NODE JS DESIGN & SCALABLE ARCHITECTURES II

everis thinking lab, date: 25/05/2018

© 2018
ignacio.ariza.victoria@everis.com

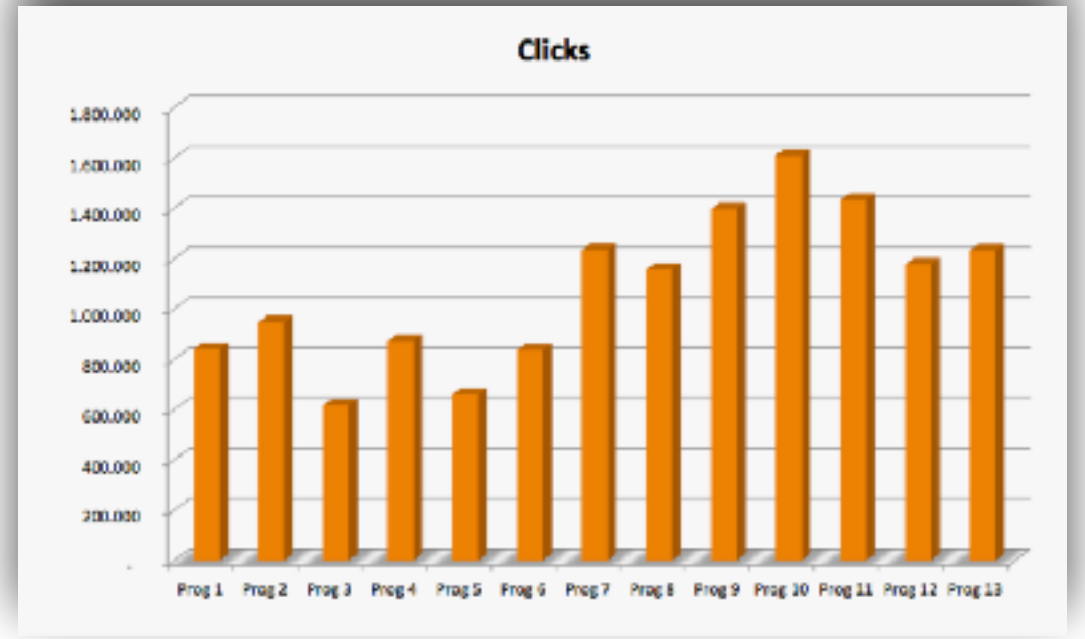
NTT DATA

Index

1 throughputs

1. Success story: MasterChef´s 2016/2017

127.385.192 **user access** today. (2015=54.584.685)
1.700.000-6.293.713 **access / tv-program** (2.5 - 3h)
5.307.716 **access / month**
968.671 **active users (2015+2016+2017)**
582/1480 **rest execution / second**



2 the second screen problem

2.1. The second screen problem

Current situation

Platform J2EE WSDL web service based / php based ,,, other backend tech.

Complication

Connectivity and Scalability to new users / 1000k. Sockets & 20.000k. access.

Stakeholders

All TV channels / audience / personal TV / Retail sectors,,,

Scope

Worldwide service.

Problem Statement

Concurrence && Scalability.

Hypotheses

Current technology / emerging technologies.

2 the second screen problem

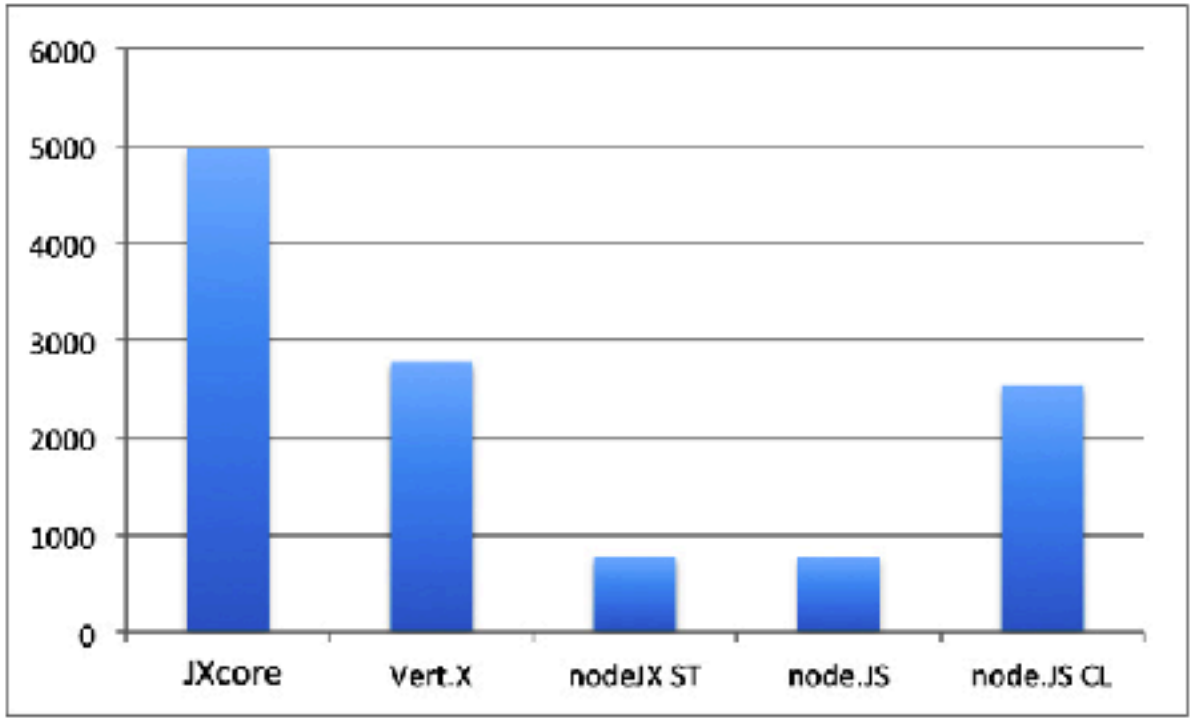
2.2. node js vs other async frameworks

Below chart shows the number of **requests handled by each platform per second at 'hello world' message.**

The performance gain behind JXcore MT is simply due to sharing the http server load across separate threads and the V8 blocks under the same process. As a result, **there is no latency because of the multi process communications.**

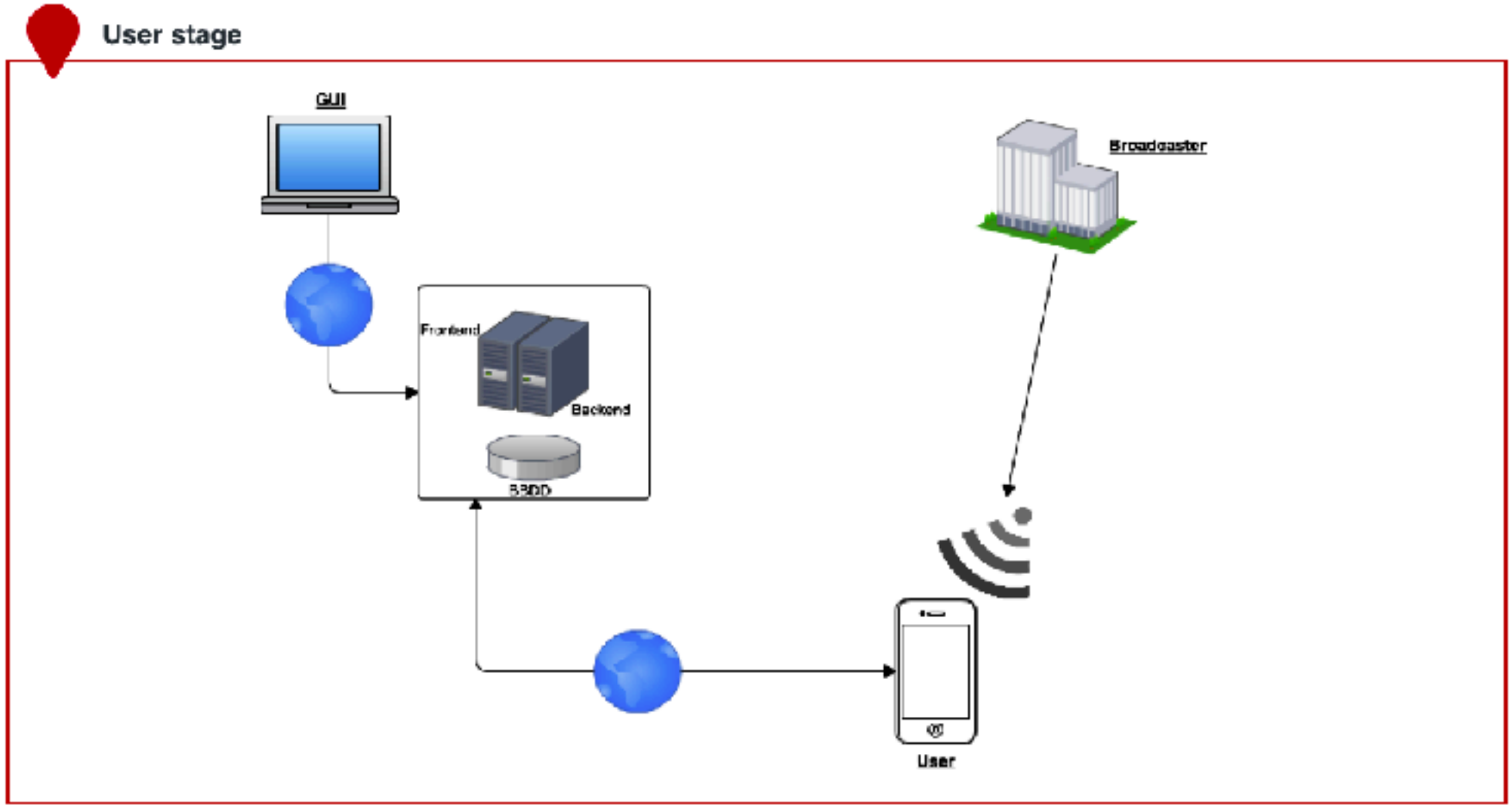
The developer community node JS is currently higher of vert.x.

CLV program -
AWS testing with node js = 40.000 request/sec with 48 cores 98 GB ram, ssd based.
3.456.000.000 req/day with 5 was host.



3 immersive tv

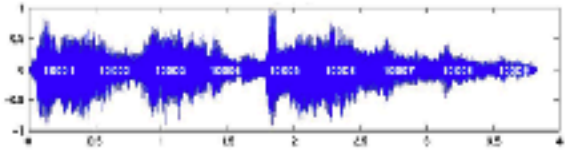
3.1. user stage concept



3.2. user stage concept: shells

Device framework

TV Event



Watermarking recognition system

Smartphones
3g/4g/internet

Platform framework

Management of realtime events

Provision framework

Multimedia contents and events.

4 designing the solution

4.1. software solution, technologies

device backend

Node JS (scalability && concurrency)

Mongodb (fast read / write persistent json)

Redis fast storage

Socket.io high realtime connectivity

Socket.proxy

backoffice provision

Mysql data storage

Web && processes

operating system

4.2. operating system tuning

4096 bytes per socket.
TCP RECYCLE (REUSE TIMEWAIT STATE)
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle (0 default value)
root@ubuntu:~# cat /etc/security/limits.conf
...
ubuntu soft nofile 1000000
ubuntu hard nofile 1000000
ubuntu soft nproc 200000
ubuntu hard nproc 200000
root@ubuntu:~# cat /etc/sysctl.conf
...
fs.file-max = 6815744
net.core.rmem_max = 25165824
# Maximum TCP Send Window
net.core.wmem_max = 25165824
# others
net.ipv4.tcp_rmem = 4096 16384 25165824
net.ipv4.tcp_wmem = 4096 16384 25165824
net.ipv4.tcp_syncookies = 1
 # this gives the kernel more memory for tcp which you need with many (100k+) open socket connections
 # 786432 = 3GB
net.ipv4.tcp_mem = **786432** 1048576 **26777216**
net.ipv4.tcp_max_tw_buckets = 360000
net.core.netdev_max_backlog = 2500
vm.min_free_kbytes = 65536
vm.swappiness = 0
net.ipv4.ip_local_port_range = 1024 65535
net.core.somaxconn = 65535
kernel.shmmax=37396480

sysctl -p
sysctl -p /etc/stsctl.conf

ulimit -n 20000000 (20k sockets open)
cat /proc/sys/fs/nr_open (1041576=> 20000000)

root@ubuntu:~# cat /etc/security/limits.conf (mongodb)
...

Limit	Soft Limit	Hard Limit	Units
Max cpu time	unlimited	unlimited	seconds
Max file size	unlimited	unlimited	bytes
Max data size	unlimited	unlimited	bytes
Max stack size	8388608	unlimited	bytes
Max core file size	0	unlimited	bytes
Max resident set	unlimited	unlimited	bytes
Max processes	**1000000**	**1000000**	processes
Max open files	**1000000**	**1000000**	files
Max locked memory	65536	65536	bytes
Max address space	unlimited	unlimited	bytes
Max file locks	unlimited	unlimited	locks
Max pending signals	643901	643901	signals
Max msgqueue size	819200	819200	bytes
Max nice priority	0	0	
Max realtime priority	0	0	
Max realtime timeout	unlimited	unlimited	us

see: https://blog.jayway.com/2015/04/13/600k-concurrent-websocket-connections-on-aws-using-node-js/

4.3. online hot parameters change

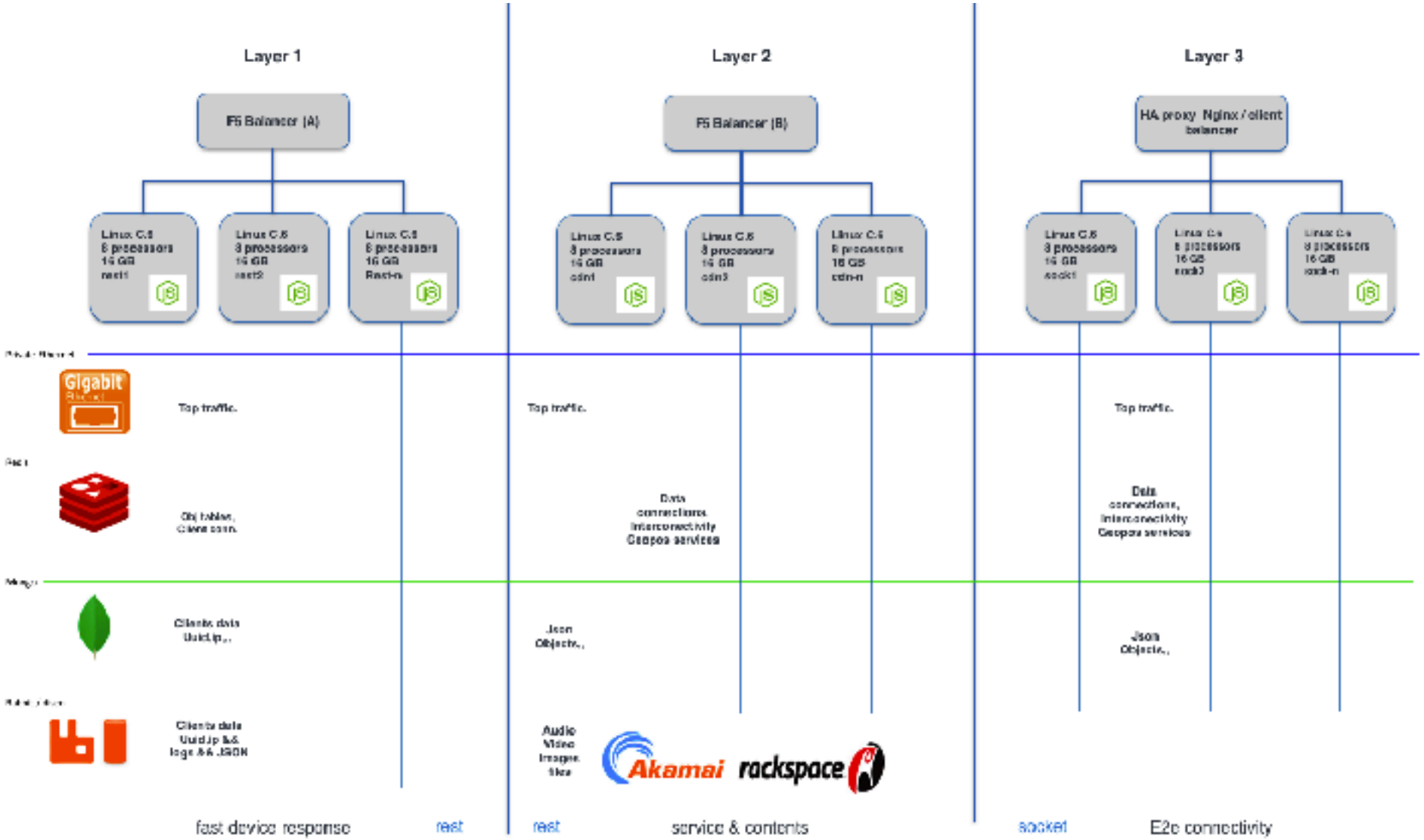
in productive environments there are situations in which it is necessary to change certain parameters of operating system process configuration, which must be done in hot, for example increase the number of processes that a bbdd /socket or process must open,

first of all, you must known the id process (ps -ef | grep "mongodb") 11980
cat /proc/11980/limits
this file shows the actual process limits,

you can increase this value:
echo -n "Max processed=200000:320000" >/proc/11980/limits

5 architecture overview

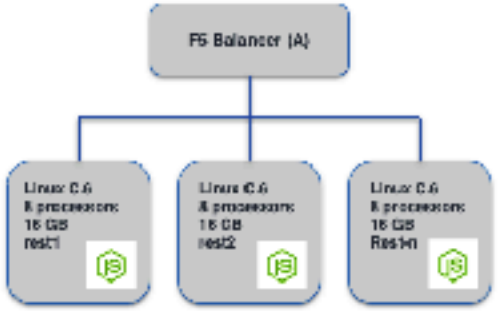
5.1. layers



6 login overview & balancing methods

6.1. layers

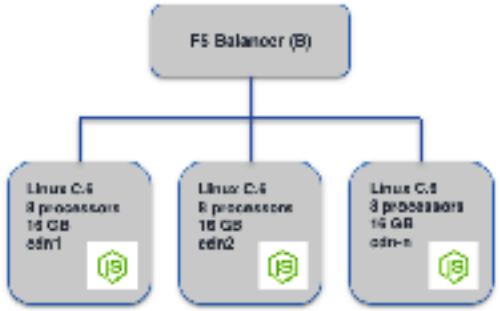
Layer 1



4..N rest backends at 8100 port. traffic http.
65 rest micro services for mobile devices (register user, apps, load json data..,etc)
8 process by cluster & processors

rest

Layer 2



3..N rest backends of cdn services at 80 / 81 http / https port.
21 rest different services on line (social network, push,reports, map audiences,..etc)
CDN services with resizing on the fly images
8 processors by cluster

Akamai rackspace

service & contents

Layer 3

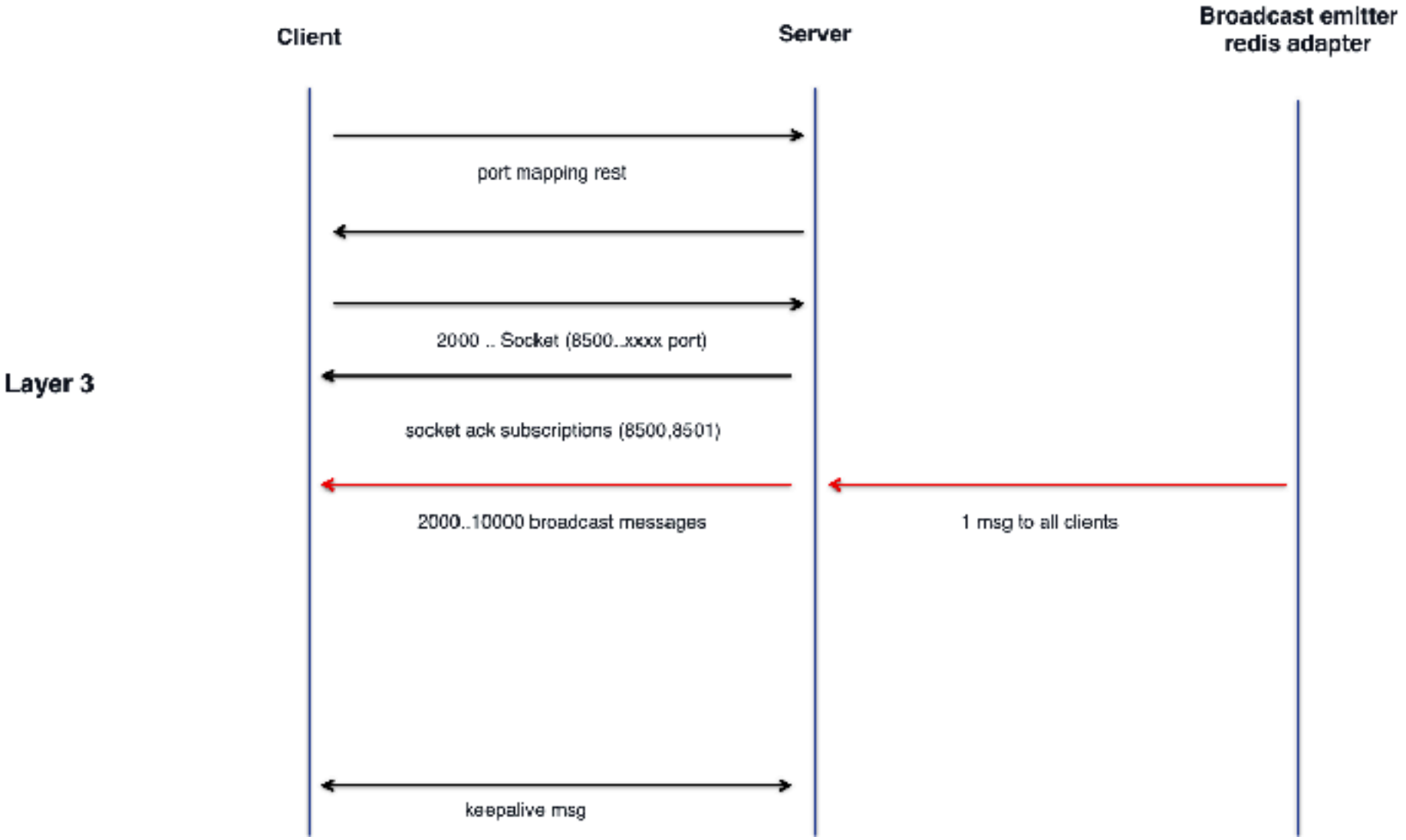
1..5 host for 200K user per host. 1000k sockets max.
5 integration services (send event,reload json events,..).

Master->controllers->servers
E2c user connect services (chat)
Geopost services for virtual reality II games...

socket

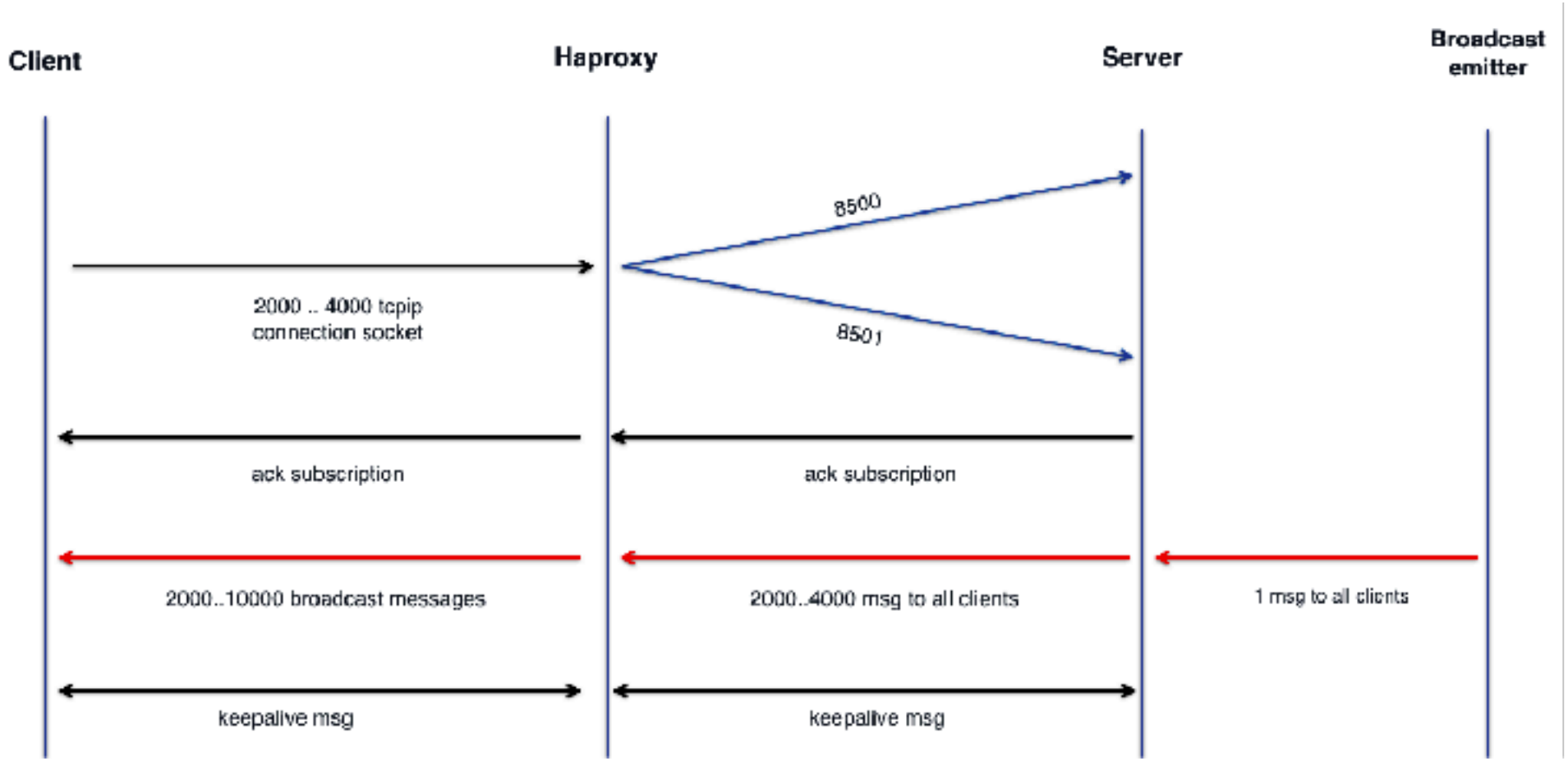
6 login overview & balancing methods

6.2. rolling client (I)



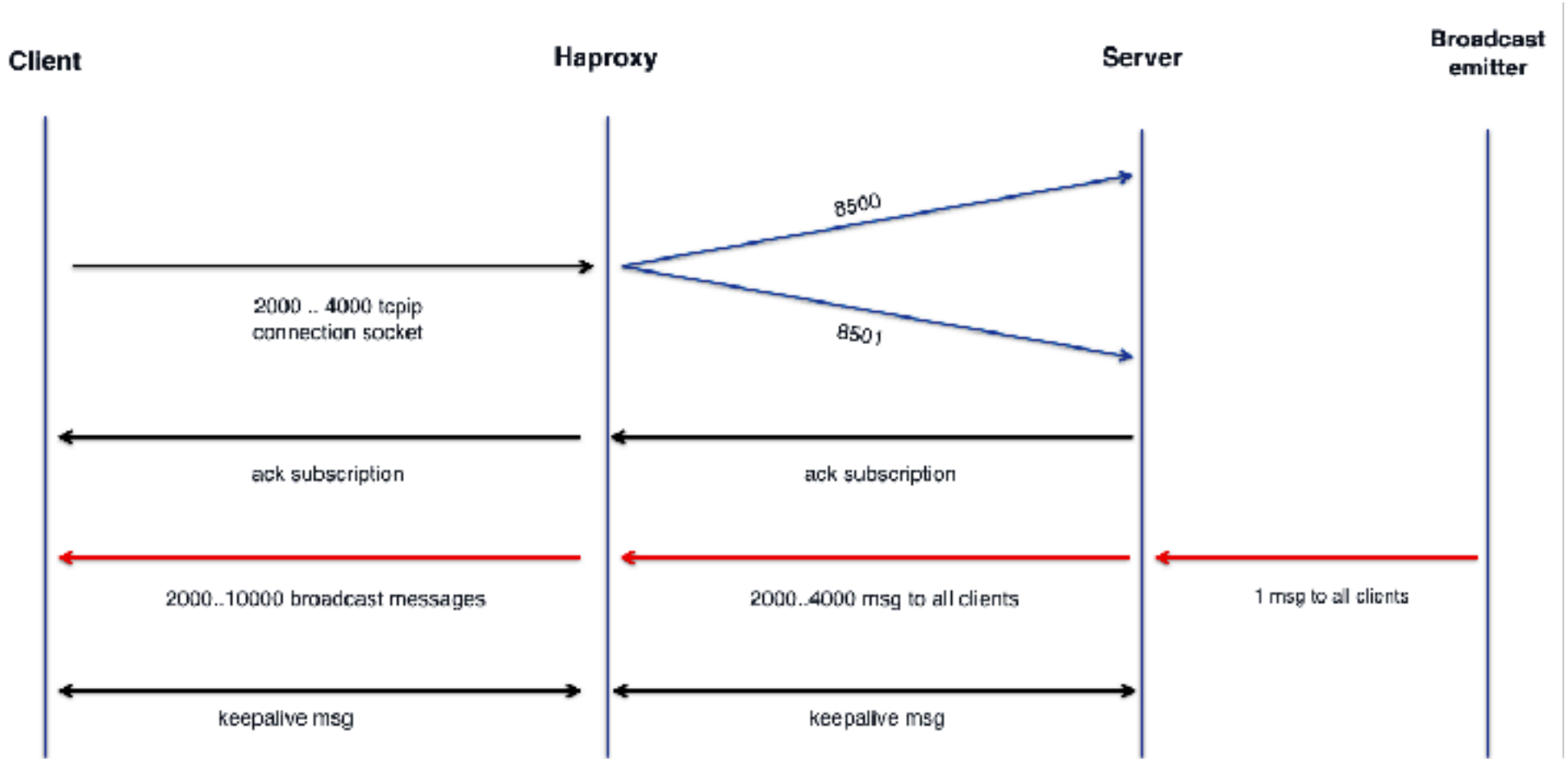
6 login overview & balancing methods

6.3. haproxy (II)



6 login overview & balancing methods

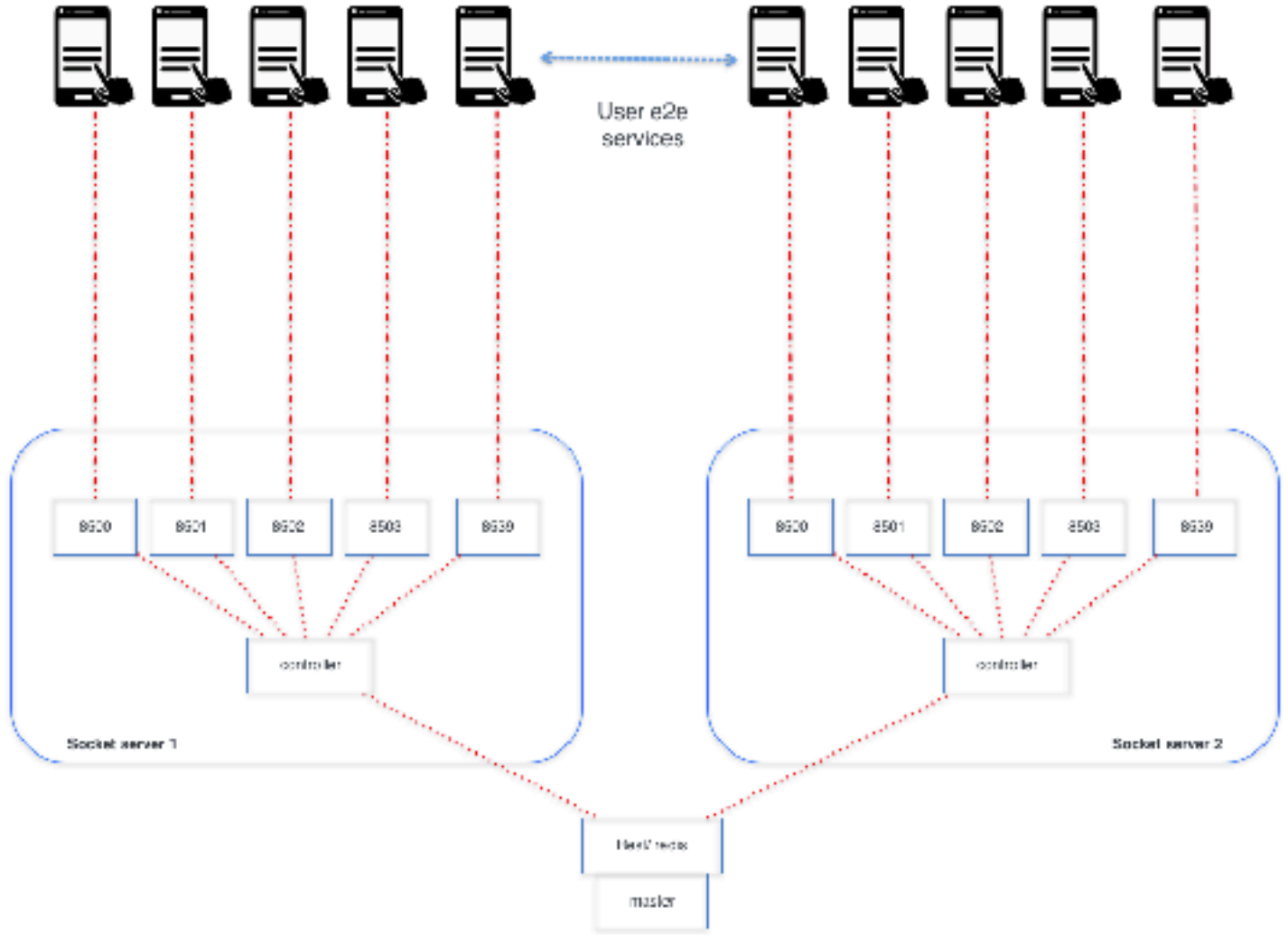
6.4. haproxy (II)



6 login overview & balancing methods

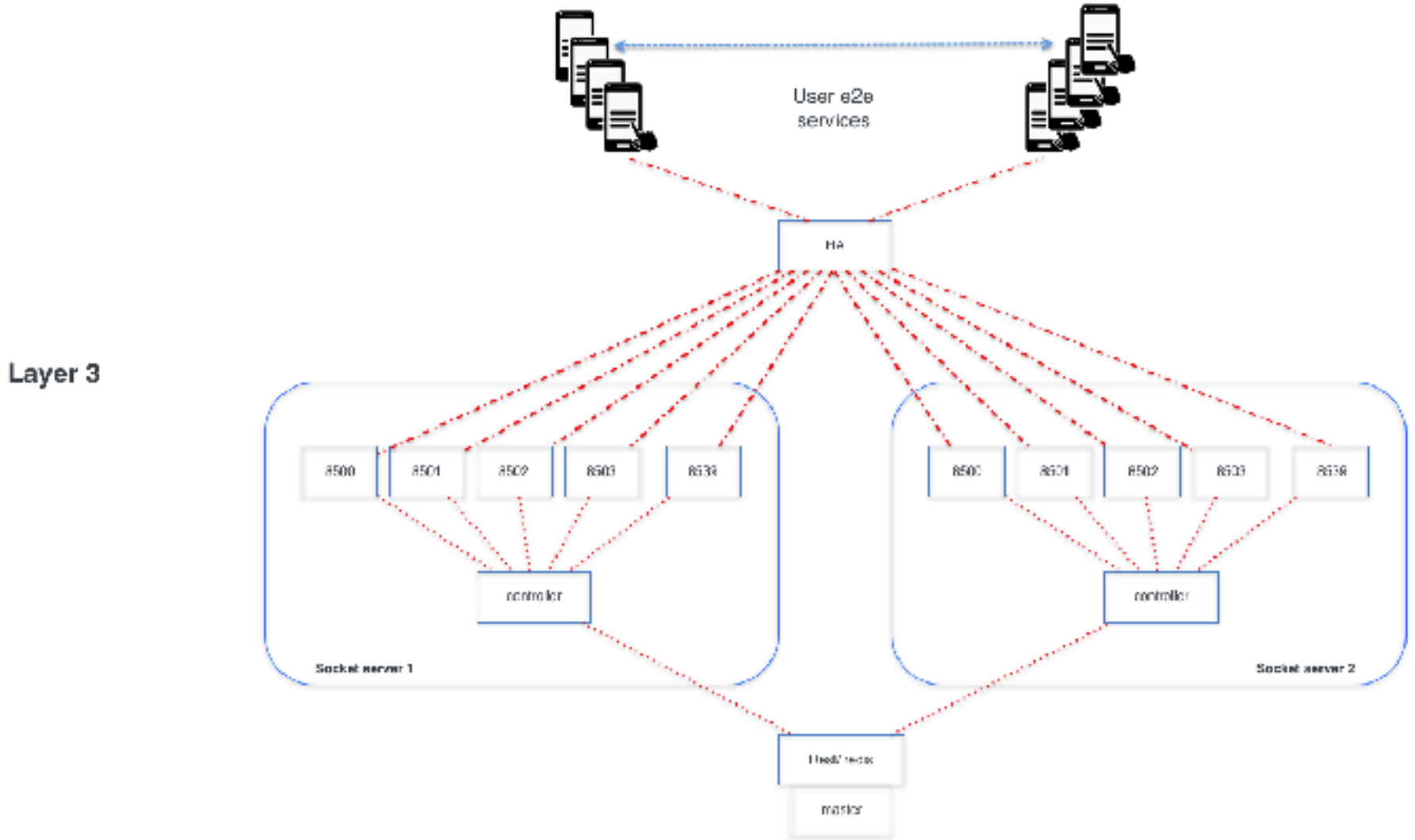
6.4. broadcast / e2e message flow

Layer 3



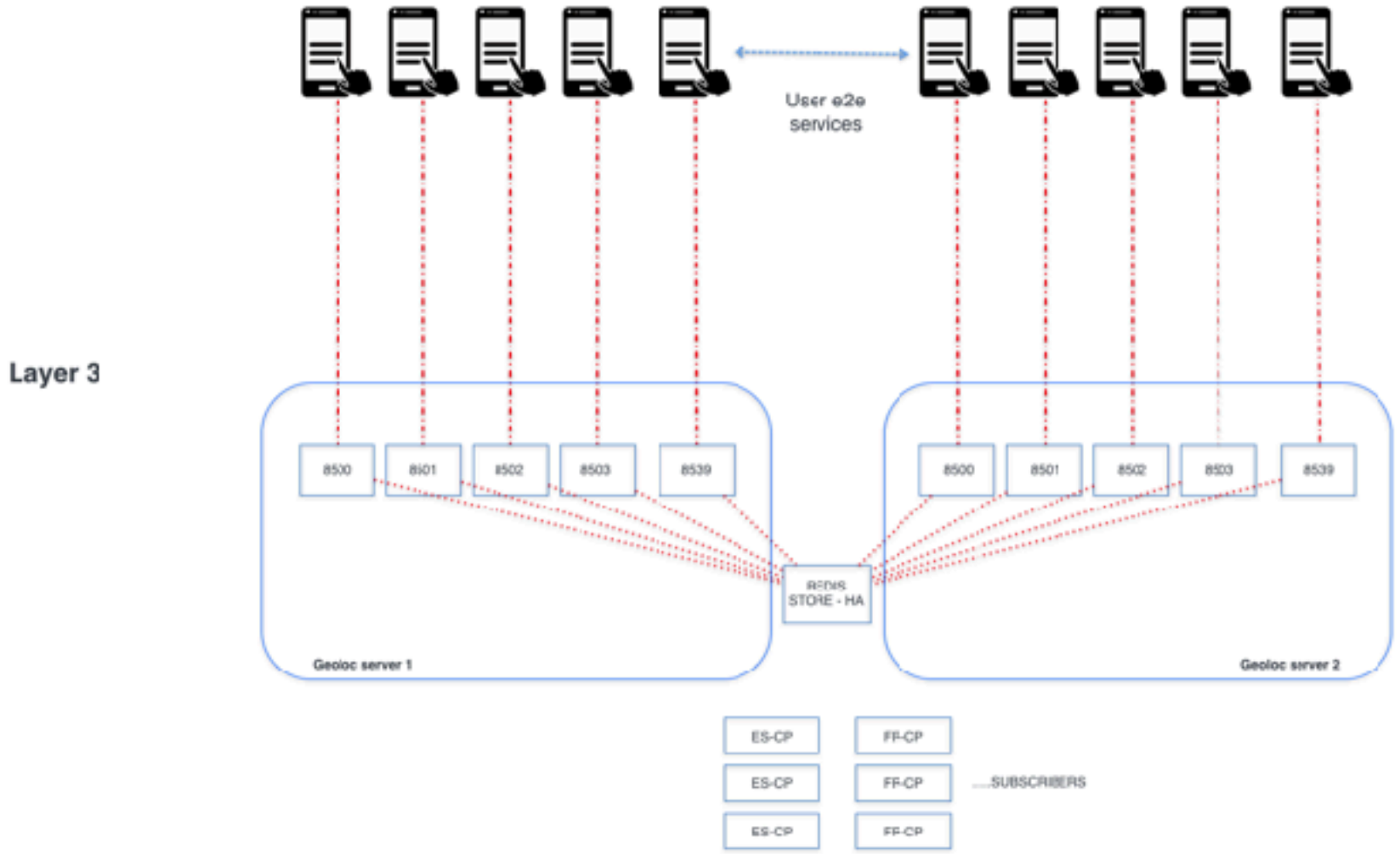
6 login overview & balancing methods

6.5. HA / e2e message flow



6 login overview & balancing methods

6.6. geolocation / message flow



7 heuristic source code recommendations

7.1. source code recommendations: layer 1

Layer 1 (rest) microservices

Very quick rest services
400K-1.300K rest / host
Max 200k/sec.
20-40 ms max.

Emacs5/Emacs6
Express framework.
Good idea to use cluster module

Insert code directly at route scope, avoiding functions calls if its possible.

Avoid use of https protocol, instead use symmetric encription functions.

Its better to use structured programming that object oriented programming

Same content to all devices = memory cache or redis.

Use standard control statements flow (if,else,for,while instead foreach, for in, async,,,)

You can make your own modules in C++ and compile them for use in node js

You can compile node js code with Jxcore

7 heuristic source code recommendations

7.2. source code recommendations: layer 2

Layer 2 (services&contents)

Service model rest services
< 400K-600K rest / host
Max 200k/sec.
50-80ms max.

- Framework oriented services (Route & service based arquitecture to scale api rest services) && code shell design.
- http or https access or simmetric encription.
- Mongo && mariadb && redis store pool connectors.
- Async + lib code to parallel process
- TDD
- Tracking and geolocalization services
- Audience maps on line reports.
- CDN server multimedia images && resizing on the fly.

Layer 1 & 2 REST testing

- LoadTest: https://github.com/alexfernandez/loadtest
- Nperf fork: https://github.com/zanchin/node-http-perf
- added 'post' method to original project.
- Apache ab software.

7 heuristic source code recommendations

7.3. source code recommendations: layer 3

Layer 3
(socket)
e2e connectivity

Socket service
100K 1000K users
1..n / 1..1

- Tunning ssoo net & process kernel parameters is required.
- Express + socket.io based library
- Master -> controller -> server separate process.
- (1 master -> 5 controllers -> 100 servers)
- Fork node subprocess better than cluster module (better memory management)
- Redis store for link users
- 50 active ports (8500 – 8550) , 4.000 sockets / port = 200.000 sockets/host
- Real time redis allocacion tables.
- 2 ways to balance ,client redis-rest based vs haproxy && nginx-plus proxy)

Redis rest
Vs
haproxy
Nginx-plus

- Redis-rest redirect client to ip and port (2 request are needed, 1 persistent)
- HA-NGX consume 2 persistent connections (in / out) increase net resources.
- Failover proxys are needed.
- HA-NGX more faster socket connection (one request)

7 heuristic source code recommendations

7.4 refactor levels

Level 1

using bitwise operators,,,
'async waterfall' to avoid callback hells,,,
grunt / gulp linting rules (avoing deep loops),,,
..

Level 2

Using latest versions (ssoo + node + libs)
..

50.000k x

```
function factorial(n){
  if (!(n^=0))
   return 1;
  else
   return (n * factorial(n - 1));
}
```

Node 1.0.0 = 11496 ms

Node 5.0.0 = 8828 ms

Javascript

C language

Level 3

Using c,c++ at critical modules
..

50.000k x

```
long double fact(int n){
  if (!(n^=0))
   return 1;
  else
   return (n * fact(n - 1));
}
```

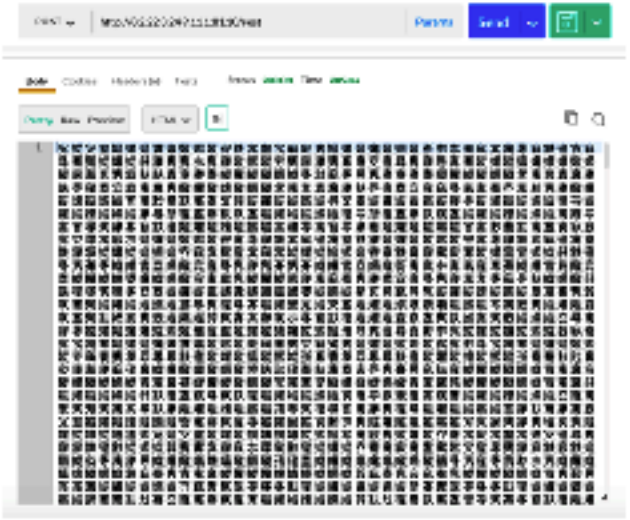
node 1.0.0 = 0.053 ms
node 5.0.0 = 0.038 ms
node 9.2.0 = 0.031 ms

8 security recommendations

8.1 security recommendations

code

Post methods.
Simmetrical coding algoritms are often sufficient to ofuscate data (instead https)
:
var b = a ^ 342951123 ➔
:
a = b ^ 342951123
Based user tokens at middlewares are useful

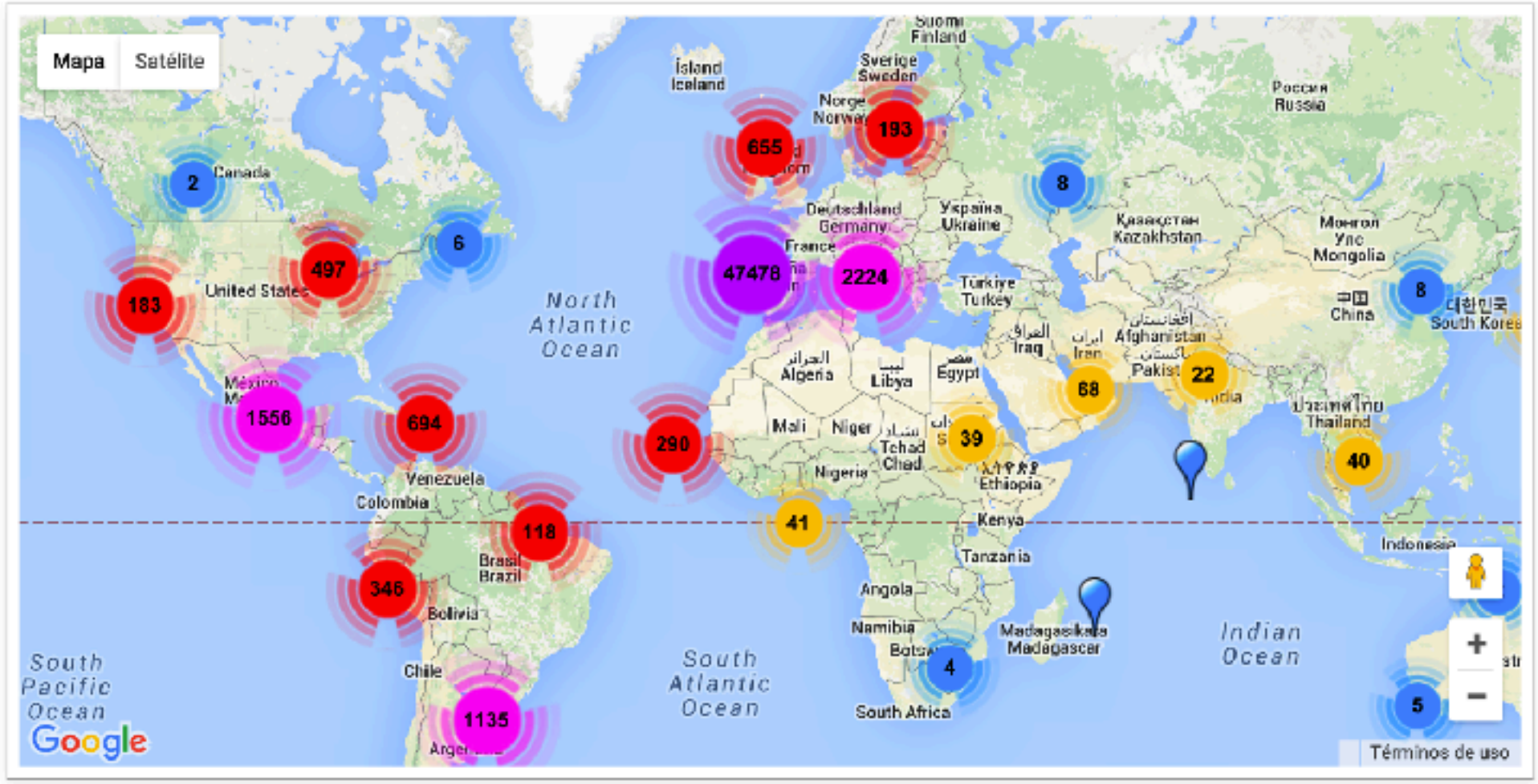


Access and content

(1) Application load balancer (alb - amazon)
(2) Firewall rules apply (only the neccesary ports)
(3) ssh certificate to access cloud .
(4) fail2ban to capture attack (ips to jail for 24 or 48 h).
(5) geoip all incomming connections to track it.
(6) its quite useful to generate and distribute fake directories, like the autentic content.
(7) designing automated procedures (cron daemons) to restore the original terms (json) of content and exchange objects every few minutes.
(8) user/pass to all databases (redis&mariadb&mongo&rabbitmq)

9 geolocation world maps

9.1 geolocation mongodb ip database service example



10 everis i+d projects (2018 -2019)

10 Here we have some of our innovation ideas

Urban Traffic Control Network
Everis Smart Agent
Biometric identification module
Multi language chat translator module
Guided drone with facial recognition
Visual representation systems based on gestural interfaces
Audio/video watermarking management

Thanks !

ありがとう

ignacio.ariza.victoria@everis.com

ignacio.ariza@gmail.com

show me the code!

https://github.com/iav2014