cs 664 computer networks

Homework 2

Name: Morgan Bergen

WSU ID: B493R546

Date: Tuesday, June 18, 2024

Due: Friday, June 21 2023

1a. What is HTTP, and what is its primary purpose in web communication? Compare and contrast the key features and improvements of HTTP/1, HTTP/2, and HTTP/3. Discuss how each protocol addresses performance, efficiency, security, and other aspects of web communication.

The hyper text transfer protocol (HTTP) is an application layer protocol which is implemented in two programs: a client program and a server program which execute on different end systems, and communciate to each other by exchanging HTTP messages. The primary purpose of HTTP is to transfer resources between a client and a server. HTTP/1.1 is a text based protocol that is stateless and connectionless. HTTP/2 is a binary protocol that is stateful and connection oriented. HTTP/3 is a binary protocol that is stateful and connection oriented. HTTP/2 and HTTP/3 are improvements over HTTP/1.1 in that they are more efficient, secure, and performant. HTTP/2 and HTTP/3 use multiplexing to allow multiple requests and responses to be sent over a single connection. HTTP/2 and HTTP/3 use header compression to reduce the size of the headers. HTTP/2 and HTTP/3 use server push to send resources to the client before the client requests them. HTTP/3 uses QUIC to provide security and performance improvements over HTTP/2.

HTTP/1.1

- A text based protocol that supports `GET`, `POST`, `PUT`, `DELETE`, and `HEAD` methods that are used to request and send resources between a client and a server in plain text. For example, a HTTP request message would look like this:

```
GET /index.html HTTP/1.1\r\n
Host:  www.example.com\r\n
User-Agent:  Mozilla/5.0 (Macintosh; Intel Mac OS X)
    10.15; rv: 80.0) Gecko/20100101 Firefox/80.0\r\n
Accept: text/html, application/xhtml+xml\r\n
Accept-Language: en-us, en; q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
\r\n
```

- Each and every request opens a new connection which can cause latency and overhead.
- When a client and server communciate a series of requests may be made back to back a periodically regular intervals, this is where the two persistent and non-persistent connections features

differentiate from each other in their behavior of how request objects are transmitted.
- Chunked transfer encoding is used to send resources in chunks, which allows data to be sent in a series of blocks, which allows the server to start transmitting content before knowing the total size of the content.

HTTP/2

- Binary based protocol making parsing more efficient and less prone to errors.
- Multiplexing allows for multiple requests and responses to be simultaeously sent over a single connection.
- Header compression helps reduce the often large and repetitive HTTP headers, this is done by using a header table that stores the most common headers and their values, and then only sending the index of the header in the table.
- Provides load control over streams and a tcp connection, this helps in managing resources by avoiding che response time in half.ongestion and overloading the network.
- Improved efficiency with the use of network resoureces with better load times and the server push feature.

HTTP/3

- Built on top of the udp protocol rather than the tcp protocol, allowing for faster connections and improved performance.
- Reduced latency by combining connection establishment with a cryptographic handshake allowing data to be transmitted with fewer round trips.
- Multiplexing without head of line blocking. Head of line blocking is when a packet is lost or delayed, all packets behind it are also delayed. Multiplexing without head of line blocking allows for packets to be sent and received out of order.
- Encrypted connection by default enchange security and privacy.
- Stream prioritization and dependency which allow clients to prioritize certain streams over others.

1b. Explain the concept of a persistent HTTP connection and its advantages? How does it differ from a non-persistent HTTP connection? Provide examples of scenarios where each type of connection may be beneficial.

Explanation of persistent HTTP connection

A persistent HTTP connection is where a tcp connection opens to a server which allows for the client and server to send and receive multiple objects over a single tcp connection, then when transmission ends the tcp connection closes. The advantage of a persistent HTTP connection is that it has as little as one RTT (round trip time) for all referenced objects; this cuts the response time in half. Subsequent requests and responses between the same client and server can be sent over the same connection. There are three features in persistent HTTP which is pipelining, multiplexing, and configurable timeout intervals. Pipelining allows for requests for objects to be sent back to back without waiting for replies to pending requests. Multiplexing allows for multiple requests and responses to be sent over a single connection. And configurable timeout intervals is where the HTTP server closes a connection when it isn't in use for a certain amount of time, closing these connections helps free up resources, allowing for the server to handle more active connections and handle requests more efficiently.

Explanation of non-persistent HTTP connection

A non-persistent HTTP connection is where a tcp connection opens to a server, sends a single object, then closes the tcp connection. The advantage of a non-persistent HTTP connection is that it is simple and easy to implement.

Example Scenarios

For a website that has a single object like a small image, a non-persistent HTTP connection would be beneficial because the object can be sent and received quickly. For a single object a non-persistent connection can be simpler and more efficient than a persistent connection.

For multimedia streaming that require continuous data transfer can benefit from persistent connections to maintain a smooth and stable stream. Persistent connections can be used to send a receive multiple objects over a single connection, which can be beneficial for streaming video and audio.

1c. How does HTTP maintain statelessness in its communication? What are the advantages and disadvantages of statelessness in HTTP?

In statelessness if a client asks for the same object twice in a period of a few seconds, the server does not respond by saying that it just served the object to the client, because the server sends requested files to clients without storing any state information about the client the server is stateless. So HTTP is a stateless protocol because the server does not maintain any information about the client. The way that statelessness is maintained is through self contained requests and no built in session tracking. So each HTTP request does not remember any previous interactions of a request that contains details such as method, url, headers, and body. HTTP does not provide a way to maintain session information across multiple requests, session information must be implemented through cookies and url parameters. The advantage to statelessness is that it is a simple design, it can scale, and it is easy to implement, and it is not complex. The disadvantage to statelessness is that it can be inefficient because there is increased overhead due to redundant information and repeated requests.

1d. Explain the request-response model in HTTP. What are the key components of an HTTP request and an HTTP response.

There are two types of HTTP messages request and response. The request messages could include methods such as `POST`, `HEAD`, `GET`, `PUT`, and `DELETE`. The response methods could include status codes that appear in the 1st line in the server-to-client response message. The client sends a request in an web browser for example to initate communication by sending any number of methods for example a `GET` method HTTP request to the server. The server then processes that request and prepares a response message to send back to the client. The server sends the response and the client receives a response message and renderes the response for example a web page and display that web page in a web browser.

The key components of an HTTP request are the request line which contain the `GET`, `POST`, `PUT`, `DELETE`, or `HEAD` methods, the url of the resource, and the version of the HTTP protocol, the header lines which contain the header field name, the header field value, which are the `User-Agent` which contains the browser and operating system information, `Accept` which contains the media types that the client can accept, `Accept-Language` which contains the language that the client can accept, and `Accept-Encoding` which specifies the content that the encoding client can handle, and the entity body which contains the data that is being sent to the server.

Here is an example of a HTTP request message

```
GET /index.html HTTP/1.1\r\n
Host:  www.example.com\r\n
User-Agent:  Mozilla/5.0 (Macintosh; Intel Mac OS X)
    10.15; rv: 80.0) Gecko/20100101 Firefox/80.0\r\n
Accept: text/html, application/xhtml+xml\r\n
Accept-Language: en-us, en; q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Connection:  keep-alive\r\n
\r\n
```

The key components of a HTTP response messages is comprised of three sections: an initial status line, six header lines, and then the entity body. The initial statue lines contain the HTTP version, the status code which is a three digit code indicating the result of the request, and the corresponding status message. Status code examples are

- `200 OK` - request succeeded, requested object later in this message
- `301 Moved Permanently` - requested object moved to a new location specified later in the message in the location field.
- `400 Bad Request` - the request message is not understood by the server.
- `404 Not Found` - the requested document is not found on the server
- `505 HTTP Version Not Supported` - the server does not support the HTTP protocol version used in the request message.

The header lines contain the status line which indicates the HTTP version, the status code, and the status message. The header lines which contain the `Connection: close` header line to tell the client that it is going to close the tcp connection after sending the message. The `Date:` header line indicates the time and date when the HTTP response was created and sent to the server. The `Server:` header line which indicates that the message was generated by an apache web server and is analogous to the `User-agent:` header line in a header request message. The `Last-Modified:` header line which indicates the time and date when the requested object was last modified. The `ETag` header lines which contains the unique identifier for the requested object. The `Accept-Ranges:` header line which indicates that the server supports byte range requests. The `Content-Length:` header line which indicates the number of bytes in the object being sent. The `Content-Type:` header line which indicates the type of object being sent. The entity body contains the data that is being sent to the client.

Here is an example of a HTTP response message

```
HTTP/1.1 200 OK
Date:  Tue, 08 Sep 2020 00:53:20 GMT
Server:  Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
    PHP/7.4.9 mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT
ETag:  "1d3-530a3a8f2f7c0"
Accpted-Ranges:  bytes
Content-Length: 2651
Content-Type:  text/html; charset=UTF-8
\r\n
<!DOCTYPE html>
```

```
<html>
...
```

2a. What is a cookie in the context of web technology? How is it different from other methods of storing user data on the client side?

A cookie addresses the problems that a http server is stateless, and it is often important for a web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user's identity. Cookies allow sites to keep track users. Cookies are used for authorization, shopping carts, recommendations, user session state, general session management, other tracking, and personalization.

Cookies contain 4 components:

1. A cookie header line in the http response message
2. A cookie header line in the http request message
3. A cookie ile kept on the user's end system and managed by the user's browser
4. A back-end database at the web site

A user accesses the web using safari from their home computer and contacts Amazon.com for the first time. She has already visited the eBay site and when the request comes to the amazon web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number. The amazon web server then responds to the user's browser, including it in the HTTP response a `Set-cookie:` header, which contains the identification number, for example `Set-cookie: 1678`. When the browser receives the HTTP response message, it sees the `Set-cookie:` header. The browser then appends a line to the special cookie file that it manages, this includes the hostname of the server and the identification number in the header. Every time the user requests a web page, her browser consults her cookie file, extracts her identification number for the site, and puts a cookie header line that includes the identification number in the HTTP request. Amazon's web server looks up the identification number in its back-end database and knows who the user is so then it can server her personalized content.

Cookies are different from other methods of storing user data on the client side because cookies are stored on the client side and are managed by the user's browser. Cookies are stored in a special cookie file, stored as key value pairs, with a storage size limited to about 4 kb, they are persistent, accessible, and can be used to track users. Cookies are sent with every HTTP request and response, and it's use case differs from other methods for the purpose of session management, tracking, and personalization.

2b. What is the difference between first-party cookies and third-party cookies? Provide examples of when each type is used and potential privacy concerns associated with third-party cookies.

First-party cookies are used to track user behavior on one given website and are set by the website that the user is visiting. Third-party cookies track user behavior across multiple websites without the user ever choosing to visit the tracker site. Usage examples of first-party cookies are login sessions, when you log into a website like your email, first party cookies will remember your login status so you don't have to re-enter your credentials on each page. Another example is shopping carts, when you add items to your shopping cart on a site like Amazon, first party cookies will remember the items in your cart as you navigate the site. Usage examples of third-party cookies are advertising, when you visit a website that has ads from

a third-party ad network, the ad network will use third-party cookies to track your behavior across multiple sites and serve you targeted ads. Another example is social media, when you visit a website that has social media buttons from a third party social media site, the social media site will use third-party cookies to keep track of your behavior across multiple sites. The potential privacy concerns is that the tracking may be invisible to the user, it could be an invisible link rather than displaying ad triggering the HTTP `GET` to tack.

2c. What is web caching, and why is it essential in the context of web perforamce and user experience?

A web cache is also called a proxy server, this is a network entity that satisfies HTTP requests on behalf of an orgin web serber. The web cache has its own disk storage and keeps copies of recently requested objects in this storage. When a client requests an object, the cache checks to see if it has a copy of the object. If it does, the cache returns the object to the client; if it does not have a copy of the object, the cache requests the object from the origin server, then returns the object to the client and stores a copy of the object in its disk storage. A web cache can substantially reduce the response time for a client request, especially if the bottleneck bandwidth between the client and the origin server is less than the bandwidth between the client and the cache. The web cache can also substantially reduce traffic on an institution's access link to the internet. And web caching can also reduce the load on the origin server.

2d. Explain the relationship between browser caching and conditional `GET` requests in web development. How do conditional `GET` requests improve caching efficiency?

Conditional `GET` requests are a type of HTTP request where the browser asks the server for a resource only if it has been modified since the last time the browser cached it. This is done through `If-Modified-Since` header line which is included with the date and time it cached the resource and the `If-None-Match` where the browser sends this header with an entity tage, a unique identifier assigned to the reosurce by the server. The server compares this ETag with the current Etag of the resource. A browser caching stores resources locally to reduce load times. Conitional `GET` requests validate the cached resources by checking with the server whether they are still up to date.

A conditional `GET` requests improve caching efficiency by reducing the amount of data that needs to be transferred between the client and the server. The server can respond with a `304 Not Modified` status code if the resource has not been modified since the last time the browser cached it. This allows the browser to use the cache resource instead of downloading the resource again. This reduces the amount of data that needs to be transferred between the client and the server, which improves the performance of the web application.

3a. What is DNS, and what is its primary function in computer networking and the internet?

The domain name system DNS translates hostnames to IP addresses. The DNS is a distributed database implemented in a hierarchy of DNS servers, and an application-layer protocol that allows hosts to query the distributed database. The DNS servers are implemented in a distributed hierarchical database. DNS provides other important services in addition to translating host names to IP adresses such as host aliasing, where a host with a complicated sometimes said to be a canonical hostname can have one or more alias names. DNS can also provide mail server aliasing, where a mail server can have multiple alias names, and load distribution, where multiple IP addresses are returned in response to a single hostname query. DNS also provides load distribution among replicated servers, which as replicated web servers. DNS also provides a means of distributing the database over multiple DNS servers.

3b. Explain the hierarchical structure of the DNS naming system, including the roles of top-level domains (TLDs), second-level domains, and sub domains.

The DNS naming system is a hierarchical structure that is organized into a tree structure. The root of the tree is the root domain, which is represented by a single dot. Below the root domain are the top-level domains (TLDs), which are the highest level of the DNS hierarchy. The TLDs are divided into two categories: generic TLDs (gTLDs) and country code TLDs (ccTLDs). The gTLDs are generic TLDs that are not associated with a specific country, such as `.com`, `.org`, and `.net`. The ccTLDs are country code TLDs that are associated with a specific country, such as `.us`, `.uk`, and `.ca`. Below the TLDs are the second-level domains which are the authoriative DNS servers, which are the next level of the DNS hierarchy. The second-level domains are the domain names that are registered by organizations or individuals, such as `example.com`, `example.org`, and `example.net`. Below the second-level domains are the subdomains, which are additional levels of the DNS hierarchy. The subdomains are used to create additional levels of organization within a domain, such as `www.example.com`, `mail.example.com`, and `blog.example.com`.

3c. What are authoritative DNS servers, and how do they differ from recursive DNS servers? How do authoritative DNS servers help resolve domain names?

Authoritative DNS servers provide publicly accessible DNS records that map the names of hosts to IP addresses. An organization's authoritative DNS server houses these DNS records. An organization can choose to implement its own authoritative DNS server to hold these records; alternatively, the organization can pay to have these records stored in an authoritative DNS server of some service provider. Most universities and large companies implement and maintain their own primary and secondary authoritative DNS server.

Recursive DNS servers are DNS servers that are used to resolve domain names on behalf of clients. When a client wants to resolve a domain name, it sends a query to a recursive DNS server. The recursive DNS server then sends queries to authoritative DNS servers to resolve the domain name. The recursive DNS server caches the results of these queries to improve performance and reduce the load on the authoritative DNS servers. The recursive DNS server then sends the resolved IP address back to the client. The recursive DNS server is responsible for resolving domain names on behalf of clients.

3d. What is a TTL (Time-to-Live) values in DNS records, and why is it important? How does it affect caching and DNS propagation?

The TTL (Time-ro-Level) value in DNS records (distributed database storing resource records) is a value that specifies the amount of time that a DNS record can be cached by a DNS resolver. In the RR format: there is (`name`, `value`, `type`, `ttl`) where the `ttl` is the time to live. The TTL value is set by the authoritative DNS server when it sends the DNS record to the recursive DNS server. The recursive DNS server then caches the DNS record for the amount of time specified by the TTL value. Once any name server learns the mapping of a domain name to an IP address, it stores the mapping in its cache. Then it returns a cached mapping in response to a query. TTL affects caching and DNS propagation because it determines how long a DNS record can be cached by a DNS resolver. Caching improves response time and reduces the load on authoritative DNS servers. DNS propagation is the process of updating DNS records on the internet, TTL values will determine when a DNS record needs updating. Cache entries time out after the TTL value expires and TLD servers will cached the DNS records for the TTL value in their local name server.

If a named host changes their IP address it may not be known to all DNS servers until the TTL value expires and goes out of date.

4a. What are two technical challenges associated with streaming video that we discussed in class and could you please explain them briefly?

The technical challenges associated with streaming video is scalability which addresses how we can deliver and reach 1 billion users, and heterogeneity which addresses how we can deliver video to a wide range of devices. In streaming stored video a server-to-client bandwidth will vary over time, with changing network congestion levels, and with changing numbers of users. Additionally the technical challenge continuous playout constraint during client video playout, playout wiming must match the original timing, however therer can be network delays that are variable so the client-sdie buffer must match continuous playout constraint. Other challenges include client interactivity and having to accomidate pause, fast-forward, rewind, and jump through video. Additionally other challenges include video packet oss and retransmission.

4b. What is the role of a buffer in streaming video playback? How does it contribute to a smooth and uninterrupted viewing experience?

Playout buffering on the client side plus playout delay compensate for network-added delay and delay jitter. In HTTP streaming a videos is stored on a server as a file with a url. When the user wants to see the video, the client establishes a TCP connection with the server and issues a HTTP `GET` request for the url containing the video file. The server sends the video file within a HTTP response message and on the client side the bytes are collected in a client application buffer. Once the number of bytes in this buffer exceed a predetermined threshold, the client application begins playback specifically the streaming video appliation periodically grabes video frames from the client application beffer, decompresses the frames, and displays them on the users screen. The buffer is used to store video frames that have been received from the server but not yet displayed them to the user. The buffer allows the client to store video frames in advance so that the video can be played back smoothly and without interruption. The buffer helps to compensate for network delay and delay jitter by storing video frames within the client application buffer as well.

4c. What is Dynamic Adaptive Streaming over HTTP (DASH), and how does it work to deliver video content to viewers? How does it differ from traditional streaming methods?

In DASH a video is encoded into several different versions, with each version having different bit rate and different quality levels. The client dynamically requests chunks of video segments of a few seconds in length. When the amount of available bandwidth is hight, the client naturally selects chunks from a high-rate version; and when the available bandwidth is low, it naturally selects from a low-rate version. The client selects different chunks one at a time with HTTP `GET` request messages. DASH allows clients with different internet access rates to stream in video at different encoding rates. DASH is different from traditional streaming methods because it allows clients to dynamically request video segments at different bit rates and quality levels. DASH allows clients to adapt to the available bandwidth if the available end-to-end bandwidth changes during the session.

4d. Content distribution networks (CDNs) typically adpot one of two different server placement philosophies. Name and briefly describe them.

The two different server placement philosophies are enter deep or bring home. The enter deep philosophy involves deploying server clusters deep within a network of internet service providers ISPs by positioning server clusters in numerous locations close to end users, this method aims to minimize the number of hops

and links that data packets must traverse, thereby reducing user-perceived delay and improving throughput. The bring home philosophy involves placing server clusters at fewer, centralized locations typically at internet exchange points. So instead of spreading out servers within individual ISPs, CDNs build large clusters in strategic locations, which can simplify maintenance and management at the cost of potentially higher delay and lower throughput for end users.