Questions and answers on distributed systems

Extracted from the distributed systems lecture

Questions and answers on distributed systems: Extracted from the distributed systems lecture

Table of Contents

Purpose of this Q and A type document

Abstract

The purpose of this list is to check and enhance the *understanding* of Distributed Systems.

Many questions go like "list some features of ....". I do not expect you to know each and every feature. It is good if you know some or the most important ones and I will not count beans here.

Chapter 1. Distributed Systems

1. Why would you design a system as a distributed system? List some advantages of distributed systems.

2. List some disadvantages or problems of distributed systems that local only systems do not show (or at least not so strong)

3. List three properties of distributed systems

4. Give a definition of middleware and show in a small diagram where it is positioned.

   ## Tip

   As the name "middleware" implies this software must be somewhere between other software layers, probably mediating between those layers. Almost everything we have touched in this lession is a form of middleware. Does middleware reduce work for its users?

5. What is the transparency dogma in distributed systems middleware and what is wrong with it?

   ## Tip

   Remember Jim Waldo's note on distributed computing? Middleware tries to hide the fact of distribution. Does this work *completely* ? What happens in case of errors?

6. What is a Single-point-of-failure and how can distribution help here?

   ## Tip

   Think about an important web service which needs to be available all the time

Chapter 2. Socket Based Client/Server Systems

1. What kind of reliable connection is provided by a tcp/ip based socket? Is this reliability enough for distributed calls or when does it break down?

   ## Tip

   Think about failure behavior, e.g. you are trying to order something on the web and suddenly the communication breaks down.

2. What are the delivery guarantees with: best effort, at least once, at most once. Which one is mostly used in distributed systems?

   ## Tip

   The best way to tackle this question is to draw a small diagram with a requester and server or client and server and draw both the request and the response into it. Now think about the problems caused when a) the request does not get through, b) the server crashes during request processing and c) the response is lost on its way back.

3. What is the advantage if your server side processing uses threads instead of a single process?

   ## Tip

   Think about what happens if your server receives multiple requests at the same time

4. What is the problem behind keeping state for a client on a server?

   ## Tip

   Think about different client scenarios where clients disappear or a rush of clients happens.

5. What is a proxy? Give an example of where a proxy can be used.

   ## Tip

   Do you always know that there is a proxy between a client and a certain service?

Chapter 3. Remote Calls

1. What are the differences between a local call and a remote call?

Tip

Think about efficiency, reference semantics, possible errors, different hardware etc.

2. What are stub and skeleton and why are they needed in remote prodecure calls?

Tip

Again the easiest answer comes with the help of a little picture that shows a local method call from one object (A) to another one (B). Now imagine that object B is remote on another machine but object A should NOT notice this. This turns the original object B into something different which can forward a call to the real object B now living on a different system.

3. Explain "call by reference" vs. "call by value".

Tip

Can remote calls ever be "by reference"? Or is it only "reference semantics" that is achieved by shipping things back and forth and make it look like a local call did directly manipulate a remote object?

4. How does so called marshalling solve the problem of different byte ordering of sender and receiver?

Tip

Some hardware has different opinions about how an integer e.g. should look. There a different ways to solve this communication problem but they all involve some form of agreement on how things should be assembled and disassembled.

5. Is appending characters to a file an idempotent operation?

Tip

Will it make a difference if one response is lost?

6. What is the purpose of a portmapper?

Tip

What would a client need to know if there was no portmapper? How would this affect your ability to change services on the server side?

Chapter 4. Distributed Objects

1. Explain the concept of an object reference in a distributed system. Why do remote objects ned one and who creates it?

   ## Tip

   The identity of a local object is its memory address in most cases. Would this be a usable concept of identity for distributed systems? Can you directly create a remote object on a remote system or do you need the help of the remote system to achieve this?

2. What is the purpose of an Interface Definition Language? Why does CORBA not just use the Java interface construct?

   ## Tip

   Remember that most distributed (calls, objects, components, services etc.) all know the distinction of interface and implementation. The implementation depends on your programming language, platform etc. The Interface is much more like a general rule.

3. Remote objects built with Java RMI are usually registered in a so called "registry". Why?

4. List several ways to invoke a method on a remote object

   ## Tip

   How could you invoke a message on an object that you have'nt seen at compile time of your program? Do you always want to wait for a response directly?

5. How does a server know that one of his remote objects provided by him is no longer used by clients and can be collected? How does Java RMI handle this problem and what alternatives are there?

   ## Tip

   You can put responsibilities for cleaning up objects either on the client or on the server with or without protocol support. Leaving it to the client can be risky, just as leaving it to the server programmer. How about hiding it in the protocol?

6. Activation is a technology used to load remote objects into a server on demand (i.e. when a client invokes a method on this object. Explain why this is much better than pre-loading remote objects?

   ## Tip

Think about large numbers of remote objects (e.g. when a database is represented as remote objects)

Chapter 5. Distributed Systems Services

1. What services are frequently provided by distributed systems middleware and what is their function?

2. What is the difference between functional and non-functional requirements? Which one is usually overlooked and causes the biggest problems later?

   ## Tip

   Non-functional requirements are often identical to the so called "ilities".

3. What kind of naming services do you know? Why do distributed systems need this function?

   ## Tip

   Does the telephone system know naming services?

4. Why do you need a locking service in distributed systems? What kind of problems does a locking service prevent?

5. What is a distributed deadlock and why are they hard to detect?

   ## Tip

   Can you detect a distributed deadlock locally on one machine?

6. List three of the eight fallacies of distributed computing? Why are they fallacies?

7. De-activation is a technology used to preserve server resources where a server which provides remote objects to clients can de-activate those remote objects, e.g. if they haven't been used for a while. Clients should not know about this. What must the server do to avoid surprises for the clients?

   ## Tip

   When a request comes in for a de-activated remote object. What must the server do to re-create (activate) the object again?

8.

A distributed persistent object has two identities (a row in a database e.g. and an identity as a remote object e.g. in an Object Request Broker. Where would you hold these two identities?

Tip

If a call from a client arrives at a server, how would the server find the right persistent identity of the remote object?.

9. Two remote objects are related and should be updated in one operation. What kind of service do you need for this if you assume that both objects can be on different systems and that multiple operations can happen concurrently?

Tip

Booking a trip is an example of this kind of problem: the booking service needs to both reserve a flight and book a hotel room in one go. If only one part of this composite request succeeds the whole operation would be useless.

10. Explain distributed 2-phase commit. Why is it called a voting algorithm?

Tip

The trick is in realizing that ANY time something can go wrong in a distributed system without the participants knowing about it automatically.

11. What happens during a flat transaction if one of the participants calls for a rollback?

Chapter 6. Distributed Systems Security

1. What is the core problem of passwords and why are they even worse in distributed systems

   ## Tip

   Think about who must know passwords and what this means in distributed systems

2. What is a distributed denial-of-service attack and how does it work?

3. What is the big advantage of public key mechanisms in distributed systems?

   ## Tip

   Here the important question is: what do you NOT have to know when public keys are used?

4. You receive a public key certificate from a caller. What should you do to verify this certificate?

   ## Tip

   Think about lost or stolen keys

5. Which web mechanism is used to combine data about a specific user across different web sites?

   ## Tip

   To combine data from visits to different web sites one anchor or key is necessary that stands in for the user.

6. What is end-to-end security in distributed systems? Is this a problem with e.g. e-mail?

   ## Tip

   Think about intermediates located between sender and receiver. What role do they play and what can they do? Do they affect security?

   .

7. What is the purpose of a firewall?

.

8. You want to send mail anonymously. What infrastructure can you use? How does it work?

Tip

Quite a tough problem because both authentication as well as full anonymity is very hard to achieve on the internet. Remember the onion-pattern of packing requests.

Chapter 7. Distributed Components

1. What were the reasons that middleware moved from distributed objects to distributed components?

   ## Tip

   Is it ok to use a local design (interfaces) for a distributed system? Or are distributed systems different and in what respect? Would you use fine grained object methods for remote objects?

2. List some features of components. If you have a software company - why would building components be an interesting idea?

   ## Tip

   Building components only makes sense if you can sell them many times. What would keep you from selling a component to different customers? What do components need to work everywhere? How are all these environments different?

3. Component technology separates business logic from infrastructure like security, transactions etc. The infrastructure services are nowadays provided automatically by tools in combination with runtime containers.Why?

   ## Tip

   Who writes business logic? Who knows how to write transactions, security code etc.? If you need to write a bookkeeping application, do you want to create transaction processing code? Deal with how security works?

4. Enterprise Java Beans distinguishes stateful and stateless beans. Which ones scale better?

   ## Tip

   Which one is tied to resource consumption? Resources are typically what makes servers critical.

5. Enterprise Java Beans has a so called "deployment descriptor" which contains a lot of configuration information. It says descriptively what transaction behavior is required by the component, security roles defined by the component and if the component want the container to store data or whether the component wants to do this by itself. Now, why does EJB put those informations into a descriptor instead of the component code?

6. Enterprise Java Beans introduced so called "local interfaces" in version 2.0. Before that every enterprise java bean object was a remote object. Now two EJBs residing in one VM can talk to each other through

local interfaces. Why did EJBs add those local interface?

Tip

An interesting question is whether a local EJB call could ever be so fast as a regular java method call or if there will always something more involved with an EJB call. What could that MORE be? What does EJB offer and promise to the installed components?

7. Modern component technology is organized after two meta-design patterns: Separation of concerns and separation of context. Can you explain what gets separated and why?

Tip

A software product usually needs to handle many different concerns, not only how the business logic works. And it should run in many different contexts with the code possibly unchanged.

Chapter 8. Distributed Systems Management

1. Applications used to be single images which were installed locally on machines and which were largely self-contained. Distributed objects and components changed this radically and introduced a lot of new problems. Explain which problems are new with distributed objects or components.

   ## Tip

   Draw a little diagram of objects or components that work together on different hosts. Some even share the use of certain objects. Now you want to take one of the hosts involved down to change a disk drive. What kind of things would you like to know in advance?

2. Components should be "black boxes" for their clients. Clients should only know interfaces to be independent of changes in the implementations. Is this abstraction useful for distributed system management or does it need to know more about internals?

   ## Tip

   This is the place where you will recognize that interface/implementation is just another abstraction useful for software designers and programmers - like object-orientation.

3. Distributed System Management relies on an information model of those systems. List some items which need to be in this information model.

   ## Tip

   Put yourself into the position of a responsible person for the operations of a large company. What do you want to know about the running systems?

4. List services a distributed management package needs to provide.

   ## Tip

   How could system management software make your live easier if you are responsible for running a lot of distributed applications? What do you need to know in case of problems?

Chapter 9. Designing Distributed Systems

1. Name two mechanisms that can be used to ensure performance in distributed systems

   ## Tip

   Think about the "distributed fallacies": if a request takes a long time between requester and sender, what could you do with the data? Do you always get the data just when they are requested or can you separate cient request from backend request?

2. List some architectures that you need to define for distributed systems

   ## Tip

   If you want to avoid performance problems, what do you need to know about other services? If you want to cache data, what do you need to know about the data? If your user base might grow considerably over the time, what should you be able to do with your machines?

3. Why is extreme testing a part of necessary architectural validation of your design? What can happen if you delay it until the end of development?

   ## Tip

   Think about "paper-pushing" projects that end in desaster

Chapter 10. Web Services

1. Do web services have something similiar to a naming service? And to an Interface Definition Language?

   ## Tip

   How does a client detect a web service? How does it find out about service methods?

2. Web services use an XML based protocol (SOAP) to send requests and responses. What are the advantages and disadvantages of using XML for remote calls?

   ## Tip

   XML is a text based protocol. Who reads those messages? Can you use all programming languages for web services?

3. Web Services are said to be "firewall friendly" because they can pass through most firewalls. Why can they do this? Why can't CORBA, dcom or RMI do this? Is this technically justified?

   ## Tip

   Sometimes just using a different name for a technology can make it look either more harmless or more dangerous.

4. The best practices for the use of web services across the internet reflect a lot of common sense for distributed applications. What was recommended?

   ## Tip

   Web services are intended to work across the internet. What kind of communication qualities can you expect with respect to reliability? Speed?

Chapter 11. Peer-to-Peer Computing

1. How can peers locate other peers?

   ## Tip

   Think about peers in the local network and peers at the internet.

2. Some p2p systems use a central server. What are the advantages and disadvantages of this architecture?

   ## Tip

   As usual, a good start for the answers is always to ask about the behavior in case of failures (client side, server side) and how big the consequences are. Another way to tackle those questions is to ask how the respective topology provides the typical distributed services like searching, finding, transactions, security (authentication) etc.

3. Some p2p systems avoid central servers completely. What are the advantages and disadvantages of this architecture?

   ## Tip

   See the hint from above "central server".

4. What does the term "edge of the internet" mean?

   ## Tip

   Take one of your machines at home with some form of connectivity to the internet through dialup. What are its properties?

Chapter 12. Reprsentational Architectures

1. What are the differences between a URL, URI and URN?

2. What are the basic operations behind the WWW, a tuple-space, a wikiwiki? Pick one and compare it to a remote procedure mechanism used to build some application

   ## Tip

   One way to answer this is to take the basic operations of e.g. the http protocol (really just the basic commands) and compare them to the remote interfaces used in a typical RMI application (e.g. one of our games). Which one looks much more complicated?

3.

4.