Chabu

Multiple Channels over TCP/IP

About this document

The original GDoc:
https://docs.google.com/document/d/1Wqa8rDi0QYcqcf0ecD8GW53nMVXj3ZFSmcF81zAa8g/edit?usp=sharing

Abstract

This is a protocol to run on top of a protocol like TCP. Chabu would be in the OSI-model in layer 6, the presentation layer.

Chabu does not handle packet loss, data corruption, …

Chabu transmits multiple channels over a single TCP connection.

A channel is like a tcp connection, having a «in» and a «out» stream.
Each stream has its own flow control. That means, if the receiver does not consume a streams data, the other streams are not blocked.
There are up to 256 channels.

Global Parameters

There are some definitions that must be the same at both communication partner sides.
Those parameters are exchanged at communication start. If there is a incompatible mismatch, the connection will be aborted.

Protocol version

The current value is «1».

Version	What changed
1	Initial version

Byte order

The default is «big endian», known as network byte order.
The byte order can be defined for the driver at compile time.
Note: In this document, big endian is used for the examples.

Maximum Payload Size

The default is 1400.
Valid values are 1 .. 0xFFFF.
This influences the amount of memory needed to be held in transmitter and receiver.

Coding

Integral types

This protocol makes use of unsigned integer types: UINT8 and UINT16.

Blocks

Block are the parts transmitted over the tcp connection.

Structure

Type	Name	Meaning

UINT8	CID	channel id
UINT16	PS	payload size
UINT16	SEQ	sequence number
UINT16	ARM	arm number
UINT8 ...	Payload	The raw bytes to be transferred

CID - Channel Id

0 .. 255

PS - Payload size

The payload is the number of bytes contained in the block. The next block starts immediately after the payload.

SEQ - Sequence block number

The absolute index the block. The first block starts with 0.
If PS is 0, the SEQ is not used.
If SEQ reaches UINT16_MAX it wraps around to 0.

ARM - Arming block number

At startup, before the first ARM is received, it is initialized with UINT16_MAX.
The absolute index, up to what index the opposite endpoint can send blocks with ISEQ.
E.g. if ARM is 10, the sender can send up to block with SEQ=10.

Start up

Protocol parameter specification

In the beginning both endpoints send the protocol parameter specification and compare to the values received from the opposite endpoint.
If there is a mismatch, the connection is aborted.

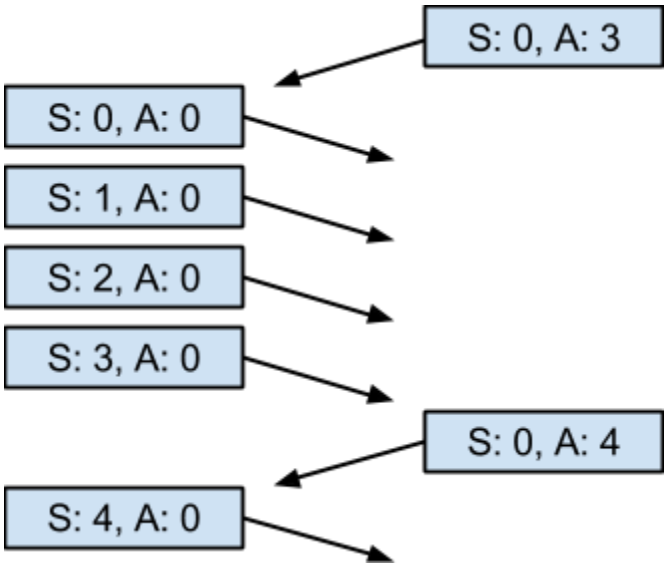
Type	Name	Meaning
UINT8	Version	Protocol version
UINT8	BO	Byte order: 0 = little endian, 1 = big endian
UINT16	MPS	Maximum Payload Size
UINT8	MCID	Maximum channel id

Reading is done in 2 steps. First the «Version» is read, because depending from this, the amount of bytes to follow depends.

Data handling

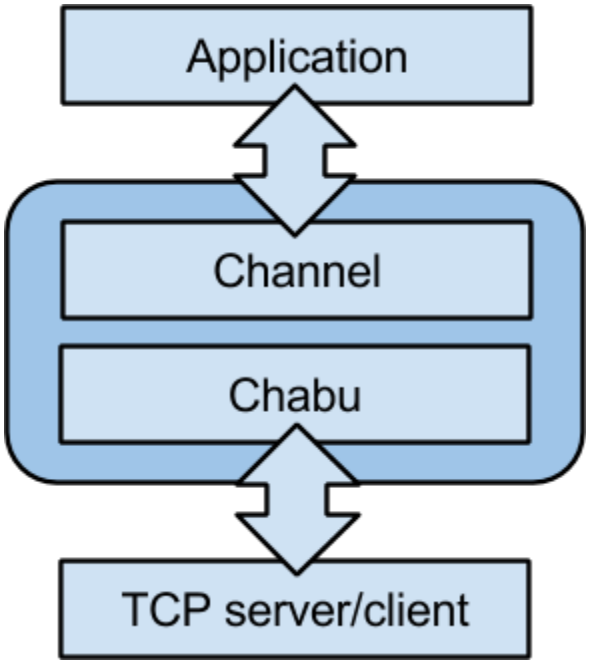
In the beginning each endpoint has ARM = UINT16_MAX, so no endpoint is allowed to send payload.
When an endpoint is ready to receive data, it sends a block with PS = 0 and a ARM value to signal up to how many blocks it is ready to receive.

Flow Control



Chabu integration

The under laying TCP layer



Configuration Example

FTP like service

FTP uses 2 TCP connection. One for the control commands and one for the data transfer. With Chabu, this can be just like 2 channels over a single TCP connection.

Channel 0: used for the control commands. Those are expected to be very short always. So length of 1 receive buffer on both side seems to be OK.
Channel 1: used for data transfer. Here the bandwidth shall be high. To avoid time loss because of round trip time to receive the next ARM, the count of rx buffers is higher. E.g. 10.

Test suite

Two applications communicate with each other. They have 2 TCP connections.

1. Control connection to exchange the information about test data to be transferred over the test connection
2. Test connection: Chabu connection

Type	Name	Meaning

UINT8	CMD RES	Command Result
UINT64	T	Time in ns, e.g. System.nanoTime()
UINT8	CID	Channel
UINT32	TX_CNT RX_CNT RX_BUF_CNT	Number of bytes to transfer or consume

Commands

For all commands, except "Time broadcast", the T is the scheduled time when this shall be started. T is always increasing.

1. Time broadcast, T is the current time when command is generated. This command will be produced regularly, so the receiver can implement a synchronization.
   CMD = 0, T
2. Close application
   CMD = 1, T
3. Await connection, act like server, waiting for connection
   CMD = 2
4. Start Connection
   CMD = 3, T, MCID, MPS
5. Close connection
   CMD = 4, T
6. Add channel
   CMD = 5, T, RX_BUF_CNT
7. Action
   CMD = 6, T, CID, RX_CNT, TX_CNT

Results

All 100ms, for all channels that had rx/tx event, on result record is produced.

1. RES = 1, T, CID, RX_CNT, TX_CNT

Test procedure

The master can either run with a predefined plan or with a random schedule.

A result file is written. It contains the values as CVS and can be examined in Excel