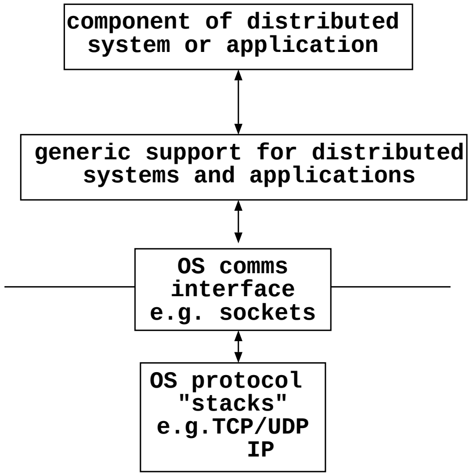
Communications support for Distributed Systems and Applications



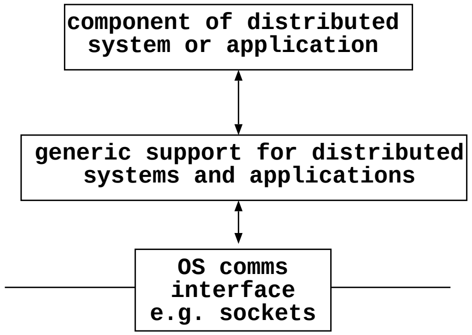
The OS interface

sockets provide a programmer’s interface to a selection of communications protocols

sockets are created and used by system calls
to send
- byte streams
- packets of unstructured bytes (datagrams)

to e.g. IP-address/port-number

ALTERNATIVELY: the OS interface may be designed to support distributed objects and the API may be defined in terms of objects’ ports with system-wide naming of port-IDs
e.g. Mach, Chorus,......



"generic support for distributed systems and applications" builds on OS-level communications services to support the development and execution of distributed software

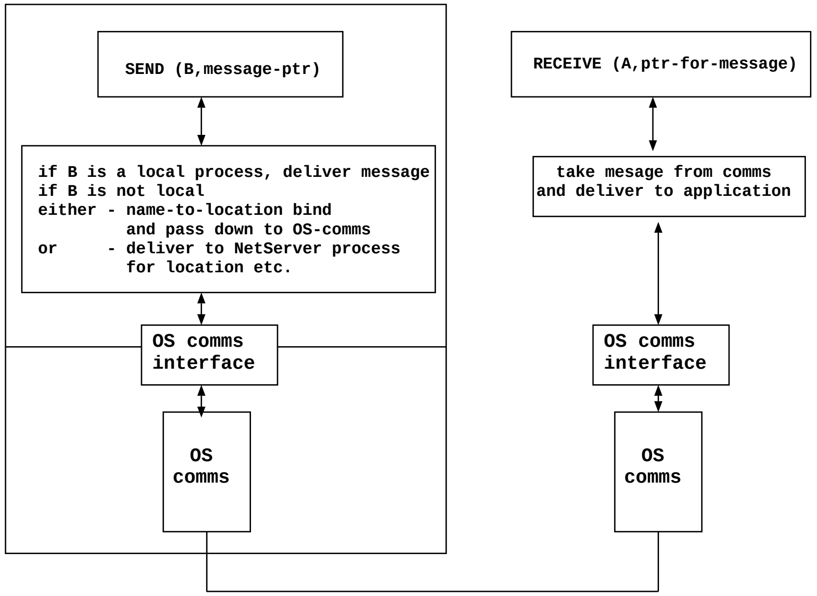
As discussed in the introduction, at this level we need support for (depending on the software model which defines the communicating entities):

- naming
- location
- name to location binding
- communication
    - message delivery?
    - call to server-interface.procedure (args)?
    - object invocation?
- authentication
- access control

Asynchronous message passing

ref 1B concurrent systems, Dip/2G OS Foundations

Message passing maps naturally onto distributed communication, provided the communicating entities are named and located system-wide



Message-passing middleware

IBM Message service: MQSeries

one-to-one message-passing

used under e.g. CICS transaction processing

http://www.software.ibm.cam/ts/mqseries

TIBCO TIB message passing

multicasts to "subjects"
need a naming scheme and yellow pages service
any process who has registered interest in a subject receives messages on it

used with e.g. Oracle/Illustra DB

The messages are structured so that language - level type systems may be built above them

Message systems have a larger proportion of the middleware market than O-O systems
e.g. IBM 24%, TIBCO 17% (1998?)

Early middleware research

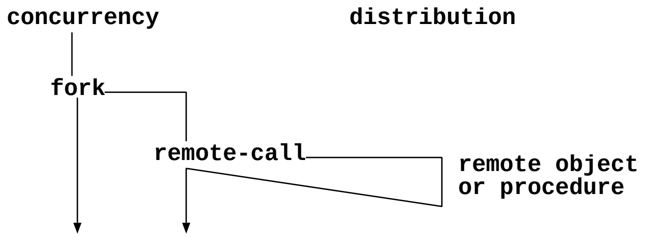
message-passing is difficult to program
- matching requests and replies

software structure and pattern of use tends
to be based on client/server or object models

language-level communication paradigm is
    procedure call
    object invocation

multi-threaded programming became commonplace
even at the application level

so use a thread to make a blocking remote service
call or remote object invocation and continue
local work in other threads



RPC systems were developed in research projects
(e.g. Mayflower and Unison RPC, Cambridge CL, mid 80's,
ANSA RPC under Alvey, then APM, now Citrix Cambridge)
then became incorporated into standards such as
ISO-ODP, OSF-DCE

RPC is built above request-response message passing
(see later) but message passing may not be visible to
and programmable at the application level

give both object invocation and message passing to the application?

with message passing only:
    doesn't extend language-level paradigm
    doesn't model service invocation well

with object invocation only:
    doesn't support large objects and streams well
    difficult to get immediate response to events

suppose an object is a source of events to which
an application should respond asap:

polling:

client polls server at some period
response is delayed by on average half that period
  either: overload comms with polling
  or: respond sluggishly

synchronous callback:

server calls interested clients on event occurrence
clients can delay server
need multi-threaded servers
complex to program for delayed threads

current O-O middleware platforms provide event services

Java RMI/Jini + events
   single language, proprietary

OMG-CORBA event notification service 1998
   multi-language, open interoperability

CEA (Cambridge event architecture)
   early 1990's research
   extend any O-O platform

we now go on to consider RPC in some detail

OMG CORBA

Further information on the object management group's
common object request broker architecture
(OMG CORBA) may be found starting from:

http://www.omg.corba/

There is too much information available!
- but you can select a suitable tutorial

Perhaps the most useful is the CORBA 2.2 1998
specification at:

http://www.omg.corba/cichpter.html

WARNING: DO NOT DOWNLOAD THE WHOLE DOCUMENT
it is two inches of double-sided A4.

there is an option to select from it by chapter.
The following chapters are of interest:

object model
overview
IDL syntax and semantics
ORB interface

Let me know if you would like any of these to be on
the course website.