Service Oriented Architecture with Python for MMO Games

Harry Liang
harry@happylatte.com

About Happylatte

- Previously Exoweb, member of PSF Sponsors
- 10+ years of experience in Python
- Now on journey to build smartphone classics

10+ million downloads

millions of MAUs

High Noon 2


FIND AN OPPONENT!
SALOON
193
CLICK TO BUY
CLICK TO BUY
CLICK TO BUY
CLICK TO BUY
CLICK TO BUY
CLICK TO BUY

HighNoon Server Platform

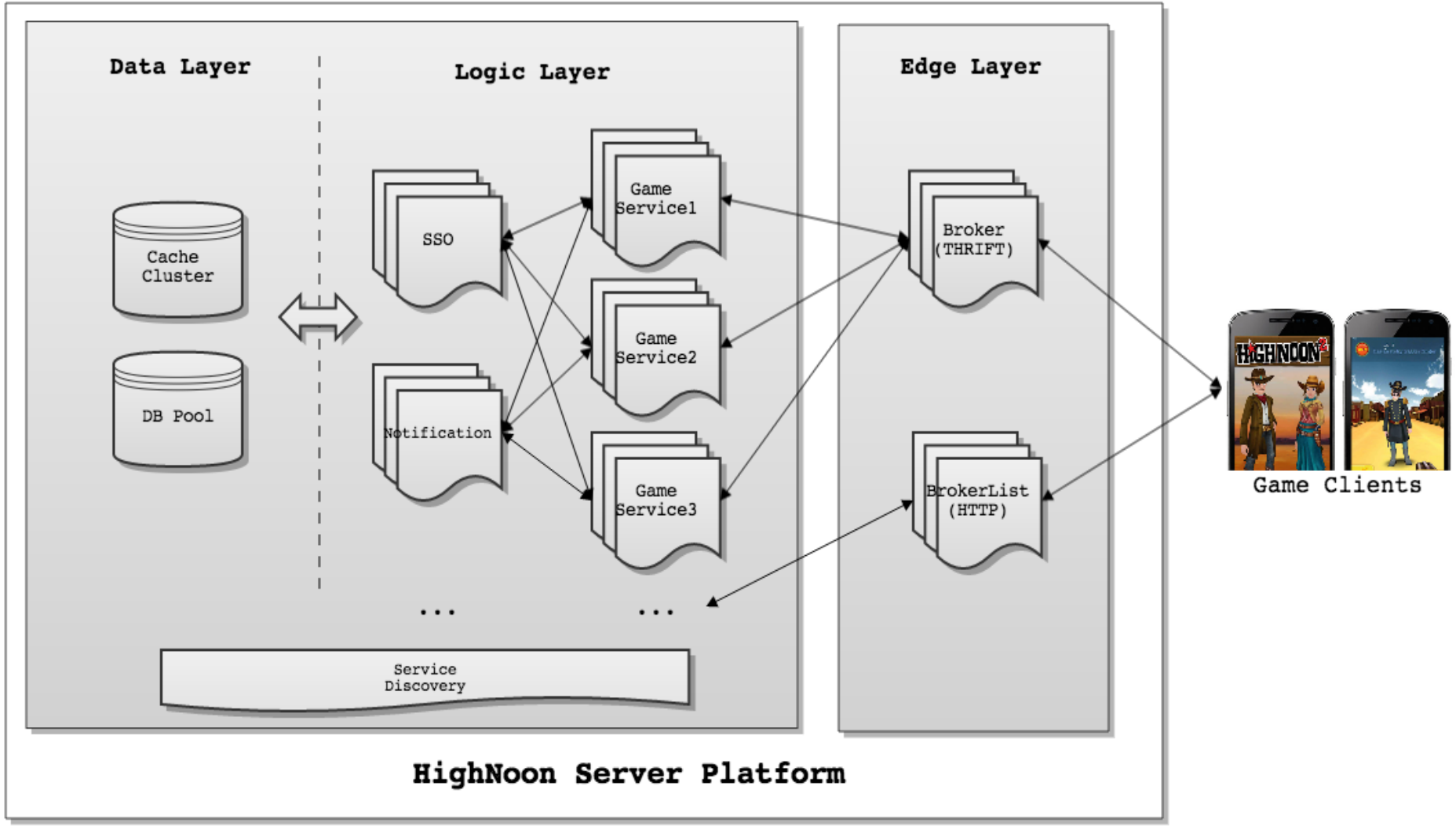
Service Oriented Architecture

Totally Python
ZeroMQ + Twisted, Eventlet/Gevent

100+ service instances running

Key Principles

- Point-to-Point Messaging
- Discovery & Presence
- Fault Tolerance
- Load Balance
- Stateful vs. Stateless


Data Layer
Logic Layer
Edge Layer
Cache Cluster
DB Pool
SSO
Notification
Game Service1
Game Service2
Game Service3
Broker (THRIFT)
BrokerList (HTTP)
Service Discovery
Game Clients
HighNoon Server Platform

Edge Layer

Interface between game clients and the platform

- Publically accessible
- Delegate real work down to logic layer
- Exist to isolate logic from interface specifics

HappyBroker

- Packets Routing

  static:

  msg type -> worker type (load balance)

  dynamic:

  session id × msg type -> stateful service instance

- Protocol Translation

  ZMQ frames <-> Thrift packets

- Keep Client Connections

BrokerList

The only known endpoint by the client

Publishes where the brokers are:

```
{
    "version":"0.1.0",
    "brokerList":{
        "ec2-54-208-53-1.compute-1.amazonaws.com:1055":0.067,
        "ec2-54-208-53-2.compute-1.amazonaws.com:1055":0.0418
    }
}
```

Logic Layer

Service = n * Servers + m * Clients

- Asynchronous Networking
- Point-to-point Messaging
- Request-Reply/Push-Pull/Pub-Sub
- Synchronous Client API

Service: Server

While True:

    session, req_msg = zmq_sock.read()

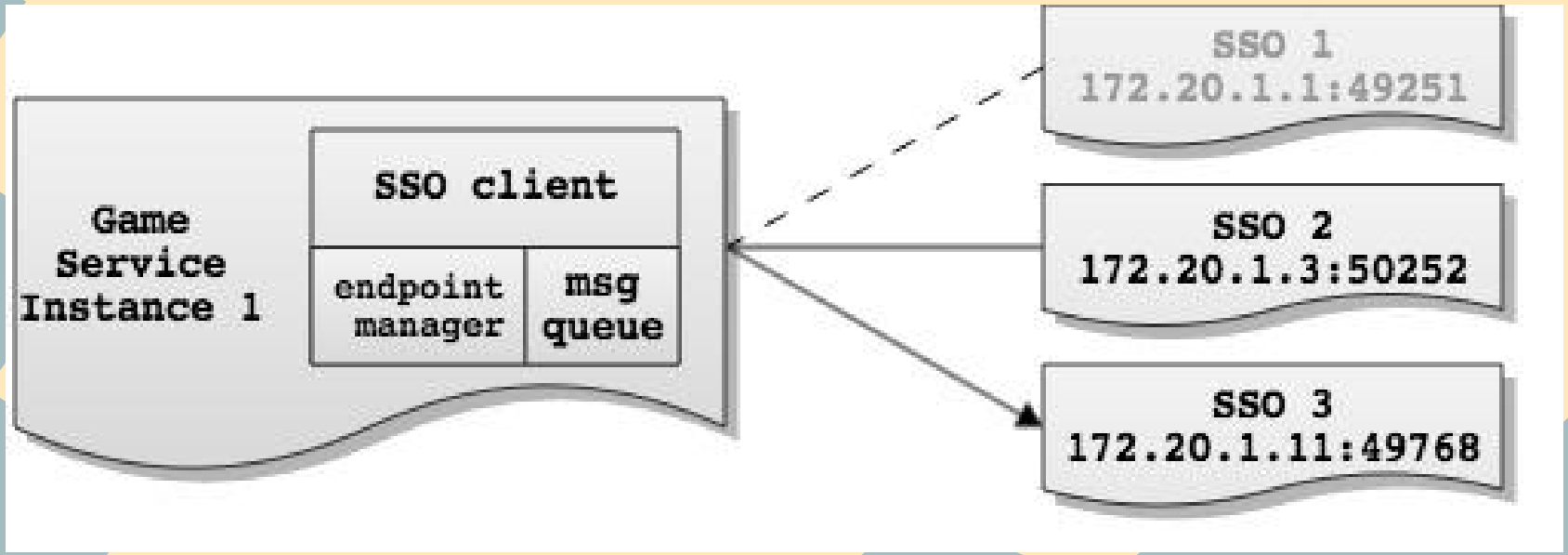
    handler = handlers.get(‘on_’ + req_msg.name)

    spawn(handler, session, req_msg)

Service: Client

sso_client = get_service(‘sso’, ‘REP’)

sso_client.login(user=’harry’, passwd=’my_dirty_secret’)



Yellow Service

Yellow Server:

- Monitor Service State
- Provision Service Endpoint Queries

Yellow Client:

- For service provider: Register to Yellow Server
- For service consumer: Query endpoints from Yellow Server

Data Layer

DB

A Pgpool of PostgreSQLs

Cache

A pre-sharded Redis Cluster with Twemproxy & Sentinel

Strengths

Scalability

Reliability: thanks, ZeroMQ

Almost..

- 0 network conf
- 0 downtime

To be implemented..

Distributed logging and monitoring

Auto-scaling

Q & A

blurrcat@gmail.com

Thanks!

We’re hiring!
https://www.happylatte.com