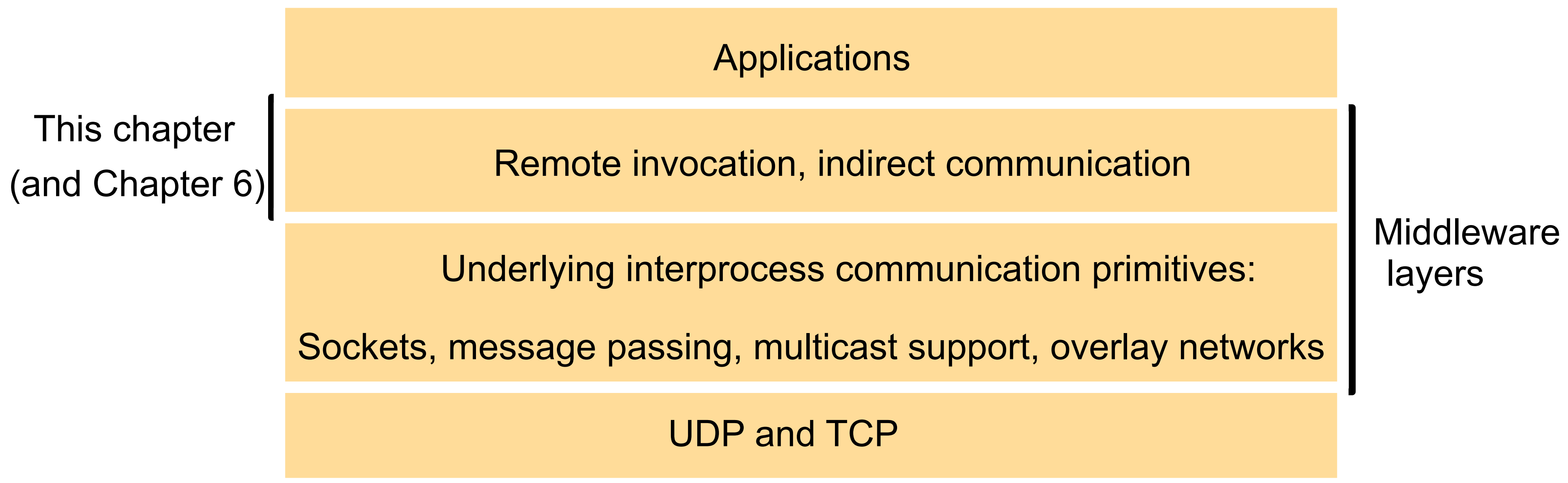
Universität Stuttgart
Master of Science

Remote Method Invocation

Marco Aiello

Figure 5.1
Middleware layers


Applications
This chapter
(and Chapter 6)
Remote invocation, indirect communication
Underlying interprocess communication primitives:
Sockets, message passing, multicast support, overlay networks
Middleware layers
UDP and TCP

Middleware programming models

- Distributed objects and remote object invocation is the model explained:
  - illustrated by Java RMI
- CORBA:
  - it provides remote object invocation between a client program written in one language and a server program written in another language
  - our book uses Java CORBA to illustrate the use of CORBA
  - another language commonly used in CORBA is C++
- Other programming models
  - remote event notification
  - remote SQL access
  - distributed transaction processing

External data representation

- It masks the differences due to different computer hardware and software.
- CORBA CDR
  - only defined in CORBA 2.0 in 1998, before that, each implementation of CORBA had an external data representation, but they could not generally work with one another. That is:
    - the heterogeneity of hardware was masked
    - but not the heterogeneity due to different programmers (until CORBA 2)
  - CORBA CDR represents simple and constructed data types (sequence, string, array, struct, enum and union)
    - note that it does not deal with objects
  - it requires an IDL specification of data to be serialised
- Java object serialisation
  - represents both objects and primitive data values
  - it uses reflection to serialise and deserialise objects– it does not need an IDL specification of the objects

CORBA IDL example

```
struct Person {
        string name;
        string place;
        long year;
} ;
interface PersonList {
        readonly attribute string listname;
        void addPerson(in Person p) ;
        void getPerson(in string name, out Person p);
        long number();
};
```

Figure 5.2

CORBA has a struct

remote interface

remote interface defines methods for RMI

parameters are *in*, *out* or *inout*

- Remote interface:
  - specifies the methods of an object available for remote invocation
  - an interface definition language (or IDL) is used to specify remote interfaces. E.g. the above in CORBA IDL.
  - Java RMI would have a class for *Person*, but CORBA has a *struct*

•

Figure 5.3
Remote and local method invocations

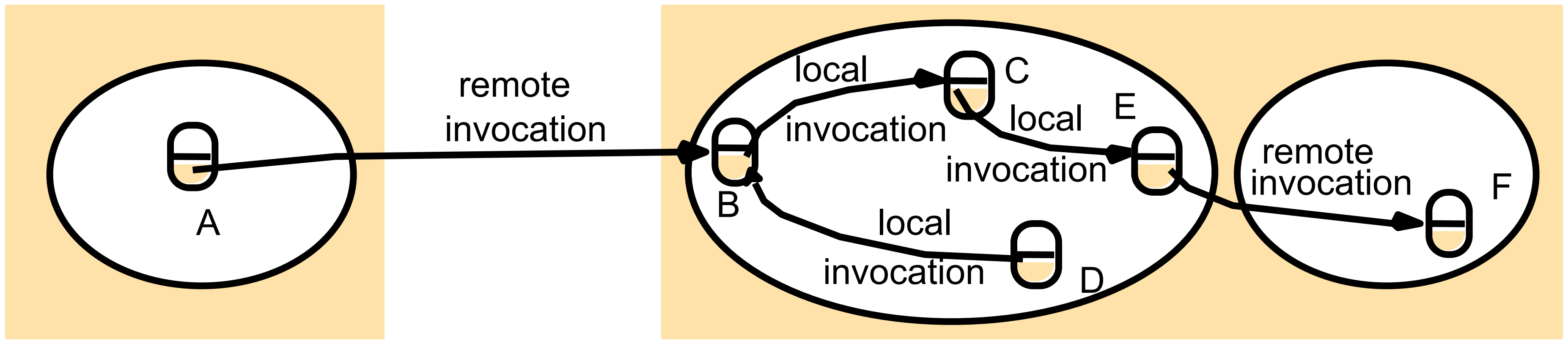


Figure 5.4
A remote object and its remote interface

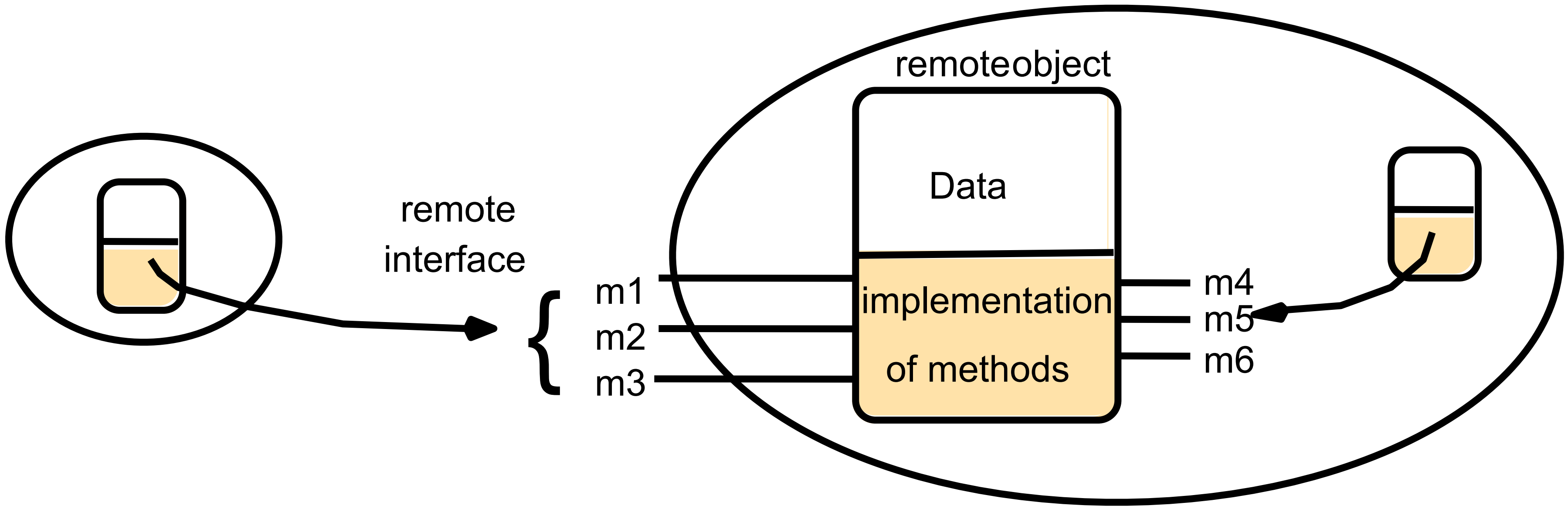


Figure 5.14
Instantiation of remote objects

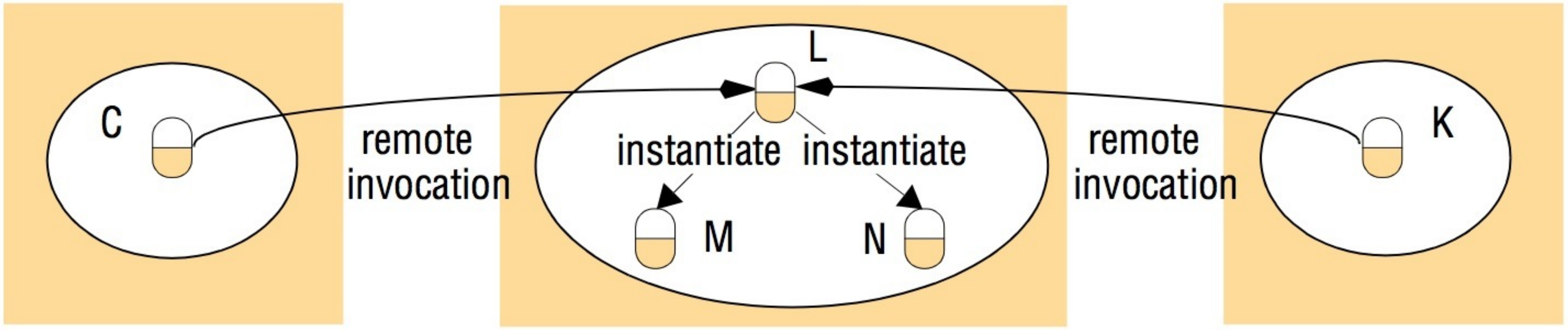


Figure 5.5
Invocation semantics

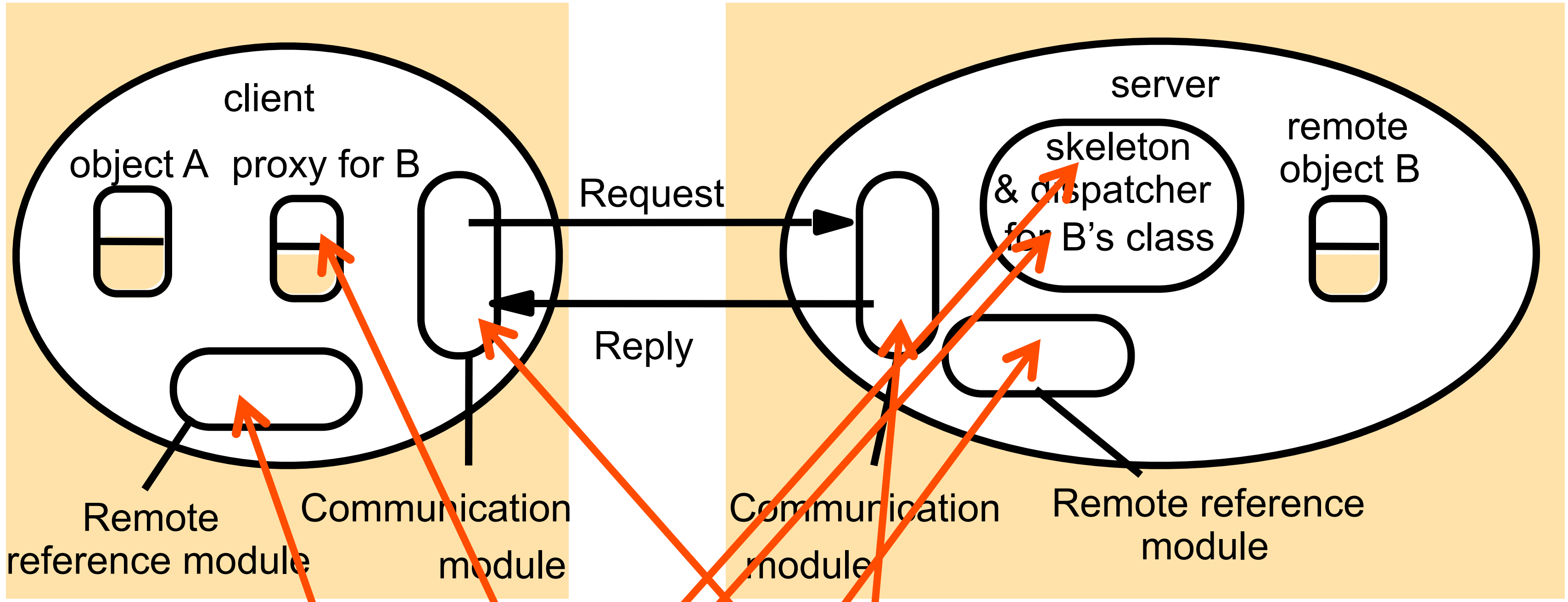
Fault tolerance measures			*Invocation semantics*
Retransmit request message	*Duplicate filtering*	*Re-execute procedure or retransmit reply*	
No	Not applicable	Not applicable	*Maybe*
Yes	No	Re-execute procedure	*At-least-once*
Yes	Yes	Retransmit reply	*At-most-once*

Invocation semantics: failure model

- *Maybe*, *At-least-once* and *At-most-once* can suffer from crash failures when the server containing the remote object fails.
- *Maybe* - if no reply, the client does not know if method was executed or not
  - omission failures if the invocation or result message is lost
- *At-least-once* - the client gets a result (and the method was executed at least once) or an exception (no result)
  - arbitrary failures. If the invocation message is retransmitted, the remote object may execute the method more than once, possibly causing wrong values to be stored or returned.
  - if *idempotent* operations are used, arbitrary failures will not occur
- *At-most-once* - the client gets a result (and the method was executed exactly once) or an exception (instead of a result, in which case, the method was executed once or not at all)

The architecture of remote method invocation

Figure 5.6



Proxy - makes RMI transparent to [illegible] s
remote interface. Marshals reques
results. Forwards request.

carries out Request-reply protocol

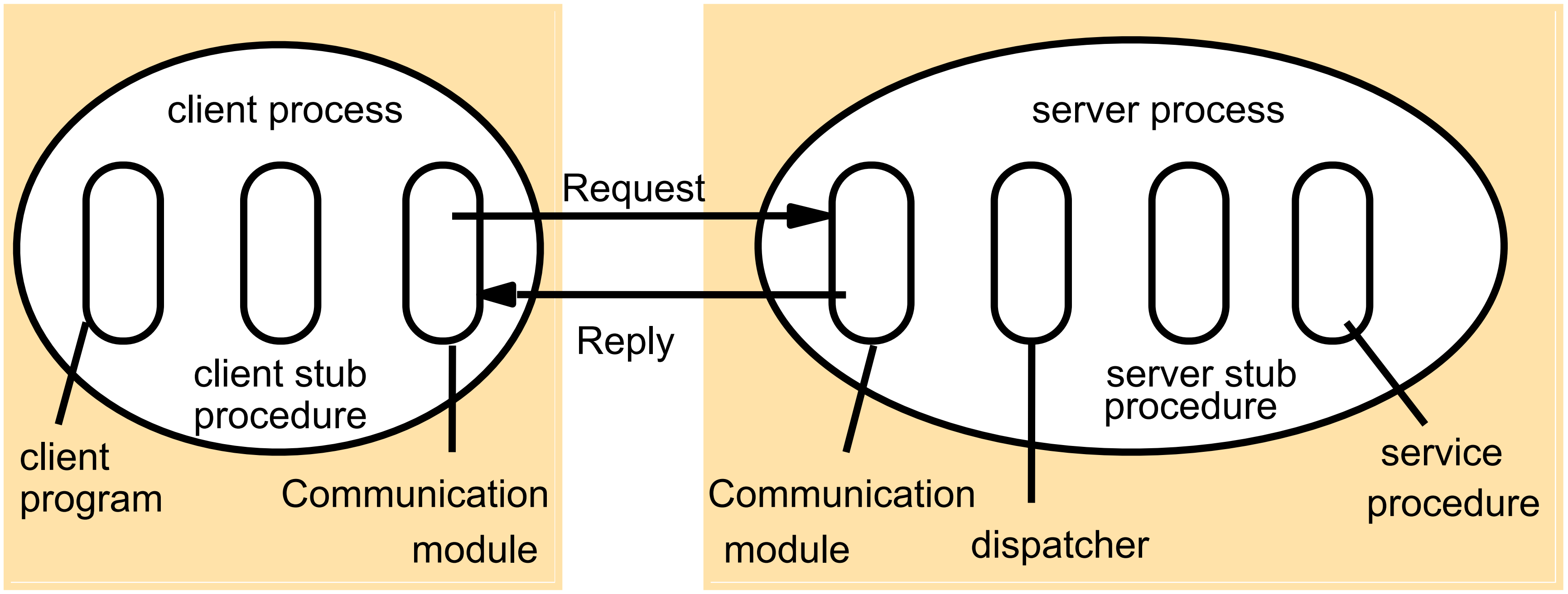
translates between local and remote object references and creates remote object references. Uses remote object table

terface.

Invokes

RMI software - between application level objects and communication and remote reference modules

Figure 5.7
Role of client and server stub procedures in RPC


client process
Request
server process
Reply
client stub procedure
server stub procedure
client program
Communication module
Communication module
dispatcher
service procedure