Ali Geniş - 2739571 - CENG519

Term Project Phase 2 Report

Implementation Details

My chosen topic is implementing a covert channel into the TTL values. TTL value is an 8-bit field in the IP header. The most common values for TTL are 64, 128, or 255 which is its maximum. In this phase, I implemented a covert channel with three different encoding parameters. The sender from sec accepts string messages as parameters and converts them to the desired encoding.

1. ASCII Encoding

This part of the encoding is very straightforward. The ASCII representation of the character is used as a TTL value. Printable ASCII chars are between 32 and 126.

2. Binary Encoding

This encoding is also straightforward. The given message is converted to zeros and ones. Then a predetermined 0 and 1 value is set as a TTL for a specified packet. I used 64 as a 0 and 128 as a 1 in my implementation.

3. Difference Encoding

This is another approach I encountered while reading Zender's paper [1]. It is described for binary, but I modified this encoding a little bit. For the first letter, it is exactly the same as ASCII encoding. After that following letters are encoded according to these principles:

- If you add new ASCII to old encoding and it exceeds 255, subtract it from the previous encoding.
- If you subtract the new ASCII from the old encoding and it is below 1, add it to the previous encoding.
- If both adding and subtracting are available and valid, choose the operation with equal probability.

Experiments

I measured two aspects of my three configurations: capacity in bits/packet and RTT.

1. Capacity:

1.1. ASCII:

The value can range from 32 to 126. Which is 94 different values which can be represented with 7 bits. So the capacity of this configuration is 7 bits/packet. Since the TTL value is 8 bits, it can be said that this configuration is using 87.5% of its maximum capacity.

1.2. Binary:

In this approach, a predetermined TTL value is sent for binary value 0 or 1. It is straightforward that capacity is 1 bit/packet. This method utilizes 12.5% of the maximum capacity.

1.3. Difference:

It is very similar to ASCII encoding, but I assume that this is hard to detect compared to standard ASCII version of it. The data encoded is the same as the standard ASCII, so the capacity is.

Tablo 1: Capacity

Encodings	Capacity(bits/packet)	Capacity(utilized/max. Available) x100
ASCII	7	87.5
Binary	1	12.5
Difference	7	87.5

2. RTT (Round Trip Time):

The experiment is done with 32 packets for each configuration. Average RTT and 95% confidence intervals can be seen in Figure 1 and Figure 2. (values are in seconds)

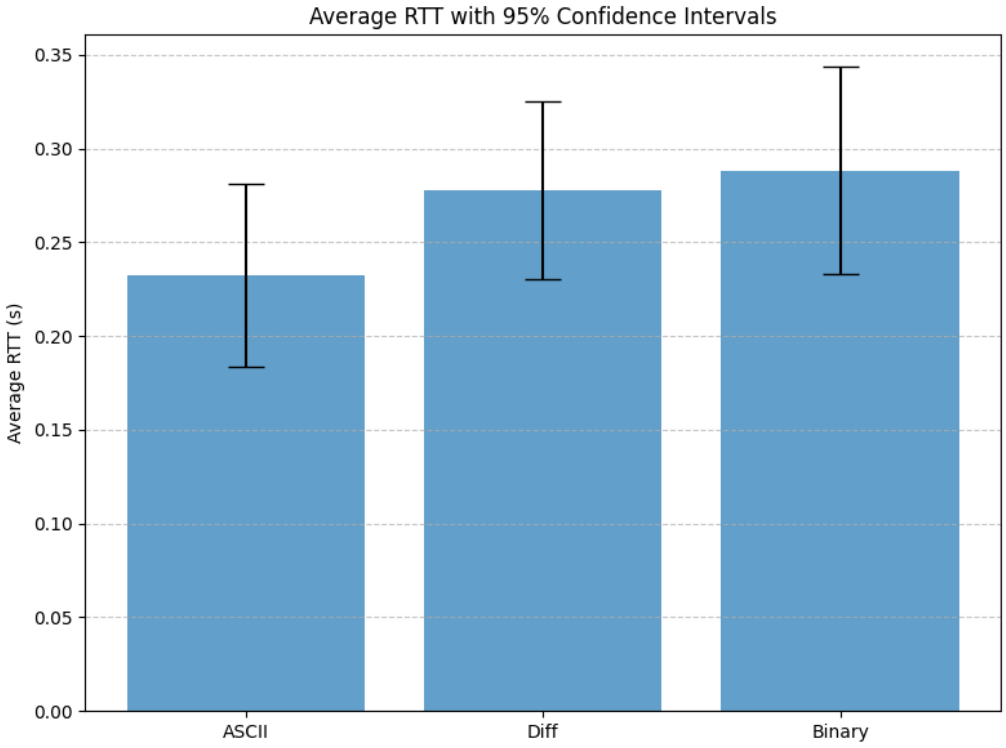
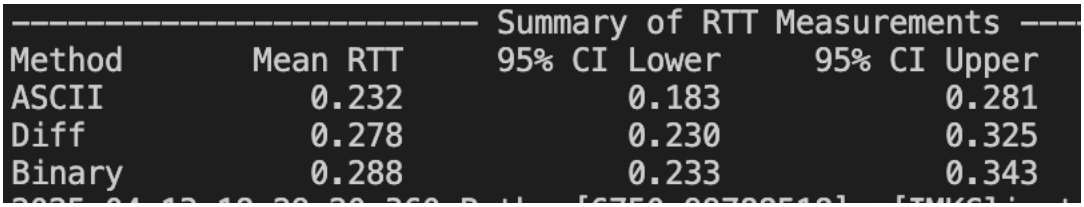


Figure 1



--- Summary of RTT Measurements ---

Method	Mean RTT	95% CI Lower	95% CI Upper
ASCII	0.232	0.183	0.281
Diff	0.278	0.230	0.325
Binary	0.288	0.233	0.343

Figure 2: Numerical Values for RTTs

Assumptions / Limitations

- UDP with acknowledgment is used in implementation. TCP is not used since the TTL value can be set once per connection. So for different TTL values, the connection needs to be reestablished for each packet, which is not meaningful.
- 'Receiver' is assumed to be available before running 'Sender'. If not, the sender waits infinitely for the acknowledgement. I can overcome this by removing acknowledgement, but it results in another problem. Methods that depend on previous packets, such as binary and diff is won't work properly if they miss any packet.

References

[1] Zander, Sebastian & Armitage, Grenville & Branch, Philip. (2007). An Empirical Evaluation of IP Time To Live Covert Channels. 42 - 47. 10.1109/ICON.2007.4444059.