CS 455: INTRODUCTION TO DISTRIBUTED SYSTEMS [LAB SESSION 1]

Shrideep Pallickara
Computer Science
Colorado State University

1

Why abstractions are important

- Abstraction is the key to managing **complexity**
- Good abstractions turn a difficult task into two manageable ones
  1. Defining and implementing abstractions
  2. Using abstractions to solve problem

2

Other suggestions that will make life easier: A stitch in time … [1/2]

- **Comment** your code
  - This is especially true in places where you are performing bitwise manipulations
- **Name** your variables so that you can know what they are anywhere in the code
- Keep functions **short**
- Check for **invariant violations** in your code
- **Test** the functionality of the small pieces

Other suggestions that will make life easier: A stitch in time … [2/2]

- Your code should run on CS department machines
- Use a version control system
  - Commit Often!
- Follow the guidelines from the beginning
  - build system, directory structure, etc.
- Follow the milestone plan as closely as possible

ASSIGNMENT

5

Two components

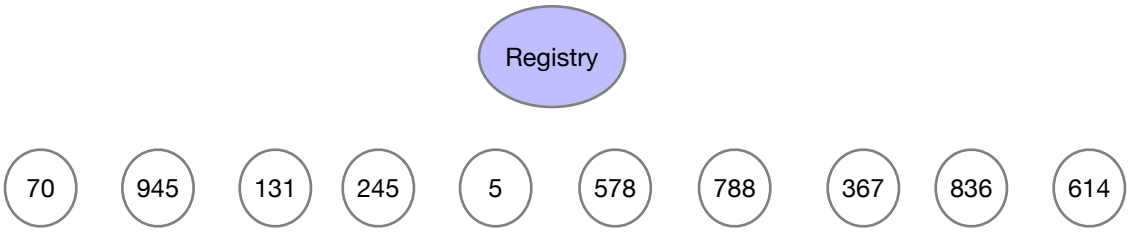
- Registry
  - Exactly one
- Messaging Node
  - Multiple

6

Functions

- Registry
  - Assign unique identifiers to individual messaging nodes
  - Construct an overlay by telling messaging nodes **who** they should connect to
- Messaging node
  - Send and receive messages

Example: Sequence of node arrivals and ID generation

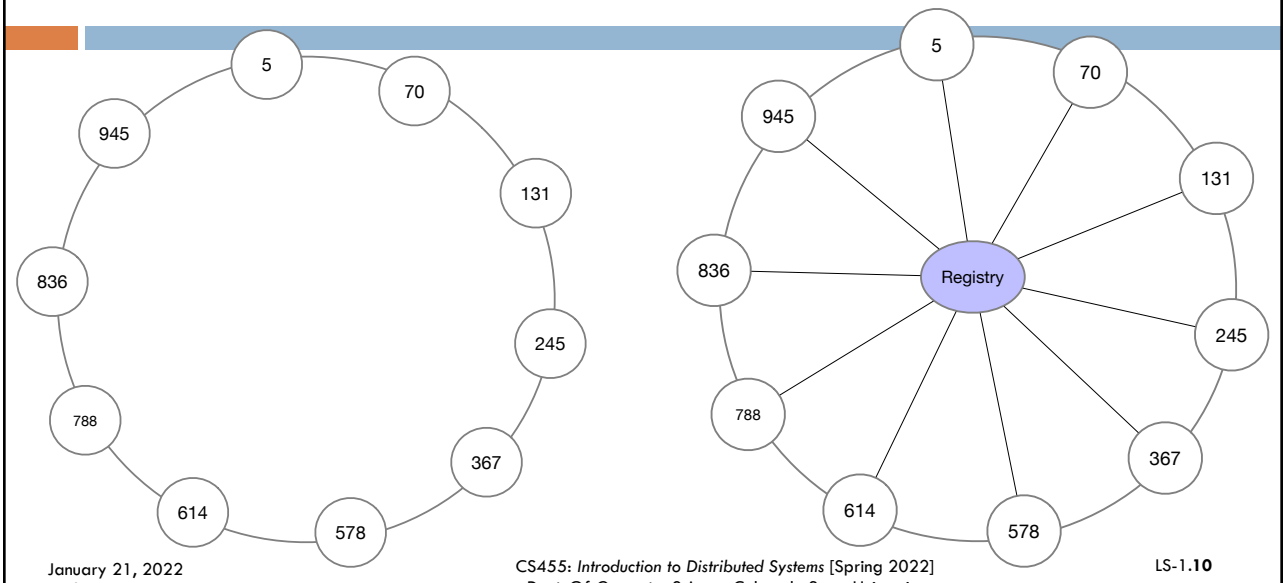


How is an overlay created?

- The Registry tells each messaging node who they should connect to
  - The messaging node does not know how nodes there are
  - It only knows about nodes that the registry tells it to connect to

9

What the topology looks like



10

Your programs will be working with two different data representations

- In memory: This is where you have your **data structures** such as lists, arrays, hash tables, trees, etc.
- Data that you will sending over the network
  - You do this as a self-contained **sequences of bytes**
  - Do references or pointers make sense here?
    - No!
    - So, the sequence-of-bytes representation will look VERY different from data structures in memory

So, we do need some translation between these representations

- Translation from in-memory to network-bound byte sequence
  - **Marshalling**
    - Also called serialization or encoding
- Translation from network-bound sequence to in-memory representation (i.e. restoration of data structure)
  - **Unmarshalling**
    - Also called deserialization or decoding

Marshalling and Unmarshalling

- Marshalling
  - **Pack** fields into a byte array
- Unmarshalling
  - **Unpack** byte array and *populate* fields that comprise the wire format message

Example: Data Structure [1/3]

```
public class WireFormatWidget {

    private int type;
    private long timestamp;
    private String identifier;
    private int tracker;

    ...

}
```

Example: Marshalling [2/3]

```
public byte[] getBytes() throws IOException {
        byte[] marshalledBytes = null;
        ByteArrayOutputStream baOutputStream = new ByteArrayOutputStream();
        DataOutputStream dout =
            new DataOutputStream(new BufferedOutputStream(baOutputStream));

        dout.writeInt(type);
        dout.writeLong(timestamp);

        byte[] identifierBytes = identifier.getBytes();
        int elementLength = identifierBytes.length;
        dout.writeInt(elementLength);
        dout.write(identifierBytes);

        dout.writeInt(tracker);

        dout.flush();
        marshalledBytes = baOutputStream.toByteArray();

        baOutputStream.close();
        dout.close();
        return marshalledBytes;
    }
```


Example: Unmarshalling [3/3]

```
public WireFormatWidget(byte[] marshalledBytes) throws IOException {
        ByteArrayInputStream baInputStream =
            new ByteArrayInputStream(marshalledBytes);
        DataInputStream din =
            new DataInputStream(new BufferedInputStream(baInputStream));

        type = din.readInt();
        timestamp = din.readLong();

        int identifierLength = din.readInt();
        byte[] identifierBytes = new byte[identifierLength];
        din.readFully(identifierBytes);

        identifier = new String(identifierBytes);

        tracker = din.readInt();

        baInputStream.close();
        din.close();
    }
```

How to send data

```
public class TCPSender {

    private Socket socket;
    private DataOutputStream dout;

    public TCPSender(Socket socket) throws IOException {
        this.socket = socket;
        dout = new DataOutputStream(socket.getOutputStream());
    }

    public void sendData(byte[] dataToSend) throws IOException {
        int dataLength = dataToSend.length;
        dout.writeInt(dataLength);
        dout.write(dataToSend, 0, dataLength);
         dout.flush();
    }
}
```


How to receive data [1/2]

```
public class TCPReceiver implements Runnable {

    private Socket socket;
    private DataInputStream din;


    public TCPReceiver(Socket socket) throws IOException {
        this.socket = socket;
        din = new DataInputStream(socket.getInputStream());
    }

    public void run() {
       ...
     }

}
```

How to receive data [2/2]

```
public void run() {

        int dataLength;
        while (socket != null) {
            try {
               dataLength = din.readInt();

               byte[] data = new byte[dataLength];
                din.readFully(data, 0, dataLength);

            } catch (SocketException se) {
                System.out.println(se.getMessage());
                break;
            }  catch (IOException ioe) {
                System.out.println(ioe.getMessage()) ;
                break;
            }
        }
}
```


A simple breakdown of classes

- cs455.overlay.wireformats
  - Protocol
  - Event [This is an interface with the getType() and getBytes() defined]
  - EventFactory [Singleton instance]
  - Register
  - Deregister
  - ConnectionsDirective
  - TaskInitiate
  - DataTraffic
  - TaskComplete
  - TaskSummaryRequest
  - TaskSummaryResponse

A simple breakdown of classes

- cs455.overlay.util
  - OverlayCreator
  - StatisticsCollectorAndDisplay

A simple breakdown of classes

- cs455.overlay.transport
  - TCPServerThread
  - TCPSender
  - TCPReceiverThread

A simple breakdown of classes

- cs455.overlay.node
  - Node [Interface with the onEvent(Event) method]
  - Registry
  - MessagingNode