

Department of Electrical and Computer Systems Engineering
Monash University

Information and Networks, ECE3141

Lab 4: Probing the Network

Authors: Dr. Mike Biggar

Dr. Gayathri Kongara

(This update: 7 April 2022. Some material from an earlier
version by Ahmet Şekercioğlu & David McKechnie)

1. Introduction

In this experiment, you will become familiar with the *Wireshark Network Protocol Analyser*¹, and the *ping* and *tracert* utilities as you investigate network operation.

By unpacking and interpreting the actual data packets exchanged on the network, Wireshark helps us to see what happens when computers communicate and exchange information. Experience gained with Wireshark in this laboratory will help you complete the further investigations of the TCP/IP protocol stack in subsequent laboratories and understand the operation of Internet protocols.

We will also introduce the utilities *ping* and *tracert* which send test packets over a network to give us connectivity, delay and route information².

In this laboratory class, you will carry out the following as you investigate network operation:

- Become familiar with the use of Wireshark. We will also use *ping* and *tracert* for some of the exercises in this lab, but it is important to get some experience with Wireshark as well; we will use it more extensively in the next lab and it is important to understand what packet capture software such as this can do
- See how we can interpret the protocols running between machines on a network by unpacking the packets exchanged between them
- Understand the different addressing mechanisms in place to identify devices on a network.
- Understand why different Internet applications run different protocols at the same level of the protocol stack.
- See how *ping* and *tracert* use the ICMP protocol to determine important network connectivity information.

Important note on the installation and use of Wireshark.

¹ Wireshark was previously known as *Ethereal*. You have already seen in lectures some examples of Wireshark and Ethereal packet captures.

² *Ping* and *tracert* (or *tracert*) are separate from Wireshark. They are present in some form in any computer, and do not need to be installed.

If you're going to complete this lab using the laboratory computers:

Wireshark is already installed on the computers in our classroom. You will not have to install anything, but if you're going to complete all the tasks in this lab exercise using lab computers during the allocated 2-hour lab session, you will need to be very well prepared. It would be a good idea to try the lab computers some time in advance of the lab session when the room is empty, to familiarise yourself with Wireshark and make sure you don't lose time during the lab session trying to understand how it works. Installing it on your own computer might be easier and will allow you to complete some of the exercises before the lab class.

If you're going to complete this lab using your own computer:

If you are installing Wireshark on your computer, it is very important that you ensure you can capture packet data using Wireshark BEFORE you start the lab session. You don't need to do the analysis beforehand, but you need to be sure that you can do the capture. Wireshark is free to download and install on your computer and it poses no risk (eSolutions allows it on Monash computers). If, however, you are uncomfortable about installing unfamiliar software on your computer, we suggest you use the laboratory computers.

If you're going to complete this lab online:

We are aware that there are sensitivities about downloading and running this software for our offshore students, particularly those in China, because it analyses local network traffic and reports IP and mac addresses. For these students, we recommend you perform this lab (and the next) by using the video recording provided in Moodle; it is a screen recording showing the running of Wireshark and the data it collects, which you can use to complete the lab without having to install anything and without having to share any of your network addressing information.

The rest of this lab exercise is written as if you are installing and running Wireshark on your own computer. If you're going to be doing this lab using the computer lab PCs, then you can obviously jump over the installation description. If you're going to be using the provided screen recording instead of installing Wireshark yourself, you should read through the descriptions about Wireshark (so you know what it does), but you will rely on the recording for your data.

2. Pre-lab

Wireshark Installation: If you need/want to install Wireshark on your computer, do so well before the laboratory class. Visit www.wireshark.org to download and install the appropriate version of Wireshark for your computer (Windows, MacOS and Linux are all supported). You should be able to do all we need by doing a standard install, accepting all the default configurations. Do a test (capture instructions are provided at the start of Section 7 below) to make sure you can capture packet data when you go to a website. If you aren't able to capture anything, make sure the correct interface (e.g. Wi-Fi) is selected (go to "Capture" at the top of the window, then select "Options" and look at the "Input" tab).

You must also read through this entire laboratory description and complete the pre-lab online quiz. Failure to do so will not only mean loss of marks for the quiz, but you will also struggle to understand the principles under investigation and will almost certainly fail to complete the laboratory on time.

2.1 Pre-reading & lecture material

You should review the lecture material on ICMP, ping and tracer, and read the introductory material below for this laboratory (Sections 3 & 4) concerning Wireshark, ping and tracer, before proceeding with this Pre-lab exercise. To avoid having to read it all again during the lab, you might like to highlight a few things that you know you'll need to find again, such as how to search through the captured packet data.

Note that some topics relevant to this lab are also covered in the lecture material this week (week 6). Make sure you go through the learning modules on IP and Control Protocols in preparation for this lab.

2.2 Browser cache

Do some reading about how a browser cache works, and why it is useful. In particular, you should make sure you understand under what circumstances web data might be retrieved from the cache rather than the web server, and what impact this might have on the data exchanged between your browser and the web server.

3. Wireshark

A protocol analyser is a tool for capturing, displaying, and analysing the packets exchanged in a network. By observing protocols in action and seeing the sequence of message exchanges between two protocol entities, getting into the details of the packet contents can lead to a deeper understanding of the networking world.

The basic tool for observing the messages exchanged between executing protocol entities is called a “packet sniffer”, which captures messages being sent/received from/by your computer. It will also typically store and/or display the contents of the various protocol fields in these captured messages. (That is, it will partly interpret the protocols used inside the packets, to make it easier to see what is going on.) A packet sniffer by itself is passive: it observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer takes a copy of packets that are sent/received from/by the applications and protocols executing on your computer.

Figure 1 shows the structure of a packet sniffer. At the right of the figure are the protocols (in this case, Internet protocols) and applications (such as a web browser or email client) that normally run on your computer. The packet sniffer, shown on the left, is an addition to the usual software in your computer, and consists of two parts. The packet capture library receives a copy of every link-layer frame that is sent from or received by your computer. Because of the layered protocol structure discussed in lectures, messages exchanged by higher layer protocols such as HTTP, TCP, UDP, DNS, or IP (IPv4 or IPv6)³ are all eventually encapsulated in link layer frames that are transmitted over physical media such as an Ethernet cable, Wi-Fi radio connection or 4G/5G cellular network. In Figure 1, the assumed physical medium is an Ethernet cable, and so all upper layer protocols are eventually encapsulated within an Ethernet frame. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

³ The University network (and probably your home gateway) now supports both IPv4 (32-bit long IP addresses) and IPv6 (128-bit long IP addresses). When we refer to an “IP datagram”, without giving a version number, this implies that the datagram could either be in IPv4 or IPv6 format.

In fact, the packet sniffer captures all frames seen on the Ethernet interface, and that includes frames not intended for your PC. These will be frames that your Network Interface Card receives and checks to see if they are intended for you (i.e. it will check the MAC address⁴), but when the addresses don't match it will reject them and so not pass them up to the next (Network) layer. Nevertheless, Wireshark will capture such packets.

The second component of a packet sniffer is the packet analyser, which displays the contents of all fields within a protocol message. To do this, the packet analyser must 'understand' the structure of all messages exchanged by the different protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol in Figure 1. The packet analyser understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. It understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string GET, POST or HEAD. However, as we note later, encrypted exchanges using the secure HTTPS protocol cannot be understood above TCP.

You will understand in more detail what all these protocols do as the lectures progress. For now, the important thing to understand is that we have this layered structure of protocols contained within other protocols (each performing its required function), and the packet analyser helps to unpack these to tell us what is going on.

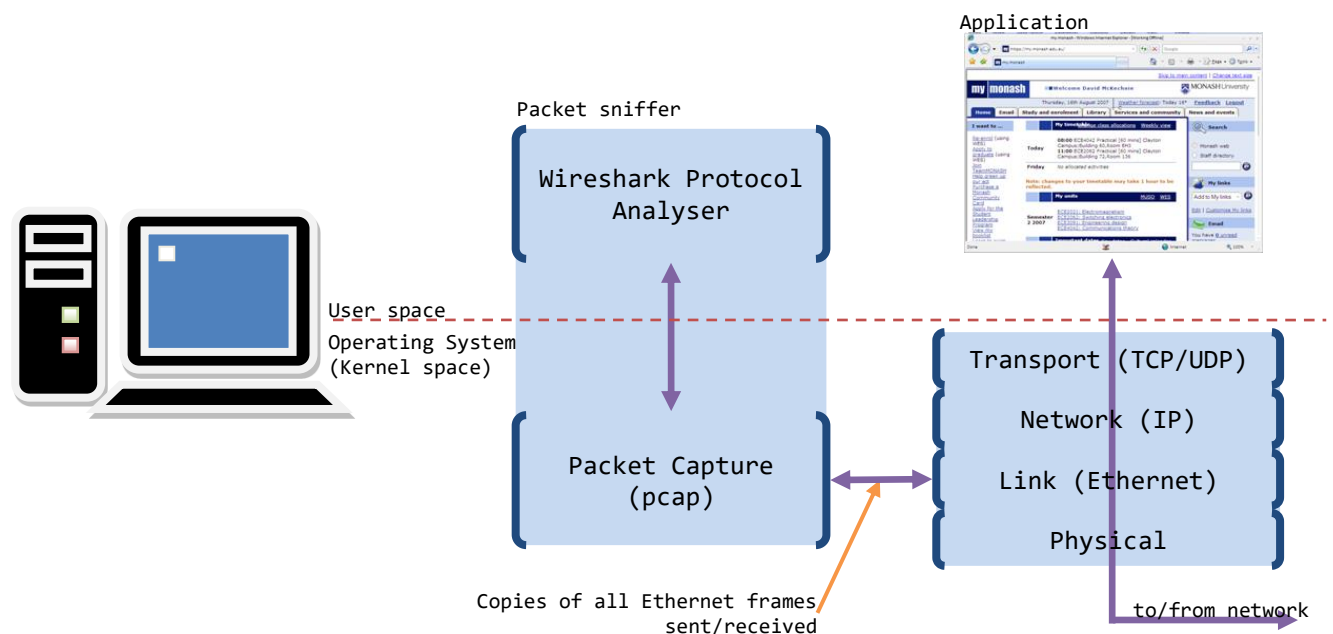


Figure 1. Packet Sniffer in Operation

We will be using the Wireshark packet sniffer [1] for this laboratory, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the

⁴ We will meet the MAC address when we discuss the Data Link layer. A summary for now is that it is a unique physical address associated with your network interface card (NIC), and it is used to identify your machine on its local network. The IP address may change, but the MAC address will not. Therefore, it can be used in Wireshark to identify and filter out only packets intended for or coming from your NIC

protocol stack. (Technically speaking, Wireshark is a packet analyser that uses a packet capture library in your computer). Wireshark is a free network protocol analyser that runs on Windows, Unix based systems, and Mac OSX. It is an ideal packet analyser for experimenting with network protocols – it is stable, has a large user base and well-documented support that includes a user-guide [2], detailed wiki [3] and FAQ [4], rich functionality that includes very many display filters for analysing different protocols, and a well-designed interface. It can also operate on several different computer interfaces.

Wireshark can be run in two modes, depending on whether you want to look at data already captured, or to capture fresh data. In the first mode, the application opens existing capture files for analysis; only the upper block of the packet sniffer in Figure 1 is run. In the second mode, the Wireshark application activates the *pcap* block to capture a frame sequence on the selected network interface in real time and can then store the result in a packet capture file.

3.1 Wireshark's Graphical User Interface (GUI)

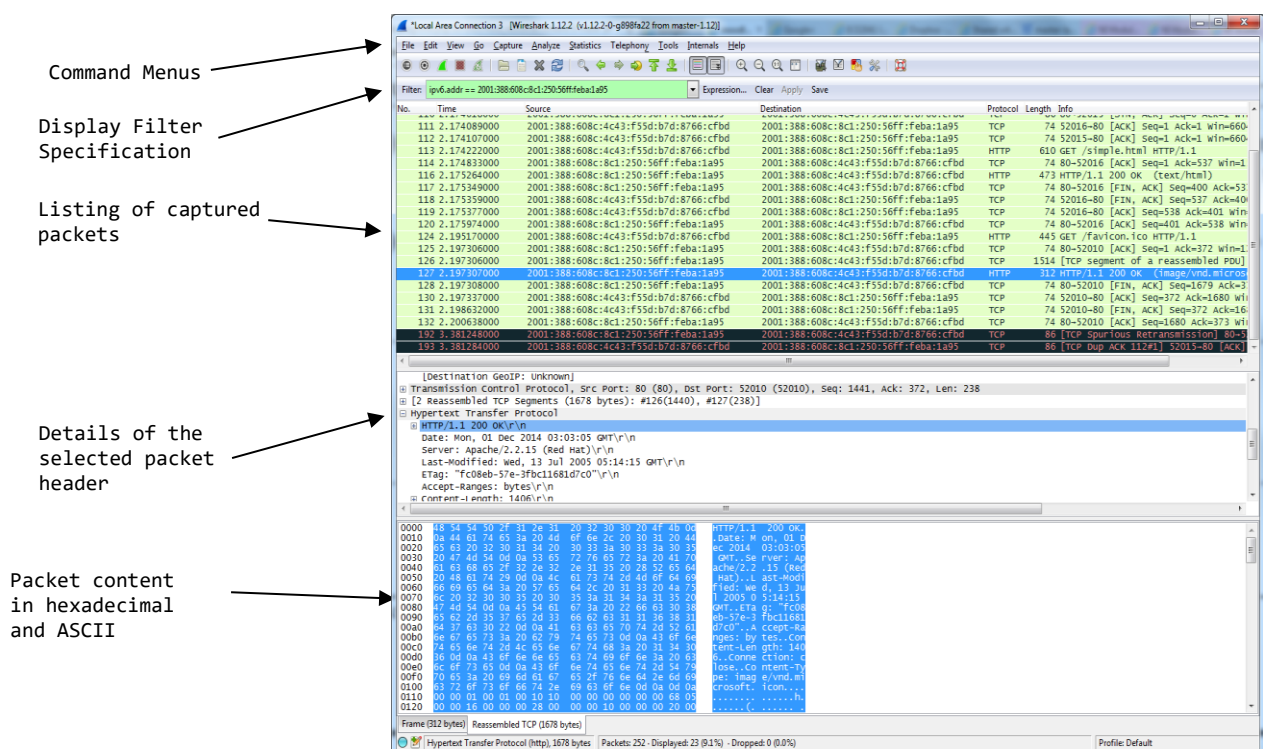


Figure 2. Wireshark Graphical User Interface

When you run Wireshark and open a data file (that is, open a previous capture that has been stored in a file), the GUI shown in Figure 2 is displayed.⁵ The interface has five major components:

The command menus are standard pull-down menus located at the top of the window. Of particular interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, or exit the application. The Capture menu allows you to begin packet capture.

The packet-listing window displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol header), the time at which the packet was captured, the packet's source and destination IP (network layer) addresses, the protocol type, and protocol specific

⁵ When you first start Wireshark, you will not see any data until you start a capture, or open a file containing a pre-captured packet trace.

information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet: i.e., the protocol that is the ultimate source or sink for this packet.

The packet header details window provides details about the packet selected (highlighted) in the packet listing window. Details about a selected packet include information about the Ethernet frame and IP datagram. The amount of Ethernet and IP layer detail displayed can be expanded or minimised by clicking on the + or – button to the left in the packet details window. If the packet has been carried over TCP or UDP, these details will also be displayed, and can similarly be expanded or minimised. Finally, details about the highest-level protocol that sent or received this packet are also provided.

The packet contents window displays the entire contents of the captured frame, in both ASCII and hexadecimal format.

The packet display filter is towards the top of the Wireshark GUI. A protocol name or other information can be entered into this field (as you would a search field) to filter the information displayed in the packet listing window (and hence the packet header and packet contents windows). This is an important tool, because it helps find the packets of interest among what could be a very large set of captured data.

3.2 How to filter captured packet data

To analyse just the packets we're interested in, it's important to be able to filter the collected data properly. We can apply a filter so that only the http packets are shown, or only those with a particular destination address, or only those with a particular source address, etc.

For instance, you could identify the Ethernet MAC address⁶ or IP address of the computer, and then filter using:

```
eth.dst == your MAC address
```

or

```
ip.dst==X.X.X.X
```

You can filter for many different things, and you can also combine filter criteria. For instance,

```
"http and ip.addr==104.116.211.48"
```

will only show HTTP protocol exchanges that also contain the indicated IPv4 address. (You can use "&&" instead of "and" to combine filter conditions.) Note that *ip.dst* means "the destination IP address", while *ip.addr* means "the appearance of this IP address (i.e. it could be source or destination)". Guess what *ip.src* means? With a little practice, you will get used to the way it works. The colour of the filter specification window will change to tell you when your filter expression is valid and, as an alternative to typing them in, you can use the "Expression" builder (button at top-right in Wireshark "Expression...") to create filters.

Of course, you're going to need to know the IP address(es) and MAC ("Physical") address of the interface card to help in your filtering of Wireshark captures. If you're using a Windows PC, type *ipconfig/all* in a Command window. On MacOS, go to "About this Mac" from the Apple icon in the top-left corner, then "System Report" on the Overview tab, then "Network" and "Locations" under Network). The "Capture Interfaces" window in Wireshark will also give you these addresses.

⁶ See the earlier footnote introducing the MAC address.

4. Ping and Tracert

As you will understand from lectures, data packets we send over the Internet are commonly sent through many routers in multiple ISPs and, depending on network conditions at the time, may even be sent via countries that we don't necessarily expect. The two utility programmes *tracert*⁷, and *ping* are tools we can use to explore the structure and performance of the Internet at the router level⁸. If you have some way of correlating IP addresses with physical locations, you can even draw a 'map' of the Internet and visualize the relationships of the backbones as shown in Figure 3. These two utility tools utilise the Internet Control Message Protocol (ICMP). The *ping* tool can tell us whether a host connected to the Internet is reachable and also an estimate of the RTT (round trip time) from us to the remote host and back again. The *tracert* tool allows us to observe the route (in terms of router hops) that an IP packet takes from the local host to the remote host⁹.

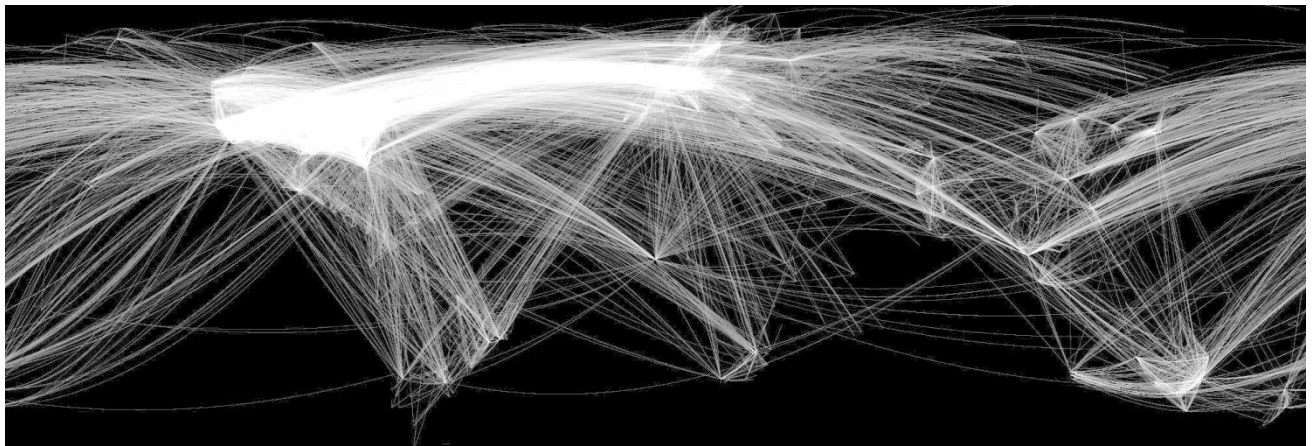


Figure 3 - Map of the internet compiled from data gathered using ping and traceroute at the Dimes Project (no longer active). Image copyright Chris Harrison 2007.

5. Running “Ping”

Choose a local website (in or near your home city) and one overseas. Pick URLs for which you are reasonably confident of their geographic location¹⁰. Note that this is not at all trivial. Many organisations host their websites on “Content Delivery Networks” (CDNs) which have networked servers all over the world. So, while you might think you're connecting to a server in the US, for instance, you're actually talking to one close by in your home country somewhere. The best choices are those mid-size organisations who are most likely to host their own websites on their own premises. Universities used to be good choices, but increasingly they tend to use external hosting services now. You should use several pieces of information to conclude the geographic location: geolocation sites may be useful (though they are not 100% reliable), ping times should be consistent, and the router names mentioned in a *tracert* response (see Section 6 below) should be consistent with a route to the target city.

⁷ The Unix/Linux/OSX equivalent of *tracert* is the *traceroute* command.

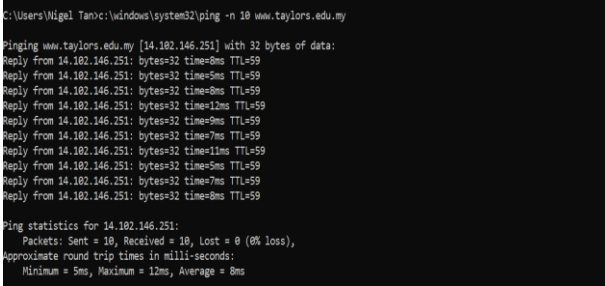
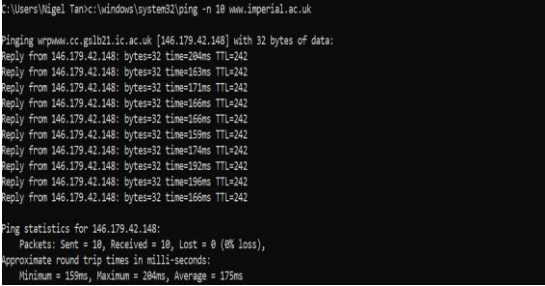
⁸ We are not talking about Wireshark here; *ping* and *tracert* are included as utilities with the computer operating system. They are separate from Wireshark.

⁹ We need to be careful here. As you will discover in lectures, most commonly the route for each IP packet is determined on a hop-by-hop and packet-by-packet basis. It could be different for different packets, and there is not one single route that is taken. However, by sending a sequence of ICMP packets, *tracert* obtains a “snapshot” of the route that packets are commonly taking at the time to get from your machine to the indicated destination.

¹⁰ How can you know that the server is in a particular geographic location? Even though the organisation may be headquartered in a particular city, will their server(s) always be there?

(Don't waste too much time trying to find suitable websites. It's not the main point of the exercise and there are more important things to look at. It's actually very hard to find machines which are known to be in a particular city. From Melbourne, you could use the Monash machine aussie.eng.monash.edu.au (it's not a web server, so your browser can't access it, but you can ping it). You could also use <http://ptsv2.com/> which is a test site in multiple locations, one of which seems to be in Melbourne. A suitable local URL for Sunway students would be www.taylors.edu.my. Possible overseas websites could be UK universities like www.imperial.ac.uk or www.bristol.ac.uk, though after you get to the "janet" academic network you will just see "*" responses which indicate a timeout waiting for a response – probably due to machines programmed not to respond to ICMP messages.)

Run ping on the URLs of both your chosen machines/websites¹¹ and note the statistics of the RTT (Round Trip Time) obtained for each¹².

URL1: www.taylors.edu.my	URL2: www.imperial.ac.uk
 <p>RTT =</p> <p>Minimum = 5 ms</p> <p>Maximum = 12 ms</p> <p>Average = 8 ms</p>	 <p>RTT</p> <p>Minimum = 159 ms</p> <p>Maximum = 204 ms</p> <p>Average = 175 ms</p>
<p>Comment on any differences in ping time to these web servers obtained by you and your partner¹³. Can you suggest reasons for the differences?</p> <p>The difference could exist due to the difference in access time. This could be dependent on the network traffic at that specific point of time. If there is a larger network traffic at a specific time in day, then there could be a higher RTT delay. There could also be this difference in time due as to the geographical location of the host server. The host server of the first website could be nearer to my geographical location than the second website host's server. This could explain the time difference between the RTT delay</p>	

¹¹ If you were ping the Monash machine, you would type "ping aussie.eng.monash.edu.au" in a command window on a Windows machine, or in a Terminal window on OSX. (With the Mac, you might want to include a "count" parameter "ping -c 5 aussie.eng.monash.edu.au" because the default count is 64 for MacOS, and that requires a bit of patience. The default is 4 for Windows.)

¹² Note that not all servers will respond to ping; their administrators could have set them to ignore such requests. If your "ping" request gets no response (i.e. repeatedly times out), then try another website.

¹³ If you and your partner are doing this exercise in the computer lab at Monash, then we wouldn't expect much difference! In this case, you might like to do this test from home before you come to the lab.

6. Running “tracert”

Now run tracert, targeting the same two URLs that you pinged. As with ping, tracert is run in a Command window on the Windows operating system¹⁴, and in a Terminal window on OSX¹⁵, but the command for the Mac is “traceroute”.

Note that, as originally designed, tracert/traceroute operated as we have described in lectures, using ICMP control messages, just the same as ping. Since then, there has been more flexibility built in and in fact, by default on Unix-based operating systems (MacOS, Linux), it will use UDP segments on IP instead of ICMP. This means that traceroute and ping can get different responses. In particular, you might be able to ping a machine but find that traceroute gets blocked by some machine along the path and all other attempts to penetrate further into the network lead to a timeout. To reduce this problem, we can force the use of ICMP with a “-I” option. This is not necessary on Windows PCs. So, for all use in this laboratory you should use the commands

tracert (machine name or URL) on Windows

traceroute -I (machine name or URL) on MacOS

What observations do you make about the route taken by packets to get to your destination website? Where are the greatest sources of delay?

```
C:\Users\Nigel Tan>tracert www.taylors.edu.my

Tracing route to www.taylors.edu.my [14.102.146.251]
over a maximum of 30 hops:

  1  5 ms    1 ms    <1 ms   192.168.0.1
  2  3 ms    8 ms    8 ms    210.186.211.254
  3  4 ms    5 ms    6 ms    10.55.49.47
  4  8 ms    5 ms    5 ms    10.55.41.212
  5  6 ms    19 ms   8 ms    ipserverone-2.myix.my [218.100.44.111]
  6  9 ms    15 ms   14 ms   103.21.181.163
  7  9 ms    8 ms    17 ms   103.197.56.131
  8  29 ms   16 ms   8 ms    14.102.146.251

Trace complete.
```

```
C:\Users\Nigel Tan>tracert www.imperial.ac.uk

Tracing route to wrpwww.cc.gslb21.ic.ac.uk [146.179.42.148]
over a maximum of 30 hops:

  1  4 ms    1 ms    <1 ms   192.168.0.1
  2  9 ms    3 ms    7 ms    210.186.211.254
  3  159 ms  167 ms  163 ms  10.55.49.45
  4  25 ms   11 ms   11 ms   10.55.100.228
  5  219 ms  168 ms  208 ms  10.55.200.123
  6  157 ms  160 ms  158 ms  ge-11-2-0.londhx-ban2.ja.net [193.62.157.37]
  7  159 ms  182 ms  174 ms  ae24.londhx-sbr1.ja.net [146.97.35.197]
  8  160 ms  158 ms  159 ms  ae29.londpg-sbr2.ja.net [146.97.33.2]
  9  165 ms  173 ms  168 ms  ae30.londtw-sbr2.ja.net [146.97.33.6]
 10  182 ms  187 ms  162 ms  ae26.londtw-ban1.ja.net [146.97.35.218]
 11  159 ms  171 ms  163 ms  imperial.ja.net [146.97.140.22]
 12  *      *      *      Request timed out.
 13  162 ms  158 ms  165 ms  bdr-rt1-bdr-fw1-prod.net.ic.ac.uk [192.195.105.2]
 14  160 ms  167 ms  161 ms  core-rt1-bdr-rt1.net.ic.ac.uk [194.82.153.196]
 15  160 ms  162 ms  274 ms  vl-3285-dcs1-rt3.net.ic.ac.uk [146.179.110.14]
 16  163 ms  161 ms  161 ms  wrpwww.cc.gslb21.ic.ac.uk [146.179.42.148]

Trace complete.
```

¹⁴ Across different versions of Windows, it is not always obvious how to run a Command window, and there are also many different ways to run it. You could just type “cmd” in the search box in the lower-right corner of the desktop, or type “Win+R” to bring up a “Run” window, then type “cmd”.

¹⁵ Terminal would normally be in the Utilities folder inside “Applications”.

Observations:

As we can see in the above figure 1, we can see that there are a total of 8 hops with the greatest source of delay at the 8th Hop. We can also see that in figure 2 that there are a total of 16 hops with the greatest source of delay at the 5th hop.

Explanations:

The `tracert` command is used to display the time it takes for a packet of information to travel between a local computer and a destination IP address or domain. In short, it basically shows the route that a packet will take.

Comment on any differences in the `tracert`/`tracert` output between you and your lab partner¹⁶.

Based on the above two figures, we can note that the number of hops for the local website will be lesser than the number of hops for the international website. We can also notice that there are a few “Request Timed Out” in some of the line when the `tracert` command is used on the international website. The time taken for the hops of the international website is also much longer than the local website.

There could be many reasons why this observation exists. The first reason would be that there is a difference in the geographical location. My lab partner would have accessed the website using his house’s router while I would have accessed the website using Monash University Malaysia’s router. The second reason could be due to the fact that the router used was busy with other request, which could result in a longer delay.

For the long distance URL you have chosen, make an estimate of the propagation time¹⁷ (that is, time that the signal spent actually in transit along telecommunication links). Assume that all the links will be on fibre and that the propagation speed is $2c/3$. How does your estimate compare with the RTT (or components of it as presented by `tracert`)? What do you think is the cause of the difference? What percentage of the total RTT do you estimate to be due to propagation? Please note the percentage and provide it when you go to “Feedback” at the end of this lab.

The URL used is www.imperial.ac.uk.

Firstly, we would need to obtain the approximate RTT and the distance between the supposed geographical location of the host server. The RTT obtained can be seen above which is 175 ms and the distance between Malaysia and the United Kingdom (UK) is 10576 km (Based on Google).

¹⁶ Again, you’re not going to see a difference between two computers in the same lab at Monash.

¹⁷ This can only be an estimate, because we do not know (even between successive routers as listed in the `tracert` report) the actual fibre path taken. But for the purpose of this exercise, a rough approximation, making reasonable assumptions, is good enough.

The estimated propagation time can then be calculated using the formula which is $[(10576 * 1000) / (2 * c/3)] * 2$. We will get $52.88 \text{ ms} * 2 = 105.76 \text{ ms}$ which approximates to about 105 ms.

This propagation time is much lesser compared with the RTT found by using the tracert function.

We can get the percentage of the total RTT due to propagation by taking the propagation time divided by the RTT

We would get $\frac{105.76}{175} * 100 = 60.43 \%$.

This means that 60.43 % of the time, the signal will be transmitted along the telecommunication links.

Theoretically, the estimated propagation time should be equal to the RTT that was obtained using the tracert function. There could be a number of reasons why there is such a stark difference. The first reason could be that there is network traffic delay which is dependent on your geographical location and the time period in which the website was accessed at. The second reason could be that RTT includes that bit rate of the router and the propagation delay in which the propagation does not. Hence, this is why the propagation time will be smaller in magnitude compared with the RTT. There could also be processing delay and launching delay involved.

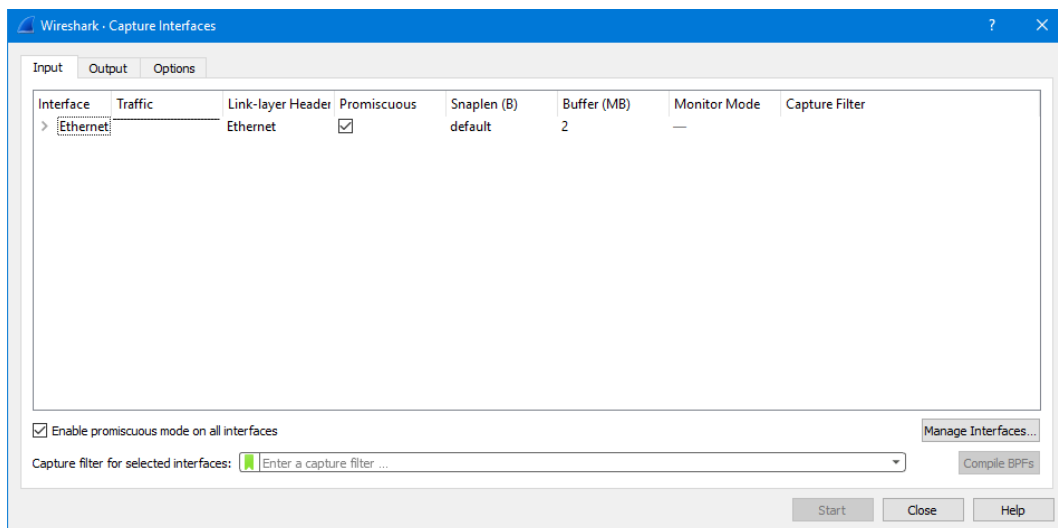
7. Packet capture

We will now use Wireshark to capture and interpret the data packets seen on your computer's network interface.

- a) Run the **Wireshark** application. (It won't start capturing data yet.)
- b) Open a web browser, too, because we're about to see what happens when the browser requests data from your chosen websites.
- c) Use the following commands to clear the browser's cache:
 - i) Internet Explorer users: Tools > Delete Browsing History
 - ii) Firefox users: Tools > Clear Private Data

- iii) Chrome: Click on the “three dots” icon in the top-right, select “More tools” and then “Clear browsing data” (or just type Ctrl+Shift+Del).
- iv) Safari: “Clear History” under the History pull-down menu at top.
- d) To begin packet capture, select the Capture pull down menu in **Wireshark** and select **Options**. You will see the window called **Capture Interfaces** similar to that shown below. Like the image below, the “Start” button will appear greyed out at first. You should select the interface you are using (e.g. wired Ethernet or perhaps WiFi), then the “Start” button will become active and, if you click on it, packet capture will begin. From that point on, you can toggle capture on/off by typing Ctrl-E or you can click on the Start or Stop entries under the Capture menu.

Note that we will be running multiple different versions of Wireshark on multiple different types of computer operating systems and with computers connecting over different networks. As a result, Wireshark’s behaviour and the screens you encounter may look a little different to what is shown here. For example, the “Capture Interfaces” window shown below is from one of the lab PCs which has only Ethernet available – so there is only one option to tick. If your computer has WiFi, there will be more options. It is REALLY IMPORTANT that you try capturing some packet data BEFORE the lab session to ensure that you are able to do so during the session. If you have problems, see if you can solve them with your regular lab partner or raise them on the Discussion Forum.



- e) All packets **seen** by this network interface are now being captured by Wireshark. Note the emphasis on ‘seen’; as discussed earlier, all packet activity on the line connected to the network interface will be recorded, including multicast and broadcast packets which are not directly addressed to your computer¹⁸.
- f) If necessary, CTRL+E lets you terminate packet capture operation, but don’t stop it yet.
- g) While Wireshark is running, put the URL of your chosen local website (i.e. from Section 5) into the browser and hit “Enter”.
- h) After the web page comes up, go back to Wireshark and stop the capture (Ctrl+E, or select “Stop” under Capture, or click the square stop icon – whichever is relevant). The main Wireshark window now will display all packets captured since you began packet capture; it should be similar to Figure 2. You now have live packet data that

¹⁸ Note that you may also see both IPv4 and IPv6 packets in the same capture; these can coexist on the same network.

contains all Data Link Layer frames seen by your selected network interface. The message exchanges with the web server for your URL should appear somewhere in the listing of packets captured¹⁹. But there could be many other types of packets displayed as well! Even though the only action you took was to download a web page, there could be other processes running on your computer that are unseen by the user (e.g. synchronising with Dropbox if you use it), as well as possibly other activity on the network²⁰.

- i) Type `http` (make sure it is in lower case) into the display filter specification window at the top of the main Wireshark window. Then select Apply (to the right of where you entered the filter) or hit return. This will cause only HTTP messages to be displayed below.
- j) Can you find the HTTP GET message that was sent from your computer to the HTTP server? When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment and HTTP message header information will all be displayed in the packet-header window. Click on the + to maximise the amount of information displayed about the HTTP protocol
- k) As we have discussed, Wireshark captures EVERY frame seen on the selected interface, including some that your computer will ignore because they are not intended for it.

How could you identify and filter the captured data so that you only see the data frames specifically addressed to your computer?

First, we have to note down the source address and use the `http` filter to get the GET message. Then, we can identify and filter the captured data by using the syntax of `ip.addr == 192.168.0.198`. When we apply this filter, we can then see the data frames that is specifically addressed to my computer.

How can you identify the first exchange of IP packets between your PC and the remote website²¹? (Note that, before sending anything to the remote website, there is, of course, a DNS query to find its IP address. That's not what we are interested in here.)

Firstly, we would need to get the `ipv6` address by using the DNS query. This can be done by using the filter option in wireshark and typing in the command

```
Ipv6.addr == XXX address (Ipv6 address)
```

The first packet that can be seen in the list would be the first exchange of the IP packets. The remote website will be the packet intended for the beginning of a three-way handshake (SYN)

¹⁹ Note that modern web browsers will commonly set up multiple parallel connections with a web server. You may need to be aware of this to make sense of the packets you capture and inspect.

²⁰ Remember, of course, that if you are doing this during an online lab session and using the same computer, then there is a lot of activity on your network interface supporting that!

²¹ Would that exchange involve HTTP? If not, you better remove your `http` filter, or it won't be visible to you.

What is the function carried out by that exchange?

The function carried out by that exchange would be to establish end to end communication between the web server and the PC. The Transmission Control Protocol (TCP) is intended for end-to-end communication.

By looking at the timestamp that Wireshark records for each packet, we can compare the time at which the first IP packet left your PC on its way to the remote website, and the time a response was received. Record the times and the differences. How do they compare with your ping RTT? Should they be the same?

The website used: www.taylors.edu.my

Since $t_1 = 5.673462$ s and $t_2 = 5.681147$ s
The time difference is 7.685 ms

The ping time difference obtained is 7.681 ms

The two values are quite similar to each other. In theory, both of the value should have the same value as the route is relatively short for the local website. RTT can also be said to be defined as the time taken for the signal to be sent until an acknowledgement is received by the sender.

- l) Repeat the Wireshark packet capture for the other website (the international one).

Is the comparison between Wireshark recorded times and ping times any different in this case? How and why did they differ?

The website used: www.imperial.ac.uk

Since $t_1 = 8.199896$ s and $t_2 = 8.398626$ s
The time difference obtained would be 198.73 ms.

The ping time difference obtained would be 194.25 ms.

The two values obtained will be different from each other. This is due to the longer route that has to be taken between the international web server and the PC. Since it is an international web server, it could take a longer time due to the huge traffic. In Wireshark, the entire http request will be sent to the web server while in ping, only a single packet is sent so the effect felt by the traffic network will be mitigated.

- m) Suppose you wanted to repeat the above exercise (that is, going to the same URL while performing a Wireshark capture). What difference would it make whether you cleared the browser history before doing so? Feel free to try it and see.

Explain the answer to the above question with reference to the browser cache and how it operates.

The browser cache can be defined as a temporary storage area in a disk or memory for files downloaded by browser or holds the most recently downloaded web pages. The browser cache will determine whether if the browser has a local copy or not which will enable the page to load faster. If the browser history is cleared, then Wireshark would need to prompt for the http request. If the browser history is not cleared, then Wireshark would not need to prompt for another http request. So, we can deduce that caching will make the page to load faster.

8. Captured packet statistics

Now that we have the ability to capture packets, we can do some analysis of the types and amounts of traffic on the network. For instance, we might be interested to know what proportion of IP packets were carrying HTTP protocol messages. You captured data while your browser went to a web page, but some of the packets were carrying out necessary tasks before you could actually use HTTP (e.g. DNS and the TCP set-up), and there could have been other processes happening in your PC and elsewhere on the network. So, what proportion of packets were actually http packets?

A challenge we face in trying to look deeply into packet contents in these days is that much of the world wide web is now “secured”, in that (as you probably noticed in your regular usage) most websites use https (the secure, encrypted version of http)²². Most browsers even give you a warning these days if you go to an unsecured site. We will learn a little about https at the end of the semester, but the important thing to note is that we cannot look inside the packet contents if encryption is used. Instead of seeing “http” contents in TCP segments, we will just see “TLSv1.3” (or something similar); TLS stands for Transport Layer Security and it provides the encrypted exchanges for https.

You can still investigate and analyse some aspects of the network messaging and protocols when https is used (for instance, the DNS is the same, the three way handshake with TCP is the same), but if we want to look at http exchanges, we will need to find an unsecured web site. These are relatively rare! If your chosen web sites from earlier on were not secured (i.e. did not use https, and did not show a little padlock at the top of your browser), then you can continue using the packets you’ve already captured. Otherwise, you might like to find an unsecured site and do a fresh capture when you point your browser to that site. One possibility is the Australian Bureau of Meteorology (www.bom.gov.au) which (as of 26 March 2022, at least!) still uses unsecured http.

- a) Look under the “Statistics” menu, to see what tools are available to analyse the captured packets. What proportion of the total packets you captured were http packets?

²² Even when you enter URLs beginning http://..., most often you are redirected to one that starts https://....

What answer do you get? Is it what you expect? Did your partner get very different answers? Why?

The total number of packets captured is 798 packets. The number of HTTP packets captured is 5 packets. The proportion of the HTTP packets received to the total number of packets received is 0.62 %. No, my lab partner did not get very different answer. This is because we only requesting for one URL and we only sent the request once (http request and http response).

What other interesting things can you find using the statistics tools?

We could see that there are the “Endpoints” section where there is a list of communication nodes such as MAC and Ipv4 addresses. It will also show the packet lengths and other information such as the number of other types of packets and about the captured file property.

- b) Now use the statistics tools in Wireshark to obtain the percentage of frames using each different type of Transport Layer protocol (at least TCP and UDP, but you might find others, such as the new “QUIC” protocol). Which is the dominant Transport Layer protocol used by each of the following?
- i) Downloading a pdf (or other) document from a website (again, because of the problem of analysing any secured sites, you might need a suggestion of an unsecured source; the bom again provides one:
<http://www.bom.gov.au/climate/current/statements/scs73.pdf>
 - ii) Playing a video clip. Most video streaming sites (including Youtube) are also secured, but here is one video link that is not (i.e. it uses http, not https):
http://clips.vorwaerts-gmbh.de/big_buck_bunny.mp4

We discussed in lectures why DNS uses UDP transport, but can you find any other application or data source that uses UDP? (Hint²³: Try setting up a zoom connection.)

Record the results you obtain in each case. Are they what you would expect? Make sure you discuss the results with a demonstrator, but you can proceed with the laboratory while you are waiting, if necessary.

Downloading PDF

TCP: 90.3%

UDP: 9.7 %

Playing a video clip from Youtube

TCP: 95.7 %

UDP: 4.3 %

Zoom Call

TCP: 7.9 %

UDP: 92.1 %

The results are expected.

²³ Actually, quite a big hint!

TCP is dominant in downloading PDF and playing a video clip from Youtube as it is reliable and it is able to ensure that all the packets are sent in order. The transmission time is not an important criterion when a PDF is being download or a Youtube video being played (not a real time streaming connection) although it does have some weight to it. Thus, UDP will not be dominant when downloading PDF.

UDP is dominant when we are having a zoom call. The transmission time is more important compared with reliability. UDP will not provide any guarantee for delivery but it will reduce the delay or overhead from the protocol. Thus, TCP will not be dominant when a zoom call.

9. Dissecting how Tracert works

In Section 6 above, you ran *tracert* (or *tracert* on a Mac) to discover the routers traversed by packets to get to a chosen URL.

- a) Run *tracert* again (don't forget the "-I" flag if you are using MacOS or Linux), but this time do so while capturing the packet data using Wireshark.
- b) Use the filter capability in Wireshark to find just those ICMP packets being sent from your computer towards the target URL. (If you filter only on "icmp", you will get the responses as well as the queries. Here we are interested in the icmp messages going FROM your computer.)

What do you observe about the "TTL" field (in the IP header) of the ICMP query packets? Can you explain it?

URL Used: www.taylors.edu.my

The Time-To-Live (TTL) field can be said to be a number or value in the Internet Protocol (IP) that limits the number of hops of an ICMP packet. It can be also be said that TTL tells the network router if a packet should be discarded or not based on the time it has remained in the network. The ICMP will first set the TTL to 1 and a response stating that TTL has expired will be received. The TTL will be incremented by the ICMP by 1 until a ping reply is obtained. This will indicate that the ICMP packet has arrived at the destination.

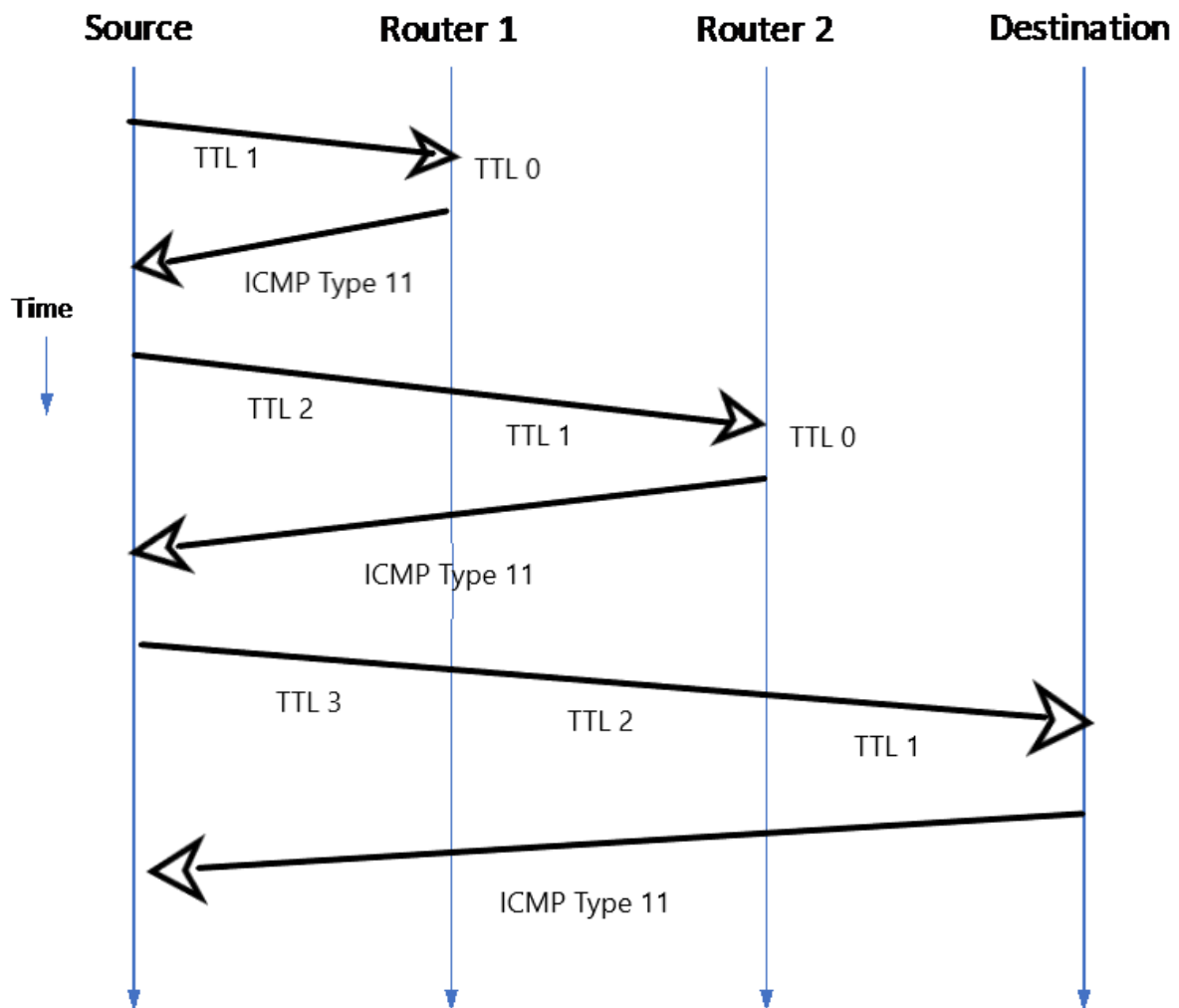
- c) Now look at the ICMP messages coming BACK to your computer.

What is the "Message Type" of the ICMP packets (i.e. you need to expand the ICMP data where the packet header information is displayed) being sent to your PC? Are they all the same? Why?

The "Message Type" of the ICMP packets being sent TO my PC will be Type 0 (Echo Reply) and Type 3 (Destination Unreachable). The "Message Type" of the ICMP packets are not all the same. There will be different error messages depending on what is happening

in my computer. If the destination is unreachable, then the type 3 Message Type will be sent to my PC (or the original sender). If there is an echo reply, then a type 0 Message Type will be sent to my PC (or the original sender). Type 11 message could also be sent. We will receive this message when we are waiting for a ping.

- d) From your understanding of this experiment, sketch²⁴ on the following timeline diagram (for which just two intermediate hops are shown between end computers) a representation of how the ICMP protocol is used by traceroute to provide a map of the 'hops' from computer to computer. Can you think of a network fault situation where this sort of information would be useful?



²⁴ Just use the drawing tools in Word – we do not need great accuracy.

10. Conclusion

In this laboratory exercise, you have been introduced to the Wireshark packet capture software and two utilities (*ping* and *tracert*) that can be used to probe network operation. By comparing the results that we obtain from each tool, we're able to confirm proper understanding of these network tools.

Before finishing, could each student please click on the "Feedback" icon for this laboratory on Moodle, to record the percentage of RTT to the remote website you probed using *tracert* which you estimated was due to propagation (Section 6 above). This will allow us to look at the overall average result. There is also an opportunity to provide some brief, anonymous feedback on this laboratory exercise.

Finally, please submit this completed report via Moodle by the stated deadline. In so doing, please be aware of the following:

- Even if you have had a mark assigned to you during the lab session, this mark will not be registered unless you have also submitted the report.
- Your mark may also not be accepted or may be modified if your report is incomplete or is identical to that of another student.
- By uploading the report, you are agreeing with the following student statement on collusion and plagiarism, and must be aware of the possible consequences.

Student statement:

I have read the University's statement on cheating and plagiarism, as described in the *Student Resource Guide*. This work is original and has not previously been submitted as part of another unit/subject/course. I have taken proper care safeguarding this work and made all reasonable effort to make sure it could not be copied. I understand the consequences for engaging in plagiarism as described in *Statue 4.1 Part III – Academic Misconduct*. **I certify that I have not plagiarised the work of others or engaged in collusion when preparing this submission.**

11. References

- [1] Wireshark Network Protocol Analyser home page. URL <http://wireshark.org/>
- [2] Wireshark Network Protocol Analyser User's Guide. URL <http://wireshark.org/docs/>
- [3] Wireshark Network Protocol Analyser wiki. URL <http://wiki.wireshark.org/>
- [4] Wireshark Network Protocol Analyser FAQ. URL <http://wireshark.org/faq.html>

– END –