Transport layer protocols

Lecture 16:

Operating Systems and Networks

Behzad Bordbar

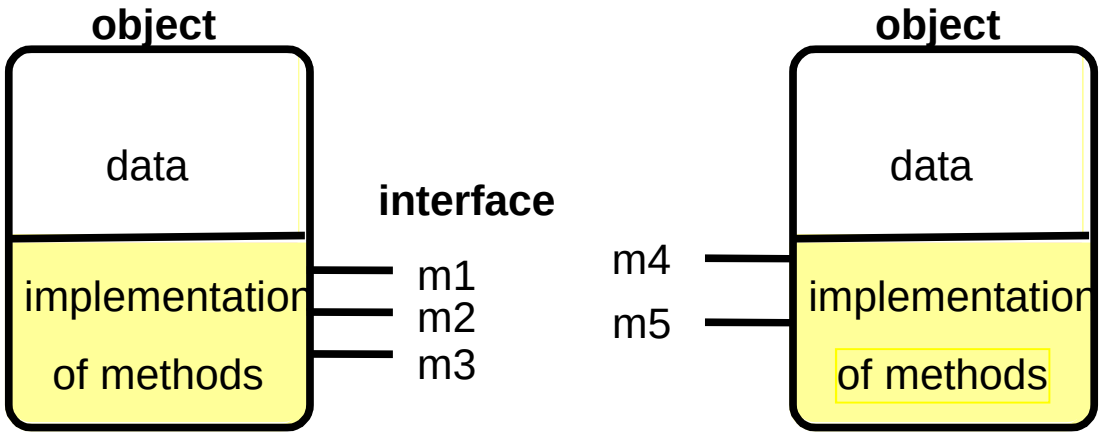
Recap

- Interprocess communication
- ❑ Synchronous and Asynchronous communication
- ❑ use of Socket for comm.
- ❑ various types of failure
- ❑ “no global time”
- ❑ Synchronous and Asynchronous interaction model
- ❑ Java API for UDP ….

Overview

- Distributed applications programming
  - distributed objects model
  - RMI, invocation semantics
  - RPC
  - events and notifications
- Products
  - Java RMI, CORBA, DCOM
  - Sun RPC

Objects

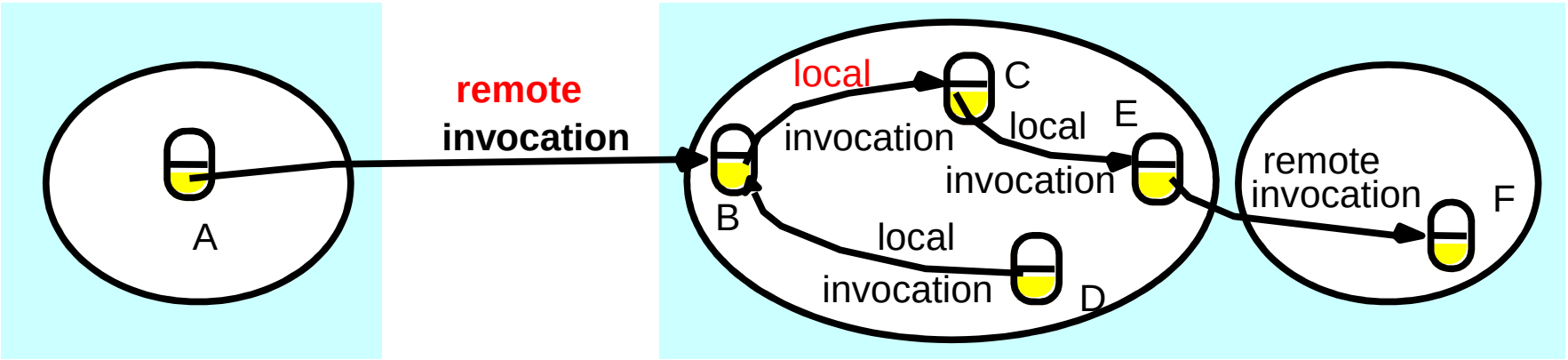


- Objects = Data (attributes) + Operations (methods)
  - encapsulating Data and Methods
  - State of Objects: value of its attributes
- Interact via interfaces:
  - define types of arguments and exceptions of methods

The object (local) model

- Programs:
  - a collection of objects
- Interfaces
  - the only means to access data, make them remote?
- Actions
  - via method invocation
  - interaction, chains of invocations
  - may lead to exceptions, specified in interfaces
- Garbage collection
  - reduced effort, error-free (Java, not C++)

In contrast: distributed object model



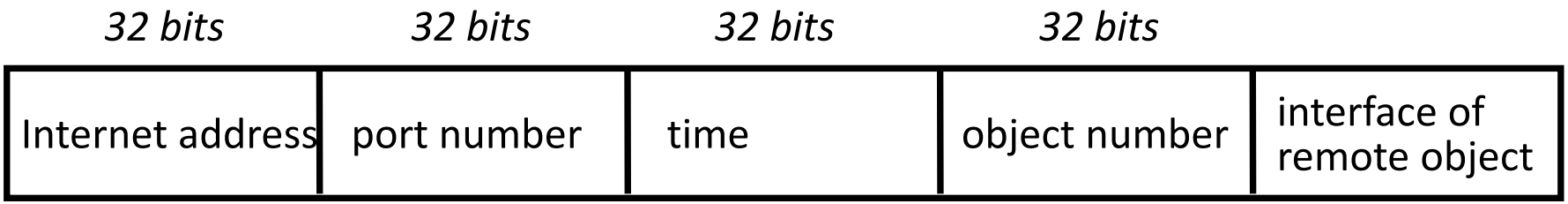
- Objects distributed (client-server models)
- Extend with
  - Remote object reference
  - Remote interfaces
  - Remote Method Invocation (RMI)

Remote object reference

- Object references
  - used to access objects which live in processes
  - can be passed as arguments, stored in variables,...
- Remote object references
  - object identifiers in a distributed system
  - must be unique in space and time
  - error returned if accessing a deleted object
  - can allow relocation (as in CORBA)

Remote object reference

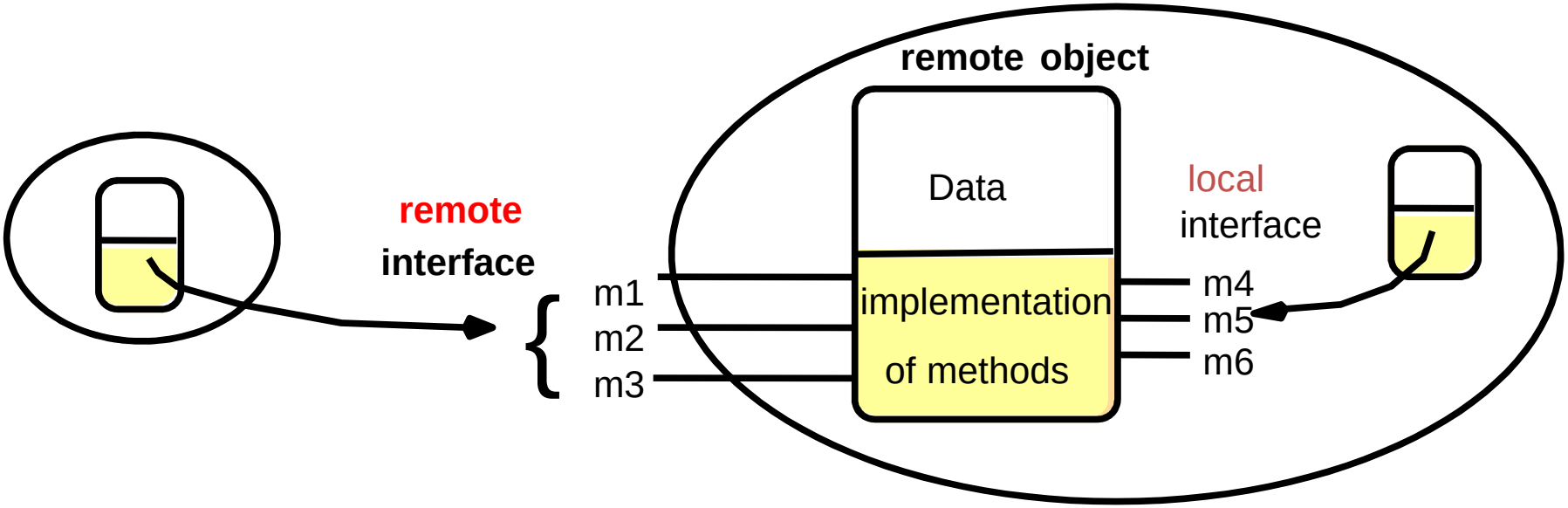
- Constructing unique remote object reference
  - IP address, port, interface name
  - time of creation, local object number (new for each object)
- Use the same as for local object references
- If used as addresses
  - cannot support relocation (alternative in CORBA)



Remote interfaces

- Specify externally accessed
  - variables and procedures
  - no direct references to variables (no global memory)
  - local interface separate
- Parameters
  - input, output or both,
  - instead of call by value, call by reference
- No pointers
- No constructors

Remote object and its interfaces



- CORBA: Interface Definition Language (IDL)
- Java RMI: as other interfaces, keyword *Remote*

Handling remote objects

- Exceptions
  - raised in remote invocation
  - clients need to handle exceptions
  - timeouts in case server crashed or too busy
- Garbage collection
  - distributed garbage collection may be necessary
  - combined local and distributed collector
  - cf Java reference counting

RMI issues

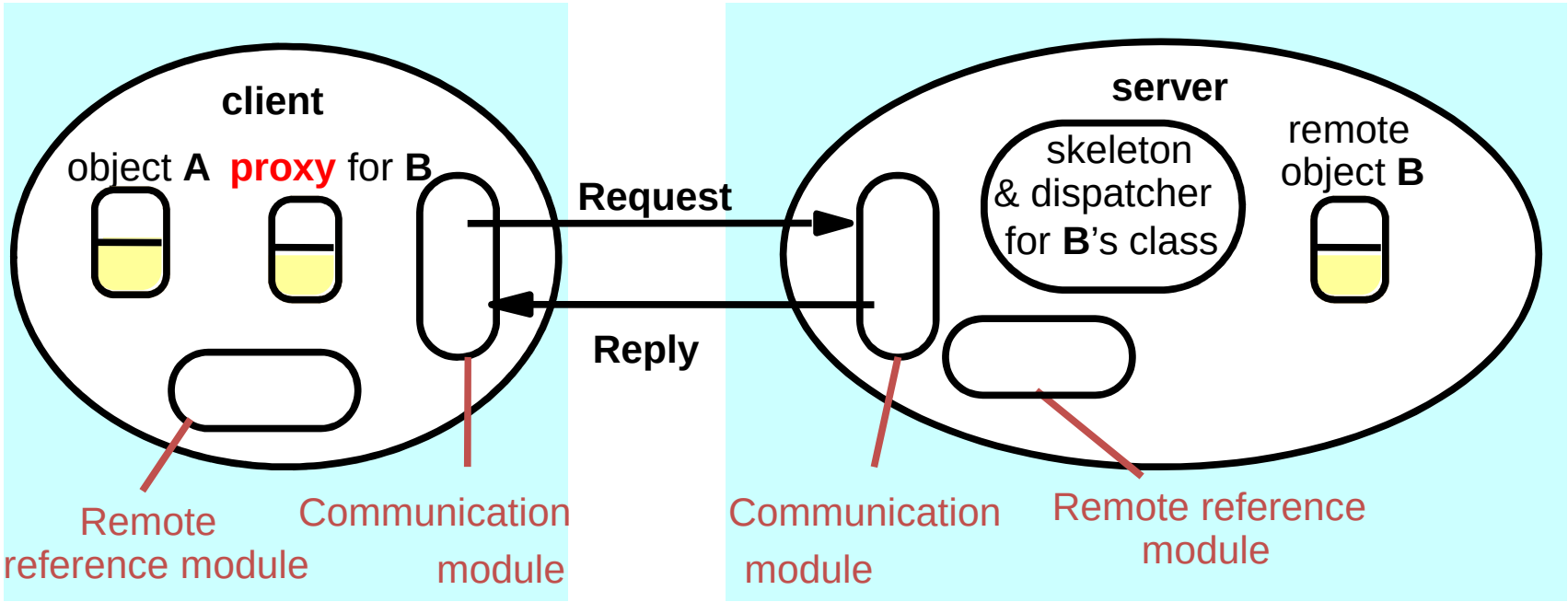
- Local invocations
  - executed exactly once
- Remote invocations
  - via Request-Reply (see *DoOperation*)
  - may suffer from communication failures!
    - ❑retransmission of request/reply
    - ❑message duplication, duplication filtering
  - no unique semantics…

Invocation semantics summary

Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	***Maybe***
Yes	No	Re-execute procedure	***At-least-once***
Yes	Yes	Retransmit reply	***At-most-once***

Re-executing a method sometimes dangerous...

Implementation of RMI



Object A invokes a method in a remote object B:
communication module, remote reference module, RMI software.

Communication modules

- Reside in client and server
- Carry out Request-Reply jointly
  - use unique message ids (new integer for each message)
  - implement given RMI semantics
- Server's communication module
  - selects dispatcher within RMI software
  - converts remote object reference to local

Remote reference module

- Creates remote object references and proxies
- Translates remote to local references (object table):
  - correspondence between remote and local object references (proxies)
- Directs requests to proxy (if exists)
- Called by RMI software
  - when marshalling/unmarshalling

RMI software architecture

- Proxy (for transparency)
  - behaves like local object to client
  - forwards requests to remote object
- Dispatcher
  - receives request
  - selects method (methodID) and passes on request to skeleton
- Skeleton
  - implements methods in remote interface
    - unmarshals data, invokes remote object
    - waits for result, marshals it and returns reply

Binding and activation

- The binder
  - mapping from textual names to remote object references
  - used by clients as a look-up service (cf Java RMIregistry)
- Activation
  - objects active (within running process) and passive (=implementation of methods + marshalled state)
  - activation = create new instance of class + initialise from stored state
- Activator
  - records location of passive and active objects
  - starts server processes and activates objects within them

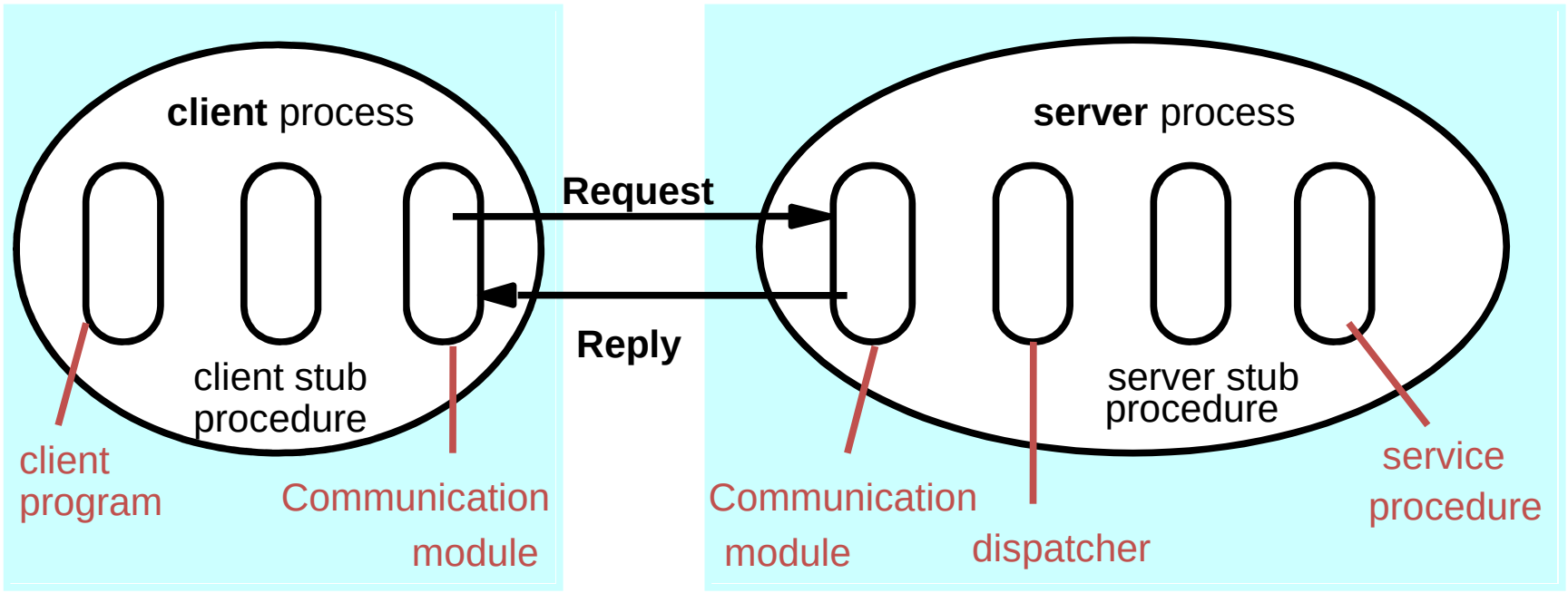
Object location issues

- Persistent object stores
  - stored on disk, state in marshalled form
  - readily available
  - cf Persistent Java
- Object migration
  - need to use remote object reference and address
- Location service
  - assists in locating objects
  - maps remote object references to probable locations

Remote Procedure Call (RPC)

- RPC
  - historically first, now little used
  - over Request-Reply protocol
  - usually at-least-once or at-most-once semantics
  - can be seen as a restricted form of RMI
  - cf Sun RPC
- RPC software architecture
  - similar to RMI (communication, dispatcher and stub in place of proxy/skeleton)

RPC client and server



Implemented over Request-Reply protocol.

Summary

- Distributed object model
  - capabilities for handling remote objects (remote references, etc)
  - RMI: maybe, at-least-once, at-most-once semantics
  - RMI implementation, software architecture
- Other distributed programming paradigms
  - RPC, restricted form of RMI, less often used

Further reading: chapter 5