DISTRIBUTED SYSTEMS: READING QUESTIONS

3

27 OCT. 2016

ALI ABDULMADZHIDOV

1. How are the OSI model layers mapped to the adapted (middleware-centric) reference model? Discuss in terms of functionality.

Let's go from bottom.

OSI	Middleware-centric
Physical	Hardware
Data link	OS
Network	OS
Transport	OS
Session	Middleware
Presentation	Middleware
Application	Application

2. Give some examples of general-purpose protocols, which belong to the middleware layer. What types of distribution transparency do they provide?

RPC - access,

L2TP - access, location, migration

3. Give some examples of systems evolving from combining persistent/transient, synchronous/asynchronous and discrete/streaming communication. Which combination corresponds to RPC?

RPC can combine synchronous and asynchronous communication looking for task that is needed to execute on servers side. Combination of that can make program behavior easier to understand and give asynchrony in places where fully synchronous code can make whole system slow.

For example some messengers use discrete and streaming communication. Streaming for video and audio calls and discrete for messaging.

Also some "secure" messengers, Telegram for example, uses persistent communication for usual chats and transient for secure/secret chats with end-to-end encryption.

4. Can we simply use pointers as parameters in RPC calls without extra care? Why?

Reference or pointer to local data on caller's machine's memory can most likely be referring to something completely different on server instead of what we were trying to recipient.

5. In multicast RPC, should the client wait for all responses? Why?

It depends of reasons for multicast RPC. For example, if we send job also to second backup server to gain fail transparency, client'll wait only for first response. But if we have several servers that make their part of job, and client needs whole result it have to wait answer from all servers.

6. Consider implementing an application using RPC on top of streams (TCP) and datagrams (UDP) sockets, respectively. What would be your main challenge?

RPC can operate on TCP or UDP. RCP + UDP become stateless protocol and RPC + TCP grants slower but reliable connection.

In TCP/IP networks, the authors of RPC were faced with the problem of mapping program numbers to generic network services. They chose to have each server provide both a TCP and a UDP port for each program and each version. Generally, RPC applications will use UDP when sending data, and only fall back to TCP when the data to be transferred doesn't fit into a single UDP datagram. (http://www.tldp.org/LDP/nag/node128.html)

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?

ZeroMQ sockets can support many-to-one communication, but standard Berkeley sockets uses just one-to-one way. Also, ZeroMQ sockets support multicasting (one to many).

An interesting side effect of combining asynchronous with connection-oriented
communication, is that a process can send message while system is setting up connection and is'nt ready to process request. After thatconnection request and subsequent messages goes queued at the sender's side and while a separate thread that is part of ZeroMQ's library will make sure that eventually the connection is set up and all queued messages are sended to the recipient.

ZeroMQ supported 3 communication patterns: request-reply, publish-subscribe and pipeline.

Request-reply - used in client-server communication. Client uses REQ socket and exptect respond. Server uses REP socket.

Publish-subscribe client subscribes to specific messages and only this messages will be transfered to him. Sever uses PUB socket and client SUB socket.

Pipeline pattern - when one process pushes results and other pulls it in like in pipe.

8. What is the role of the message brokers? Why/when are they needed?

Message broker is a part of distributed system that takes to itself all problems with message converting for different systems. Simple example, if we have ASCII-text message formated by newline symbol that is sending from source to destination and destination machine understands messages formatted only by spaces, message broker will convert it from one format to another.