INTERPROCESS COMMUNICATION

Reference: George Coulouris, Jean Dollimore and Tim Kindberg, “Distributed Systems Concepts and Design”, Fifth Edition, Pearson Education, 2012

Topics

- Client-server Communication
- Group Communication

Client-Server Communication

- The client-server communication is designed to support the message exchange in typical client-server interactions.
- In the normal case, request-reply communication is synchronous because the client process blocks until the reply arrives from the server.
- Asynchronous request-reply communication is an alternative that is useful where clients can afford to retrieve replies later.

Client-Server Communication

- Often built over UDP datagrams
- Client-server protocol consists of request/response pairs, hence no acknowledgements at transport layer are necessary
- Avoidance of connection establishment overhead
- No need for flow control due to small amounts of data are transferred

Client-Server Communication

- The request-reply protocol was based on a trio of communication primitives: doOperation, getRequest, and sendReply shown in Figure 12.

Client-Server Communication

- The request-reply protocol is shown in Figure 12.

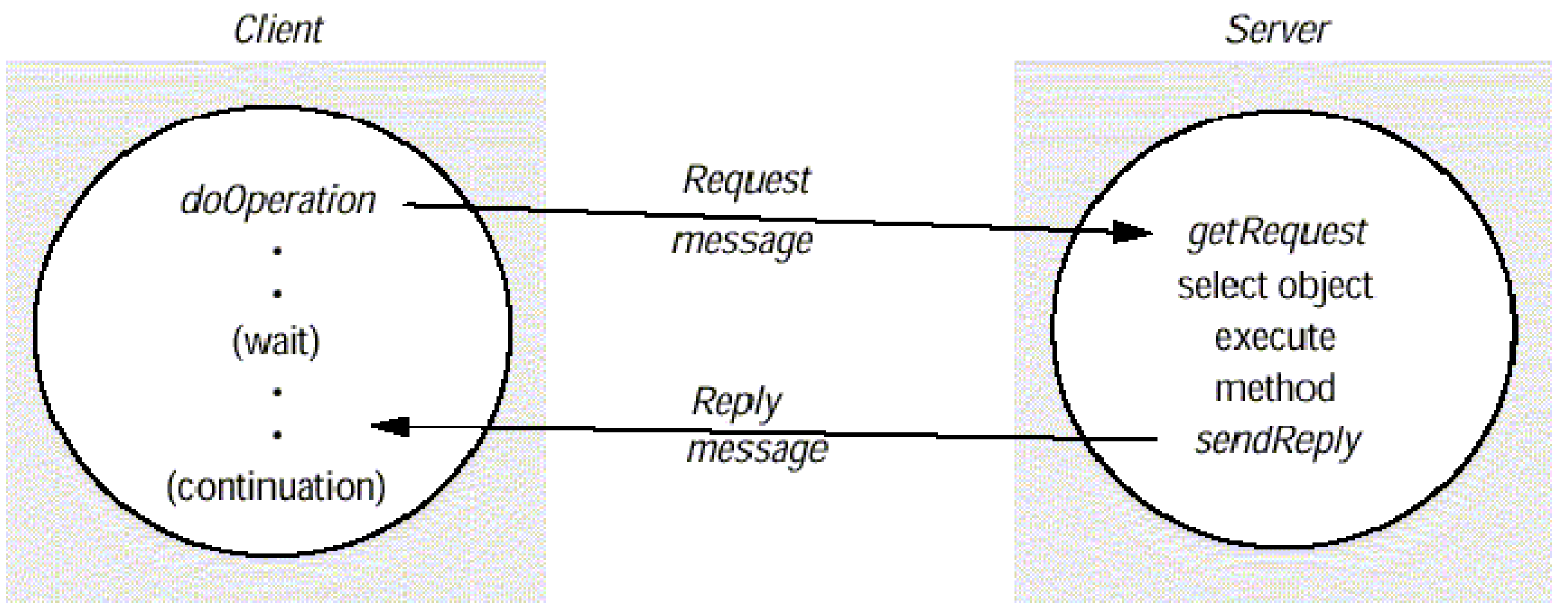


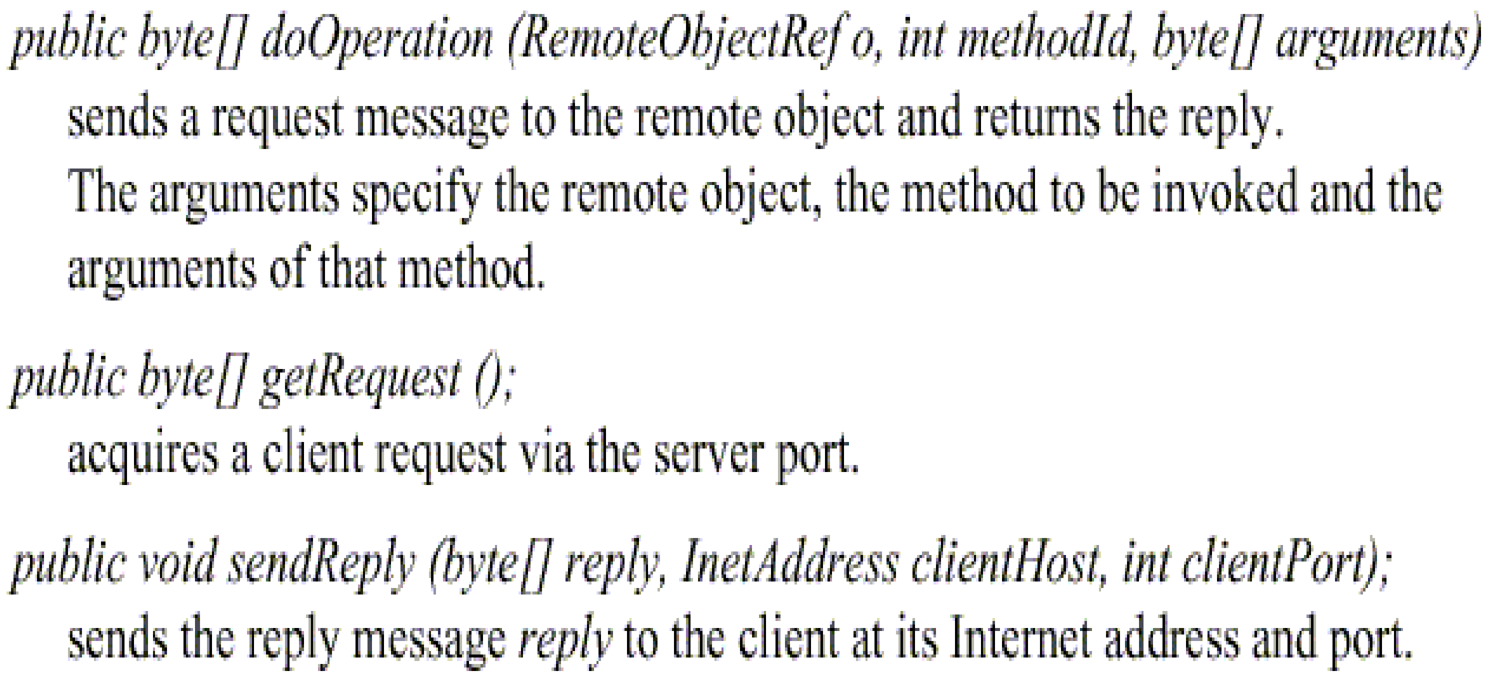
Figure 12. Request-reply communication

Couloris,Dollimore and Kindberg Distributed Systems: Concepts & Design Edn. 4 , Pearson Education 2005

Client-Server Communication

- The designed request-reply protocol matches requests to replies.
- If UDP datagrams are used, the delivery guarantees must be provided by the request-reply protocol, which may use the server reply message as an acknowledgement of the client request message.
- Figure 13 outlines the three communication primitives.

Client-Server Communication



public byte[] doOperation (RemoteObjectRef o, int methodId, byte[] arguments)
sends a request message to the remote object and returns the reply.
The arguments specify the remote object, the method to be invoked and the arguments of that method.

public byte[] getRequest ();
acquires a client request via the server port.

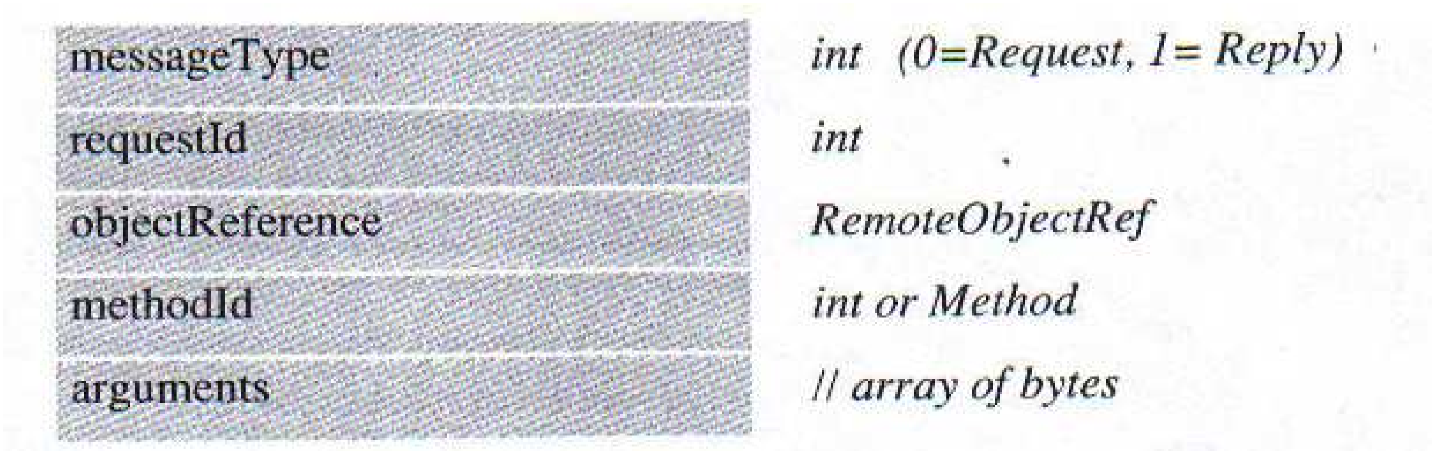
public void sendReply (byte[] reply, InetAddress clientHost, int clientPort);
sends the reply message *reply* to the client at its Internet address and port.

Figure 13. Operations of the request-reply protocol

Couloris,Dollimore and Kindberg Distributed Systems: Concepts & Design Edn. 4 , Pearson Education 2005

Client-Server Communication

- The information to be transmitted in a request message or a reply message is shown in Figure 14.



messageType	*int (0=Request, 1= Reply)*
requestId	*int*
objectReference	*RemoteObjectRef*
methodId	*int or Method*
arguments	*// array of bytes*

Figure 14. Request-reply message structure

Couloris,Dollimore and Kindberg Distributed Systems: Concepts & Design Edn. 4 , Pearson Education 2005

Client-Server Communication

- In a protocol message
  - The first field indicates whether the message is a request or a reply message.
  - The second field request id contains a message identifier.
  - The third field is a remote object reference .
  - The forth field is an identifier for the method to be invoked.

Client-Server Communication

- Message identifier
  - A message identifier consists of two parts:
    - A requestId, which is taken from an increasing sequence of integers by the sending process
    - An identifier for the sender process, for example its port and Internet address.

Client-Server Communication

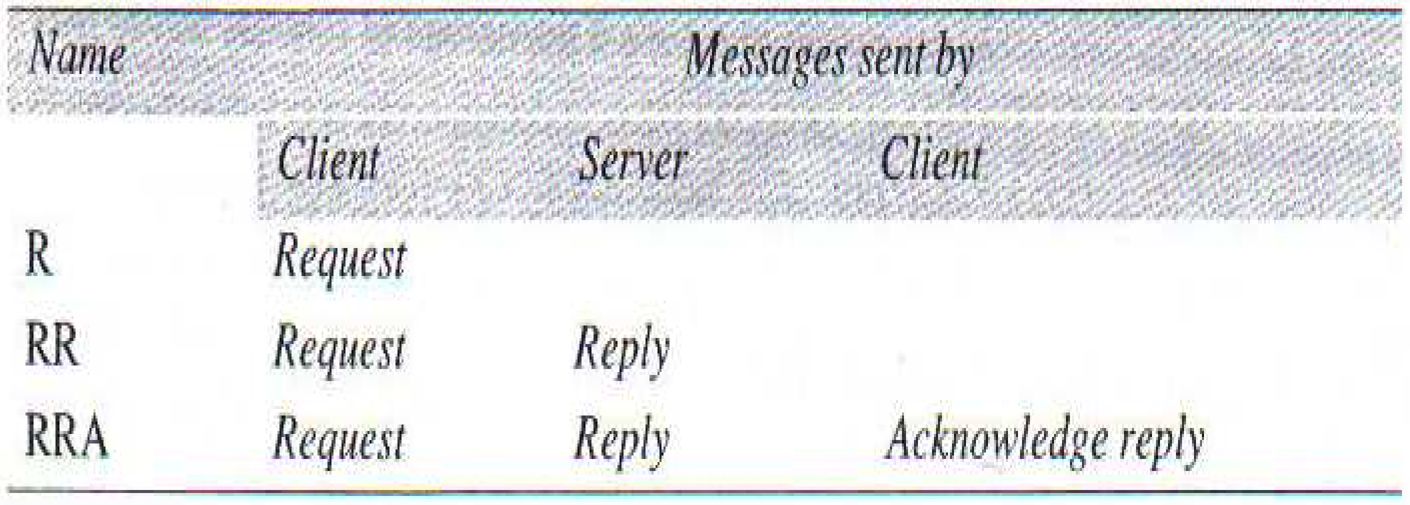
- Failure model of the request-reply protocol
  - If the three primitive doOperation, getRequest, and sendReply are implemented over UDP datagram, they have the same communication failures.
    - Omission failure
    - Messages are not guaranteed to be delivered in sender order.

Client-Server Communication

- RPC exchange protocols
  - Three protocols are used for implementing various types of RPC.
    - The request (R) protocol.
    - The request-reply (RR) protocol.
    - The request-reply-acknowledge (RRA) protocol.

    (Figure 15)

Client-Server Communication



Name	Messages sent by		
	Client	Server	Client
R	Request		
RR	Request	Reply	
RRA	Request	Reply	Acknowledge reply

Figure 15. RPC exchange protocols

Client-Server Communication

- In the R protocol, a single request message is sent by the client to the server.
- The R protocol may be used when there is no value to be returned from the remote method.
- The RR protocol is useful for most client-server exchanges because it is based on request-reply protocol.
- RRA protocol is based on the exchange of three messages: request-reply-acknowledge reply.

Client-Server Communication

- HTTP: an example of a request-reply protocol
  - HTTP is a request-reply protocol for the exchange of network resources between web clients and web servers.

Client-Server Communication

- HTTP protocol steps are:
  - Connection establishment between client and server at the default server port or at a port specified in the URL
  - client sends a request
  - server sends a reply
  - connection closure

Client-Server Communication

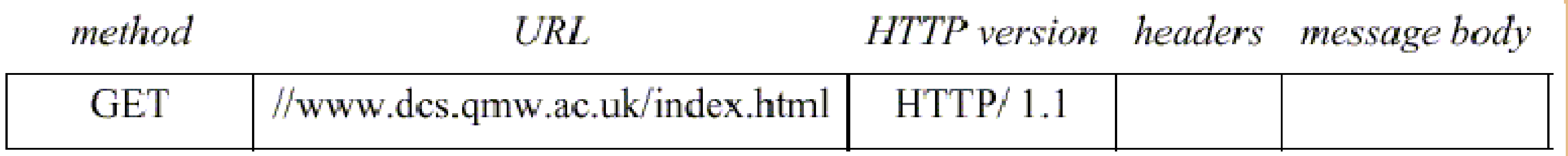
- HTTP 1.1 uses persistent connections.
  - Persistent connections are connections that remains open over a series of request-reply exchanges between client and server.
- Resources can have MIME-like structures in arguments and results.

Client-Server Communication

- A Mime type specifies a type and a subtype, for example:
  - text/plain
  - text/html
  - image/gif
  - image/jpeg

Client-Server Communication

- HTTP methods
  - GET
    - Requests the resource, identified by URL as argument.
    - If the URL refers to data, then the web server replies by returning the data
    - If the URL refers to a program, then the web server runs the program and returns the output to the client.



method	*URL*	*HTTP version*	*headers*	*message body*
GET	//www.dcs.qmw.ac.uk/index.html	HTTP/ 1.1		

Figure 16. HTTP request message

Client-Server Communication

- HEAD
    - This method is similar to GET, but only meta data on resource is returned (like date of last modification, type, and size)

Client-Server Communication

- POST
  - ❖ Specifies the URL of a resource (for instance, a server program) that can deal with the data supplied with the request.
  - ❖ This method is designed to deal with:
    - Providing a block of data to a data-handling process
    - Posting a message to a bulletin board, mailing list or news group.
    - Extending a dataset with an append operation

Client-Server Communication

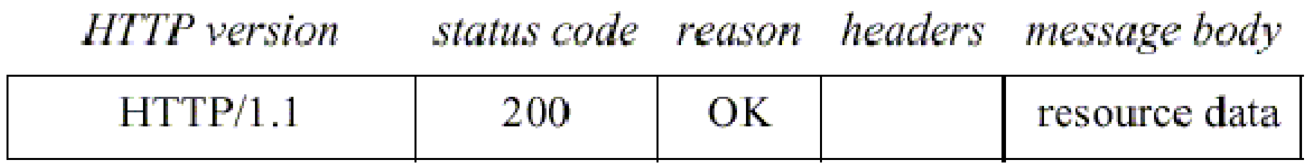
- PUT
  - Supplied data to be stored in the given URL as its identifier.
- DELETE
  - The server deletes an identified resource by the given URL on the server.
- OPTIONS
  - A server supplies the client with a list of methods.
  - It allows to be applied to the given URL

Client-Server Communication

- TRACE
  - The server sends back the request message

Client-Server Communication

- A reply message specifies
  - The protocol version
  - A status code
  - Reason
  - Some headers
  - An optional message body



HTTP version	*status code*	*reason*	*headers*	*message body*
HTTP/1.1	200	OK		resource data

Figure 17. HTTP reply message

Group Communication

- The pairwise exchange of messages is not the best model for communication from one process to a group of other processes.
- A multicast operation is more appropriate.
- Multicast operation is an operation that sends a single message from one process to each of the members of a group of processes.

Group Communication

- The simplest way of multicasting, provides no guarantees about message delivery or ordering.
- Multicasting has the following characteristics:
  - ➢ Fault tolerance based on replicated services
    - ❖ A replicated service consists of a group of servers.

Group Communication

- Client requests are multicast to all the members of the group, each of which performs an identical operation.

- Finding the discovery servers in spontaneous networking
  - Multicast messages can be used by servers and clients to locate available discovery services in order to register their interfaces or to look up the interfaces of other services in the distributed system.

Group Communication

- ➢ Better performance through replicated data
    - ❖ Data are replicated to increase the performance of a service.
- ➢ Propagation of event notifications
    - ❖ Multicast to a group may be used to notify processes when something happens.

Group Communication

- IP multicast
  - IP multicast is built on top of the Internet protocol, IP.
  - IP multicast allows the sender to transmit a single IP packet to a multicast group.
  - A multicast group is specified by class D IP address for which first 4 bits are 1110 in IPv4.

Group Communication

- The membership of a multicast group is dynamic.
- A computer belongs to a multicast group if one or more processes have sockets that belong to the multicast group.

Group Communication

- The following details are specific to IPv4:
  - Multicast IP routers
    - IP packets can be multicast both on local network and on the wider Internet.
    - Local multicast uses local network such as Ethernet.
    - To limit the distance of propagation of a multicast datagram, the sender can specify the number of routers it is allowed to pass- called the time to live, or TTL for short.

Group Communication

❖ Multicast address allocation

- Multicast addressing may be permanent or temporary.
- Permanent groups exist even when there are no members.
- Multicast addressing by temporary groups must be created before use and cease to exit when all members have left.
- The session directory (sd) program can be used to start or join a multicast session.
- session directory provides a tool with an interactive interface that allows users to browse advertised multicast sessions and to advertise their own session, specifying the time and duration.

Group Communication

- Java API to IP multicast
  - The Java API provides a datagram interface to IP multicast through the class MulticastSocket, which is a subset of DatagramSocket with the additional capability of being able to join multicast groups.
  - The class MulticastSocket provides two alternative constructors , allowing socket to be creative to use either a specified local port, or any free local port.

  (Figure 18)

Group Communication

```
import java.net.*;
import java.io.*;
public class MulticastPeer{
   public static void main(String args[]){
    // args give message contents and destination multicast group (e.g. "228.5.6.7")
    MulticastSocket s =null;
    try {
        InetAddress group = InetAddress.getByName(args[1]);
        s = new MulticastSocket(6789);
        s.joinGroup(group);
        byte [] m = args[0].getBytes();
        DatagramPacket messageOut = new DatagramPacket(m, m.length, group, 6789);
        s.send(messageOut);
        byte[] buffer = new byte[1000];
        for(int i=0; i< 3;i++) {                         // get messages from others in group
             DatagramPacket messageIn = new DatagramPacket(buffer, buffer.length);
             s.receive(messageIn);
             System.out.println("Received:" + new String(messageIn.getData()));
             }
        s.leaveGroup(group);
    }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
    }catch (IOException e){System.out.println("IO: " + e.getMessage());
    }finally {if(s != null) s.close();}
  }

}
```

Figure 18. Multicast peer joins a group and sends and receives datagrams

Group Communication

- ❖ A process can join a multicast group with a given multicast address by invoking the joinGroup method of its MulticastSocket.
- ❖ A process can leave a specified group by invoking the leaveGroup method of its MulticastSocket.
- ❖ The Java API allows the TTL to be set for a MulticastSocket by means of the setTimeToLive method. The default is 1, allowing the multicast to propagate only on the local network.

Thank You