Streaming a million likes/second
Real-time interactions on Live Video

Akhilesh Gupta

Realtime Engineer, LinkedIn

QCon by InfoQ

tiny.cc/qconlive

Login with LinkedIn credentials

?

Which was the largest live stream in the world?
(largest number of concurrent viewers)

INDIA VS NEW ZEALAND WORLD CUP

Largest Live Stream

>25.3M viewers


PUMA
AMERICAN EAGLE
UNREAL
REAL TIME
REAL TIME
REAL TIME

Distributed Systems

Start small and add simple layers to your architecture


PUMA
AMERICAN EAGLE
UNREAL
REAL TIME
REAL TIME
REAL TIME

What is a Live Video?

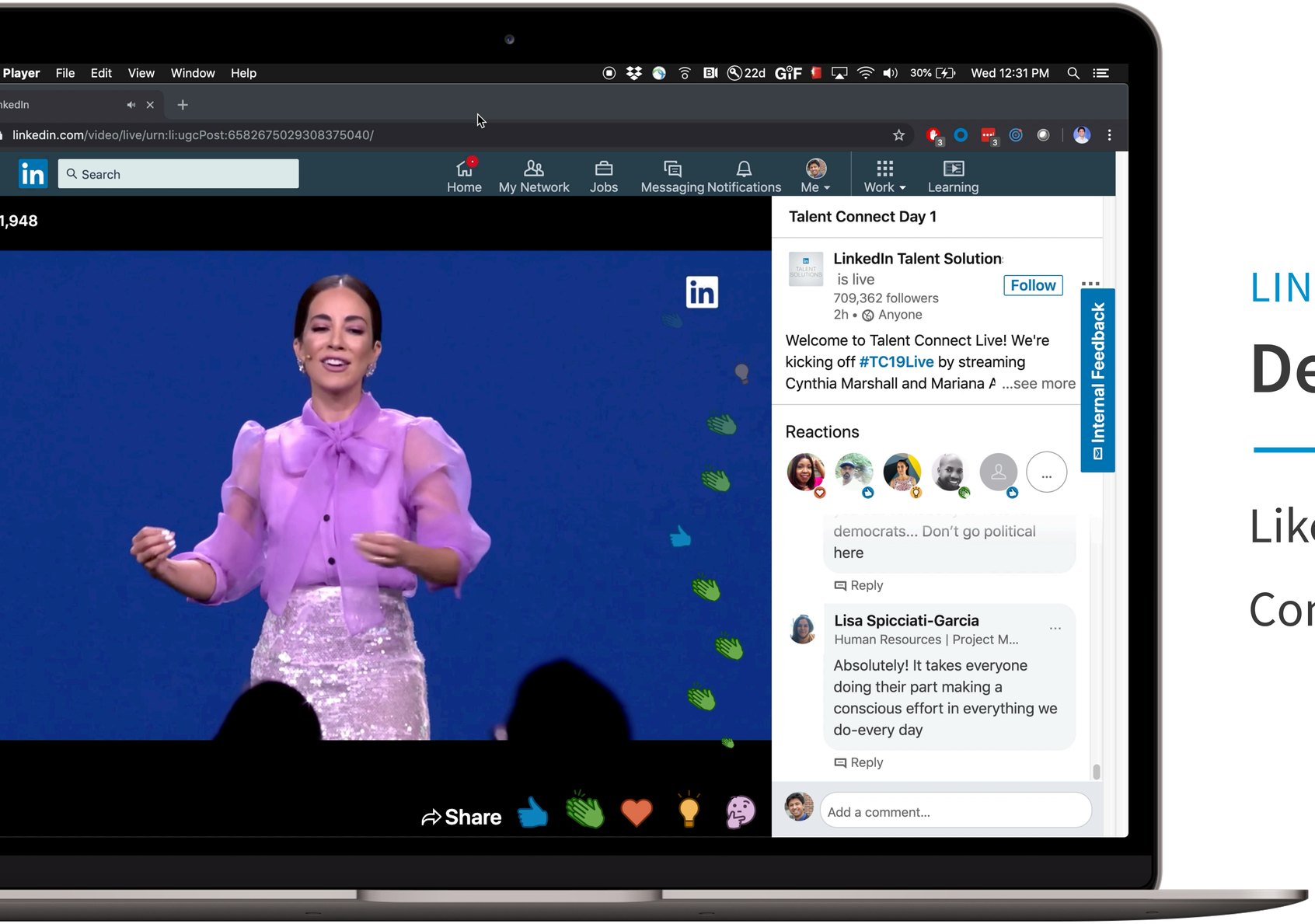
What are real-time interactions on Live Video?

LINKEDIN LIVE

Mobile

Likes/Comments

Concurrent Viewer Counts



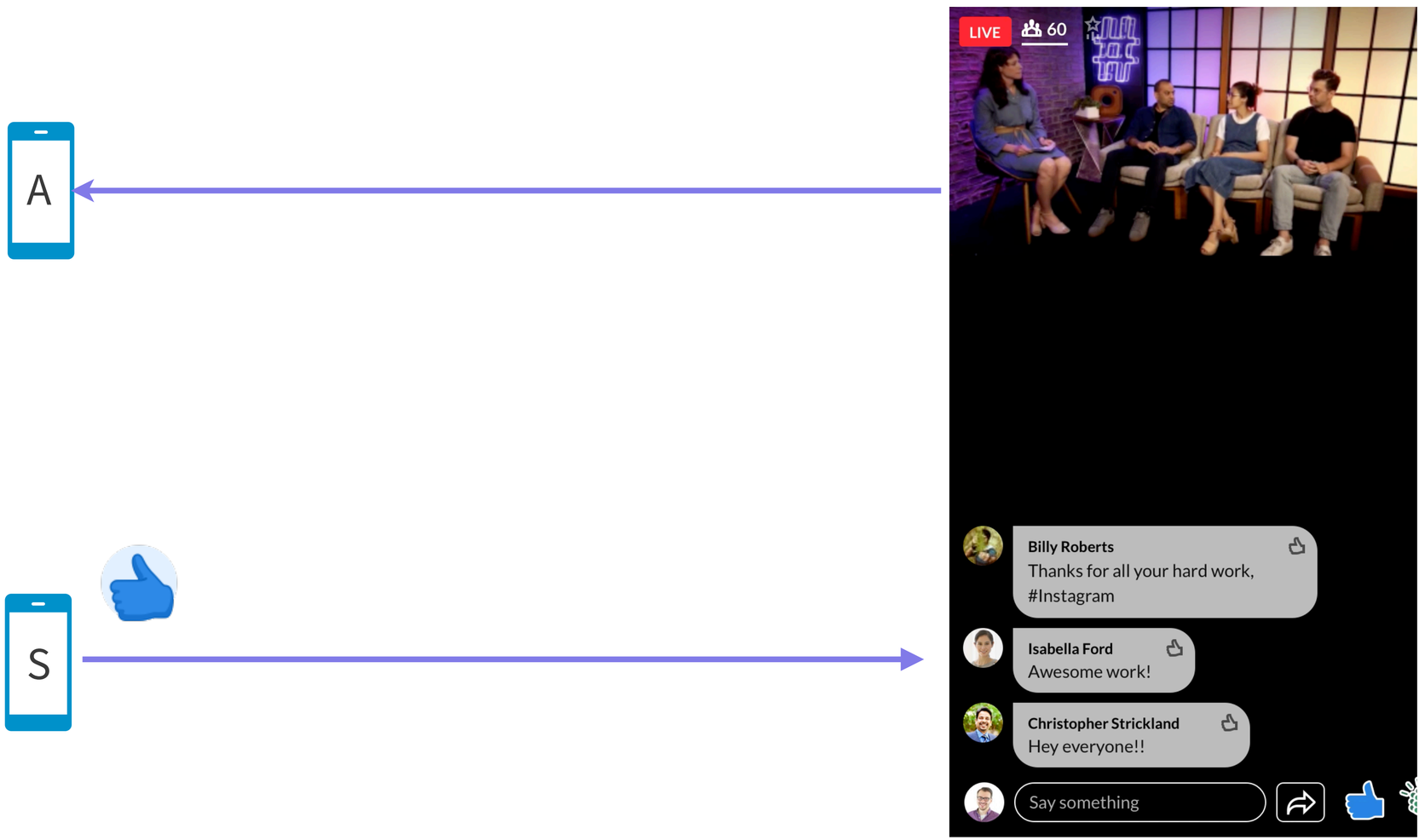
LINKEDIN LIVE

Desktop

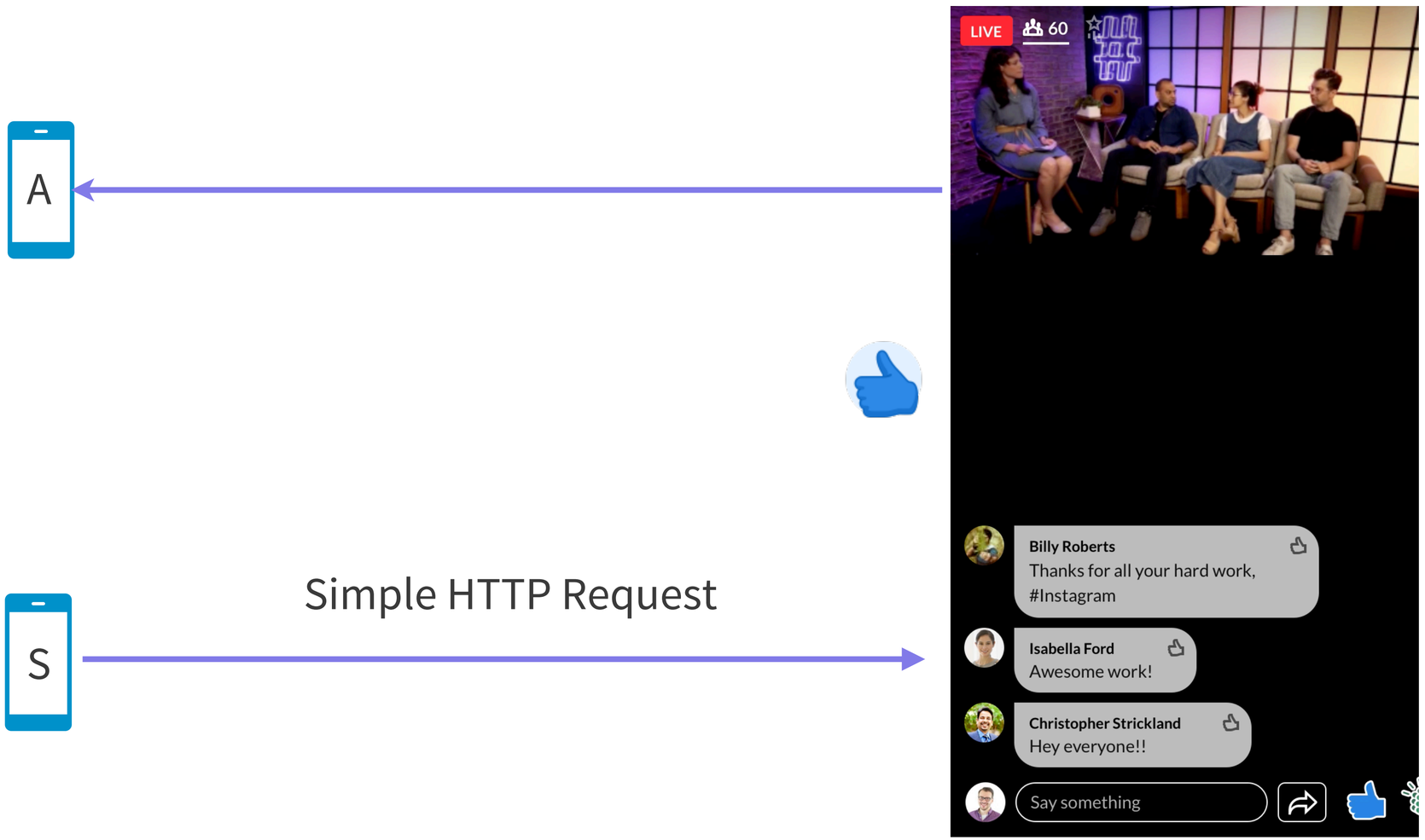
Likes/Comments

Concurrent Viewer Counts

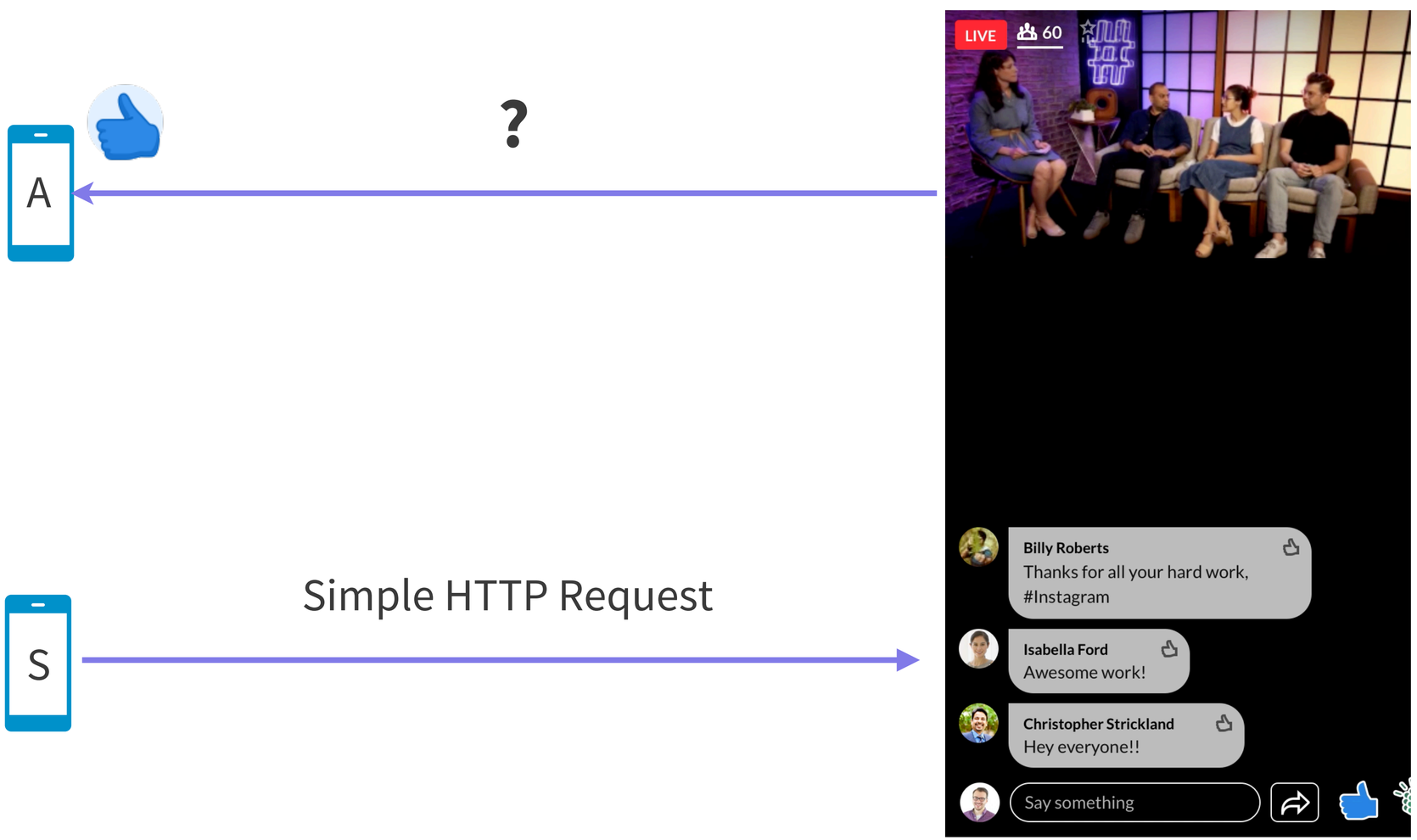
Distribution of Likes



Distribution of Likes

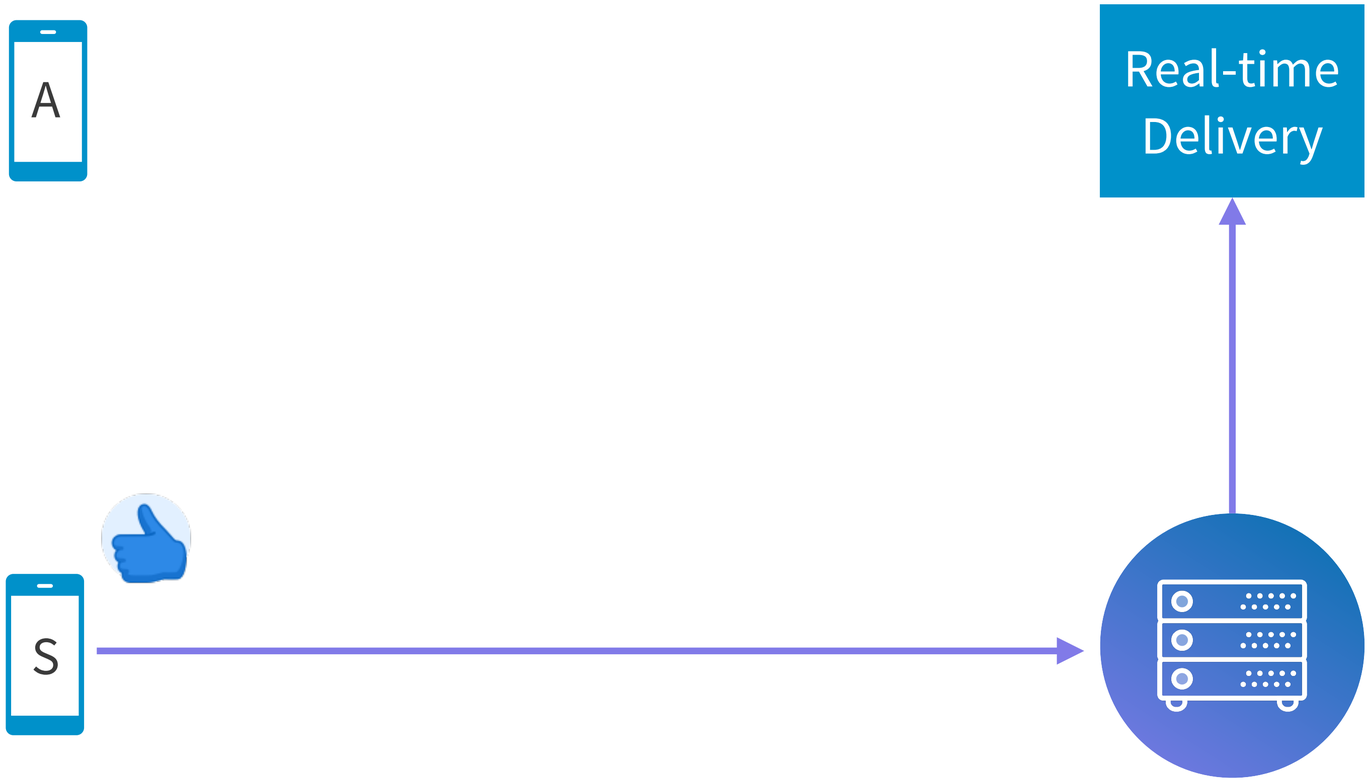


Distribution of Likes

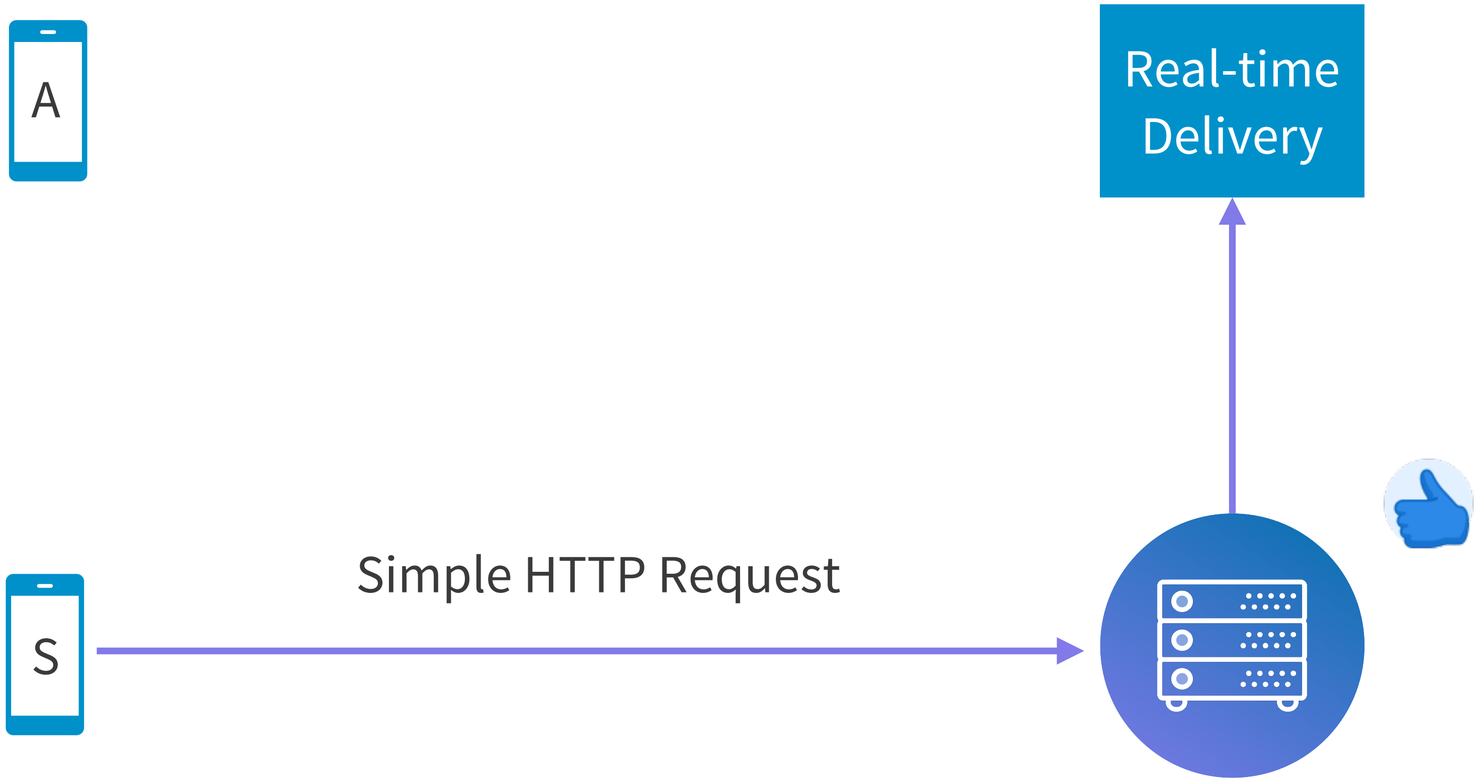


Challenge 1:The delivery pipe

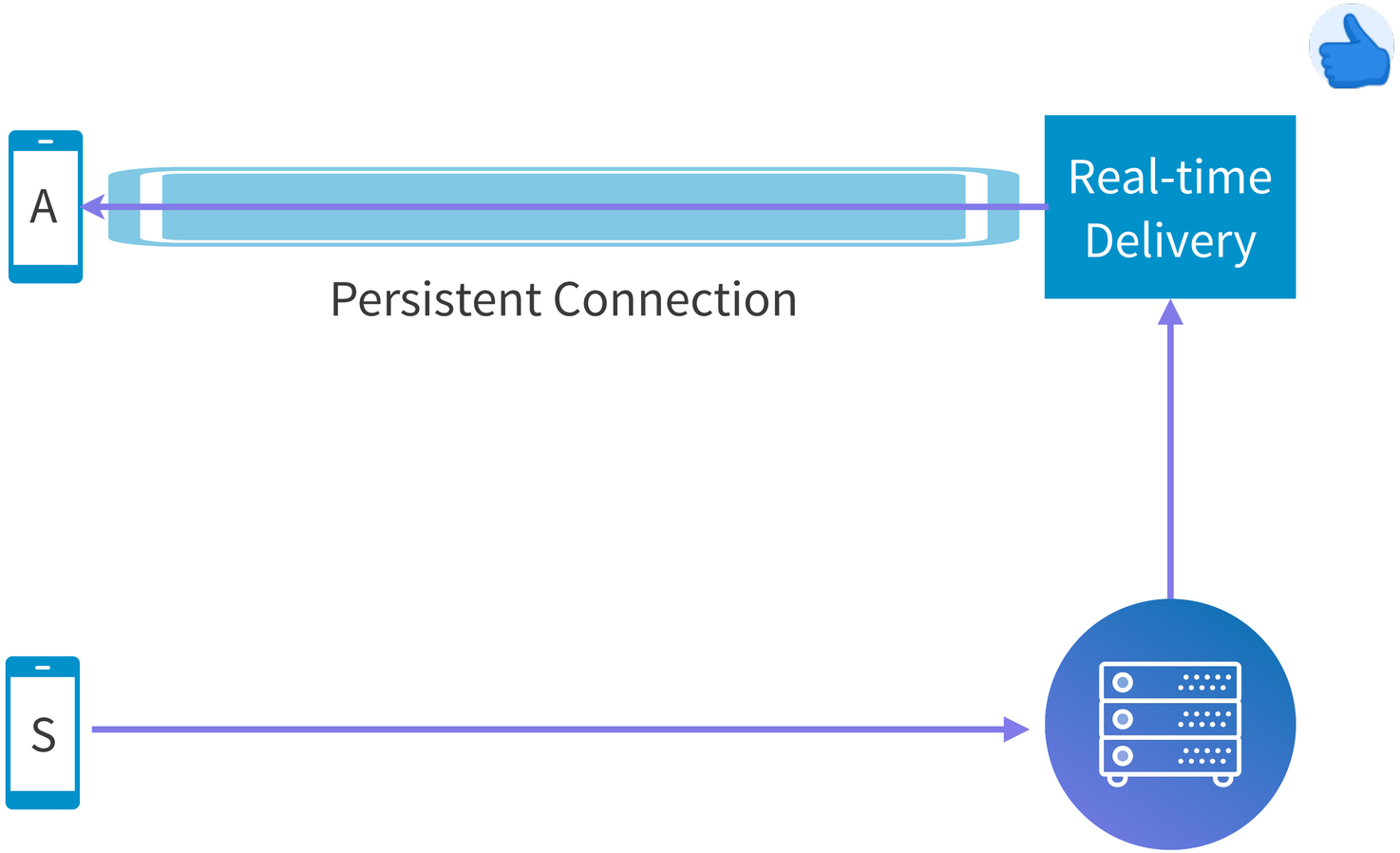
The delivery pipe



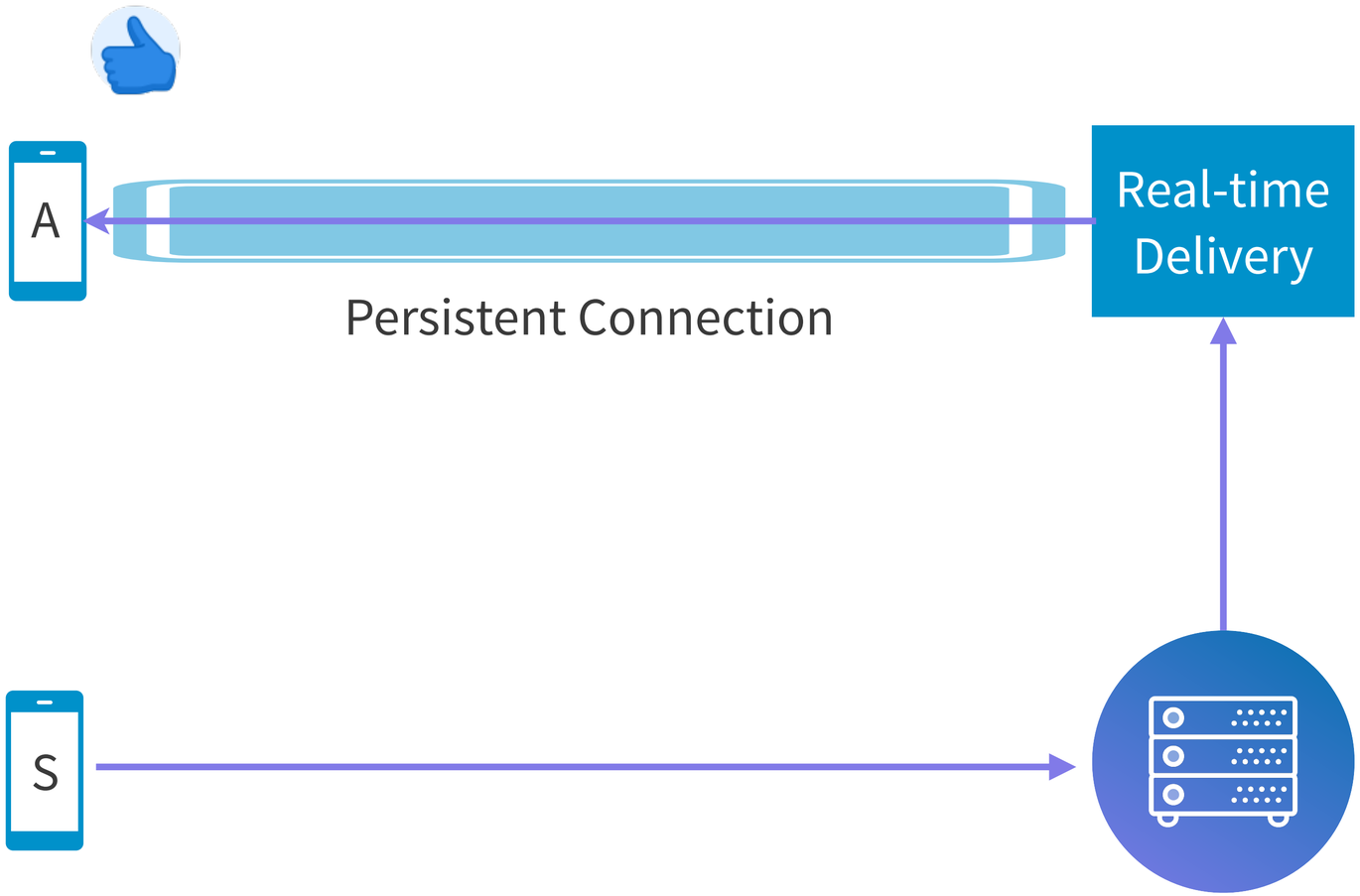
The delivery pipe



The delivery pipe



The delivery pipe



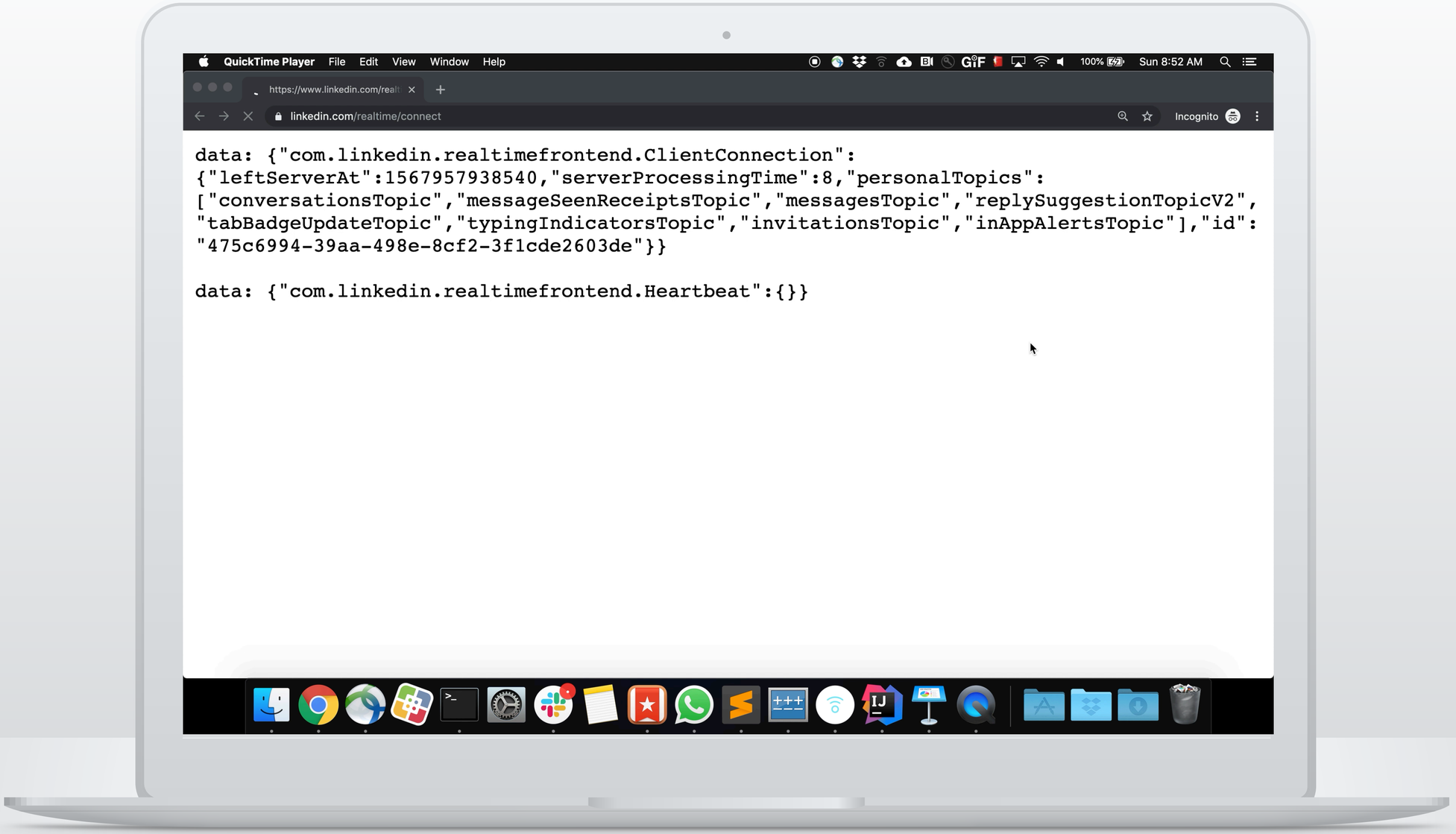
Persistent Connection

Persistent Connection

TRY IT!

Login at linkedin.com

tiny.cc/realtime


QuickTime Player File Edit View Window Help
100% Sun 8:52 AM
https://www.linkedin.com/realt
linkedin.com/realtime/connect
Incognito
data: {"com.linkedin.realtimefrontend.ClientConnection":
{"leftServerAt":1567957938540,"serverProcessingTime":8,"personalTopics":
["conversationsTopic","messageSeenReceiptsTopic","messagesTopic","replySuggestionTopicV2",
"tabBadgeUpdateTopic","typingIndicatorsTopic","invitationsTopic","inAppAlertsTopic"],"id":
"475c6994-39aa-498e-8cf2-3f1cde2603de"}}
data: {"com.linkedin.realtimefrontend.Heartbeat":{}}

HTTP Long Poll with Server Sent Events

EVENTSOURCE INTERFACE CLIENT REQUEST

```
GET /connect HTTP/1.1
Accept: text/event-stream
```

HTTP Long Poll with Server Sent Events

EVENTSOURCE INTERFACE SERVER RESPONSE

```
HTTP/1.1 200 OK
- RESPONSE HEADERS -
Content-Type: text/event-stream
```

HTTP Long Poll with Server Sent Events

EVENTSOURCE INTERFACE SERVER RESPONSE

```
HTTP/1.1 200 OK
- RESPONSE HEADERS -
Content-Type: text/event-stream

- RESPONSE BODY -
data: {“like” object}
```

HTTP Long Poll with Server Sent Events

EVENTSOURCE INTERFACE SERVER RESPONSE

```
HTTP/1.1 200 OK
- RESPONSE HEADERS -
Content-Type: text/event-stream

- RESPONSE BODY -
data: {“like” object}
..
data: {“comment” object}
```

HTTP Long Poll with Server Sent Events

EVENTSOURCE INTERFACE SERVER RESPONSE

```
HTTP/1.1 200 OK
- RESPONSE HEADERS -
Content-Type: text/event-stream

- RESPONSE BODY -
data: {“like” object}
..
data: {“comment” object}
..
data: {“like” object}
```

EventSource Client Libraries

WEB

```
var evtSource =
new EventSource("https://www.linkedin.com/realtime/connect");

evtSource.onmessage = function(e) {
  var likeObject = JSON.parse(e.data);
}
```

EventSource Client Libraries

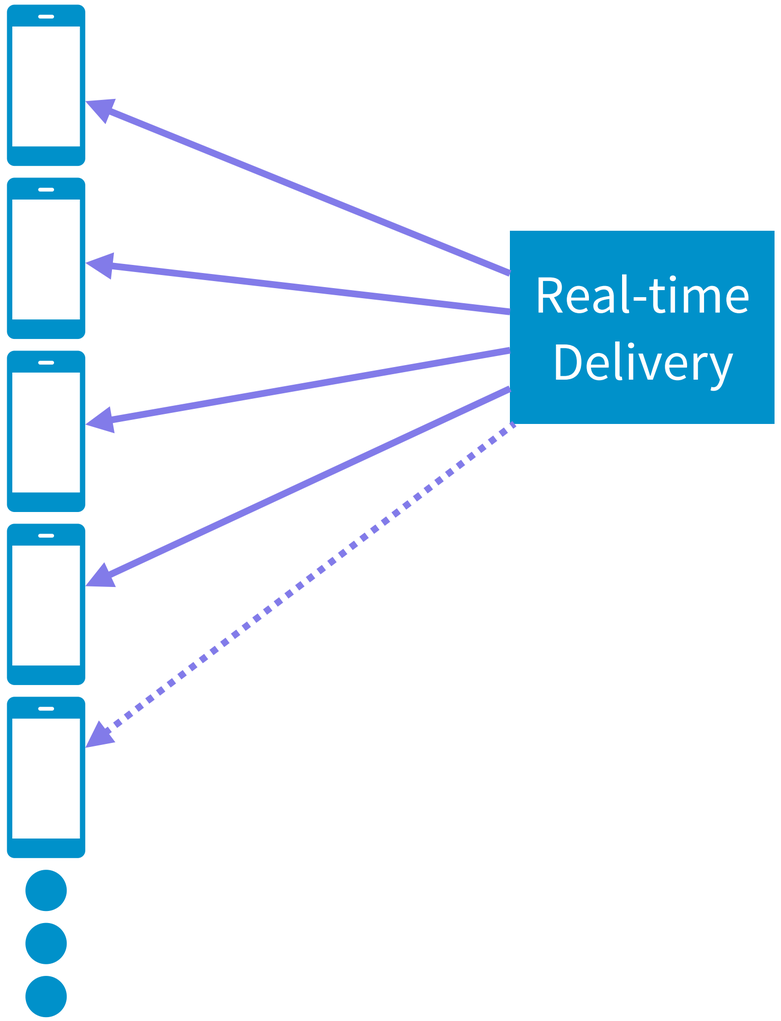
ANDROID & IOS SUPPORT

Android: https://github.com/tylerjroach/eventsource-android

iOS Swift: https://github.com/inaka/EventSource

Next Challenge?

Next Challenge: Multiple Connections



Challenge 2: Connection Management

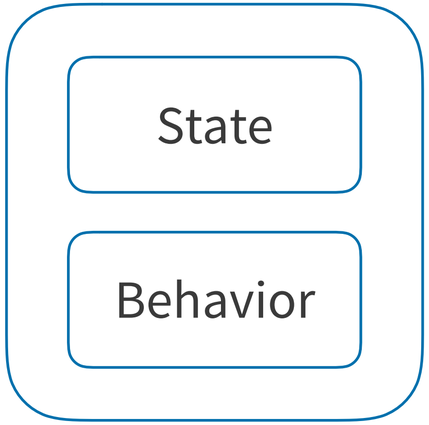
Connection Management

PLAY & AKKA

Akka is a toolkit for building highly concurrent, distributed, and resilient message-driven applications

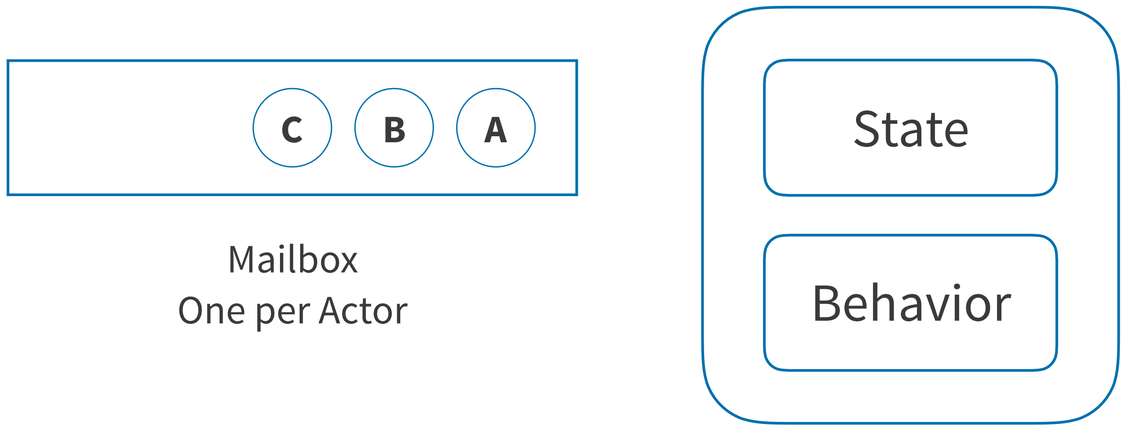
Connection Management

AN AKKA ACTOR



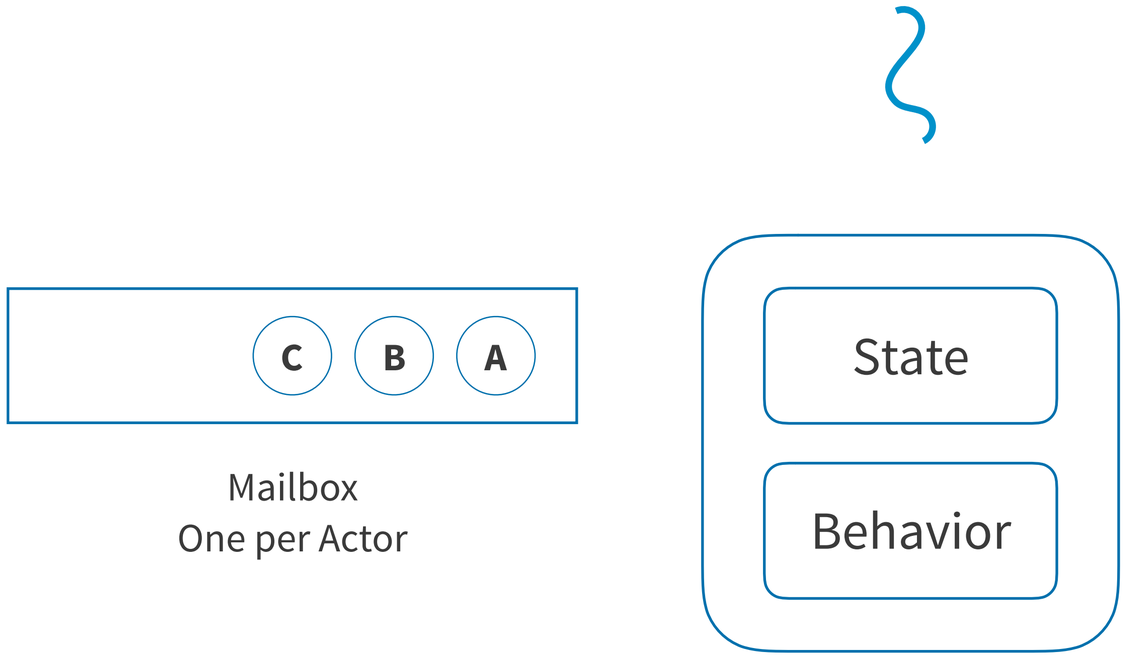
Connection Management

AN AKKA ACTOR



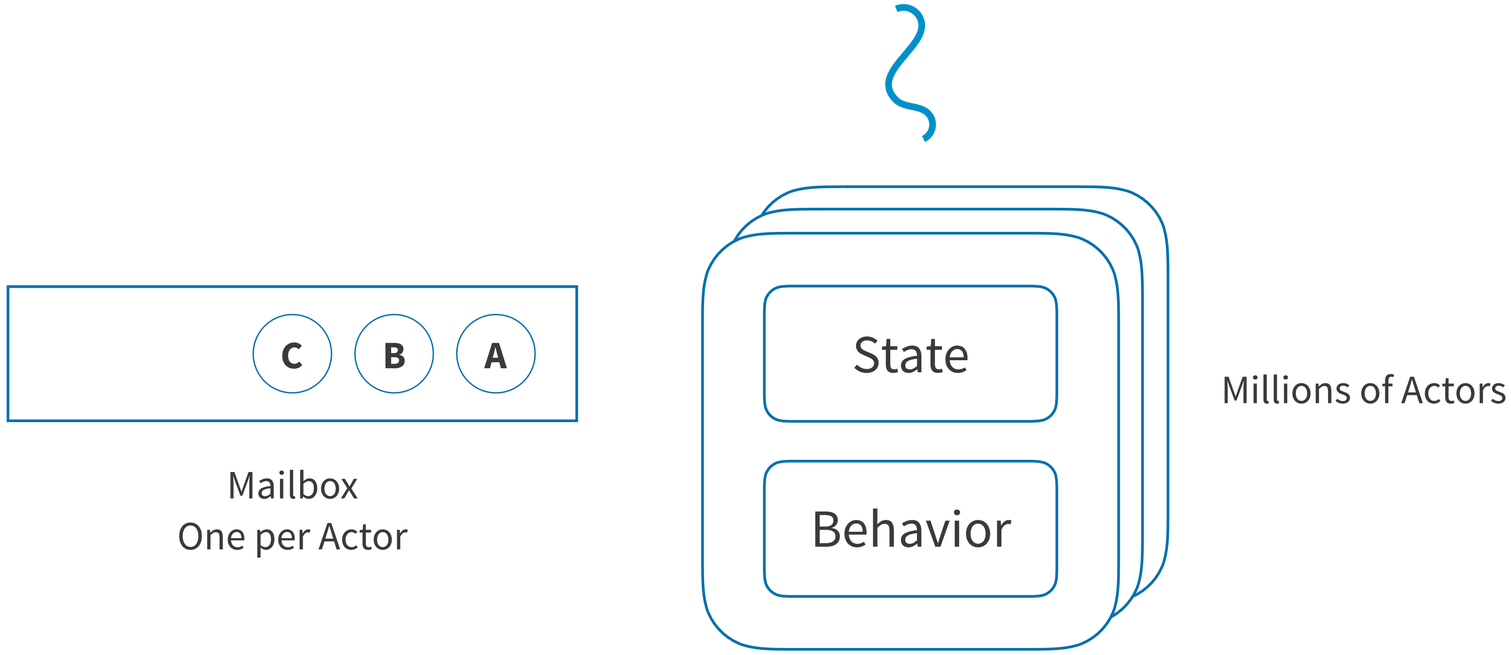
Connection Management

AN AKKA ACTOR



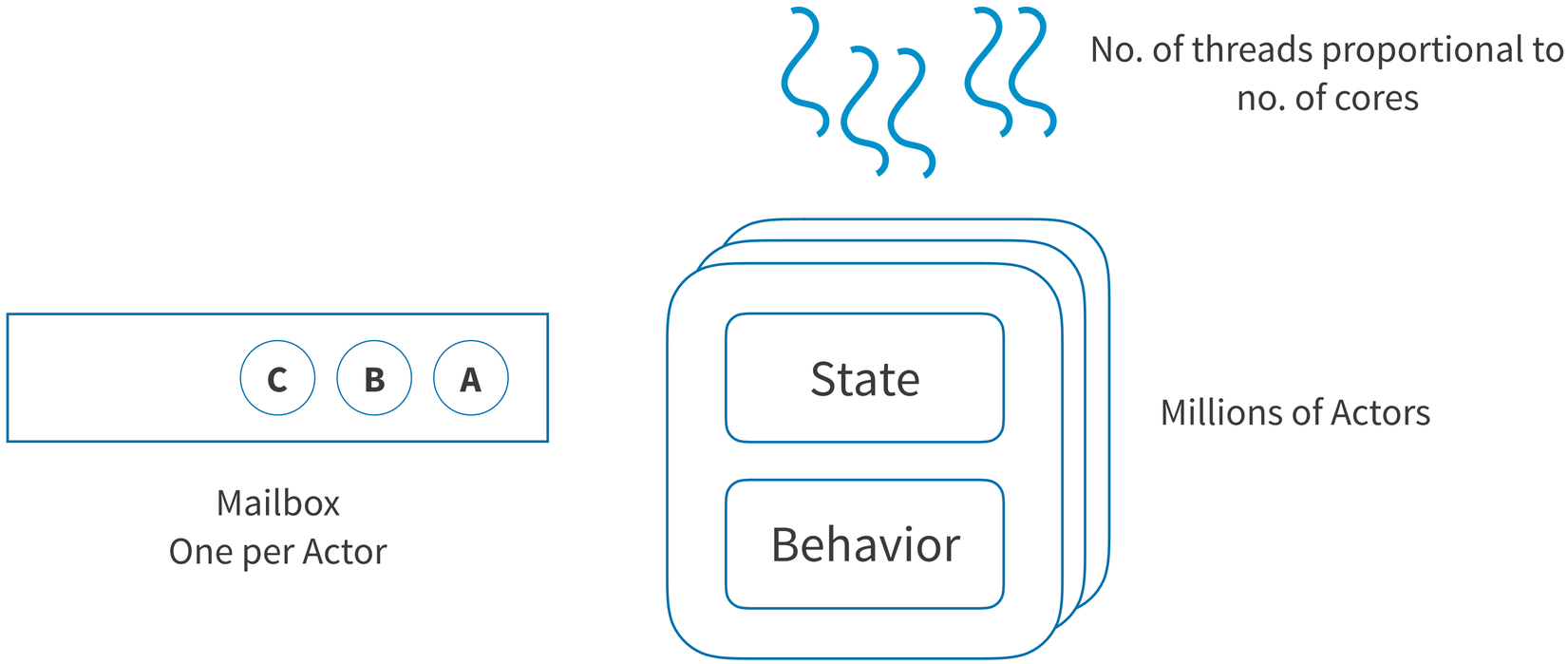
Connection Management

AN AKKA ACTOR



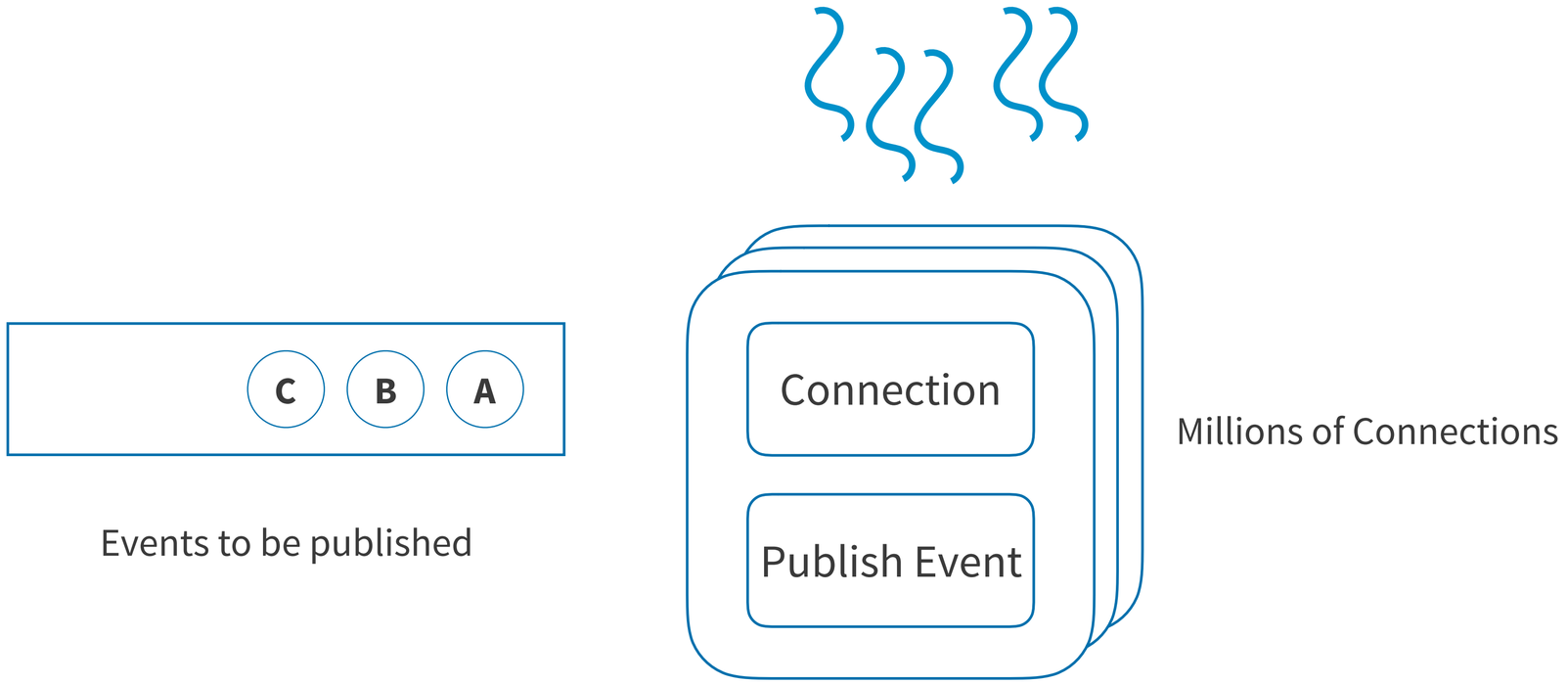
Connection Management

AN AKKA ACTOR



Connection Management

AN AKKA ACTOR



Server: Accepting a EventSource connection

INITIALIZE AN AKKA ACTOR

```
// Client A connects to the server and is assigned connectionIdA
public Result listen() {




}
```

Server: Accepting a EventSource connection

INITIALIZE AN AKKA ACTOR

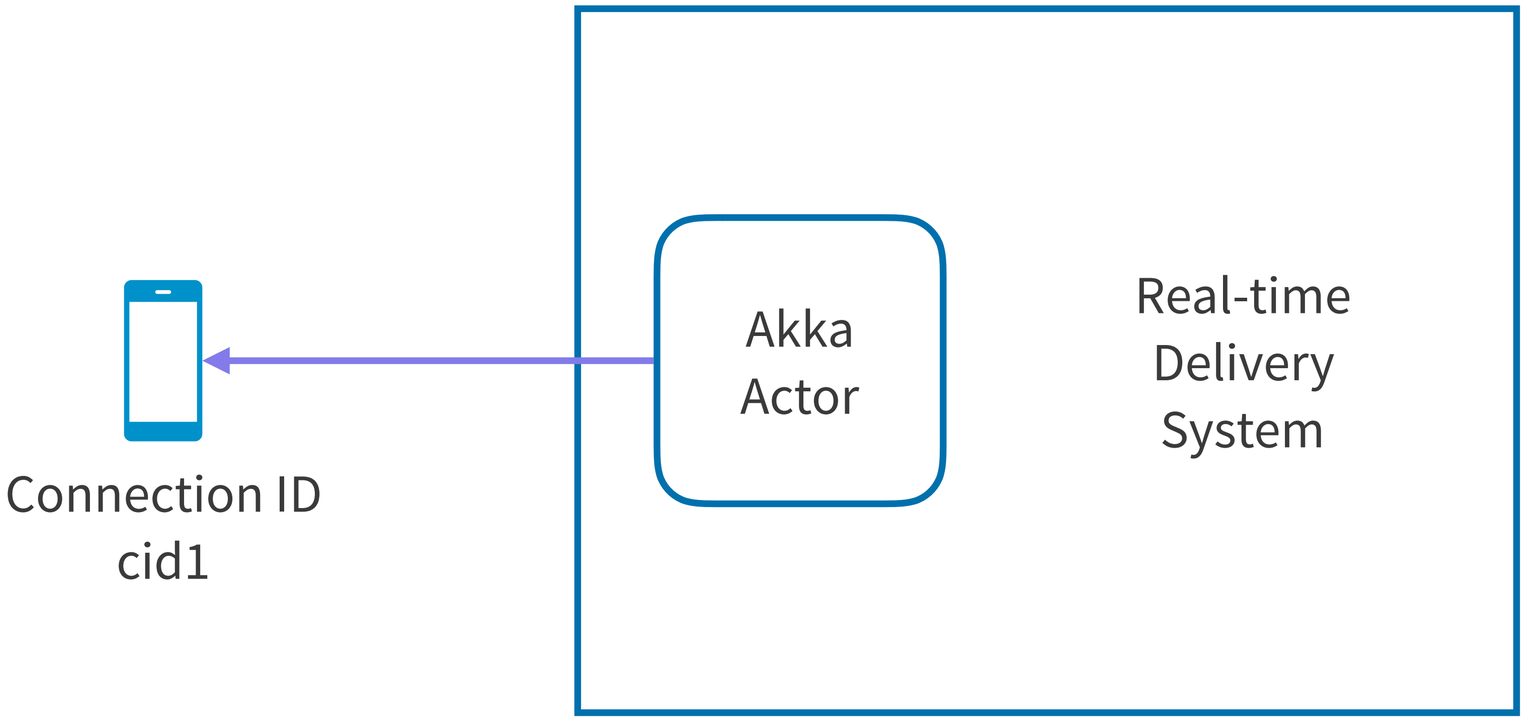
```
// Client A connects to the server and is assigned connectionIdA
public Result listen() {
 return ok(EventSource.whenConnected(eventSource -> {
   String connectionId = UUID.randomUUID().toString();









 }));
}
```

Server: Accepting a EventSource connection

INITIALIZE AN AKKA ACTOR

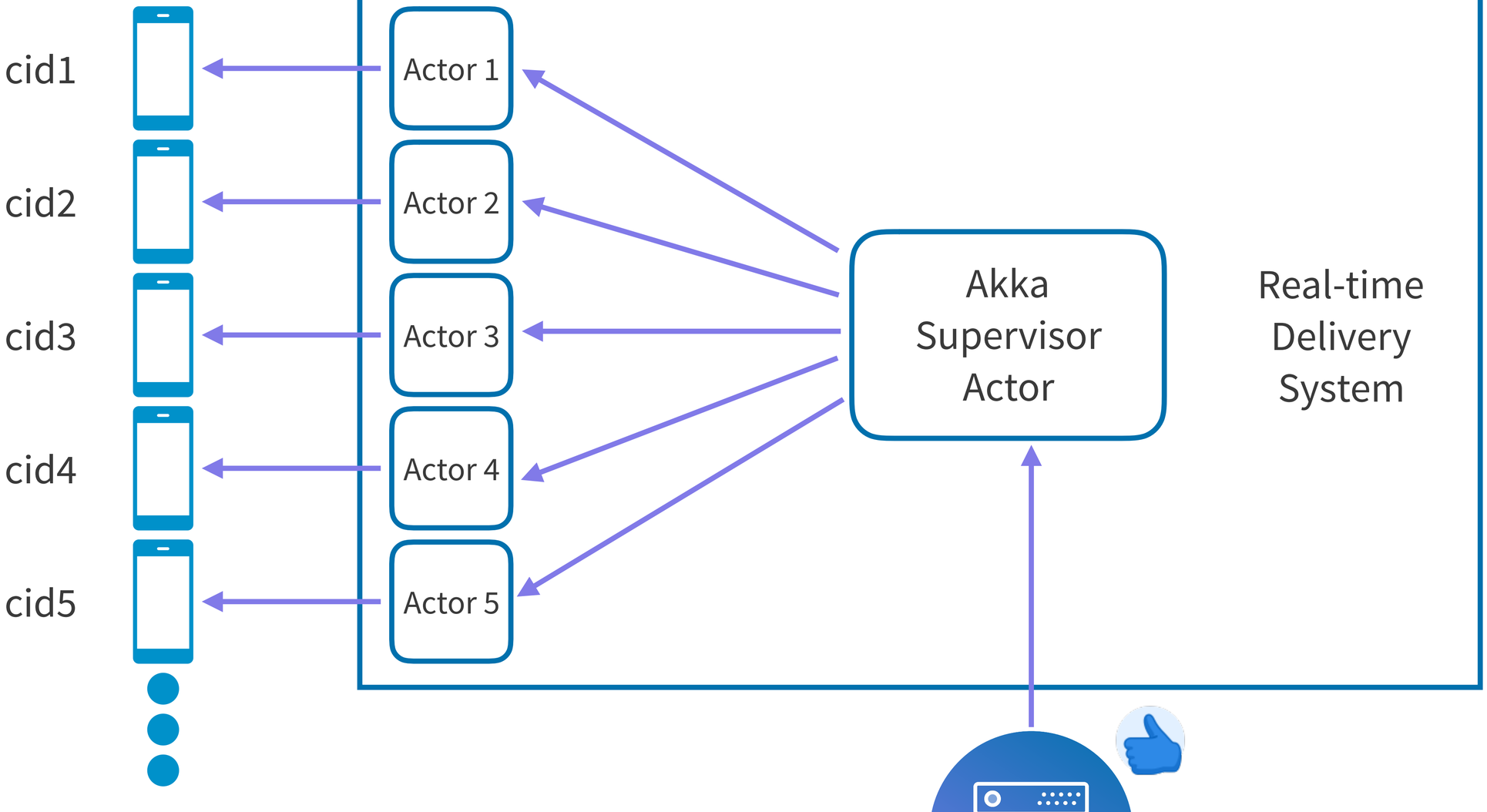
```
// Client A connects to the server and is assigned connectionIdA
public Result listen() {
 return ok(EventSource.whenConnected(eventSource -> {
   String connectionId = UUID.randomUUID().toString();
   // construct an Akka Actor to manage connection
   _actorSystem.actorOf(
     ClientConnectionActor.props(connectionId, eventSource),
     connectionId
   );
 }));
}
```

Publish Events Using Akka Actors



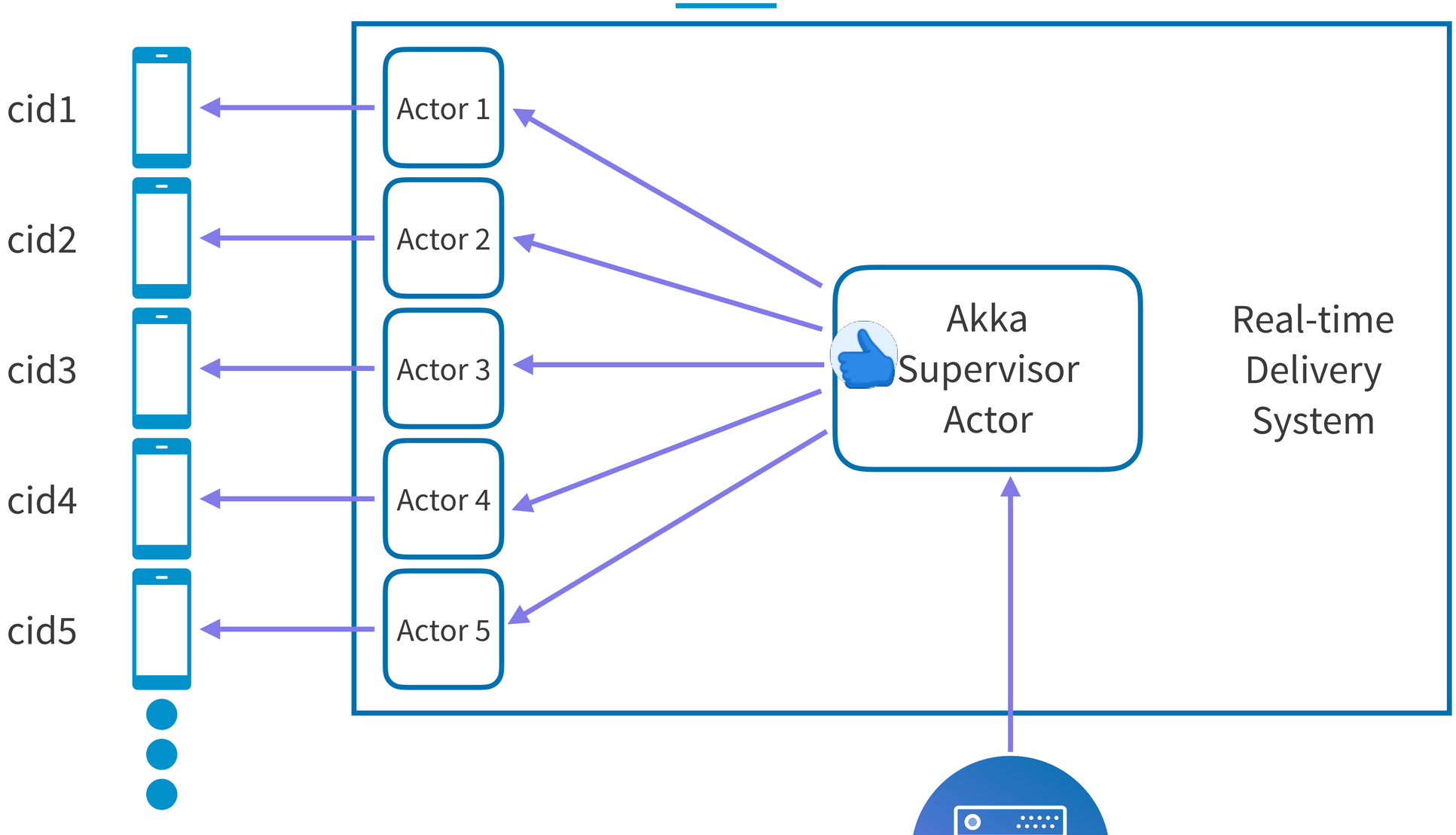
Publish Events Using Akka Actors

EVENT PUBLISH FROM PUBLISHER



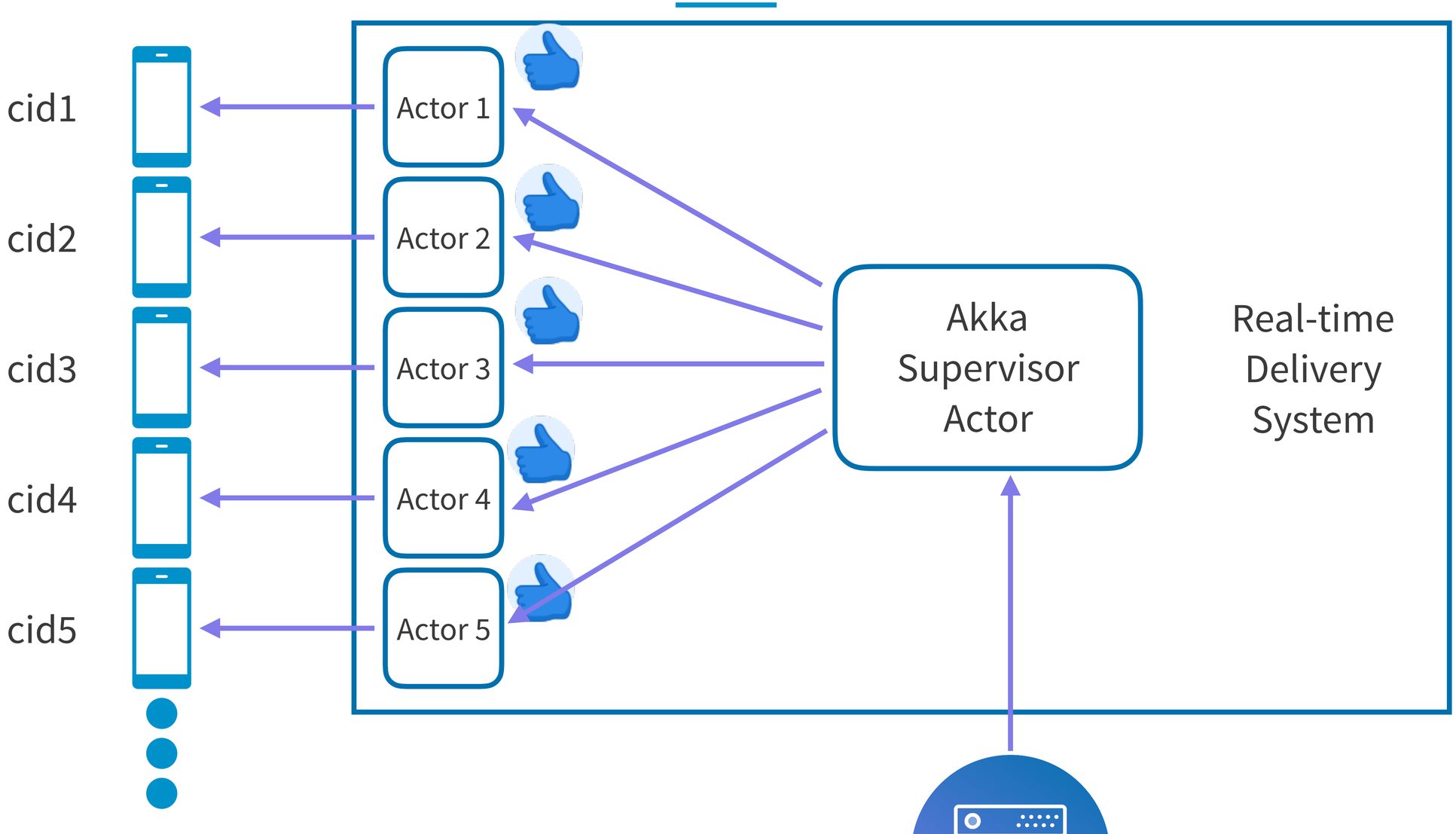
Publish Events Using Akka Actors

EVENT PUBLISH FROM SUPERVISOR TO CHILD ACTORS



Publish Events Using Akka Actors

EVENT PUBLISH FROM CHILD ACTORS TO CLIENT CONNECTIONS



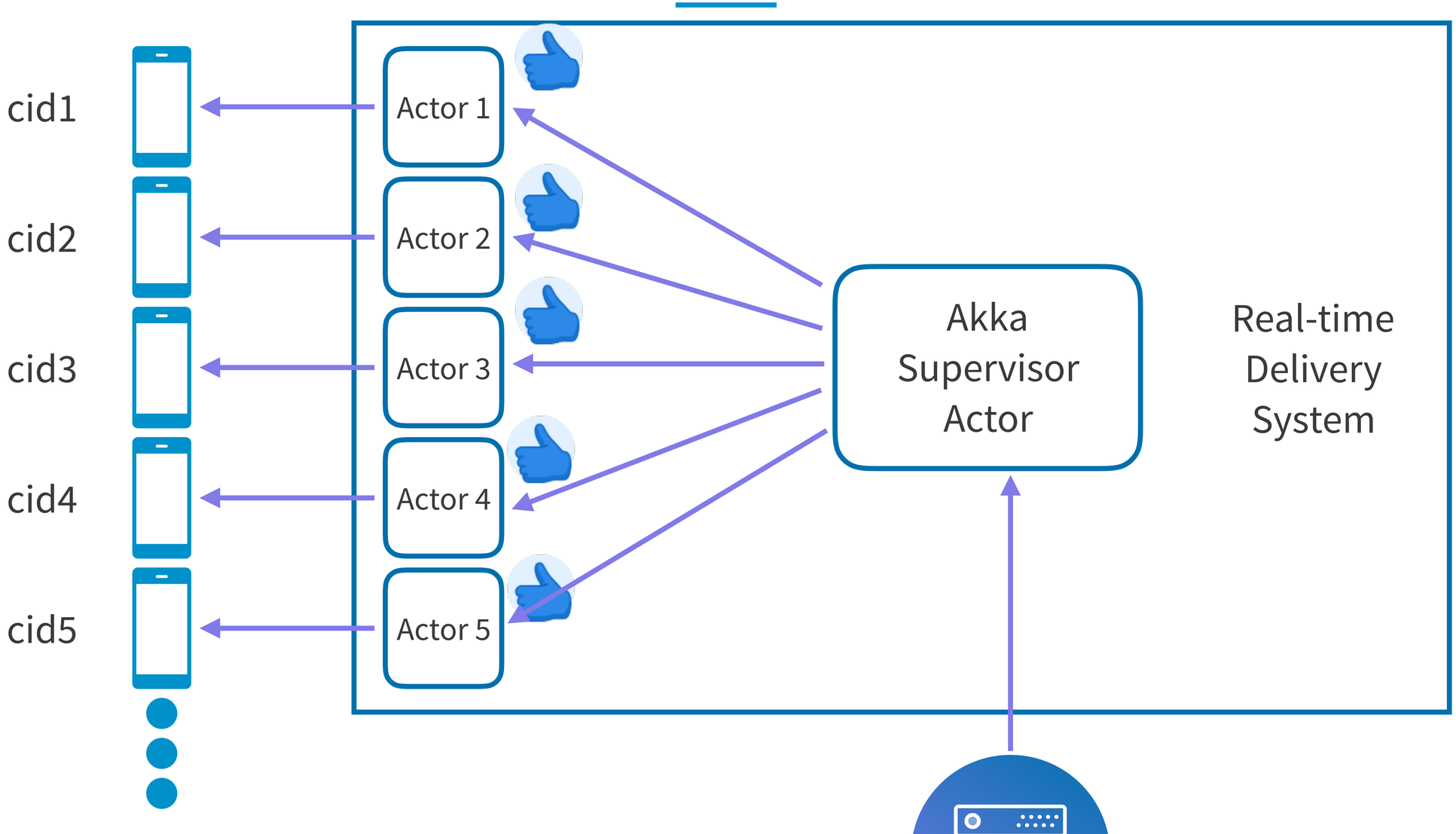
Publishing on EventSource

EVENT PUBLISH FROM CHILD ACTORS TO CLIENT CONNECTIONS

```
eventSource.send(<like object>);
```

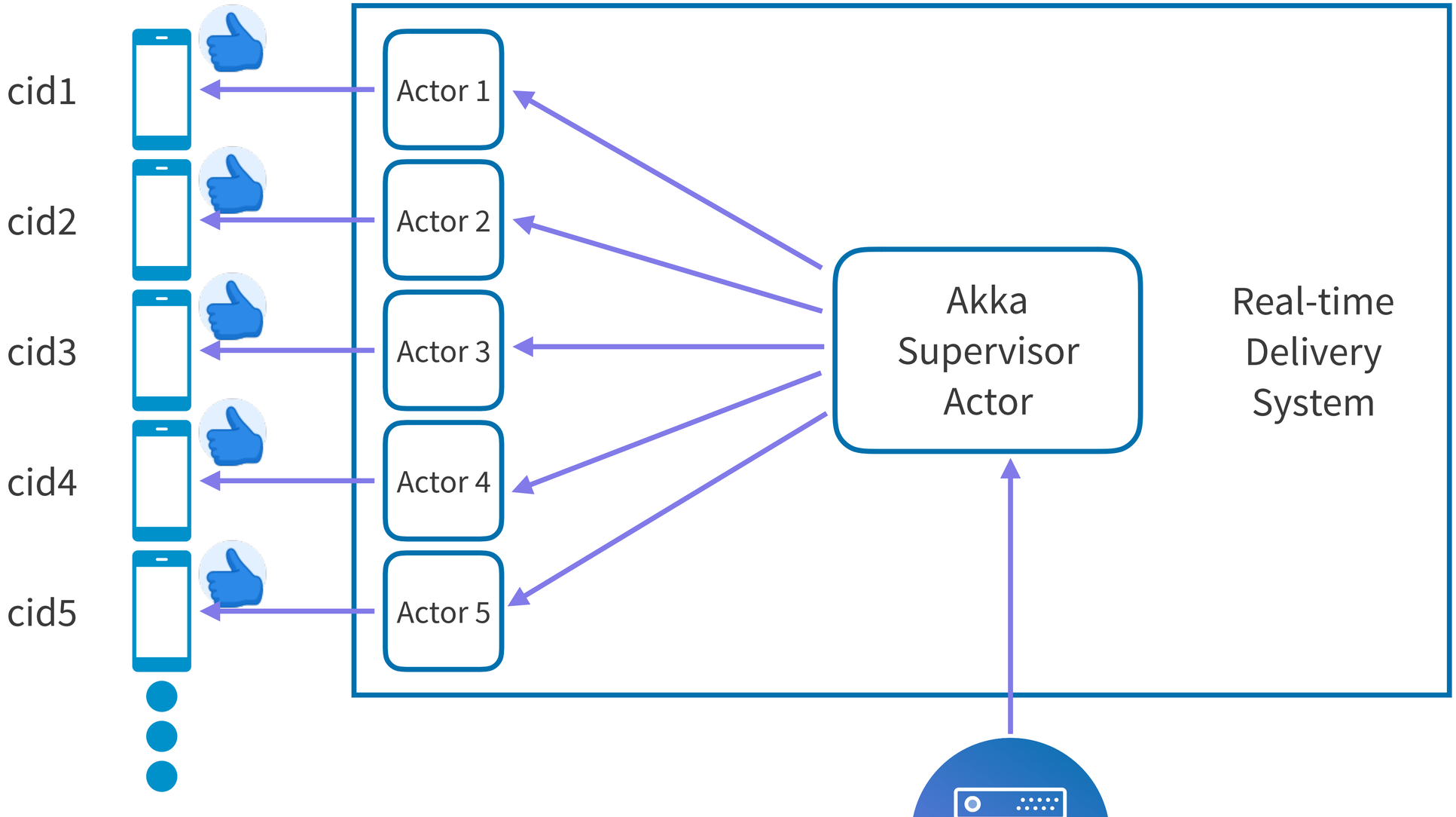
Publish Events Using Akka Actors

EVENT PUBLISH FROM CHILD ACTORS TO CLIENT CONNECTIONS



Publish Events Using Akka Actors

EVENT PUBLISH FROM CHILD ACTORS TO CLIENT CONNECTIONS

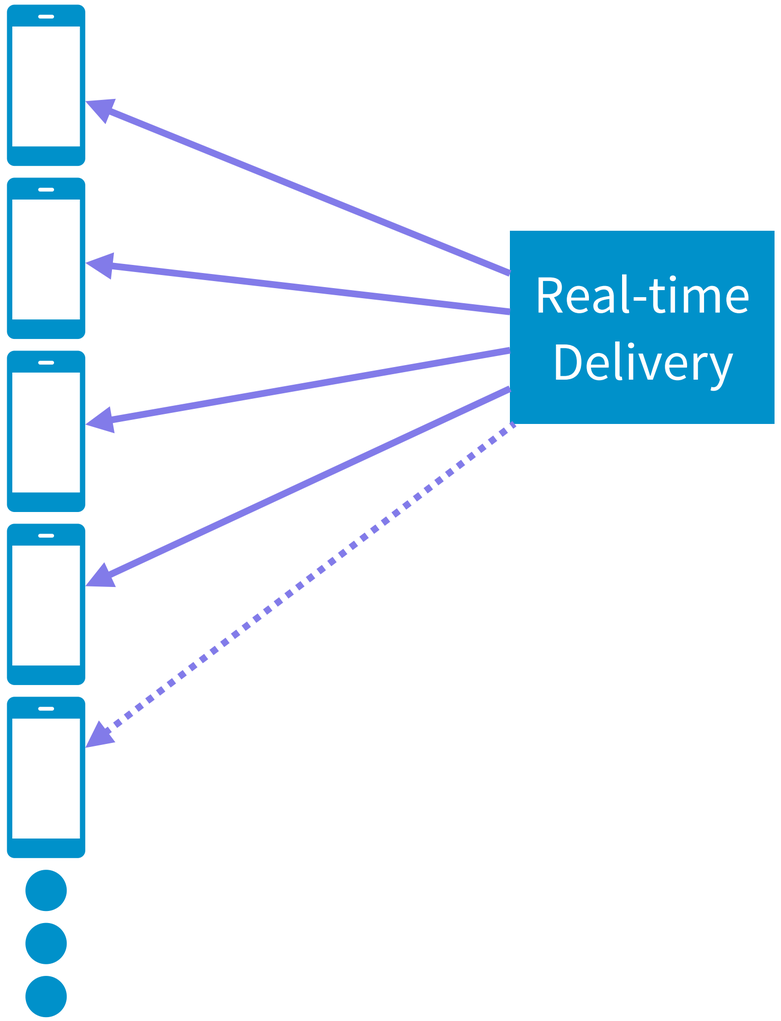


Received Events on the Clients

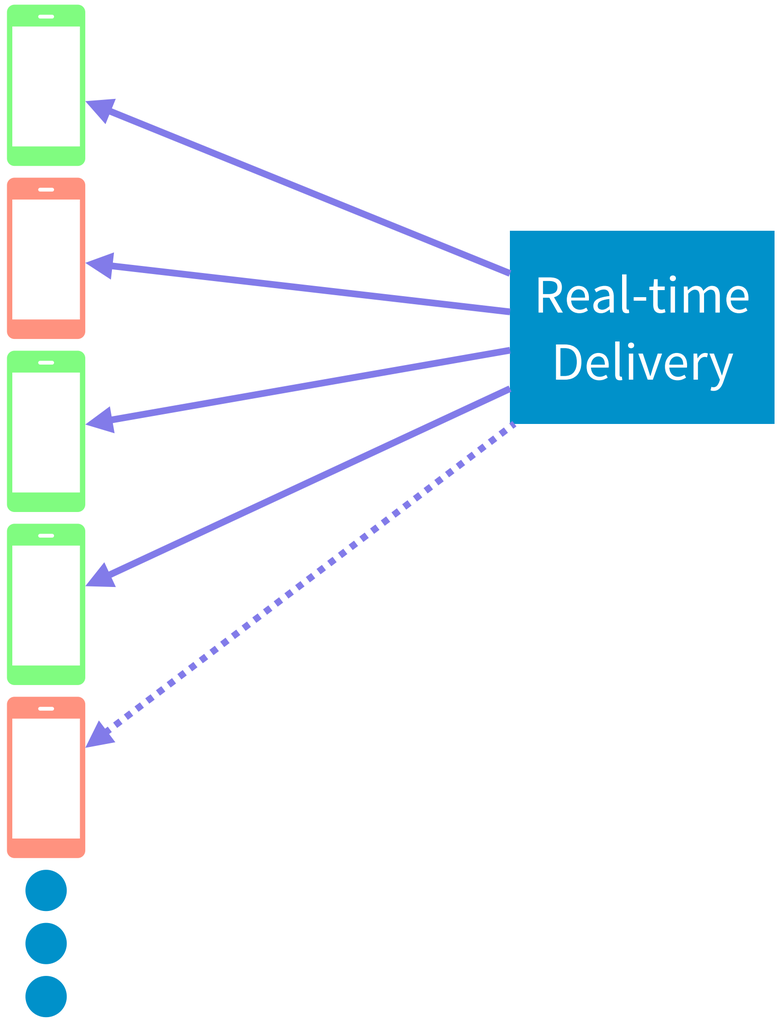
EVENTSOURCE INTERFACE

```
HTTP/1.1 200 OK
- RESPONSE HEADERS -
Content-Type: text/event-stream

- RESPONSE BODY -

data: {“User A liked the video…” object}
```

Next Challenge?



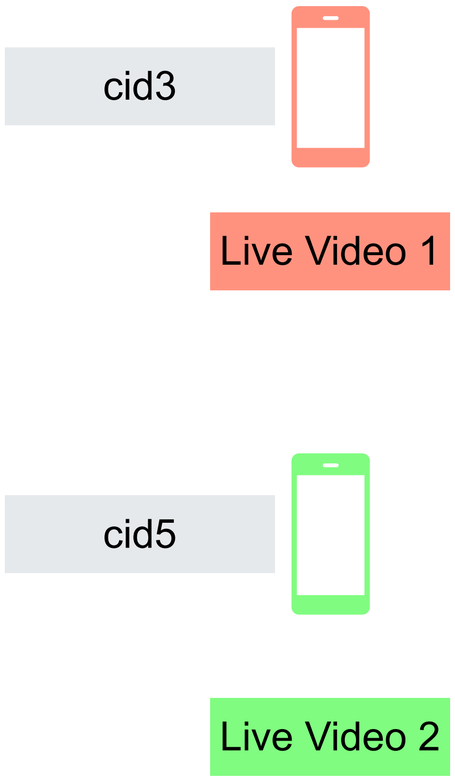
Next Challenge: Multiple Live Videos

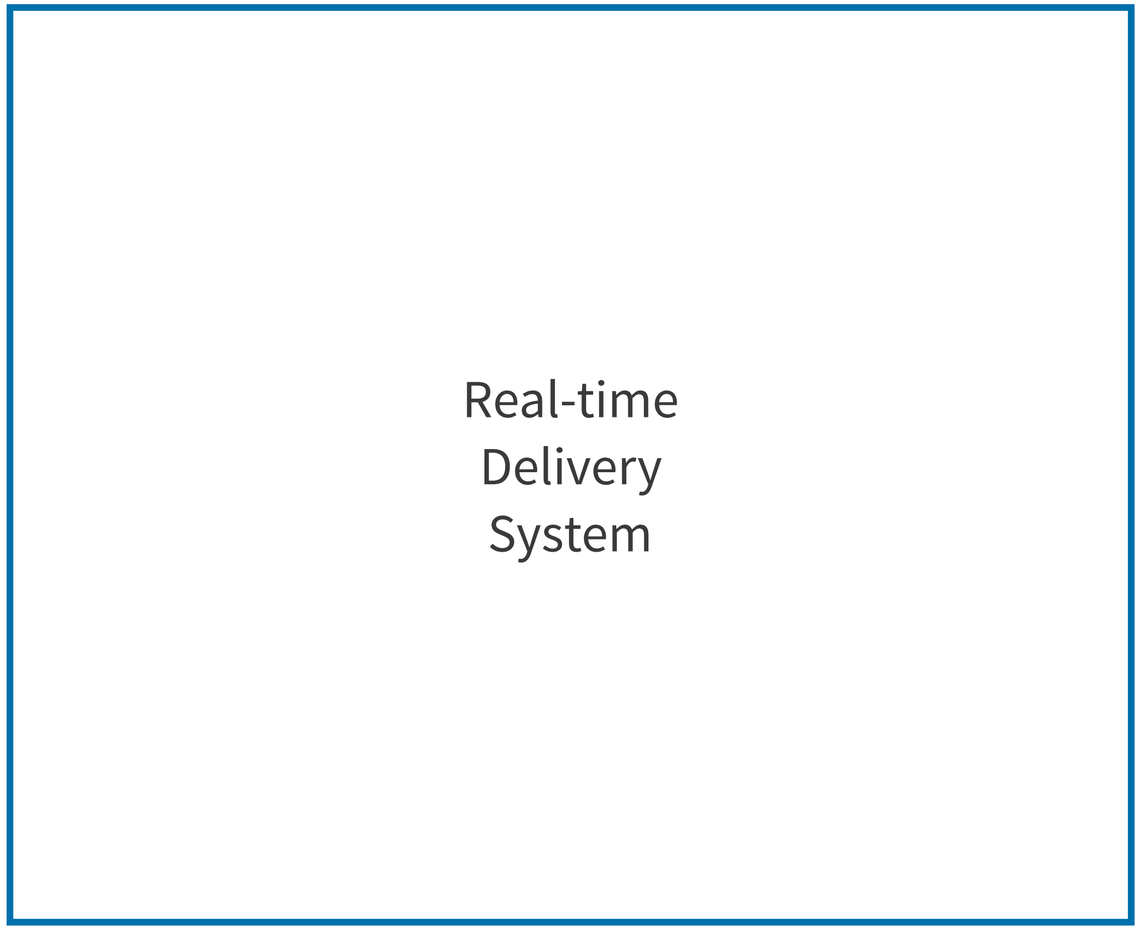


Challenge 3: Multiple Live Videos

Multiple Live Videos

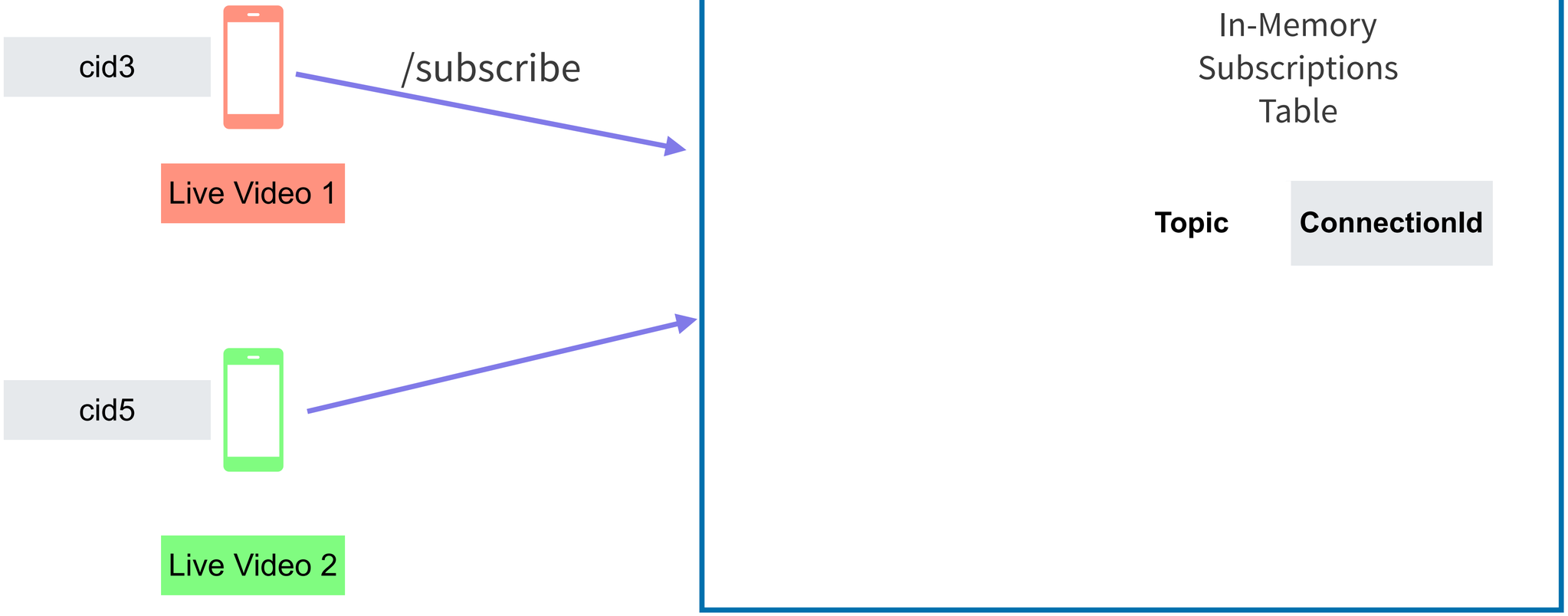
START WATCHING LIVE VIDEO





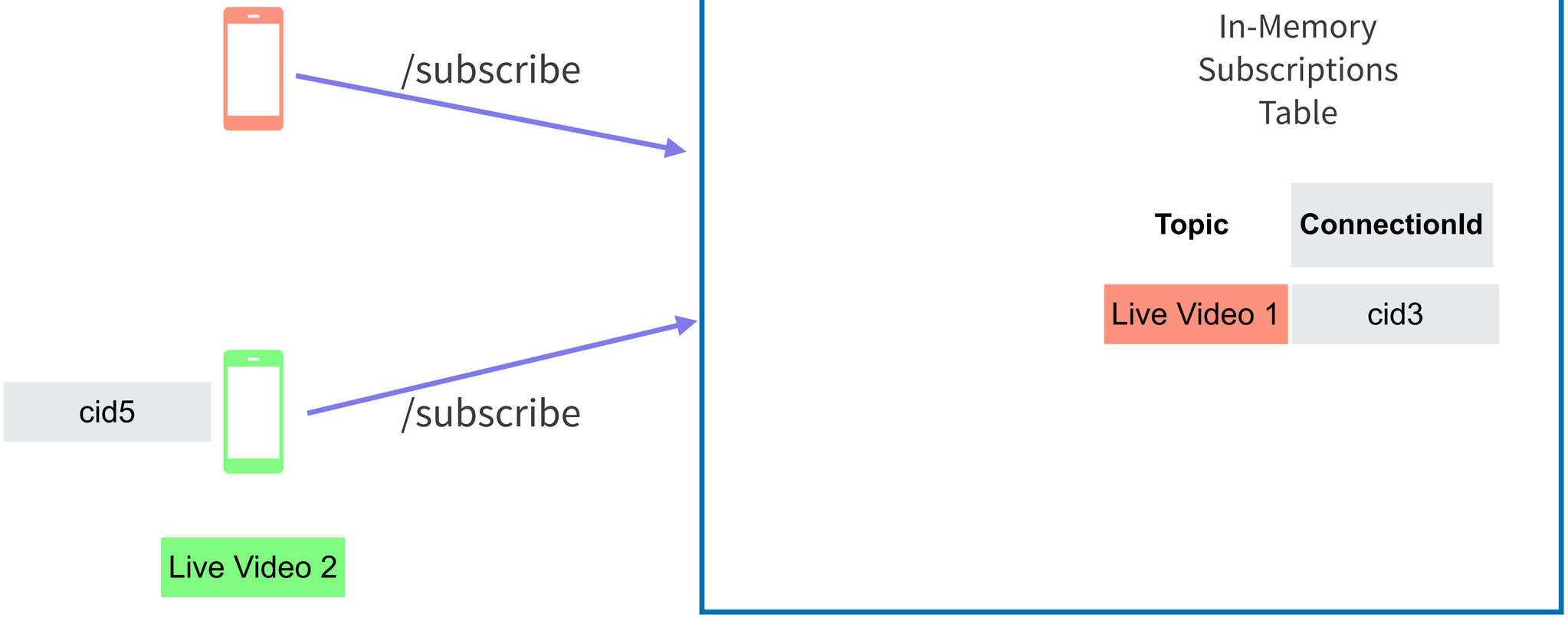
Viewers subscribe to Live Videos

HTTP SUBSCRIPTION REQUEST



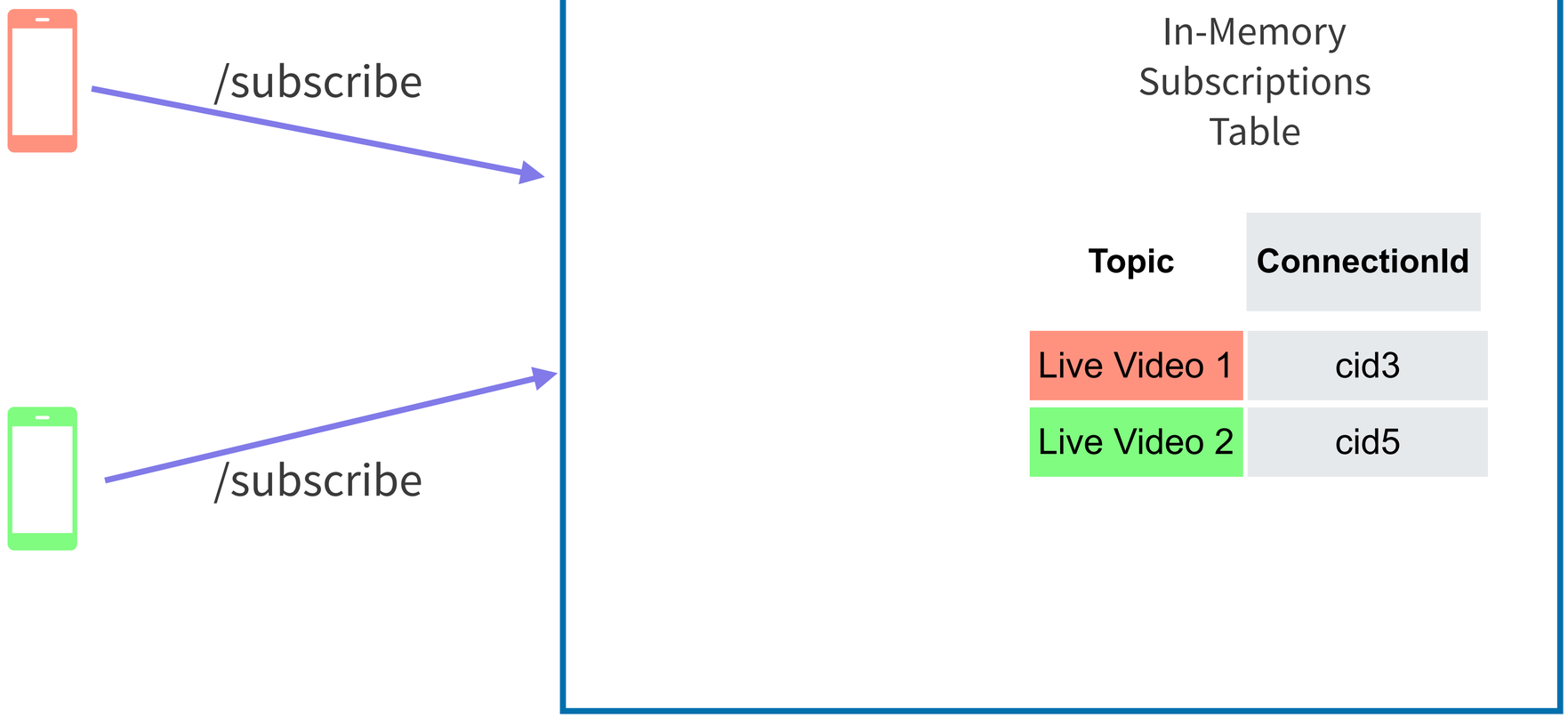
Viewers subscribe to Live Videos

IN-MEMORY SUBSCRIPTION ENTRY

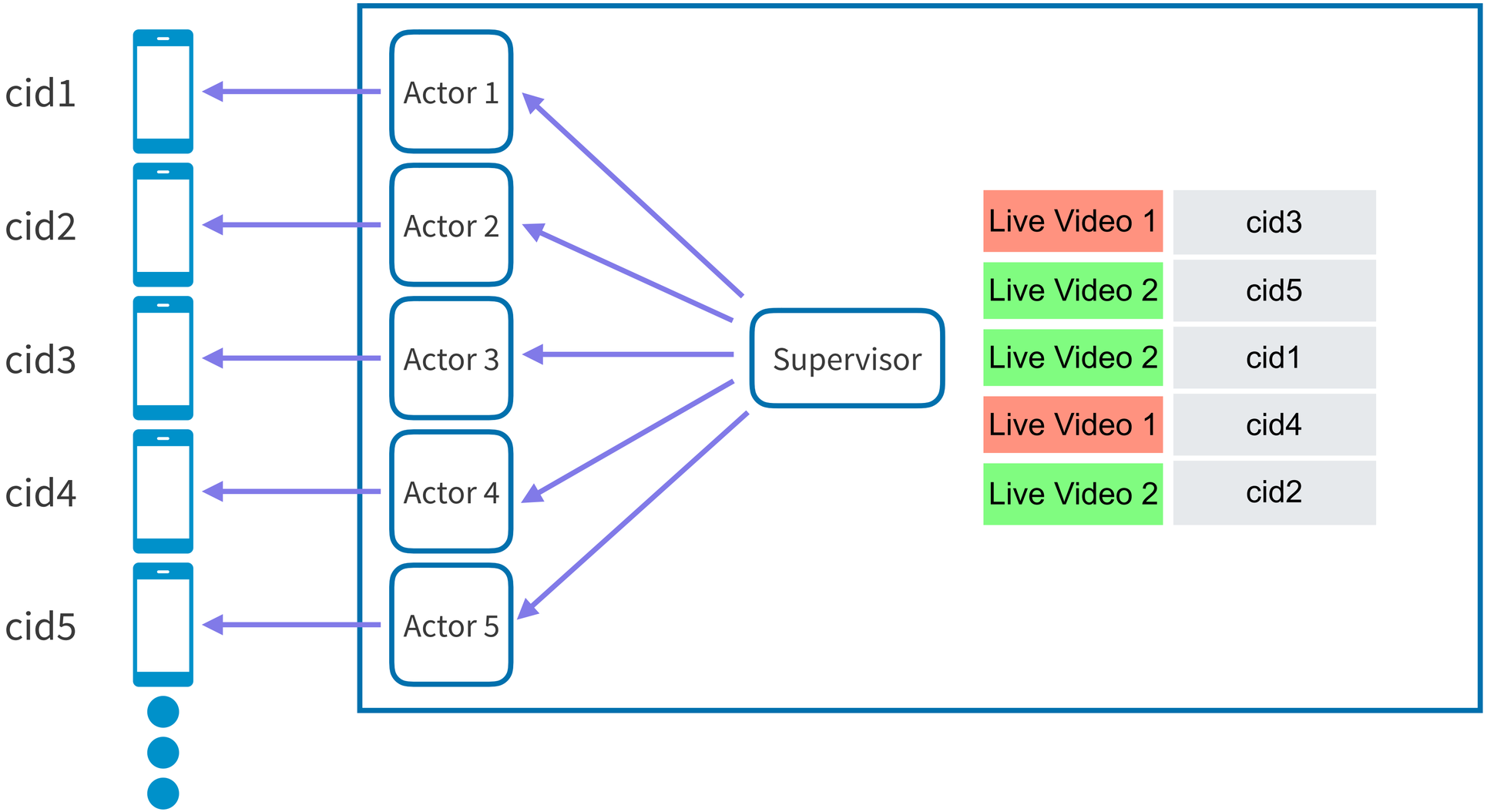


Viewers subscribe to Live Videos

IN-MEMORY SUBSCRIPTION ENTRY

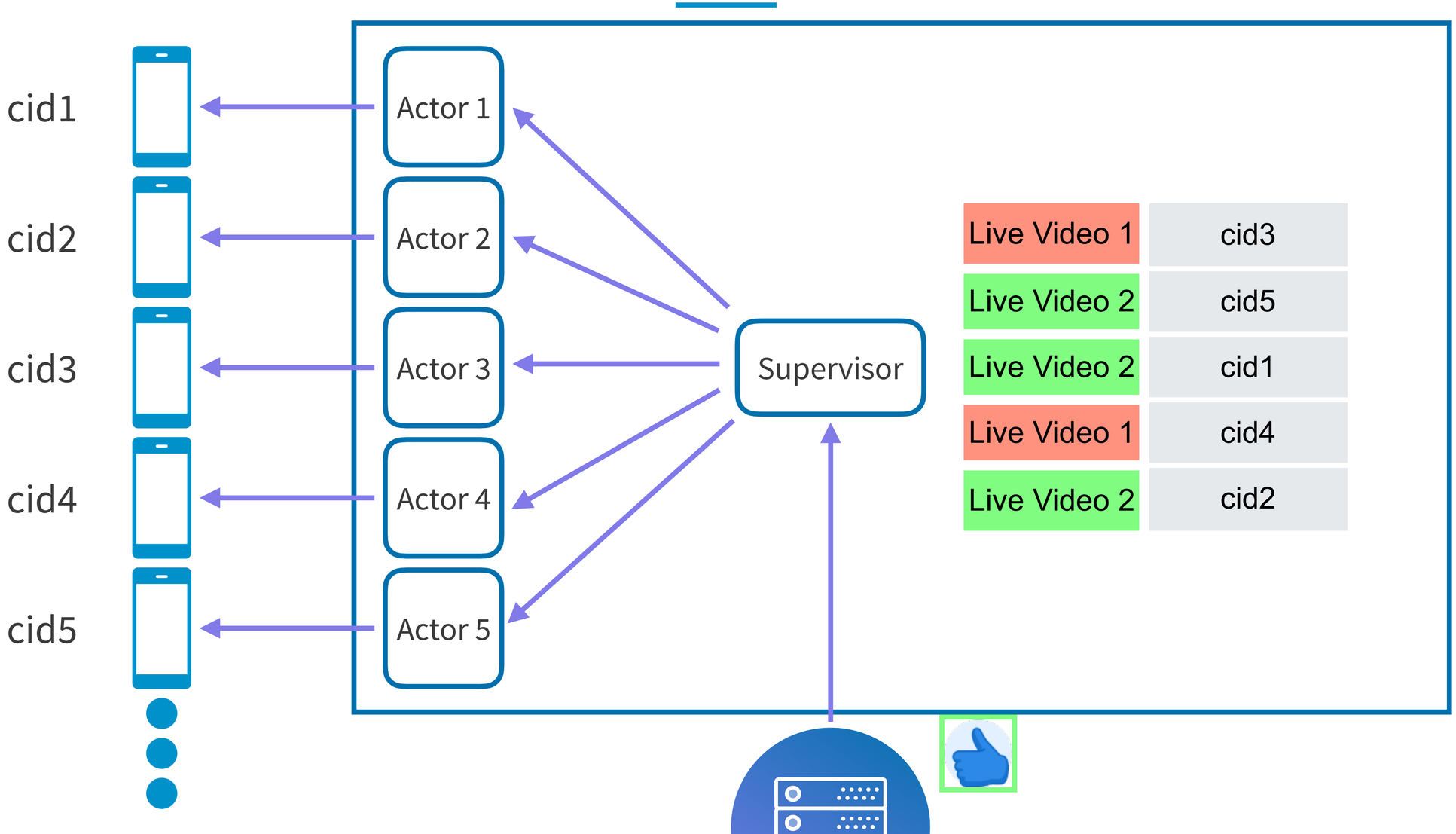


Publish using Subscription



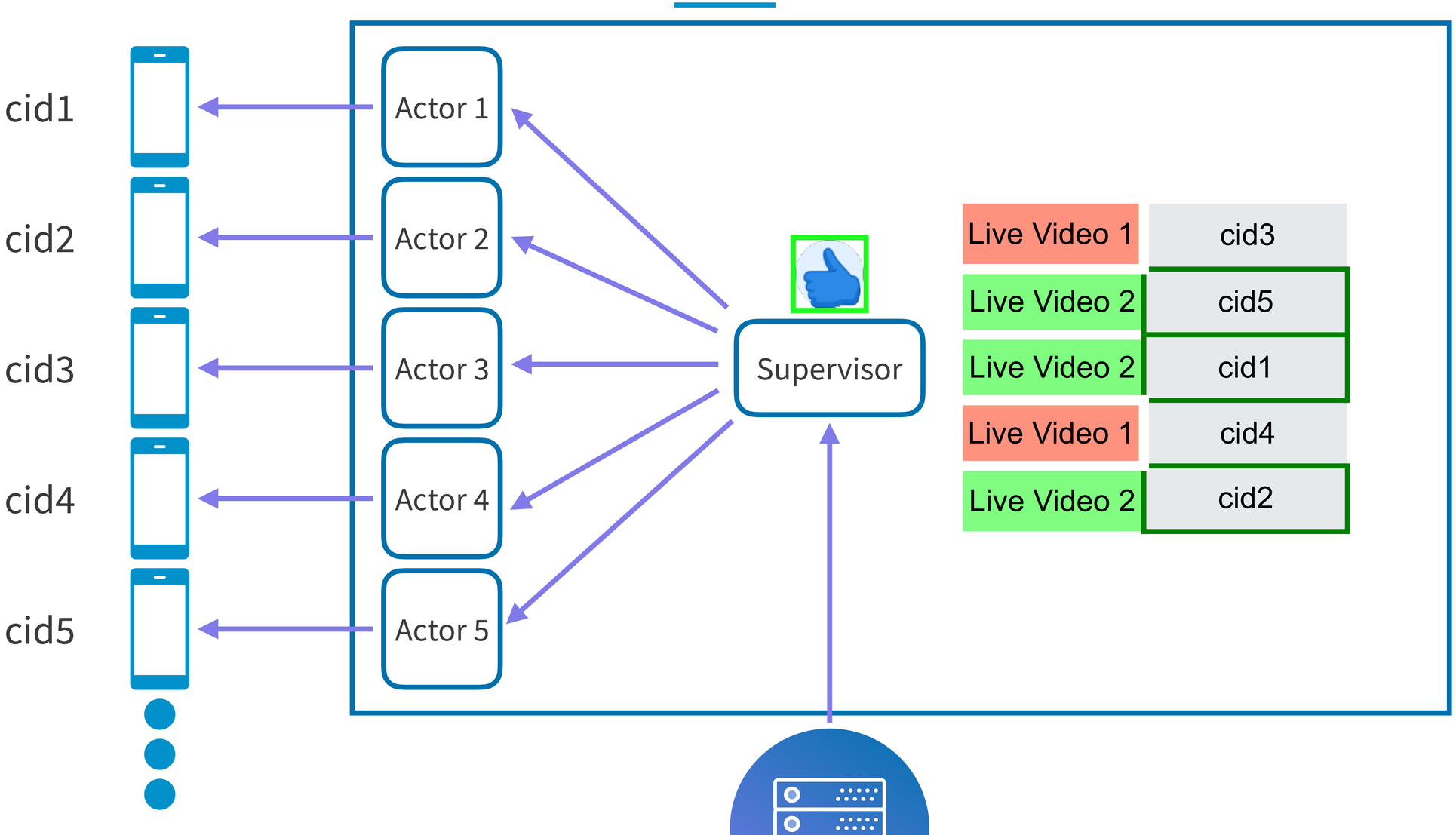
Publish using Subscription

PUBLISHER PUBLISH TO SUPERVISOR ACTOR



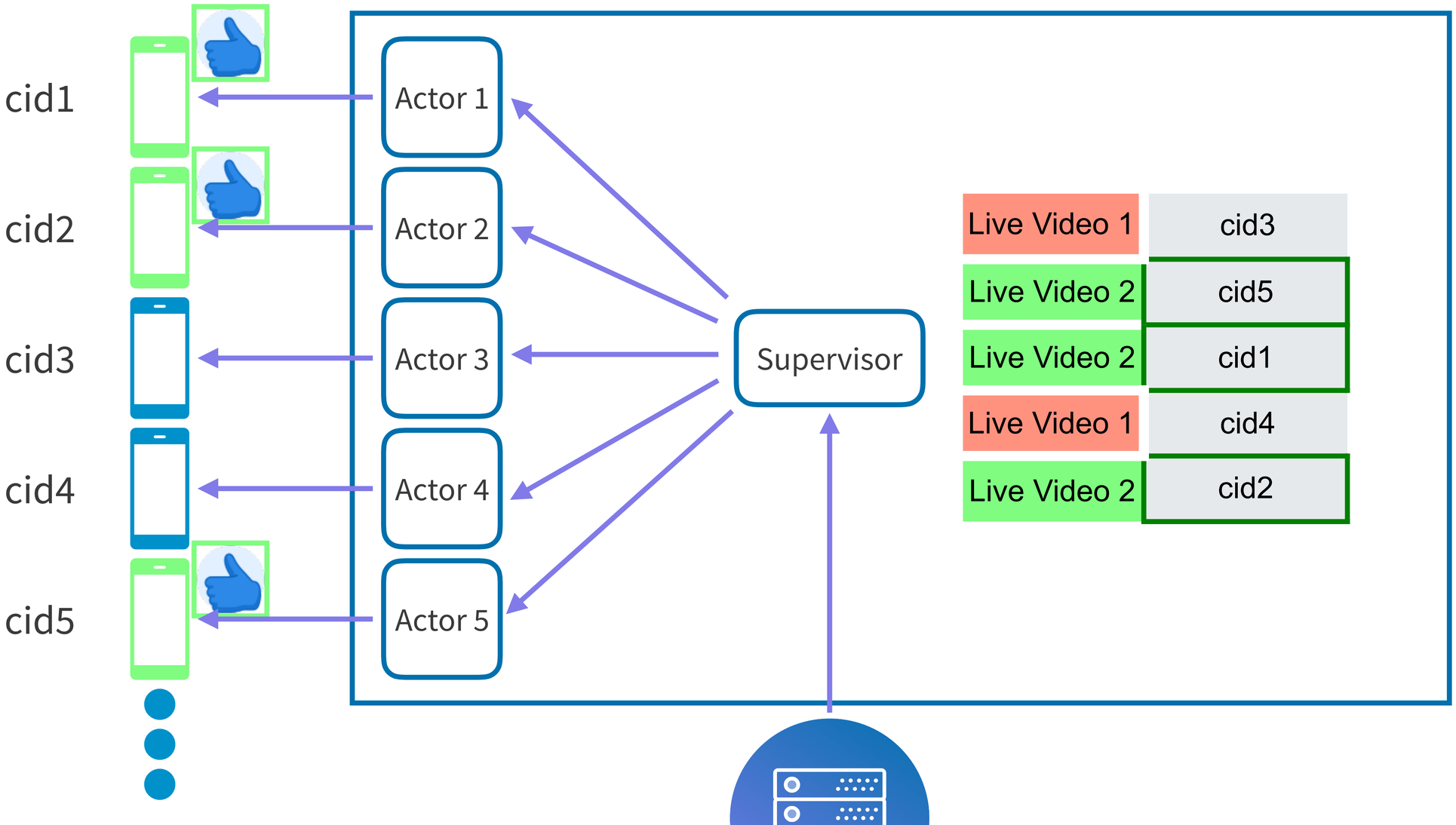
Publish using Subscription

SUBSCRIBER LOOKUP FROM IN MEMORY TABLE



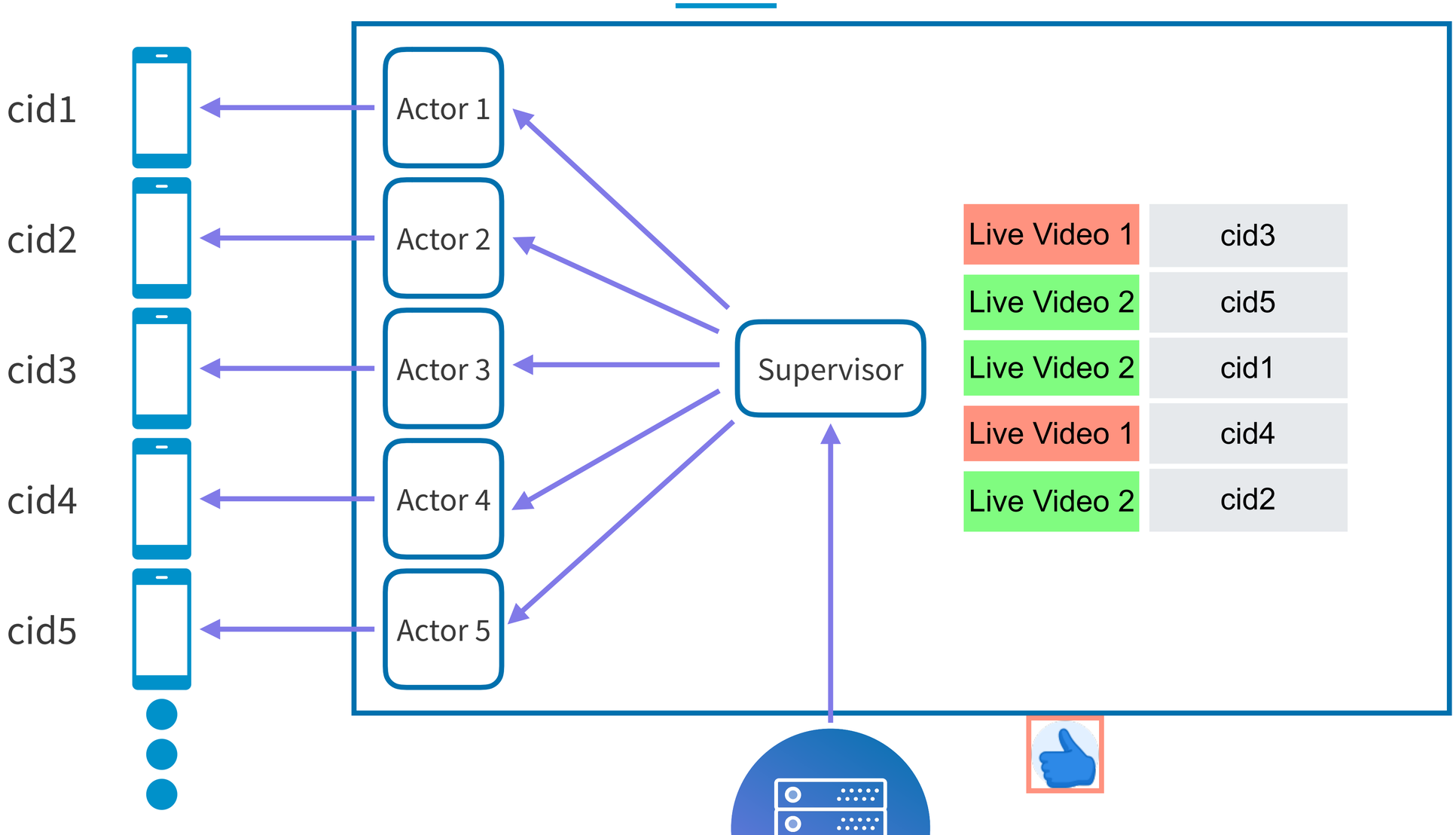
Publish using Subscription

PUBLISH TO CLIENT CONNECTIONS



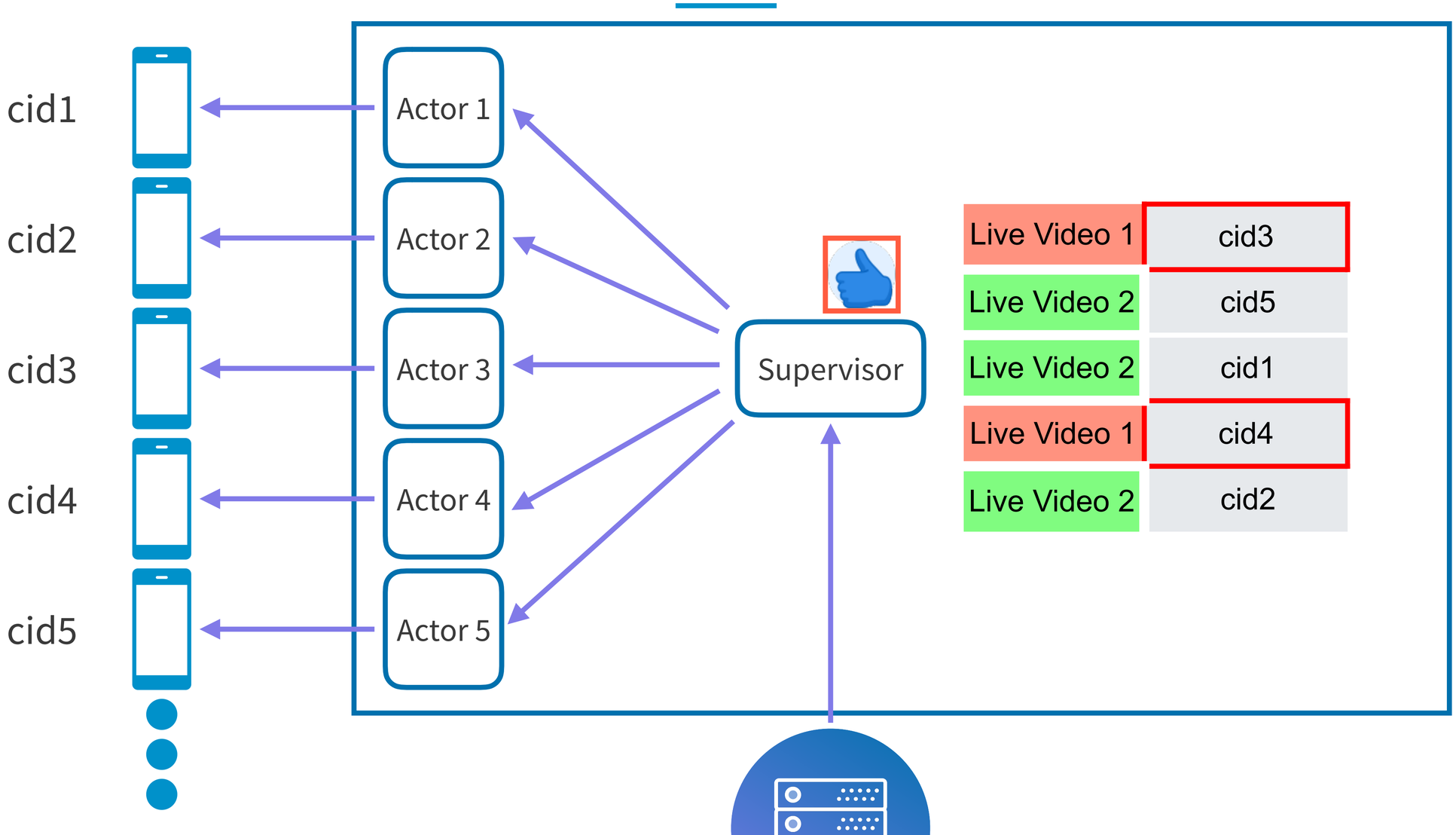
Publish using Subscription

PUBLISHER PUBLISH TO SUPERVISOR ACTOR



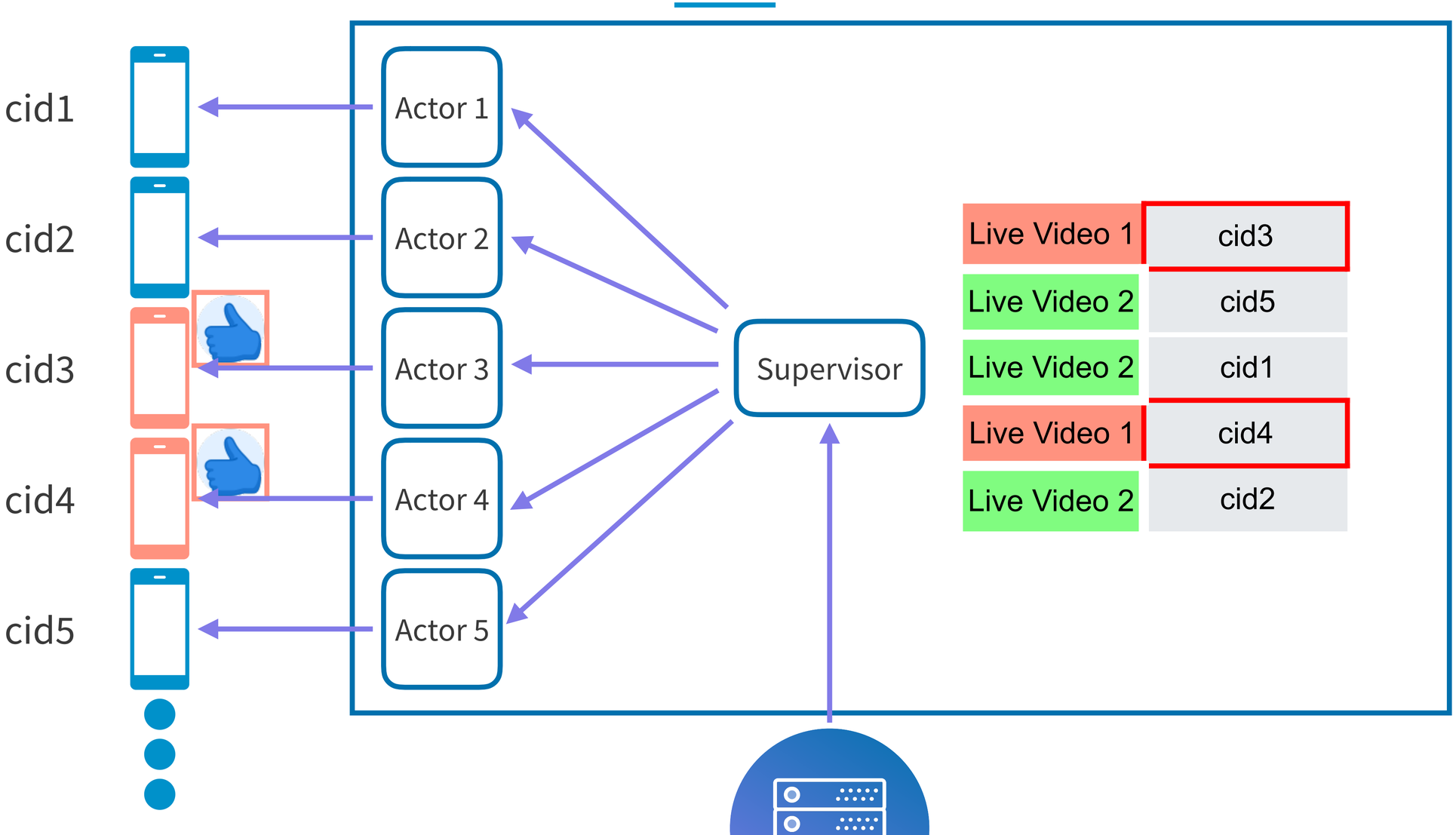
Publish using Subscription

SUBSCRIBER LOOKUP FROM IN MEMORY TABLE

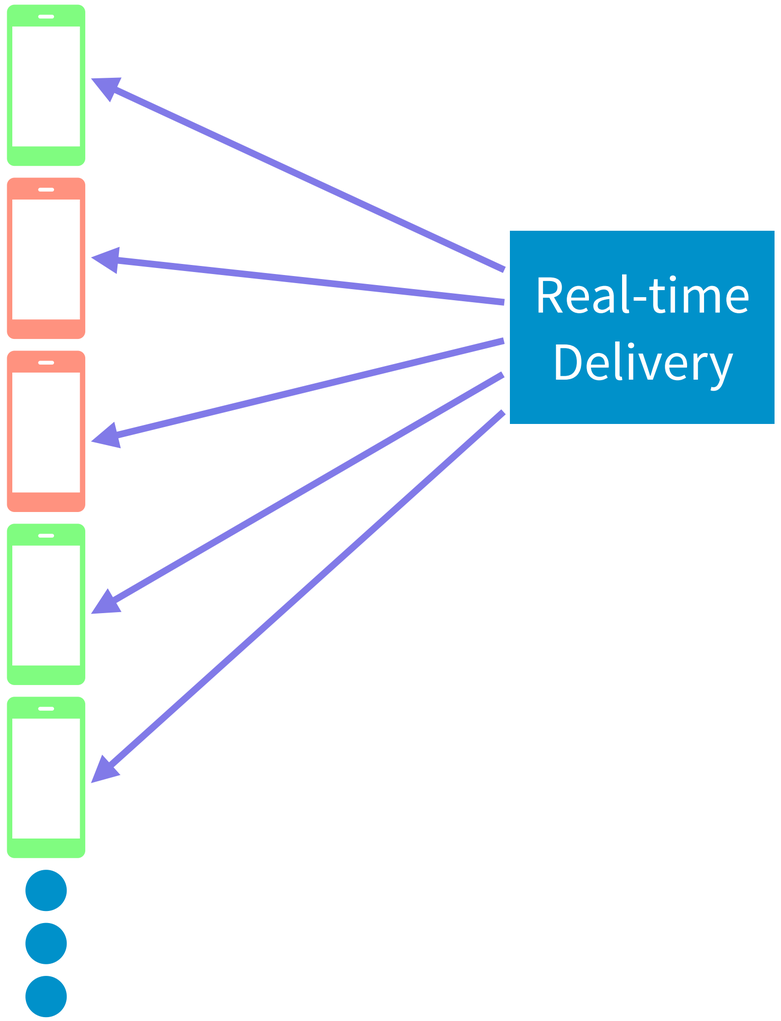


Publish using Subscription

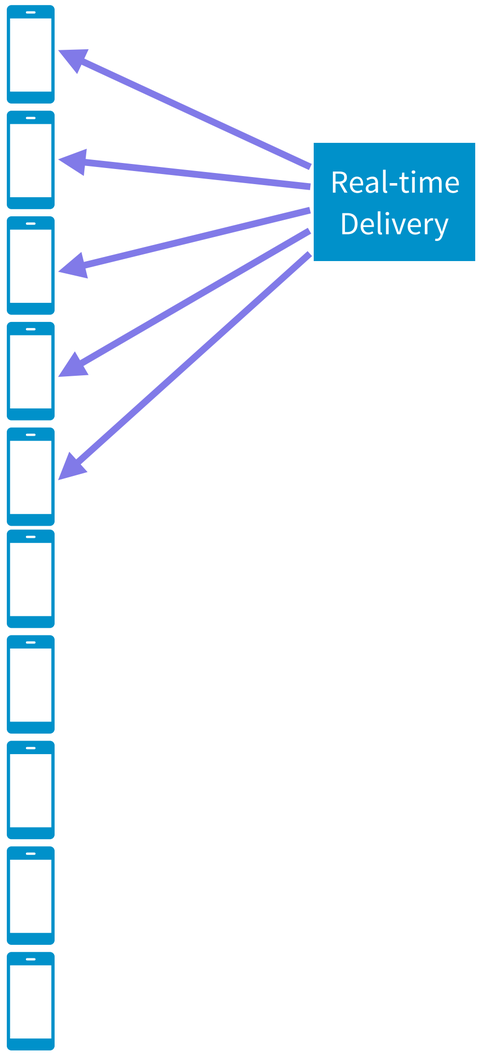
PUBLISH TO CLIENT CONNECTIONS



Next Challenge?



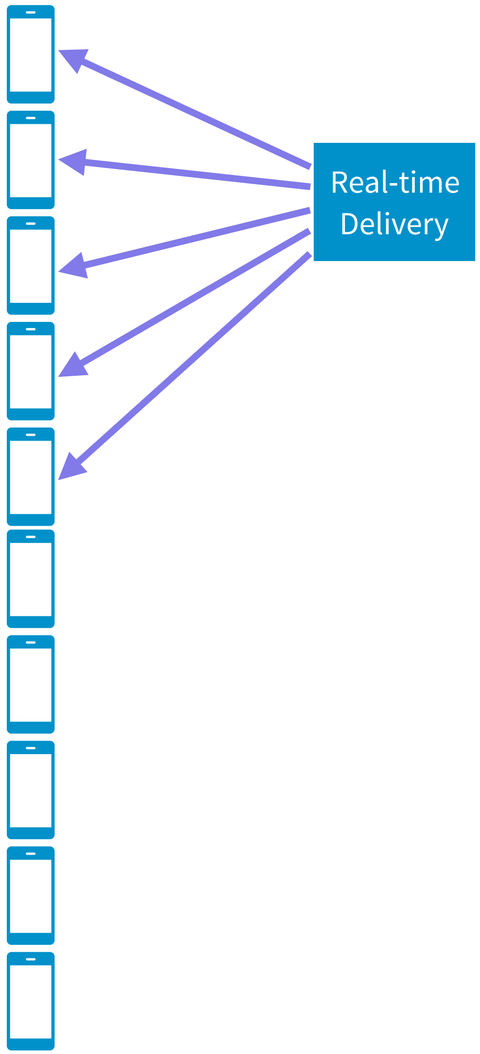
Next Challenge: 10K Concurrent Viewers



Challenge 4: 10K Concurrent Viewers


PUMA
AMERICAN EAGLE
UNREAL
REAL TIME
REAL TIME
REAL TIME

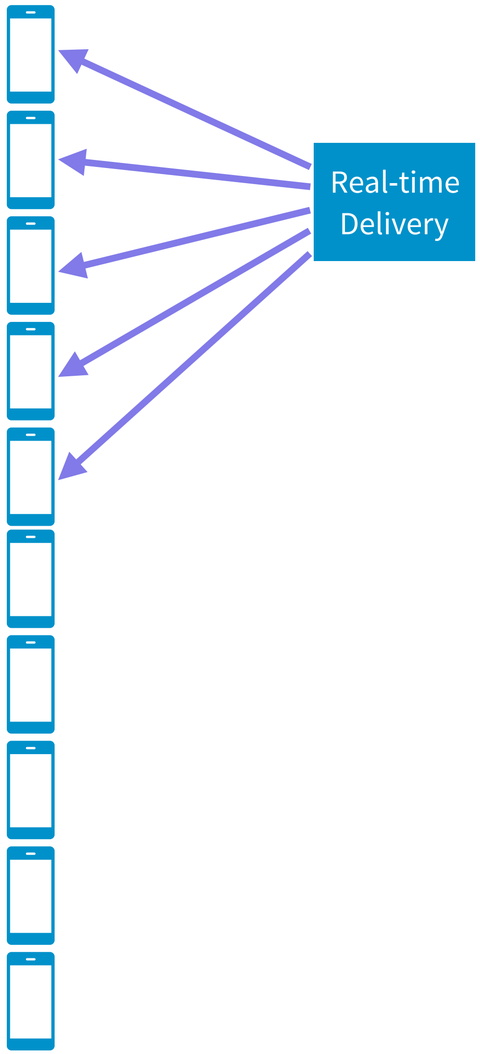
10K Concurrent Viewers



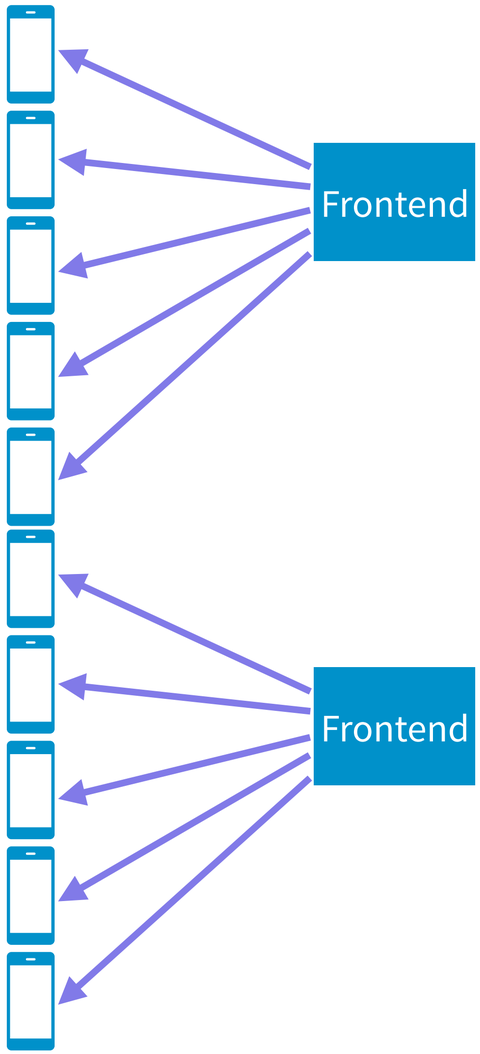
SCALING BACKEND SYSTEMS

When in doubt, add a machine!

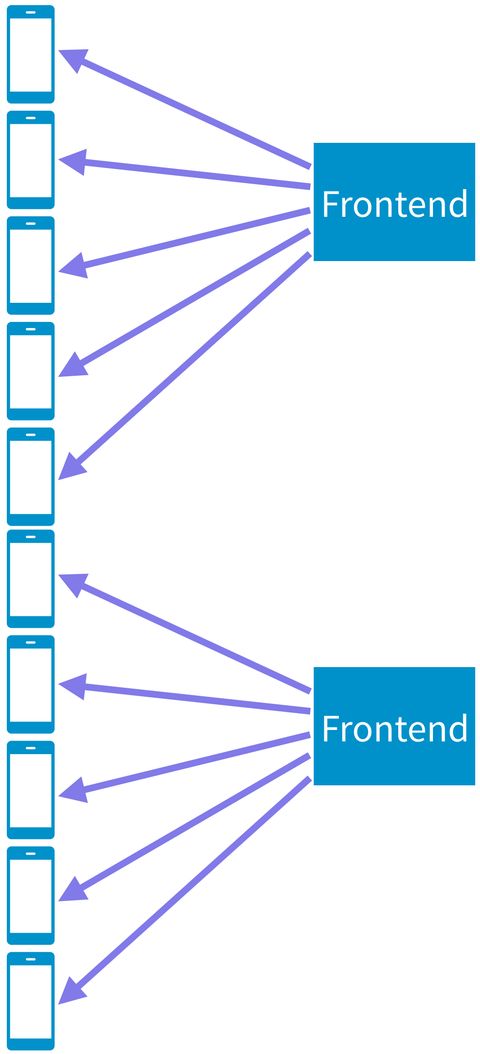
10K Concurrent Viewers



10K Concurrent Viewers

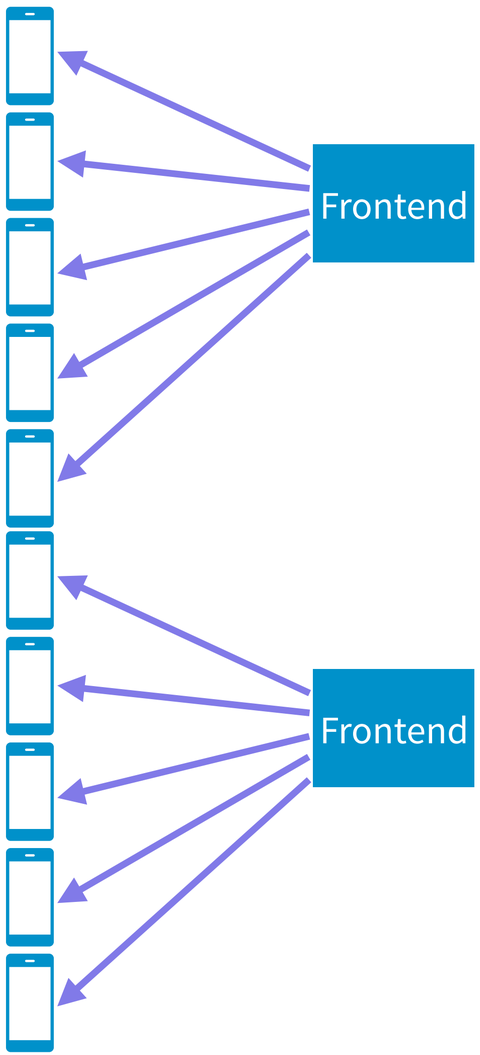


10K Concurrent Viewers



Real-time Dispatcher

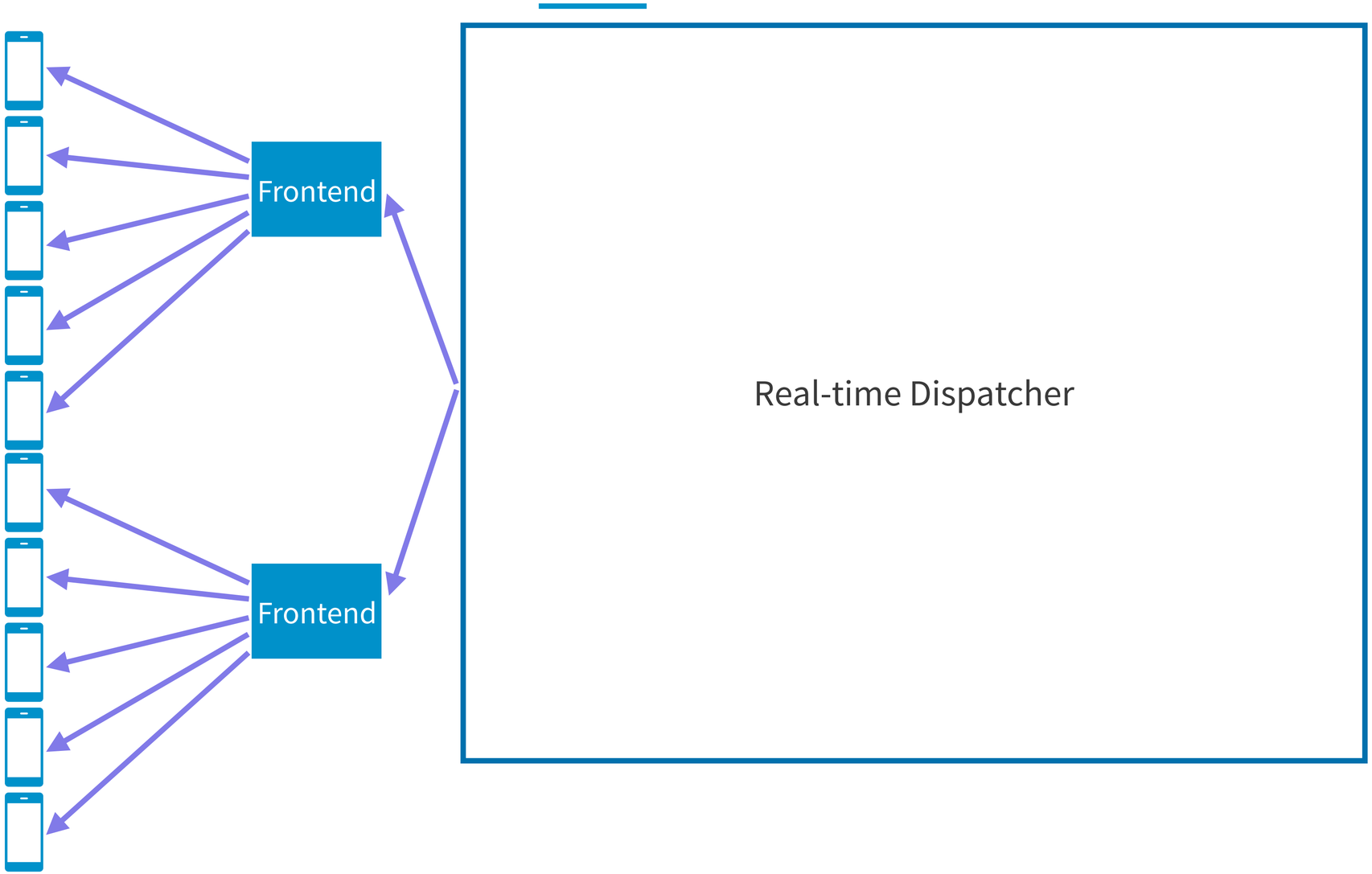
DISPATCHES BETWEEN FRONTEND NODES



Real-time Dispatcher

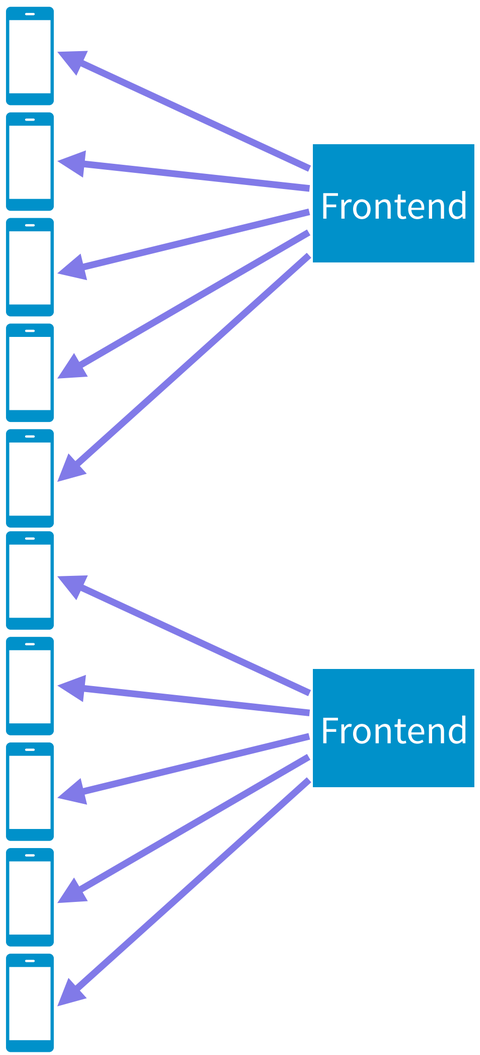
Real-time Dispatcher

DISPATCHES BETWEEN FRONTEND NODES



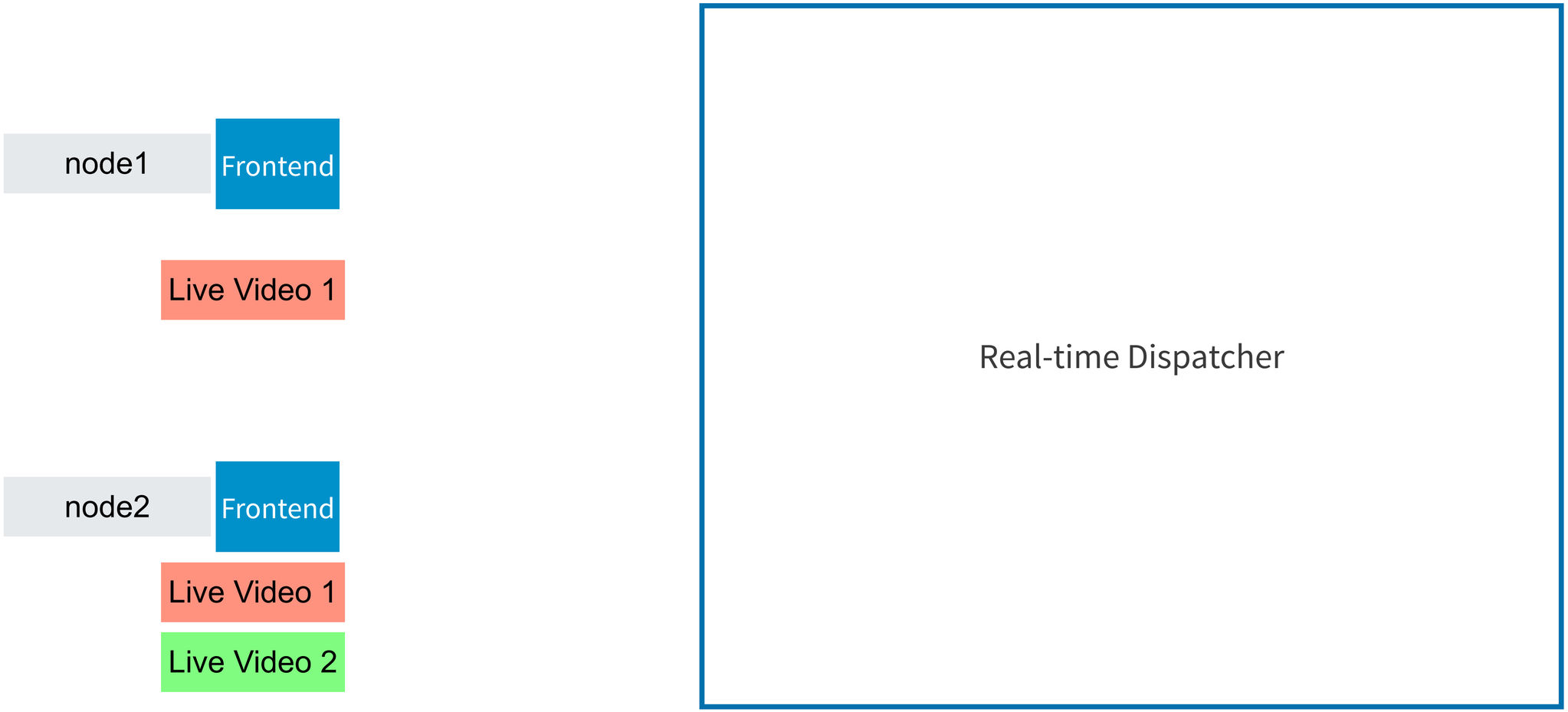
Real-time Dispatcher

DISPATCHES BETWEEN FRONTEND NODES



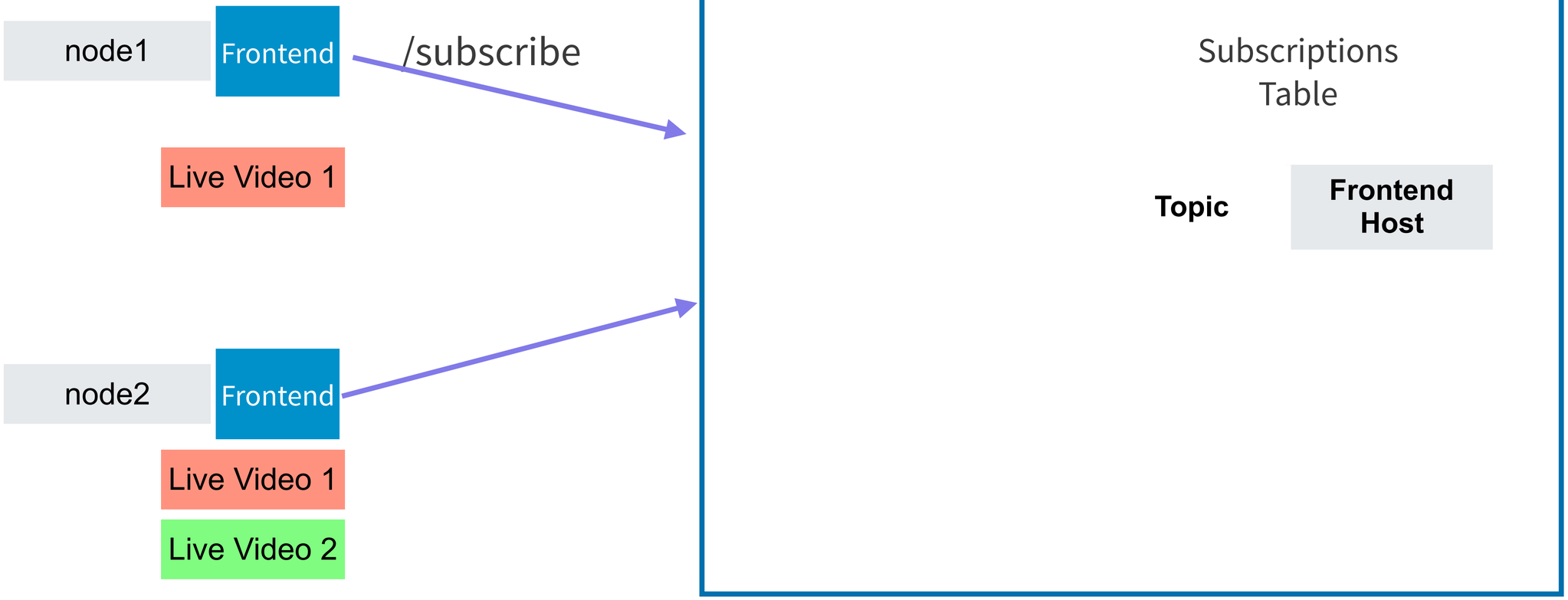
Real-time Dispatcher

Frontend Node Subscriptions



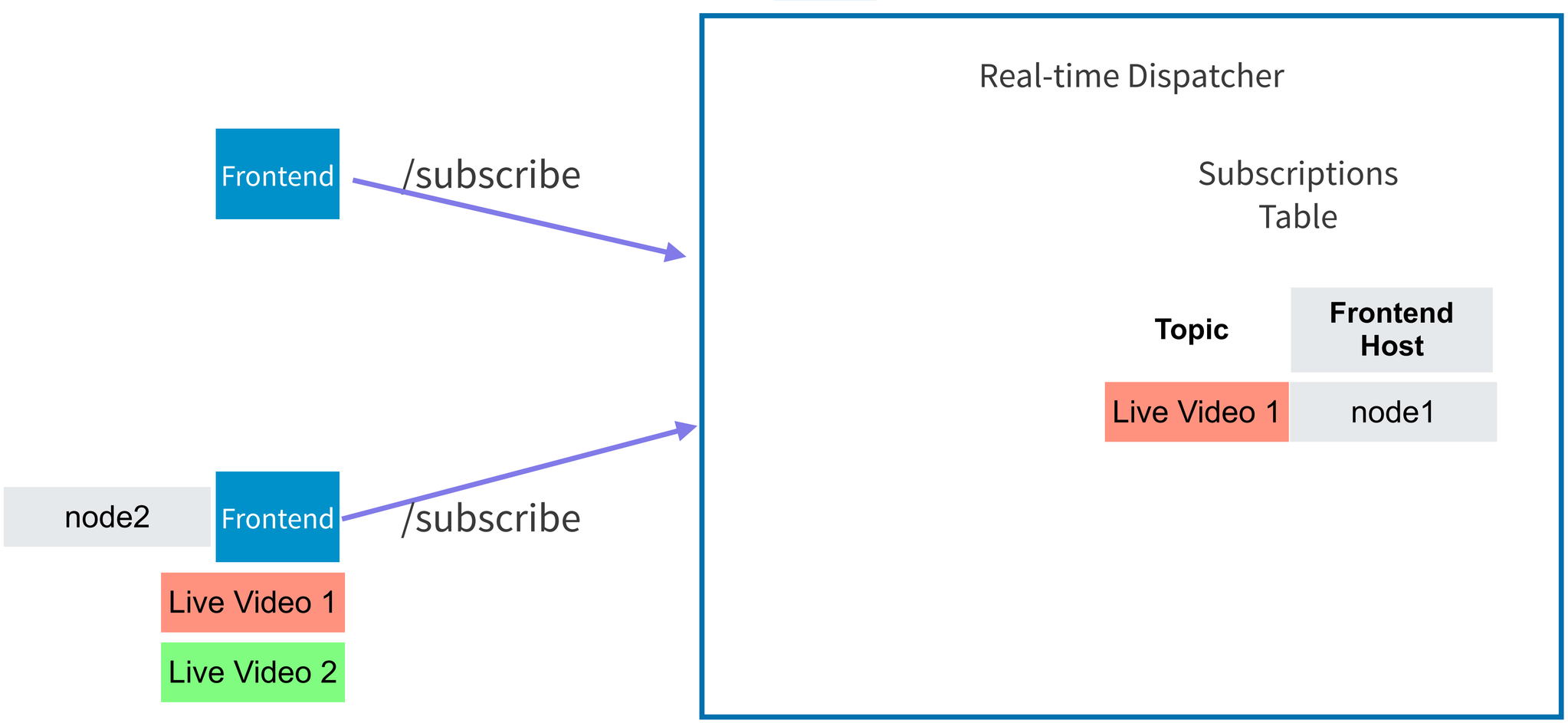
Frontend Node Subscriptions

HTTP SUBSCRIPTION REQUEST



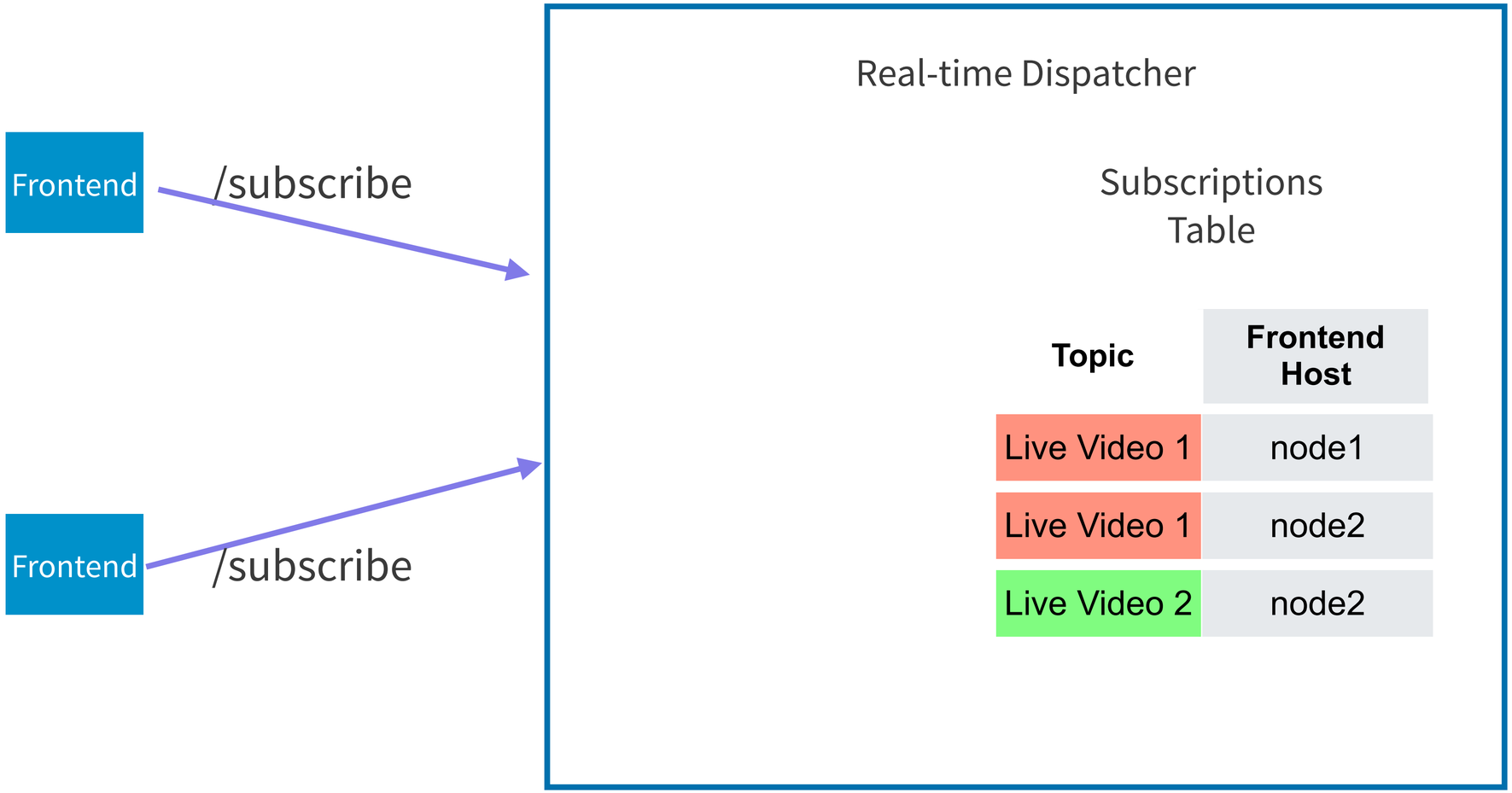
Frontend Node Subscriptions

SUBSCRIPTIONS TABLE ENTRY



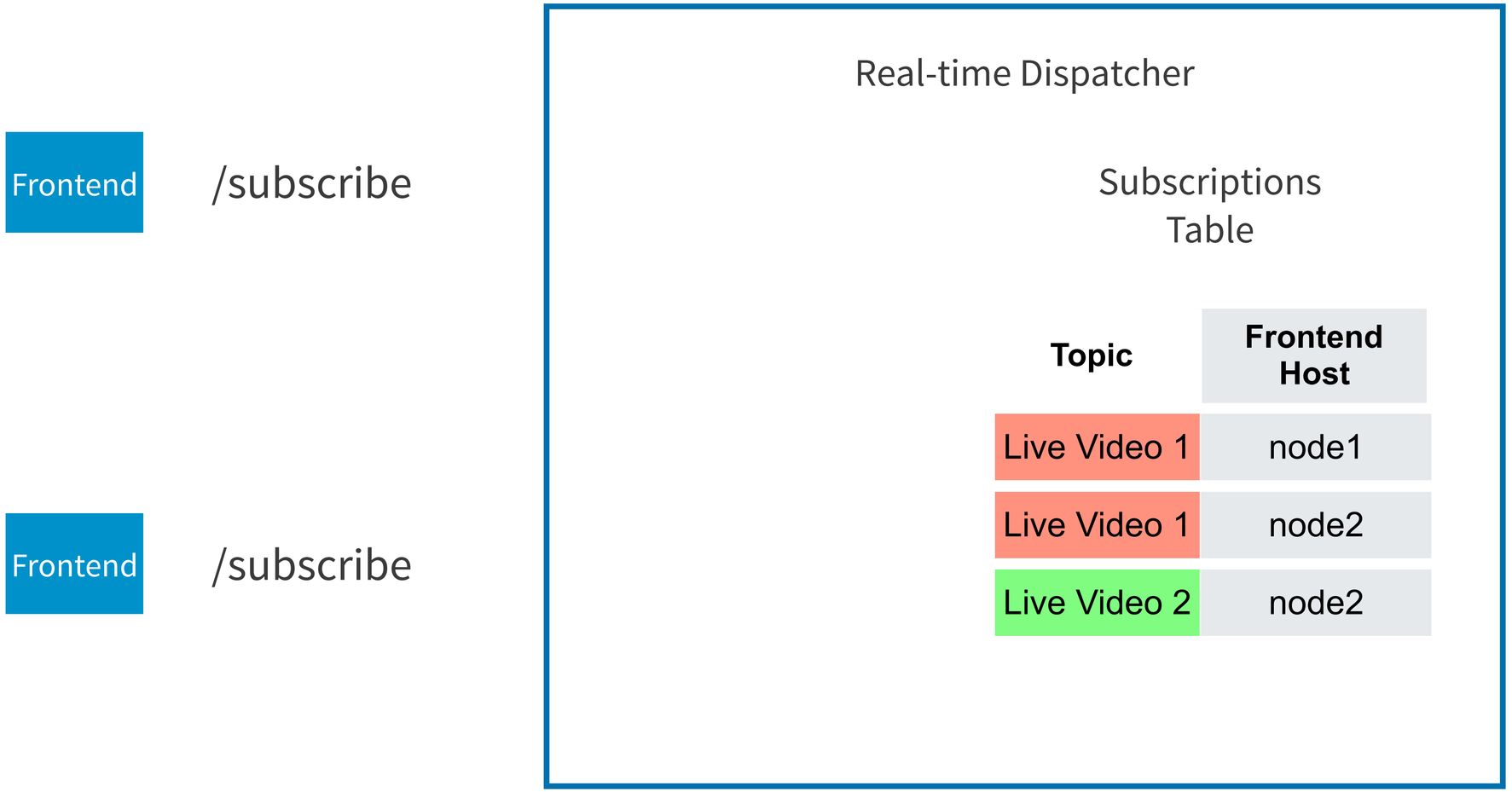
Frontend Node Subscriptions

SUBSCRIPTIONS TABLE ENTRY

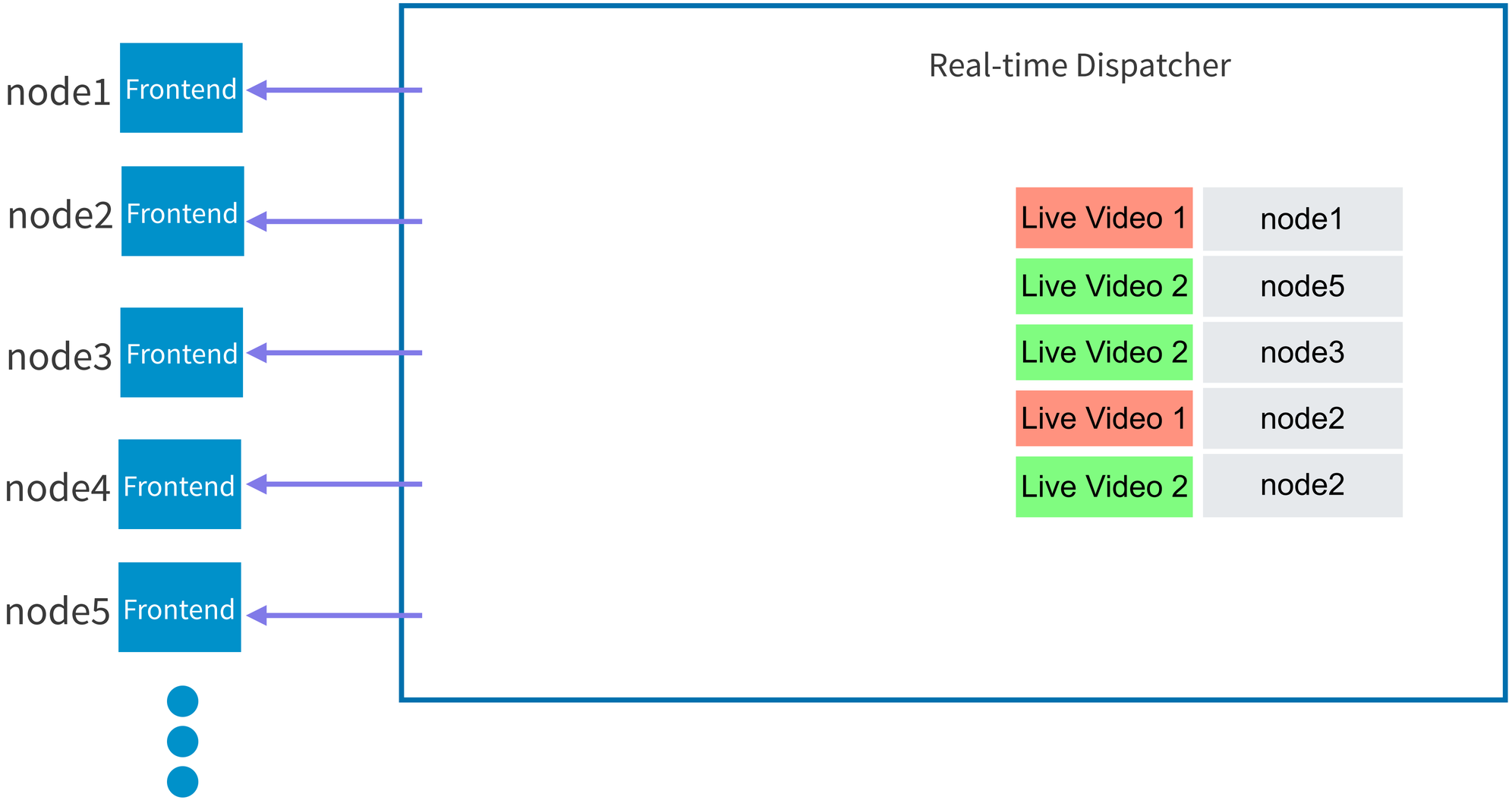


Frontend Node Subscriptions

SUBSCRIPTIONS TABLE ENTRY

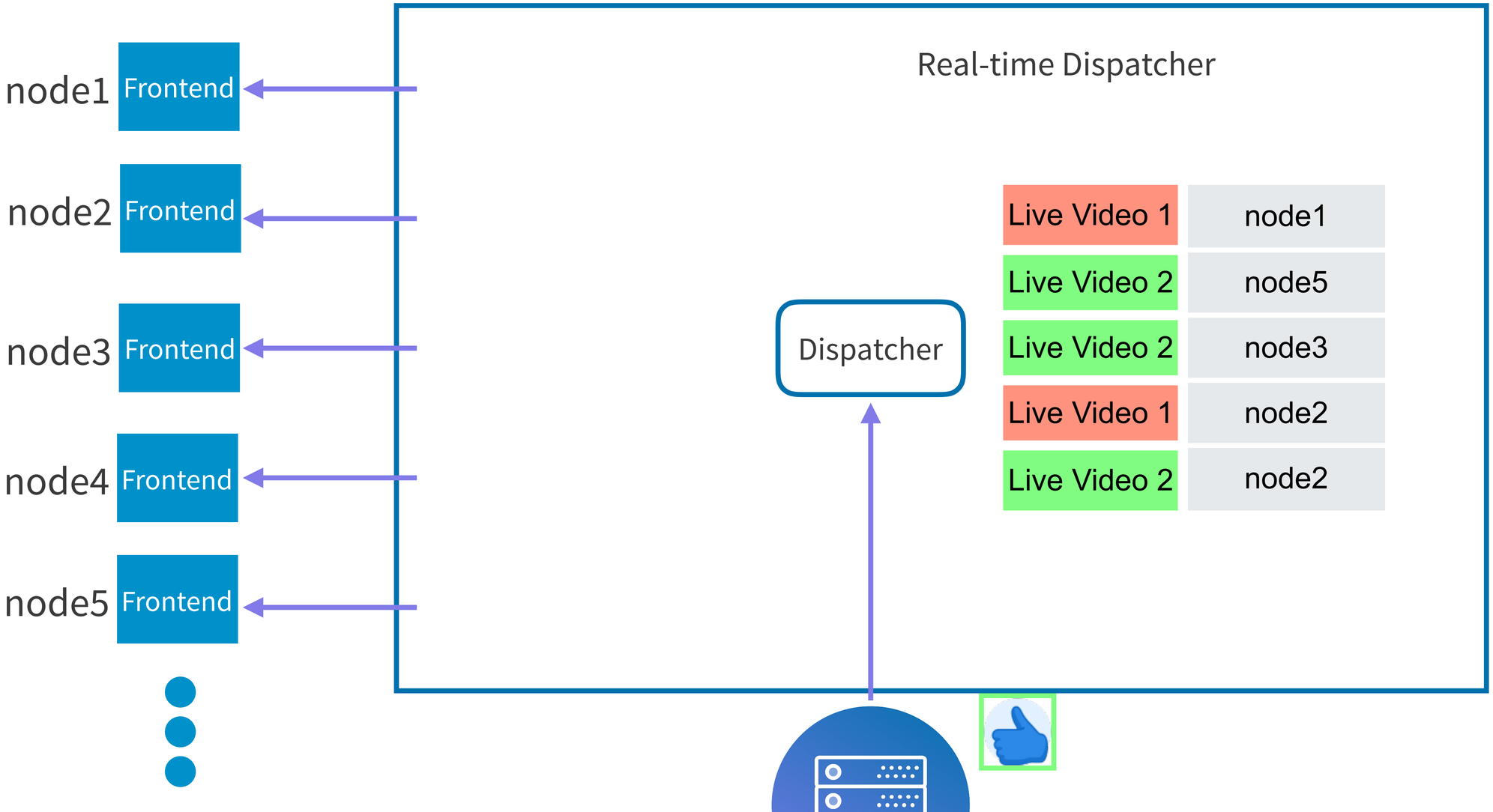


Publish to Frontend Nodes



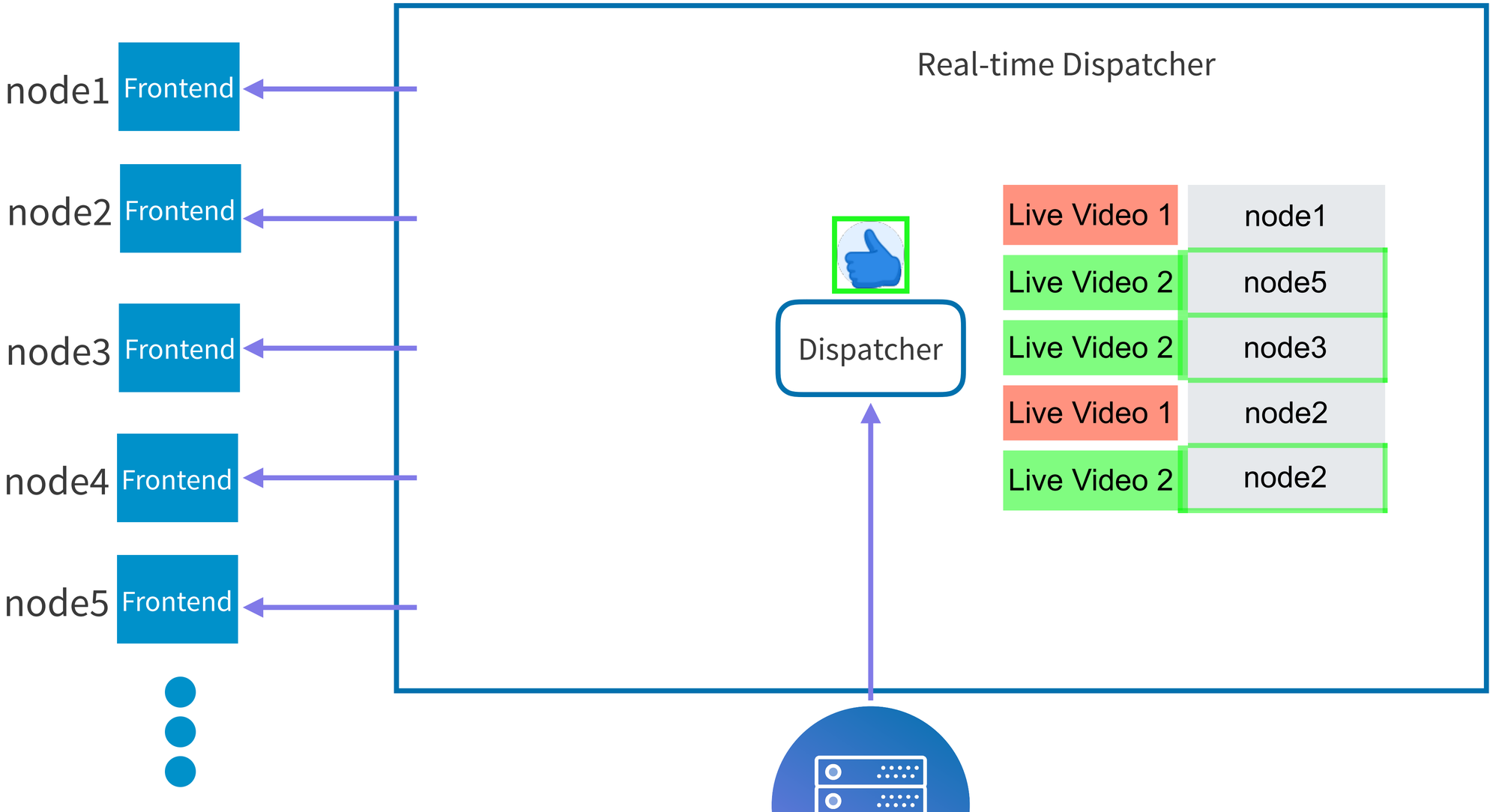
Publish to Frontend Nodes

PUBLISHER PUBLISH TO DISPATCHER



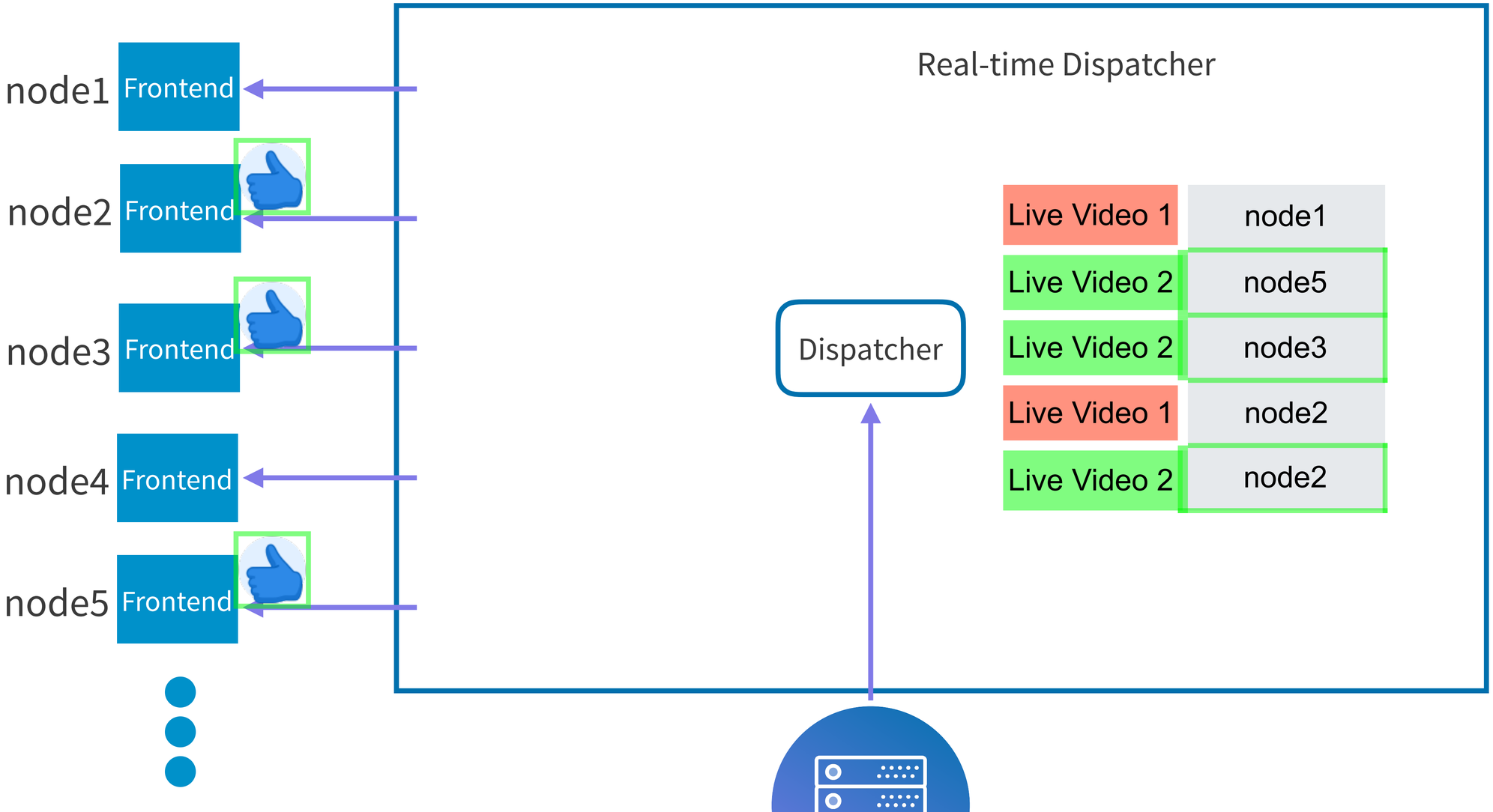
Publish to Frontend Nodes

SUBSCRIBER LOOKUP

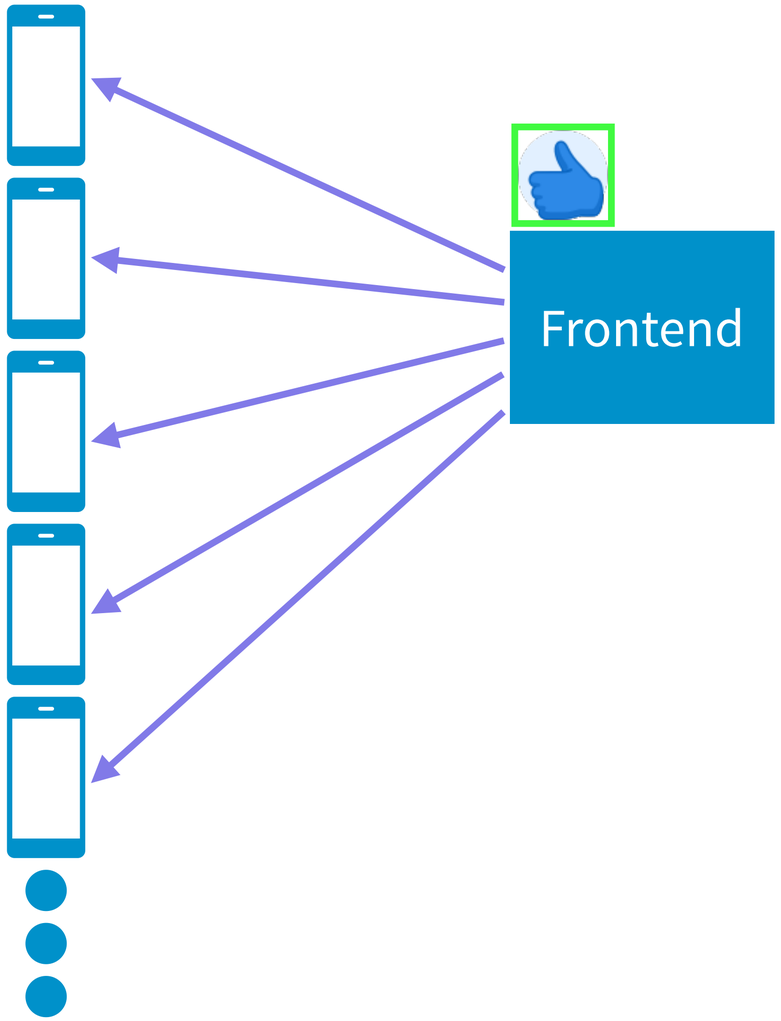


Publish to Frontend Nodes

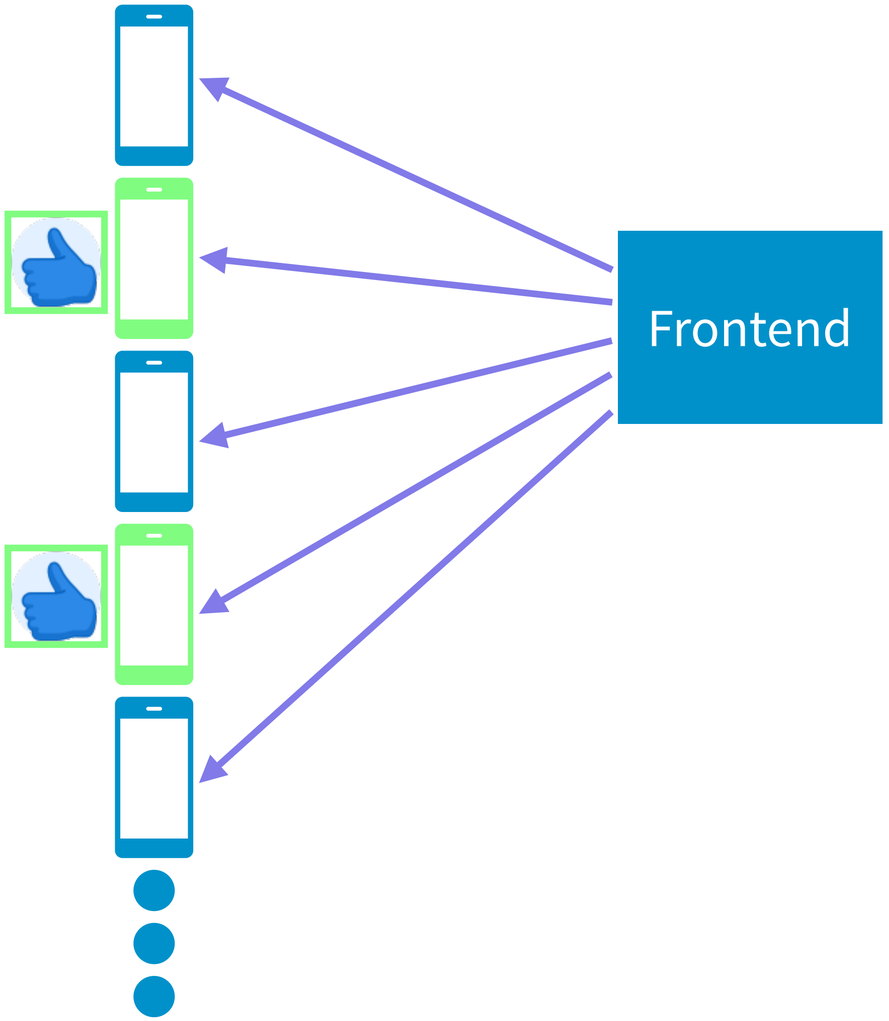
PUBLISH TO SUBSCRIBED FRONTEND NODES



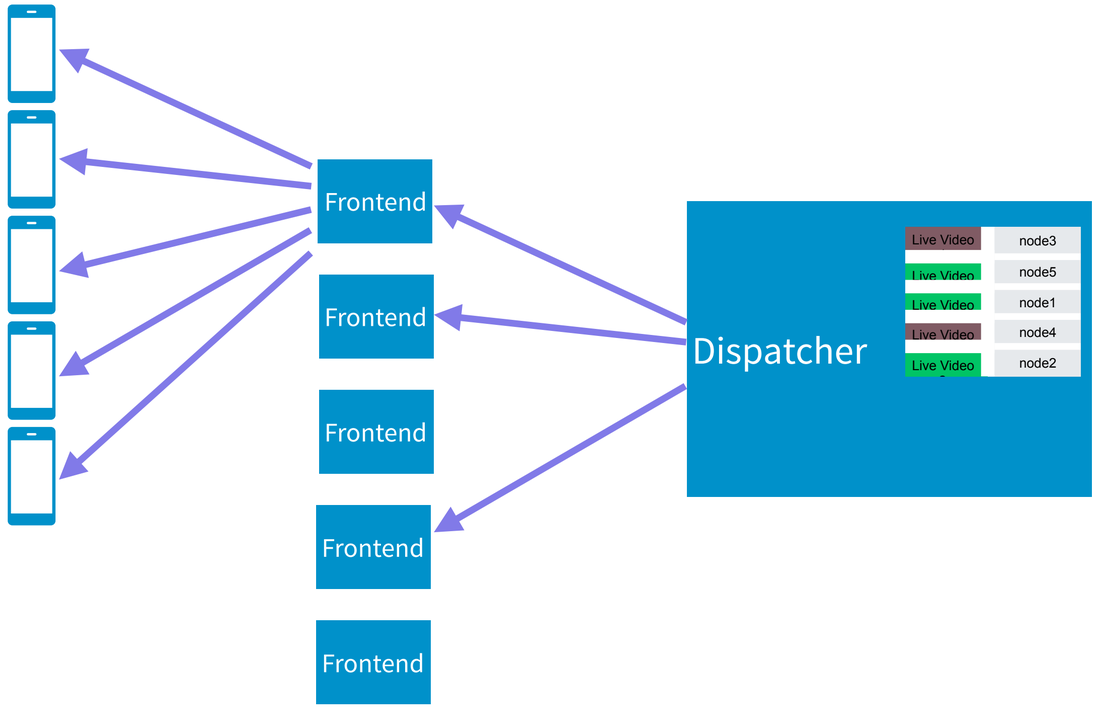
Frontend Node to Client Publish



Frontend Node to Client Publish

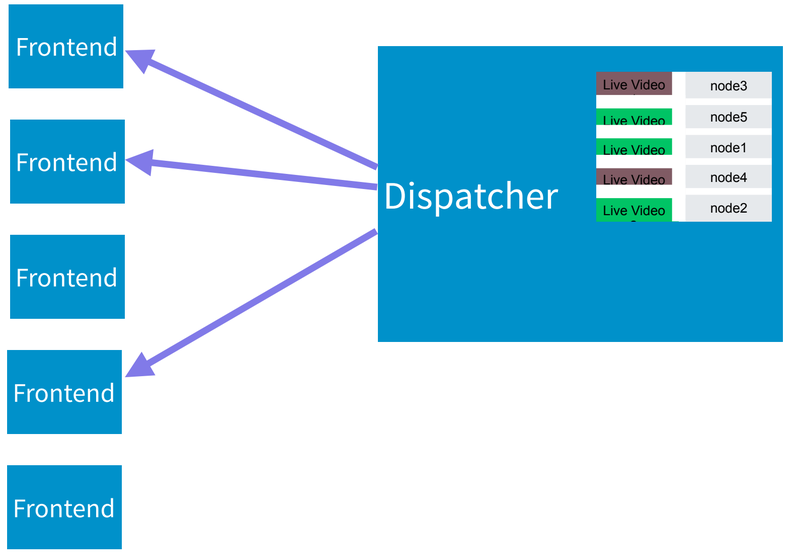


What’s the next challenge?



Challenge 5: 100 Likes/second

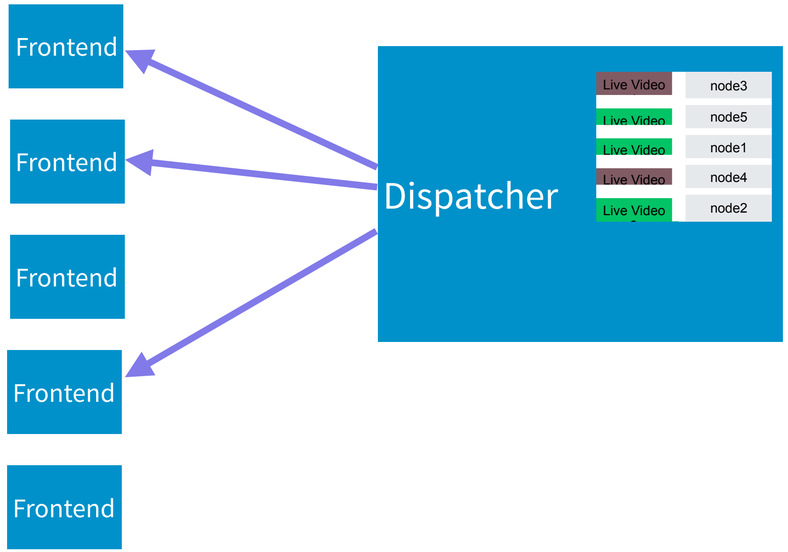
100 Likes/second



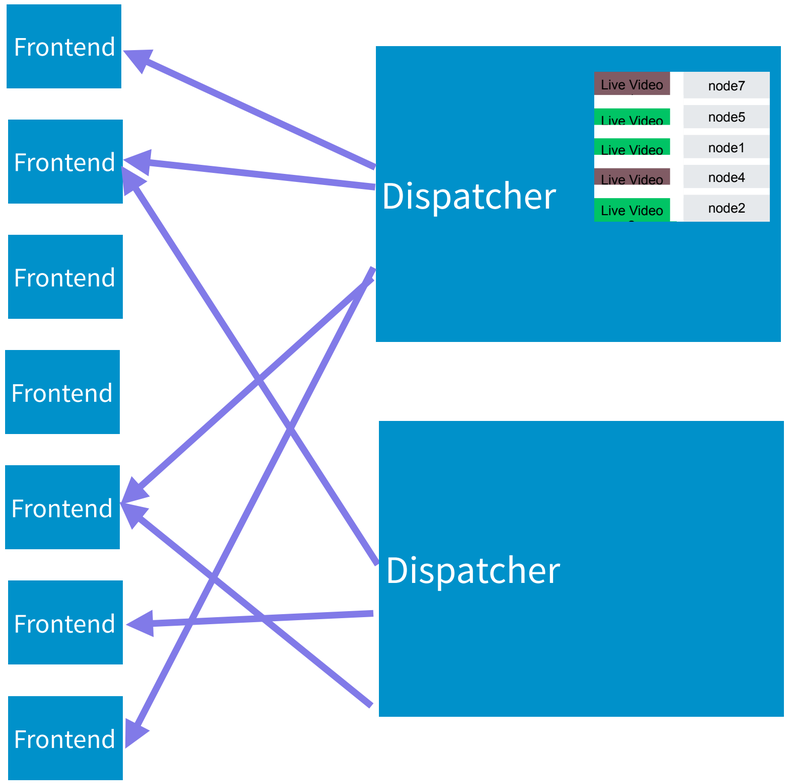
SCALING BACKEND SYSTEMS

When in doubt, add a machine!

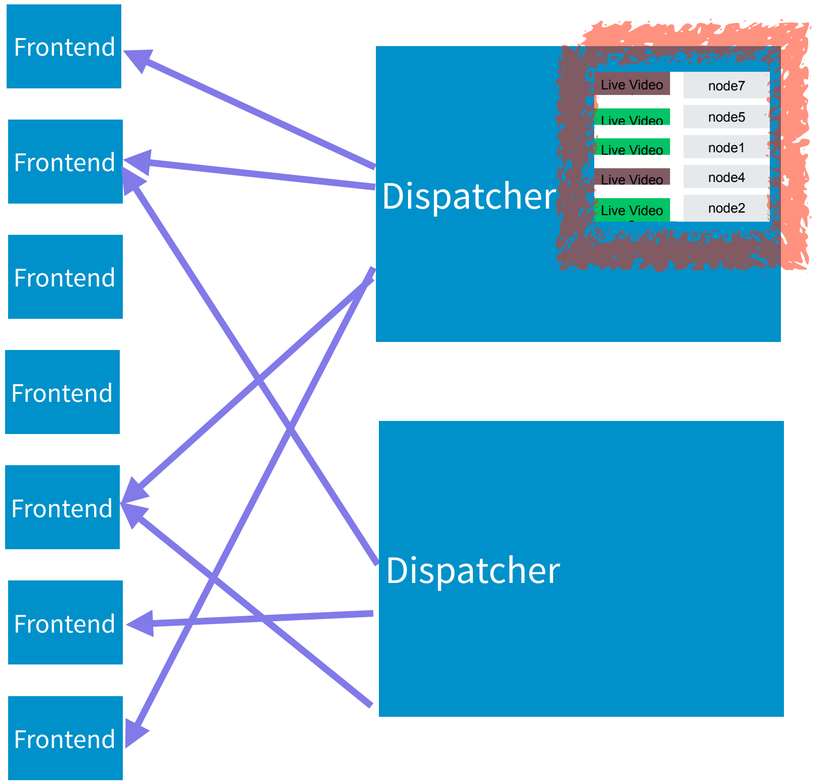
100 Likes/second



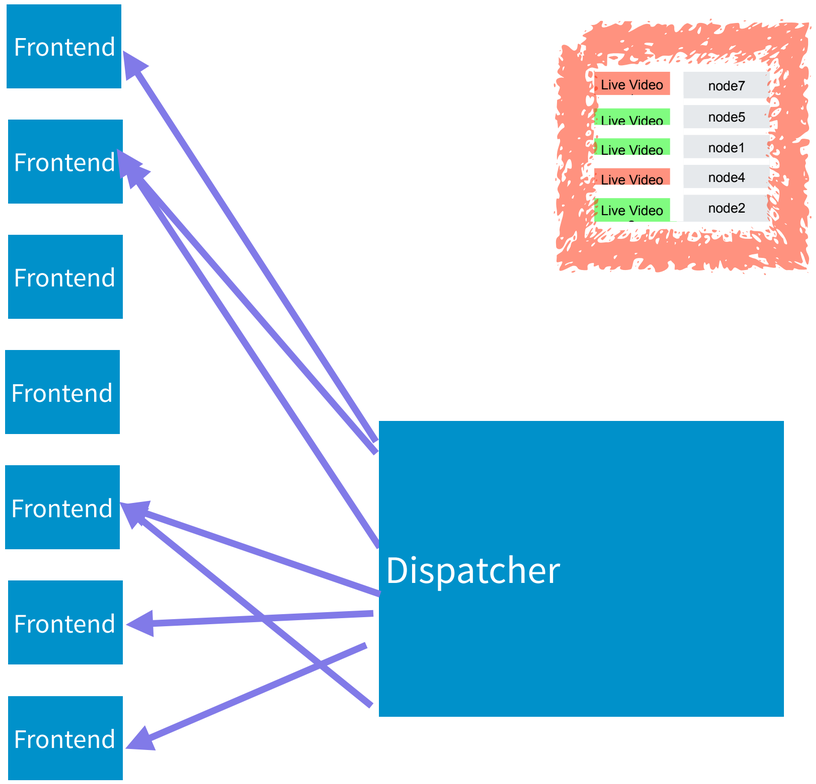
100 Likes/second



100 Likes/second

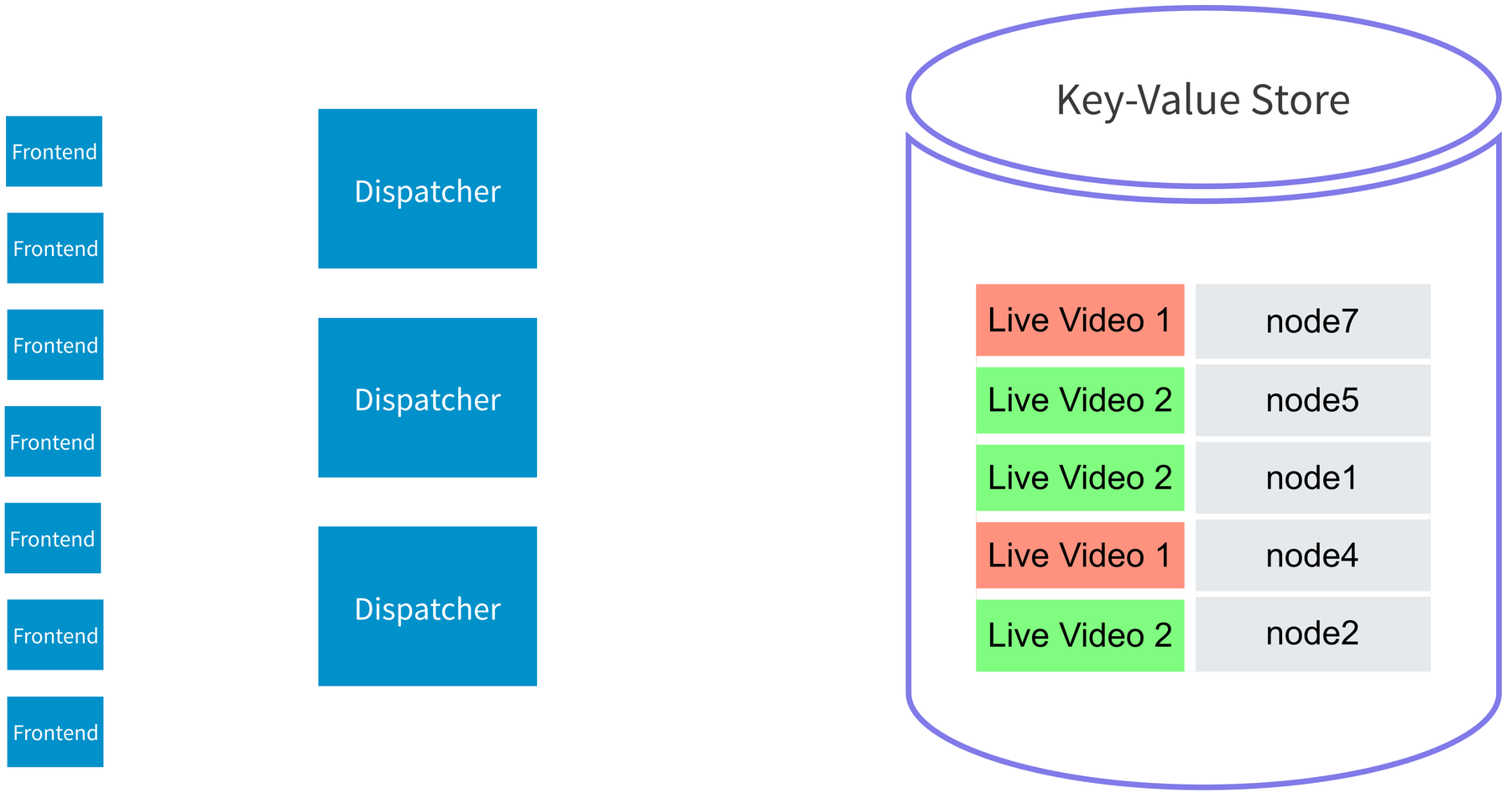


100 Likes/second

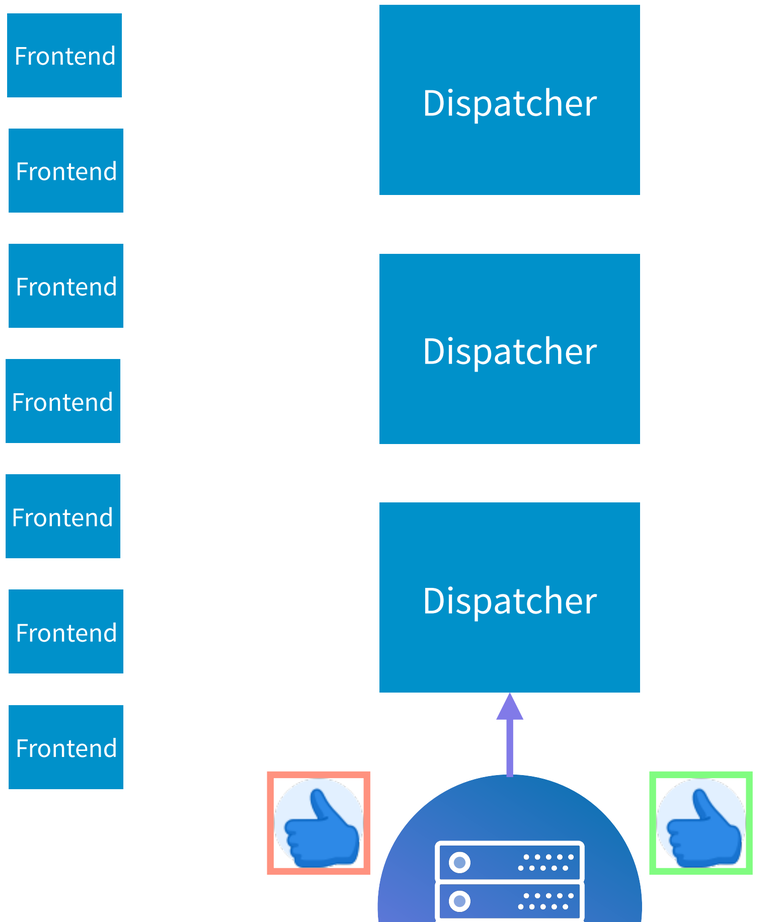


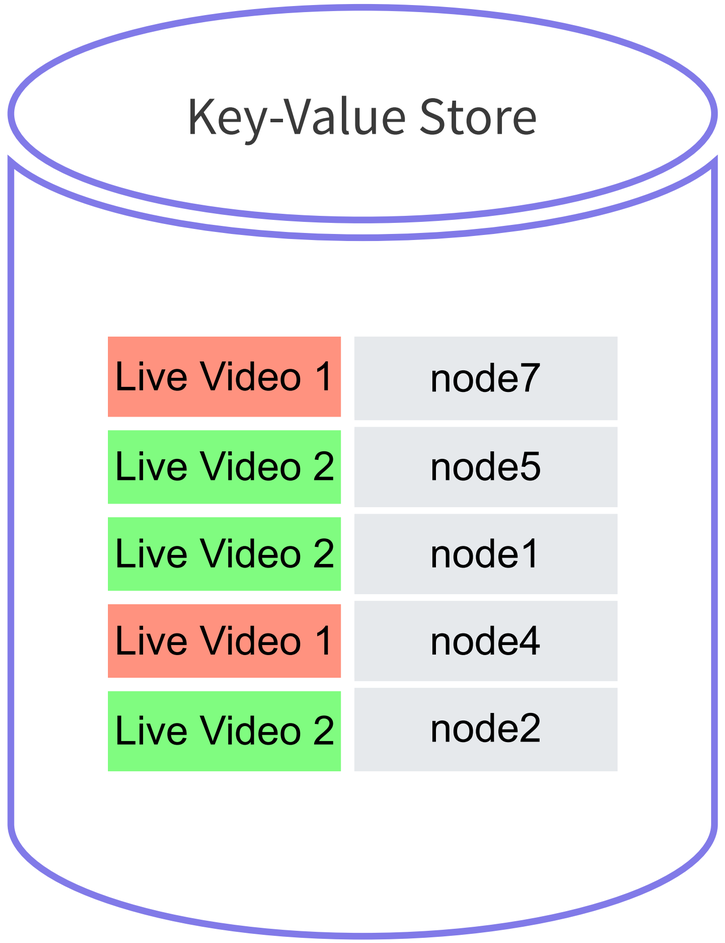
Key-Value Store for Subscriptions

COUCHBASE/REDIS ETC.

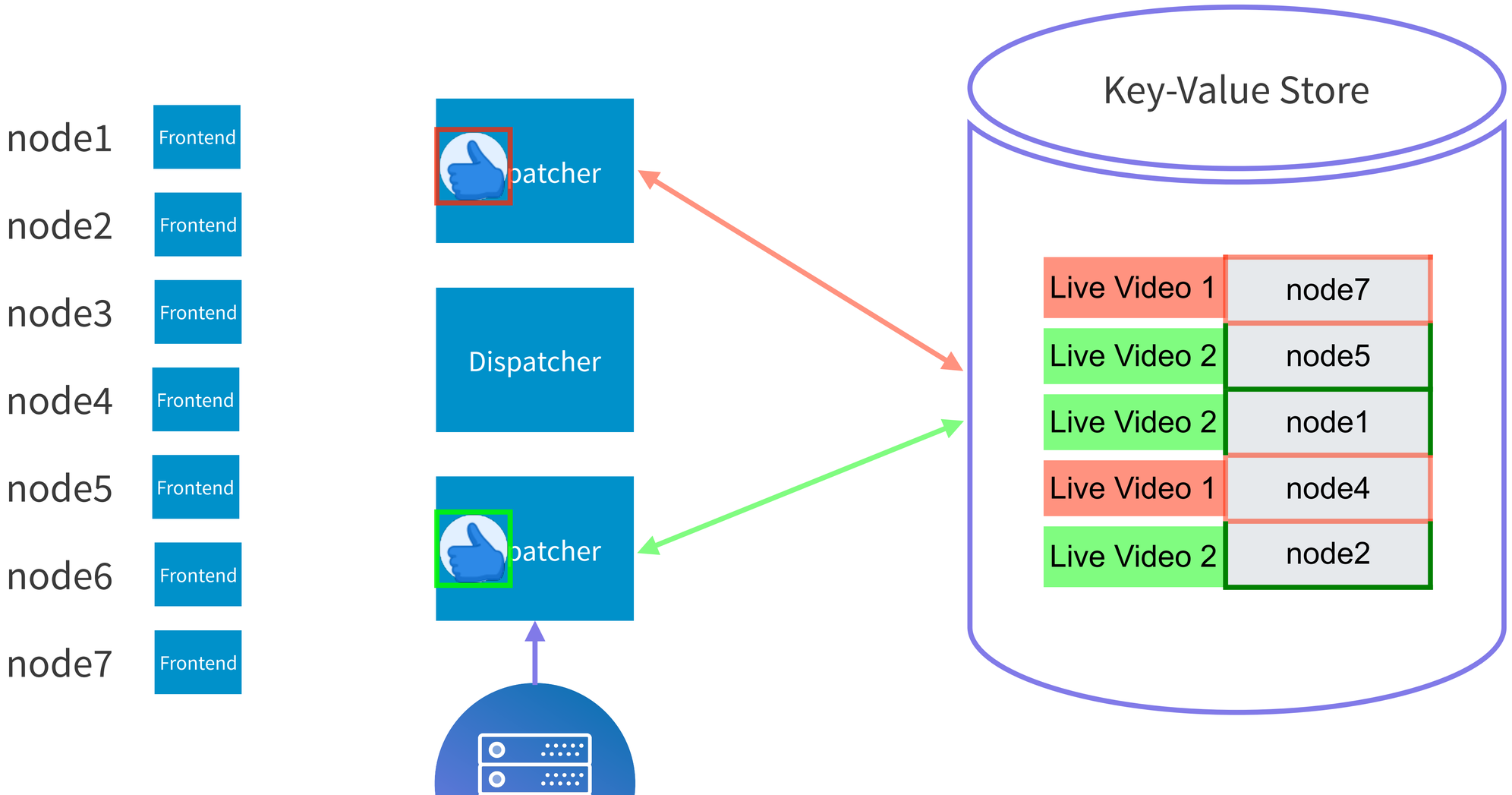


Publish to Dispatcher Nodes

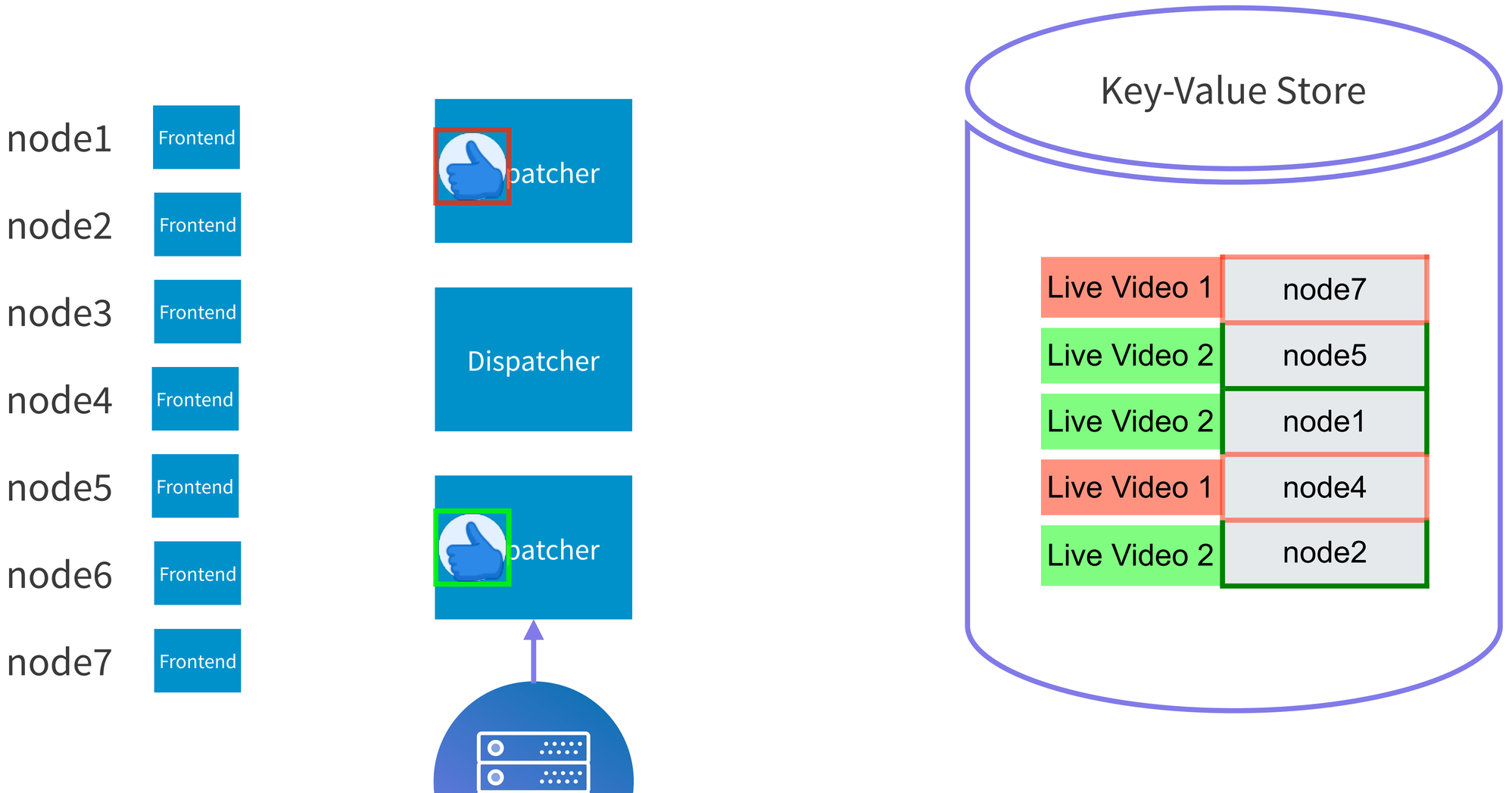




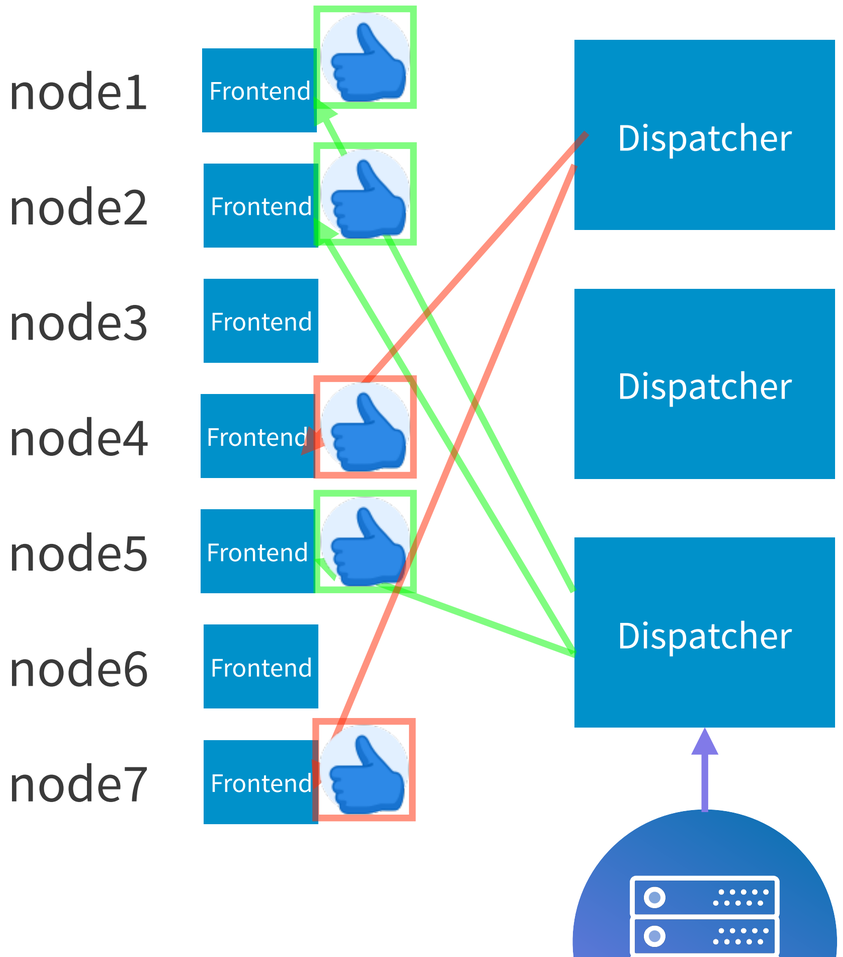
Subscriber lookup from KV Store

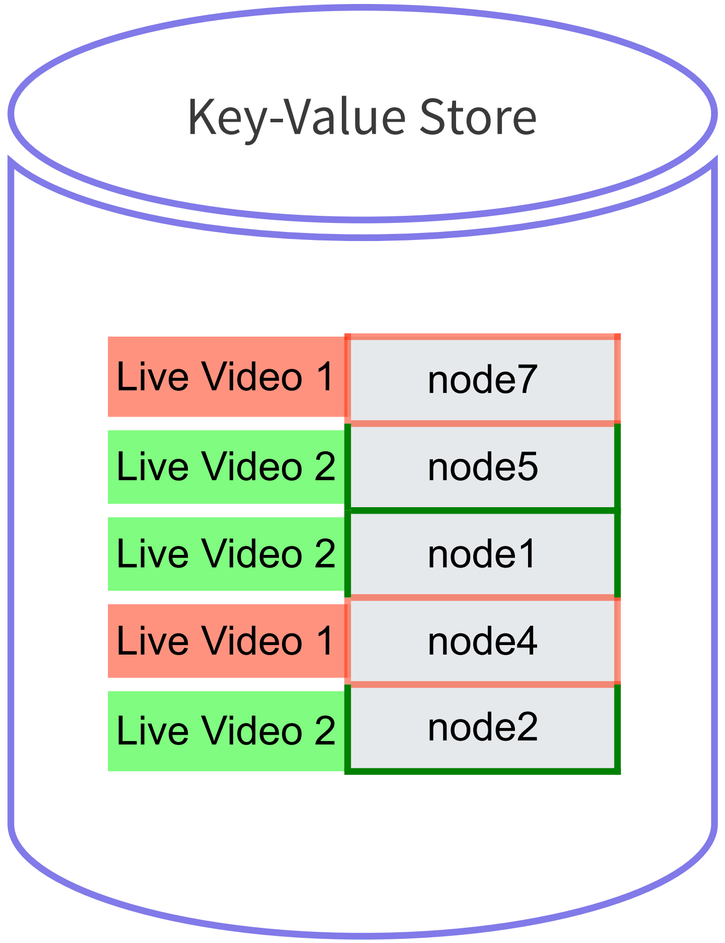


Publish to Frontend Nodes

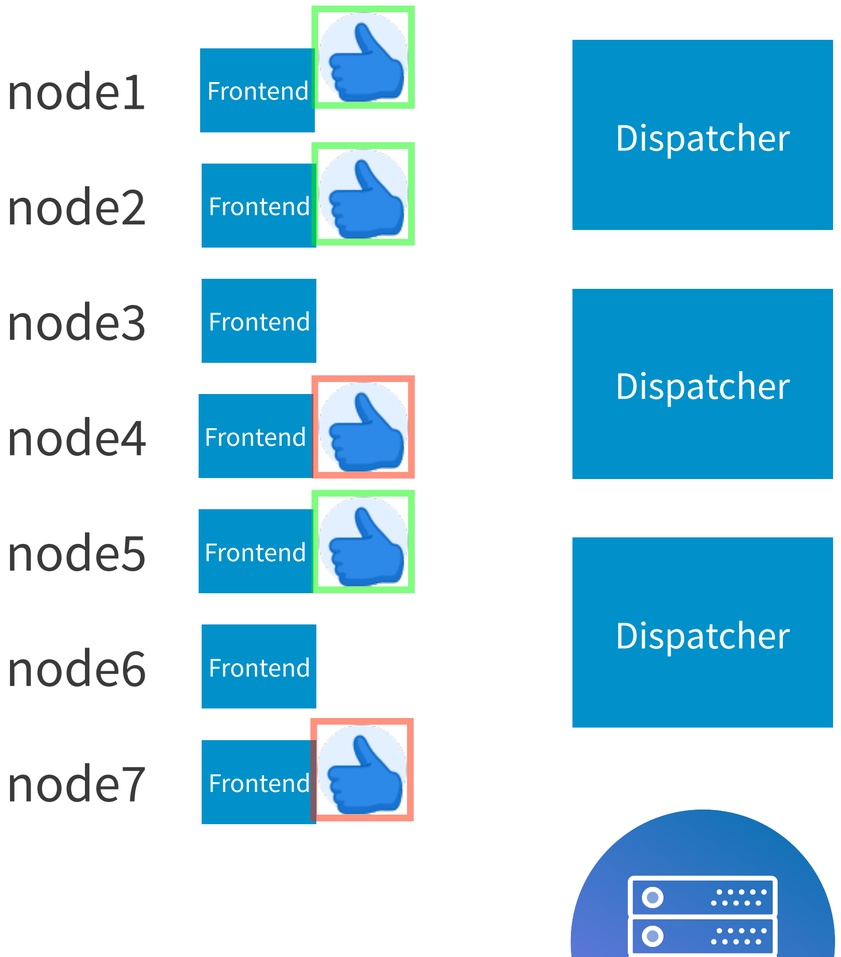


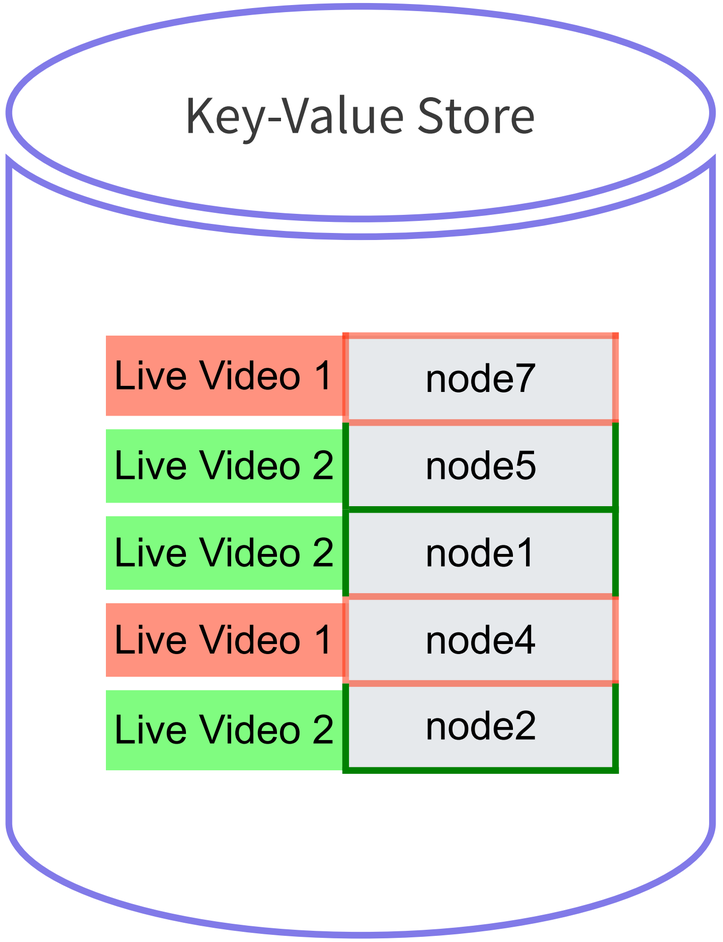
Publish to Frontend Nodes





Publish to Frontend Nodes





Challenge 6: 100 Likes/s, 10K Viewers

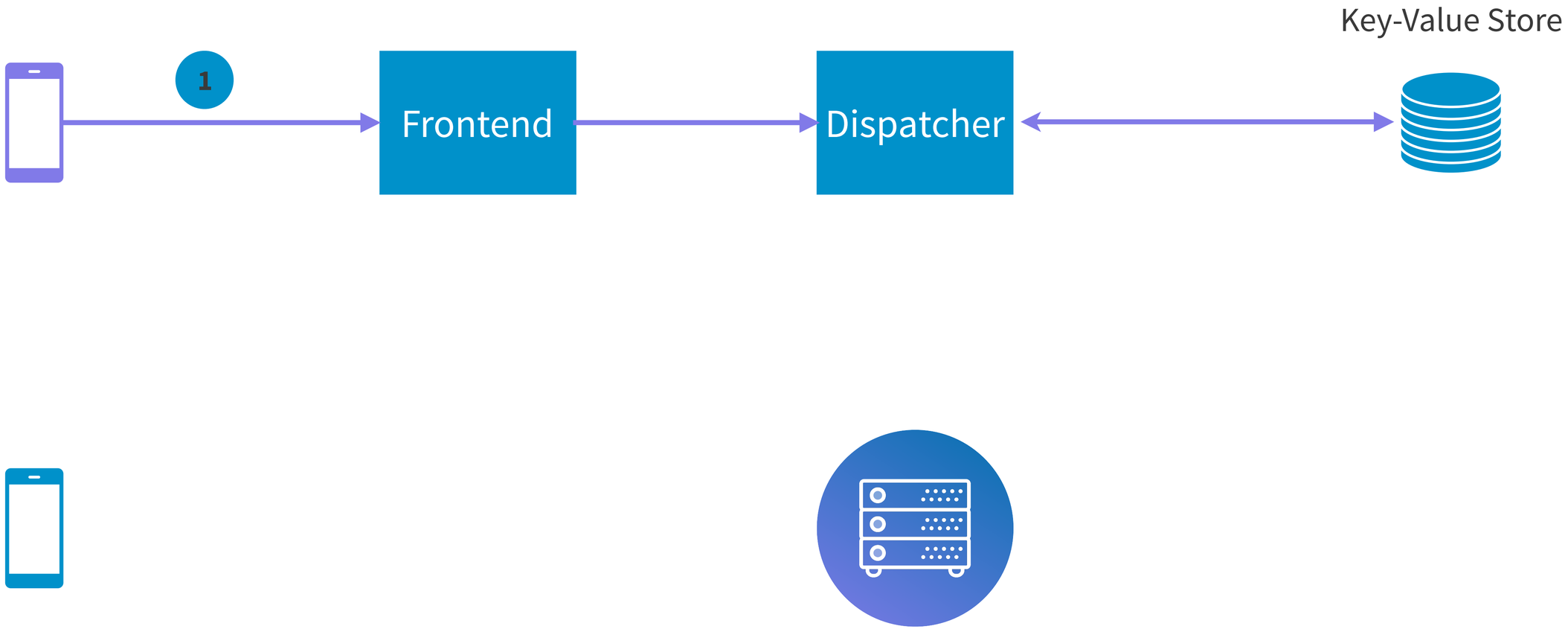
Distribution of **1M** Likes/s

LINKEDIN LIVE

Mobile

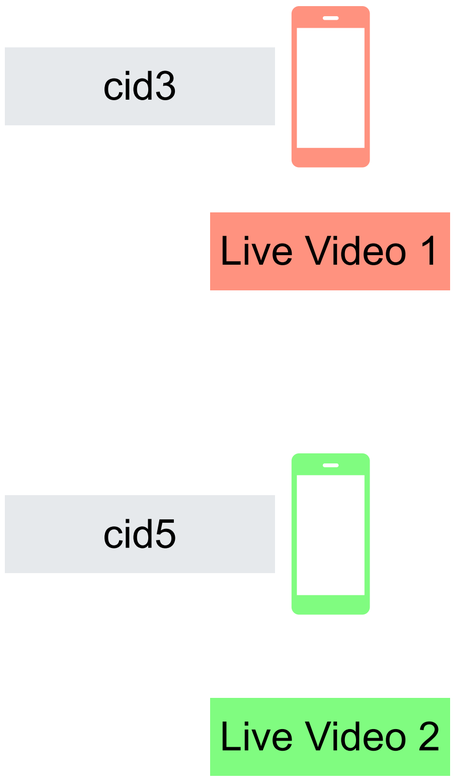
Likes

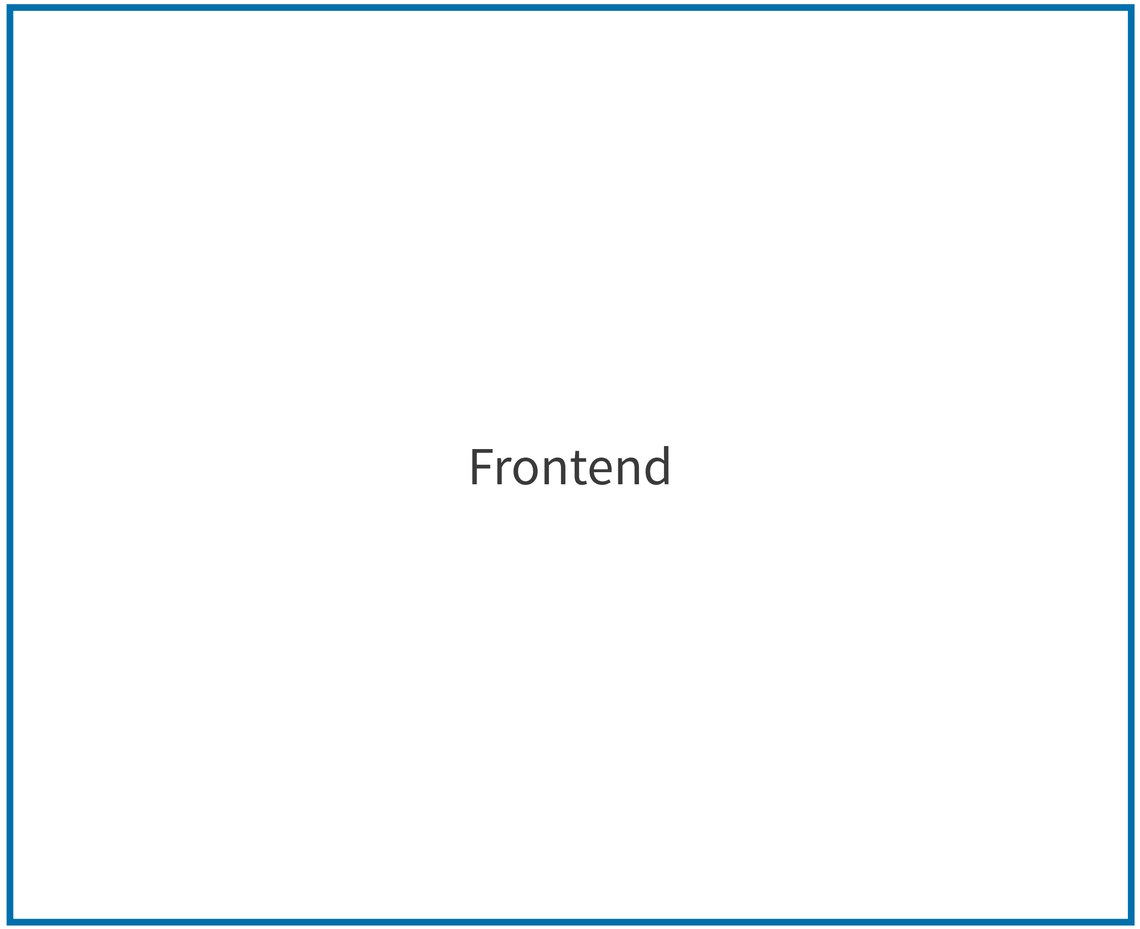
The Subscription Flow



Viewers subscribing to Live Video Likes

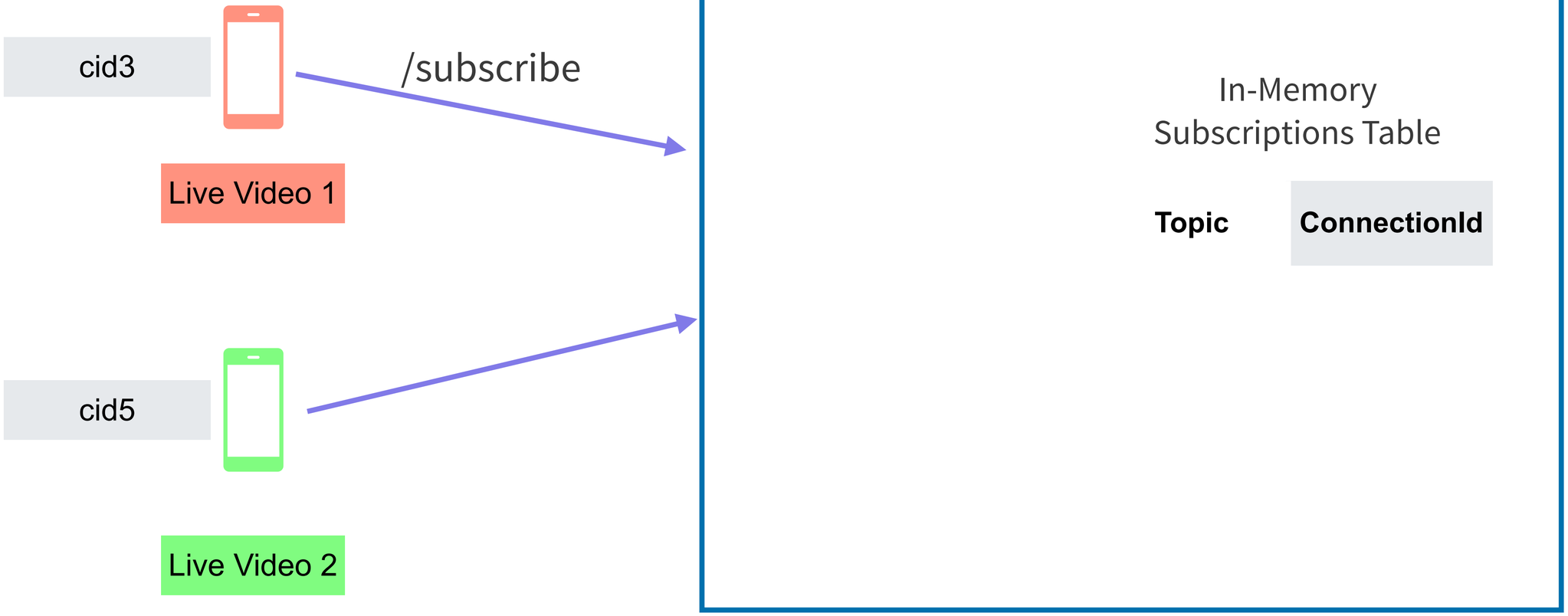
START WATCHING LIVE VIDEO





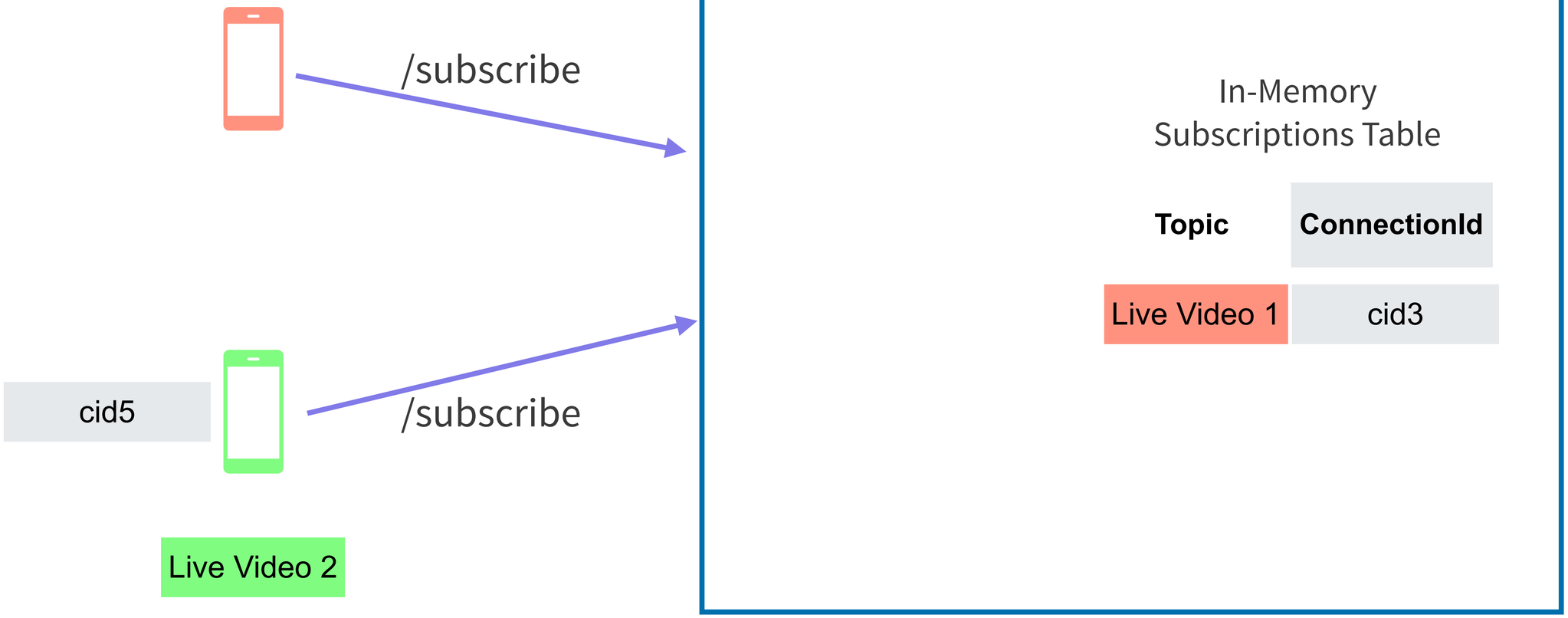
Viewers subscribing to Live Video Likes

HTTP SUBSCRIPTION REQUEST



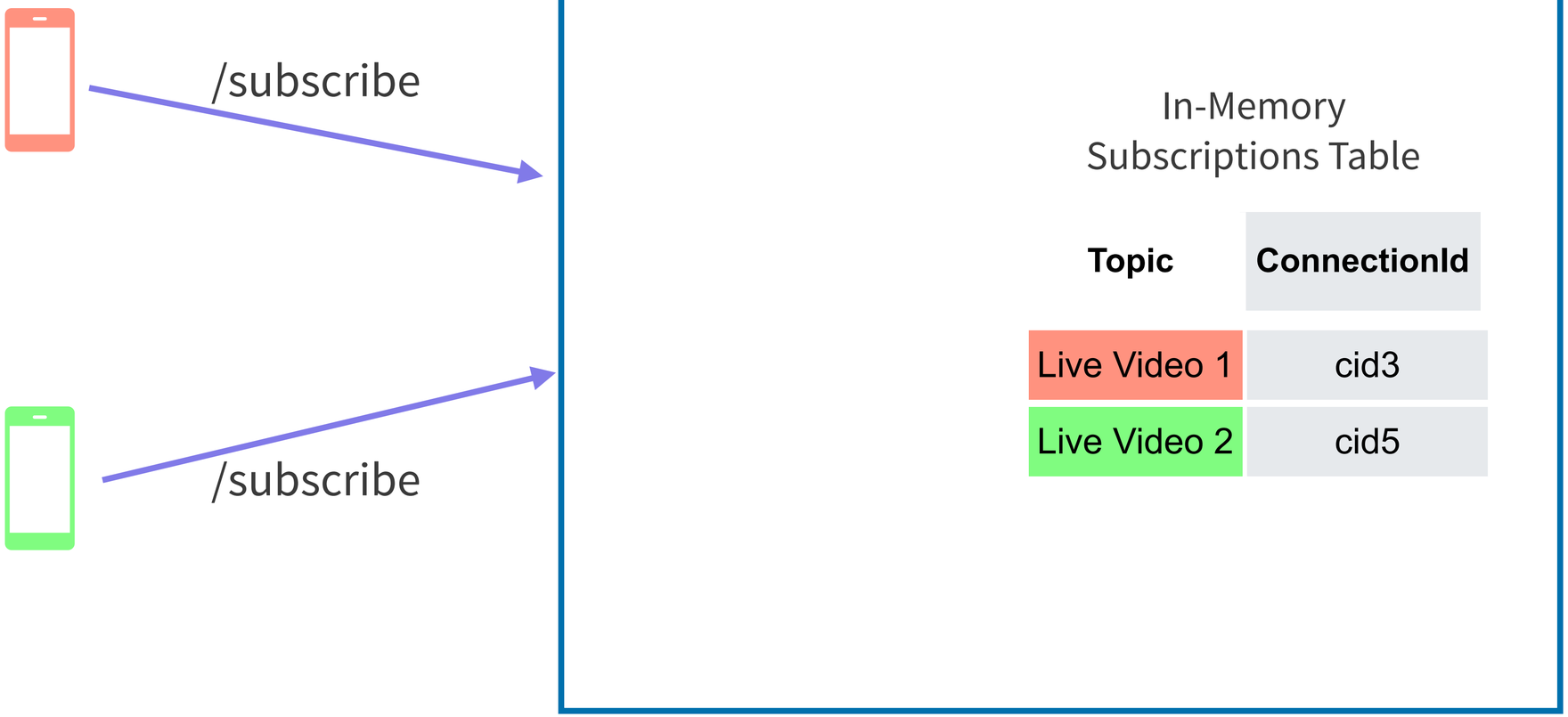
Viewers subscribing to Live Video Likes

FRONTEND IN-MEMORY SUBSCRIPTION ENTRY



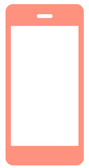
Viewers subscribing to Live Video Likes

FRONTEND IN-MEMORY SUBSCRIPTION ENTRY

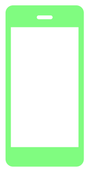


Viewers subscribing to Live Video Likes

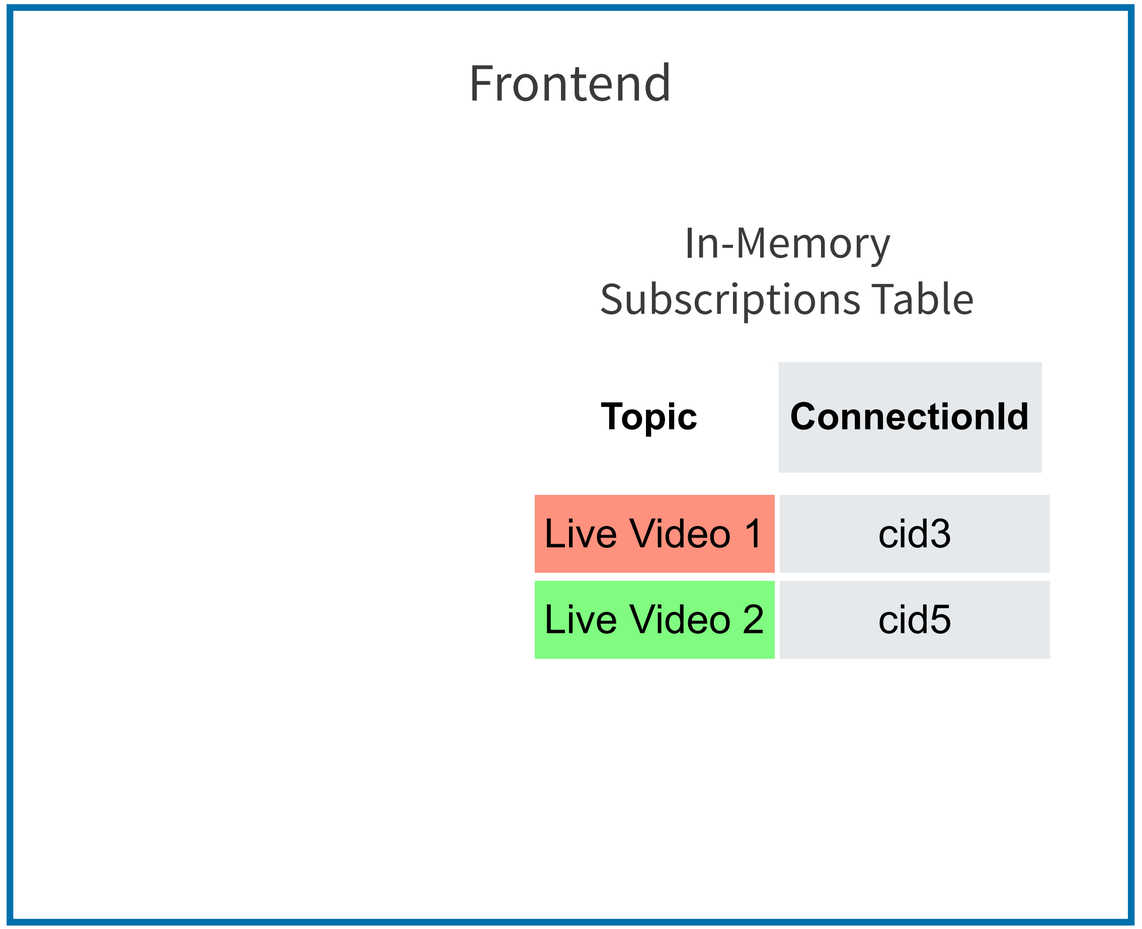
FRONTEND IN-MEMORY SUBSCRIPTION ENTRY



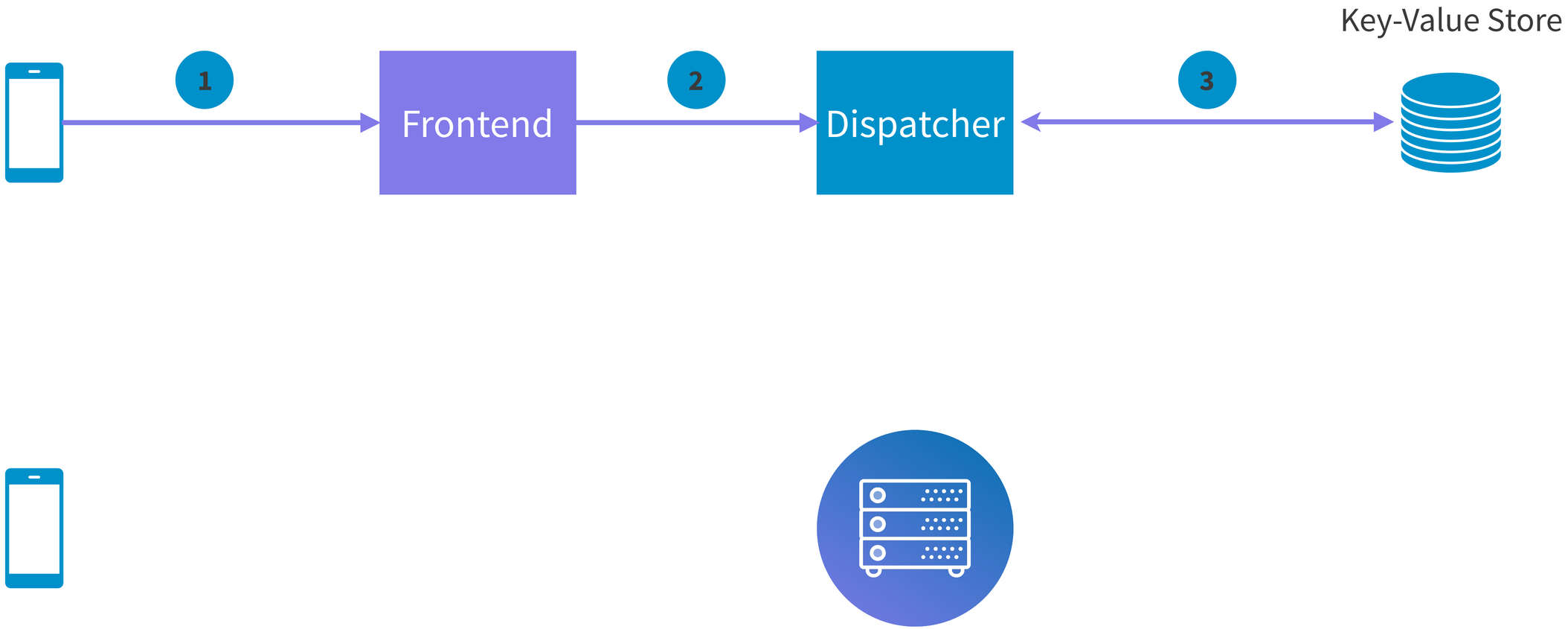
/subscribe



/subscribe

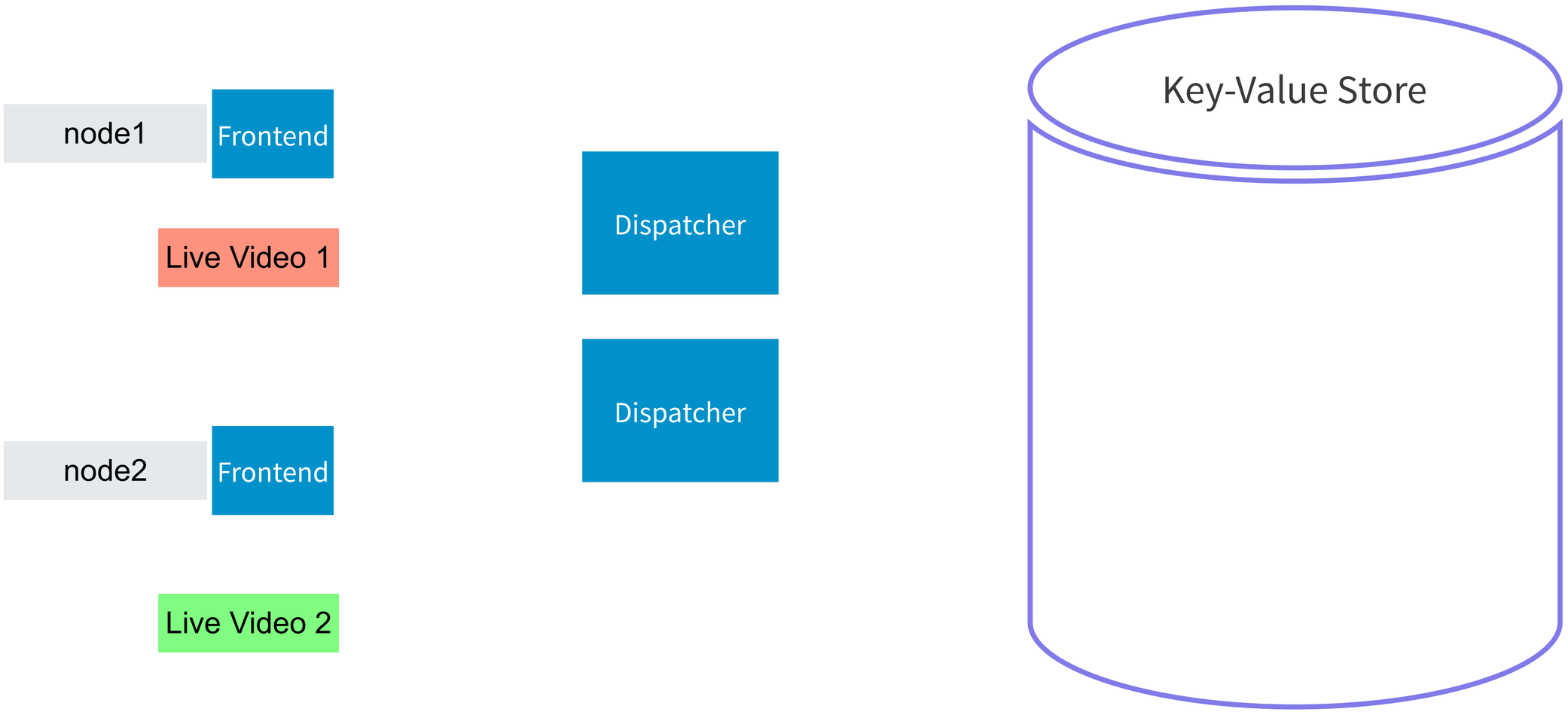


The Subscription Flow



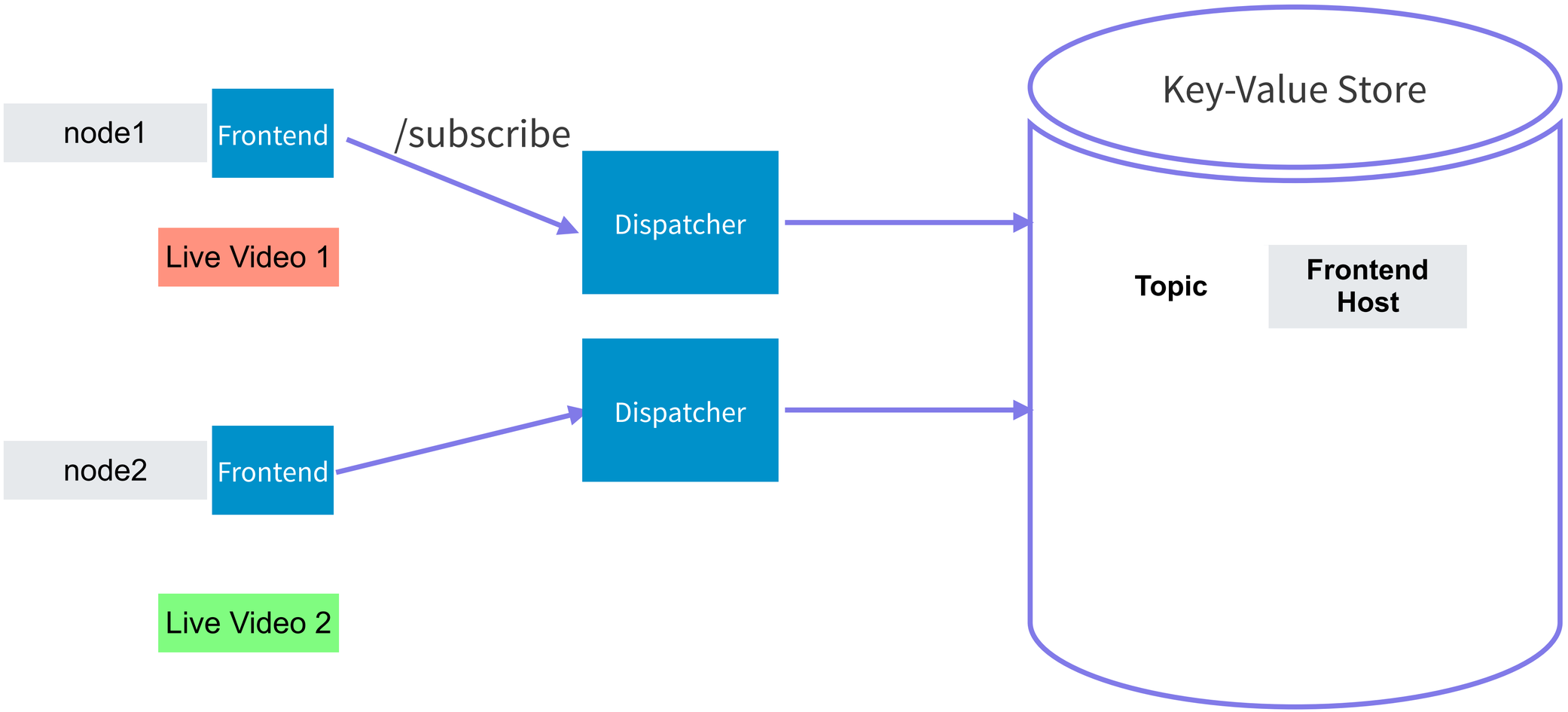
Frontend Node Subscriptions in Dispatcher

FRONTEND RECEIVES SUBSCRIPTION REQUEST



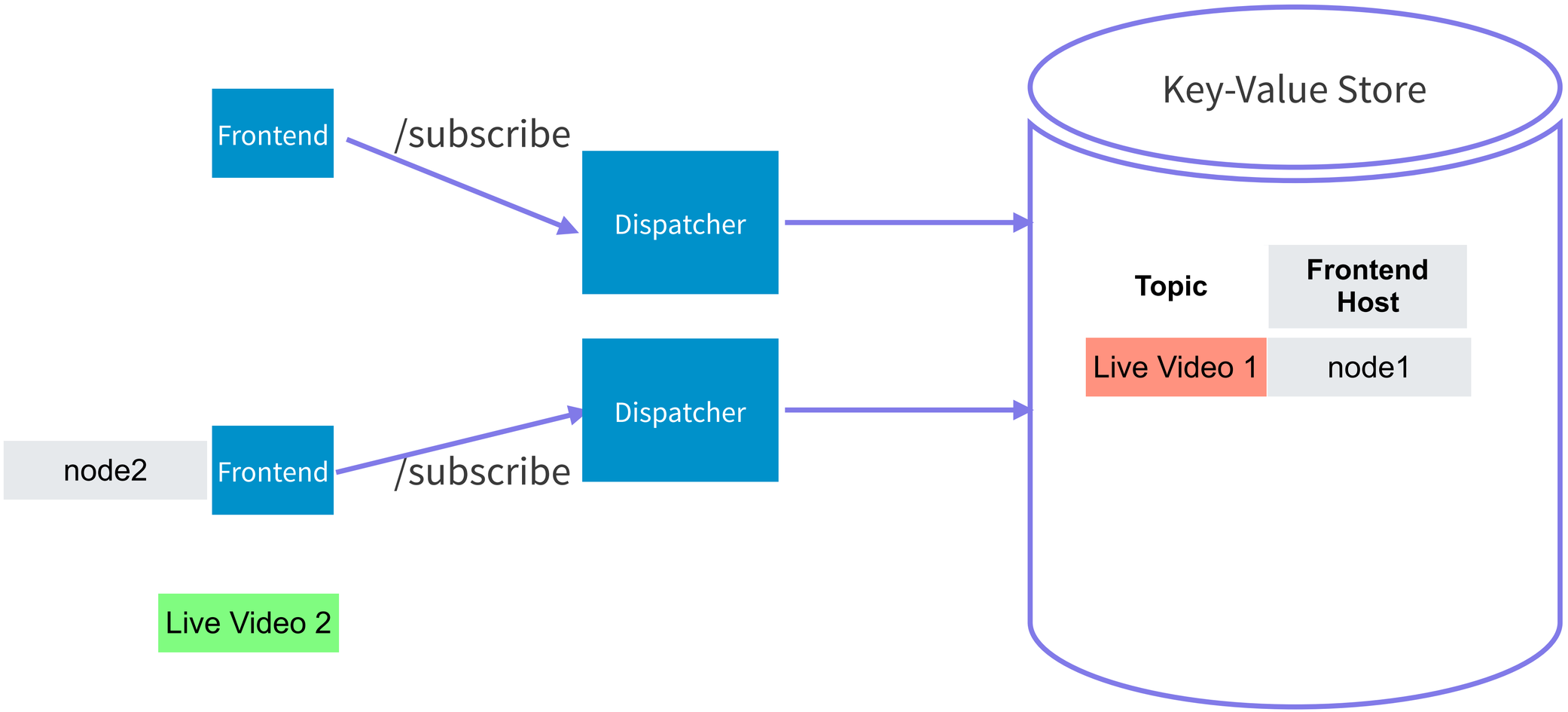
Frontend Node Subscriptions in Dispatcher

FRONTEND HTTP SUBSCRIPTION REQUEST TO DISPATCHER



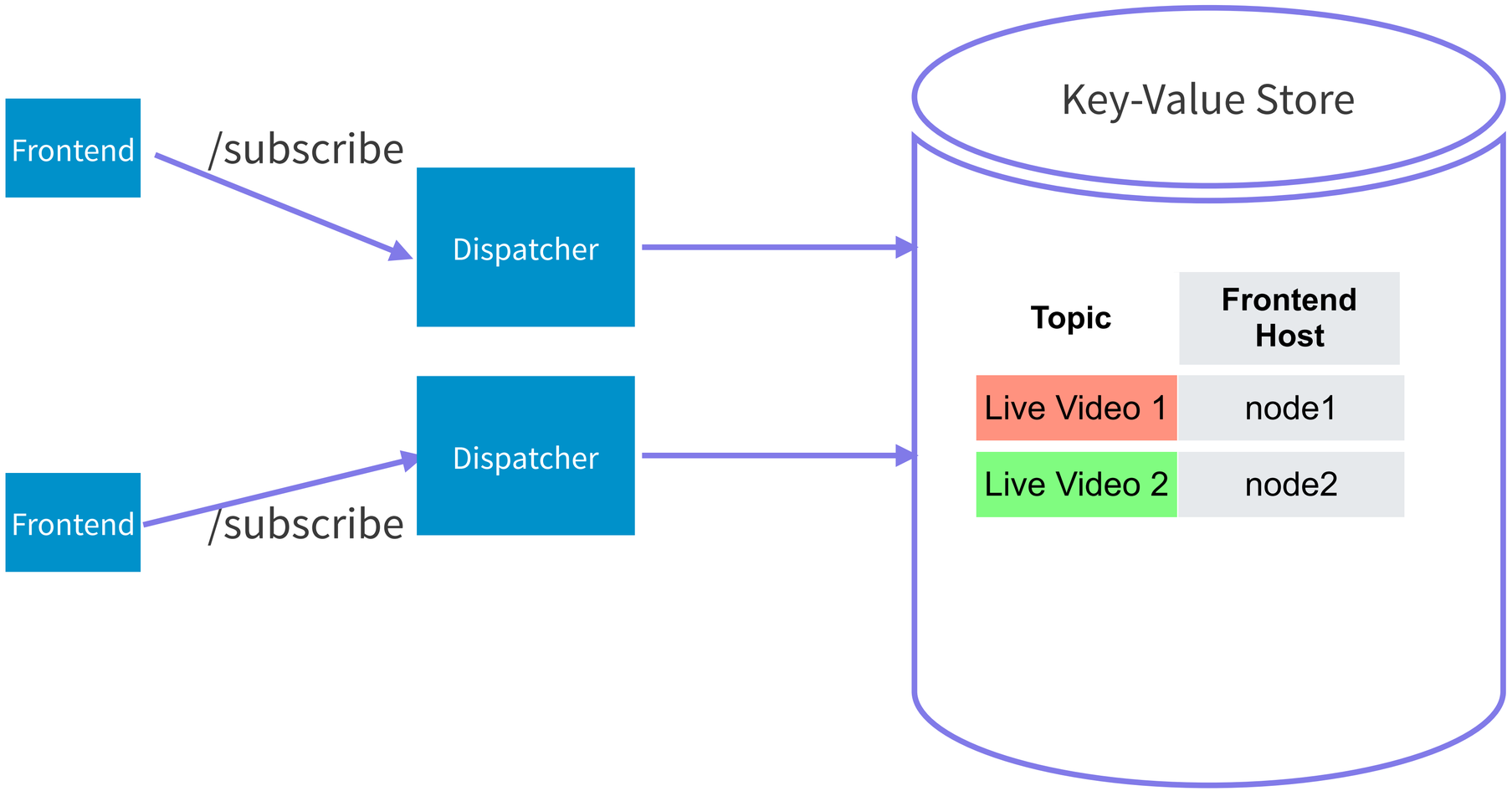
Frontend Node Subscriptions in Dispatcher

DISPATCHER SUBSCRIPTION ENTRY IN KV STORE

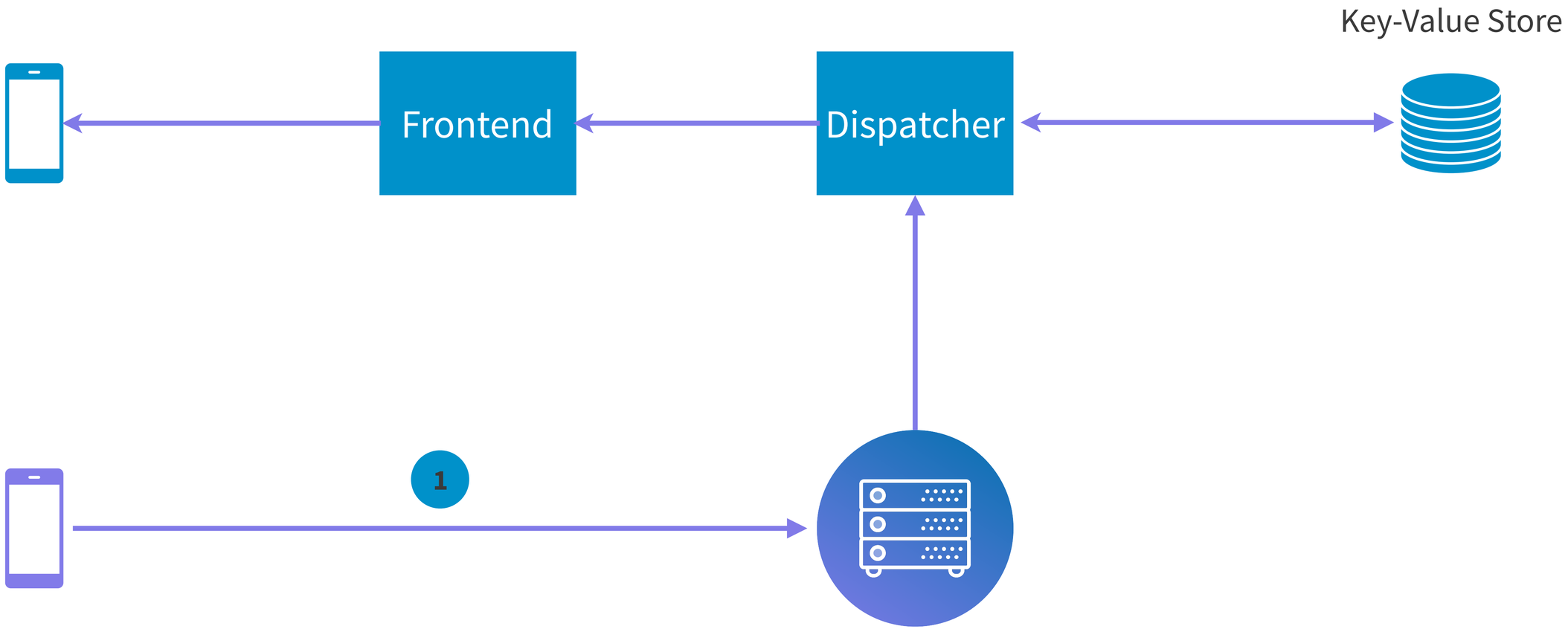


Frontend Node Subscriptions in Dispatcher

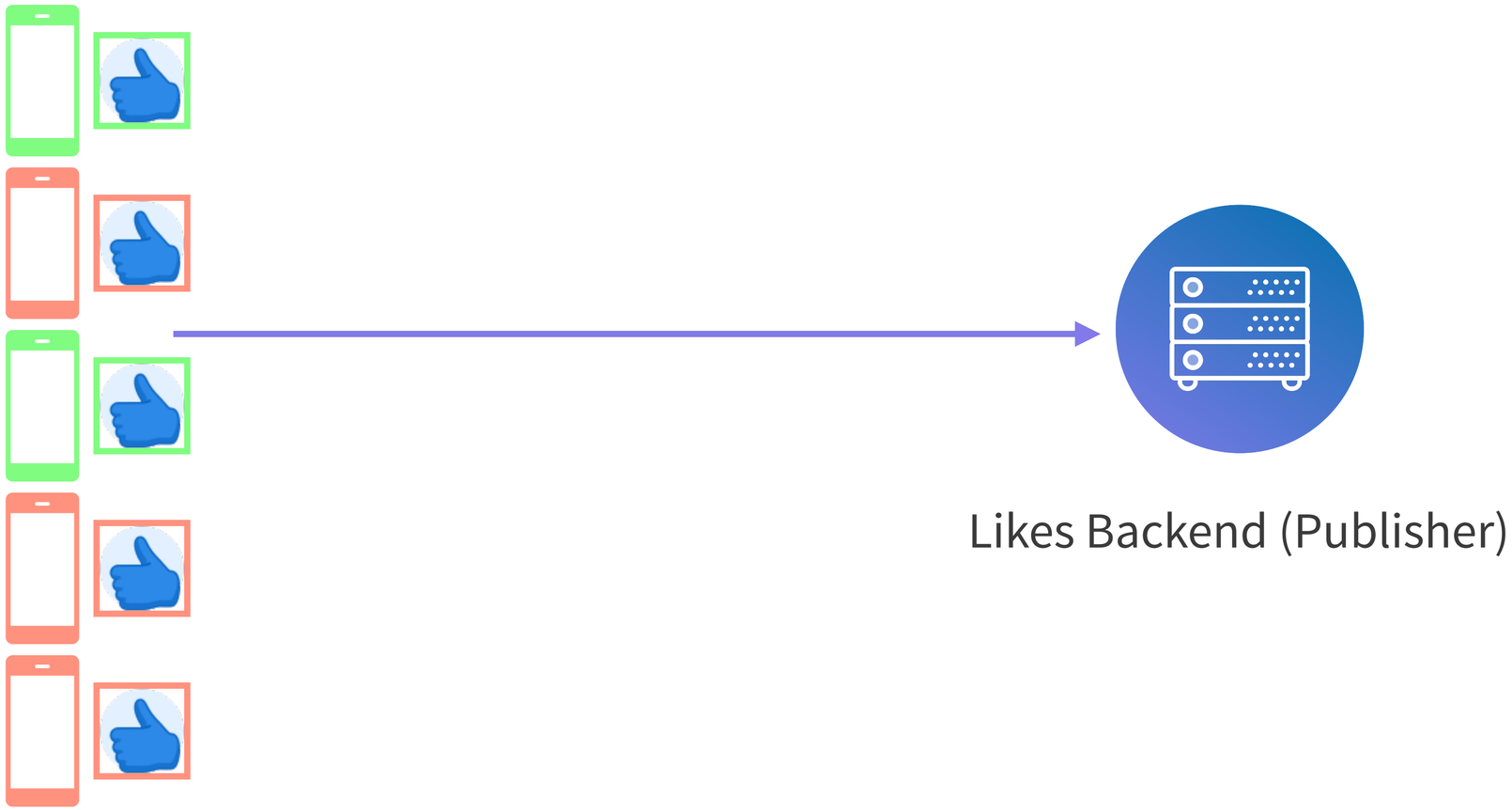
DISPATCHER SUBSCRIPTION ENTRY IN KV STORE



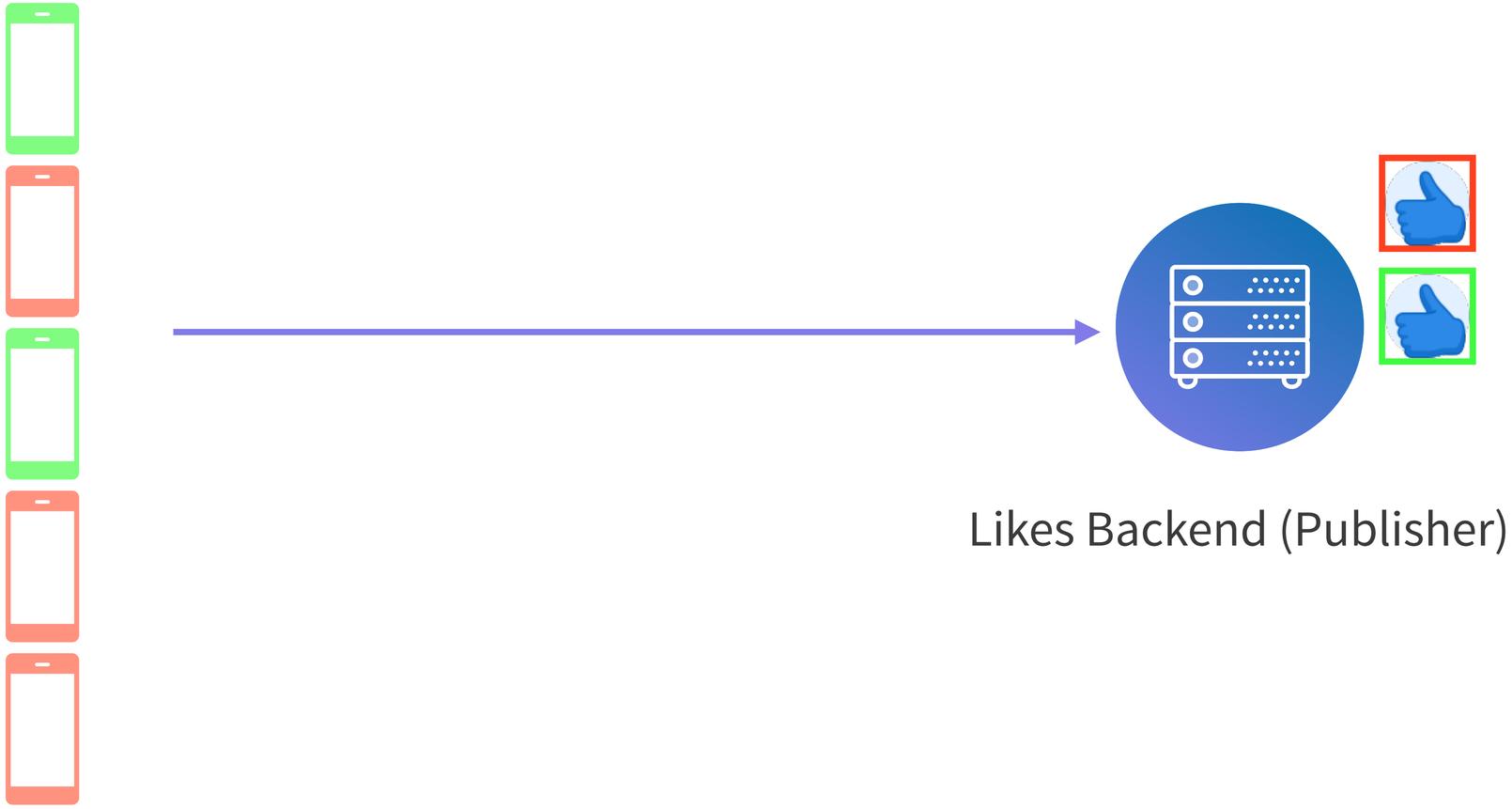
The Publish Flow



Viewers liking Live Videos

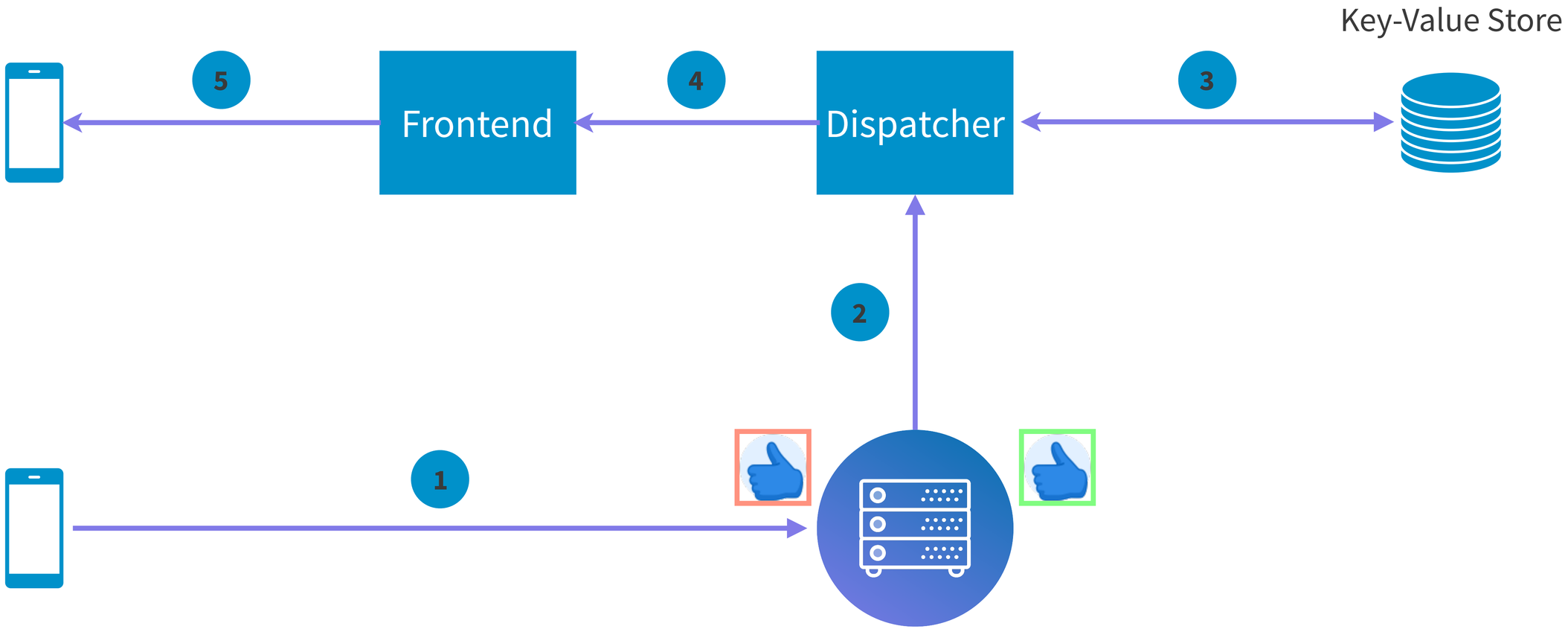


Likes Published to Backend

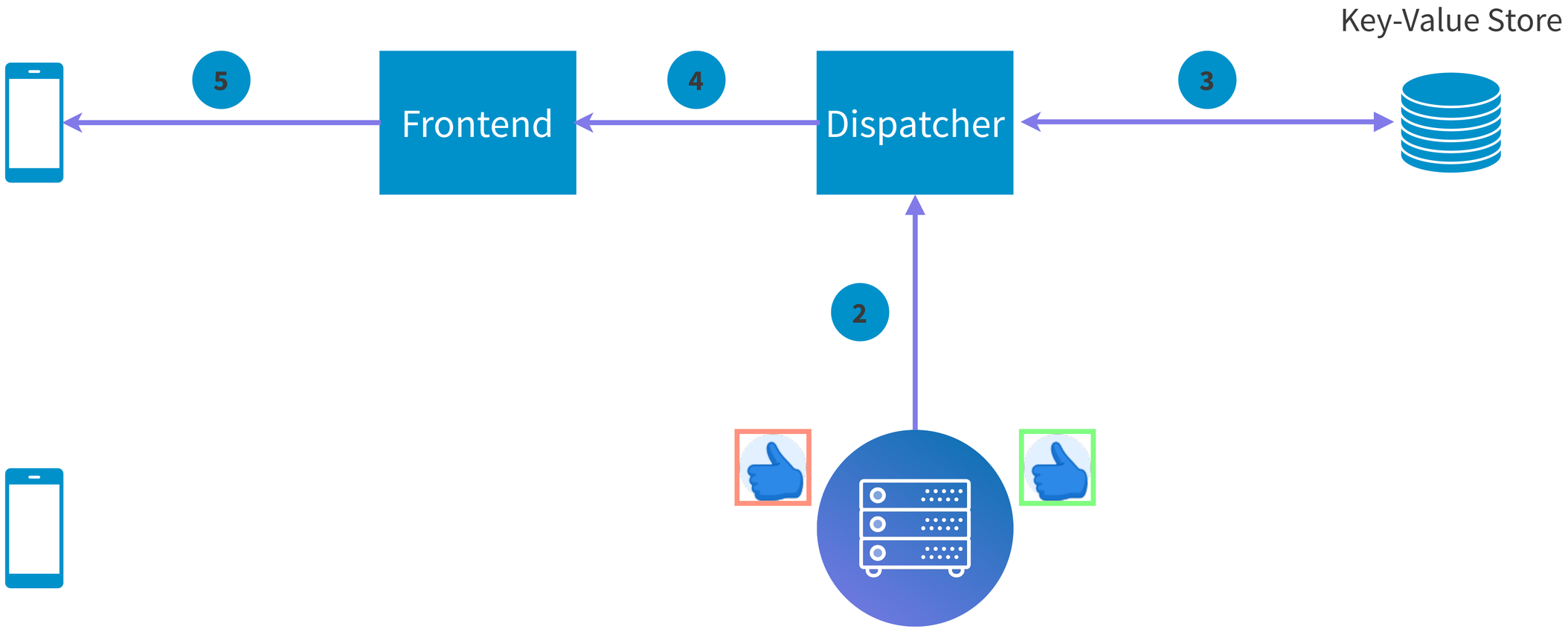


Likes Backend (Publisher)

The Publish Flow

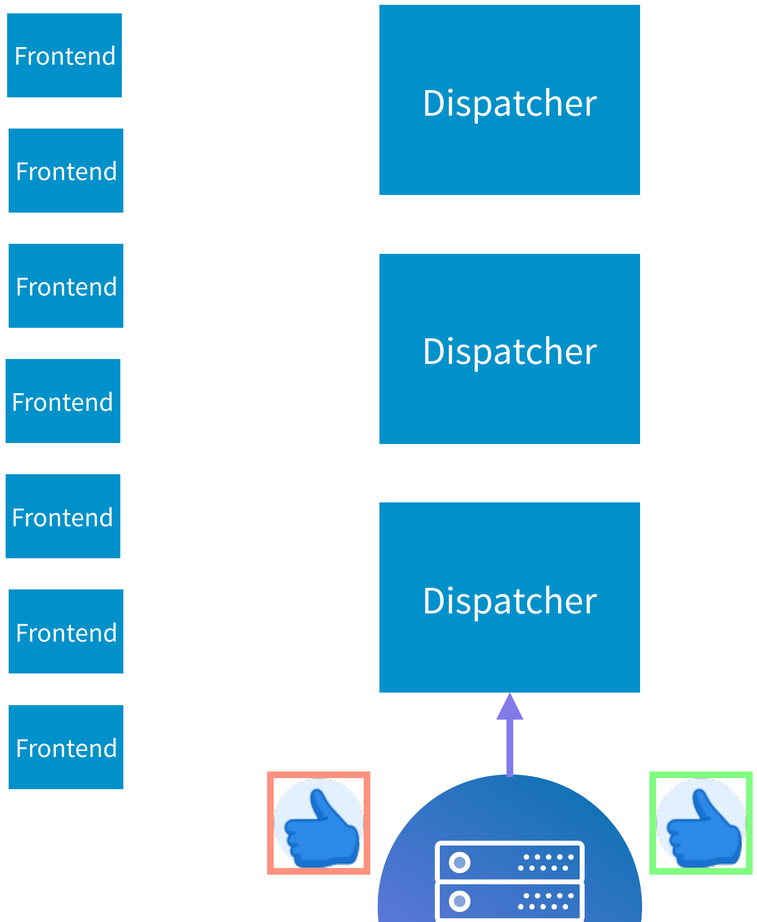


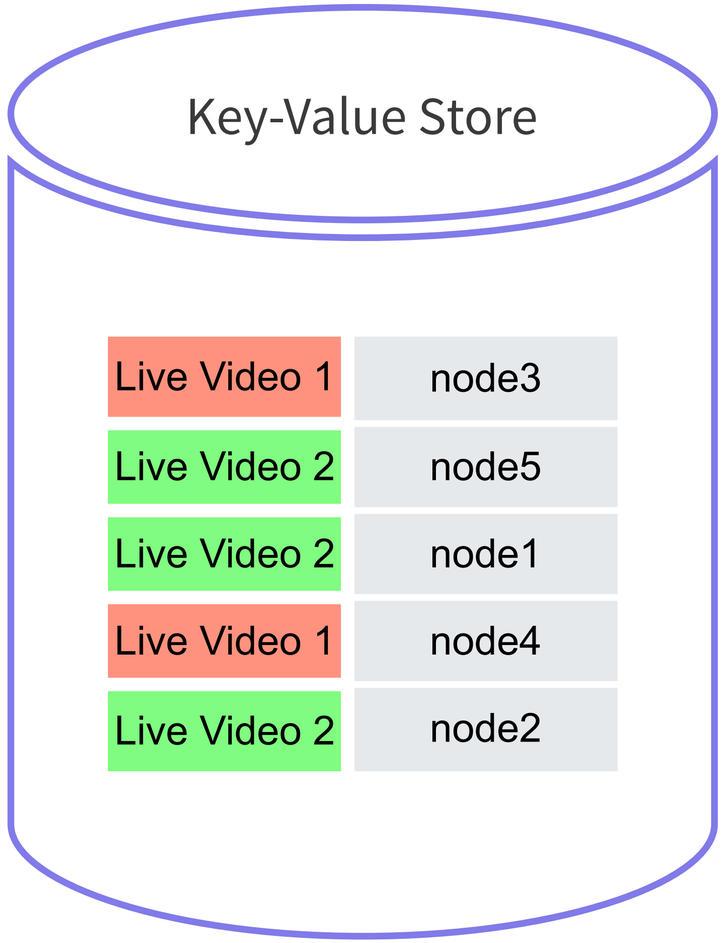
The Publish Flow



Dispatcher Publish to Frontend Nodes

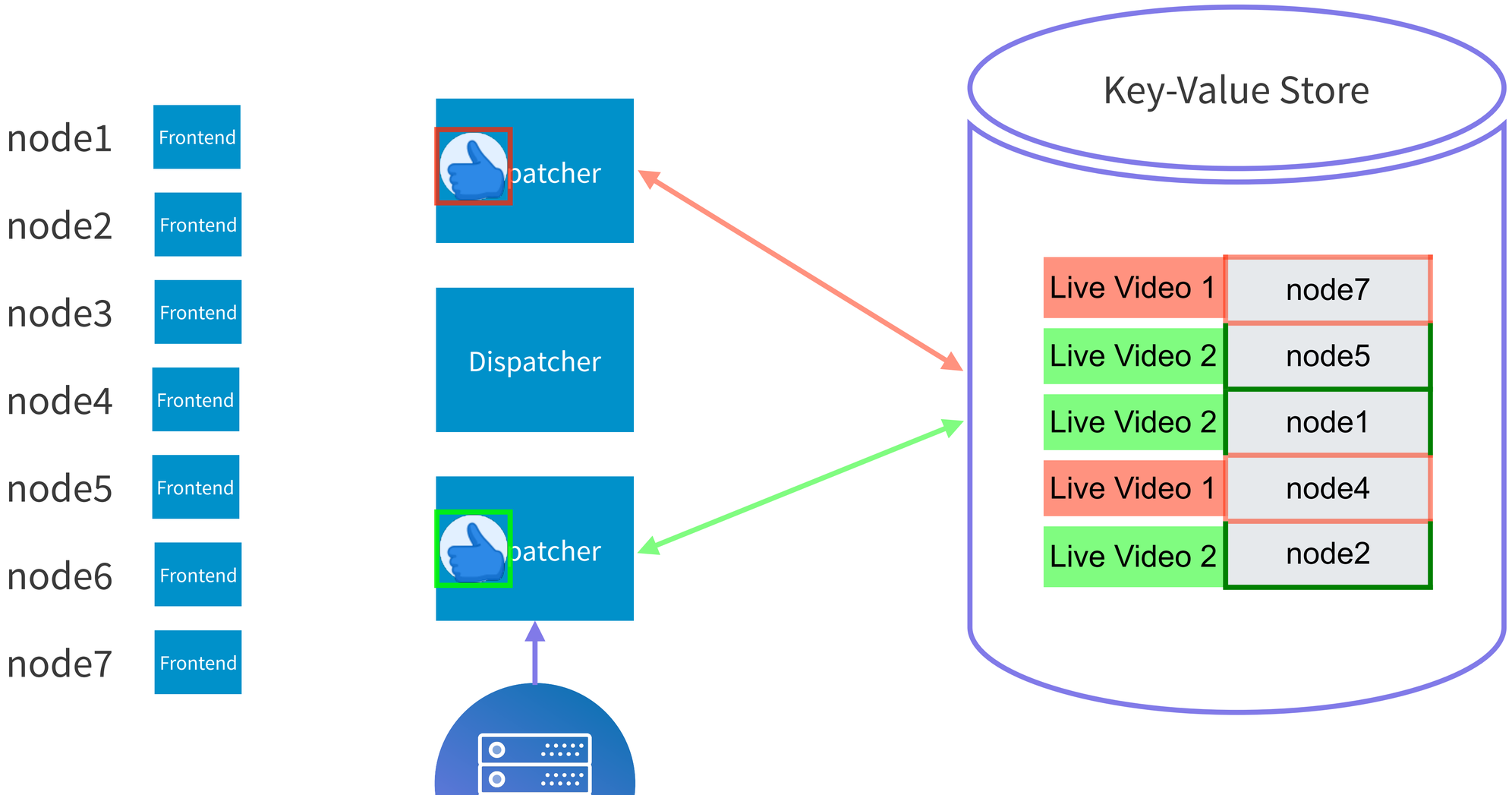
BACKEND PUBLISH TO DISPATCHER NODES





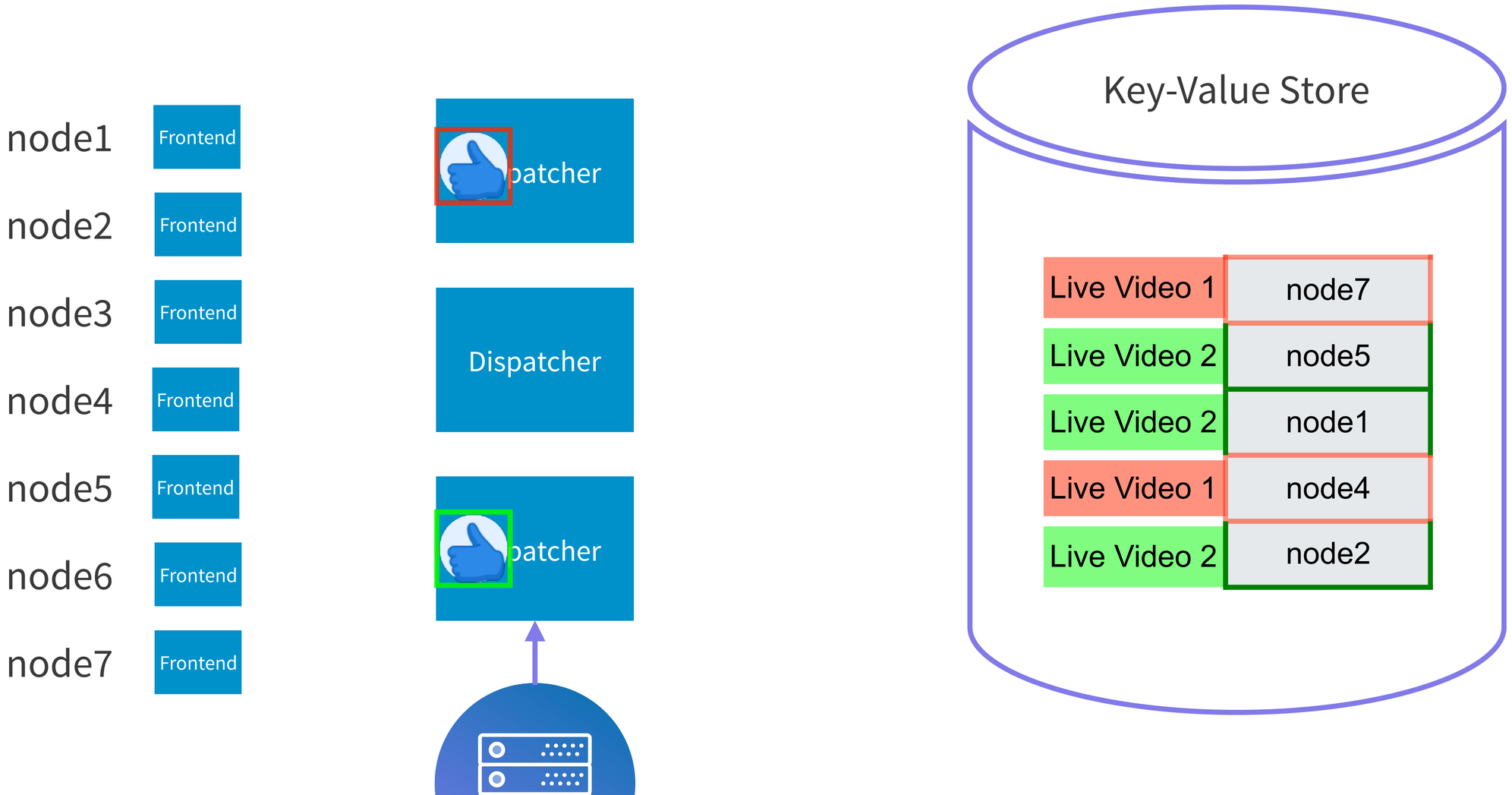
Dispatcher Publish to Frontend Nodes

SUBSCRIBER LOOKUP FROM KV STORE



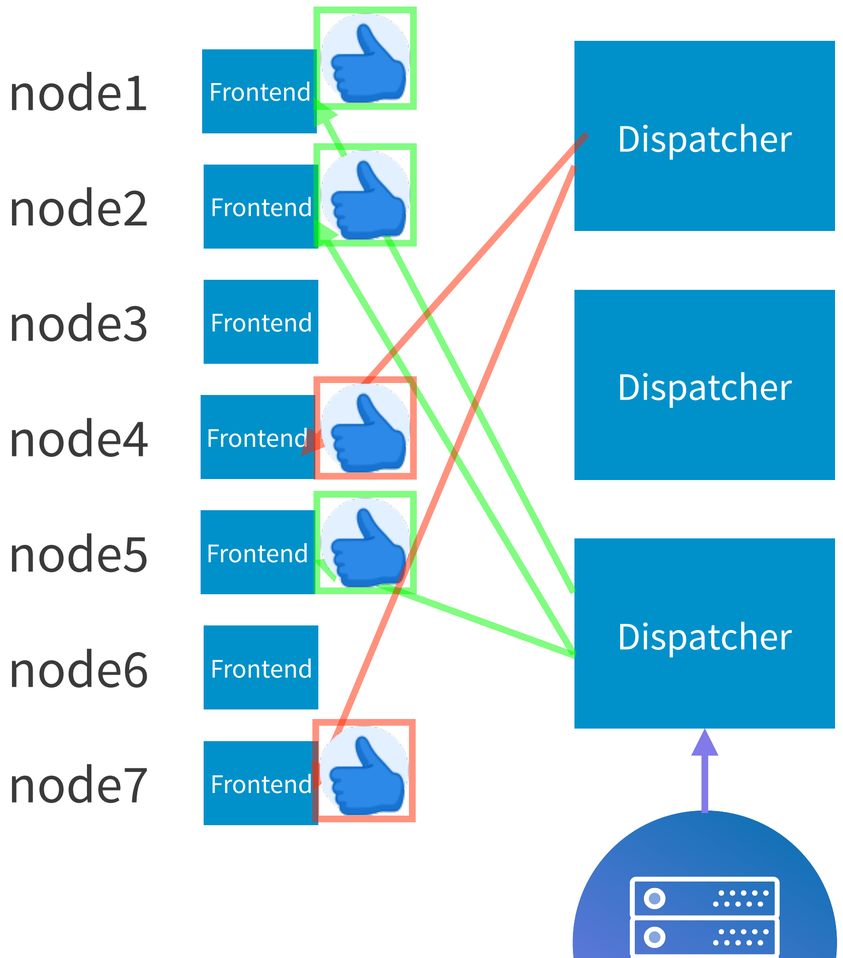
Dispatcher Publish to Frontend Nodes

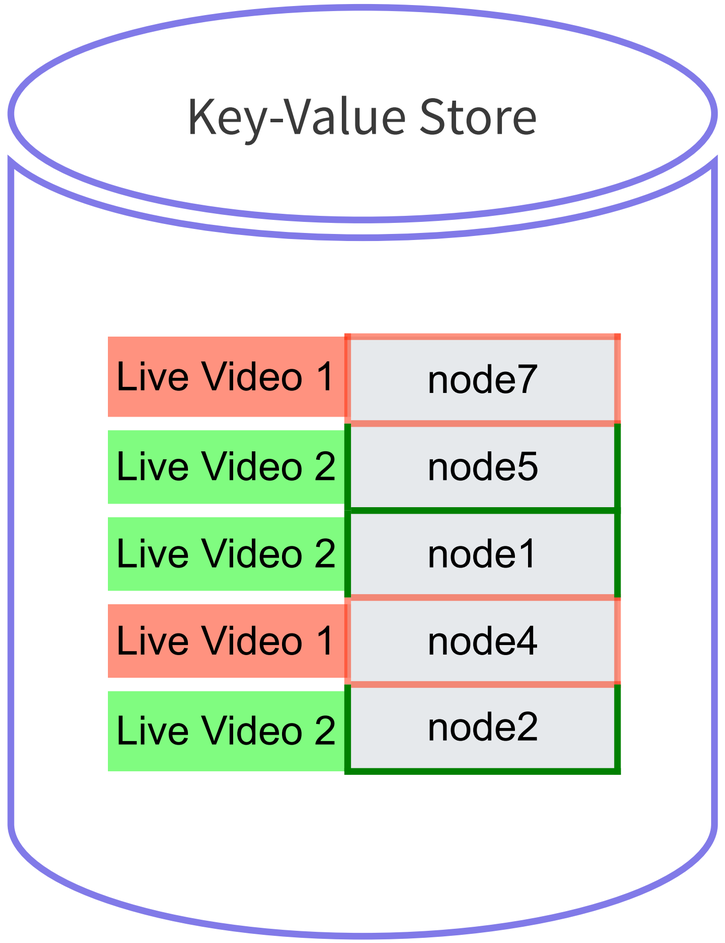
SUBSCRIBER LOOKUP FROM KV STORE



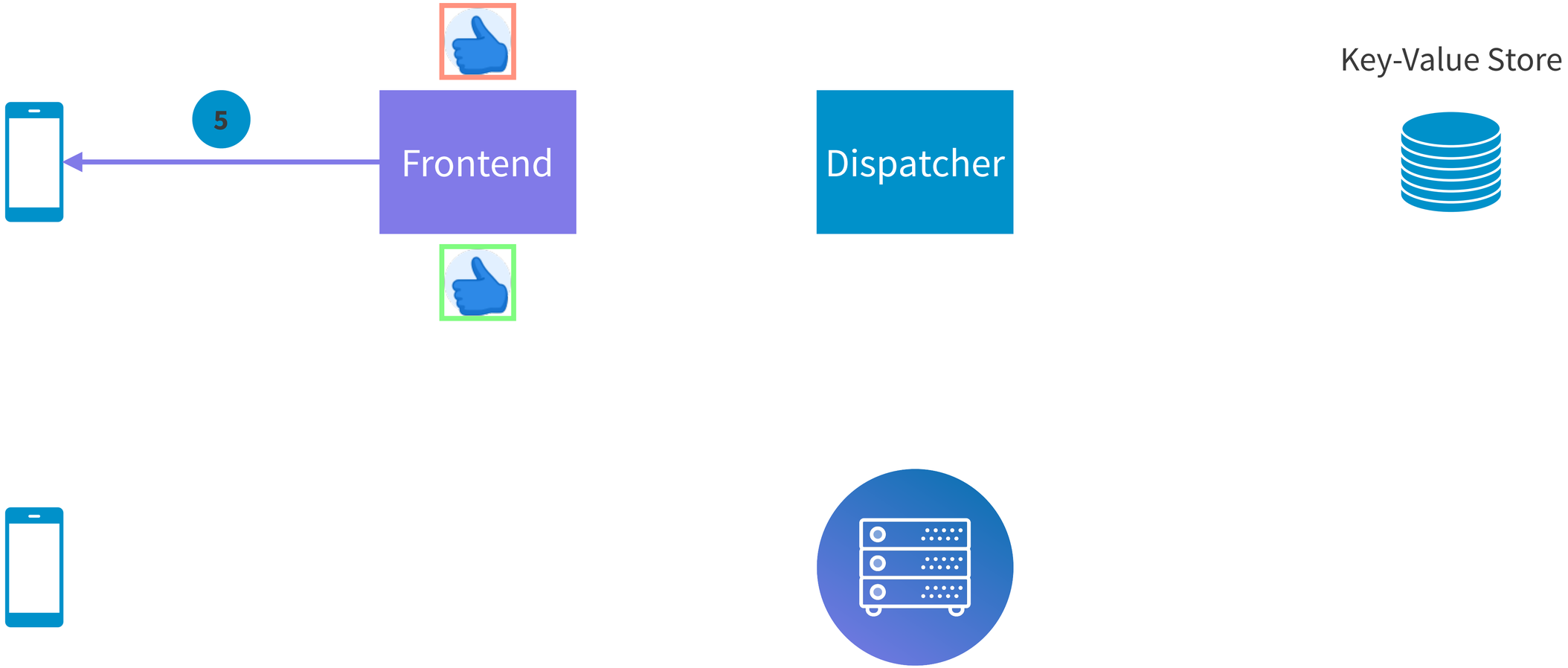
Dispatcher Publish to Frontend Nodes

PUBLISH TO FRONTEND NODES



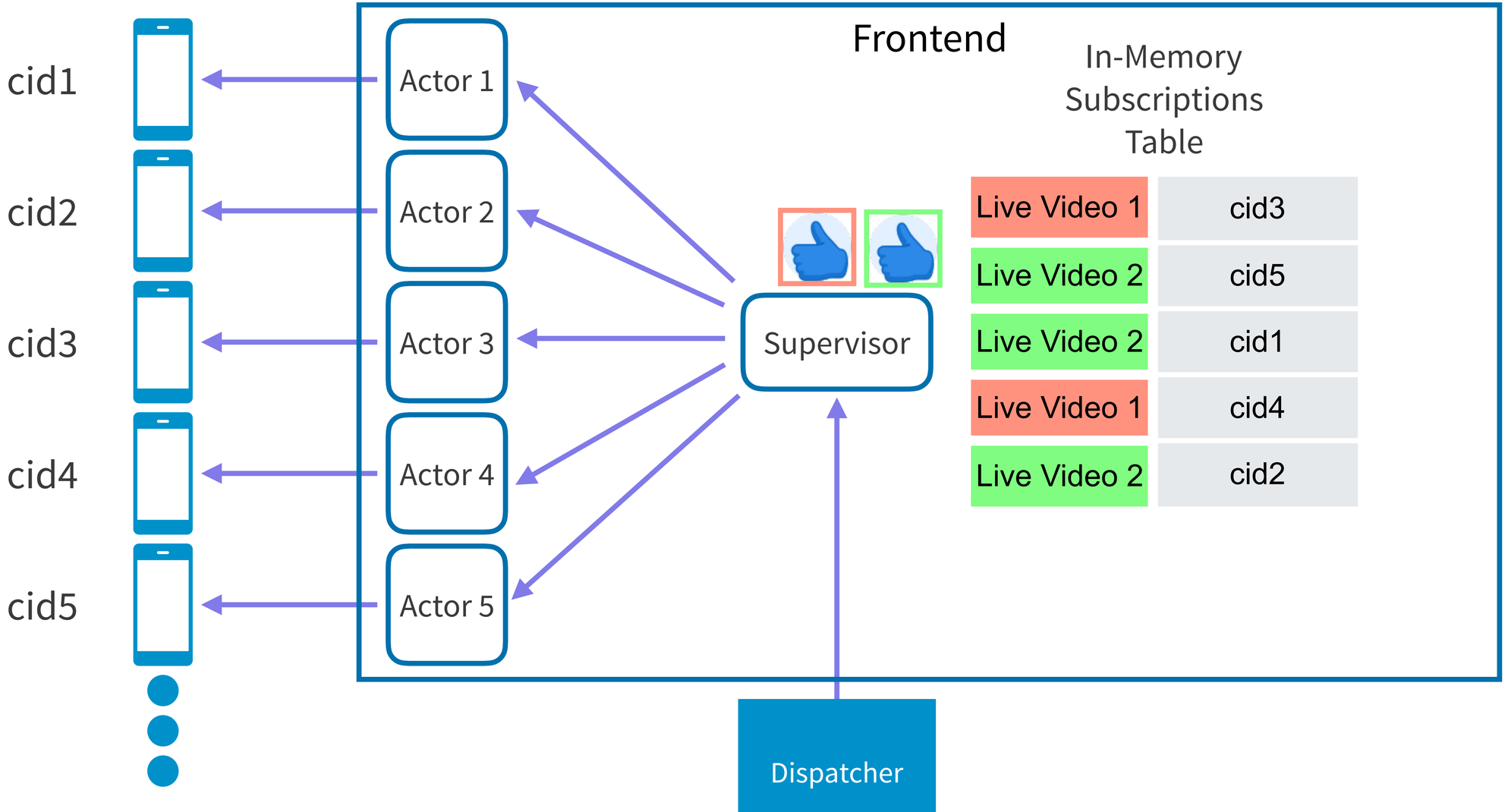


The Publish Flow



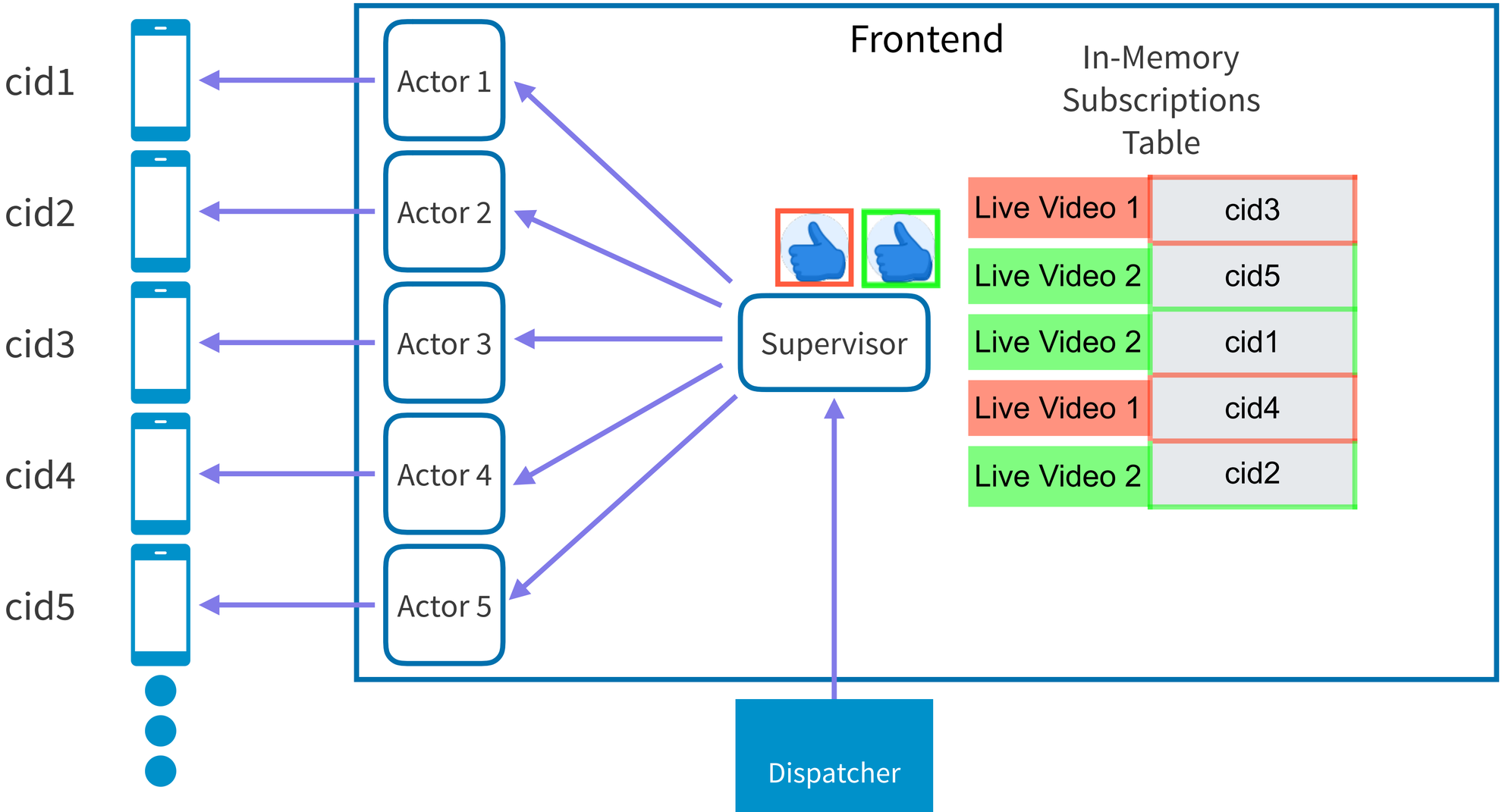
Frontend Publish to Mobile Clients

DISPATCHER PUBLISH TO FRONTEND NODE SUPERVISOR ACTOR



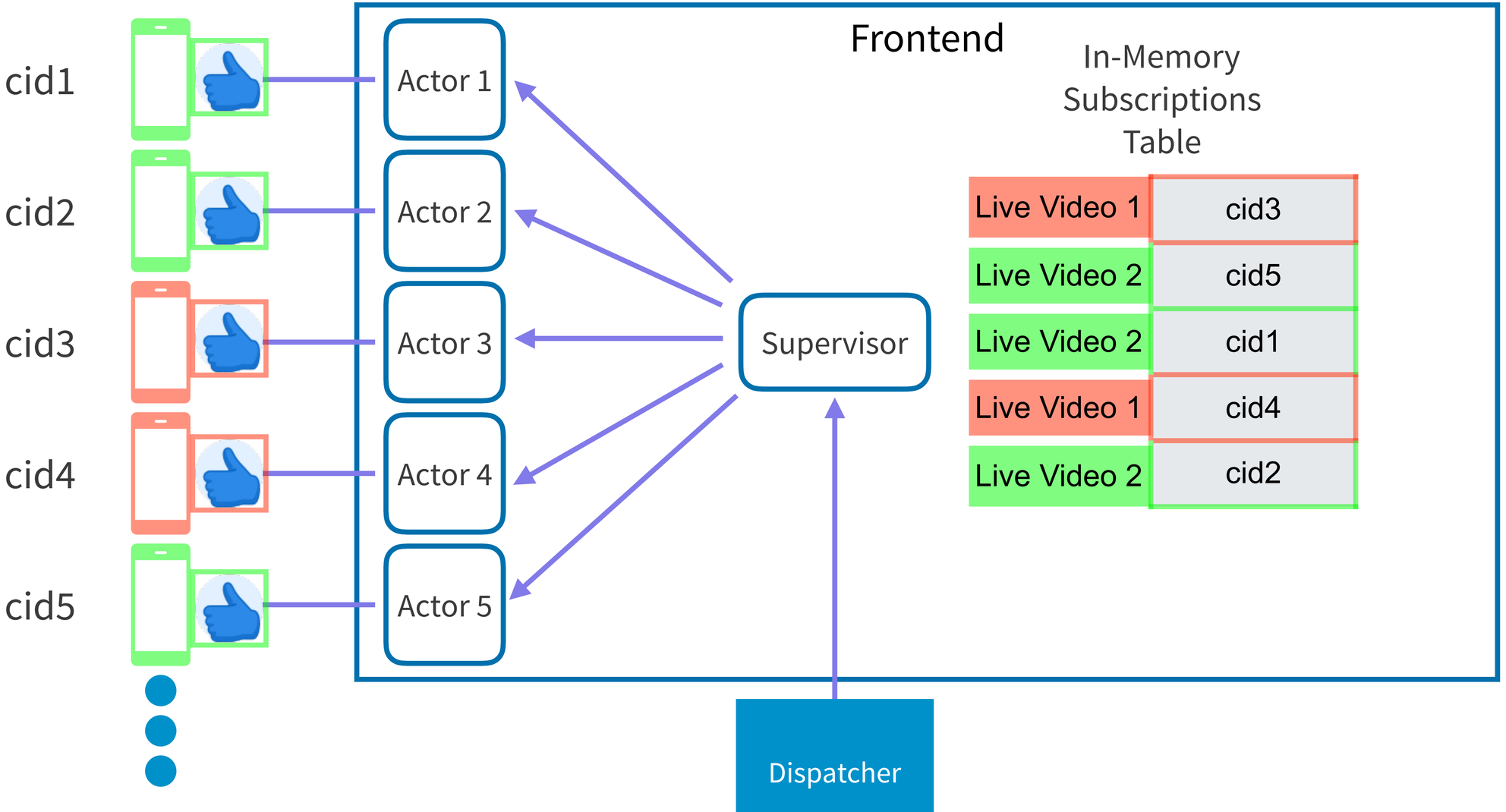
Frontend Publish to Mobile Clients

SUBSCRIBED CLIENT CONNECTION LOOKUP FROM IN-MEMORY TABLE



Frontend Publish to Mobile Clients

PUBLISH TO MOBILE CLIENTS

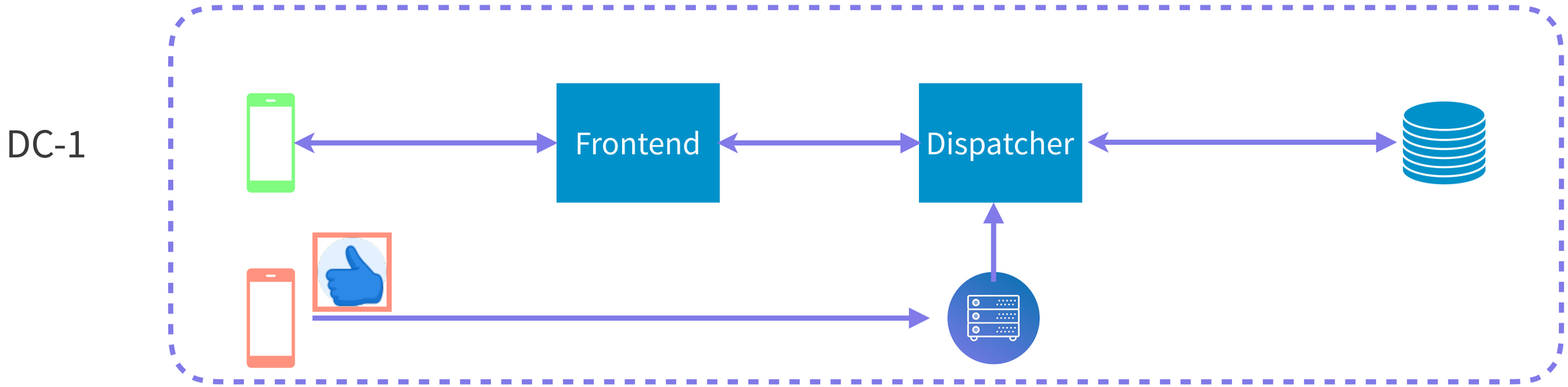


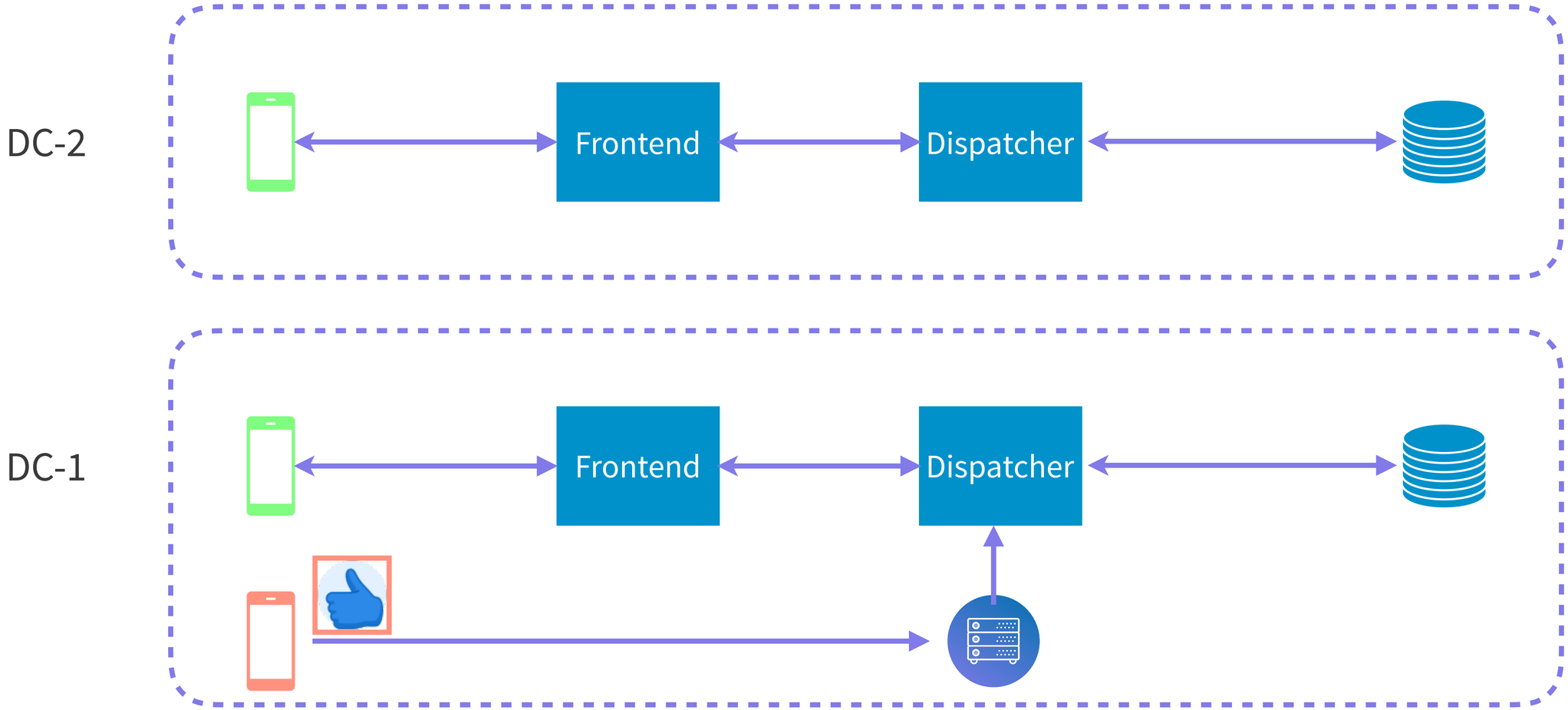
Bonus Challenge: Another data center

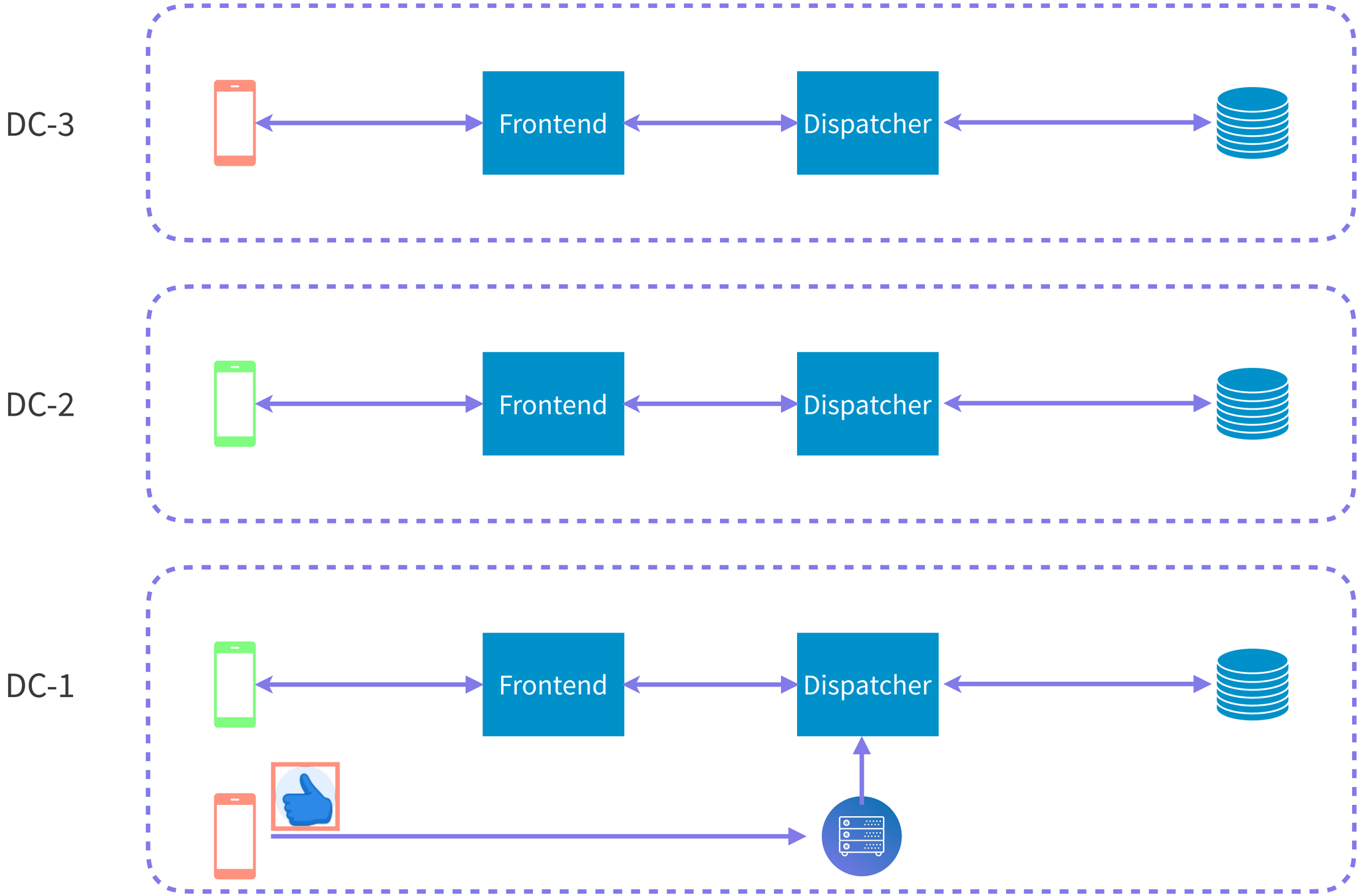

PUMA
AMERICAN EAGLE
UNREAL
REAL TIME
REAL TIME
REAL TIME

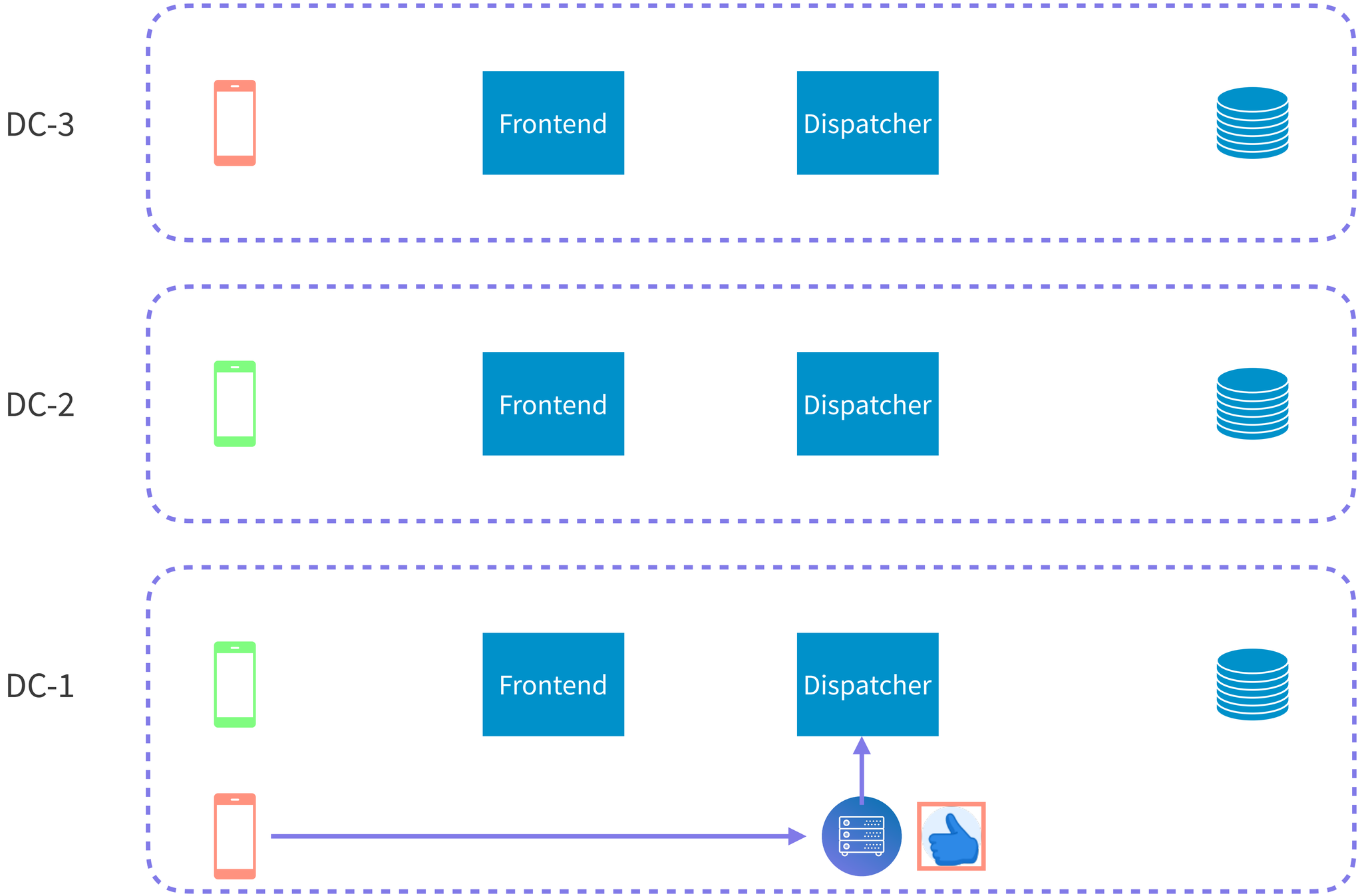
Distributed Systems

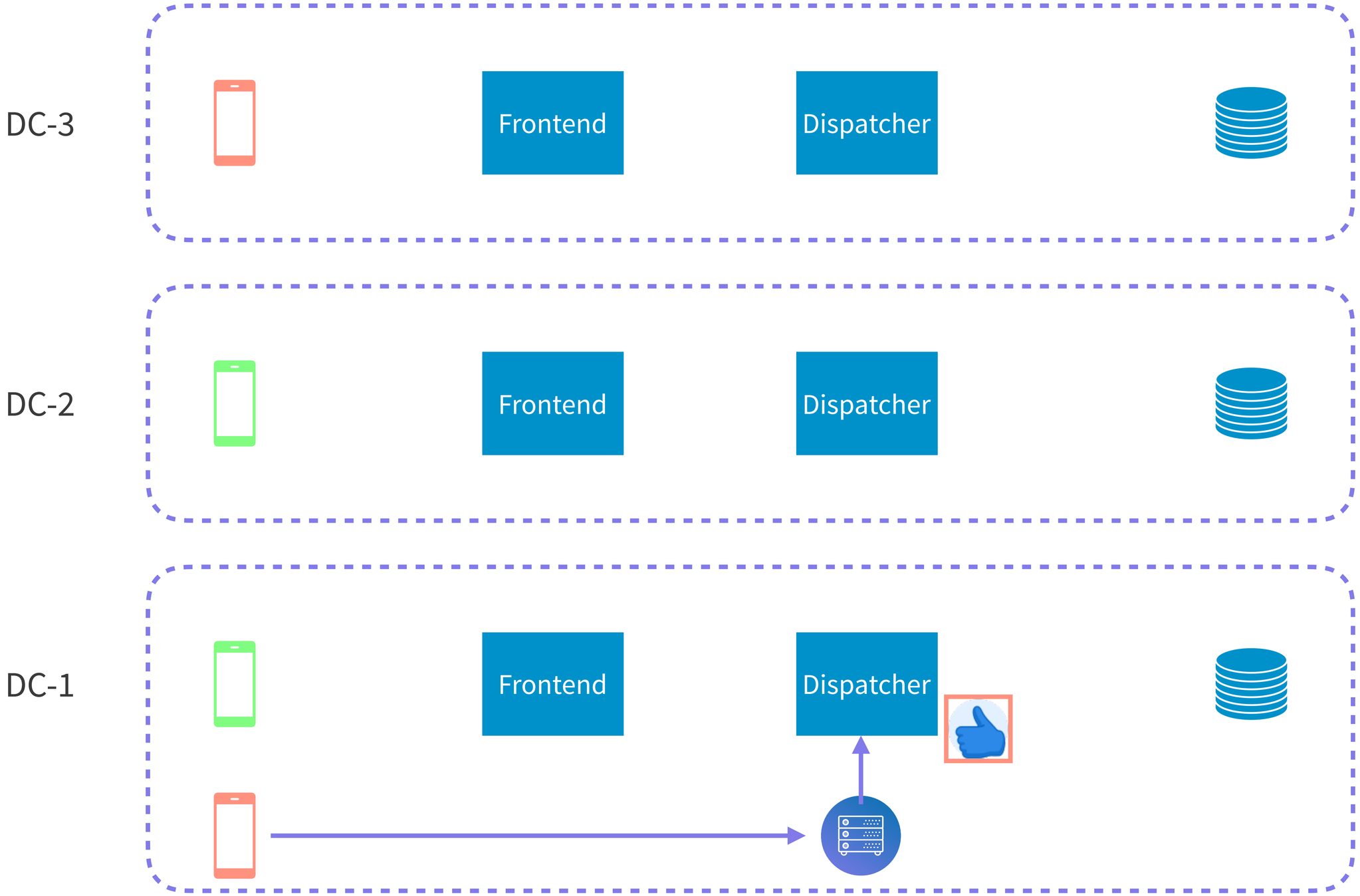
Start small and add simple layers to your architecture

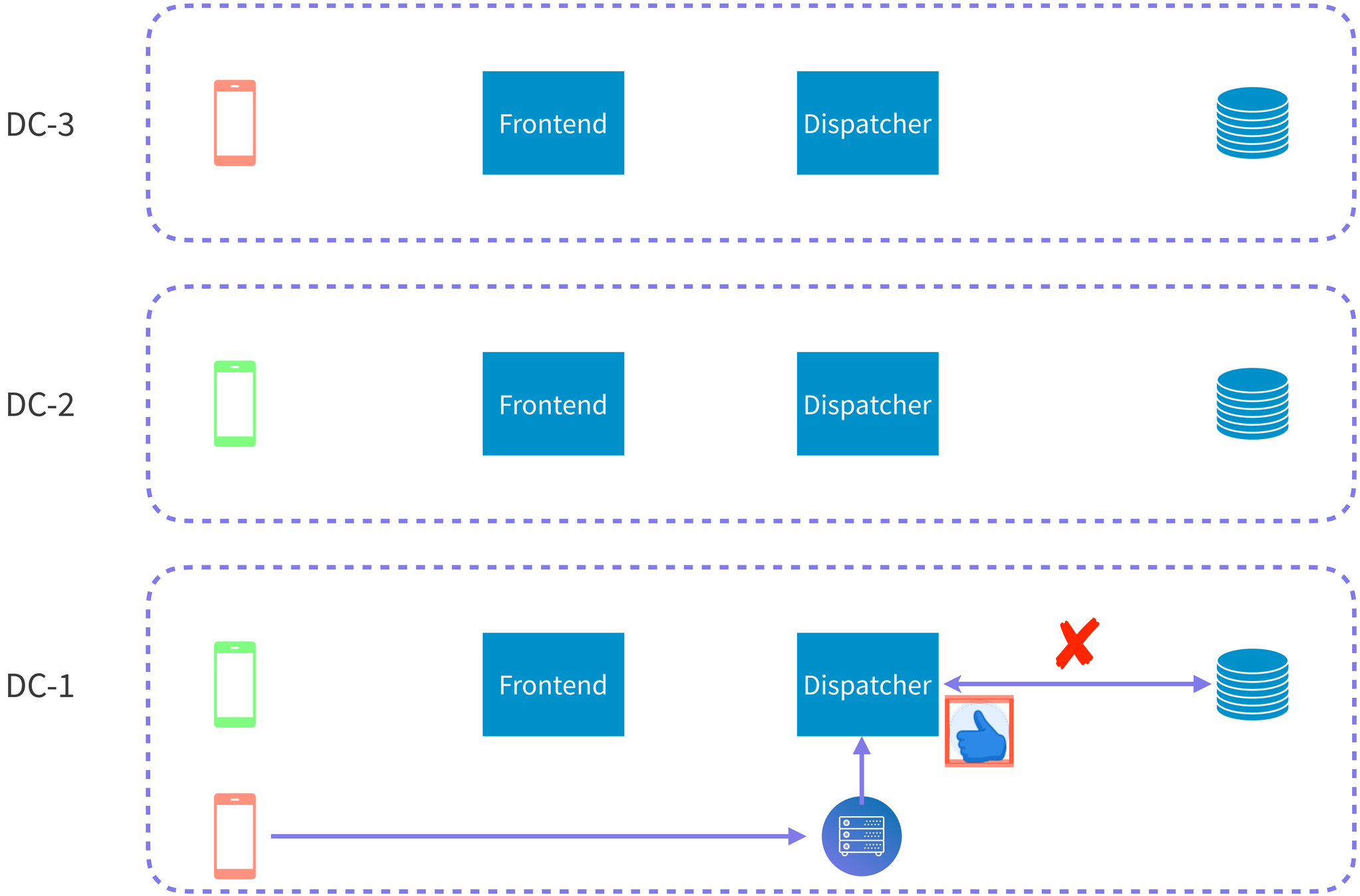

DC-1
Frontend
Dispatcher

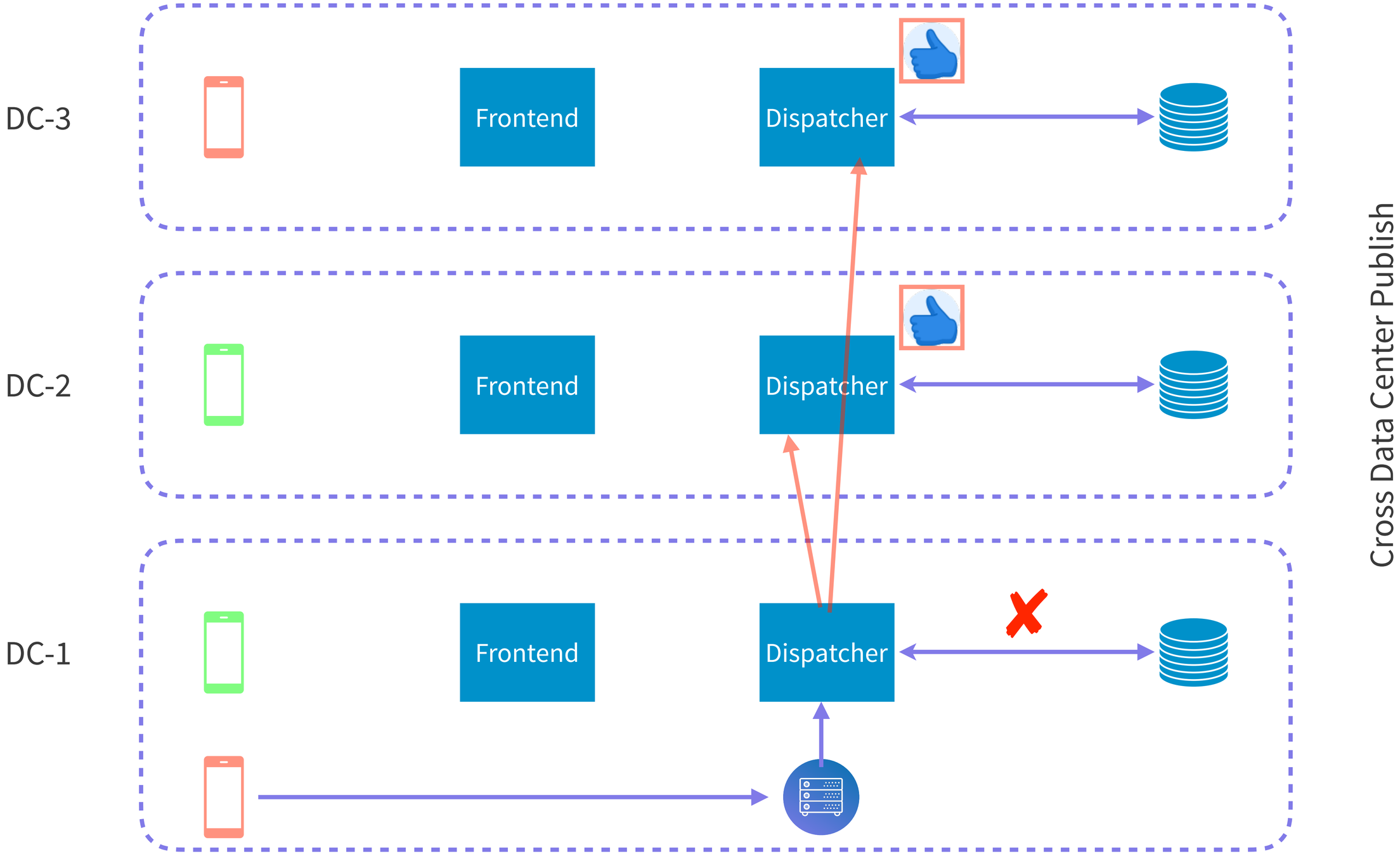

DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher

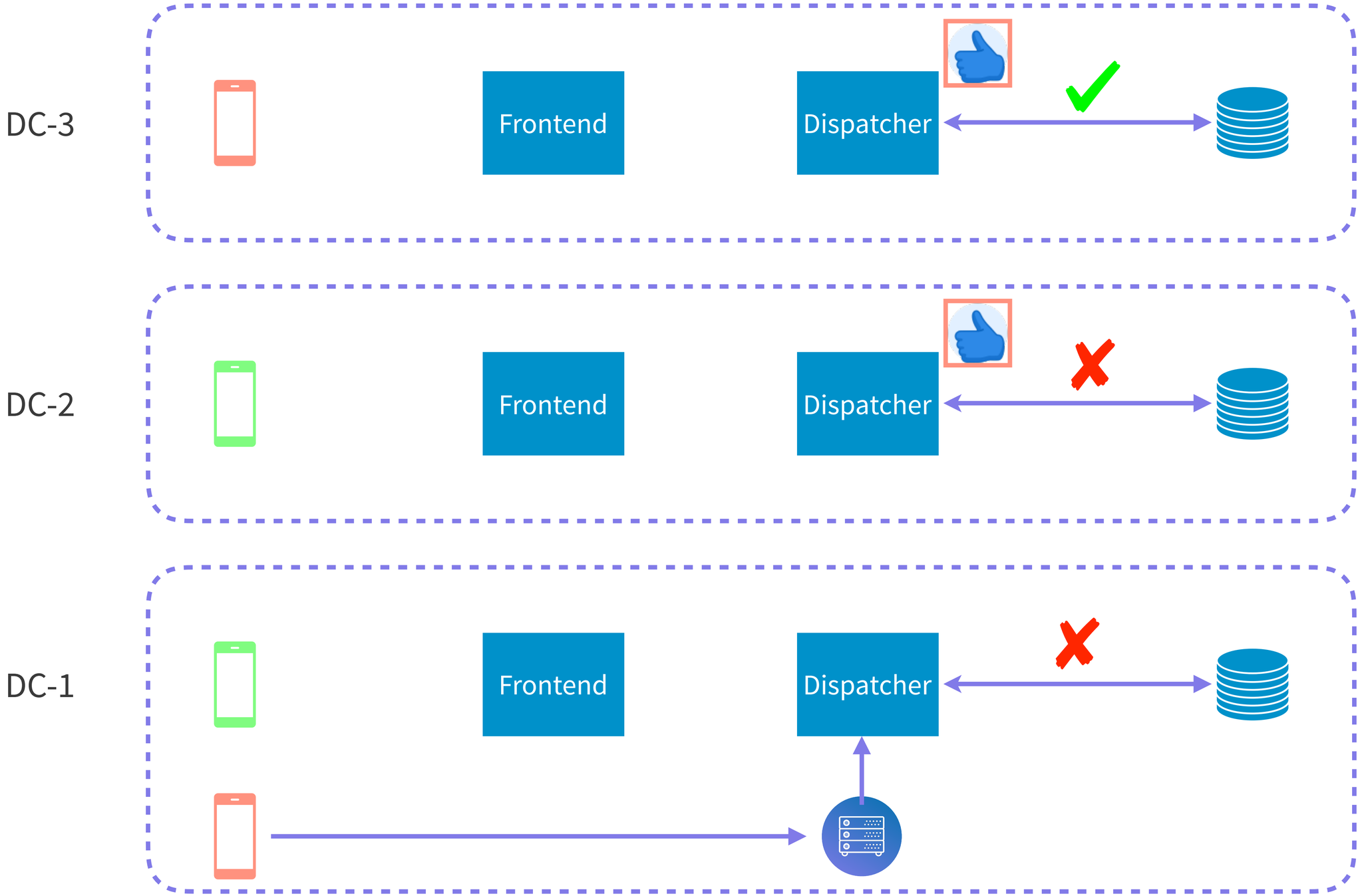

DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher

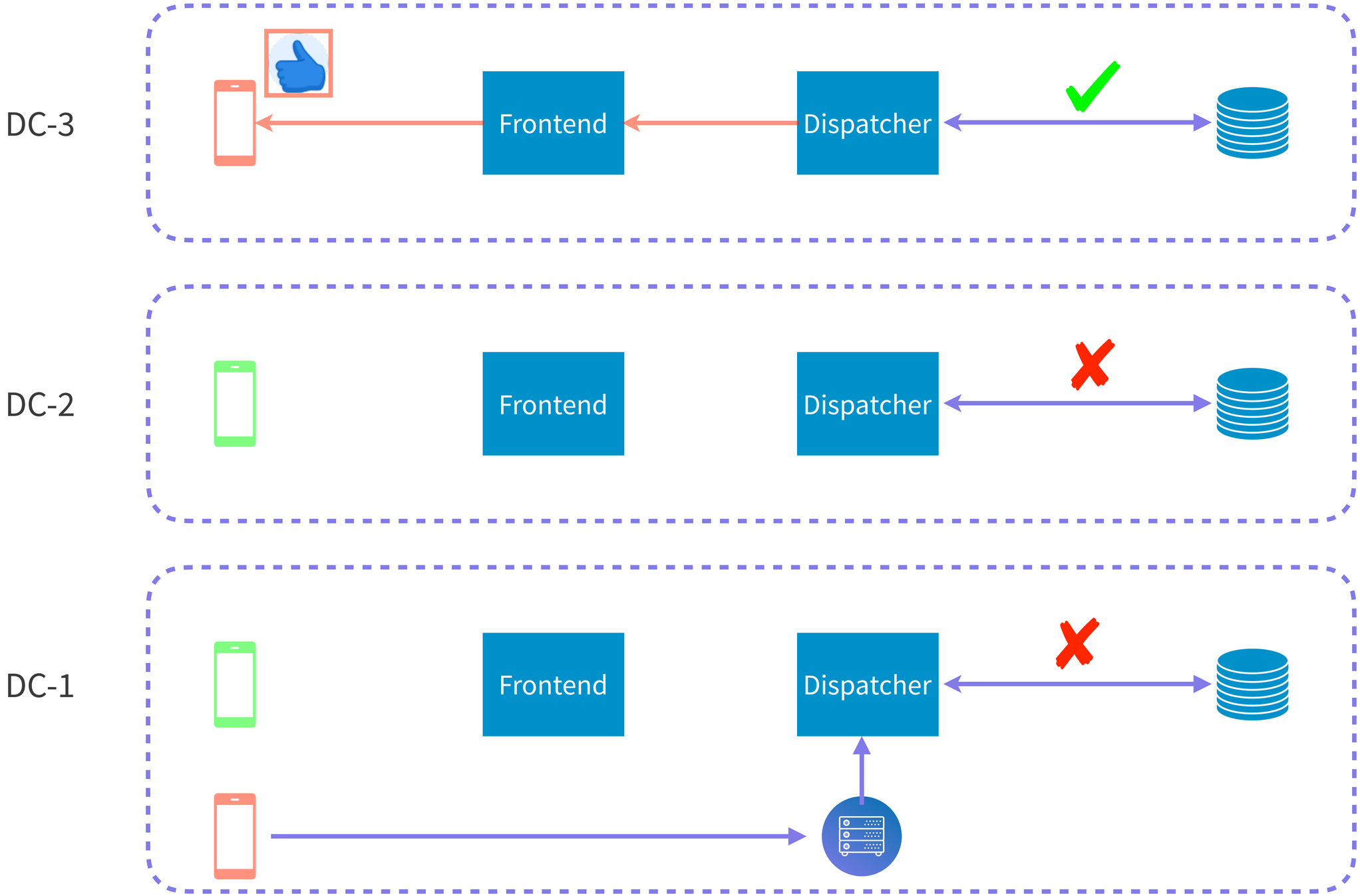

DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher


DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher

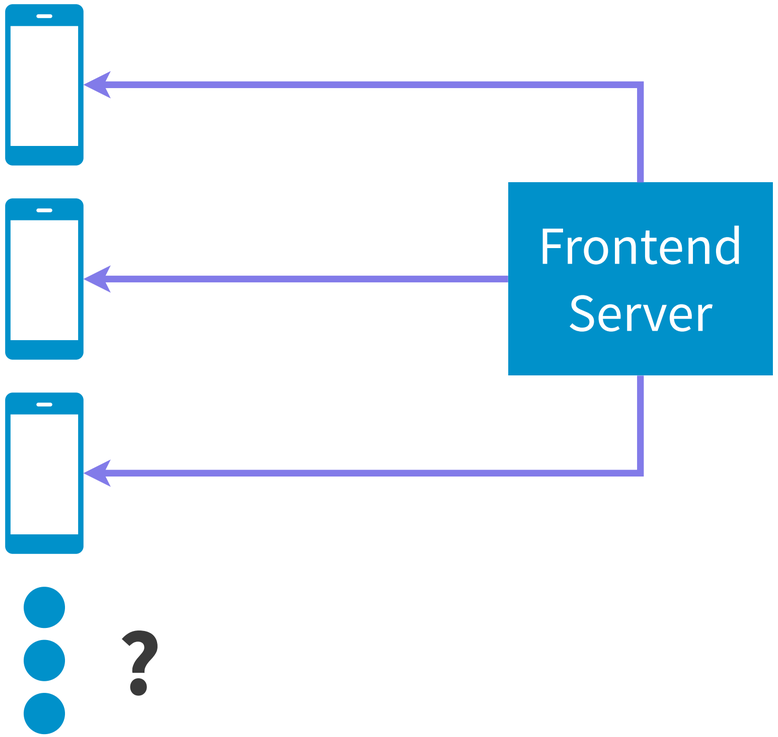

DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher


DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher
Cross Data Center Publish


DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher


DC-3
Frontend
Dispatcher
DC-2
Frontend
Dispatcher
DC-1
Frontend
Dispatcher

Performance & Scale



How many persistent connections can a single machine hold?

> 100,000

ROYAL WEDDING

Second Largest Stream

>18M

viewers

© time.com

> 100,000

We can hold those 18M connections with just 180 machines

LINKEDIN ENGINEERING BLOG

tiny.cc/linkedinscaling

>100K

LinkedIn Realtime Performance

>100K

Persistent Connections/machine

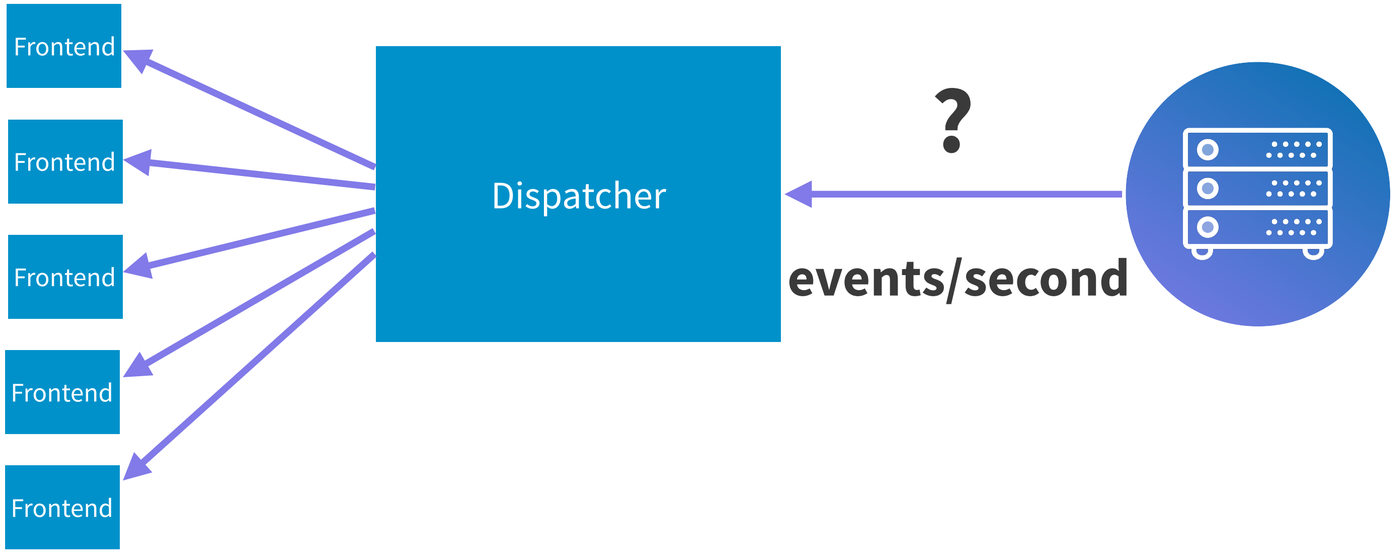
5K/s

Dispatcher Publish QPS/machine

75ms

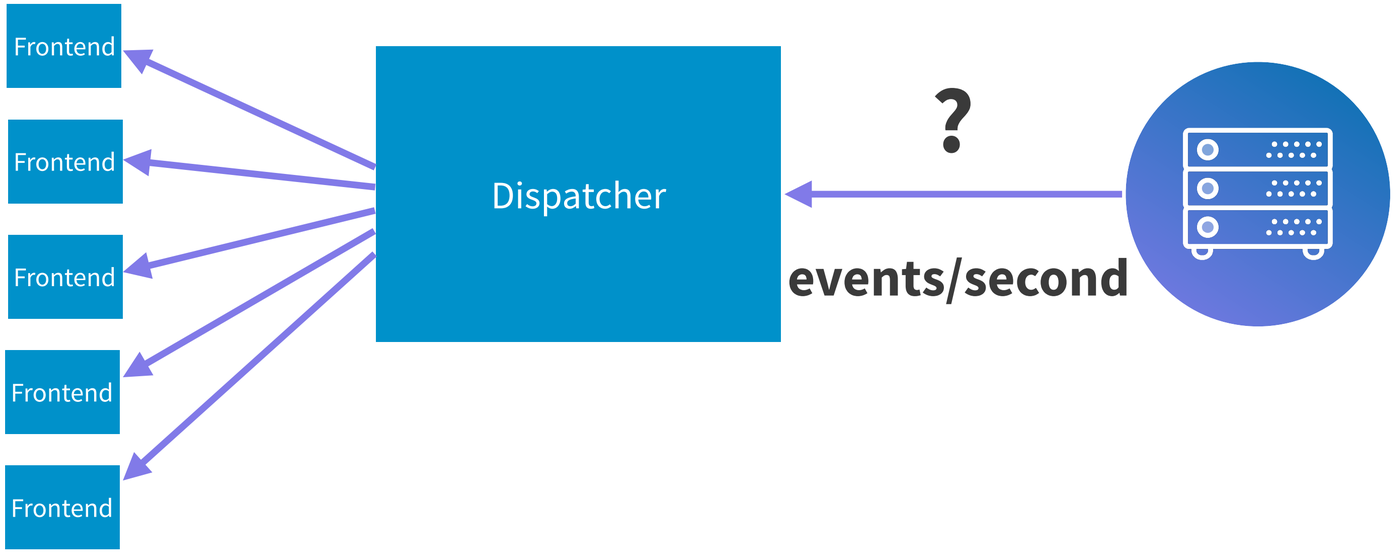
E2E Publish Latency

Dispatcher Performance



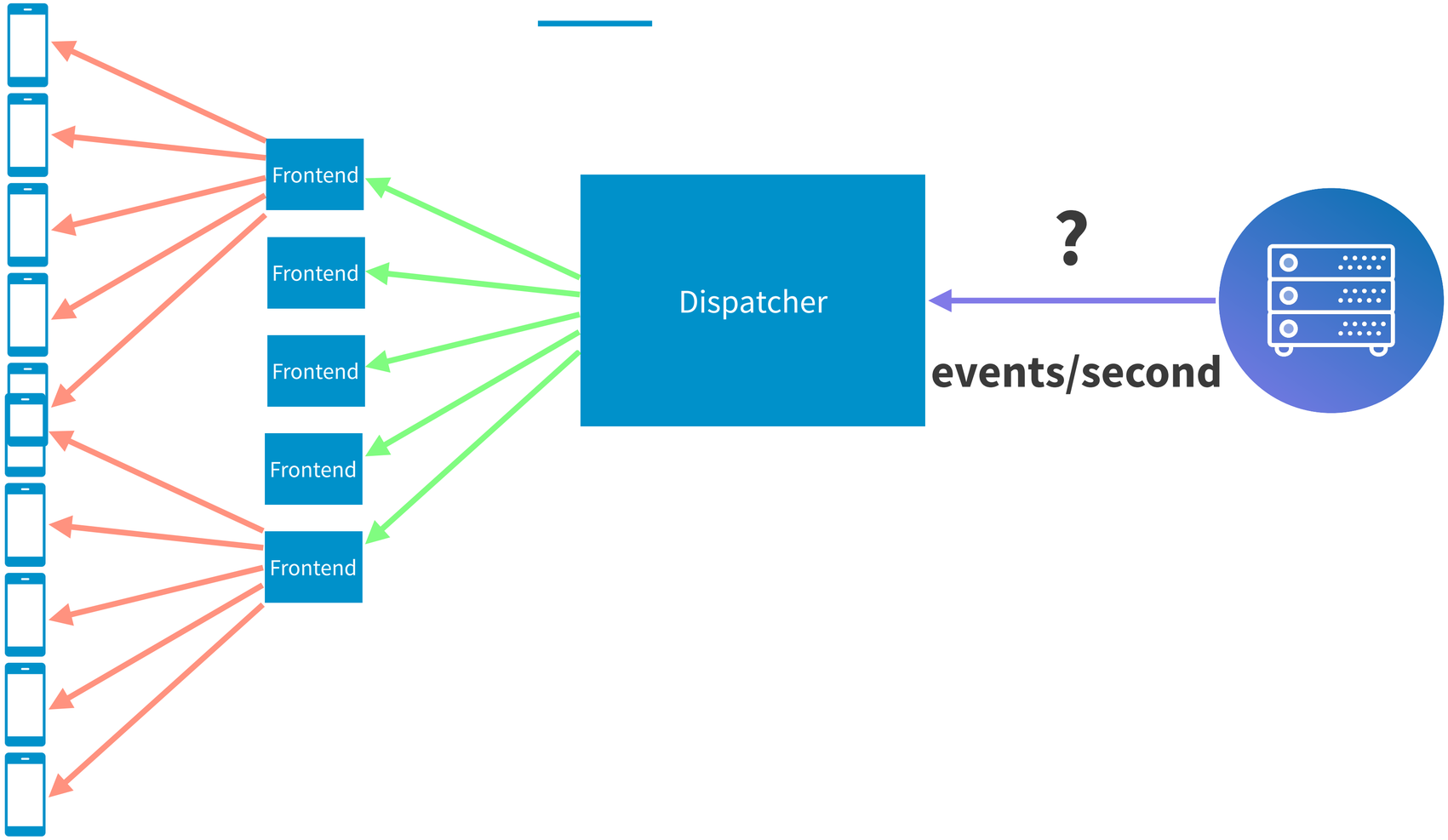
How many events per second can be published to a single dispatcher machine?

Dispatcher Performance



Number of frontend nodes to publish to is limited per event

Dispatcher Performance



5K

We can publish 50K likes per second with just 10 machines

LinkedIn Realtime Performance

>100K

Persistent Connections/machine

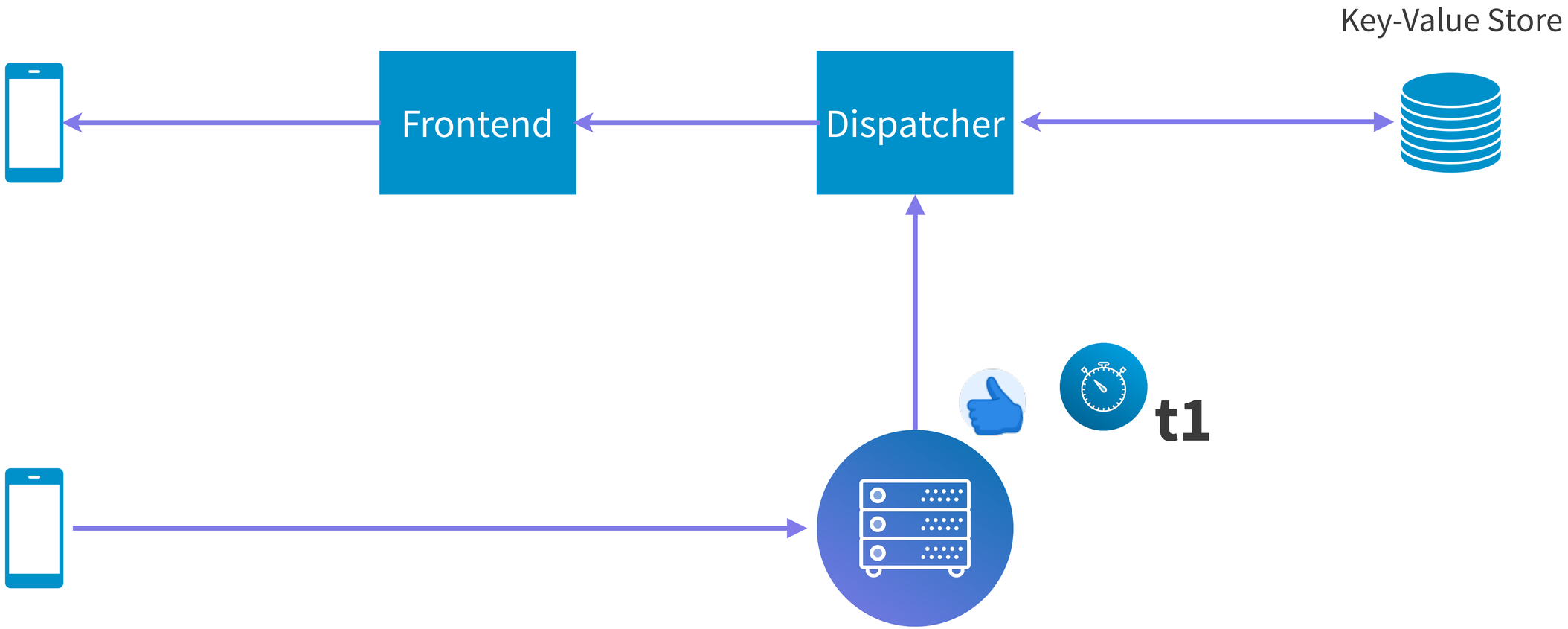
5K

Dispatcher Publish QPS/machine

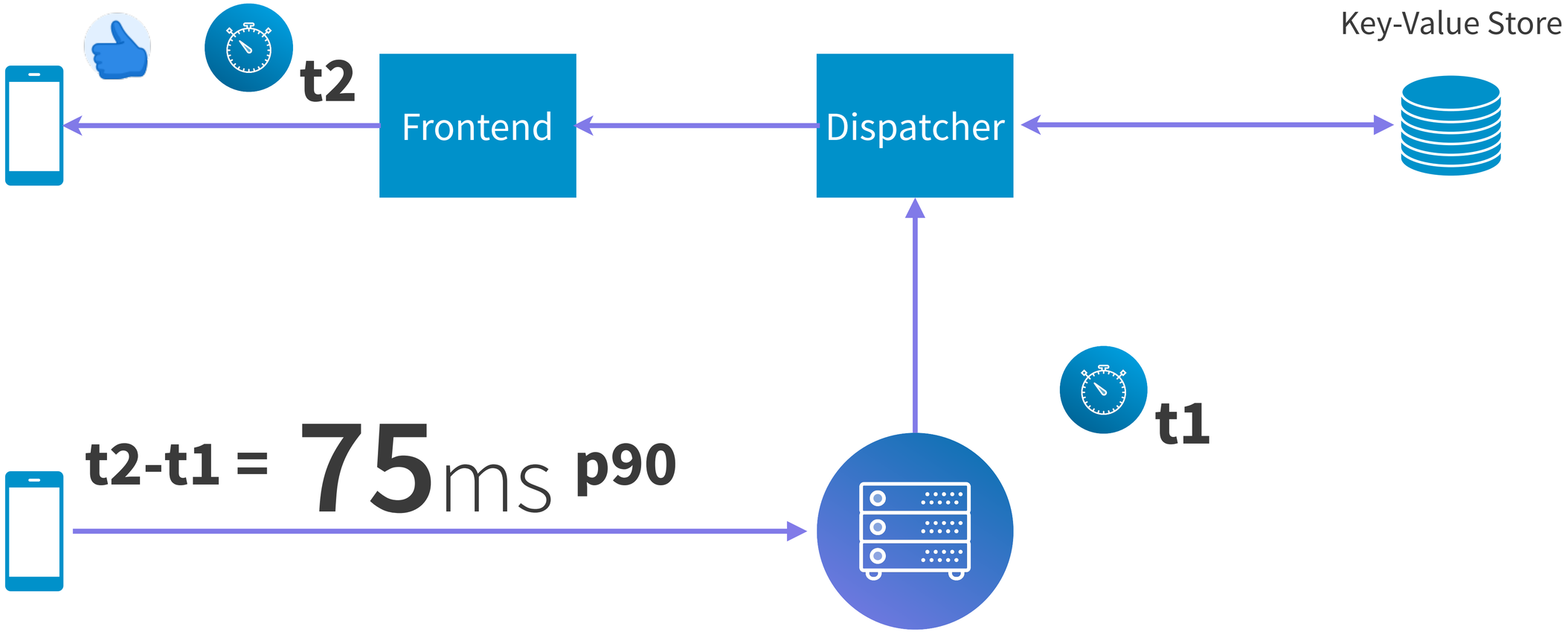
75ms

E2E Publish Latency

E2E Publish Latency



E2E Publish Latency



LinkedIn Realtime Performance

>100K

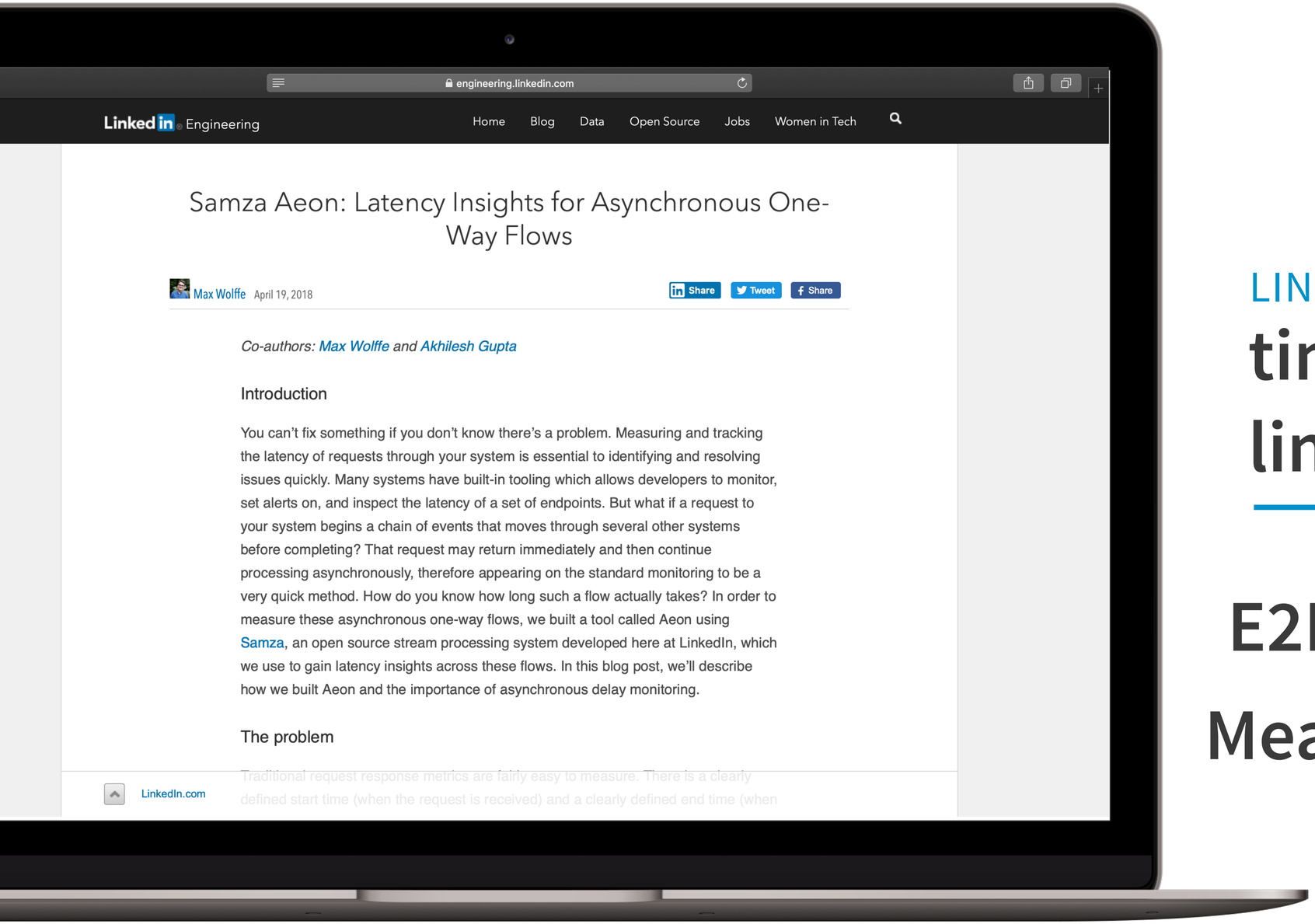
Persistent Connections/machine

5K

Dispatcher Publish QPS/machine

75ms

E2E Publish Latency

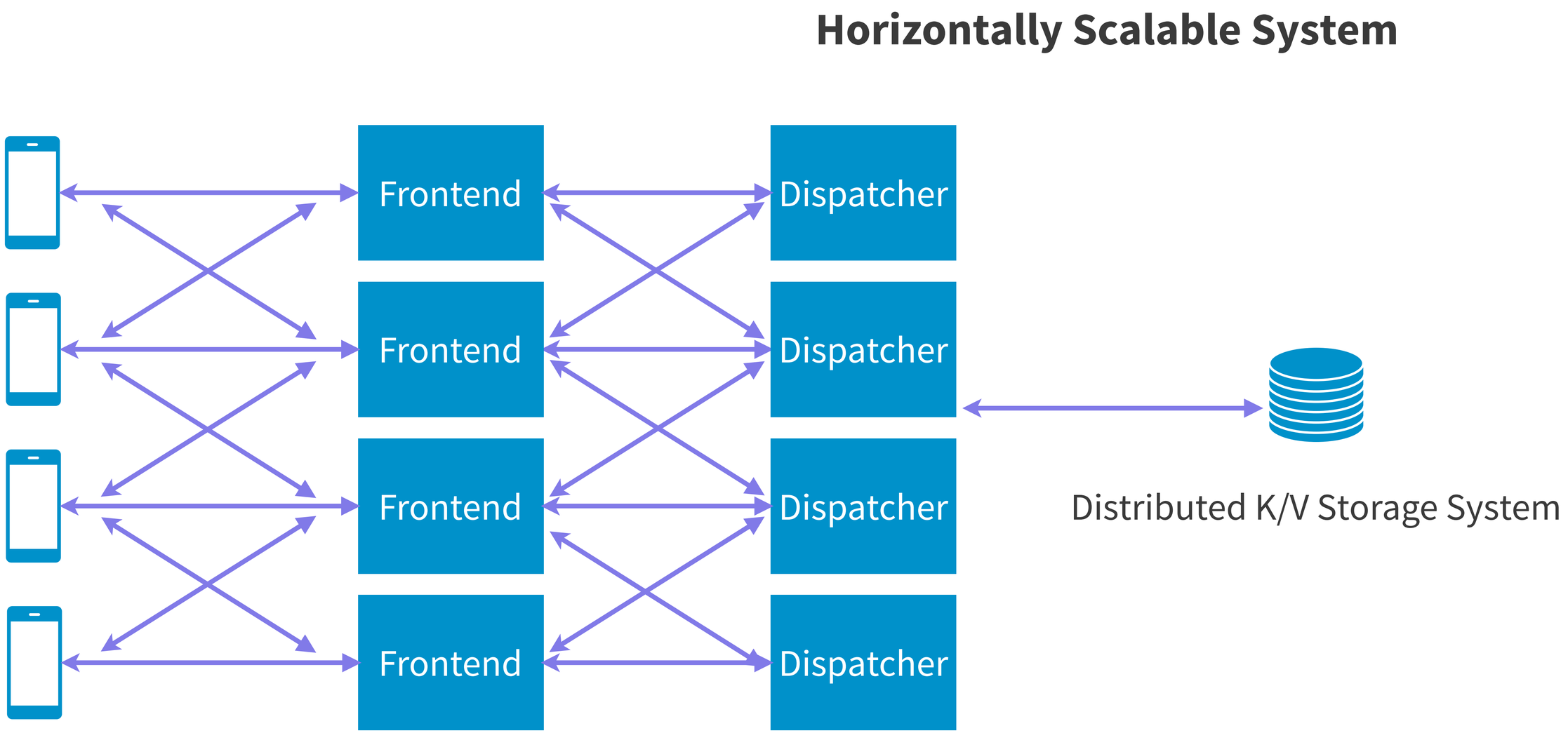


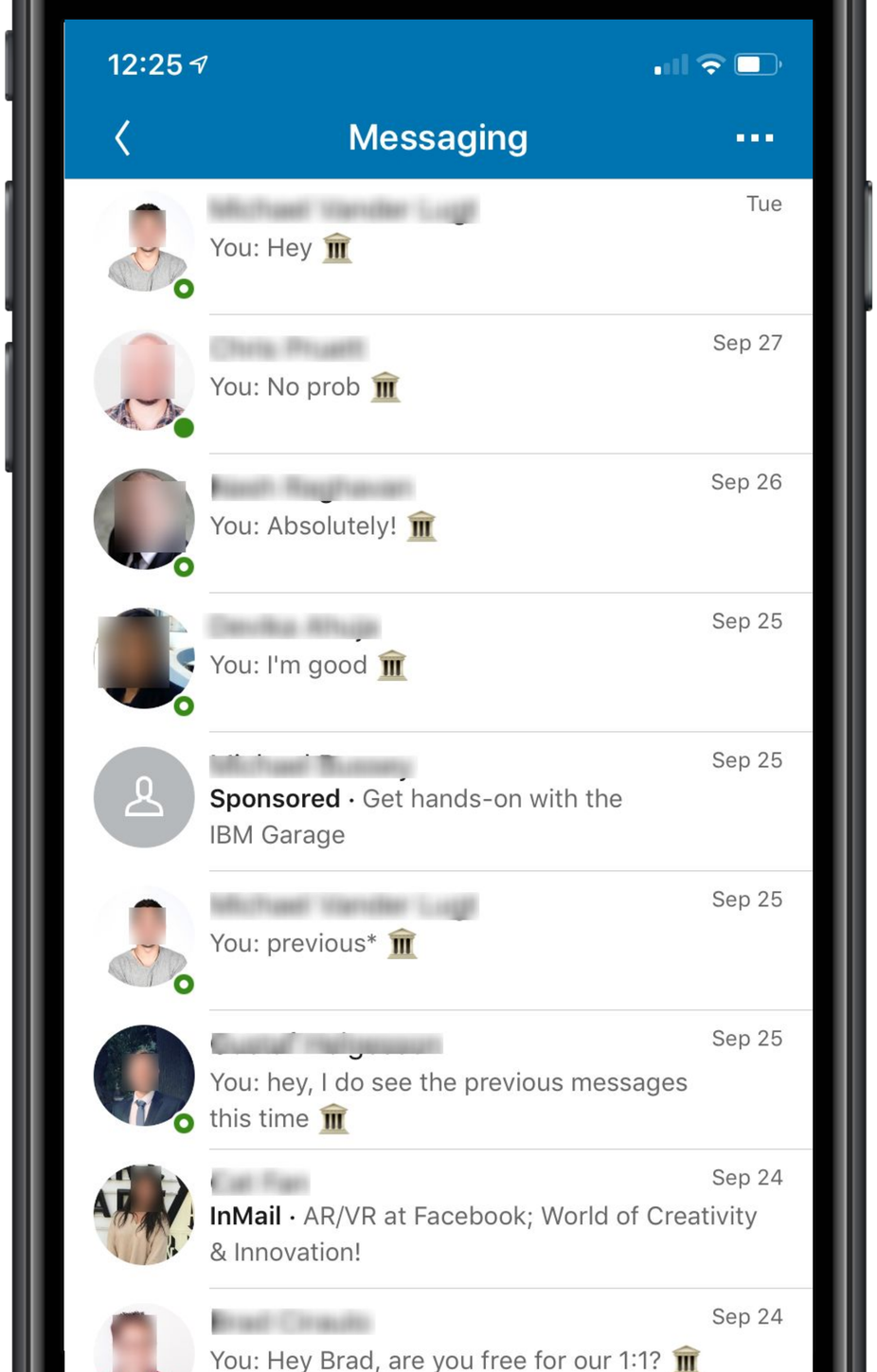
LINKEDIN ENGINEERING BLOG

tiny.cc/linkedinlatency

E2E Publish Latency Measurement System

Why does this system scale?

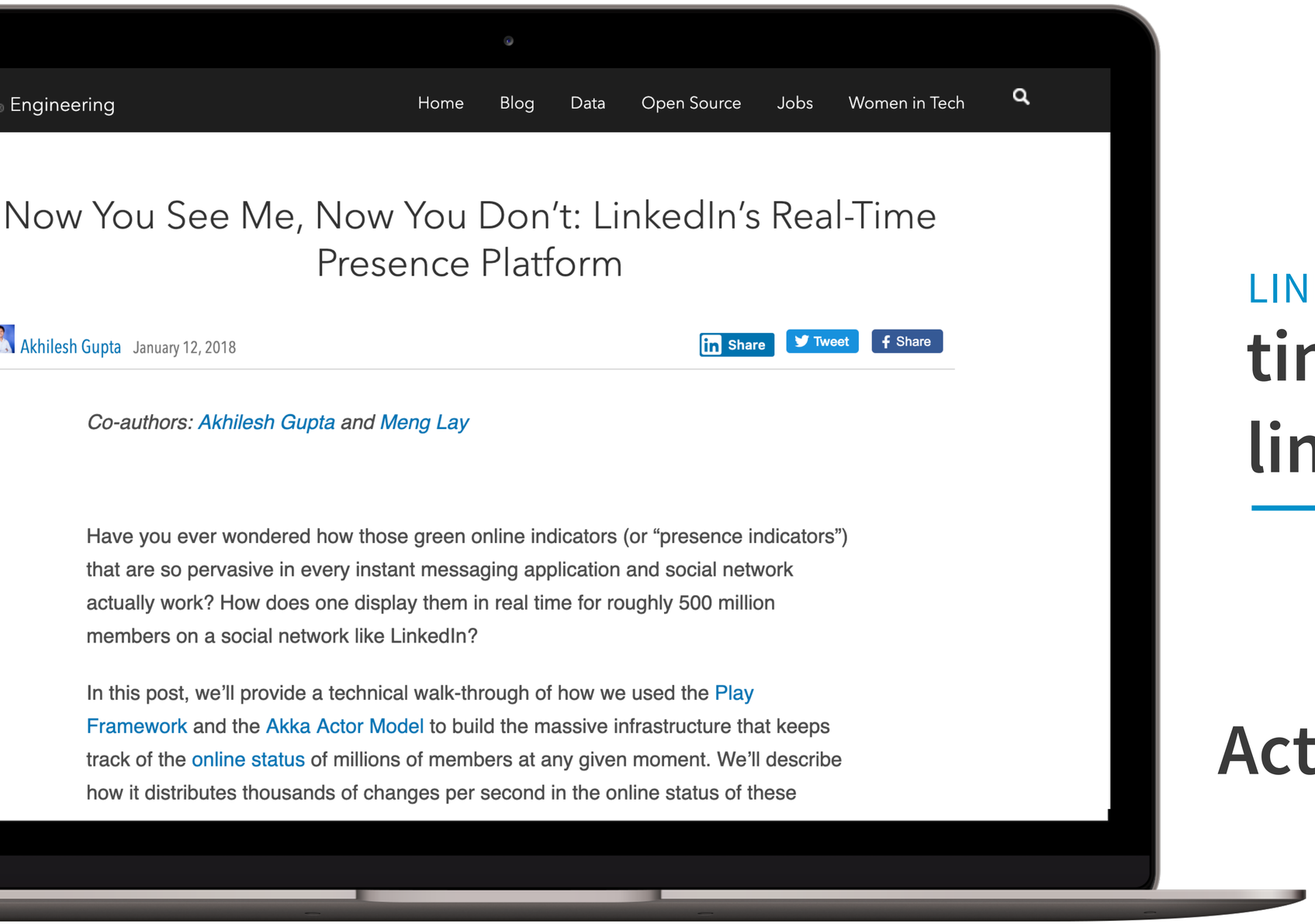




LINKEDIN REALTIME TECHNOLOGY

Active Status/Presence

Online/Offline indicators



LINKEDIN ENGINEERING BLOG

tiny.cc/linkedinpresence

Active Status/Presence

Key takeaways

REALTIME INTERACTIONS

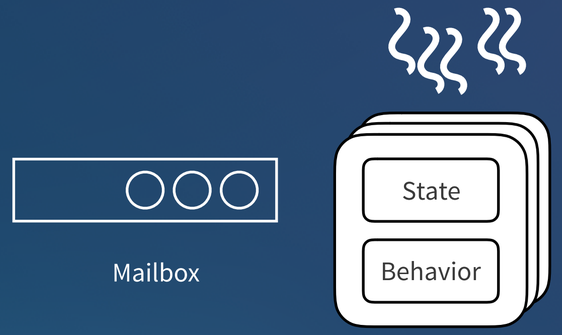
Enable dynamic instant experiences on your apps

```
HTTP/1.1 200 OK
- RESPONSE HEADERS -
Content-Type: text/event-stream
```

EVENTSOURCE/SSE

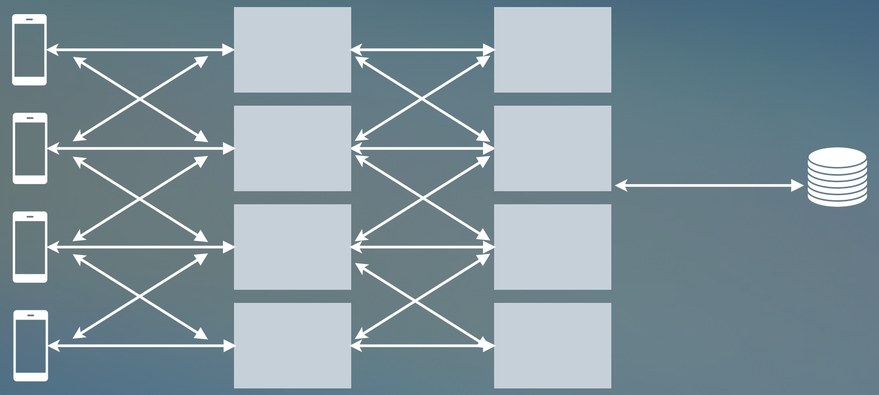
Built-in support in most server frameworks

Readily available client libraries



Play Framework and Akka Actors

Powerful frameworks to manage millions of connections

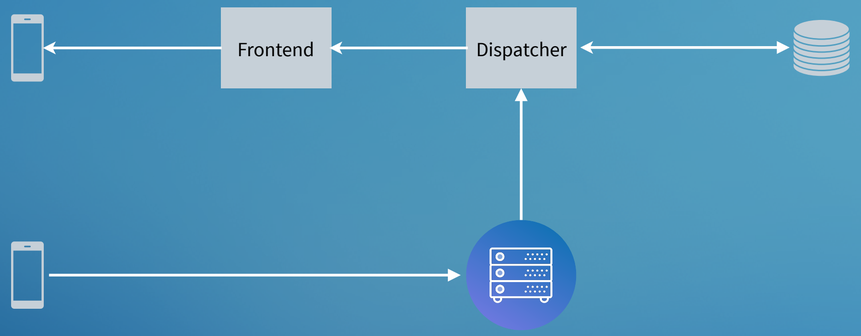


Distributed Systems

Start small and add simple layers to your architecture

SCALING BACKEND SYSTEMS

When in doubt, add a machine!



REALTIME INTERACTION PLATFORM

Can be built on any server/storage technology

Horizontally scalable

REALTIME INTERACTION PLATFORM

You can do it for your app!

REALTIME ENGINEER, LINKEDIN

@agupta03

tiny.cc/qcon2020

AMA at 1.40pm at Guild, Mezzanine