Application Layer

Stand-alone Applications

Applications that don't require data exchange over the network and can work on their own are called stand-alone applications. For example, many video and image editors, word processors and CAP tools are such applications. Currently also these applications are becoming connected, as there are, for example, photo editing applications that allow you to share images over the Internet. Soon we will probably have no standalone applications, except in critical security systems and real time systems.

Networked Applications

Unlike standalone applications, need to connect to other applications over the network. There are some applications that are inherently networked, such as e-mail, web browser and FTP tools for file transfer. On the other hand, there are applications that we have made connected because of their utility. A good example is a word processor which lets you collaborate with other authors around the globe simultaneously on a shared document.

Protocols

In order to communicate over the network, these applications need to follow certain rules so that there is consistency. The combinations of rules and suggestions are called protocols. They can be different aspects of these protocols, but some of the basic aspects that every application layer protocol should provide, include the type of data or payload format. That can be exchange by these protocols and the way it should be formatted and packaged.

Web

The web is the most dominant way for humans to exchange and consume information and media over the Internet. The key technologies that enable this exchange of information are the following. HTML or the Hypertext Markup Language that defines how the content or webpages should be formatted. A web browser, the application that reads the HTML files and presents them in a human understandable formats.

URI

A URI or Uniform Resource Identifier, is a way to address pages and information on the web. This is how you and the web browser know where to go to find the next piece of information. For example, www.google.com is a URI which tells you want to go for a Google search. Finally, HTTP, the hypertext transfer protocol. You have a browser that will access the information and URI, the location from where to access it and HTML is the format of the information.

HTTP

Now the question is how will the browser send a request and receive the HTML pages? That is where HTTP comes in. It's the protocol via which applications communicate. Is not the only one, but is the one you encounter the most. Out of these technologies, the one that concerns us the most is HTTP. So let's look at it in the light of the characteristics we defined for application layer protocols previously, namely payload formats, message types, message syntax and rules. HTTP is the protocol via which your browser sends a request to remote server over the Internet. It is inherently a stateless protocol, meaning that it keeps no information about the state of the connection. Each subsequent request is

sent via a new TCP connection.

Keep Alive

There is an HTTP property called keep alive, which tells the server to keep the connection open for subsequent requests. As we are dealing with embedded systems that have limited memory and might not need tocommunicate that frequently, it is recommended to disable keep-alive to save up memory.

Video Processing Via HTTP

The WAV format is supported by all modern browsers. For video, H.264 is supported by all major browsers. And there are hardware decoders available for it too. It also offers good compression, but it might not be royalty free and requires a license. Among free video codecs, VP8 and Theora are the formats supported by the major browsers except for Internet Explorer. There are different protocols that allow you to stream media instead of merely downloading them in chunks.

Progressive Downloads

Progressive downloads, meaning downloading files in small chunks and streaming technically work the same way. A large file is cut into small pieces and then transferred. However, when progressively downloading a video file, you might have to wait for the whole file to download before you can play it. Or you can play it right away but you can move forward past the portion that has been already downloaded.

Streaming Protocols

You need streaming protocols to achieve fast forward, rewind, and play, pause features. The benefit of streamingmedia via HTTP is that the traffic goes through a port which is rarely blocked. It is also an affordable solution as the infrastructure is already available. And HTTP streaming solutions can scale according to the user's network. That is, the bit rate can be increased or decreased based on the user's network conditions. A few examples of HTTP video streaming protocols are HLS by Apple, MPEG-DASH developed under MPEG. It has been an international standard since 2012. And then there is Microsoft Smooth Streaming, supported by IIS, Microsoft's web server.