Alexandros Karakasidis

Assignment 1

Part I

Goal 1: The type of resources shared in the Web is mainly documents. Other types of files that are shared by the means of the Web are multimedia files, executable programs etc. However, there can be access via web protocols to other resources, such as printers and web telephony.

The two problems of sharing are security and unwanted communication. In the case of the Web, most of the security issues deal with setting up a secure channel between a client and a server. The approach most commonly used for this purpose is the use of the Secure Socket Layer (SSL) and of its standardized update Transport Layered Security (TLS).

Considering the problem of unwanted communication, the main solutions have to deal with the policy followed by the user. In specific, a type of unwanted communication is the pop-up windows. The user may avoid this by disabling the execution of mobile code (e.g. Javascript) by the browser. One other way of avoiding unwanted communication is by using proxy servers. These can filter out some of the non-desired traffic. However some unwanted communication such as advertisements, cannot be avoided.

Goal 2: The following table depicts the transparency offered by the web for each of the types of transparency:

Type of Transparency	Transparency offered by the Web
Access Transparency	High. The user does not know whether the resource accessed is for instance on a Unix or a Windows machine
Location Transparency	High. The user accesses a resource by its URL. It does not know where this resource is actually located.
Migration Transparency	High. The user accesses a resource by its URL. It does not know where this resource is actually located.
Relocation Transparency	None. A resource cannot move from a server to another while in use.
Replication Transparency	Not always Possible. This type of transparency exists when we use, for example, server clusters. On the other hand, by using a number of different mirrors, this type of transparency does not exist,
Concurrency Transparency	High. A user cannot know how many other users may be viewing a Web site, or requesting a document. All users have mainly read permissions of the documents.
Failure Transparency	Low. If, for instance a DNS server crashes, then the user cannot connect to the desired Web site even though this might be working properly.
Persistent Transparency	High. The user does not know if a retrieved document was stored in the hard disk, or directly displayed from the memory.
Performance Transparency	Not always Possible. This type of transparency exists only if server clusters are used, or if mirrors of a server exist
Scaling Transparency	High. New Web sites are created every day, without an impact to the system's structure.

Goal 3: The Web is a highly open system. There can be interaction between clients and servers, no matter what their underlying network or operating system is. The offered services follow standard rules, which are self – descriptive regarding their semantics and syntax. This is achieved by the utilization of markup languages (HTML, XML).

The rules used in the Web are formalized in protocols. Briefly these are: HTTP, FTP and MIME for file retrieval, DNS for name resolution, and TLS for secure connections. Other protocols closely related to HTTP, are the Internet Message Access Protocol (IMAP) for the exchange of mail messages and the Network News Transfer Protocol (NNTP). HTML provides for neutral and complete specifications over the form of the documents. It exhibits high interoperability, since it facilitates access and location transparency to a high extend. Documents are liked with each other, regardless of their host operating system or their location. It is highly portable, since pages developed by means of HTML and Javascript can be server by any host on the Web. It is also extendable, by using helper applications that can be added as modules, to the client web browsers, and it can provide definitions for the internal state of the system using status codes.

The Web also separates policies from mechanisms. Some examples are the following:

- Cookies: A mechanism mainly used for purposes of personalization. The Web server sends to the client a short text file. The client's policy might be to accept all cookies, cookies originating only from known web sites, none, etc.
- Mobile Code: Javascript, Java etc. This is transferred from the server to the client and is executed locally. The client can allow or not the execution of Mobile Code.

Goal 4:

- Scalability along Size. The web is scalable along size, since it can accept more and more users. In order to achieve this, techniques such as caching, mirroring and server clustering are facilitated in order to achieve better response times and reduce latencies.
- Geographical Scalability. The web is geographically scalable, since it is highly distributed and replication used by the means of mirrors offers high availability and global access.
- Administrative Scalability. The web is organized in a distributed administrative architecture, which allows it to easily expand. Local clusters can have their own mirrors or caches in order to minimize latencies.

Part II

a) The paper focuses on a principle called the end-to-end argument. This principle, suggests that functions placed at low levels of a system, forming a communications subsystem, may be redundant or of little value when compared with the cost of providing them at that low level. The main argument is that lower architecture levels cannot provide reliability. An alternative approach called "end-to-end check and retry" is proposed, which is presented by suggesting an example according to which, in order to have consistent file transfer, the entire file should be checked for consistency after the completion of the operation. If the checksum value is wrong, the file has to be retransmitted. Finally some examples are provided in order to validate the end-to-end argument. The main points of the authors presented in these examples are the following:

- "*The end-to-end check of the file transfer application must still be implemented no matter how reliable the communication system becomes*."
- *"Since the lower level subsystem is common to many applications, those applications that do not need the function will pay for it anyway"*
- *"The low-level subsystem may not have as much information as the higher levels, so it cannot do the job as efficiently"*
- Layered architectures are not sufficient because *"What the application wants to know is whether or not the target host acted on the message"*
- *"Real time transmission of speech has tighter constraints on message delay than on bit-error rate. Most retry schemes significantly increase the variability of delay."*
- A layered communications subsystem cannot provide for security
- Retransmission of individual packets in a network communication, is worse than fully checking a file for consistency and then retransmit if necessary
- In order to ensure FIFO delivery, *"an independent mechanism at a higher level than the communication subsystem must control the ordering of actions."*

The original text is quoted here for reference purposes.

b) The arguments of the paper's authors are all based on the hypothesis that lower architecture layers need not provide perfect reliability. TCP is in the lower layers of the architecture stack and provides for reliability. Most of the arguments presented, fall by the technologies employed by the Internet stack protocols and especially TCP. Thus, TCP violates the argument that "*the end-to-end check of the file transfer application must still be implemented no matter how reliable the communication system becomes*."

The argument that *"since the lower level subsystem is common to many applications, those applications that do not need the function will pay for it anyway"* can fall having in mind that in modern layered architectures, applications use only the necessary resources.

The next statement that *"the low-level subsystem may not have as much information as the higher levels, so it cannot do the job as efficiently",* is invalid, since in the contemporary layered system architectures, each layer receives the appropriate amount of information in order to fulfill its task.

One other statement that sounds strange, having in mind the internet protocols, is that layered architectures are not sufficient because *"What the application wants to know is whether or not the target host acted on the message;"* This is achieved at present by means of layered protocols.

One other argument against layered network protocols is that *"real time transmission of speech has tighter constraints on message delay than on bit-error rate. Most retry schemes significantly increase the variability of delay."* Speech or other real time types of communication neither require error correction codes nor acknowledgements, since this kind of transmissions contain redundant information, the loss of which does not affect the overall communication scheme. UDP offers these functions.

The arguments of the authors that a layered communications subsystem cannot provide for security are not valid, since the utilization by Internet protocols of SSL and symmetric and asymmetric encryption techniques, along with digital signatures, are a good counter argument.

It is also supported that retransmission of individual packets in a network communication, is worse than fully checking a file for consistency and then retransmit if necessary. This is obviously not correct since the overhead in the second case is significantly greater than in the first case.

Another statement that falls, considering the operation of TCP, is that, in order to ensure FIFO delivery, *"an independent mechanism at a higher level than the communication subsystem must control the ordering of actions."* TCP while being a communications subsystem, ensures FIFO delivery.

c) The most well known example of system which violates the end – to – end argument is the internet, where the communication is fault tolerant and achieved through a layered protocol suite. Files are copied between hosts, without having to be checked for consistency at either end when transmission ends.