The Server Architecture Behind Killzone Shadow Fall

Jorrit Rouwé
Lead Game Tech, Guerrilla Games

Overview

- Hosting Choices
- Server Architecture
- Match Making
- Zero Downtime
- Updating Without Patching
- How Telemetry Helped Fix Issues

Introduction

- Killzone Shadow Fall is a First Person Shooter
- 24 player team based gameplay
- Warzone = Customizable Game
  - Selection of maps / missions / rules
- Official and community created Warzones
- Multiple active games in a Warzone
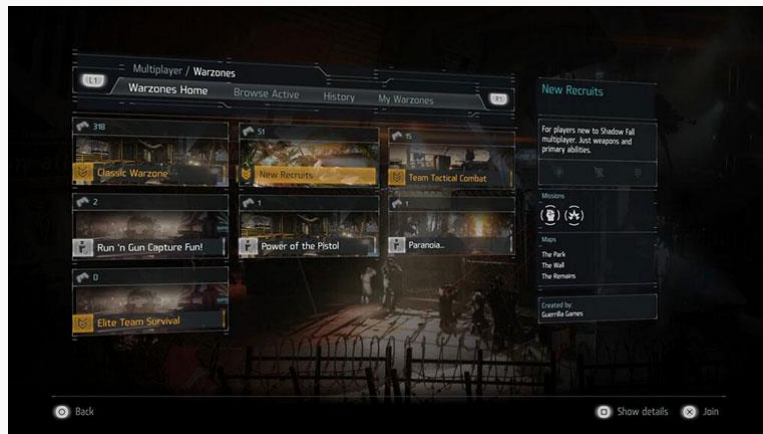
- Bottom right screenshot shows 300+ players in 1 warzone

Hosting Choices

DEVCON2014

Hosting Choices - Need for Scaling

- Killzone 3 used traditional hosting
  - Have to manage many environments
  - Adding / scaling environment needs planning
  - Shared load testing environment with other titles

- Example environments: Guerrilla Internal, FPQA, FQA, Demo, Press, Production, Load Test

Hosting Choices - Need for Scaling (2)

- For Killzone Shadow Fall we needed flexibility
  - Launch title = no Public Beta
  - Effect PlayStation®Plus pay wall?

- Used Amazon for easy scaling

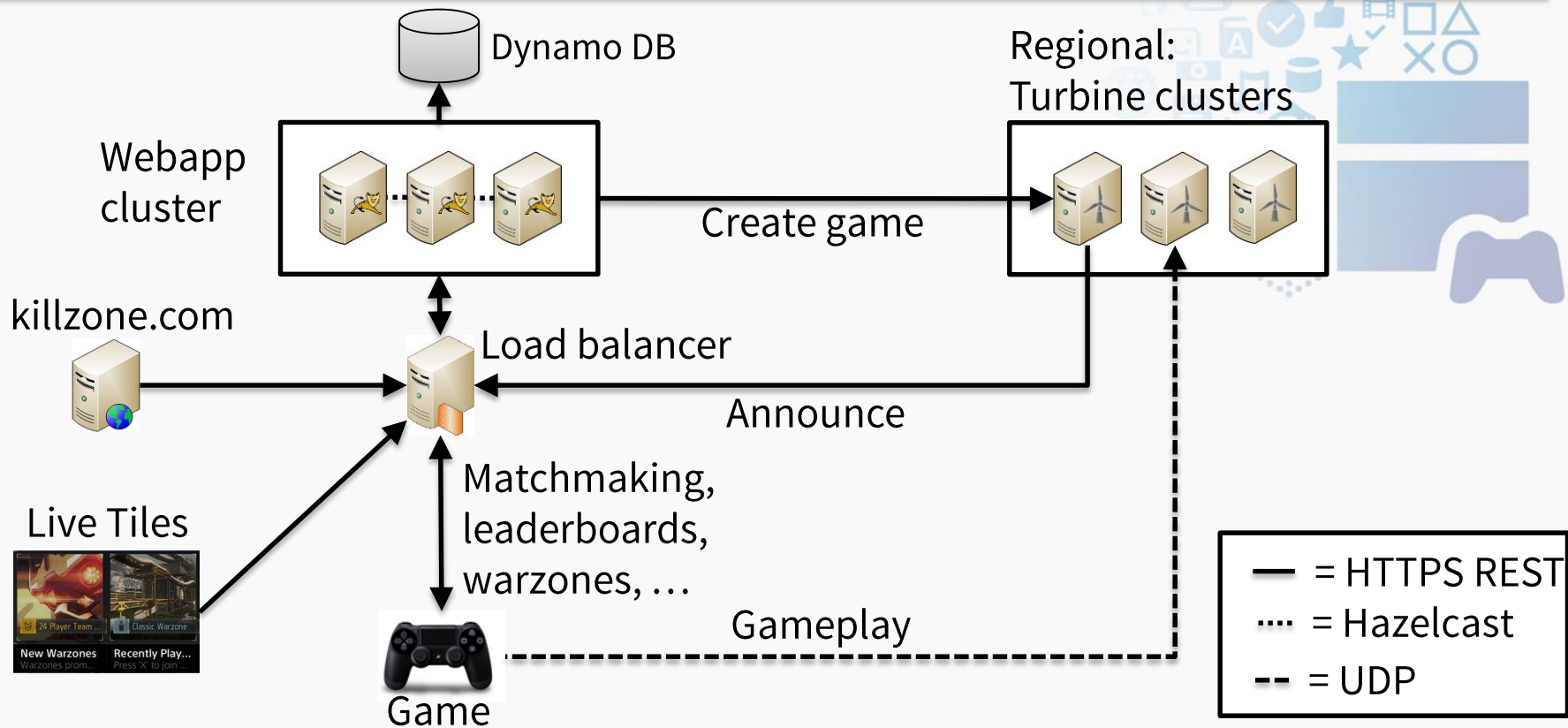- Load testing
  - 100K sims
  - 100 real players

- Some servers still physically hosted in Sony data centers

Server Architecture

Server Architecture - Overview

- Most components use standard web technology
  - Java
  - Tomcat
  - REST

- Simpler hosting by using Amazon components
  - Storage: Dynamo DB / S3 / Redshift
  - Deployment: Cloud Formation / Elastic Beanstalk
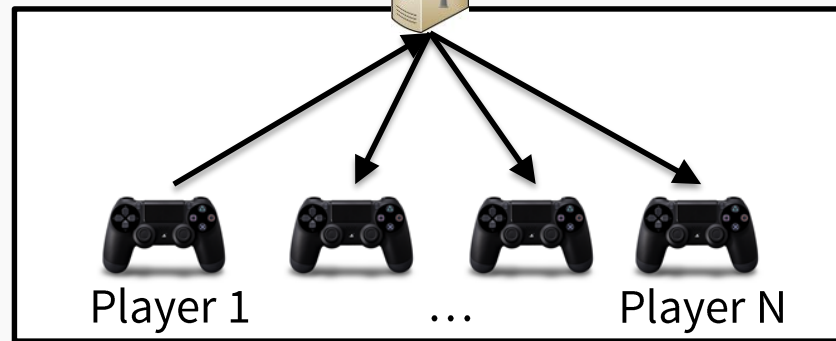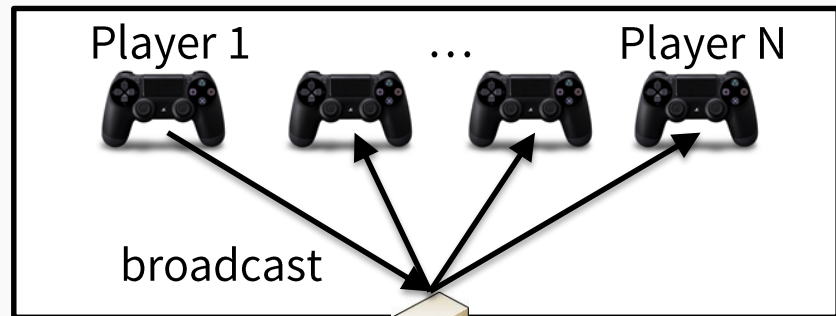 - Monitoring: Cloud Watch

Server Architecture - Overview (2)

Dynamo DB

Regional:
Turbine clusters

Webapp
cluster

Create game

killzone.com

Load balancer

Announce

Live Tiles

Matchmaking,
leaderboards,
warzones, …

Gameplay

— = HTTPS REST

···· = Hazelcast

-- = UDP

Game

- Webapps are all running the same components
- Turbine hosted partially in Sony data centers, partially in AWS
- Hazelcast is peer to peer communication library

Server Architecture - Gameplay (Turbine)

- C++ / libRUDP
- Messaging hub between players in a game
- Game logic runs on the PS4™
- Advantages:
  - Prevents NAT issues
  - Reduces required bandwidth
- Disadvantages:
  - We pay for bandwidth
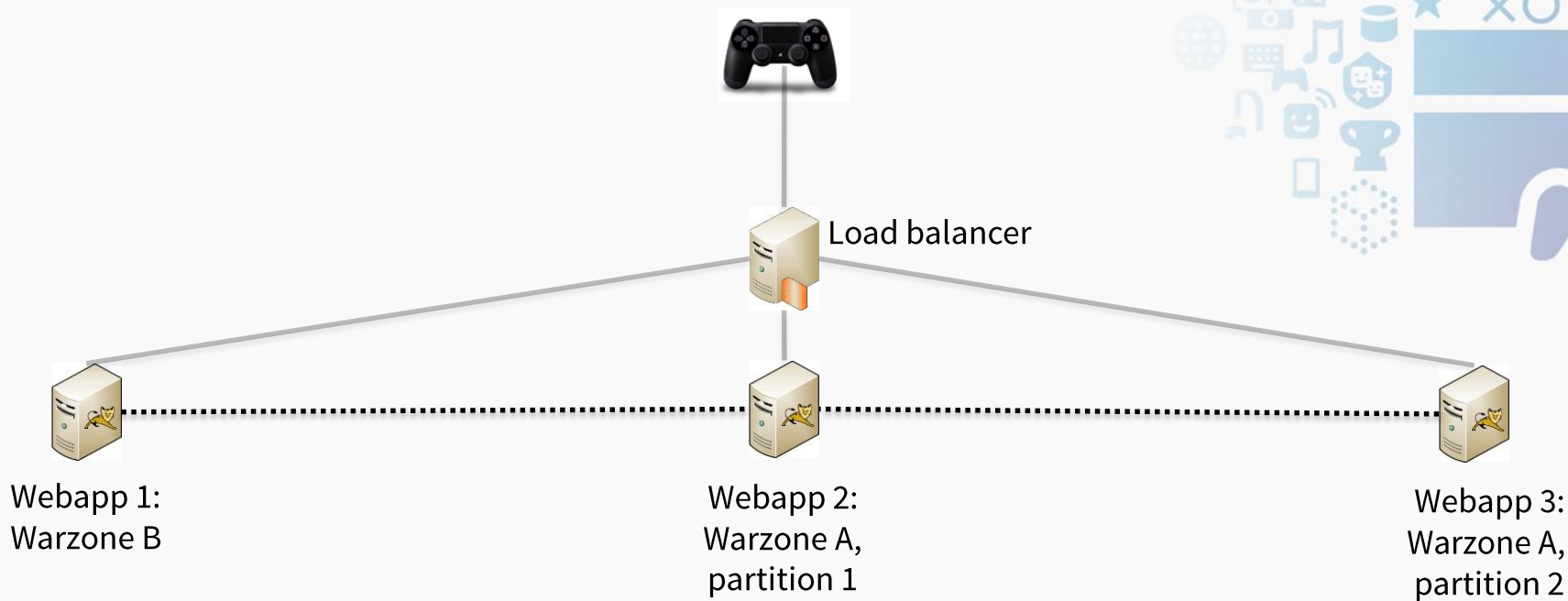 - Introduces extra hop

Match Making

Match Making - General Principles

- Linear scaling
  - State stored in memory
  - Minimal communication between servers

- Fault tolerant
  - All servers equal
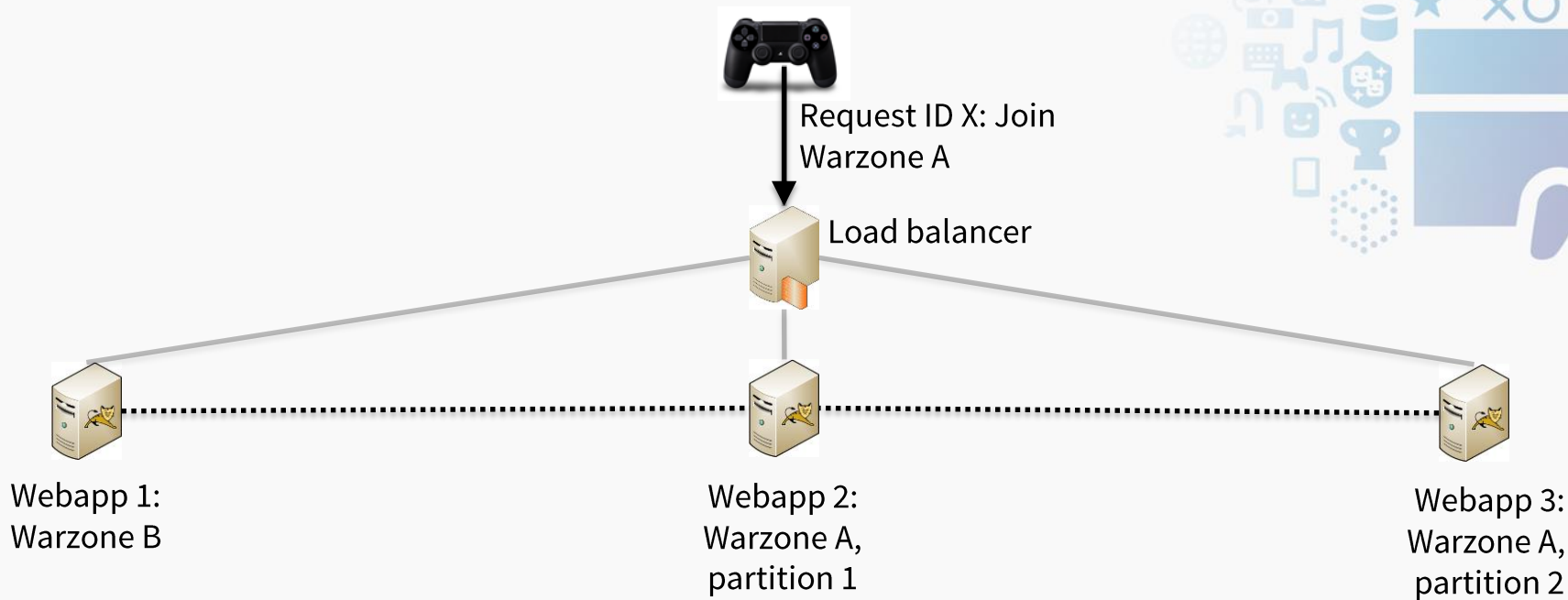 - Client repeats requests
  - Client sends match updates

- One server not aware of games hosted on other server
- Since state is stored in memory clients need to repeat requests to reconstruct state in case of server outage

Match Making - Algorithm

Load balancer

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2
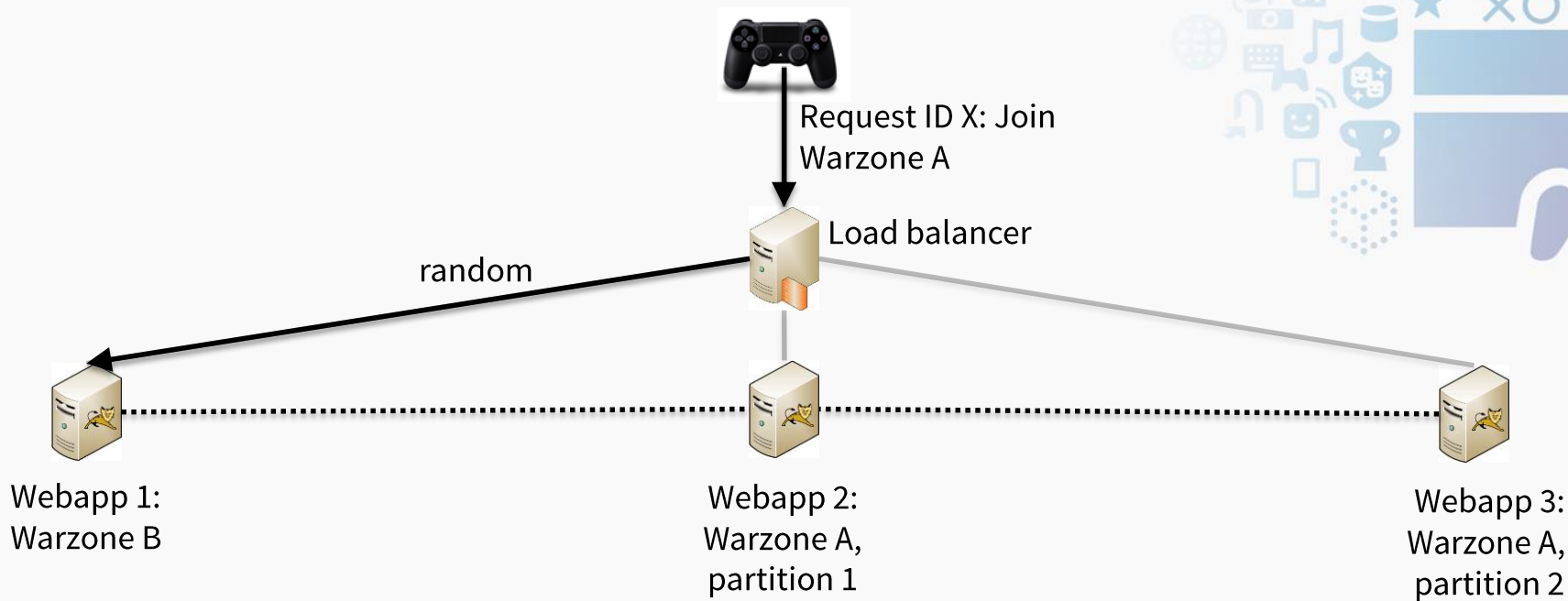
- Warzone B is small
- Warzone A is too big to be hosted by 1 server so is split up in 2 partitions

Match Making - Algorithm

Request ID X: Join Warzone A

Load balancer

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2

- Client creates request with random ID X

DEVCON2014

Match Making - Algorithm

Request ID X: Join Warzone A

Load balancer

random

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2

Match Making - Algorithm

Request ID X: Join Warzone A

Load balancer

Shared map:
Who hosts what warzone

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2

- Using shared Hazelcast map

DEVCON2014

Match Making - Algorithm

Request ID X: Join
Warzone A

Load balancer

In memory:
Pending request

Forward: Warzone A,
partition consistent_hash(X)

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1
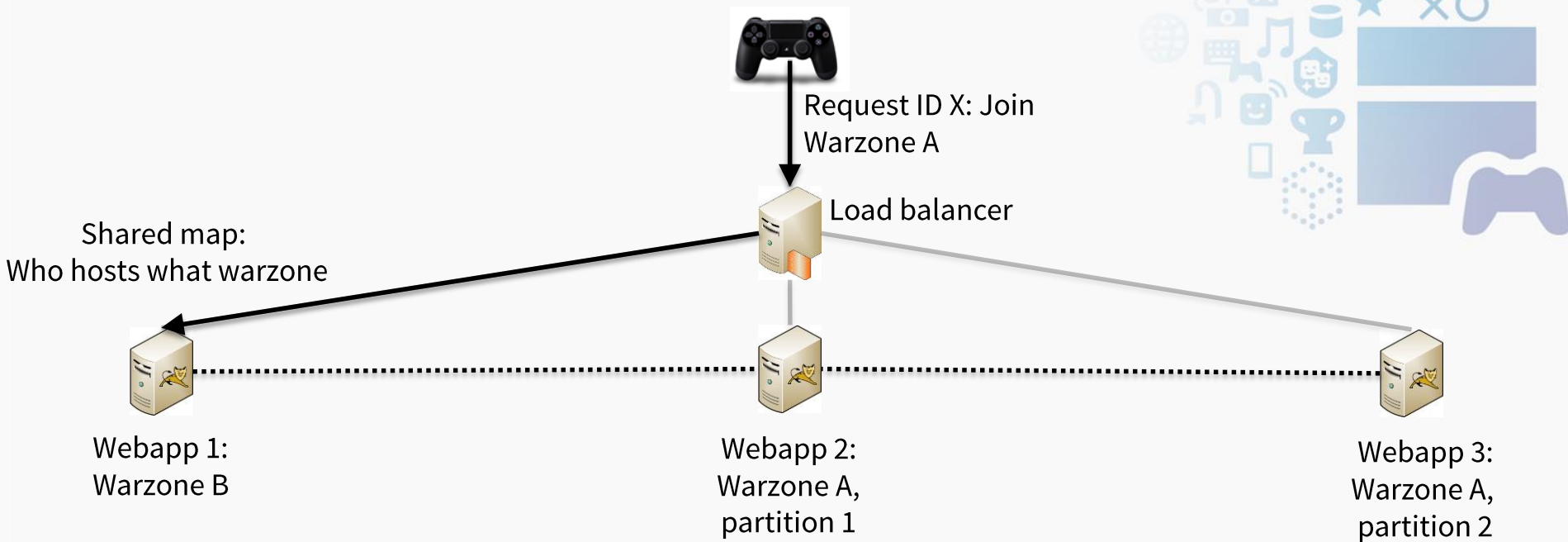
Webapp 3:
Warzone A,
partition 2
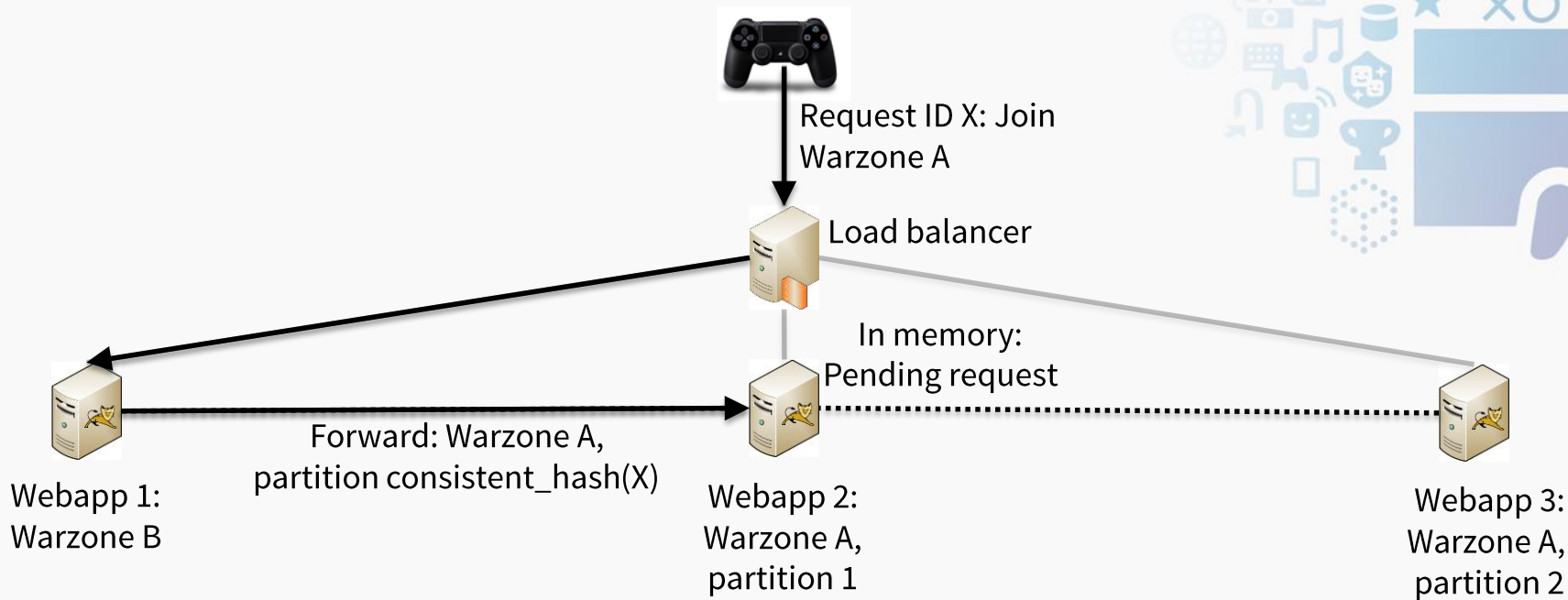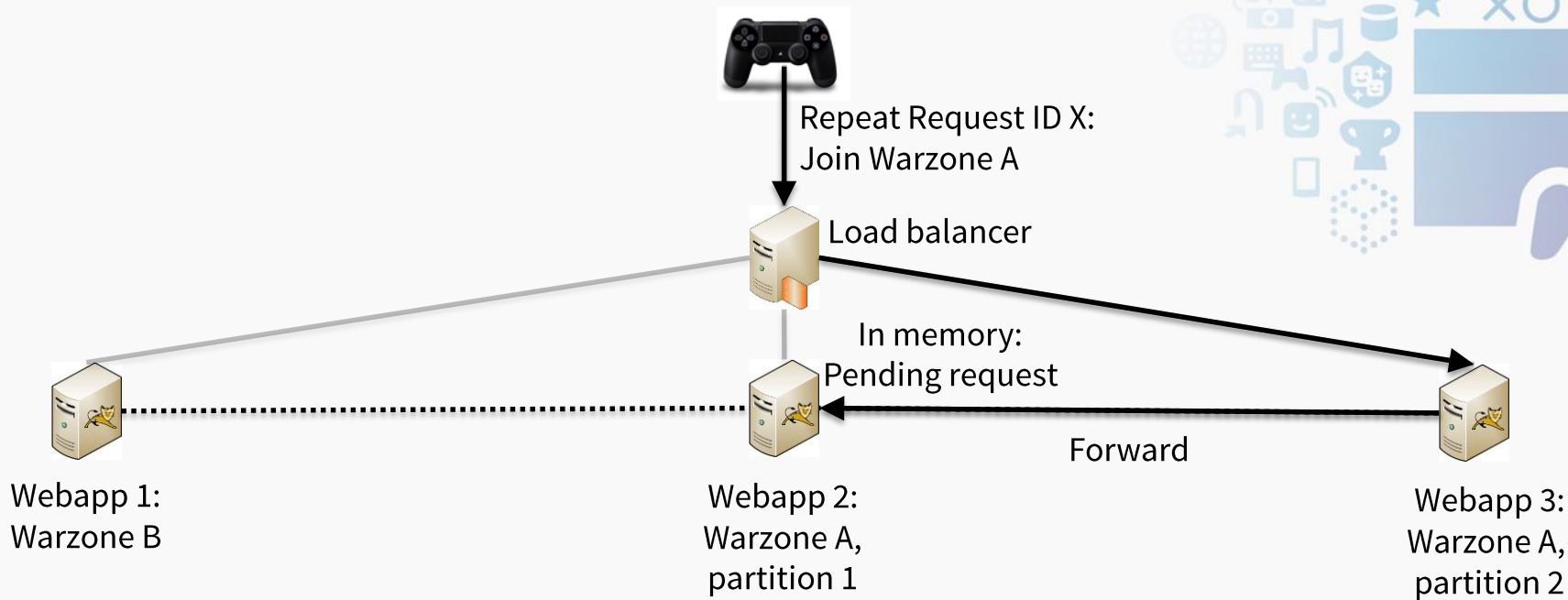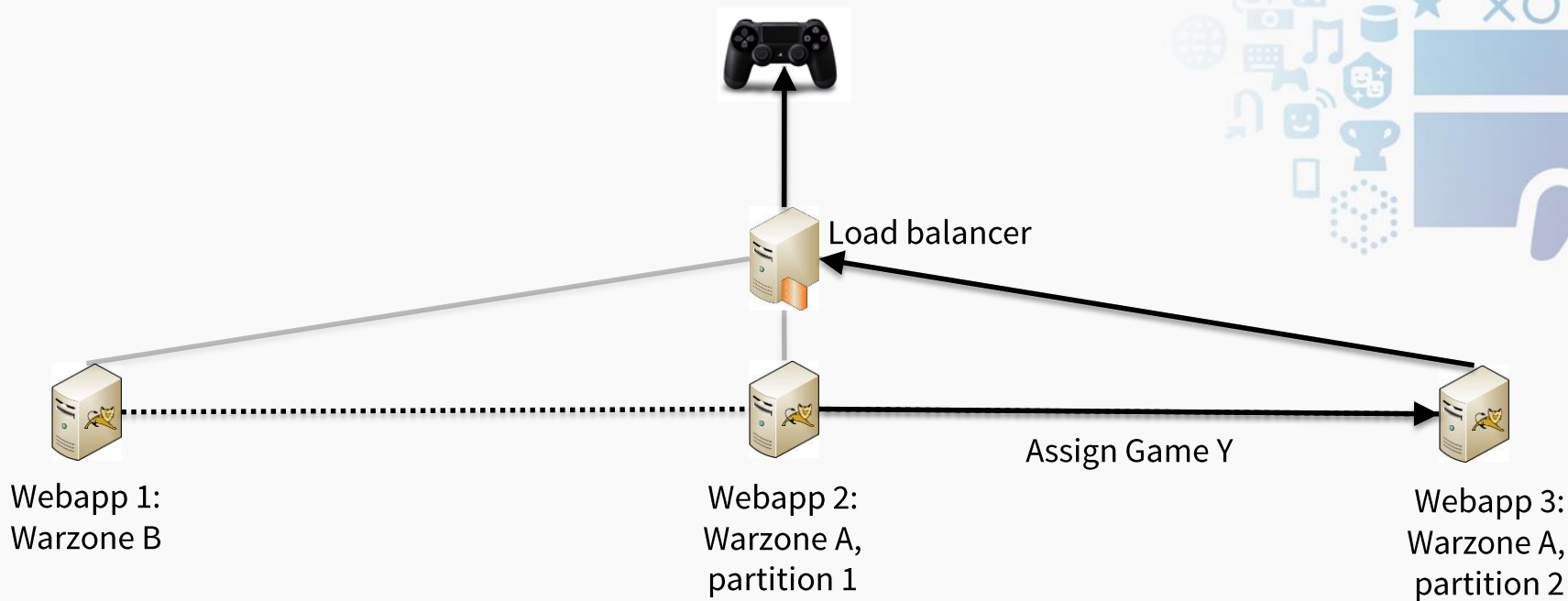
- Lets say match was not found first time, so webapp 2 returns match
making progress status

Match Making - Algorithm

Repeat Request ID X:
Join Warzone A

Load balancer

In memory:
Pending request

Forward

Webapp 1:
Warzone B

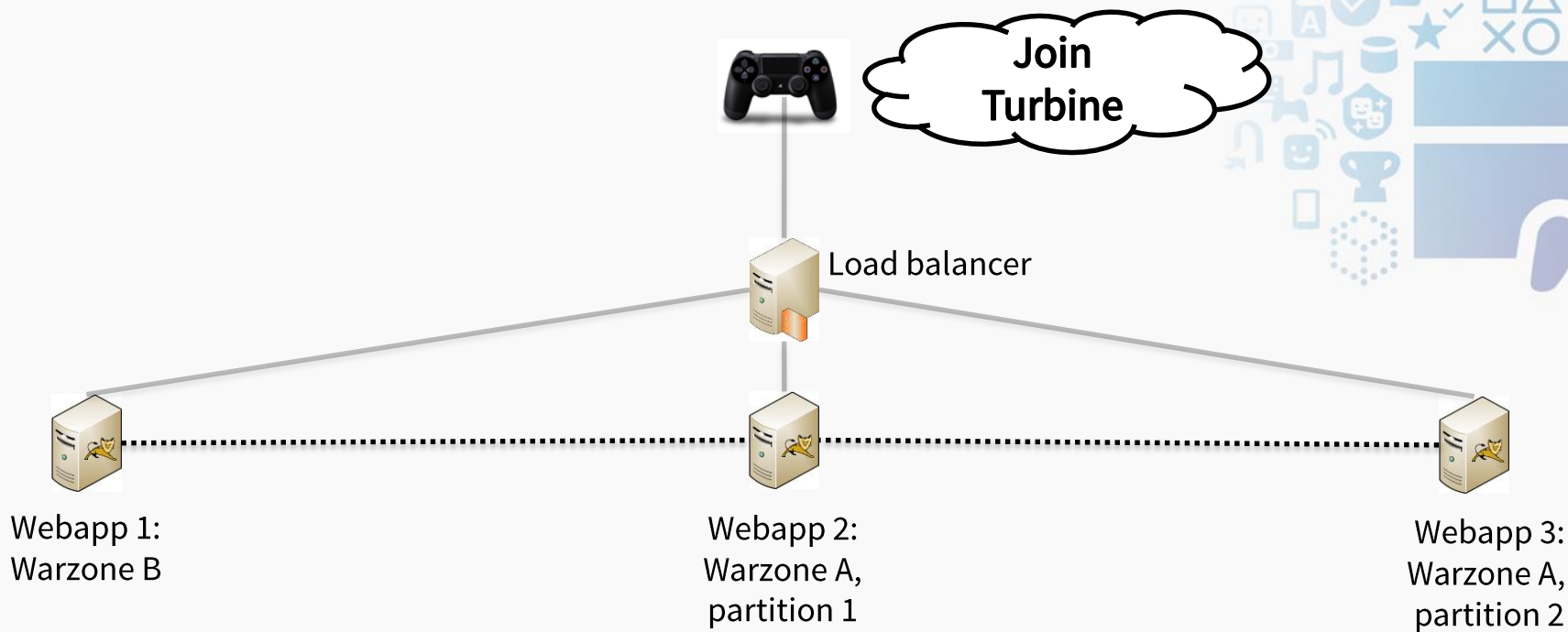Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2

- After 5 seconds client repeats request
- Request has same ID so ends up on webapp 2 again

Match Making - Algorithm

Load balancer

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Assign Game Y

Webapp 3:
Warzone A,
partition 2

Match Making - Algorithm

Join Turbine

Load balancer

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2

- Game uses IP / port returned by game assignment to connect to Turbine

Match Making - Algorithm

Game Y, Warzone A:
Match Update

Load balancer

In memory:
players in Game Y

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Forward: Warzone A,
partition consistent_hash(Y)

Webapp 3:
Warzone A,
partition 2
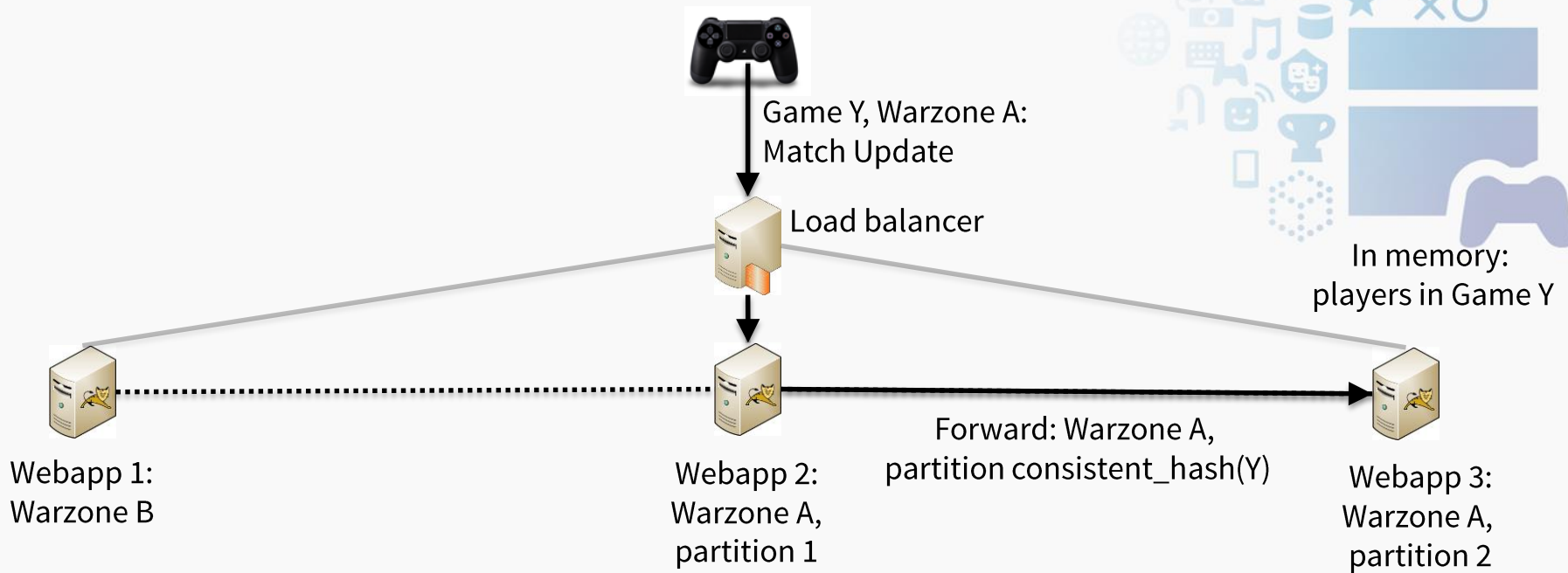
- One client in game send match update when player joins (and periodically in 30 second intervals). Game can be hosted on different webapp because consistent_hash(Y) is used

Match Making - Algorithm

Load balancer

Webapp 1:
Warzone B

Webapp 2:
Warzone A,
partition 1

Webapp 3:
Warzone A,
partition 2
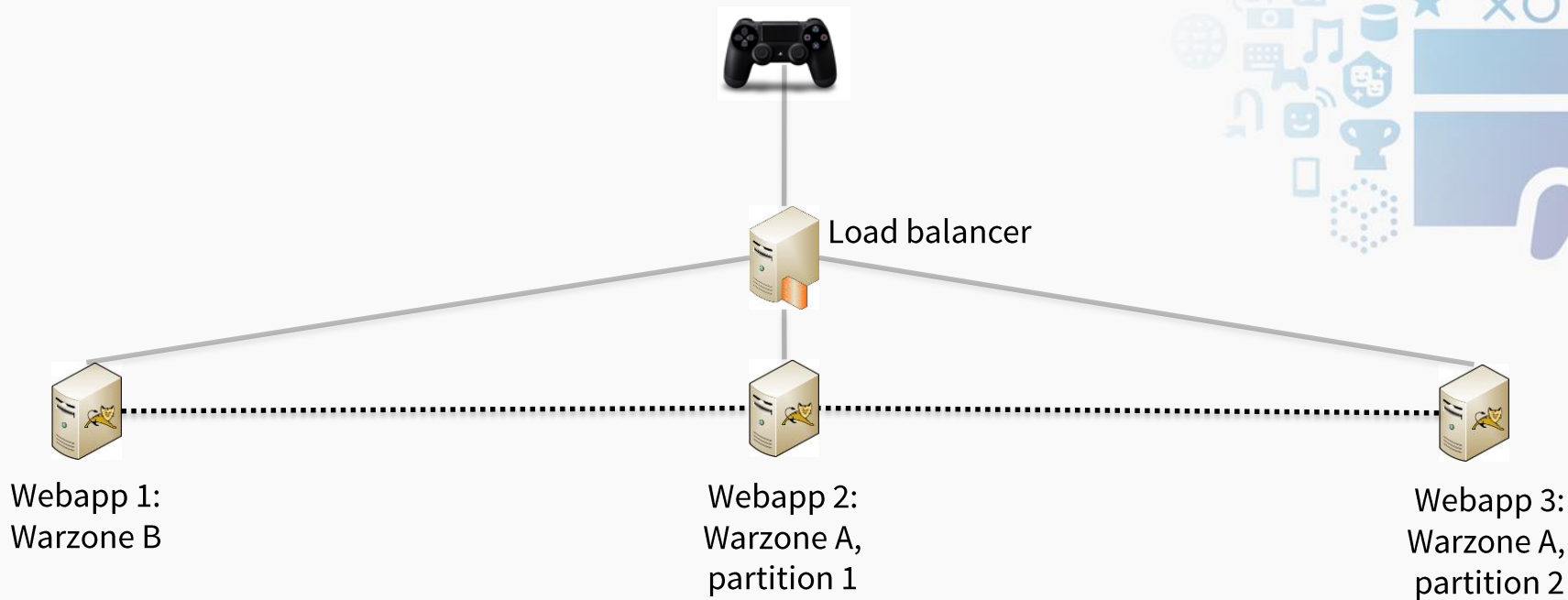
No Cross Partition Matching

- Max partition size around 10K players
- Requests only match with games on same partition
- Requests can create new game on different node

Match Making - Finding Game

- Greedy algorithm for finding best game
- Weighted scoring, in priority order:
  - Region based on ping
  - Good / bad ping players
  - Biggest games first
  - Long unbalanced first
  - ELO Glicko skill
  - Don't join game you just left

DEVCON2014

- Ping all data centers on application startup
- Match outside closest region after period of time
- Good ping < 100 ms

Match Making - Round End

- Merge small game into large
- Put players back into own region
- Balance team sizes
- Balance skill

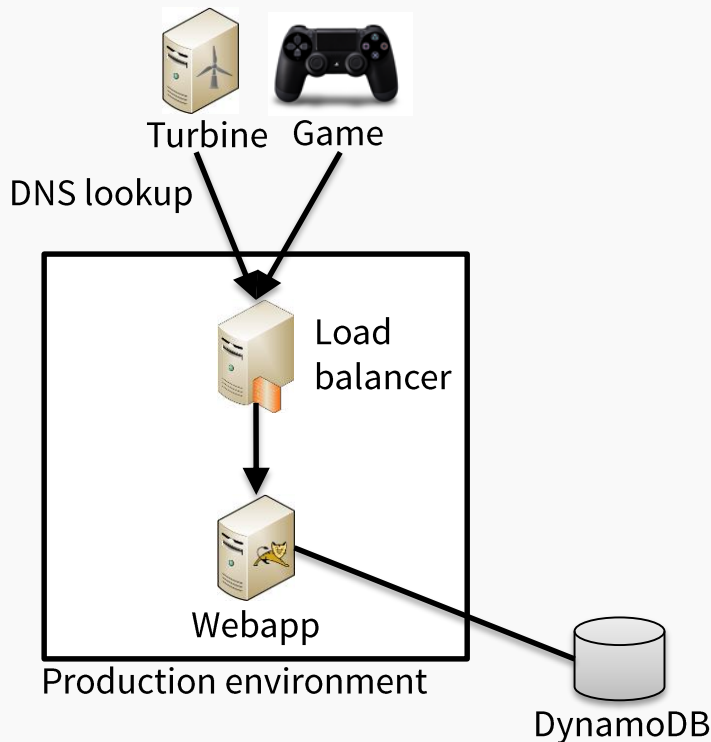- Take best 1 or 2 players from 1 team and exchange for worst players on other team

Zero Downtime
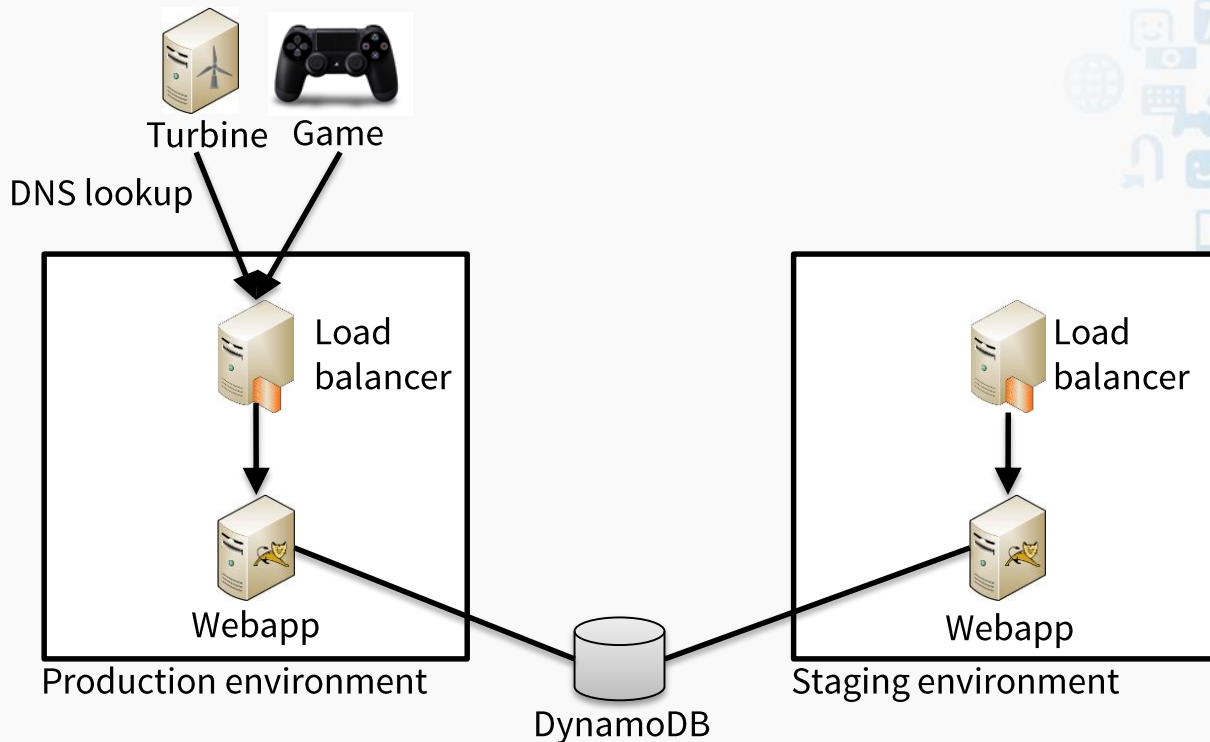
Zero Downtime - Overview

- Killzone 3 often had hour long maintenance

- Zero downtime deployment / scaling

- Manage our own deployments
  - Being on call 24/7 makes us want to fix issues!

- Every service redundant
  - So far only 2 outages (< 1 hr) due to our servers failing
  - Survived multiple 'cable pulls'

Zero Downtime Deployment - Webapp

Turbine Game

DNS lookup

Load
balancer

Webapp

Production environment

DynamoDB

Zero Downtime Deployment - Webapp

Turbine   Game

DNS lookup

Load balancer

Webapp

Production environment

DynamoDB

Load balancer

Webapp

Staging environment
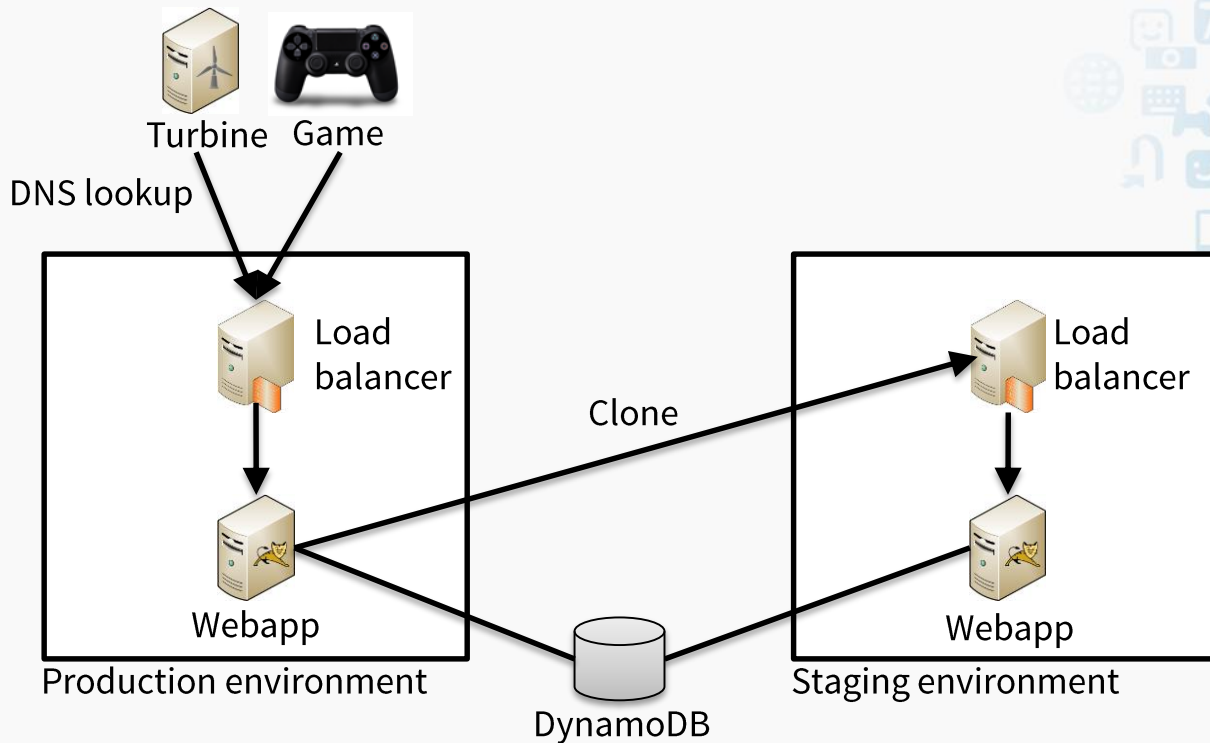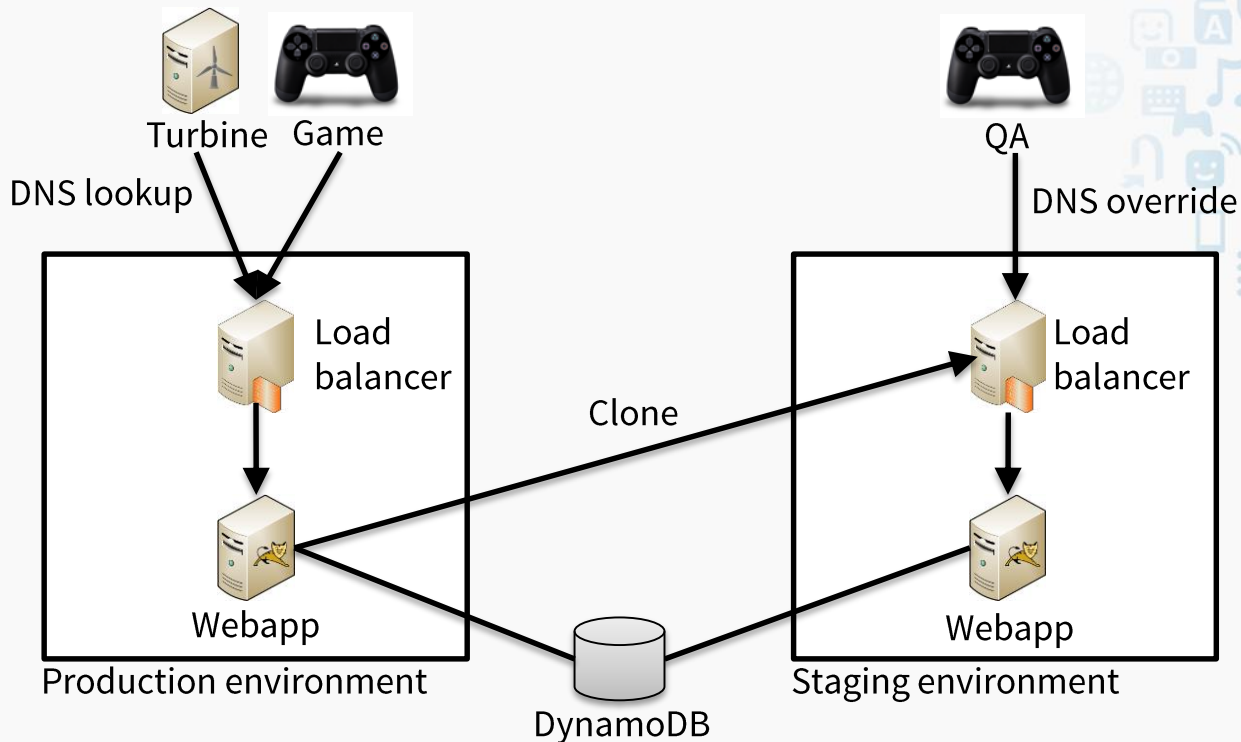
- Use Amazon to create second server set
- Database is shared
- New server is backwards compatible with old server

Zero Downtime Deployment - Webapp

Turbine   Game

DNS lookup

Load balancer

Clone

Load balancer

Webapp

Production environment

DynamoDB

Webapp

Staging environment

- Clone handles request on production environment and replicates the request to the staging environment (but ignores result). Clone warms caches / load balancer. Take care not to write 2x to database.

DEVCON2014

Zero Downtime Deployment - Webapp

Turbine   Game

DNS lookup

QA

DNS override

Load balancer

Load balancer

Clone

Webapp

Webapp

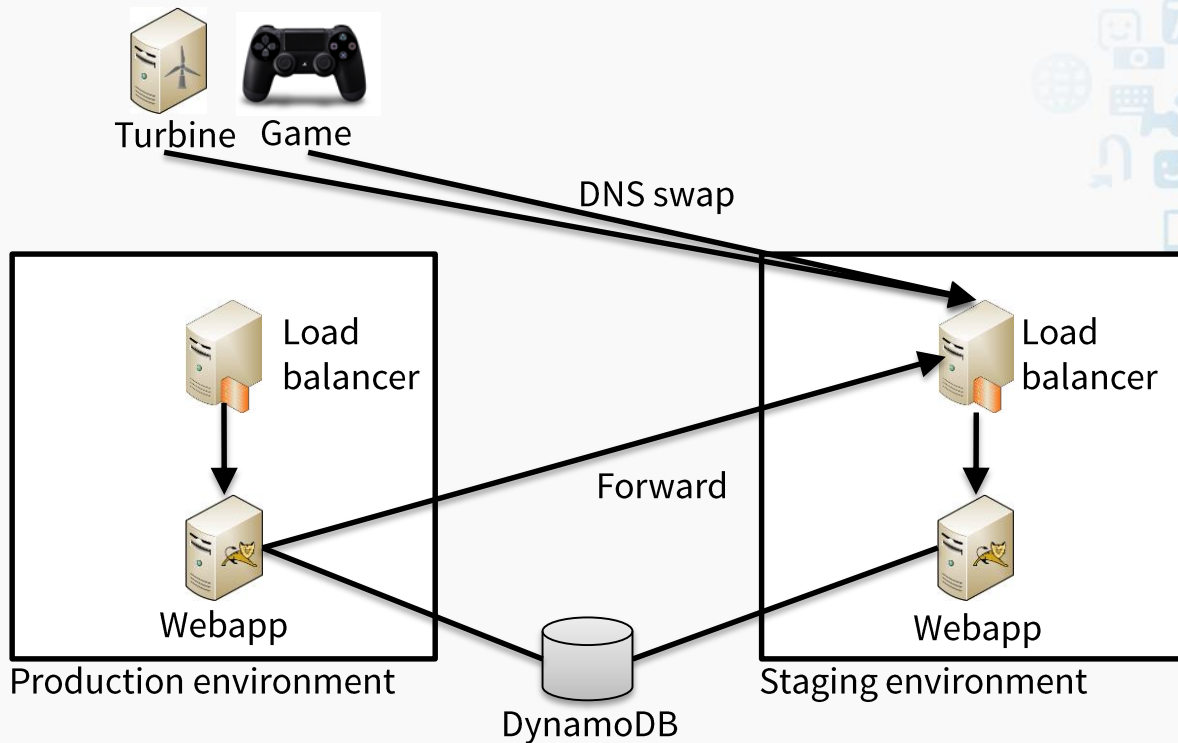Production environment

Staging environment

DynamoDB

- QA can now test the staging environment

Zero Downtime Deployment - Webapp
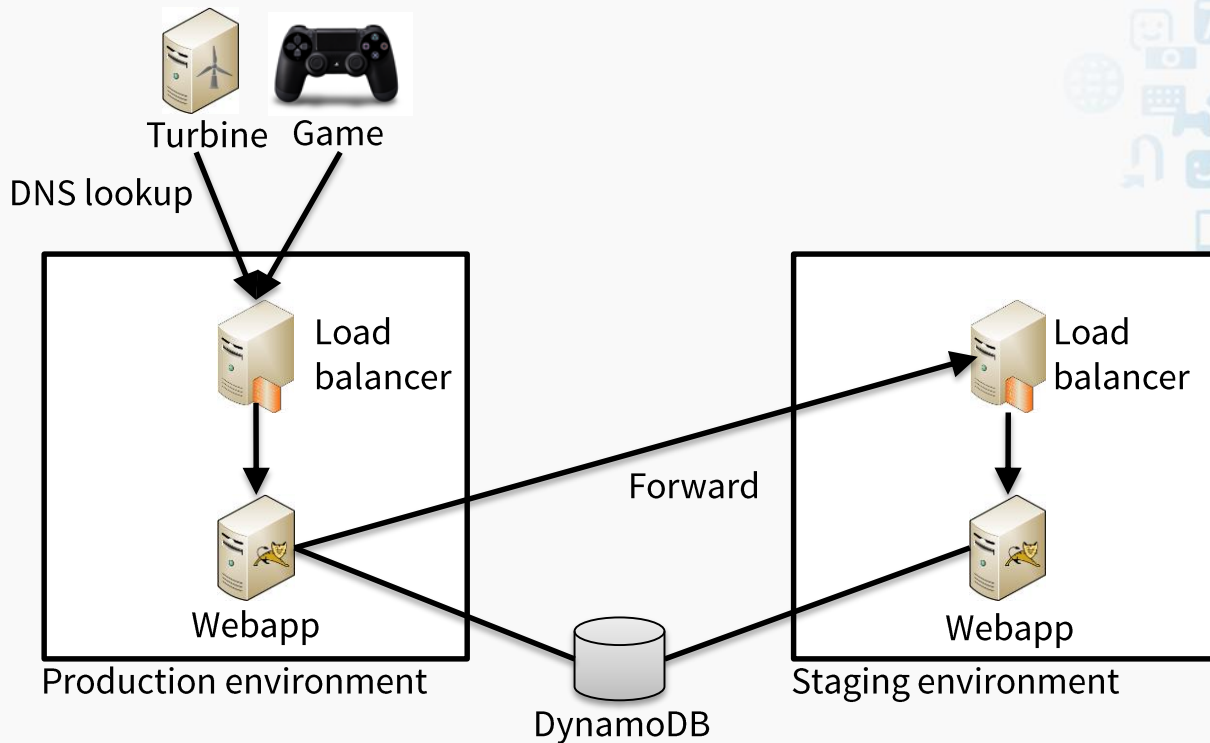
- Forward does not process request on production environment but only on staging environment

Zero Downtime Deployment - Webapp

Turbine   Game

DNS swap

Load balancer

Load balancer

Forward

Webapp

Webapp

Production environment

Staging environment

DynamoDB

Zero Downtime Deployment - Webapp

Turbine   Game

Wait 24h

DNS swap

Load balancer

Load balancer

Forward

Webapp

Webapp

Production environment

Staging environment

DynamoDB

- Some ISP's don't care about DNS record expiry, so you need to wait for 24h

DEVCON2014

Zero Downtime Deployment - Webapp

Turbine Game

DNS swap

Load balancer

Webapp

DynamoDB

Staging environment

- Delete old production environment

Zero Downtime Deployment - Turbine

Games A

Games B

Turbine 1

Turbine 2

announce

Load balancer

Zero Downtime Deployment - Turbine

Games A   Games B

Turbine 1   Turbine 2

maintenance
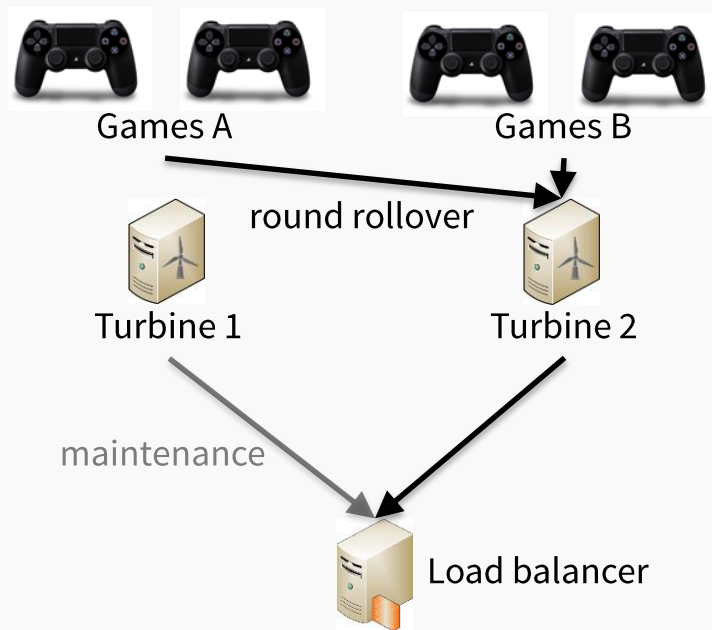
Load balancer
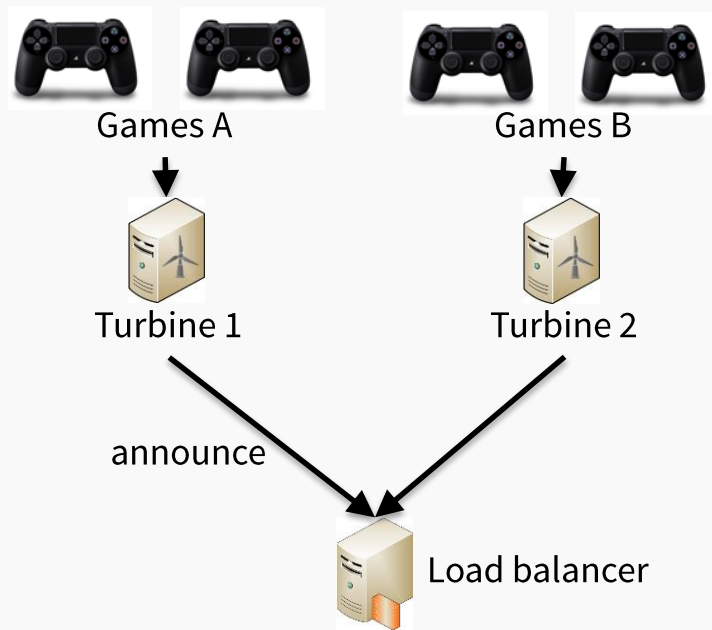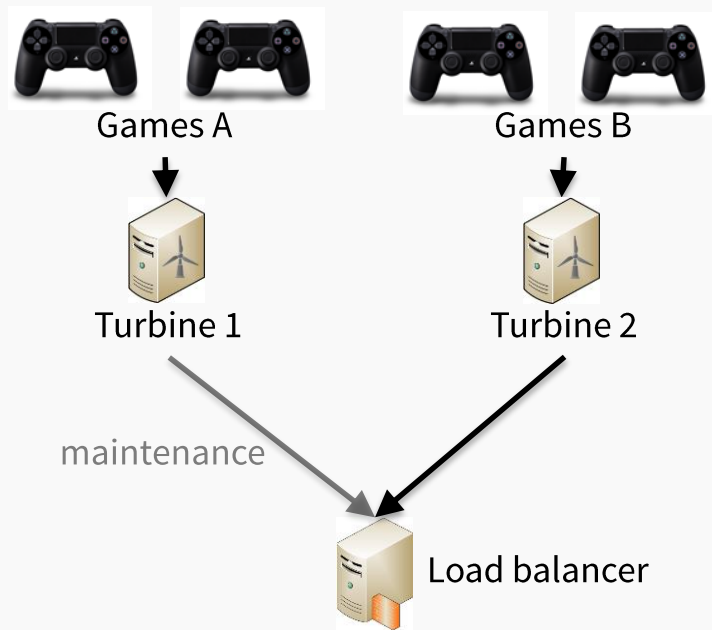
- 50% of machines put in 'maintenance mode'
- Maintenance mode allows current games to keep playing / new players can still join game

Zero Downtime Deployment - Turbine

Games A

Games B

round rollover

Turbine 1

Turbine 2

maintenance

Load balancer

- On round rollover (after approx. 20 minutes) game is moved to new Turbine server

Zero Downtime Deployment - Turbine

Games A   Games B

Turbine 1 (new)   Turbine 2

announce

Load balancer

Zero Downtime Deployment - Turbine

Games A   Games B

Turbine 1 (new)   Turbine 2

announce   maintenance

Load balancer

- Other 50% put in maintenance mode

Zero Downtime Deployment - Turbine

Games A   Games B

round rollover

Turbine 1 (new)   Turbine 2

announce   maintenance

Load balancer

Zero Downtime Deployment - Turbine

Games A   Games B

Turbine 1 (new)   Turbine 2 (new)

announce

Load balancer

Zero Downtime Deployment - Turbine

Games A   Games B

round rollover

Turbine 1 (new)   Turbine 2 (new)

announce

Load balancer

Zero Downtime Deployment - Turbine

Games A   Games B

round rollover

Turbine 1 (new)   Turbine 2 (new)

announce

Load balancer

Do A/B testing!

- Updating 1 server first and let it run for a couple of days
- Check telemetry results / crashes to validate behavior because validating in QA is difficult due to amount of testers needed

Update Without Patching

Update Without Patching

- Patch release cycle very long
  - 1 week for emergency patches
  - 4+ weeks for normal fixes

- Developed server side tweaks system
  - Patching gameplay settings (e.g. weapon stats)
  - Adding collision volumes
  - Changes versioned, each game round uses specific version
  - Update to latest version on round rollover
  - Fixed 50+ exploits!

- Server tweak is simple key / value pair
- Using special notation to patch game settings e.g.
  (ClassName)ObjectName.AttributeName = Value

How Telemetry Helped Fix Issues

Telemetry

Game → Collector (Events) → S3 (Plain text) → Hive (Aggregation) → Redshift → Tableau (Visualization)

Webapp →

Very Powerful!
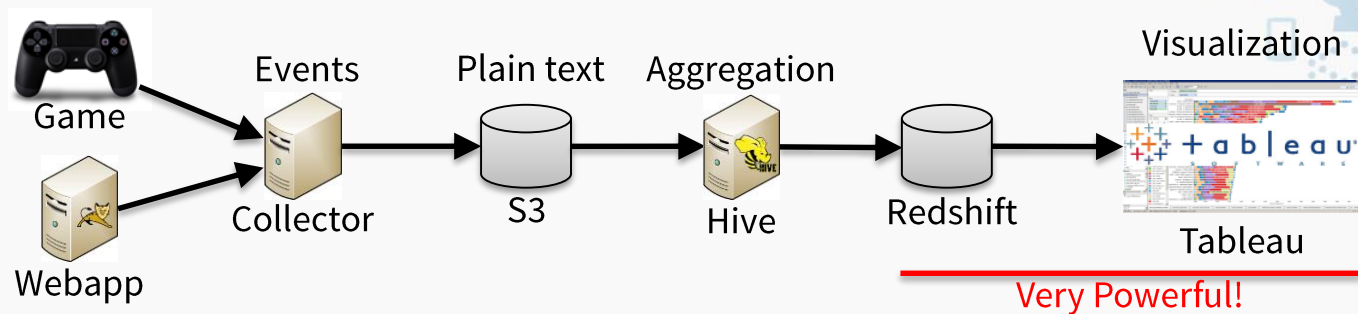
- Can drill down to individual user for complaints

- Example 'Player Spawn' + 'Player Die' event gets aggregated by Hive to 'Player Life' table in Redshift

Errors Per Hour Over Time

PlayStation®Plus check

PSN^{SM} authentication

Turbine fix

Error Name
- E_AC_AGE_RESTRICT
- E_AC_FAILED
- E_AH_DATA_DOWNLOAD
- E_AH_INFO_DOWNLOAD
- E_GM_BAD_GAME_STATE
- E_GM_FULL
- E_GM_OBJ_LOST
- E_GM_SYNC_FAILED
- E_GM_SYNC_FAILED_2
- E_MENU_NOT_AVAILABLE
- E_MMS_MODERATED
- E_MMS_REQUEST_FAILED
- E_NGT_CONNECT
- E_NGT_DECRYPTION
- E_NGT_DISCONNECTED
- E_NGT_KICKED
- E_NGT_SEND
- E_NGT_TIMEOUT
- E_NGT_TOO_LARGE
- E_NGT_VERIFY
- E_NGT_WOULD_BLOCK
- E_NP_AUTH_FAILED
- E_NP_NO_DNS
- E_NPM_INIT
- E_OPM_DOWNLOAD_FAILED
- E_PC_FAILED
- E_PPM_DOWNLOAD_FAILED
- E_PRS_INIT
- E_PS_DOWNLOAD
- E_PS_NO_RESPONSE
- E_SEM_DISCONNECT
- E_SEM_TIMEOUT
- E_SS_DOWNLOAD
- E_SS_VERSION
- E_SS_VERSION_2
- E_STM_GET_SESSION_DATA
- E_STM_REQUEST_FAILED
- E_TM_CONTEXT
- E_TM_REGISTER
- E_TM_UNLOCK_STATE
- E_WA_INIT

DEVCON2014
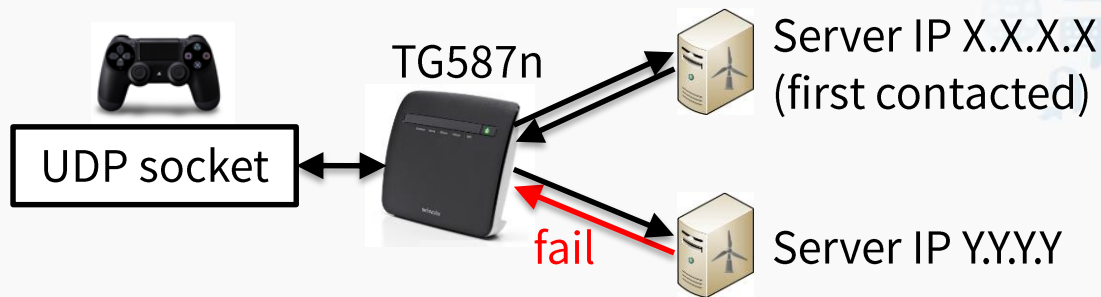
- PlayStation®Plus Check = check to see if you have purchased
  PlayStation®Plus and can play online

Errors Per Hour Per Router

Technicolor modem triggers errors on round roll over

Technicolor TG587n

TG587n

UDP socket

Server IP X.X.X.X
(first contacted)

fail

Server IP Y.Y.Y.Y
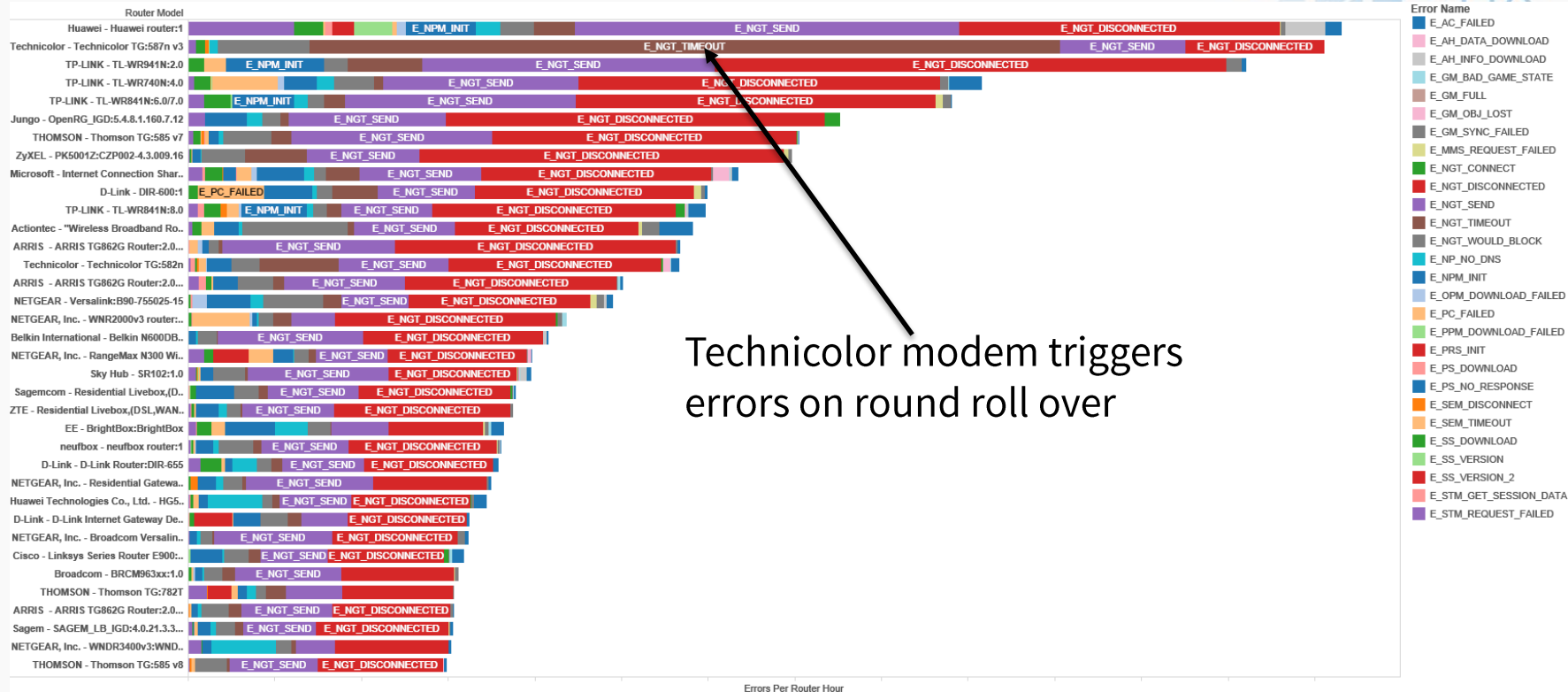
- 1 UDP socket fails to send to multiple IP's!
- Need to allocate UDP socket per IP

And Then We Released Patch 1.10
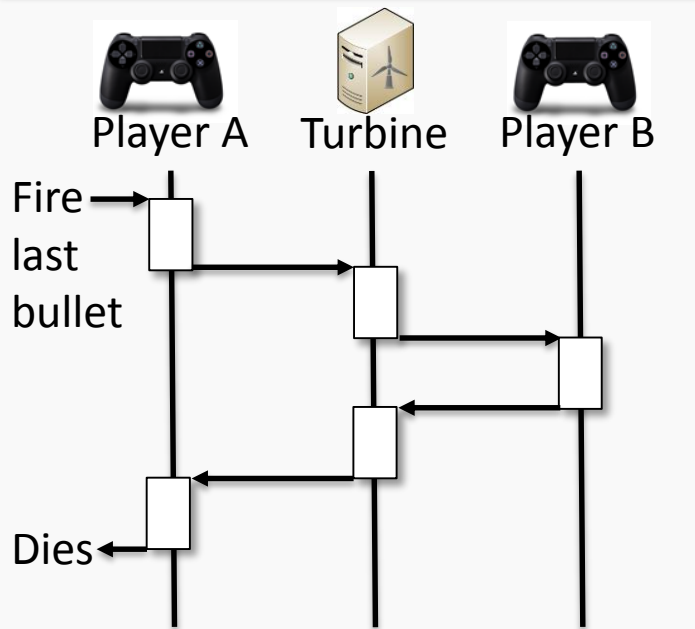
FIX THE LAGGING!!!!!

I have no idea what you need to do to fix the issue but it is sooooo much worse since the 1.10 patch came out. If its not fixed very soon im going to trade the game!!

173 votes

Vote

danieladkins7 shared this idea · February 19, 2014 · Flag idea as inappropriate...

- Community complained about 5+ second lag
- We didn't see it, couldn't measure it

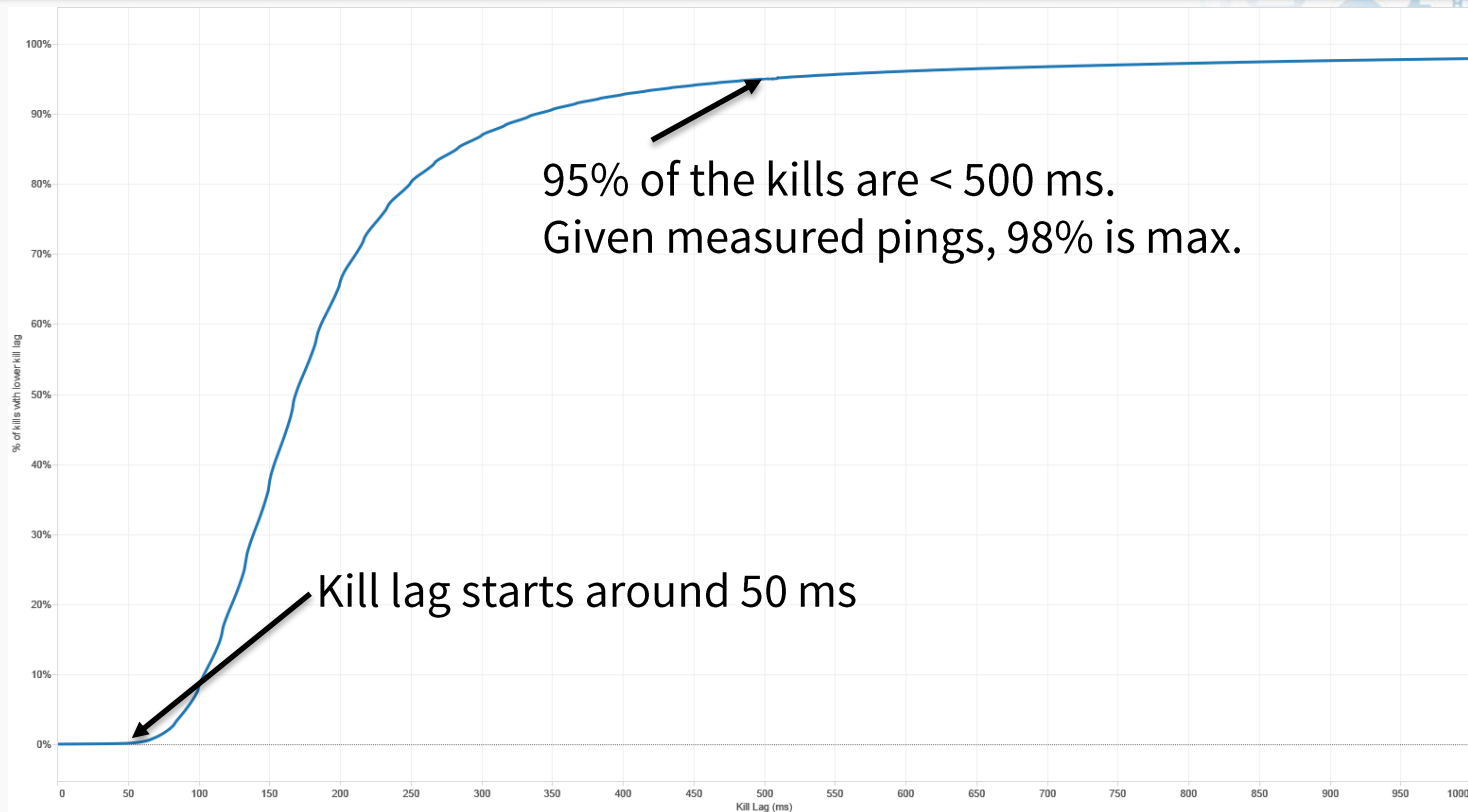# Key Metric - Kill Lag

Kill Lag = Ping$_A$ + Ping$_B$ + Processing

- Server & client run at 60 Hz
  - Processing ≈ 5 * 0.5 * 1/60 ≈ 40 ms
- Best ping ≈ 5 ms
  - PingA + PingB ≈ 10 ms
- Best Kill Lag ≈ 50 ms
- Consider laggy kill > 500 ms

Kill Lag Measured

95% of the kills are < 500 ms.
Given measured pings, 98% is max.

Kill lag starts around 50 ms

- For every kill you make we send telemetry on what the kill lag was
- Graph shows percentage of kills that took lower than X-axis ms

DEVCON2014

Laggy Kills Over Time

- US-East gateway failure
- Backup gateway high packet loss
- libRUDP resend bug

libRUDP fix

Switch to Amazon

Release patch 1.10

Switch back

A/B test
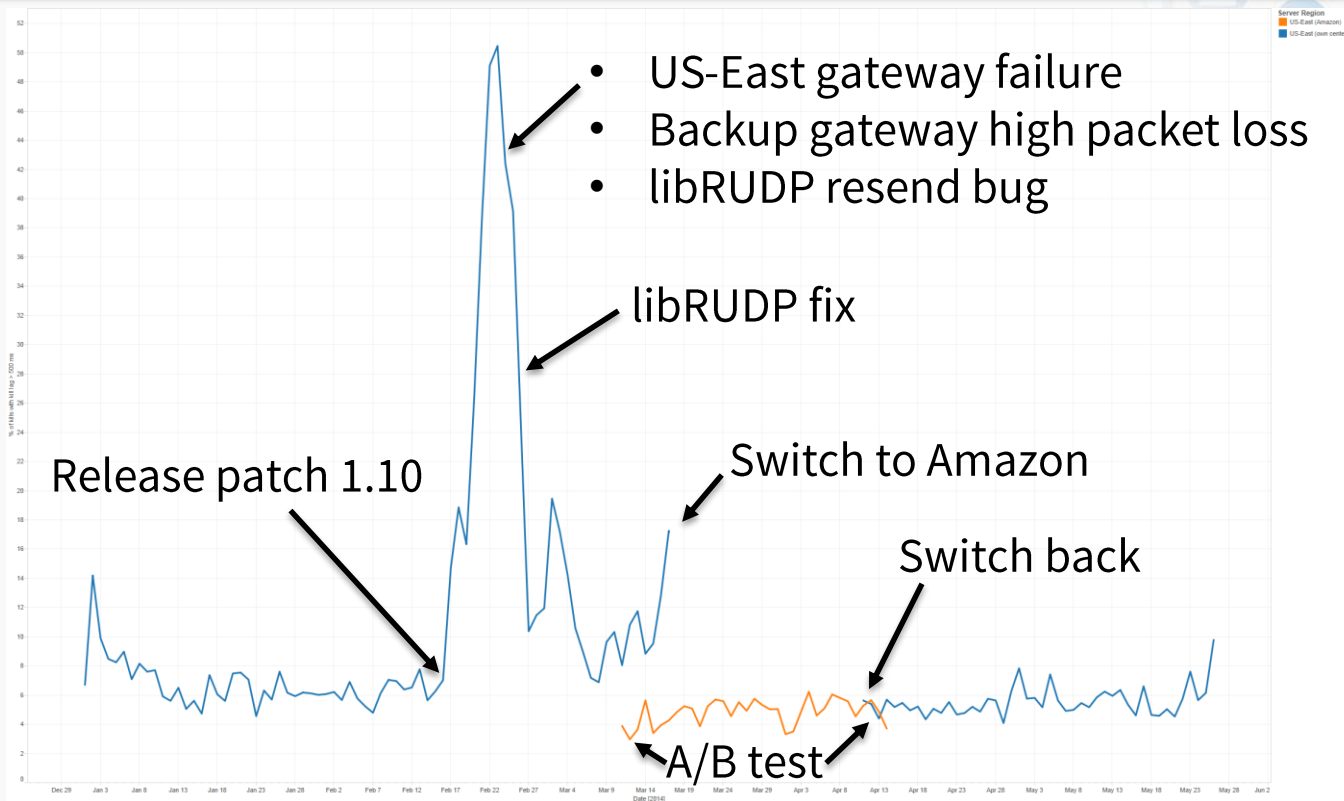
- Vertical axis is percentage of laggy kills
- Graph shows US-East only (other regions were unaffected – which is why we didn't see it)

Amazon isn't perfect either

One server triggers
a lot of network errors

Terminate, and it's
automagically replaced
by a working instance
☺

Error Count

Mar 15 Mar 16 Mar 17 Mar 18 Mar 19 Mar 20 Mar 21 Mar 22 Mar 23 Mar 24 Mar 25 Mar 26 Mar 27 Mar 28 Mar 29 Mar 30 Mar 31 Apr 1

Date [2014]
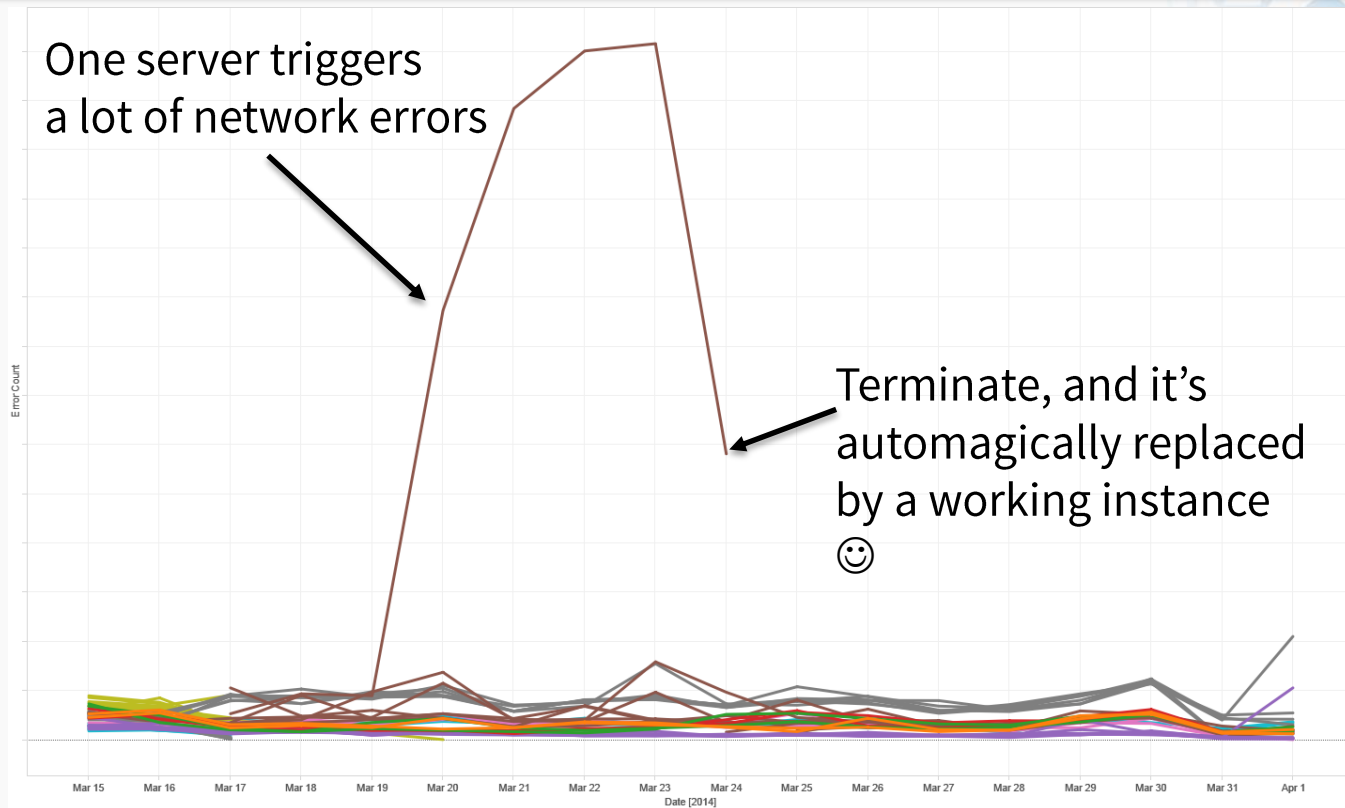
- Vertical axis shows error count per server
- Color is region

Average Ping Between Regions

Player Region	Server Region							
	Tokyo	London	US-East	US-West	Sydney	Sao Paulo	Istanbul	Singapore
Tokyo	**49.6**	351.0	220.4	161.7	170.0	397.5	354.0	119.1
London	323.1	**73.9**	154.5	214.8	395.9	296.7	129.6	312.2
US-East	232.6	152.7	**78.3**	119.7	271.2	206.3	217.7	292.6
US-West	203.8	200.5	134.9	**90.8**	240.9	249.5	281.6	263.1
Sydney	254.4	411.5	319.4	245.1	**93.3**	446.2	448.0	239.7
Sao Paulo	369.6	298.8	225.6	255.7	421.9	105.1	373.8	437.0
Istanbul	380.1	179.2	221.1	273.7	449.0	350.4	136.0	359.8
Singapore	245.6	376.1	331.3	301.0	295.1	474.8	401.1	153.8

- No region where next closest region < 100 ms
- Matchmaking tries to avoid cross region!
  - Smaller regions / warzones don't have enough players
  - Friends may want to play cross region
  - 98%+ success in large, 80% in smallest region

Telemetry - Conclusion

- You NEED telemetry to fix your game
  - Difficult to draw conclusions from forums
  - Previous examples very difficult to find

- Consider key metrics
  - For us 'Kill Lag'
  - Tie into monitoring

- Monitoring = real time check that number of laggy kills stays constant, if it increases we get called