

(Q) Describe: ...

Dr. Barry Porter  
SCC.311: Remote Invocation  
Lecture starts at 15:00 BST

(Q) Describe: SCC 311 | Dr. Barry Porter

# 1 SCC 311 | Dr. Barry Porter

Housekeeping...

- Our lab stream starts this week
- Lab work is based on Java, using RMI as an example middleware
- We've posted an "RMI primer", for you to start this week, and will

shortly post the first two coursework stages

- We have a practice marking session for stage 1 in week 4, with a real marking session for stage 1 + stage 2 in week 7

(Q) Describe: SCC 311 | Dr. Barry Porter

## 2 SCC 311 | Dr. Barry Porter

### Overview

- Remote invocation in the general sense is just accessing any remote resource, using a particular protocol
- We'll look at general protocol variations, then focus on:
- RPC as a particular protocol
- Java RMI as one possible implementation of RPC
- REST as a different remote invocation protocol

(Q) Describe: SCC 311 | Dr. Barry Porter

### 3 SCC 311 | Dr. Barry Porter

#### Remote Invocation

This is the act of accessing a remote "thing" (procedure / object)

Achieved through the use of message passing over a network

Message exchange is handled through an agreed protocol

The semantics of this protocol depend on the context of your application

Different protocols will offer different reliability, scalability, and performance

The implementation of this protocol, and associated tools, is a communication middleware



(Q) Describe: SCC 311 | Dr. Barry Porter

## 4 SCC 311 | Dr. Barry Porter

### Protocol styles

- R: no value needs to be returned from the server / no confirmation is needed; client can "fire and forget" in a non-blocking way

- RR: typical "request-reply" protocol: if reply from server is lost in transit, request may be repeated by client

- RRA: server needs to know the client got its reply, e.g. to allow resources to be released or coordinate with other communications

Style Messages sent by

Client Server Client

R Request - -

RR Request Reply -

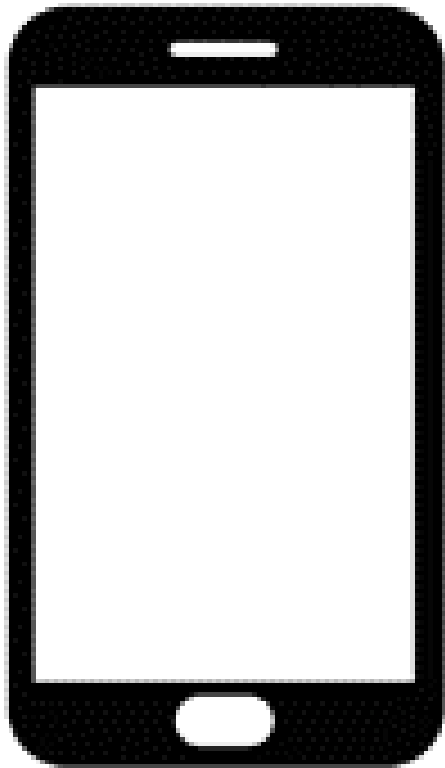
RRA Request Reply Acknowledgement

(Q) Describe: SCC 311 | Dr. Barry Porter

## 5 SCC 311 | Dr. Barry Porter

Let's build a text chat server...

```
socket = new Socket(serverName, serverPort);  
console = new DataInputStream(System.in);  
streamOut = new DataOutputStream(socket.getOutputStream());  
line = console.readLine();  
streamOut.writeUTF(line);  
streamOut.flush();  
console.close();  
streamOut.close();  
socket.close();  
client
```



(Q) Describe: SCC 311 | Dr. Barry Porter

## 6 SCC 311 | Dr. Barry Porter

Let's build a text chat server...

```
server = new ServerSocket(port);
socket = server.accept();
streamIn = new DataInputStream(new
BufferedInputStream(socket.getInputStream()));
boolean done = false;
while (!done) {
String line = streamIn.readUTF();
System.out.println(line);
}
socket.close();
streamIn.close();
server
```



(Q) Describe: SCC 311 | Dr. Barry Porter

## 7 SCC 311 | Dr. Barry Porter

An RPC chat server

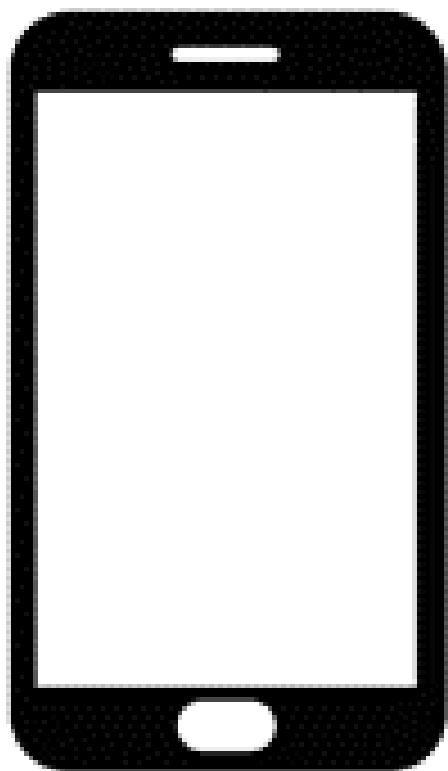
```
server = ChatServer.connect(serverName);  
line = console.readLine();  
server.sendMessage(line);  
void sendMessage(String line) {  
    System.out.println(line)  
}  
main(...)  
server = new ChatServer();  
server  
client
```

- Build from the top down, not bottom-up
- Directly call a remote procedure, rather

than handing all of the piping yourself







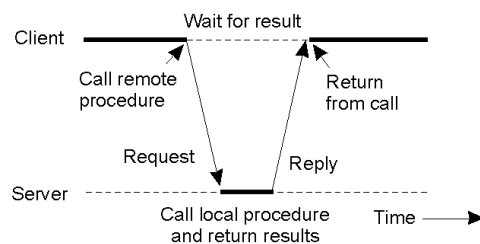
(Q) Describe: SCC 311 | Dr. Barry Porter

## 8 SCC 311 | Dr. Barry Porter

What is RPC?

- Remote Procedure Call
- In essence, the idea is:
- it's nice that we can define and call functions for local programs
- why not extend this to a distributed system, so that we can call an apparently

local function and that function call actually happens on a remote computer?

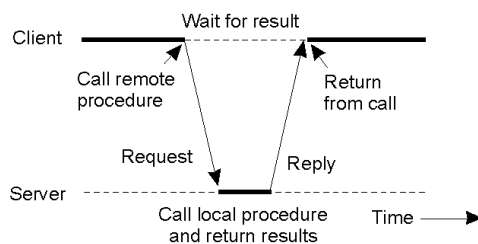


(Q) Describe: SCC 311 | Dr. Barry Porter

## 9 SCC 311 | Dr. Barry Porter

What is RPC?

- Remote Procedure Call
- One of the simplest forms of communication middleware
- Provides a high-level request-response mechanism to build distributed apps
- Usually synchronous, meaning that the client blocks while waiting for the procedure (cf. function) call to complete



(Q) Describe: SCC 311 | Dr. Barry Porter

## 10 SCC 311 | Dr. Barry Porter

### RPC and Middleware

- Examples: XML-RPC, JSON-RPC, SOAP
- Interaction between processes is done using defined interfaces

Application

RMI, RPC and indirect comm.

Request reply protocol

External Data Representation

Operating System

Middleware

layers

(Q) Describe: SCC 311 | Dr. Barry Porter



## 11 SCC 311 | Dr. Barry Porter

### Programming with Interfaces

- Separation of interface and implementation
- IDL (Interface Definition Language): programming language-independent

notation of parameters and types

- Client software does not need to know the details of the implementation,

cf. abstraction

- Important for platform and language independence
- Also important to support the evolution of software

myObject:myClass

myInterface

(Q) Describe: SCC 311 | Dr. Barry Porter

## 12 SCC 311 | Dr. Barry Porter

Implementing RPC

N.B. Proxies, stubs, dispatchers are generated automatically by an appropriate IDL compiler

Request

Reply

Communication modules

Object A

Remote

Object B

Proxy

Client Stub

Server Stub

Dispatcher

(Q) Describe: SCC 311 | Dr. Barry Porter

## 13 SCC 311 | Dr. Barry Porter

Key components: client side

- Proxies
- Masquerade as a local version of the remote interface
- Redirect calls to client stubs
- May perform other actions (see smart proxies)
- Client stub
- Carries out marshalling (flattening) of a call into a request message sent to

remote end

- Also unmarshalls returning replies
- One stub per interface procedure

(Q) Describe: SCC 311 | Dr. Barry Porter

## 14 SCC 311 | Dr. Barry Porter

Key components: server side

Dispatchers

Receive incoming messages and direct them to an appropriate server stub

Server stubs (skeletons)

Unmarshalls message and then invokes appropriate code body

Also marshals reply values and initiates transmission back to the client

(Q) Describe: SCC 311 | Dr. Barry Porter



## 15 SCC 311 | Dr. Barry Porter

What we get from this...

```
obj = RPCService.getRemoteObject(serverName);
```

`obj.callFunction("hi")` After the initial acquisition of this object, from the middleware, use of the object then looks exactly like a normal local call – neat!

(Q) Describe: SCC 311 | Dr. Barry Porter

## 16 SCC 311 | Dr. Barry Porter

...but there might be a dragon or two...

- A lot of our current understanding on the theory of RPC comes from the original researchers and designers of the Java language<sup>1</sup>

- Remote calls have different latency to local calls
- Memory access models are different if we pass references around
- Partial failures are possible

[1] J. Waldo, G. Wyant, A. Wollrath, and S. Kendall. A note on distributed computing. Technical report, 1994

(Q) Describe: SCC 311 | Dr. Barry Porter

## 17 SCC 311 | Dr. Barry Porter

### Protocol guarantees

- What delivery guarantees does the exchange protocol give?
- Referred to as ‘call semantics’ in the book
- Local procedure calls = ‘exactly once’ guarantee
- But for RPC?
- Different guarantee types are possible depending on the protocol implementation

(Q) Describe: SCC 311 | Dr. Barry Porter

## 18 SCC 311 | Dr. Barry Porter

Focus on the underlying protocol

Let's focus on the communications module for RPC which will provide a protocol that mimics the semantics of a local call

- For the sake of this discussion let's assume the underlying protocol is UDP
- (note RPC is more commonly implemented with TCP in modern middleware)
- Problems
- Request message may get lost
- Reply message may get lost
- Client may crash
- Server may crash

(Q) Describe: SCC 311 | Dr. Barry Porter



## 19 SCC 311 | Dr. Barry Porter

Lightweight protocol semantics

Maybe semantics

Send request to server; which sends back a reply

No guarantees at all if anything goes wrong

At least once semantics

Sends message and if reply not received after a given time, the message is re-sent (failure assumed after n re-sends)

- Will guarantee the call is made “at least once”, but possibly multiple times
- Ideal for idempotent operations (i.e. same effect)

(Q) Describe: SCC 311 | Dr. Barry Porter

## 20 SCC 311 | Dr. Barry Porter

Lightweight protocol semantics

At most once semantics Client Server Call

Re-send

Log

results

Execute code

Detect

duplicate

Request (#1473)

Request (#1473)

Reply (#1473)

Reply (#1473)

(Q) Describe: SCC 311 | Dr. Barry Porter

## 21 SCC 311 | Dr. Barry Porter

Lightweight protocol semantics

- Local procedure calls have an even stronger exactly once semantic
- So far, for RPC, we have:

Semantics Fault tolerance measures

Retransmit request Duplicate filtering Re-execute procedure or  
retransmit reply

Maybe No No N/A

At least once Yes No Re-execute procedure

At most once Yes Yes Retransmit reply

(Q) Describe: SCC 311 | Dr. Barry Porter

## 22 SCC 311 | Dr. Barry Porter

RPC protocol semantics

- Exactly once semantics
- In this case the procedure will be carried out once (completely) or not at all

(operation aborted)

- This builds on the "at most once" protocol, but also adds support for atomicity

- We'll cover this topic in our lectures on fault tolerance and dependability

(Q) Describe: SCC 311 | Dr. Barry Porter



## 23 SCC 311 | Dr. Barry Porter

From RPC to RMI

- Remote Method Invocation (RMI) is the Java-specific built-in middleware technology which implements the RPC concept, integrating it seamlessly with the Java language
- RMI implements remote objects in an almost transparent way: you can pass object references into remote function calls to create complex object reference graphs which span continents
- The "almost" part is that RMI chooses to expose a new class of exceptions on all remote calls, via RemoteException, which must be caught in the caller

(Q) Describe: SCC 311 | Dr. Barry Porter

## 24 SCC 311 | Dr. Barry Porter

### RMI Basics

Example: You need to develop a Java mobile app to access the Google Maps Service

Your Mobile @ Lancaster

Google Servers @ Dublin





(Q) Describe: SCC 311 | Dr. Barry Porter

## 25 SCC 311 | Dr. Barry Porter

### RMI Basics

- RMI allows one Java object to call methods on another Java object in a different JVM

- The intention is to make distributed programming as easy as standard

Java programming

Local

Object

Remote

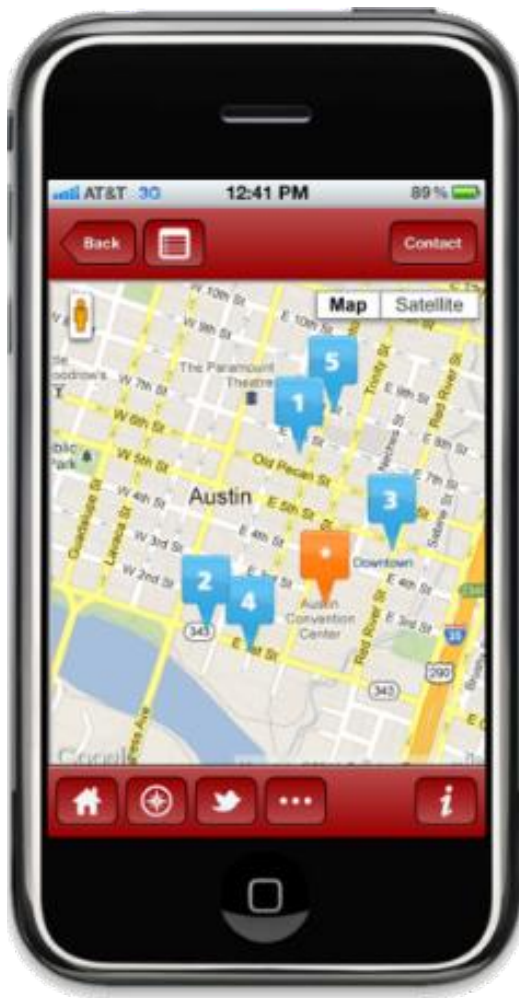
Object

Client JVM

Server JVM

Method parameters

Result or exception



(Q) Describe: SCC 311 | Dr. Barry Porter



## 26 SCC 311 | Dr. Barry Porter

### RMI Basics

- RMI uses interfaces to specify a remote object: we define an interface which extends from `java.rmi.Remote`
- We define a class which implements this interface; we can then instantiate an object from this class which can be advertised for remote access
- A client program only needs access to the interface type (not the class), and can then acquire a reference to the remote object of this type via the RMI middleware service

(Q) Describe: SCC 311 | Dr. Barry Porter

## 27 SCC 311 | Dr. Barry Porter

### RMI Basics

- The advertisement (at the server) and lookup (at the client) of remote objects is done through a special service called the RMI registry
- We execute the registry at the command line using the command `rmiregistry`
- This service associates names with object references
- The registry often runs on the same host as a server system, but does not need to
- A server and client both need to talk to the same registry service, on the same host, to advertise and look up a named object

(Q) Describe: SCC 311 | Dr. Barry Porter

## 28 SCC 311 | Dr. Barry Porter

RMI Basics

RMIRegistry

Server

```
naming.rebind("rmi://www.google.com/Map  
Service", RemoteObjectReference)
```

Client

```
naming.lookup("rmi://www.google.com  
/MapService")
```

Interface Remote Object

Client Program Server Program

Step1: Bind to name Step2: Lookup name

Step3: Method Invocation

(Q) Describe: SCC 311 | Dr. Barry Porter

## 29 SCC 311 | Dr. Barry Porter

REST

- ...and now for something different!
- So far we've covered remote procedure call and RMI
- These are not the only forms of remote invocation

(Q) Describe: SCC 311 | Dr. Barry Porter



## 30 SCC 311 | Dr. Barry Porter

### REST

Representational State Transfer (REST) is a set of resource-oriented architectural principles

- RPC/RMI are operation-/transaction-oriented
- RPC: readStudent(1234)
- REST: GET /students/1234
- Properties:
- Every resource is addressable using a Uniform Resource Identifier (URI)
- To change the state of the system: transition resources
- HTTP-based: basic HTTP verbs and status codes (universal interface)
- Self-descriptive: responses include description and next step(s) links
- Stateless: data required to transition between states is in request

(Q) Describe: SCC 311 | Dr. Barry Porter

## 31 SCC 311 | Dr. Barry Porter

HTTP Methods (verbs)

The universal/uniform interface of REST

•<https://www.restapitutorial.com/lessons/httpmethods.html>

Verb CRUD Safe? Idempotent?

POST Create

GET Read

PUT Update/Replace

PATCH Modify

DELETE Delete

(Q) Describe: SCC 311 | Dr. Barry Porter

## 32 SCC 311 | Dr. Barry Porter

HTTP Methods (verbs)

Read specific student GET /students/1234

- Read all students

GET /students

- Create a student

POST /students

- Update specific student

PUT /students/1234

- Delete specific student

DELETE /students/1234

(Q) Describe: SCC 311 | Dr. Barry Porter

## 33 SCC 311 | Dr. Barry Porter

HTTP Methods (verbs)

- Read specific student email address(es)

GET /students/1234/email

- Update specific student email address

PUT /students/1234/email/1

- Delete specific student email address

DELETE /students/1234/email/2

(Q) Describe: SCC 311 | Dr. Barry Porter



## 34 SCC 311 | Dr. Barry Porter

### HTTP Content Types

- The content type used by each verb, in both the request and the response message, is configurable
- This is done using headers which can be included in the request and response
- Common content types are text/html, text/xml, text/json, image/jpeg, etc.
- The sender of a request can also specify the content types that it is expecting and can process, as part of its request message

(Q) Describe: SCC 311 | Dr. Barry Porter

## 35 SCC 311 | Dr. Barry Porter

### HTTP Status Codes

1xx Informational

2xx Success

200 Resource was read, updated, or deleted

201 Resource was created

3xx Redirection

301 Resource has permanently moved to a new URI

4xx Client Error

400 Bad request

403 Not authorized to perform this action

404 Resource not found

5xx Server Error

- <https://www.restapitutorial.com/httpstatuscodes.html>

(Q) Describe: SCC 311 | Dr. Barry Porter

## 36 SCC 311 | Dr. Barry Porter

### HTTP Semantics

The most obvious feature of REST is that servers do not hold any per- client state

- Instead, the client sends the current state with every request (this is what cookies are)

- This allows servers to consume fewer memory resources, and also allows a client request to hit any server, because the request carries all of the state

(Q) Describe: SCC 311 | Dr. Barry Porter

## 37 SCC 311 | Dr. Barry Porter

### HTTP Semantics

- REST also has a general assumption of idempotence, meaning that an operation will only ever have a single effect (repeating the same operation, with the same state, has no effect)

- Keeps servers slender
- Very useful in distributed environments
- Multiple ‘servers’
- Unreliable network

(Q) Describe: SCC 311 | Dr. Barry Porter



## 38 SCC 311 | Dr. Barry Porter

### HTTP Summary

Because it is a text-based format, is very simple, and is not language-specific, HTTP has become a kind of general interoperability protocol

- A wide range of other protocols have been designed which can operate on top of HTTP, taking advantage of this common carrier
- Linking back to RPC, the Web Services framework is a language-independent RPC solution which is built on top of HTTP

(Q) Describe: SCC 311 | Dr. Barry Porter

## 39 SCC 311 | Dr. Barry Porter

Further reading

- CDKB, ch 5
- also optionally ch 4 for background
- TvS, pp. 145-158, 68-98, 99-134
- REST API Tutorial: <https://www.restapitutorial.com/>

(Q) Describe: ...

(Q) Describe: ...