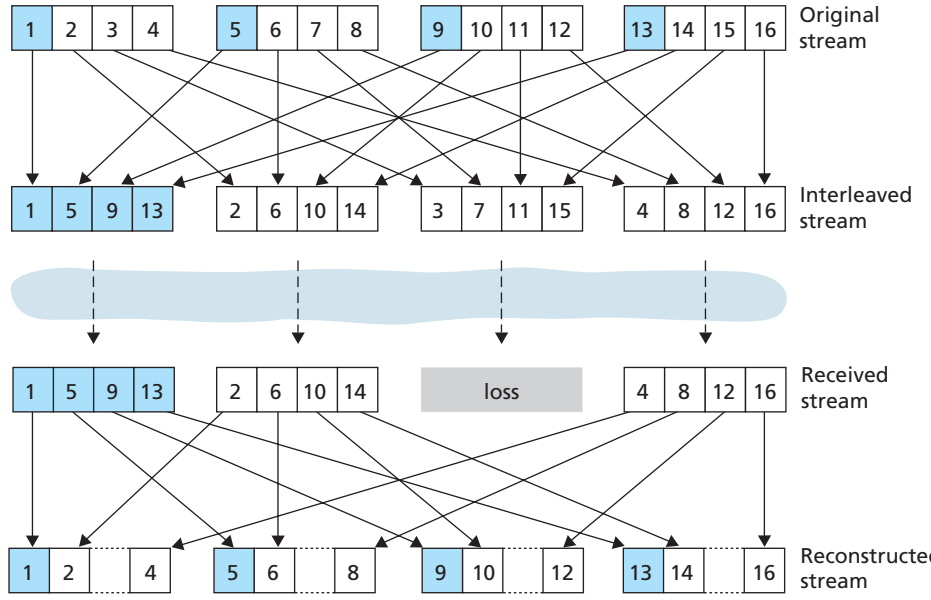

Figure 7.9 ♦ Sending interleaved audio

Error Concealment

Error concealment schemes attempt to produce a replacement for a lost packet that is similar to the original. As discussed in [Perkins 1998], this is possible since audio signals, and in particular speech, exhibit large amounts of short-term self-similarity. As such, these techniques work for relatively small loss rates (less than 15 percent), and for small packets (4–40 msecs). When the loss length approaches the length of a phoneme (5–100 msecs) these techniques break down, since whole phonemes may be missed by the listener.

Perhaps the simplest form of receiver-based recovery is packet repetition. Packet repetition replaces lost packets with copies of the packets that arrived immediately before the loss. It has low computational complexity and performs reasonably well. Another form of receiver-based recovery is interpolation, which uses audio before and after the loss to interpolate a suitable packet to cover the loss. Interpolation performs somewhat better than packet repetition but is significantly more computationally intensive [Perkins 1998].

7.3.4 Case Study: VoIP with Skype

Skype is an immensely popular VoIP application with over 50 million accounts active on a daily basis. In addition to providing host-to-host VoIP service, Skype offers host-to-phone services, phone-to-host services, and multi-party host-to-host

video conferencing services. (Here, a host is again any Internet connected IP device, including PCs, tablets, and smartphones.) Skype was acquired by Microsoft in 2011 for over $8 billion.

Because the Skype protocol is proprietary, and because all Skype's control and media packets are encrypted, it is difficult to precisely determine how Skype operates. Nevertheless, from the Skype Web site and several measurement studies, researchers have learned how Skype generally works [Baset 2006; Guha 2006; Chen 2006; Suh 2006; Ren 2006; Zhang X 2012]. For both voice and video, the Skype clients have at their disposal many different codecs, which are capable of encoding the media at a wide range of rates and qualities. For example, video rates for Skype have been measured to be as low as 30 kbps for a low-quality session up to almost 1 Mbps for a high quality session [Zhang X 2012]. Typically, Skype's audio quality is better than the "POTS" (Plain Old Telephone Service) quality provided by the wire-line phone system. (Skype codecs typically sample voice at 16,000 samples/sec or higher, which provides richer tones than POTS, which samples at 8,000/sec.) By default, Skype sends audio and video packets over UDP. However, control packets are sent over TCP, and media packets are also sent over TCP when firewalls block UDP streams. Skype uses FEC for loss recovery for both voice and video streams sent over UDP. The Skype client also adapts the audio and video streams it sends to current network conditions, by changing video quality and FEC overhead [Zhang X 2012].

Skype uses P2P techniques in a number of innovative ways, nicely illustrating how P2P can be used in applications that go beyond content distribution and file sharing. As with instant messaging, host-to-host Internet telephony is inherently P2P since, at the heart of the application, pairs of users (that is, peers) communicate with each other in real time. But Skype also employs P2P techniques for two other important functions, namely, for user location and for NAT traversal.

As shown in Figure 7.10, the peers (hosts) in Skype are organized into a hierarchical overlay network, with each peer classified as a super peer or an ordinary peer. Skype maintains an index that maps Skype usernames to current IP addresses (and port numbers). This index is distributed over the super peers. When Alice wants to call Bob, her Skype client searches the distributed index to determine Bob's current IP address. Because the Skype protocol is proprietary, it is currently not known how the index mappings are organized across the super peers, although some form of DHT organization is very possible.

P2P techniques are also used in Skype **relays**, which are useful for establishing calls between hosts in home networks. Many home network configurations provide access to the Internet through NATs, as discussed in Chapter 4. Recall that a NAT prevents a host from outside the home network from initiating a connection to a host within the home network. If *both* Skype callers have NATs, then there is a problem—neither can accept a call initiated by the other, making a call seemingly impossible. The clever use of super peers and relays nicely solves this problem. Suppose that when Alice signs in, she is assigned to a non-NATed super peer and initiates a session to that super peer. (Since Alice is initiating the session, her NAT permits this session.) This session allows Alice and her super peer to

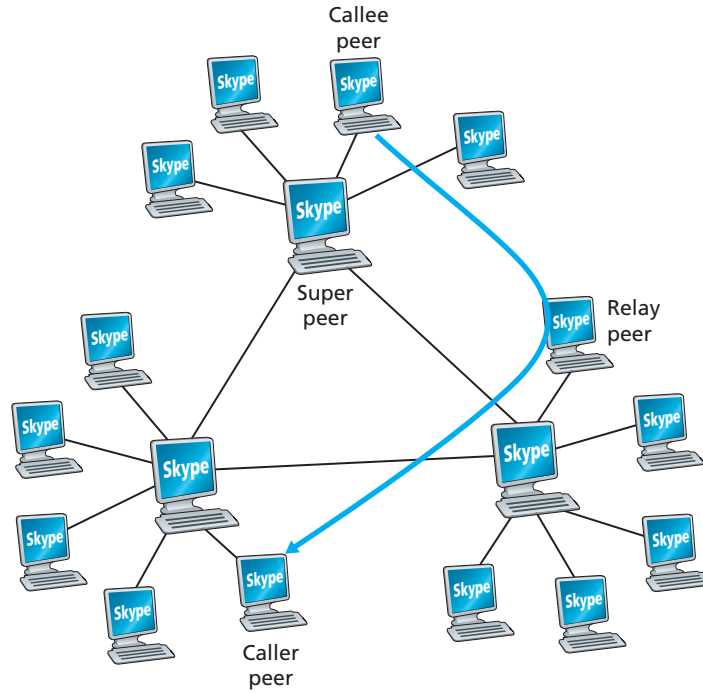

Figure 7.10 ♦ Skype peers

exchange control messages. The same happens for Bob when he signs in. Now, when Alice wants to call Bob, she informs her super peer, who in turn informs Bob's super peer, who in turn informs Bob of Alice's incoming call. If Bob accepts the call, the two super peers select a third non-NATed super peer—the relay peer—whose job will be to relay data between Alice and Bob. Alice's and Bob's super peers then instruct Alice and Bob respectively to initiate a session with the relay. As shown in Figure 7.10, Alice then sends voice packets to the relay over the Alice-to-relay connection (which was initiated by Alice), and the relay then forwards these packets over the relay-to-Bob connection (which was initiated by Bob); packets from Bob to Alice flow over these same two relay connections in reverse. And *voila!*—Bob and Alice have an end-to-end connection even though neither can accept a session originating from outside.

Up to now, our discussion on Skype has focused on calls involving two persons. Now let's examine multi-party audio conference calls. With $N > 2$ participants, if each user were to send a copy of its audio stream to each of the $N - 1$ other users, then a total of $N(N - 1)$ audio streams would need to be sent into the network to support the audio conference. To reduce this bandwidth usage, Skype employs a clever distribution

technique. Specifically, each user sends its audio stream to the conference initiator. The conference initiator combines the audio streams into one stream (basically by adding all the audio signals together) and then sends a copy of each combined stream to each of the other $N - 1$ participants. In this manner, the number of streams is reduced to $2(N - 1)$. For ordinary two-person video conversations, Skype routes the call peer-to-peer, unless NAT traversal is required, in which case the call is relayed through a non-NATed peer, as described earlier. For a video conference call involving $N > 2$ participants, due to the nature of the video medium, Skype does not combine the call into one stream at one location and then redistribute the stream to all the participants, as it does for voice calls. Instead, each participant's video stream is routed to a server cluster (located in Estonia as of 2011), which in turn relays to each participant the $N - 1$ streams of the $N - 1$ other participants [Zhang X 2012]. You may be wondering why each participant sends a copy to a server rather than directly sending a copy of its video stream to each of the other $N - 1$ participants? Indeed, for both approaches, $N(N - 1)$ video streams are being collectively received by the N participants in the conference. The reason is, because upstream link bandwidths are significantly lower than downstream link bandwidths in most access links, the upstream links may not be able to support the $N - 1$ streams with the P2P approach.

VoIP systems such as Skype, QQ, and Google Talk introduce new privacy concerns. Specifically, when Alice and Bob communicate over VoIP, Alice can sniff Bob's IP address and then use geo-location services [MaxMind 2012; Quova 2012] to determine Bob's current location and ISP (for example, his work or home ISP). In fact, with Skype it is possible for Alice to block the transmission of certain packets during call establishment so that she obtains Bob's current IP address, say every hour, without Bob knowing that he is being tracked and without being on Bob's contact list. Furthermore, the IP address discovered from Skype can be correlated with IP addresses found in BitTorrent, so that Alice can determine the files that Bob is downloading [LeBlond 2011]. Moreover, it is possible to partially decrypt a Skype call by doing a traffic analysis of the packet sizes in a stream [White 2011].

7.4 Protocols for Real-Time Conversational Applications

Real-time conversational applications, including VoIP and video conferencing, are compelling and very popular. It is therefore not surprising that standards bodies, such as the IETF and ITU, have been busy for many years (and continue to be busy!) at hammering out standards for this class of applications. With the appropriate standards in place for real-time conversational applications, independent companies are creating new products that interoperate with each other. In this section we examine RTP and SIP for real-time conversational applications. Both standards are enjoying widespread implementation in industry products.