session has been established between an email client and an email service. To account for information being transmitted between client and service, TCP uses sequence numbers. Two different sequence numbers are established for transmission from the client to the service and from the service to the client.

In the next segment, the email client is sending information to the email service. The sequence number shows the value representing the first byte of the email message in the segment.

### TCP - Transport Control Protocol

```
Source Port:      1254 de-noc
Destination Port: 25 smtp
Sequence Number:  2173105768
```

```

Ack Number:          2247743657
TCP Offset:        5 (20 bytes)
Reserved:          %0000
TCP Flags:         %00010000 ...A....
                        0... .. (No Congestion Window Reduction)
                        .0.. .. (No ECN-Echo)
                        ..0. .. (No Urgent pointer)
                        ...1 .. Ack
                        .... 0... (No Push)
                        .... .0.. (No Reset)
                        .... ..0. (No SYN)
                        .... ...0 (No FIN)

Window:            65273
TCP Checksum:      0x895B
Urgent Pointer:    0

```

A second packet or segment was immediately sent to the service by the client. The sequence number shows the value representing the first byte of the email message in the second packet. This sequence number can be used to show how much information was in the last segment.

#### TCP - Transport Control Protocol

```

Source Port:       1254 de-noc
Destination Port: 25 smtp
Sequence Number:  2173107228
Ack Number:       2247743657
TCP Offset:       5 (20 bytes)
Reserved:         %0000
TCP Flags:        %00011000 ...AP...
                        0... .. (No Congestion Window Reduction)
                        .0.. .. (No ECN-Echo)
                        ..0. .. (No Urgent pointer)
                        ...1 .. Ack
                        .... 1... Push
                        .... .0.. (No Reset)
                        .... ..0. (No SYN)
                        .... ...0 (No FIN)

Window:           65273
TCP Checksum:     0x877D
Urgent Pointer:   0
No TCP Options

```

```

Sequence number frame 2          2173107228
Sequence number frame 1          2173105768
Difference                        1460

```

By subtracting the sequence number of the first segment from the sequence number of the second segment, we can calculate the number of bytes found in the first segment. In this case, the TCP payload or data being carried by TCP is 1460 bytes.

The sequence number also allows TCP to recorder segments that are received out of order. By re-arranging the received segments based on their sequence numbers, TCP provides properly sequenced information to the application.

## 5. Acknowledgements

Continuing our example of an email being sent from a client to a service, the packet below shows how the service acknowledges the receipt of the data from the client. The TCP session supporting the email service sends an acknowledgement packet to the client. Note that the TCP Flags area contains a 1 in the Ack field. This tells the client that the information previously sent has been received by the service.

The Ack Number field contains a pointer to the most recently received information by the service.

### TCP - Transport Control Protocol

```
Source Port:      25 smtp
Destination Port: 1254 de-noc
Sequence Number:  2247743657
Ack Number:       2173108210
TCP Offset:       5 (20 bytes)
Reserved:         %0000
TCP Flags:        %00010000 ...A....
                  0... .. (No Congestion Window Reduction)
                  .0.. .. (No ECN-Echo)
                  ..0. .. (No Urgent pointer)
                  ...1 .. Ack
                  .... 0... (No Push)
                  .... .0.. (No Reset)
                  .... ..0. (No SYN)
                  .... ...0 (No FIN)

Window:          65535
TCP Checksum:    0x472C
Urgent Pointer:  0
No TCP Options
```

```
Ack Number          2173108210
Sequence number frame 1  2173105768
Bytes acknowledged    2432
```

By using the same subtraction process we used in discussing the sequence number accounting, we can calculate how much information has been acknowledged. The calculation shows that the last two frames contained 2432 bytes of data. In fact, the transmitted information was only 2431 bytes. The acknowledgement number indicates the "byte to be processed next" which is one number more than the last byte of the frame.

Since TCP is a full duplex protocol, the sequence and acknowledgement fields in the packets are used to account for data being sent from the client to the service and from the service to the client. Also, the sequence and acknowledgement fields are used for the entire term of the session. It is possible to determine how much data was sent in a session by capturing the first sequence numbers and subtracting them from the last acknowledgement numbers to compute the actual number of bytes in the data flow.

## 6. Retransmission of Missing Information

TCP retransmits missing information by using missing acknowledgements. If a TCP process sends data and receives no acknowledgement for the data, the process simply retransmits the information after a time-out period. That is, TCP waits until an acknowledgement should have arrived and then retransmits the sent data if the acknowledgement does not show up.

The acknowledgement may not arrive because the information transmitted did not make it to the destination or because the acknowledgement was lost on its trip to the original sender. In either case, there was no positive acknowledgement so the data is retransmitted.

## 7. Data Flow Management

Returning to the original example using an FTP session, we can discover how TCP manages data flow. The captured packet below contains a field labeled Window. The TCP window field is used to provide information about the session buffer space available on the sending process. The buffer is used to temporarily store data as it arrives and before the receiving client or service has time to process it.

### Step 1.

In the example below, the client is telling the service that 5840 bytes of buffer space is available for the service if and when it decides to send data to the client. The service is obliged to send 5840 bytes or less to the client. Otherwise, the client buffer will fill up and the session might terminate abnormally.

#### TCP - Transport Control Protocol

```
Source Port:      46139
Destination Port: 21 ftp
Sequence Number:  3213898066
Ack Number:       0
TCP Offset:       10 (40 bytes)
Reserved:         %0000
TCP Flags:        %00000010 .....S.
Window:           5840
TCP Checksum:     0x3ABA
Urgent Pointer:   0
```

### Step 2.

The Window field sent by the service tells the client that the entire receive buffer space is available for the client to send information to the service. The maximum window size is 65535. This field will change values as the session progresses and the buffer fills and empties.

#### TCP - Transport Control Protocol

```
Source Port:      21 ftp
Destination Port: 46139
Sequence Number:  2062083365
Ack Number:       3213898067
TCP Offset:       11 (44 bytes)
Reserved:         %0000
TCP Flags:        %00010010 ...A..S.
Window:           65535
TCP Checksum:     0xAD48
Urgent Pointer:   0
```

### Step 3.

In the third step, the Window field indicates that the client is telling the service that it has 1460 bytes of buffer space available. The service cannot transmit more than 1460 bytes to the client without overrunning the buffer and possibly causing the session to be terminated. The 1460 byte limit seen here is temporary and will change depending on many factors on the client.

#### TCP - Transport Control Protocol

```
Source Port:      46139
Destination Port: 21 ftp
Sequence Number:  3213898067
Ack Number:       2062083366
TCP Offset:       8 (32 bytes)
Reserved:         %0000
TCP Flags:        %00010000 ...A....
Window:           1460
TCP Checksum:     0x1A7D
Urgent Pointer:   0
```

During the normal flow of information between client and service, the Window values will change. If the Window value is zero, the transmission in one direction will stop until the window size increases, allowing the sending processes to communicate again. If the client or service is not taking data out of the buffer, the window size will stay at zero. This allows the communication to flow based on capability of the sender and receiver rather than the speed of the network.

## 8. Orderly Session Shutdown

At the end of the transmission process, it is necessary to terminate the session between the client and the service. TCP uses a four-step process to end the session.

Depending on the application, the client or the service will begin the termination process by sending a Fin or Final Data flag to the corresponding process signaling that the final data segment has been transmitted.

### Step 1.

#### TCP - Transport Control Protocol

```
Source Port:      25 smtp
Destination Port: 1254 de-noc
Sequence Number:  2247743800
Ack Number:       2173108221
TCP Offset:       5 (20 bytes)
Reserved:         %0000
TCP Flags:        %00010001 ...A...F
                  0... .. (No Congestion Window Reduction)
                  .0.. .. (No ECN-Echo)
                  ..0. .. (No Urgent pointer)
                  ...1 .... Ack
                  .... 0... (No Push)
                  .... .0.. (No Reset)
                  .... ..0. (No SYN)

      .... ...1 FIN
```

```
Window:                65524
TCP Checksum:          0x469C
Urgent Pointer:        0
No TCP Options
```

In the segment header above, the email service is signaling the client that the last data segment has been transmitted. Note the FIN flag is set to 1 in the TCP Flags area.

## Step 2.

### TCP - Transport Control Protocol

```
Source Port:           1254 de-noc
Destination Port:      25 smtp
Sequence Number:       2173108221
Ack Number:            2247743801
TCP Offset:            5 (20 bytes)
Reserved:              %0000
TCP Flags:             %00010000 ...A....
                       0... .. (No Congestion Window Reduction)
                       .0.. .. (No ECN-Echo)
                       ..0. .. (No Urgent pointer)
                       ...1 .. Ack
                       .... 0... (No Push)
                       .... .0.. (No Reset)
                       .... ..0. (No SYN)
                       .... ...0 (No FIN)

Window:                65130
TCP Checksum:          0x83A7
Urgent Pointer:        0
```

In step 2, email client acknowledges the FIN flag sent by the service. This ends the session from the service to the client.

## Step 3.

### TCP - Transport Control Protocol

```
Source Port:           1254 de-noc
Destination Port:      25 smtp
Sequence Number:       2173108221
Ack Number:            2247743801
TCP Offset:            5 (20 bytes)
Reserved:              %0000
TCP Flags:             %00010001 ...A...F
                       0... .. (No Congestion Window Reduction)
                       .0.. .. (No ECN-Echo)
                       ..0. .. (No Urgent pointer)
                       ...1 .. Ack
                       .... 0... (No Push)
                       .... .0.. (No Reset)
                       .... ..0. (No SYN)
                       .... ...1 FIN
```

**Window:** 65130  
**TCP Checksum:** 0x83A7  
**Urgent Pointer:** 0

In step 3, the email client announces to the service that no more data is to be sent. The FIN flag is sent by the client to the service.

#### Step 4.

##### TCP - Transport Control Protocol

**Source Port:** 25 smtp  
**Destination Port:** 1254 de-noc  
**Sequence Number:** 2247743801  
**Ack Number:** 2173108222  
**TCP Offset:** 5 (20 bytes)  
**Reserved:** %0000  
**TCP Flags:** %00010000 ...A....  
0... .. (No Congestion Window Reduction)  
.0.. .. (No ECN-Echo)  
..0. .... (No Urgent pointer)  
...1 .... Ack  
.... 0... (No Push)  
.... .0.. (No Reset)  
.... ..0. (No SYN)  
.... ...0 (No FIN)

**Window:** 65524  
**TCP Checksum:** 0x469B  
**Urgent Pointer:** 0

In step 4, email service acknowledges the FIN flag sent by the client. This ends the session from the client to the service. This step ends the connection between the client and the service.

## 9. time-out Process Based on Throughput

One of the more difficult design issues related to TCP was the determination of the time-out value. How long should a process wait for an Ack before it retransmits?

The designers of TCP realized that network conditions will vary and that a dynamic value was necessary. As a session is established, TCP calculates two values. The Round Trip Time (RTT) is determined by measuring how long it takes for a transmitted segment to be acknowledged. The second value calculated is the Smoothed Round Trip Time (SRTT). The SRTT is a moving average of the RTT over time. Over the length of the session, the RTT and the SRTT is adjusted as response times lengthen or are reduced due to network conditions.

The dynamic changes to the time-out values allow TCP to adapt itself to the constantly changing network environment.

## 10. Checksum for Data Quality Management.

The final reliability component to be discussed in this white paper is the TCP Checksum. When TCP prepares a segment for transmission, it calculates a checksum value. The value is based on the TCP segment payload, the

TCP segment header and the IP addresses of the two devices participating in the TCP session. The checksum is then inserted into the TCP header as seen below in the TCP Checksum field.

#### TCP - Transport Control Protocol

```
Source Port:      46139
Destination Port: 21 ftp
Sequence Number:  3213898067
Ack Number:       2062083366
TCP Offset:       8 (32 bytes)
Reserved:         %0000
TCP Flags:        %00010000 ...A....
Window:           1460
TCP Checksum:     0x1A7D
Urgent Pointer:   0
```

The TCP segment is transmitted to the target process. The receiving TCP process then repeats the TCP checksum calculation. If the calculated checksum does not match the checksum sent in the original segment, TCP assumes that the data in the segment was jumbled during transmission. The segment is discarded by the receiving process, as if it were never received. Bad data is never processed by TCP when the checksums do not match.

## Summary

The Transmission Control Protocol is one of the key Internet protocols. This white paper introduced 10 elements of TCP that contribute to the reliability of the protocol. We have just scratched the surface of the processes involved in a successful TCP session. More study is necessary to fully understand how the protocol operates.

## Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge. Check out the following Global Knowledge courses:

[Understanding Networking Fundamentals](#)  
[TCP/IP Networking](#)  
[Network+ Boot Camp](#)

For more information or to register, visit [www.globalknowledge.com](http://www.globalknowledge.com) or call **1-800-COURSES** to speak with a sales representative.

Our courses and enhanced, hands-on labs offer practical skills and tips that you can immediately put to use. Our expert instructors draw upon their experiences to help you understand key concepts and how to apply them to your specific work situation. Choose from our more than 700 courses, delivered through Classrooms, e-Learning, and On-site sessions, to meet your IT and management training needs.

## About the Author

Ted Rohling has been a contract instructor with Global Knowledge since 1995. With over 40 years of experience in information technology, telecommunications, and security, Ted teaches in the Networking and Security

product lines and focuses on TCP/IP, Networking Fundamentals and CISSP Preparation. He currently holds the CISSP certification and has previously held various certifications from Nortel, Cisco and Microsoft. His educational background includes a BBA in Management Science, and MA in Information and Computer Management, and an MS in Educational Human Resource Development.