IK2218 Protocols and Principles of the Internet EP2120 Internetworking

Homework 3

Solutions due: 19:00, October 6, 2023 Review due: 19:00, October 10, 2023

1. Socket API (15 p)

The pseudo-code sample below (with most details omitted) describes an application that uses the socket interface (API) for communication

```
s = socket(...); bind(s, ...); while (true) {
recvfrom(s, ...); if (fork() == 0) {
            ProcessRequest(...); sendto(s, ...); exit();
    }
}
```

(a) Is the sample code for a client or server? Does it use TCP or UDP? Explain your (5 p) answer.

Ans:

The given sample code represents a server that uses UDP for communication. Because it binds to a socket (using bind) and monitors for incoming data (using 'recvfrom'), the code is structured as a server. Servers often bind to a specified port and listen for incoming client connections or data. In the code sample, the usage of the 'recvfrom' and 'sendto' functions shows UDP communication. UDP is a connectionless, datagram-based protocol that is frequently used for real-time applications and straightforward request-response exchanges, both of which require little overhead and minimal delay. Because each packet (datagram) is treated separately in UDP and there is no specific connection setup, it is appropriate for use in situations where speed is more important than reliability.

(b) The textbook gives two examples of communication using the socket interface: (5 p) 1) connection-oriented, concurrent communication, and 2) connectionless, iterative communication. Characterize the communication in the sample code using the same terminology.

Ans:

UDP (User Datagram Protocol), a connectionless transport protocol, is used in the sample code. There is no explicit establishing of a connection between the sender and recipient with UDP. Each message (datagram) is self-contained, and the sender does not need to connect before transferring data.

The code sample uses a loop (while (true)) to simultaneously handle all incoming messages while continuously listening for them using recvfrom. When a message is received, it uses the fork() function to start a new process and handle the request concurrently. ProcessRequest is executed individually by each child process. The idea of "concurrent communication" as it is presented in the textbook, in which several clients can communicate with the server simultaneously, is consistent with the concurrent processing of incoming requests. In order to make the most effective use of the server's resources, each client's request is processed simultaneously without waiting for the completion of earlier requests.

Therefore, the sample code denotes connectionless, iterative communication using UDP where each message is independent and concurrent communication by process of forking which handles incoming requests concurrently.

(c) In practice, this kind of communication is not frequently used. Give an explanation (5 p) why the designer of this particular application still may have decided to use it.

Ans:

Some of the reasons why designer of this particular application may have used it are :

1) Due to the fact that TCP uses procedures for connection setup, acknowledgment, and retransmission, UDP has a lower overhead than TCP. Because of its quicker transmission, UDP can be favored in situations where minimizing latency and obtaining higher data transfer speeds are essential, like online gaming.
2) For some applications, timely data transfer is more important than guaranteed delivery, hence real-time communication is necessary.
3) UDP is appropriate for situations when data needs to be transferred simultaneously to numerous receivers since it supports broadcasting and multicasting. UDP might be effective, for instance, in live video streaming to several viewers.
4) To meet the unique requirements of the application, developers can use UDP to design specialized error-handling techniques.

5) To retain compatibility and connectivity, new applications may choose to use UDP while communicating with older or legacy systems.

2. Web (35 p)

Suppose that you are using your web browser and click on a link on a web page, which causes the following HTTP request to be sent:

```
GET /feathers/swordfish.html HTTP/1.1
Host: www.duck.org
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5)
Accept-Encoding: gzip,deflate,sdch Accept-Language: en-US,en;q=0.8
```

(a) Which web document does the browser request? Answer by giving the URL. The (5 p) answer should be a complete and correct URL.

Ans: URL: http://www.duck.org/feathers/swordfish.html

(b) Describe the TCP connection policy that the client requests. (2 p)

Ans:

A persistent or keep-alive connection between the client and server is requested, displayed by the "Connection: keep-alive" header. This means that, if possible, the connection should be maintained open and used for any further requests to the same server after the original request and answer. This improves performance and lowers latency, especially when loading several resources from the same server in a brief period of time, by eliminating the overhead of creating a new TCP connection for each HTTP request. It is a method created to reduce the overhead involved in creating and tearing down TCP connections.

(c) The server gives the following response: (3 p)

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Date: Sun, 20 Sep 2020 12:34:56 GMT
Server: Apache/2.2.3 (Red Hat)
Content-Length: 286
Cache-control: public, max-age=600 Keep-Alive: timeout=8, max=120
Content-Type: text/html; charset=utf-8

more data...
```

Describe the TCP connection policy with which the server responds.

Ans:

In the server's response, the header "Connection: Keep-Alive" denotes that the server accepts the client's request to establish a persistent or keep-alive connection. This indicates that the server is ready to maintain the connection for additional requests from the client. Also, it provides specific timeout and max connection value for managing the keep-alive connection efficiently.

(d) Is the returned object cacheable? If so, describe how it may be cached. (5 p)

Ans:

Yes the returned object is cacheable as the server’s response includes a cache header. The “public” in the header denotes that the response can be cached in both private and shared caches. Also, the “max age= 600” implies that the response can be cached for a max of 600 secs. The response can be stored by client and intermediate caches for 10 minutes. Also, the date header is used for expiration and validation of the cache.

(e) Assume that the web object that the client fetches is an HTML file that references (15 p) four other web objects (which do not, in turn, use any further web objects).

The browser can use two different strategies to speed up loading the page: 1) multiple non-persistent connections in parallel, and 2) a single persistent connection with pipelining. For each of the two cases, find the time it takes from that the user clicks on the link until the web page can be presented on the screen. That is, the total time it takes to fetch all objects that are needed for the page.

The round-trip time between client and server is RTT. Assumed that all objects are very small, and the connection is fast, so transmission time is negligible, and so is processing time on the server. Moreover, you need not consider the time it takes to close down a TCP connection (it takes place in the background).

Ans:

1. Multiple Non-Persistent connections in parallel:

   We have five separate HTTP request/response transactions (one for the HTML file and one for each of the four objects).

   A new TCP connection must be established for each transaction, adding to the RTT overhead.This scenario's time frame can be computed as follows:

   Time to fetch the HTML file + (first object + second object + third object + fourth object)

   Total time = 2+2= 4RTTs

2. Single Persistent Connection with Pipelining:

   Here, overall HTTP transactions (HTML file and four objects) are sent over a single persistent connection with pipelining. There is no need to wait for the RTT between requests because pipelining enables many requests to be sent before receiving responses.

   In this case, the time required can be computed as follows:

   Time to fetch the HTML+ (first object + second object + third object + fourth object (no additional RTT, as it can be pipelined))

   Total time = 2 RTT (HTML) + 1 RTT (subsequent requests) = 3 RTT

(f) Page load time is important on the web, but it is not the only thing that matters. (5 p) Not considering page load time performance, give two advantages of persistent connections over non-persistent connections.

Ans:

Persistent connections have various advantages over non-persistent connections that go beyond just faster page loading. Here are two main advantages:

- Reduced Connection Latency/Overhead:
  Persistent connections substantially lowers the latency or overhead by creating and tearing down the TCP connections for each HTTP request. This lower latency helps in efficient utilization of resources by handling multiple requests and responses in a single connection. Also, the requests can be sent immediately without delay which minimizes the latency and response and interaction of the web application.
- Better support for keep-alive and pipelining:
  Persistent connections allow the reusing of the same connection for multiple requests and responses resulting in a smooth user experience and reducing the impact on the resources of the server. Also, these connections

support pipelining where HTTP requests are sent to server for processing, and the server responds sequentially to the requests therefore improving the efficiency of webpage loading.

3. DNS (25 p)

Use the "dig" lookup tool to get the IP address of KTH's web server "www.kth.se". Try different name servers, and specify the DNS server as an argument to dig (the server's IP address prepended with '@'). Use the following four commands:

dig @193.0.14.129 www.kth.se dig @213.108.25.4
www.kth.se dig @130.235.20.5 www.kth.se dig
@130.237.72.200 www.kth.se

(a) Explain the results: Describe the responses from the four DNS servers. What do (20 p) the responses say? Note that the result may differ depending on your location. Try to run the commands on KTHs network, and outside KTH.

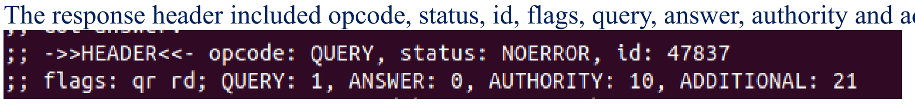
We distinguish between four kinds of name servers: root, TLD, authoritative, and local. You can deduce just from studying the responses what kind of DNS server it is (the flags are useful here, among other things). For each of the four cases, describe the kind of name server that is responding to your query. Explain what the response contains. Also, for each name server, explain whether or not the answer contains the IP address you are looking for. You only need to discuss the responses at a general level – you should not discuss or describe the details of the messages, the different fields, their contents, etc. However, you probably need to study the responses carefully in order to be able to explain them.

Ans:

1. dig @193.0.14.129 www.kth.se

   The DNS server with IP address 193.0.14.129 seems to be a Root DNS server.

   The response header included opcode, status, id, flags, query, answer, authority and additional.

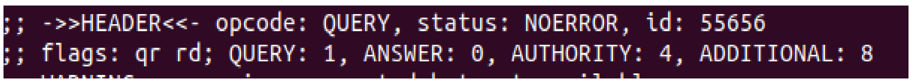


```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47837
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 10, ADDITIONAL: 21
```

   In the above output 'opcode' indicates standard query, status with 'noerror ' means that the query was successful and the id includes the unique identifier given to the query. In the flag section 'qr' denotes response, 'rd' denotes recursion desired. The 'Query: 1' denotes there was one query in the request, 'AUTHORITY:10 ' contains info about authoritative name servers which in this case is 10, and 'ADDITIONAL : 21' denotes that additional resources are included in the response.
   Also, the IP address of "www.kth.se" is not directly shown in the response.

2. Dig @213.108.25.4 www.kth.se dig

   The DNS server with the IP address 213.108.25.4 is probably a TLD DNS server in charge of the.se domain when this request is made.



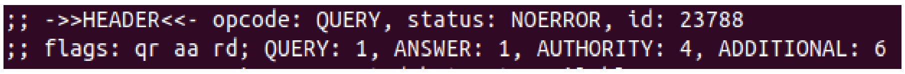
```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55656
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 8
```

   In the above output 'opcode' indicates standard query, status with 'noerror ' means that the query was successful and the id includes the unique identifier given to the query. In the flag section 'qr' denotes response, 'rd' denotes recursion desired. The 'Query: 1' denotes there was one query in the request, 'AUTHORITY: 4 ' contains info about authoritative name servers which in this case is 4, and 'ADDITIONAL : 8' denotes that additional resources are included in the response.

   Similarly to the first address, the IP address of "www.kth.se" is not directly shown in the response but displays an authoritative name server for the domain of "www.kth.se".

3. dig @130.235.20.5 www.kth.se dig

The DNS server with IP address 130.235.20.5 seems to be the authoritative server for “kth.se”.



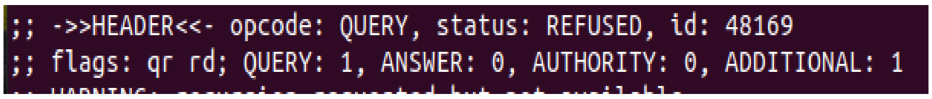
```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23788
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 6
```

In the above output ‘opcode’ indicates standard query, status with ‘noerror ‘ means that the query was successful and the id includes the unique identifier given to the query. In the flag section ‘qr’ denotes response, ’aa’ indicates that “Authoritative answer” flag is set, ‘rd’ denotes recursion desired. The ‘Query: 1’ denotes there was one query in the request, ‘AUTHORITY: 4 ‘ contains info about authoritative name servers which in this case is 4, and ‘ADDITIONAL : 6’ denotes that additional resources are included in the response.

The IP address for “kth.se” can be seen in the ”answer section” (130.237.28.40)

4. dig @130.237.72.200 www.kth.se

The DNS server with the IP address 130.237.72.200 appears to be a local DNS server because it refused the query made, although if it was an authoritative server, the response should have contained the phrase 'NOERROR'.



```
;; ->>HEADER<<- opcode: QUERY, status: REFUSED, id: 48169
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
```

In the above output ‘opcode’ indicates standard query, status with ‘REFUSED ‘ means that DNS server refused to process the query and the id includes the unique identifier given to the query. In the flag section ‘qr’ denotes response, ’aa’ indicates that “Authoritative answer” flag is set, ‘rd’ denotes the recursion desired. The ‘Query: 1’ denotes there was one query in the request, ‘AUTHORITY: 0 ‘ it doesn’t contain info about authoritative name servers, and ‘ADDITIONAL : 1’ denotes that additional resources are included in the response.

(b) DNS uses UDP, not TCP. Unlike TCP, UDP does not guarantee delivery of data. (5 p) What happens if a DNS message (query or response) is lost? Is it a problem, and if so, how is it handled?

Ans:

DNS mainly uses UDP for communication in the transport layer . UDP is a connectionless and less unreliable protocol when compared to the Transmission control protocol(TCP) and the delivery of data is not guaranteed everytime and there is inbuilt function for retransmission of lost packets. Therefore methods such as Retransmissions, timeout and retry, multiple queries, recursive resolution & time to live are used.

- When a DNS query or response is lost the sender can retransmit the message and if the response is received, the transmission is processed as usual.
- A DNS resolver may consider a query lost and retry it if it sends a request to a DNS server and doesn't hear back within a certain timeout period.
- DNS resolvers occasionally have the capability to simultaneously transmit the same request to a number of DNS servers. Responses from the other servers are ignored and are only used by the first server to respond with a valid response. The effect of lost messages can be reduced with the help of this method.
- Recursive resolution, where the resolver asks many servers for the same information, is a request that can be made in an DNS query.
- A TTL value is included in DNS resource entries to indicate how long the data should be regarded as valid. The impact of lost interaction is reduced because even if a query or response is lost, the resolver can still use the cached data until it expires.

4. Email (25 p)

Internet email consists of several components that have different functions. Consider the case when Alice wants to send an email to Bob. The email will be transferred in two steps, via two different mail servers, before Bob gets the mail.

(a) What are the two mail servers? Describe their main functions. (5 p)

Ans:

- Outgoing Mail Server:
  This is also known as SMTP (Simple Mail Transfer Protocol) server which is mainly responsible for sending outgoing emails from client to receiver (who in this case is Alice and Bob). When Alice sends the mail, the email client contacts the SMTP server who then formats the mail correctly and forwards it to Bob's mail address. The SMTP server can also check the recipient's domain to know the next hop server for the email.

- Incoming Mail Server:
  This server is mainly responsible for receiving and storing the emails that are sent to Bob's address. POP3(Post Office Protocol version 3) and IMAP (Internet Message Access Protocol) are the two protocols primarily used in this server. POP3 is responsible for downloading mails (storing locally) to Bob's email client which are later deleted after downloading. Where as IMAP handles the emails on the server and keeps them in sync with Bob's email client allowing him to access mails from multiple devices. Also, the Incoming Mail server is responsible for authenticating Bob's access & organizing mails to Bob's email client.

(b) For each of the two transfers, explain what protocols are used, both at the appli- (5 p) cation layer and at the transport layer.

Ans:

1. Transfer from Alice's email client to the outgoing mail server:
   - **Application Layer Protocol:** ***SMTP (Simple mail transfer protocol)*** is responsible for exchange of emails between the sender (Alice) and Outgoing Mail server.

   - **Transport Layer Protocol:** SMTP uses the ***Transmission Control Protocol (TCP)*** at the transport layer to communicate between Alice's email client and the SMTP server, TCP makes sure that data is reliably and sequentially transmitted.

2. Transfer form Bob's incoming mail server to Bob's email client r:
   - **Application layer**: ***POP3 (Post Office Protocol version 3) & IMAP (Internet Message Access Protocol)*** are the two protocols used here for the retrieval, synchronization, and Management of emails from the server to Bob's email client
   - **Transport Layer Protocol:** Both the protocols operate over TCP for reliable and secure transfer of data between Bob's email client and the server.

(c) For each of the two transfers, explain what party is client and what party is server, (15 p) and how the client gets the location (the domain name) of the server. Be specific when you describe how the domain names are obtained

Ans:

a) Transfer from Alice's Email Client to the Outgoing Mail Server:
The client in this first transfer is Alice's email client, and the server is Outgoing Mail Server (SMTP). To determine the location of the SMTP server, Alice's email client depends on the email configuration settings provided by her email service provider. The domain name of the SMTP server is typically found in the SMTP settings. This domain name is used by Alice's email client to find an SMTP server for email service.

b) Transfer from Bob's Incoming Mail Server to Bob's Email Client:

In this transfer Bob’s email client is the client and Incoming Mail server (POP3 or IMAP Server) is the server. To determine the location of the Incoming Mail server , Bob’s email client relies on similar email configuration settings given by email service provider. The domain name of both POP3 and IMAP are specified in the setting of the server allowing Bob’s email client to locate the server for his email service.