Bringing Riak to the Mobile Platform

Kresten Krab Thorup
Hacker

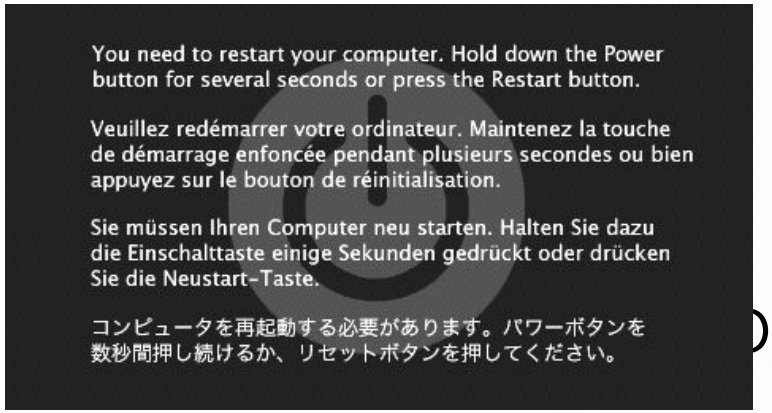
@drkrab

About the Speaker

- **Language Geek** Emacs/TeX Hacker, Objective C, NeXT, GNU Compiled Java, Java Generics, Ph.D., Erlang/Erjang
- **Developer** J2EE AppServer, CORBA/RMI, XA-TM
- **Trifork CTO** QCon & GOTO Conferences Technology Adoption; Riak; Erlang

Outline

- **Riak** and
- **Protocol** onization
- **BucketDB**, a Riak clients for mobile



@drkrab

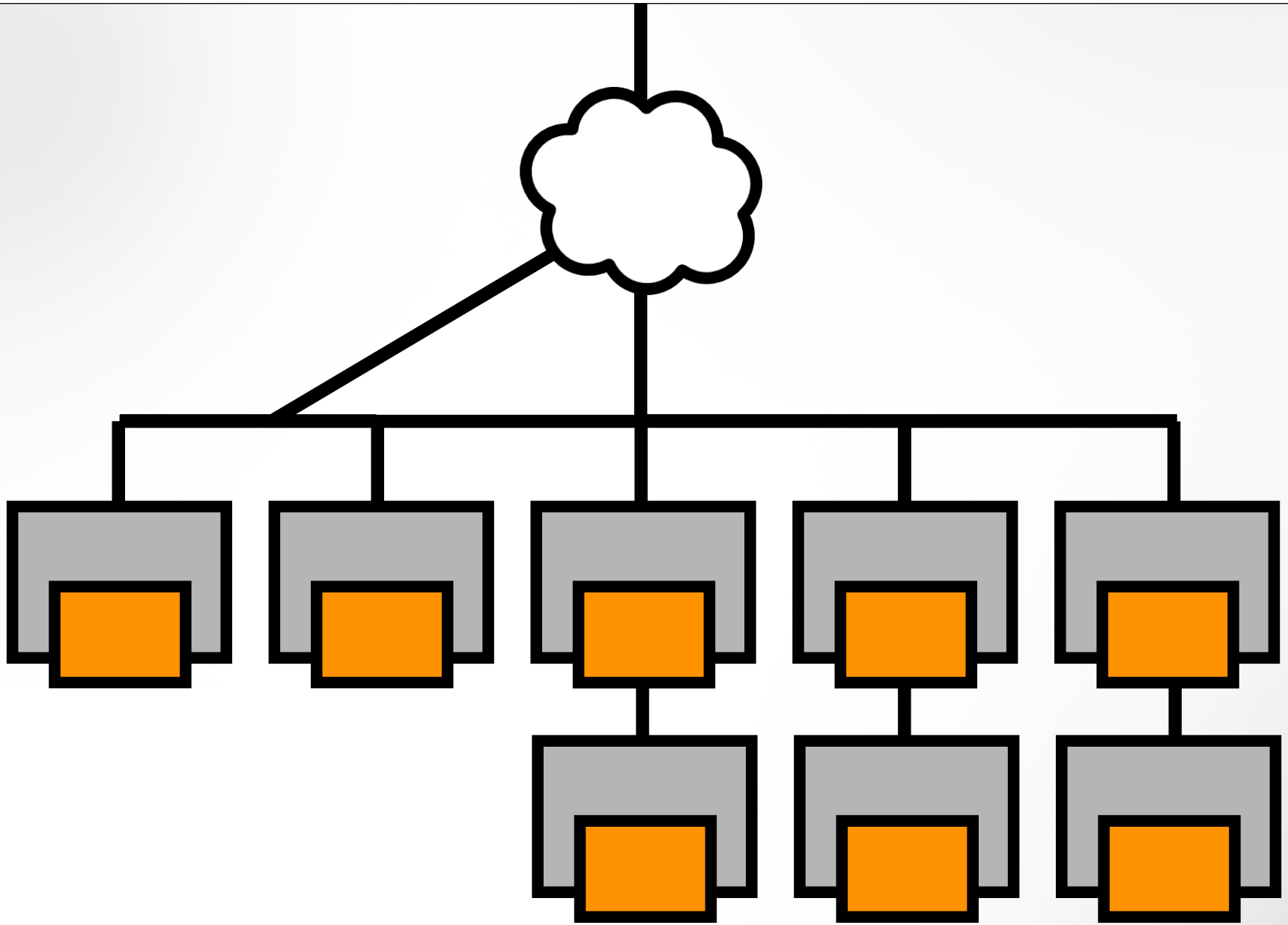
Outline

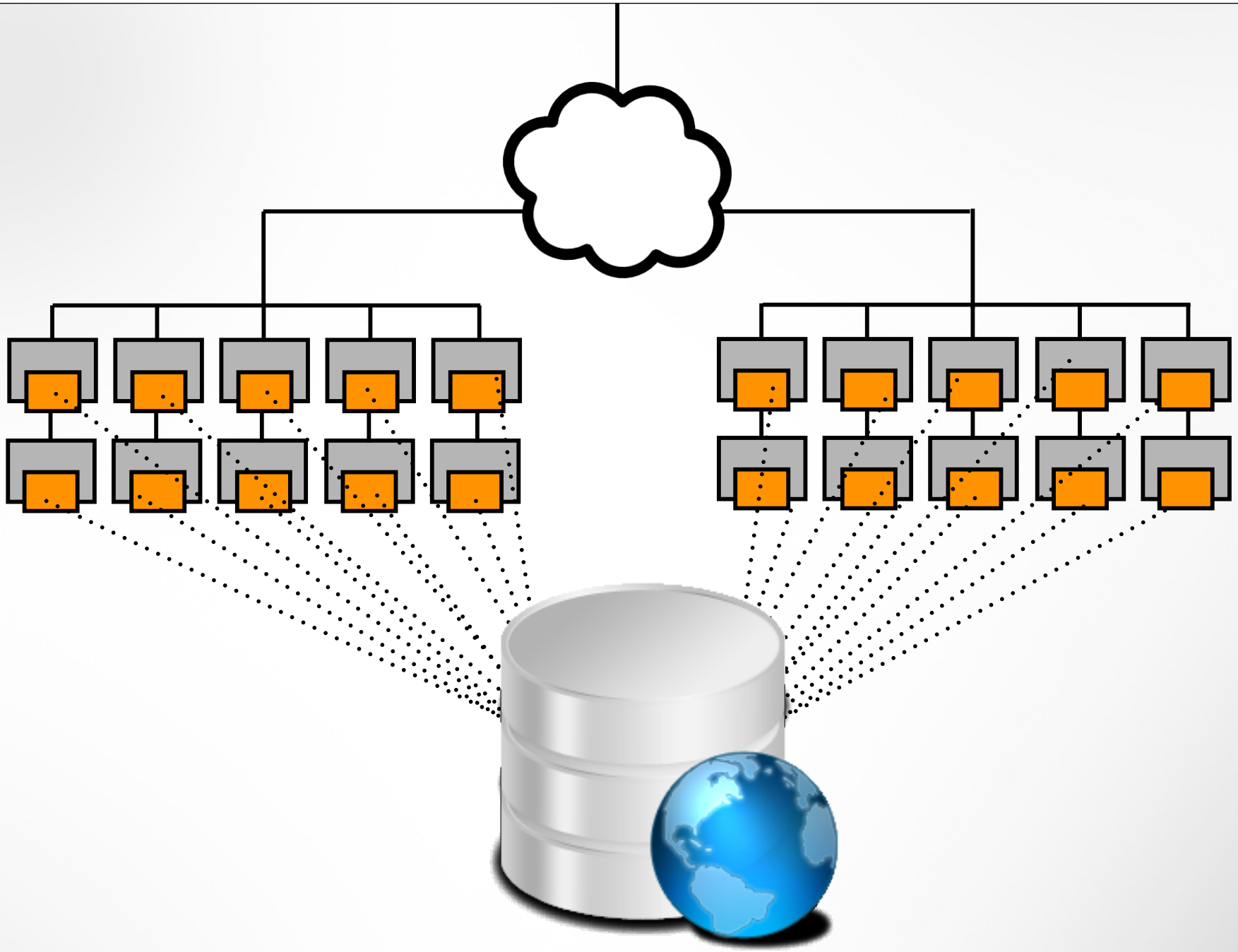
- **Riak** and it's Data Model
- **Protocol** for Key/Value synchronization
- **BucketDB**, a Riak clients for mobile

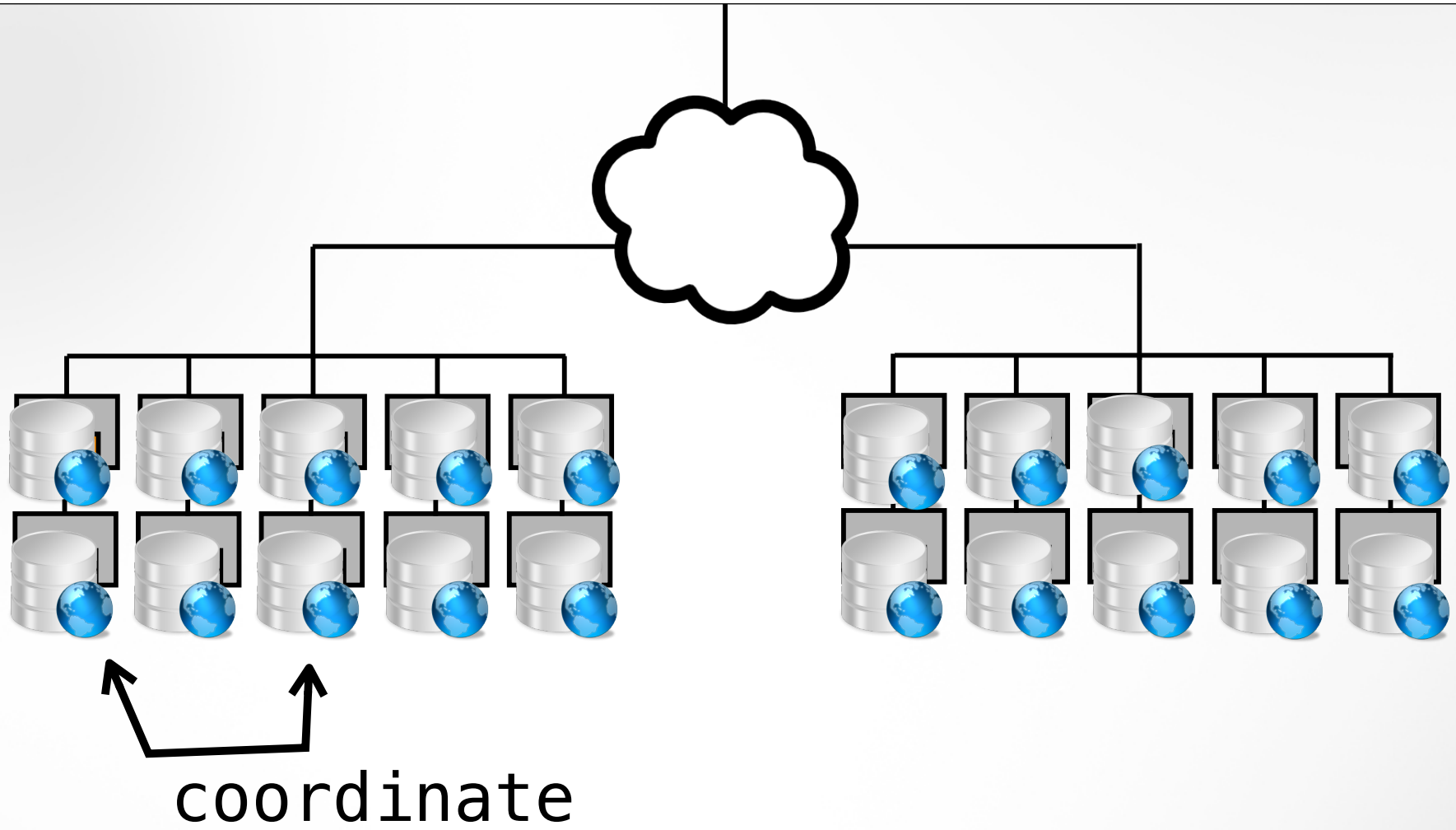
@drkrab

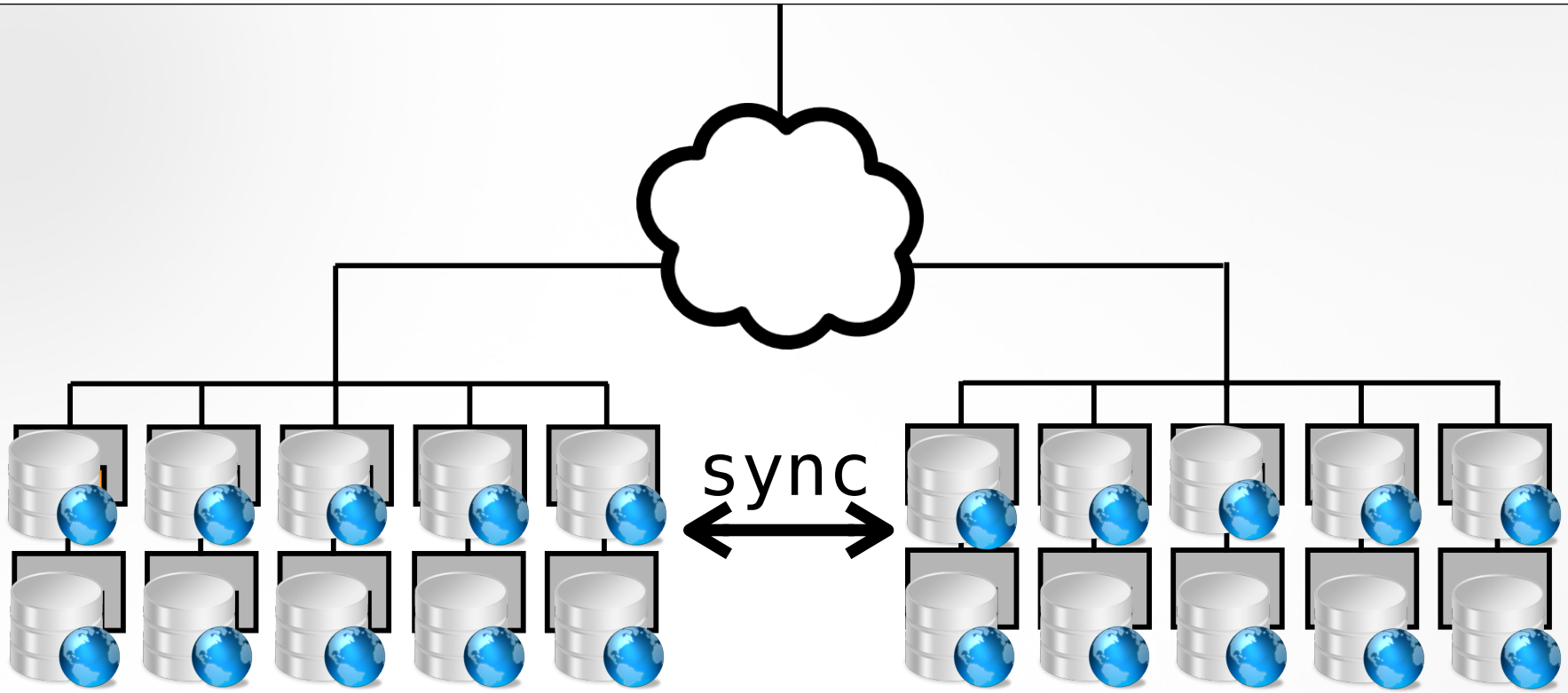
What is Riak, anyway?

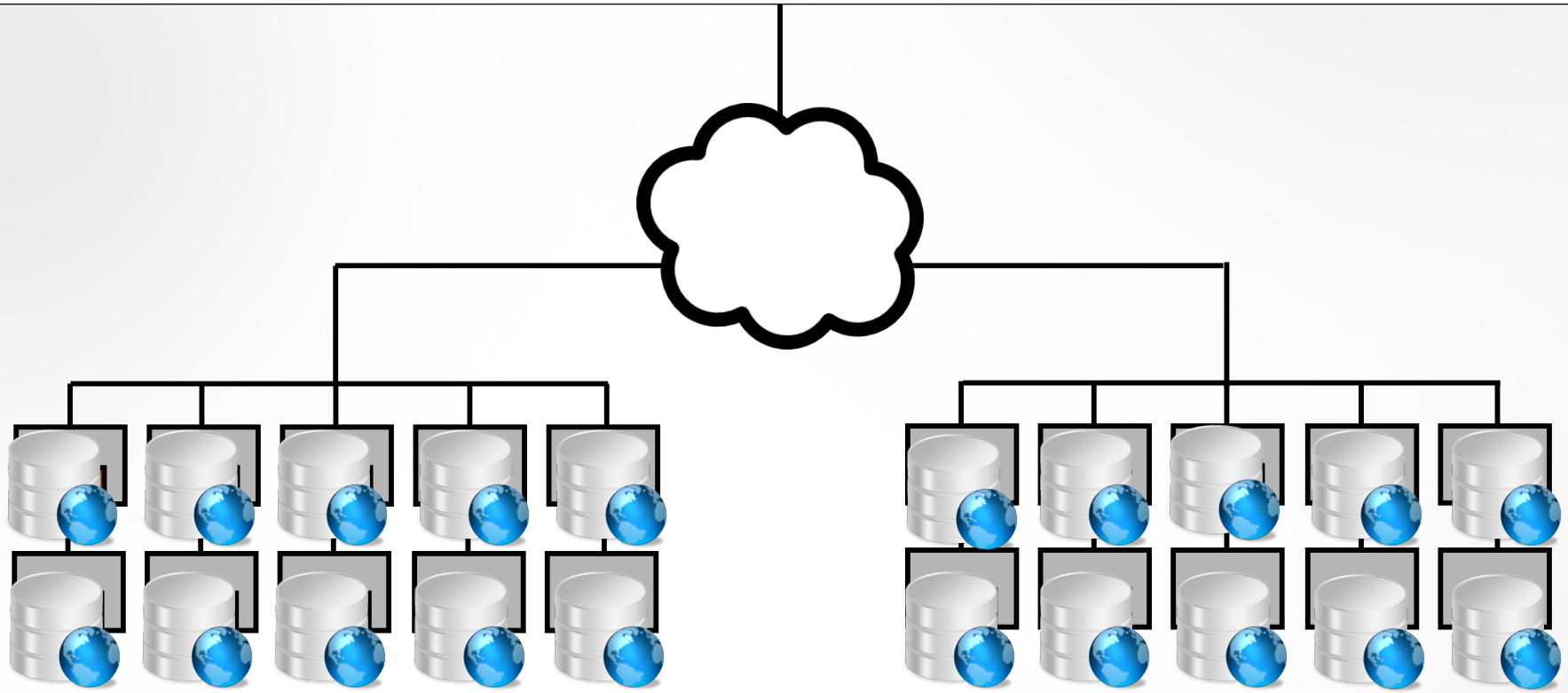
@drkrab






coordinate

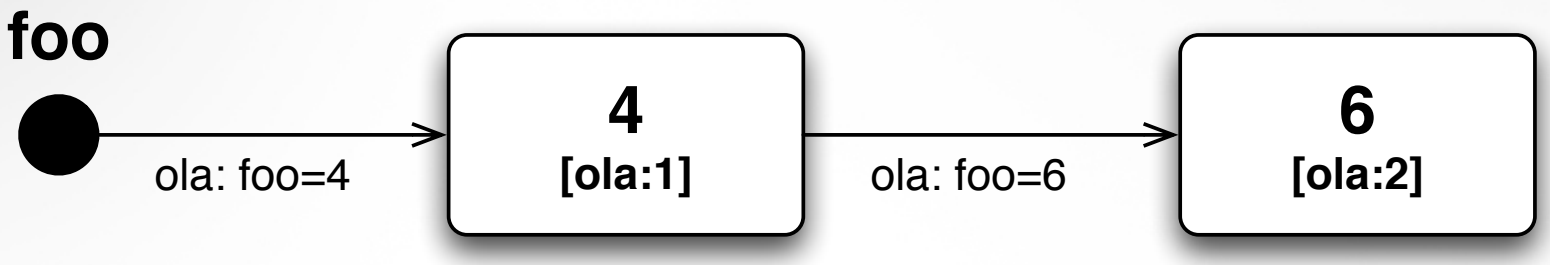

sync

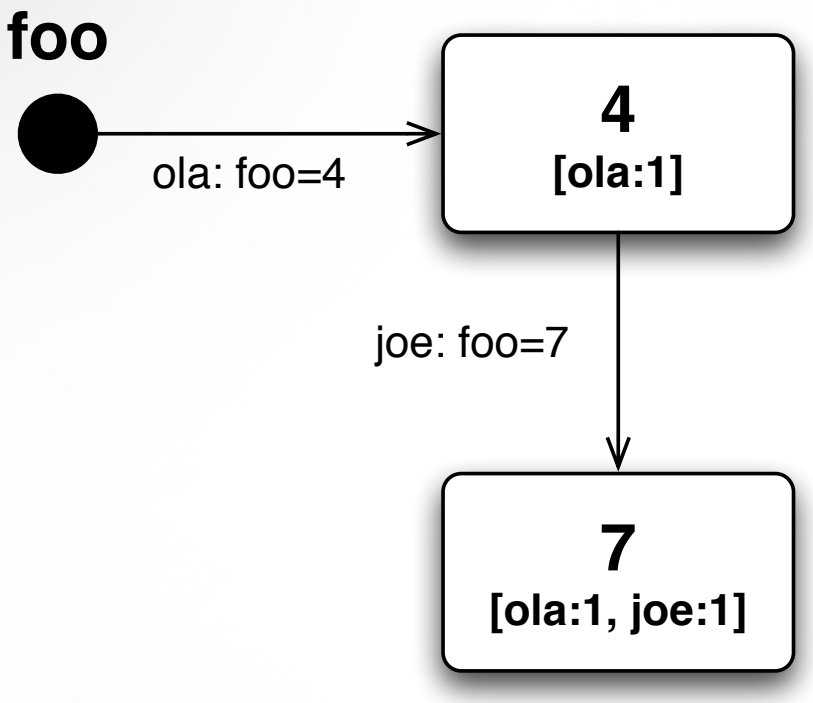


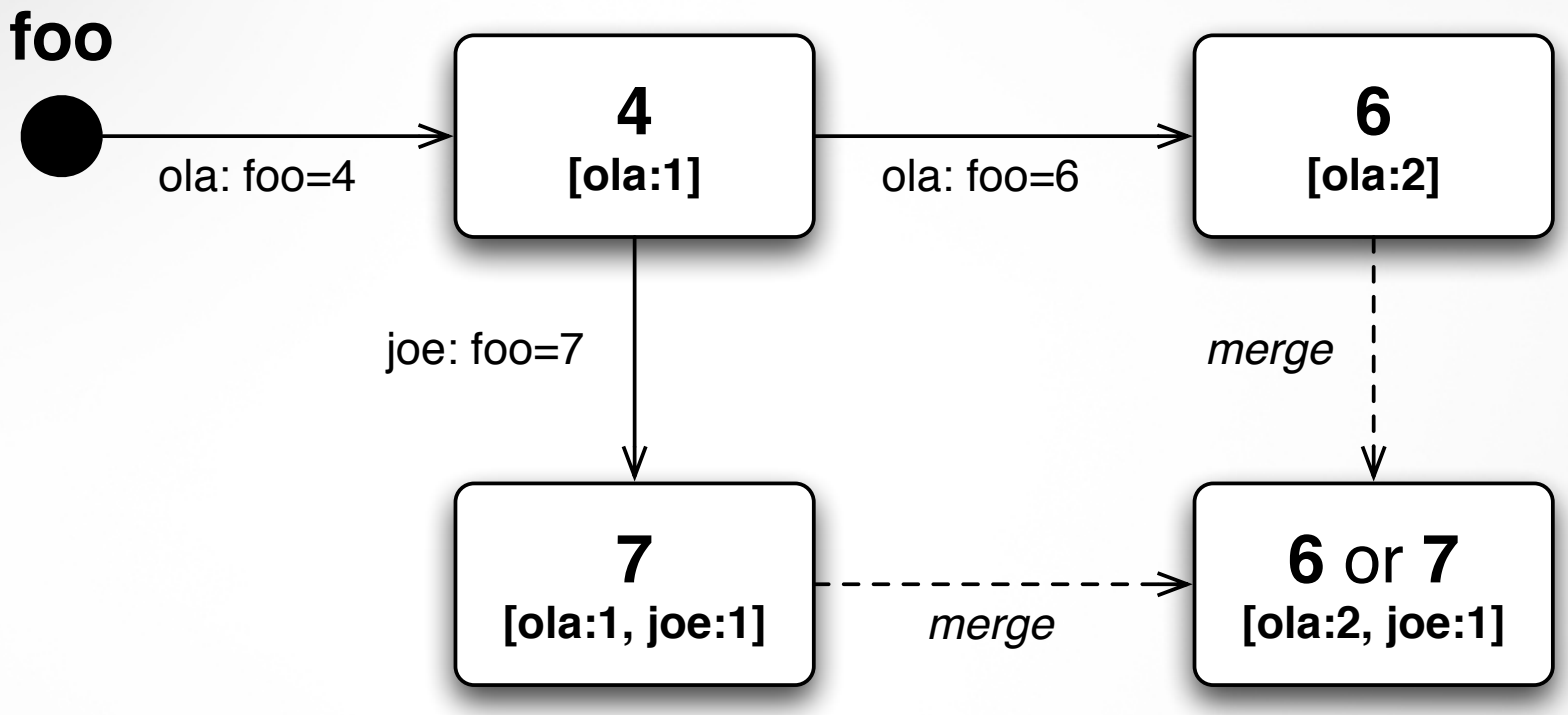
riak

- scalable and available
- system captures write conflicts
- resolve lazily (read repair)

Common Medicine Card


foo
ola: foo=4
4
[ola:1]
ola: foo=6
6
[ola:2]


foo
ola: foo=4
4
[ola:1]
joe: foo=7
7
[ola:1, joe:1]


foo
ola: foo=4
4
[ola:1]
ola: foo=6
6
[ola:2]
joe: foo=7
merge
7
[ola:1, joe:1]
merge
6 or 7
[ola:2, joe:1]

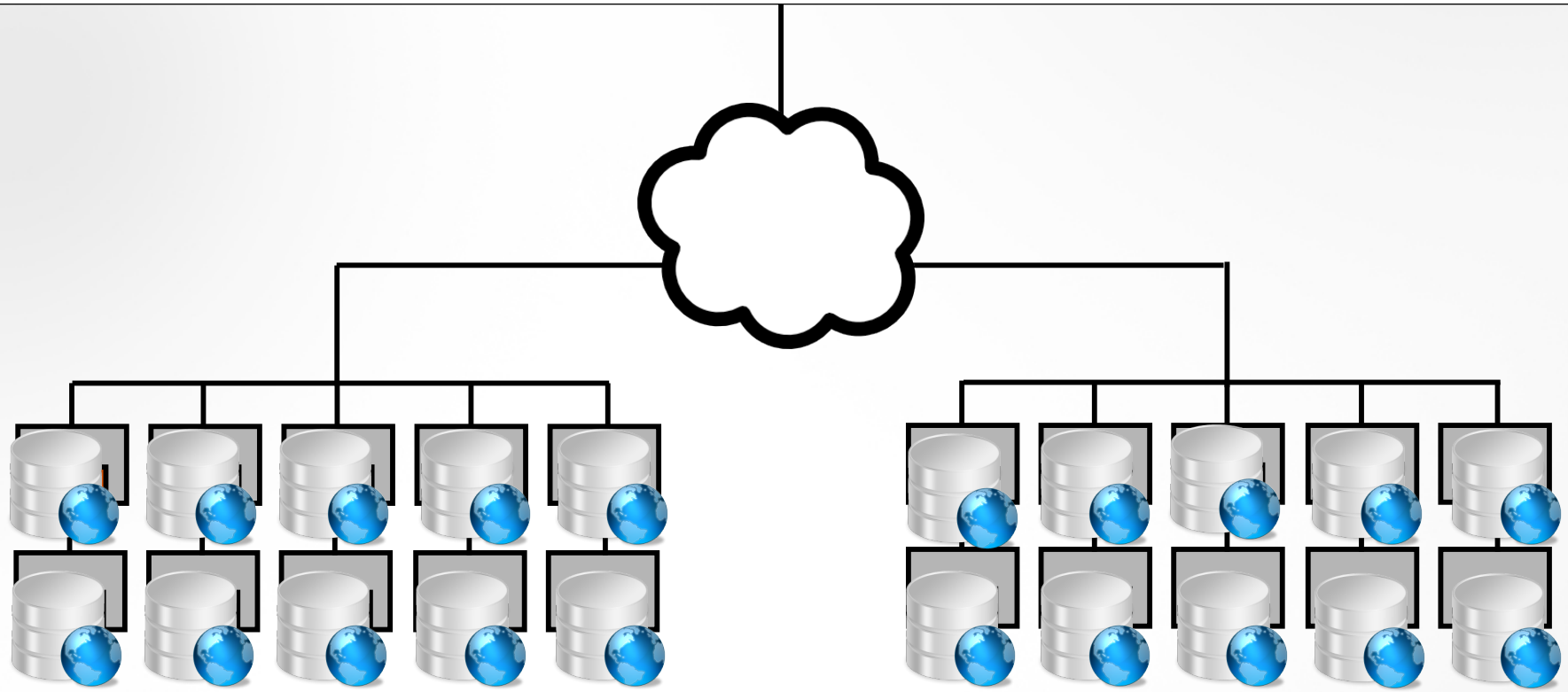
Pseudo Code!

Riak API review

```
connect(ClientID) -> Client
```

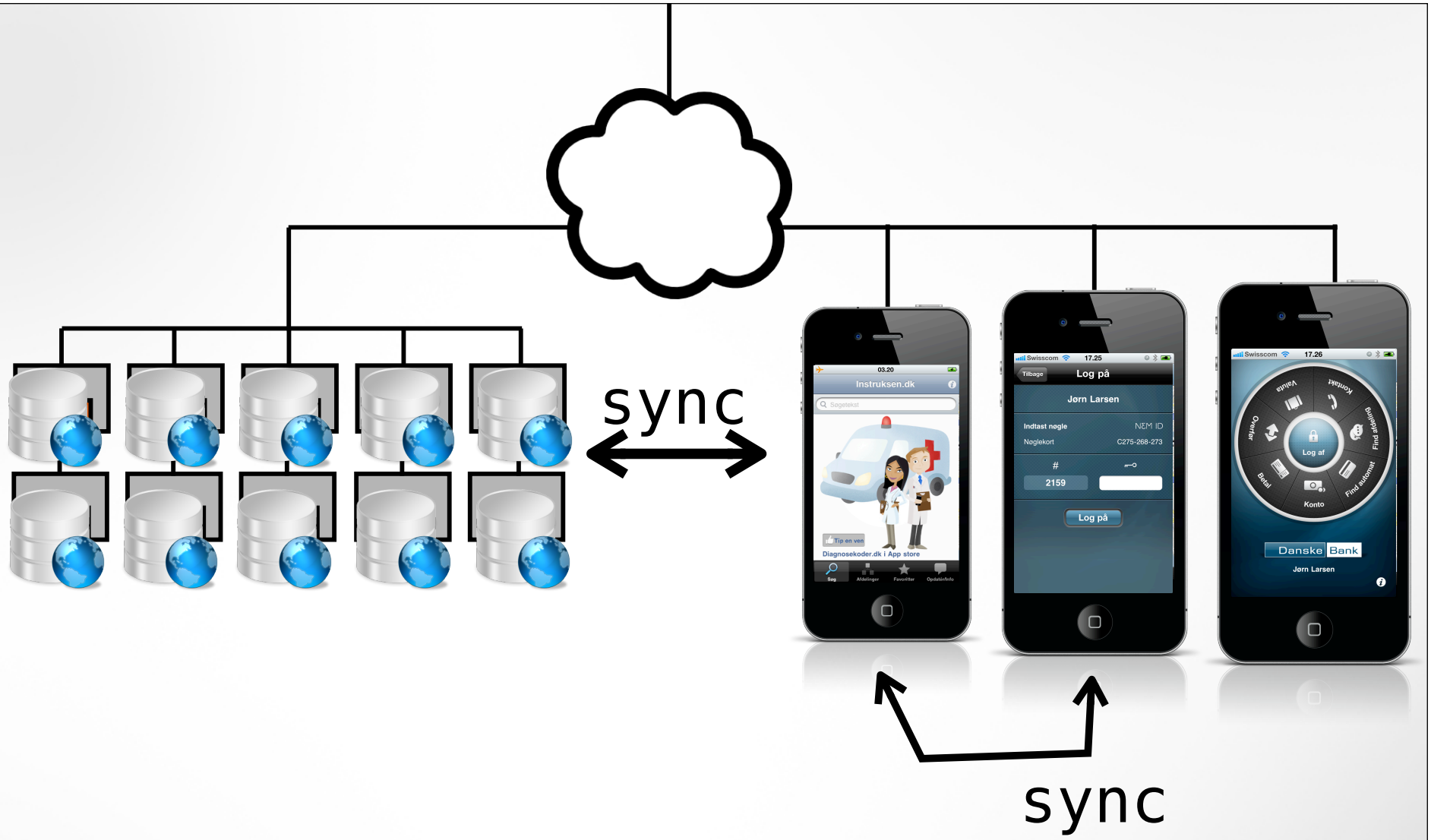
```
BKey  :: {Bucket, Key}
Datum :: {ContentType, RawData}
```

```
Client:insert(BKey, Datum) -> {ok, VClock} | ...
Client:lookup(BKey) -> {ok, VClock, [Datum]} | ...
Client:update(BKey, VClock, Datum) -> ok | ...
Client:delete(BKey, VClock) -> ok | ...
```



riak

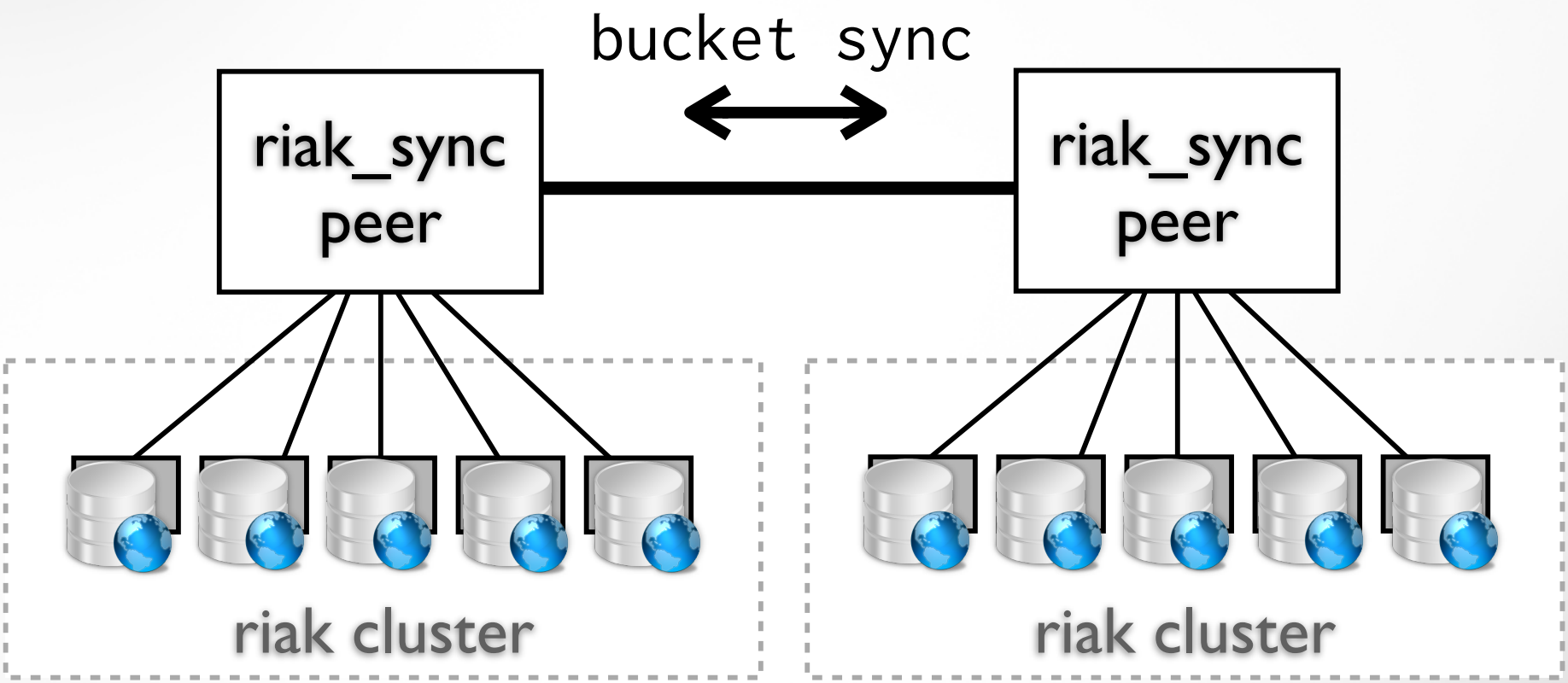
- scalable and available
- system captures write conflicts
- resolve lazily (read repair)


sync
Instruksen.dk
Diagnosekoder.dk i App store
Log på
Jørn Larsen
Indtast nøgle
NEM ID
Nøglekort
C275-268-273
2159
Log på
Danske Bank
Jørn Larsen
Log af
Konto
sync

Today: Two Things

- **RiakSync,** Two-way Replication Protocol
- **BucketDB,** the Mobile client

Not to be confused with `riak_repl`, part of the commercial Riak EDS



```
%% “client” peer
> LocalStore = riak:client_connect(‘riak@127.0.0.1’)

> riak_sync:sync(LocalStore, <<“bucket”>>,
                “172.1.35.204”, 8082)


%% Other site has riak_sync running...
(riak@172.1.35.204)1> riak_sync:start()
```

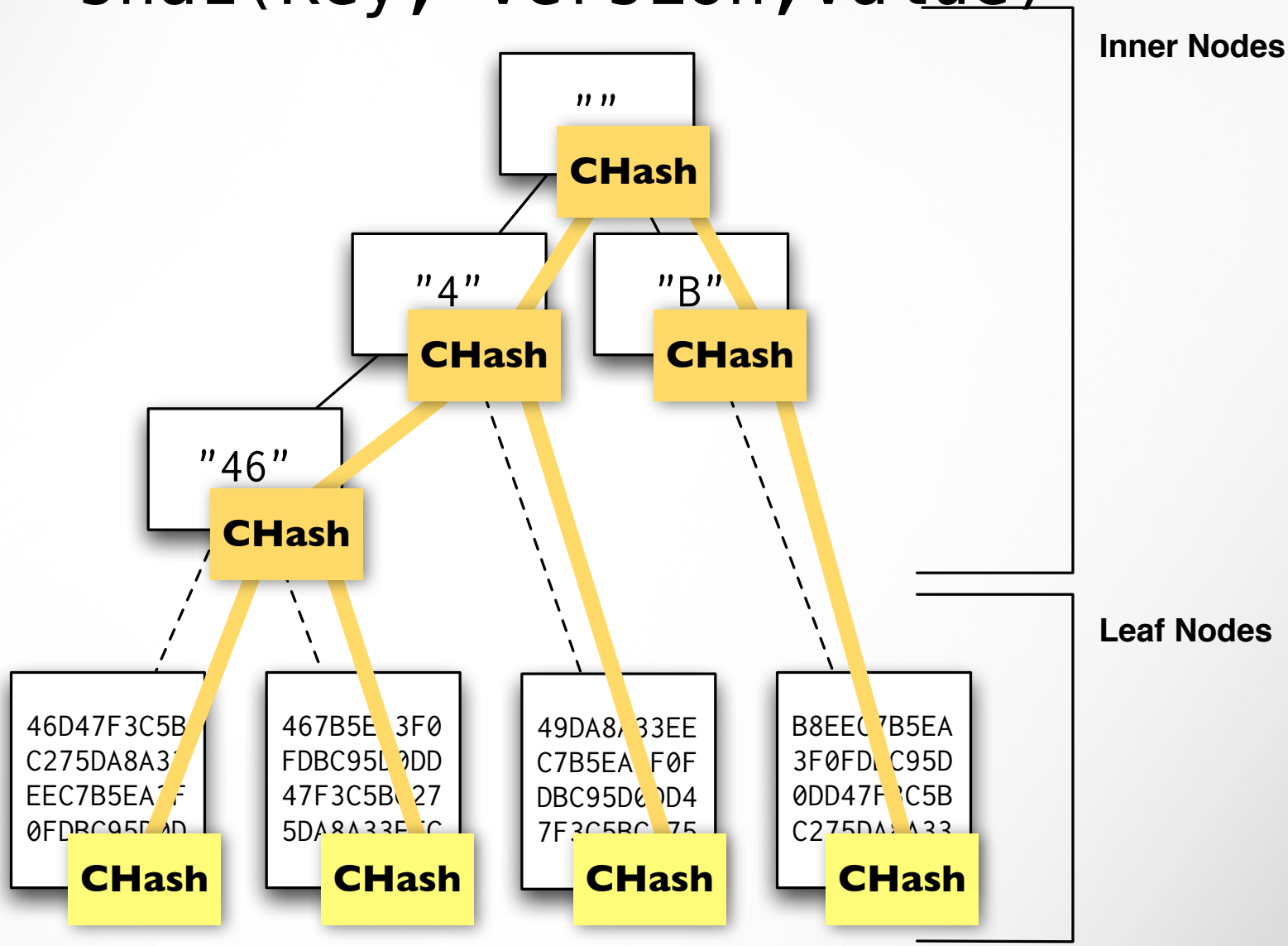
```
{riak_sync, [{pb_ip, “172.1.35.204”},
             {pb_port, 8082},
             {riak, ‘dev@127.0.0.1’}]}
```

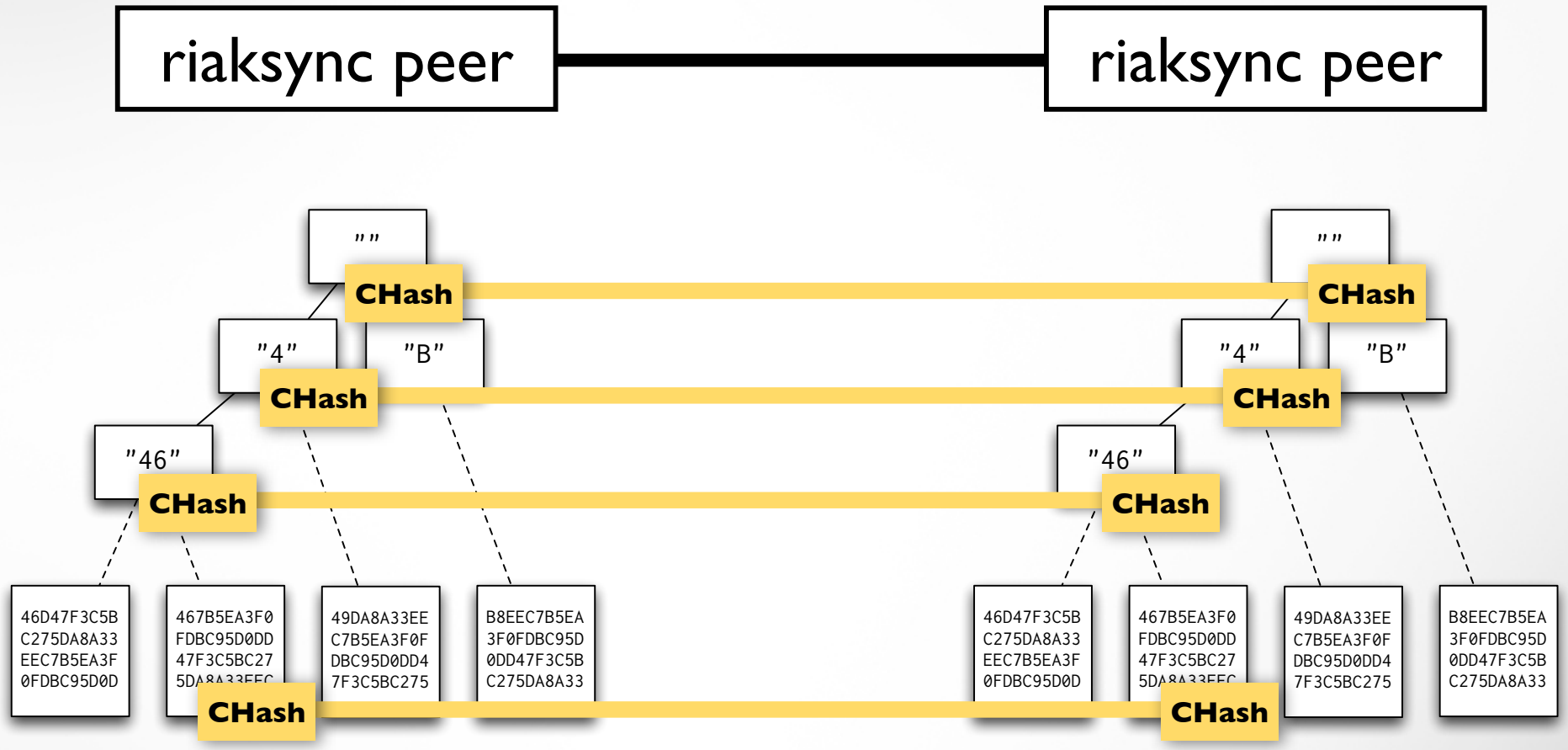
Riak Sync Protocol

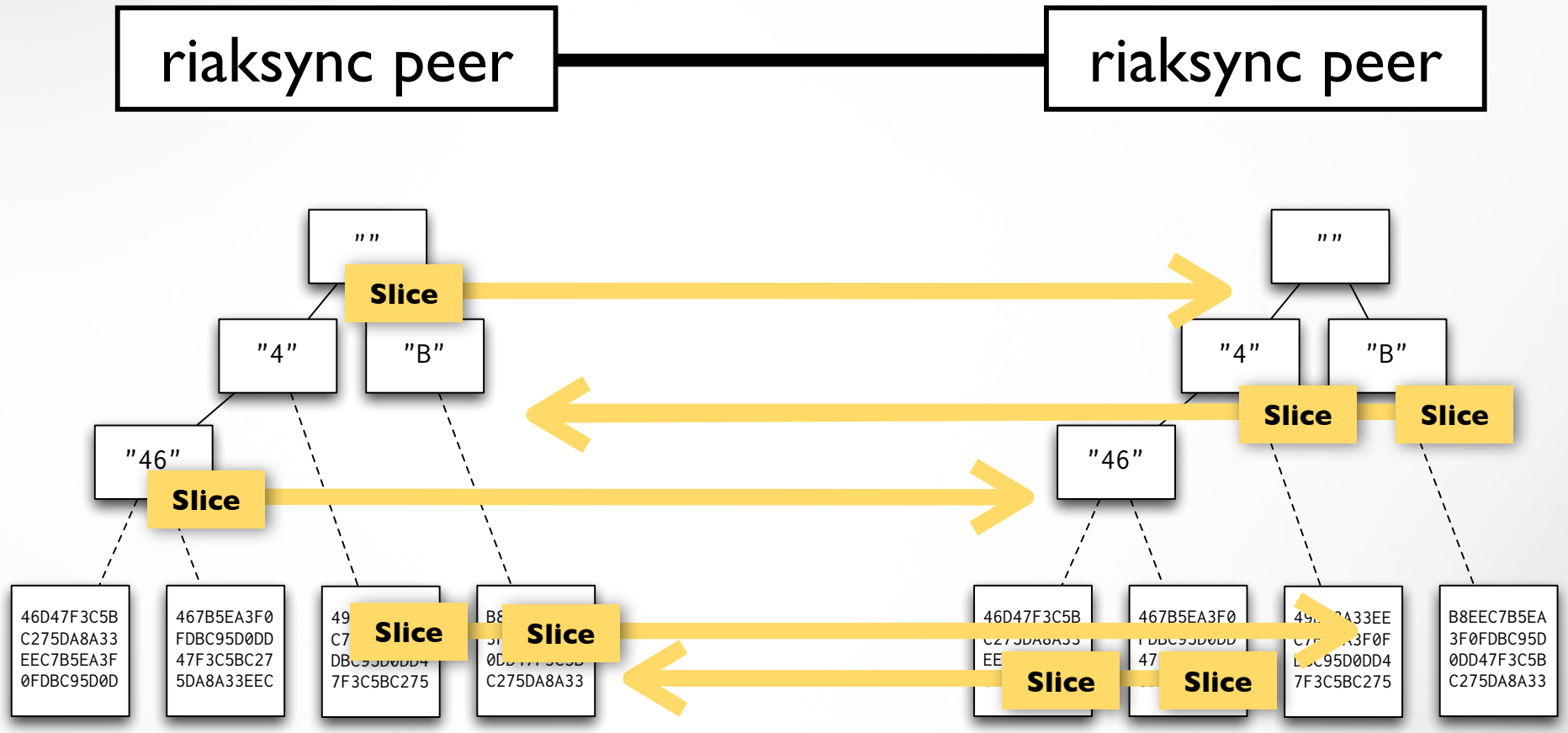
- Riak bucket replication
- Asymmetric work load
- Designed for high-latency networks

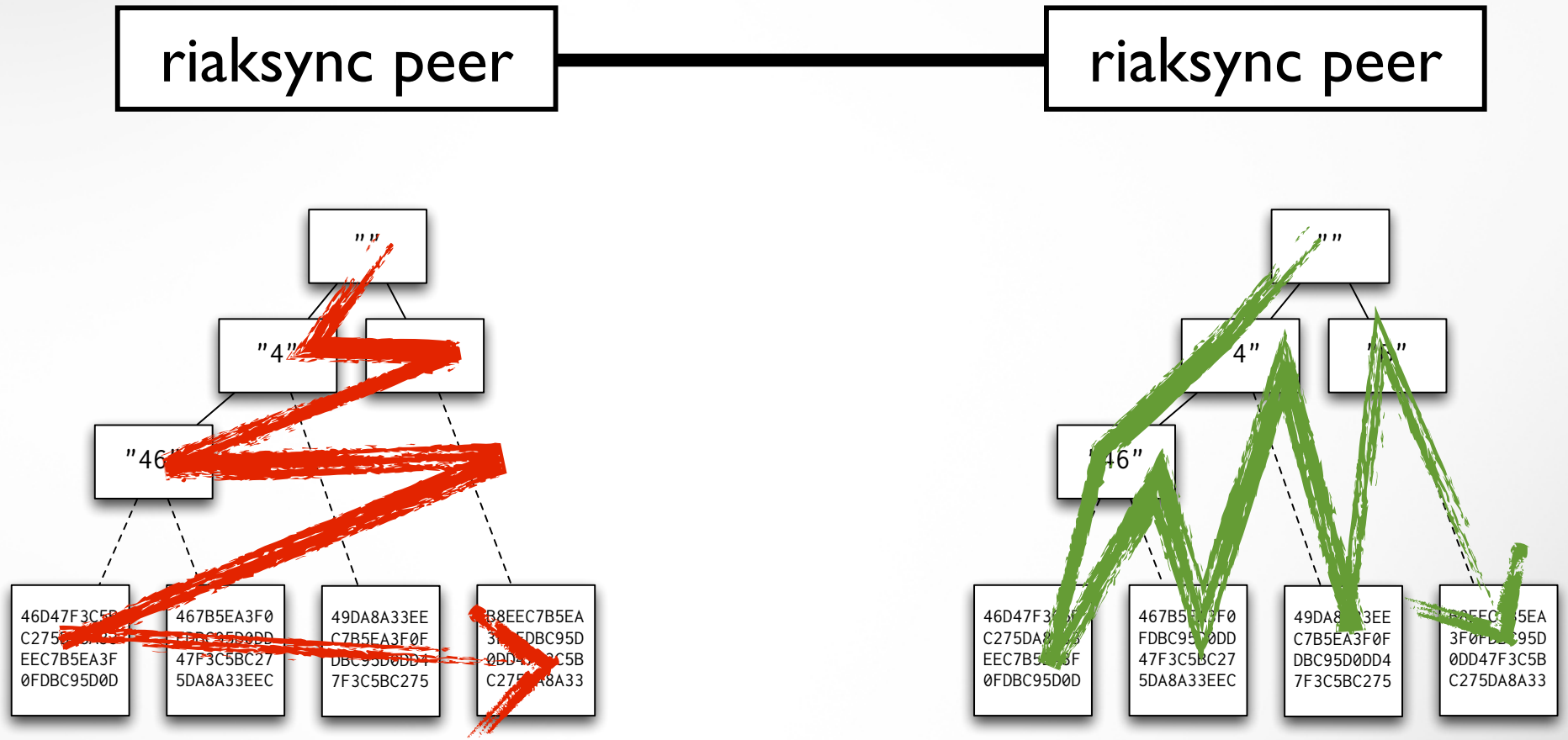
- **Very different design criteria than normal Riak cluster-cluster**

```
StorageKey = sha1(Key)
CHash = sha1(Key, Version,Value)
```



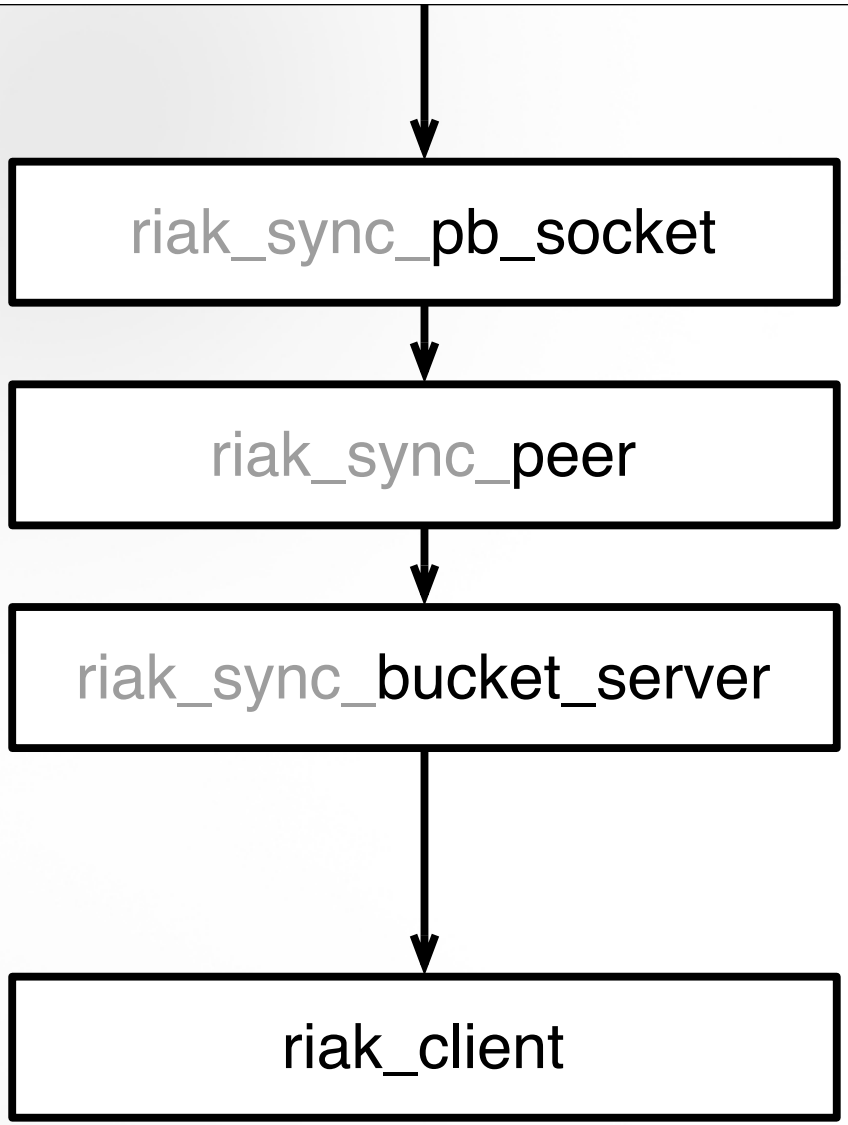

riaksync peer
riaksync peer
""
CHash
"4"
"B"
CHash
"46"
CHash
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
49DA8A33EE
C7B5EA3F0F
DBC95D0DD4
7F3C5BC275
B8EEC7B5EA
3F0FDBC95D
0DD47F3C5B
C275DA8A33
CHash
""
CHash
"4"
"B"
CHash
"46"
CHash
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
49DA8A33EE
C7B5EA3F0F
DBC95D0DD4
7F3C5BC275
B8EEC7B5EA
3F0FDBC95D
0DD47F3C5B
C275DA8A33
CHash


riaksync peer
riaksync peer
""
Slice
"4"
"B"
"46"
Slice
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
Slice
Slice
7F3C5BC275
C275DA8A33
""
"4"
"B"
Slice
Slice
"46"
46D47F3C5B
467B5EA3F0
Slice
Slice
7F3C5BC275
B8EEC7B5EA
3F0FDBC95D
0DD47F3C5B
C275DA8A33


riaksync peer
riaksync peer
""
"4"
"46"
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
49DA8A33EE
C7B5EA3F0F
DBC95D0DD4
7F3C5BC275
B8EEC7B5EA
3F0FDBC95D
0DD47F3C5B
C275DA8A33
""
4"
"46"
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
49DA8A33EE
C7B5EA3F0F
DBC95D0DD4
7F3C5BC275
3F0FDBC95D
0DD47F3C5B
C275DA8A33

Sync Protocol

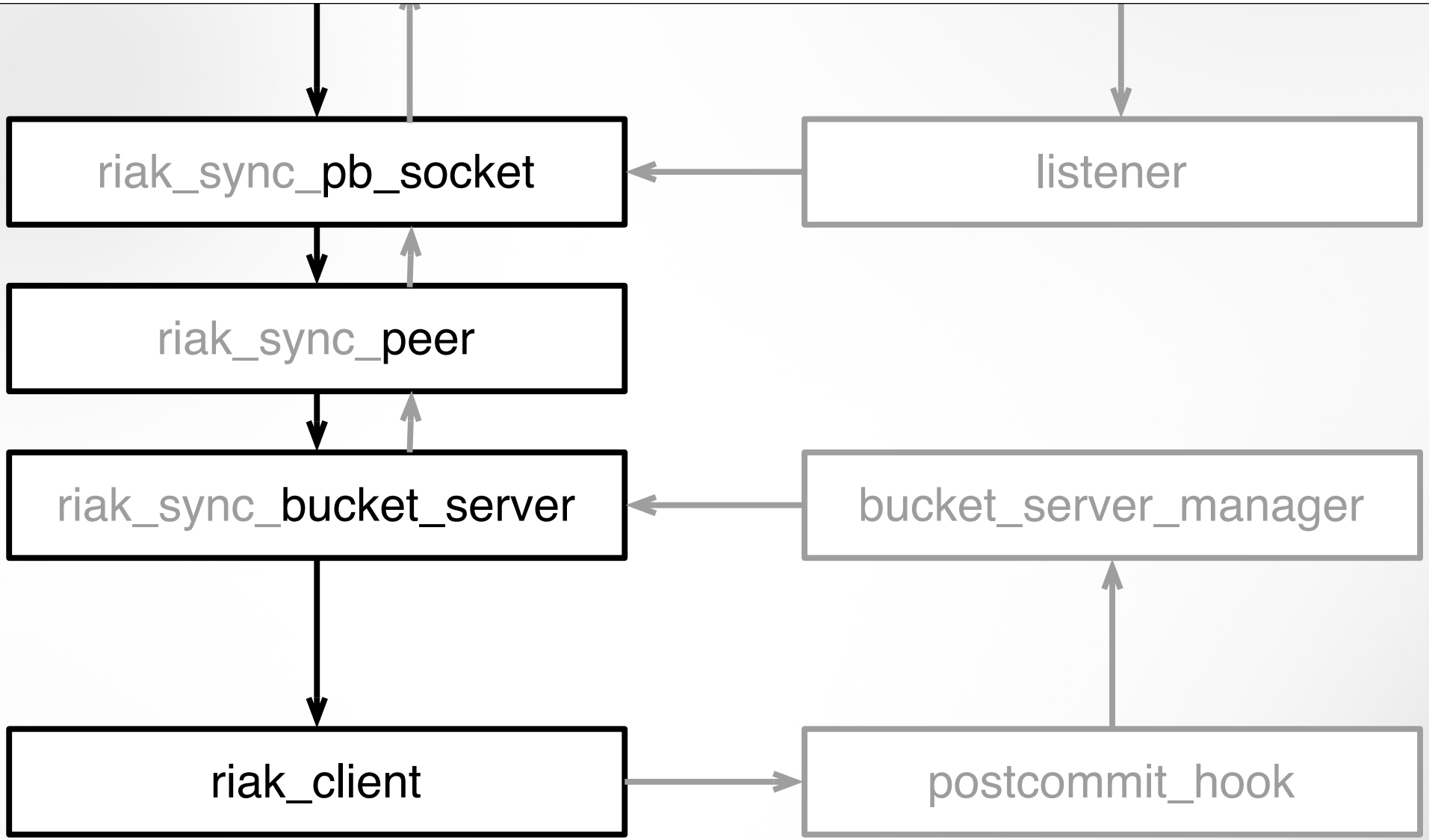
- Parallelizes quickly (client does breadth first)
- Many small messages (will benefit from Nagle's alg)
- Server peer reorders requests to reduce session/buffer state
- Client needs (almost) no session state



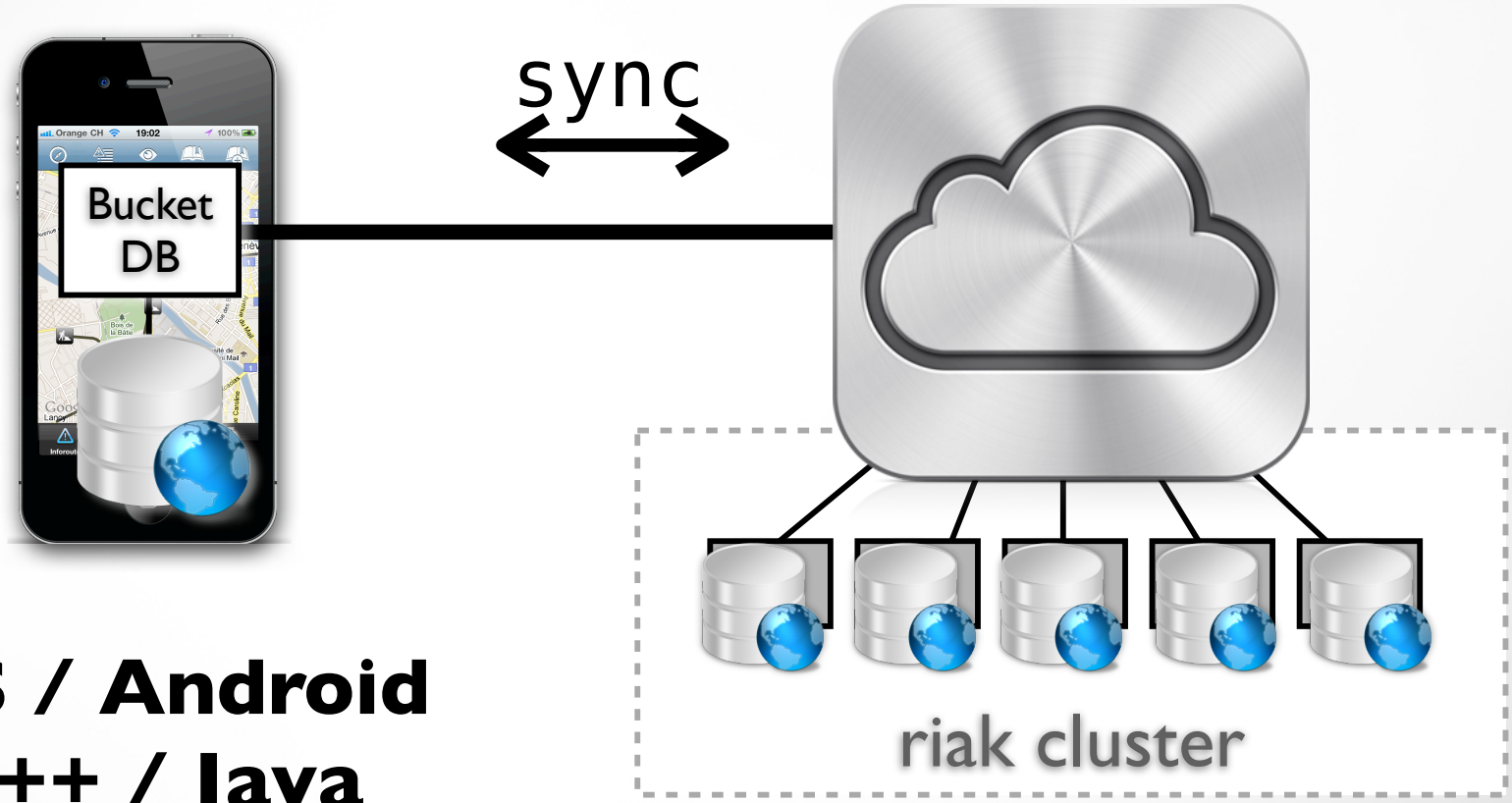
wire protocol handler

session management

merkle_tree server

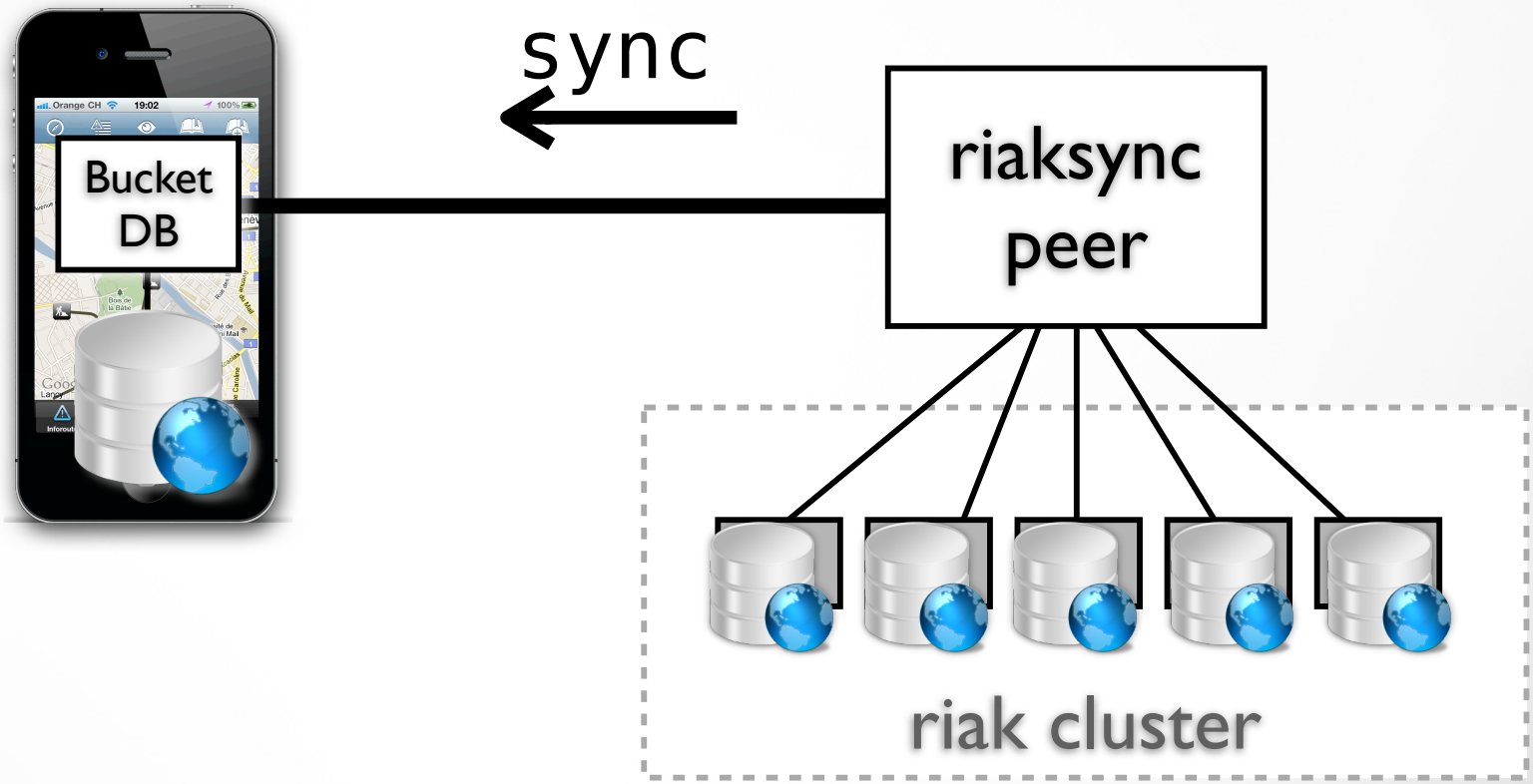

riak_sync_pb_socket
listener
riak_sync_peer
riak_sync_bucket_server
bucket_server_manager
riak_client
postcommit_hook

Mobile Riak: BucketDB

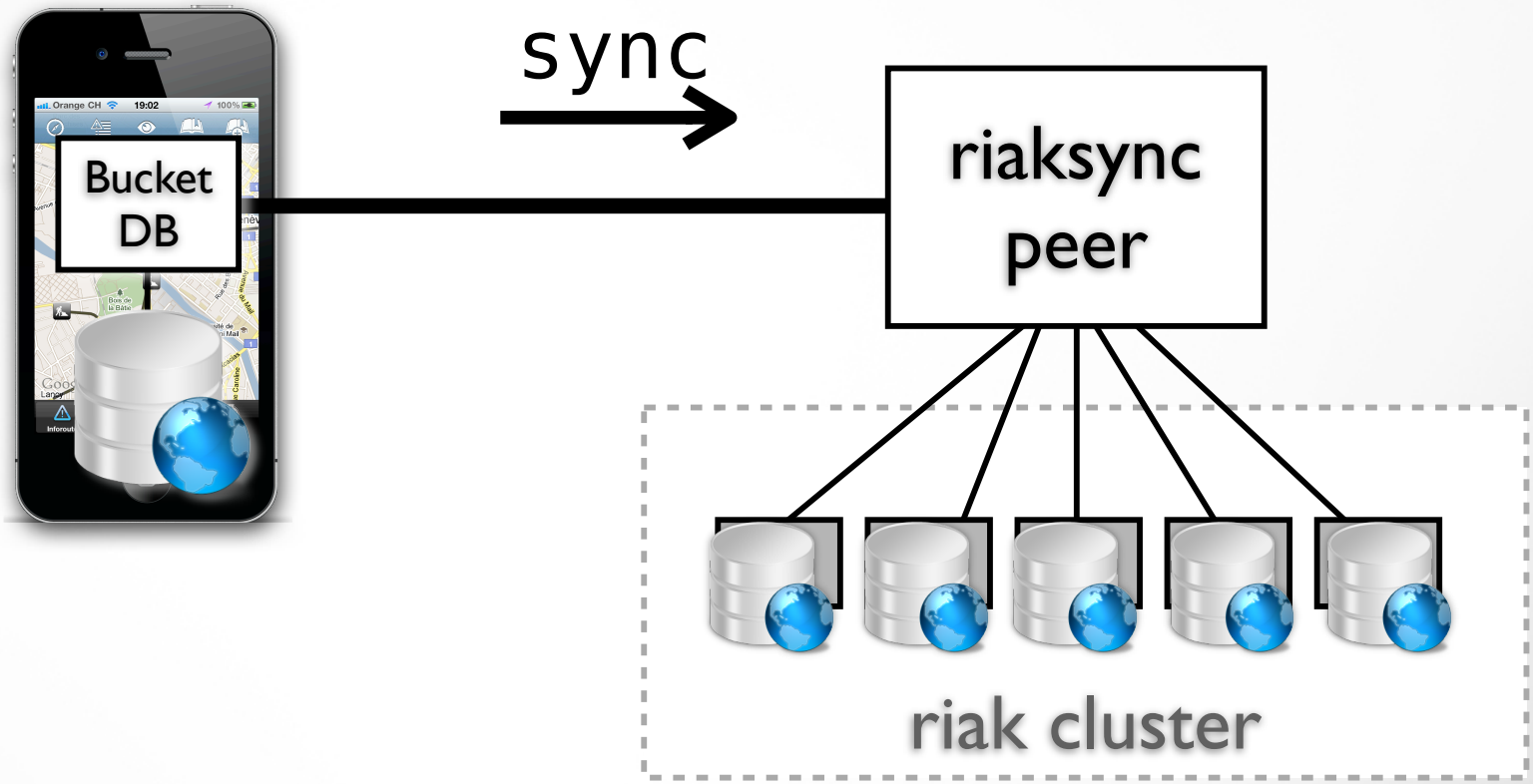


iOS / Android
C++ / Java
Desktop

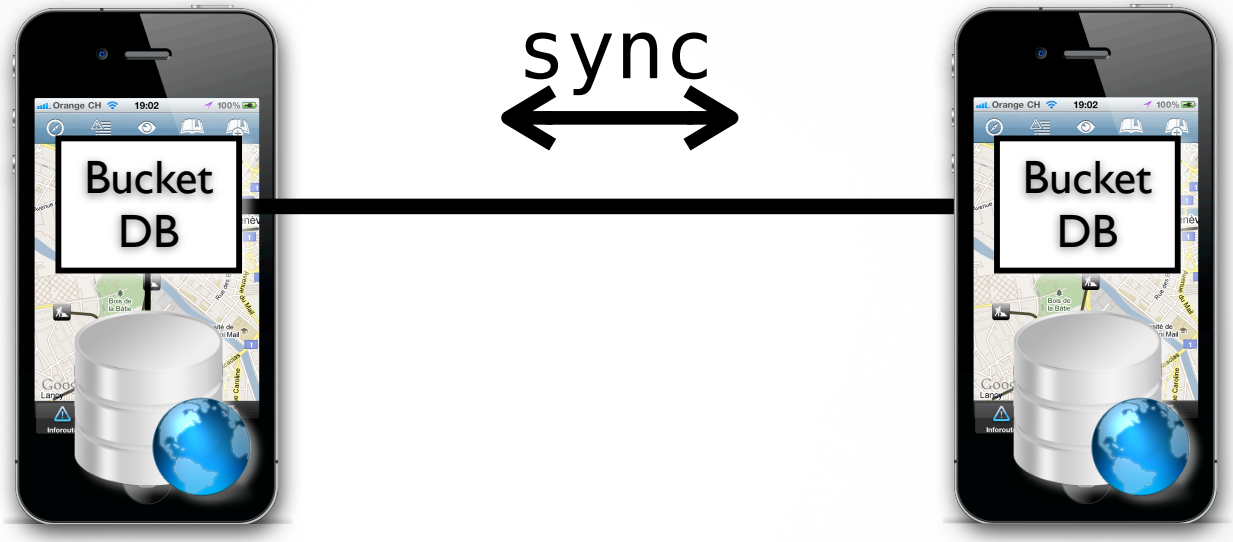
Content Distribution

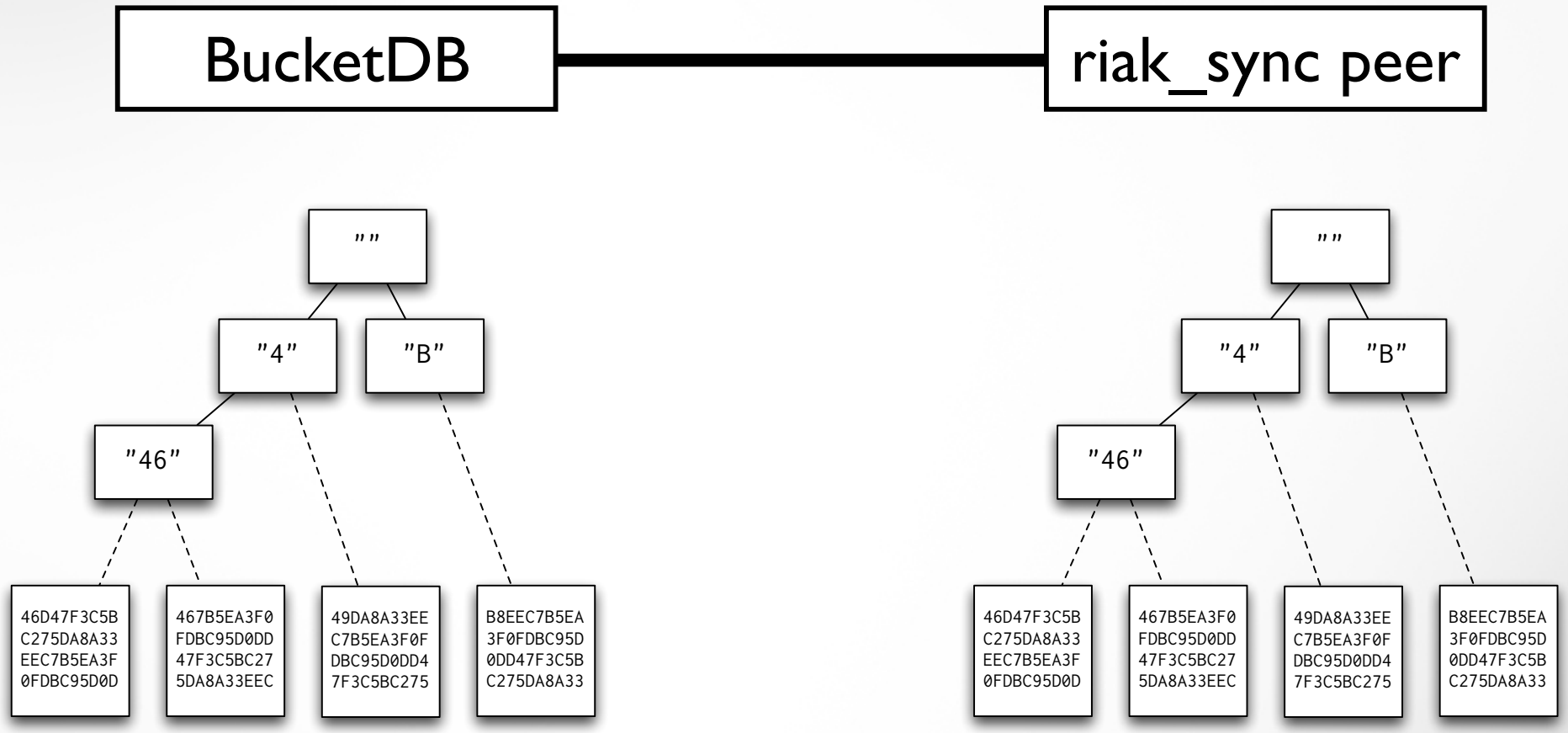


Reliable Queueing

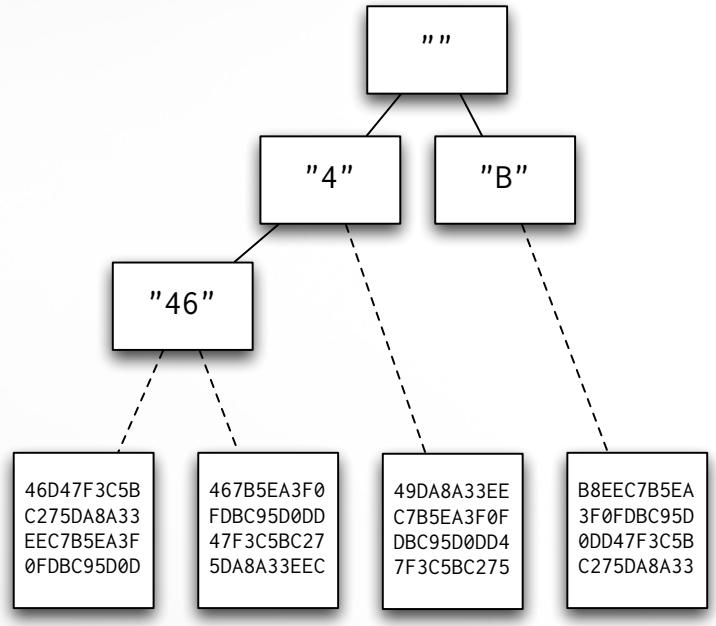


Peer Sync




BucketDB
riak_sync peer
""
"4"
"B"
"46"
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
49DA8A33EE
C7B5EA3F0F
DBC95D0DD4
7F3C5BC275
B8EEC7B5EA
3F0FDBC95D
0DD47F3C5B
C275DA8A33
""
"4"
"B"
"46"
46D47F3C5B
C275DA8A33
EEC7B5EA3F
0FDBC95D0D
467B5EA3F0
FDBC95D0DD
47F3C5BC27
5DA8A33EEC
49DA8A33EE
C7B5EA3F0F
DBC95D0DD4
7F3C5BC275
B8EEC7B5EA
3F0FDBC95D
0DD47F3C5B
C275DA8A33

BucketDB



- Client stores data in local data store as a HASH TREE
- Always “ready to sync” (no session state, no cpu/memory intensive computation)

DATA

ROW_ID	KEY	CONTENT TYPE	DATA

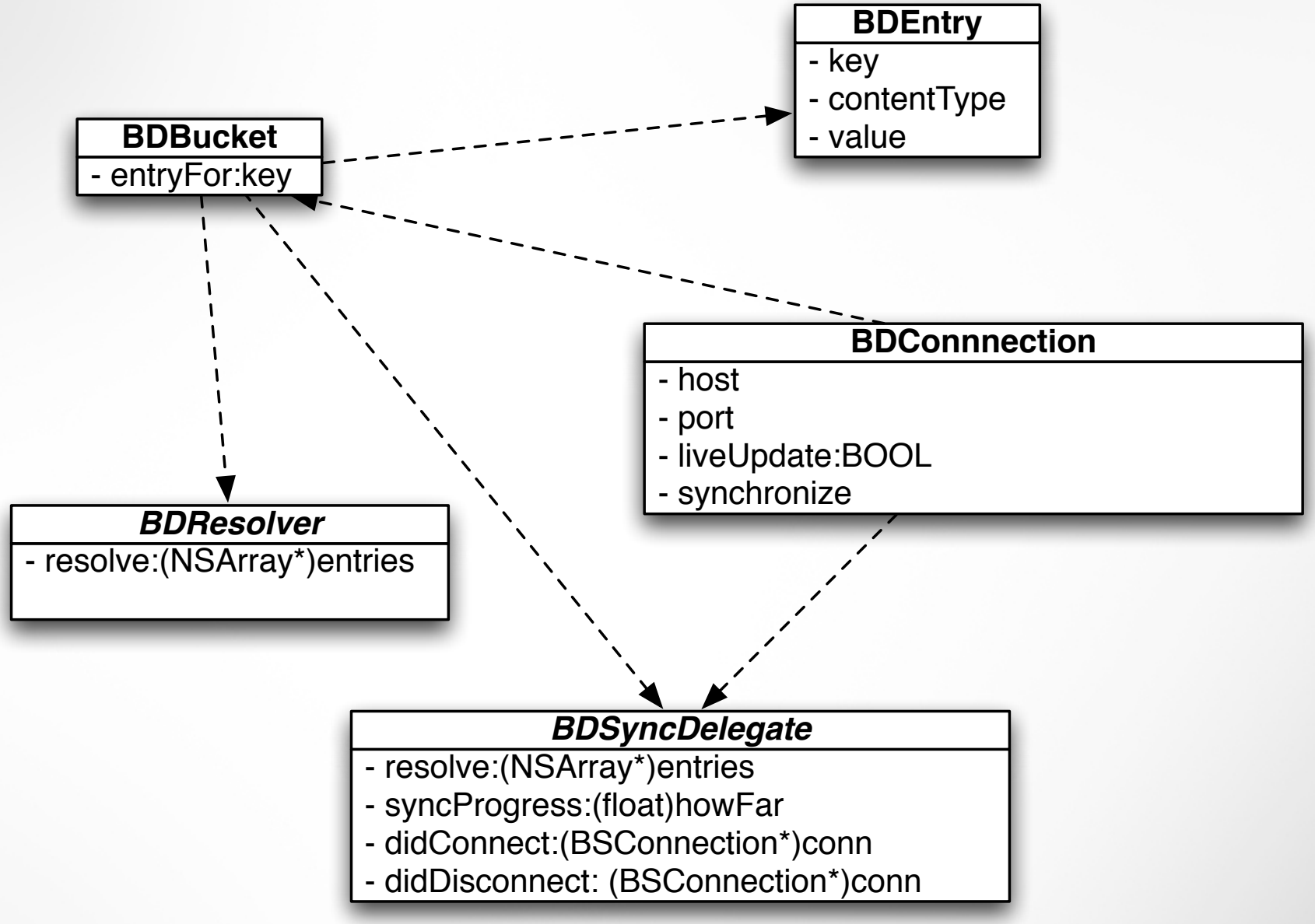
TRIGGER

MERKLE_LEAF

HKEY	VCLOCK	CHASH	KEY

MERKLE_INNER

PATH	CHASH	CHILDREN


BDEntry
- key
- contentType
- value
BDBucket
- entryFor:key
BDConnnection
- host
- port
- liveUpdate:BOOL
- synchronize
BDResolver
- resolve:(NSArray*)entries
BDSyncDelegate
- resolve:(NSArray*)entries
- syncProgress:(float)howFar
- didConnect:(BSConnection*)conn
- didDisconnect: (BSConnection*)conn

Other Items to Discuss

- Security models
  - Provide own “socket factory” / credentials
  - Document- or Record-oriented
- Handling Deleted data
- Future: Index/Search, local M/R

Summary

- Riak Data Model
- RiakSync, a protocol for Key/Value synchronization
- BucketDB, Riak clients for mobile

Thank You

@drkrab

Thank You

@drkrab