Intro

A publisher produces an event, and notifies its subscribers. It also includes extra information with the notification, to specify the type of the notification, and its own ID (so the subscriber knows where the notification has come from).

Subscribers tell publishers they want to subscribe to one or more types of event from that publisher.

Implementation is not that easy – for example, how can a client learn about publishers? Also, what if there are a lot of subscribers? It can then take a long time to notify them all.

Event Types & Attributes

Events come in different types, and also have attributes (e.g. the id of the object that generated the event).

When subscribing to an event, the type can be specified, and criteria for the attribute values can also be specified.

Scalability

Proxy nodes can be used to create scalable notification trees. Now the publisher informs the proxy, and the proxy can handler notification of the subscribers. If you duplicate the proxies, you can also perform load balancing, and distribute the proxies geographically.

At this point you'll need a manager to decide where to put the proxies and how best to balance the load.

A notification manager can be used to register publishers and subscribers – it can be a large database. For reliability, there can be more than one notification manager. This can also help with discovery.

Brokers

Some applications require brokers/proxies.

A broker decouples the publisher from its subscribers, and can do a few things:

- forwarding – the work of sending notifications
- filtering – reduces load
    - according to e.g. content

- allow subscription to patterns of events
- notification mailboxes – notifications need to be delayed until the subscriber is ready to receive them

Characteristics of Distributed Event-Based Systems

- heterogeneous
  - publishers can be very different, but use same communication protocols
- asynchronous
  - notifications are sent asynchronously
  - difficult to make this robust
    * can't check if we received all the notifications, or if we missed one

3rd Party Systems

You can use third-party systems like APNS (Apple Push Notification Service) or Google CM (Firebase Cloud Messaging).

How long notifications are valid for is important because space is allocated to them while they're valid.

Practice

Event notification can happen between components on the same computer as well as across different computers (IPC vs. networked communication).

Analysis

- Different subscribers may receive event notifications in different orders, or may not receive some at all.
- The time it takes for a notification to arrive is unpredictable.
- Sometimes a subscriber may wish to only receive notifications e.g. every hour, rather than as the events happen.
- Sometimes the subscriber may not be the object to which a notification should be sent.

  - E.g. proxies/brokers

Jini Distributed Event Notification

Requirements:

- Specify an interface that can be used to send notifications
- Specify the information a notification must contain
- Allow various degrees of assurance on delivery of a notification
- Provide support for different notification scheduling policies
- Explicitly allow for brokers that can collect, hold, filter, and forward notifications

Registration of interest for an event type is typically limited to a specific duration with a lease.

All of the classes defined in the specification are in the net.jini.core.event package.

Interfaces and Classes

A `RemoteEventListener` is defined by an interface with just the method `notify()`.

Events are represented by the `RemoteEvent` class, which contains:

- the type of the event
- a reference to the source object for the event
- a sequence number identifying the instance of the event type
- an object that was passed in when the subscriber registered interest in the event

[...]

What's nice about this is that there's a standardised system.

Push Notifications

These are associated with remote applications and can be received by the user at any time.

Model

Three interacting components:

- The OS push notification service (OSPNS)
- The app publisher (which registers with the push notification service)
- [...]

How It Works

1. User installs an app
2. User runs the app – unique ids for the app and the device are registered with the OSPNS
3. The unique identifiers are passed back to the app from the OSPNS, as well as to the app publisher.
4. The app publisher receives and stores this data.

Then the publisher defines:

- The audience for a push notification
- Whether the message should be scheduled or sent immediately

Now the publisher can send notifications via the API according to the schedule.

Transactions

It's possible to allow event subscriptions within the scope of a transaction. Participants within the transaction can see events that are hidden from entities outside the transaction.

Consider notifications to be part of a transaction. Subscribers send an acknowledgement to confirm receipt of the notification or set of notifications.

Transactions are for grouping a set of actions into one greater (possibly atomic) action.

Questions

- What is the "evented web"?
- What are web notifications?
- What are the main issues with a distributed event notification system?