Welcome to Network Communication part 2, the final module in the Polycom Fundamentals series. This module is approximately 9 minutes long.
In order to understand how videoconferencing works it’s important to understand the underlying technologies at work behind the scenes. In this short module we will talk further about network communication models.

We will begin with a recap of the TCP/IP model and where some familiar protocols fit into it.
Here is another look at the diagram depicting the TCP/IP model and related protocols; we are going to take a look at how these tie together by digging a bit deeper into some of them.

This all starts really with the standard called Ethernet. Ethernet is a huge subject which covers local area networking, but here we will just focus on a small part of how Ethernet defines the way in which our payload moves through the network.

Ethernet separates the stream of data into frames, as we have already mentioned, but what does that mean?
A frame is just a name for the arrangement of data encapsulated with a source and destination address. Ethernet sits at Layer 1 and 2 on the OSI model, or in the Link layer in the TCP/IP model, and the addressing model which sits at this layer is Media Access Control, or MAC addresses.

Switches can be described as ‘Layer 2’ devices, meaning that they operate using the MAC address to direct frames. This means that each frame does not need to take the same route through the network, adding the address information allows it to go wherever the network sends it and still end up in the same place. This, though, only works within the same local network, as we will see shortly.
This is what a frame looks like:

- The preamble allows synchronization to the clock on the signal. This is required to ensure that each frame is sent at exactly the correct time.
- The Start Frame Delimiter (SFD) indicates the beginning of the frame.
- The source and destination address are MAC addresses. The destination can be to one device, but it can also be a broadcast which goes to all devices on the network, or a multicast to a sub-set of devices.
- Next comes the length of the payload. This will be a value from 0 (empty) to 1500 (maximum size).
- After the length information we have our payload, which must be from 46-1500 bytes in total. If the length is less than 46, padding is added so that this section is at least 46 bytes long. 1500 bytes is the maximum transmission unit, or MTU, size allowed by the Ethernet standard. Included in here is any other header information from any other layers in use.
- Finally comes a Frame Check Sequence (FCS), which performs error checking on the frame. This is calculated at the source and then recalculated at the destination. If the calculations do not give the same answer then the frame is assumed to be corrupt and is discarded.
So that takes care of networking within a local area, but what about sending something outside your own network? Clearly this is something which we do a lot, sending email, surfing the internet, videoconferencing and so on. Well, routers understand Layer 2 and Layer 3 which uses a different addressing system. Layer 3 takes care of the Internet Protocol standard, which uses IP addresses. IP addresses can of course be used to route data wherever we need.
Remember that IP is a connection-less protocol, so any information sent without the protection of TCP at the next layer up may be lost or discarded. Here is what an IPv4 packet looks like:

It identifies the version of the IP protocol being used, specifies the header length and the total packet length, provides a Type Of Service (TOS) identifier, gives source and destination IP addresses and of course has the payload in there too, along with some information which shows which protocol the payload requires. There are also some additional options but we won’t get into those here.
The Transport layer gives us two options for encapsulation of each payload. The first is UDP, which is also connection-less. The second is TCP, which is connection-oriented, which is to say it makes sure the other end is ready to receive data and checks at regular intervals to ensure that everything is going OK. For videoconferencing our signaling is usually sent using TCP, as we want to be sure the signaling is setup correctly, and the media is usually sent using UDP, partly because TCP would be too slow, but also because in real time everything is moving so quickly that if one or two packets are dropped you’re likely not to notice as the picture will be rebuilt so quickly.
Because UDP is not very complex, it doesn’t have to add much information in its header, as you can see here; it is really just the source and destination port numbers (remember we don’t need any other addressing information because this is already taken care of by the Internet and Link layers), the length of the header plus the payload (known collectively as a datagram), and a checksum.
TCP requires a bit more information to add the reliability needed. Again we see the source and destination port numbers and a checksum, but quite a bit of other stuff too.

- There is a sequence number for tracking the delivery of each TCP segment, an acknowledgment which is equal to the sequence number of the previously received packet +1
- Next comes an acknowledgment which says that it is OK to send the next packet in the sequence
- We also see here a number of flags – these can be used to define the type of message, such as URG (Urgent), ACK (Acknowledgment), EOM (End Of Message), RST (Reset link), SYN (Synch), FIN (Finish)
- The window gives the amount of data that can be transmitted before an acknowledgment is required
- Additionally there is an urgent pointer, which enables an urgent message to be processed immediately rather than waiting its turn in line

TCP also uses an options field which we won’t look at further.
And finally, we come to the Application layer, which is where the real-time video and audio enter the scene, as it were. As we know, our video and audio is looked after by Real-time Transport Protocol or RTP, and it’s partner in crime, RTCP, or Real-time Transport Control Protocol.
Here's a quick look at the RTP packet. It is slightly more sophisticated than at the Transport layer as it can define certain variables about the media being sent, for example if this particular payload is the start of a new video frame.

- **Ver** gives us the version of RTP being used
- **P** is for padding and is used to indicate if the payload contains padding blocks at the end to fill the space up to a minimum required size
- **X** indicates that there is a header extension following the standard header
- **CC** gives the count of contributing resources in the stream
- **M** is our marker, which can indicate a significant packet such as the start of a new video frame
- **Payload Type**
- **Sequence number** is the same as in TCP – it allows the far end to make sure everything is unpacked in the right order
- **The timestamp** indicates the date and time the packet was produced
- **The Synchronization Source Identifier** acts as a unique reference for the origin of this specific media stream
- **The Contributing Source Identifiers** are used to indicate if this stream contains information from multiple contributing sources
So there we have it; the videoconferencing data plus the RTP header is encapsulated by the Transport layer as UDP, then encapsulated by the Internet layer with the source and destination IP addresses, then encapsulated by the Link layer with the source and destination MAC addresses, then sent through the network.

At the other end, it is unpacked again at each level until the endpoint at the Application layer receives the RTP and unpacks it.
Thank You