cmux: Connection Mux

cmux is a generic Go library to multiplex connections based on their payload. Using cmux, you can serve gRPC, SSH, HTTPS, HTTP, Go RPC, and pretty much any other protocol on the same TCP listener.

How-To

Simply create your main listener, create a cmux for that listener, and then match connections:

```
// Create the main listener.
l, err := net.Listen("tcp", ":23456")
if err != nil {
    log.Fatal(err)
}

// Create a cmux.
m := cmux.New(l)

// Match connections in order:
// First grpc, then HTTP, and otherwise Go RPC/TCP.
grpcL := m.Match(cmux.HTTP2HeaderField("content-type", "application/grpc"))
httpL := m.Match(cmux.HTTP1Fast())
trpcL := m.Match(cmux.Any()) // Any means anything that is not yet matched.

// Create your protocol servers.
grpcS := grpc.NewServer()
grpchello.RegisterGreeterServer(grpcS, &server{})

httpS := &http.Server{
    Handler: &helloHTTP1Handler{},
}

trpcS := rpc.NewServer()
trpcS.Register(&ExampleRPCRcvr{})

// Use the muxed listeners for your servers.
go grpcS.Serve(grpcL)
go httpS.Serve(httpL)
go trpcS.Accept(trpcL)

// Start serving!
m.Serve()
```

Take a look at other examples in the GoDoc.

Docs

- GoDocs

Performance

There is room for improvment but, since we are only matching the very first bytes of a connection, the performance overheads on long-lived connections (i.e., RPCs and pipelined HTTP streams) is negligible.

TODO(soheil): Add benchmarks.

Limitations

- *TLS*: `net/http` uses a type assertion to identify TLS connections; since cmux's lookahead-implementing connection wraps the underlying TLS connection, this type assertion fails. Because of that, you can serve HTTPS using cmux but `http.Request.TLS` would not be set in your handlers.
- *Different Protocols on The Same Connection*: `cmux` matches the connection when it's accepted. For example, one connection can be either gRPC or REST, but not both. That is, we assume that a client connection is either used for gRPC or REST.
- *Java gRPC Clients*: Java gRPC client blocks until it receives a SETTINGS frame from the server. If you are using the Java client to connect to a cmux'ed gRPC server please match with writers:

```
grpcl := m.MatchWithWriters(cmux.HTTP2MatchHeaderFieldSendSettings("content-type",
"application/grpc"))
```

Copyright and License