Newcomers in golang

Traps that could be avoided

Motiejus Jakštys
motiejus@uber.com
@mo_kelione

Who Am I

- Previous: telco, online games, Amazon.
  - ~4 years of Erlang professionally.
- Lucky to join Ringpop team in Amsterdam:
  - Library for building distributed systems in Golang.
- Moved to Vilnius in Nov 2016.

Agenda:

- Personal experience of ignorance, fight and submission to Golang.
- Things happen in teams. Some team experiences on the way.

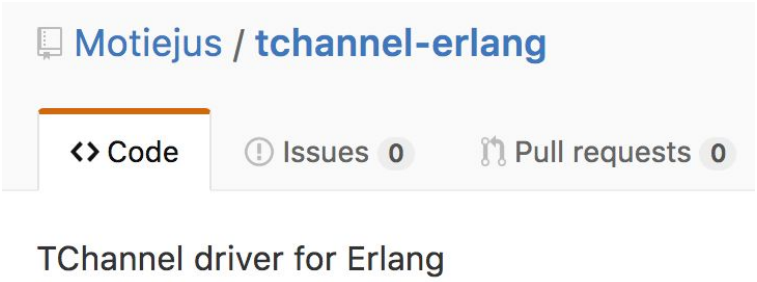
Ignorance

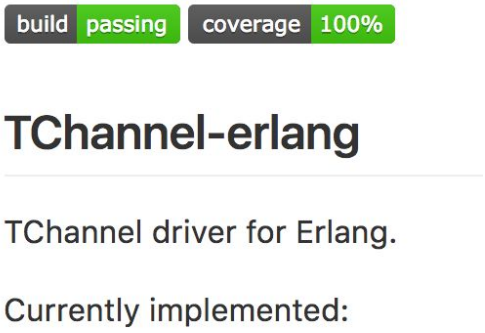
- I joined Uber in Feb 2016.
- I was ignoring Golang at first.
- My first thought was that Go is like Erlang, but with worse runtime tooling. I bet everyone would switch to it.

Image credit: https://followyourgps.files.wordpress.com/

Fight

- Most services use an in-house RPC protocol *tchannel*.
- I created an Erlang tchannel library.

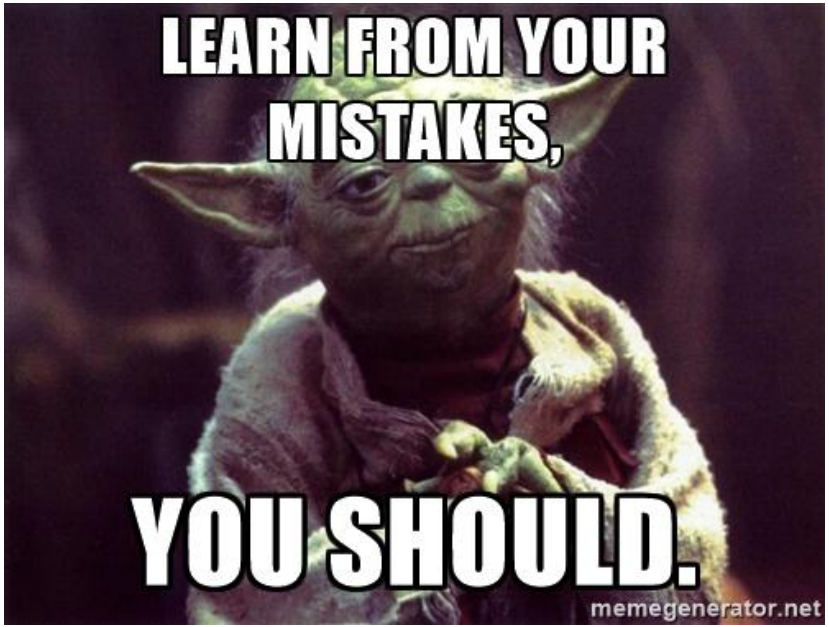




I spent a lot of time for nothing.

- After tchannel-erlang was usable, lots more work was required to make real applications in Erlang.
- Out of thousands of engineers, I am one of the ***few*** that know Erlang.

Scrapped it.



End of fight, a.k.a submission

- In mid-2016 an email was sent from someone important:

Backend Programming Language Strategy

we have converged on Go and Java as the language choices for building backend services.

At that point I realized I am comfortable with Go.

Learnings from a new team

If you are new, think again before you do this
(<6 months in Go)

DO: use standard error

Java-style:

```
type UserError struct {
        Msg string
}
func (e UserError) Error() string {
        return e.Msg
}
type SystemError struct {
        Msg string
}
func (e SystemError) Error() string {
        return e.Msg
}
```

Go-style:

```
import "error"
```

DO: use json library with structs

Python-style (free form json):

```
namespace = fmt.Sprintf("%s",
                            value.(map[string]interface{})["Namespace"])
reportData := []map[string]string{}
for _, dataValue := range
value.(map[string]interface{})["Data"].([]interface{}) {
    partlyData := make(map[string]string)
    for k, v := range dataValue.(map[string]interface{}) {
        partlyData[k] = v.(string)
    }
    reportData = append(reportData, partlyData)
}
```

Go-style:

```
type Namespace struct {
    UUID             string           `json:"uuid"`
    AcceptedTargets []string          `json:"acceptedTargets"`
    FinishedTargets []string          `json:"finishedTargets"`
    Results         []ExecutionResult `json:"results"`
}
```

It works, but we were afraid to change it.

DO: helpful instructions

```
$ make help
bins           Build all binaries
clean          Removes all build objects
db-bootstrap   bootstrap database schemas
db-connect     connect to local postgresql
db-reset       reset database
db-start       start Backend database locally
db-stop        stop the Backend database
```

- Effortless to write and generate.
- Pleasure to use.

NOT SURE: goto error

Still not sure about this one.

From linux/block/scsi_ioctl.c:

```
int sg_scsi_ioctl(...) {
    /* .. .. .. */
    err = -EFAULT;
    if (copy_from_user(rq->cmd, sic->data, cmdlen))
        goto error;
    /* .. .. .. */
error:
    blk_put_request(rq);

    return err;
}
```

DO: worry about dependencies

Google re-implemented crypto in Go for a reason.

Because they don’t want to depend on anything C.

```
$ cloc ~/code/go/src/crypto
     258 text files.
     258 unique files.
      65 files ignored.

github.com/AlDanial/cloc v 1.70  T=1.42 s (135.7 files/s, 41382.9 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Go                             163           4776           4948          35018
Assembly                        30           1888           2639           9592
-------------------------------------------------------------------------------
SUM:                           193           6664           7587          44610
-------------------------------------------------------------------------------
```

My last advice

Just go with the language (flow). You will be fine.

Questions?

Motiejus Jakštys
motiejus@uber.com
@mo_kelione