CYBERINFRASTRUCTURE
INTEGRATION RESEARCH CENTER
PERVASIVE TECHNOLOGY INSTITUTE

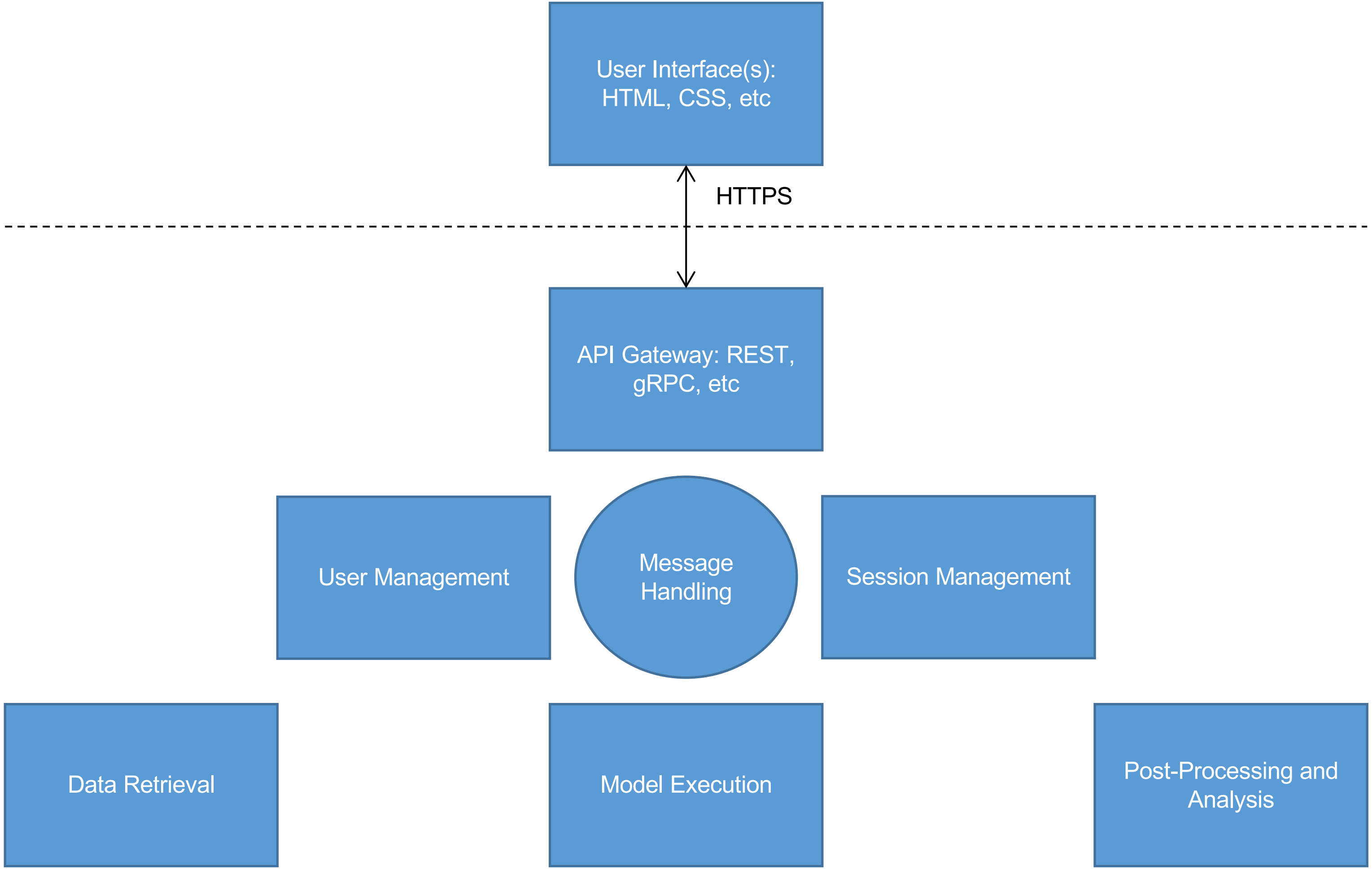
Data Serialization and Binary RPC

January 27th 2022

Suresh Marru

Assignment 1 anti-patterns

- “If it did not happen on your GitHub repo it did not happen”.
- Software engineering project vs Architectural thinking.
- Confusion with “Session Management” Service.
- “interactive” user interface.
- Focus on usability of application vs ease of deploy of your system (build from source on a laptop).



User Interface(s): HTML, CSS, etc

HTTPS

API Gateway: REST, gRPC, etc

User Management

Message Handling

Session Management

Data Retrieval

Model Execution

Post-Processing and Analysis

Assignment Requirements

- Each of the components in the previous diagram is a microservice.
  - Each must run as a separate process.
- You must use at least 3 different programming languages
  - For example, one service in Python, one in Go, one in Java, ...
- You must use at least one DB technology
  - Only one microservice can connect to each DB
- You must choose and implement an internal communication strategy for your microservices
- You must define your API based on this lecture and other discussions
- Prototype your continuous integration and deployment
  - Your entire system must be easily deployable by your peers, graders, and instructors

Task or Job State Diagram

- Do not assume your service has a Boolean operation.
- Your core functional services are accomplishing a task.
- A task has a state transition

  <Operating Systems 101>
- Are you thinking on state diagram for your individual services and for your system as a whole?

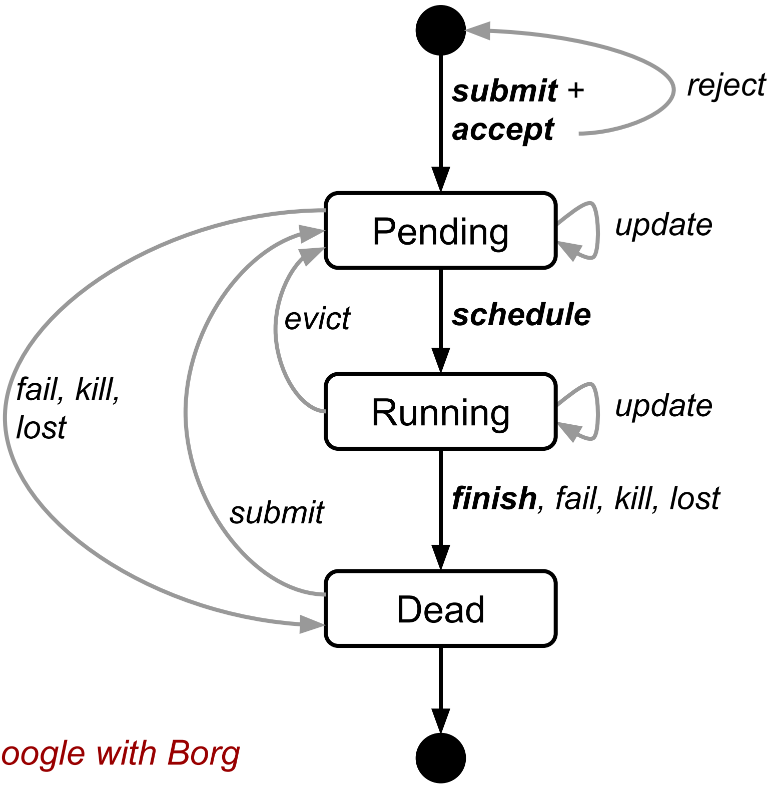
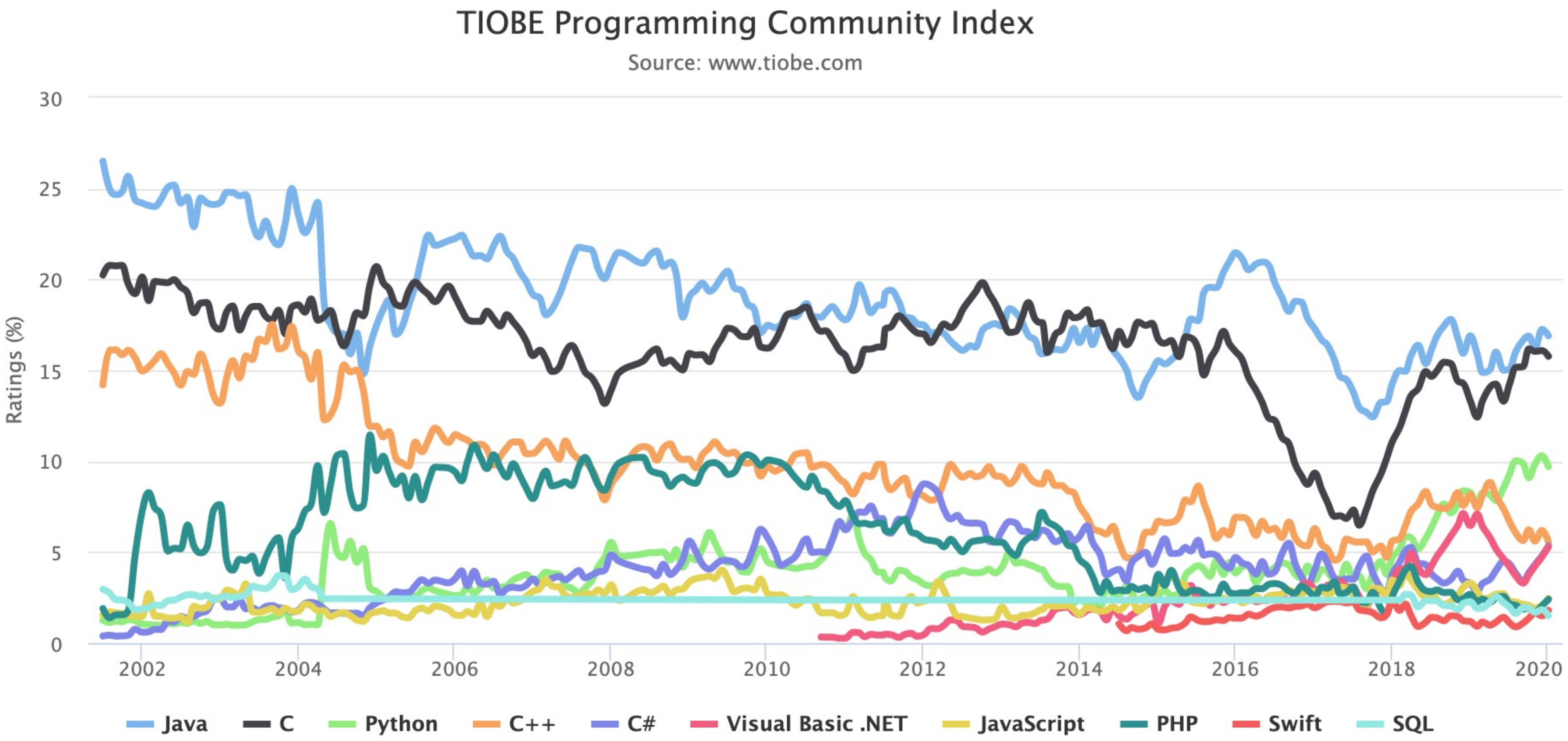


Image Source: Verma et al, Large-scale cluster management at Google with Borg

Thrift, ProtoBuff and gRPC

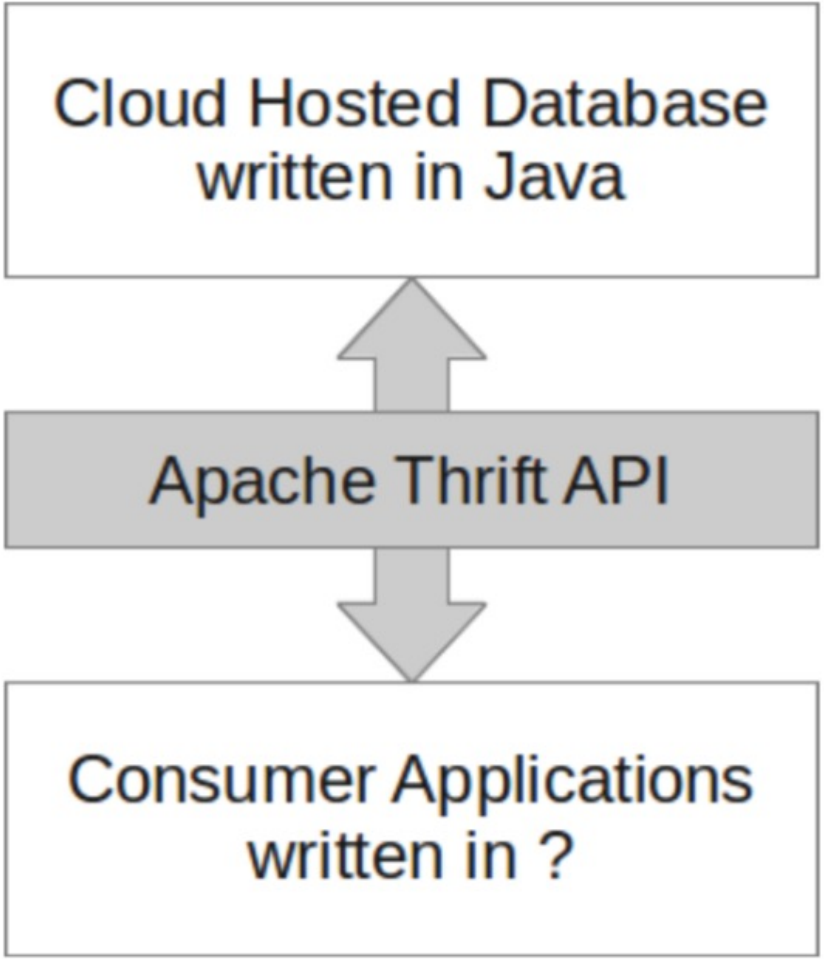
Reference Papers

- Verma, Abhishek, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. "Large-scale cluster management at Google with Borg." In *Proceedings of the Tenth European Conference on Computer Systems*, pp. 1-17. 2015.
  - https://dl.acm.org/doi/pdf/10.1145/2741948.2741964
- Burns, Brendan, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. "Borg, omega, and kubernetes." *Queue* 14, no. 1 (2016): 70-93.
  - https://dl.acm.org/doi/pdf/10.1145/2898442.2898444



TIOBE Programming Community Index

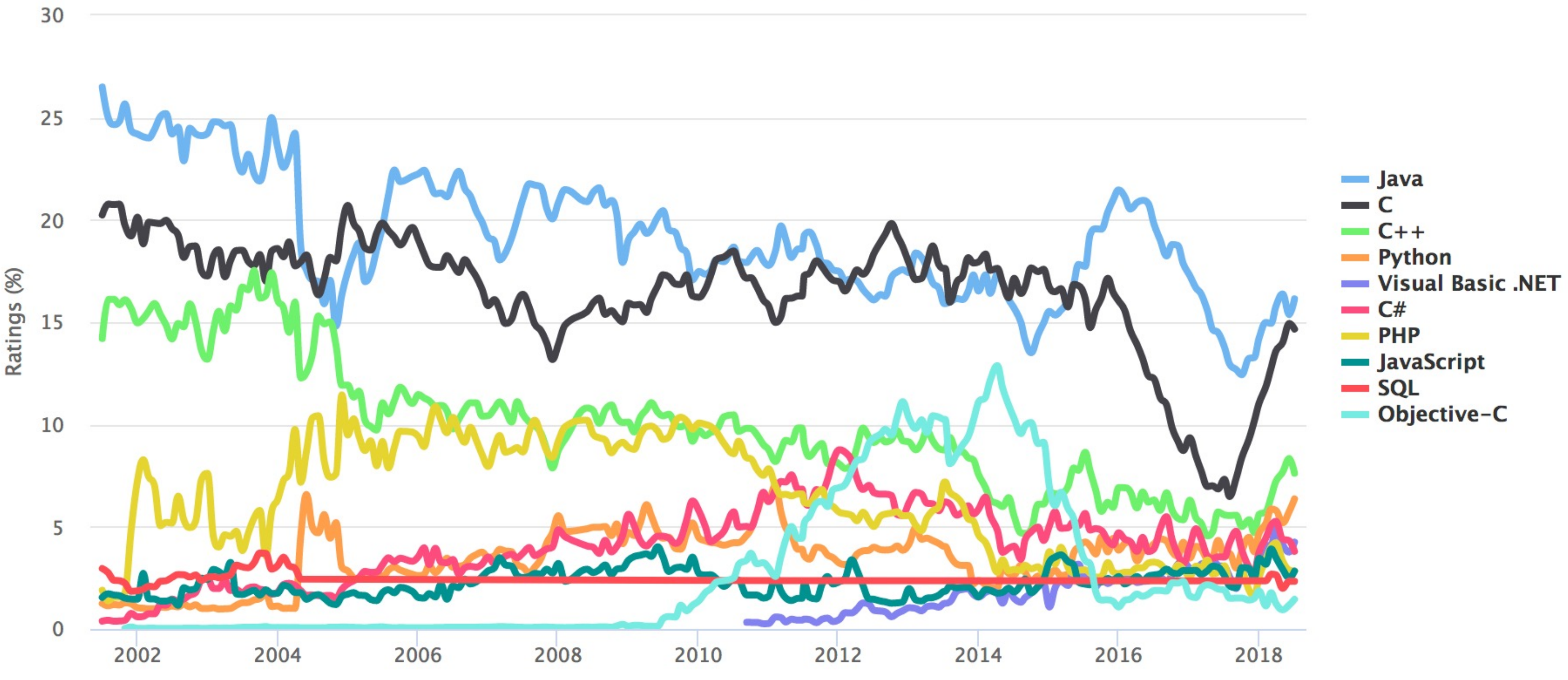
Source: www.tiobe.com



Cloud Hosted Database written in Java

Apache Thrift API

Consumer Applications written in ?



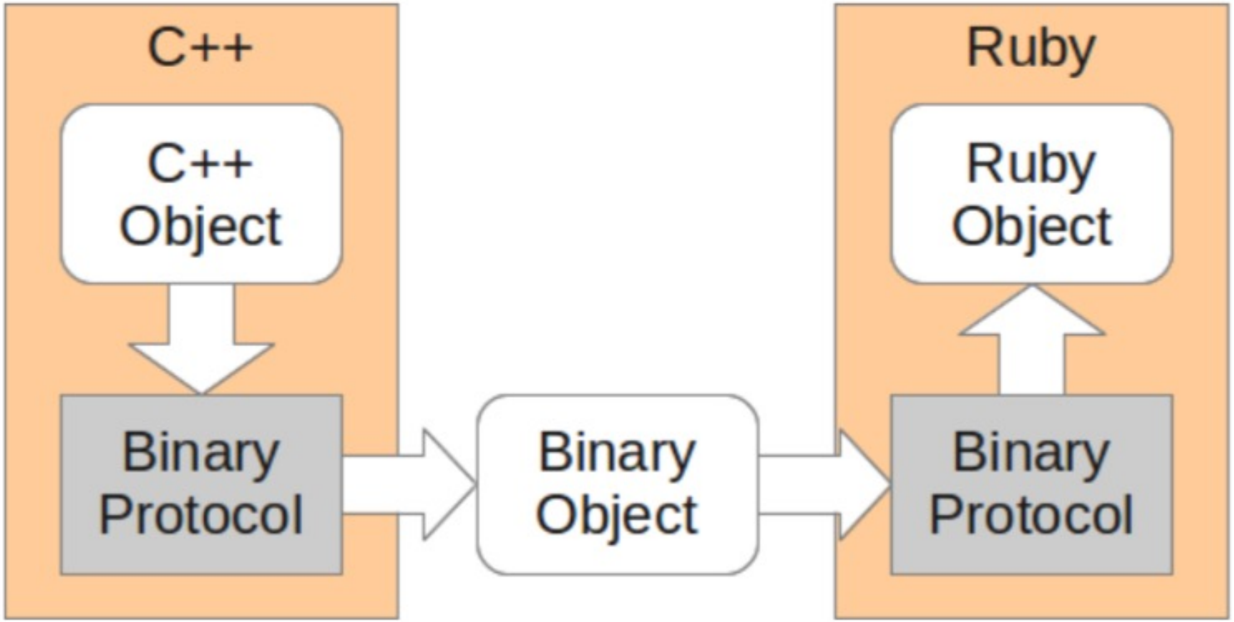
TIOBE Programming Community Index

Source: www.tiobe.com

Programming Language “polyglotism”

- Modern distributed applications are rarely composed of modules written in a single language.
- Weaving together innovations made in a range of languages is a core competency of successful enterprises.
- Cross language communications are a necessity, not a luxury.
- In your projects you need to demonstrate this by using three or more languages.

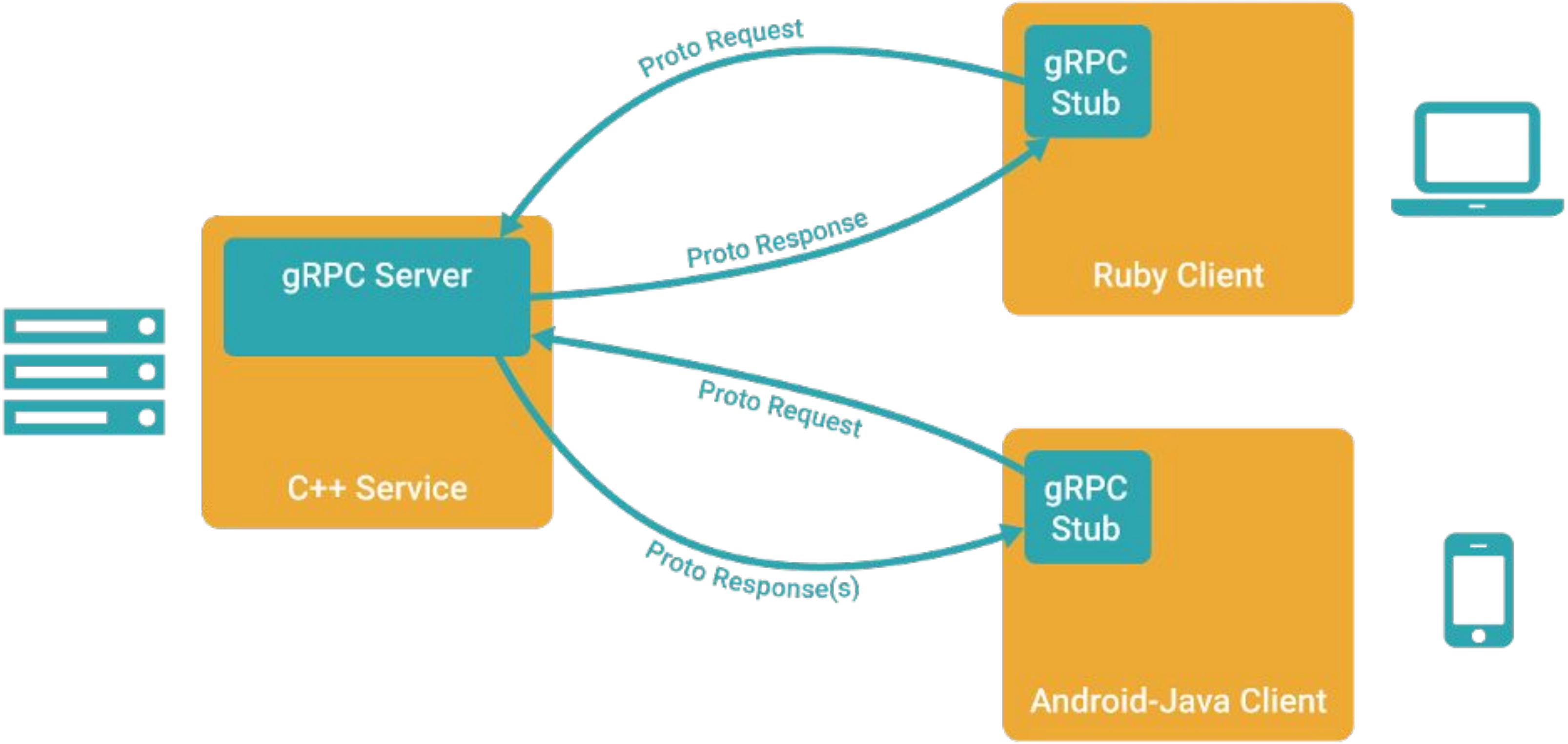
Cross-Language Communications



Source: Randy Abernethy. *The Programmer’s Guide to Apache Thrift, Manning Publications Co.*

Other Motivations

- Large-scale distributed systems actually composed of microservices
  - Allows loosely-coupled and even multilingual development
  - Scalability: things, cores, devices, nodes, clusters, and data centers (DCs)
- Communication predominantly structured as RPCs
  - Many models of RPC communication
  - Terminology: Client uses a stub to call a method running on a service/server
  - Easiest interfaces (synchronous, unary) resemble local procedure calls
- Translated to network activity by code generator and RPC library
  - High-performance interfaces (async, streaming) look like Active Messaging
- Long way from textbook description of RPCs!

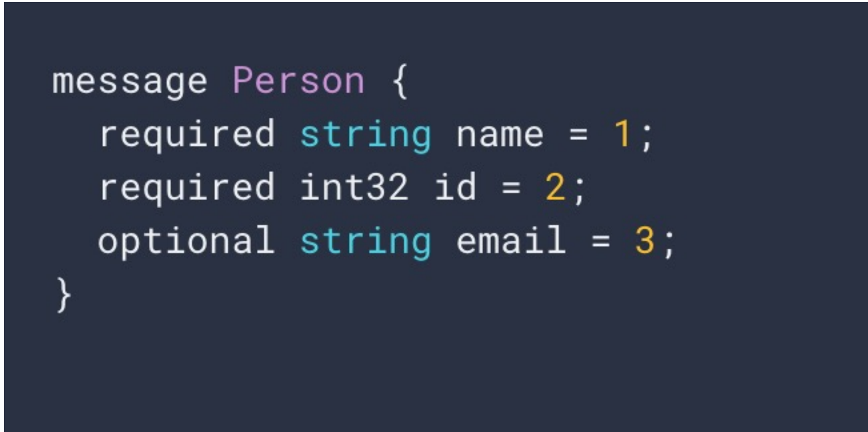

Proto Request
gRPC Stub
Proto Response
gRPC Server
Ruby Client
Proto Request
C++ Service
gRPC Stub
Proto Response(s)
Android-Java Client

Protocol Buffers

- “a language-neutral, platform-neutral, extensible way of serializing structured data for use in communications protocols, data storage, and more.”
- “Protocol buffers are a flexible, efficient, automated mechanism for serializing structured data – think XML, but smaller, faster, and simpler. ”
  - https://developers.google.com/protocol-buffers/docs/overview
- Started internally within Google in 2001 and Opened in 2008.

Protocol Buffers Contd.

- IDL (Interface definition language)
  - Describe once and generate interfaces for any language.
- Data Model
  - Structure of the request and response.
- Wire Format
  - Binary format for network transmission.



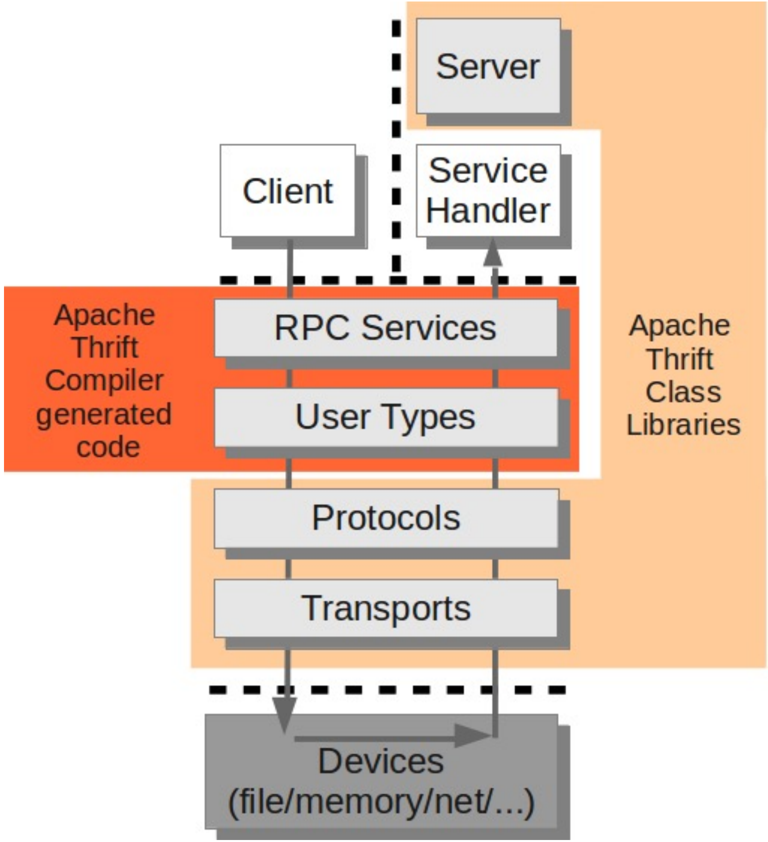
```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
}
```

Apache Thrift

- Thrift is Facebook's implementation of Proto Buff open sourced under Apache.
- A high performance, scalable cross language serialization and RPC framework.
- Provides a full RPC Implementation with generated clients, servers, everything but the business logic.
- Thrift is is fast and efficient, solutions for minimal parsing overhead and minimal size.

Thrift for RPC Services

- User Code
  - client code calls RPC methods and/or [de]serializes objects
  - service handlers implement RPC service behavior
- Generated Code
  - RPC stubs supply client side proxies and server side processors
  - type serialization code provides serialization for IDL defined types
- Library Code
  - servers host user defined services, managing connections and concurrency
  - protocols perform serialization
  - transports move bytes from here to there

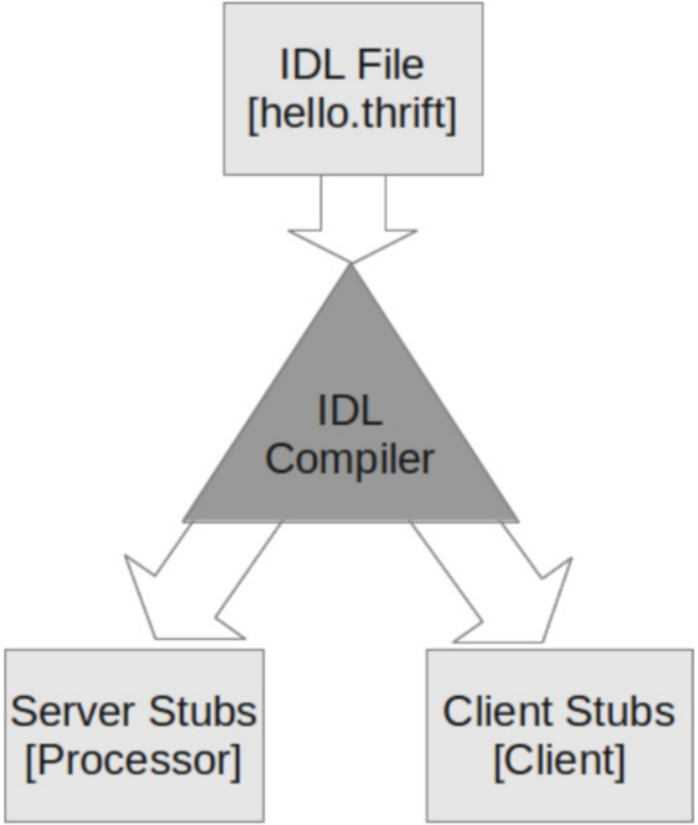


Server
Client
Service Handler
Apache Thrift Compiler generated code
RPC Services
User Types
Apache Thrift Class Libraries
Protocols
Transports
Devices (file/memory/net/...)

Source: Randy Abernethy. *The Programmer's Guide to Apache Thrift, Manning Publications Co.*

4 Simple Steps to Create a RPC microservice

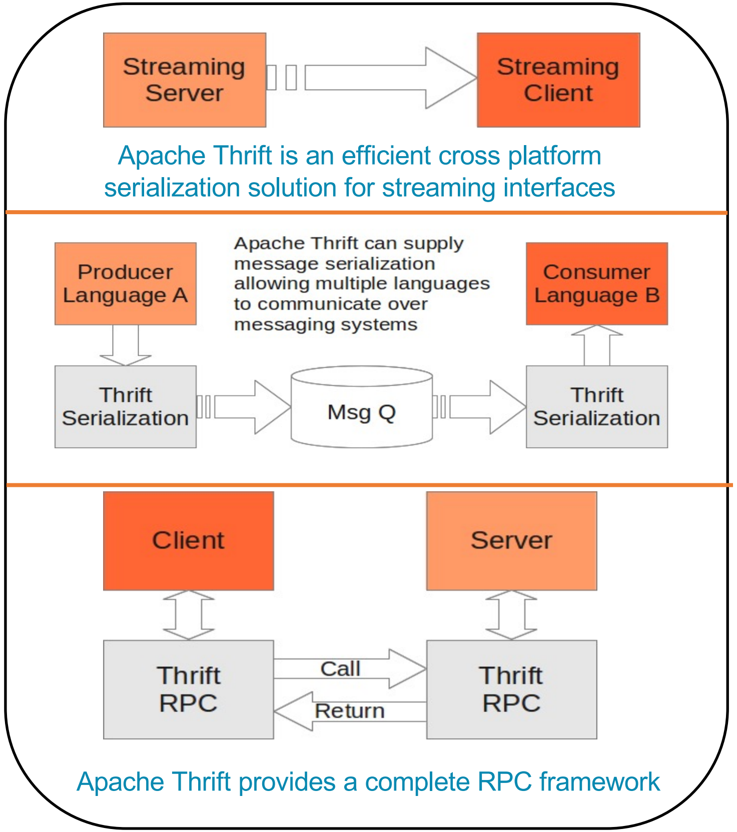
1. Define the service in a language neutral "Interface Description Language".
2. Compile the IDL to generate Server and Client "stubs" in desired programming languages.
3. Plug the server implementation in the pre-generated server stub.
4. Call the remote services as if they are making local method calls.



Source: Randy Abernethy. *The Programmer's Guide to Apache Thrift, Manning Publications Co.*

Thrift: Multiple Communication Schemes

- Streaming – Communications characterized by an ongoing flow of bytes from a server to one or more clients.
  - Example: An internet radio broadcast where the client receives bytes over time transmitted by the server in an ongoing sequence of small packets.
- Messaging – Message passing involves one way asynchronous, often queued, communications, producing loosely coupled systems.
  - Example: Sending an email message where you may get a response or you may not, and if you do get a response you don’t know exactly when you will get it.
- RPC – Remote Procedure Call systems allow function calls to be made between processes on different computers.
  - Example: An iPhone app calling a service on the Internet which returns the weather forecast.

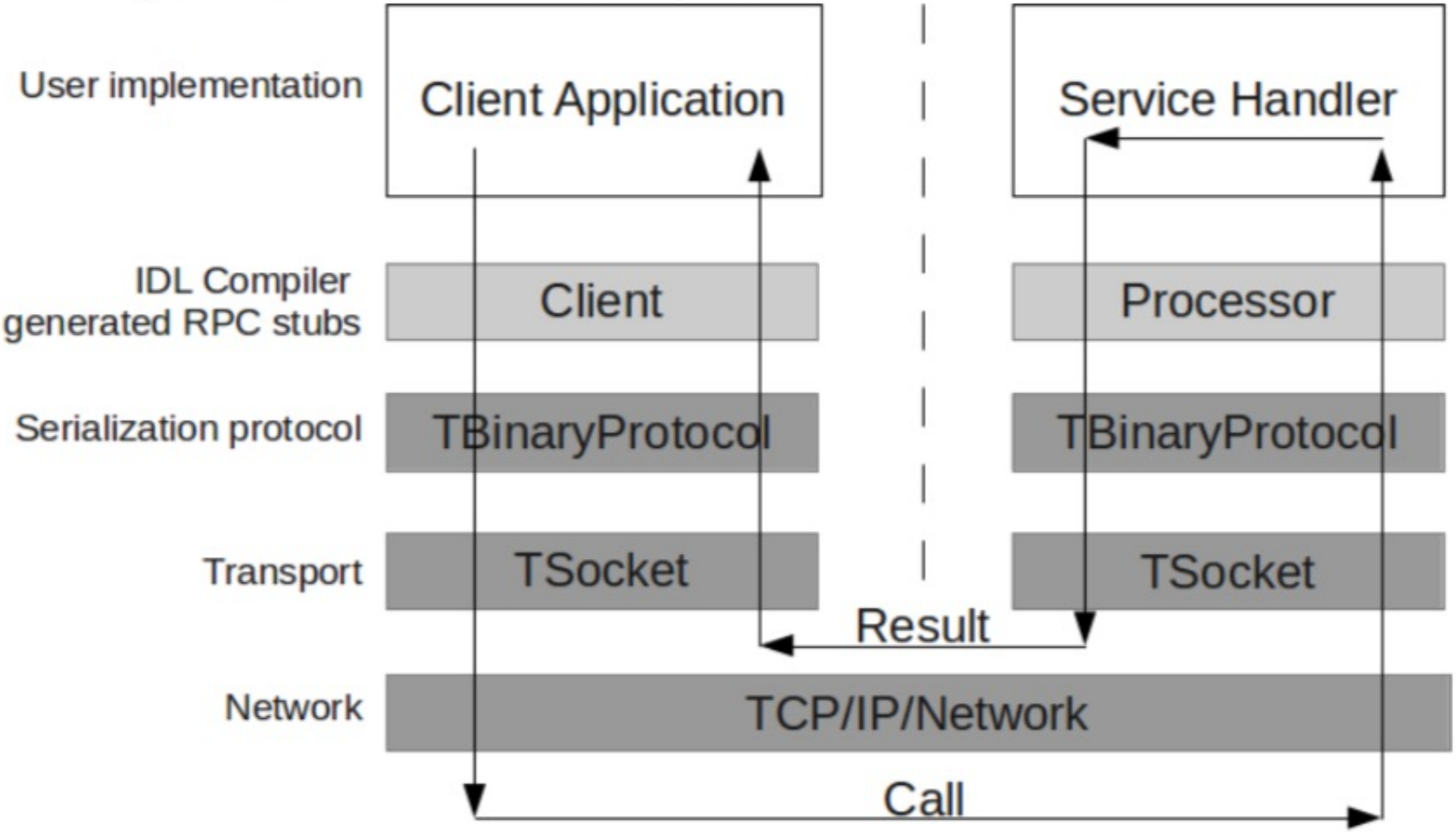


Apache Thrift is an efficient cross platform serialization solution for streaming interfaces

Apache Thrift can supply message serialization allowing multiple languages to communicate over messaging systems

Apache Thrift provides a complete RPC framework

Source: Randy Abernethy. *The Programmer’s Guide to Apache Thrift, Manning Publications Co.*



User implementation
Client Application
Service Handler

IDL Compiler generated RPC stubs
Client
Processor

Serialization protocol
TBinaryProtocol
TBinaryProtocol

Transport
TSocket
TSocket

Result

Network
TCP/IP/Network

Call

Source: Randy Abernethy. *The Programmer's Guide to Apache Thrift, Manning Publications Co.*

GRPC

- Google open sourced in Feb 2015
- **Transport**: `HTTP/2`
- **Wire format**: `Protocol Buffers v3` (Binary)
- **Service definition**: `Protocol Buffers IDL`
- Libraries in ~10 languages (native C, Go, Java)
- Microservices framework

What is gRPC for? (from official FAQ)

- Low latency, highly scalable, distributed systems
- Developing mobile clients which are communicating to a cloud server
- Designing a new protocol that needs to be accurate, efficient and language independent
- Layered design to enable extension e.g. authentication, load balancing, logging and monitoring etc