Source Ethernet ID: See Destination Ethernet ID above.

Type (protocol ID): 16 bits. Indicates the protocol being used, e.g. 0x0800 for IP, 0x809b for Appletalk, 0x8137 for Netware IPX/SPX. This is the mechanism by which different protocols can coexist on the same Ethernet.

Data: This consists of the IP packet (in the case of the IP protocol). Its length is inferred by subtracting the lengths of the other fields from the overall frame length.

CRC: 32 bits. An error-checking field for this frame.

Postamble: End-of-frame indicator, a special 8-bit pattern.

4.6 Putting It All Together

Suppose that in our earlier examples, **svr** is running on Venus, and **wps** is running on Honda. Suppose we have the following IP and Ethernet (MAC) addresses:

machine	IP address	MAC address
Earth	192.0.0.0	0x0123456789ab
Mars	192.0.0.1	0x1123456789ab
Venus	192.0.0.2	0x2123456789ab
Saturn	192.0.0.3	0x3123456789ab
Jeep	193.0.0.0	0x4123456789ab
Honda	193.0.0.1	0x5123456789ab

All of the machines here have **Class C** IP addresses, which consist of a 24-bit network number and 8-bit host-within-network number. For example, Mars is host 1 on network 192.0.0.0. Here is what will happen when **wps** executes

```
write(SD,argv[2],strlen(argv[2]));
```

at Honda. Recall that argv[2] is either “w” or “ps”; let’s say it’s “ps”. Recall also that SD is a socket which **wps** has already opened in TCP. Also, the call which **wps** made earlier to connect() had connected this socket to port 2000 at Venus, and the OS at Honda had assigned **wps** an ephemeral port number, say 3056.

So, the effect of the write() is that the socket number SD and the string argv[2] will be sent from **wps**, which is running in the Application Layer at Honda, to the Session Layer at Honda.

The Session Layer will find in its records that this socket is for ephemeral port 3056 on TCP. So, the Session Layer at Honda will pass “ps” and this port number to TCP in the Transport Layer at Honda. (Note that this passing is done by a simple function call, since we are at the same machine.)

TCP at Honda will first have to decide how much “chunking” to do — none in this case, since the data consist of only two bytes! TCP will now prepare a TCP packet containing that data, using the TCP packet format shown above:

TCP will first note that ephemeral port 3056 is associated with the destination port and IP numbers 2000 and 192.0.0.2, respectively. TCP will then fill in the packet, putting 3056 and 2000 for the Source and Destination Port numbers; it will fill in the Sequence number, etc., and finally put “ps” into the Remaining Bytes (i.e. data) field. After creating this packet, TCP at Honda will pass it to IP in the Network Layer of Honda, along with the information that the destination will be 192.0.0.2.

IP at Honda will now prepare an IP packet. It will fill in 193.0.0.1 for the Source IP address, and 192.0.0.2 for the Destination IP. Note carefully that for the Remaining Bytes field in this case, IP will put in the entire packet that it received from TCP.

IP at Honda will now decide how to route the IP packet it has created. It first will ask whether the destination host, is on the same network as Honda. The answer to that question will be no, since Honda is on the network 193.0.0 and Venus is on 192.0.0. So, IP at Honda will need to send the packet to a router on Honda’s network. There are two such routers, Jeep and Citroen.

IP at Honda will send the packet to Jeep. (This will probably have been hand-coded; more on this in our unit on routing.) Note that IP at Honda will not know that Venus is just one hop away from Jeep; it merely knows that the first step should be Jeep. So, IP will pass the packet, plus Jeep’s Ethernet address, 0x4123456789ab, to the Data Link Layer at Honda.

The Data Link Layer will then create an Ethernet frame. It will put 0x4123456789ab for the Destination Ethernet ID, and 0x5123456789ab for the Source Ethernet ID. It will fill in 0x0800 for the Type field. And it will put in the entire IP packet it received from the Network Layer in for the current Data field. Finally, the Data Link Layer will pass this Ethernet frame to the Ethernet device driver on Honda.

The Ethernet device driver on Honda will then put the frame on network B. All NICs on that network will see it, including the NIC 0x4123456789ab on Jeep.

That NIC will say, “Oh, this frame is for me!” Note that the NIC will not notice that the ultimate destination of the frame is Venus; all the NIC cares about is the Destination Ethernet address, which it has seen is its own. All the NIC will do is pass this frame up to the next layer at Jeep, [24] which will be the Data Link Layer.

The Data Link Layer at Jeep will strip off the Ethernet IDs and other Ethernet-related information. The stripped-down frame is now the IP frame which IP at Honda had produced. The Data Link Layer will know this, since the Type field in the Ethernet frame stated that the protocol was IP. The Data Link Layer at Jeep will now pass the IP packet to IP in the Network Layer at Jeep.

IP will now look at the Destination IP Address in the packet, 192.0.0.2. Since that does not match Jeep’s own address, 193.0.0.0, Jeep knows that it needs to route this packet. IP also notices that the Destination IP Address is on network number 192.0.0, i.e. network A, which Jeep’s machine is attached to via another NIC. So, Jeep will be able to relay the packet directly to Venus:

At this point the packet will go down the protocol stack at Jeep, just like we saw earlier at Honda. The Ethernet Source ID will be Jeep’s, i.e. 0x4123456789ab, and the Ethernet Destination ID will be Venus’, 0x2123456789ab. The frame will be placed onto network A, and picked up by Venus.

[24] Note that it is the same protocol stack as that of Mars. They are the same machine, but have two different NICs and thus different names.

The frame will then go up the protocol stack at Venus like it did at Jeep, but in this case IP at Venus will discover that the Destination IP Address is that of Venus. Thus the packet will not be routed from Venus, but instead will continue to go up the protocol stack. IP at Venus will see in the Transport Layer Protocol field that this is a TCP packet (and thus not, for example, UDP). IP will now strip off the IP-only fields from the packet, leaving the original TCP packet. IP will pass the packet to TCP, along with information on the Source IP address.

TCP will note the Destination Port, strip off the TCP-only information, and send the remainder to the Session Layer, along with the Destination Port number. The Session Layer will send the remainder to read(), and **svr** will be able to read the “ps”.

When **svr** writes back to **wps**, a similar sequence of events will occur. Note also that when **wps** first called connect(), a similar sequence of events occurred then too, as Honda’s TCP and Venus’ TCP exchanged messages in order to set up a connection.

5 Application-Layer Protocols

Some applications, such as FTP, e-mail, etc. are so common that they have their own protocols.

For example, e-mail uses the Simple Mail Transfer Protocol (SMTP). Suppose you are lm@abc.com and are sending e-mail to uvw@xyz.org, and your message is going to be a simple one-line greeting:

```
Hi, how have you been?
```

The client, which will be either the e-mail utility that you use, or an OS function called by that utility, will first establish a TCP connection to the SMTP server at the remote machine, at port 25 of the server. The following messages would then be sent by the client:[25]

```
HELO abc.com
MAIL FROM: lm@abc.com
RCPT TO: uvw@xyz.org
DATA
Hi, how have you been?
.
QUIT
```

(The end of the e-mail message itself is indicated by a period on a separate line, as shown above.)

Similarly, HTTP, the protocol used for Web access, consists of a set of commands similar to the HELO, MAIL FROM:, RCPT TO:, DATA and QUIT commands we saw for SMTP above. The HTTP server is at port 80.

[25]There will be responses from the server for each one, but we will ignore them here.