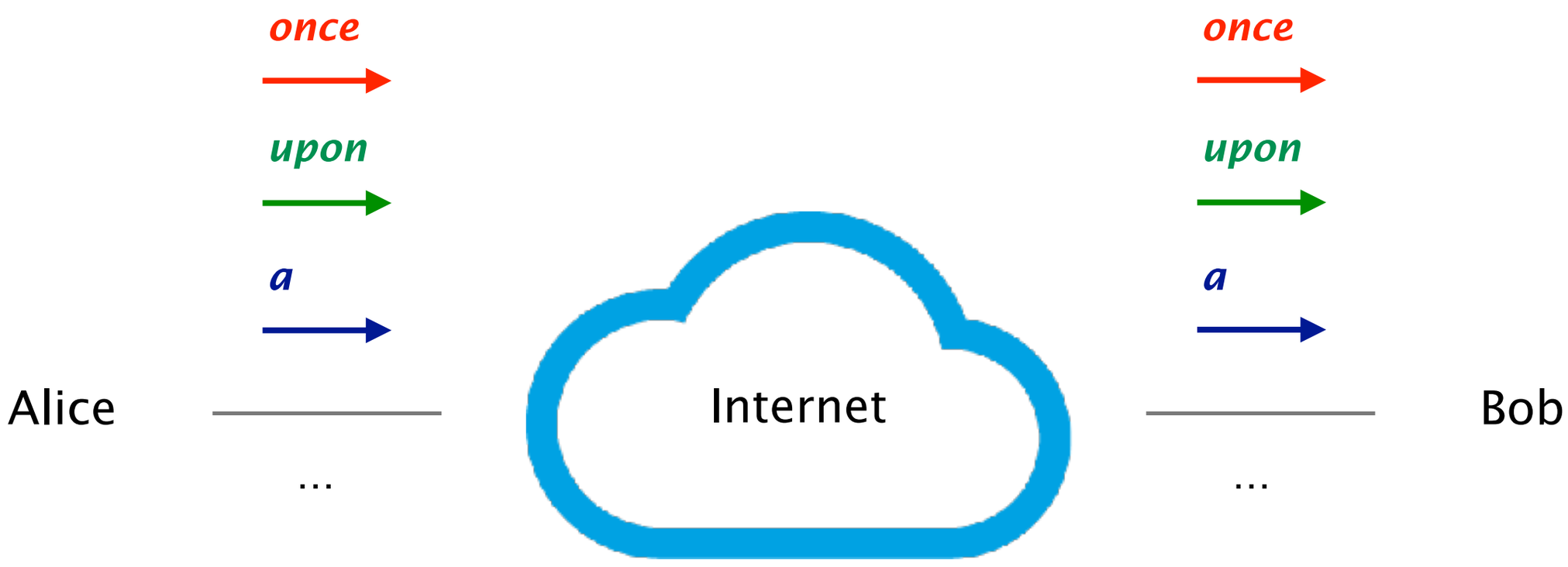
Now, it's your turn

...to design a Internet protocol

instructions given in class

Let’s consider that Alice wants to transmit a text to Bob, word-by-word, via the Internet



Your job is to design a reliable transport protocol running on Alice's and Bob's computer

	property
correctness	Bob should read exactly what you've typed in the same order, without any gap
timeliness	Bob should receive the complete text as fast as possible minimize time until data is transferred
efficiency	Minimize the use of bandwidth don't send too many packets

The number in front of you is

your group number

Your task

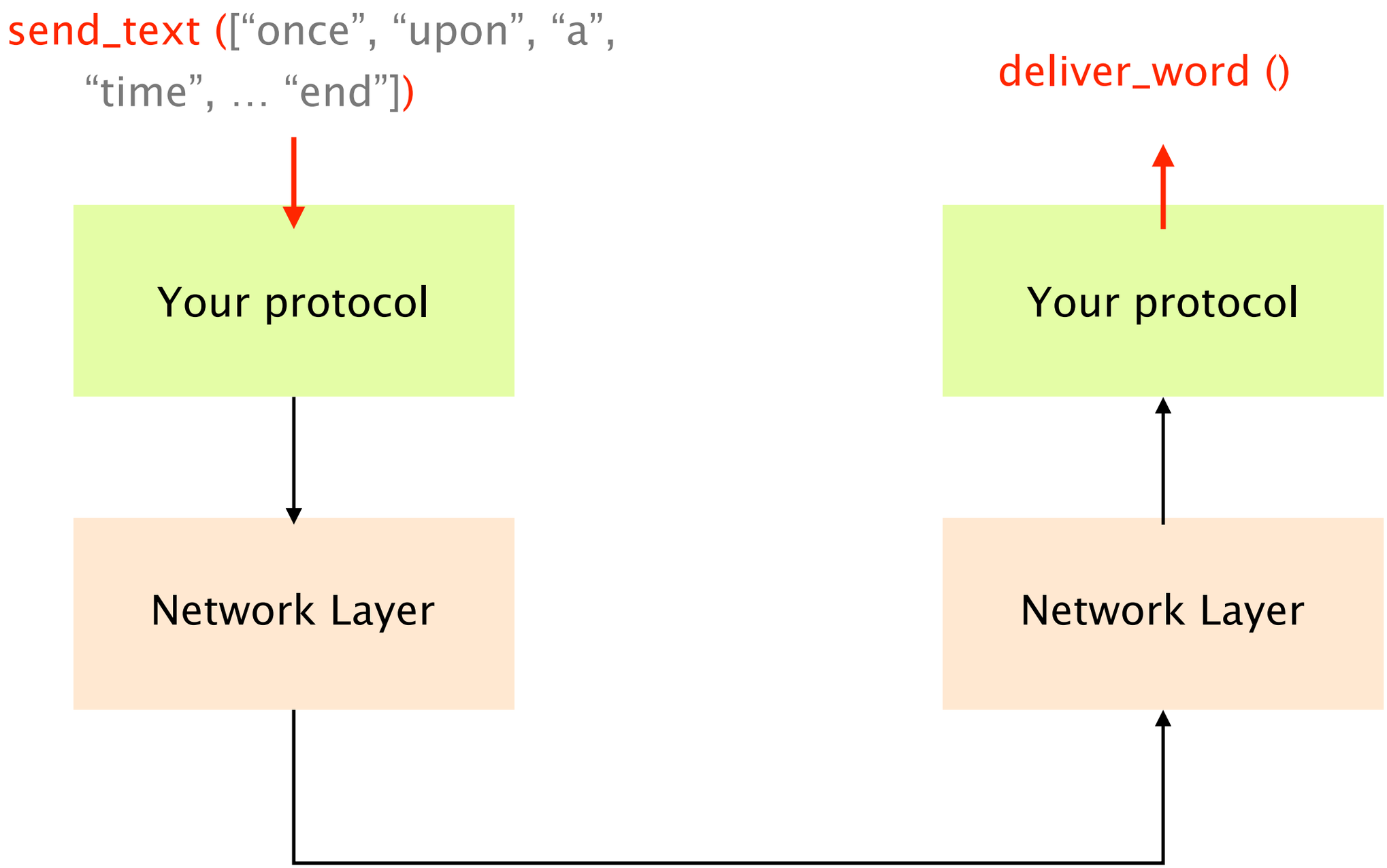
Design a protocol that can deal with packet loss, corruption, reordering and duplication

Design a protocol that can deal with packet loss, corruption, reordering and duplication

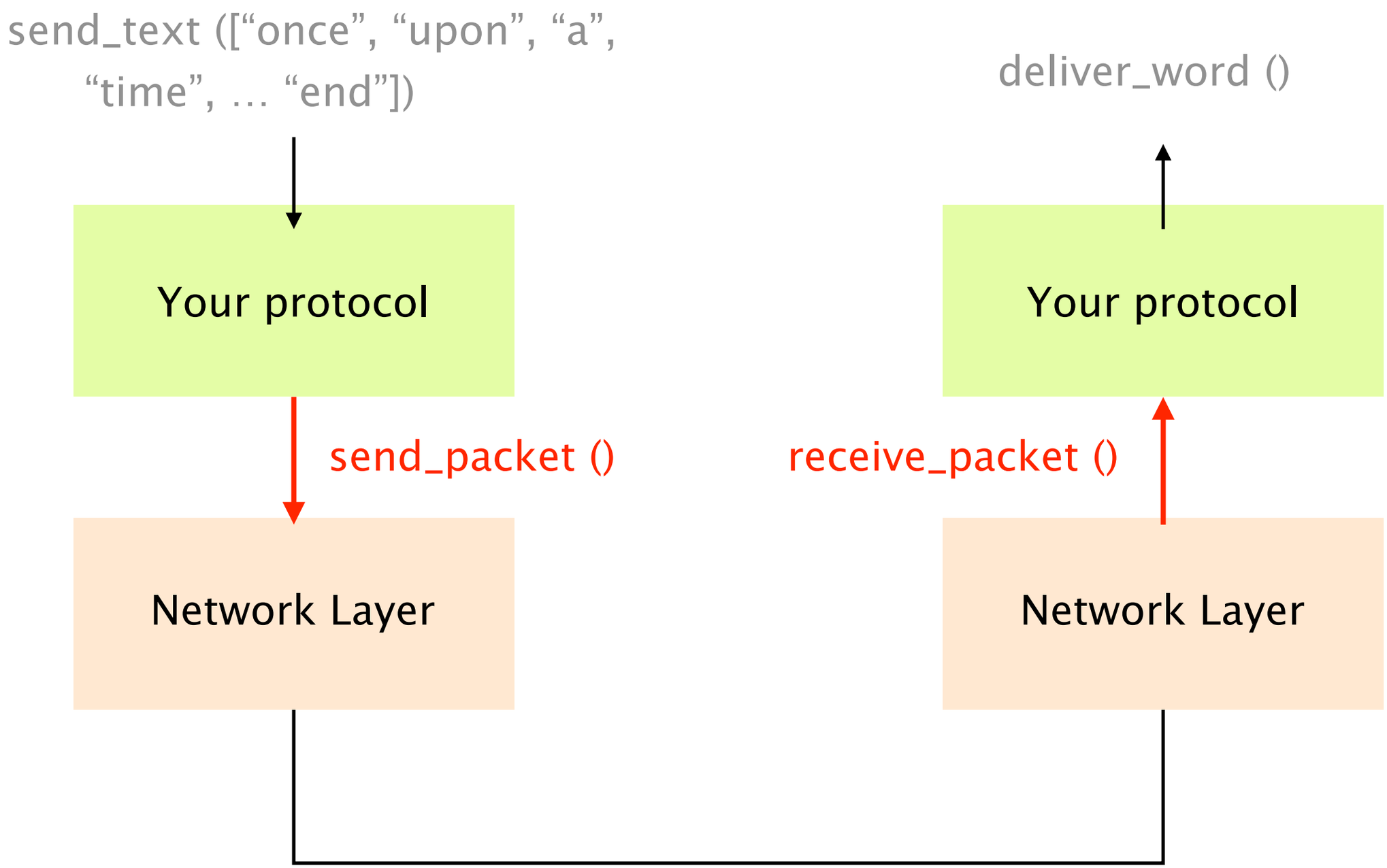
what fields do you add to the packets?

what code do you run on the end-points?

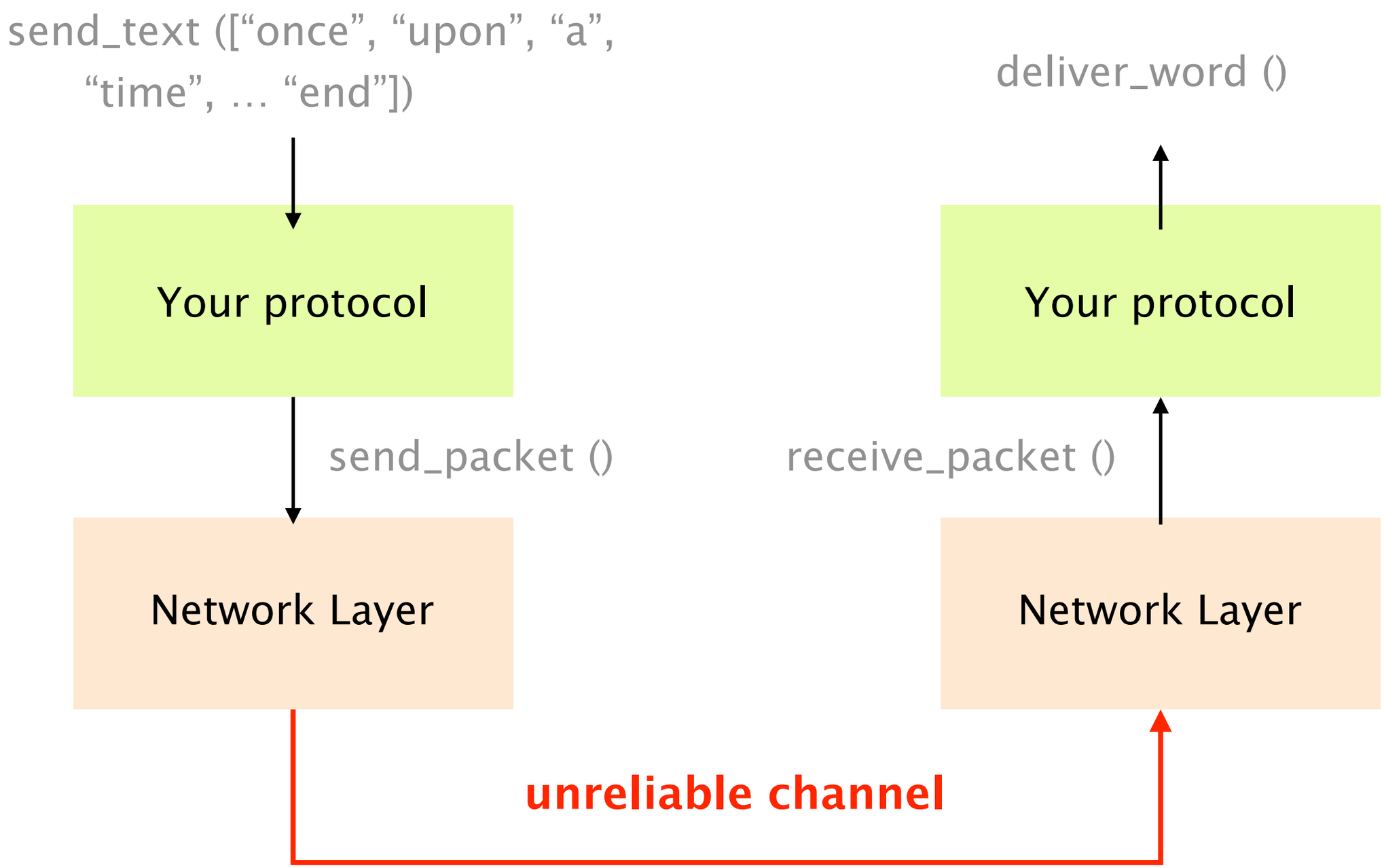
Your protocol receives a list of words on one host, and deliver them, in order, one-by-one, on another host



Your protocol uses 2 primitives of the network layer: `send_packet` and `receive_packet`



Packets can be lost, corrupted, reordered or duplicated



first Write down the pseudo-code of a protocol
that sends at most 1 word/packet at a time.
Each packet can be lost, corrupted or duplicated.

then Think about how you would extend your protocol
so that it can send *multiple* words/packets at a time.
How you deal with packet reordering?

output The procedure you run on the sender and receiver
The header(s) you need to add to the packets
An idea of how you support >1 outstanding packets

You have ~15 minutes.

Any group member should be able to present its group's protocol

Solution for the single packet case

Alice

```
for word in list:
    send_packet(word);
    set_timer();

upon timer going off:
    if no ACK received:
        send_packet(word);
        reset_timer();
upon ACK:
    pass;
```

Bob

```
receive_packet(p);
if check(p.payload) == p.checksum:
    send_ack();

    if word not delivered:
        deliver_word(word);
else:
    pass;
```