Byte Stuffing

Before closing our discussion of PPP framing, let us consider a problem that arises when any protocol uses a specific bit pattern in a flag field to delineate the beginning or end of the frame. What happens if the flag pattern itself occurs elsewhere in the packet? For example, what happens if the flag field value of 01111110 appears in the information field? Will the receiver incorrectly detect the end of the PPP frame?

One way to solve this problem would be for PPP to forbid the upper-layer protocol from sending data containing the flag field bit pattern. The PPP requirement of transparency discussed above obviates this possibility. An alternative solution, and the one taken in PPP and many other protocols, is to use a technique known as **byte stuffing**.

PPP defines a special control escape byte, 01111101. If the flag sequence, 01111110 appears anywhere in the frame, except in the flag field, PPP precedes that instance of the flag pattern with the control escape byte. That is, it "stuffs" (adds) a control escape byte into the transmitted data stream, before the 01111110, to indicate that the following 011111110 is *not* a flag value but is, in fact, actual data. A receiver that sees a 01111110 preceded by a 01111101 will, of course, remove the stuffed control escape to reconstruct the original data. Similarly, if the control escape byte bit pattern itself appears as actual data, it too must be preceded by a stuffed control escape byte. Thus, when the receiver sees a single control escape byte by itself in the data stream, it knows that the byte was stuffed into the data stream. A pair of control escape bytes occurring back to back means that one instance of the control escape byte appears in the original data being sent. Figure 5.35 illustrates PPP byte stuffing. (Actually, PPP also XORs the data byte being escaped with 20 hexadecimal, a detail we omit here for simplicity.)

5.7.2 PPP Link-Control Protocol (LCP) and Network-Control Protocols

Thus far, we have seen how PPP frames the data being sent over the point-to-point link. But how does the link get initialized when a host or router on one end of the PPP link is first turned on? The initialization, maintenance, error reporting, and shutdown of a PPP link is accomplished using PPP's **link-control protocol** (**LCP**) and family of PPP network-control protocols.

Before any data is exchanged over a PPP link, the two peers (one at each end of the PPP link) must first perform a considerable amount of work to configure the link, in much the same way that a TCP sender and receiver must perform a three-way handshake (see Section 3.5) to set the parameters of the TCP connection before TCP data segments are transmitted. Figure 5.36 illustrates the state transition diagram for the LCP protocol for configuring, maintaining, and terminating the PPP link.

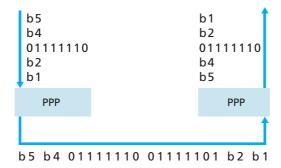


Figure 5.35 ♦ Byte stuffing

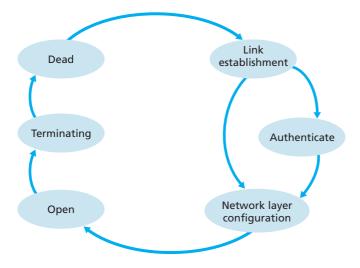


Figure 5.36 ◆ PPP link-control protocol states

The PPP link always begins and ends in the dead state. When an event such as a carrier detection or network administrator intervention indicates that a physical layer is present and ready to be used, PPP enters the link-establishment state. In this state, one end of the link sends its desired link configuration options using an LCP configure-request frame (a PPP frame with the protocol field set to LCP and the PPP information field containing the specific configuration request). The other side then responds with a configure-ack frame (all options acceptable), a configure-nak frame (all options understood but not acceptable), or a

configure-reject frame (options not recognizable or not acceptable for negotiation). LCP configuration options include a maximum frame size for the link, the specification of an authentication protocol (if any) to be used, and an option to skip the use of the address and control fields in PPP frames.

Once the link has been established, link options negotiated, and the authentication (if any) performed, the two sides of the PPP link then exchange network layer—specific network-control packets with each other. If IP is running over the PPP link, the IP control protocol [RFC 1332] is used to configure the IP protocol modules at each end of the PPP link. IPCP data are carried within a PPP frame (with a protocol field value of 8021), just as LCP data are carried in a PPP frame. IPCP allows the two IP modules to exchange or configure their IP addresses and negotiate whether or not IP datagrams will be sent in compressed form. Similar network-control protocols are defined for other network-layer protocols, such as DECnet [RFC 1762] and AppleTalk [RFC 1378]. Once the network layer has been configured, PPP may then begin sending network-layer datagrams—the link is in the opened state and data has begun to flow across the PPP link. The LCP echo-request frame and echo-reply frame can be exchanged between the two PPP endpoints in order to check the status of the link.

The PPP link remains configured for communication until an LCP terminate-request packet is sent. If a terminate-request LCP frame is sent by one end of the PPP link and replied to with a terminate-ack LCP frame, the link then enters the dead state.

In summary, PPP is a link-layer protocol by which two communicating link-level peers, one on each end of a point-to-point link, exchange PPP frames containing network-layer datagrams. The principal components of PPP are:

- Framing. A method for encapsulating data in a PPP frame, identifying the beginning and end of the frame, and detecting errors in the frame.
- Link-control protocol. A protocol for initializing, maintaining, and taking down the PPP link.
- Network-control protocols. A family of protocols, one for each upper-layer network protocol, that allows the network-layer modules to configure themselves before network-level datagrams begin flowing across the PPP link.

5.8 Link Virtualization: A Network as a Link Layer

Because this chapter concerns link-layer protocols, and given that we're now nearing the chapter's end, let's reflect on how our understanding of the term *link* has evolved. We began this chapter by viewing the link as a physical wire connecting two communicating hosts, as illustrated in Figure 5.2. In studying multiple access