Some More Details on IP and TCP

Norman Matloff
Dept. of Computer Science
University of California at Davis

June 9, 1999

1 IP

1.1 Review

The Internet Protocol (IP) is the Network layer section of the TCP/IP suite. For a sending node, IP will receive a packet from either TCP or UDP in the Transport layer—a "chunk" of a message in the case of TCP, officially called a **segment**, or an entire message in the case of UDP, officially called a **datagram**—then determine initial routing, then encapsulate the packet with an IP header and send it to the Link layer. The encapsulated packet is now called an **IP datagram** (not to be confused with the UDP term).

Note that the IP datagram will typically pass through several different networks on its way to its destination. It will be passed from network to network by **routers**, each one of which executes IP code.

On the receiving end, the reverse occurs.

1.2 Packet Format

Byte 0: The first four bits are the Version Number (currently4, going to 6), and the other four bits are the Header Length in words.

Byte 1: Type of Service field, intended to give priority to some packets but not used much in practice.

Bytes 2-3: Length field, giving length of the entire IP datagram including data.

Bytes 4-5: Datagram ID number; used in the case of fragmentation (see below).

Byte 6, Bits 0-2: Flags: First bit is reserved, value 0; second bit is 1 if Do Not Fragment flag; third bit is 1 if More Fragments Remaining flag. (No other flags in IP V.4.)

Byte 6, Bits 3-7, and Byte 7: Offset (position of this fragment in relation to the original datagram).

Byte 8: Time to Live field. If this equals, say, k, then this packet will be allowed k more hops through the network. If it hasn't reached its destination by then, it is discarded, to prevent infinite routing loops.

Byte 9: Transport-layer protocol (e.g. 0x01 for ICMP, 0x06 for TCP, 0x11 for UDP, 0x59 for OPSF route-optimization messages).

Bytes 10-11: Checksum, to check for errors within this datagram.

Bytes 12-15: Source IP address.

Bytes 16-19: Destination IP address.

Bytes 20-whatever: Options including blank padding to make an integral number of words.

Remaining Bytes: Data. Remember, from the point of view ofthe IP layer, the "data" consists of a TCP or UDP packet(or other packet from a higher layer).

1.3 Fragmentation

Consider an IP datagram originating at X, say from TCP, with destination Y. Each network which this datagram passes through on its way from X to Y will have its own limitation on the size of a frame, called the **maximum transfer unit** (MTU). For example, Ethernet does not allow a frame to be longer than 1500 bytes. A PPP phone connection often sets this limit as well. For FDDI, the MTU is 4500 bytes.

Suppose that X is on an FDDI network, and that the TCP layer at X forms this datagram (at that layer, called a segment) at size 4500 bytes. Say the IP layer sends out that 4500-byte datagram in one piece. But among the various networks this datagram traverses on its way to Y, some of them might consist of, say, Ethernets, in which case our datagram would not be transmittable.

To cope with this problem, IP occasionally does **fragmentation** of a datagram before placing it on a network. This means that the given datagram is split into two or more smaller ones with sizes under the MTU for that network.

Note that once a datagram is split in this manner, it is never reassembled again until it reaches its destination, even though subsequent networks which it traverses may have larger MTUs. The various fragments may be widely separated as they make their way to Y, even arriving out of order, but the IP layer will be able to piece them together again via their common ID number (Bytes 4-5 in the IP format), the More Fragments Remaining flag (Byte 6), and the Offset field (Byte 7).

An Offset value of k means, "This fragment will contain data bytes starting with the k-th byte of the original datagram (calling the first byte 0)." If for example the first fragment is 228 bytes long, its More Fragments Remaining flag will be set, and the second fragment will have the same ID as the first but with its offset value being 228. Say there is a third fragment, and that the size of the second fragment had been 50. Then the second fragment will have its More Fragments Remaining flag set, and the third one will again have the same ID, with an Offset value of 278.

So, we now see that "chunking" can occur at several different layers. The original message may have been broken down into segments by TCP at the Transport layer, and each segment might be broken into fragments at the Network layers of various routers that our message passes through. Furthermore, each fragment might itself be broken down by, say, HDLC, when passing through a phone link at the Link layersomewhere en route. In each case, the corresponding layer will also do the reassembly: IP and TCP will reassemble fragments into datagrams, and datagrams/segments into full messages, respectively, at the destination Y; but

HDLC will reassemble HDLC frames back into an IP fragment right away, on the receiving end of the phone line.

If IP sets the Do Not Fragment flag and a datagram reaches a network which it is too big for, the router there will send back an ICMP message to the source node X, and simply discard the datagram.

1.4 Error Control

The Network layer, IP does not do error control for the data, a job handled by TCP in the Transport layer.[1] But the IP format does include a Checksum field, whose purpose is to check the IP datagram header. After all, if the header is in error, say in its destination address, then the datagram will be useless, and of course it could not be checked at the destination—since it would never get there!

The Checksum field consists of the (slightly modified) sum of all the IP datagram's header, taken as 16-bit words. A receiving IP layer will compute this sum itself, and compare with the claimed value. The sum itself is 1s-complement, meaning ordinary addition with the carries out of the most significant bit added to the sum. Note that the Checksum field must be recomputed by each router before sending on to the next one, since the Time to Live field changes in each hop.

2 TCP

As noted earlier, TCP breaks a message (or more accurately, the message stream) into "chunks," formally called **segments**, translates sockets to IP addresses, and passes the encapsulated packages to IP. The reverse occurs at the receiving end.

2.1 Segment Format

Bytes 0-1: Source Port.

Bytes 2-3: Destination Port.

Bytes 4-7: Sequence Number.

Bytes 8-11: ACK.

Byte 12: The first four bits comprise the Header Length field (the other four bits are 0s), meaning the number of words (not bytes) from the beginning of the packet to the first data byte within the packet. Most of the header is of fixed length, so we would not need this, except for the fact that the Options field below is of variable length. The software thus uses this field to deduce where in the packet the data starts.

Byte 13: Flags, which are various bits giving control information such as a PUSH command (which tells the host not to continue accumulating bytes to send; "send whatever you have now, without waiting for more").

[1] There may also be some error control at the Link layer. Ethernet, for instance, has a CRC field; it checks that field but simply discards frames which are in error, rather than signaling for retransmission. HDLC also has a CRC field, and it does signal for retransmission.

Bytes 14-15: Advertised Window, a value that the recipient uses to say, "OK, you can now send me this many bytes without waiting for an ACK."

Bytes 16-19: Checksum (two bytes), for error checking, and a two-byte Urgent Pointer field.

Bytes 20-whatever: Options field.

Remaining Bytes: Data, e.g. your e-mail message in the case of **sendmail**. (Note: No length field is needed for specifying the amount of data, since this can be deduced from a similar field in the IP header which will contain this TCP packet.)

2.2 Slidow-Window Protocol

Each byte in the stream sent in a TCP connection is given a sequence number. The initial sequence number, not necessarily 0, is negotiated when the connection is established. Sequence numbers from A to B are separate from those from B to A. The Sequence Number field will state the number of the first byte in the current segment.

A sending TCP layer may wait to send until enough bytes from a given stream are accumulated to "make it worthwhile," in the sense of overhead. Setting the Push flag will force TCP to send all of the bytes accumulated in the stream so far.

A receiving TCP layer will try to **piggyback** its ACKs with its own data. So a segment, say from A to B, will contain not only the data A is sending to B, but also A's ACK of receipt of data from B. A value of k in the ACK field means that the receiving TCP layer has received all bytes up through sequence number k-1 correctly, and is now expecting byte k.

If a sending TCP layer does not receive an ACK for a given segment within the timeout period, it will send again. If its original transmission had merely been late in arriving at the receiving TCP layer, then there may be both duplicate transmissions and duplicate ACKs, which both sides must check for. The typical cause for lack of an ACK is that the buffers are full at some intermediate router and the segment is discarded there.

A receiving TCP layer can change the window size dynamically. This information too is piggybacked with the receiving layer's own data to be sent.

2.3 Error Control

The Checksum field is computed similarly to that of IP, except that it sums the entire segment, both header and data. Unlike Link-layer error control, there is no such thing as NAK; the receiving TCP layer simply discards the segment, and lets the sender timeout.