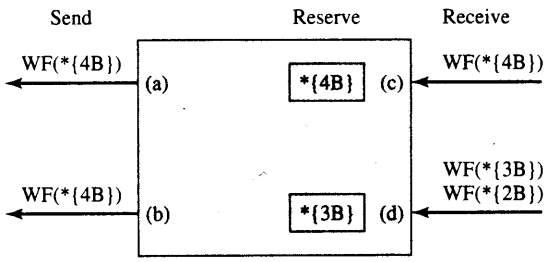

FIGURE 10.7 Wildcard-filter reservation example.

is the largest of the resource requests from all receivers, independent of the number of senders. The WF-style reservation request is symbolically represented by WF(*{Q}), where the asterisk denotes wildcard sender selection and Q denotes the flowspec. WF style is suitable for applications that are unlikely to have multiple senders transmitting simultaneously. Such applications include the audioconferencing example (page 822).

Figure 10.7 shows the WF-style reservation. For simplicity, the flowspec is assumed to be of one-dimensional quantity in multiples of some base resource quantity B. Interface (c) receives a request with a flowspec of 4B from a downstream node, and interface (d) receives two requests with flowspecs of 3B and 2B. The requests coming from interface (d) are merged into one flowspec of 3B so that this interface can support the maximum requirement. When forwarded by the input interfaces, the requests from interfaces (c) and (d) are further merged into a flowspec of 4B. The audioconference example corresponds to the case where all receivers request the same resource quantity.

The *fixed-filter (FF) style* creates a distinct reservation for each sender. Symbolically, this reservation request can be represented by FF(S1{Q1}, S2{Q2}, . . .), where S*i* is the selected sender and Q*i* is the resource request for sender *i*. The total reservation on a link for a given session is the sum of all Q*i*'s. The videoconferencing example (page 822) corresponds to this reservation style.

Figure 10.8 shows the FF-style reservation. Interface (c) receives a request with a flowspec of 4B for sender S1 and 5B for sender S2 and forwards FF(S1{4B}) to (a) and FF(S2{5B}) to (b). Note that the FF-style reservation is shared by all destinations. Interface (d) receives two requests: FF(S1{3B}, S3{B}) and FF(S1{B}). Interface (d) then reserves 3B for S1 and B for S3 and forwards FF(S1{3B}) to (a) and FF(S3{B}) to (b). Interface (a) then merges the two requests received from (c) and (d) and forwards

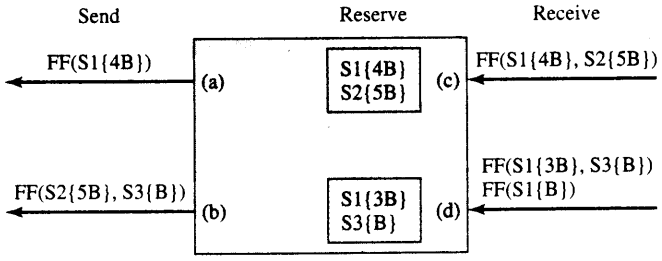

FIGURE 10.8 Fixed-filter reservation example.

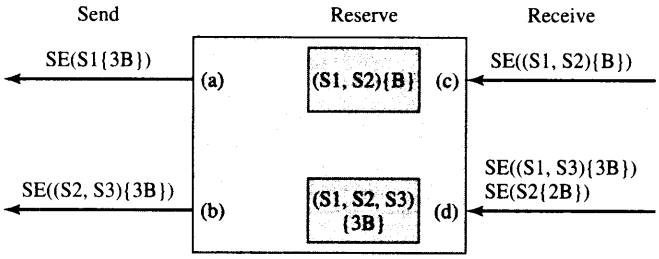

FIGURE 10.9 Shared-explicit reservation example.

FF(S1{4B}) upstream. Interface (b) packs the two requests from (c) and (d) and forwards FF(S2{5B}, S3{B}) upstream.

The *shared-explicit (SE) style* creates a single reservation shared by a set of explicit senders. Symbolically, this reservation request can be represented by SE(S1, S2, ... {Q}), where S*i* is the selected sender and Q is the flowspec.

Figure 10.9 shows the SE-style reservation. When reservation requests are merged, the resulting filter spec is the union of the original filter specs, and the resulting flowspec is the largest flowspec. This example corresponds to the layered-encoded videostreaming example.

10.2.4 Soft State

The reservation states that are maintained by RSVP at each node are refreshed periodically by using Path and Resv messages. When a state is not refreshed within a certain time-out period, the state is deleted. This type of state that is maintained by a timer is called *soft state* as opposed to hard state where the establishment and teardown of a state are explicitly controlled by signaling messages.

Because RSVP messages are delivered as IP datagrams with no reliability requirement, occasional losses can be tolerated as long as at least one of the K consecutive refresh messages gets through. Currently, the default value of K is 3. Refresh messages are transmitted once every R seconds, where the default value is 30 seconds. To avoid periodic message synchronization, the actual refresh period for each message should be randomized, say, using a uniform distribution in the range of $[0.5R, 1.5R]$. Each Path and Resv message carries a TIME_VALUES object containing the refresh period R. From this value, a local node can determine its state lifetime L, which is $L \geq 1.5 \times K \times R$.[2]

There is a trade-off between the refresh traffic overhead period and recovery time. The amount of refresh traffic overhead is *(Path_size + Resv_size)/R* bits/second. A short refresh period increases the refresh traffic overhead, while a long refresh period lengthens the recovery time.

Although a time-out will eventually occur in a path or reservation state if the corresponding refresh message is absent, RSVP can speed up state removals by utilizing

[2]The RSVP specification instead gives this formula: $L \geq (K + 0.5) \times 1.5 \times R$.

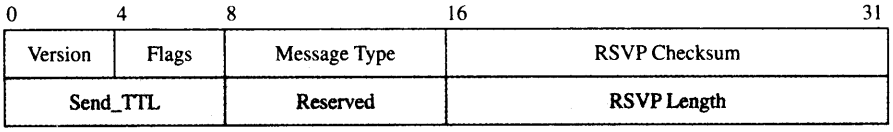

FIGURE 10.10 Format of common header.

teardown messages. There are two types of teardown messages: **PathTear** and **ResvTear**. A PathTear message travels from the point of origin, following the same paths as the Path messages towards the receivers, deleting the path state and dependent reservation state along the paths. A ResvTear message travels from the point of origin towards the upstream senders, deleting the reservation state along the way.

10.2.5 RSVP Message Format

Each RSVP message consists of a common header and a body consisting of a variable number of objects that depends on the message type. The objects in the message provide the information necessary to make resource reservations. The format of the common header is shown in Figure 10.10. The current protocol version is 1, and no flag bits are currently defined. The seven message types are Path, Resv, PathErr, ResvErr, PathTear, ResvTear, and ResvConf.

The RSVP checksum uses the one's complement algorithm common in TCP/IP checksum computation. The Send_TTL represents the IP TTL value with which the message was sent. It can be used to detect a non-RSVP hop by comparing the Send TTL value in the RSVP common header and the TTL value in the IP header. The RSVP length field indicates the total length of the RSVP message in octets, including the common header.

The format of the objects that follow the common header is shown in Figure 10.11. The length field indicates the total object length in octets. The length must be a multiple of four.

The Class-Num field identifies the object class. The C-Type value identifies the subclass of the object. The following object classes are defined:

NULL: A NULL object is ignored by the receiver.

SESSION: A SESSION object specifies the session for the other objects that follow. It is indicated by the IP destination address, IP protocol number, and destination port number. The session object is required in every message.

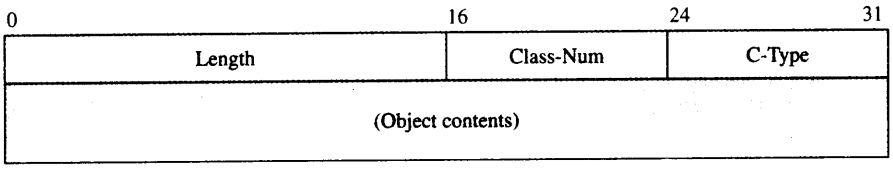

FIGURE 10.11 Format of each object.

RSVP_HOP: This object carries the IP address of the RSVP-capable router that sent this message. For downstream messages (e.g., Path from sender to receiver) this object represents the previous hop; for upstream messages (e.g., Resv from receiver to sender) this object represents the next hop.

TIME_VALUES: This object contains the value of the refresh period R.

STYLE: This object defines the reservation style information that is not in the flowspec or filterspec objects. This object is required in the Resv message.

FLOWSPEC: This object defines the desired QoS in a Resv message.

FILTER_SPEC: This object defines the set of packets that receive the desired QoS in a Resv message.

SENDER_TEMPLATE: This object provides the IP address of the sender in a Path message.

SENDER_TSPEC: This object defines the sender's traffic characteristics in a Path message.

ADSPEC: This object carries end-to-end path information (OPWA)[3] in a Path message.

ERROR_SPEC: This object specifies errors in PathErr and ResvErr, or a confirmation in ResvConf.

POLICY_DATA: This object carries policy information that enables the policy module in a node to determine whether a request is allowed.

INTEGRITY: This object carries cryptographic and authentication information that is used to verify the contents of an RSVP message.

SCOPE: This object provides an explicit list of senders that are to receive this message. The object may be used in Resv, ResvErr, or ResvTear messages.

RESV_CONFIRM: This object carries the receiver IP address that is to receive the confirmation.

RSVP messages are built from a common header followed by a number of objects. For example, the format of a *Path message* is given as follows:

```
<Path message> ::= <Common Header>[<INTEGRITY>]
            <SESSION><RSVP_HOP><TIME_VALUES>[<POLICY_DATA> ...]
            [<sender descriptor>]
<sender descriptor> ::= <SENDER_TEMPLATE><SENDER_TSPEC>[<ADSPEC>]
```

Another important example is the Resv message, which is given as follows:

```
<Resv message> ::= <Common Header>[<INTEGRITY>]
            <SESSION><RSVP_HOP><TIME_VALUES>
            [<RESV_CONFIRM>][<SCOPE>][<POLICY_DATA> ...]
            <STYLE><flow descriptor list>
```

The flow descriptor list depends on the reservation styles. For wildcard-filter (WF) style, the list is

```
<flow descriptor list> ::= <WF flow descriptor>
   <WF flow descriptor> ::= <FLOWSPEC>
```

[3]OPWA stands for one pass with advertising. It refers to a reservation model in which downstream messages gather advertisement information that the receiver(s) can use to learn about the end-to-end service.

For fixed-filter (FF) style, the list is given by

```
<flow descriptor list> ::= <FLOWSPEC><FILTER_SPEC> |
            <flow descriptor list><FF flow descriptor>
   <FF flow descriptor> ::= [<FLOWSPEC>] <FILTER_SPEC>
```

For shared-explicit (SE) style, the flow descriptor list is given by

```
<flow descriptor> ::= <SE flow descriptor>
   <SE flow descriptor> ::= <FLOWSPEC><filter spec list>
      <filter spec list> ::= <FILTER_SPEC> |
            <filter spec list><FILTER_SPEC>
```

10.3 DIFFERENTIATED SERVICES

The intserv model was a first step toward providing QoS in the Internet. However, the intserv model requires a router to keep a flow-specific state for each flow that the router is maintaining. This requirement raises some concerns. First, the amount of state information increases proportionally with the number of flows. Thus routers may need large storage spaces and high processing power. Second, the intserv model may make the routers much more complex, since they need to implement the RSVP protocol, admission control, packet classifier, and especially a per flow packet-scheduling algorithm. Because of the scalability and complexity issues associated with the intserv model, IETF has introduced another service model called the **differentiated services (DS)** or **(diffserv) model**, which is intended to be simpler and more scalable. Scalability is achieved in two ways. First, per flow service is replaced with per aggregate service. Second, complex processing is moved from the core of a network to the edge.

Unlike the intserv model, which requires an application to make a resource reservation for each flow, the diffserv model aggregates the entire customer's requirement for QoS. A customer or organization wishing to receive differentiated services must first have a **service level agreement (SLA)** with its service provider. An SLA is a service contract between a customer and a service provider that specifies the forwarding service that the customer will receive. An SLA includes a **traffic conditioning agreement (TCA)**, which gives detailed service parameters such as service level, traffic profile, marking, and shaping. An SLA can be static or dynamic. Static SLAs are negotiated on a relatively long-term basis (e.g., weekly or monthly) between the humans representing the customer and the provider. Dynamic SLAs change more frequently, and thus must use a protocol such as a "bandwidth broker" to effect SLA changes.

If a customer wishes to receive different service levels for different packets, it needs to mark its packets by assigning specific values in the type-of-service (TOS) field (renamed to the DS field) in accordance with the requested service treatments.[4] Packet marking at the customer premises may be done at a host or at a customer's router as shown in Figure 10.12. In the diffserv model, different values of DS field correspond to different packet forwarding treatments at each router, called the **per-hop**

[4]Please refer to Figure 8.4 for the placement of the TOS/DS field within the IP header.

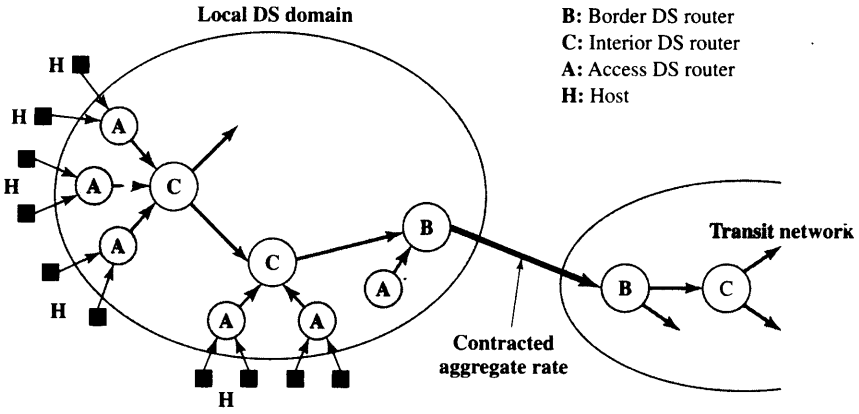

FIGURE 10.12 Router types in differentiated services.

behaviors (PHBs). Unlike the intserv model whereby the router reserves some resources for each flow, the router in the diffserv model only allocates resources on an aggregate basis for each PHB.

To ensure that the traffic entering the service provider's network conforms to the rules specified in the TCA, the ingress router (entry border router) in the provider's network needs to support traffic classification and traffic conditioning. A **traffic classifier** performs traffic classification by matching the content of some portion of the packet header with some pre-defined sets so that packets can be directed to appropriate data paths (e.g., appropriate buffers) inside a router to receive appropriate treatments. The content may come from the DS field. It may also come from multiple fields consisting of IP source address, IP destination address, protocol ID number, source port number, and destination port number. A **traffic conditioner** performs traffic conditioning by combining elements such as metering, marking, shaping, and dropping. Detailed elements of a traffic conditioner are described in Section 10.3.3.

10.3.1 DS Field

A diffserv-capable router in the interior of a network relies on the value of the DS field of each packet and uses buffer management and scheduling mechanisms to deliver the specific PHB. The value of the DS field is set at network boundaries.

The DS field is intended to supersede the existing definitions of the IPv4 TOS octet and the IPv6 traffic class octet. As shown in Figure 10.13, six bits of the DS field are used as a *differentiated services codepoint (DSCP)* to indicate the PHB a packet should experience at each node. The other two bits in the octet are currently unused (CU) and should be ignored by a router.

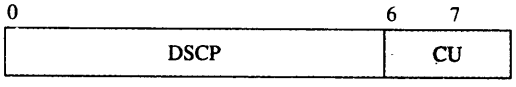

FIGURE 10.13 The structure of the DS field.

A diffserv-capable router uses the entire six-bit DSCP field as a table index to a particular packet-handling mechanism. Although implementations are encouraged to use the recommended DSCP-to-PHB mappings, network operators may choose a different DSCP for a given PHB. In such a case, the operator may need to re-mark the codepoint at the administrative boundary of its network.

To maintain some form of backwards compatibility with current TOS practice, certain codepoints are reserved. In particular, a default PHB codepoint of 000000 remains for use for conventional best-effort traffic, and the codepoint 11x000 is for network control traffic. If a packet is received with an unrecognized codepoint, the packet will be forwarded as if it were marked with the default PHB codepoint.

10.3.2 Per-Hop Behaviors

The standard best-effort treatment that routers perform when forwarding traffic is defined as the default (DE) PHB. To provide additional classes of service, the IETF has defined additional PHBs. So far two additional PHBs have been defined: the expedited forwarding PHB and assured forwarding PHB.

Expedited Forwarding PHB (EF PHB) provides a low-loss, low-latency, low-jitter, assured-bandwidth, end-to-end service through DS domains. To the end hosts, the service received by using EF PHB is equivalent to a "virtual leased line." Such service is often called a "premium service."

To ensure very low latency and assured bandwidth, the aggregate arrival rate of packets with EF PHB at every router should be less than the aggregate minimum allowed departure rate. Thus every router must be configured with a minimum departure rate for EF PHB independent of other traffic. In addition, the aggregate arrival rate must be shaped and policed so that it is always less than the minimum configured departure rate. The observant reader would notice that the service obtained using EF PHB is essentially equivalent to CBR service in ATM.

Packets that are marked as EF PHB are encoded with codepoint 101110. When EF packets enter a router, they are placed in a queue that is expected to be short and served quickly so that EF traffic maintains significantly lower levels of latency, packet loss, and jitter. Several types of queue-scheduling mechanisms such as priority queueing and weighted fair queueing may be adopted to implement the EF PHB. Note that the diffserv model specifies the PHB but not the mechanism to provide it.

Assured Forwarding PHB (AF PHB) delivers the aggregate traffic from a particular customer with high assurance (i.e., high probability of the traffic being delivered to the destination) as long as the aggregate traffic does not exceed the traffic profile (e.g., the subscribed information rate). The customer, however, is allowed to send its traffic beyond the traffic profile with the caveat that the excess traffic may not be given high assurance. Unlike EF PHB, AF PHB is not intended for low-latency, low-jitter applications.

Four independent AF classes in the AF PHB group have been defined to offer several levels of forwarding assurances. Within each AF class, packets are assigned to one of three possible drop-precedence values. If there is congestion in a router, the drop precedence of a packet determines the relative importance of the packet within the AF class.

A router must maintain the sequence of IP packets of the same microflow (that is, application flow) belonging to the same AF class regardless of the drop precedence values.

An IP packet associated with AF class i and drop precedence j is marked with AF codepoint AFij. The recommended values of the AFij codepoints are as follows: AF11 = 001010, AF12 = 001100, AF13 = 001110, AF21 = 010010, AF22 = 010100, AF23 = 010110, AF31 = 011010, AF32 = 011100, AF33 = 011110, AF41 = 100010, AF42 = 100100, AF43 = 100110. Within each AF class, the codepoint AFx1 yields lower loss probability than the codepoint AFx2, which in turn yields lower loss probability than the codepoint AFx3.

One service that the AF PHB group can implement is the olympic service, which consists of three service classes: bronze, silver, and gold. The objective is to assign packets to these three classes so that packets in the gold class experience lighter load than packets in the silver class and in the bronze class. The bronze, silver, and gold service classes can be mapped to the AF classes 1, 2, and 3, respectively. Packets within each class may be further mapped to one of the three drop-precedence values.

RED AND RIO BUFFER MANAGEMENT

The different packet drop precedence levels in differentiated services IP indicate that packets are to receive different priorities in accessing buffers during periods of congestion. In Chapter 7 we presented buffer management techniques based on queue-size allocation and queue-threshold admission policies, including Random Early Detection (RED) that can be used with packet flows generated by sources that can adapt to congestion. The prime example of such flows are packets generated by TCP sources.

[Clark and Fang 1998] proposed an extension of RED to provide different levels of drop precedence for two classes of traffic. The algorithm is called RED with IN/OUT or RIO for short. Packets are classified as being inside (IN) or outside (OUT) depending on whether they conform to some profile. Two average queue lengths are maintained: Q_{IN} the average number of IN packets, and Q_T the average number of total (IN + OUT) packets in the buffer. IN packets are dropped according to a RED algorithm that is based on Q_{IN} and corresponding packet-dropping thresholds. OUT packets are dropped according to Q_T and corresponding thresholds. The thresholds and packet-dropping probabilities are selected so that OUT packets are dropped more aggressively than IN packets. Consequently as congestion sets in, OUT packets will be more likely to be dropped. RIO and RIO-like algorithms can therefore be used to provide different levels of packet-drop precedence.

10.3.3 Traffic Conditioner

Traffic conditioning is an essential function of a diffserv-capable router. This section describes the elements that can be combined to perform traffic conditioning. These

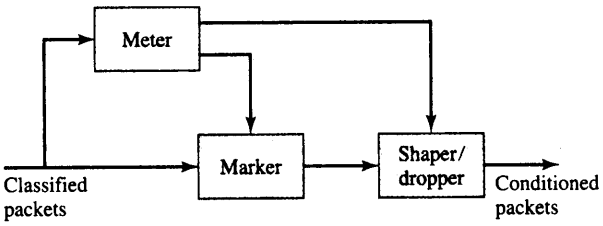

FIGURE 10.14 The functional diagram of a traffic conditioner.

elements include meters, markers, shapers, and droppers. Figure 10.14 shows the block diagram of a traffic conditioner.

A border router uses a traffic classifier to identify the service class that should be given to the traffic. Once the traffic is classified, it is typically submitted to a meter. The **meter** measures the traffic to check conformance to a traffic profile, such as the rate and burst size, and provides inputs to other elements. There are various types of meters. The most common one is the token bucket meter that can be used to check conformance against peak rate, average rate, maximum burst size, and other traffic parameters.

A **marker** sets the DSCP in a packet header. Markers may perform marking on unmarked packets (DSCP = 000000) or may re-mark previously marked packets. Traffic that is deemed to be nonconforming may be re-marked to a lower service level. Marking can also be provided by the service provider as a value-added service. For example, the service provider may mark the customer's traffic based on certain multifield classification.

A **shaper** delays packets so that the traffic at the output of a shaper is compliant with the traffic profile. Ingress routers may shape the incoming traffic that is deemed noncompliant to protect the DS domain. Egress routers may shape the outgoing traffic before it is forwarded to a different provider's network. This type of shaper ensures that the traffic will conform to the policing action in the subsequent network.

A **dropper** discards traffic that violates its traffic profile. A dropper may be thought of as a shaper with zero buffer size.

10.3.4 Bandwidth Broker

In Sections 10.3.2 and 10.3.3 we have introduced several service classes and mechanisms for providing differentiated services. The problem remains of allocating and controlling the bandwidth within a diffserv domain so that the objectives of an organization are met. One possible approach is to have users individually decide which service to use, but this approach is unlikely to achieve the desired objectives. Another approach is to have an agent for each domain, called a **bandwidth broker**, that tracks the current allocation of traffic to various services and that handles new requests for service according to organizational policies and the current state of traffic allocation [Nichols 97].

A bandwidth broker manages the bandwidth in a domain in several ways. It is responsible not only for the allocation of traffic to the various service classes within the domain but also for the setting up of packet classifiers and meters in the edge routers.

The bandwidth broker maintains a policy database that specifies which users are allowed to request what services at what time. The bandwidth broker first authenticates each requester and then decides whether there is sufficient bandwidth to meet the particular service request.

The bandwidth broker also maintains bilateral agreements with bandwidth brokers in neighboring domains. For flows that request service to a destination in a different domain, the bandwidth broker checks to see that the requested flow conforms to the prearranged allocation through the appropriate next-hop domain. The bandwidth broker then informs the appropriate neighboring bandwidth broker of the new rate allocation and configures the appropriate border router to handle the new flow. The neighboring bandwidth broker then configures its border router to handle the allocated packet flow. This bilateral agreement approach is viewed as a viable means of reaching agreement on the exchange of diffserv traffic across multiple domains.

It is anticipated that, at least initially, static preallocation of bandwidth in bilateral agreements will predominate. Nevertheless, bandwidth brokers are viewed as capable of handling the range of possible operating points up to dynamic per flow set up of bilateral agreements.

10.4 NETWORK INTERCONNECTION MODELS

An end-to-end communication path between two hosts often traverses multiple networks with different technologies. Figure 10.15 shows an example where a host connected to network 1 implemented with a given technology communicates with another host connected to network 3 with the same technology. These two networks in turn use the service of network 2 that implements a different technology. Networks 1 and 3 may be called the *client networks* and network 2 the *server network*, since network 2 provides communication services to the other networks. Also, note that the concept of

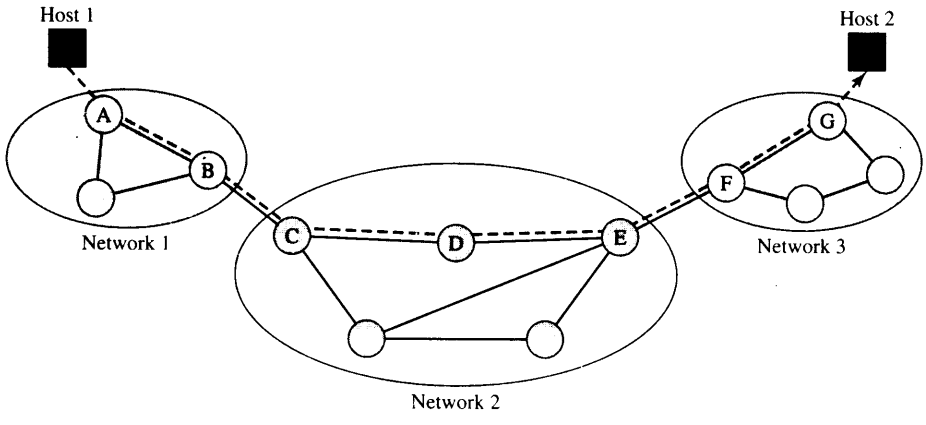

FIGURE 10.15 Interconnection of networks with different technologies.

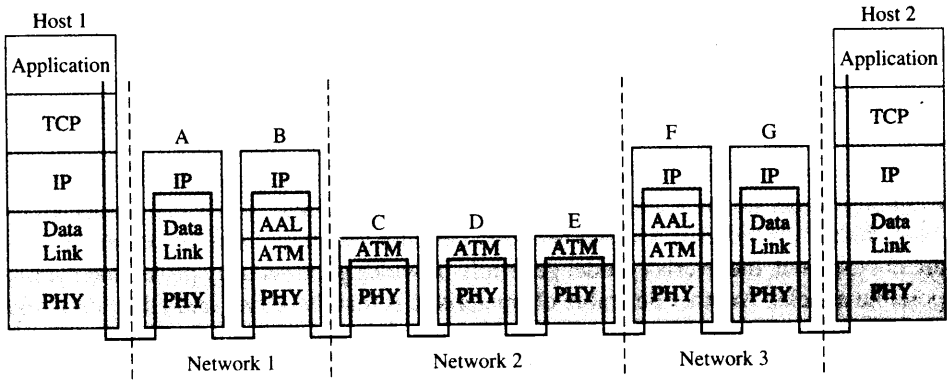

FIGURE 10.16 End-to-end protocol stack in the data plane.

client/server network can be recursive in the sense that a server network can further be a client to another server network. Figure 10.16 illustrates an example where the client networks may be IP networks while the server network may be an ATM network. The figure also shows how data from host 1 is processed at each layer in each node as the data flow traverses toward host 2.

The situation illustrated in Figure 10.15 arises in different forms. For example, nodes B and F could contain Packet-over-SONET (POS) interfaces that connect to a network 2 that provides SONET transport connections. Another important example has nodes B and F map IP streams onto optical wavelengths that traverse an all-optical network 2.

An important consideration in interconnecting networks of different technologies lies in the control-plane (i.e., routing and signaling) protocols. The issue is how the paths from the client networks across the server network are selected and who is responsible for the selection. If the networks run the same control-plane protocols (e.g., OSPF and RSVP) and allow transparent distribution of control-plane information, the interworking problem usually becomes simpler. However, this approach may not be appropriate in certain cases. For example, the provider of the server network may not want to expose the details of its own network to the client networks for security or other reasons. Another example preventing networks from running the same control plane is that the server network may use a different addressing scheme from the client networks. The following sections introduce two common approaches for interconnecting networks of different technologies.

10.4.1 Overlay Model

In the **overlay model,** the server network operates an independent control plane from the client networks. The client and server networks interact through a user-network

interface much as telephones interact with the telephone network in setting up connections. The overlay model allows the server network to prevent the control-plane information (e.g., routing information) from propagating to the client networks, so that the internal details of the server network are hidden from the client networks. As far as the client networks are concerned, the client networks would receive the service from the server network in the form of logical links connecting their border nodes; for example, border nodes B and F of the client networks in Figure 10.16 are connected by a logical link provided by a server network. Because the internals of the server network are not exposed, the overlay model generally provides good security protection against possible attacks. Another desirable characteristic is that the server and client networks can be developed independently. The overlay model is common when different service providers operate the server and client networks.

EXAMPLE **IP over ATM with MPOA**

Multiprotocol Over ATM (MPOA) is a solution designed to overlay an IP network over an ATM network. Figure 10.17 shows an example of an MPOA network configuration where the client IP networks are connected to the server ATM network via edge devices (EDs) (the figure only shows two EDs for simplicity). The ED contains an *MPOA Client (MPC)*, which is primarily used to source and sink ATM VCCs (shortcut paths) in the server ATM network and to interface with the client IP network. The ATM network also contains *MPOA Servers (MPSs)* that perform IP-to-ATM address resolution mapping and forward IP packets if needed.

Initially, packets from host 1 to host 2 would follow the default path from MPC1 (ingress MPC) through each router residing in the MPS, and finally to MPC2 (egress MPC), as shown in Figure 10.17. When the ingress MPC detects a long-lived flow, that MPC will try to establish a shortcut VCC to the egress MPC.[5] A flow is deemed to be long-lived when a certain number of packets of the same flow have passed through the ingress MPC in a certain time interval. Unfortunately, the ingress MPC does not know which egress MPC is the correct exit point for this particular flow as the flow is identified by an IP address while the egress MPC is identified by an ATM address. To find the egress MPC, the ingress MPC first performs an ATM address discovery of the corresponding egress MPC by sending through the default path an *address resolution request* first to MPS1 and subsequent MPSs until the request reaches the MPS associated with the egress MPC (in this case, MPS3). Because MPS3 knows that the flow exits through MPC2, MPS3 can then send an *address resolution reply* containing the ATM address of MPC2 (the egress MPC) back to the ingress MPC. Once the ingress MPC discovers the ATM address of the egress MPC, the ingress MPC can setup a VCC (or shortcut path) to the egress MPC and transfers subsequent packets through the VCC. The VCC can be automatically torn down through aging if the VCC is inactive after a certain amount of time.

[5]The shortcut is viewed as desirable because it reduces the traffic load that the routers handle.

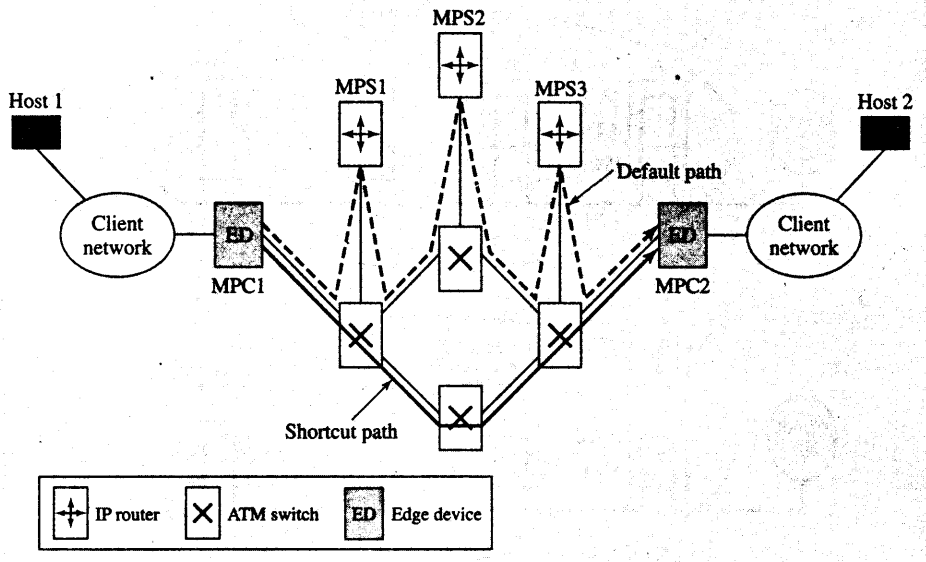

FIGURE 10.17 Example of network topology.

10.4.2 Peer-to-Peer Model

The **peer model** uses a single instance of the control plane for the server and client networks. As a result, the client network knows the topology of the server network, and the client network can directly determine an end-to-end connection through the server network. One advantage of the peer model is that it does not require an address resolution protocol to interwork address spaces, and thus simplifies address administration. However, the peer model may introduce some security issues to the server network. Moreover, the interdependence between the client and server networks makes it difficult to develop one network without affecting the other network.

EXAMPLE **IP+ATM**

In IP+ATM networks, ATM switching and IP routing are combined in a node. In one approach, IP routers initially forward new packets of a particular flow hop-by-hop using the destination-based lookup. As shown in Figure 10.18a, initially cells with a default VCI x are reassembled into packets and forwarded by the IP layer. When a predetermined number of packets of the same flow have been received, each node may decide to bypass its IP layer by notifying its upstream node to use another free VCI for the flow. For example, node A notifies the upstream client IP network to use VCI y_1 for the subsequent cells of the same flow. When node A receives a similar notification from B

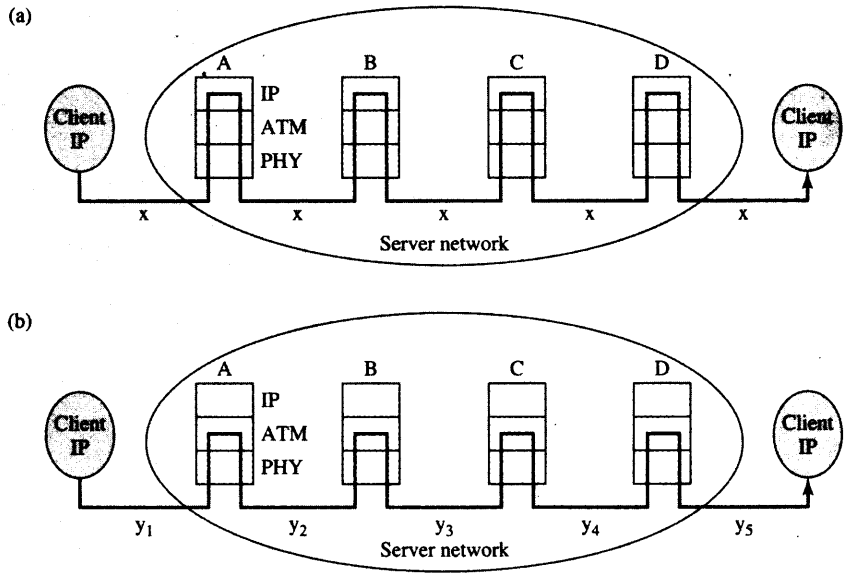

FIGURE 10.18 IP+ATM with (a) hop-by-hop forwarding at the IP layer, and (b) cut-through forwarding at the ATM layer.

to use VCI y_2, node A configures its ATM connection table so cells with incoming VCI y_1 can be directly switched to node B with outgoing VCI y_2 without going though an IP forwarding layer. When other nodes along the path perform the same process, all subsequent packets between the client IP networks will be forwarded through an ATM connection (also called a cut-through path) from node A to node D, as shown in Figure 10.18b. After the ATM connection is established, each node examines the flow periodically. If the flow is inactive after a certain time out, the connection is deleted and subsequent packets will be forwarded through IP and the process for re-establishing the cut-through path repeats.

The issue of overlay versus peer models in network interconnection typically flares up with the emergence of each new service network technology. The reasonable approach is to use the overlay model first and to contemplate the peer model in the long term. Most recently the issue of overlay versus peer models has arisen in the context of interconnecting IP client networks over next-generation SONET networks and optical networks. Unlike the case of ATM networks that had its own established routing and addressing scheme, SONET and optical networks are more amenable to using extensions of IP-based protocols such as OSPF and RSVP and so the long-term chances of some deployment of the peer model approach are more reasonable.

10.5 MPLS

IP routers use destination-based forwarding (i.e., lookup based on the IP destination address) to determine the next hop of a packet. The longest-prefix match required in IP address lookup was traditionally implemented in software and viewed as too slow for core networks. Although recent advances in IP address lookup techniques and hardware implementations have allowed destination-based packet forwarding to perform at higher speed, lookup techniques based on simpler forwarding information provide certain advantages as embodied in Multiprotocol Label Switching (MPLS).

The label-switching paradigm in ATM performs lookup based on a short and fixed-length label (i.e., VPI/VCI) to determine the corresponding next hop for each cell. Label switching uses a label to directly index into a connection table entry to determine the next hop, lending itself to a simple hardware lookup implementation capable of a very high forwarding rate. Label-switching forwarding is also considered more attractive than destination-based forwarding because of its flexibility. While the path taken by a packet is determined solely by the destination address of a packet with destination-based forwarding, label switching allows different streams of packets with the same destination to follow different predetermined routes. Thus label switching has been considered a better candidate for facilitating traffic engineering (that is, mapping packet flows to the network resources efficiently).

MPLS was developed out of a number of proposals for transporting IP over ATM networks. In the 1990s the availability of standardized ATM switches prompted some Internet service providers (ISPs) to adopt label switching in their IP-over-ATM overlay networks so that the configuration of the VCC routes could effectively map the traffic flows to the network resources. The overlay model, however, has the drawback that two network infrastructures need to be managed separately, each with its own addressing, routing, and management systems. Consequently, several approaches to integrating IP and ATM were proposed. One early proposal, called *IP switching* developed by a company called Ipsilon, uses the model described in the IP+ATM example in the preceding section. IP switching establishes and tears down ATM cut-through paths for carrying IP flows based on flow activities, and is said to follow a *traffic-driven label assignment* approach because a label is assigned when a new traffic flow appears.

The Cell Switch Router (CSR) proposed by Toshiba is a similar approach to providing cut-through paths. Besides using the traffic-driven label assignment approach, the CSR also allows a new label to be assigned when a new IP forwarding table entry (or a new IP route) is found. This approach is naturally called *topology-driven*, since IP routing determines the assignment of labels. For example, when a node receives new routing information and adds the corresponding new route to its routing table, the node triggers the assignment of a label associated with the new route. Finally, the CSR also allows a new label to be assigned by means of a specific request (called *request-driven label assignment*). The request is usually initiated by a signaling protocol, which instructs each node along the path to assign a new label entry in the forwarding table. The signaling protocol typically allows the connection to follow a specific explicit route.

Cisco's tag switching is a proposal that was initially designed to operate over a variety of data-link technologies. To work on different data links, tag switching

proposes a new *tag* (or shim header) that is inserted between the link-layer header and the network-layer header. The label information, among other relevant information, is carried in the tag. If the data-link layer is ATM, the label information may be carried in VPI/VCI fields and thus the tag is not used. Another feature of tag switching includes support of multiple levels of connection hierarchy by stacking multiple tags. This feature essentially generalizes the two levels of connection hierarchy in ATM (that is, by using VCI and VPI) to multiple levels. Finally, tag switching includes topology-driven and request-driven label assignment.

Another notable proposal, developed by IBM, is called Aggregate Route-Based IP Switching (ARIS). In the ARIS proposal, connections are pre-established toward each destination. Because connections from multiple sources to a given destination typically form a tree, ARIS allows further optimization of label usage whereby multiple incoming connections can be merged into a single outgoing connection. The benefit of merging is further explained below.

Although differences exist among these various proposals, the label-switching paradigm shares a common principle where forwarding and control components in a node are separated, as shown in Figure 10.19. At the early stage of the development of this technology, the forwarding component was usually handled by ATM data path while the control component by IP control protocols. Later, it became clear that the data path could be handled by many other technologies. The separation approach has the advantage of allowing each component to be developed and modified independently. We show how this separation also allows ISPs greater flexibility in defining the services they offer.

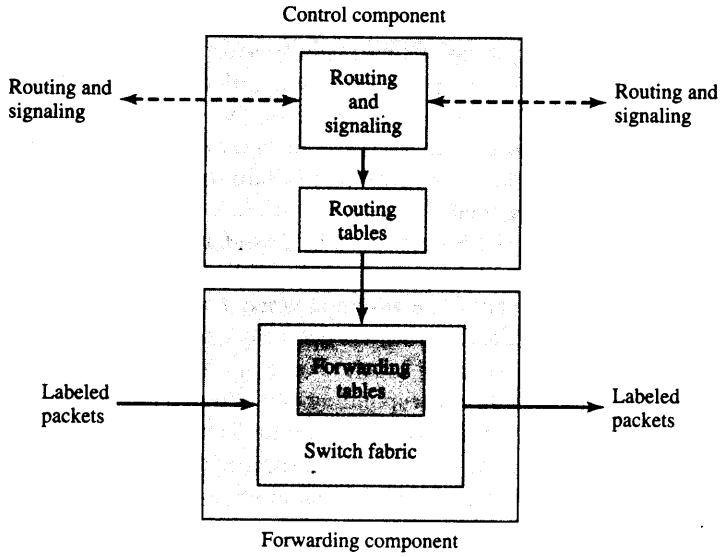

FIGURE 10.19 Separation of forwarding and control components in label-switching paradigm.

Interest in the application of label-switching paradigm to the IP core network prompted some birds-of-the-feather (BOF) meetings at the IETF in 1996, which were subsequently followed by a formation of a new working group called the **MultiProtocol Label Switching (MPLS)**. The "multiprotocol" stresses the fact that MPLS should be able to support any layer-3 protocol, although the initial focus would only be on the IPv4 and IPv6. Additionally, MPLS should be able to operate on top of a variety of layer-2 technologies such as ATM, frame relay, PPP, and Ethernet. Note that MPLS does not refer to one particular protocol, but *a set of protocols* that enable MPLS networking.

A router that supports MPLS is called a **label-switching router (LSR)**. Unlike a traditional datagram router, an LSR separates the control and forwarding components as shown in Figure 10.19. The control component runs traditional IP routing protocols and new MPLS signaling protocols. The forwarding component, which can be implemented by any layer-2 technology, performs label switching and provides fast data paths. An *MPLS domain* consists of a contiguous set of LSRs that are capable of executing standardized MPLS signaling protocols and perform routing and forwarding functions in that domain. An LSR that interfaces to a traditional router is called an *edge LSR*. With respect to the direction of the traffic flow, it is often useful to distinguish between an *ingress LSR* and an *egress LSR*. The ingress LSR receives traffic from a non-MPLS router while the egress LSR sends traffic to a non-MPLS router. It is important to note ingress and egress LSRs are defined only for a particular traffic flow. Thus an LSR in an MPLS domain can act as an ingress LSR for a particular traffic flow, an egress LSR for another traffic flow, or a plain LSR for yet another traffic flow.

10.5.1 Fundamentals of Labels

In MPLS, a *unidirectional* connection through multiple LSRs is called a **label-switched path (LSP)**. LSP generalizes the basic concept of a connection or virtual circuit through a technique called LSP merging whereby multiple incoming connections may be merged into one outgoing connection leading to a multipoint-to-point tree rooted at an egress LSR, as illustrated in Figure 10.20. Observe that the multipoint-to-point tree can be made to match the shortest-path tree described in Chapter 7. Thus the routes followed by packets toward a particular egress LSR can be made to match those in the corresponding connectionless network. The matching can be done through the topology-driven label assignment approach.

A group of packets that are forwarded in the same manner (e.g., along the same route) in an MPLS domain are said to belong to the same **forwarding equivalence class (FEC)**. In general, an FEC may have many granularities. At one end of the spectrum, an FEC could be associated with the flow of a particular application for a particular source and destination host pair. At the other end, an FEC could be associated with all the flows destined to an egress LSR. These different levels of granularity allow MPLS to be used in a wide range of situations. The assignment of a particular packet to a particular FEC is done only at an ingress LSR where the label for the packet is also created. From then on, other LSRs use the label to determine the next hop and the label is removed at an egress LSR.

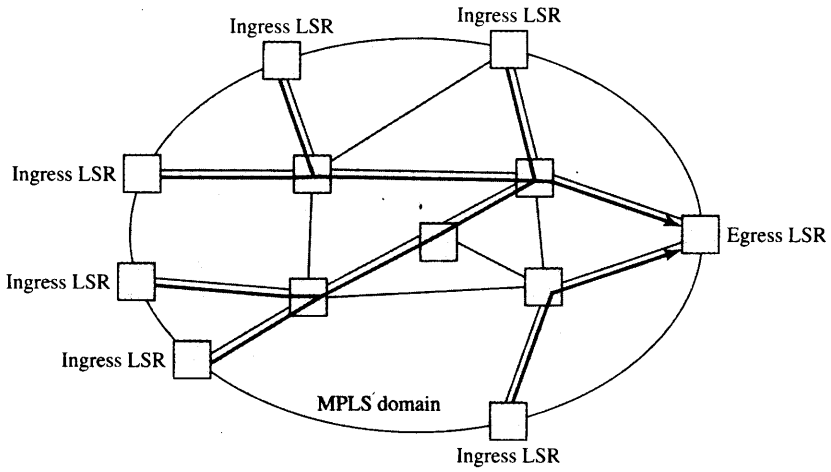

FIGURE 10.20 Multipoint-to-point tree rooted at the egress LSR.

MPLS labels are encoded in different ways depending on the layer-2 technologies. As shown in Figure 10.21a, the label is encoded in the VCI and/or VPI fields for ATM. For other layer-2 technologies that do not support label fields, such as PPP or Ethernet, an MPLS header is inserted between the layer-2 and layer-3 (IP) headers as shown in Figure 10.21b. The 32-bit MPLS header contains a 20-bit label field, a 3-bit experimental field, a 1-bit hierarchical stack (S) field, and an 8-bit time-to-live (TTL) field.

10.5.2 Label Stack and LSP Hierarchy

When a label is attached to a packet at an ingress LSR, the LSR is said to perform a *label push*. Subsequent LSRs in the MPLS domain only swap the incoming label to the outgoing label. When the packet leaves the MPLS domain, the egress LSR performs a *label pop* by removing a label.

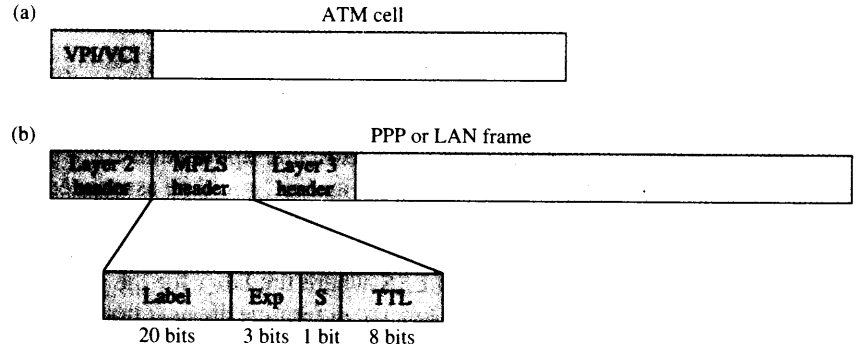

FIGURE 10.21 MPLS labels in ATM and PPP/LAN.

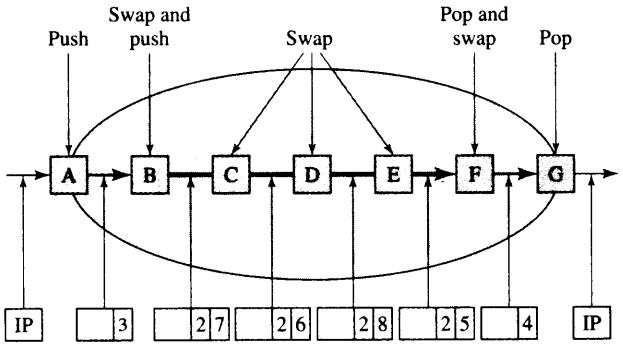

FIGURE 10.22 LSP hierarchy.

It is possible for an MPLS header to consist of a stack of m labels (or of depth m). Each label in the stack has the S bit equal to zero except the last one (i.e., the bottom of the stack) where the S bit is equal to one. In a given LSR, the label at the top of the stack is the only one that determines the forwarding decision. An unlabeled packet is a packet with an empty stack (depth 0) and is forwarded at the IP layer.

Figure 10.22 illustrates the use of label stack. LSR A receives an unlabeled packet from a traditional IP router destined for LSR G. First, A pushes on a label (3) that is agreed upon by B and forwards the packet to B. Then B replaces the incoming label with the label (2) that is agreed upon by F. B also pushes on another label (7) that is agreed upon by C and forwards the packet to C, which replaces the incoming label at the top of the stack (7) with the outgoing label (6) and forwards the packet to D. Similarly, D replaces the incoming label with the outgoing label (8), and E replaces the incoming label with the outgoing label (5). When F receives the packet, it pops the label stack and replaces the incoming label (3) that is used by B with an outgoing label (4). Finally, G pops the label stack and forwards the unlabeled packet using normal IP forwarding. In this example, we see a label stack of depth two implements two levels of **LSP hierarchy**. The first level LSP traverses through <A, B, F, G> while the second level LSP traverses through <B, C, D, E, F> *e*. The actual physical path taken by a packet is <A, B, C, D, E, F, G>. Referring to the concept of tunneling described in Section 8.3.4, we observe that the second level LSP provides a tunnel between B and F to packets from the first level LSP. Similarly, the first level LSP provides a tunnel between A and G to packets arriving at A and destined to an external network via G.

To see the benefit of LSP hierarchy, let us suppose that there are n ingress LSRs $A_1, A_2, \ldots, A_n$, connected to B, and n egress LSRs $G_1, G_2, \ldots, G_n$ connected to F. An LSP is to be established between A_i and G_i for each i, such that each LSP traverses through LSRs $<A_i$, B, C, D, E, F, and $G_i>$. Without LSP hierarchy, n LSPs would have to be established between B and F. With two-level LSP hierarchy, only one LSP is needed between B and F. Thus LSP hierarchy provides scalability advantage in terms of LSP usage. We will see that multiple levels of LSP hierarchy greater than two have applications in generalized MPLS.

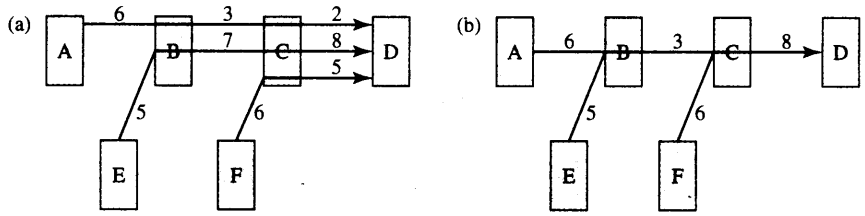

FIGURE 10.23 Label merging.

10.5.3 VC Merging

Suppose that an MPLS network needs to forward packets on three routes: <A, B, C, D>, <E, B, C, D>, and <F, C, D>. One approach is to establish three LSPs: LSP <A, B, C, D>, LSP <E, B, C, D>, and LSP <F, C, D>. Figure 10.23a shows these LSPs and the associated labels on each LSP. Another approach is to establish one multipoint-to-point LSP as shown in Figure 10.23b.

When a separate LSP needs to be established for each LSR pair in a network with N LSRs (called full-meshed LSPs), the connection table at each LSR requires $O(N^2)$ entries. This approach places heavy demand on the usage of labels at each LSR and does not scale to large networks. With multipoint-to-point LSPs, an LSR would "merge" multiple incoming labels for the traffic flows intended to the same egress LSR to the same outgoing label. For example, incoming labels 5 and 6 at LSR B in Figure 10.23 are merged to an outgoing label 3. Label merging essentially reduces the connection table entries from $O(N^2)$ to $O(N)$. This property gives MPLS a great degree of scalability and makes it appropriate for very large networks.

In ATM systems capable of performing **VC merging**, incoming VCs intended for the same egress LSR are merged to the same outgoing VC. Thus the cells intended for the same egress LSR become indistinguishable at the output of a switch, as they carry the same VC value. However, if cells belonging to different packets for the same egress LSR are interleaved, the receiver will not be able to reassemble the packets. To avoid this problem, a mechanism is needed to ensure that cells within the same packet are not interleaved with other cells intended for the same egress LSR. One mechanism is to reassemble cells into packets prior to VC merging. The "end-of-packet" bit available in the cell payload type can be used to reassemble packets with AAL5 encapsulation. Figure 10.24 shows how VC merging reshuffles the output streams to ensure that cells belonging to different packets never interleave. Packet reassembly for VC merging requires an additional amount of buffering. It turns out that VC merging incurs only a minimal buffer overhead.

10.5.4 Label Distribution Protocols

MPLS requires a protocol to distribute label bindings between LSRs. In general, label bindings between two LSRs can be distributed by either an upstream LSR or a downstream LSR. The MPLS architecture dictates that a downstream LSR distributes label

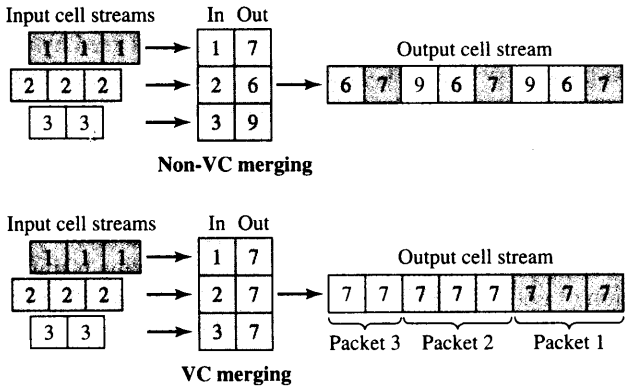

FIGURE 10.24 Output streams for VC merging and non-VC merging.

bindings to an upstream LSR. Figure 10.25 illustrates the label distribution process. Here, suppose that LSR 1 (the upstream LSR) just detects that LSR 2 (the downstream LSR) is its next hop for FEC = 10.5/16. LSR 1 then sends a label request message for FEC = 10.5/16 to LSR 2. Upon receiving the message, LSR 2 responds with a label binding message that specifies the FEC-label binding. From then on, LSR 1 can forward packets destined to 10.5/16 with label value 8 to LSR 2.

In this example, called the *downstream-on-demand* mode, LSR 2 distributes a label binding in response to an explicit request from LSR 1. Another label distribution mode, called the *unsolicited downstream* mode, allows LSR2 to distribute a label binding even if it has not been explicitly requested.

MPLS also allows LSP setups to be initiated in two ways. In *ordered control* an LSR distributes an FEC-label binding only if the LSR is the egress LSR for that FEC or the LSR has already received a label binding for that FEC from its next hop. In *independent control* each LSR independently binds a label to an FEC and distributes the binding to its peer.

Several MPLS label distribution protocols have been specified in IETF. **Label distribution protocol (LDP)** [RFC 3036] adopts the topology-driven label assignment approach. Initially, an LSR sends Hello messages over UDP periodically to its neighbors to discover potential LDP peers. When an LDP peer is discovered, the LSR tries to establish a TCP connection to its peer. Once the TCP connection is established, the two LSRs may negotiate session parameters such as label distribution option, valid label ranges, and valid timers. Successful negotiation between the two LSRs completes the establishment of an LDP session. The two LSRs then may exchange LDP messages over the LDP session. Several LDP messages are defined with some notable ones including label request, mapping, and label withdraw. Label request message is used by an LSR to request

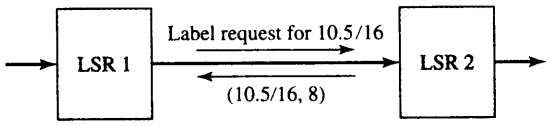

FIGURE 10.25 Label distribution.

a binding for an FEC from its peer. Label mapping message is used by an LSR to advertise FEC-label bindings to its peer. Label withdraw message is used by an LSR to indicate to its peer to stop using specific FEC-label bindings that were previously advertised.

An extension to RSVP to establish traffic-engineered LSPs has also been specified under the name **RSVP-TE** [RFC 3209]. Unlike LDP that establishes LSPs along the same routes as specified by the routing protocol, RSVP-TE adopts the request-driven label assignment approach and allows an *explicitly routed* LSP between each LSR pair. In explicit routing the route taken by a packet is determined by a single node, usually the ingress LSR. One useful application of explicit routing is traffic engineering where maximization of network resource utilization translates to effective mapping of traffic flows on the network topology. Figure 10.26a illustrates how resources may be inefficiently utilized by using hop-by-hop routing. Suppose that node 3 assumes that the shortest path to node 8 is via node 4. Thus node 3 forwards packets from nodes 1 and 2 that are destined to node 8 through node 4. This routing approach may cause some links (e.g., link 4–6) to be congested, while other links (e.g., link 5–7) are lightly loaded. As shown in Figure 10.26b, explicit routing allows node 2 to establish the explicitly routed LSP through nodes 3, 5, 7, and 8 if it determines that link 4–6 is already heavily utilized. Thus explicit routing allows for a more flexible mapping of traffic to the network topology.

RSVP-TE extends the RSVP Path message to include a label request object to request a label binding. In response to the request, the label binding is distributed upstream by extending the RSVP Resv message to include a label object. Thus RSVP-TE follows the downstream-on-demand label distribution mode. Explicit routing is implemented by including an explicit route object (ERO) in the Path message. The ERO typically specifies the list of nodes along the explicit route. RSVP-TE also allows an LSP to be associated with particular setup and holding priorities. An LSP with a higher setup priority is allowed to preempt another LSP with a lower holding priority. When a particular link does not have sufficient bandwidth for a new LSP that has a higher setup priority, an existing LSP with a lower holding priority may be torn down. The bandwidth released by the existing LSP can then be used for setting up the new LSP.

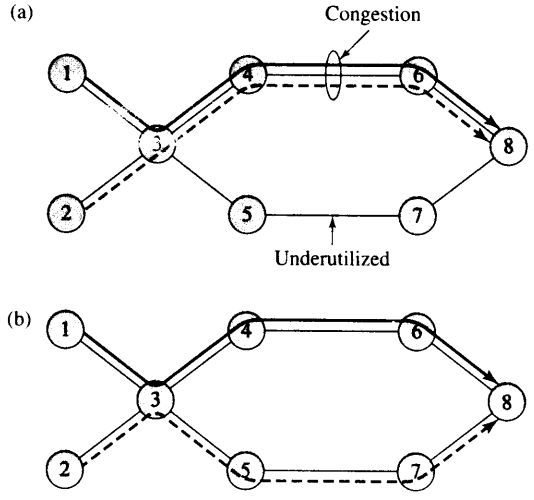

FIGURE 10.26 Traffic mapping with (a) hop-by-hop routing and (b) explicit routing.

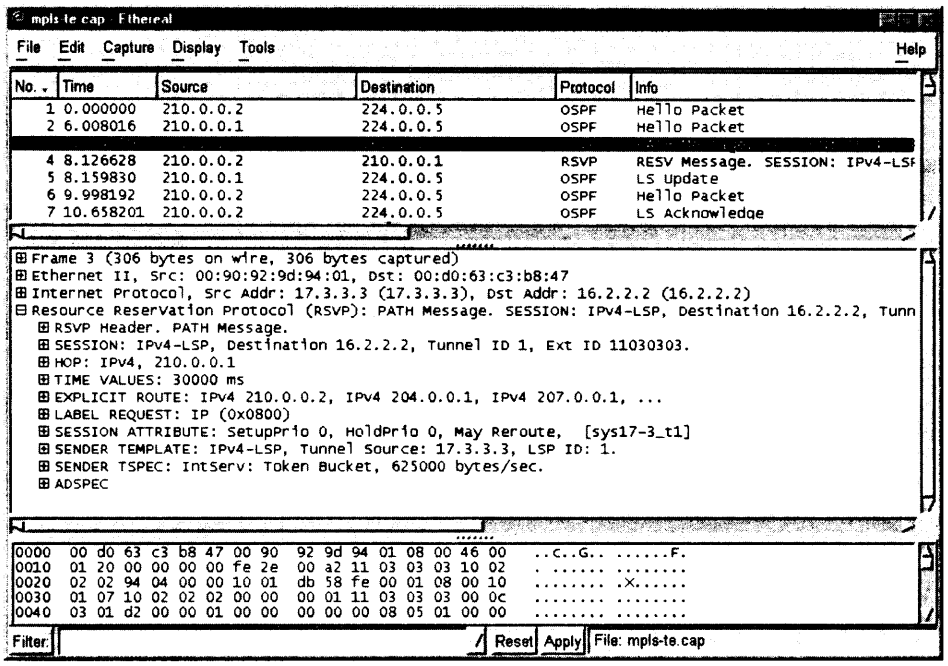

FIGURE 10.27 Example of an RSVP Path message for MPLS label request.

Figure 10.27 provides an example of an RSVP Path message containing a label request. The middle pane shows that the session involves an IPv4 LSP and the explicit route field, which contains a list of routers that are to be used in the explicit route. Figure 10.28 shows the RSVP Resv message that travels in the reverse direction of the corresponding Path message and assigns the label 16 to the preceding label request.

10.5.5 MPLS Support for Virtual Networks

MPLS allows ISPs to configure LSPs to create *virtual networks* that support particular classes of traffic flows. In principle an ISP could configure a separate LSP to carry each class of traffic between each pair of edge LSRs. A more practical solution is to merge LSPs of the same traffic class to obtain multipoint-to-point flows that are rooted at an egress LSR. The LSRs serving each of these flows would be configured to provide the desired levels of performance to each traffic class.

MPLS can also be used to create **Virtual Private Networks (VPNs).** A VPN provides wide area connectivity to an organization located in multiple sites. MPLS can provide connectivity among VPN sites through LSPs that are dedicated to the given VPN. The LSPs can be used to exchange routing information between the various VPN sites, transparently to other users of the MPLS network, thus giving the appearance of a dedicated wide area network. The scalability of the ISP network can be improved by providing hierarchical LSPs where the first level LSP is used to identify a particular VPN site and the second level LSP is used to route traffic within the ISP network. VPNs

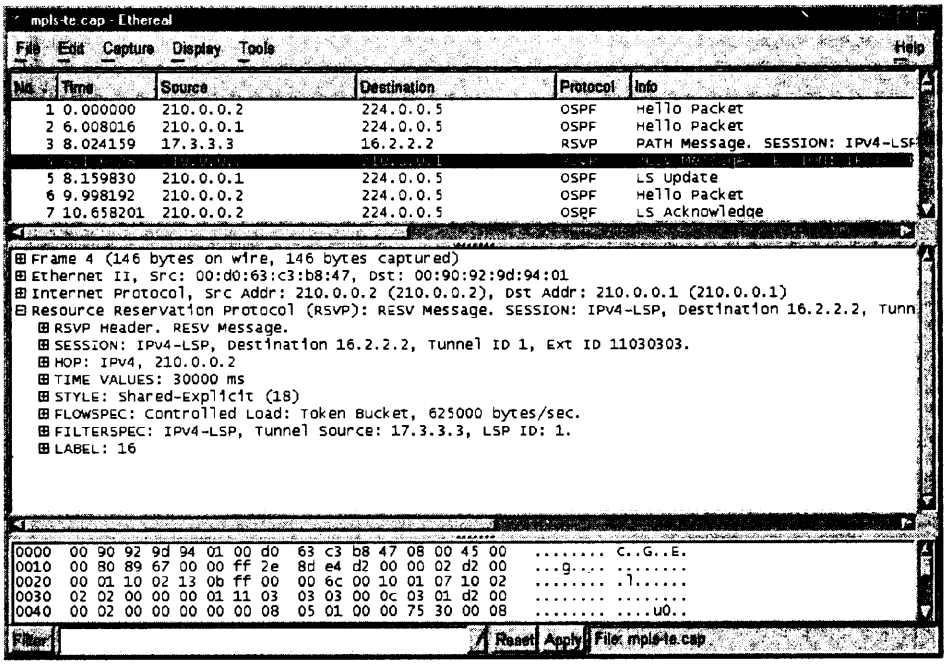

FIGURE 10.28 Example of RSVP Resv message response to label request.

involve many security issues related to ensuring privacy in the public or shared portion of a network. These issues are discussed in Chapter 11.

10.5.6 Survivability

Survivability refers to the capability of a network to maintain existing services in the face of failures. Datagram IP networks address survivability by means of dynamic routing. When a link or a node fails, dynamic routing eventually discovers the new topology and reconfigures the routing tables correctly. Dynamic routing typically restores traffic in the order of seconds or minutes depending on the convergence time of the protocol. However, as packet networks carry more mission-critical and high-priority traffic, there is increasing interest in designing a packet network, in particular an MPLS network, that can restore traffic much faster. In this sense MPLS attempts to provide a measure of the survivability afforded by SONET networks.[6]

In general, traffic can be restored upon failure by means of **restoration** or **protection** mechanisms. With restoration, new paths are established after a failure occurs and the affected traffic is then rerouted through the new paths. With protection (or protection switching), **protection paths** are used as backups for the normal paths **(working paths)** and are pre-established. When a failure occurs, the affected traffic is

[6]Protection and restoration in SONET networks is discussed in Chapter 4.

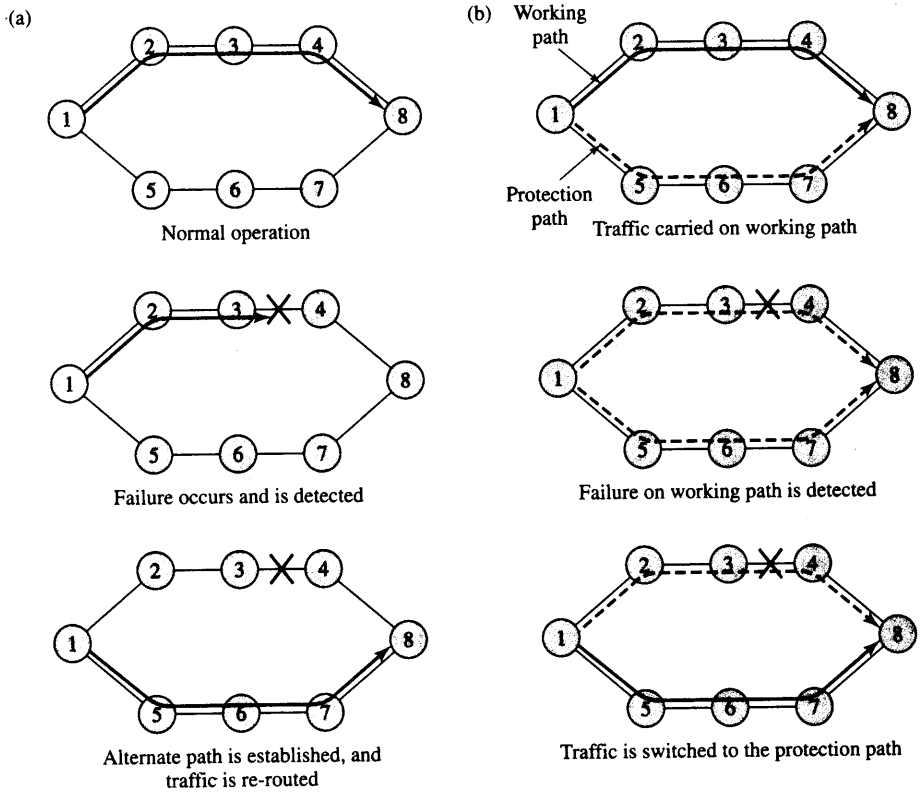

FIGURE 10.29 Survivability with (a) restoration and (b) protection.

switched to the protection paths. The working and protection paths are usually disjoint to ensure that a failure will not simulataneously affect both paths. Figure 10.29 illustrates the differences between restoration and protection. Since restoration requires a path to be established after a failure is detected, restoration typically takes longer to restore traffic than protection. On the other hand, restoration typically requires less resources (e.g., bandwidth) than protection, since the resources are only activated after a failure occurs in restoration.

Protection and restoration mechanisms can be designed with different areas of coverage and different repair methods. **Path protection/restoration** is intended to protect against any failure along the active path (the path that carries traffic). Path protection relies on **global repair** where the node that performs protection/restoration is located at a particular node (typically the ingress node) and may be located far from failure. When a failure along the active path is detected, a failure notification message needs to be forwarded to the node that can perform protection/restoration. **Local repair** relies on a local node (typically the node upstream of the failure) to perform protection/restoration and is intended to protect against a failure in a particular local area. Since a failure can be directly detected without relying on the failure notification message, local repair in general restores traffic faster than global repair. Local repair can be designed to deal

with **link protection/restoration** or **node protection/restoration**. Link protection/restoration protects a path from a given link failure while node protection/restoration protects a path from a given node failure.

Protection mechanisms may be configured in a number of options. The common options include:

- 1+1 (1 plus 1) where traffic is copied to both working and protection paths at the ingress node. The egress node selects one of the paths based on integrity of the paths (e.g., bit error rate).
- 1:1 (1 for 1) where traffic is sent to the working path during normal operation. When a failure occurs, the traffic is switched to the protection path which can be shared among a number of flows that are selected so that they are unlikely to fail at the same time.

FROM TELEGRAMS TO CIRCUITS TO DATAGRAMS TO VIRTUAL CIRCUITS TO ...

In Chapter 1 we saw that the telegraph network used message switching, a special case of datagram packet switching. The invention of analog voice transmission led to a telephone network based on the switching of circuits. The proliferation of computers in the 1980s triggered extensive research and development into packet-switching technologies of two basic forms: datagram and virtual circuit. The simplicity and robustness of IP led to the growth of the datagram-based Internet. ATM based on virtual-circuit packet switching introduced low-cost high-performance switching, and in MPLS, we see the use of a virtual-circuit approach as a means of optimizing the core IP network.

We have witnessed the evolution of communication networks from packet switching to circuit switching and then to packet switching again. We have also witnessed the tension between virtual-circuit and datagram packet switching and interestingly their apparent coalescence. Thus we see that the balance of the prevailing technologies becomes manifest in network architectures that can favor one extreme or the other or even a blend of these.

Vinton Cerf once made the remark, "Today: you go through a circuit switch to get to a packet switch. Tomorrow: you go through a packet switch to get to a circuit switch." Hmmm ... what about the day after tomorrow?

10.5.7 GMPLS

In a transport network, a node typically forwards traffic based on time slots, wavelengths, or physical ports. For example, forwarding in a TDM switch may constitute the transfer of information from slot i in an incoming TDM frame to slot j in an outgoing frame. Although the specific forwarding mechanism in a transport network is different from that in a packet-based MPLS network, observe that these different types

of networks are fundamentally connection oriented. Therefore, signaling protocols that are developed for setting up and tearing down connections in a packet network can also be applied to a transport network.

Generalized MPLS (GMPLS) has been proposed to reuse much of the MPLS control component to select paths and to set up and tear down different types of connections. GMPLS allows a node with multiple switching technologies (e.g., packet switching, TDM switching, and possibly optical switching) to be controlled by a unified control component. This approach promises to reduce the cost of the network through the integration of the network control and management infrastructure.

In GMPLS, an LSR may have different types of interfaces: packet-switch capable (PSC) interface, time-division multiplex (TDM) interface, lambda switch capable (LSC) interface, fiber-switch capable (FSC) interface. Thus the definition of an MPLS label needs to be generalized so that a label can also be encoded as a time slot, a wavelength, or a spatial identifier (e.g., a port). This new definition allows the same MPLS control component to configure a TDM switch, a lambda cross-connect, or a fiber cross-connect.

The concept of LSP hierarchy in MPLS can be generalized to include LSPs at different levels to have different types of LSP (e.g., MPLS virtual circuit, SONET TDM circuit, and optical lightpath), as illustrated in Figure 10.30. This generalization allows multiple virtual circuits to be tunneled over a TDM circuit. In turn, multiple TDM circuits can be tunneled over a lightpath (i.e., wavelength connection). Tunneling in a GMPLS-based network generally follows a certain hierarchy based on the multiplexing capability of the LSP type. For example, it generally does not make sense to tunnel TDM circuits over a virtual circuit.

The generalization of the concept of an LSP in GMPLS is extremely powerful because the signaling and routing protocols of MPLS can then be generalized for use in this much broader context. The MPLS signaling capabilities (e.g., RSVP-TE) can be used to convey label requests and label objects along an explicit path. Extensions to these protocols are required to support not only virtual circuits, but also TDM circuits, lightpaths, and fibers.

In MPLS, routing protocols such as OSPF are used to disseminate information about the state of links in the network that are useful in the selection of paths that meet

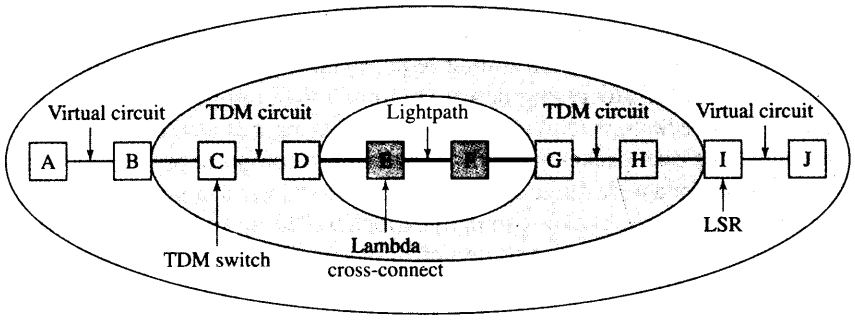

FIGURE 10.30 Hierarchical LSPs of different types.

certain requirements. Extensions to these routing protocols in GMPLS can play a very important role in disseminating information about the state and capabilities of optical and TDM links in the network. For example, OSPF LSAs may be extended to carry information about a link such as signal quality, protection capability, whether the link is part of a bundle, and the type of interfaces available to terminate the connection.

Optical networks using dense wavelength division multiplexing (DWDM) are likely to create many parallel links between two adjacent nodes. Current IP routing protocols such as OSPF establish a routing adjacency over each link and flood the link state information for each adjacency throughout the network. **Link bundling** has been introduced to cope with the scalability problem of IP routing. The basic idea is to aggregate these parallel links (called *component links*) into a single *bundled link*. Adjacency is now maintained over the bundled link and thus routing scalability improves.

When two adjacent nodes are connected by many component links, the manual configuration process of each link can be laborious and prone to human errors. The **Link Management Protocol (LMP)** is a new protocol that can be used to automate the association and verification of the component links. LMP decouples the control channel from the data channels so that a transparent data channel such as a lightpath can also be supported.

LMP provides four basic functions: control channel management, link connectivity verification, link property correlation, and fault isolation. Control channel management establishes and maintains connectivity between adjacent nodes. Link connectivity verification ensures the correct associations of all component links. Link property correlation ensures that the link IDs, protection modes, and other link properties between the adjacent nodes are properly correlated. Finally, fault isolation can isolate link and channel failures independent of the data format used by the channel.

10.6 REAL-TIME TRANSPORT PROTOCOL

Traditional real-time communications has taken place over circuit-switched networks that can provide low transfer delay for a steady stream of information. Telephone calls and low-bit rate videoconferencing are the primary examples of this type of communications. Packet networks were developed primarily for the transport of data information that did not have such stringent timing requirements. Advances in compression algorithms and computer processing power combined with improvements in transmission bandwidth and packet-switching capability are making it possible to support real-time communications over packet networks. Advances in computer processing in particular make it feasible for a wide range of media types to coexist in a multiplicity of multimedia applications. Real-time packet communications would make it possible to exploit the ubiquity of the Internet. Packet communications would also bring capabilities such as multicasting that are not easily provided by circuit switched networks.

Real-time packet communications, however, must deal with impairments inherent in packet networks. This includes packet delay and jitter, packet loss, and delivery of out-of-sequence or duplicate packets (see Section 5.3.2). The **Real-Time Transport**

Protocol (RTP) [RFC 1889] provides end-to-end transport functions for applications that require real-time transmission, such as audio or video over unicast or multicast network services. RTP services include: payload type identification, sequence numbering, timestamping, and delivery monitoring. RTP normally runs on top of UDP, but it can also run over other suitable networks or transport protocols. RTP does not provide any means to ensure the timely delivery of information or quality-of-service guarantees. For this it must depend on the services of the underlying network layer such as diffserv IP. Instead, RTP provides the mechanisms for dealing with impairments such as jitter, and loss, as well as for timing recovery and intermedia synchronization.

The **RTP Control Protocol (RTCP)** is a companion protocol for monitoring the quality of service observed at a receiver and for conveying this and other information about participants to the sender. This capability is especially useful in situations where the sender can adapt its algorithm to the prevailing network conditions (e.g., available bandwidth or network delay/jitter).

RTP is intentionally not a complete protocol and is intended to be sufficiently flexible that it can be incorporated into the application processing, instead of being implemented as a separate layer. The use of RTP in a particular application therefore requires one or more companion documents. A *profile* specification document defines attributes and/or modifications and extensions to RTP for a class of applications (e.g., audio and video). A *payload format* document in turn defines how a particular audio or video encoding is to be carried over RTP (e.g., MPEG2 video or ADPCM audio). [RFC 1890] specifies an initial set of payload types. [RFC 2250] describes a packetization scheme for MPEG video and audio streams.

The RTP/RTCP protocols are in wide use in the Internet supporting audio and video streaming applications as well as Internet telephony and other real-time applications. Table 10.1 lists the Payload Type assignment numbers that are used to specify the type of encoding in RTP. Additional payload type numbers can be defined dynamically through non-RTP means such as SIP signaling discussed later in the chapter.[7]

10.6.1 RTP Scenarios and Terminology

Let us introduce some terminology using first an audioconference example and then an audiovisual conference example.

An *RTP session* is an association among a group of participants communicating using RTP. Suppose for now that the RTP session involves an audioconference. The chair of the conference makes arrangements to obtain an IP multicast group address and a pair of consecutive UDP port numbers which identify the RTP session. The first port number (even) is for RTP audio and the other port number (odd) is for the corresponding RTCP stream. The address and port information is distributed (by some means beyond the scope of RTP) to the participants.

Once the audioconference is underway, each participant can send fixed-duration blocks of audio information as payloads in an RTP PDU, which in turn is incorporated in

[7] In Chapter 12 we discuss many of the encoding schemes listed in Table 10.1.

TABLE 10.1 RTP Payload Type Numbers.

PT	Encoding name	Audio/Video (A/V)	Clock rate	Channels
0	PCMU	A	8000	1
2	G726-32	A	8000	1
3	GSM	A	8000	1
4	G723	A	8000	1
7	LPC	A	8000	1
8	PCMA	A	8000	1
9	G722	A	8000	1
10	L16	A	44100	2
11	L16	A	44100	1
12	QCELP	A	8000	1
13	CN	A	8000	1
14	MPA	A	90000	
15	G728	A	8000	1
18	G729	A	8000	1
dyn	GSM-HR	A	8000	1
dyn	GSM-EFR	A	8000	1
dyn	L8	A	var.	var.
26	JPEG	V	90000	
28	nv	V	90000	
31	H261	V	90000	
32	MPV	V	90000	
33	MP2T	AV	90000	
34	H263	V	90000	
96–127	dynamic	?		
dyn	BT656	V	90000	
dyn	H263-1998	V	90000	
dyn	MP1S	V	90000	
dyn	MP2P	V	90000	
dyn	BMPEG	V	90000	

a UDP datagram. The RTP header specifies the type of audio encoding. It also includes sequence and timestamp information that can be used to deal with packet loss and reordering and to reconstruct the encoder clock. Each source of a stream of RTP packets is identified by a 32-bit *Synchronization Source (SSRC)* ID that is carried in the RTP header. All packets for a given SSRC use the same timing and sequence number space in order to allow receivers to regroup and resynchronize the given packet sequence.

Each RTP periodically multicasts a receiver report on the RTCP port. The report provides an indication of how well the RTP packets are being received. The report also identifies who is participating in the conference. Upon leaving the conference, each site transmits an RTCP BYE packet.

Now consider the case of an audiovisual conference. Typically, *each media is transmitted using a separate RTP session*, that is, using separate multicast and UDP port pair addresses. The audio and video RTP sessions are treated as completely separate except that their association is indicated by a unique name that is carried in their RTCP packets. This allows the synchronized playback of the audio and video streams.

There are a number of reasons for using a separate RTP session for each media stream. Different QoS or discard treatments can be provided for each stream (e.g., under

congestion video packets are discarded and audio packets kept). Some users may choose or only be capable of receiving audio and not video. In the case of layered coding, different terminals may choose to receive a different set of layered video streams either because of their processing capability or the available bandwidth in their direction.

RTP allows for the use of devices called *mixers*. A mixer is an intermediate system that receives RTP packets from one or more sources, possibly changes the data format, and combines the packet in some manner and then forwards new RTP packets. For example, a mixer may combine several audio streams into a single stream. The mixer usually needs to make timing adjustments among the streams so it generates new timing for the packet sequence it produces. All the packets thus generated will have the mixer SSRC identified as their synchronization source. The mixer inserts in each RTP packet header a *Contributing Source (CSRC)* list of the sources that contributed to the combined stream.

RTP also allows for devices called *translators* which relay RTP packets with the SSRC left intact. Translators include devices that convert format without mixing, replicators from multicast to unicast, and application-level filters in firewalls.

10.6.2 RTP Packet Format

Figure 10.31 shows the packet header format for RTP. The first three rows (12 bytes) are found in every packet. The fourth row (CSRC) is used when a mixer has handled the information in the payload.

The RTP packet fields are used as follows:

Version (V): This 2-bit field identifies the version of RTP. The current version is two.

Padding (P): This 1-bit field is used to indicate that the packet contains one or more additional padding bytes that are not part of the payload. The last byte of the padding contains a count of how many padding bytes are to be ignored, including itself.

Extension (X): When the 1-bit extension field is set, the fixed header must be followed by exactly one header extension.

CSRC count (CC): This 4-bit number specifies the number of CSRC identifiers that follow the fixed header.

Marker (M): This 1-bit is defined by a profile, and it is intended to mark significant events such as frame boundaries in the packet stream.

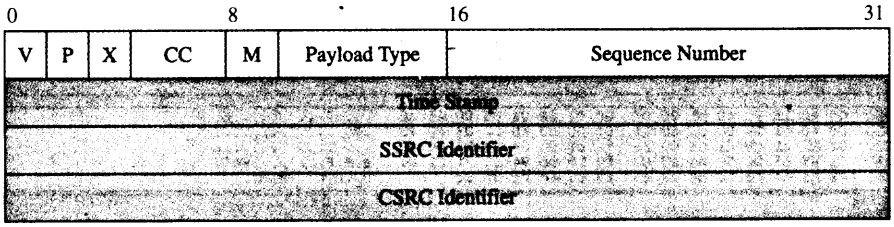

FIGURE 10.31 RTP packet header format.

Payload type (PT): This 7-bit field identifies the format of the RTP payload and determines its interpretation by the application.

Sequence number: This 16-bit field is incremented by one each time an RTP packet is sent. The number can be used by the receiver to detect packet loss and to recover packet sequence. The initial value is selected at random.

Timestamp: This 32-bit number specifies the sampling instant of the first byte in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time, so that the number can be used for synchronization and jitter calculations. The initial value is selected at random.

SSRC: The randomly chosen number is used to distinguish synchronization sources within the same RTP session. It indicates where the data was combined, or the source of the data if there is only one source.

CSRC list: This list of 0 to 15 32-bit items specifies the contributing sources for the payload contained in the packet. The number of identifiers is given by the CC field.

Figure 10.32 shows an example of an RTP packet in a voice call using Microsoft® Netmeeting®. The details of the RTP packet in the middle pane show that the call uses ITU G.723 voice coding which operates at around 6 kbps. From the clock rate column in Table 10.1 it can be seen that timestamps in the RTP packet are generated using an

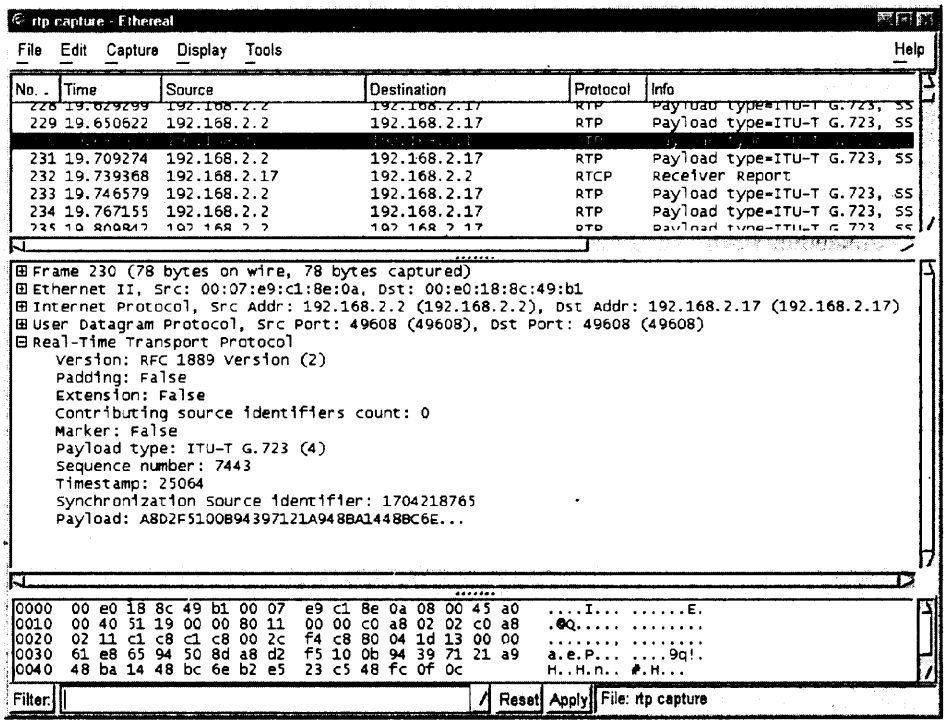

FIGURE 10.32 Example of an RTP packet.

8 kHz clock. Note that the Synchronization Source (SSRC) is given by 1704218765. We will return to this example shortly.

10.6.3 RTP Control Protocol (RTCP)

RTCP involves the periodic transmission of control packets to all participants in the session. The primary function of RTCP is to provide feedback on the quality of the data distribution, which can be used for control of adaptive encodings or to diagnose faults in the distribution. This feedback information is sent in the form of RTCP sender and receiver reports.

RTCP also carries a persistent transport-level identifier called the Canonical name (CNAME) that is used to keep track of each participant. For example, CNAME can be given by user@host or by a host in single user systems. CNAME is used to associate multiple RTP sessions, for example, to synchronize audio and video in related RTP sessions to achieve lip synch.

Since all participants are required to send RTCP packets, a mechanism has been developed to control the rate at which RTCP packets are transmitted. Each participant is able to independently determine the number of participants from the RTCP packets it receives. This information is used to adjust the RTCP transmission intervals.

RTCP can optionally provide minimal session control information such as participant identification. A higher-level session control protocol is needed to provide all the support required by a given application. Such a protocol is beyond the scope of RTCP.

RTCP defines several types of packets to carry different types of control information:

Sender Report (SR): The SR is used to distribute transmission and reception statistics from active senders.

Receiver Report (RR): The RR is used to distribute reception statistics from participants that are not active senders.

Source Description (SDES): SDES provide source description items such as, CNAME, e-mail, name, phone number, location, application tool/version, etc.

BYE: This message indicates the end of participation by the sender.

APP: Application-specific functions that are defined in profile specifications.

The SR packets provide a sender report and several reception reports. The sender report information includes: a "wall clock" time given by a Network Time Protocol timestamp which is seconds elapsed since 0 hour January 1, 1900; and the same time instant as the NTP timestamp but using the clock used to produce the RTP timestamps. This correspondence can then be used for intra- and intermedia synchronization. The sender report also gives the number of RTP packets transmitted as well as the total number of payload bytes transmitted by the sender since starting transmission. Each reception report provides statistics on a single synchronization source: fraction of RTP data packets lost since previous SR or RR packet was sent; cumulative number of RTP data packets lost since beginning of reception; extended highest sequence number received; interarrival jitter; last SR timestamp, and delay since last SR. The RR packets are the same as SR packets except that a sender report is not included.

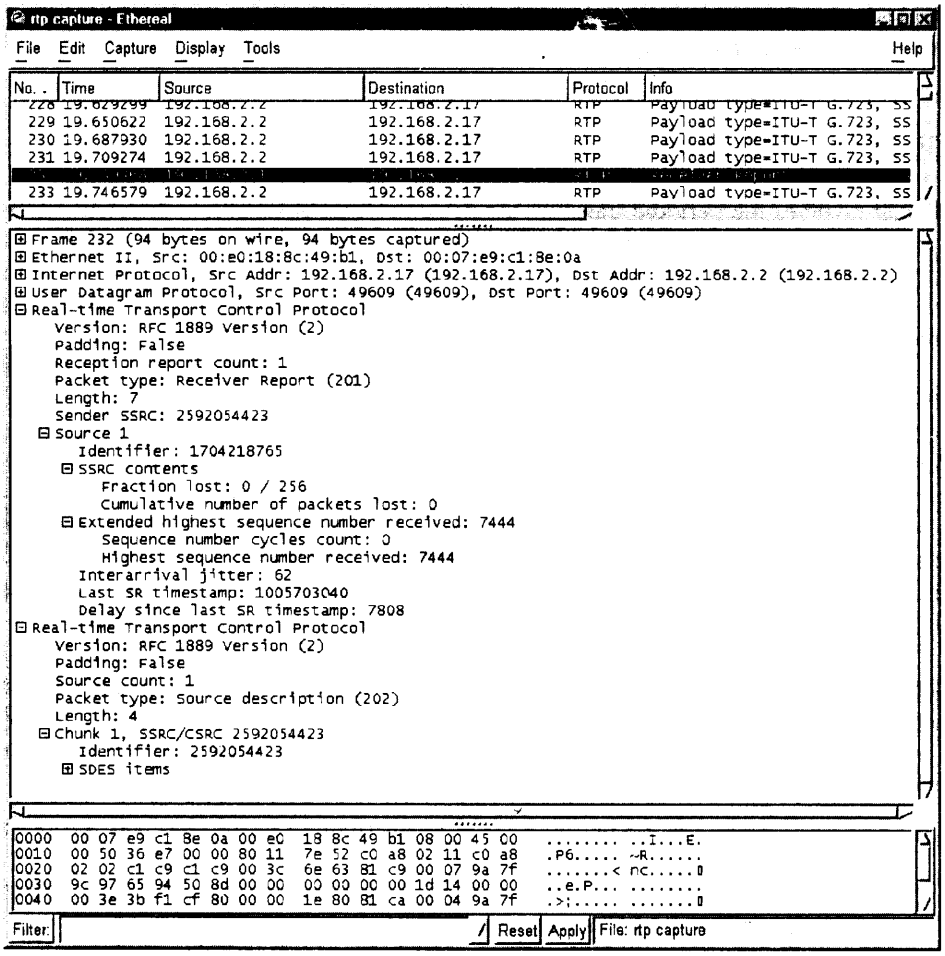

FIGURE 10.33 Example of an RTCP Receiver Report packet.

An example of an RTCP Receiver Report packet is shown in Figure 10.33. The Receiver Report is for the RTP voice call session introduced in the example of Figure 10.32. The middle pane shows that the packet type number (201) identifies the packet as a receiver report. Note that Source 1 in the report has SSRC number 1704218765, which corresponds to the source in the previous example. The receiver report provides information on the total number of packets received and the number of lost packets. It also provides information on the sequence number count and on the jitter of the packet interarrival times. The SDES in the middle pane, if expanded, would show the sender's name and other information.

RTCP was designed to provide minimal control functionality. In particular RTCP does not provide explicit membership control and session setup. The intent is that a separate session control protocol would provide this functionality. In the next section we introduce several protocols and recommendations that can provide this functionality.

10.7 SESSION CONTROL PROTOCOLS

A *session* is an association involving the exchange of data between two or more Internet end systems. Session control refers to the various functions that are involved in the setting up, maintaining, and termination of a session. As an example of a session, consider a multimedia videoconference meeting.[8] Before the meeting can be set up, the time and participants in the session need to be specified. Technical details such as media encoding, transport protocols, addresses and port numbers also need to be specified. The session needs to be announced, that is, participants and relevant servers need to receive the description of the session. At the appropriate time, the session needs to be initiated and maintained. In the case of the videoconference, conference control is also required. Each of these functions (description, announcement, initiation, and conference control) involves a different set of protocols. The IETF is developing protocols that provide all of these functions: Session Description Protocol (SDP), Session Announcement Protocol (SAP), Session Initiation Protocol (SIP), and Simple Conference Control Protocol (SSCP). The Real-Time Streaming Protocol (RTSP) has been developed to control the playback of streaming media. In this section, we focus on the IETF's Session Initiation Protocol and ITU's H.323 protocols, which are key protocols for the deployment of multimedia applications.

10.7.1 Session Initiation Protocol

The **Session Initiation Protocol (SIP)** [RFC 2543] is an application-layer control protocol that can be used to establish, modify, and terminate multimedia sessions or calls with one or more participants. These multimedia sessions can be an Internet telephony call, a multimedia videoconference, a distance learning session, or multimedia distribution. SIP and extensions of SIP can also be used for instant messaging and event notification (i.e., voicemail notification, stock notification, and callback notification). SIP can be used to manage other types of sessions, such as distributed games.

The participants in the SIP session can be people or various types of media devices (e.g., media servers). The participants in a SIP session can communicate via multicast or via a mesh of unicast connections. SIP uses invitation messages to set up a session, and these messages include session descriptions. SIP provides support for user mobility by enabling users and resources to be located using location-independent names. SIP applications dynamically register their current location when they are launched. SIP is designed to be independent of lower-layer transport protocols, so SIP can operate over UDP or over TCP.

SIP is similar to HTTP in that it is a text-based client/server protocol with syntax very similar to HTTP. A *transaction* consists of the issuing of a request by a client and the returning of one or more responses by one or more servers. Basic signaling functions are implemented with one or more transactions. As in the case of HTTP, each SIP request invokes a *method* in a server. SIP provides six methods: INVITE,

[8] A more interesting example might be a distributed game.

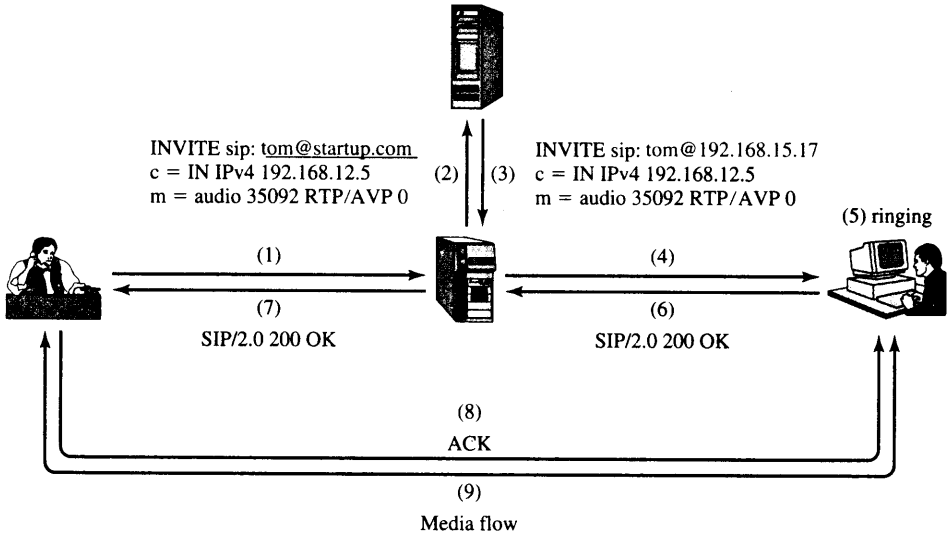

FIGURE 10.34 SIP message exchange.

ACK, OPTIONS, BYE, CANCEL, REGISTER. INVITE and ACK are the most basic methods and are used to initiate calls.

Consider the example in Figure 10.34 where Steve calls Tom. Steve is at his desk and he has a phone that uses SIP to set up connections. After Steve has indicated that he wants to call Tom, the SIP client in Steve's phone prepares an INVITE message which resembles an HTTP request message. The message includes an identifier for TOM, sip: tom@startup.com, as well as information about the media encoding, the protocol that is to be used (UDP or TCP), and the port numbers. The client sends the INVITE message to the IP address and port corresponding to Tom's identifier if it has the information. If it does not know the address, the client sends the message to a local SIP proxy server as shown in step 1 in Figure 10.34 using port 5060. (IP addresses are assigned dynamically by DHCP so typically the INVITE message will usually go to the proxy server.)

The responsibility of the SIP proxy server is to route the INVITE message to the IP address of the device that Tom is currently using. As shown in step 2 in the figure, the proxy server may need to consult Tom's SIP registrar. Whenever Tom launches a SIP application, the application sends a registration message with the current IP address to Tom's registrar. In step 3 the registrar responds to the proxy server with the IP address where Tom can be reached currently, say his workstation. The proxy server then issues a SIP INVITE request with the precise IP address to Tom's workstation (step 4). The user agent server in the workstation alerts Tom of the incoming call (step 5) and returns a success indication (OK 200) to the proxy server (step 6). The proxy server forwards the success indication to Steve's phone (step 7). Finally Steve's SIP client confirms receipt of the response message by sending an ACK message to Tom's device, either directly or through the proxy server (step 8). The call setup is now completed. In step 9

Tom's and Steve's devices can begin exchanging voice information directly. Note that the path followed by the INVITE message may be completely different than that used in the exchange of voice information.

The above example illustrates that SIP system has two components: user agents and network servers. The *user agent* is software in an end system that acts on behalf of a human user. The user agent has two parts: a protocol client, called *User Agent Client (UAC)*, is used to initiate a call; a protocol server, called *User Agent Server (UAS)*, is used to answer a call. Together the UAC and UAS allow for peer-to-peer operation using a client/server protocol.

The function of the network servers is to do the call routing to establish a call, that is, to find the desired user in the network. The network servers can be of two types: proxy and redirect. A *proxy server* receives a request, determines which server to send it to, and then forwards the request. Several servers may be traversed as a request flows from UAC to UAS. The eventual response traverses the same set of servers but in the reverse direction. It is quite possible for a user to have several SIP applications active at a given time. SIP allows a proxy server to *fork* a request and forward it simultaneously to several next-hop servers. Each of these branches can issue a response, so SIP provides rules for merging the returning responses. A *redirect server* does not forward a request, and instead it returns a message to the client with the address of the appropriate next-hop server.

Let us examine more closely how a session is set up. To establish a call, an INVITE request is sent to the UAS of the desired user. In general, the IP address or hostname of the desired user is not known. As a result, this information must be obtained from a name, such as e-mail address, telephone number, or other identifier. Typically the first INVITE request message has the following form:

```
INVITE sip:tom@startup.com SIP/2.0
Via SIP/2.0/UDP 168.192.12.5
From: sip:steve@startup.com
To: sip:tom@startup.com
Call-ID: 12345@startup.com
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 885

c=IN IPv4 192.168.12.5
m=audio 35092 RTP/AVP 0
```

The INVITE request contains the callee's SIP address followed by the SIP version. When the message is originated and each time the SIP message passes through a SIP device, the IP address of the device is attached using a Via header. The Via headers indicate the path taken by a request so far and ensures that responses take the same path as the request. The SIP message also contains a From and a To header. A Call-ID provides a unique identifier for a particular invitation. Every request message has a command sequence (CSeq) header that contains the request method and a single decimal sequence number. The Content-Type header specifies the format that is used in

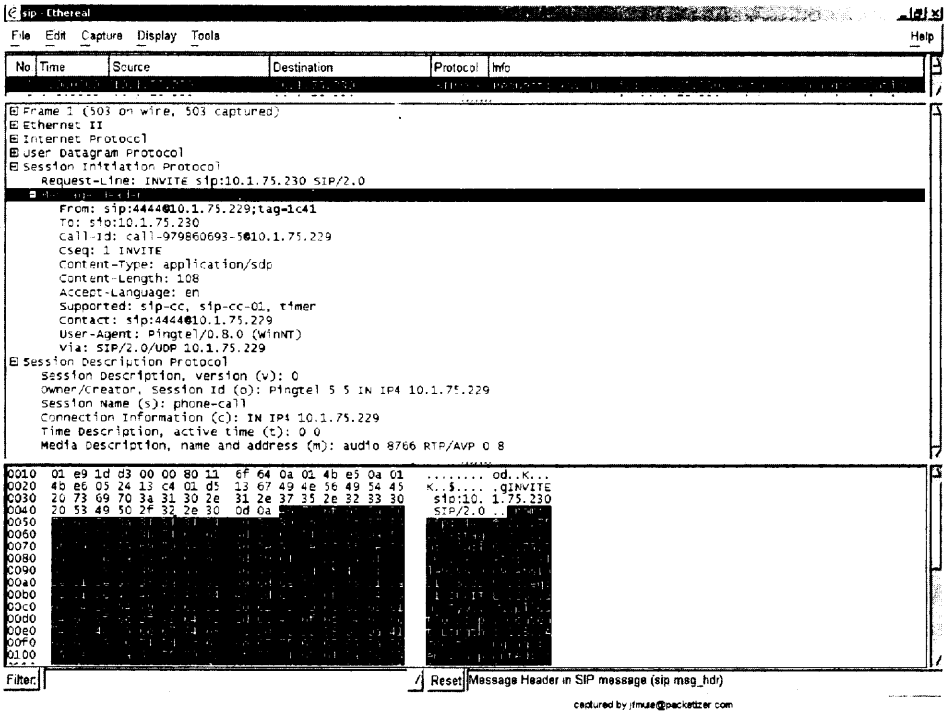

FIGURE 10.35 Example of SIP INVITE request.

the body of the message. The default is application/sdp which indicates that the *Session Description Protocol* format is used to provide information about the number and types of media streams in the session. The Content-Length gives the size of the message body in octets. The content of the message follows after a carriage return and a line feed. In the above example, the content gives the IP address of Steve's phone and specifies that an audio media encoding using mu-law PCM on UDP port 35092 is requested. Figure 10.35 shows an example of a SIP INVITE request packet.

The UAC sends the INVITE request to an appropriate network server, which in turn may proxy or direct the call to other servers until a server is found that knows the IP address of the desired user. All responses to a request contain the same values in the Call-ID, CSeq, To, and From fields in order to enable responses to be matched with requests. The response to an INVITE request from the UAS contains a reach address that the UAC can use to send further transactions *directly* to the UAS. Consequently the *SIP network servers do not need to maintain call state*. The response to the INVITE also provides the information about the media content for the callee.

The REGISTER method is used to send location information to a SIP server. For example, a user can send REGISTER to help a server map an incoming address into an outgoing address that can reach the user or a proxy that know how to reach the user. A typical example involves a user sending a REGISTER message to its usual SIP server giving it a temporary forwarding address.

BYE is used to terminate a connection between two users. OPTIONS is used to request information about the capabilities of a callee without setting up a call. CANCEL is used to terminate a pending request.

INTERNET TELEPHONY AND IP TELEPHONY

The terms *Internet telephony* and *IP telephony* are used to distinguish between two approaches to providing telephone service over IP that are different in a subtle but fundamental way. Internet telephony is used to describe telephone service using the classical Internet approach where the service control is in the end user's system (e.g., the PC). The SIP protocol provides the means for providing the end systems with telephone service control. This approach is characterized by smart and powerful end systems.

IP telephony is used to describe telephone service over IP in which service control is provided by intelligence inside the network. This follows the traditional telephone network setting where the terminal (e.g., telephone) is a very low-cost low-functionality device. The actual setting up of a connection across the network is done by switches that the terminal equipment attaches to.

The interworking of telephone service that spans the Internet and the telephone network is addressed by the introduction of gateways, as provided by H.323 and related approaches.

10.7.2 H.323 Multimedia Communication Systems

ITU-T Recommendation H.323 consists of a set of standards to provide support for real-time multimedia communications on LANs and packet networks that do not provide QoS guarantees. H.323 evolved out of the H.320 videoconferencing standards for ISDN. H.323 terminals and equipment can carry voice, video, data, or any combination of these. H.323 addresses call control, multimedia management, bandwidth management, and interfaces to other networks. In particular it provides a means for interworking telephone-based and IP-based conferencing.

As shown in Figure 10.36, an H.323 network involves several components. In addition to H.323 terminals, the network involves gateways, gatekeepers, and multipoint control units.

The gateways provide interworking between H.323 terminals in the packet network and other terminal types (e.g., telephone sets in a conventional telephone network). The gateway is responsible for the translation between audio and video codec formats. For example, a speech signal may be encoded using ITU-T G.729 8 kbps compression with RTP framing in the packet network and 64 kbps PCM in a telephone network. The gateway is responsible for the translation between the two media formats.

The gateway is also responsible for the mapping of signaling messages from the packet side of the network to other networks. In particular, in telephony applications the gateway performs call setup between the packet network and the public telephone

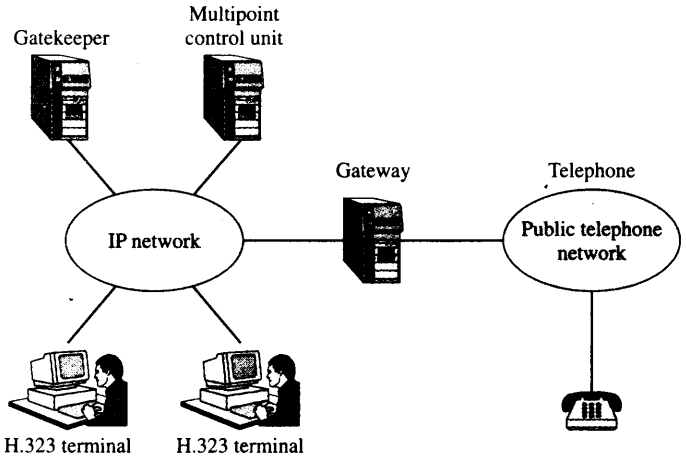

FIGURE 10.36 Components of H.323 network and interworking with a telephone network.

network. In particular, the gateways terminate H.323 signaling on the packet network side and usually ISDN signaling on the telephone network side. Part of the call setup involves establishing a path for the call across the gateway.

The gatekeepers are responsible for control of calls within an H.323 network. Gatekeepers grant permission or deny requests for connections. They manage the bandwidth that can be used by calls. Gatekeepers perform name-to-address translation, and they direct calls to appropriate gateways when necessary.

H.323 terminals in a multipoint conference can send audio and video directly to other terminals using multicasting. They can also use multipoint control units that can combine incoming audio streams and video streams and transmit the resulting streams to all terminals.

Figure 10.37 shows the scope of an H.323 terminal. H.225 specifies the call control procedures that are to be used for setting up H.323 calls in the H.323 network. The procedures use a subset of the Q.931 messages that are used in conventional ISUP signaling.[9] Terminals are addressed using either IP addresses or names (e-mail address, telephone number) that can be mapped to an IP address. H.225 also stipulates that RTP/RTCP is to be used in the packetization of the audio and video streams.

The H.245 control channel is a reliable channel (operating over TCP) that carries the control messages to set up logical channels, including the exchange of transmit and receive capabilities. The RAS control deals with registration, admission control, bandwidth management between endpoints and gatekeepers.

H.323 specifies audio and video codecs that are supported. All H.323 terminals are required to support the G.711 voice standard for log-PCM voice compression.

[9]ISUP signaling was discussed in Chapter 4.

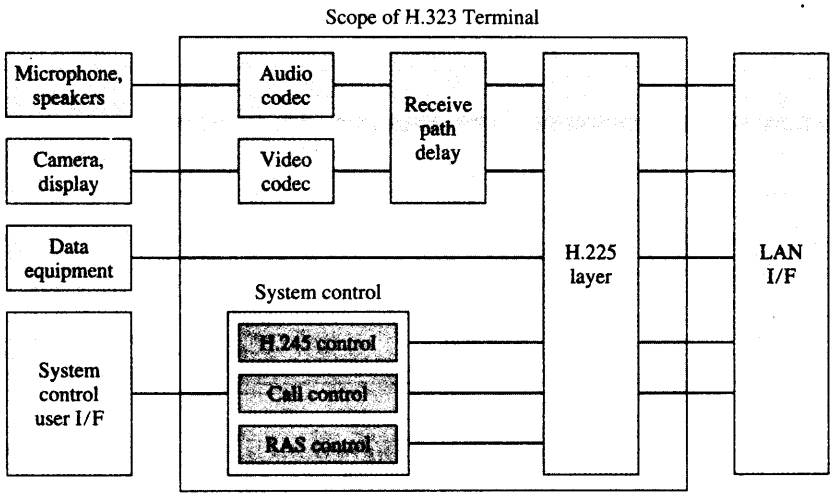

FIGURE 10.37 An H.323 terminal.

Additional audio codecs are also specified. Video is optional in H.323 terminals. QCIF H.261 video is the baseline video mode.

Consider once again the Microsoft® Netmeeting® example introduced in Section 10.6.2 in the discussion of RTP and RTCP. Figure 10.38 shows the exchange of messages that transpired in the setup of the voice call using H.323. Initially H.225 messages are used to set up the call: Frame 15 shows the setup message from 192.168.2.2 to 192.168.2.17, followed by an Alerting message response (frame 18), and completed with a Connect message (frame 22). The middle pane shows the type of information that is contained in the setup message. H.245 terminal capability messages are exchanged next to negotiate the type of media and encodings that are to be used in the session (frame 28, 29, ...). The message exchange is also used to determine that one terminal is to act as master and the other as slave in order to avoid potential conflicts that may arise. The master terminal then initiates the opening of a logical channel in one direction, and the slave follows by opening a channel in the opposite directions. The flow of RTP packets can then begin. When the session is complete, H.245 and H.225 messages are exchanged to terminate the session. Figure 10.38 only shows the first two packets in the H.245 exchange.

10.7.3 Media Gateway Control Protocols

The H.323 recommendation was developed assuming relatively powerful end systems attached to the packet network. The signaling and processing requirements are too complex for simple terminal equipment such as telephones. This has led to proposals for approaches that allow simple terminal equipment to connect to the Internet and provide telephone service. The approaches have two aspects. The first aspect involves the introduction of a *residential gateway* that is interposed between a telephone and

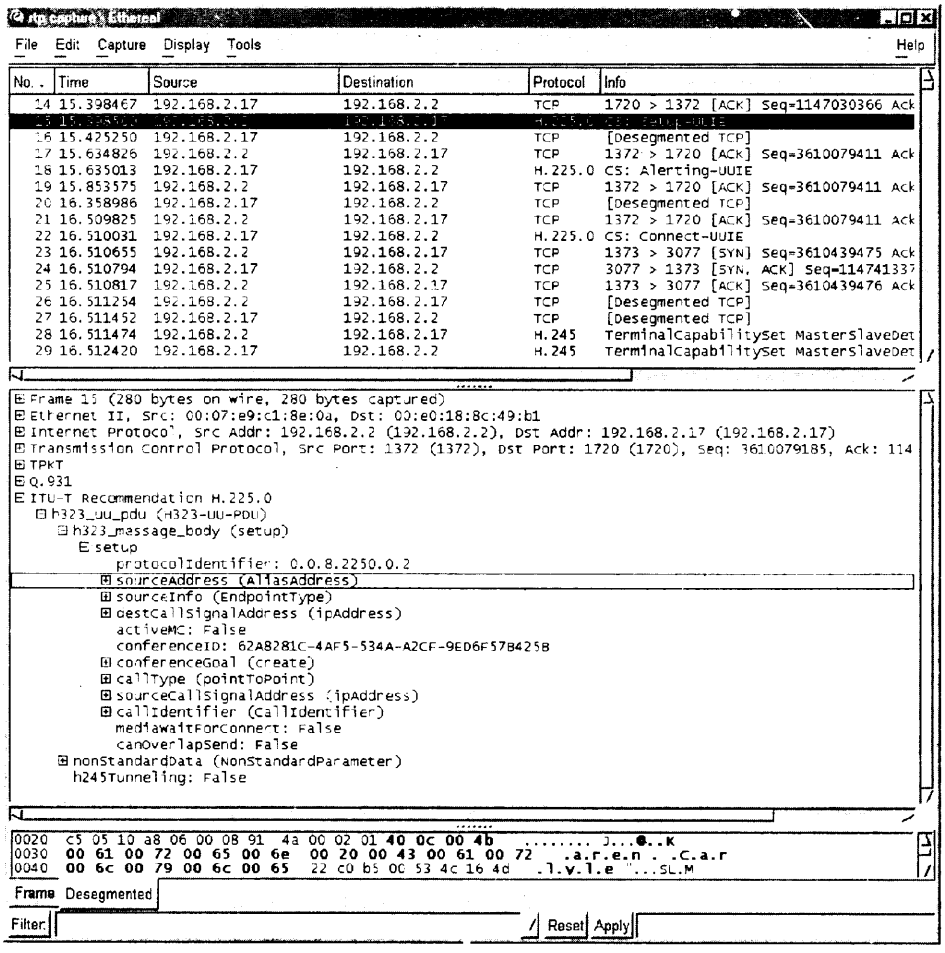

FIGURE 10.38 H.323 call setup.

the Internet and that provides the required processing capability. The second aspect involves the partitioning of the functions of the H.323 gateway into two parts. The *media gateway* is placed between the Internet and the telephone network and its role is to carry out media format conversion. The call control function is provided by *call agents* that are placed in the Internet. The residential gateways interact with the call agents to set up a telephone call. The call agents in turn interact with *SS7 gateways* that allow them to interact with the telephone signaling system. The call agents use a *media gateway control* protocol to control the setup of connections across the media gateways.[10]

[10]The media gateway control protocols are under development, so the reader is referred to the IETF website for the current status of these standards.