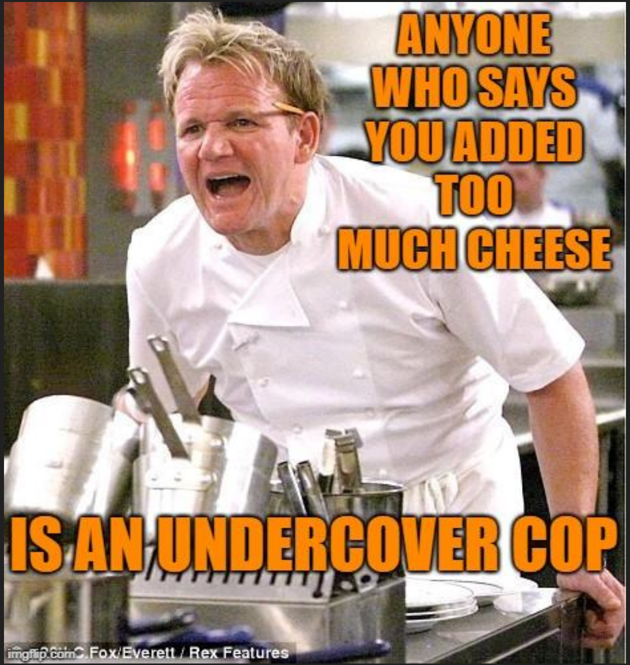
SOFT WG Update
14.08.2020

State of the spec

- ~~Added a section allowing limited-scope implementations~~
- Finalised security considerations
- Cursed at the amount of features…
- We found the spec to work :)


ANYONE WHO SAYS YOU ADDED TOO MUCH CHEESE
IS AN UNDERCOVER COP
imgflip.com
C.Fox/Everett / Rex Features

Interoperability

	SOFT|SOFT	R2FT|R2FT	R2FT|SOFT	SOFT|R2FT
Handshake	yes	yes	-	-
Multiple server sessions	yes	unstable	-	-
Single server session	yes	yes	-	-
Serial file transfer	yes	yes	-	-
Parallel file transfers	yes	yes	-	-
SHA512 Hash Check	yes	yes	-	-
Out-of-order reception	yes	unstable	-	-
Retransmissions	yes	unstable	-	-
Induce Packet Loss param	no	no	-	-
Congestion Control	no	no	-	-
Flow Control	no	unstable	-	-
File List Operation	no	no	-	-
Error Reporting	yes	yes	-	-
File Resume	no	no	-	-
Permission Metadata	no	no	-	-

Rust SOFT

Protocol Design SS20 - Johannes Abel, Joseph Birkner, Peter Okelmann

R2FT

Language: **Rust**

Test Coverage:	**64%**
Lines of Code:	**3229**
Lines of Comments:	**301**

Server:
```
cargo run -- -s
```
Client:
```
cargo run -- 127.0.0.1:42424 testdata/test.txt
```

Public Repo: https://gitlab.pogobanane.de/pogobanane/r2ft

Dependencies

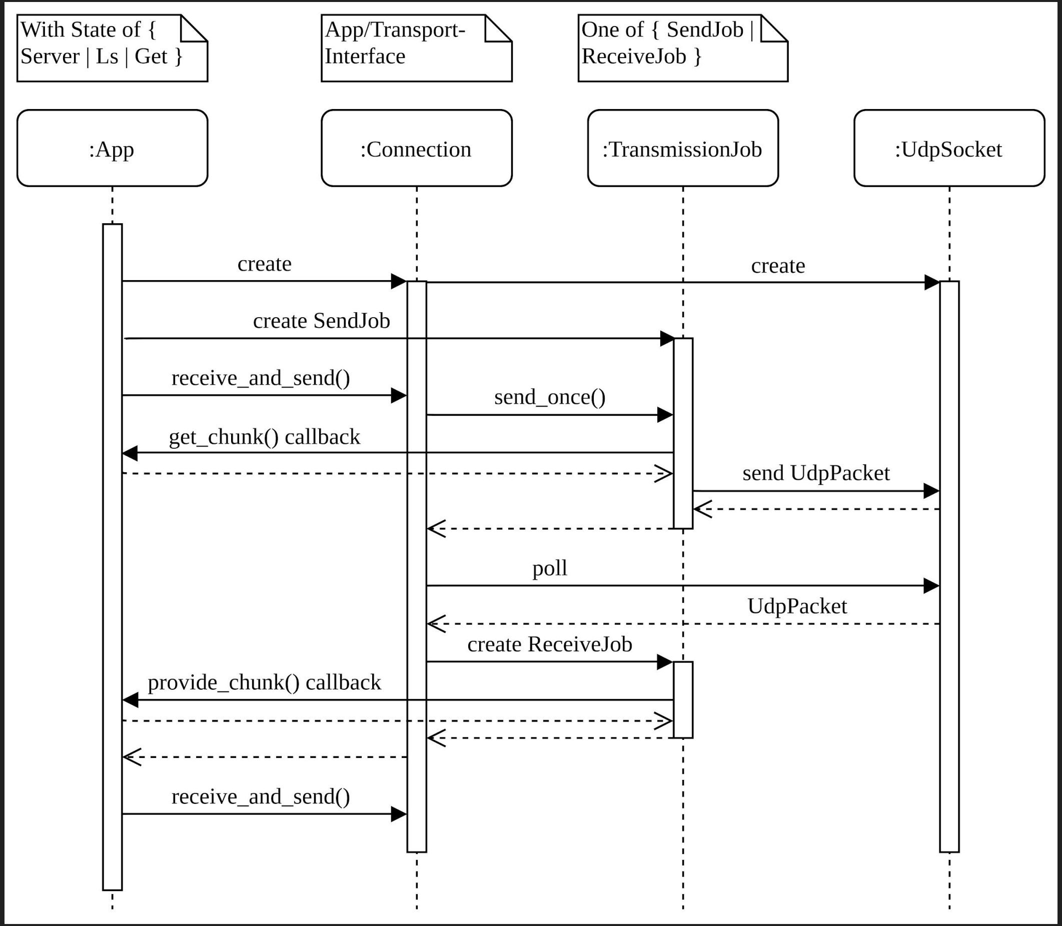
`[dependencies]`	
`clap`	Command line parsing
`env_logger`	Simple Logger
`log`	Logging macros
`byteorder`	Read and write depending on Endianness
`leb128`	Read and write LEB128 numbers
`rand`	Random number generation
`sha3`	SHA3 calculation
`itertools`	Convenience tools for iterators
`num`	For conveniently converting enums numbers
`num-derive`	“
`num-traits`	“

External code (MIT license), modified for our fnv1a32 calculations:
https://github.com/althonos/pruefung/blob/master/src/fnv/fnv32.rs

Module Structure

```
r2ft                Library crate
├── app                 SOFT application layer stuff
│   ├── frame               TLVs + (de-)serialization
│   ├── get                 File retrieval client (run method)
│   ├── ls                  File list retrieval client (run method)
│   ├── server              Runs Server (run method)
│   └── state               State machine for sending and receiving chunks
├── common              Some useful stuff for both layers, mainly traits and macros for TLV parsing
│   ├── fnv1a32             Calculation of FNV1a32
│   └── mtu                 MTU from OS retrieval
├── options                 Command line options parsing
└── transport           SOFT transport layer stuff
    ├── client              Creating a client connection to a server
    ├── common              Transport-global definitions
    ├── connection          Connection interface
    ├── frame               (De-) Serialization of UDP messages
    ├── jobs                Jobs for sending and receiving Objects over a Connection
    └── server              Listening for incoming connections as a Server
main                Binary crate for command line program
```

Control Flow



Peter Okelmann: Interface

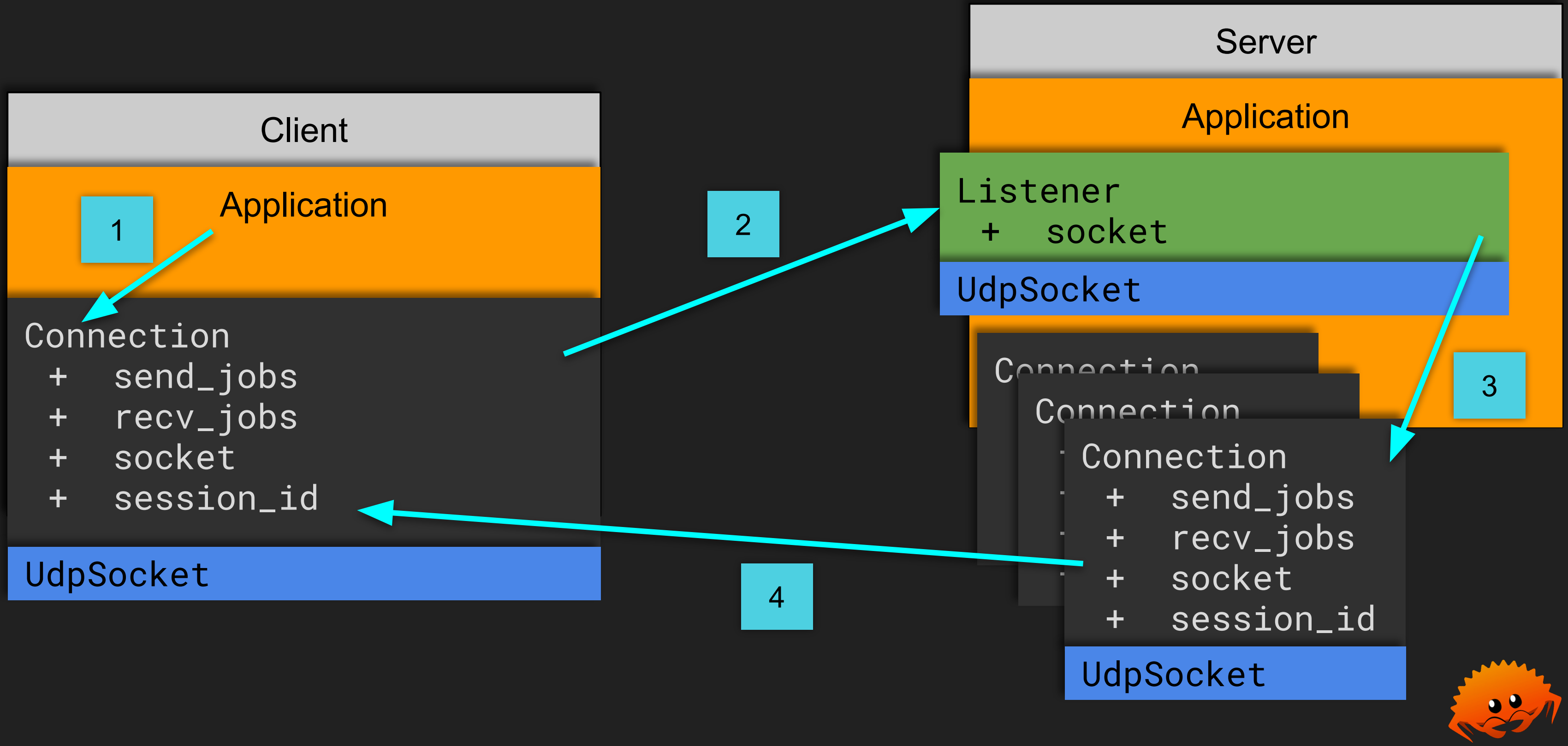
Interface: Transport - App

Goal: Allow concurrent implementation of both parts.

Design Decisions:

- Sync vs ~~async~~ rust
- Choose UdpSocket impl (std, std+thread, mio/poll, tokio, libc)
  -> std::UdpSocket.set_nonblocking()
- Collaboratively single-threaded vs ~~threaded~~

Joseph Birkner: Transport



Johannes Abel: Application

- Command line parsing done in main
- Results in call to server's run or client's get method

- both (client + server) init
  - state machine (abstracts SOFT application layer)
    - manages application layer closures and state
  - connection (abstracts SOFT transport layer)
    - directly (connect) by client, via busy wait (listen) by server

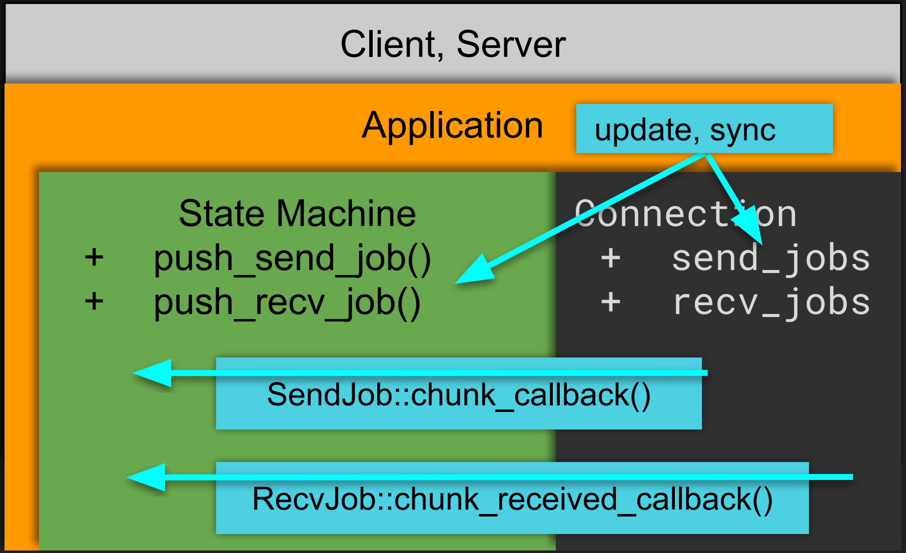
Johannes Abel: Application

- both run loops
  - get closures with corresponding state from app to transport
  - fetches send jobs from application to transport (get closure/ callback on state)
  - register new receive jobs from to application (get on state)
  - periodically call connection.receive_and_send() at connection/ transport layer
  - progresses send and receive of jobs via callbacks

- client creates initial SendJobs for file request

Johannes Abel: Application

- Symmetrical application state machine implementation
- Allows server to retrieve files from client ¯_(ツ)_/¯



🧡

Demo Time!

Any Questions?