UDP Checksum Calculation & Why UDP?

Let's look at how the UDP checksum is calculated in-depth, why UDP would ever get used, and applications that use UDP.

WE'LL COVER THE FOLLOWING

UDP detects if any changes were introduced into a message while it traveled over the network. To do so, it appends a 'checksum' to the packet as a field that can be checked against the message itself to see if it was corrupted. It's calculated the same way as in TCP. Here's a refresher with some extra information:

Checksum Calculation

1. The payload and some of the headers (including some IP headers) are all divided into 16-bit words.
2. These words are then added together, wrapping any overflow around.
3. Lastly, the one's complement of the resultant sum is taken and appended to the message as the checksum.

Note Also, note that if a message cannot be perfectly divided into 16-bit chunks, then the last word is padded to the right with zeros. This is

> bit chunks, then the last word is padded to the right with zeros. This is
>
> only for checksum calculation though! The actual message does not have these zeros.

Here’s a visual of how the checksum for a datagram is calculated:

```
1110011001100110
0101010101010101
1100010000100010
```

Assume a message can be divided into these 3 16-bit words

1 of 24

```
 1110011001100110
+0101010101010101
 ________________
```

We will start off by adding the first 2

2 of 24

```
 1110011001100110
+0101010101010101
 ----------------
                1
```

0 + 1 = 1

3 of 24

```
 1110011001100110
+0101010101010101
 ----------------
               11
```

1 + 0 = 1

4 of 24

$$\begin{array}{r} {\scriptstyle 1} \\ 1110011001100110 \\ +0101010101010101 \\ \hline 011 \end{array}$$

1 + 1 = 10 where the 1 is carried

5 of 24

$$\begin{array}{r} {\scriptstyle 1} \\ 1110011001100110 \\ +0101010101010101 \\ \hline 1011 \end{array}$$

0 + 0 + 1 (from the carry) = 1

6 of 24

$$\begin{array}{r} 111001100110\overset{1}{0}110 \\ +0101010101010101 \\ \hline 11011 \end{array}$$

0 + 1 = 1

7 of 24

$$\begin{array}{r} 111001100110\overset{1}{0}110 \\ +0101010101010101 \\ \hline 111011 \end{array}$$

1 + 0 = 1

8 of 24

$$\begin{array}{r} 11100110\overset{1}{0}110\overset{1}{0}0110 \\ +0101010101010101 \\ \hline 0111011 \end{array}$$

1 + 1 = 10 where 1 is carried

$$\begin{array}{r} 11100110\overset{1}{0}110\overset{1}{0}0110 \\ +0101010101010101 \\ \hline 110111011 \end{array}$$

0 + 1 = 1

$$\begin{array}{r} 11100110\overset{1}{0}110\overset{1}{0}110 \\ +0101010101010101 \\ \hline 1110111011 \end{array}$$

1 + 0 = 1

11 of 24

$$\begin{array}{r} 1110\overset{1}{0}110\overset{1}{0}110\overset{1}{0}110 \\ +0101010101010101 \\ \hline 01110111011 \end{array}$$

1 + 0 = 1

12 of 24

$$\begin{array}{r} 1110\overset{1}{0}110\overset{1}{0}110\overset{1}{0}110 \\ +0101010101010101 \\ \hline 101110111011 \end{array}$$

$1 + 0 + 0 = 1$

$$\begin{array}{r} 1110\overset{1}{0}110\overset{1}{0}110\overset{1}{0}110 \\ +0101010101010101 \\ \hline 1101110111011 \end{array}$$

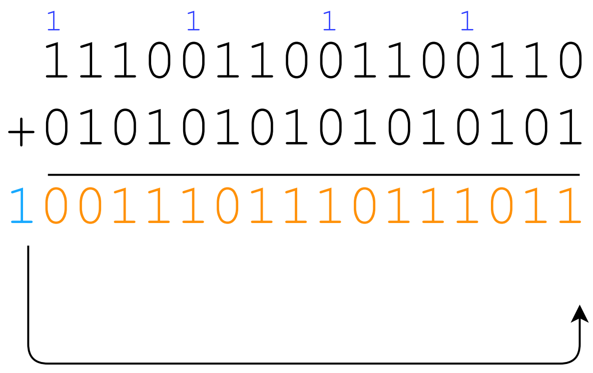
$0 + 1 = 1$

```
     1   1   1
 1110011001100110
+0101010101010101
 ----------------
   11101110111011
```

1 + 0 = 1

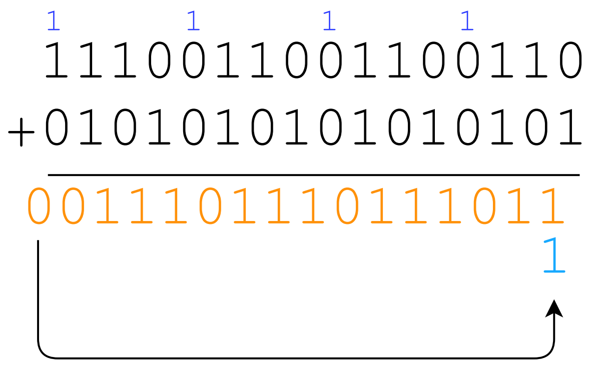
```
 1   1   1   1
 1110011001100110
+0101010101010101
 ----------------
10011101110111011
```

1 + 1 = 10. Now, notice that the 1 is an overflow, so we can wrap it around!



```
  1   1   1   1
  1110011001100110
 +0101010101010101
  ----------------
 10011101110111011
```

We bring the '1' to the right and add it to the rest of the word!



```
  1   1   1   1
  1110011001100110
 +0101010101010101
  ----------------
  0011101110111011
                 1
```

We bring the '1' to the right and add it to the rest of the word!

```
  1   1   1   1
  1110011001100110
 +0101010101010101
  ----------------
 0011101110111011
+               1
  ----------------
```

We bring the '1' to the right and add it to the rest of the word!

```
  1   1   1   1
  1110011001100110
 +0101010101010101
  ----------------
 0011101110111011
+               1
  ----------------
 0011101110111100
```

We sum using the same rules as before to get this

$$\begin{array}{r} 0011101110111100 \\ +1100010000100010 \\ \hline \end{array}$$

$\longleftarrow$ third word

We sum using the same rules as before to get this

$$\begin{array}{r} 0011101110111100 \\ +1100010000100010 \\ \hline 01111111111011110 \end{array}$$

$\longleftarrow$ Final sum

We sum using the same rules as before to get this

$$\begin{tikzcd}[column sep=large]
\begin{array}{r} 0011101110111100 \\ +\,1100010000100010 \\ \hline 01111111111011110 \end{array} \arrow[d] & \text{Final sum} \arrow[l] \\
1000000000010000 & \text{Final checksum} \arrow[l]
\end{tikzcd}$$

We sum using the same rules as before to get this

10000000000100001

The final checksum!

At the receiving end, UDP sums the message in 16-bit words and adds the sum to the sent checksum. If the result is **1111111111111111**, the message was not corrupted. If the result is otherwise, it was.

What if the checksum field gets corrupted?

If the checksum itself gets corrupted, UDP will assume that the message has an error.

Why UDP?

You might be wondering why would anyone use UDP when it has so many apparent drawbacks and doesn’t really do anything? Well, there are actually a number of reasons why UDP would be a good choice for certain applications.

1. UDP can be **faster**. Some applications cannot tolerate the load of the retransmission mechanism of TCP, the other transport layer protocol.

2. **Reliability can be built on top** of UDP. TCP ensures that every message is sent by resending it if necessary. However, this reliability can be built in the application itself.

3. UDP gives **finer control** over what message is sent and when it is sent. This can allow the application developer to decide what messages are important and which do not need concrete reliability.

4. Going on points 3 and 4, **UDP allows custom protocols to be built on top of it**.

    - In fact, Google’s transport layer protocol, **Quick UDP Internet Connections (QUIC)**, pronounced *quick*, is an experimental transport layer network protocol built on top of UDP and designed by Google. The overall goal is to reduce latency compared to that of TCP. It’s used by most connections from the Chrome web browser to Google’s servers!

5. With the significantly smaller header gives UDP an edge over TCP in terms of reduced transmission overhead and quicker transmission times.

Well Known Applications That Use UDP

Well-Known Applications That Use UDP

Xbox Live

Xbox live is built on UDP.

Yes, Xbox live runs on UDP!

Name Translation

Yes, DNS uses UDP! In the case of failed message delivery, DNS either:

1. Resends the message.
2. Sends the message to some other server.
3. Gives a failure message.

Using UDP instead of TCP makes DNS and consequently, web browsing significantly faster.

Network Management

Network management and network monitoring is done using a protocol called Simple Network Management Protocol and it runs on UDP as well.

Quick Quiz!

1

An application implements its own mechanisms of checksums and retransmissions. UDP is the ideal choice of transport layer protocol for this application.

COMPLETED 0%

1 of 4

Let’s look at some actual live UDP packets in the next lesson with TCPDUMP!