網易 NetEase
www.163.com

Building fast, scalable game server in node.js

网易杭州研究院
谢骋超
@圈圈套圈圈
@xiecc

網易 NetEase
www.163.com

Introduction to pomelo

https://github.com/NetEase/pomelo

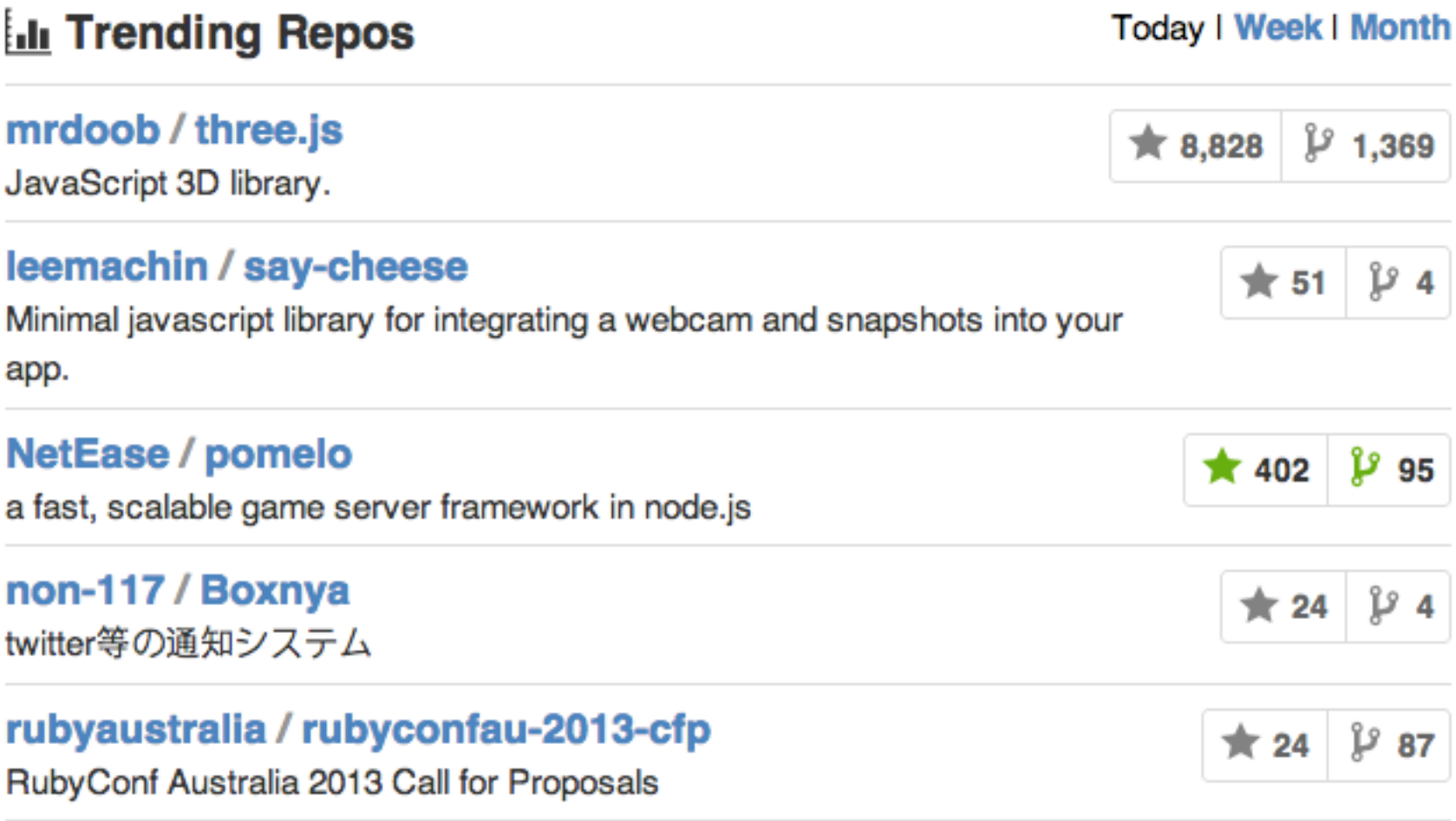
- Fast, scalable, distributed game server framework in node.js

- Open sourced in

2012.11.20

網易 NetEase
www.163.com

Github trending Repos---2nd day



Github---most popular

Most Starred Today

- NetEase / **pomelo**
- wojodesign / **simplecart-js**
- ftlabs / **fastclick**
- CloudMade / **Leaflet**
- camerond / **jquery-minical**

Most Forked Today

- binaryjs / **node-binarypack**
- binaryjs / **js-binarypack**
- na / **binaryjs**
- daeq / **programmer-site**
- hakimel / **reveal.js**

Most Starred This Week

- NetEase / **pomelo**
- wojodesign / **simplecart-js**
- ftlabs / **fastclick**
- CloudMade / **Leaflet**
- camerond / **jquery-minical**

Most Forked This Week

- binaryjs / **node-binarypack**
- binaryjs / **js-binarypack**
- na / **binaryjs**
- daeq / **programmer-site**
- hakimel / **reveal.js**

Scope of application

- Game
  - Web, socail, mobile game
  - Medium size client side game
- Realtime web
  - More scalable than other frameworks

What is this lecture about

How to create game using pomelo?

No

- Scalable game server architecture
- Extensible framework
- Performance

Category

- Scalabable game server architecture
- Extensible game server framework
- Performance

Scalability---Web and Game server

- Web server

  unlimited scalability

- Game server

World of tanks(bigworld)： 74,536 online users

MMORPG： 7k-8k maximum

Why does game server not scale?

- Long connection VS request/response
  - Game server and realtime web
  - Long connection: pull/push
  - Response time

Web response time: 2000ms

Game、realtime web: 100ms

How to solve

- Node.js to rescue

Perfect for:

Fast scalable network applications

Real-time application

Erlang and node.js

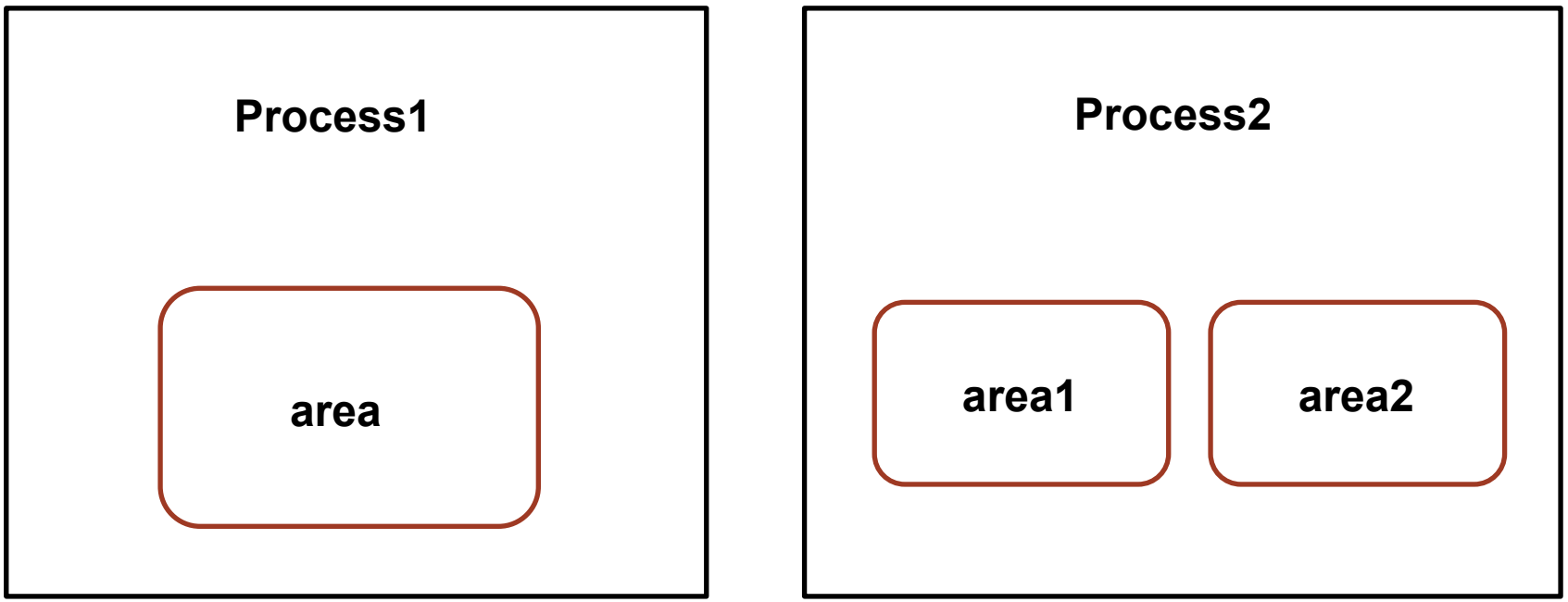
Why does game server not scale

- Web app（not realtime），no realtime interact
  - No coordinates, no adjacency
  - Partition randomly, stateless

- Game server, realtime interact
  - Have coordinate, no adjacency
  - Partition by area, stateful

How to solve

- Paritition by area
  - One area in one process （or many areas in one process）
  - The scalability of server is limited by process

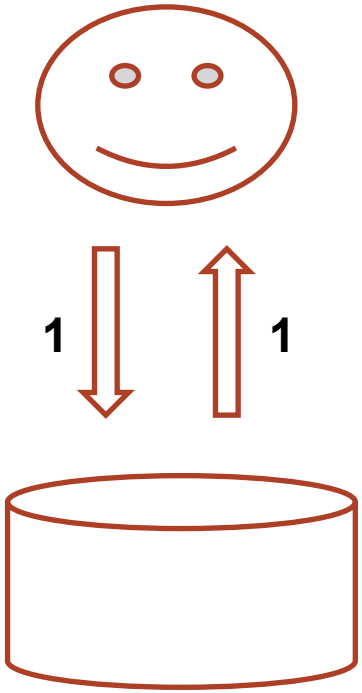


Why does game server not scale

Broadcast

MESSAGES IN

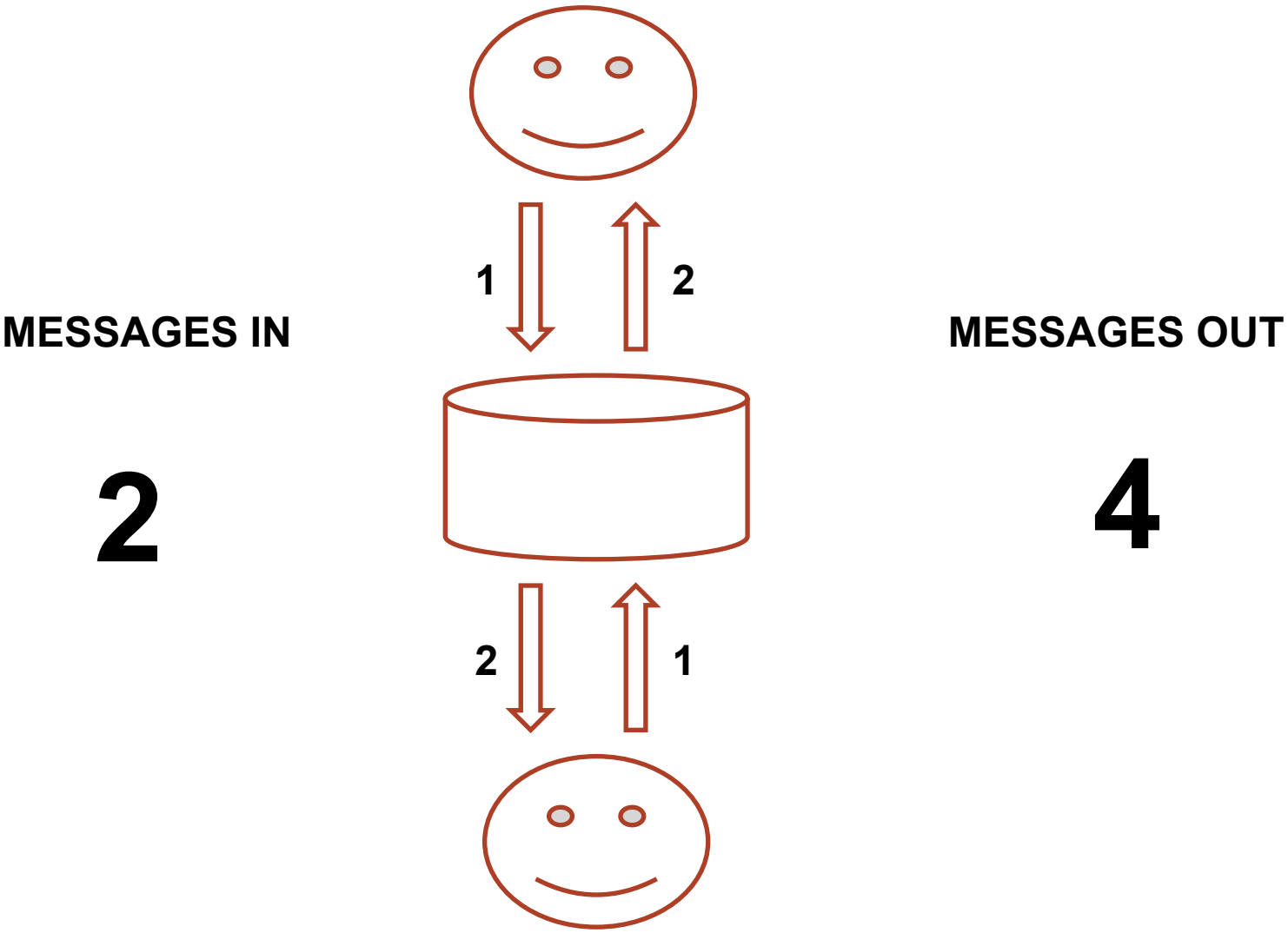
1



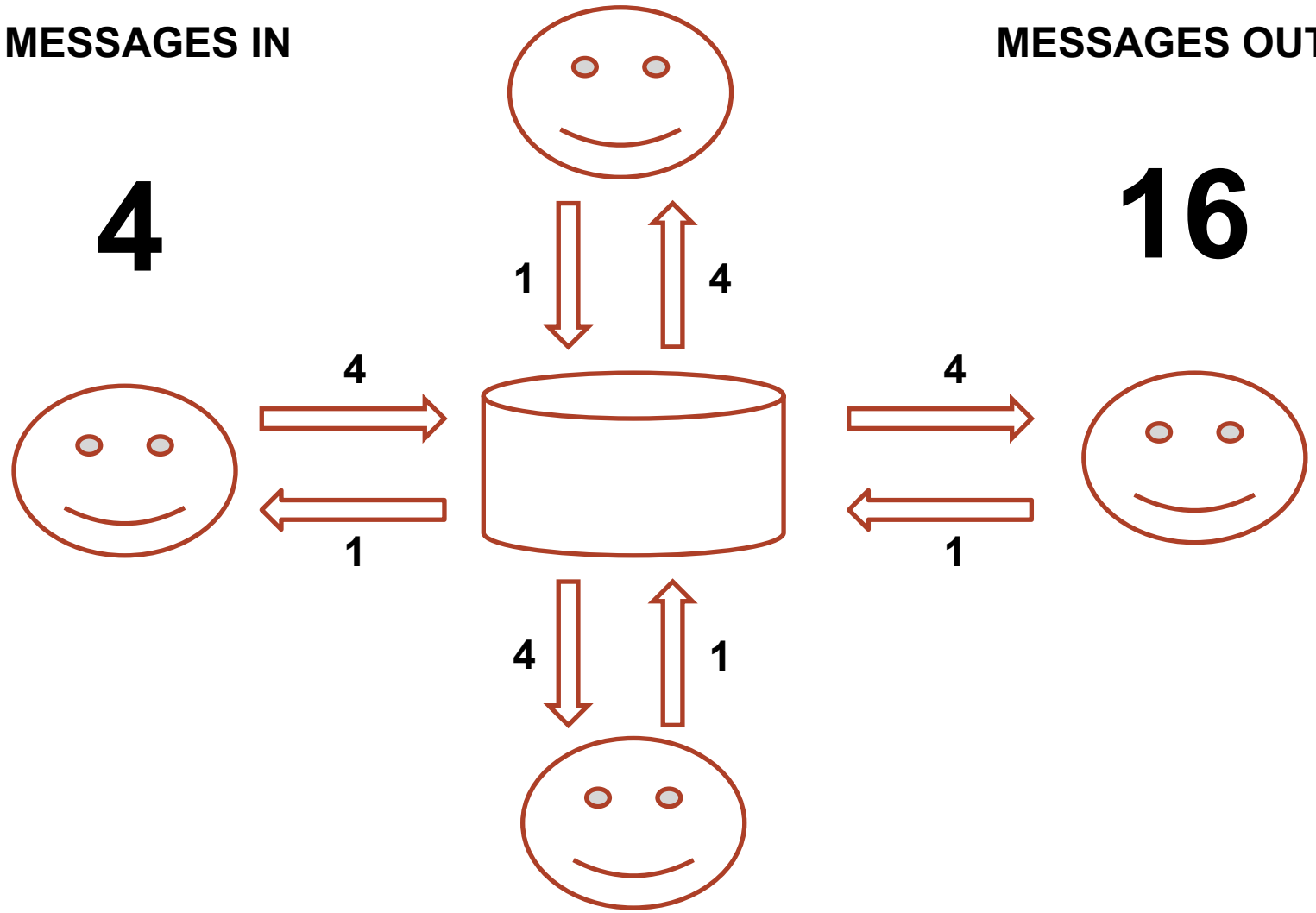
MESSAGES OUT

1

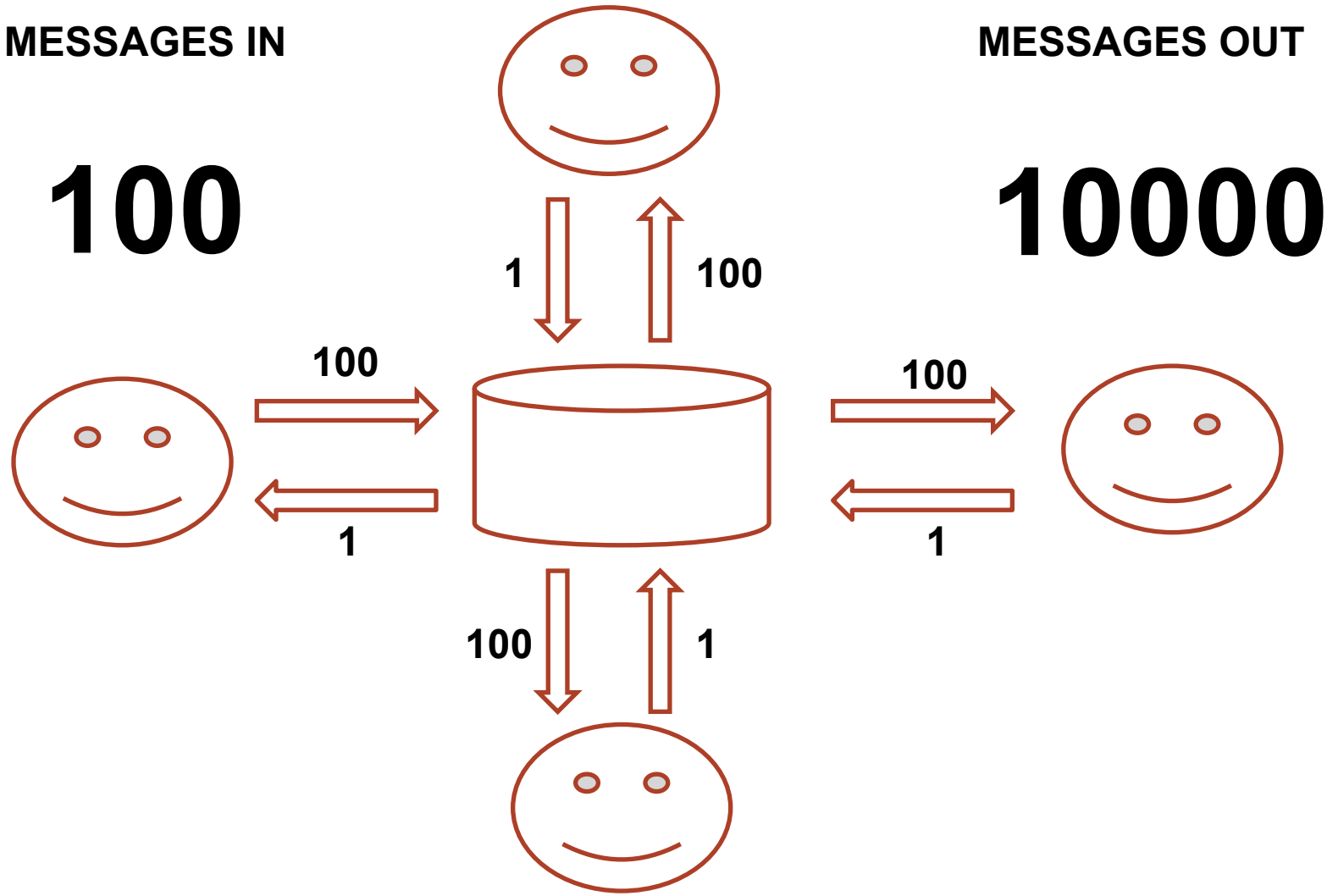
Why does game server not scale



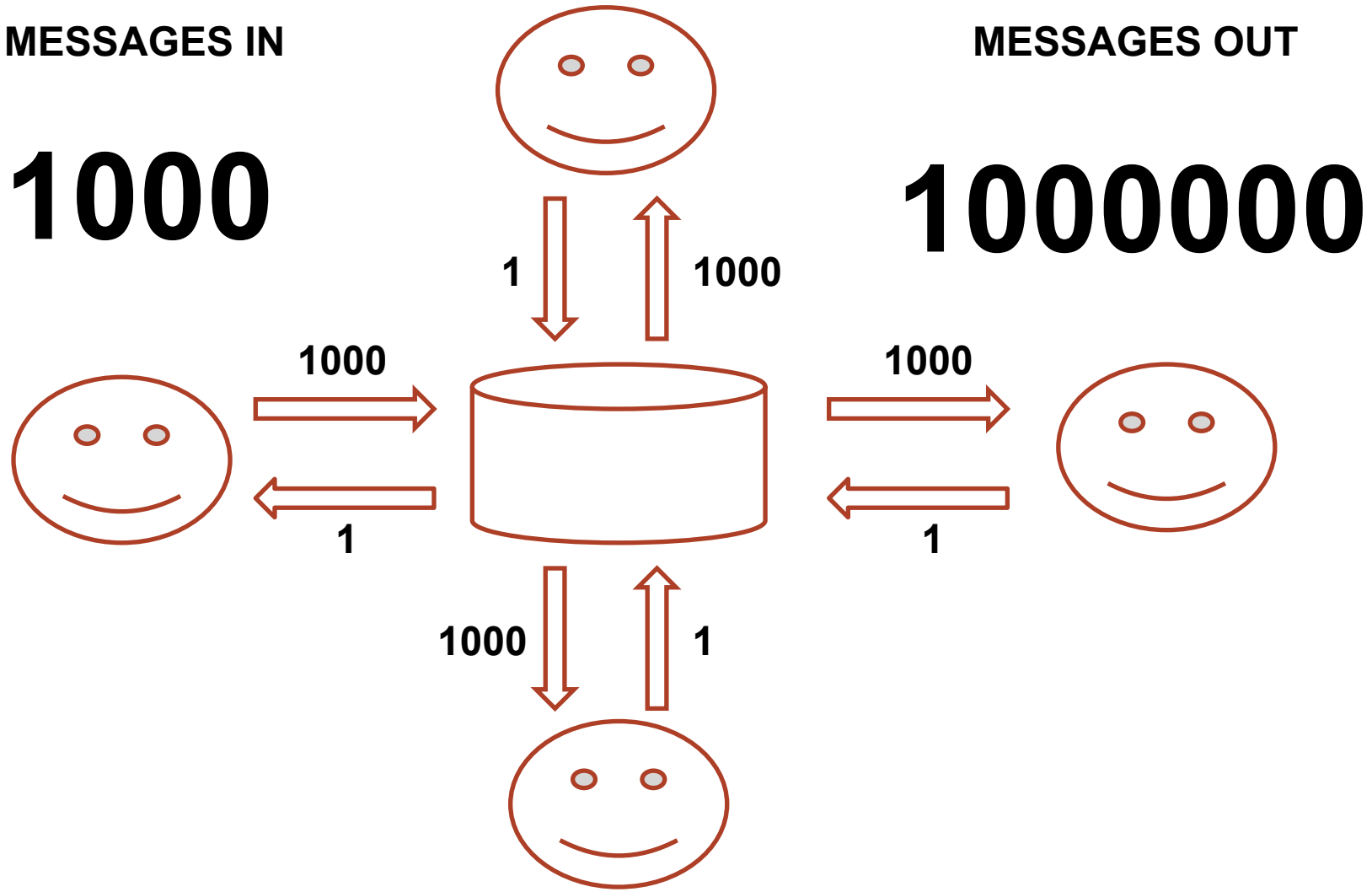
Why does game server not scale



Why does game server not scale

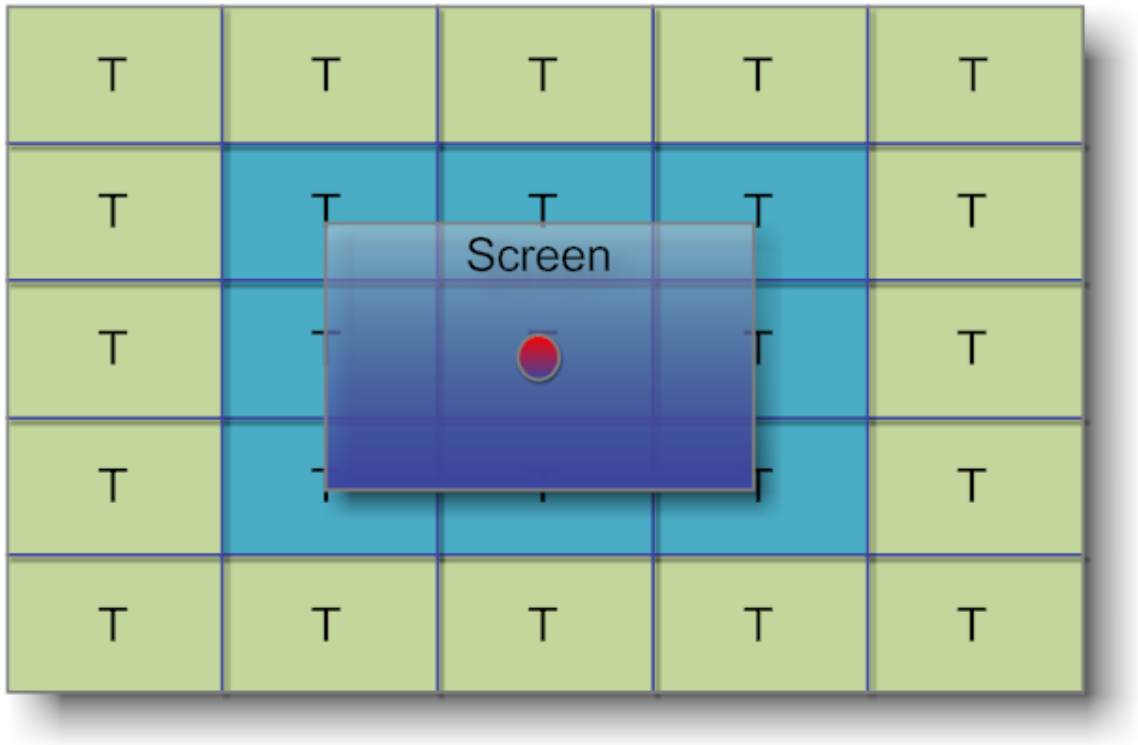


Why does game server not scale



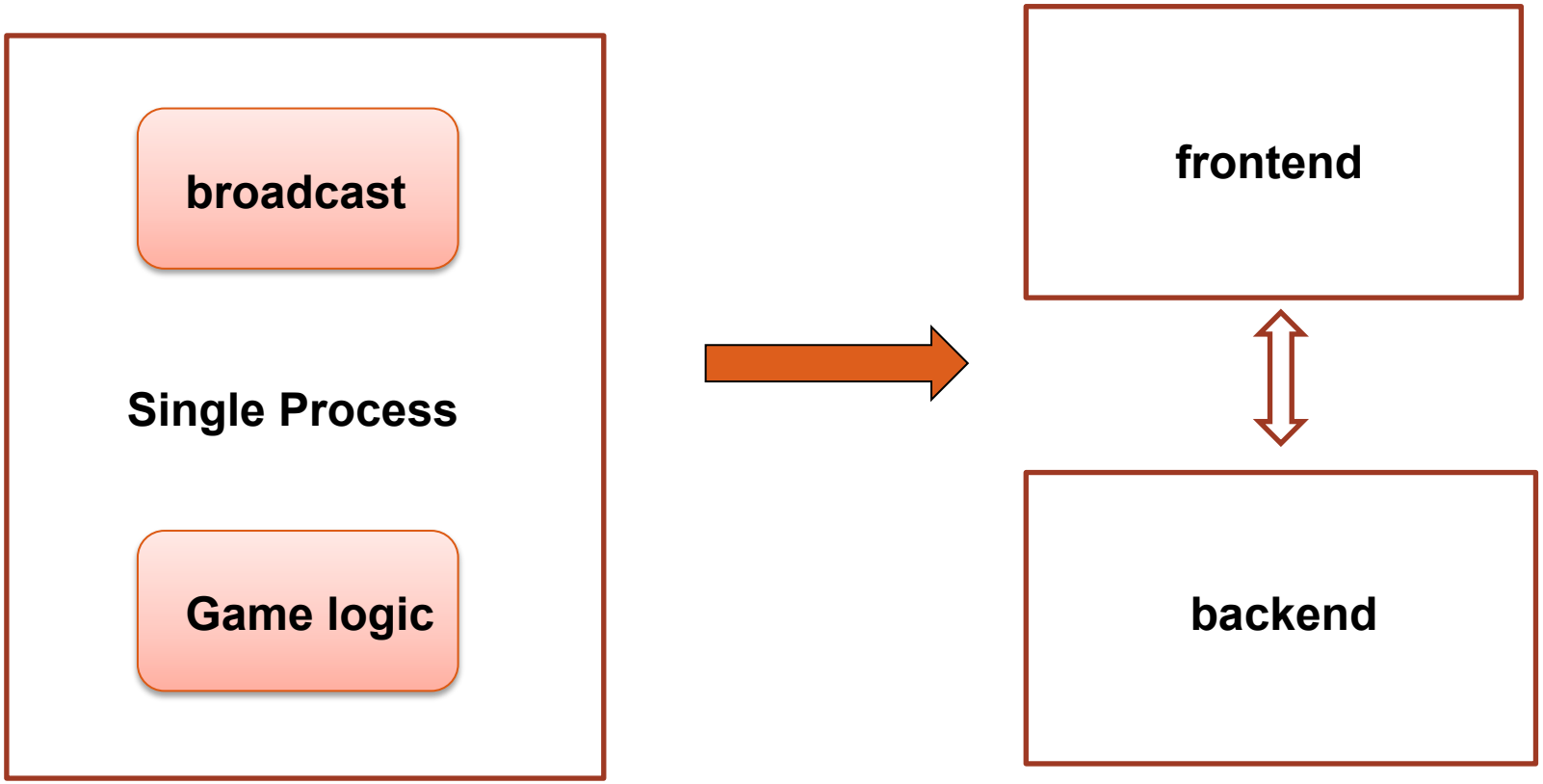
How to solve --- broadcast

- AOI --- area of interested module: pomelo-aoi



How to solve

- Split process, seperate load , frontend is stateless



Why does game server not scale

- Tick
  - setInterval(tick, 100)

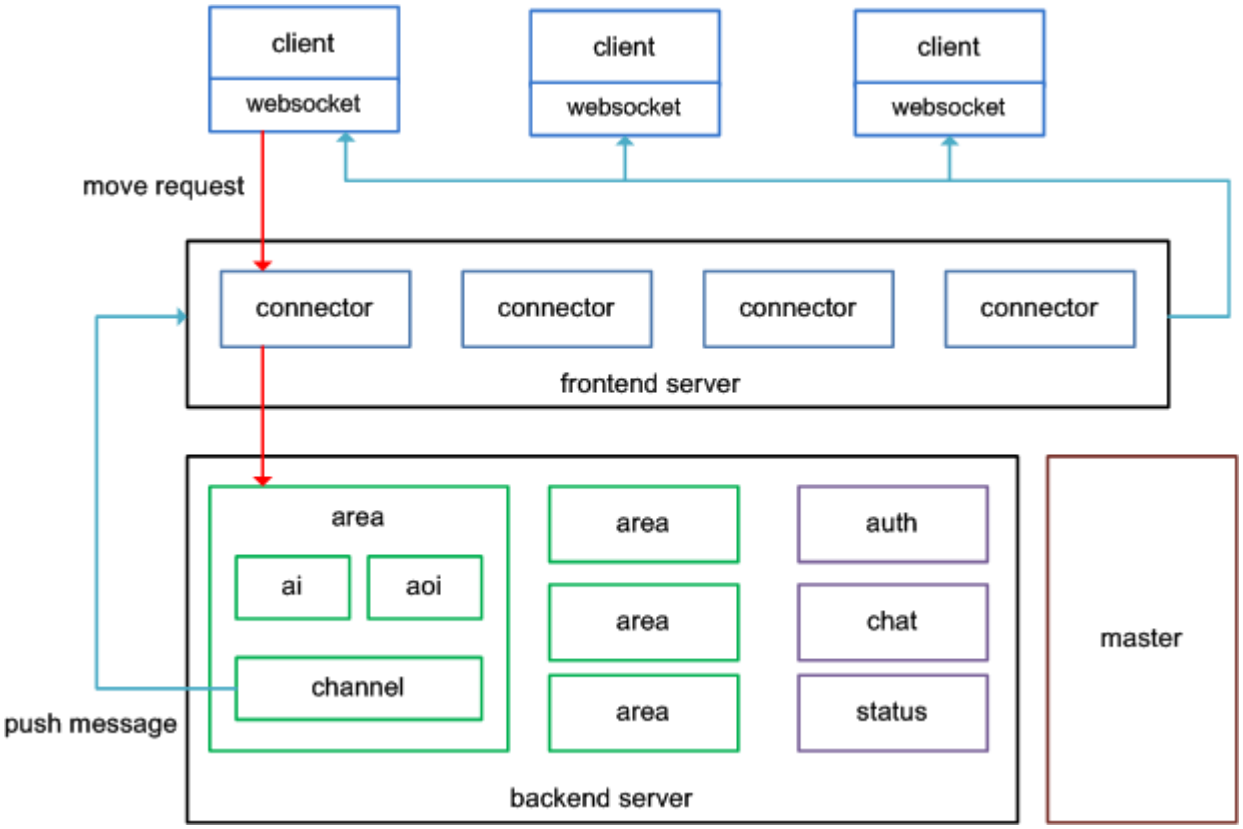
- What does every tick do?
  - Update every entity in the scene(disappear， move, revive)
  - Refresh mob
  - Driving ai logic(monster, player)

Tick must be far less than 100ms

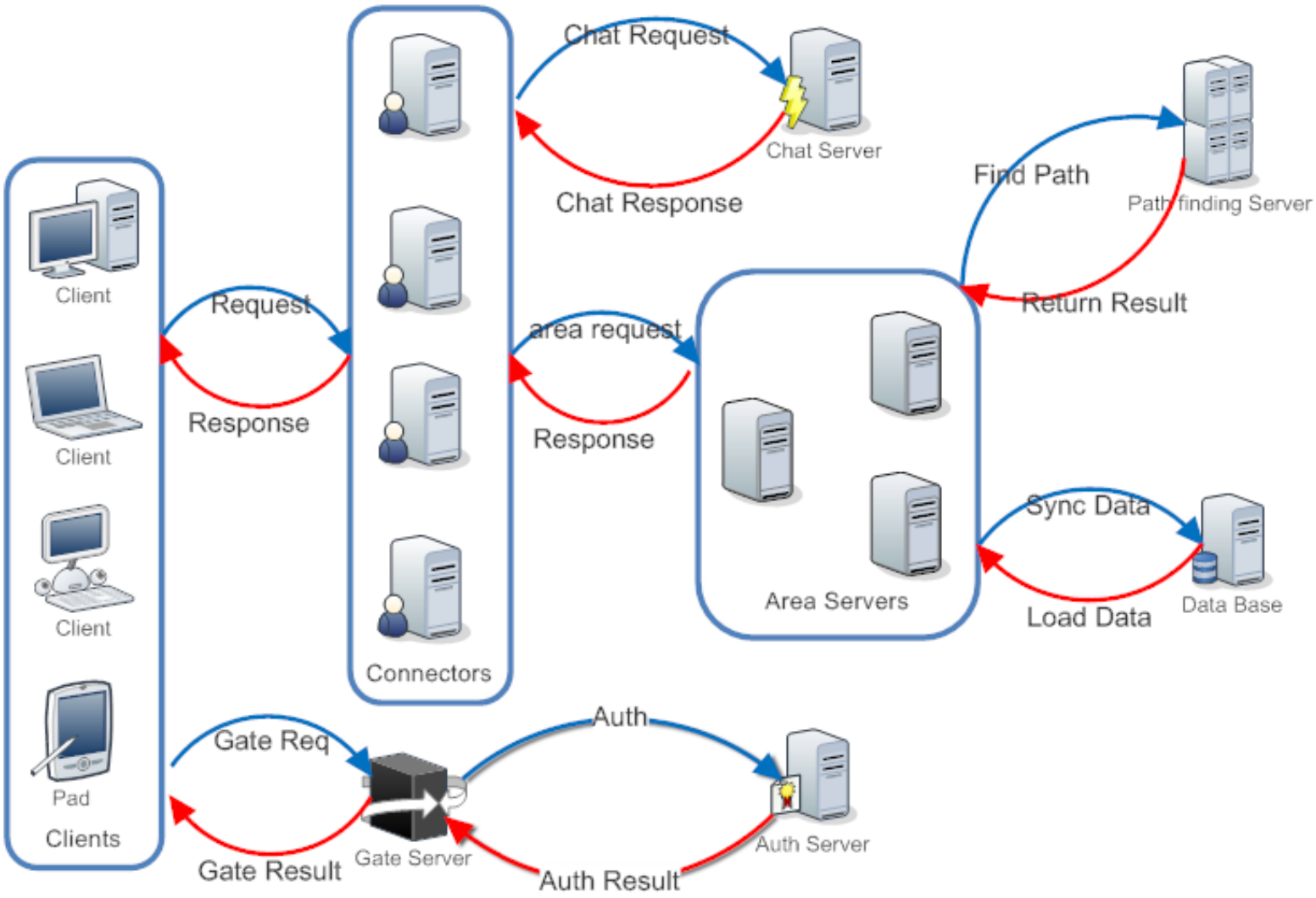
Problem of tick

- The entity number should be limited
- Pay attention to update algorithm: AI etc.
- GC, full gc should never happen
  - V8 is good at GC when memory is under 500M
  - Memory must be limited
  - Try to divide process
- Multi-thread may have some logic problem
  - node.js is single thread

At last--- runtime architecture



Runtime architecture--lordofpomelo



Problem of runtime architecture?

- How many codes for this complicated architecture?
- A lot of servers and server types, how to manage?
- The server side rpc is complicated, how to simplify?
- A lot of processes
  - How many servers do we need?
  - How to spot the problem on multiple servers?
  - Is it too heavy, not efficient?

With pomelo

- Achieve this architecture---almost zero Code
- Server types and servers extention --- simple
- Servers invocation --- Simple， zero config, no stub
- A lot of processes
  - One machine, small amount of resources
  - Single console， quick spot problem, no different to single process
  - Lightweight, extremely quick to start up

Scalability and node.js

Node.js shine

- A lot of network I/O, broadcast
- Multi-process, single thread
- Lightweight

Category

- Scalability of Game server
- Extensible game server framework
- Performance

Framework --- Extensibility

- Difference between framework and project
  - Base architecture over function
  - Extensible： config(DSL)， extention points
  - Everything is replacable： underlying protocal, router, application component, service, admin console
  - Modularize---reusable module
  - Pomelo specific---servers management

Category --- extensibility

- Server abstraction
- App abstraction
- App extention point
- Modularize

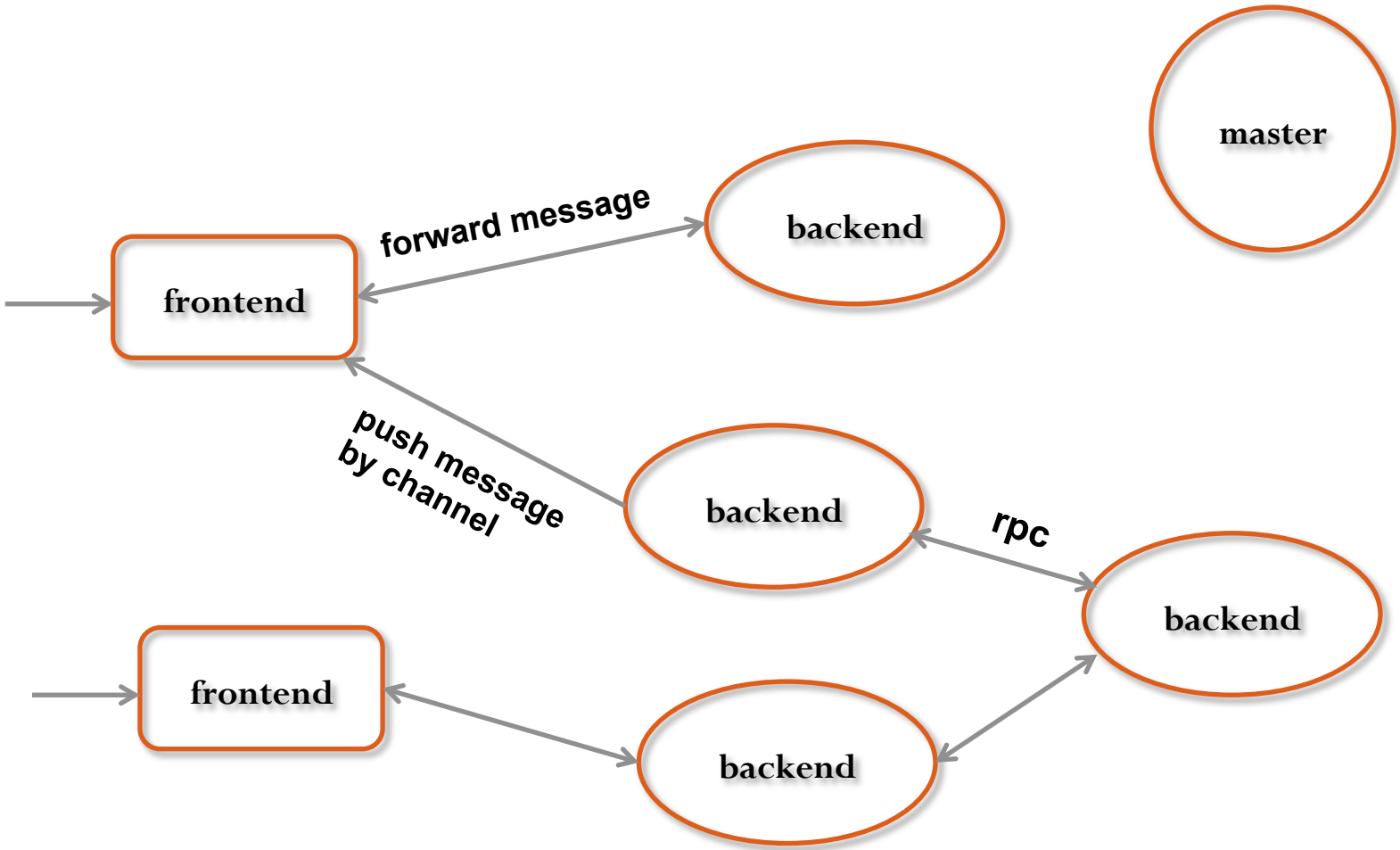
Abstract of Servers

- Pomelo --- distributed(multi-process) app architecture

  why?

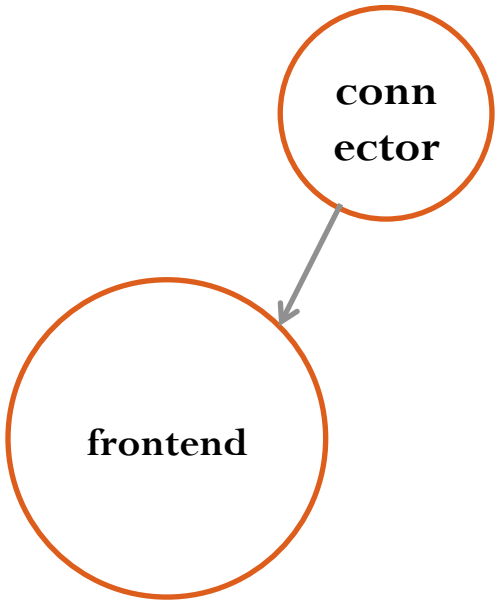
- State
  - Web app---stateless， nginx or apache handle processes
- App servers interaction
  - Web app has not interaction
- Before node.js, too heavy for multiple processes

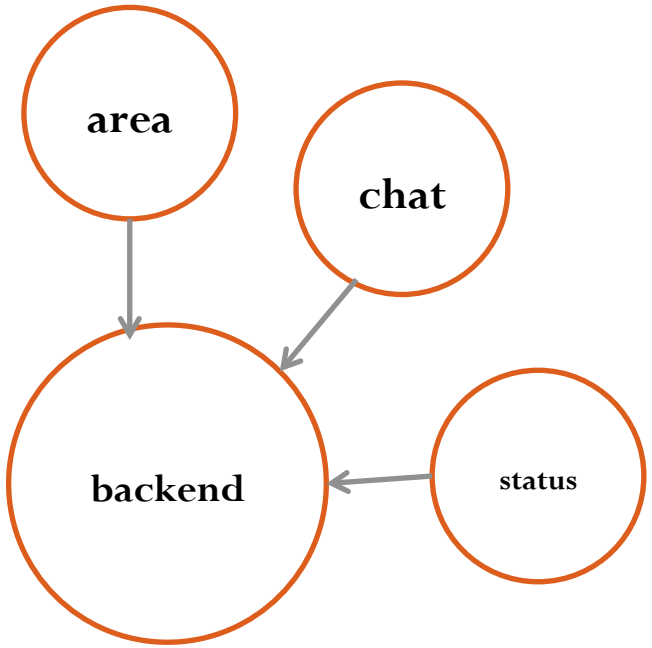
Abstract of servers



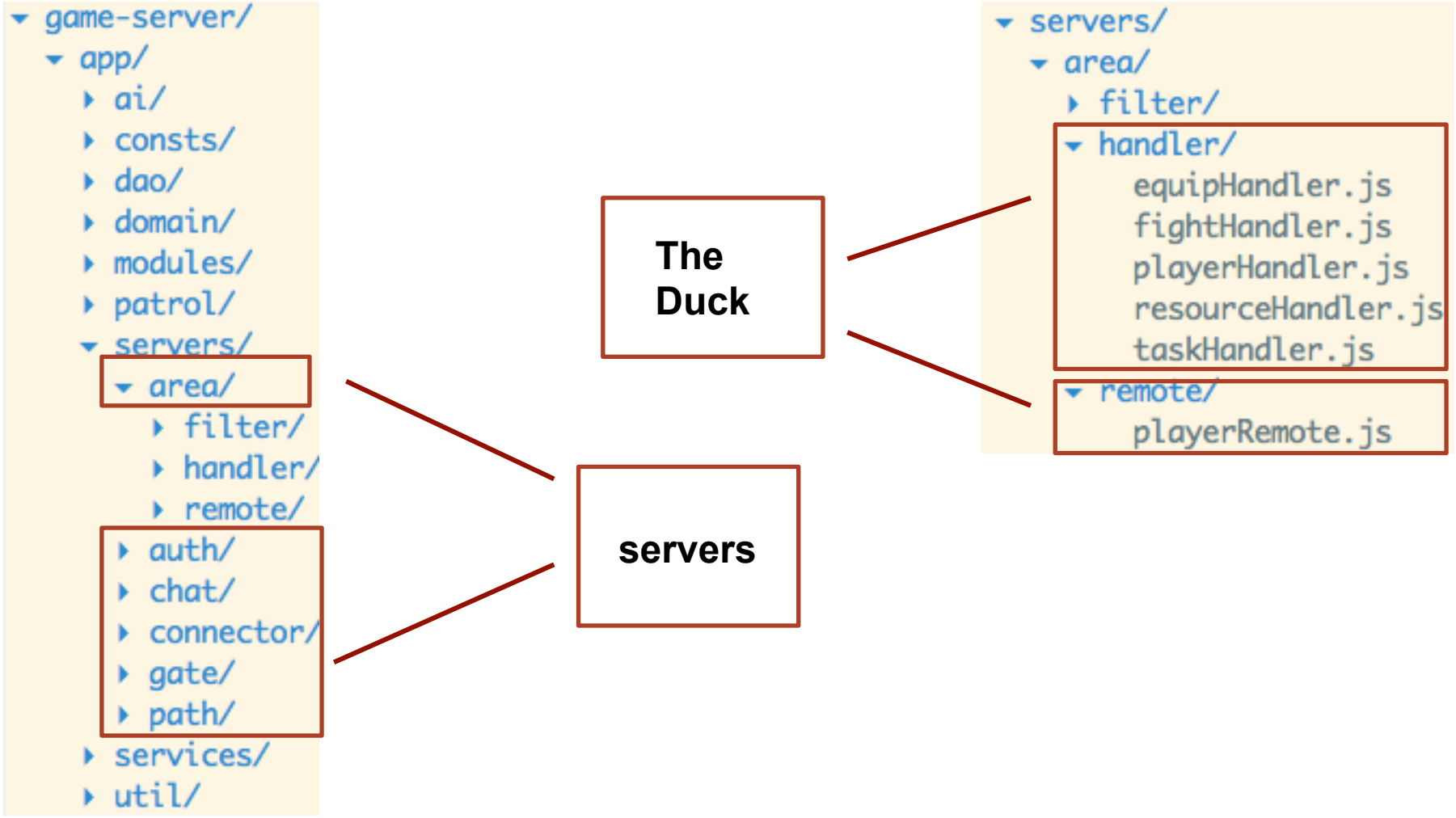
Abstract of servers

- Duck type





Abstract of servers



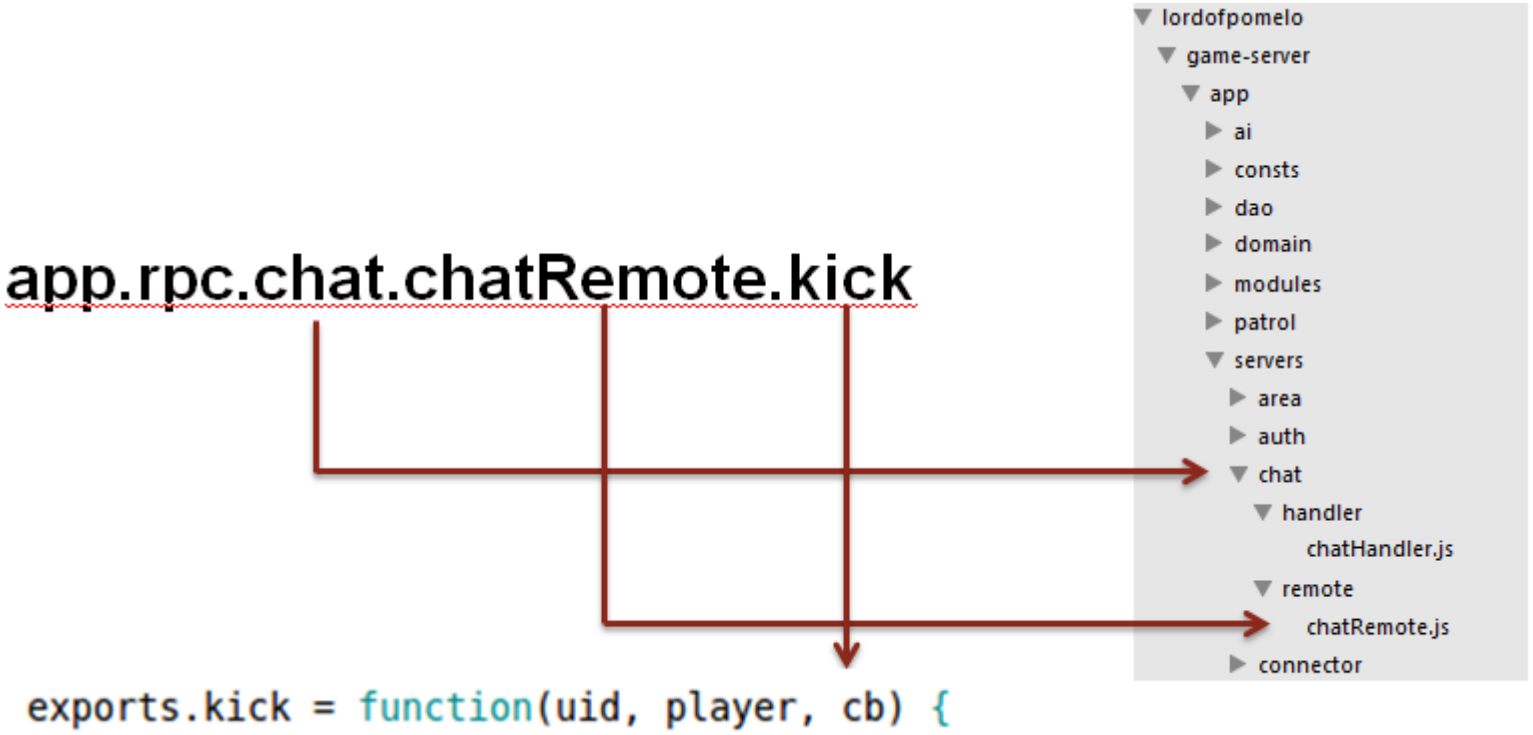
Abstract of servers

- Easy to extend

```
{
  "development":{
    "connector": [
      {"id": "connector-server-1", "host": "127.0.0.1", "port": 3150, "wsPort":3010},
      {"id": "connector-server-2", "host": "127.0.0.1", "port": 3151, "wsPort":3011}
    ],
    "area": [
      {"id": "area-server-1", "host": "127.0.0.1", "port": 3250, "area": 1},
      {"id": "area-server-2", "host": "127.0.0.1", "port": 3251, "area": 2},
      {"id": "area-server-3", "host": "127.0.0.1", "port": 3252, "area": 3}
    ],
    "chat":[
      {"id":"chat-server-1","host":"127.0.0.1","port":3450}
    ]
   }
}
```

Convention over configuration

- rpc --- based on server abstract



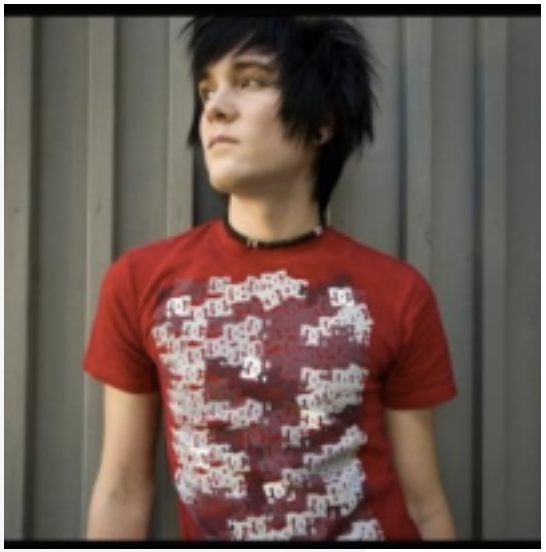
Abstract of Application

- We need an expressive DSL

- Flexible
- Support multi-servers config
- Support a lot of extention points

Json and XML does not fit our needs.

Application DSL



```
var app = pomelo.createApp();
app.set('name', 'chatofpomelo');
app.defaultConfiguration();

// app configure
app.configure('production|development', function() {
        // route configures
        app.route('chat', routeUtil.chat);
        app.route('connector', routeUtil.connector);
        // remote configures
        app.set('remoteConfig', {
                cacheMsg: true,
                interval: 30
        });
        // filter configures
        app.filter(pomelo.filters.timeout());
       // mysql configures
        app.loadConfig('mysql', app.get('dirname') + '/config/mysql.json');
});

// start app
app.start();
```

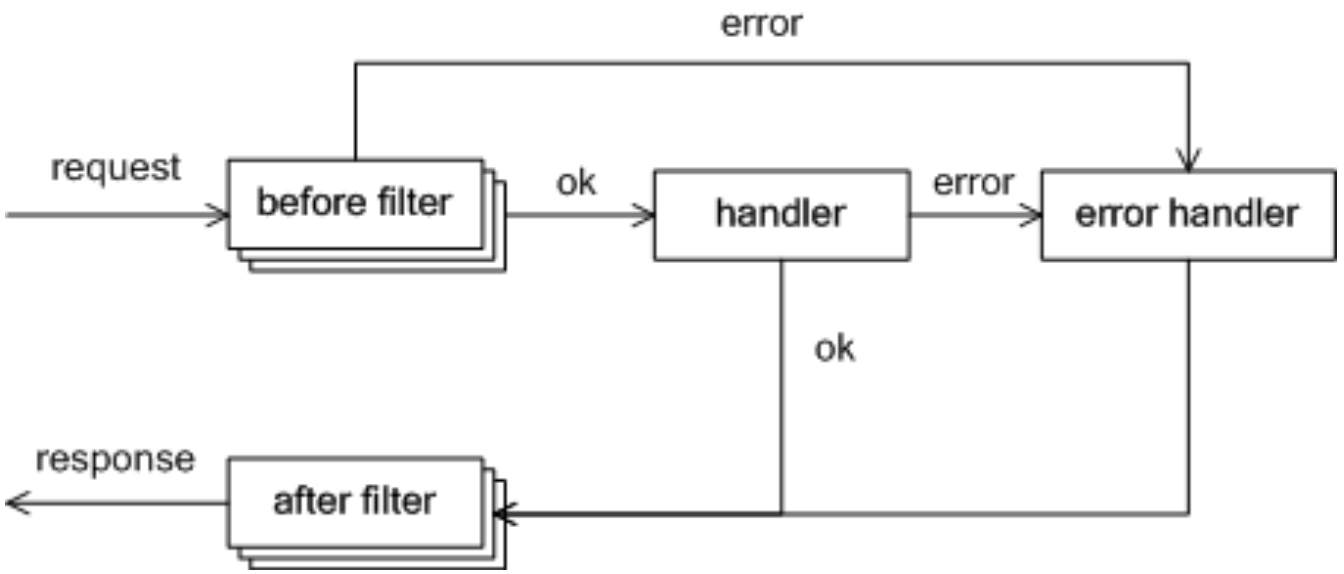
Application DSL --- configure

Multiple server config:

```
app.configure('production|development', function() {
});

app.configure('production|development', 'chat', function() {
});

app.configure('production', 'connector|area|auth', function(){
});
```

Filter

app.filter(pomelo.filters.timout());



Filter

```
Filter.prototype.before = function(msg, session, next) {
  session.__startTime__ = Date.now();
  next();
};

Filter.prototype.after = function(err, msg, session, resp, next) {
  var start = session.__startTime__;
  if(typeof start === 'number') {
    var timeUsed = Date.now() - start;
    var log = {
      route : msg.__route__,
      args : msg,
      time : utils.format(new Date(start)),
      timeUsed : timeUsed
    };
    con_logger.info(JSON.stringify(log));
  }
  next(err,msg);
};
```

Router

- Route to specific server

  based on session state, dynamic

- app.route('chat', routeUtil.chat);

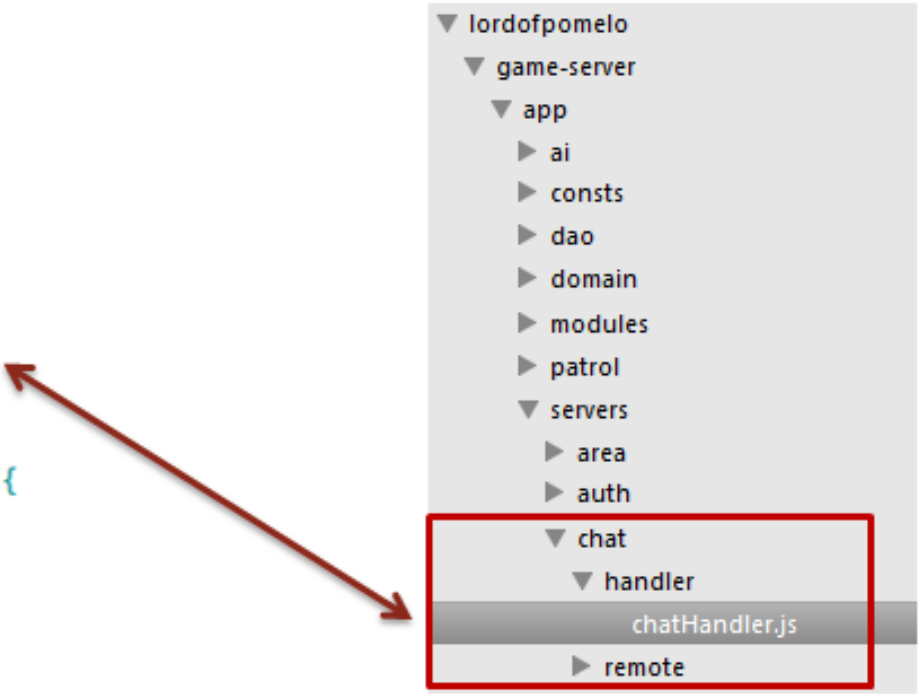
```
exp.chat = function(session, msg, app, cb) {
    var chatServers = app.getServersByType('chat');

    if (!chatServers || chatServers.length === 0) {
      cb(new Error('can not find chat servers.'));
    return;
    }

    var res = dispatcher.dispatch(session.rid, chatServers);

    cb(null, res.id);
};
```

Extensibility– request and transparent route

- Request and transparent route

```
pomelo.request('chat.chatHandler.send', {
  msg: 'hello'}, function(response) {
  var result = response.data;
  console.log('result:', + result);
});
```

```
  handler.send = function(req, session, next) {
    channelService.pushMessageByUids({
      route: route, msg: msg
    });
    next(null, {
      code: OK
    });
  };
```

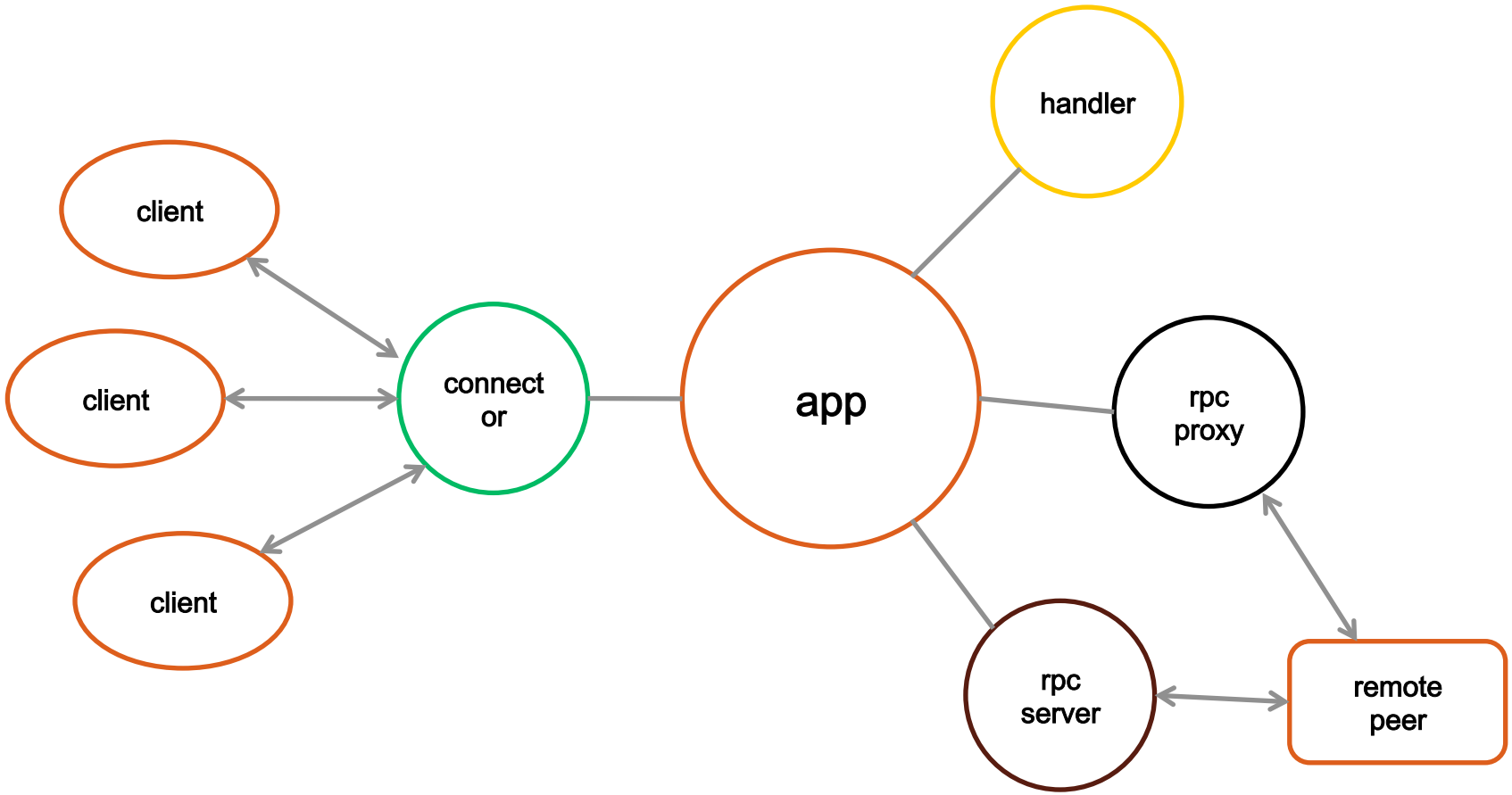


Extensibility --- Component

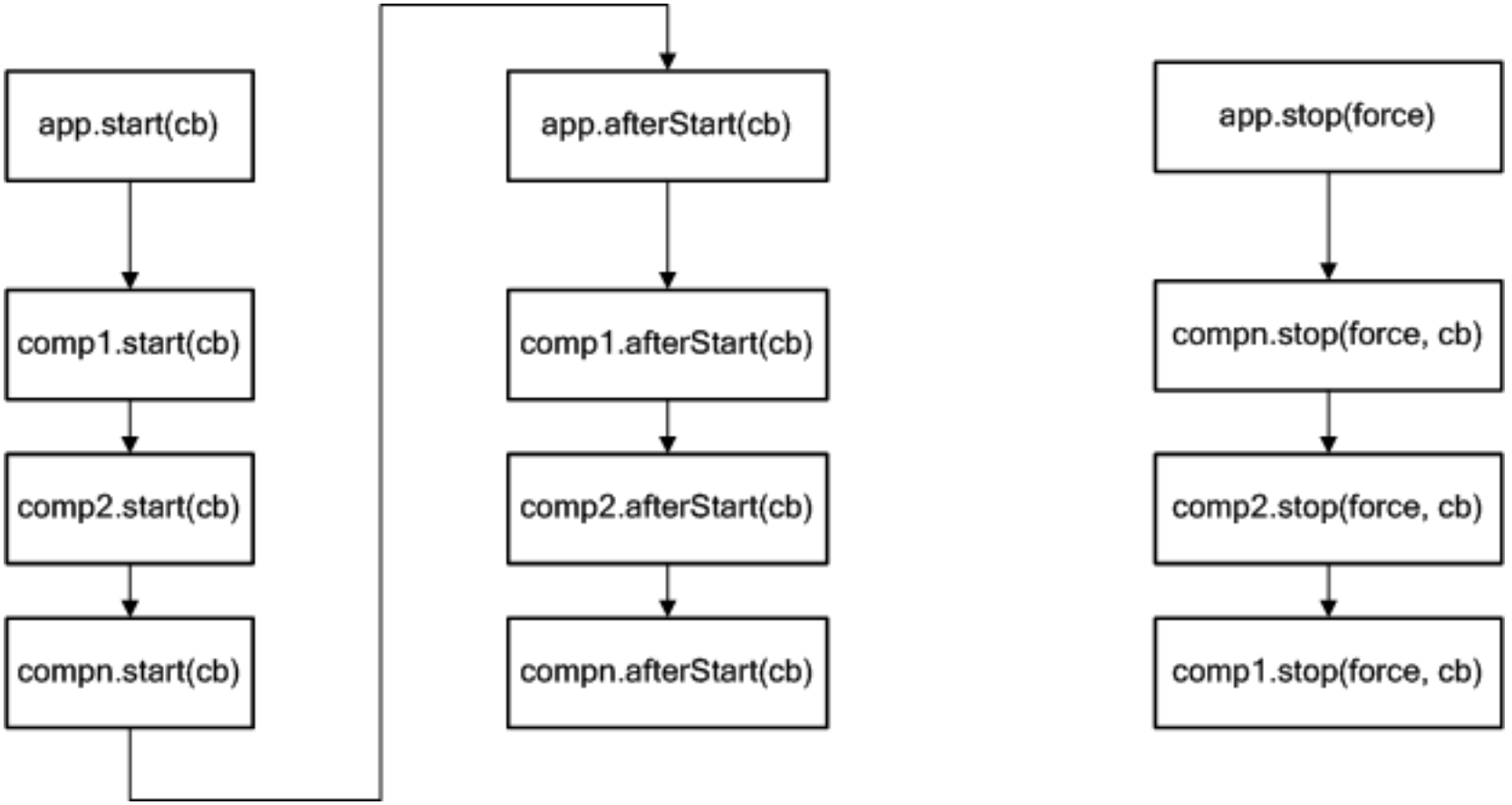
```
app.load(component, options};


app.configure('production', 'area', function() {
        app.load(pomelo.sync, {
                path:__dirname,
                dbclient:dbclient'
        });
});
```

Extensibility--- App component

Pomelo is a collection of components

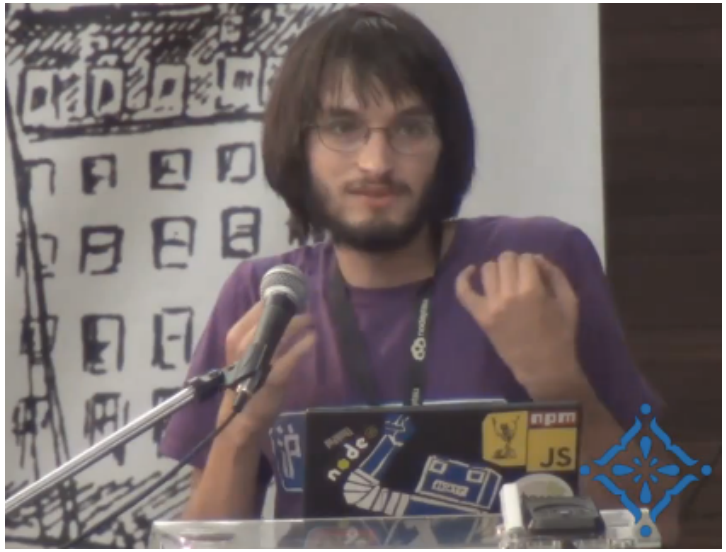


Component

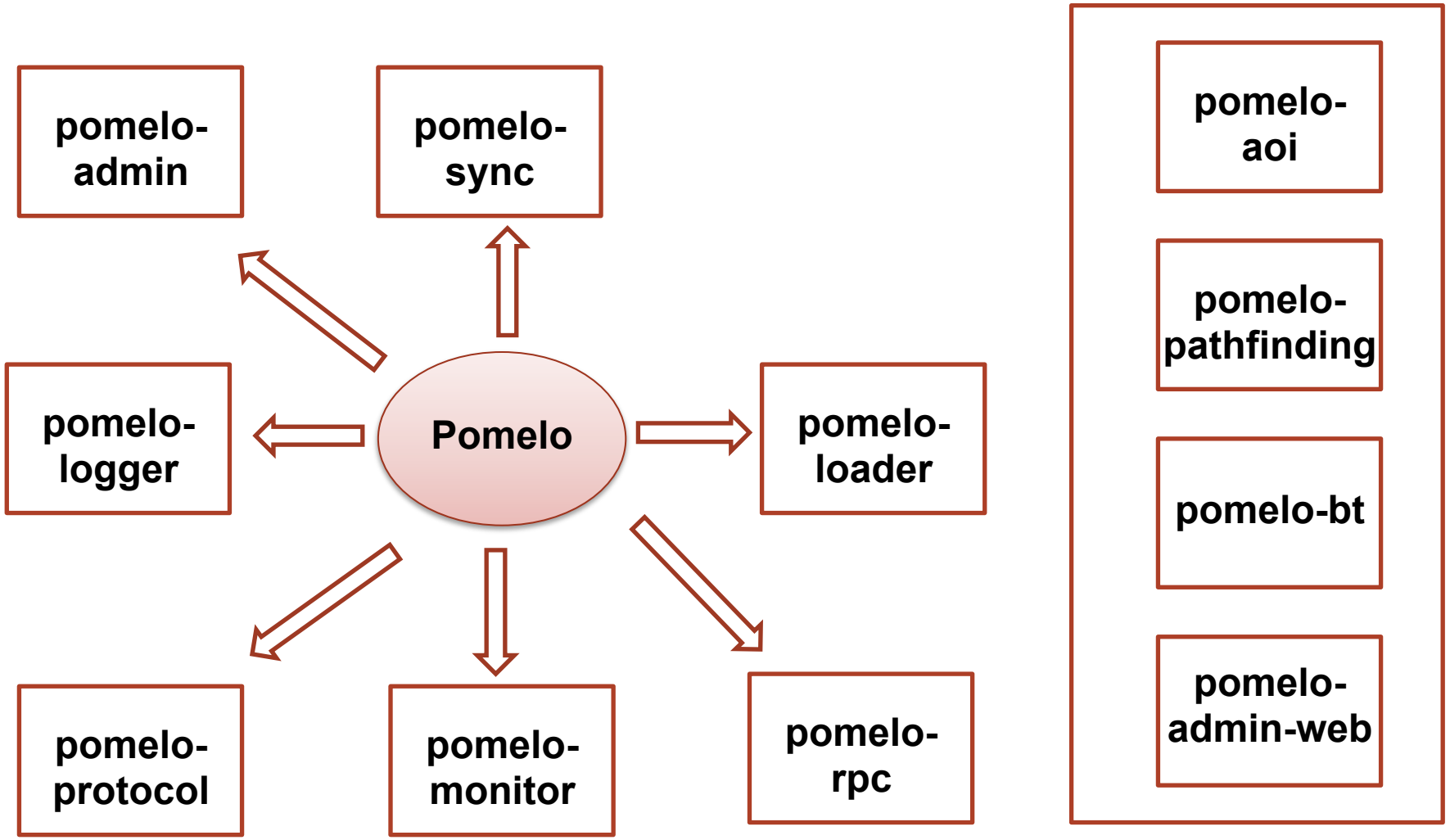


Modularize---npm module based design

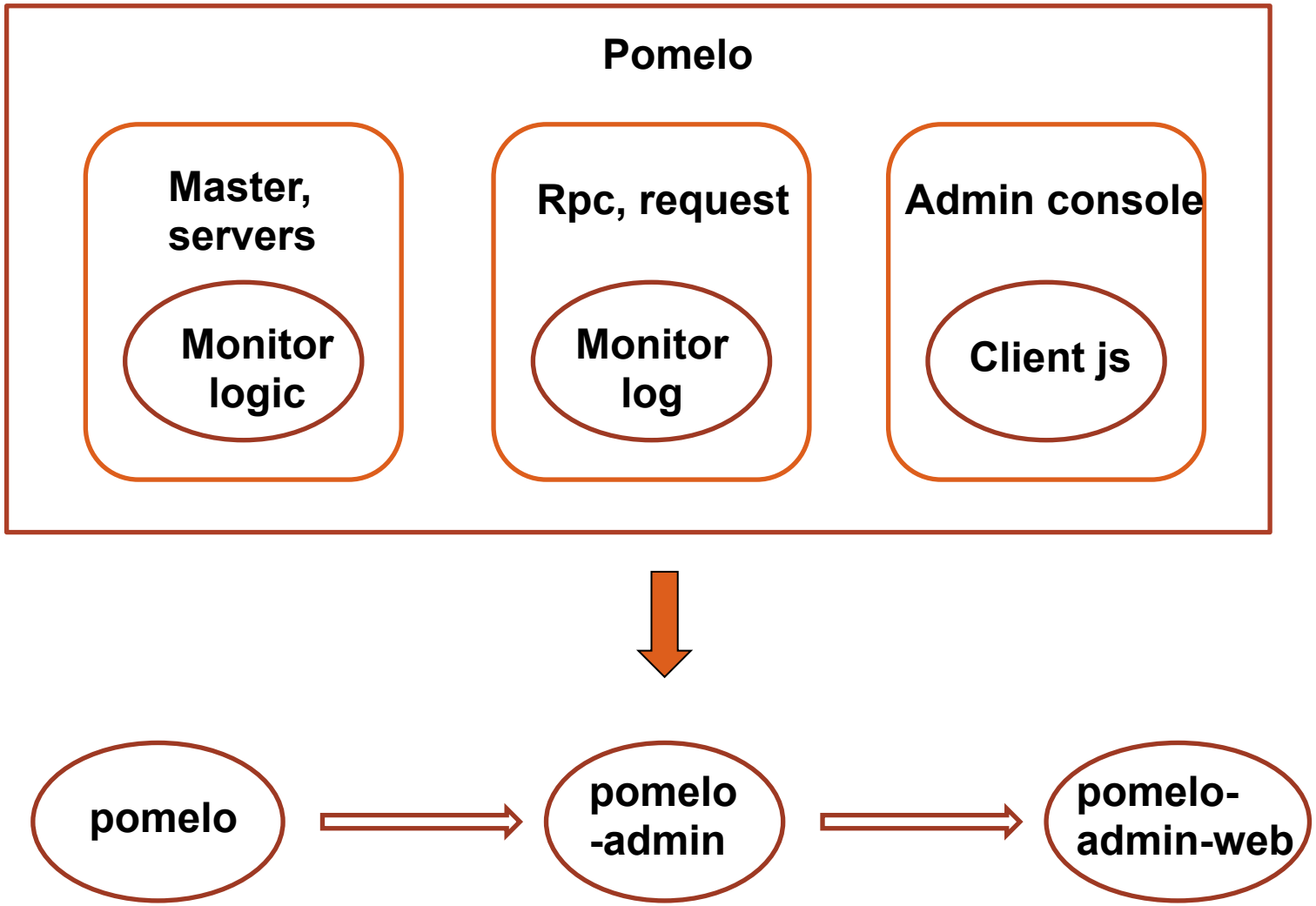
- James Halliday(substack) --- linux philosophy



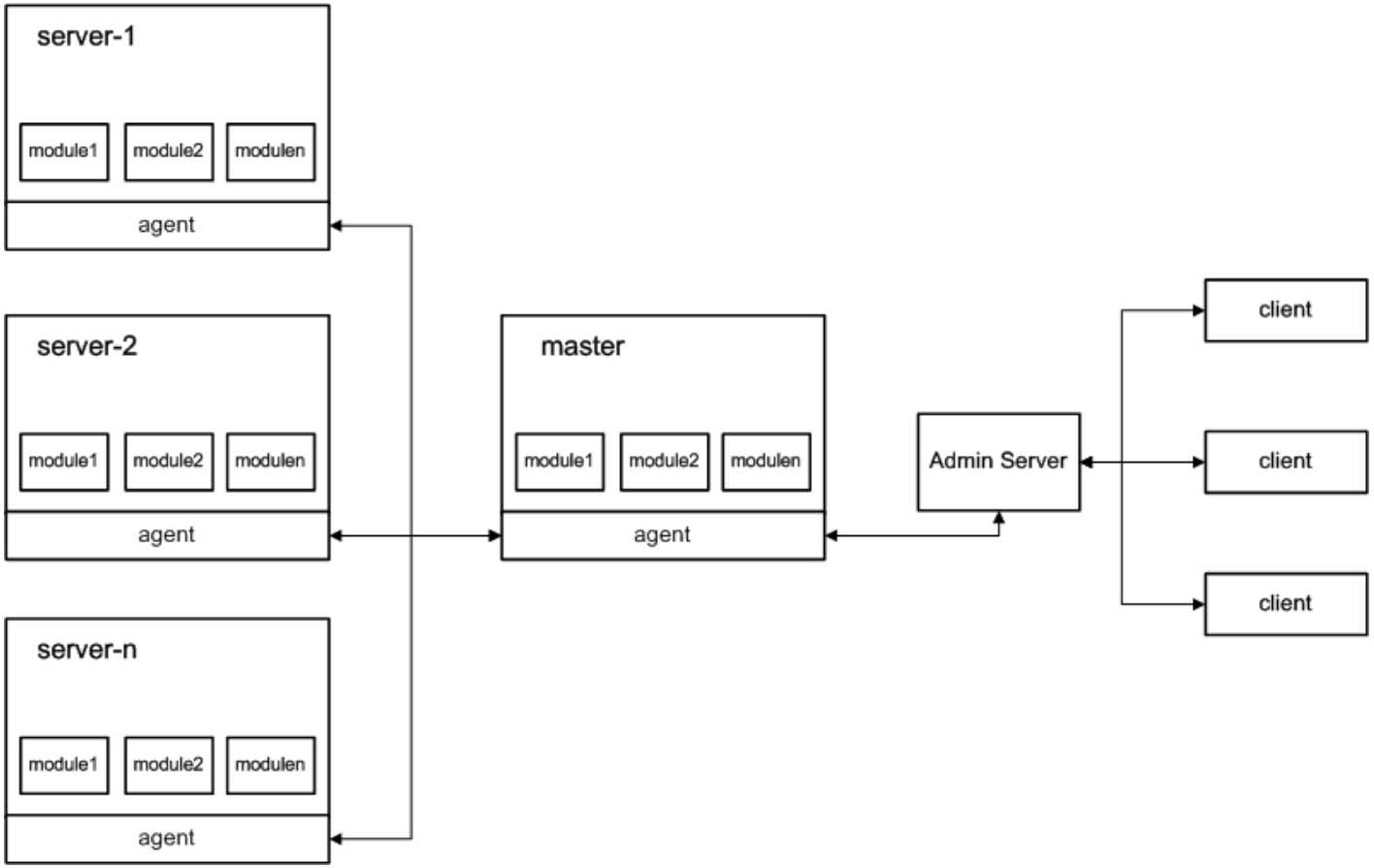
Modularize---npm module based design



Admin console



Admin console extensibility



Extensibility ---Node.js shine again

- Strong DSL ability
- Dynamic， easy for COC
- Module organization， npm， all components are loosely coupled

Category

- Scalabable game server architecture
- Extensible game server framework
- Performance

Performance --- overview

- The indicator
  - The max online users
  - Response time/throughput
  - Single area or game?

- The variation
  - Game logic: round or realtime, room or infinite
  - Map size, character density
  - Balance of areas
  - Test parameters: Think time, test action

Performance --- target

- Area online users
  - next-gen: • Proven to support
  - Socket.io: 25,000 concur

SEND	MEM	CPU
1W	250~	20-30
2W	316~	40-75
2W3	366~	80-90
2W5	428~	90-100

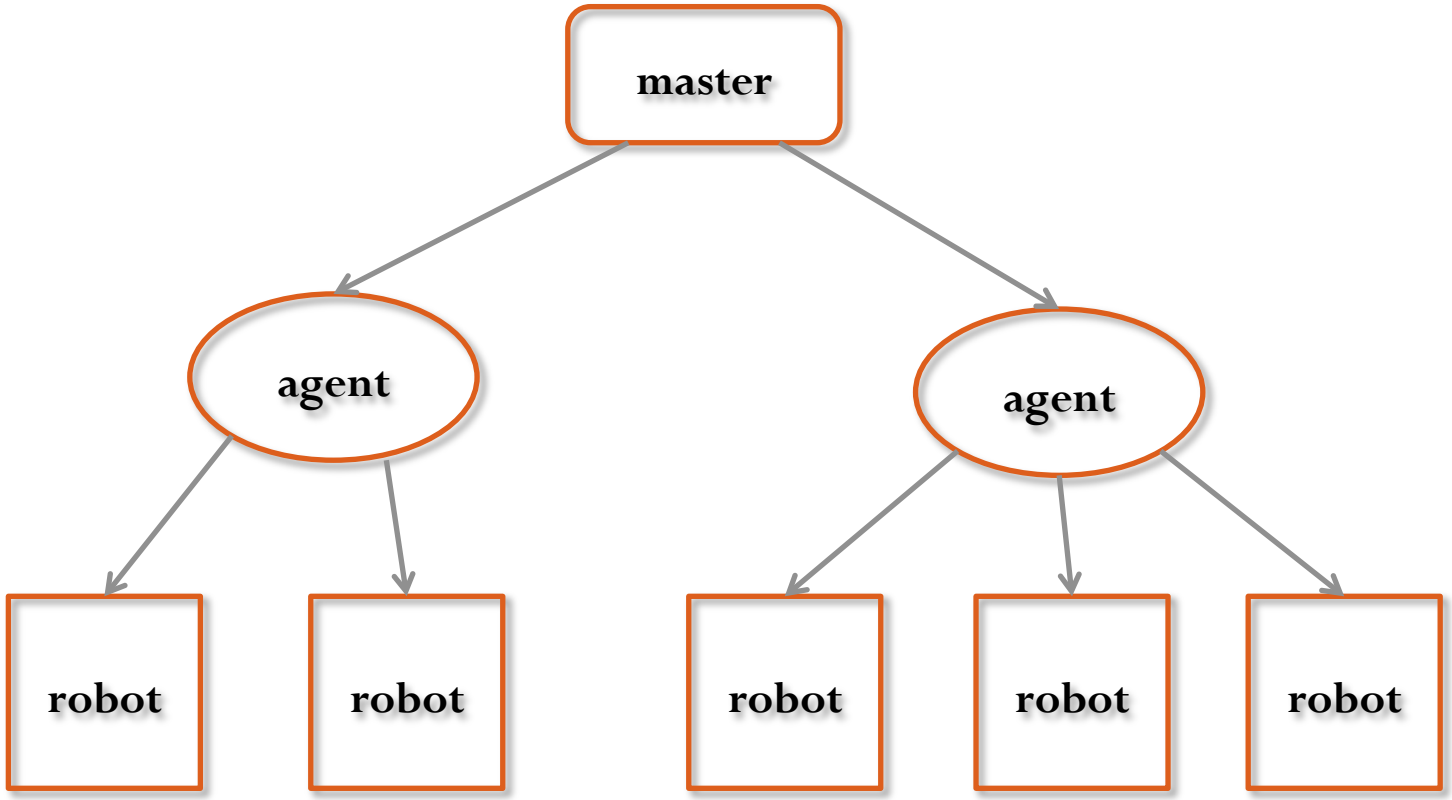
EMIT	MEM	CPU
2W	319~371	48-90
2W3	390~423	95-100

Broadcast	MEM	CPU
300	170~	75-90
400	200`	95-100

- But in the real world
  - The real online data: maximum 1,000 concurrent users per area, 8,000 concurrent users per group game servers

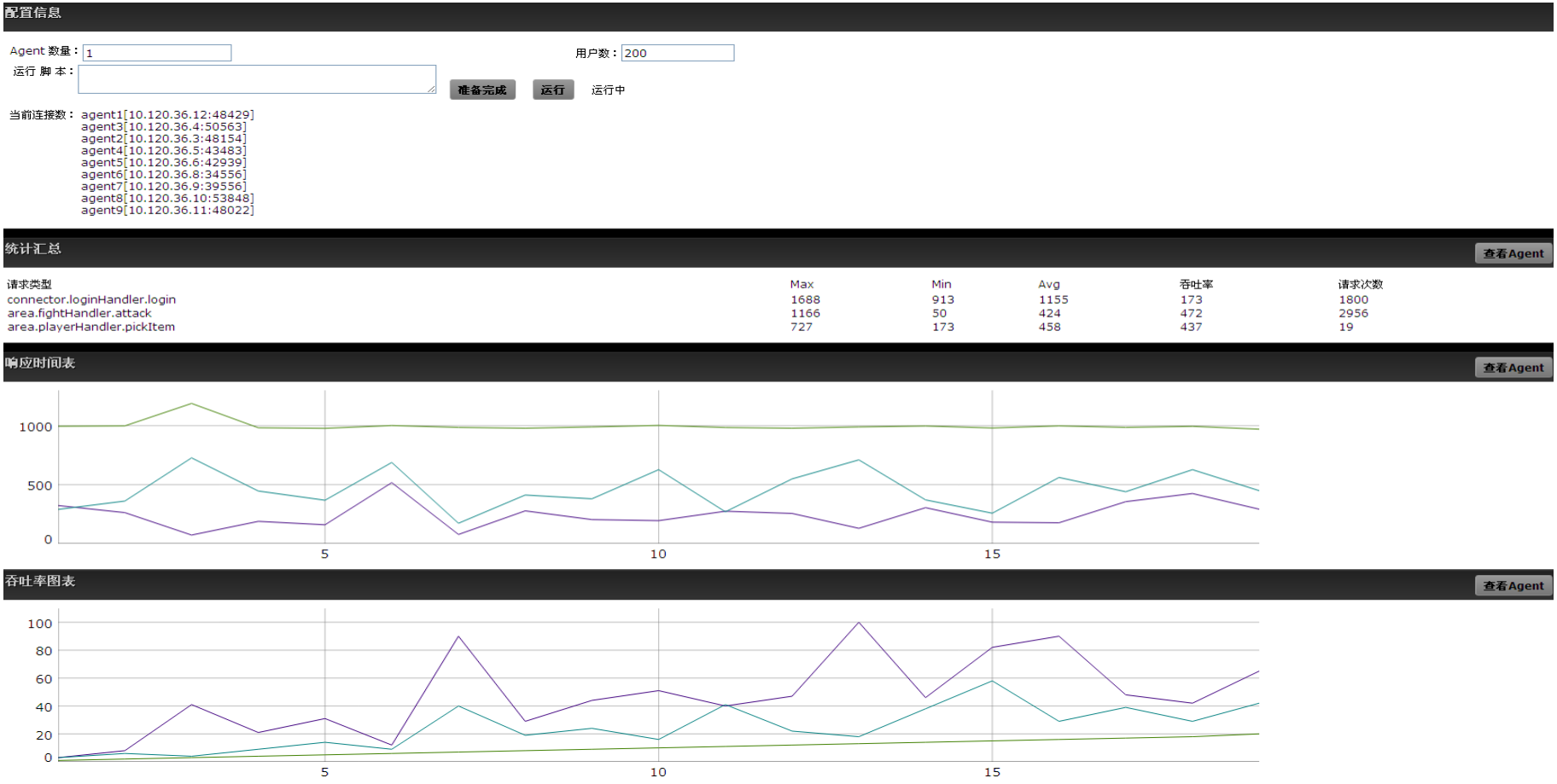
Performance --- tools

Stress testing for websocket--pomelo-robot



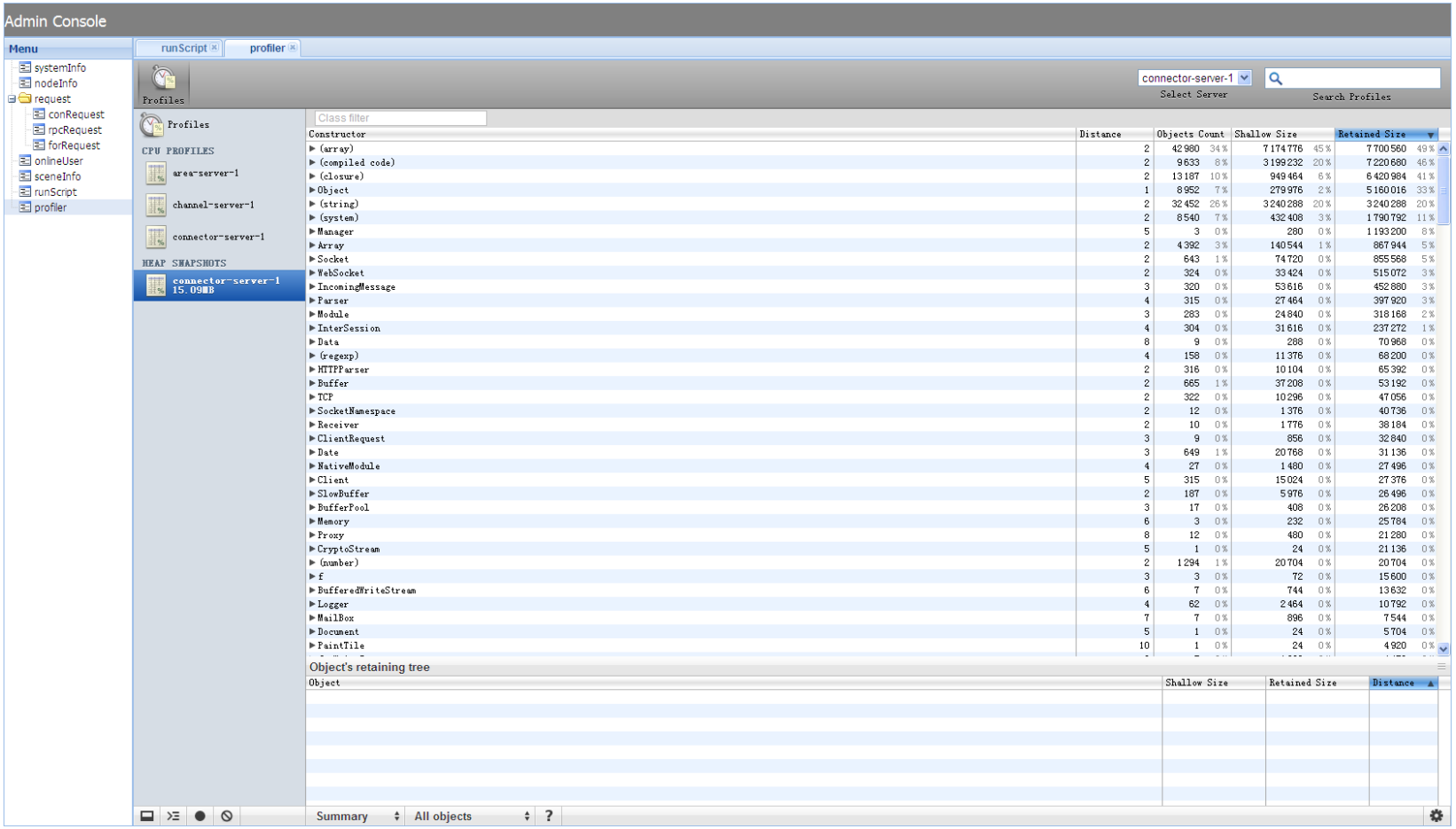
Performance --- tools

Stress test console



Performance --- tools, profiler

Server profiler, choosing servers



Performance --- stress testing

- Stress on single area, increasing step by step
- Real game logic simulation
  - Roam, fight, pick
  - Think time: 2s~4s

Performance --- hardware

- CPU , 24 cores

```
processor       : 23
vendor_id       : GenuineIntel
cpu family      : 6
model           : 44
model name      : Intel(R) Xeon(R) CPU           E5649  @ 2.53GHz
stepping        : 2
cpu MHz         : 2533.675
cache size      : 12288 KB
```

- Mem, 48G

```
MemTotal:       49556132 kB
MemFree:        30924028 kB
Buffers:          717540 kB
Cached:         15988664 kB
SwapCached:           12 kB
```

Performance --- progress

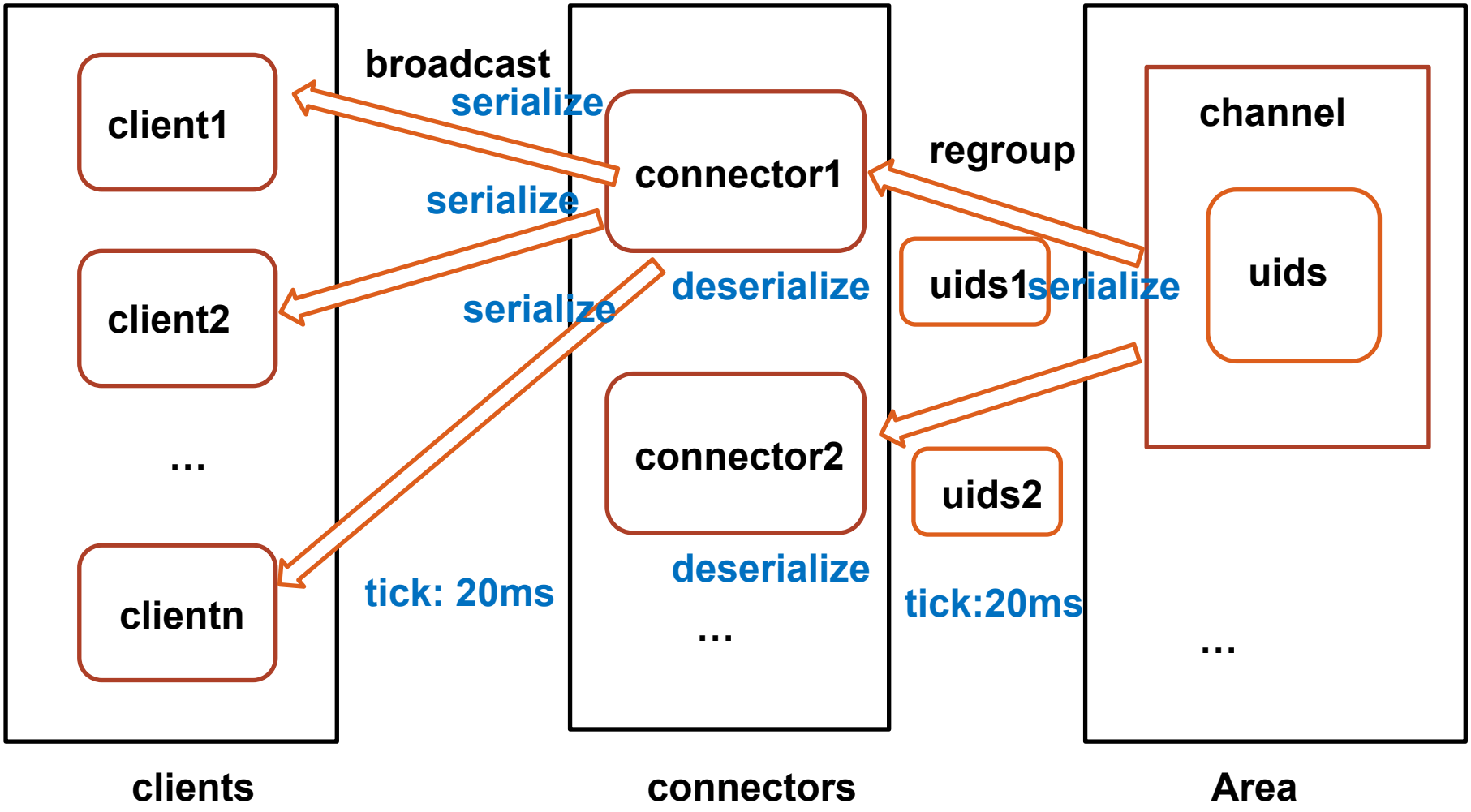
- 6 rounds
- Online users: 200 to 1000…
- Response time: less than 200ms
  - Enter scene: 200ms
  - Other requests: 100ms

Performance --- broadcast

200 online users, connector 100% cpu

Self	Total	Function
25.60%	25.60%	(program)
13.67%	13.93%	▶ Socket._write
12.12%	12.13%	▶ exports.encodePacket
4.89%	11.28%	▶ Buffer
4.50%	4.64%	▶ Buffer.write
4.40%	4.44%	▶ exports.toArray
3.62%	3.62%	(garbage collector)
2.81%	51.83%	▶ exp.flush
2.20%	30.52%	▶ WebSocket.write
2.04%	2.04%	▶ Logger.log
1.76%	49.40%	▶ Socket.emit
1.70%	1.70%	▶ parser.encodePacket

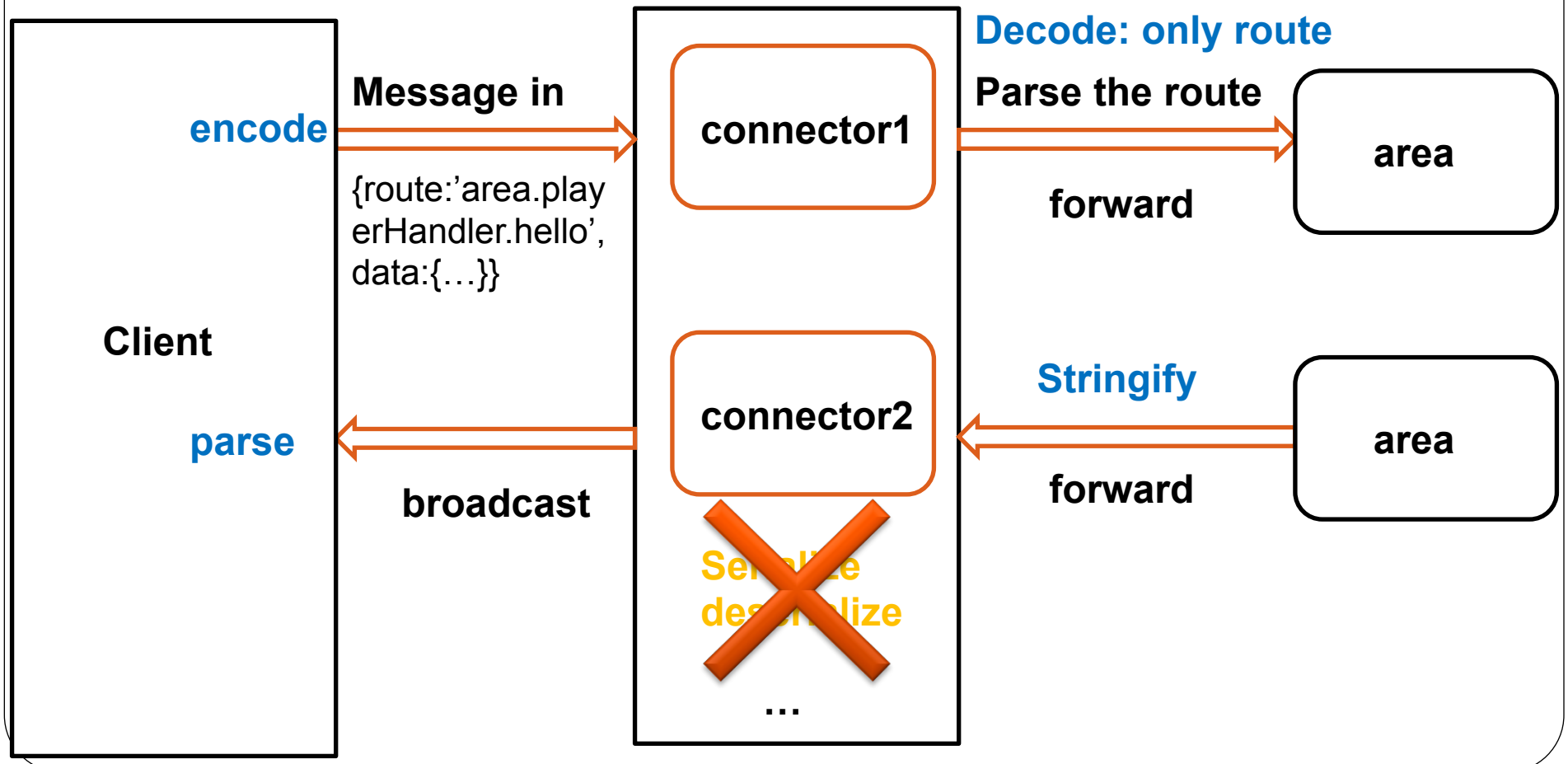
Performance --- channel, where is wrong?



Performance --- connector

- Connector--- the middle man

What do I need data for?



Performance --- the package

- Pomelo-protocal

  Only parse head for route information:

```
-----------------------------------------------
| 4 bytes | 1 byte(route length) | route | body |
-----------------------------------------------
```

\0\0\0\3\34connector.loginHandler.login{“username”:”xcc”,……..}

Performance---CPU

- 场景及出生点
- 寻路问题
- 怪数量及动态寻路
- AOI计算
- 大量的解压包
- dataApi查找换成MAP

Performance---IO

- 网络传送数据路过大
- 数据未批量发送
- 用户断开空转
- 数据同步日志过于频繁

Performance---Memory

- 数据抽取模式（拉推）
- 冗余数据去除
- 内存泄漏及GC

Performance --- the result

- 1600 onlines

配置信息

Agent 数量：2　　用户数：100

运行 脚 本：

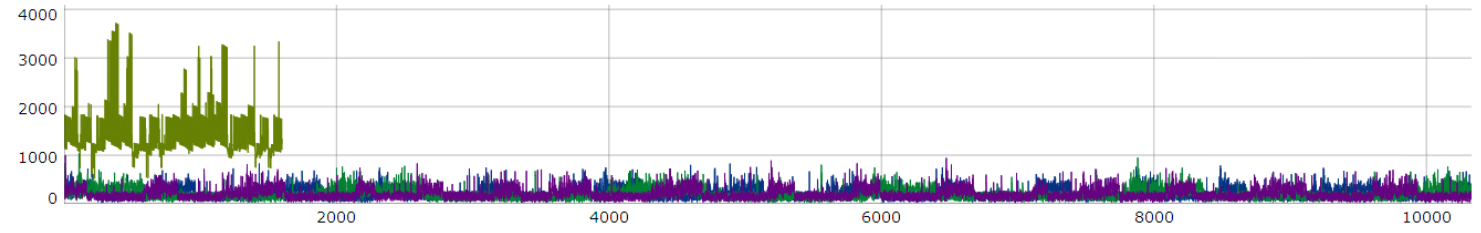
准备完成　运行　运行中

当前连接数：agent5[10.120.36.5:50515]
agent3[10.120.36.4:57031]
agent4[10.120.36.4:57032]
agent6[10.120.36.5:50517]
agent13[10.120.36.10:57001]
agent9[10.120.36.8:40073]
agent16[10.120.36.11:53165]
agent10[10.120.36.8:40074]
agent2[10.120.36.3:51566]
agent1[10.120.36.3:51567]
agent12[10.120.36.9:41907]
agent15[10.120.36.11:53168]
agent7[10.120.36.6:51712]
agent8[10.120.36.6:51713]
agent11[10.120.36.9:41910]
agent14[10.120.36.10:57004]

统计汇总　查看Agen

请求类型	Max	Min	Avg	吞吐率	请求次数
connector.loginHandler.login	3724	519	1410	142	1600
area.fightHandler.attack	832	16	214	935	12491
area.playerHandler.pickItem	1037	17	203	985	31057
area.playerHandler.move	999	10	210	952	10331

响应时间表　查看Agen



吞吐率图表

Performance --- the result

- 1600 onlines , server load

Isn't that amazing? **no**

```
top - 17:17:02 up 124 days, 14:53,  4 users,  load average: 0.18, 0.37, 0.35
Tasks: 334 total,   2 running, 332 sleeping,   0 stopped,   0 zombie
Cpu(s):  4.2%us,  1.1%sy,  0.0%ni, 94.4%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:  49556132k total, 21296868k used, 28259264k free,   781604k buffers
Swap:  2096440k total,       12k used,  2096428k free, 17581444k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
23159 pomelo    20   0  712m 110m 6152 S   35  0.2   2:49.90 node
23142 pomelo    20   0  638m  65m 6140 S   25  0.1   1:00.79 node
23150 pomelo    20   0  697m  89m 6136 S   19  0.2   1:03.42 node
23144 pomelo    20   0  672m  74m 6136 S   17  0.2   1:00.87 node
23147 pomelo    20   0  630m  87m 6132 R   16  0.2   0:58.85 node
23162 pomelo    20   0  674m  71m 5968 S    3  0.1   0:09.18 node
23168 pomelo    20   0  611m  71m 5960 S    2  0.1   0:24.23 node
23153 pomelo    20   0  669m  54m 6136 S    1  0.1   0:02.78 node
  745 monitor   20   0 12.2g  83m  11m S    0  0.2   0:16.67 java
16983 monitor   20   0 12.4g 538m  11m S    0  1.1   6:40.51 java
23137 pomelo    20   0  667m  65m 5988 S    0  0.1   0:03.04 node
```

Performance --- the result

- 800 onlines, fight each other

Performance --- the result

- Server load, fight each other

```
 - 14:37:59 up 125 days, 12:14,  4 users,  load average: 0.63, 0.52, 0.67
ks: 330 total,   4 running, 326 sleeping,   0 stopped,   0 zombie
(s):  6.9%us,  1.2%sy,  0.0%ni, 91.6%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
:  49556132k total, 22283880k used, 27272252k free,   789140k buffers
p:  2096440k total,       12k used,  2096428k free, 18671912k cached
```

ID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
47	pomelo	20	0	697m	90m	6152	R	80	0.2	1:39.26	node
35	pomelo	20	0	697m	89m	6136	R	28	0.2	0:29.28	node
38	pomelo	20	0	701m	92m	6132	S	28	0.2	0:30.90	node
30	pomelo	20	0	694m	87m	6132	S	25	0.2	0:34.36	node
32	pomelo	20	0	701m	92m	6136	R	24	0.2	0:31.83	node
50	pomelo	20	0	671m	66m	5952	S	7	0.1	0:07.32	node
56	pomelo	20	0	606m	66m	5960	S	4	0.1	0:09.39	node
41	pomelo	20	0	597m	34m	5952	S	1	0.1	0:02.45	node
44	pomelo	20	0	658m	25m	5948	S	1	0.1	0:02.59	node

TODO

- Performance
  - Servers rpc， sock.io → tcp
  - Network protocal, json is wasteful
- Fault-tolerant
- Different clients support

網易 NetEase
www.163.com

Sites

- Pomelo home: http://pomelo.netease.com
- Github: https://github.com/NetEase/pomelo
- Demo: http://pomelo.netease.com/lordofpomelo
- Weibo: @pomelonode @圈圈套圈圈

Q&A