UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Excercise Sheet 1
Solution

Lecture Distributed Systems
Summer Term 2020

Exercise 1: Heterogeneity and Openness

a) Since both computers are connected to the Internet, we can assume that the Internet protocols deal with the differences between networks.

- Because the computers have **different hardware**, the differences in the representation of the data elements in the request and response messages from client to server must be considered. A common standard is defined for each type of data element that must be transferred between the server and the clients.
- Since computers run **different operating systems**, we have to cope with different operations to send and receive messages or to express invocations. Therefore, at the Java/C++ level, a common operation would be used which will be translated to the particular operation according to the operating system it runs on.
- We have two **different programming languages** C++ and Java, they use different representations for data structures such as strings, arrays, records. A common standard will be defined for each type of data structure that must be transmitted between the object and its clients and a way of translating between that data structure and each of the languages.
- We may have **different implementors**, e.g. one for C++ and the other for Java. They will need to agree on the common standards mentioned above and to document them.
- Also, to add new services in an existing open system, the standards mentioned in point a) must have been agreed. Briefly repeated:
  - The distributed system uses a common set of communication protocols (probably Internet protocols).
  - It uses an defined standard for representing data items (to deal with heterogeneity of hardware).
  - It uses a common standard for message passing operations (or for invocations).
  - It uses a language independent standard for representing data structures.

  But for an open distributed system, the standards must have been agreed and documented before the implementation is provided. This means that the implementations must comply with those standards. In addition, the interface to the service must be published so that, when it is added to the system, both existing and new clients can use it. The publication of the standards allows parts of the system to be implemented by different software vendors and to work together later without problems.
- Each request to access a protected operation must include the identity of the user making the request. The problems are:
  - Defining the identities of the users. Using these identities (IDs) in the list of users who are allowed to access the protected operations at the implementation of the BLOB object and in the request messages.
  - Ensuring that the identity supplied comes from the user it purports to be and not some other user pretending to be that user.
  - Preventing other users from replaying or tampering with the request messages of legitimate users.

  Additionally:
  The information returned as a result of a protected method must be encrypted if it can be intercepted by unauthorized users.

Exercise 2: Migration of resources

A migration of resources is carried out, for example, to

- reduce communication delays (placing objects on a server in the local network from which they are most frequently used),
- balance the compute load and/or memory utilization between different servers.

Exercise 3: Response time of client applications

The response time of client applications is affected by the following factors:

a) the utilization of the server,

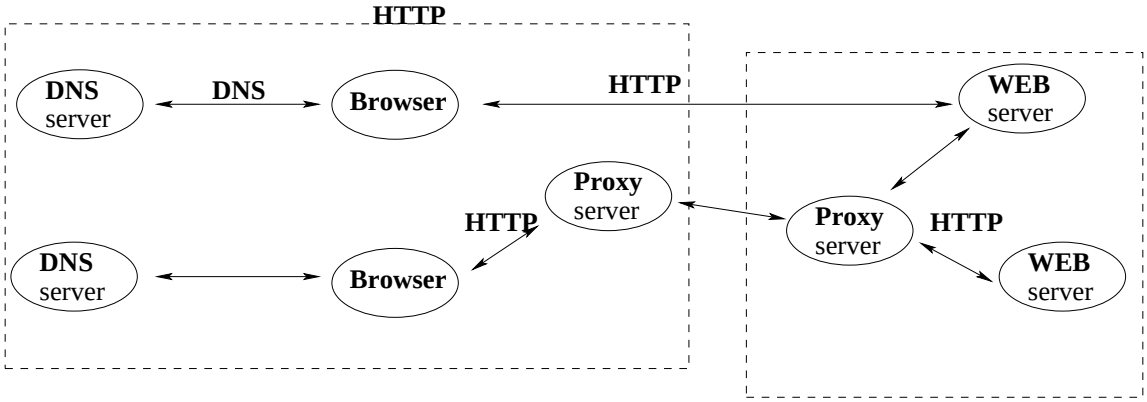
b) the latency when exchanging request and response messages (due to the OS layers and the middleware in the client and server),

c) the load in the network.

Caching helps with all of the above problems (especially client caches). Proxy server caches help with a). Replication of the service also helps with a). The use of lightweight communication protocols helps with b).

Exercise 4: Architecture of known client/server applications

WEB:

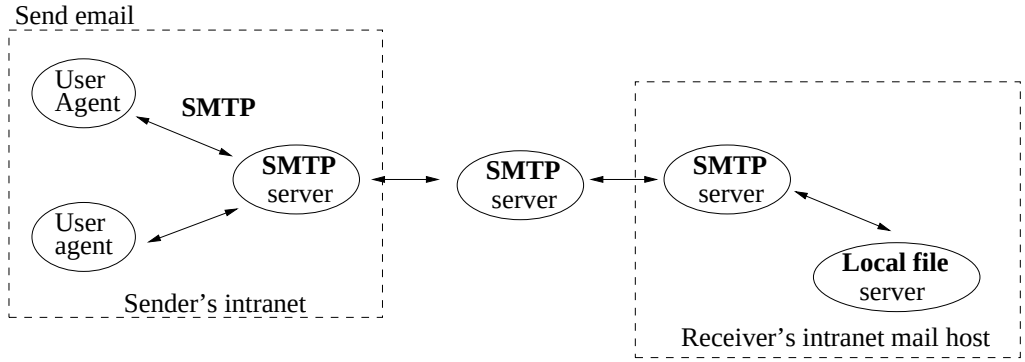


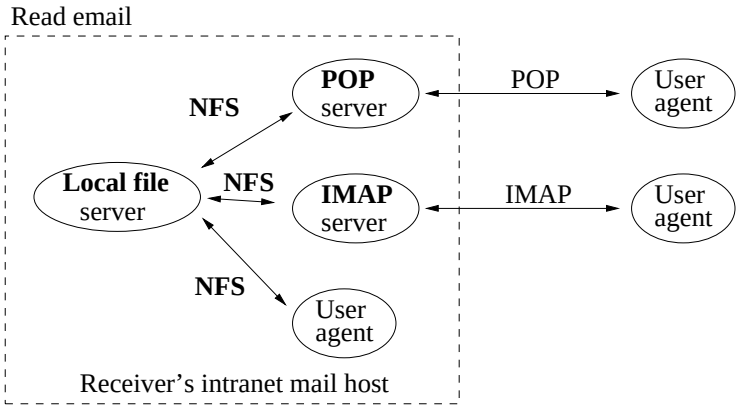
Browsers are clients of the domain name server (DNS) and the web server. Some intranets are configured with a proxy server in between. Proxy servers reduce network delays and traffic when they are in the same location as the client. If they are at the same location as the server, they form a security control point and can reduce the load on the server. The responsibility for data consistency lies with the proxy servers – they periodically check the modification time of the pages at the source web server.

Web pages are stored in the local file system of the server. All the information on the Web is therefore distributed among many Web servers. Replication is not part of the Web protocol, but a frequently visited Web site can provide multiple servers with identical copies of the file system. HTTP requests can be distributed between these servers (for example, via DNS load balancing). In addition, proxy servers support replication by caching recently used pages, just like web browsers.

DNS servers can also be included in all application architectures described below, but are omitted to facilitate discussion.

Email:





Send messages: The User Agent is a client of a local SMTP server and passes each outgoing message to the SMTP server for delivery. The local SMTP server uses email routing tables to determine a route for each message and then forwards the message to the next SMTP server on the chosen route. Each SMTP server processes and forwards each incoming message until the domain name in the message address matches the local domain. In this case, it attempts to deliver the message to local recipients by storing it in a mailbox file on a local disk or file server.

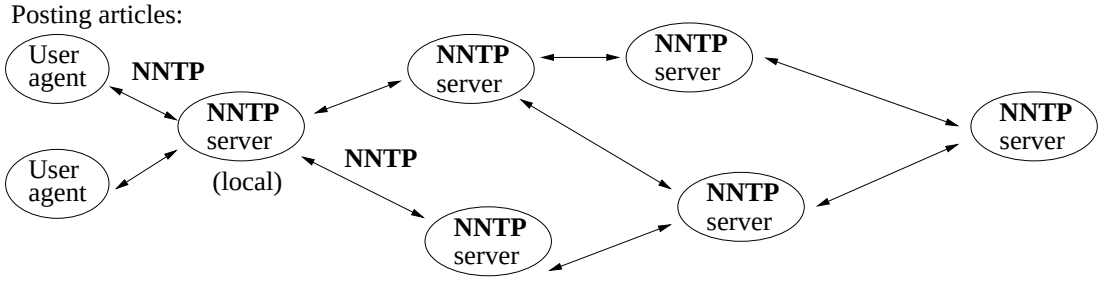
SMTP servers do not necessarily need to have email routing tables for all destinations. Instead, they pass messages for unknown destinations to another server that probably has a matching routing table.

Messages are only stored at their destination (permanently). This means that the email service is based on partitioning only, even if a message with multiple recipients is replicated on multiple destination computers.

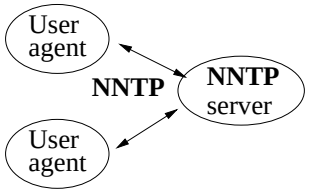
Receive messages: The user agent (the user's mail reader) is either a client of the local file server or a client of an email delivery server such as a POP or IMAP server. In the first case, the user agent reads the messages directly from the mailbox file. In the latter case, the user agent requests information about the contents of the user's mailbox file from a POP or IMAP server and receives the messages from that server for presentation to the user. POP and IMAP are protocols specifically designed to support email access over WAN and slow network connections.

News:

Posting articles:



Browsing articles:



Post news articles: The user agent (the user program for creating and sending news) is a client of a local NNTP server and passes each outgoing article to the NNTP server for delivery. Each article is assigned a unique identifier. Each NNTP server holds a list of other NNTP servers for which it is a news creator – they are registered to receive articles from it. Periodically, he contacts all registered servers, delivers all new articles to them, and requests all new articles they already have, although he does not have them yet (the unique identifier helps determine which ones these are). This replicates the articles of a newsgroup on all the servers that need them.

Reading news articles: The user agent (the user's news reader) is a client of a local NNTP server. The user agent requests updates for all newsgroups for which the user has registered, and displays the news.

Exercise 5: Session tracking

Session tracking using cookies:

A cookie consists of a name, a value, and one or more attributes. An attribute can be e.g. EXPIRES=dateValue, Version or SECURE.

The session ID is transferred to the client within a cookie via the browser or a script so that the server can recognize it and process information specific to it. Each time you return, the text information from the cookie is sent to the server. The cookies can be transmitted in the headers of requests and responses or through a script such as JavaScript or within Java applets and always remain stored locally on the client.

Cookies make it possible to uniquely identify a client, which means that user profiles can be created within a website that could also be misused to link to personal data. In order to be able to partially understand the further surfing behaviour of a user, third-party cookies are required, which are usually found in advertising banners or other media content on a website. If the user has visited one of the associated websites, the information can be linked.

However, the user has the option to allow or block cookies, where a more detailed distinction can be made. In the case of public computers, care should be taken to ensure that cookies are not stored permanently.

Session tracking using URL rewriting:

URL rewriting is used, for example, when the session ID is appended to the end of a URL in order to trace the session of a client, which is browser-independent and rather simple to implement. The problem here is a little on the server side, since the identifiers have to be assigned to the sessions here.

For the user the extension of the URL is visible and has the disadvantage that this URL is no longer accessible after a while, because it refers to a certain session.

Furthermore, the length of the URL may be problematic, since browsers, HTTP protocols, search engine indexing, etc. can only handle a maximum length of 255 characters per URL.

Exercise 6: Server implementation

a) In the single-threaded case, a cache hit takes 15 and a cache miss 90 milliseconds. The weighted average is 2/3*15 + 1/3*90, so the average request takes 40 milliseconds and the server can process 25 requests per second. For a multithreaded server, waiting for disk requests overlaps, so each request takes 15 milliseconds and the server can process 66 2/3 requests per second.

b) Yes, for two reasons: First, threads need memory to set up their own private stack. As a result, many threads can take up so much memory that the server no longer works properly. Another serious reason is that independent threads for an operating system run in a chaotic way. In a virtual memory system, it can be difficult to create a relatively stable resident set, leading to many page faults and thus many I/Os. So using many threads can lead to a performance degradation resulting from page thrashing.