Building Thick Clients w/ Tower

github.com/LucioFranco
twitter.com/lucio_d_franco
luciofran.co

What is Tower?

Tower is a library of modular and reusable components for building robust networking clients and servers.

Who uses Tower?

- Noria
- Rust Runtime for AWS Lambda
- Linkerd2 Proxy
- Toshi
- TiKV (soon)
- And more

What is Tower?

Service: Maps a `Request` to some `Future<Response>`

```rust
pub trait Service<Request> {
    type Response;
    type Error;
    type Future: Future<Item = Self::Response, Error = Self::Error>;

    fn call(&mut self, req: Request) -> Self::Future;
}
```

Caution: there is more to this trait but it is simplified for the talk

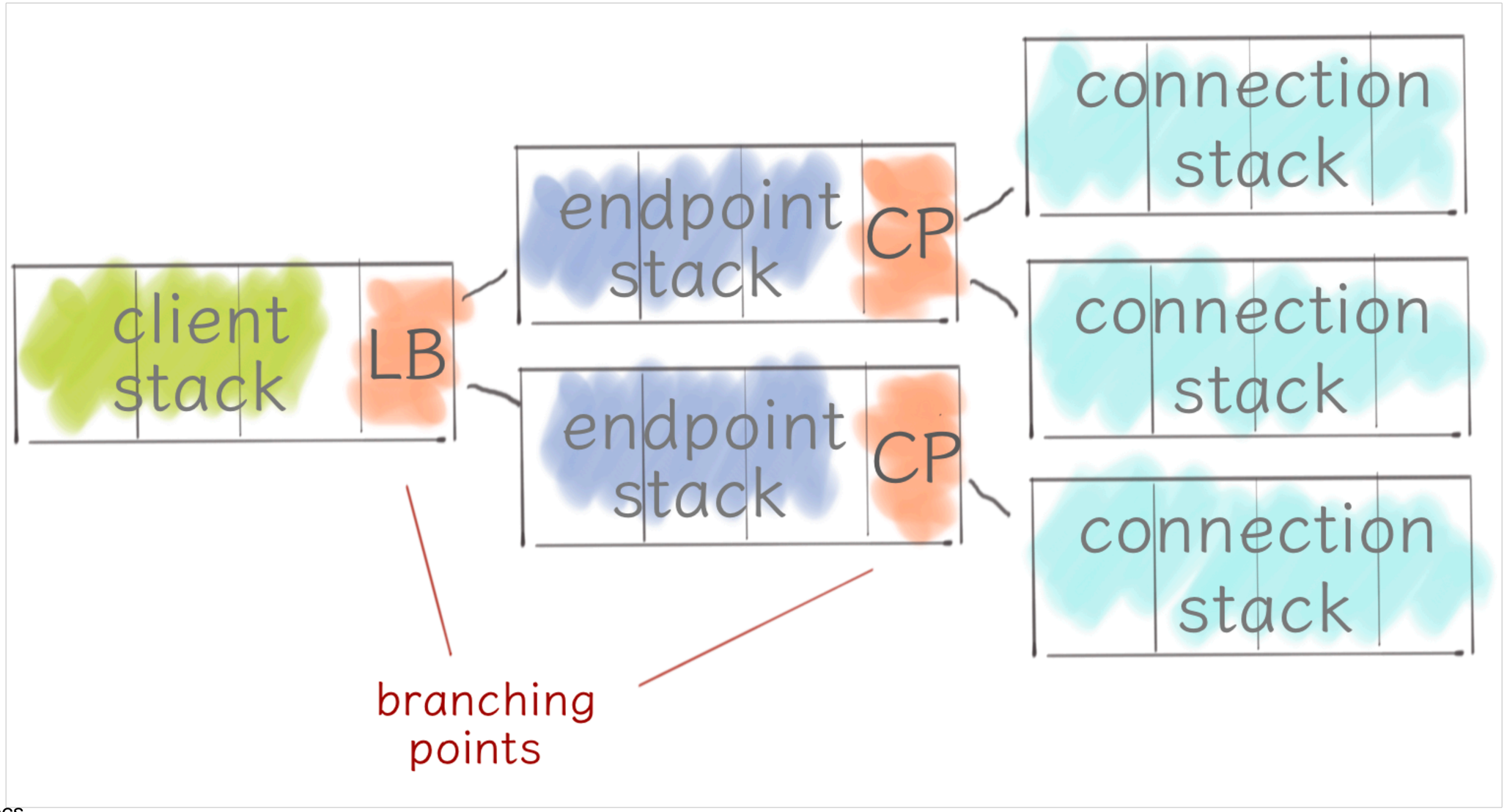
What is Tower?

MakeService: A service factory, `make_service(some_target) -> impl Future`

```
pub trait MakeService<Target, Request> {

    type Service: Service<Request, Response = Self::Response, Error = Self::Error>;
    type Future: Future<Item = Self::Service, Error = Self::MakeError>;


    fn make_service(&mut self, target: Target) -> Self::Future;

}
```

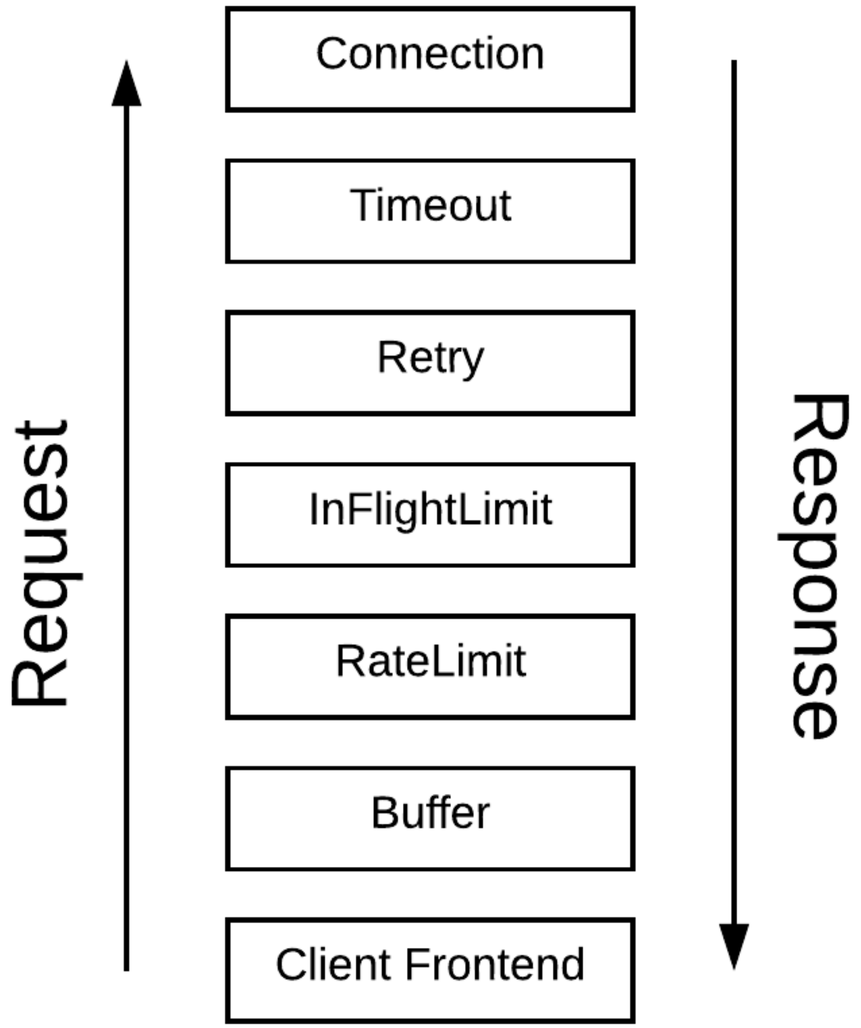
Caution: there is more to this trait but it is simplified for the talk

Clients



Credit: Finagle docs

Clients


Connection
Timeout
Retry
InFlightLimit
RateLimit
Buffer
Client Frontend
Request
Response

Layer / Middleware

Layer: any `Service` that *wraps* another `Service`

```rust
pub trait Layer<S, Request> {
    //....
    type Service: Service<Request, Response = Self::Response, Error = Self::Error>;
    fn layer(&self, inner: S) -> Result<Self::Service, Self::LayerError>;
}
```

Caution: there is more to this trait but it is simplified for the talk

Stack

```
let connector = Connector::new(HttpConnector::new(1));
let hyper = Connect::new(connector, Builder::new());


 ServiceBuilder::new(hyper)
     .add(BufferLayer::new(5))
     .add(RetryLayer::new(policy))
     .add(InFlightLimitLayer::new(5))
     .add(RateLimitLayer::new(5, Duration::from_secs(1)))
     .build()
```

This produces a MakeService…

Caution: the service builder api is not finalized

Stack

Reconnect maps a `MakeService` to a `Service`

```
let mut client = Reconnect::new(maker, dst);


client.call(request).and_then(|response| {…});
client.call(request).and_then(|response| {…});
client.call(request).and_then(|response| {…});
```

Client Frontends

```
pub struct Consul<T>
where
    T: HttpService<Bytes>,
{
    scheme: String,
    authority: String,
    inner: Buffer<IntoService<T>, Request<Bytes>>,
}
```

Client Frontend

```
/// Get a list of all Service members
pub fn get(&mut self, key: &str) -> impl Future<Item = Vec<KVValue>, Error = Error> {
    let url = format!("/v1/kv/{}", key);
    let request = match self.build(&url, Method::GET, Bytes::new()) {
        Ok(req) => req,
        Err(e) => return Either::A(future::err(e)),
    };

    Either::B(self.call(request))
}
```

What’s available today?

- Core: tower-service 0.2 on crates.io
- Layers: retries, in flight limit, rate limit, timeout, filter
- MakeService: reconnect watch, and balance
- Transports: Hyper, tokio, h2
- Frontends: grpc, Consul and Http
- Servers: tower-web, warp, and grpc

Questions?