Innopolis University

SYSTEM AND NETWORKING ENGINEERING

Distributed Systems

READING QUESTIONS 3

Communication

Student	**ID**
Grebennikov Sergey	47611

Lecturer
Konstantin Urysov

November 9, 2017

Readings:

- **[DS]** Chapter 4. Distributed systems: principles and paradigms, Andrew S. Tanenbaum, Maarten Van Steen

Questions:

1. How are the OSI model layers mapped to the adapted (middleware-centric) reference model? Discuss in terms of functionality.
   Answer:
   Compared to the OSI model, the session and presentation layer have been replaced by a single middleware layer that contains application-independent protocols. These protocols do not belong in the lower layers. Network and transport services have been grouped into communication services as normally offered by an operating system, which, in turn, manages the specific lowest-level hardware used to establish communication. (Figure 1)

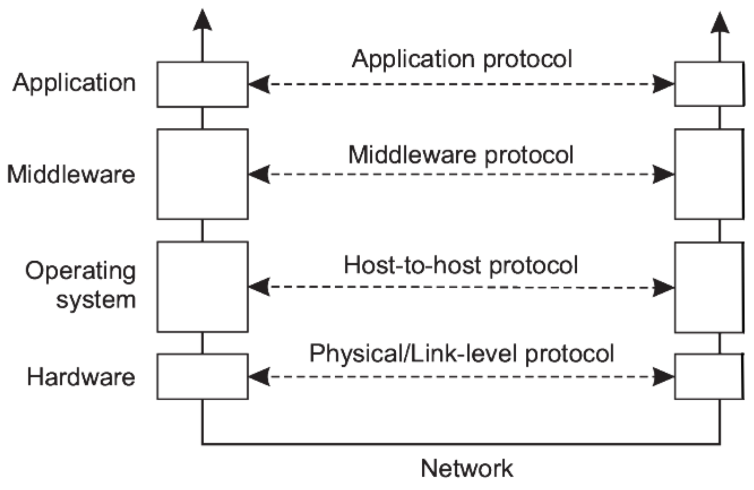

Figure 1: An adapted reference model for networked communication.

2. Give some examples of general-purpose protocols, which belong to the middleware layer. What types of distribution transparency do they provide?
   Answer:
   Domain Name System, authentication protocols, authorization protocols, commits protocols, and locking protocols belong to the middleware layer and offer a general-purpose, application-independent service. Domain Name System provides location transparency; authentication, authorization and commits protocols provide access transparency; locking protocols provide concurrency transparency.

3. Give some examples of systems evolving from combining persistent/transient, synchronous/asynchronous and discrete/streaming communication. Which combination corresponds to RPC?
   Answer:
   Message-queuing systems are combination of persistence with synchronization at request submission. Audio and video streams are combination of streaming and synchronous communication. Transient communication with synchronization after the request has been fully processed corresponds with RPC (Remote Procedure Calls).

4. Can we simply use pointers as parameters in RPC calls without extra care? Why?
   Answer:
   No, it is a big problem to pass the pointer as parameters in RPC calls. Since the pointer will refer to the local data structure somewhere in the caller's main memory and as a result, the value of the passed pointer will refer to a completely different place.

5. In multicast RPC, should the client wait for all responses? Why?
   Answer:
   It all depends. When the server replicates for fault tolerance, we can decide to wait until the first response or until most servers return the same result. On the other hand, if the servers were replicated to perform the same job, but in different parts of the input, their results might have to be merged before the client can continue.

6. Consider implementing an application using RPC on top of streams (TCP) and datagrams (UDP) sockets, respectively. What would be your main challenge?
   Answer:
   TCP and UDP are visible to applications through sockets. The purpose of the socket interface was to provide a file abstraction. Yet sockets are quite low level for many applications, thus, RPC (Remote Procedure Call) appeared as a way to hide communication details behind a procedural call bridge heterogeneous environments (Figure 2).

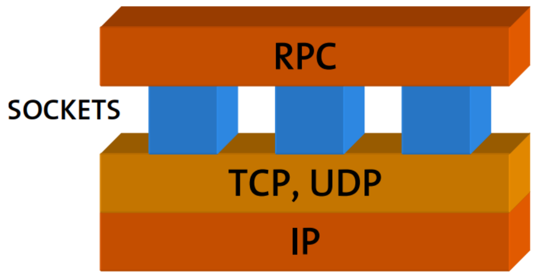

Figure 2: Remote Procedure Call

There are two alternatives for the implementing of a distributed application:

- Bottom-Up:
  - First design network protocol
  - Build program that follows the protocol using sockets
- Top-Down:
  - Design program first
  - Partition the program in different modules
  - Describe the set of procedures that make up the module interface
  - Place modules on different network hosts
  - Add the network protocol to make the procedures communicate

There are several chalanges that would be:

- client and server can reside on different computers and run on different operating systems
- the only form of communication is by sending messages (no shared memory, no shared disks, etc.)
- some minimal guarantees are to be provided (handling of failures, call semantics, etc.)
- generic solution is more preferable than a one time hack

But the main chalange is calls may fail due to network problems. Therefore we have the two transport layer protocol (TCP and UDP). TCP is a connection-oriented protocol with the ability to acknowledge receipt of packets at both ends. Acknowledgement ensures that the lost/corrupt packets can be retransmitted upon request. While UDP is an unreliable connectionless protocol that neither guarantees that the packets will ever reach the destination nor that they will arrive in the same order they were sent. Therefore, it is preferable to use a TCP connection for remote procedure calls than UDP.

7. What are the differences between ZeroMQ and Berkeley (traditional) sockets? What is the side effect of asynchronous connection-oriented communication? What are the main communication patterns supported by ZeroMQ?
   Answer:
   In ZeroMQ model an actual message transmission generally takes place over TCP connections, a socket

may be bound to multiple addresses, and ZeroMQ sockets also support multicasting (one-to-many communication). A side effect of asynchronous connection-oriented communication is that a process can request a connection setup, and subsequently send a message even if the recipient is not yet up-and-running and ready to accept incoming connection requests, let alone incoming messages. The three most important communication patterns supported by ZeroMQ are request-reply, publish-subscribe, and pipeline.

8. What is the role of the message brokers? Why/when are they needed?
   Answer:
   A message broker acts as an application-level gateway in a message-queuing system. Its main purpose is to convert incoming messages so that they can be understood by the destination application.

9. Gossiping protocols can be used for data aggregation. Each node P_i maintains a variable v_i. When two nodes i and j gossip, they set their variables v_i and v_j to $(v_i + v_j)/2$. Show that, eventually, all nodes will have computed the average of the initial values. What happens in the case when $v_1 = 1$, and $v_i = 0$ (for $i \neq 1$)?
   Answer:
   When node P_i contacts node P_j , they each update their value as:

$$v_i, v_j \leftarrow \frac{(v_i + v_j)}{2}$$

   Obviously, after this exchange, both P_i and P_j will have the same value. In fact, it is not difficult to see that eventually all nodes will have the same value, namely the average of all initial values.

   If all nodes P_i have set v_i to zero, except for P_1 who has set v_1 to 1:

$$v_i \leftarrow \begin{cases} 1 & \text{if } i = 1 \\ 0 & \textit{otherwise} \end{cases}$$

   If there are N nodes, then eventually each node will compute the average, which is $1/N$. As a consequence, every node P_i can estimate the size of the system as being $1/v_i$.

10. Where do we see gossip protocols at work in real-life computing? Think of an example.
    Answer:
    A gossip algorithm implements detection of nodes in a peer-to-peer network.