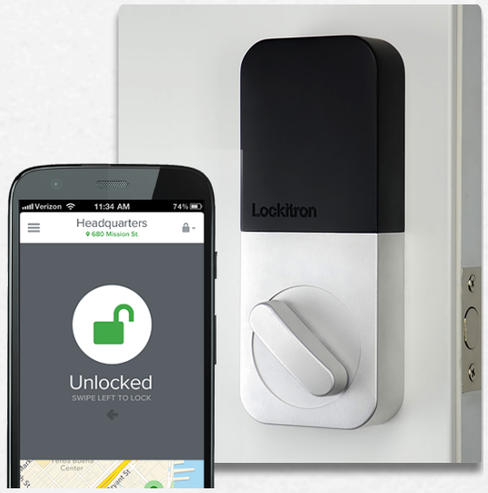
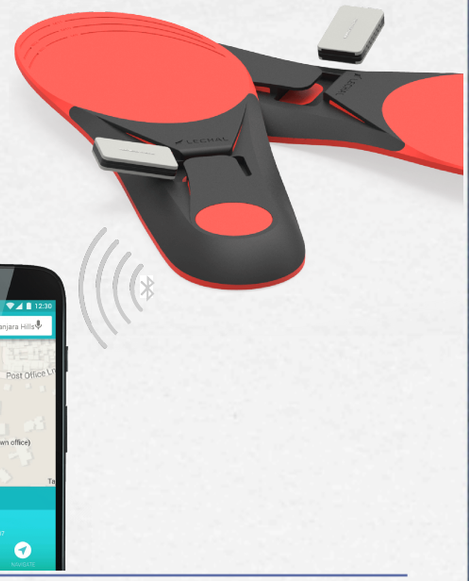
Location-based publish/subscribe

Unil | HEC | dop lab
Benoît Garbinato
distributed object programming lab

The Internet of Objects (IoT) is becoming a reality





Software support | Limitations

- Platform-specific APIs*, i.e., lack of standard
- Distinct APIs to manage the content aspect and the context aspect of communication
- Multiplication of APIs to learn and master
- Difficulty to scale when the number of sensing and connected objects explodes

*Application Programming Interfaces

Software support | Limitations

facing an api jungle when developing

in particular no programming support
that seamlessly combines

communication

+

sensory input

Open Challenges

each connected object can be seen as a moving producer and consumer of contextual information that needs to be tracked

mobile app developers face complex development and deployment issues even for simple context-aware services

development

multiple hardware, operating systems, protocols, etc.

⬇

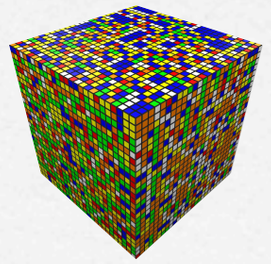
the api jungle challenge

deployment

massive tracking, messaging, testing, monitoring, etc.

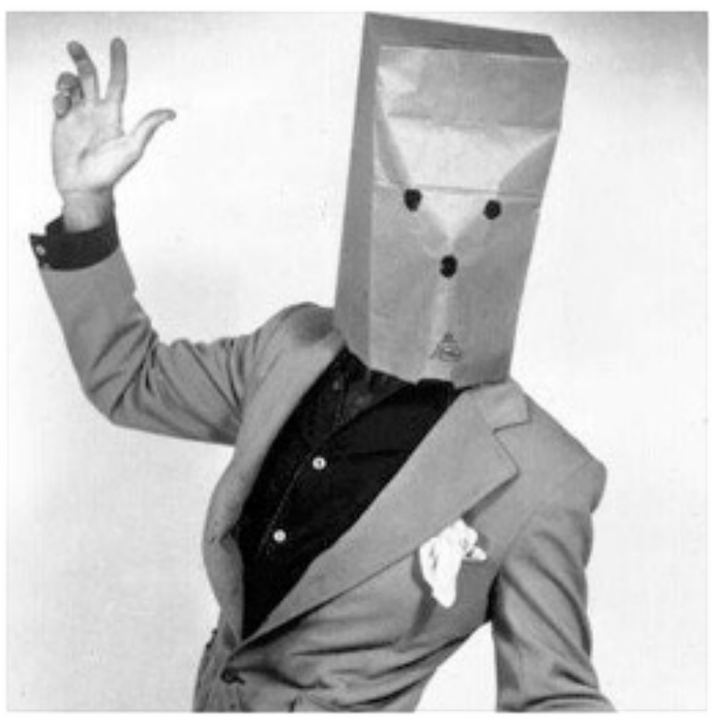
⬇

the scalability challenge



Publish/Subscribe

as starting point

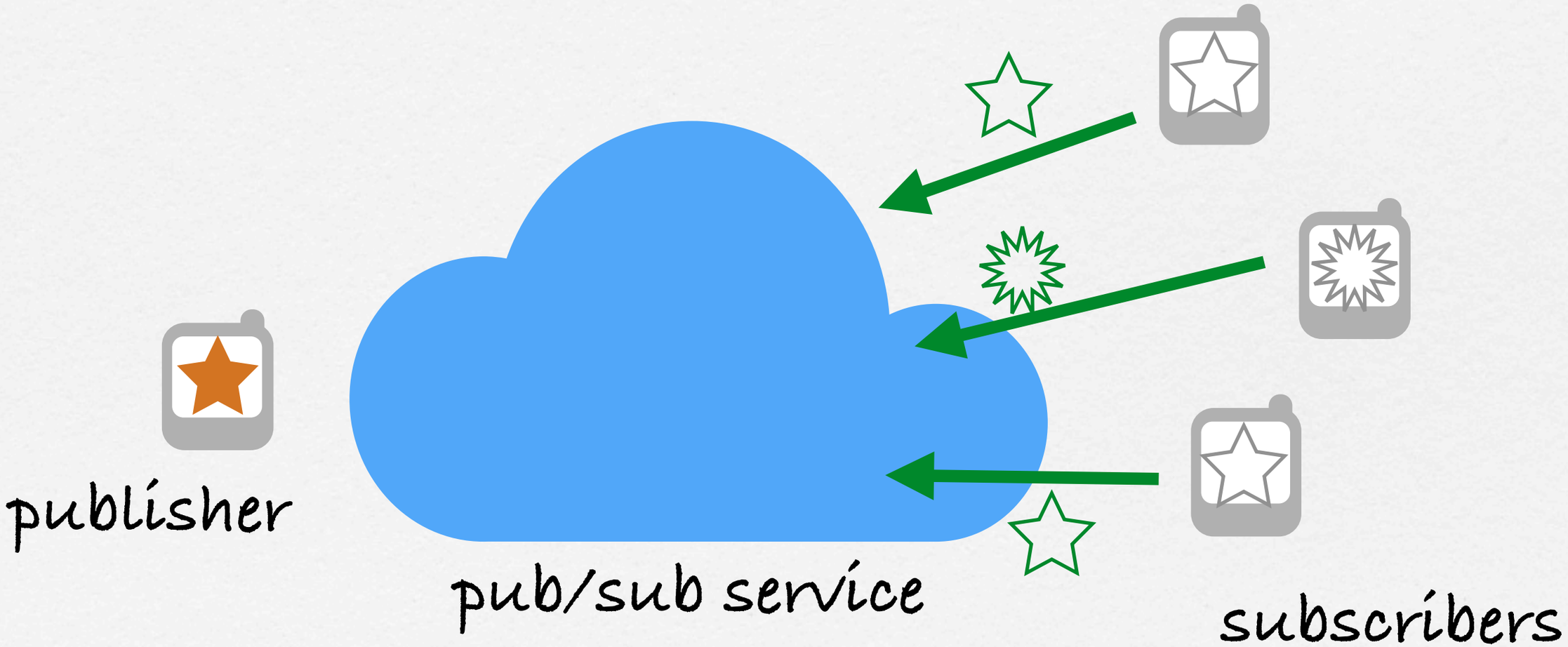


anonymous

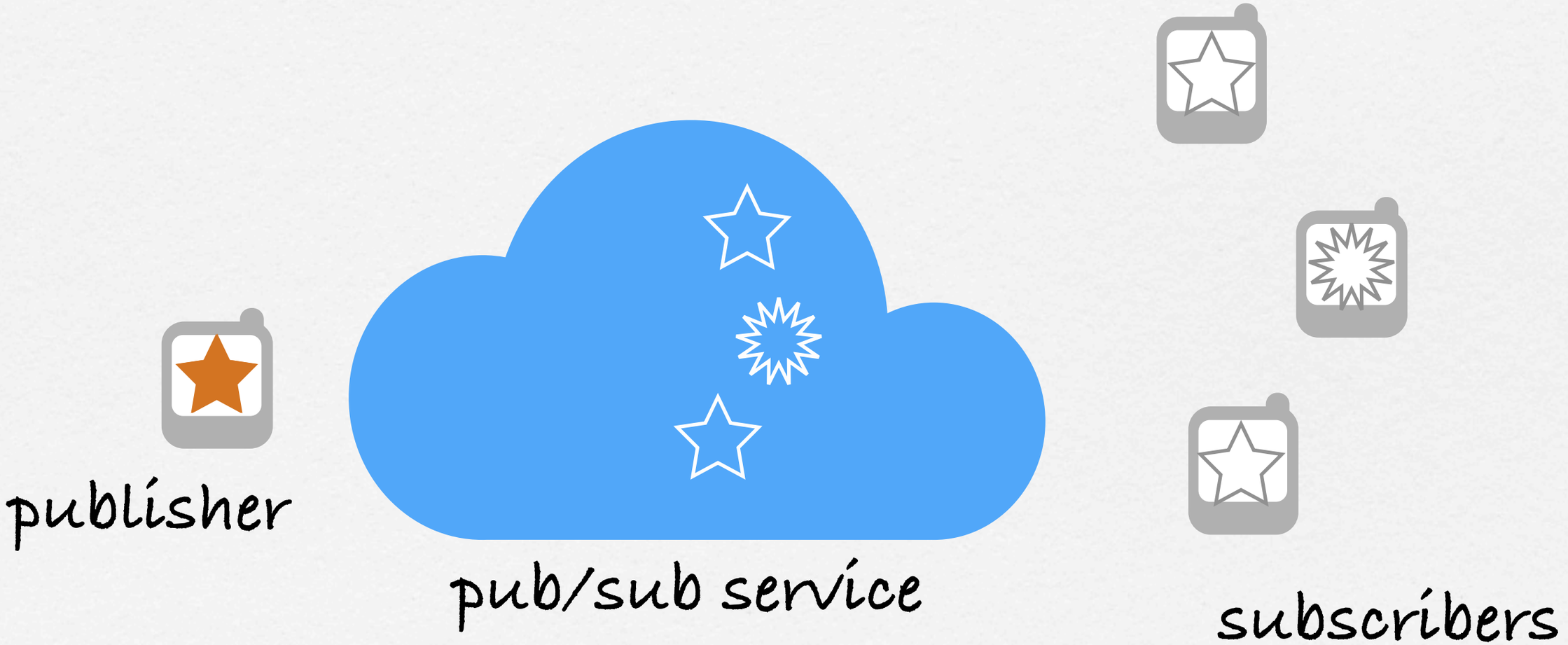
asynchronous

Publish/Subscribe
as starting point

Publish/Subscribe
subscriptions are created

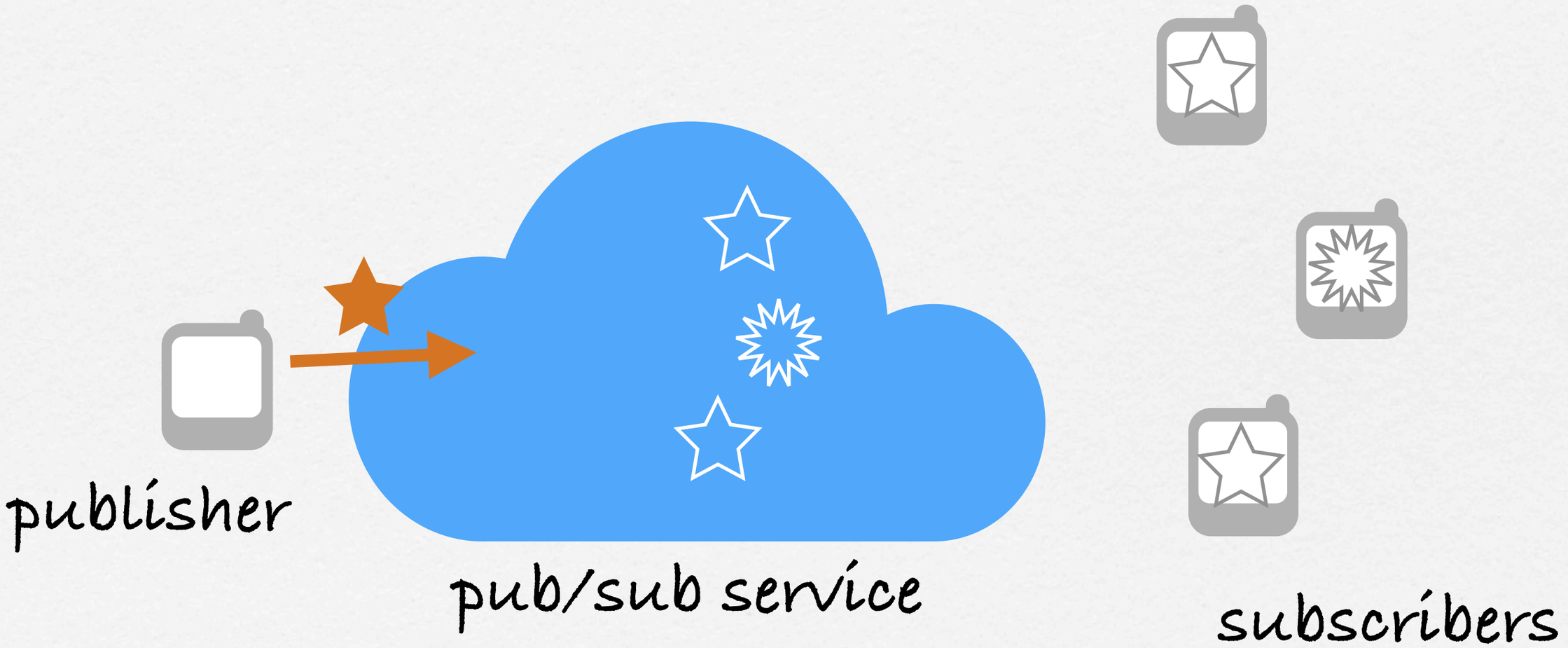


Publish/Subscribe
subscriptions are created



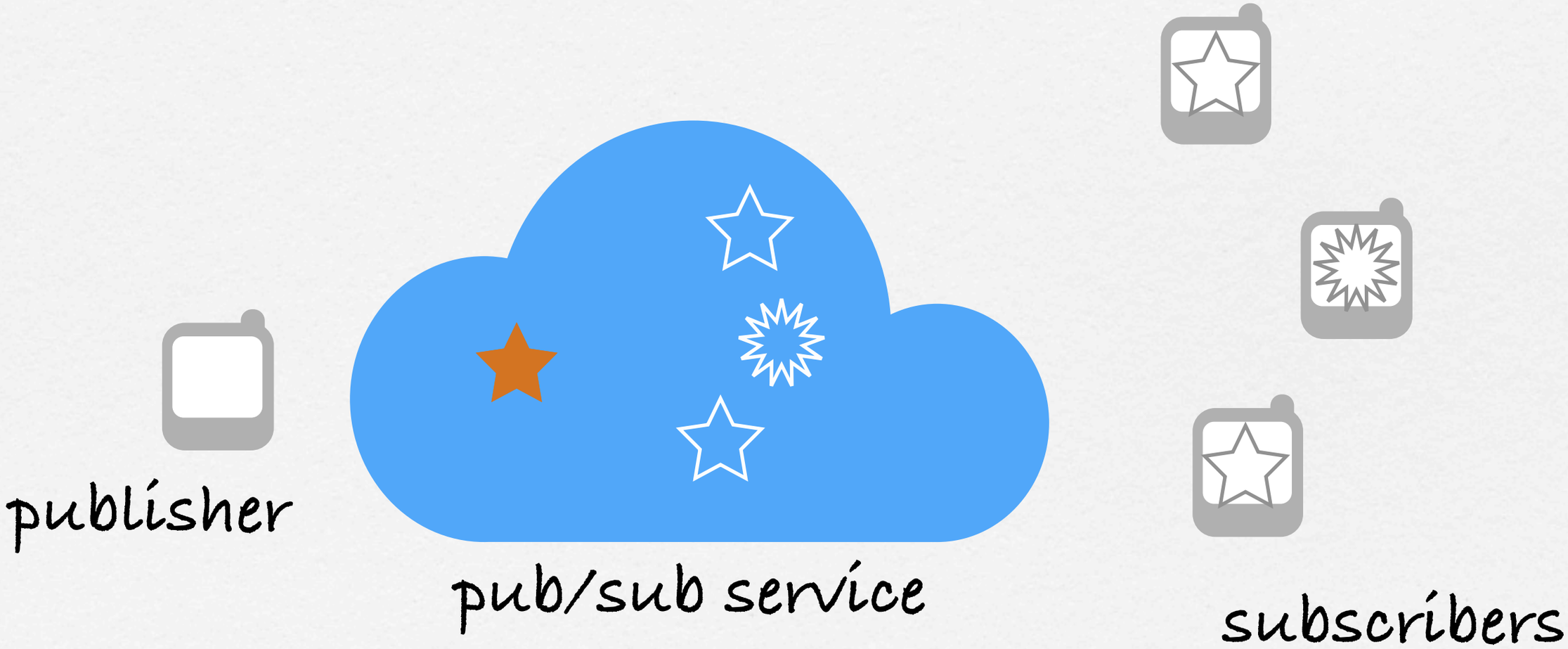
Publish/Subscribe

message is published

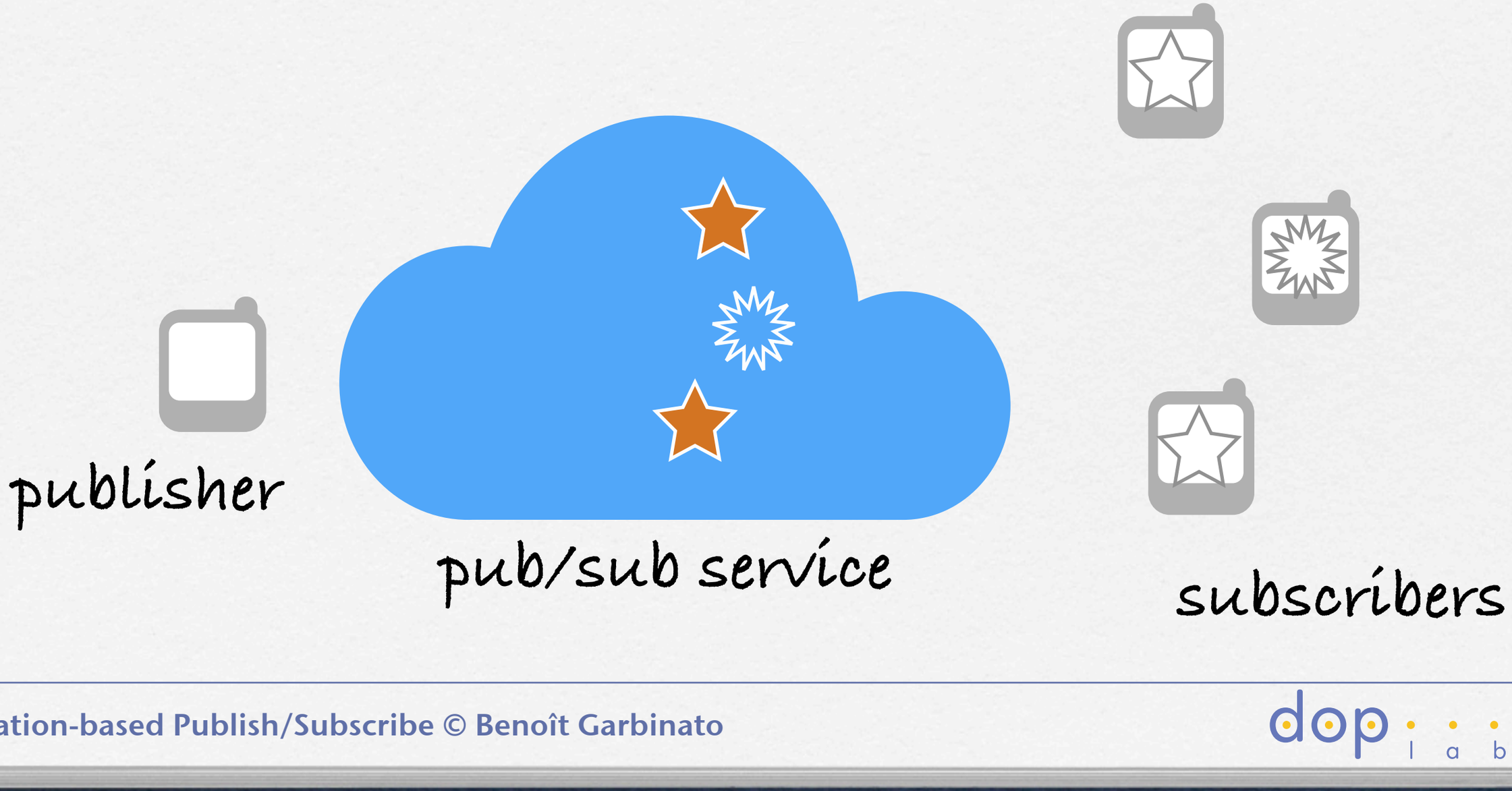


Publish/Subscribe

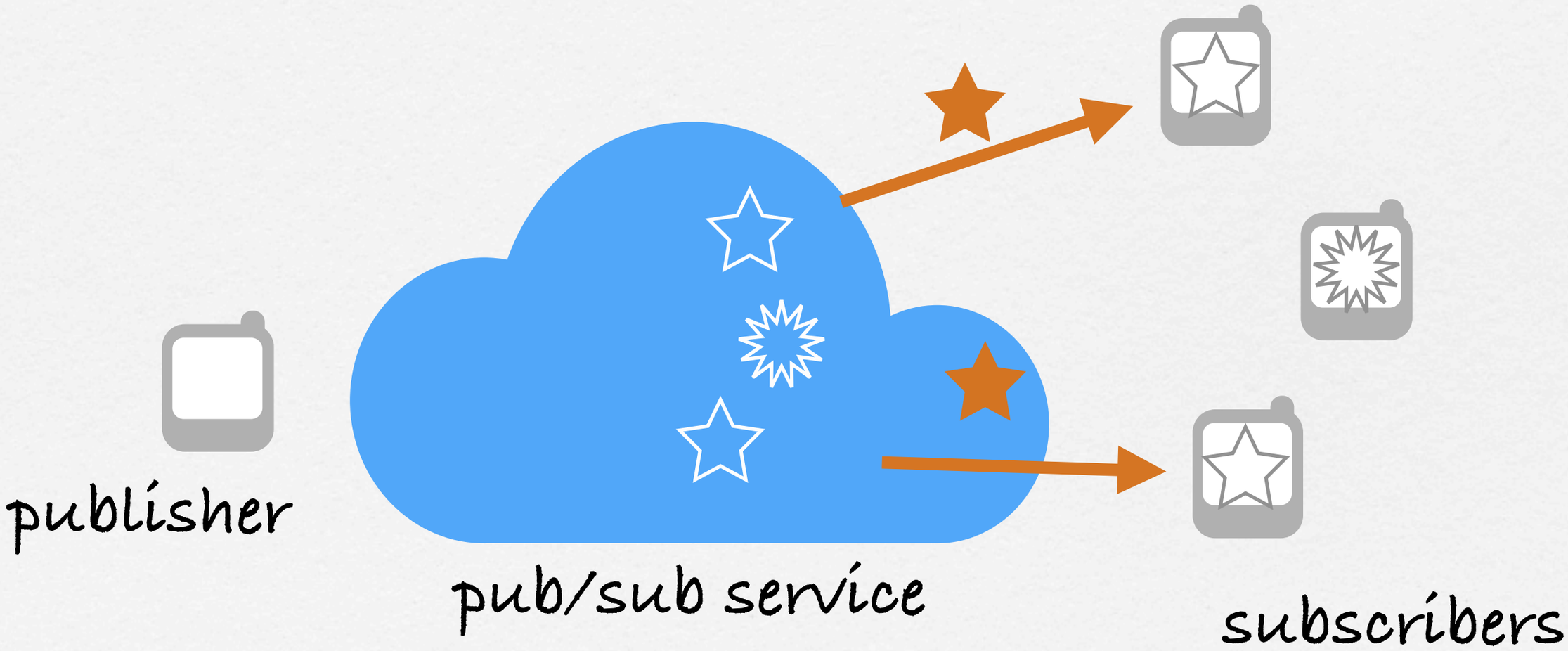
message is published



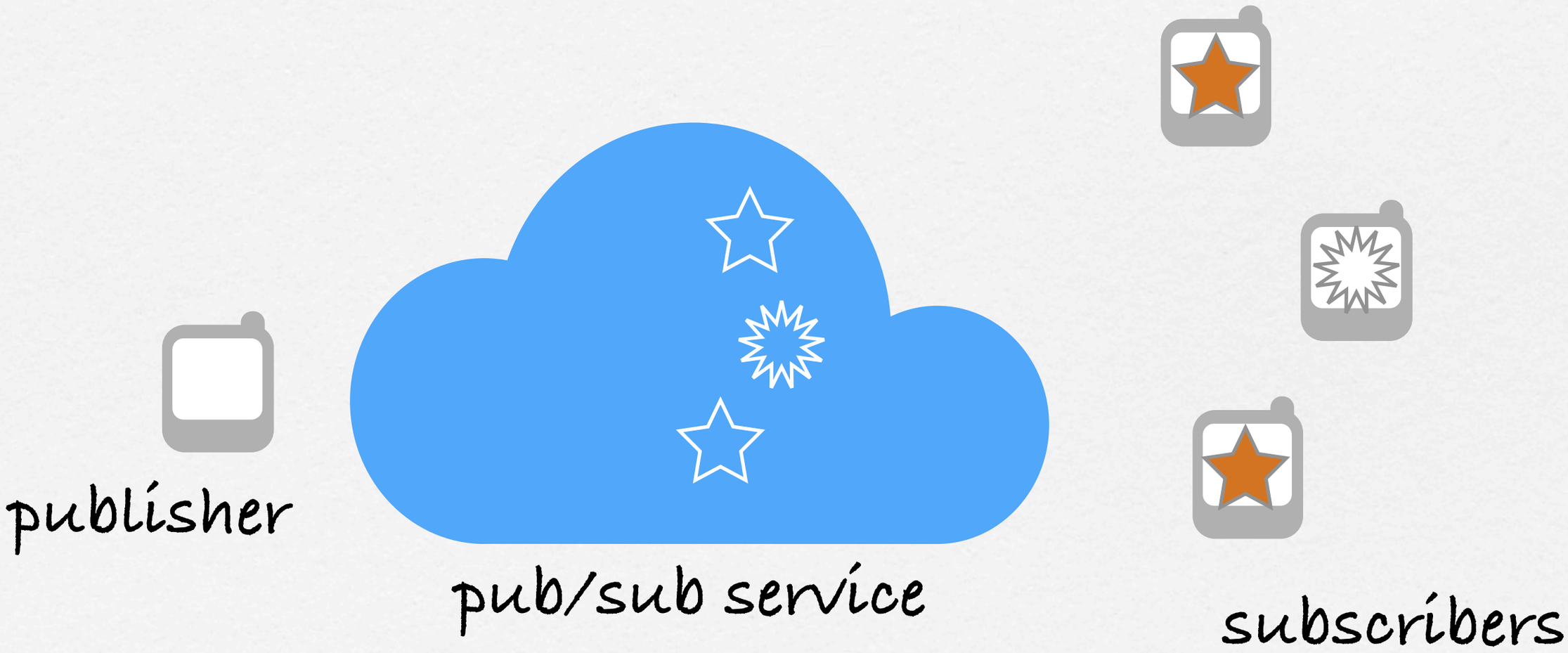
Publish/Subscribe
content match occurs



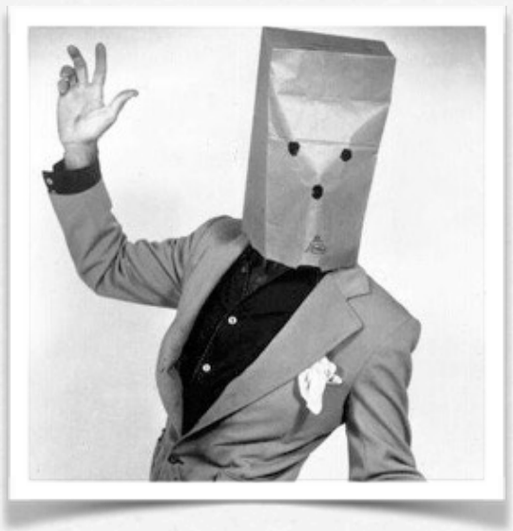
Publish/Subscribe
message is delivered



Publish/Subscribe
message is delivered



Publish/Subscribe

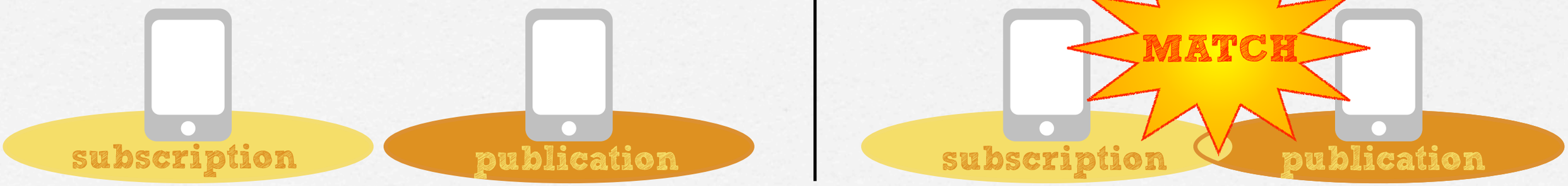


anonymous

+

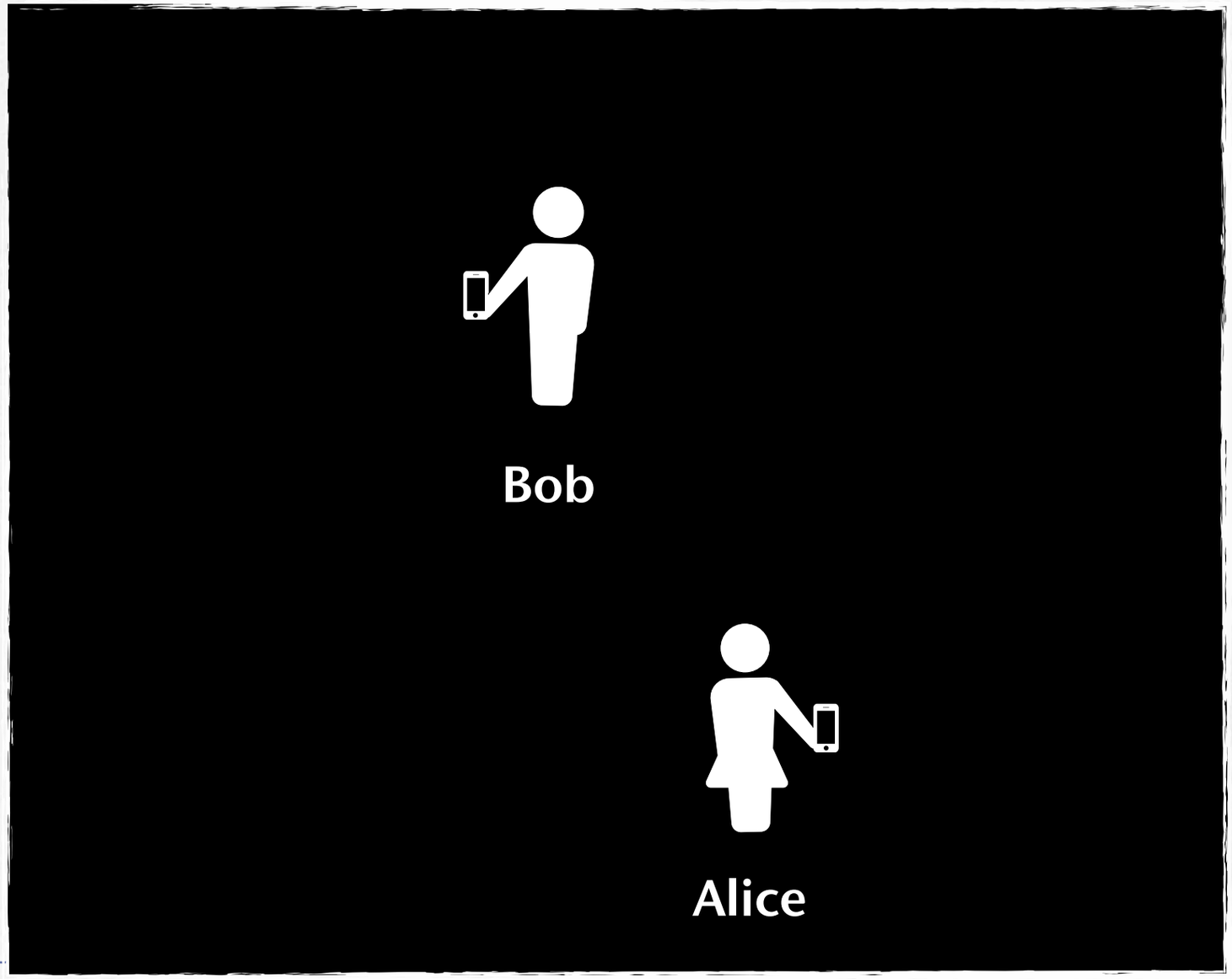
asynchronous

Location Awareness

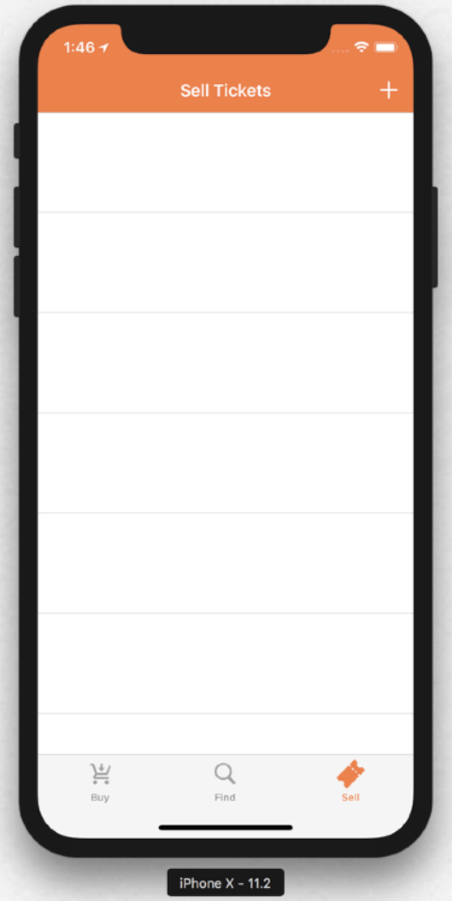


Location-based Publish/Subscribe
The iTicket app example

Bob



Alice

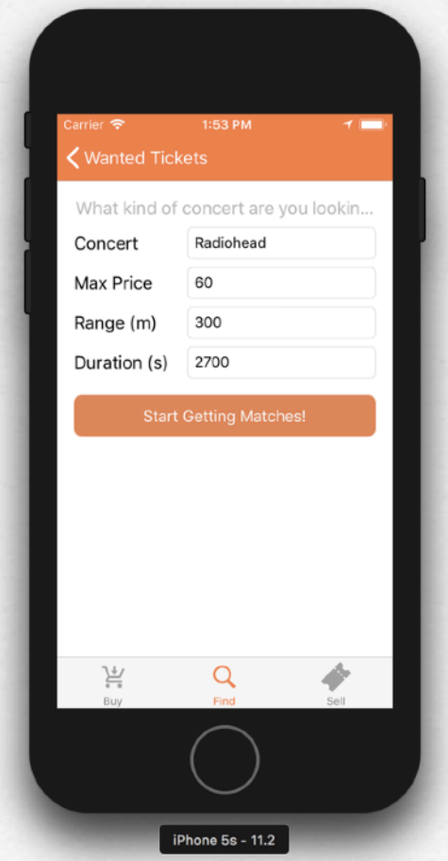


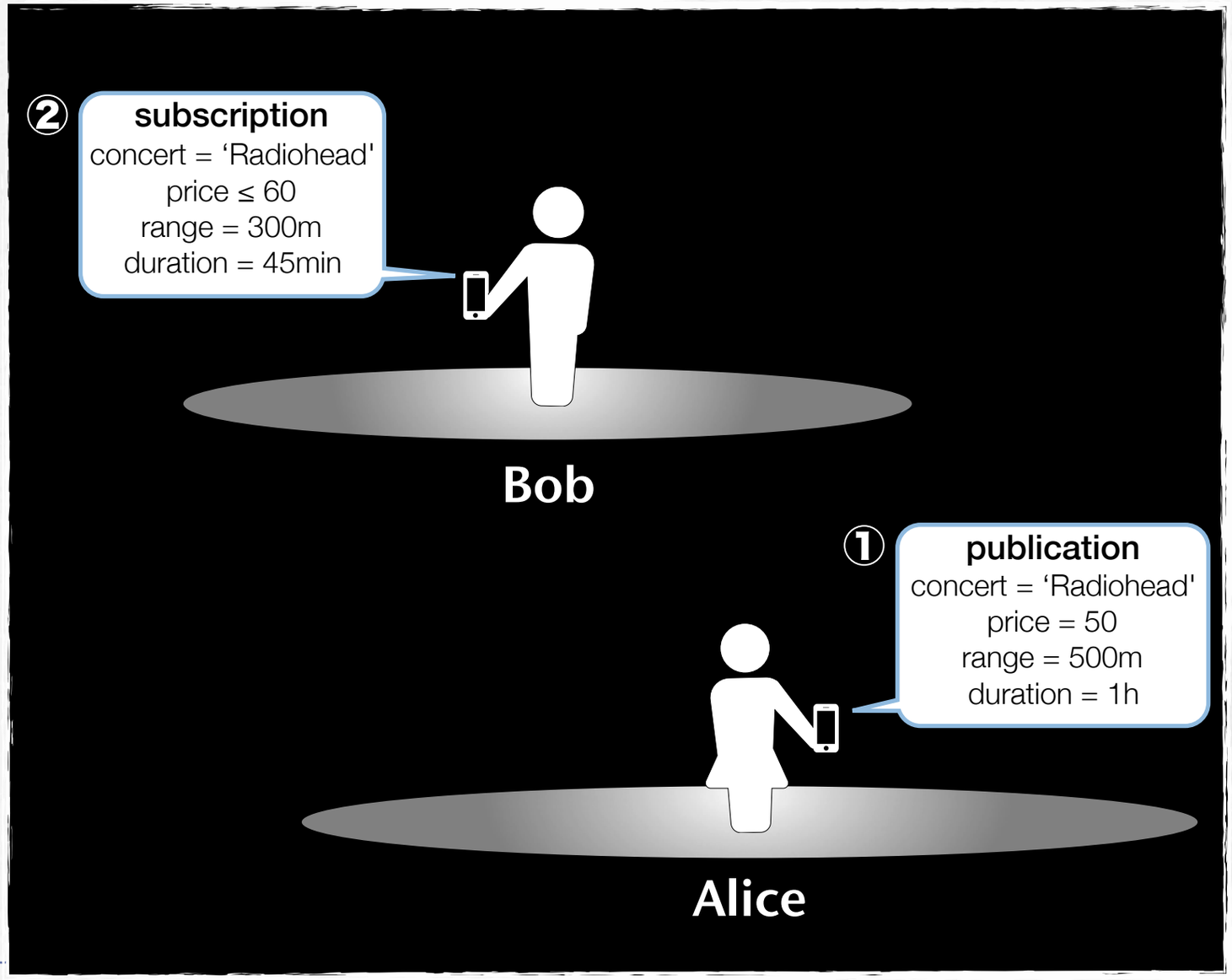
dop lab

Location-based Publish/Subscribe

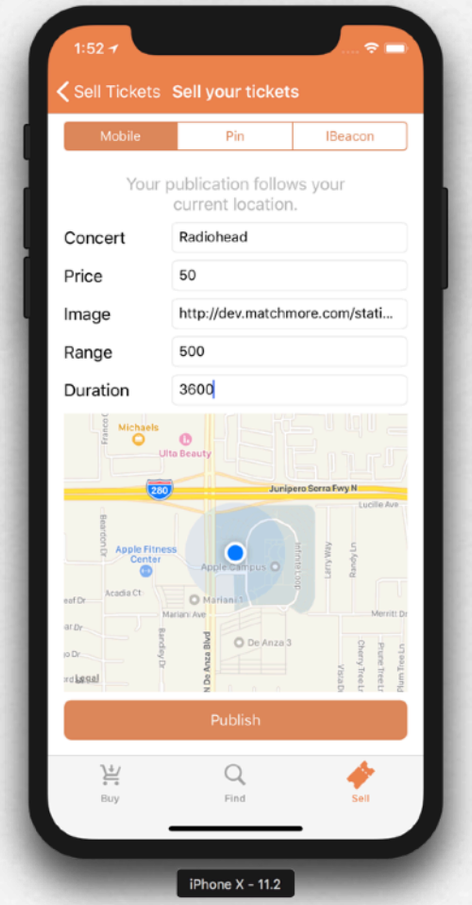
The iTicket app example

Bob





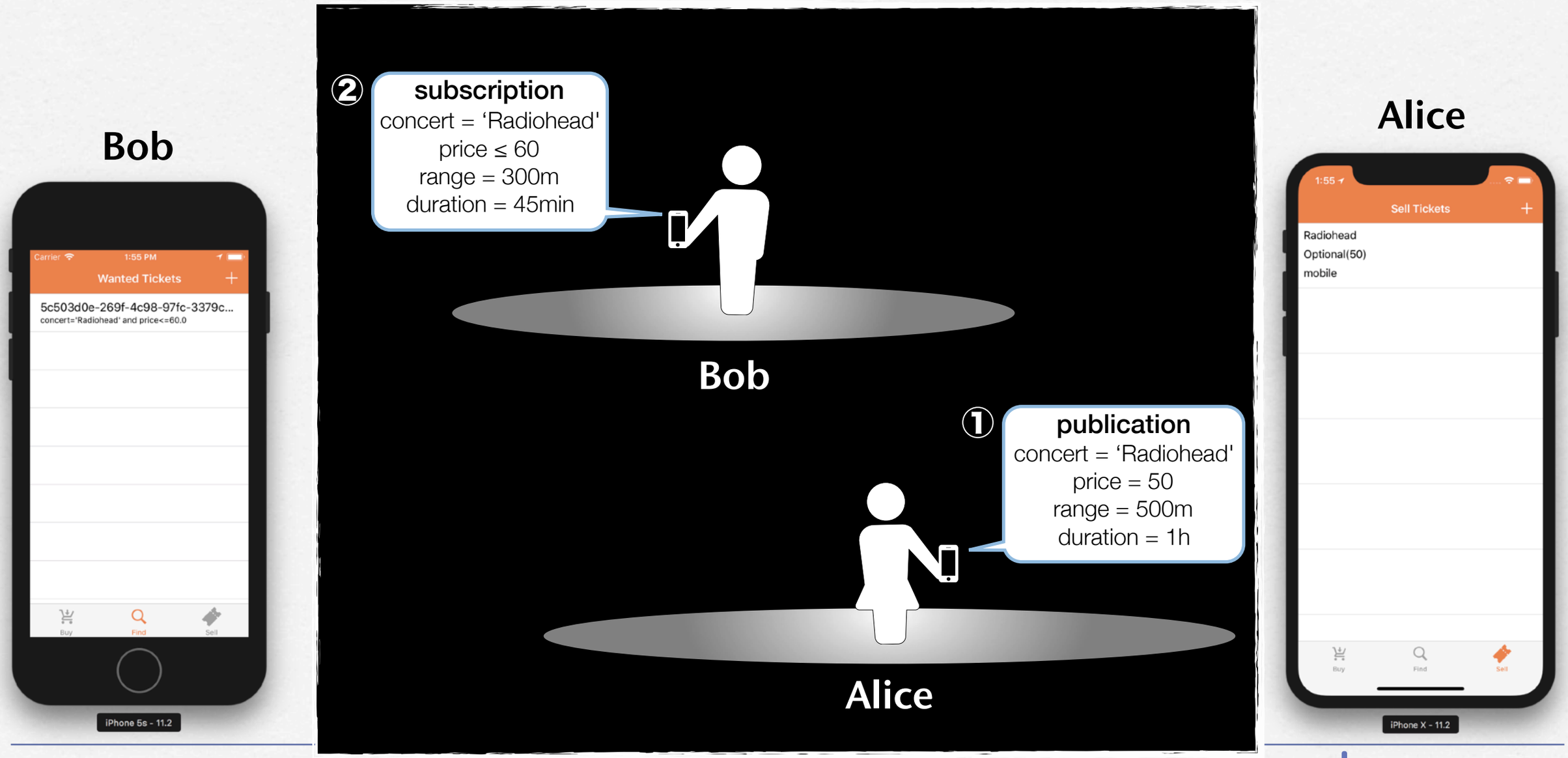
Alice



dop lab

Location-based Publish/Subscribe

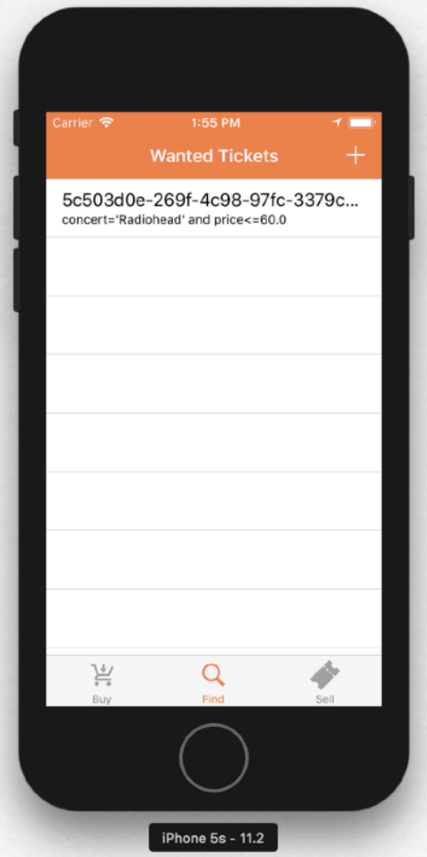
The iTicket app example

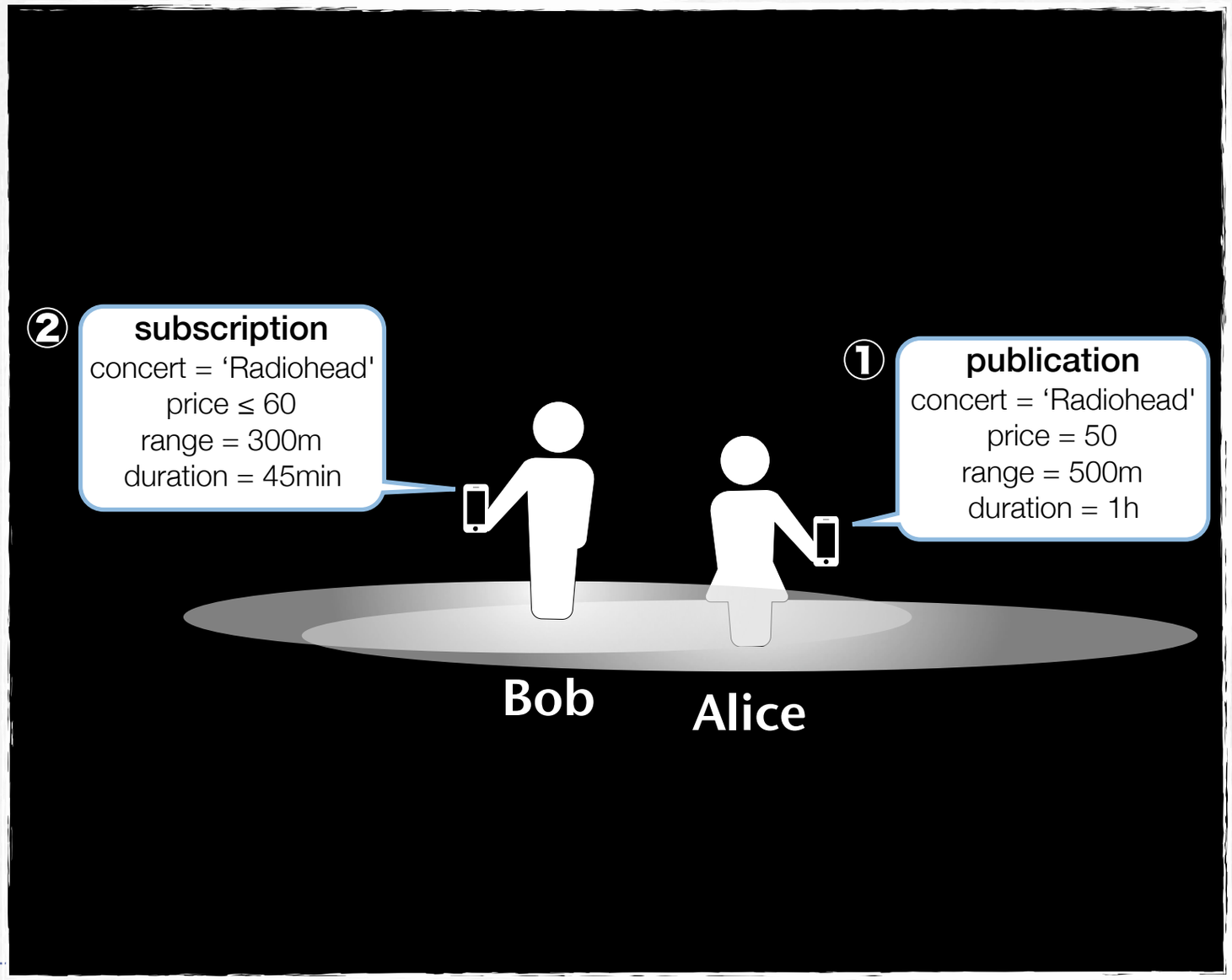


dop lab

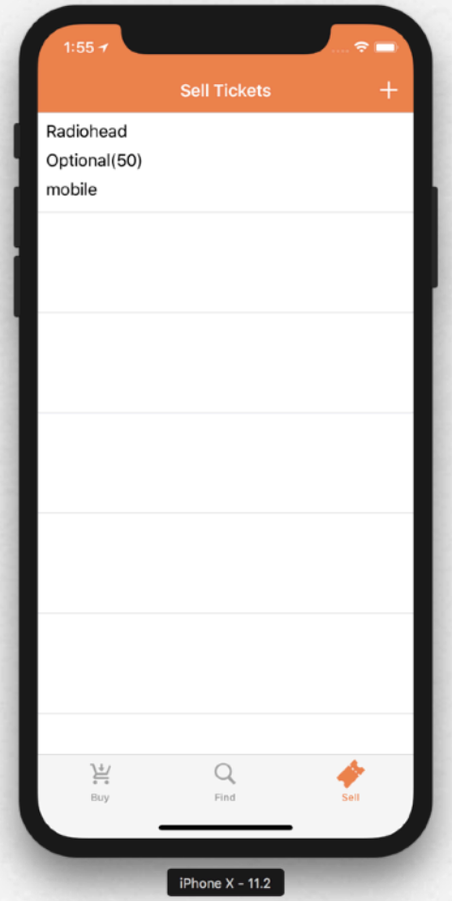
Location-based Publish/Subscribe
The iTicket app example

Bob





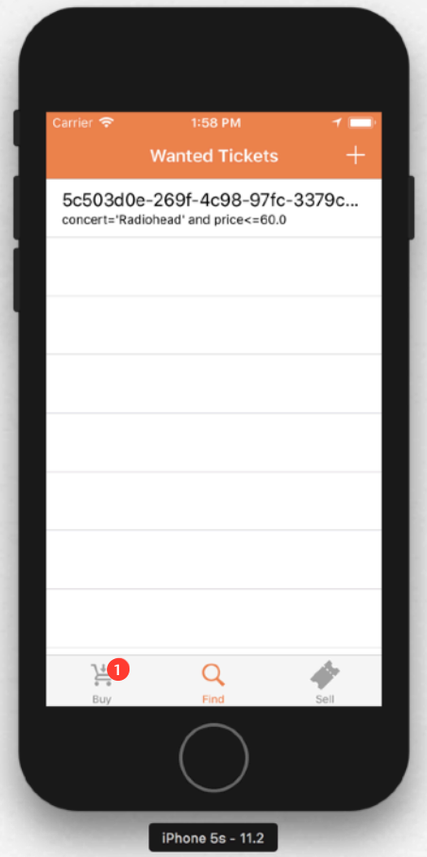
Alice

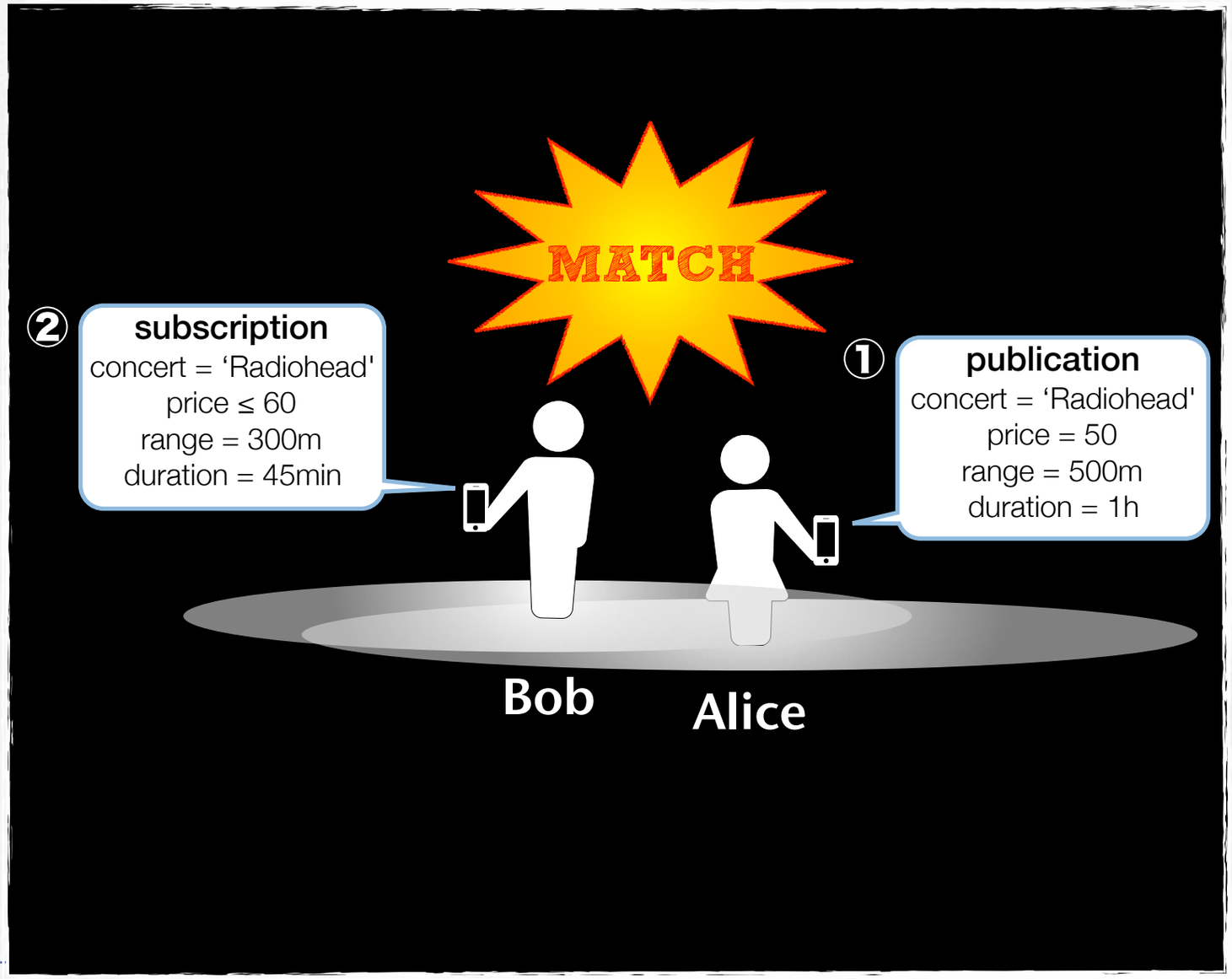


dop lab

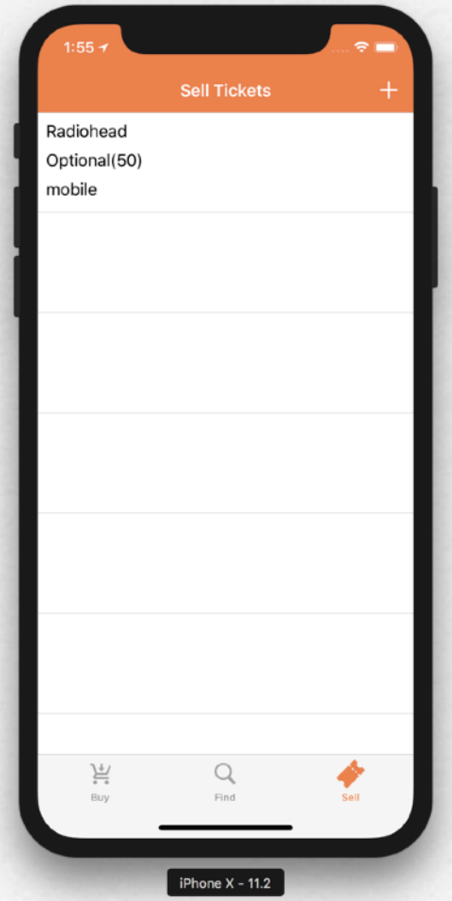
Location-based Publish/Subscribe
The iTicket app example

Bob



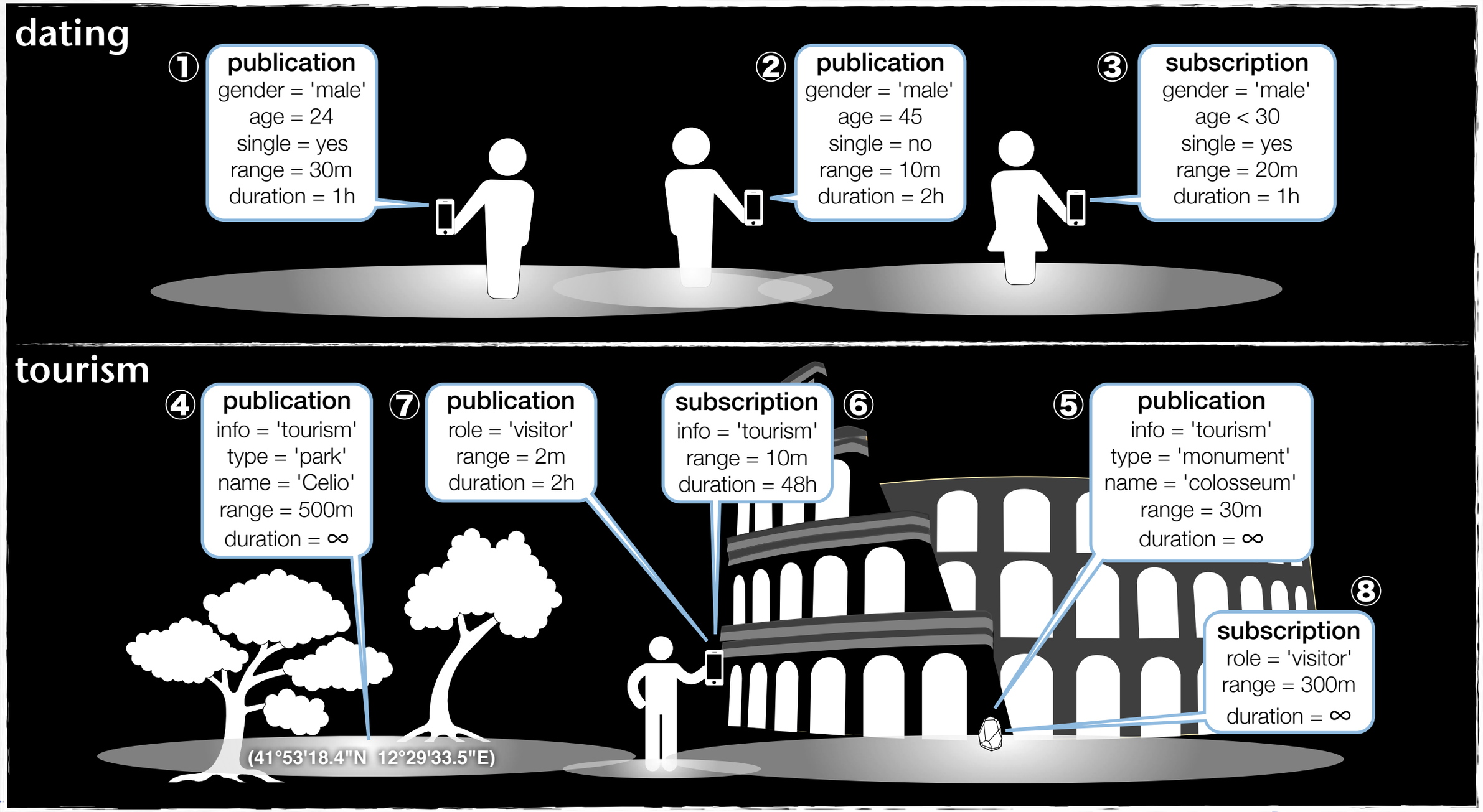


Alice



dop lab

Location-based Pub/Sub | Other Examples



Location-based Publish/Subscribe

with matchmore BETA

Location-based Publish/Subscribe with matchmore BETA

iOS +

```swift
import UIKit
import Alps
import AlpsSDK

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    var matchDelegate: MatchDelegate! = nil
    var alpsManager: AlpsManager! = nil

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool
        {

        MatchMore.apiKey =
            "eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiJ9.eyJpc3MiOiJhbHBzIiwic3ViIjoiYzM5MzFhNDgtYmQ4Mi00NDVmLWI2NTYtMTEyN2ZkY2FiYjBlIiwiYXVkIjpbIlB1Y
            mxpYyJdLCJuYmYiOjE1MTExODMxOTgsImlhdCI6MTUxMTE4MzE5OCwianRpIjoiMSJ9.ZvZ-
            cWwlUJv_dPpn1pSUoHoRT-7yoH4HjFqofnaDxMk5ZSwh0v9yn2HmnxejixinApGr-P-PAXcbisFuREVgPA"
        MatchMore.worldId = "c3931a48-bd82-445f-b656-1127fdcabb0e"

        MatchMore.startUsingMainDevice { result in
            guard case .success(let mainDevice) = result else { print(result.errorMessage ?? ""); return }
            print("⛰ Using device: ⛰\n\(mainDevice.encodeToJSON())")

            // Start Monitoring Matches
            self.matchDelegate = MatchDelegate { matches, _ in
                print("⛰ You've got a new match!!! ⛰\n\(matches.map { $0.encodeToJSON() })")
            }
            MatchMore.matchDelegates += self.matchDelegate

```

Location-based Publish/Subscribe with matchmore BETA

iOS + Swift

```
35
36          // Create New Publication
37          MatchMore.createPublication(publication: Publication(topic: "Test Topic", range: 20, duration: 100, properties: ["test":"true"]),
                completion: { result in
38              switch result {
39              case .success(let publication):
40                  print("🏔 Pub was created: 🏔\n\(publication.encodeToJSON())")
41              case .failure(let error):
42                  print("🌋 \(String(describing: error?.message))")
43              }
44          })
45
46          // Polling
47          //MatchMore.startPollingMatches()
48          //self.createPollingSubscription()
49
50          // Socket (requires world_id)
51           MatchMore.startListeningForNewMatches()
52           self.createSocketSubscription()
53
54          // APNS (Subscriptions is being created after receiving device token)
55          // PermissionsHelper.registerForPushNotifications()
56
57          MatchMore.startUpdatingLocation()
58      }
59      return true
60  }
```

Location-based Publish/Subscribe with matchmore BETA

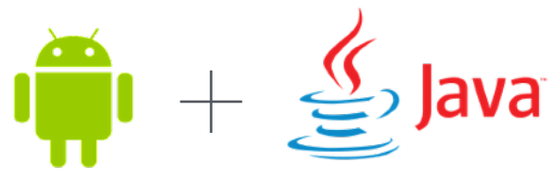
iOS +

```swift
// Subscriptions

func createSocketSubscription() {
    let subscription = Subscription(topic: "Test Topic", range: 20, duration: 100, selector: "test = true")
    subscription.pushers = ["ws"]
    MatchMore.createSubscription(subscription: subscription, completion: { result in
        switch result {
        case .success(let sub):
            print("🏔 Socket Sub was created 🏔\n\(sub.encodeToJSON())")
        case .failure(let error):
            print("🌋 \(String(describing: error?.message))")
        }
    })
}

func createPollingSubscription() {
    let subscription = Subscription(topic: "Test Topic", range: 20, duration: 100, selector: "test = true")
    MatchMore.createSubscription(subscription: subscription, completion: { result in
        switch result {
        case .success(let sub):
            print("🏔 Polling Sub was created 🏔\n\(sub.encodeToJSON())")
        case .failure(let error):
            print("🌋 \(String(describing: error?.message))")
        }
    })
}
```

Location-based Publish/Subscribe with matchmore BETA



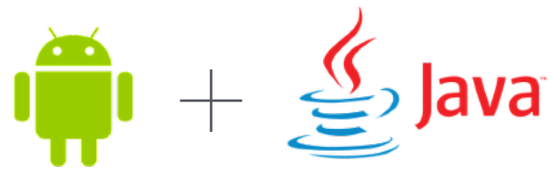
```
// Configuration of api key/world id
MatchMore.config(new MatchMoreConfig(
                    RuntimeEnvironment.application,

                    "eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiJ9.eyJpc3MiOiJhbHBzIiwic3ViIjoiYzM5MzFhNDgtYmQ4Mi00NDVmLWI2NTYtMTEy
                    N2ZkY2FiYjBlIiwiYXVkIjpbIlB1YmxpYyJdLCJuYmYiOjE1MTExODMxOTgsImlhdCI6MTUxMTE4MzE5OCwianRpIjoiMSJ9.ZvZ-
                    cWwlUJv_dPpn1pSUoHoRT-7yoH4HjFqofnaDxMk5ZSwh0v9yn2HmnxejixinApGr-P-PAXcbisFuREVgPA", // api key
                    "c3931a48-bd82-445f-b656-1127fdcabb0e", // world id
                    null, // custom server protocol
                    null, // custom server url
                    true, // callback in UI Thread
                    true) // debug log
);


// Getting instance. It's static variable. It's possible to have only one instance of matchmore.
MatchMoreSdk matchMore = MatchMore.getInstance();

// Creating main device.
matchMore.startUsingMainDevice(device -> {
            Log.d("Examples", device.getId());
            return Unit.INSTANCE; // this is important (b/c kotlin vs java callbacks differ in implementation)
        }, e -> {
            Log.d("Examples", e.getMessage());
            return Unit.INSTANCE;
        });
```

Location-based Publish/Subscribe with matchmore BETA




```java
// Creating publication
Publication publication = new Publication("Test Topic", 20d, 100d);
matchMore.createPublication(publication,
        createdPublication -> {
            Log.d("Examples", publication.getId());
            return Unit.INSTANCE;
        }, e -> {
            Log.d("Examples", e.getMessage());
            return Unit.INSTANCE;
        });

// Creating subscription
Subscription subscription = new Subscription("Test Topic", 20d, 100d, "");
matchMore.createSubscription(subscription,
        createdSubscription -> {
            Log.d("Examples", subscription.getId());
            return Unit.INSTANCE;
        }, e -> {
            Log.d("Examples", e.getMessage());
            return Unit.INSTANCE;
        });
```

Location-based Publish/Subscribe with matchmore BETA

```
49  // Getting Matches
50  matchMore.getMatchMonitor().addOnMatchListener(
51                  (matches, device) -> {
52                      Log.d("Examples", device.getId());
53                      return Unit.INSTANCE;
54          });
55
56  // Deleting all devices
57  matchMore.getDevices().deleteAll(() -> {
58              Log.d("Examples", "All devices were deleted");
59              return Unit.INSTANCE;
60          }, e -> {
61              Log.d("Examples", e.getMessage());
62              return Unit.INSTANCE;
63          });
```

Visit dev.matchmore.com to learn more!