computers are bad

You are receiving this facsimile because you signed up for fax delivery of this newsletter. To stop delivery, contact Computers Are Bad by email or fax.

https://computer.rip - me@computer.rip - fax: +1 (505) 926-5492

2021-03-27 the actual osi model

I have said before that I believe that teaching modern students the OSI model as an approach to networking is a fundamental mistake that makes the concepts less clear rather than more. The major reason for this is simple: the OSI model was prescriptive of a specific network stack designed alongside it, and that network stack is not the one we use today. In fact, the TCP/IP stack we use today was intentionally designed differently from the OSI model for practical reasons.

Teaching students about TCP/IP using the OSI model is like teaching students about small engine repair using a chart of the Wankel cycle. It's nonsensical to the point of farce. The OSI model is not some "ideal" model of networking, it is not a "gold standard" or even a "useful reference." It's the architecture of a specific network stack that failed to gain significant real-world adoption.

Well, "failed to gain real-world adoption" is one of my favorite things, so today we're going to talk about the OSI model and the OSI network stack.

The story of the OSI model basically starts in the late '70s with a project between various standards committees (prominently ISO) to create a standardized network stack which could be used to interconnect various systems. An Open Systems Interconnection model, if you will.

This time period was the infancy of computer networking, and most computer networks operated on vendor-specific protocols that were basically overgrown versions of protocols designed to connect terminals to mainframes. The IBM Systems Network Architecture was perhaps the most prominent of these, but there were more of them than you could easily list.

Standardized network protocols that could be implemented across different computer architectures were relatively immature. X.25 was the most popular, and continues to be used as a teaching example today because it is simple and easy to understand. However, X.25 had significant limitations, and was married to the telephone network in uncomfortable ways (both in that it relied on leased lines and in that X.25 was in many ways designed as a direct analog to the telephone network). X.25 was not good enough, and just as soon as it gained market share people realized they needed something that was more powerful, but also not tied to a vendor.

The OSI network stack was designed in a very theory-first way. That is, the OSI conceptual model of seven layers was mostly designed before the actual protocols that implemented those layers. This puts the OSI model in an unusual position of having always, from the very start, been divorced from actual working computer networks. And

while this is a matter of opinion, I believe the OSI model to have been severely over-engineered from that beginning.

Unlike most practical computer networks which aim to provide a simple channel with few bells and whistles, the OSI model attempted to encode just about every aspect of what we now consider the "application" into the actual protocols. This results in the OSI model's top four layers, which today are all essentially "Application" spelled in various strange ways. Through a critical eye, this could be viewed as a somewhat severe example of design over function. History had, even by this time, shown that what was needed from computer networks was usually ease of implementation and ease of use, not power.

Unfortunately, the OSI model, as designed, was powerful to a fault.

From the modern perspective, this might not be entirely obvious, but only because most CS students have been trained to simply ignore a large portion of the model. Remember, the OSI model is:

1. Please (Physical)
2. Do (Data Link)
3. Not (Network)
4. Throw (Transport)
5. Sausage (Session)
6. Pizza (Presentation)
7. Away (Application)

Before we get too much into the details of these layers, let's remember what a layer is. The fundamental concept that the OSI model is often used to introduce is the concept that I call network abstraction: each layer interacts only with the layer below it, and by doing so provides a service to the layer above it.

Each layer has a constrained area of concern, and the protocol definitions create a contract which defines the behavior of each layer. Through this sort of rigid, enforced abstraction, we gain flexibility: the layers become "mix and match." As long as layers implement the correct interface for above and expect the correct interface from below, we can use any implementation of a given layer that we want.

This matters in practice. Consider the situation of TCP and UDP: TCP and UDP can both be dropped on top of IP because they both expect the same capabilities from the layer under them. Moreover, to a surprising extent TCP and UDP are interchangeable. While they provide different guarantees, the interface for the two is largely the same, and so switching which of the two software uses is trivial (in the simple case where we do not require the guarantees which TCP provides) [1].

So, having hopefully grasped this central concept of networking, let's apply it to the OSI model, with which it was likely originally taught to us. The presentation layer depends on the session layer, and provides services to the application layer. That's, uhh, cool. Wikipedia suggests that serializing data structures is an example of something which might occur at this layer. But this sort of presupposes that the session layer does not require any high-level data structures, since it functions without the use of the presentation layer. It also seems to suggest that presentation is somehow dependent on session, which makes little sense in the context of serialization.

In fact, it's hard to see how this "fundamental concept" of the presentation layer

applies to computing systems because it does not. Session and presentation are both "vestigial layers" which were not implemented in the IP stack, and so they have no real modern equivalent. Most teaching of the session and presentation layers consists of instructors grasping for examples--I have heard of things like CHAP as the session layer--which undermine the point they are making by violating the actual fundamental concept of layered networking.

Now that we all agree that the OSI model is garbage which does not represent the real world, let's look at the world it does represent: the OSI protocols, which were in fact designed explicitly as an implementation of the OSI model.

Layer 1

No one really defines layer 1, the physical layer, because it is generally a constraint on the design of the protocols rather than something that anyone gets to design intentionally. The physical layer, in the context of the OSI stack, could generally be assumed to be a simple serial channel like a leased telephone line, using some type of line coding and other details which are not really of interest to the network programmer.

Layer 2

Layer 2, the data link layer, provides the most fundamental networking features. Today we often talk of layer 2 as being further subdivided into the MAC (media access control) and LLC (logical link control) sublayers, but to a large extent this is simply a result of trying to retcon the OSI model onto modern network stacks, and the differentiation between MAC and LLC is not something which was contemplated by the actual designers of the OSI model.

The data link layer is implemented primarily in the form of X.212. In a major change from what you might expect if you were taught the IP stack via the OSI model, the OSI data link link layer and thus X.212 provides reliability features including checksumming and resending. Optionally, it provides guaranteed order of delivery. X.212 further provides a quality of service capability.

Specifically related to order of delivery, X.212 provides a connection-oriented mode and a connectionless mode. This is very similar (but not quite the same) to the difference between TCP and UDP, but we are still only talking about layer 2! Keep in mind here that layer 2 is essentially defined within the context of a specific network link, and so these features are in place to contend with unreliable links or links that are themselves implemented on other high-level protocols (e.g. tunnels), and not to handle routed networks.

X.212 addressing is basically unspecified, because the expectation is that addresses used at layer 2 will be ephemeral and specific to the media in use. Because layer 2 traffic cannot directly span network segments, there is no need for any sort of standardized addressing.

As with most layers, there are alternative implementations available for the data link layer, including implementations that transport it over other protocols.

Layer 3

OSI layer 3, the network layer, provides a more sophisticated service which is capable of moving bytes between hosts with basically the same semantics we expect in the IP world. Layer 3 is available in connection oriented and connectionless modes, much like layer 2, but now provides these services across a routed network.

The two typical layer 3 protocols are Connectionless Network Protocol and Connection Oriented Network Protocol, which are basically exactly what they sound like.

OSI addressing at these layers is based on Network Service Point Addresses or NSAPs. Or, well, it's better to say that NSAPs are the current standard for addressing. In fact, the protocols are somewhat flexible and historically other schemes were used but have been largely replaced by NSAP. NSAP addresses are 20 bytes in length and have no particular structure, although there are various norms for allocation of NSAPs that include embedding of IP addresses. NSAPs do not include routing information as is the case with IP addresses, and so the process of routing traffic to a given NSAP includes the "translation" of NSAPs into more detailed addressing types which may be dependent on the layer 2 in use. All in all, OSI addressing is confusing and in modern use depends very much on the details of the specific application.

Layer 4

Layer 4, the transport layer, adds additional features over layer 3 including multiplexing of multiple streams, error recovery, flow control, and connection management (e.g. retries and reconnects). There are a variety of defined layer 4 protocol classes called TP0 thru TP4, which vary in the features that they offer in ways that do not entirely make sense from the modern perspective.

Because layer 4 offers general messaging features, it is perhaps the closest equivalent to the TCP and UDP protocols in the IP stack, but of course this is a confusing claim since there are many elements of UDP and TCP found at lower levels as well.

The selection of one of the five transport layer "levels" depends basically on application requirements and can range from very high reliability (TP4) to low latency given unreliable network conditions, with relaxed guarantees (TP0 or TP1).

Layer 5

The session layer adds management of associations between two hosts and the status of the connection between them. This is a bit confusing because the IP model does not have an equivalent, but it might help to know that, in the OSI model, connections are "opened" and "closed" using the session protocol X.215 (which causes actions which cascade down to the lower layers). The concept of connection state in the OSI session layer is much higher level than TCP's basic concept of handshakes, and connections are expected to be more permanent.

More interestingly, though, the session layer is responsible for very high-level handling of significant network errors by gracefully restarting a network dialog. This is not a capability that the IP stack offers unless it is explicitly included in

an application.

The session layer manages conversations through a token mechanism, which is somewhat similar to that of token-ring networking or the general “talking stick” concept. Multiple tokens may be in use, allowing for half-duplex or duplex interactions between hosts.

Like basically every layer below it, Layer 5 comes in connection-oriented and connectionless flavors. The connectionless flavor is particularly important since it provides powerful features for session management without the requirement for an underlying prepared circuit--something which is likewise often implemented at the application layer over UDP.

Layer 6

Layer 6, the presentation layer, is another which does not exist in the IP stack. The session layer is a bit hard to understand from the view of the IP stack, but the presentation layer is even stranger.

The basic concept is this: applications should interact using abstract representations rather than actual wire-encoded values. These abstract values can then be translated to actual wire values based on the capabilities of the underlying network.

Why is this even something we want? Well, it’s important to remember that this network stack was developed in a time period when text encoding was even more poorly standardized than now, and when numeric representation was not especially well standardized either (with various types of BCD in common use).

So, for two systems to be able to reliably communicate, they must establish an acceptable way to represent data values... and it is likely that a degree of translation will be required. The OSI presentation layer, defined by X.216, nominally adjusts for these issues by the use of an abstract representation transformed to and from a network representation. There are actually a number of modern technologies that are similar in concept, but they are seldom viewed as network layers [2].

Layer 7

Finally, the application layer is actually where, you know, things are done. While the application layer is necessarily flexible and not strongly defined, the OSI stack nonetheless comes with a generous number of defined application layer protocols. While it’s not particularly interesting to dig into these all, it is useful to note a couple that remain important today.

X.500, the directory service application protocol, can be considered the grandparent of LDAP. If you think, like all sane people, that LDAP is frustratingly complicated, boy you will love X.500. It was basically too complex to live, but too important to die, and so it was pared down to the “lightweight” LDAP.

Although X.500 failed to gain widespread adoption, one component of X.500 lives on today, nearly intact: X.509, which describes the cryptographic certificate feature of the X.500 ecosystem. The X.509 certificate format and concepts are directly used

today by TLS and other cryptographic implementations, including its string representations (length-prefixed) which were a decent choice at the time but now quite strange considering the complete victory of null-terminated representations.

X.400, the messaging service protocol, is basically the OSI version of email. As you would expect, it is significantly more powerful and complicated than email as we know it today. For a long time, Microsoft Exchange was better described as an X.400 implementation than an email implementation, which is part of why it is a frightening monstrosity. The other part is everything about modern email.

And that is a tour of the OSI network protocols. I could go into quite a bit more depth, but I have both a limited budget to buy ISO standards and a limited attention span to read the ones I could get copies of. If you are interested, though, the OSI stack protocols are all well defined by ITU standards available in the US from ISO or from our Estonian friends for much cheaper. For a fun academic project, implement them: you will be perhaps the only human alive who *truly* understands the OSI model ramble your data communications professor indulged in.

[1] Contrast SCTP, which provides an interface which is significantly different from the UDP and TCP bytestream, due to features such as multiple streams. Not unrelatedly, SCTP has never been successful on the internet.

[2] I think that this is actually a clue to the significant limitations of the OSI model for teaching. The OSI model tends to create a perception that there is one “fixed” set of layers with specified functions, when in actual modern practice it is very common to have multiple effective layers of what we would call application protocols.