Protocols Are Interface Rules

Paul S. Wang, Sofpower.com

May 25, 2023

Generally, a *protocol* is a set of rules and conventions to follow for two parties to interface with each other to achieve some specific cooperation. These are independent parties such as individuals, organizations, clients and servers, or even countries. For example in foreign affairs, diplomatic protocols must be strictly followed. While ordinary people must follow precise rules when writing a letter, making a phone call, or conduct business with a bank.

For example when making a phone call we first dial a number (address of our target), when the call has been picked up, we say "Hello", and after talking we hang up the phone on both ends. These rules form the telephone protocol. Thus, protocols are everywhere, especially where there is an interface between independent systems.

But, in this article we will focus only on protocols for networking, in particular the Internet and the Web. Improved understanding in this direction will make you a better computational thinker and can help you apply the knowledge to make interfacing with others more efficient and effective in everything you do.

This post is part of our *Computational Thinking* (CT) blog where you can find many other interesting and useful articles.

What Are Networking Protocols

For various computing devices from different vendors, under different operating systems, to communicate on a network, a detailed set of rules and conventions must be established for all parties to follow. Such rules are known as *networking protocols*.

Each separate device connected on a network is known as a *host*. Example hosts include workstations, laptops. smartphones, smart TVs, Amazon Echo

speakers, servers on the cloud, and so on. Networking makes many different services available. Each networking service follow its own specially designed protocols. Protocols govern such details as

- Address format of hosts and processes
- Data format
- Manner of data transmission
- Sequencing and addressing of messages
- Initiating and terminating connections
- Establishing services
- Accessing services
- Data integrity, privacy, and security

Thus, for a *process* (an executing program) on one host to communicate with another process on a different host, both processes must follow the same protocol. The *Open System Interconnect* (OSI) *Reference Model* (Figure 1) provides a standard layered view of networking protocols and their interdependence. The corresponding layers on different hosts, and inside the net-

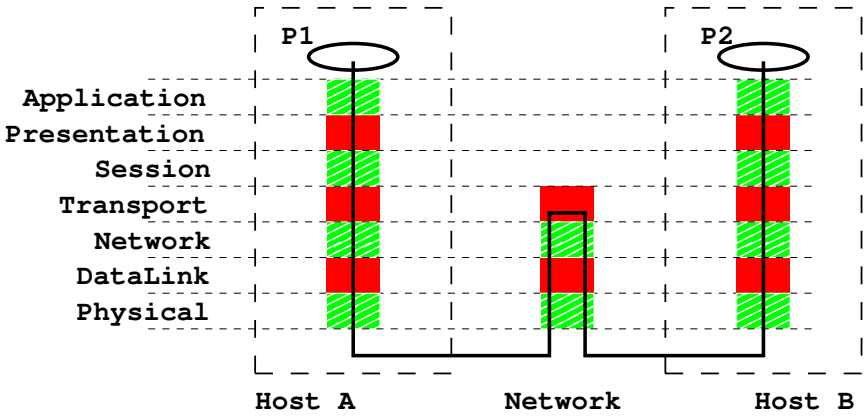

Figure 1: Networking Protocol Layers (OSI Model)

work infrastructure, perform complementary tasks to enable data exchange between the communicating processes (`P1` and `P2` in Figure 1).

Here is a brief summary of the OSI layers:

1. Physical–Provides mechanical and electrical connection specifications
2. Data Link–Organizes bits into frames
3. Network–Provides inter-network and packet movements
4. Transport–Enables data transport from source host to destination host
5. Session–Creates and ends a communication session
6. Presentation–Encodes, decodes, and translates data
7. Application–Gives network access to an end-user process

The physical, data-link, and network layers are handled by the network nodes and the transport, session, presentation, and application layers are the responsibility of the communicating hosts (Figure 2).

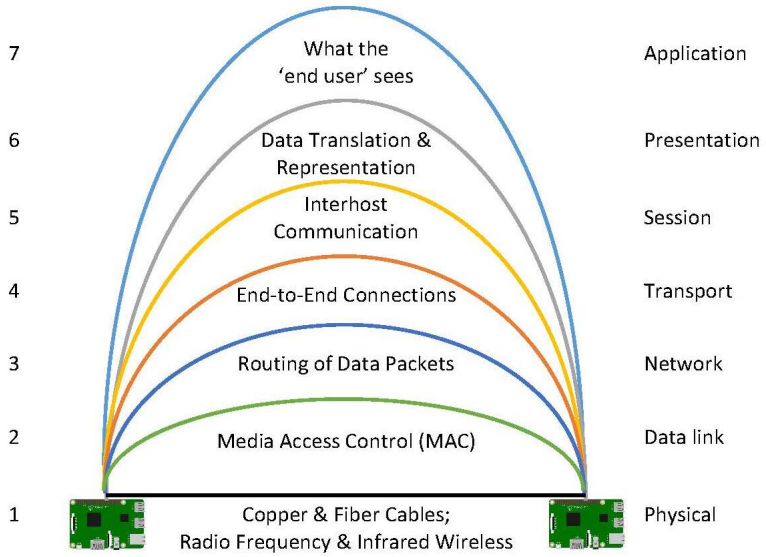

Figure 2: Another View of OSI Layers

Internet Protocols

Internet, the network that made the world a global viillage, started as a US Defense Department research project, the ARPANET. In 1971, ARPANET

had 15 nodes (23 hosts) in the United States (Figure 3)—A: Cambridge, MA, B: Stanford, CA, C: Pittsburgh, PA, D: Urbana, IL, E: Cleveland, OH, F: Los Angeles, CA, G: Lexington, MA, H: Salt Lake City, UT, I: Santa Barbara, CA, J: Mountain View, CA.

Figure 3: 1971 ARPANET

Among common networking protocols, the Internet Protocol Suite is the most widely used. The basic IP (*Internet Protocol*) is a *network layer* protocol. The TCP (*Transport Control Protocol*) and UDP (*User Datagram Protocol*) are at the *transport layer*. The HTTP (*Hypertext Transfer Protocol* is at the *application layer* and is used for the Web.

> **CT concept–*follow protocols***: *Only by following protocols can independent parties that may have never met cooperate smoothly.*

Protocols can affect the effectiveness and speed of the entire system.

Networking protocols are no mystery. Think about the protocol for making a telephone call. You (a client process) must pick up the phone, listen for the dial tone, dial a valid telephone number, and wait for the other side (the server process) to pick up the phone. Then you must say "hello," identify yourself, and so on. This is a protocol from which you cannot deviate if you want the call to be made successfully through the telephone network, and it is clear why such a protocol is needed. The same is true of a computer program attempting to talk to another computer program through a computer network. The design of efficient and effective networking protocols for different network services is an important area in computer science.

IP Addresses

Just like phones on a telephone neteork, every host on the Internet has its own network address that identifies the host for communication purposes. The addressing technique is an important part of a network and its protocols. An Internet IPv4 address is represented by 4 bytes in a 32-bit quantity. For example, `csail`, a host at MIT, has the IP address `128.30.2.109`. This *dot*

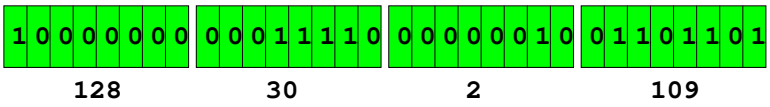

Figure 4: IPv4 Address

notation (or *quad notation*) gives the decimal value (0 to 255) of each byte. To accommodate its explosive growth, the Internet also uses IPv6, which supports 128-bit addresses. The IP address is similar to a telephone number in another way: the leading digits are like area codes, and the trailing digits are like local numbers.

Basically, the Internet transmits information by routing *data packets* (a well-organized block of data) from a source IP address to a destination IP address.

Domain Names

Because of its numerical nature, an IP address is easy on machines but hard on users. Therefore, any host may also have a *domain name* composed of words, rather like a postal address.

With domain names, the entire Internet name space for hosts is recursively divided into disjoint domains in a tree structure (Figure 5), similar to a file tree.

For example, the domain name `csail.mit.edu` identifies a host `csail` in the CSAIL Department at MIT. Domain names are for convenience. Some hosts may not need or have a domain name. The address for `frodo` puts it in the `csail` local domain, within the `mit` subdomain, which is under the `edu` *top-level domain* (TLD) for U.S. educational institutions. Other TLDs include `org` (nonprofit organizations), `gov` (U.S. government offices), `mil` (U.S. military installations), `com` (commercial outfits), `net` (network service providers), `uk` (United Kingdom), `cn` (China), and so forth. Within a local

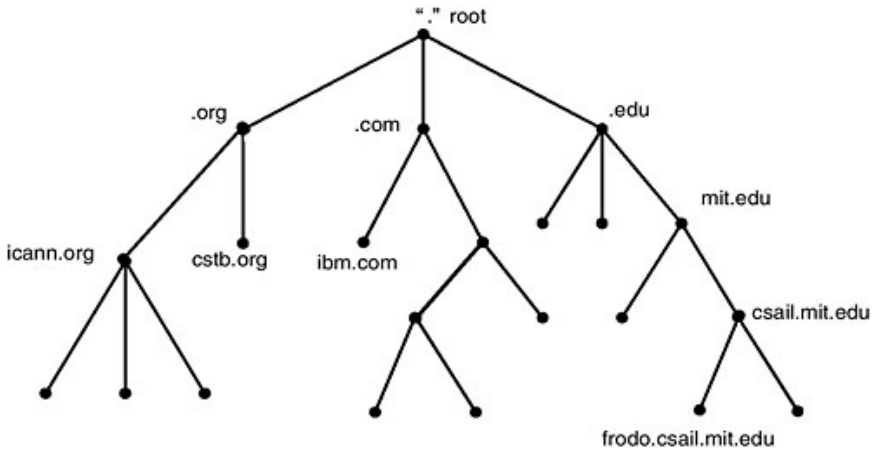

Figure 5: The Domain Name Tree

domain (for example, `csail.mit.edu`), you can refer to machines by their hostname alone (for example, `monkey`, `dragon`, `frodo`), but the full address must be used for machines outside.

All network applications accept a host address given either as a domain name or as an IP address. In fact, a domain name is first translated to a numerical IP address before being used to locate a host. The translation is done automatically via the dns service, an Internet based domain-to-IP dynamic mapping service.

Domain names are not created equal. Some are much better than others. People pay good money for great domain names that can help their business or cause.

Domain Registration

Anyone can obtain a domain name, often for creating a website, or some other purpose. To obtain a domain name, you need the service of a *domain name registrar*. Most will be happy to register your new domain name for a very modest yearly fee. Once registered, the domain name is property that belongs to the *registrant*. No one else can register for that particular domain name as long as the current registrant keeps the registration in good order.

ICANN accredits commercial registrars for common TLDs, including `.com`, `.net`, `.org`, and `.info`. Additional TLDs include `.biz`, `.pro`, `.aero`, `.name`, and `.museum`. Restricted domains (for example, `.edu`, `.gov`, and

`.us`) are handled by special registries such as `net.educause.edu` (for `.edu`), `nic.gov` (for `.gov`), and `nic.us` (for `.us`). Country-code TLDs are normally handled by registries in their respective countries.

Accessing Domain Registration Data

The registration record of a domain name is often publicly available. The standard Internet *whois* service allows easy access to this information. You can do this on the Web at `www.internic.net/whois.html`, for example. On Linux/Unix systems, easy access to whois is provided by the **whois** command,

whois *domain_name*

which lists the domain registration record kept at a registrar. Try the **Demo:** `WhoIs` demo at our companion website that can retrieve desired domain registration records.

Packet Switching

Data on the Internet are sent and received in *packets*. Thus, the Internet is a packet switching network. Similar to a letter, a packet envelops a small block of data with address information so the data can be routed through intermediate nodes on the network which is shared by all connected users. The network uses routing algorithms to efficiently forward packets to their final destinations.

Because there are multiple routes from the source to the destination host, the Internet is very reliable and can operate even if parts of the network are down. Figure 6 shows the structure details of an IPv4 packet. As you can see, the packet has two parts—the packet header contains destination, source, and other meta information (functioning as an envelop) and the packet body contains the actual data to be sent through the Internet to the destination.

We now turn our attention to the most important service on the Internet—the World Wide Web (WWW) or simply the Web.

Hypertext Transfer Protocol

As you may already know, Web browsers and Web servers communicate following HTTP, the *Hypertext Transfer Protocol*. It does not matter which

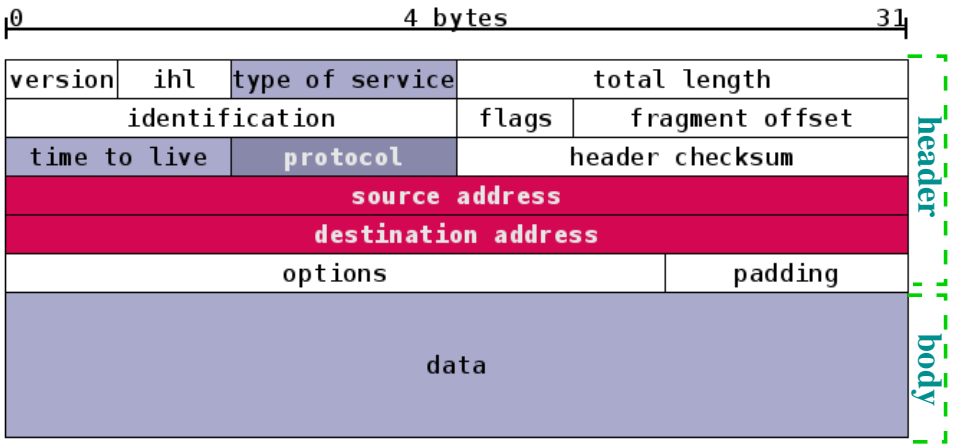

Figure 6: IPv4 Packet

browser is contacting what server; as long as both sides use the same protocol, everything will work.

In the early 1990s, HTTP gave the Web its start. HTTP/1.0 was standardized in the first part of 1996. Important improvements and new features have been introduced in HTTP/1.1, and it is now the stable version.

HTTP is an application layer (Figure 1) protocol that sits on top of TCP/IP, which provides reliable two-way connection between the Web client and Web server. We don't need all the details to understand the basics of HTTP.

1. A Web client, usually a browser but can be any user agent (UA), sends an HTTP *query* to a server.

2. A Web server, upon receiving a query, sends back an HTTP *response*.

A query and a response form an HTTP *transaction*. Each transaction stands alone and has no protocol-provided means to be correlated with any other transaction. Figure 7 illustrates an HTTP transaction.

A simple HTTP transaction goes as follows:

1. *Connection*—A browser (client) opens a connection to a server.

2. *Query*—The client requests a resource controlled by the server.

3. *Processing*—The server receives and processes the request.

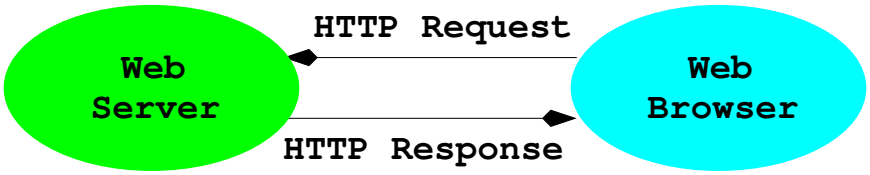

Figure 7: An HTTP Transaction

4. *Response*—The server sends the requested resource, or an error, back to the client.

5. *Termination*—The transaction is finished, and the connection is closed unless it is kept open for another request immediately from the client on the other end of the connection.

HTTP governs the format of the query and response messages (Figure 8). Basically, each query or request consists of an *initial line,* one or more *header lines* and an optional *body.* The initial line and header lines are textual (ISO-

initial line	(different for query and response)
HeaderKey1: *value1*	(zero or more header fields)
HeaderKey2: *value2*	
	(an empty line with no characters)
Optional message body contains query or response data.	
Its data type and size are given in the headers.	

Figure 8: HTTP Query and Response Formats

8859-1). Each line should end in RETURN and NEWLINE, but it may end in just NEWLINE.

The initial line identifies the message as a query or a response.

- A query line has three parts separated by spaces: a *query method* name, a local path of the requested resource, and an HTTP version number. For example,

```
GET   /path/to/file/index.html   HTTP/1.1
HOST: domain_name
```

or

```
POST   /path/script.php   HTTP/1.1
HOST: domain_name
```

The `GET` method requests the specified resource and does not allow a message body. A `GET` method can invoke a server-side program by specifying the CGI or active-page path, a question mark, and then a *query string*:

```
GET /CT_join.php?name=value1&email=value2   HTTP/1.1
HOST:  computize.org
```

Unlike `GET`, the `POST` method allows a message body and is designed to work with HTML forms for collecting input from Web users. The `POST` message body transmits name-value pairs just like a query string.

- A response (or status) line also has three parts separated by spaces: an HTTP version number, a status code, and a textual description of the status. Typical status lines are

  ```
  HTTP/1.1   200   OK
  ```

  for a successful query or

  ```
  HTTP/1.1   404   Not Found
  ```

  when the requested resource cannot be found.

- When the HTTP response contains a message body, the `Content-Type` and `Content-Length` headers are set so the client will know how to process it.

The Web borrowed the content type designations from the Internet email system and uses the same MIME (Multipurpose Internet Mail Extensions) defined content types. Hundreds of standard MIME content types are listed at the IANA (Internet Assigned Numbers Authority) site (`iana.org`).

The content type information allows browsers to decide how to process the incoming content. HTML, text, images, audio, and video may be handled

by the browser directly. Other types, such as PDF and Flash, are usually handled by plug-ins or external helper programs.

When using a browser to access the Web, the HTTP messages between it and the Web servers are kept behind the scenes. But it is possible to expose these messages and gain real experience with HTTP. See **Demo:** `Http` at our companion website.

While HTTP transmits information in the open, HTTPS (HTTP Secure) is a secure protocol that simply applies HTTP over a secure transport layer protocol *Transport Layer Security* (TLS 1.2) that is derived from the earlier *Secure Sockets Layer* (SSL).

Secure Website Login

Login is also required for shopping and other business on the Web. Websites often have areas and services reserved for members who are registered or have accounts with the site/organization. Each user often must log in to gain access to member-only or account-specific information or services. Access to a login page and all pages under login control normally uses HTTPS (Secure HTTP) to protect the user ID and password from eavesdropping en route in the network. For better security, it is now standard practice to require all websites, without regard to login, to use HTTPS instead of HTTP. Any

Figure 9: The HTTPS Lock Symbol

webpage that uses HTTPS will display a lock symbol at the beginning of the location or address line (Figure 9). The symbol simply means HTTPS

is being used for the page and you can click on the lock symbol to check on the *digital certificate* for SSL/TSL being used. It is a good idea that you do check it. The FBI has warned the public not be fooled by the lock symbol. It at best tells you that the HTTPS protocol is being used on that particular webpage. **A phishing website can use HTTPS as well. Yet it has been designed to steal your personal information**. HTTPS is a good thing and all website should use it but we need to understand its limitations.

What Is a Digital Certificate?

In secure communication, the very first concern is that the parties are actually who they say they are. A *digital certificate* is a document (computer file) signed by a *certificate authority* (CA) that can vouch for the identity of the certificate holder. A CA is usually a well-established third party that is in the business of verifying credentials and issuing certificates digitally signed by the CA. Certificates can be issued for different purposes and different domains: Web server, email, digital signature, payment systems, and so on. SSL/TLS certificates are widely used by Web servers to enable HTTPS access. The largest CAs include Symantec[1], Comodo Group, Go Daddy, Thawte, and GlobalSign.

A CA issues a digital certificate for a customer after carefully verifying the identity and legitimacy of the person or organization (the client). Each certificate is a digital ID and contains the identity of the client, the client's public key, the expiration date of the certificate, and details of the issuing CA. A digital certificate is often issued for a certain specific purpose and is installed in applications that use it for that particular security purpose.

Figure 10 shows some details of the Web server certificate used by the US Social Security Administration on its `secure.ssa.gov` site. Digital certificates follow standardized formats, for example, X509v3, and are used by security programs within applications, such as Web servers, Web browsers, and email clients. Servers that support HTTPS need to install valid SSL/TLS certificates. A certificate not issued by a widely recognized CA or has expired can cause a browser warning about the certificate's status, allowing the end user to accept or reject the certificate.

Certificates for CAs are issued by other CAs. A *root CA* is one that signs its own certificate.

[1]Symantec acquired VeriSign's Security Business.

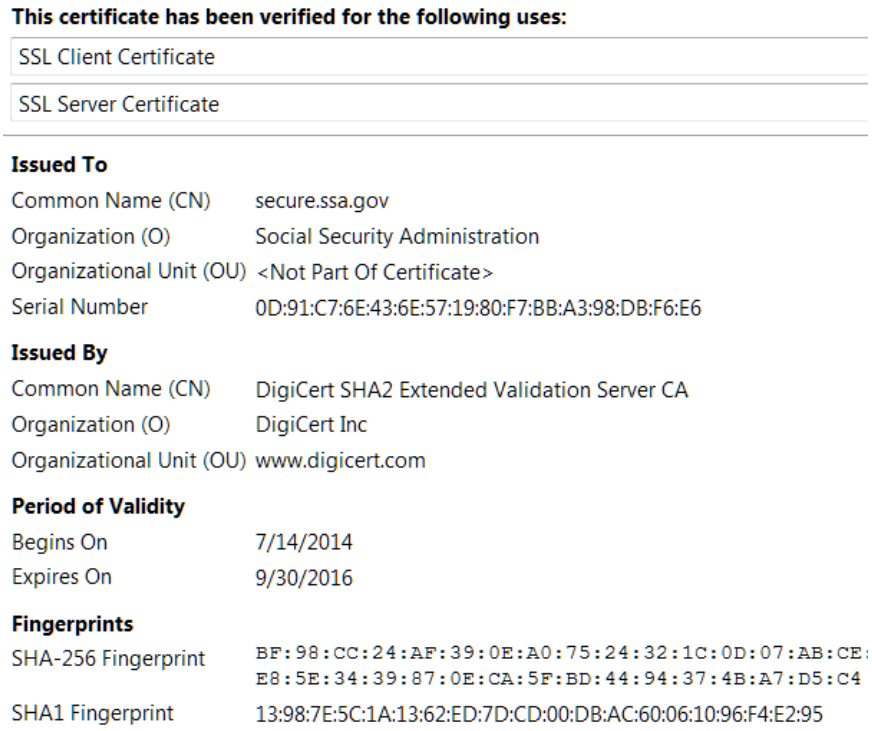

Figure 10: Sample Web Server Certificate Details

Organizations often set up their internal certification system with a root CA controlled by their own company. This way, large companies and organization can issue digital certificates for internal use, without paying fees to commercial CAs.

HTTPS and SSL/TLS

Web servers support HTTPS for secure communication between the client and the server.

HTTPS is HTTP (Hypertext Transfer Protocol) over *Secure Socket Layer* (SSL) or the newer *Transport Layer Security* (TLS) protocol (Figure 11). Note different network services use different *network ports* on servers. For example, port 80 for HTTP and port 443 for HTTPS are standard. However, nonstandard ports can use protocols of their own design. SSL/TLS developed from SSL 1.0, 2.0, and 3.0 to TLS 1.0, 1.1, and 1.2. SSL/TLS provides secure communication between client and server by allowing mutual authen-

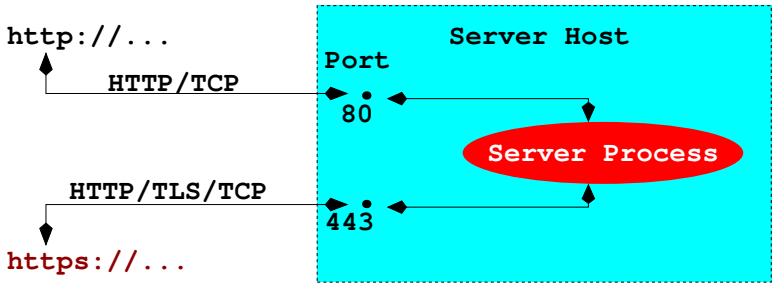

Figure 11: HTTP vs. HTTPS

tication, the use of digital signatures for integrity, and data encryption for confidentiality. To enable HTTPS, a server needs to install a valid Web server certificate and enable SSL/TLS.

SSL/TLS may be placed between a reliable connection-oriented transport protocol layer, such as TCP/IP, and an application protocol layer, such as HTTP (Figure 12).

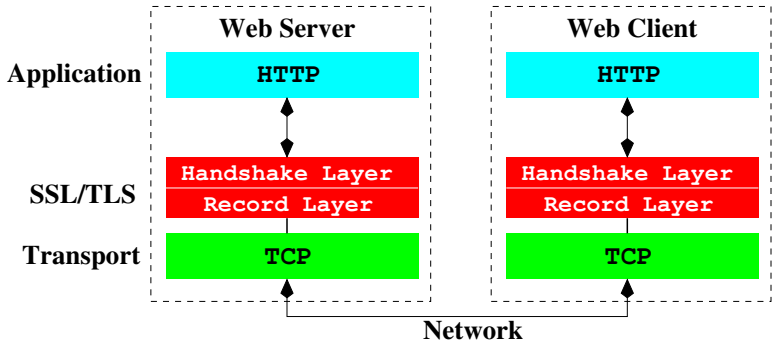

Figure 12: HTTPS Protocol Layers

Basically, TLS sets up secure communication in two steps:

1. The *handshake phase*—Mutual authentication and securely agreeing upon a randomly generated *session key* to be used in the next phase

2. The *session data phase*—Following the Record layer protocol, using the session key for encryption of messages between the client and server

The handshake phase uses *public-key cryptography* for security, while the session data phase uses the more efficient symmetric encryption for speed.

Each new SSL/TLS connection will establish a new session key. Figure 13 illustrates the TLS handshake process from a user viewpoint.

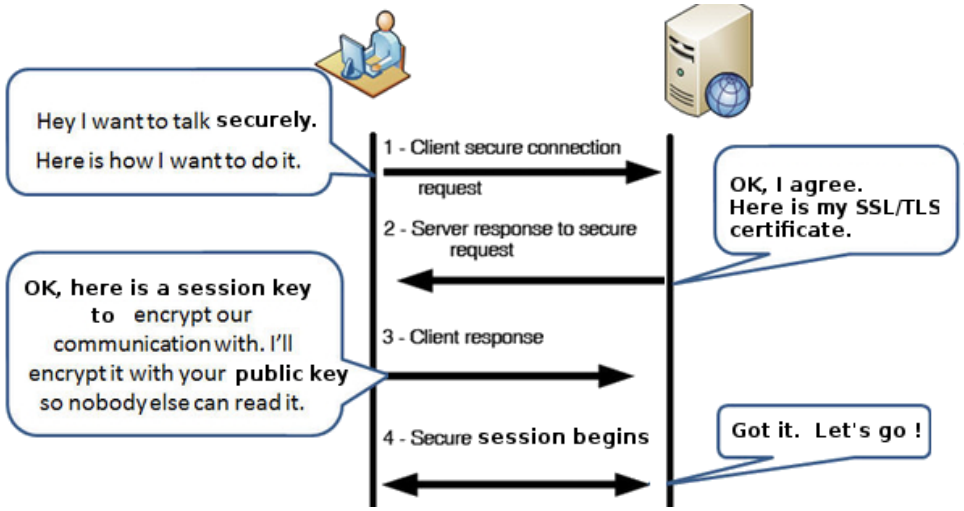

Figure 13: Basic TLS Handshake

Finally

Hosts and processes on a network follow well-define protocols to interface with one another. In this article we focused on the Internet protocols, IP, TCP/IP, UDP/IP, HTTP(S), SSL/TLS.

There of course are many other networking protocols including SMTP (Simple Mail Transfer Protocol) for email, FTP/SFTP for file transfer, SSH for secure shell, and DHCP for dynamic host configuration (automatically assign local IP addresses to connecting host on a LAN dynamically), and many others. A list of current Internet services, their protocols and assigned port numbers can be found at the IANA site. Port numbers for Internet services are assigned in various ways, based on three ranges: standard system ports (0-1023), user ports (1024-49151), and the dynamic and/or private Ports (49152-65535).

> **CT concept–*Protocols are interface rules***: *Only by following exact and well-defined rules can unrelated and independent entities interact successfully.*

The Internet is wonderful and used by everyone globally all the time. In fact if someone says "the Internet is down!", then it usually means disaster or at least a major inconvenience for all involved. The Internet as an infrastructure is like running water or electric power, you can hardly do without.

And all the various Internet services depend on protocols to work smoothly and efficiently. Computational thinkers, we must realize that paying attention to interfaces with machines, programs, organizations, or other people is important. And if something goes wrong at the interface, something bad can happen!