Unit 2 Message-oriented and peer to peer systems

Unit Outcomes. Here you will learn

- to program messaging among Java applications using the JMS standard
- to program a simple P2P system using JMS
- describe characteristics and benefits of P2P DS
- explain why peers need to implement a routing facility, giving at least two reasons
- explain how prefix routing works giving a simplified example

Further Reading: Sun JMS tutorial, CDK2005 10

Contents

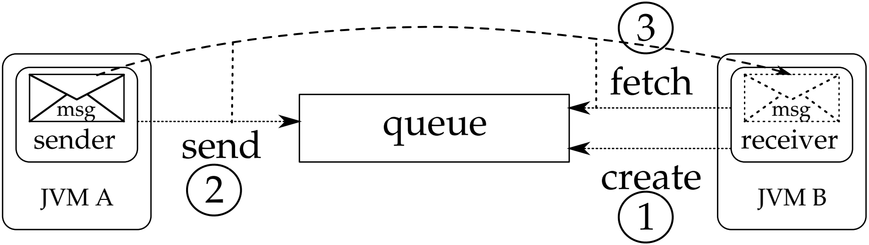
1. Java Messaging Service (JMS)
   - Purpose and design
   - Direct messaging
   - Mini JMS example

2. Peer to peer systems
   - Characteristics
   - Routing overlay
   - Broadcast routing
   - Prefix routing
   - Prefix routing to objects
   - Distributed hash table (DHT)

Java Messaging Service (JMS)

Purpose and design

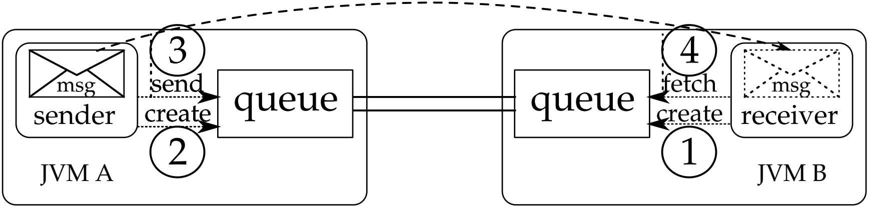
- JMS facilitates asynchronous messaging among JVMs:



- sending always asynchronous
- receiving can be synchronous or asynchronous
- J2EE application servers are supposed to manage the queues
- Manaray, ActiveMQ implement JMS but not the rest of J2EE

Direct messaging

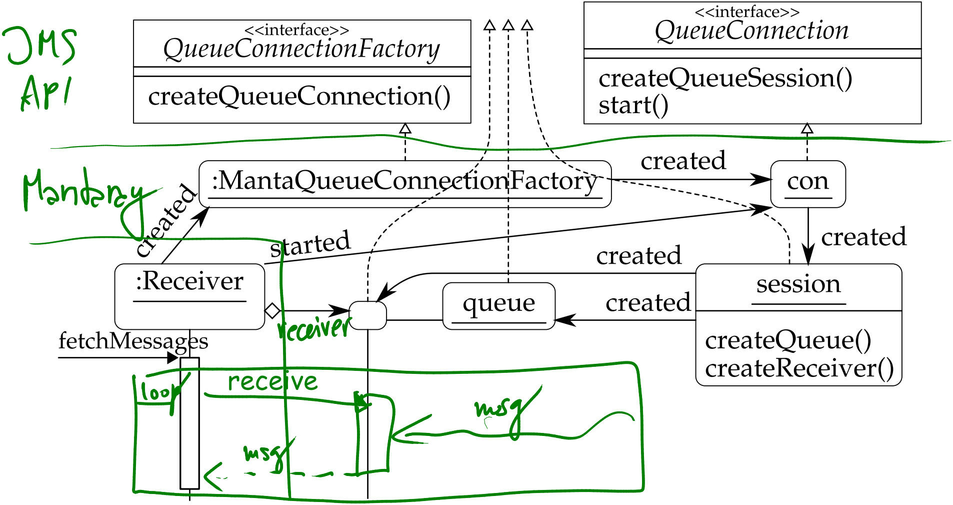
- Mantaray lets the peers manage the queue:



- no need for any central server except for bootstrapping:
  - on LAN automatic discovery of remote queues using broadcast
  - on WAN need WAN Bridge — a lightweight server to help establish connection between peers' queues

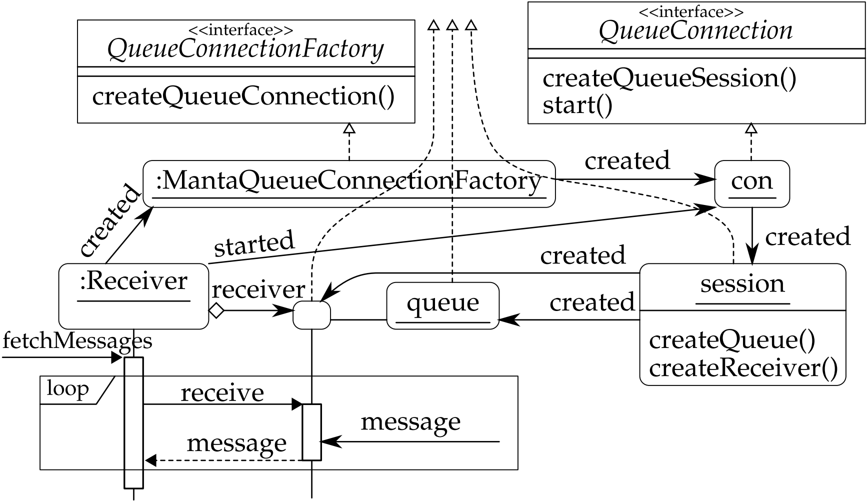
Mini JMS example — synchronous receiver

- initialisation creates the following objects and then interacts as shown:



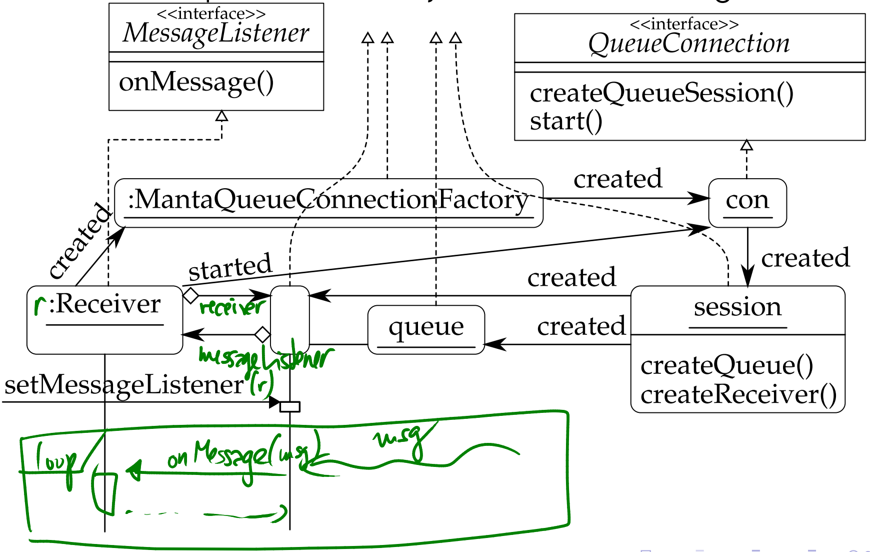
Mini JMS example — synchronous receiver

- initialisation creates the following objects and then interacts as shown:



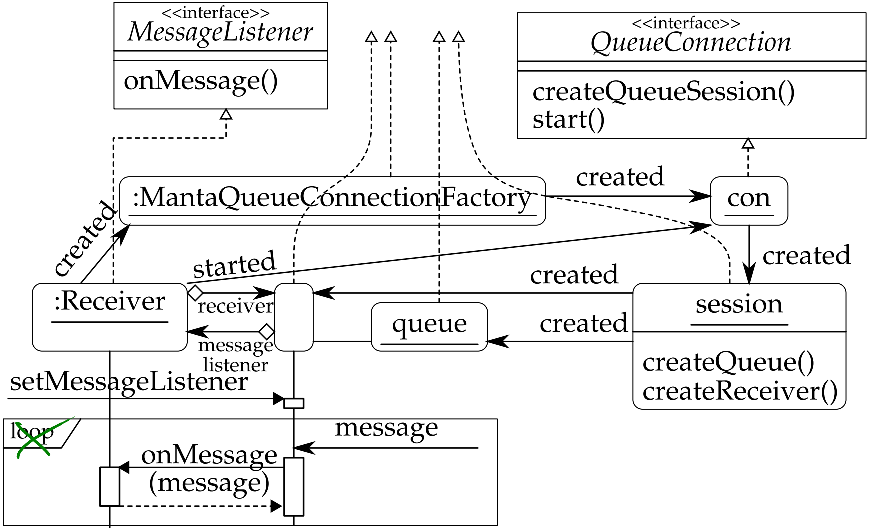
Mini JMS example — asynchronous receiver

- as before except `Receiver` object serves as message listener:



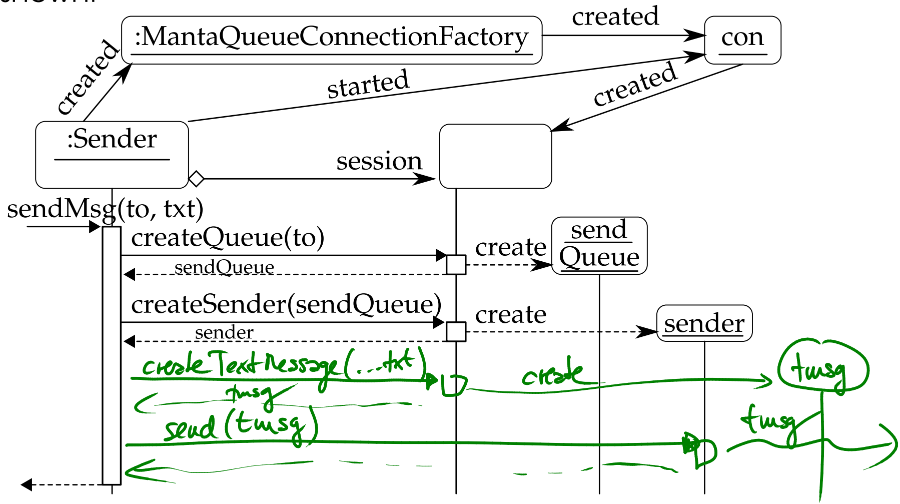
Mini JMS example — asynchronous receiver

- as before except `Receiver` object serves as message listener:



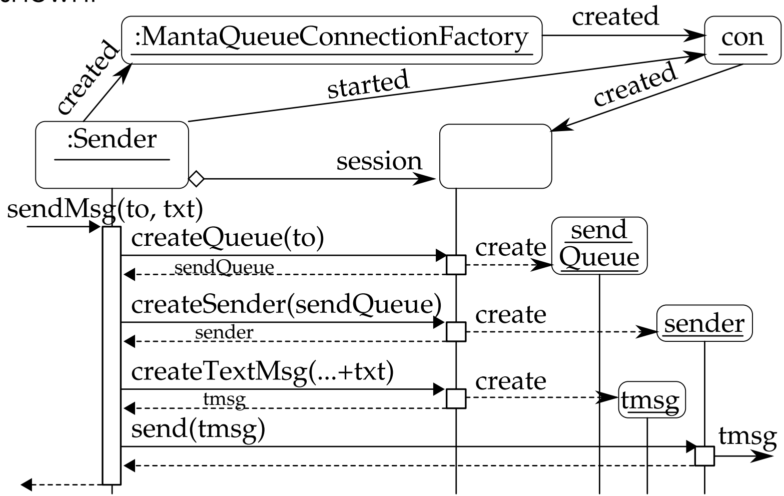
Mini JMS example — sender

- initialisation creates the following objects and then interacts as shown:



Mini JMS example — sender

- initialisation creates the following objects and then interacts as shown:



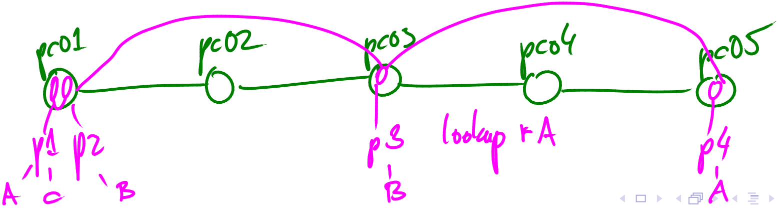
Peer to peer systems
Characteristics

- in a P2P system:
  - all peers contribute resources
  - all peers functionally equivalent
    but may hold different data
  - each item of data is placed in multiple nodes
- benefits:
  - high scalability
  - peer failure transparency
  - potential for anonymity (why?)

eg freenet, tor

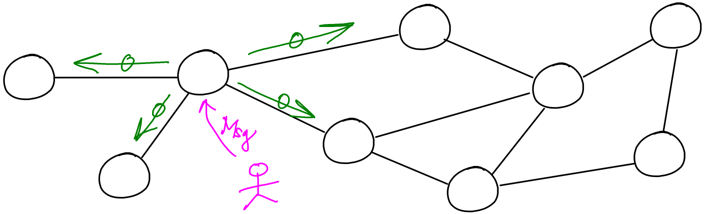
Routing overlay

- peers must include routing functionality — why?
  - messages often addressed to resources, not peers
  - peers can change their IP:
    - relocated to different computer
    - computer physically moves in a network
- peers and objects have logical addresses
- addressing and routing provided by P2P middleware



Broadcast routing

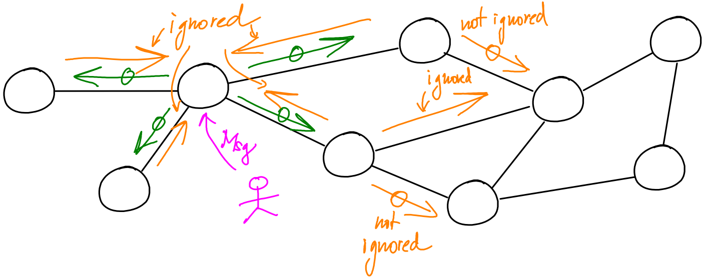
- each peer knows IDs of several neighbour peers
- peer forwards all messages to all neighbours



- only practical for broadcasting to all peers

Broadcast routing

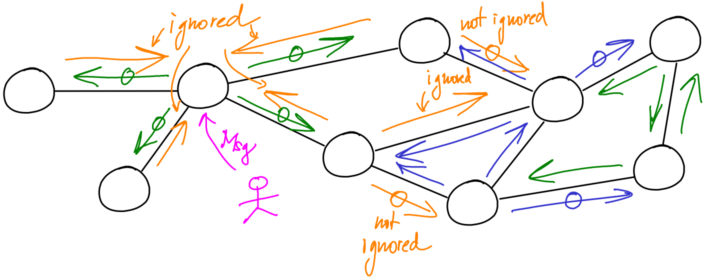
- each peer knows IDs of several neighbour peers
- peer forwards all messages to all neighbours



- only practical for broadcasting to all peers

Broadcast routing

- each peer knows IDs of several neighbour peers
- peer forwards all messages to all neighbours

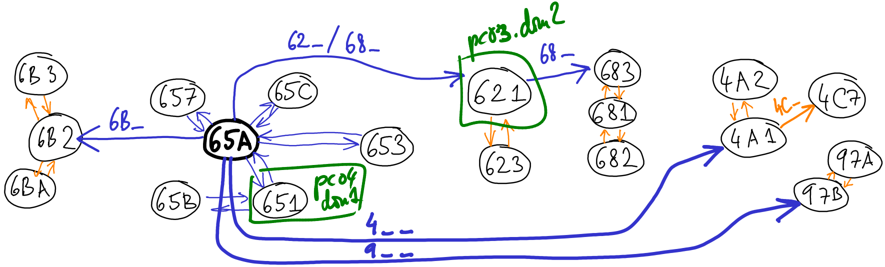


- only practical for broadcasting to all peers

Prefix routing (1/2)

- peers have routing tables:
  - logical name → Internet name of neighbour to forward to
- table must cover all possible names
- names — usually *Globally Unique Identifiers* (GUIDs)
  - 128 bits long (16 bytes)
  - when randomly generated, only rarely not unique

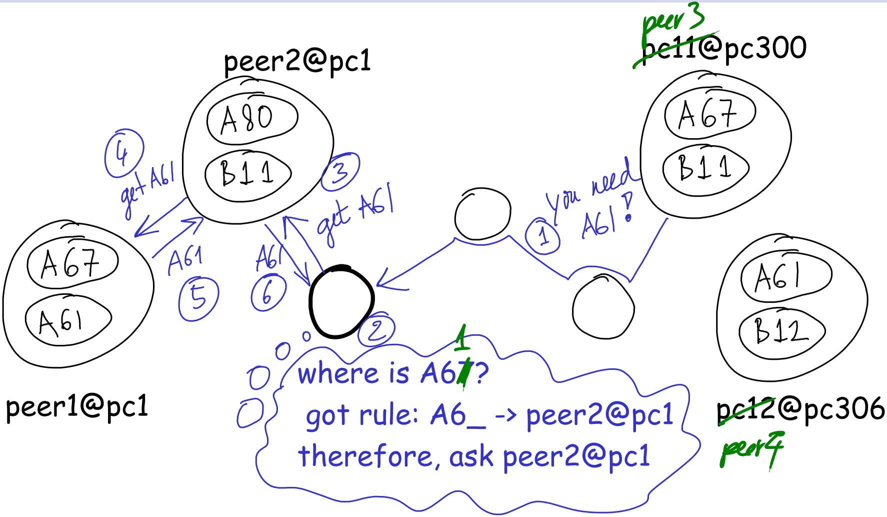
Prefix routing (2/2)



Routing table at peer with address 65A:

level 1		level 2		level 3	
651	pc0**4**.dom1	62 _	621 pc03.dom2	4 _ _	4A1 pc02.dom5
652	pc01.dom1	68 _	621 pc03.dom2	9 _ _	97B pc07.dom3
657	pc02.dom**2**	6B _	6B2 pc11.dom6	⋮	
65B	pc0**4**.dom1	⋮			
65C	pc08.dom1				
⋮					

Prefix routing to objects



peer2@pc1

~~pc11~~@pc300

peer1@pc1

~~pc12~~@pc306

where is A6~~7~~?

got rule: A6_ -> peer2@pc1

therefore, ask peer2@pc1

Distributed hash table (DHT)

- a very common pattern for P2P systems:
  a very large map: key $\mapsto$ value
  - key = resource name — can route to a peer that has the value
  - often key = resource name = hash of the value
- eg recent versions of BitTorrent use DHT for tracking peers who participate in the distribution of some file
  - value = a block of a shared file
  - key = its hash value as shown in the torrent descriptor

Learning Outcomes

Learning Outcomes. You should now be able to

- describe the characteristics and benefits of a P2P DS
- explain why peers need to implement a routing facility, giving at least two reasons
- explain how prefix routing works giving a simplified example
- explain and modify a simple P2P system programmed using JMS