CSCI 5105

Instructor: Abhishek Chandra

Today

- Communication in Distributed Systems
  - Overview
  - Types
- Remote Procedure Calls (RPC)
- Remote Method Invocation (RMI)

Communication

- How do program modules/processes communicate on a single machine?

Communication in Distributed Systems

- "Distributed" processes
  - Located on different machines
- Need communication mechanisms
- Goal: Hide distributed nature as far as possible

Communication in Distributed Systems

- Networking primitives and protocols (e.g.: TCP/IP)
- Advanced communication models: Built on networking primitives
  - Remote Procedure Calls (RPC)
  - Remote Method Invocation (RMI)
  - Messages
  - Multicast

5

Types of Communication

- Defined by two main properties:
  - Persistence
  - Synchronization

6

Persistence

- Persistent communication
  - Messages are stored until receiver is ready
  - Sender/receiver don't have to be up at the same time
- Transient communication
  - Message is stored only so long as both sending/receiving applications are executing
  - Discard message if it can't be delivered to receiver

7

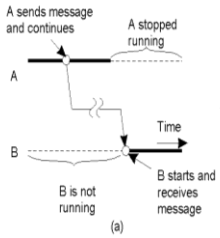
Synchronization

- Synchronous communication
  - Sender blocks until message is delivered to receiver
  - Variant: block until receiver processes the message
- Asynchronous communication
  - Sender continues immediately after it has submitted the message
- Several combinations of persistence and synchronization

8

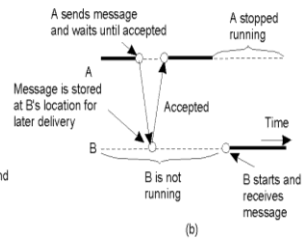
Persistence-Synchronization Combinations

Persistent Asynchronous communication



Example: Email

Persistent Synchronous communication

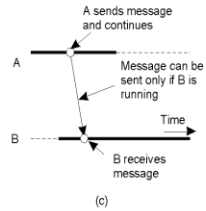


Example: Message Queuing

9

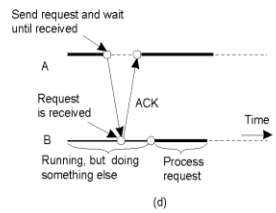
Persistence-Synchronization Combinations

Transient Asynchronous communication



Example: UDP, One-way RPC

Receipt-based Transient Synchronous Communication

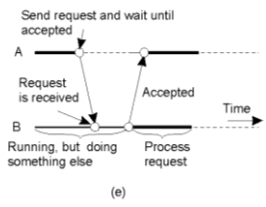


Example: Message-passing

10

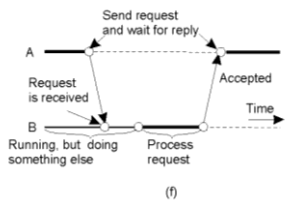
Persistence-Synchronization Combinations

Delivery-based Transient Synchronous Communication



Example: Asynchronous RPC

Response-based Transient Synchronous Communication



Example: RPC, RMI

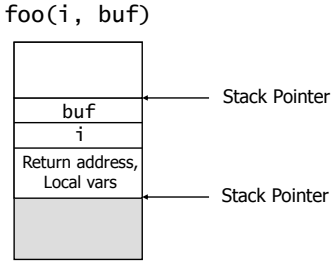
11

Remote Procedure Calls (RPC)

- Goal: Make distributed computation look like centralized computation
- Idea: Allow processes to call procedures on other machines
  - Make it appear like normal procedure calls

12

Local Procedure Calls



13

RPC Operation

- Challenges:
  - Hide details of communication
  - Pass parameters transparently
- Stubs
  - Hide communication details
  - Client and server stubs
- Marshalling
  - Flattening and parameter passing

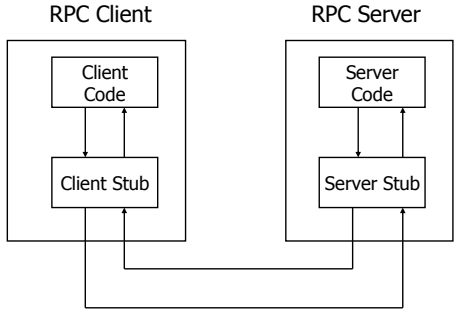
14

Stubs

- Code that communicates with the remote side
- Client stub:
  - Converts function call to remote communication
  - Passes parameters to server machine
  - Receives results
- Server stub:
  - Receives parameters and request from client
  - Calls the desired server function
  - Returns results to client

15

Basic RPC Operation



16

Parameter Passing: Local Procedures

- Pass-by-value
  - Original variable is not modified
  - E.g.: integers, chars
- Pass-by-reference
  - Passing a pointer
  - Value may be changed
  - E.g.: Arrays
- Pass-by-copy/restore
  - Copy is modified and overwritten to the original
  - E.g.: in-out parameters in Ada

17

Parameter Passing: RPC

- Pass-by-value
  - Send the value in standard format
- Pass-by-reference
  - Can we pass pointers?
- What about complex data structures (linked lists, trees, graphs)?

18

Marshalling

- Converting parameters into a byte stream
- Problems:
  - Heterogeneous data formats: Big-endian vs. little-endian
  - Type of parameter passing: by-value vs. by-reference

19

Heterogeneous Data Formats

- Use a standard data format
- Examples: Network byte order, XDR (Extended Data Representation)
- Decide on a protocol for parameter ordering

20

Stub Generation

- Most stubs are similar in functionality
  - Handle communication and marshalling
  - Differences are in the main server-client code
- Application needs to know only stub interface
- Interface Definition Language (IDL)
  - Allows interface specification
  - IDL compiler generates the stubs automatically

21

Binding

- How does the client stub find the server stub?
  - Needs to know remote IP address/port no.
- Port mapper
  - Daemon on server machine maintaining server bindings
  - Listens on a well-known port
- Server stub registers its port no. and service name with portmapper
  - Client gets this binding by querying portmapper
- Server's IP address can be obtained from a directory service

22

Synchronous RPC

- RPC Performed in a synchronous manner
  - Client blocks until results come back
- What if client wants to do something else?

23

RPC Variants

- Asynchronous RPC
  - Server sends ACK as soon as request is received
  - Executes procedure later
- Deferred synchronous RPC
  - Use two asynchronous RPCs
  - Server sends reply via second asynchronous RPC
  - Callback to the client
- One-way RPC
  - Client does not even wait for an ACK from the server

24

Multicast RPC

- RPC sent to multiple servers
  - Could use multiple concurrent one-way or asynchronous RPCs
- Why use multicast RPC?
- How to handle responses?
  - First one or majority response
  - Aggregate multiple responses

25

Remote Method Invocation (RMI)

- RPCs applied to distributed objects
- Class: object-oriented abstraction
- Object: instance of class
  - Encapsulates data
  - Exports methods: operations on data
  - Separation between interface and implementation

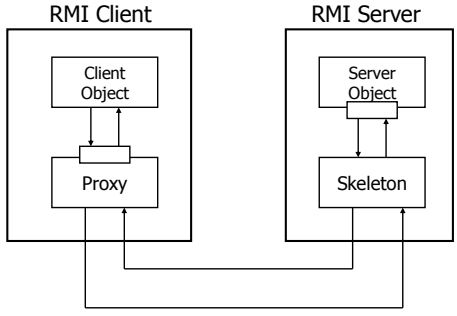
26

Distributed Objects

- Object resides on one machine, interfaces reside on other machines
- Remote object: State on different machine
- Local object: State on the same machine
- RMIs allow invoking methods of remote objects
  - Use proxies, skeletons, binding
  - Allow passing of object references as parameters

27

Basic RMI Operation



28

Proxies and Skeletons

- Proxy: client stub
  - Maintains server ID, endpoint, object ID
  - Does parameter marshalling
  - In practice, can be downloaded/constructed on the fly
- Skeleton: server stub
  - Does demarshalling and passes parameters to server
  - Sends result to proxy

29

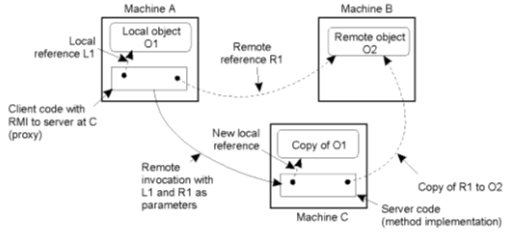
Binding a Client to an Object

- Loading a proxy in client address space
- Implicit binding:
  - Bound automatically on object reference resolution
- Explicit binding:
  - Client has to first bind object
  - Call method after binding

30

Parameter Passing

- Less restrictive than RPCs
  - Supports system-wide object references
  - Copy local objects, pass references of remote objects



31