# Coding PopVote

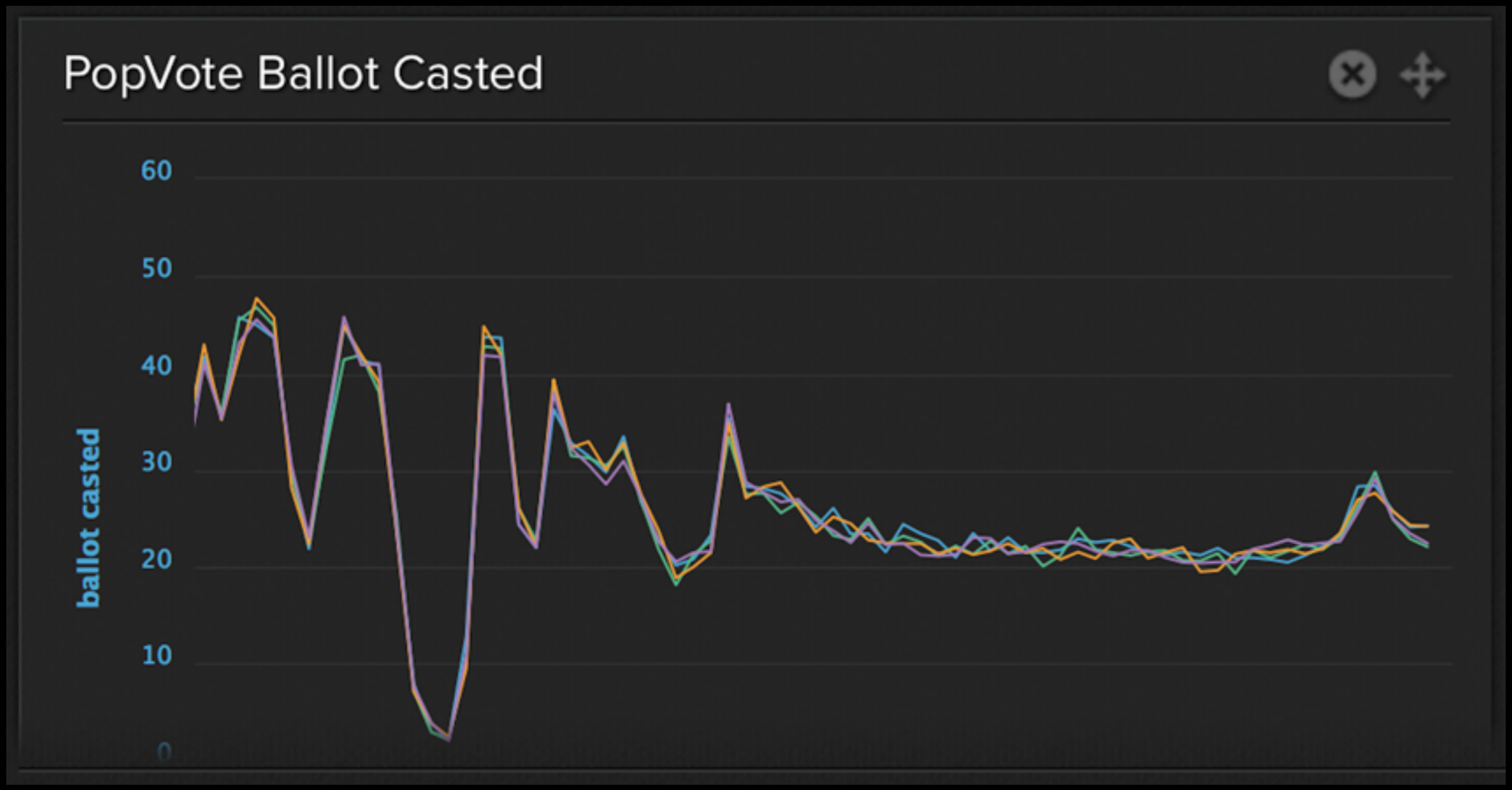
Patrick Cheung

PopVote backend developer

# Why am I here?

47 votes in 1 second

*highest throughput in any second*




first voting day (20 June)

> 70% votes casted in less then 180 seconds

*may include duplicated votes*

redis

puppet

tornado

flask

supervisord

fluentd

statsd

nginx

mcollective

fabric

mysql

uwsgi

vagrant

ubuntu

pycrypto

itsdangerous

python

r10k

pycaptcha

csshx

sqlalchemy

json

async io

uuid

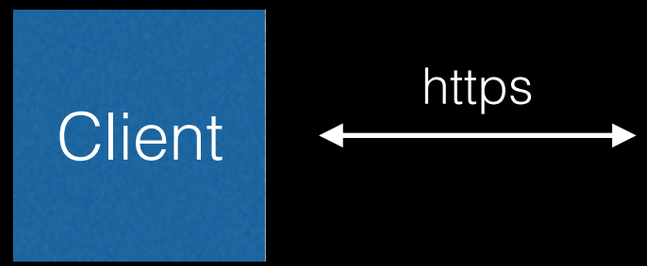
pubsub

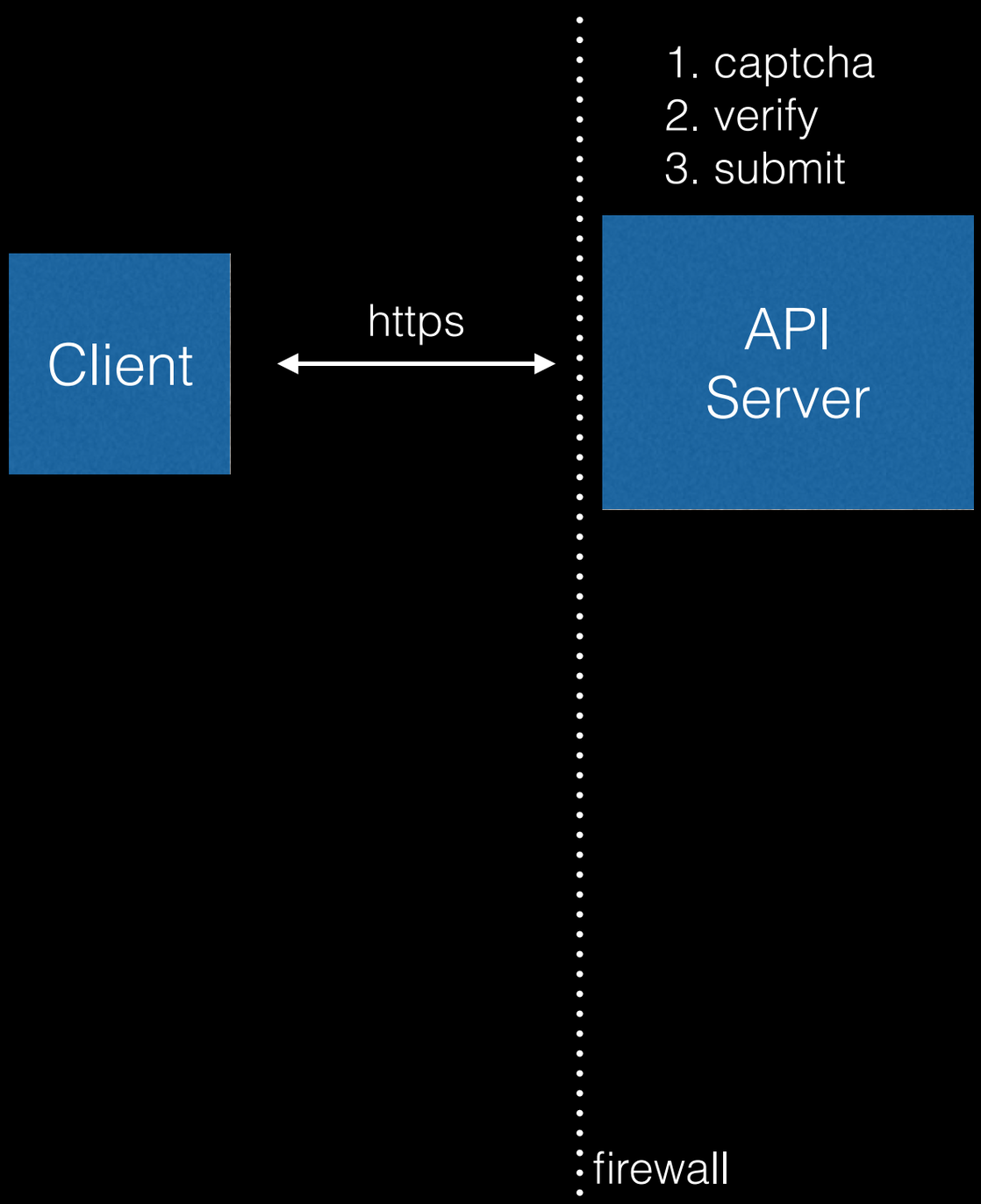
boto

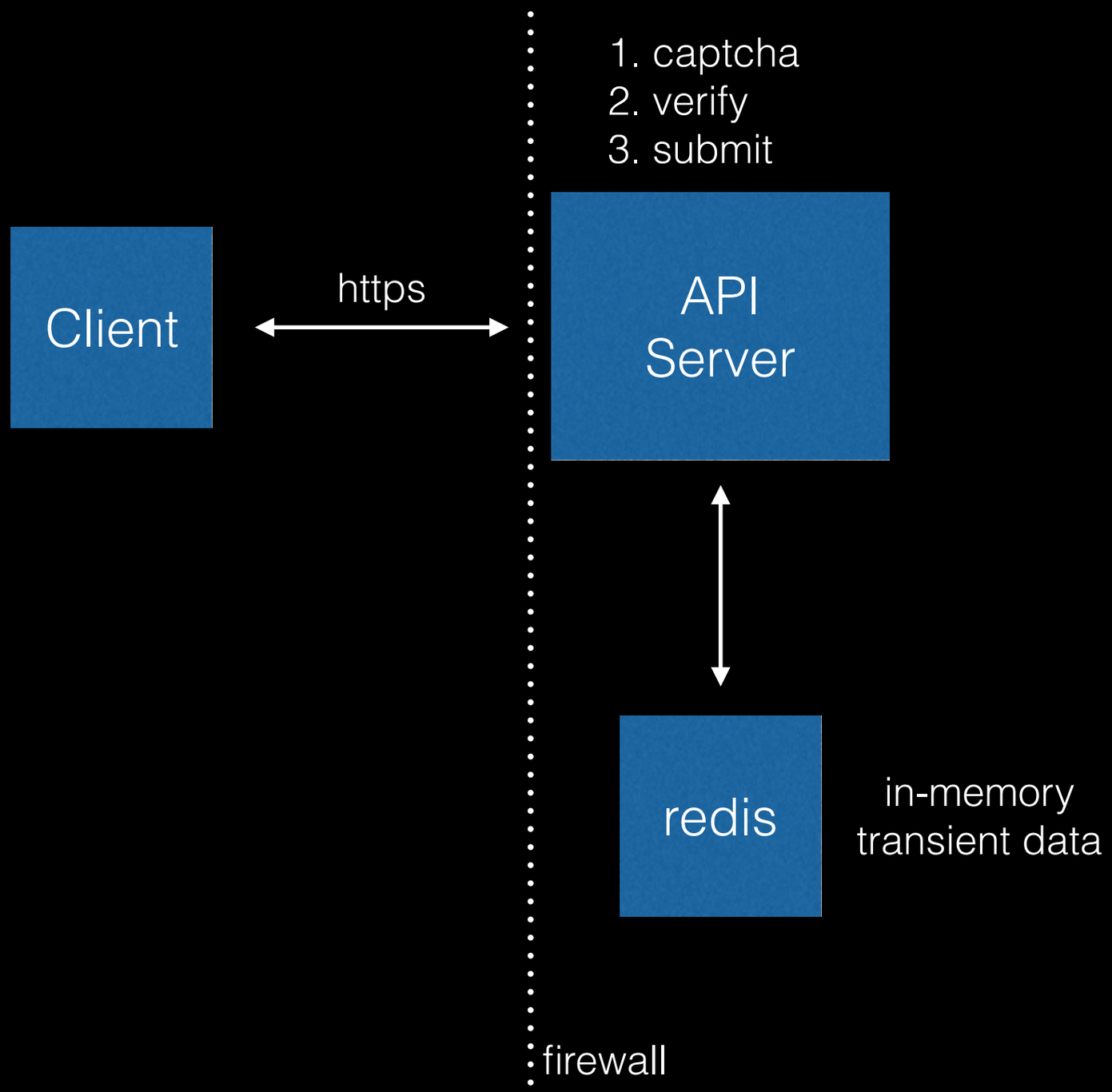
jwt

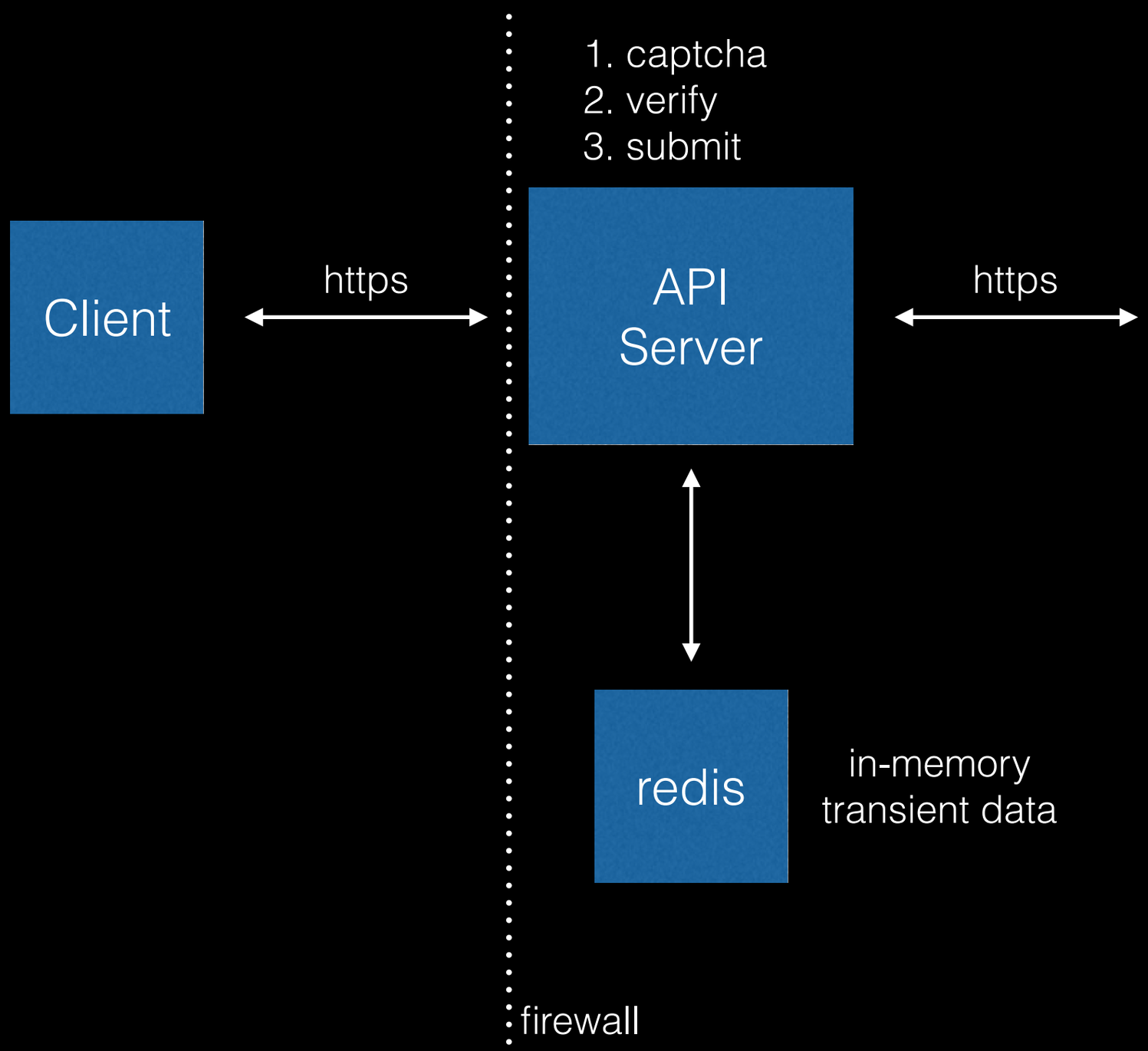
- 100% written in Python
- 18,000 line of code
- API Server
- Ballot Server
- Control Panel

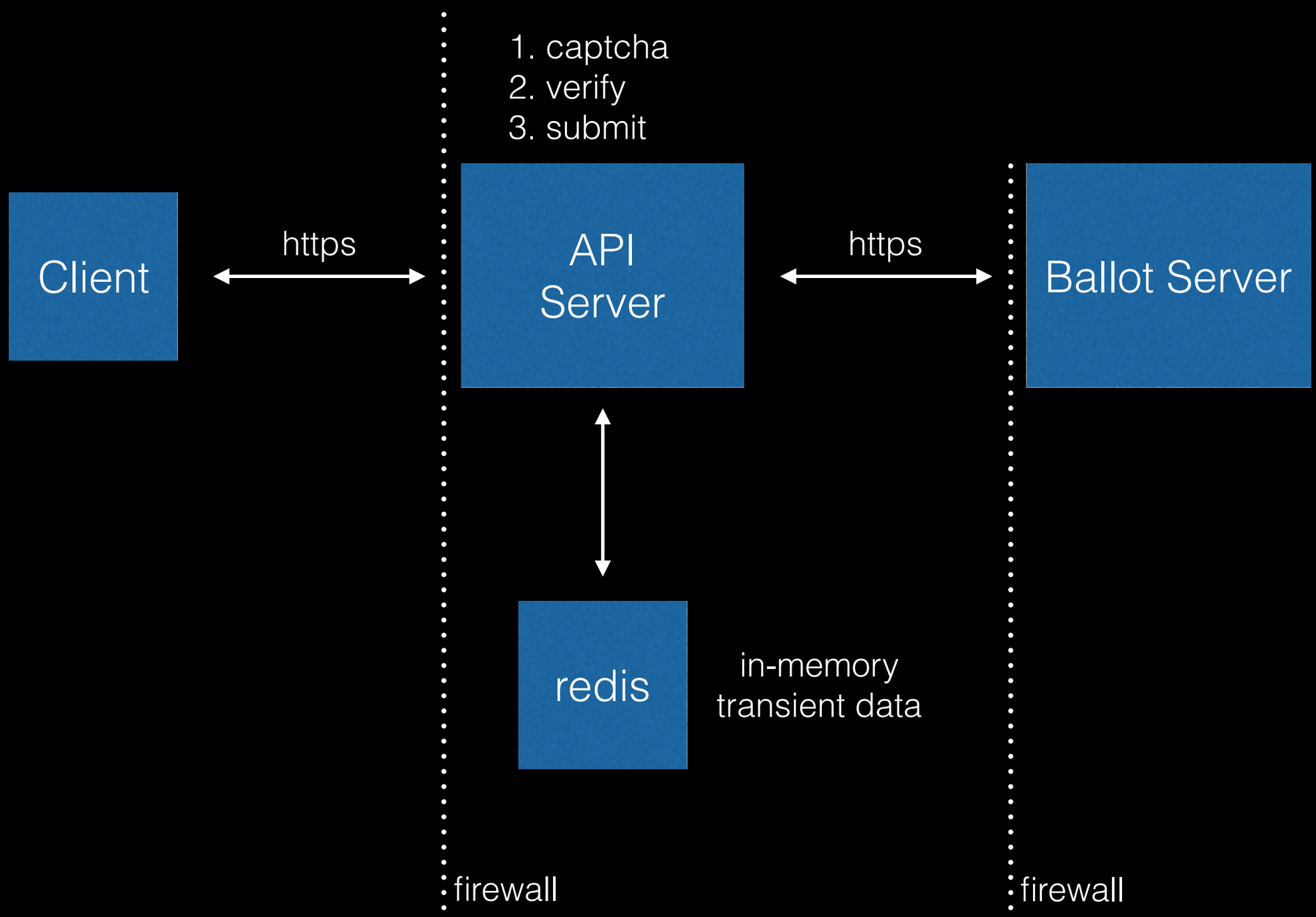
Client

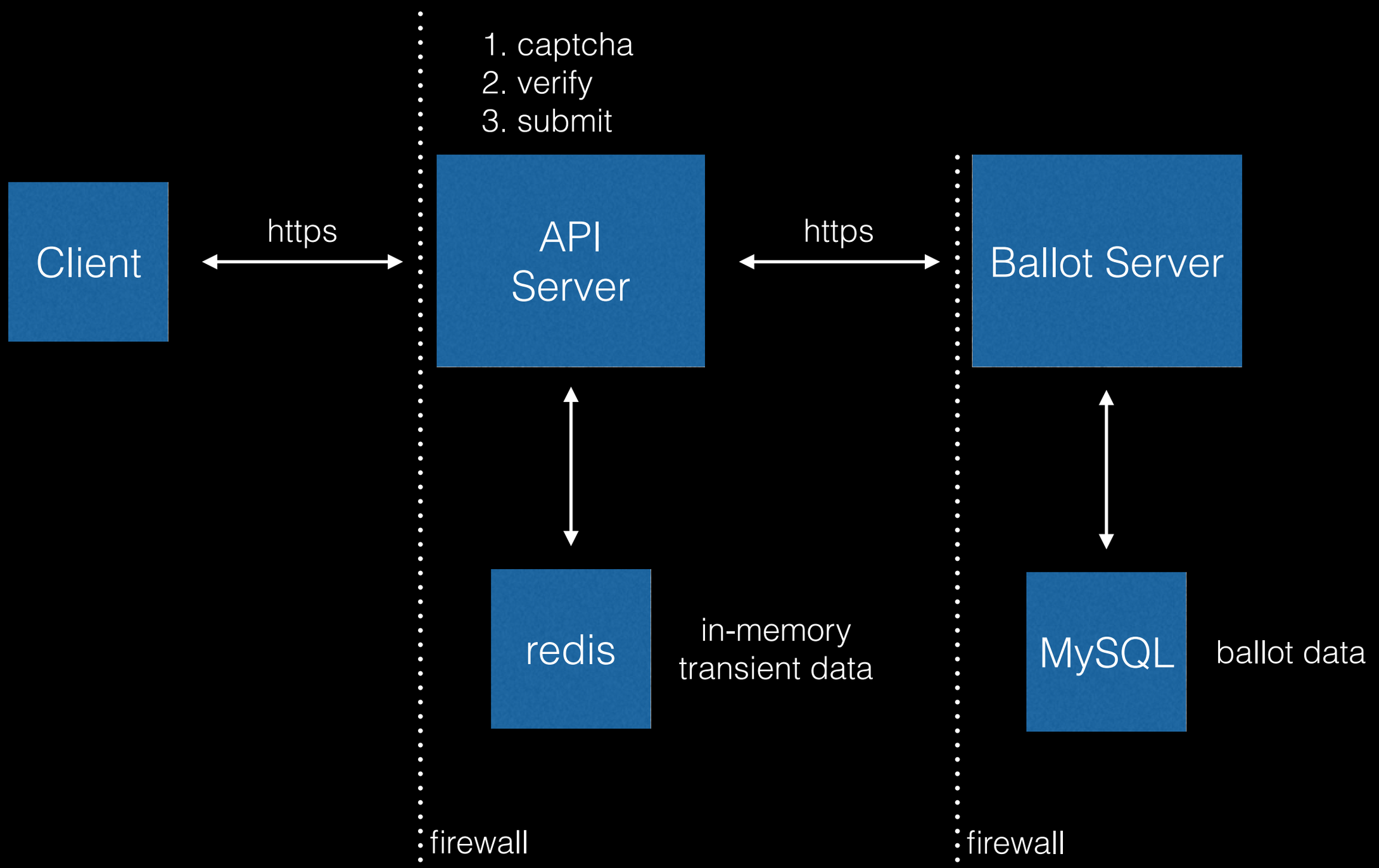

Client
https

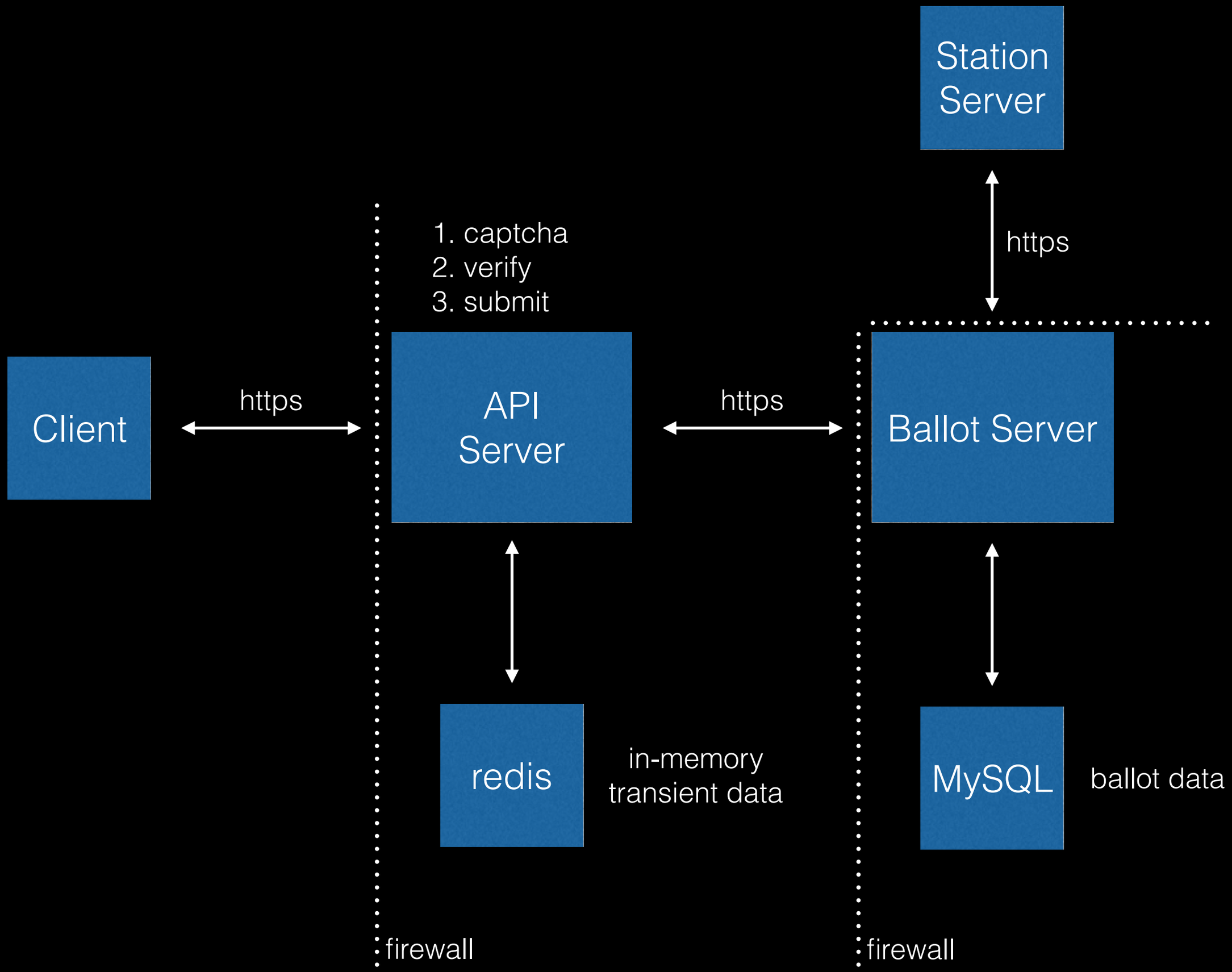

1. captcha
2. verify
3. submit
Client
https
API
Server
firewall


1. captcha
2. verify
3. submit
Client
https
API
Server
redis
in-memory
transient data
firewall


1. captcha
2. verify
3. submit
Client
https
API
Server
https
redis
in-memory
transient data
firewall


1. captcha
2. verify
3. submit
Client
https
API
Server
https
Ballot Server
redis
in-memory
transient data
firewall
firewall


1. captcha
2. verify
3. submit
Client
https
API
Server
https
Ballot Server
redis
in-memory
transient data
MySQL
ballot data
firewall
firewall


Station Server
https
1. captcha
2. verify
3. submit
Client
https
API Server
https
Ballot Server
redis
in-memory transient data
MySQL
ballot data
firewall
firewall

# developing a system

# developing a system

- short development timeframe

# developing a system

- short development timeframe
- system load gradually increase

# developing a system

- short development timeframe
- system load gradually increase
- optimise system with real load

# developing a system

- short development timeframe
- system load gradually increase
- optimise system with real load
- not mission critical

# developing popvote

- short development timeframe
- system load gradually increase
- optimise system with real load
- not mission critical

# developing popvote

- short development timeframe
- system load gradually increase ❌
- optimise system with real load
- not mission critical

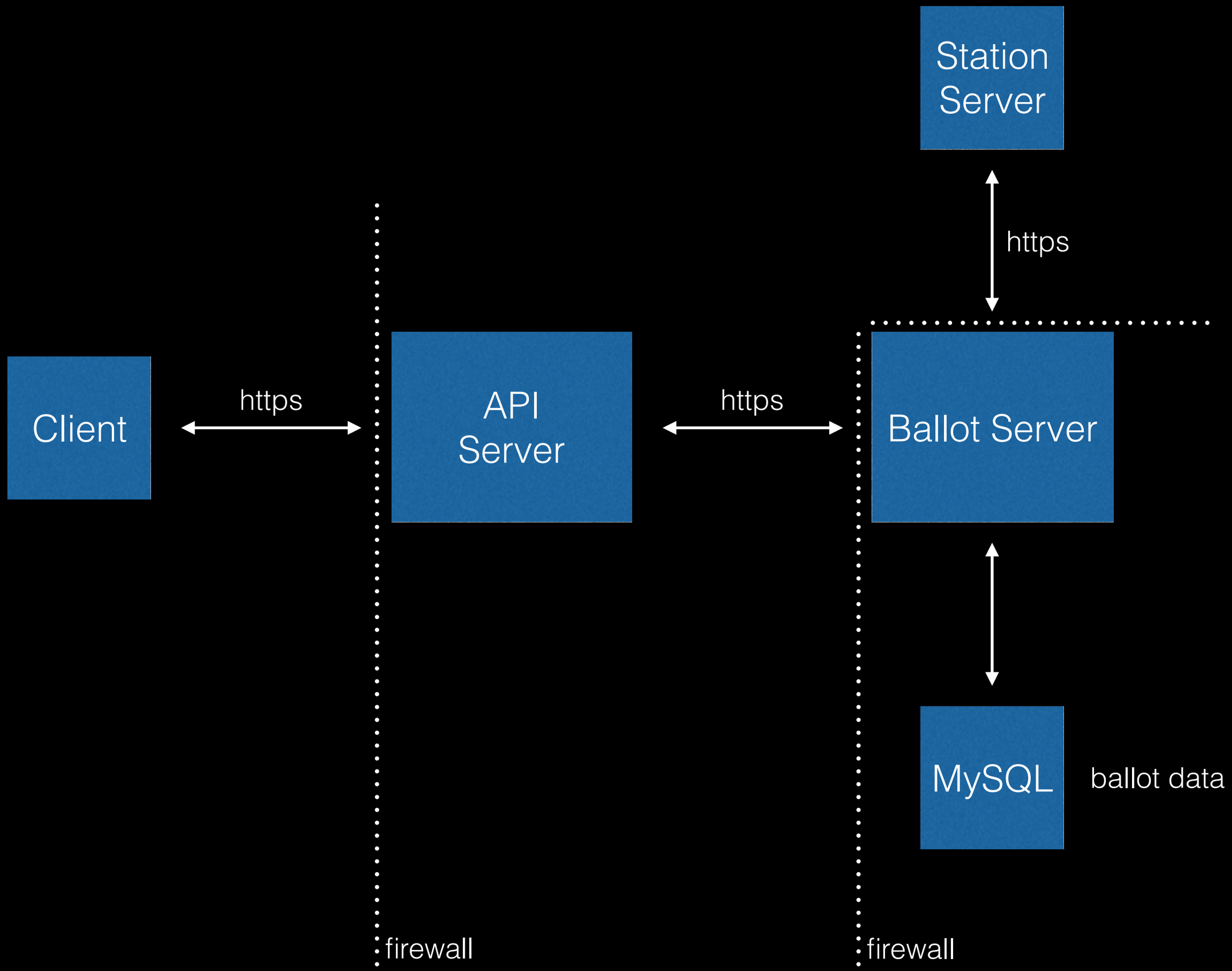
# developing popvote

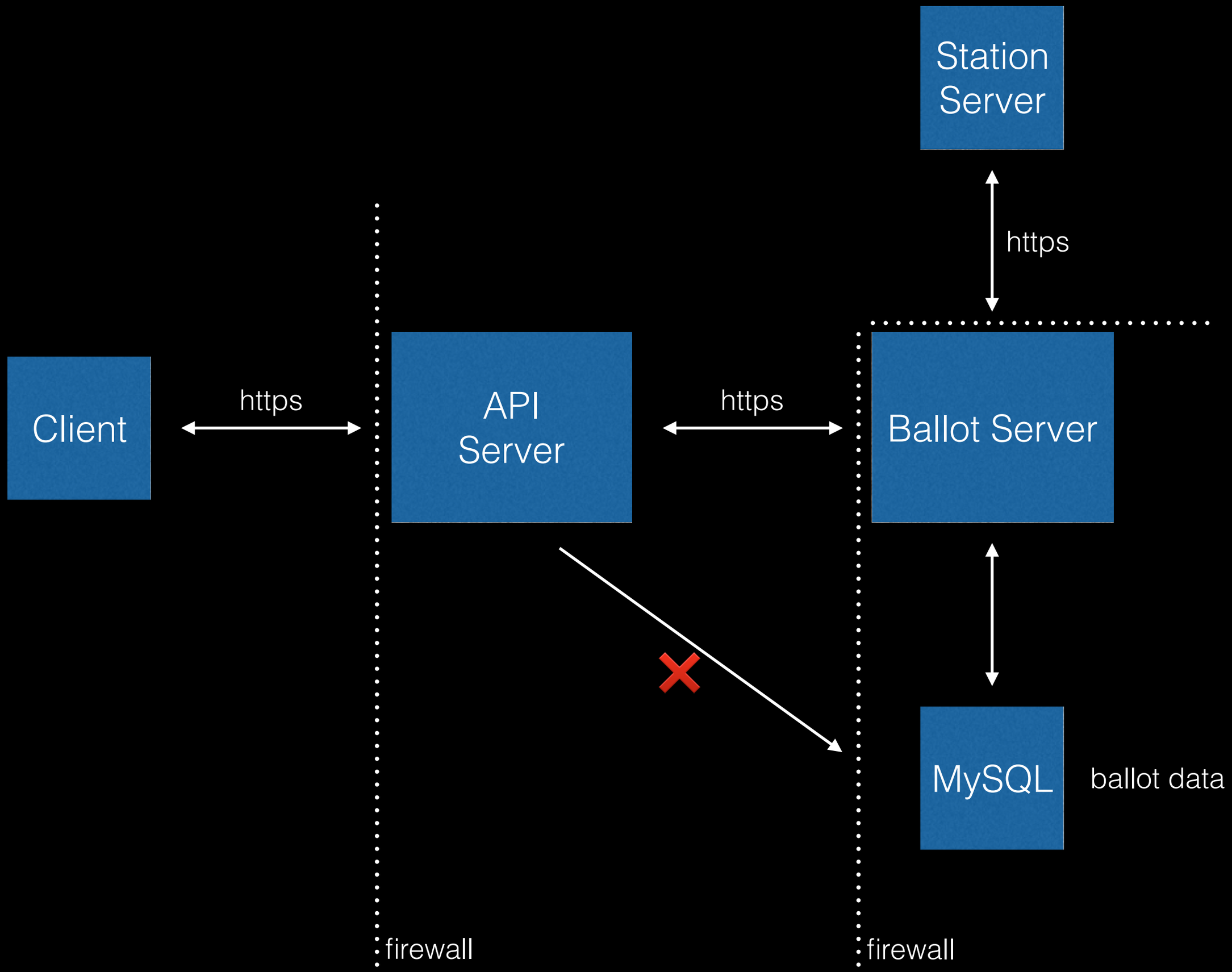
- short development timeframe
- system load gradually increase ❌
- optimise system with real load ❌
- not mission critical

# developing popvote

- short development timeframe
- system load gradually increase ❌
- optimise system with real load ❌
- not mission critical ❌

Public-facing web servers do not have access to the database


Station Server
https
Client
https
API Server
https
Ballot Server
MySQL
ballot data
firewall
firewall

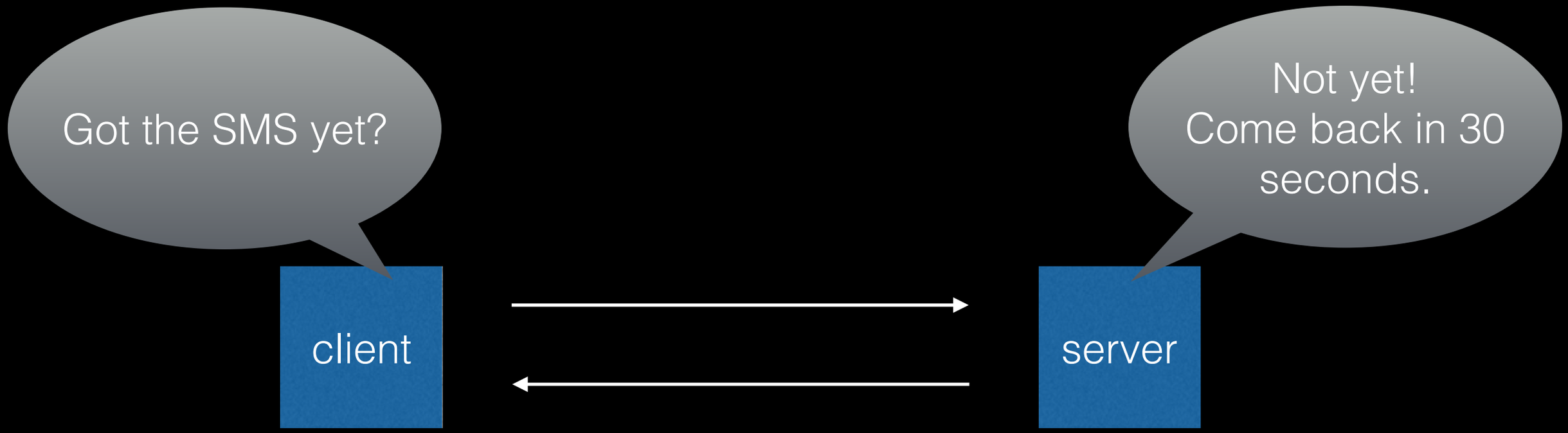

Station Server
https
Client
https
API Server
https
Ballot Server
MySQL
ballot data
firewall
firewall

Users are notified immediately when they are successfully verified

client

server

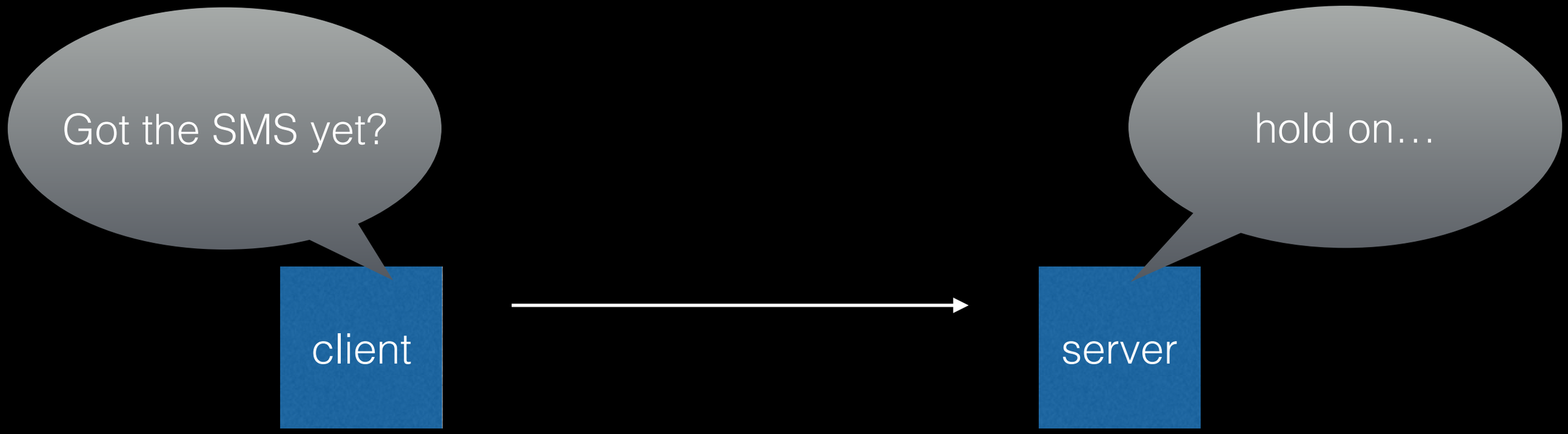

Got the SMS yet?
client
server

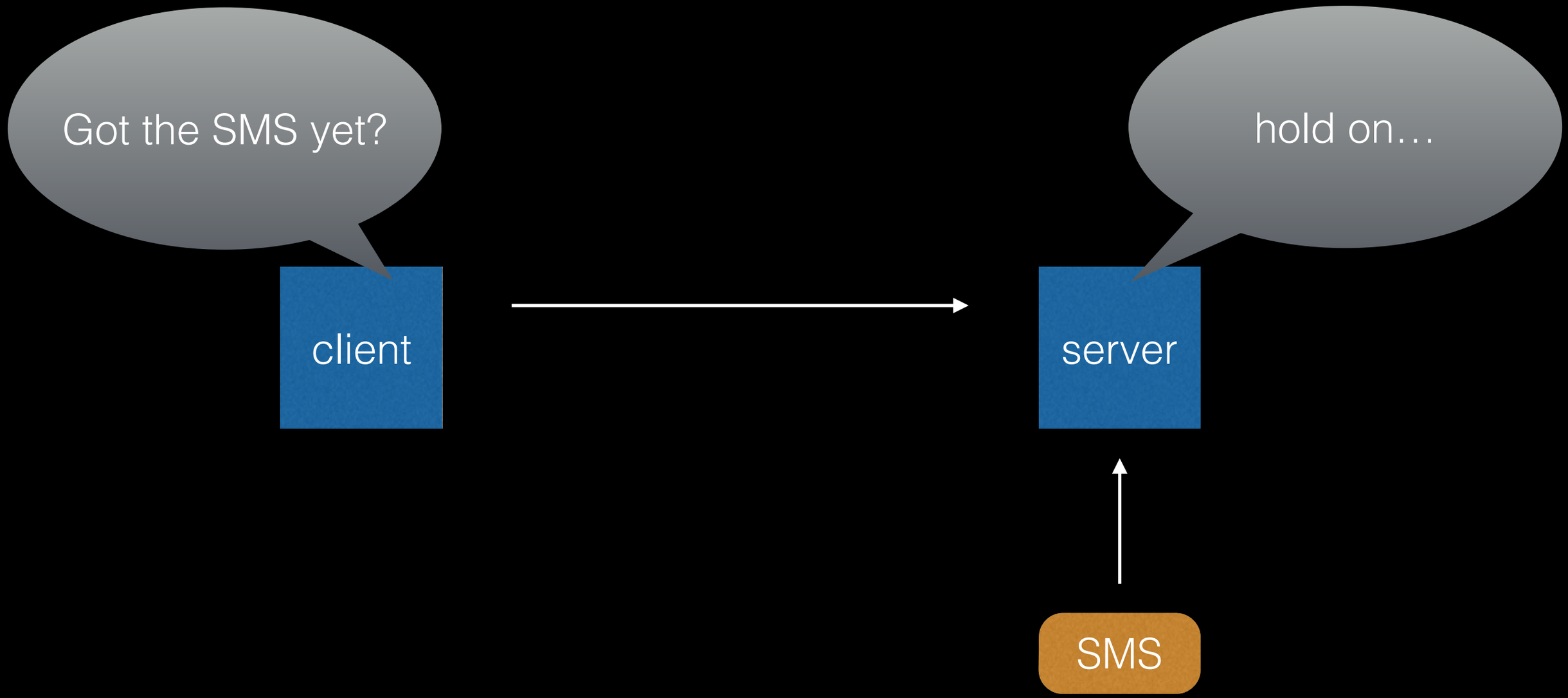

Got the SMS yet?
Not yet!
Come back in 30
seconds.
client
server

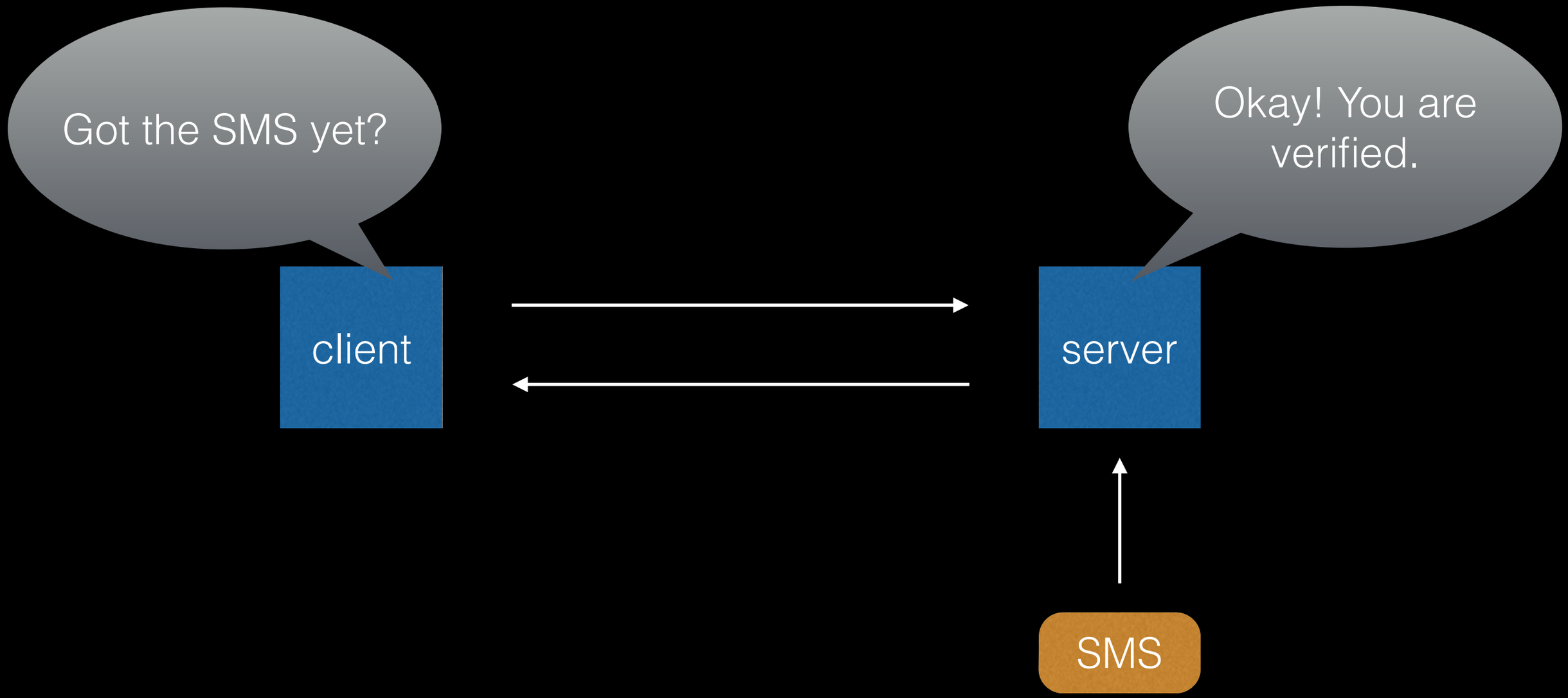
client

server


Got the SMS yet?
client
server


Got the SMS yet?
client
server
hold on…


Got the SMS yet?
client
server
hold on…
SMS


Got the SMS yet?
client
server
Okay! You are verified.
SMS

- Tornado Web Server and Async IO
- One thread can handle multiple clients at the same time
- Keep client connection opens
- Server subscribes to SMS events through redis
- Very fast response

# Deployment

a mix of Puppet+CloudFormation

- Cloud infrastructure created with CloudFormation
- Make use of EC2 Auto-scaling and Multi-AZ for high availability
- Application installed automatically when server start
- Automatically configured by Puppet Master

# Design highlights

# API Design

## PopVote Public Channels API

Date: 22 April 2014

Version: 0.5

## Overview

The Public Channels API implements an interface for the client to submit ballot data in a voting session. This is a three-step process:

1. Submit voter info.

   The client submits voter info.

   The client receives an access token, which is required in all subsequent steps.

# API Design

- Client sends/receives
- Clear separation between client and server code
- Rapid client development
- Makes server simple

# Stateless Design

- Avoid storing session state on server
- Session state exchanged with client in encrypted form
- Good for privacy
- Good for server performance and operation

# Duplicate votes

- Duplicate vote checked at the very last step
- Vote is recorded if not duplicate
- Minimise participation checking by attacker
- Makes web server simple
- Database access restricted to ballot server

# Further Improvement

- Diversify choice of cloud providers
- Application containment using Docker
- Consolidate server resources