Chapter 4: Interprocess Communication

From **Coulouris, Dollimore, Kindberg and Blair**
Distributed Systems: Concepts and Design

Edition 5, © Addison-Wesley 2012

Outlines

- Introduction – Interprocess Communication
- Application Program Interface (API) for Internet Protocols
- External Data Representation and Marshalling

Introduction: Interprocess communication

- This chapter (Chapter 4) and the next chapter (Chapter 5) are concerned with the **communication aspects** of middleware.
- **Chapter 5** examines **remote invocation**.
- **Chapter 3** discussed the Internet transport-level protocols **UDP** and **TCP** without saying how middleware and application programs could use these protocols.
- The **next section** of this chapter introduces the **characteristics of interprocess communication** and then discusses User Datagram Protocol **(UDP)** and Transport Control Protocol **(TCP)** from a programmer's point of view.

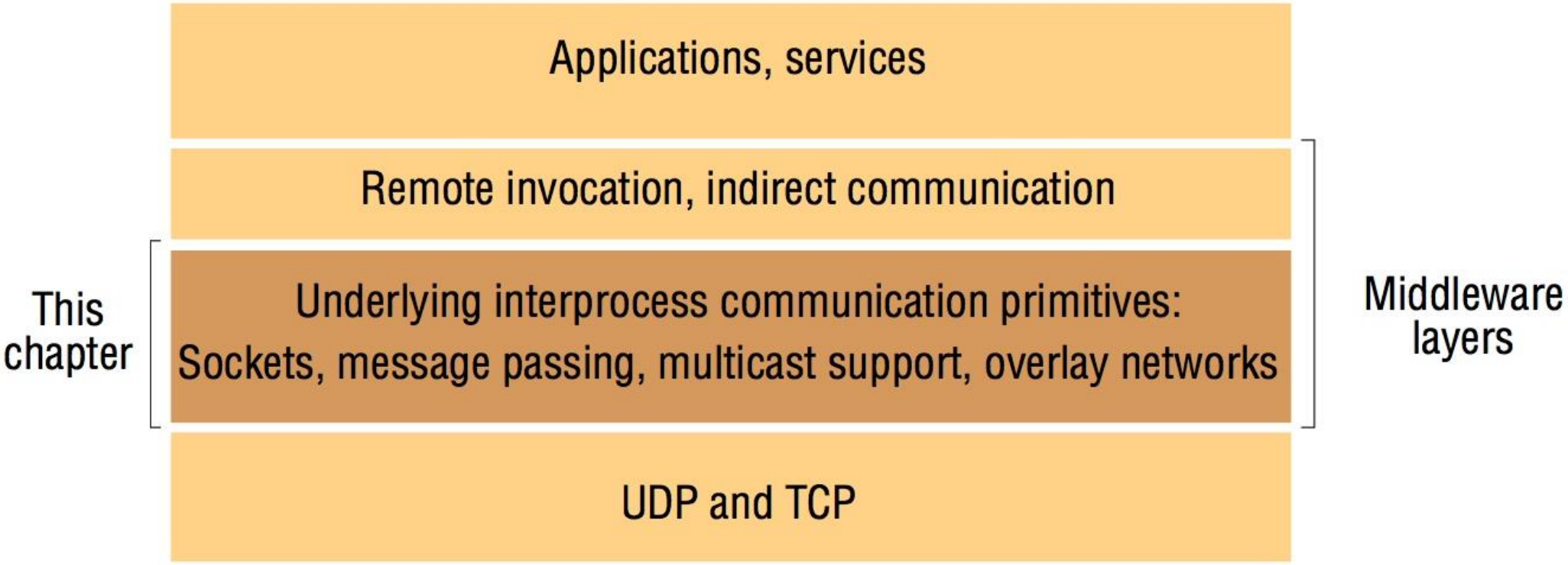
Introduction:
Interprocess communication

- **DS processes** communicate with each other to request services or to provide services.
- **Interprocess communications** are handled through massage passing over the underlying network.
- **TCP/IP** protocol is used to mask the network heterogeneity.
  - Network Heterogeneity: Different network types, different data types, …
- **Middleware** is used to mask the platforms and OS heterogeneity. It enables the processes to do interprocess communication using a known language to all different platforms and OS.

Introduction:
Interprocess communication

- **Interprocess communication** involves using two operations: send / receive.
- **Interprocess communication** uses either UDP (independent packets called datagrams) or TCP protocols (stream).
- **Marshalling** and **Unmarshalling** are used to prepare and restore the massages in a universal format.

Figure 4.1
Middleware layers



The API of internet protocols:

The characteristics of Interprocess communication

1. **Synchronous & Asynchronous Communication.**
2. **Massage Destinations.**
3. **Reliability.**
4. **Ordering.**

The API of internet protocols:
The characteristics of Interprocess communication

1. **Synchronous & Asynchronous Communication**:

- A queue is associated with each message destination.
- Sending processes cause messages to be added to remote queues and receiving processes remove messages from local queues.

The API of internet protocols:
The characteristics of Interprocess communication

- In the ***synchronous*** form of communication:
  - The sending and receiving processes synchronize at every message. In this case, both ***send*** and ***receive*** are ***blocking*** operations.
  - Whenever a ***send*** is issued the sending process (or thread) is blocked until the corresponding receive is issued.
  - Whenever a ***receive*** is issued by a process (or thread), it blocks until a message arrives.

The API of internet protocols: The characteristics of Interprocess communication

- In the ***asynchronous*** form of communication:
  - The use of the ***send*** operation is ***nonblocking*** in that the sending process is allowed to proceed as soon as the message has been copied to a local buffer, and the transmission of the message proceeds in parallel with the sending process.
  - The ***receive*** operation can be ***blocking*** and ***non-blocking***.
  - In the non-blocking, the receiving process proceeds with its program after issuing a receive operation, which provides a buffer to be filled in the background, but it must separately receive notification that its buffer has been filled, by polling or interrupt.

The API of internet protocols: The characteristics of Interprocess communication

2. **Message Destinations:** Chapter 3 explains that in the Internet protocols, messages are sent to pairs of (*Internet address*, *local port*).
3. **Reliability:** Reliable communication is defined in terms of validity and integrity.

- As far as the **validity** property is concerned, a point-to-point message service can be described as **reliable** if messages are guaranteed to be delivered despite a 'reasonable' number of packets being dropped or lost.
- In contrast, a point-to-point message service can be described as **unreliable** if messages are not guaranteed to be delivered in the face of even a single packet dropped or lost.
- For **integrity,** messages must arrive uncorrupted and without duplication.

The API of internet protocols: The characteristics of Interprocess communication

4. **Ordering:** Some applications require that messages be delivered in *sender order* – that is, the order in which they were transmitted by the sender. The delivery of messages out of sender order is regarded as a failure by such applications.

The API of internet protocols: The characteristics of Interprocess communication

Sockets:

- Both forms of communication (**UDP** and **TCP**) use the **socket** abstraction, which provides an endpoint for communication between processes.
- **Interprocess communication** consists of transmitting a message between a socket in one process and a socket in another process, as illustrated in **Figure 4.2**.
- For a process to receive messages, its **socket** must be bound to a local port and one of the internet addresses of the computer on which it runs.
- **Messages** sent to a particular internet address and port number can be received only by a process whose socket is associated with that internet address and port number.
- **Processes** may use the same socket for sending and receiving messages.

Here, the general concept of a socket is used as a point of communication.

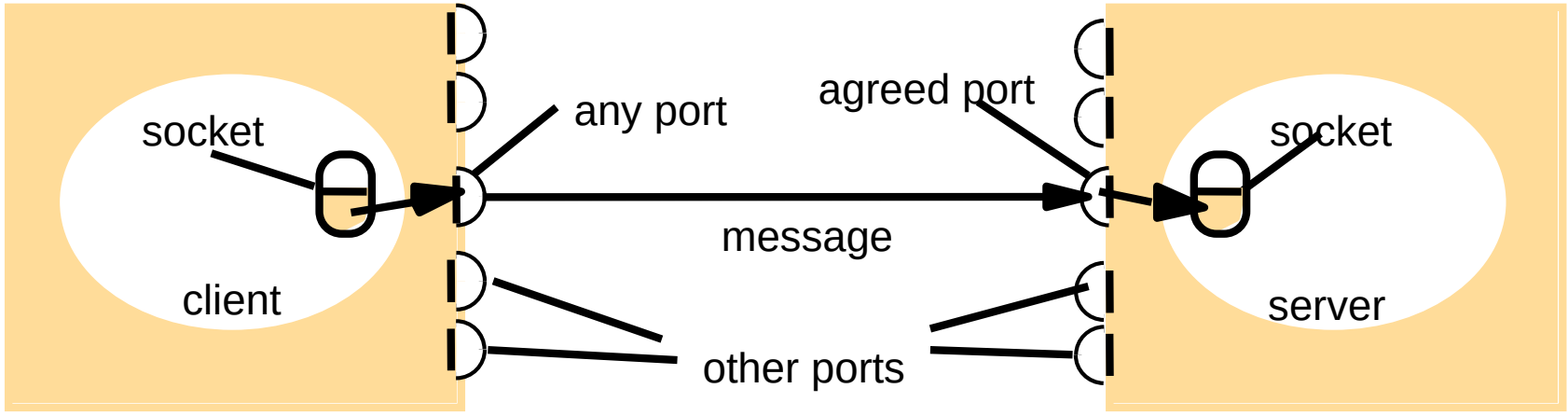
However, programming wise, sockets are used in TCP and datagrams are used in UDP.

The API of internet protocols: The characteristics of Interprocess communication

Sockets:

- Each **computer** has a large number (2^{16}) of possible port numbers for use by local processes for receiving messages.
- Any **process** may make use of multiple ports to receive messages, but a process cannot share ports with other processes on the same computer.
  - In other words, if a port is occupied by one process, other processes cannot use it.
- **Processes** using IP multicast are an exception in that they do share ports.
- Any number of **processes** may send messages to the same port.
- Each **socket** is associated with a particular protocol – either UDP or TCP.

Figure 4.2
Sockets and ports



The API of internet protocols: The characteristics of Interprocess communication

UDP datagram communication:

- Sending a massage without acknowledgment.
- UDP is non-blocking send and blocking receive for datagram.
- Timeout is used to make the process not to block forever.
- UDP massage may expose failures 1) omission failure: where some datagrams are dropped 2) ordering: out of sender order delivery.
- UDP is efficient (no overheads) but unreliable. It is used for applications like IP telephony (VoIP).

Datagram packet format:

array of bytes containing massage | length of massage | internet address | port number

The API of internet protocols: The characteristics of Interprocess communication

TCP stream communication:

- The API to TCP protocol provides the abstraction of a stream of bytes to which data may be written and from which data may be read.
- The following **characteristics** of the network are hidden by the stream abstraction:
    1. Massage sizes.
    2. Lost massages (acknowledgment).
    3. Flow control (speed of transmission).
    4. Massage duplication and ordering.
    5. Massage destination (communication establishment).

The API of internet protocols:
The characteristics of Interprocess communication

1. Massage sizes:

- The application can choose how much data it writes to a stream or reads from it.
- It may deal in very small or very large sets of data.
- The underlying implementation of a TCP stream decides how much data to collect before transmitting it as one or more IP packets.
- On arrival, the data is handed to the application as requested.
- Applications can, if necessary, force data to be sent immediately.

The API of internet protocols:
The characteristics of Interprocess communication

2. Lost massages (acknowledgment):

- The TCP protocol uses an acknowledgement scheme.
- As an example of a simple scheme, the sending end keeps a record of each IP packet sent and the receiving end acknowledges all the arrivals.
- If the sender does not receive an acknowledgement within a timeout, it retransmits the message.

3. Flow control (speed of transmission):

- The TCP protocol attempts to match the speeds of the processes that read from and write to a stream.
- If the writer is too fast for the reader, then it is blocked until the reader has consumed sufficient data.

The API of internet protocols:
The characteristics of Interprocess communication

4. Massage duplication and ordering:

- Message identifiers are associated with each IP packet, which enables the recipient to detect and reject duplicates, or to reorder messages that do not arrive in sender order.

5. Massage destination (communication establishment):

- A pair of communicating processes establish a connection before they can communicate over a stream.
- Once a connection is established, the processes simply read from and write to the stream without needing to use internet addresses and ports.
- Establishing a connection involves a connect request from client to server followed by an accept request from server to client before any communication can take place.
- This could be a considerable overhead for a single client-server request and reply.

The API of internet protocols: The characteristics of Interprocess communication

The following are issues related to stream communication:

- Matching of data items.
- Blocking.
- Threads.

The API of internet protocols: The characteristics of Interprocess communication

Matching of data items:

- Two communicating processes need to agree as to the contents of the data transmitted over a stream.
- For example, if one process writes an *int* followed by a *double* to a stream, then the reader at the other end must read an *int* followed by a *double*.
- When a pair of processes do not cooperate correctly in their use of a stream, the reading process may experience errors when interpreting the data or may block due to insufficient data in the stream.

The API of internet protocols: The characteristics of Interprocess communication

Blocking:

- The data written to a stream is kept in a queue at the destination socket.
- When a process attempts to read data from an input channel, it will get data from the queue, or it will block until data becomes available.
- The process that writes data to a stream may be blocked by the TCP flow-control mechanism if the socket at the other end is queuing as much data as the protocol allows.

Threads:

- When a server accepts a connection, it generally creates a new thread in which to communicate with the new client.
- The advantage of using a separate thread for each client is that the server can block when waiting for input without delaying other clients.

The API of internet protocols: The characteristics of Interprocess communication

Failure Model:

- Lost packets are re-transmitted when a timeout takes place.
- Corrupted packets are re-transmitted (checksum is used).
- Duplicated packets are dropped (sequence number is used).

Using Checksum in Error Detection

The basic principle of **checksum** is:

- Sender adds up the data that will be sent.
- Sender appends the negative value of the sum at the end of the data and sends it to the receiver.
- Receiver adds up the received data along with checksum value.
  - If the result is zero: No Error
  - If the result is non-zero: Error

Example: Suppose sender wants to send: 11 9 6 10

11 + 9 + 6 + 10 = 36

Sender sends: 11 9 6 10 -36

Suppose receiver receives: 11 9 6 10 -36

11 + 9 + 6 + 10 + -36 = zero (No Error)

The API of internet protocols: The characteristics of Interprocess communication

Use of TCP:

Many frequently used services run over TCP connections, with reserved port numbers. These include the following:

- **HTTP:** The Hypertext Transfer Protocol is used for communication between web browsers and web servers.
- **FTP:** The File Transfer Protocol allows directories on a remote computer to be browsed and files to be transferred from one computer to another over a connection.
- **Telnet:** Telnet provides access by means of a terminal session to a remote computer.
- **SMTP:** The Simple Mail Transfer Protocol is used to send mail between computers.

External data representation and data marshalling

- **Information** in running programs (processes) is represented as a data structure or primitive data types.
- **Information** is converted to a sequence of bytes before transmission.
- **Data structures** and types vary among different architectures.
- **Sender** should convert the data to an agreed on external format (e.g., ASCII Unicode, or later formats used for marshalling / unmarshalling).

External data representation and data marshalling

- An agreed standard for the representation of data structures and primitive data types is called **external data representation**.
- **Marshalling:** is the process of taking a collection of data items and assembling them into a form suitable for transmission in a massage.
- **Unmarshalling:** is the process of disassembling them on arrival to produce an equivalent collection of data items at the destination.
- **Marshalling:** External data representation.
- **Unmarshalling:** Rebuilt data structures and primitive data.

External data representation and data marshalling

Three approaches to external data representation and marshalling:

1) **CORBA's (Common Object Request Broker Architecture)** common data representation, which is concerned with an external representation for the structured and primitive types that can be passed as the arguments and results of remote method invocations in CORBA.
2) **JAVA object serialization** which is concerned with the flattening and external data representation of any single object or tree of objects that may need to be transmitted in a message or stored on a disk. It is for use only by Java.
3) **XML (Extensible Markup Language)** which defines a textual format for representing structured data. it is used to represent the data sent in messages exchanged by clients and servers in web services.