Indirect Communication

Chapter 6 from the TextBook

Indirect Communication

- Point-to-point communication (Examples?)
- Indirect communication (how?)
- The techniques covered so far are based on a direct coupling between a sender and a receiver (Direct coupling?)
- In contrast, indirect communication avoids this direct coupling and hence inherits interesting properties

Indirect Communication

- Two key properties stem from indirect communication:
  - Space uncoupling
    - Effect?
  - Time uncoupling
    - Effect?
- Is RMI space-coupled?
- Is RMI time-coupled?
- Is message-passing space-coupled?
- Is message passing time-coupled?

Indirect Communication?

- Indirect communication is defined as communication between entities in a distributed system through an intermediary with no direct coupling between the sender and the receiver(s).
- Indirect communication paradigms:
  - Publish-subscribe systems
  - Message queues

Indirect Communication

- Good for:
  - Environments where the users connect and disconnect very often (e.g., mobile environments)
  - Event dissemination where receivers may be unknown or change often
  - Scenarios with large number of participants
  - Anticipated change (how?)
- Limitations?

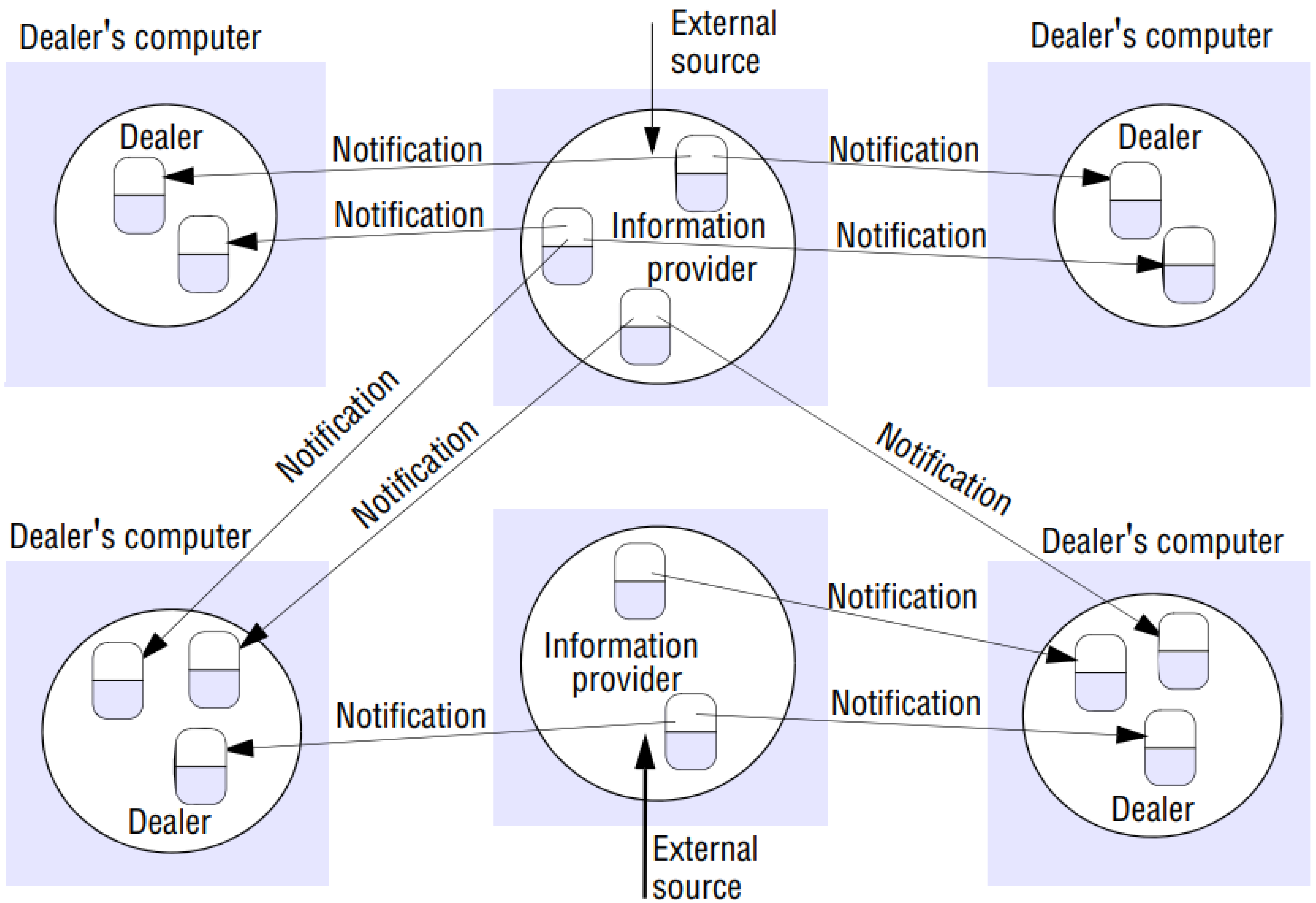
Publish-Subscribe Systems

- Also known as “distributed event-based systems”
- A publish-subscribe system is a system where *publishers* publish structured events to an event service and *subscribers* express interest in particular events through *subscriptions* which can be arbitrary patterns over the structured events.
- A given event will be delivered to potentially many subscribers, and hence publish-subscribe is fundamentally a one-to-many communication paradigm.
- Applications?

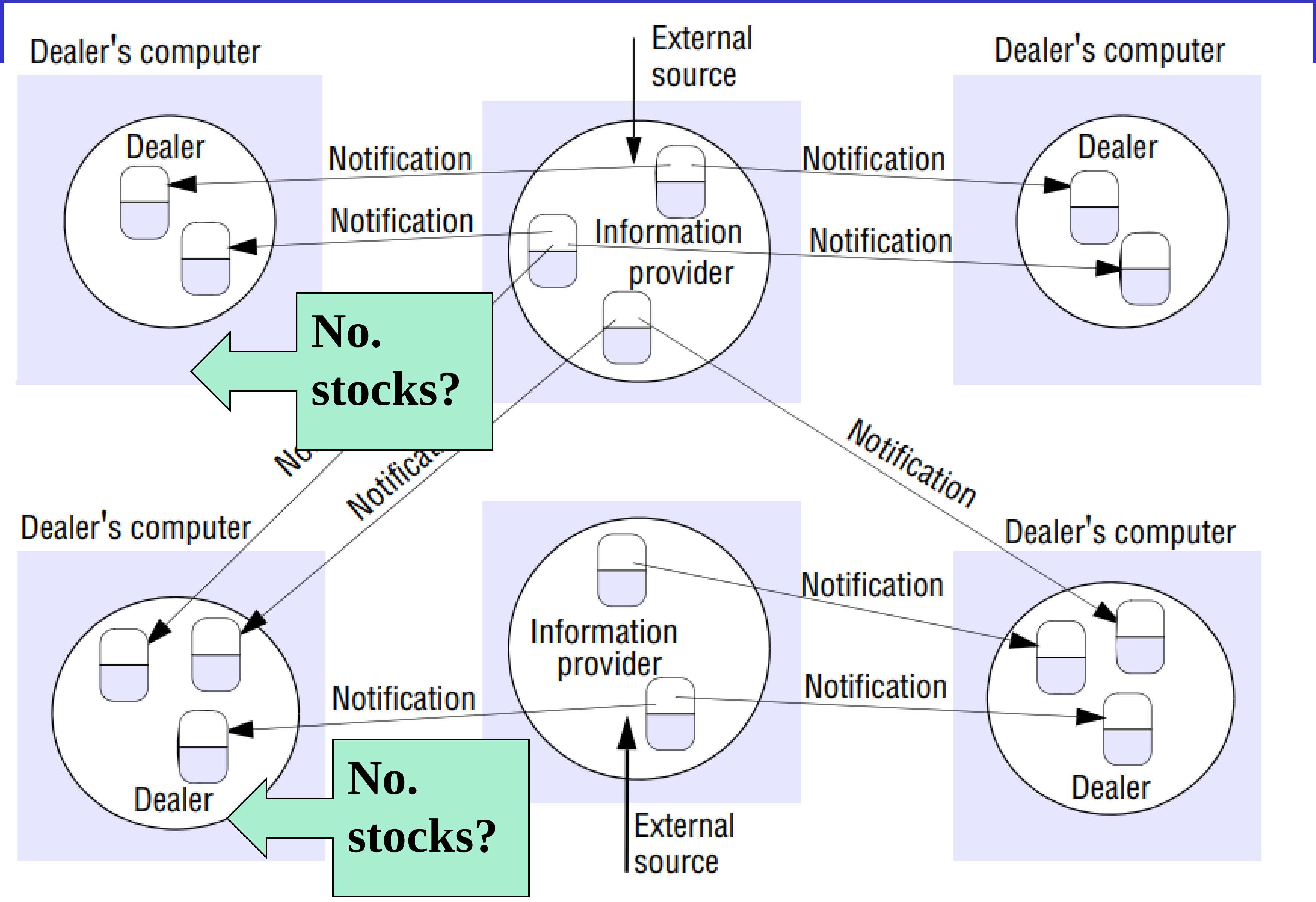
Example: Dealing Room System

- Consider a simple dealing room system whose task is to allow dealers using computers to see the latest information about the market prices of the stocks they deal in.
- The market price for a single named stock is represented by an associated object.
- The information arrives in the dealing room from several different external sources in the form of updates to some or all of the stocks.
- Dealers are typically interested only in their own specialist stocks.

Example: Dealing Room System



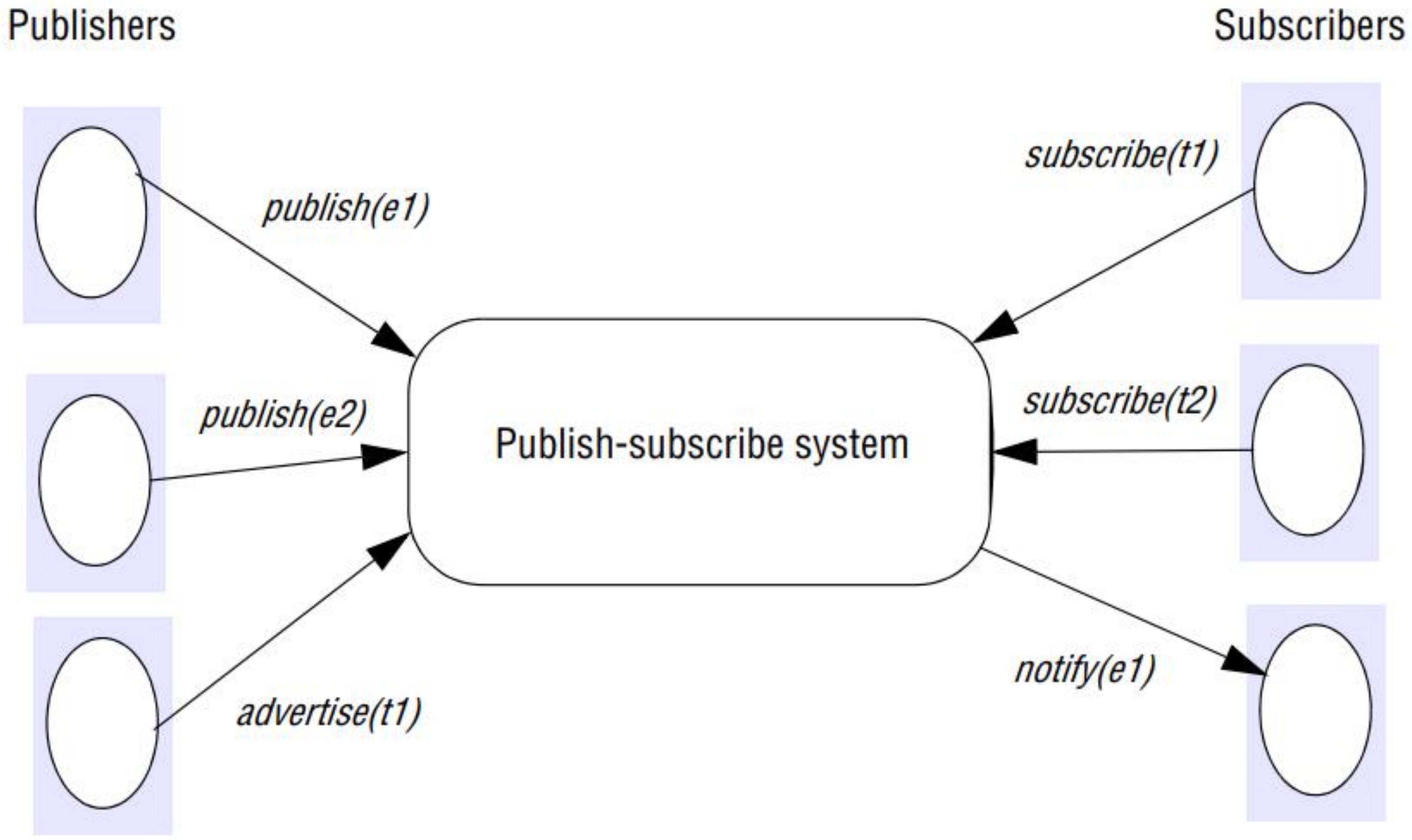
Example: Dealing Room System



Characteristics of Publish-Subscribe Systems

- Heterogeneity: All that is required is that event-generating objects publish the types of events they offer, and that other objects subscribe to patterns of events and provide an interface for receiving and dealing with the resultant notifications.
- Asynchronicity: Notifications are sent asynchronously by event-generating publishers to all the subscribers that have expressed an interest in them to prevent publishers needing to synchronize with subscribers

Publish-Subscribe Programming Model



Publish-Subscribe Programming Model
Filter Model

1) Channel-based: publishers publish events to named channels and subscribers then subscribe to one of these named channels to receive all events sent to that channel.
2) Topic-based: each notification is expressed in terms of a number of fields, with one field denoting the topic.
3) Content-based: Content-based approaches are a generalization of topic-based over a range of fields in an event notification.

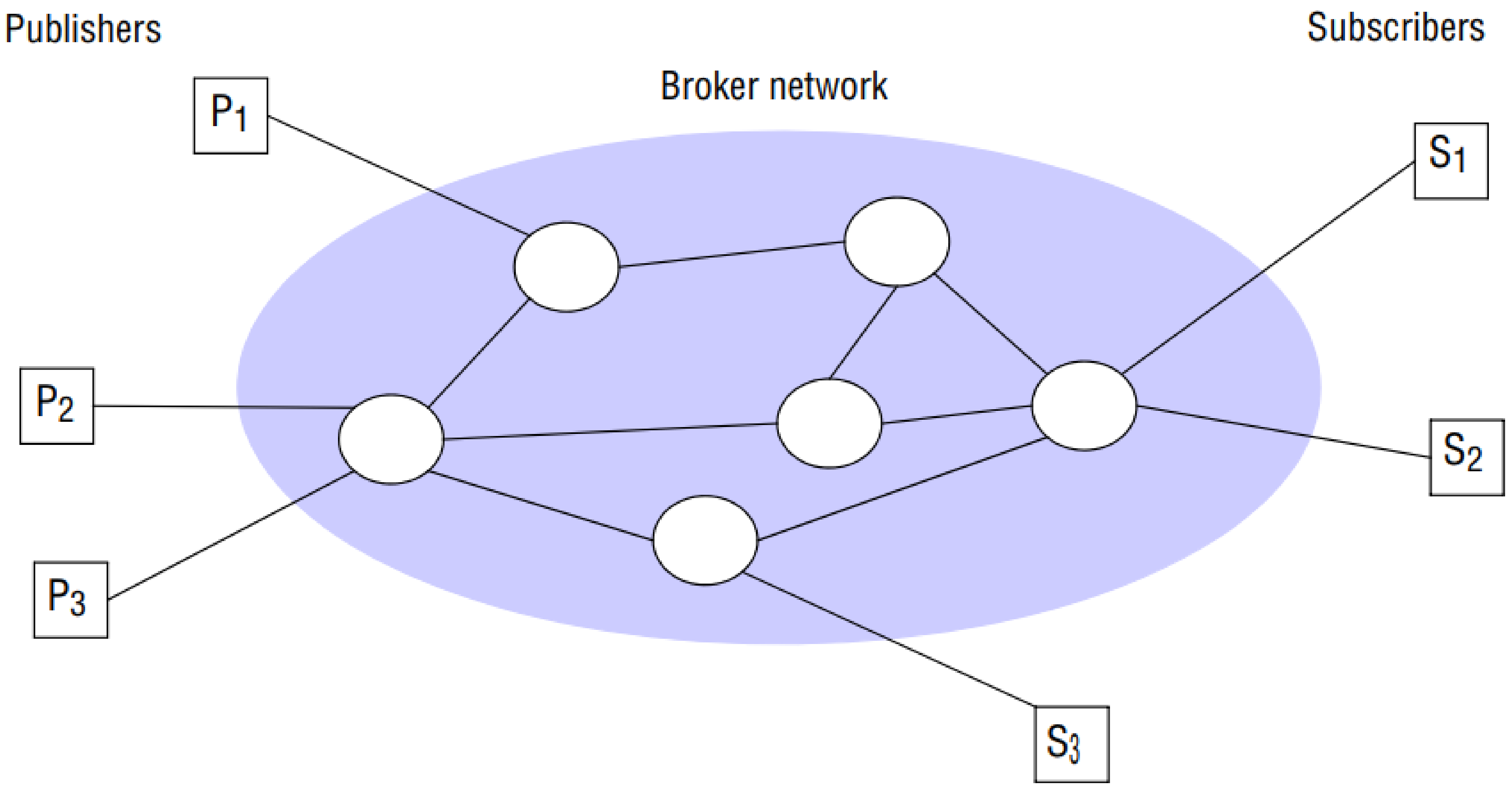
Publish-Subscribe Programming Model
Filter Model

- Example: Alexander is interested in the topic of publish-subscribe systems, where the system in question is the ‘CORBA Event Service’ and where the author is ‘Tim Kindberg’ or ‘Gordon Blair’. (Filter?)

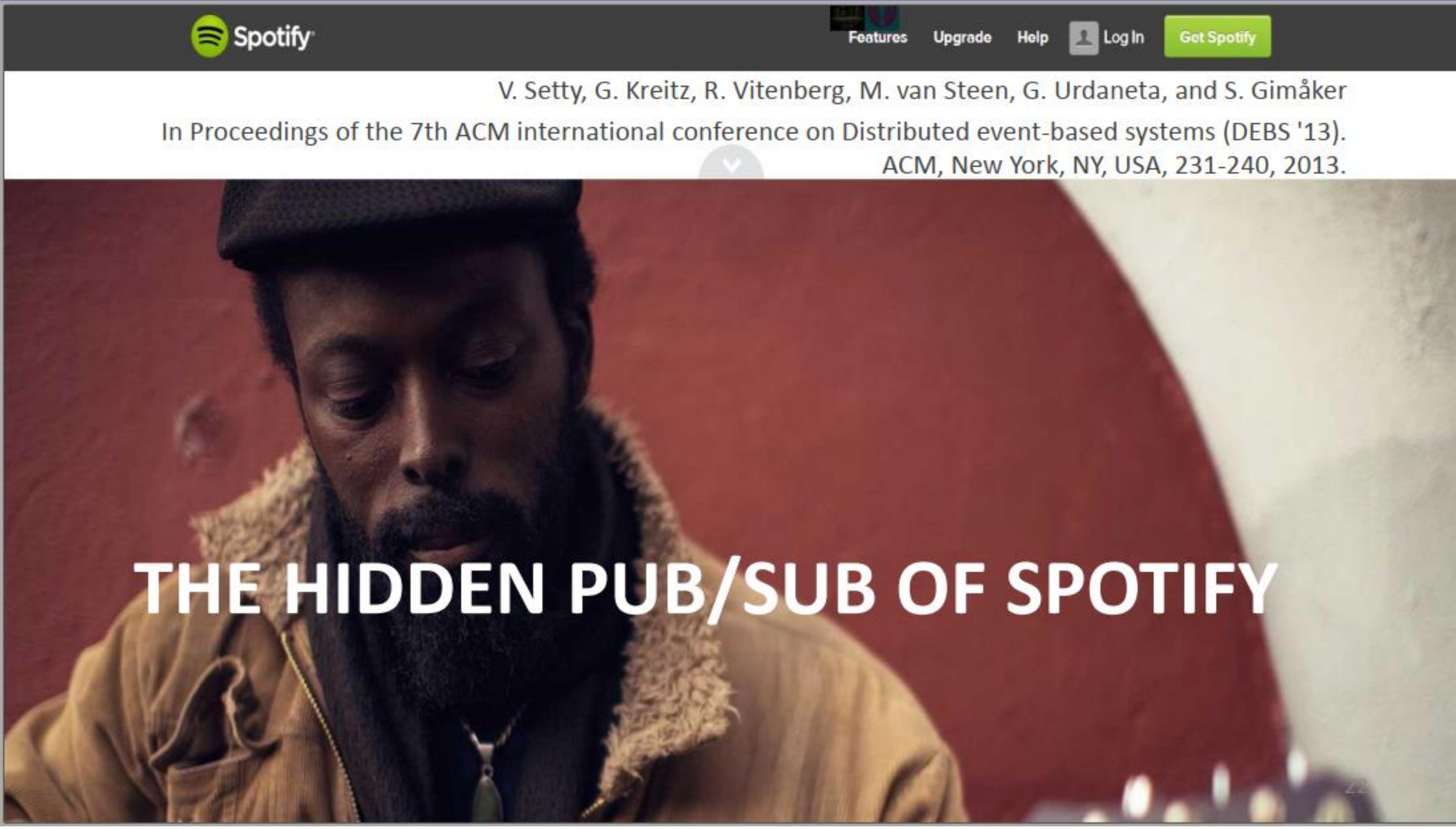
Publish-Subscribe Programming Model **Concerns?**

- The main concern is to deliver events efficiently to all subscribers
- Centralized implementation using **brokers** (How?)
  - Scalability
  - Failure handling
- Distributed implementation

Distributed Implementation



Case Study: Spotify



V. Setty, G. Kreitz, R. Vitenberg, M. van Steen, G. Urdaneta, and S. Gimåker

In Proceedings of the 7th ACM international conference on Distributed event-based systems (DEBS '13). ACM, New York, NY, USA, 231-240, 2013.

THE HIDDEN PUB/SUB OF SPOTIFY

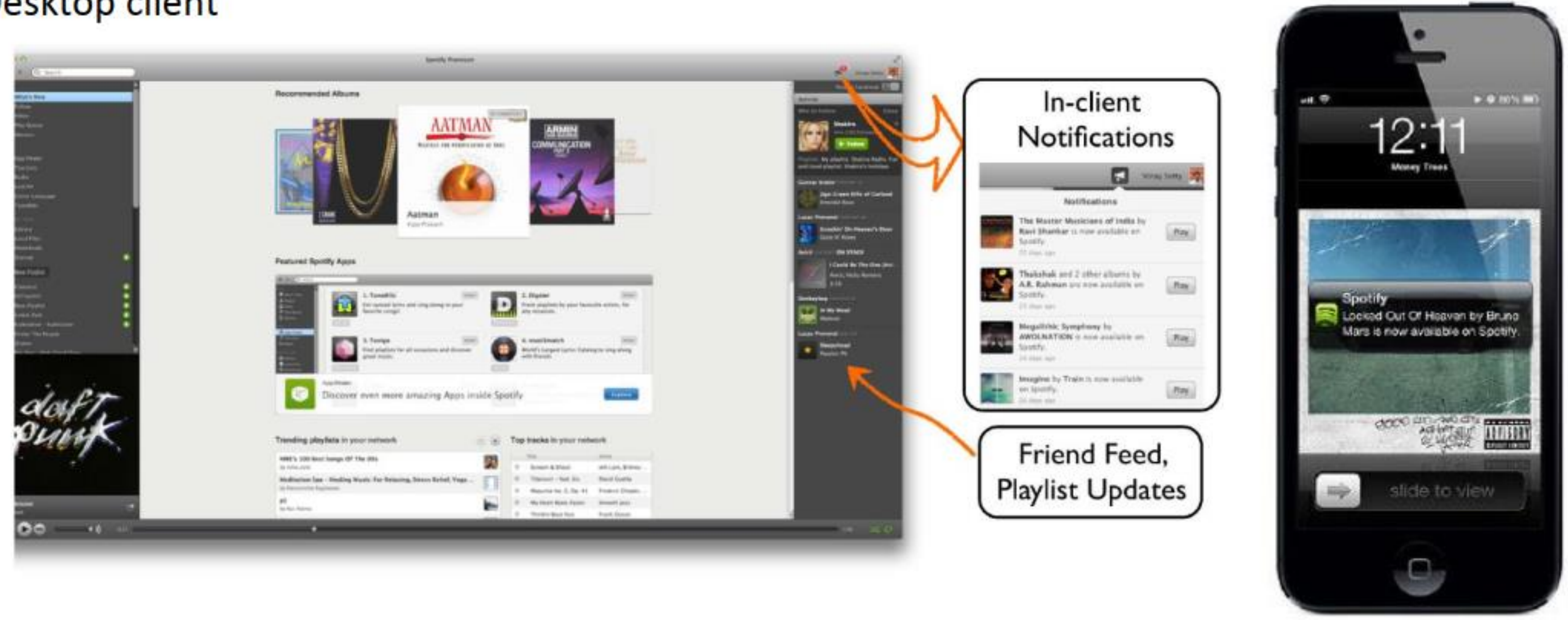
Those slides are from a previous offering of the course “Distributed Systems (5DV147)”

Case Study: Spotify

- Topic-based subscriptions
- Hybrid engine
  - Relay events to online users in real time
  - Store and forward selected events to offline users
- Brokers are organized as a DHT (distributed hash table) overlay that spans three sites in Sweden, UK, and USA.
- Design to scale
  - Stores approx., 600 million subscriptions at any given time
  - Matches billions of publication events every day

Case Study: Spotify

Desktop client


In-client Notifications
Friend Feed, Playlist Updates
Spotify
Locked Out Of Heaven by Bruno Mars is now available on Spotify.
12:11
slide to view

Those slides are from a previous offering of the course “Distributed Systems (5DV147)”

Case Study: Spotify

Topic-based subscription

subscription(user_name, topic_name)

- Types of topics
  - Friends (Spotify + Facebook): FB friends who are Spotify users and by sharing music
  - Playlists (URI): other users playlists (updates), “Collaborative” playlists or only modifiable by creator
  - Artists pages (follow artist): new albums or news related to artist

Case Study: Spotify

Publication events

- All events delivered in real time (best effort and guaranteed delivery) to online users
- Some notifications are sent by email to retrieve in the future
- Example, new album from famous artist added
  - Instant notification sent to online followers
  - Email notification to offline followers
  - Event persisted so that (new) followers can retrieve it in the future (e.g., from another device)

Those slides are from a previous offering of the course "Distributed Systems (5DV147)"

Case Study: Spotify

Publication events

- Friend feed
  - – Event notification to all friends following user
    - Play a track, create or modify playlist, add a favorite(artist, track, album)
    - Publish event on Facebook wall (optional)

Case Study: Spotify

Publication events

- Playlist updates
- – Event notifications when
  - A playlist is modified (adding or removing track, renaming playlist) via friend feed
- Synchronize playlist across all devices of all subscribers of the playlist

Those slides are from a previous offering of the course “Distributed Systems (5DV147)”

Case Study: Spotify

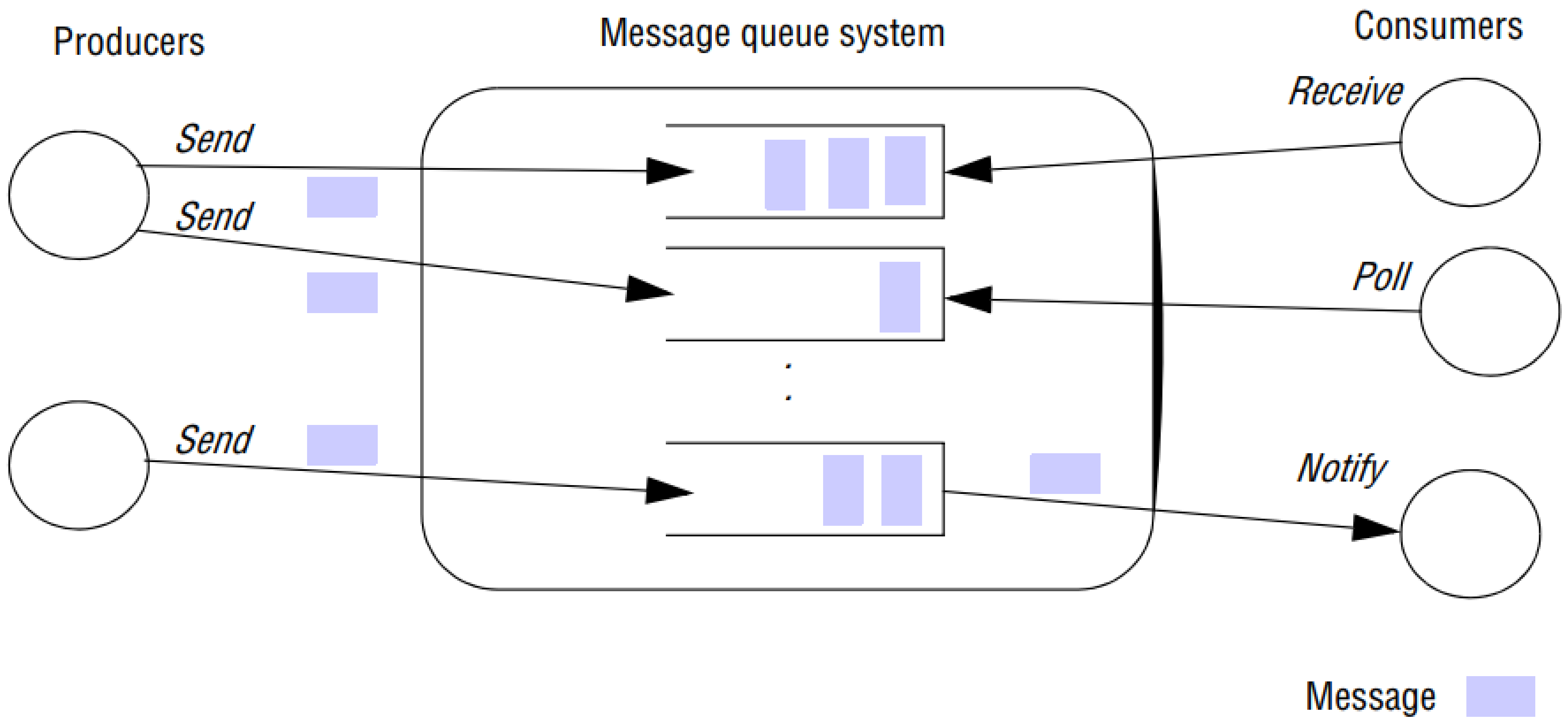
Publication events

- Artist pages
- – Notification sent to followers of artist when
  - New album added in Spotify
  - Playlist created by artist

Message Queues

- Whereas groups and publish-subscribe provide a one-to-many style of communication, message queues provide a point-to-point service using the concept of a message queue as an indirection, thus achieving the desired properties of space and time uncoupling.
- They are point-to-point in that the sender places the message into a queue, and it is then removed by a single process.
- Message queues are also referred to as Message-Oriented Middleware.

Message Queues



- Three styles of receive are generally supported
- Marshalling?

RedHat Certification – Chapter 8
Creating Messaging Applications with JMS

- Using Asynchronous Messaging Concepts
  - Messaging solutions use the concept of queues or destinations to facilitate the transfer of data from one application to another.
  - Developers use queues as an intermediary between applications.
- Consider a front-end web application for an e-commerce business sending data to a back-end order fulfillment system to fill the order.
  - When does the back-end system need to process the order?
- **Coupling? Message format?**

RedHat Certification – Chapter 8
Creating Messaging Applications with JMS

- Using Asynchronous Messaging Concepts
  - Messaging solutions use the concept of queues or destinations to facilitate the transfer of data from one application to another.
  - Developers use queues as an intermediary between applications.
- Consider a front-end web application for an e-commerce business sending data to a back-end order fulfillment system to fill the order.
  - When does the back-end system need to process the order?
- **Coupling? Message format?**

Chapter 8
Using a **Queue** for Point-to-Point Messaging

Figure 8.1 from RH student guide removed for Copyright preservation.

- Point-to-point messaging?
- Pull-based model
- In the point-to-point model, a queue consumer typically must acknowledge successful processing of the message, or it is put back onto the queue to be retried.

Chapter 8
Using a **Topic** for Publish-Subscribe Messaging

Figure 8.2 from RH student guide removed for Copyright preservation.

- Publish-subscribe messaging?
- Push-based model
- Durability?

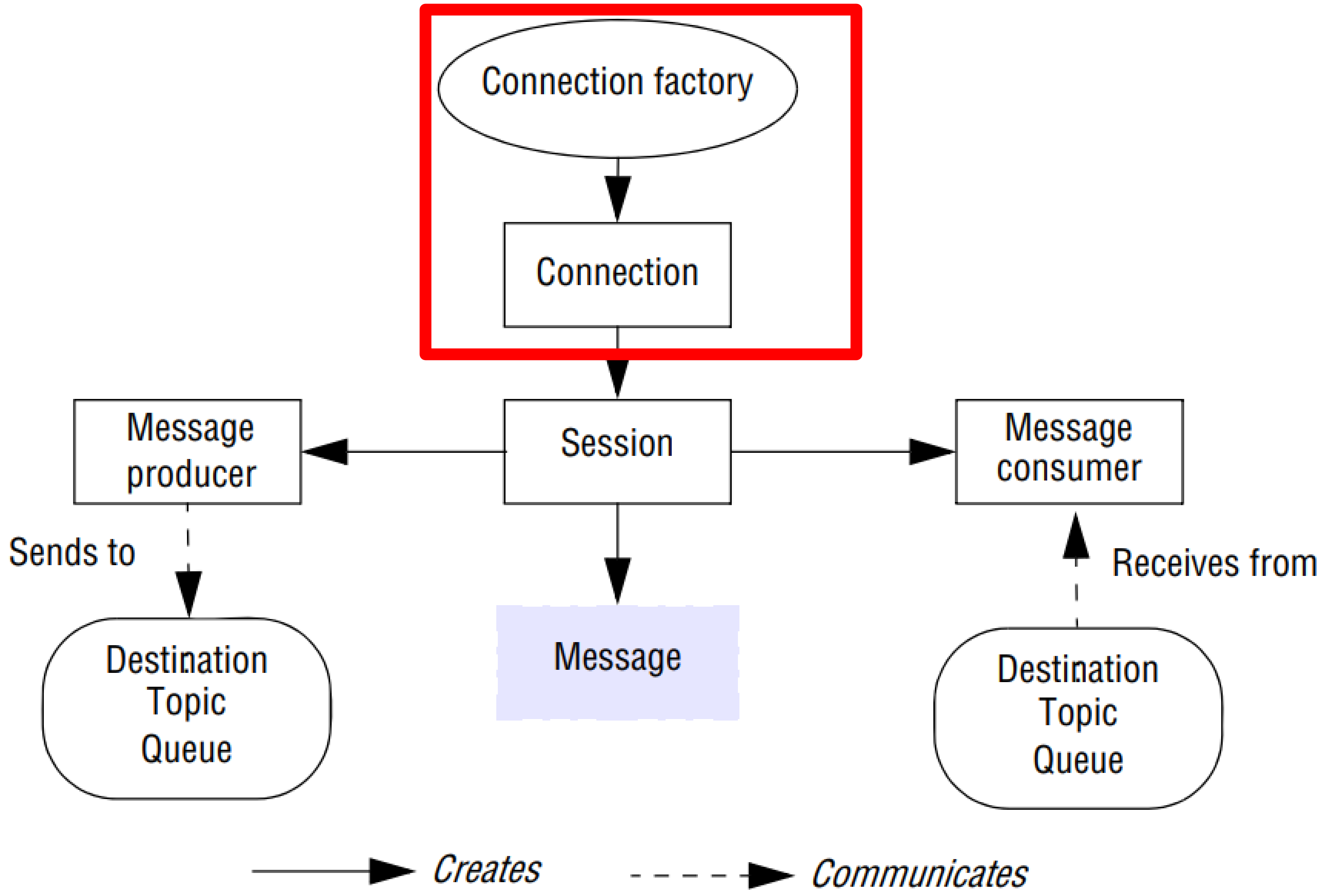
Chapter 8
Java Message Service (JMS) API Specification

Table JMS Concepts from Chapter 8: RH student guide removed for Copyright preservation.

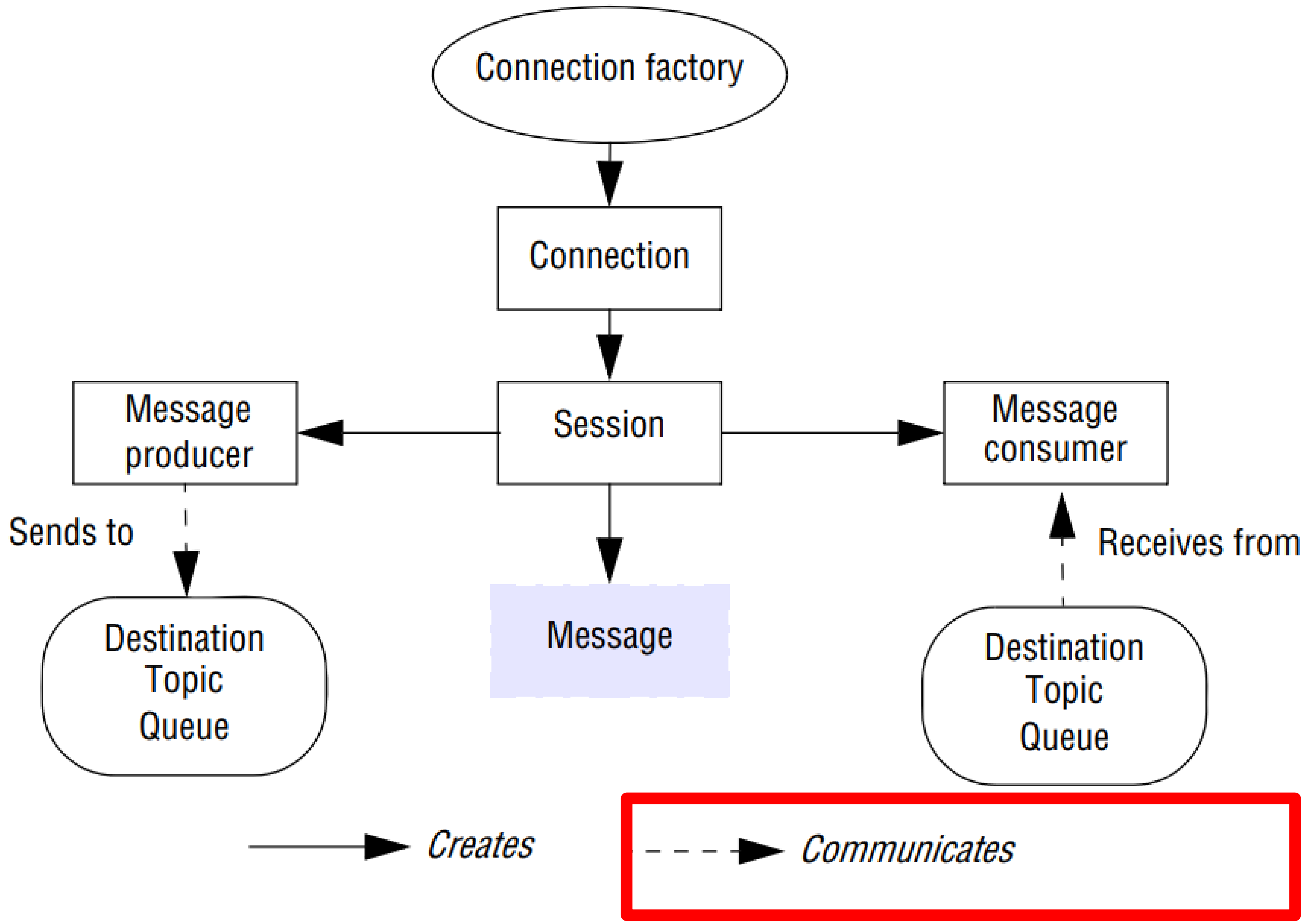
Chapter 8
Java Message Service (JMS) API Specification

Table JMS Concepts from Chapter 8: RH student guide removed for Copyright preservation.

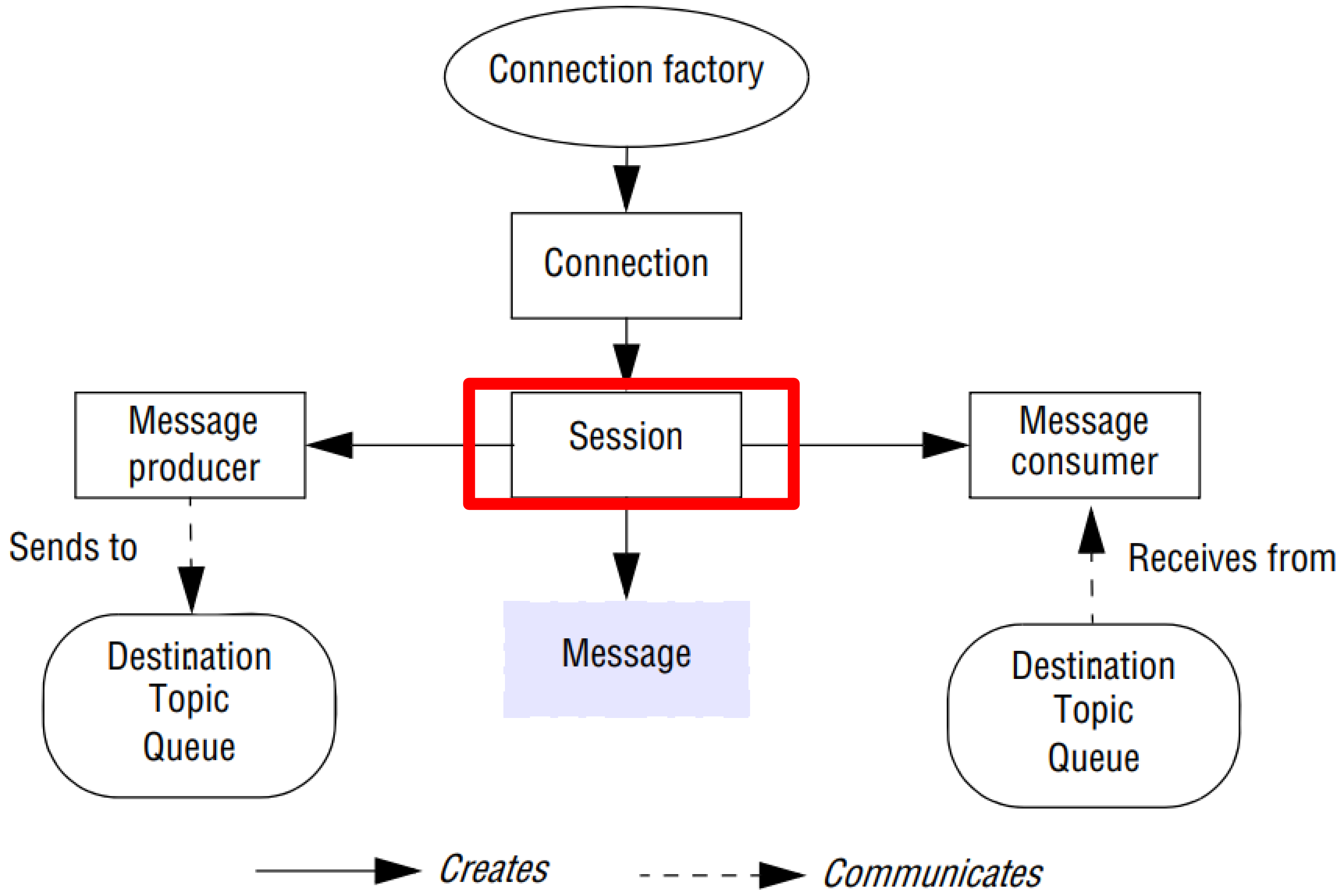
The programming model offered by JMS



The programming model offered by JMS



The programming model offered by JMS



Chapter 8
Describing JMS Architecture

- JBoss Enterprise Application Platform 7 leverages an **Apache ActiveMQ Artemis** messaging broker as an embedded JMS provider.
- JBoss EAP 7 deploys and manages Artemis as the **messaging-activemq** subsystem.

Required Readings

- Chapter 6: **Distributed Systems: Concepts and Design**, 5th Edition. George **Coulouris**, Cambridge University. Jean Dollimore, Formerly of Queen Mary, University of London.
- Red Hat Application Development 1: Programming in Java EE Edition 2
  - Chapter 8: “Describing Messaging Concepts” section