1. (10 pt.) Design and describe an application-level protocol to be used between an Automatic Teller Machine, and a bank's centralized computer. Your protocol should allow a user's card and password to be verified, the account balance (which is maintained at the centralized computer) to be queried, and an account withdrawal (i.e., when money is given to the user) to be made. Your protocol entities should be able to handle the all-too-common case in which there is not enough money in the account to cover the withdrawal. Specify your protocol by listing the messages exchanged and their formats, and the action taken by the Automatic Teller Machine or the bank's centralized computer on transmission and receipt of messages. Sketch the operation of your protocol for the case of a simple withdrawal with no errors, using a diagram to illustrate the messages exchanged. Explicitly state the assumptions made by your protocol about the underlying end-to-end transport service.

Answer:

Assumption:

No message lost and error in the middle of transmission.

Stateless

<table>
<thead>
<tr>
<th>Command_ID</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOGIN</td>
<td>This command is for the client to issue a login request to CC with client_ID and the corresponding password.</td>
<td>&lt;LOGIN, client_ID, password&gt;</td>
</tr>
<tr>
<td>BAL_CHAN</td>
<td>This command is for the client to issue a balance checking request to CC with client_ID.</td>
<td>&lt;BAL_CHAN, client_ID&gt;</td>
</tr>
<tr>
<td>WITHDRAW</td>
<td>This command is for the client to issue a withdraw request to CC with client_ID and amount</td>
<td>&lt;WITHDRAW, client_ID, amount&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command_ID</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERIFY</td>
<td>This command is for the CC to confirm the login information to the client.</td>
<td>&lt;VERIFY, client_ID&gt;</td>
</tr>
<tr>
<td>VERIFY_FAIL</td>
<td>This command is for the CC to issue a failure information of the login to the client.</td>
<td>&lt;VERIFY_FAIL, client_ID&gt;</td>
</tr>
<tr>
<td>BAL_INFO</td>
<td>This command is for the CC to return the balance information to the client</td>
<td>&lt;BAL_INFO, client_ID, balance&gt;</td>
</tr>
<tr>
<td>WD_SUCC</td>
<td>This command is for the CC to indicate that the withdraw has been handled successfully at the server side</td>
<td>&lt;WD_SUCC, client_ID, balance&gt;</td>
</tr>
</tbody>
</table>
**WD_FAIL** This command is for the CC to indicate that the withdraw fails at the server side due to not enough balance

<WD_FAIL, client_ID, balance>

**DIAGRAM:**
A simple version of the diagram is shown in the following:

```
   Listen
     |
     V
  Verify
 /    \
|      |
|      |
|      |
Verify  Verify
Success Fail
    |    |
    |    |
    |    |
Balance Checking Account Withdraw Wrong Password
    |    |
    |    |
    |    |
Show Balance Outputs Money Not Enough Money
```

**ACTIONS:**
1. LOGIN: ATM takes card and gets password input: sends user verify message which includes user id and password to Central Computer (CC)
2. VERIFY / VERIFY_FAIL: Receiving the user verify message, CC checks if the combination is correct, if yes, sends verify success message to ATM. Otherwise, sends verify failed message to ATM.
3. VERIFY: Receiving the verify success message, ATM shows further options of withdrawal and balance checking.
4. VERIFY_FAIL: Receiving the verify failed message, ATM shows wrong password alert and returns the card.
5. BAL_CHK: ATM gets balance checking requirement: sends balance checking message to CC including user id.
6. BAL_INFO: Receiving the balance checking message, CC checks this user’s balance and send balance message to ATM.
7. WITHDRAW: ATM gets account withdrawal requirement: sends withdrawal message to CC which includes the user id and the requested amount of money.
8. WD_SUCC / WD_FAIL: Receiving the withdrawal message, CC checks this user’s balance and compare it with the requested amount. If the balance can cover the withdrawal, CC updates this user’s balance information and sends allow withdrawal message to ATM. Otherwise, CC sends forbid withdrawal message to ATM.
9. WD_SUCC: Receiving the allow withdrawal message, ATM outputs money.
10. WD_FAIL: Receiving the forbid withdrawal message, ATM shows not enough balance alert and show options of withdrawal and balance checking

**Grading:**
- 2 If message format is not defined
- 3 If no diagram
2. (10 pt.) How does SMTP mark the end of a message body? How about HTTP? Can HTTP use the same method as SMTP to mark the end of a message body? Explain your answer.

Answer:

SMTP uses a line containing only a period to mark the end of a message body.

HTTP uses “Content-Length header field” to indicate the length of a message body.

No, HTTP cannot use the method used by SMTP, because HTTP message could be binary data, whereas in SMTP, the message body must be in 7-bit ASCII format.

Grading:
-3 If fail to show the mechanism of SMTP.
-3 If fail to show the mechanism of HTTP.
-1 If a clear answer (yes/no) is not given.
-2 If fail to give a correct answer to the third question.
-2 If solid explanation is not given for the third question.

3. (10 pt) What is the Apache Web Server? How much does it cost? What functionality does it currently have? You may want to look at Wikipedia to answer these questions.

Answer: (From Wikipedia)

The Apache HTTP Server, colloquially called Apache, is the world's most used web server software. Originally based on the NCSA HTTPd server, development of Apache began in early 1995 after work on the NCSA code stalled. Apache played a key role in the initial growth of the World Wide Web, quickly overtaking NCSA HTTPd as the dominant HTTP server, and has remained most popular since April 1996. In 2009, it became the first web server software to serve more than 100 million websites.

Apache is an open source software available for free.

The Apache server offers a number of services that clients might make use of. These services are offered using various protocols through different ports, and include: hypertext transfer protocol (HTTP), typically through port 80, simple mail transfer protocol (SMTP), typically through port 25, domain name service (DNS) for mapping domain names to their corresponding IP addresses, generally through port 53, and file transfer protocol (FTP) for uploading and downloading files, usually through port 21.

Grading:
-1~3 If the answer does not show the student's understanding of the concept of Apache Web Server.
-2 If the cost of the Apache Web Server is not mentioned.
Partial points will be deducted if important functions of Apache Web Server are missing.
4. (10 pt.) What is the function of a DNS server? When you click on an URL, your local DNS does not have the information to complete the job of DNS server. What is the possible process that your local DNS has to do?

**Answer: (From Wikipedia)**

The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or other resources connected to the Internet or a private network. It associates various information with domain names assigned to each of the participating entities. Most prominently, it translates more readily memorized domain names to the numerical IP addresses needed for locating and identifying computer services and devices with the underlying network protocols.

Assuming the resolver has no cached records to accelerate the process, the resolution process starts with a query to one of the root servers. In typical operation, the root servers do not answer directly, but respond with a referral to more authoritative servers, e.g., a query for "www.wikipedia.org" is referred to the org servers. The resolver now queries the servers referred to, and iteratively repeat this process until it receives an authoritative answer. The diagram illustrates this process for the host [www.wikipedia.org](http://www.wikipedia.org).

This mechanism would place a large traffic burden on the root servers, if every resolution on the Internet would require starting at the root. In practice caching is used in DNS servers to off-load the root servers, and as a result, root name servers actually are involved in only a fraction of all requests.

**Grading:**
-1 ~ -4 If main functions of a DNS are missing in the answer.
-1 ~ -4 If the process is not clearly explained. (Key words: local DNS server, top-level DNS server, root DNS server)
-1 If related terminology is not mentioned.

5. (10 pt) Comparing HTTP 1.0 with HTTP 1.1, what are the major differences between the two? Which one will perform better from a client point of view? Do you see any condition that HTTP 1.0 will perform better from a server perspective?

**Answer: (Stack Overflow: [http://stackoverflow.com/questions/246859/http-1-0-vs-1-1](http://stackoverflow.com/questions/246859/http-1-0-vs-1-1))**

**Proxy support and the Host field:**

HTTP 1.1 has a required Host header by spec.

HTTP 1.0 does not officially require a Host header, but it doesn't hurt to add one, and many applications (proxies) expect to see the Host header regardless of the protocol version.

**Example:**

GET / HTTP/1.1

Host: www.blahblahblahblah.com
This header is useful because it allows you to route a message through proxy servers, and also because your web server can distinguish between different sites on the same server.

So this means if you have blahblahbah.com and helohelohelo.com both pointing to the same IP. Your web server can use the Host field to distinguish which site the client machine wants.

**Persistent connections:**

HTTP 1.1 also allows you to have persistent connections which means that you can have more than one request/response on the same HTTP connection.

In HTTP 1.0 you had to open a new connection for each request/response pair. And after each response the connection would be closed. This lead to some big efficiency problems because of TCP Slow Start.

**OPTIONS method:**

HTTP/1.1 introduces the OPTIONS method. An HTTP client can use this method to determine the abilities of the HTTP server. It's mostly used for Cross Origin Resource Sharing in web applications.

**Caching:**

HTTP 1.0 had support for caching via the header: If-Modified-Since.

HTTP 1.1 expands on the caching support a lot by using something called 'entity tag'. If 2 resources are the same, then they will have the same entity tags.

HTTP 1.1 also adds the If-Unmodified-Since, If-Match, If-None-Match conditional headers.

There are also further additions relating to caching like the Cache-Control header.

**100 Continue status:**

There is a new return code in HTTP/1.1 100 Continue. This is to prevent a client from sending a large request when that client is not even sure if the server can process the request, or is authorized to process the request. In this case the client sends only the headers, and the server will tell the client 100 Continue, go ahead with the body.

HTTP 1.1 will perform better from a client point of view. Because HTTP 1.1 has PersistentConnection and Pipelining. Client can also get request resource from server more efficiently.

There might be some scenarios that when server does not have enough resource, HTTP 1.1 will cause extra load for the server hence decrease the performance of the server.

**Grading:**

-1 ~ -4 If major differences are missing in the answer.
-2 If the answer is wrong in choosing the one with better performance.
-2 If the reason behind the choice is not clarified.
-1 ~ -2 If the answer to the last question is not justified. This is an open question; no points will be deducted if you have clear explanation.