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

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
  - Subversion, Git, Mercurial, etc.
  - Free hosted services: Github, Bitbucket, etc.
  - Run a Subversion server in a CS machine
  - Commit Often!
- Follow the guidelines from the beginning
  - build system, directory structure, etc.
- Follow the milestone plan as closely as possible

**Assignment**

Composition of the Overlay
Overlay topology set up with $C_R = 4$

The registry tells each messaging node who it should connect to via the `Messaging_Nodes_List`.

E.g., Shortest path from C to I.

Looking at another overlay topology that could be set up with $C_R = 4$

This topology has a partition. The assignment asks you to prevent this. Nodes A, B, C, D and E have no way of communicating with F, G, H, I, and J.

Avoiding network partitions

- Create a linear topology first, and then start making the required number of connections
  - A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ E $\rightarrow$ F $\rightarrow$ G $\rightarrow$ H $\rightarrow$ I $\rightarrow$ J
  - Starting off at this point ensures that partitions will not exist.

Marshaling and Unmarshaling

- **Marshaling**
  - Pack fields into a byte array.

- **Unmarshaling**
  - Unpack byte array and populate fields that comprise the wire format message (i.e., o).

Marshalling Example

```java
public class WireFormatWidget {
    private int type;
    private long timestamp;
    private String identifier;
    private int tracker;
    ...
}
```
Marshalling Example [2/3]

```java
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;
}
```

Marshalling Example [3/3]

```java
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?

```java
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?

```java
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]

```java
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());
            catch (IOException ioe) { System.out.println(ioe.getMessage());
        } catch (IOException ioe) { System.out.println(ioe.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
  - MessagingNodeList
  - LinkWeights
  - TaskInitiate
  - Message
  - TaskComplete
  - TaskSummaryRequest
  - TaskSummaryResponse
A simple breakdown of classes

- `cs455.overlay.dijkstra`
  - ShortestPath
  - RoutingCache

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