go + grpc + protobuf

Makefile

```
all: convert
	go build grpc_server.go
	go build grpc_client.go
convert:
	go get -u google.golang.org/grpc
	go get -u github.com/golang/protobuf/proto
	go get -u github.com/golang/protobuf/protoc-gen-go
	go get -u google.golang.org/grpc
	go get google.golang.org/protobuf/cmd/protoc-gen-go
	go get google.golang.org/grpc/cmd/protoc-gen-go-grpc
	protoc --go_out=. --go-grpc_out=. ./proto/*.proto
```

proto/say.proto

```
syntax = "proto3";

option go_package = "say";

package say;

service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

message HelloRequest {
  string name = 1;
}

message HelloReply {
  string message = 1;
}
```

grpc_client.go

```
package main

import (
	"context"
	"log"
	"os"
	"time"

	pb "example.com/test_grpc/proto"
	"google.golang.org/grpc"
)

const (
	address     = "localhost:50051"
	defaultName = "world"
)

func main() {
	conn, err := grpc.Dial(address, grpc.WithInsecure(), grpc.WithBlock())
	if err != nil {
		log.Fatalf("did not connect: %v", err)
	}
	defer conn.Close()
	c := pb.NewGreeterClient(conn)

	name := defaultName
	if len(os.Args) > 1 {
		name = os.Args[1]
	}
	ctx, cancel := context.WithTimeout(context.Background(), time.Second)
	defer cancel()
	r, err := c.SayHello(ctx, &pb.HelloRequest{Name: name})
	if err != nil {
		log.Fatalf("could not greet: %v", err)
	}
	log.Printf("Greeting: %s", r.GetMessage())
}
```

grpc_server.go

```
package main

import (
	"context"
	"log"
	"net"

	pb "example.com/test_grpc/proto"
	"google.golang.org/grpc"
)

const (
	port = ":50051"
)
```

```go
type server struct {
    pb.UnimplementedGreeterServer
}

func (s *server) SayHello(ctx context.Context, in *pb.HelloRequest)
(*pb.HelloReply, error) {
    log.Printf("Received: %v", in.GetName())
    return &pb.HelloReply{Message: "Hello " + in.GetName()}, nil
}

func main() {
    lis, err := net.Listen("tcp", port)
    if err != nil {
        log.Fatalf("failed to listen: %v", err)
    }
    s := grpc.NewServer()
    pb.RegisterGreeterServer(s, &server{})
    if err := s.Serve(lis); err != nil {
        log.Fatalf("failed to serve: %v", err)
    }
}
```