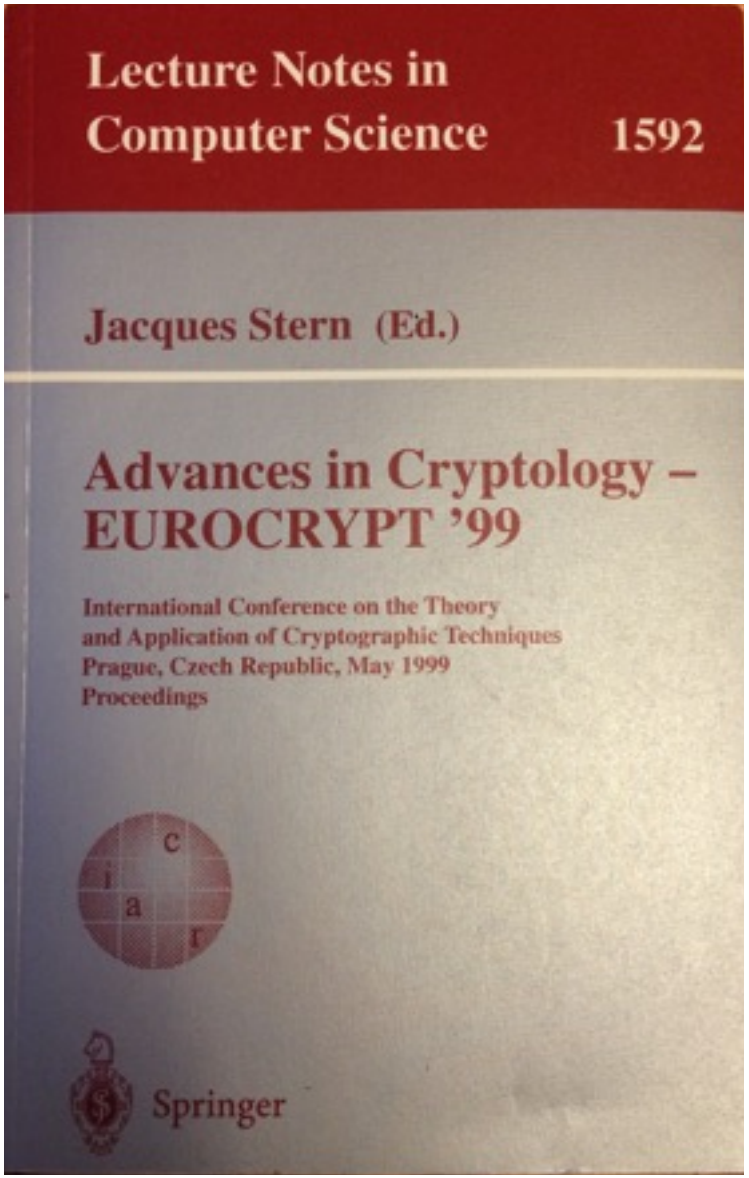
talko

Erlang in the Cloud

Talko Service Architecture

Ransom Richardson
ransomr@talko.com
@ransomr
https://medium.com/@ransomr

On the Current Composition of Zero-Knowledge Proofs



talko

Erlang in the Cloud

Intro
- Talko
- erlcloud

The Cloud is Different
- Reliability
- Deployment
- Service State

More Erlang
- Redis to Bz

Looking Forward
- AWS Lambda

Questions

“The cloud is fundamentally different”

https://www.youtube.com/watch?v=JIQETrFC_SQ

Mobile Team Communications

anytime, anywhere,
for the new style of work.

Tap to Talk
Conferencing meets messaging. On any network, or even offline.

Tap to Show
Hi-detail photos sent instantly while talking.

Always In-Sync
Flag, tag, bookmark, share to reduce anxiety, increase transparency.

Today: Calls & 'rich conversations'; iOS
Soon: Calls, 'rich conversations' & messaging; iOS, Web, Android
Next: Managed deployments & administration, integration

talko

Talko Service

- VoIP
- Audio Recording
- Registration
- Contact matching
- Group and call creation and membership
- Texts, photos, tags
- Notifications
- User awareness
- …

Service Team

Big Service Small Team

Optimize for Developer Productivity

erlcloud

https://github.com/gleber/erlcloud

- Partial AWS API Support
- Many contributors
- Implement needed APIs
- Contributions welcome!


bwbuchanan


ptrakhtman


gleber


kevinmontuori


fogfish


elbrujohalcon

Reliability

talko

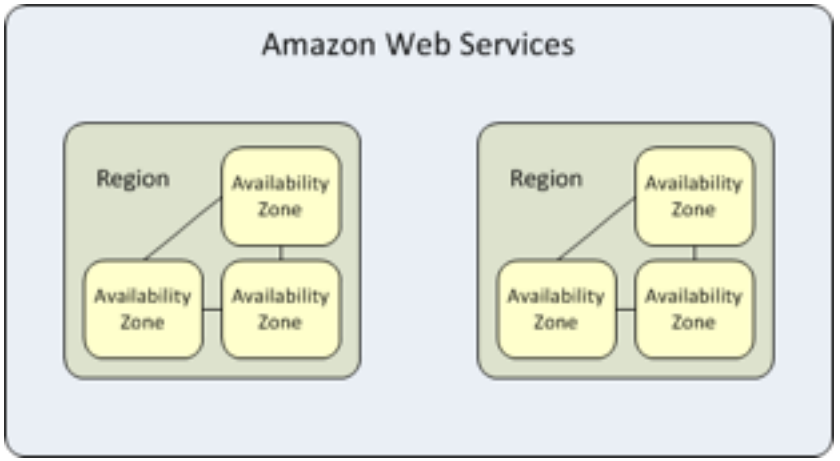
Reliability

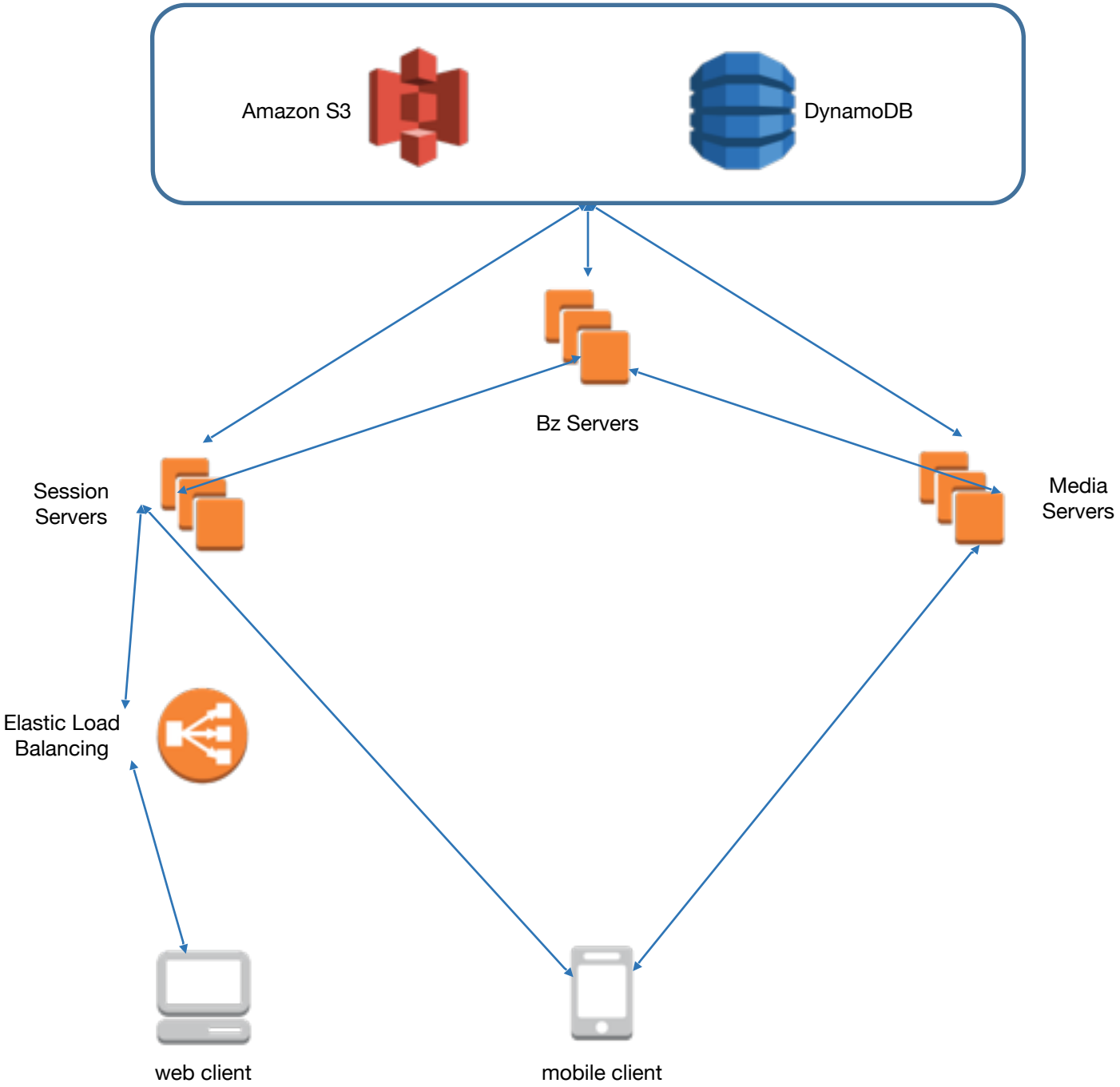
- The cloud is unreliable – expect failures
- Let it Crash!

Erlang	AWS
process	*instance*
supervisor	*Auto Scale Group*

Reliability

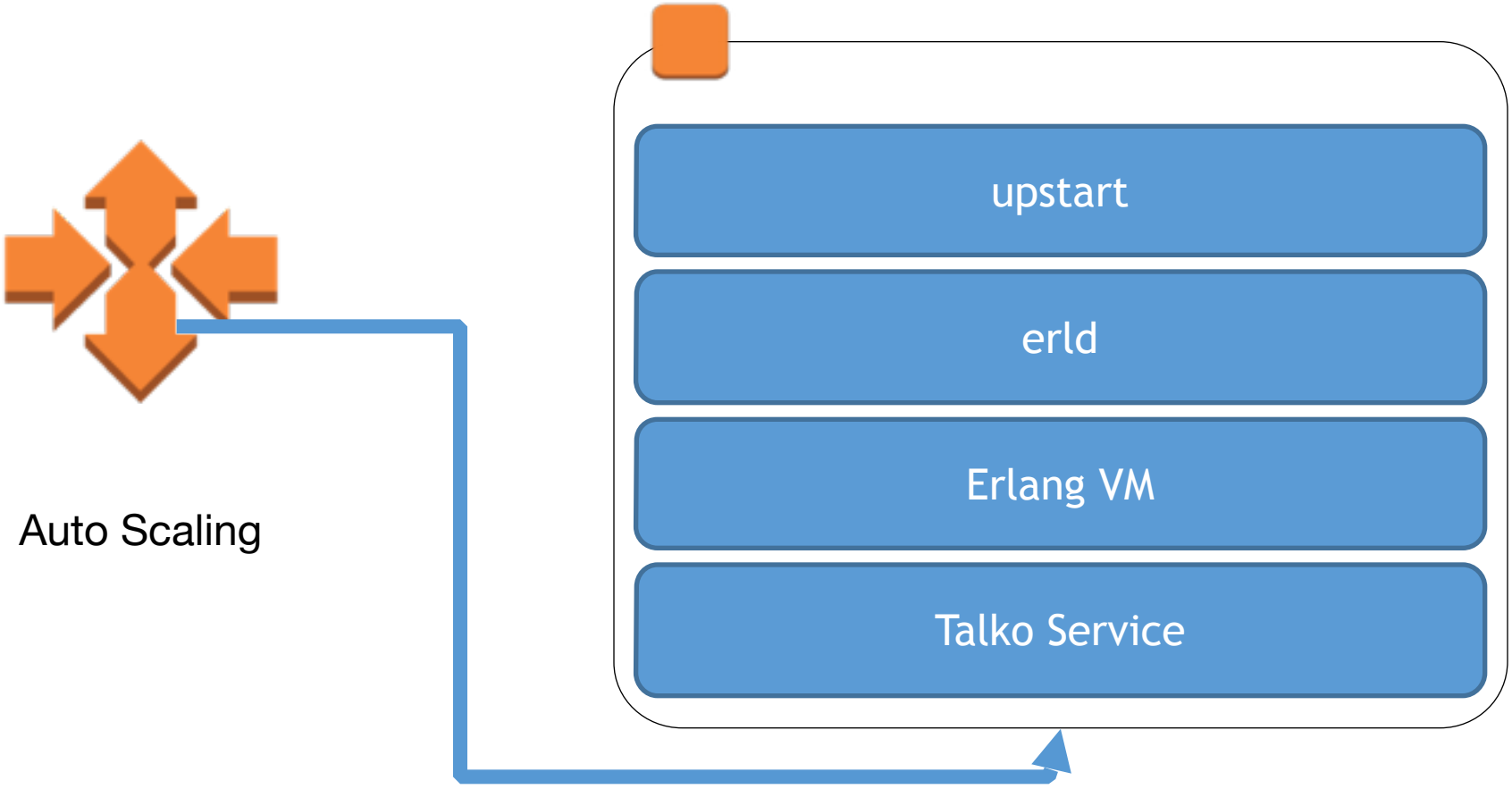
- More Than One Instance
- Different Availability Zones
- Enough extra capacity to absorb failures
- …and load spike that is caused by failures




Amazon S3
DynamoDB
Bz Servers
Session Servers
Media Servers
Elastic Load Balancing
web client
mobile client

talko

Reliability



Deployment

talko

Deployment

Two approaches:

1. Upgrade existing instances
2. Deploy all new instances

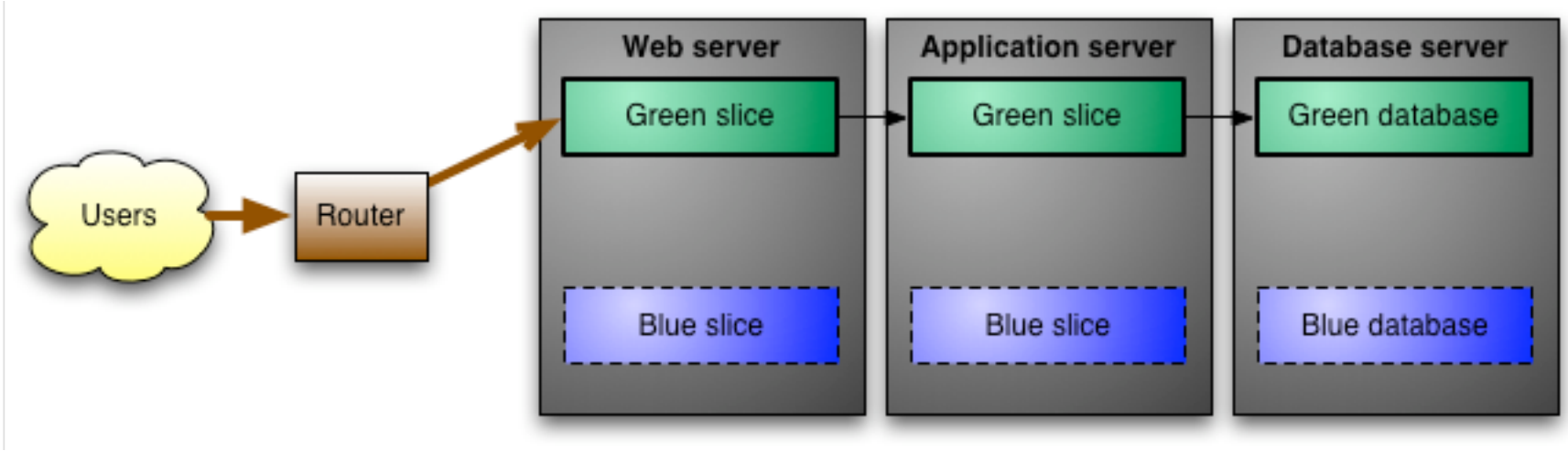
talko

Deployment – New Instances

- Easy rollback
- Need to be able to deploy quickly for reliability
- Tests reliability code path
- Known instance state
- Security

Blue/Green Deployment

- Blue service running
- Deploy Green service with new code
- Switch all traffic to Green
- Take down Blue service



http://martinfowler.com/bliki/BlueGreenDeployment.html

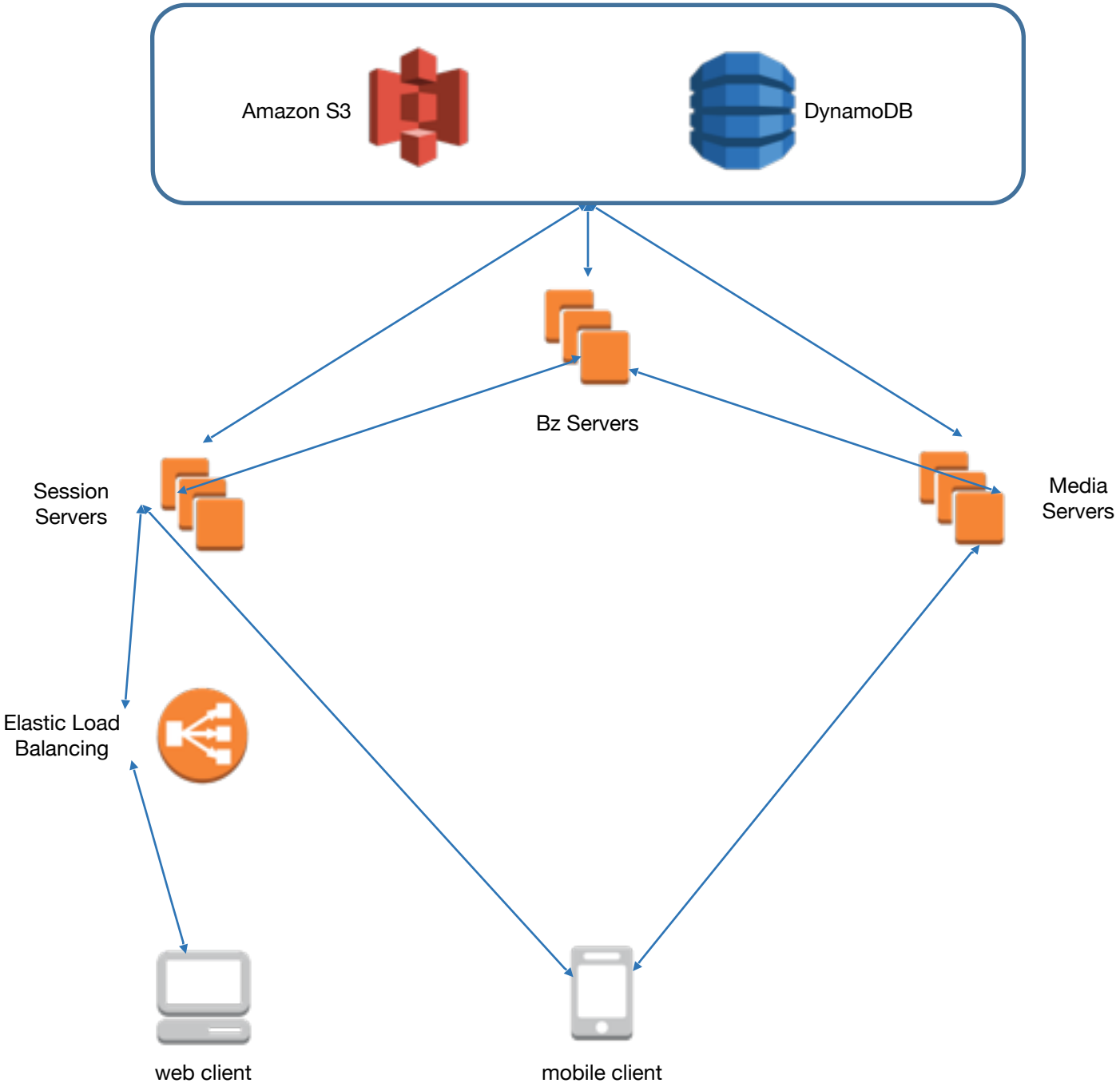
talko

Talko Deployment

State would be lost on blue/green switch
How do we preserve in-progress calls?

Modified approach:

- Deploy new instances
- Test
- Close old instances
- Wait for load to move to new instances
- Take down old instances


Amazon S3
DynamoDB
Bz Servers
Session Servers
Media Servers
Elastic Load Balancing
web client
mobile client

talko

Service State

Service State

Stateless

- Event handling code only
- Easier reliability, scaling and deployment
- Limited functionality

Stateful (gen_server)

- State and message handling code
- Harder reliability, scaling and deployment
- Full functionality

talko

Talko Services are Stateful

- In Progress Calls
- Persistent Connections
- Cross-Client Communications

How do we manage this state so we can build a reliable, scalable and deployable service?

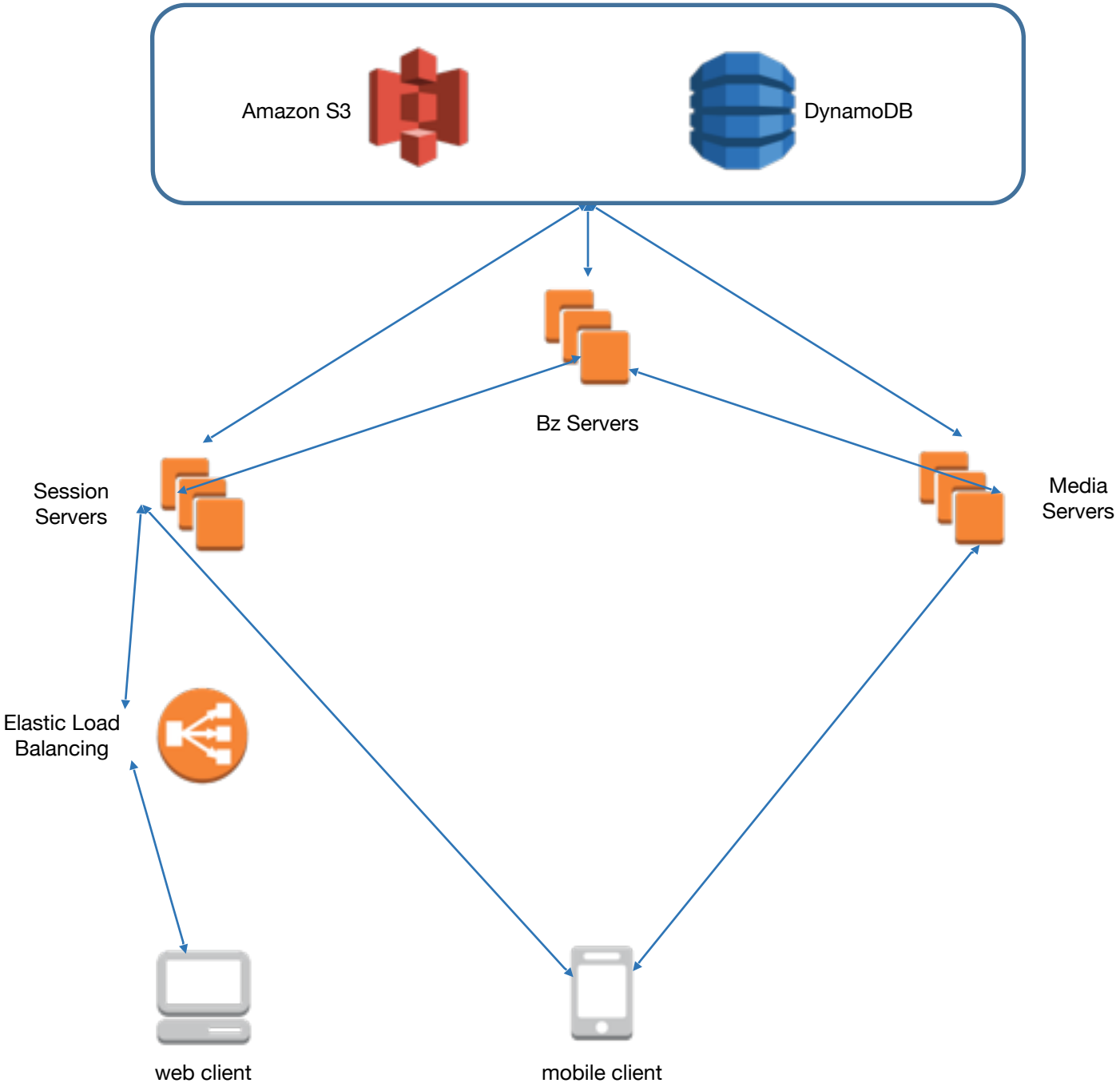
talko

Ephemeral State

- In Progress Calls
- Persistent Connections
- Cross-Client Communications

This state is ephemeral:

- No Redundancy
- Losing it results in slight (< 30s duration) user interruption


Amazon S3
DynamoDB
Bz Servers
Session Servers
Media Servers
Elastic Load Balancing
web client
mobile client

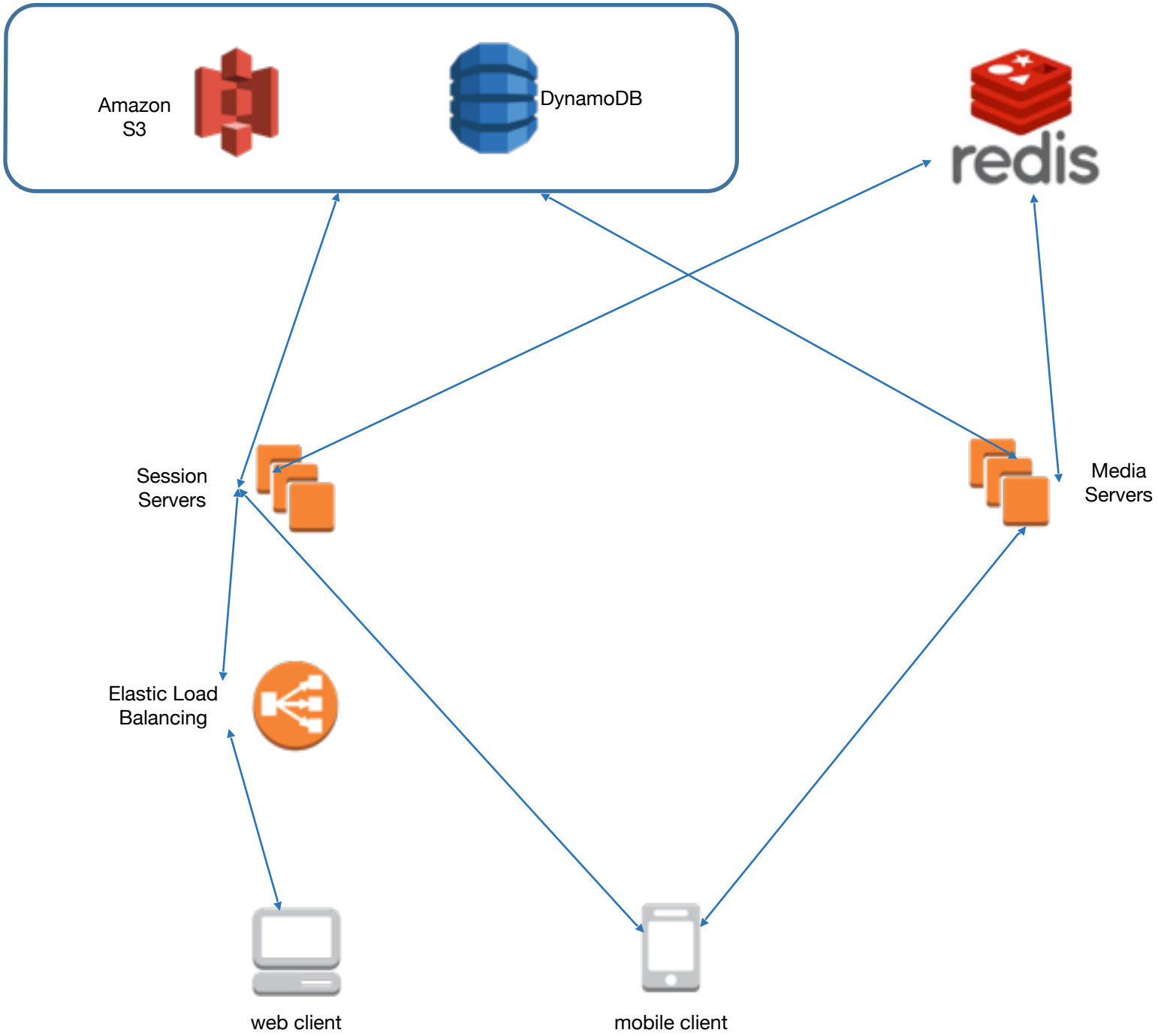
talko

Ephemeral State Challenges

- Possibility of user impact
- Need to detect and heal outages very quickly
- Takes time for load to shift to new instances
- Potential for load spikes on loss of state

More Erlang

Redis to Bz

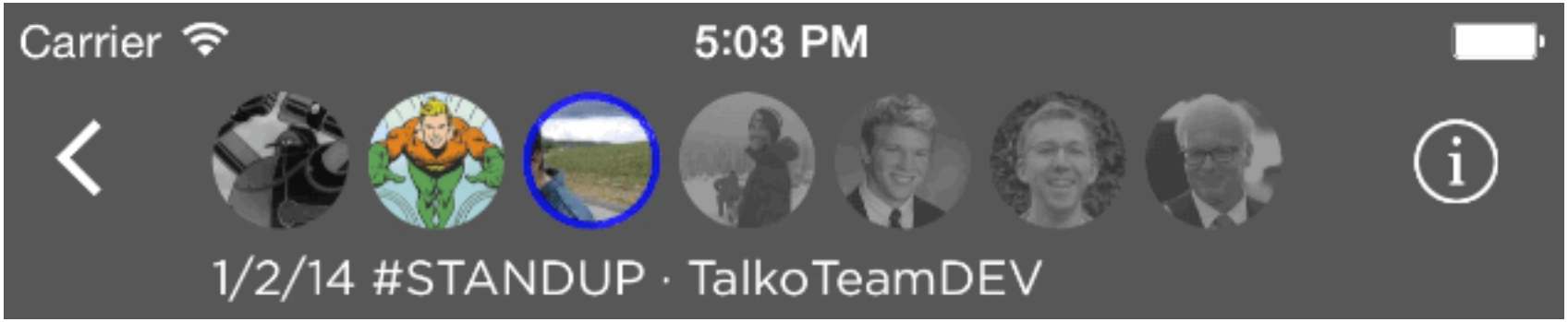

Amazon
S3
DynamoDB
redis
Session
Servers
Media
Servers
Elastic Load
Balancing
web client
mobile client

Redis to Erlang

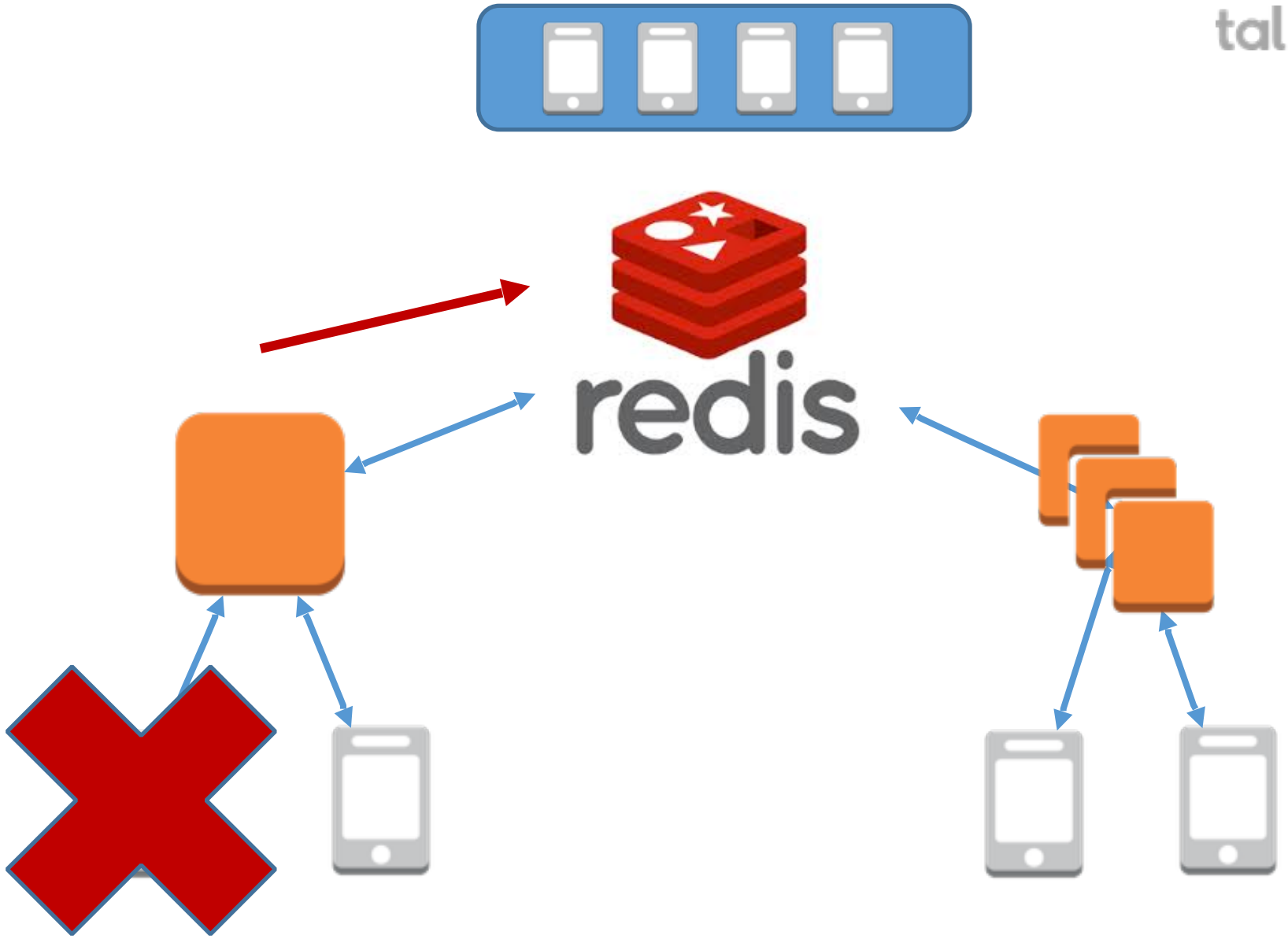
Redis for all cross-instance communication:

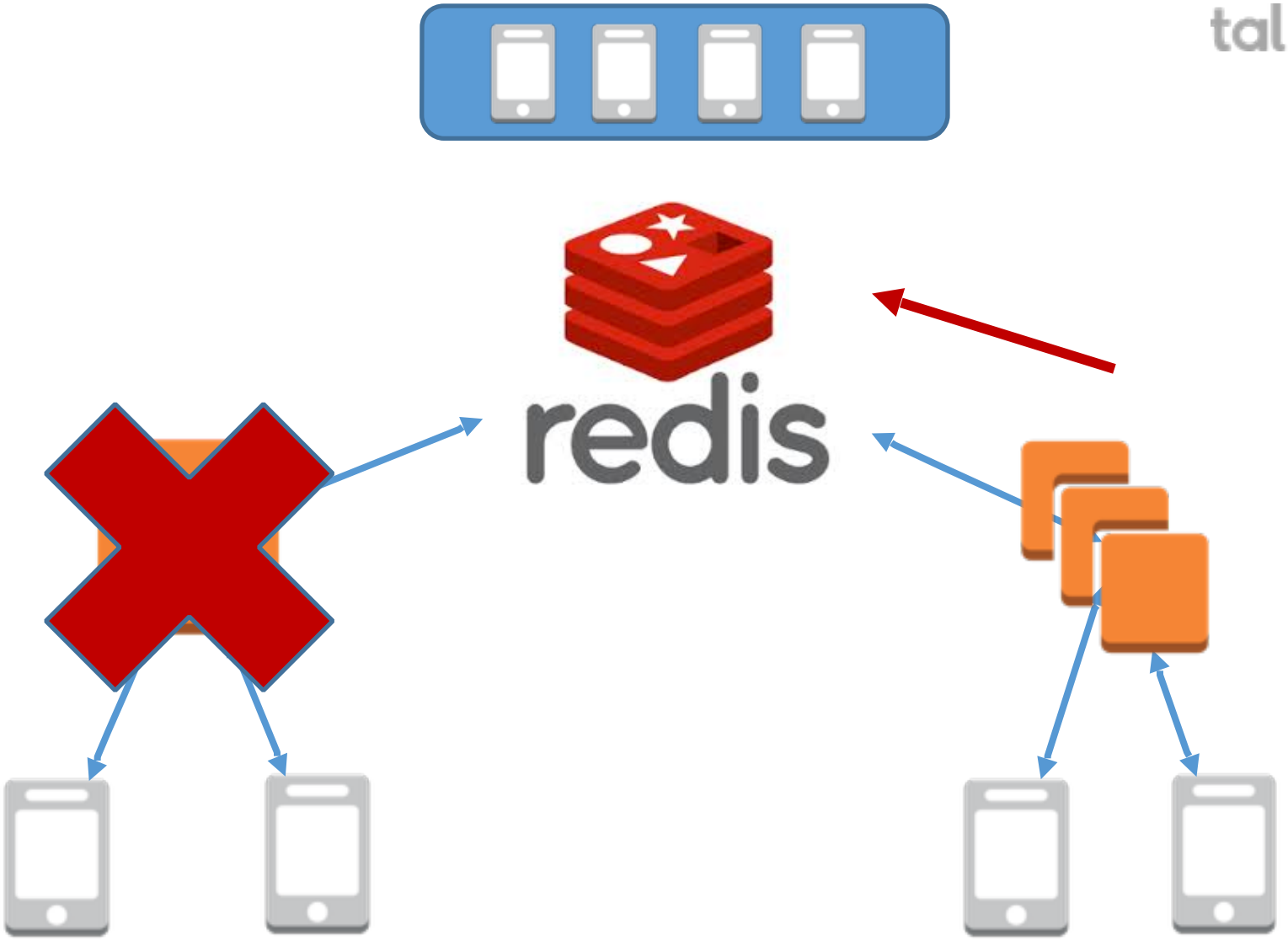
- Pub/Sub
- Ephemeral State

User Awareness



talko




redis

Redis Challenges

- No single server that can run code on shared data
- Distributed Concurrent Updates to Shared State
- Complex algorithms and complex code
- Duplicated Data

Bz Server

- Erlang server
- gen_server per shared object (account, call)
- Much simpler code – awareness expiration is trivial
- Faster (!?!?)

Looking Forward

AWS Lambda

Just write a function

Don’t worry about:

1. Reliability
2. Deployment
3. Machines/instances

But must be Stateless

talko

Questions

Distributed Erlang

We don't use it.

Should we?

Concerns:

- Clusters that cross AZs or Regions
- Speed of detecting/ healing failures
- Fully interconnected

talko

Questions?

Ransom Richardson
ransomr@talko.com
@ransomr
https://medium.com/@ransomr

Backup Slides

talko

Choosing Erlang

- https://medium.com/talko-team-talk-share-do/we-learned-us-some-erlang-ef06bd44e3c2

- Concurrency Model
- Error Handling
- Reputation
- Domain Fit

Erlang Features

We Use

- OTP
- Dialyzer
- ETS

We Don't Use

- Distributed Erlang
- Releases/Upgrades
- Mnesia

talko

Key Dependencies

- erlcloud
- cowboy & ranch
- jsx
- lager
- gproc
- rebar