LECTURE 30
COMMUNICATION IN MACH

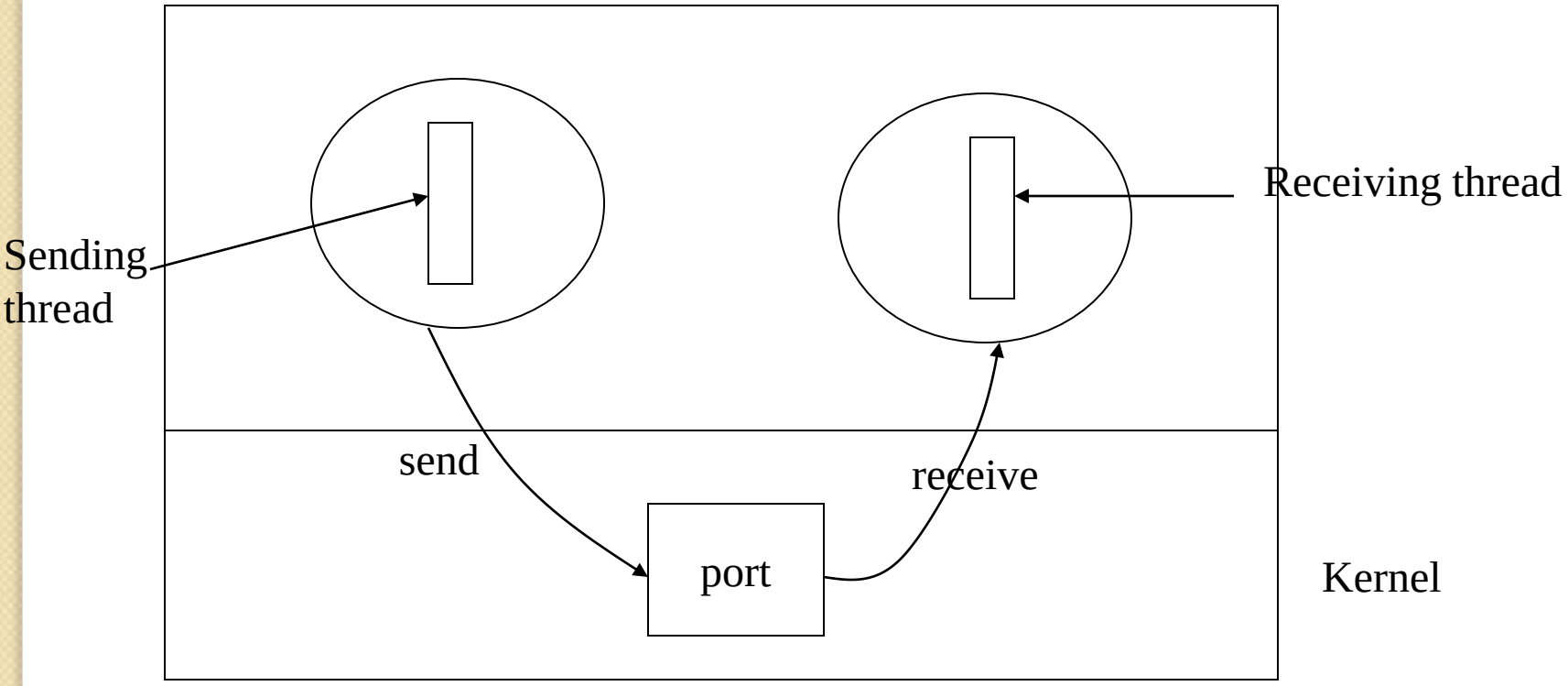
Communication in Mach

- The basis of all communication in Mach is a kernel data structure called a port.
- When a thread in one process wants to communicate with a thread in another process, the sending thread writes the message to the port and the receiving thread takes it out.
- Each port is protected to ensure that only authorized processes can send it and receive from it.
- Ports support unidirectional communication. A port that can be used to send a request from a client to a server cannot also be used to send the reply back from the server to the client. A second port is needed for the reply.

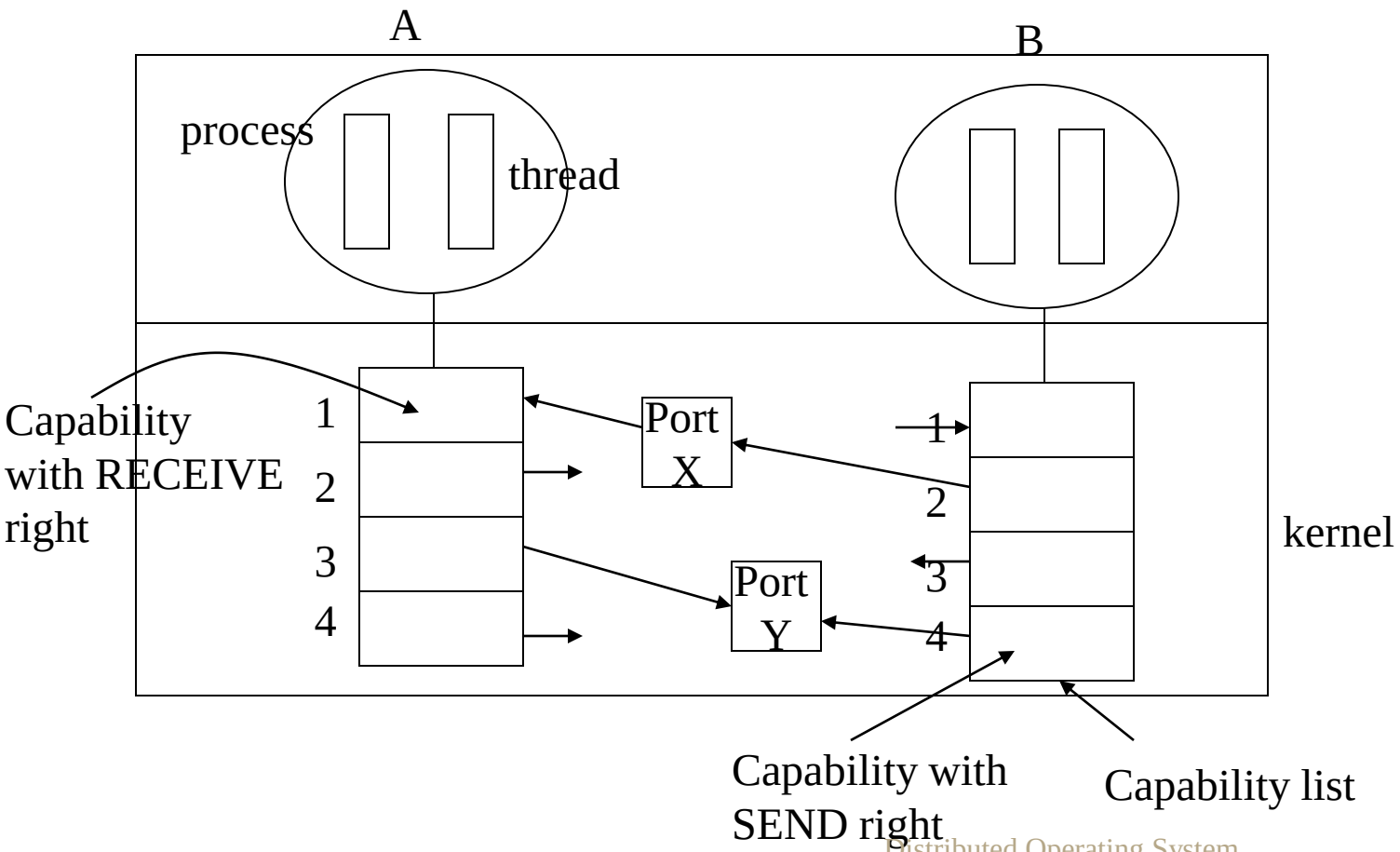
A Mach port

Message queue
Current message count
Maximum messages
Port set this port belongs to
Counts of outstanding capabilities
Capabilities to use for error reporting
Queue of threads blocked on this port
Pointer to the process holding the RECEIVE capability
Index of this port in the receiver’s capability list
Pointer to the kernel object
Miscellaneous items

Message passing via a port



Capabilities



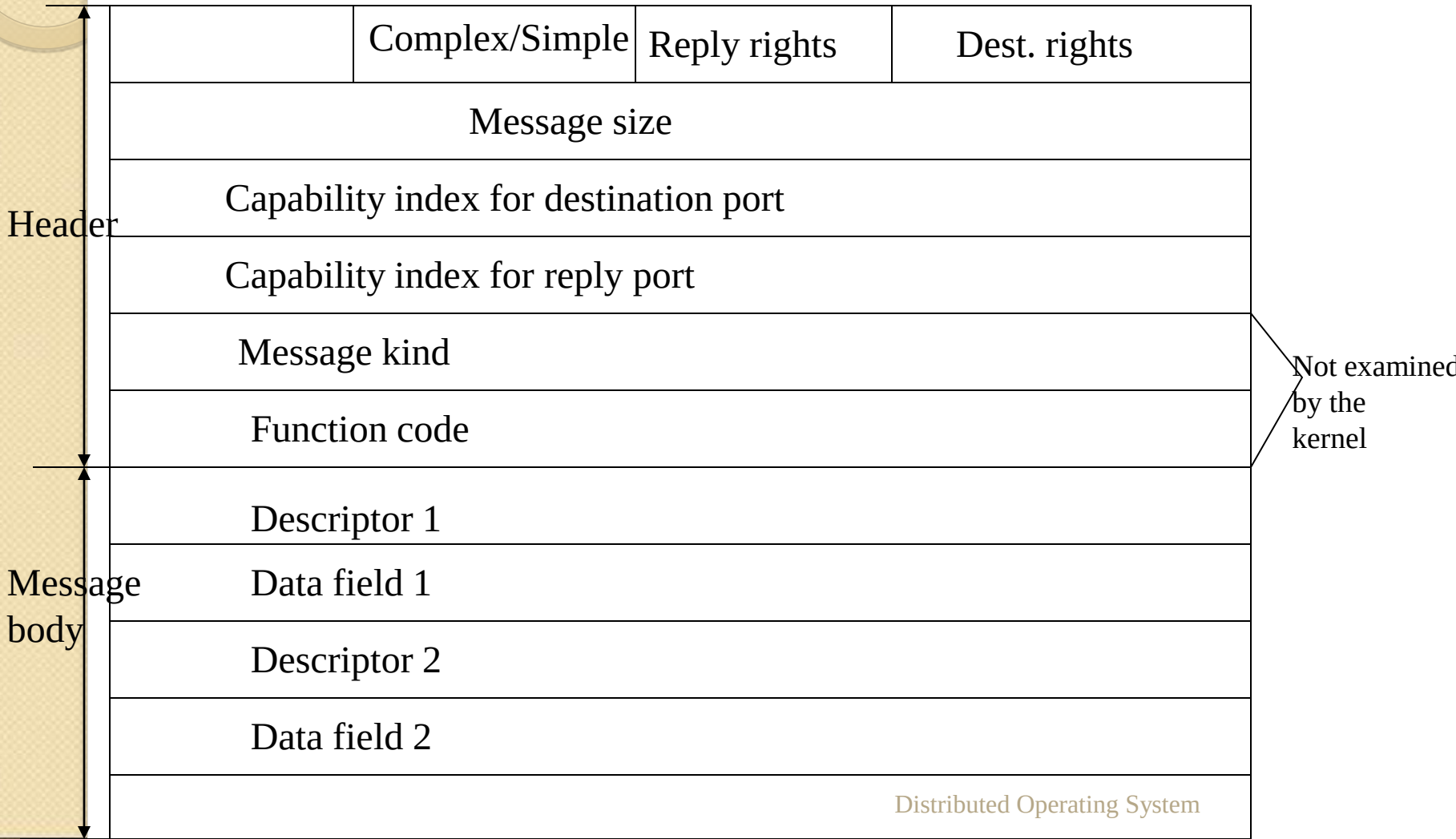
Primitives for Managing Ports

Allocate	Create a port and insert its capability in the capability list
Destroy	Destroy a port and remove its capability from the list
Deallocate	Remove a capability from the capability list
Extract_right	Extract the n-th capability from another process
Insert_right	Insert a capability in another process' capability list
Move_member	Move a capability into a capability set
Set_qlimit	Set the number of messages a port can hold

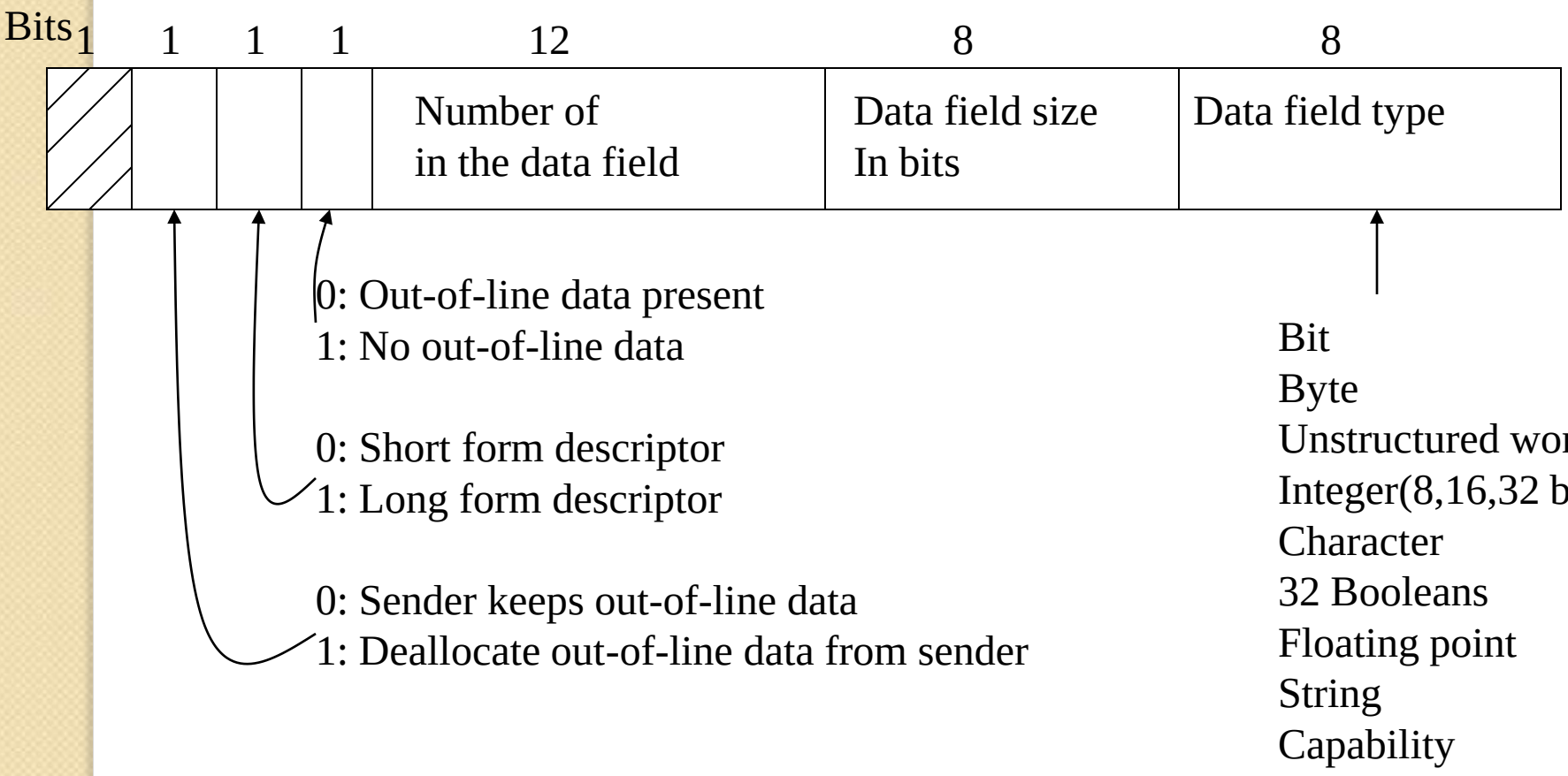
Sending and Receiving Messages

- Mach_msg(&hdr, options, send_size, rcv_size, rcv_port, timeout, notify_port);
- The first parameter, *hdr*, is a pointer to the message to be sent or to the place where the incoming message is put, or both.
- The second parameter, *options*, contains a bit specifying that a message is to be sent, and another one specifying that a message is to be received. Another bit enables a timeout, given by the *timeout* parameter. Other bits in *options* allow a SEND that cannot complete immediately to return control anyway, with a status report being sent to *notify_port* later.
- The *send_size* and *rcv_size* parameters tell how large the outgoing message is and how many bytes are available for storing the incoming message, respectively.
- *Rcv_port* is used for receiving messages. It is the capability name of the port or port set being listened to.

The Mach message format



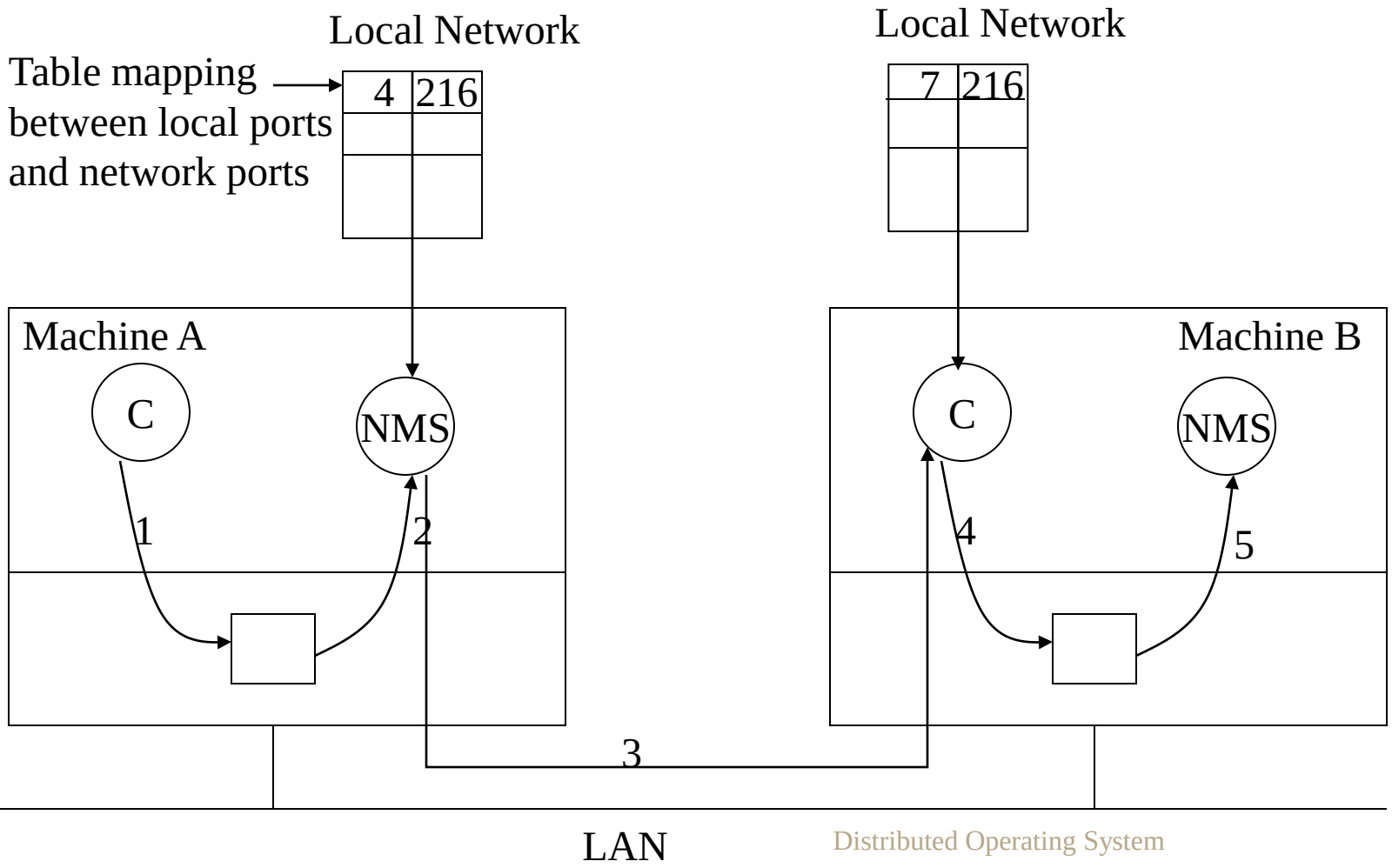
Complex message field descriptor



The Network Message Server

- Message transport from the client to the server requires five steps:
- 1. The client sends a message to the server's proxy port.
- 2. The network message server gets this message.
- 3. The network message server looks up the local port in a table that maps proxy ports onto network ports. Once the network port is known, the network message server looks up its location in other tables. It then constructs a network message containing the local message and sends it over the LAN to the network message server on the server's machine. When the remote network message server gets the message, it looks up the network port number contained in it and maps it onto a local port number.
- 4. The remote network message server writes the message to the local port just looked up.
- 5. The server reads the message from the local port and carries out the request.

NMS Cont..


Local Network
Table mapping
between local ports
and network ports
4 216
Local Network
7 216
Machine A
C
NMS
1
2
Machine B
C
NMS
4
5
3
LAN

ASSIGNMENT

- Explain the communication process in distributed operating system.