KubeCon | CloudNativeCon

North America 2018

Using gRPC for Long-lived and Streaming RPCs

Eric Anderson, Staff Software Engineer, *Google*

Intended Audience

Have used gRPC

- Made a service
- Wrote some clients

Interested in more advanced use-cases

Overview

Long-lived RPCs

Streaming RPCs

- Half Duplex
- Full Duplex

Long-lived Streaming RPCs

Overview

Long-lived RPCs

Streaming RPCs

- Half Duplex
- Full Duplex

Long-lived Streaming RPCs

Long-lived RPCs

RPCs that last minutes, hours, days

Long-lived RPC Use-cases

Reduce polling, reduced latency; "Hanging GET"

- Watches/notifications

Long-lived RPC Issues

Load balancing

- Uneven backend memory usage
- MAX_CONNECTION_AGE can accumulate connections

TCP disconnects will fail calls; may take time to be detected

Deadline not very useful*

Not great for operations that take a long time, as you need to "reconnect"

Long-lived RPC Improvements

Load balancing: Have server occasionally close RPC

- If using MAX_CONNECTION_AGE, can use MAX_CONNECTION_AGE_GRACE to auto-kill as a back-up

Time to detect disconections: Client-side Keepalive

Since not generally latency-sensitive, enable Wait-for-ready

Overview

Long-lived RPCs

Streaming RPCs

- Half Duplex
- Full Duplex

Long-lived Streaming RPCs

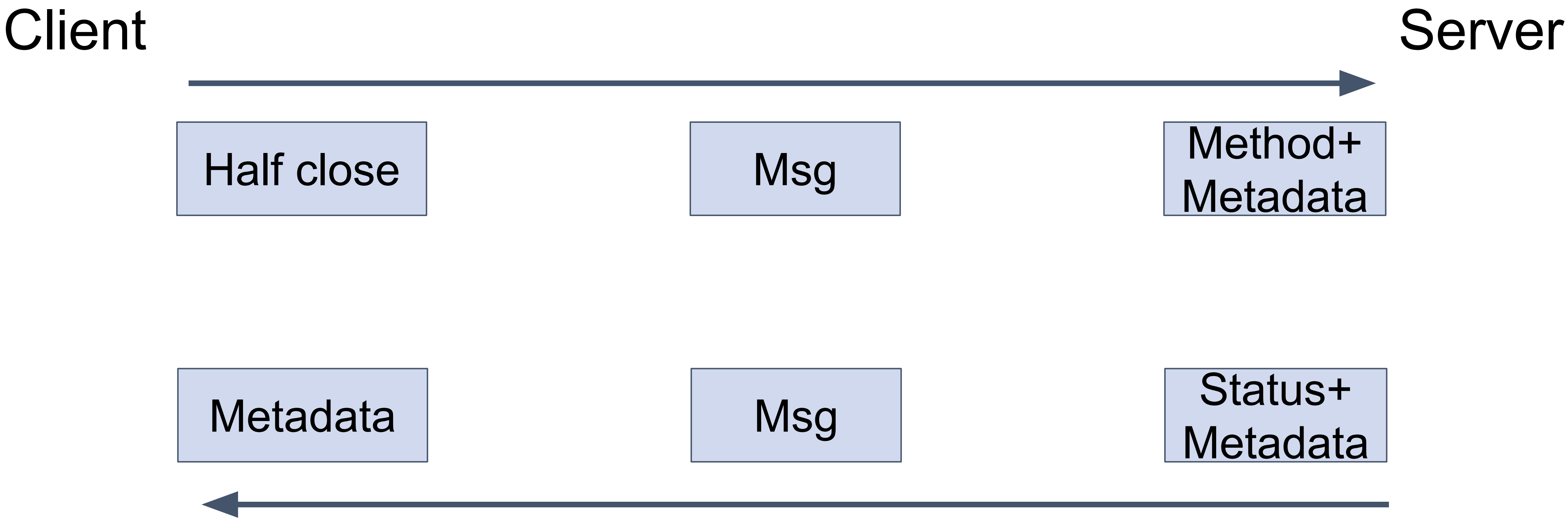
Streaming RPCs

Zero-to-many messages (instead of one)

Messages are ordered

Streaming is independent in each direction

Streaming RPCs (Unary)



Client

Server

Half close

Msg

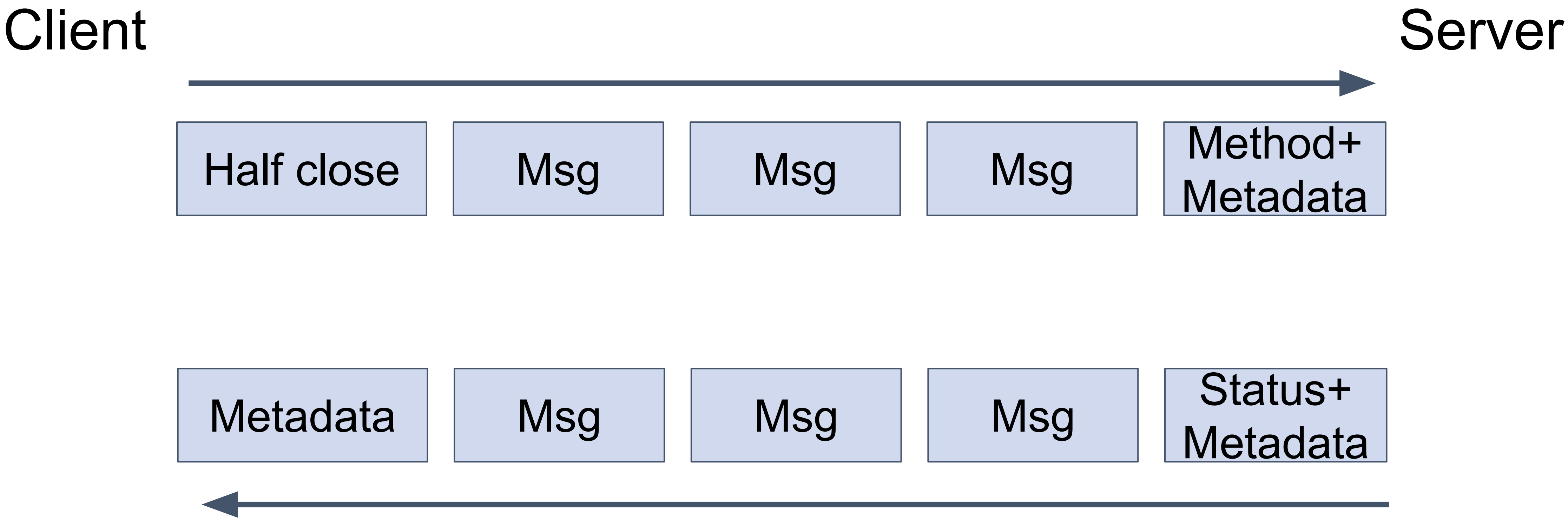
Method+ Metadata

Metadata

Msg

Status+ Metadata

Streaming RPCs (Streaming)



Client

Server

Half close

Msg

Msg

Msg

Method+
Metadata

Metadata

Msg

Msg

Msg

Status+
Metadata

Streaming RPCs

```
rpc          UnaryCall       (Request) returns        (Response);

rpc ClientStreamingCall(stream Request) returns        (Response);

rpc ServerStreamingCall       (Request) returns (stream Response);

rpc   BidirectionalCall(stream Request) returns (stream Response);
```

Streaming RPCs

Bidirectional (Bidi) Streaming

- Half duplex. Client-streaming + Server-streaming
- Full duplex. More that one side can send at a time
  - Like TCP, but with messages instead of bytes
    (close semantics are a bit different, though)
  - No implicit acks; writes are only acked by responses

Overview

Long-lived RPCs

Streaming RPCs

- **Half Duplex**
- Full Duplex

Long-lived Streaming RPCs

Half-duplex Streaming Use-cases

Latency or memory reduction (e.g., speech to text)

- multiple small messages instead of a big message

Separation of response and “end of call” (e.g., watches)

Flow Control (“push-back”)

- Bulk uploads without needing to optimize chunk sizes
- Less “jerky” than one-at-a-time chunking (gives “pipelining”)

Half-duplex Streaming Use-cases

Messages with state association

- Pinning to a backend
- Expands call lifetime (e.g., transactions)
- Reduced per-message setup cost (e.g., watches)
- Full-state followed by deltas (watches again...)

Half-duplex Streaming Issues

gRPC flow control may have large buffers (64 KB-4 MB)

gRPC flow control is point-to-point

Increased API complexity*

Server-streaming may require application-level retries* (vs framework-level)

Tracing/stats muddled or missing

Half-duplex Streaming Improvements

Flow control problems:
    use full duplex + application-level flow control

Tracing/stats: treating like unary could work okay

Overview

Long-lived RPCs

Streaming RPCs

- Half Duplex
- **Full Duplex**

Long-lived Streaming RPCs

Full-duplex Streaming Use-cases

TCP with messages

- Custom protocols

Application-level flow control (e.g., “messages,” “work items”)

Transactions

“Live” Reconfiguration

Bulk uploads, with reduced frequency of resumption

Use half-close to “hang up” instead of cancel

Full-duplex Streaming Issues

Tracing/stats systems may be overly simplistic*

API/protocol complexity*

Involved application-level retry*

Flow-control-induced deadlocking

Lack of REST conversion*

Full-duplex Streaming Improvements

Have at least one side be reading at any time

- If mixing control and data messages, use application-level flow control to limit memory usage

Overview

Long-lived RPCs

Streaming RPCs

- Half Duplex
- Full Duplex

Long-lived Streaming RPCs

Long-lived Streaming Issues

Load balancing (memory+cpu)

Tracing/stats systems may be overly simplistic*

Long-lived Streaming Improvements

Load balancing: same as long-lived RPCs

Q&A

GitHub/Gitter: @ejona86

Email: ejona@google.com (please CC mailing list)

Mailing List: grpc-io@googlegroups.com

Stack Overflow: #grpc #grpc-java