Binary protocols

Oxford University
Software Engineering
Programme
December 2018

Data Serialisation

- Data in memory needs to be stored on disk or transferred over the wire
- Lots of different approaches:
  - XML
  - JSON
  - MsgPack
  - Thrift
  - Etc

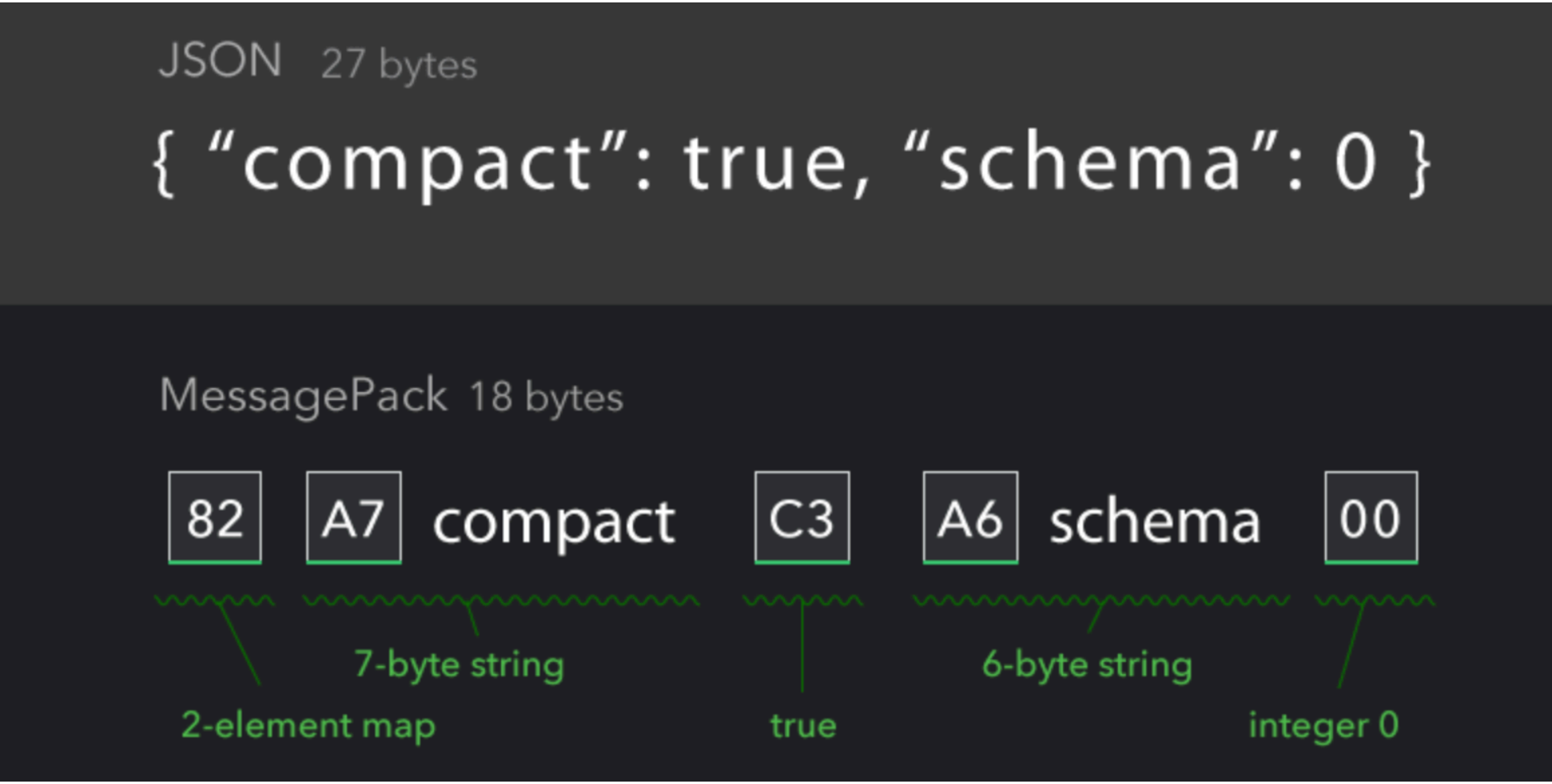
XML

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <isbn>9780262510875</isbn>
  <title>Structure and Interpretation of Computer Programs - 2nd Edition
    </title>
</book>
```

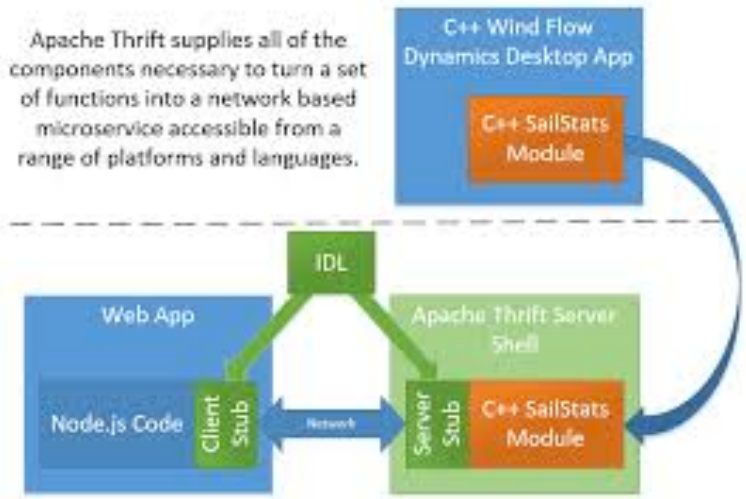
JSON

```
{
   "isbn": 9780262510875,
   "title": "Structure and
Interpretation of Computer Programs
- 2nd Edition”
}
```

MsgPack



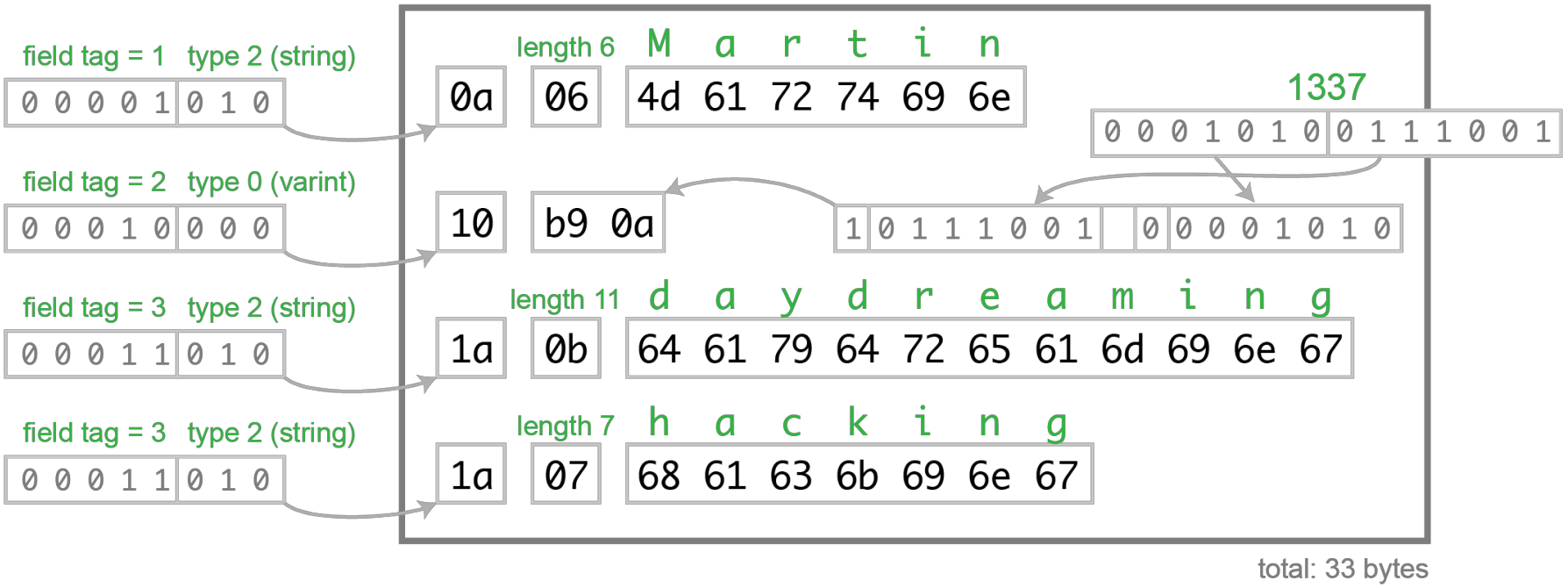
Apache Thrift



Source: The Programmer's Guide to Apache Thrift

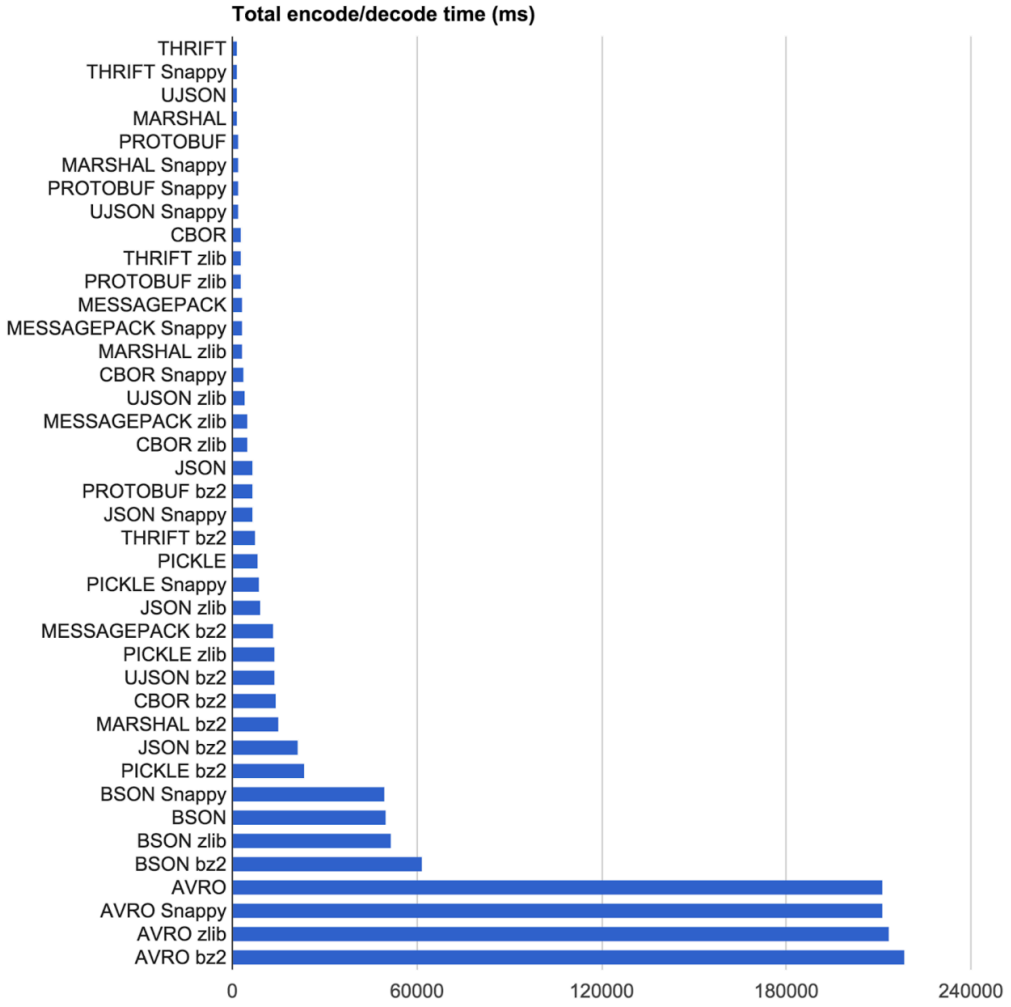
ProtoBuf

Protocol Buffers



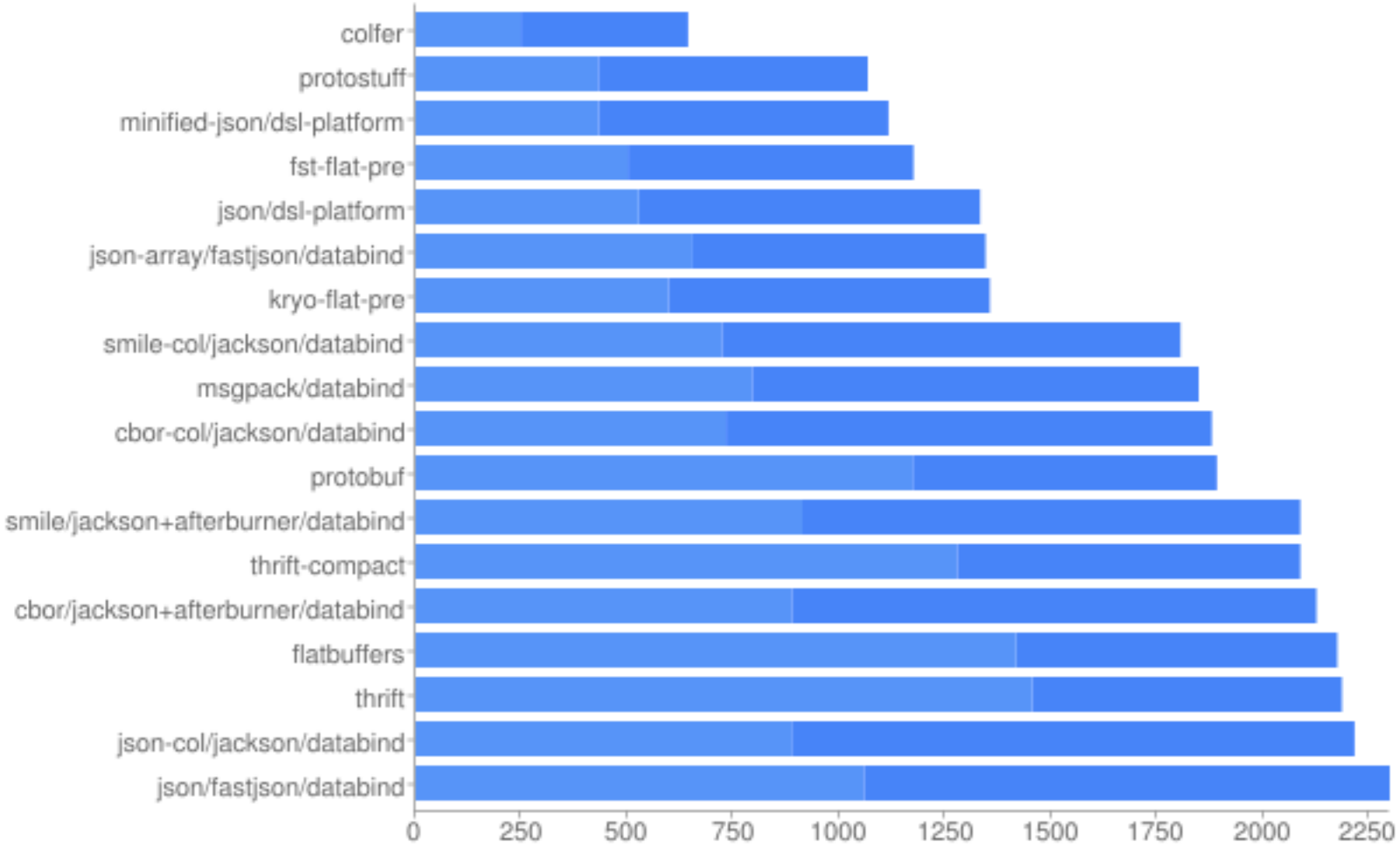
Performance

https://eng.uber.com/trip-data-squeeze/

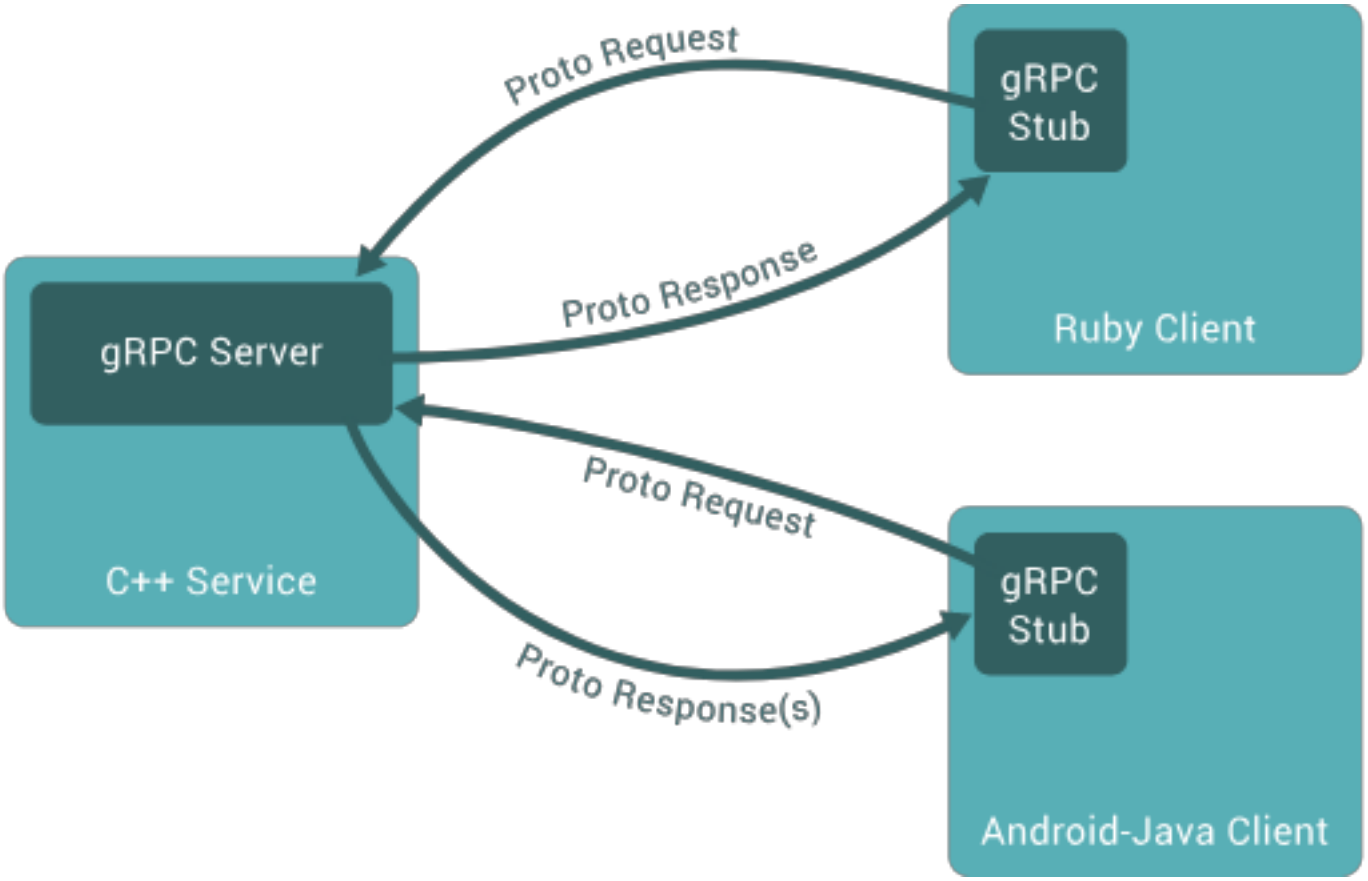


More performance

https://github.com/eishay/jvm-serializers/wiki



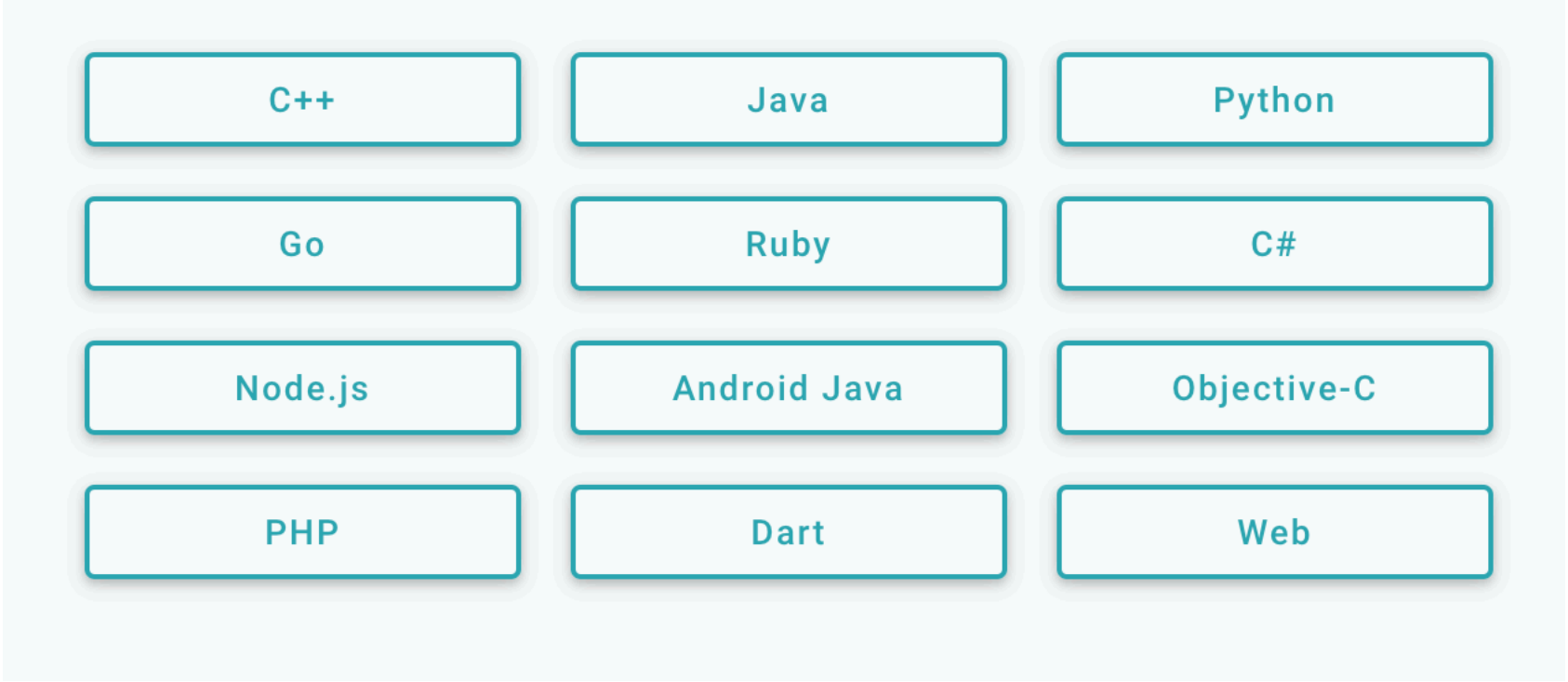
gRPC



gRPC

- ProtoBuf over HTTP/2
- Supports:
  - One-way
  - Request-Response
  - Request-Stream Response
  - Bidirectional Stream

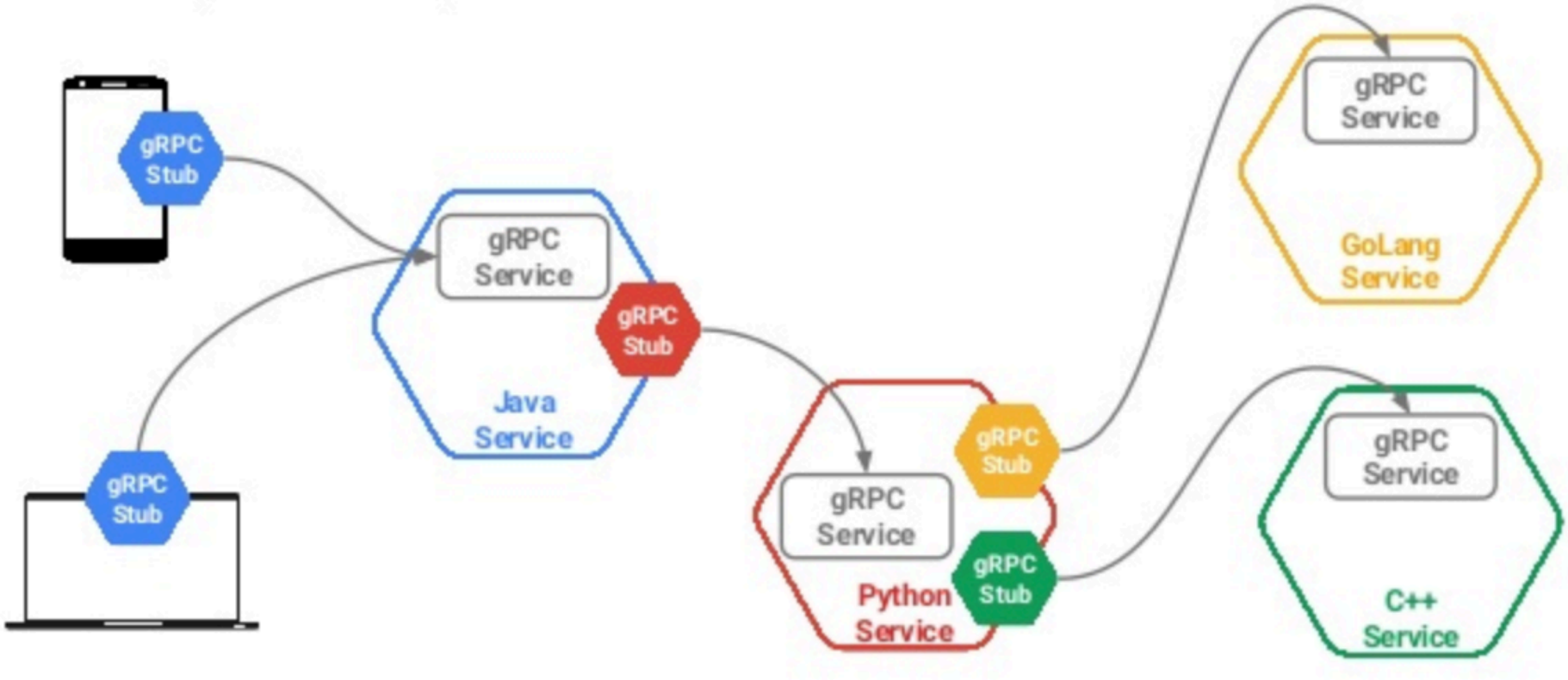
gRPC Language Support



and Ballerina

```
syntax = "proto3";

package freo.me.purchase;

// The greeting service definition.
service Purchase {
  // Sends a greeting
  rpc purchase (PurchaseRequest) returns (PurchaseReply) {}
}

// The request message containing the user's name.
message PurchaseRequest {
  string poNumber = 1;
  string lineItem = 2;
  int32 quantity = 3;
  Date date = 4;
  string customerNumber= 5;
  string paymentReference = 6;

}

message Date {
    int32 year = 1;
    int32 month = 2;
    int32 day = 3;
}
```

Interoperability



https://in.pycon.org/cfp/2017/proposals/boosting-python-web-applications-with-protocol-buffers-and-grpc~egQZb/

Ballerina and gRPC

```
type birthday record {
    int day;
    int month;
    int year;
};

endpoint grpc:Listener listener {
    host: "localhost",
    port:  config:getAsInt("GRPC_PORT")
};

@grpc:ServiceConfig
service<grpc:Service> grpcService bind listener {
    calculateAge(endpoint caller, birthday req) {
        time:Time bday = time:createTime(req.year, req.month, req.day,
        time:Time now = time:currentTime();
        int ageyears = (now.time - bday.time)/(24*365*60*60*1000);
        _ = caller -> send(ageyears, headers = ());
        _ = caller -> complete();
    }
}
```

Automatically creates this .proto

```
syntax = "proto3";
import "google/protobuf/wrappers.proto";
service grpcService {
    rpc calculateAge(birthday)
        returns (google.protobuf.Int64Value);
}
message birthday {
    int64 day = 1;
    int64 month = 2;
    int64 year = 3;
}
```

Questions