Q & A for Assignment 3

Q: In a provided frag trace, datagrams are in two fragments (the original with the mf and the second without). Is this one fragment or two fragments?

Ans: (1) In your example, each UDP datagram has two fragments, and the corresponding output should be:

> "The number of fragments created from the original datagram is: 2
>
> The offset of the last fragment is: xxx"

In addition, note that an intermediate route will only respond one ICMP error (instead of two) for this two fragments.

Q: Should we print fragment info for non-fragmented packets? The assignment mentions 0 for no fragments. But it seems silly to print 0 for every unfragmented datagram.

Ans: Follow the spec:

> "Note that 0 means no fragmentation. Print out the offset (in terms of bytes) of the last fragment of the fragmented IP datagram. Note that if the datagram is not fragmented, the offset is 0."

Q: How to find a match between packets from source and returned error messages?

Ans: Please read the Miscellaneous Section in the assignment.

"If the tracefile is captured in Linux, the source port number included in the original UDP can be used to match against the ICMP error message. …. We will not test your code with a trace file not falling in the above cases."

More notes on Win traces:

In Win traces, you need to match the sequence number between ICMP echo and ICMP error message. Both win traces work fine in this way of match.

To give you more details, you need to be careful about the "seq number" in ICMP error message that should be used for matching. Using win_trace2 as an example:

* The seq number in the first frame (which is ICMP echo) is: 0X 0222 2202 (16 bits). (refer to: https://datatracker.ietf.org/doc/html/rfc792)

* The seq number in the second frame (which is ICMP type 11 error) is also 0X 0222 2202 (16 bits). Please be advised that this seq number needs to be found from the "the first 64 bits of the original datagram's data," as defined in ICMP type 11 error. (refer to: https://datatracker.ietf.org/doc/html/rfc792)

Q: Are there any sample files for R2?

Ans: For requirement 2 of Assignment 3, you need to work on two groups of trace files, each group having 5 trace files. The two groups of sample files can be found in bright space (published together with Assignment 3)

Q: How to estimate RTT if a datagram has multiple fragments?

Ans: If you read the ICMP standard (https://datatracker.ietf.org/doc/html/rfc792) carefully, you should know that "ICMP messages are only sent about errors in handling fragment zero of fragemented datagrams. (Fragment zero has the fragment offeset equal zero)." In other words, if a datagram has multiple segments, each router only reports ONE "TTL exceeds" message (corresponding to the first fragment) to the source node. I double checked the trace file that include multiple fragments and confirmed that the above rule has been followed. For instance, the first datagram has two segments (both having TTL=1), but the source node only receives one "TTL exceeds" error message.
Therefore, in the estimation of RTT for each fragment, you should use this returned ICMP error message to match the multiple fragments sent from the source node.

Q: Why do multiple routers return an error message for UDP with same TTL in a trace file?

Ans: This phenomenon may look strange in theory, but is quite normal in practice: Multiple UDP messages with the same TTL value may hit different routers due to potential load balancing in the intermediate routers. In this case, you just list all the routers, in the same order that they appear in the trace file.

Q: How to exclude irrelevant UDP packets?

Ans: If you capture your own tracefile and do not correctly set up the wireshare configuration, your tracefile may include irrelevant UDP packets (e.g., UDP packets used by NBNS protocol). Those UDP packets should be excluded from the analysis.

One way to do that is to filter UDP packets by dest. port number. Your code should handle this part or otherwise your program may return incorrect result. For these traceroute implementations the dest. port number is incremented by 1 for each probe

so **the highest port number for any given run is equal to *33434 + (max-ttl * probes-per-hop - 1)***. The default setting for *max-ttl* is usually 32 or 64, depending on the implementation. The default *probes-per-hop* is usually 1 or 3. Using the more liberal defaults of 64 and 3, the range would be **33434-33625**. In practice, a max-ttl of 32 is usually more than enough in which case the range would be **33434-33529.**

You can use a condition check to find the right UDP packets:

(dst_port >= 33434) && (dst_port <= 33529)

Q: What exactly the meaning of the original datagram? Do you only want the number of fragments and offset for the first router? Or should we be printing these out for every router? So if there were 10 routers in the trace file then we would output this information 10 times, once for each hop?

Ans: "Original datagram" means the datagram sent from the source node. In addition, in Assignment 3, one original datagram <-------> one corresponding returned ICMP error message.

For more explanation, Refer to https://datatracker.ietf.org/doc/html/rfc792

Since the number of fragments and the offset in the last fragment remain unchanged across various original datagrams, let's just print this information once, as shown in the sample output in the first part of the assignment.

Q: I am confused about requirement 1: "The values in the protocol field of IP headers:" Are we supposed to only include the protocol fields we use? So, for Linux we only use UDP and ICMP (as shown in your example) so we shouldn't include other ones such as TCP and DNS.

Ans: Ignore TCP and DNS, and only consider ICMP and UDP.

Q: I was just wondering if you want us to show our work/code for how we get the answers for Part 2 of Assignment 3. Or is just our final answers fine?

Ans: It is fine if you only show the final answer for part 2. But I highly recommend that you show how the final answer is obtained. Giving intermediate steps can help TA mark your assignment. Otherwise, when your final answer is wrong, you may lose a large portion of your mark.

Q: From the sample output, if the source sends multiple fragments for each TTL, do we need to print out this information once or multiple times, with each corresponding to one TTL value?

Ans: Since the number of fragments and the offset in the last fragment remain unchanged across various original datagrams, let's just print this information once, as shown in the sample output in R1.

Q: Why is a fragmented packet assembled at the source node in traceroute-frag.cap? I remember you told us fragment reassembly only happens at the destination node.

Ans: The fact that Wireshark "reassembled at #xx" does not mean fragment reassembly occurs at the source node. Wireshark lists the re-assembled packet in the trace file to help people easily view the whole packet for traffic analysis purpose. The actual packet delivered on the wire is not the "reassembled" packet. Using "traceroute-frag.cap" as an example:

* packet #21 is the first fragment (1514 bytes on the wire and captured by wireshark, including 1480 bytes payload viewed from the IP layer);

* packet #22 is the second fragment (534 bytes on the wire and captured by wireshark, including 500 bytes payload viewed from the IP layer, but Wireshark lists the whole packet of total 1480+500=1980 payload so that you can quickly view fragements #21 and #22 were originally from the same packet.

In summary, the packet information shown by Wireshark does not mean that fragment reassembly happens at the source node. I emphasize again that only the destination node performs fragment reassembly.