**Wireshark Lab - TCP**

# Capturing a bulk TCP transfer from your computer to a remote server

Before beginning our exploration of TCP, we’ll need to use Wireshark to obtain a packet  
trace of the TCP transfer of a file from your computer to a remote server. You’ll do so by  
accessing a Web page that will allow you to enter the name of a file stored on your  
computer (which contains the ASCII text of Alice in Wonderland), and then transfer the  
file to a Web server using the HTTP POST method (see section 2.2.3 in the text). We’re  
using the POST method rather than the GET method as we’d like to transfer a large  
amount of data from your computer to another computer. Of course, we’ll be running  
Wireshark during this time to obtain the trace of the TCP segments sent and received  
from your computer.

Do the following:

• Start up your web browser. Go the http://gaia.cs.umass.edu/wireshark-  
labs/alice.txt and retrieve an ASCII copy of Alice in Wonderland. Store this file  
somewhere on your computer.  
• Next go to <http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html>.  
• You should see a screen that looks like:

Graphical user interface, text, application, email

Description automatically generated  
• Use the Browse button in this form to enter the name of the file (full path name)  
on your computer containing Alice in Wonderland (or do so manually). Don’t yet  
press the “Upload alice.txt file” button.  
• Now start up Wireshark and begin packet capture (Capture->Start) and then press  
OK on the Wireshark Packet Capture Options screen (we’ll not need to select any  
options here).  
• Returning to your browser, press the “Upload alice.txt file” button to upload the  
file to the gaia.cs.umass.edu server. Once the file has been uploaded, a short  
congratulations message will be displayed in your browser window.  
• Stop Wireshark packet capture. Your Wireshark window should look similar to  
the window shown below.

Graphical user interface, application, table, Excel

Description automatically generated

If you are unable to run Wireshark on a live network connection, you can download a  
packet trace file that was captured while following the steps above on one of the author’s  
computers2. You may well find it valuable to download this trace even if you’ve  
captured your own trace and use it, as well as your own trace, when you explore the  
questions below.

# A first look at the captured trace.

Before analyzing the behavior of the TCP connection in detail, let’s take a high level  
view of the trace.  
• First, filter the packets displayed in the Wireshark window by entering “tcp”  
(lowercase, no quotes, and don’t forget to press return after entering!) into the  
display filter specification window towards the top of the Wireshark window.  
What you should see is series of TCP and HTTP messages between your computer and gaia.cs.umass.edu. You should see the initial three-way handshake containing a SYN message. You should see an HTTP POST message. Depending on the version of Wireshark you are using, you might see a series of “HTTP Continuation” messages being sent from your computer to gaia.cs.umass.edu. Recall from our discussion in the earlier HTTP Wireshark lab, that is no such thing as an HTTP Continuation message – this is Wireshark’s way of indicating that there are multiple TCP segments being used to carry a single HTTP message. In more recent versions of Wireshark, you’ll see “[TCP segment of a reassembled PDU]” in the Info column of the Wireshark display to indicate that this TCP segment contained data that belonged to an upper layer protocol message (in our case here, HTTP). You should also see TCP ACK segments being returned from gaia.cs.umass.edu to your computer.

Answer the following questions, by opening the Wireshark captured packet file tcp-  
ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip (that is  
download the trace and open that trace in Wireshark; see footnote 2). Whenever possible,  
when answering a question, you should hand in a printout of the packet(s) within the trace  
that you used to answer the question asked. Annotate the printout3 to explain your  
answer. To print a packet, use File->Print, choose Selected packet only, choose Packet  
summary line, and select the minimum amount of packet detail that you need to answer  
the question.

## What is the IP address and TCP port number used by the client computer (source) that is transferring the file to gaia.cs.umass.edu? To answer this question, it’s probably easiest to select an HTTP message and explore the details of the TCP packet used to carry this HTTP message, using the “details of the selected packet header window” (refer to Figure 2 in the “Getting Started with Wireshark” Lab if you’re uncertain about the Wireshark windows.

**IP address: 192.168.1.89 (source IP)**

**TCP port number: 53665 (source), 80 (destination)**

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

## What is the IP address of gaia.cs.umass.edu? On what port number is it sending and receiving TCP segments for this connection?

**gaia.cs.umass.edu IP address: 128.119.245.12**

**Sending data on the below port:**

**TCP port number: 80**

Graphical user interface, text, application, email

Description automatically generated

**Receiving data on below address and port:**

**TCP port number: 80**

Graphical user interface, text, application

Description automatically generated

## What is the IP address and TCP port number used by your client’s computer (source) to transfer the file to gaia.cs.umass.edu?

**IP address: 192.168.1.89**

**TCP port number: 53665 is being used for the transfer of files (It is also using 53664 for FIN, ACK, and 53678 for SYN)**

Graphical user interface, text, application

Description automatically generated

Since this lab is about TCP rather than HTTP, let’s change Wireshark’s “listing of  
captured packets” window so that it shows information about the TCP segments  
containing the HTTP messages, rather than about the HTTP messages. To have  
Wireshark do this, select Analyze->Enabled Protocols. Then uncheck the HTTP box and  
select OK. You should now see a Wireshark window that looks like:  
Graphical user interface, application, table, Excel

Description automatically generated

This is what we’re looking for - a series of TCP segments sent between your computer  
and gaia.cs.umass.edu. We will use the packet trace that you have captured (and/or the  
packet trace tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-  
traces.zip; see earlier footnote) to study TCP behavior in the rest of this lab.

# **TCP Basics**

Answer the following questions for the TCP segments:

Note: Close the tab in which you've received the congratulation.

## What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as an SYN segment?

**The sequence number of the TCP SYN segment used to initiate the TCP connection between the client computer and gaia.cs.umass.edu is 0. The SYN flag is set to 1 and it indicates that this segment is a SYN segment.**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## What is the sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the Acknowledgement field in the SYNACK segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYNACK segment?

Graphical user interface, text, application

Description automatically generated

**The Sequence number of the SYNACK segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN is 0. The raw value is 2262807227.**

Graphical user interface, application

Description automatically generated

**The value of the Acknowledgement field in the SYNACK segment is 1. The raw value is 313610246.**

Graphical user interface, application

Description automatically generated

**The value of the Acknowledgement field in the SYNACK segment is determined by gaia.cs.umass.edu by adding 1 to the initial sequence number of the SYN segment that was received from the client computer. Also, the acknowledgment number of SYN, ACK is equal to the sequence number of the next ACK segment.**

**The SYN flag and Acknowledgement flag in the segment are set to 1 and they indicate that this segment is a SYNACK segment.**

Graphical user interface, application

Description automatically generated

## What is the sequence number of the TCP segment containing the HTTP POST command? Note that to find the POST command, you’ll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a “POST” within its DATA field.

**The sequence number of the TCP segment containing the HTTP POST  
the command is 1. The raw value is 523582368.**

Graphical user interface, application, table

Description automatically generated

## **What is the maximum segment size in the first ACK by the server?**

**MSS = 1460 (It is found in SYN or SYN, ACK segments, in other segments its indicated using the length field)**

A picture containing text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## What is the round-trip time(RRT) for the ACK received by the server?

**Round trip time by taking the time difference between the SYN and the SYN/ACK packets.**

**We can Calculate the RTT by subtracting the timestamp of the original data packet from the client and the timestamp of the ACK packet from the server.**

**RTT = 17:34:50.795298000 – 17:34:50.85761300 = 0.062315**

**Or**

**RTT = 0.062508000-0.000193000 = 0.062315**

**The packets are highlighted below:**

Graphical user interface, application, table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

## What is the calculated window size and checksum in the HTTP first packet?

**The window field controls the flow of data. It is a new window size that has been requested.**

**The checksum field is to ensure that the TCP segment reaches the destination without any errors. It is used for error detection.**

Graphical user interface, text, application

Description automatically generated

## Is the PUSH flag set? If yes? What is its significance?

**Yes, the PUSH flag is set. The PUSH flag instructs the receiving application to process the data as soon as possible rather than waiting for additional data to arrive, which can enhance the program's overall performance. Additionally, it can serve as a reminder for the sender to transmit data as soon as feasible.**Graphical user interface, text, application, email

Description automatically generated

## What are the font face and acknowledgment number of the HTTP packet which has the status of 200 OK?

**The font face is Arial, Helvetica, and Sans-serif. The acknowledgment number is 153041.**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

## How is the connection closed? (Hint: FIN flag) Attach the screenshot of the flags which are set at the end?

**The FIN flag is used to indicate that the sender has no more data to send.**

**There are three ways a TCP connection is closed:**

* **By sending a FIN packet to the server, the client takes the initiative to cut off the connection.**
* **By sending a FIN packet to the client, the server starts the connection from being closed.**
* **The connection is cut off by both the client and the server.**

**The ACK and FIN flags are set at the end.**

Graphical user interface, text, application, email

Description automatically generated

## Who has set the FIN bit? Is it server or client? Attach the screenshot.

**The client set the FIN flag first.**

**The FIN flag is set in two packets:**

1. **Frame 9: Source IP - 192.168.1.89 (Client), Destination IP - 128.119.245.12 (Server)**
2. **Frame 251: Source IP - 128.119.245.12 (Server), Destination IP - 192.168.1.89 (Client)**

Graphical user interface, text, application

Description automatically generated

## What is the acknowledgment number of the last TCP ACK?

**The acknowledgment number of the last TCP ACK is 779.**

Graphical user interface, text, application

Description automatically generated