CS 4453 Computer Networks

Chapter 3 Application Layer

2015 Winter

Network applications are the driving forces for the explosive development of the internet.

Internet applications include the classic text-based applications such as text email, file transfers, newsgroups etc.

Then applications as World Wide Web, instant messaging and P2P file sharing, voice-over-IP (VoIP), video conference (Skype), video distribution (YouTube), movies on demand (Netflix), on-line games (World of Warcraft), social networks (Facebook, Twitter), etc. New application are keeping invented.

3.1 Basics of network applications

Software of network applications usually need to run on the different end systems and communicate each other over the network.

Since the each lower layer of the network protocols has implemented in different network devices, usually the application software, no matter written in C, Java or Python, needs not include the parts run on network core devices, such as routers or switches.

Basically, we just need to know the interface of the lower layer, or the API of the lower layer (for example, socket).

Network application architectures

Roughly, there are two architectural paradigms used in modern network applications.

- Client-server architecture
- P2P architecture

In Client-server architecture, there are a server and many clients distributed over the network. The server is always on while a client can be randomly run. The server is listening on the network and a client initializes the communication. Upon the requests from a client, the server provide certain services to the client. Usually, there is no communication between two clients.

Original client-server architecture includes a single-server host. But now, in many cases, a single-server host is unable to keep up with all the requests from large number of clients. For this reason, a data center, housing a large number of hosts, is often used to create a powerful virtual server. A date center can have hundreds of thousands of servers, that must be powered and maintained.

In P2P architecture, the application exploits direct communication between pairs of intermittently connected hosts, called peers. The peers are not owned by the service provider, but are instead desktops and laptops controlled by users. Many of today's most popular and traffic-intensive applications are based on P2P architecture.

One of the most compelling feature of P2P architectures is their self-scalability. For example, in a P2P file sharing system, each peer generates workload by requesting files, but each peer also adds service capacity to the system by distributing files to other peers.

There are also some challenges for the P2P applications.

- ISP Friendly: most residential ISPs (DSL, cable ISPs etc) have been designed for "asymmetrical" bandwidth usage, i.e., for much more downstream than upstream traffic. But many P2P applications shift upstream traffic from servers to residential ISPs, which significant stress on the ISPs.
- Security: Since the highly distribution and openness, P2P applications can be a challenge to security.
- Incentive: the success of P2P applications depend on convincing users to volunteer bandwidth, storage and computation resources to the applications.

There are also some applications using hybrid architectures, that combines both client-server and P2P elements. For example, servers are used to track the IP addresses of users, but user-to-user messages are sent directly between user hosts without passing through servers.

Processes communicating

When a program is running, it results processes. In the case of network application, processes will run on different hosts and include processes communication over the network.

A network application consists of pairs of processes that send messages to each other. Usually, we will label one of the two processes as the client and the other server. In the case of P2P, a process can both upload and download files.

Usually, a process that initiates the communication is labeled as the client and the process that waits to be contacted to begin the session is labeled as the server.

Some important processes for the network applications:

- The interface between the process and the network:

  In network application, any message sent from one process to another must go through the underlying network. Most of these processes send and receive messages through a software interface of the underlying network called socket. Socket is an Application Programming Interface (API) between the application layer and the transport layer within a host. A few things the application developer can control on the transport layer is choice of transport protocol and fix a few transport-layer parameters such as maximum buffer and maximum segment sizes.

- Addressing processes:

  In the Internet, the host is identified by its IP address. The message must include the IP address of the sending process and the IP address of the destination process.

  In addition to the addresses, the sending process also needs to identify the receiving process (the receiving socket), because a host may run several sockets at the same time. A destination port number serves this purpose.

Transport services for applications

Networks usually provide more than one transport-layer protocols for different applications. An application developer should choose certain protocol according to the type of applications. Different protocols may provide different services.

- **Reliability of data transfer**

  In some applications, data need to be guaranteed that it sent by one end of the application is delivered correctly and completely to the other end of the application. Some of the transport protocol provides to an application such a guaranteed data delivery service, that is said to provide reliable data transfer. Some protocol does not provide reliable transfer. Some of the data sent by the sending process may never arrive at the receiving process.

- **Throughput**

  The throughput for the communications between two processes over the network means the rate at which the sending process can deliver bits to the receiving process. Since other hosts are using the network, the throughput can fluctuate with time.

  Some applications, called bandwidth-sensitive applications, need a guaranteed throughput. Some transport protocol provides the service of guaranteed available throughput at some specified rate. With this service, the application could request a guaranteed throughput of r bits/sec, and the transport protocol would then ensure that the available throughput is always at least r bits/sec.

- **Timing**

  A transport-layer protocol can also provide timing guarantees. For example, it guarantees that every bit that sender pumps into the socket arrives at the receiver's socket no more than 100 msec later.

- **Security**

  A transport layer protocol can provide an application with one or more security services.

Today's Internet can often provide satisfactory services to time-sensitive applications, but it cannot provide any timing or throughput guarantees.

3.2 HTTP and the web

In 1989, CERN (the European Center for Nuclear Research) initialized the main idea of World Wide Web. In 1994, CERN and MIT signed an agreement setting up the W3C (World Wide Web Consortium) to further developing the Web.

Web has many advantages for a lot of applications. It operates on demand so that the users receive what they want when they want it.

Search engine and hyperlinks, graphics and multimedia, web based emails, social networks, web telephony, $\cdots$, applications keep emerging.

Many tools are used for the web applications, that let the developers use web as a platform many important applications.

Overview of HTTP:

The HyperText Transfer Protocol (HTTP) the Web's application-layer protocol, is at the heart of the web. HTTP is defined in RFCs 1945 and 2616, which is implemented in two programs: a client program and a server program. HTTP defines the structure of exchanging messages and how the client and server exchange the messages.

A web page consists of objects. An object is a file, such as an HTML (HyperText Markup Language) file, a JPEG image, a Java applet, a video chip etc, that is addressable by a single URL (Uniform Resource Locator).

Most web pages consist of a base HTML file and several referenced objects. The base HTML file references the other objects in the page with the object's URLs. Each URL has two components: the hostname of the server that houses the object and the object's path name.

The web browsers (such as Internet Explorer and Google Chrome) implement the client side of HTTP and the web servers (such as Apache and Microsoft Internet Information Server) implement the server side of HTTP.

When a user requests a web page, the browser sends HTTP request messages for the objects in the page to the server. The server receives the requests and responds with HTTP response messages that contain the objects.

HTTP uses TCP as its underlying transport protocol. The HTTP client first initiates a TCP connection with the server. Once the connection is established, the browser and the server processes access TCP through their socket interfaces.

HTTP is a stateless protocol, in which the server sends requested files to clients without storing any state information about the client. The web uses the client-server architecture. The web server is always on with a fixed IP address.

HTTP connections

In many internet applications, the client and server communicate for an extended period of time. When this client-server interaction takes place over TCP, a decision should be made: each request/response pair be sent over a separate TCP connection or the sequence of request/response pairs be sent over the same TCP connection. These different connections are called non-persistent connections or persistent connections. HTTP uses persistent connections in its default mode, but HTTP client and server can be configured to use non-persistent connections as well.

- HTTP with non-persistent connections

  Suppose a web page (server) consists a base HTML file and 10 JPEG images. Under the non-persistent connection, the client first requests the web by creating a TCP connection to the server. The server then sends the base HTML file to the client over the TCP connection and then closes that TCP connection. When the client needs the JPEG image, a new TCP connection will be create. With non-persistent connection, each TCP connection only be used to send one object.

- HTTP with persistent connections

  The server leaves the TCP connection open after sending a response. Subsequent requests and responses between the same client and server can be sent over the same connection. In particular, an entire web page (base HTML base file and other objects) can be sent over a single persistent TCP connection. Moreover, multiple web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection. Usually, the HTTP server closes a connection when it isn't used for a certain time.

- Non-persistent connections have some shortcomings. To establish a new TCP connection, some simple packet exchange must perform. So create more TCP connections consumers more communication resources.
- Modern browsers can be configured to obtain parallel TCP connections. In their default modes, most browsers open 5 to 10 parallel TCP connections.

HTTP message format

HTTP specification include the definitions of the HTTP message formats. There are two formats.

- HTTP request message.
- HTTP response message

An example of HTTP request message:

```
GET /cs4453/sample.html HTTP/1.1
Host: ccc.cs.lakeheadu.ca
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

- The message is written in ASCII text.
- The first line of the HTTP request message is called the request line.
- The subsequent lines are called the header lines.
- A request line has three fields: the method, the URL and the version. A request message can contain several header lines. A carriage return and a line feed are required to indicate the end of the message.

The meaning of the header lines are as follows.

The `Connection:close` means requesting a non-persistent connection.

`User-agent:Mozilla/5.0` means the browser used is the Firefox.

`Accept-language:fr` means French is the preferred language. If there is no French Web, then the default page will be sent. `Accept-language` is one of many content negotiation headers available in HTTP.

HTTP defined several methods:

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTION	Query options for a page

An example of HTTP response message.

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 13 Jan 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified:  Tue, 13 Jan 2015 15:11:04 GMT
Content-Length: 6821
Context-Type: text/html
... ... ...
... ... ...
```

The response message contains three sections:

- an initial status line
- several header lines
- entity body (the details of this section is missed in this example).

The status line has three fields: the protocol version, a status code and a corresponding status message.

Many status codes are defined in HTTP including:

- `1xx` Informational
- `2xx` Success
- `3xx` Redirection
- `4xx` Client error
- `5xx` Server error

Some of these codes only supported by HTTP 1.1, but not HTTP 1.0. Microsoft, Nginx, and others also gave some extensions of these codes.

Some examples of the status codes:

- `200 OK`: Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource.
- `301 Moved Permanently`: This and all future requests should be directed to the given URI.
- `400 Bad Request`: The server cannot or will not process the request due to something that is perceived to be a client error.
- `403 Forbidden`: The request was a valid request, but the server is refusing to respond to it.
- `404 Not Found`: The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.

- `500 Internal Server Error` A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.
- `503 Service Unavailable`: The server is currently unavailable (because it is overloaded or down for maintenance). Generally, this is a temporary state.
- `505 HTTP Version Not Supported`: The server does not support the HTTP protocol version used in the request.

Caching and GET extension

A Web cache is a network entity that may used to response HTTP requests on the behalf of an origin Web server. The Web cache has its own disk storage and keeps copies of recently requested objects in this storage.

The user's HTTP requests are first directed to the web cache. If the cache has the object requested, it returns the object within an HTTP response message to the client browser. If the cache does not have that object, then it connects to the original server and ask for the object. After it obtained the object, it then sends the object to the client.

Although caching may reduce the response time, it could send a old version of the object to the client.

- The semantics of the GET method change to a "conditional GET" if the request message includes an If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field.
- A conditional GET method requests that the entity be transferred only under the circumstances described by the conditional header field(s).
- The conditional GET method is intended to reduce unnecessary network usage by allowing cached entities to be refreshed without requiring multiple requests (the cache need not to check if the object is updated by the server for every request) or transferring data already held by the client.

The following is an example of using conditional GET:

```
GET /fruit/kiwi.fig HTTP1.1
Host: www.exoriguecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

The response is:

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
```

Another extension of GET is “partial GET” method.

The semantics of the GET method change to a “partial GET” if the request message includes a Range header field. A partial GET requests that only part of the entity be transferred.

The partial GET method is intended to reduce unnecessary network usage by allowing partially-retrieved entities to be completed without transferring data already held by the client.

Caching can also be used in client side. A browser can reduce download times significantly by saving a copy of each image in a cache on the user's disk and using the cached copy next time for visiting the same web page. The HEAD method can be used to see if the object is updated and a new download should be performed.

Cookies

HTTP servers are designed as stateless so that a Web server can handle thousands of simultaneous TCP connections. However, in many cases a Web site needs to identify users. For this purpose, HTTP uses cookies (RFC 6265), that allow sites to keep track of users.

When a user first time visit a site, the user may provide a user identification or some information. The web site can put a `Set-Cookie` header line in the response message. This line will provide a unique cookie value to the user. The user then store the cookie in the computer and add the `Cookie` line in the subsequent requests using the cookie provided by the server.

An simple example:

First the browser sends a request:

```
GET /index.html HTTP/1.1
Host: www.example.ca
```

The server response:

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: name=<value>
Set-Cookie: name2=<value2>; Expires=Wed, 09 Jun 2021 10:18:14 GMT

(content of page)
```

The browser continues the requests:

```
GET /spec.html HTTP/1.1
Host: www.example.ca
Cookie: name=<value>; name2=<value2>
Accept: */*
```

Cookie specifications suggest that browsers should be able to save and send back a minimal number of cookies.

In particular, a web browser is expected to be able to store at least 3000 cookies of four kilobytes each, and at least 50 cookies per server or domain.

The value of a cookie can be modified by the server by sending a new `Set-Cookie` line in response of a page request. The browser then replaces the old value with the new one.

The server will store the cookies associating with the users information (user identity, address, email, credit card number etc) to its database.

The next time when the user visit the web site, the web server will get the information from the database so that the user needs not provide again.

Cookies may have different types. A session cookie, also referred as in memory cookie. When a expiry date of validity interval is not set at the `Set-Cookie` line, a session cookie is created.

When a expiry is set, the cookie is called persistent cookie.

Some secure cookie is used for security purpose for the https connection.

Dynamic Web pages

Many Web pages are used for different applications and services. The applications run inside the browser, with user data stored on servers in Internet data centers.

The advantage of this approach is that users do not need to install separate application programs, and user data can be accessed from different computers and back up by the service operator.

For these Web applications, dynamic content of web is needed. The dynamic content can be generated by programs running on the server or in the browser.

Most of these requests and responds for dynamic contents happen in the background and the user may not know because the page URL and title typically do not change.

By including client side programs, the page can present a more responsive interface than with server side program along.

Some of the most popular methods for server side are as follows.

- CGI (Common Gateway Interface) (RFC 3875): CGI provides an interface to allow web servers to talk to back-end programs and scripts that can accept input and generate HTML pages in response. These programs can be implemented in different languages such as Python, Peal, Ruby and so on. Usually, the programs invoked via CGI live in a directory called `cgi-bin`.
- PHP (Hypertext preprocessor): In this approach, little scripts are embedded inside HTML pages and have them be executed by the server to generate the page. PHP is also a powerful programming language for interfacing the Web and a server database. Usually, servers identify Web pages containing PHP from the file extension `.php` rather than `html`.

- JSP (JavaServer Pages): It is similar to PHP, but the dynamic part is written in the Java instead of in PHP. Pages using this technique have the file extension `.jsp`. ASP.NET (Active Server Pages) is Microsoft's version of PHP and JavaServer Pages.

Start with HTML 4.0, scripts are permitted to run at the client side using the tag `<script>`.

The most popular scripting language for the client side is JavaScrpt.

JavaScript is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed.

JavaScript and PHP look similar as both of them are embedded inside the HTML files, they are processed differently. In the case of PHP, the browser collect information into a long string and sends it off to the server as a request for a PHP page. The server loads the PHP file and executes the embedded PHP script to produce a new HTML page. That page is then sent back to the browser for display.

For JavaScript, the work is done locally, inside the browser. There may be no contact with the server. Consequently, the result of a JavaScript is displayed virtually instantaneously, while the PHP may have a delay.

There are some other scripts used at client side:

- VBScript: based on Visual Basic used on Windows platforms.
- Java applets: small Java programs that have been compiled into machine instructions for JVM (Java virtual machine).
- ActiveX controls: programs compiled to x86 machine language and executed on the bare hardware.

In general, JavaScript is easier to write, Java applets execute faster, and ActiveX controls run faster than others. But ActiveX is basically platform dependent while Java applets are more portable.

JavaScript is also used in server-side network programming with runtime environments such as Node.js, game development and the creation of desktop and mobile applications.