Yet Another Reliable UDP (YARU)

Introduction

An application layer transport protocol built on top of UDP[1] to add properties of ARQ (Automatic Repeat reQuest). It is designed to be small and easy to implement, while still providing the minimum functionality to enable reliable transmission of data. Therefore, it lacks most of the features of a regular TCP[2], such as handshakes, flow control, and congestion control.

Features

YARU provides the following features to handle the various shortcomings of the underlying networks and to enable forming a reliable connection.

Error Detection

Uses a simple MD5 checksum to detect bit errors in a transmitted packet.

Sequence numbers

Used for sequential numbering of packets of data flowing from sender to receiver and allowing the receiver to detect duplicate packets.

Acknowledgements

Special packets sent by receiver to indicate that a particular packet has been received correctly. Useful when the underlying network can drop packets.

Pipelining

Allows multiple packets to be transmitted concurrently, so that sender utilization can be increased over a stop-and-wait mode of operation. YARU follows the **Selective Repeat** scheme for sending acknowledgements, so each packet gets ACK'ed individually.

Timeouts

Timers are used by the sender to protect against lost packets, where each packet has its own timer. If timeout occurs, the packet has to be retransmitted.

Packet layout

Bitrange	0 63	64 79	80 207
Field	Sequence Number	Length	Checksum

- **Sequence Number:** 64-bit sequence number of the current packet. For ACK packets, this value should be the same as that of the packet being acknowledged.
- **Length:** 16-bit number, which is the length in octets of this packet including this header and the data. Should be at least `208` (size of header), and at most `65507` (maximum allowed for UDP over IP). This field should be `0` for ACK packets.
- **Checksum:** 128-bit checksum of the YARU packet, calculated by first creating the entire packet with checksum field as `0`, and then storing its MD5 hash into the correct location.

Interface to underlying network

YARU is designed to work on top of the unreliable properties of UDP such as packet dropping, out of order transmission and packet corruption. Additionally, the size of each YARU packet is limited to `65507` bytes, as mandated by the length fields in IP and UDP protocols.

Interface to application

YARU guarantees reliable delivery of packets. The application can use the following methods on YARU sockets, analogous to Python sockets[3].

- `bind(address)`: Bind the socket to the given interface and port.
- `connect(address)`: Establish a connection (no handshaking done).
- `write(data)`: Send the data (of type bytes) to the connected socket. The data should be less than `65481` bytes long.
- `read()`: Read the data sent by the connected socket, present in the buffer. If no data is available, it returns an empty string.
- `close([timeout])`: Wait for the connection to close, optionally kill it after timeout seconds (default: 120s).

Beware that there is no flow control, so it is possible to deadlock the socket in certain situations.

References

[1] RFC 768 - User Datagram Protocol
[2] RFC 761 - DoD standard Transmission Control Protocol
[3] socket — Low-level networking interface — Python