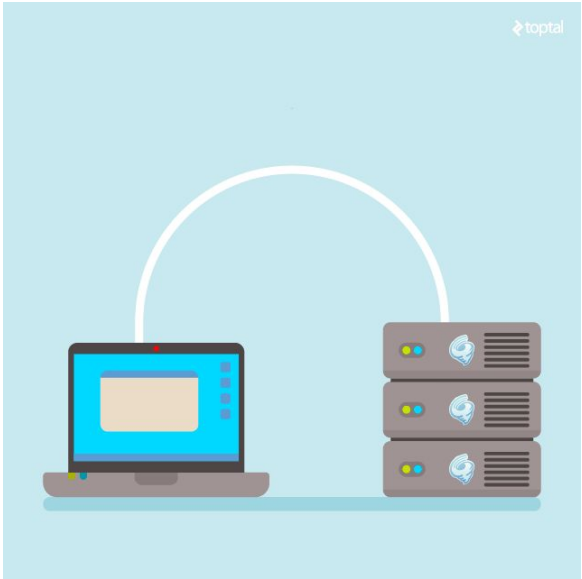
Stateful Client-Server Communication in the Era of Serverless Functions

Wenbo Zhu (Google)

O'Reilly Software Architecture Conference 2018

An old question, and yet again

How bad is stateful communication, now with “serverless functions”?



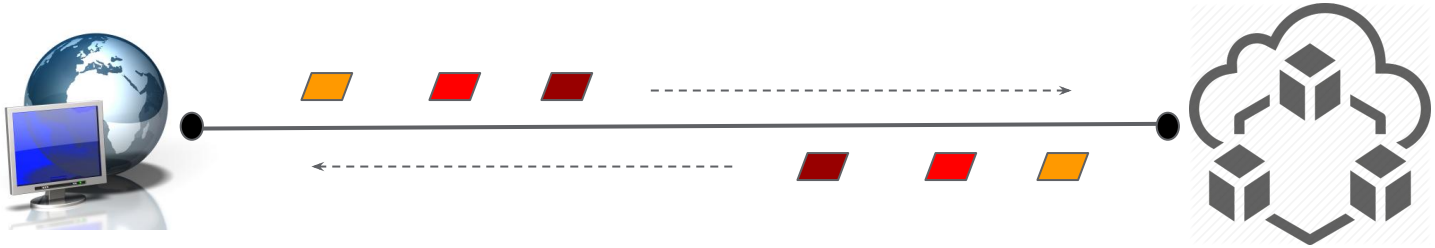
Stateful client-server communication

In a nutshell, “TCP socket” + message framing

- bidi gRPC, websockets ...

Compared to RPC (request-response)

- Full-duplex, FIFO and in-order delivery
- Non-causal semantics: server-push, one-way messages

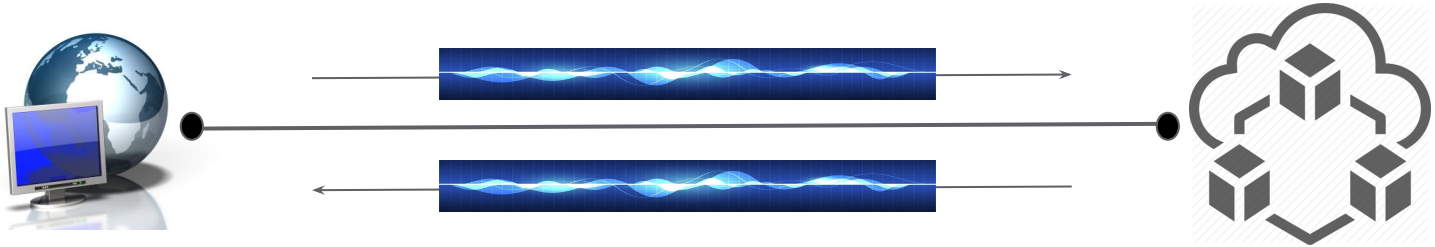


Where it works well

As an optimization to strict RPCs

- Causal relation between client and server-sent data
- Incremental & interleaved data delivery and processing
- Short-lived

“media transcoding”, i.e. full-duplex & streamed upload/download

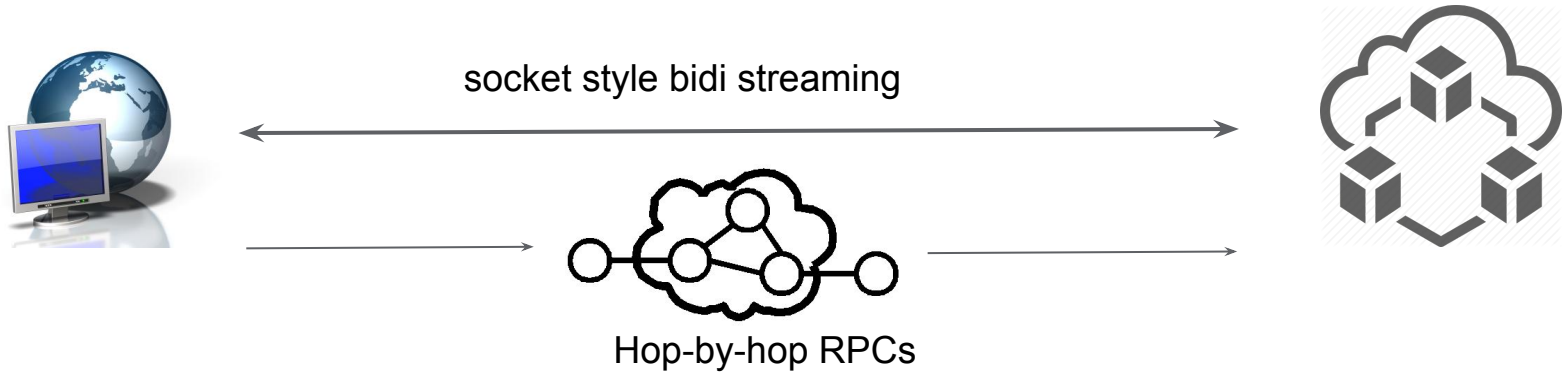


Where it becomes problematic

Spontaneous server-push or long-lived sessions

- Database watch (firestore)
- Chat, collaborative editing

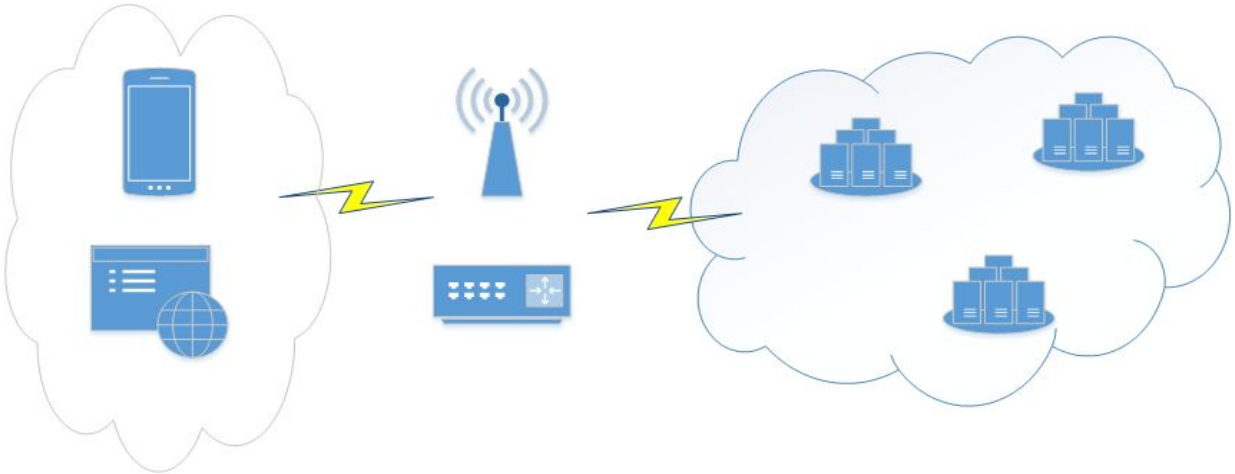
Stateless RPCs => stateful bidi streaming



Challenges

When the transport (TCP) is unreliable, i.e. Internet, cloud

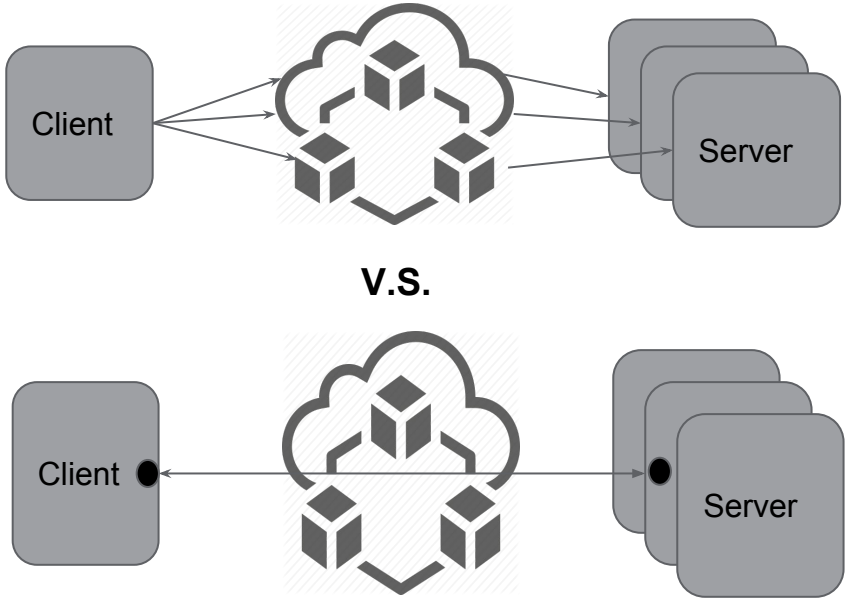
- No end-to-end ack
- Failure detection is hard or impossible



Challenges

For long-lived sessions

- Client-side: proxies, mobility
- Server-side: load-balancing, DoS prevention, security

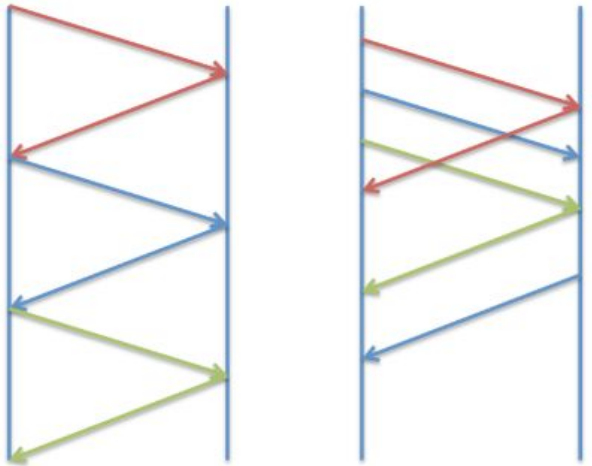


Performance benefits

- Latency: saving 1-RTT, compared to sequential polling
- Efficiency: cross-layer “batching”, esp. to skip redundant auth

Relative to the speed of light

- L1: ~2x
- L2-3: ~3x
- L4-7: ~10x



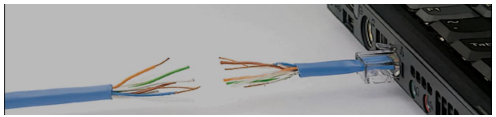
Higher-level abstraction

A single session that assumes FIFO, in-order and causal delivery

- v.s. independent & concurrent RPCs (which may fail too)

1. session = open(url)
2. session.send(msg)
3. session.on(“message”, callback)
4. session .close()

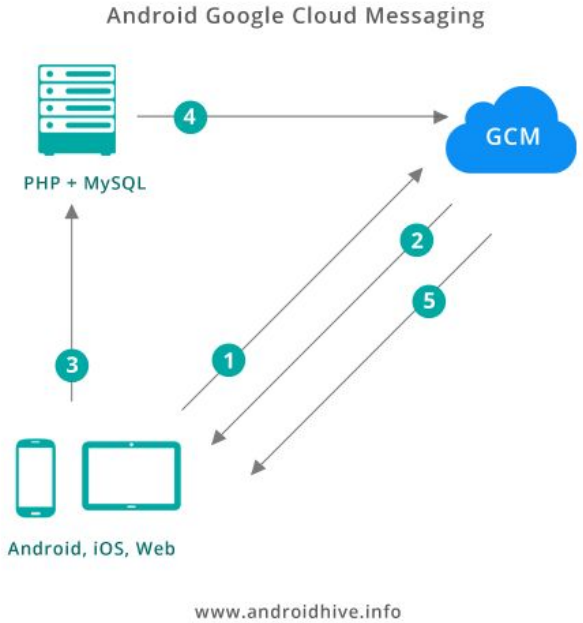
Util



State of the art

AppEngine, serverless

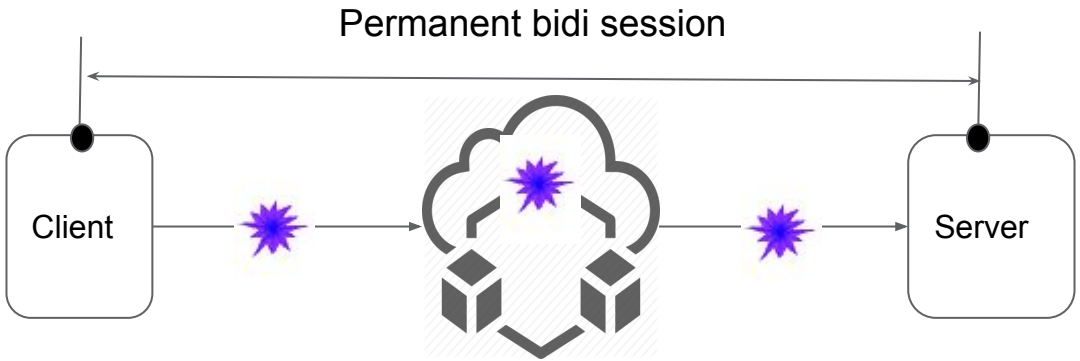
- Short sessions only, up to 60s
- Out-of-band pushes, via external messaging services



State of the art

Bidi-web (github.com/bidiweb)

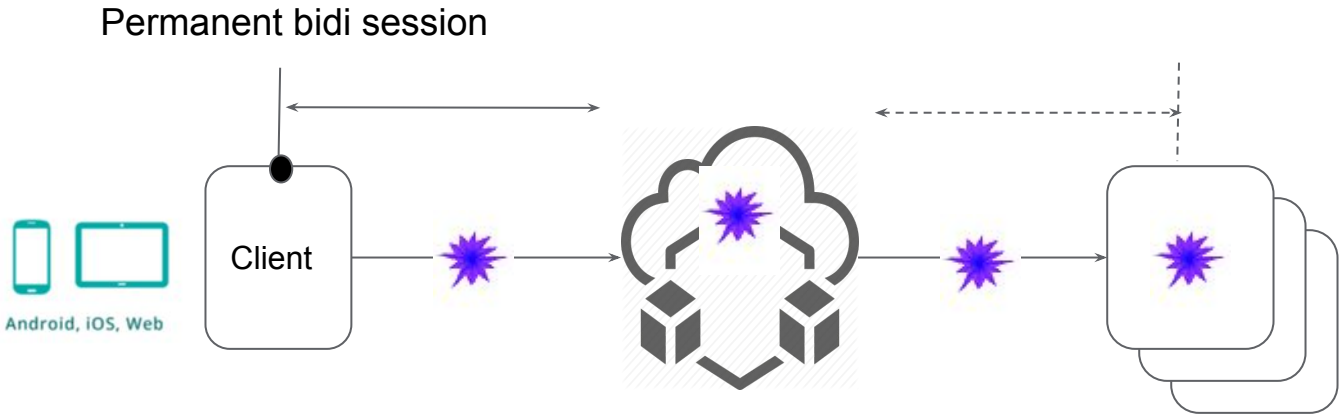
- A logical session over short & concurrent RPCs (HTTP)
- Server end-points are still stateful and long-lived



Can we have both?

Bidi-web model on serverless

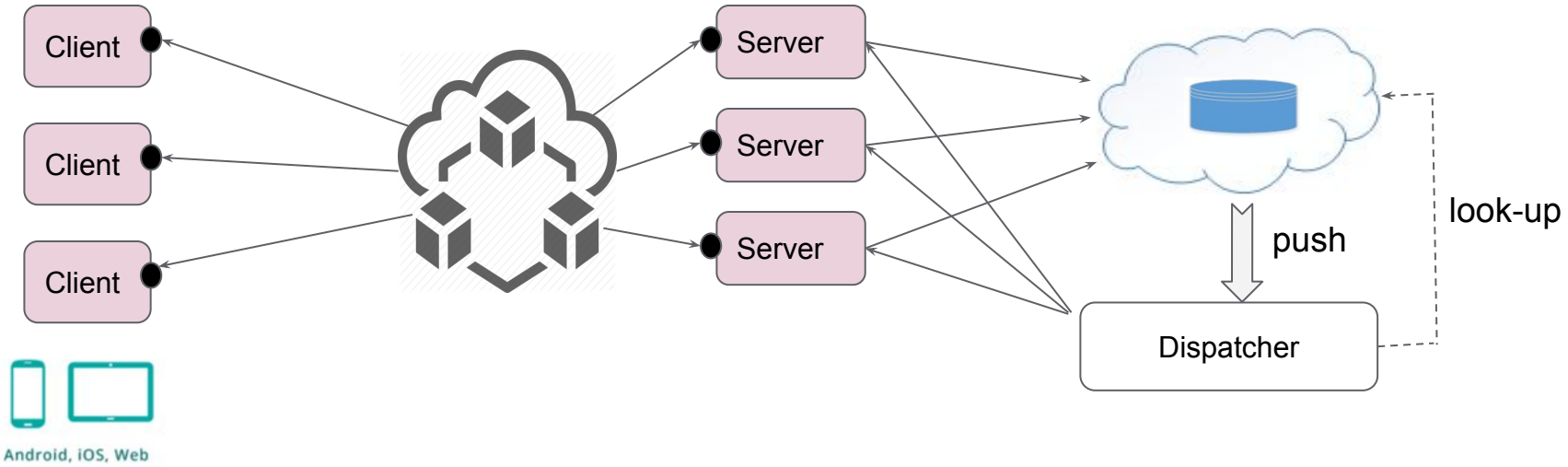
- Keep the client endpoint stateful
- Deploy the server endpoint as stateless functions



Case study: real-time chat

Stateful-server model

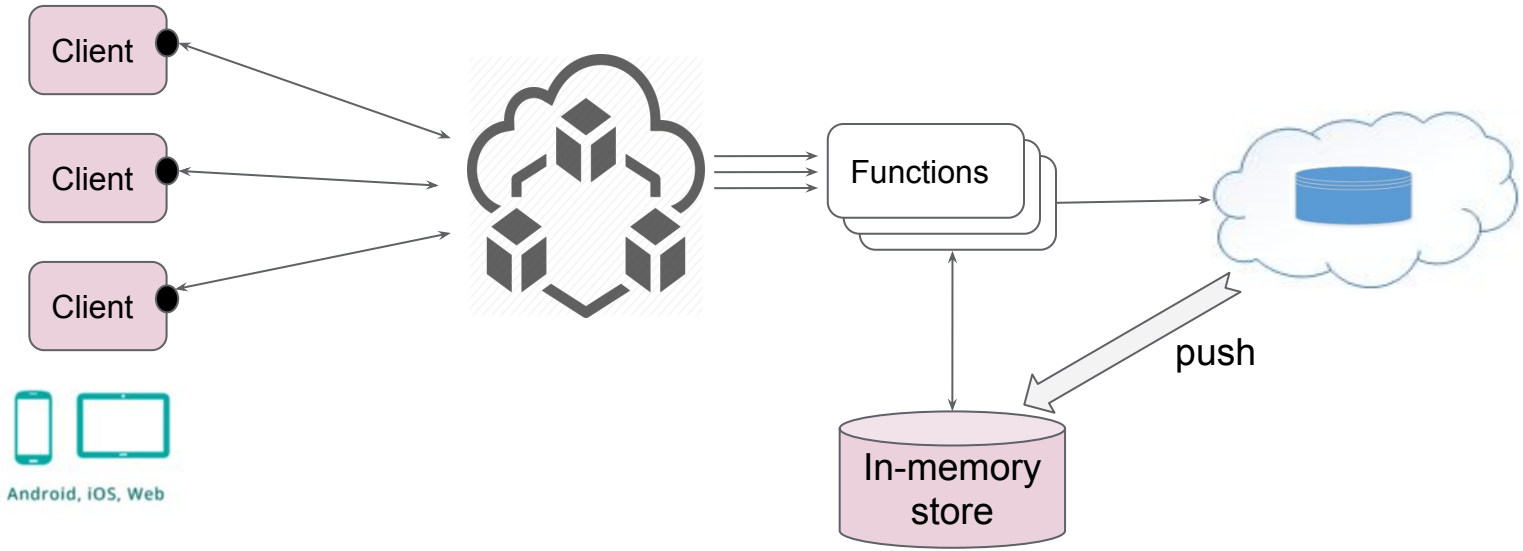
- A stateful end-point on the server-side for each client session
- Shared state for the “chat room”
- Stateless dispatcher to push messages to individual clients



Real-time chat on serverless

Externalized communication state

- In-memory store, e.g. Redis in a single cluster
- Stateless framework libraries as server functions



Accepts a new session

```
class ServerSession {
    void accept(...);
}
```

- *sid* represents an opaque client session id

Delivers a new client message

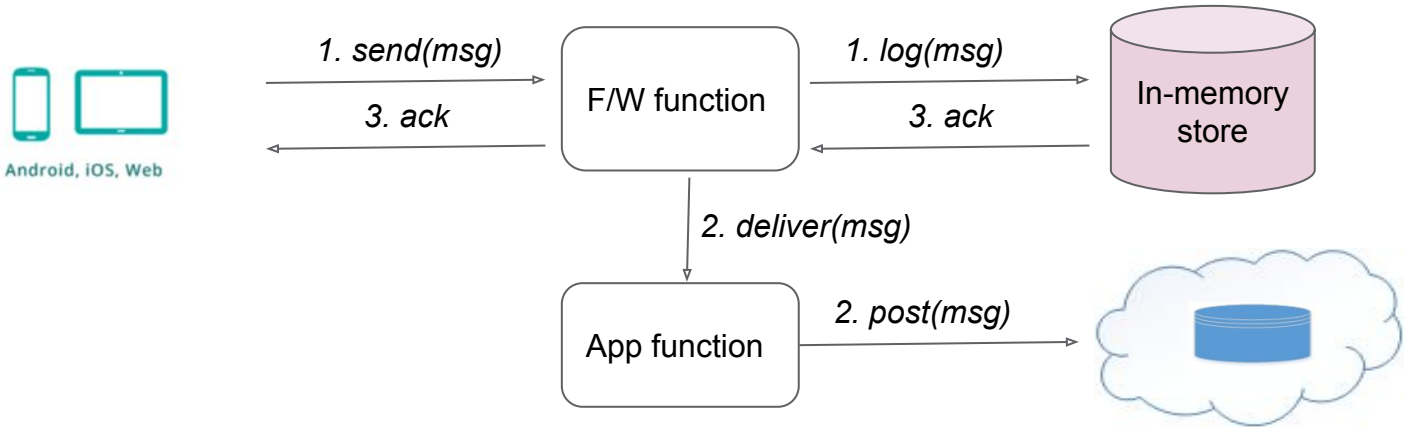
```
class ServerSession {
    void onMessage(msg);
}
```

- *send()* may include retried client messages
- *ack* may include retried server messages

Delivers a new client message

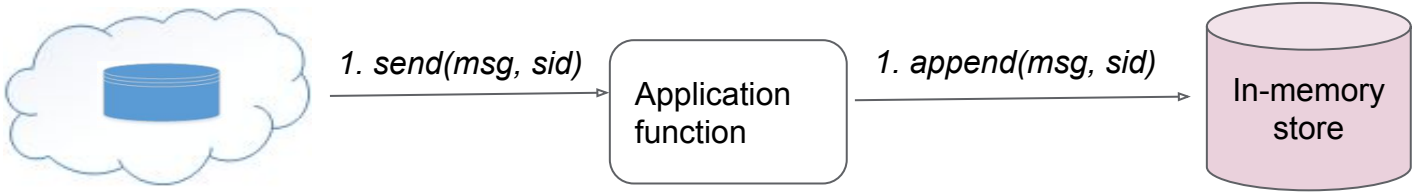
```
class ServerSession {
    void onMessage(msg);
}
```

- concurrent sends are expected
- messages are totally ordered



Delivers a new server message

```
class ServerSession {
    void send(msg):
}
```

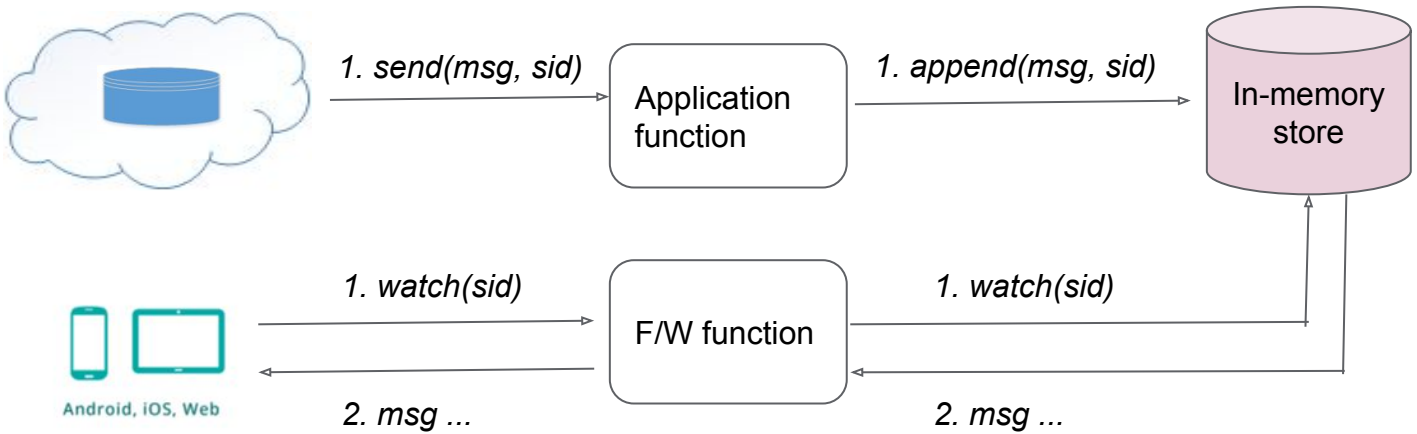


- storage layer to order messages per *sid*

Delivers a new server message

```
class ServerSession {
    void send(msg):
}
```

- *watch()* does be a short “Hanging GET”, e.g. < 60s



Framework roles

Ensure communication semantics

- Concurrency control of parallel client requests
- Message ordering, recovery, ack, flow-control ...

```
class ServerSession {
    void open(...);
    void onMessage(msg);
    void send(msg);
}
```

```
function open(sid, ...);
```

```
function onMessage(sid, ...);
```

```
function send(sid, ...);
```

Why not push services?

Compared to application-owned bidi endpoints

- Extra cost and latency
- Different ordering semantics

Future use cases

- Long-lived VR chats
- Extremely low-overhead storage, e.g. RAMCloud

Conclusion

- An intriguing architecture choice with both well-understood and new challenges
- Frameworks are expected to play a significant role

Questions, comments: web@google.com