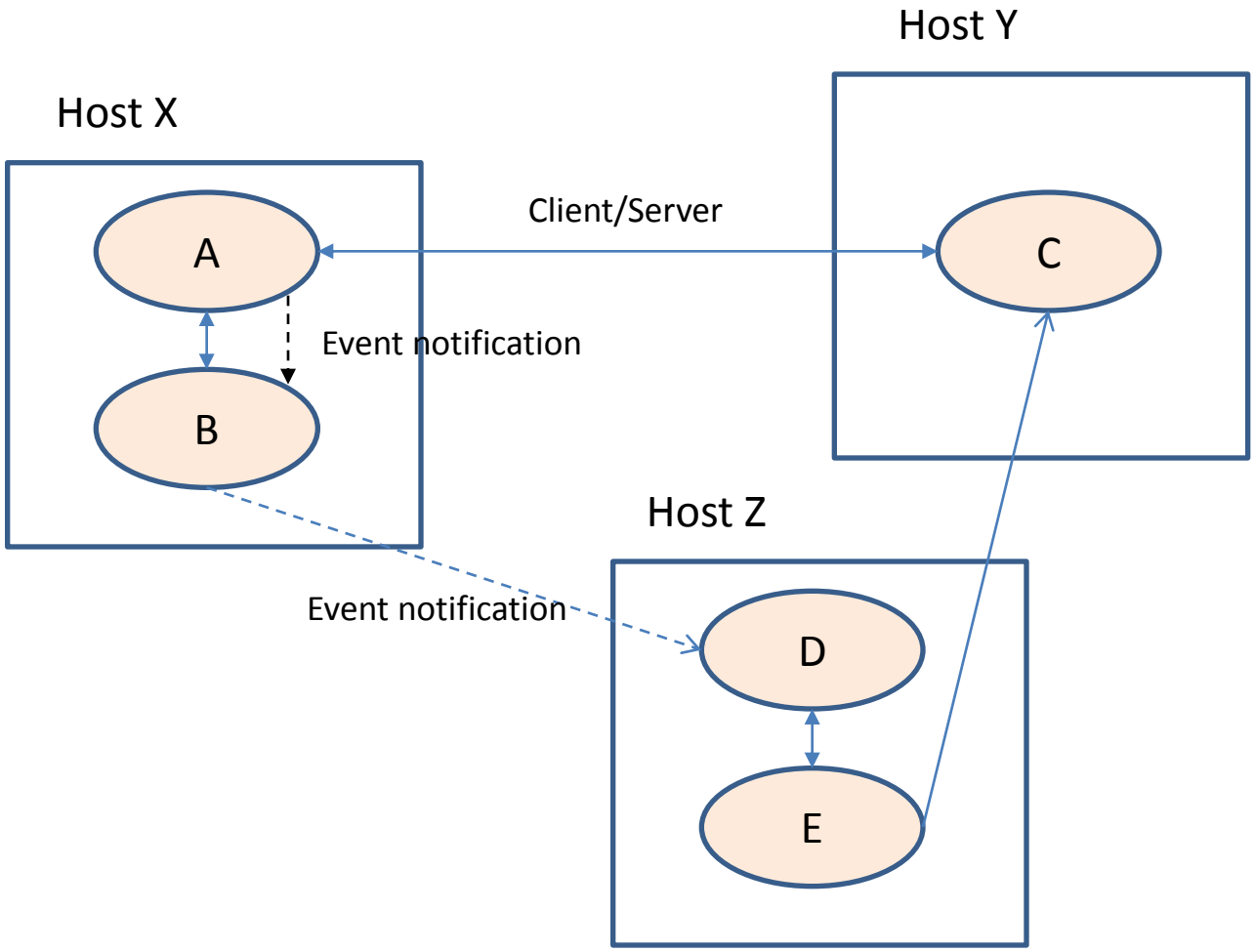
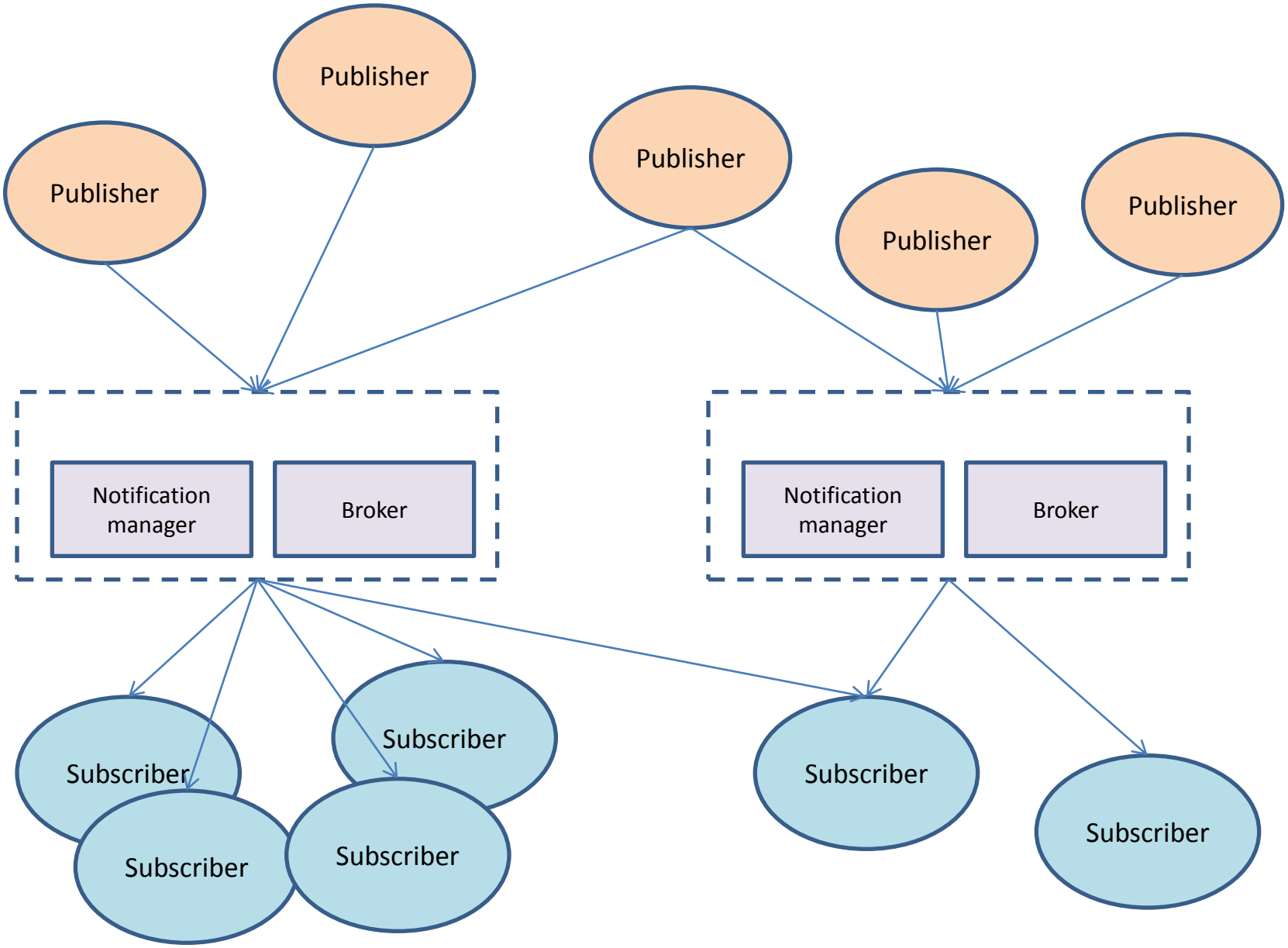
Lecture 9

Practice of Events and Alerts Notifications

The event notification service



Distributed application running on three networked hosts.


Publisher
Publisher
Publisher
Publisher
Publisher
Notification manager
Broker
Notification manager
Broker
Subscriber
Subscriber
Subscriber
Subscriber
Subscriber
Subscriber

Analysis

- Notifications of events from remote publishers may arrive in different orders to different subscribers, or may not arrive at all.

- The time it takes for a notification to arrive is *unpredictable* – it may be long, it might even be lost.

- There may be occasions in which the subscriber does not wish to have that notification as soon as possible, but only on some schedule determined by itself (e.g., after it was activated).

- There may even be times when the subscriber is not the object to which a notification of the event should be sent.

A. Jini distributed event notification

- Requirements:
  - specify an interface that can be used to send a notification of the occurrence of the event;
  - specify the information that must be contained in such a notification;
  - allow various degrees of assurance on delivery of a notification;
  - support for different policies of scheduling notification;
  - explicitly allow the interposition of brokers that will collect, hold, filter, and forward notifications.

- Registration of interest for an event type is not temporally open ended but is limited to a given duration using the notion of a lease.

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

Interfaces and classes

- A *RemoteEventListener* (the subscriber) is defined by an interface that contains a single method, *notify*, which informs about the occurrence of the event. This method returns no value.

- The basic form of a *RemoteEvent* class contains:
  - an identifier for the type of event;
  - a reference to the object in which the event occurred;
  - a sequence number identifying the instance of the event type;
  - an object that was passed in, as part of the registration of interest in the event by the subscriber.

- The *EventRegistration* class defines an object that returns the information required by the subscriber and is intended to be the return value of remote event registration calls. Instances of the EventRegistration class contain an identifier for the type of event, the current sequence number of the type of event, and a lease object for the registration of interest.

Example: EventPublisher Interface

```
public interface EventPublisher extends Remote {
  public EventRegistration register(long evId,
                                    MarshalledObject handback,
                                    RemoteEventListener toInform,
                                    long leaseLength)
          throws UnknownEventException, RemoteException;
}
```

- The second argument of the register method is a MarshalledObject that is to be handed back as part of the notification generated when the event occurs. This object is known to the subscriber and should contain any information that is needed to identify the event and to react to the occurrence of that event. This object will be passed back as part of the notification object that is passed as an argument to the notify method.
- leaseLength indicates the requested duration of the registration. This period is a request. The period actually granted by the publisher may be different. The actual duration of the registration lease is returned as part of the Lease object included in the EventRegistration object.

The EventRegistration Class

```
public class EventRegistration implements java.io.Serializable {
   public EventRegistration(long eventID,
                            Object eventSource,
                            Lease eventLease,
                            long seqNum) {...}
   public long getID() {...}
   public Object getSource() {...}
   public Lease getLease() {...}
   public long getSequenceNumber() {...}
}
```

- Objects of the class EventRegistration are meant to encapsulate the information the subscriber needs to identify a notification as a response to a registration request and to maintain that registration request.

B. Push notifications

- A push notification is a message that pops up on a mobile device. They are associated with remote applications (publishers) and can be received at any time.
  - Example: the latest (changed) score of a match, traffic updates, or a flash sale.

- For app publishers, push notifications are a way to speak directly to a user – ads, improve user experience

- All mobile platforms have support for push notifications — iOS, Android, Fire OS, Windows have their own services.

The model

- There are three interacting components:
  - The OS push notification service (OSPNS)
  - App publisher - the app publisher is enabled to work with OSPNS.
  - Client app - this is an app installed on a user's device that will receive incoming notifications.

- The app publisher registers with the OS push notification service.

- The OS service provides an application programming interface (API) to the app publisher.

- The app publisher uploads the app to the app store.

How it works

1. The user visits an OS app store, downloads and installs the app.
2. The user opens the app. Unique identifiers (IDs) for both the app and the device are registered with the OSPNS.
3. The unique identifiers are passed back to the app from the OSPN. They are also sent to the app publisher.
4. The app publisher receives and stores these data.

1. The publisher defines the audience to whom the push notification will be sent.
2. The publisher determines whether the message should be sent immediately or scheduled.
3. The publisher sends notification(s) via the API according to the schedule.

C. Use of transactions

- It is possible to allow event subscriptions within the scope of a transaction, in such a way that the notifications of these events can occur within the scope of the transaction. This means that other participants in the transaction may see some events whose visibility is hidden by the transaction from entities outside it.
- Notifications of these events will not be sent to entities that registered interest in this type of event outside the scope of the transaction until and unless the transaction is committed.
- The duration of the subscription is the minimum of the length of the lease and the duration of the transaction.
- Transactions bring robustness to event notification.

Questions

- What is the “evented web”?
- What means web notification?
- What are the main issues associated with a distributed event notification system?

- https://river.apache.org/doc/specs/html/event-spec.html
- http://www.w3.org/TR/notifications/