Introduction

The current trend in the telecommunications industry is the migration from voice-optimized switched circuit networks (SCNs) to multiservice packet networks. This migration will provide benefits in terms of economies of scale and revenue from new services. One of the greatest challenges in this migration is creating a packet-based infrastructure that will preserve the ubiquity, quality, and reliability of voice services while allowing the greatest flexibility for use of the new packet technologies. In practice, this means retaining current telephony-related services on new packet-based deployments. In this paper, we present an architecture model that can serve in the migration path to that infrastructure as well as in all-packet solutions. We recognize that most calls traverse multiple provider networks, which may use different transport technologies or call/service signaling protocols. Therefore, the next step is to create a transport-independent functional model for provider domains. This allows the solution to address, for example, introduction of new services separately from interworking between different transport technologies. The architecture model we present is one step away from physical implementations in equipment and can be used to model existing telephony equipment and protocols. Unlike protocol architectures, we do not define implementable entities such as Lucent Technologies’ elemedia® H.323 gatekeeper or the Lucent Softswitch. The elements of our model can be combined to create such entities as appropriate and can also be used to design new ones. In the European Telecommunications Standards Institute (ETSI)/Telecommunications and IP Harmonization over Networks (TIPHON) project, the model is used for standardization.

One of the greatest challenges in the migration from a traditional telephony network toward a multiservices packet network is creating a packet-based infrastructure that will preserve the ubiquity, quality, and reliability of voice services while allowing the greatest flexibility for use of the new packet technologies. In practice, this means retaining current telephony-related services on new packet-based deployments. In this paper, we present an architecture model that can serve in the migration path to that infrastructure as well as in all-packet solutions. We recognize that most calls traverse multiple provider networks, which may use different transport technologies or call/service signaling protocols. Therefore, the next step is to create a transport-independent functional model for provider domains. This allows the solution to address, for example, introduction of new services separately from interworking between different transport technologies. The architecture model we present is one step away from physical implementations in equipment and can be used to model existing telephony equipment and protocols. Unlike protocol architectures, we do not define implementable entities such as Lucent Technologies’ elemedia® H.323 gatekeeper or the Lucent Softswitch. The elements of our model can be combined to create such entities as appropriate and can also be used to design new ones. In the European Telecommunications Standards Institute (ETSI)/Telecommunications and IP Harmonization over Networks (TIPHON) project, the model is used for standardization.
phony networks, the packet network must be able to adhere to commercial and legal requirements. This implies that a packet telephony deployment should be able to provide guarantees for the availability and quality of the service. Such a deployment should be able to interwork with other packet telephony domains that may be governed by other policies, and it should allow billing for used services.

The architecture presented in this paper recognizes that most calls traverse multiple provider networks, which may use different transport technologies or call/service signaling protocols. To achieve our goals, we create a transport-independent functional model for provider domains. This allows the solution to address, for instance, introduction of new services separately from interworking between different transport technologies. Our approach groups transport-related functions and application-related functions into separate planes, which are then further subdivided into layers. We describe how this approach leads to a packet telephony architecture that can offer services on par with those offered on legacy telephone networks and provide an evolution path to broadband and mobile applications.

The architecture presented in this paper is an evolution of the ideas presented by Sijben and Spergel in an earlier paper.\(^1\) These ideas are currently being accepted by standards bodies such as the European Telecommunications Standards Institute (ETSI)/Telecommunications and IP Harmonization over Networks (TIPHON).\(^2\)

### Problem Statement

This section describes the motivation and
requirements for the architecture model presented in this paper.

**Motivation**

Currently, in the telecommunications industry, there is a growing demand for services and features that can be provided transparently over different network technologies. For example, legacy PSTN services must be available for telephone calls independent of the transport technology used—Internet protocol (IP), asynchronous transfer mode (ATM), or switched networks. Users generally prefer to retain the “feel” of traditional basic call service while demanding the rapid introduction of new services. This seems to apply to the enterprise market as well as the service provider market, both wireless and wireline. Inversely, services created for IP telephony networks may be exported to SCN users.

The problem we see is that the industry needs a way to establish these services on all networks and allow for the rapid introduction of new services. In order to do this, a model must be created that can be applied to each of the different network technologies that will exist.

Besides using different transport technologies, packet telephony networks use a variety of signaling protocols including H.323, session initiation protocol (SIP), and bearer-independent call control (BICC). As the deployment of these packet-based technologies increases, service providers will find an increasing need to interwork between the different transport technologies. This could be within their own networks or through interconnection with other service providers. Furthermore, they will want service ubiquity, meaning that all of their services will be available, independent of the underlying transport technology or the signaling protocols used. Therefore, it is necessary to devise a transport-independent, protocol-independent, and implementation-independent architecture that allows the transport aspects to be separated from the signaling and services aspects. By abstracting from the implementation, we obtain the functional architecture that we will use.

**Requirements**

Considering our proposed application space, our model must meet the following requirements:

1. It must be transport independent (that is, it must apply to networks in which the underlying transport is IP, ATM, frame relay [FR], or multiprotocol label switching [MPLS]).
2. The model must not depend on any one protocol for its implementation. It must be possible to develop new protocols to replace others without changing the basic architecture or affecting other protocols in the network.
3. The functional model must be able to support a modular implementation, thus facilitating software reuse and future evolution. It must clearly define which functions belong to each element in the architecture so that, when a new feature is required, it is clear in which element it should reside.
4. The model must be able to support multiple quality of service (QoS) levels, including per-call QoS requests.
5. The model must support end-to-end QoS across service provider boundaries.
6. The model must support end-to-end media negotiation. This will reduce unnecessary transcoding and degradation of quality levels.
7. The model must support accounting and billing for usage and also provide for prevention of unauthorized usage.
8. The model must support various charging mechanisms.
9. The model must allow intelligent call routing mechanisms and user mobility.
10. The model must allow for security in the form of authentication, authorization, and privacy.

In the remainder of this paper, we refer to these requirements by number.

**Functional Architecture**

The migration from a traditional telephony network toward a multiservices packet network poses a complex problem. One of the more challenging aspects is to devise an extendable solution. Our approach is to start with a model of the problem space and clearly separate concerns of the elements of the model. In this section, we introduce concepts that are the basis for our approach and their further sub-
divisions. First, we begin by creating functional groupings, which we call planes. Next, we describe business separation units orthogonal to these planes, which we call domains. Finally, we give a detailed description of the application plane and the transport plane.

**Planes**

As shown in Figure 1, the major functional groupings we use in our model are planes:

- The application plane, which is the part of the architecture that is specialized. Examples include voice telephony and video conferencing.
- The transport plane, which contains the generic transport functionality. It hides details of the underlying transport technology from the application. The separation of transport functionality in a separate plane addresses requirement 1.
- The management plane, which covers all management facilities. This allows us to model one management facility across multiple applications.
- The SCN plane, which consists of the functionality present in the legacy switched telephone networks. This functionality is well known and will not be further described in this paper. Actually, this plane is not a functional split but rather an instantiation of a particular set of applications on a set of transports.

**Domains**

A service domain is an abstraction of a business entity. Figure 2 shows several examples. In the figure, each service provider represents a separate service domain. The figure shows three domains:

- Domain 1: a bundled service provider offering both the transport and the telephony application;
- Domain 2: a specialized transport service provider; and
- Domain 3: a specialized telephony application service provider.

In this paper, we use the domain as a high-level concept with the following formal definition:

A service domain is a group of systems providing a service under one administration and operating under a common set of policies.

This implies that a particular service provider may consist of either a single domain or multiple domains (possibly as a result of legislation or mergers), each with its own administration or policy.

The concept of the domain allows us to move away from a homogeneous model (such as the Internet) and model not just information flows but...
also business relationships (monetary flows) and legal/regulatory constraints that will shape the architecture. Each domain will have its own business model and be subject to certain local laws and regulations. These two forces determine the technology that may be employed. For example, if a system cannot support the kind of accounting and billing needed for the business model or the locally imposed regulatory requirements, it will not be deployed. The domain paradigm also serves to remind us that, unlike in the Internet, policies are not universal, so that certain technology/protocols may be supported in one deployment but not in another.

Note that the intent of this paper is not to create a business model or describe legal/regulatory constraints. We merely create a technical functional model that can be used to support a variety of business, legal, and regulatory models.

Each domain may need to be able to enforce its policies and interwork with both peer domains and end users. To contain policies within domains, we introduce the following elements:

- The interconnect element, which enables a transport interface between transport domains. The interconnect element interfaces the local transport policies with those of another domain. To do so, the interconnect element is able to pass or block media streams. (A media stream, or flow, consists of streaming real-time voice, music, or video information for a call or session over one transport technology). One example of an interconnect element is a lightweight firewall that does not have any knowledge of applications (that is, a header rewriter and a packet filter).
- The border element, which enables an application-level service interface between service domains. The border element enforces the relevant application-level policies (for example, no calls from certain domains over a certain bandwidth, no calls from originators on blacklists, or only emergency calls when the transport

![Figure 2. Domains, interworking, and interconnection.](image)
network load exceeds a certain threshold). The border element may translate protocols and protocol profiles between its domain and the next domain. (This fulfills requirement 2.)

- The transport interface element, which provides an interface between a transport domain and an application domain. It hides the complexity and topology of the transport network from the application domain. It may also provide a QoS interface, a monitoring interface, and an access control interface.

Figure 2 gives an illustration of these domains, relationships, and elements, and it also shows the end users as their own domains. The figure shows interconnect elements between the application levels in the domains and the transport networks, border elements around the application-level service networks, and transport interface elements between the service domain and the transport domain. We can model real-world deployments by concatenating multiple domains in the manner shown in Figure 2 as desired.

Application Plane

The application plane contains the application-specific part of the architecture. Different functionalities may be present in this plane for different applications. However, in this paper, we focus on telephony applications, including voice and multimedia calls and conferencing sessions. Although call and session are defined differently, they are treated as equivalent in this model. This section deals with the functionality in the application plane needed for telephony applications and services related to such applications.

We subdivide the application plane into five layers, which are used to achieve structure and reusability. The functional grouping used for each layer is bound to the functions performed and the lifetime of the related information. The layers are defined as follows:

- The services layer contains functionality to create services, functionality to maintain data such as databases for user profiles (to enable user mobility and roaming), and routing tables (to enable number portability). The information in this layer lives rather long—orders of magnitude longer than an individual call.
- The service control (SC) layer contains functionality to control services and provides an interface between the services layer and the call control layer. Some examples are terminal registration and call routing. The information in this layer may live only as long as the user is registered. Note that behavior when a user is not registered is handled by the services layer.
- The call control (CC) layer contains functionality for call setup, teardown, and midcall services. Multiparty sessions are maintained in this layer. The information in this layer lives exactly as long as the call. This layer is the most critical in the application plane.
- The bearer control (BC) layer is required for media realization by most calls and sessions. In this paper, we use the following definition:

  A bearer is instantiated for the purpose of media communication through cooperation of the media control layer and the transport plane.

In the BC layer, the mapping is performed from the call topology to bearers for the individual media flows. This layer also negotiates details of the media flows in the bearers, such as voice quality and encoding. This allows us to meet requirement 6. Information in this layer lives as long as a media flow is desired.
- The media control (MC) layer is responsible for the realization of the media. This layer negotiates with the BC layer the properties of individual media flows (for example, media encoding and media flow addresses). The MC layer communicates with the transport plane to achieve a media path across the relevant networks with the appropriate QoS and firewall settings. The information in this layer lives as long as a media flow exists, which may be less than or equal to the lifetime of a bearer.

The description of these layers is rather abstract, yet the functions in many telecommunications systems of today are easily recognized. Using these domains, planes, and layers, we obtain a modular model that allows the maximum amount of reuse, satisfying requirement 3.

Call routing and user mobility functions may be shared among a number of call-oriented applications.
A multimedia call may have two or more bearers and as many media flows. A multiparty session has a number of bearers equal to the number of parties times the number of media. The number of media flows is at least as large as the number of bearers. There may be more media flows when the transport network has no multicast—it may have a number of media flows equal to the number of parties times the number of media.

The “Applications” and “Implementation” sections contain a number of examples that further explain the functional divisions in layers.

**Transport Plane**

The transport plane provides the transport of signaling and media packets. The transport plane is typically IP or ATM, but the model applies to Signaling System 7 (SS7) and X.25 packet switching protocol as well. Although the transport plane only arranges the bit pipes necessary to enable the application, there are two important services that the transport plane provides—QoS transport and low-level security.

**Quality of service.** QoS is an important topic in packet transport discussions, especially for IP. It is generally understood that many applications are hampered, if not made impossible, by the lack of QoS in many of today’s best-effort IP networks. It is necessary to have enough bandwidth available for the application, but bandwidth alone is not sufficient. Other parameters critical to applications that involve real-time media streams are delay, jitter, and packet loss. The ability of today’s networks to deliver the desired values for these parameters depends critically on the network load; no guarantees can be provided for the QoS delivered.

We believe that transport QoS will only be deployed if the investments allow the generation of revenue. This implies that a QoS-capable network will only give its service to paying customers, who may be end users, corporations, or other service providers.

A QoS transport network will therefore have some means of access control to protect the network from unauthorized usage and overloading. In this paper, we place end-user authentication in the application plane, and we define an interface between the transport and application planes that allows the entities in the application plane to request an end-to-end bit pipe through the transport network. A service allowing end users to request the QoS themselves would work the same way but is not explained here. How the transport network is implemented with its access control and QoS schemes is outside the scope of this paper. It does serve to show we can meet requirements 4 and 5.

**Security.** Transport security is an increasingly important topic for packet networks. One would like to achieve private, end-to-end, uninterrupted, authenticated communication. By assuming that the lines of communication in the PSTN cannot be tapped illegally, one can assume that the phone network provides this security.

On packet networks, achieving security is more difficult because in the most common form of packet network (IP over Ethernet), everyone broadcasts his/her message assuming the intended recipient will see it. Communication can be signed and encrypted in several ways. The goal of private, authenticated communication can be achieved through several techniques:

- Secure IP (IPSEC) can be used at the IP level,
- Transport layer security (TLS) protocol can be used at the transport level, or
- The application can provide its own mechanisms.

ITU-T Recommendation H.2359 provides an overview of security methods.

For the purposes of our model, we assume transport security to be covered in the transport plane because it is general to all applications. This covers requirement 10.

**Control.** We model control functionality to oversee the QoS and security policies. This functionality will oversee the end-to-end transport policies and may interface with an application domain through a transport interface element. This functionality will control the interconnect element so it can perform its functionality in accordance with requirement 7.

**Applications**

In this section, we apply the general model outlined in the previous section. This section serves as an overview of how the model may be used and how it can be applied in real-world situations. We begin by
providing a general model of the states in a phone call loosely based on the Q.931 and intelligent network (IN) call state models. Next, we extend this for supplementary services and multidomain services, showing how the model remains intact and can be refined to provide these services.

**Telephone Call**

The basic phone call consists of a number of events:

1. The call is initiated.
2. The call is authorized by the service provider.
3. The call is routed toward the destination.
4. Call-associated bearers are set up.
5. Media flows are requested for the bearers.
6. Appropriate transport is arranged for the media flows.
7. The call and bearer requests are forwarded to the next domain (or terminal).
8. The next domain confirms the bearer request.
9. The media flows are established for the bearers.
10. The transport is established for the media.
11. The call is accepted by the recipient.
12. (Optionally) the media is activated (in case the forward channel is muted to enforce billing policies).
13. (Optionally) the transport is activated.
14. The media flows.
15. The call is torn down and, along with it, the bearers, media, and transport are torn down.

**Figure 3** shows these steps in a schematic way. These 15 steps usually must be taken in this order to retain the “feel” of the SCN. If this sequence is not fol-
allowed, unexpected things could happen. For example, the perceived post-pickup delay (the time from receiver pickup to media establishment) might be very large, or the first few seconds of speech might be choppy until QoS transport is arranged for the media path.

Figure 4 shows these 15 steps in the form of an information flow. This figure, which is a subset of figures from TIPHON DTS02003, does not show special...
cases such as rejection of requests and overlap sending of dialed digits. The routing request flow performed in step 3 allows us to hide any intelligent routing mechanisms, including number portability and intelligent routing schemes (see the example in the next section). In this way, we can meet requirement 9.

Note that the bearer is set up in three phases:

1. The originator offers his capabilities for sending and receiving and commits to receive what was offered (steps 5 and 6).
2. The remote side responds and offers its capabilities, and the appropriate reverse stream can be established (steps 9 and 10).
3. When the call is accepted, the (reverse) speech channel is opened (steps 12 and 13).

Using these information flows down to the transport plane and up to the SC layer, we can implement strict access and usage control in both the transport plane (it can block any nonauthorized streams) and the application plane (it can authorize users before committing resources). This allows us to satisfy requirement 7.

This basic example is used as the foundation of the following application examples.

**Supplementary Services**

Supplementary services extend basic service. Call waiting and calling line identification (CLI) are common examples. Supplementary services can be modeled as:

- An addition to the basic information flow (that is, more information is added to the flows);
- A separate flow, for which the information elements may be piggybacked on the messaging of the basic call in an implementation; or
- A refinement of the basic flow.

The first option does not meet the requirement of modularity and hence may block future extendibility. Therefore, the latter two options are preferable. The flows shown in the previous section form the basic reference flows and hence the basic reference points in this architecture. Therefore, the CC entity has a CC-CC call setup reference point, the BC entity has a BC-BC reference point, and so on. In Figure 3, reference points are shown as black blocks on the entities.

In this section, we add new reference points to these entities and demonstrate them using a number of services from the following categories:

- Services during call setup. We use CLI as our example.
- Services operating on the call state. We discuss two such services, namely, the call transfer service and the multiparty session.
- Services based on the services layer and the SC layer. A number of services transcend the call state and, possibly, the domains. We give three examples: Internet call waiting, dynamic call screening, and prepaid calling.

**Calling line identification.** CLI is a very common service in the PSTN. IP-based services usually carry an identification of the remote party. In the phone network, this information is trustworthy. However, in an IP network, this is not necessarily true. For example, on an H.323 network, anyone could set up a call as “President@whitehouse.gov.” CLI comprises two services—CLI presentation (CLIP) and CLI restriction (CLIR). The former describes the capability to present CLI information to the receiver, while the latter represents the calling party’s ability to request that such presentation be withheld.

In our architecture, this service can be modeled as follows. Each CC entity may support a CC-CC CLI reference point, allowing the transport of CLI information as well as the information to restrict it. The CLI flows will include authentication so that the receiver can trust this flow. If the next CC entity (or its domain) is trusted to adhere to both the presentation and the restriction flows, the CLI flow will be initiated along with the call setup flow. If the receiver is not trusted or not capable of receiving this information flow, the flow can be omitted without losing the basic functionality.

**Call transfer.** The call transfer service allows a user at a terminal to signal, during a call, that the call is to be transferred to another endpoint (with or without consultation). In our model, this implies that the CC entities involved support a transfer reference point, allowing a user to trigger the CC entity to:

- Check whether the originator can be authorized to transfer this call (at all and to this destination),
Attempt to set up a call to the new destination (step 3 and beyond in the “Telephone Call” section), and

Tear down the current call (step 10 in the “Telephone Call” section).

The extra reference point is easily created in the architecture, and the flows are easily denoted. However, adding such services to a call state machine does complicate this machine, and it increases the possibility of feature interaction.

Multiparty calls. Multiparty sessions in which a user can invite additional parties to a session constitute another example of an extension to the CC functionality. This example can be modeled by adding a multiparty reference point to the relevant CC entities (although the internal behavior of the call state machine must be radically different to allow multiple users to be connected). However, support for multiparty service has an additional twist—a multiparty call requires multiparty bearers. In our model, the CC entity requests a multiparty bearer to be established. Depending on the underlying network and the available resources, the BC entity will attempt to acquire either a multicast transport pipe or a media bridge. Depending on the outcome, the existing media flows must be routed over that pipe or bridge. Subsequently, the new media flows to it. Creation of this service in our model therefore imposes the addition of multiparty capabilities in the CC layer and in the layers below.

Internet call waiting. For users with one phone line who want to be reachable while on line, Internet call waiting is an attractive service. In our architecture, we model Internet call waiting in the SC layer of the SCN and packet-based domains.

We assume, for our example, that an on-line user receives a telephone call to the line used for Internet dialup. The call routing request in the SCN will be propagated to the voice over IP (VoIP) domain (for instance, through the integrated services digital network (ISDN)13 “Call Divert on Busy” function). Arriving on the packet network, the routing request will trigger a request to the user. This example can be mapped to our architecture as follows. The SC layer in the SCN is triggered by the “Call Divert on Busy” function and queries the user over the IP network. This query can be viewed as signaling on the SC layer. The user chooses to accept the incoming call as a VoIP call, terminate the Internet session and accept the call as a regular telephone call, or reject the incoming call altogether. If a VoIP call is accepted, a call request is generated and sent to the user, and the regular flow of Figure 4 is followed. Thus the general model stays intact and is refined for the application. Other refinements are possible, as we show in subsequent examples.

Dynamic call screening. With the dynamic call screening service, a user can provide a profile that specifies how to handle calls based on the CLI of the caller (Sijben and Spergel1 provide further details). This feature is placed in the SC or the services layer of our model. When the routing request is received by the appropriate entity, a routing confirmation provides a routable address or a name to which the call needs to be routed. This functionality may be used to screen the originator identification and divert the call to the appropriate place—for example, the user’s cell phone for a selective list of callers—while the rest is sent to the voice mail system. More complex policies can be envisioned. Such policies may not only use the originator identification as a key to deliver the routing address, but also the called user’s (cell phone) location, the time of day, and/or the availability of the called user’s secretary.

Prepaid calling. In any commercial scenario, the access request triggers a check against the user’s solvency. A prepaid scenario contains a similar check. In addition, checks need to be performed regularly if the account has not run out during the call. This implies that functionality in the services layer will have to compute the per-second price of the call (possibly based on the called address) and the length of time it is allowed to last. When the account has run out, the call is likely to be terminated.

This functionality allows itself to be implemented rather easily in our architecture model. The only addition is the ability for SC to initiate termination of a call. This functionality is needed anyway for other scenarios, such as loss of authorization for the user’s registration,10
This example shows that the architecture can support both prepaid and subscription-based billing mechanisms, thus fulfilling requirement 8.

When the account is about to run out, this service functionality may inform the user of this situation and offer ways to upgrade the account. This is a typical example of an interactive service within a prepaid service. Interactive services are discussed in the next section.

Interactive Services

A number of interactive services are known in the telephone system today. We just described a kind of feedback in a prepaid scenario with the ability to upgrade the user’s account. A second example is found in interactive voice response systems that form the front end of numerous companies (“Dial ‘1’ for support; Dial ‘2’ for complaints”). In this example, the interaction with the calling user occurs before the call is routed to the final destination. A third example is interactive services that act like an endpoint of the call (such as voice mail systems or information numbers). In this section, we fit these services into our architecture.

We use the example of a gateway application with interaction with the user. In such a setting, the user may be requested to provide an account number, a personal identification number (PIN), and, subsequently, a called number (this is a typical toll bypass application). These requests can be embedded in the access and routing request in the flows in Figure 4. A user who dials in to such a gateway has not provided proof of identity by simply connecting to the gateway. For these requests, the toll bypass service executes a challenge to the user to establish caller identity.

The access and routing requests trigger interaction—an information flow between the service and the user. This interaction is executed through in-band voice messages in the case of a voice-only terminal (the typical PSTN phone) but may be executed as out-of-band signaling to packet-based, feature-rich terminals. In our architecture, the SC layer will have to decide how to signal these requests (out-of-band and/or in-band) based on the used terminal.

The decision as to how this information flow is signaled to the user is mostly an implementation decision. In the architecture, we see that the requests are mapped onto the media stream but do not change information content. This is depicted in Figure 5, which shows how the access request triggers an authentication information exchange implemented using the messages “Please enter PIN” and “Digit
string.” This is a fairly elementary authentication request, as also described in TIPHON DTS0200310 and in a standards contribution by Sijben,14

Depending on the application, the information flows can be mapped to protocols in different ways. In this example, one may choose to incorporate H.248 information elements for signals in the H.225.0 (remote access service [RAS] or Q.931) messages or introduce a hypertext transfer protocol (HTTP) channel to show the message on a browser-like screen. If the signaling is implemented in the media stream in the form of voice messages and dual-tone multi-frequency (DTMF) detection, the media must be connected to the user. This places the requirement on such implementations that the media for calls that may require such a service flows over a controllable point. Through such a point, the media can be rerouted so that either the terminal has to provide this capability or the network needs to use a pivot point under the control of the service.

Similar statements could be made for other typical IN services, such as the exchange necessary for account updates for prepaid users, but they could also be made for endpoint-like services such as voice-mail systems, of which the control could be out-of-band while, when appropriate, the media is passed in-band.

Implementation

In this section, we map the functional architecture and its flows back to real-world standard protocols and show how they fit into the architecture. Figure 6 gives a high-level overview of the best known protocols (H.323, SIP, and BICC) and their mapping to the architecture model. The figure shows physical elements, such as terminal, softswitch (providing protocol-independent CC functionality and services like the Lucent Softswitch), and media gateway as well as the protocols between them mapped to the layers in the architecture model. In the figure, protocol representations are as follows:

- The H.323 communication is represented by the labels “H.225.0,”15 “H.245,”16 and “H.248.”17
- The SIP5 communication is represented by the labels “SIP,” “SDP” (session description protocol),18 and “MEGACO” (media gateway control protocol)—another name for H.248.
- The BICC6 communication is labeled “BICC.” This communication also uses H.248 for the vertical interface.
- Media streams are labeled “RTP” (real-time transport protocol).19

In the following sections, we describe the mappings to these protocols in detail. The mappings show that requirement 2 is satisfied. We describe the mappings in the form of examples, which assume the following setting: an IP-based user endpoint (such as a terminal or a SIP server) can communicate with a network service entity as a front end while another network entity performs the back-end functionality of call admission control and call routing. An interconnect element is implemented as a controllable firewall/media gateway to be controlled by the first network element.

H.323 Flows

Figure 7 shows the mapping of the information flows 1 to 14 of Figure 4 to H.323 messaging flows. The BC, CC, and SC layers are implemented as one gatekeeper. The MC layer is implemented as the media gateway, and the services layer is implemented as the back-end gatekeeper. In this example, the protocol used to communicate with the IP transport is resource reservation protocol (RSVP).20 In our example, the QoS transport is spread into several domains. The domain we describe here has RSVP between the terminal and the interconnect element; what happens beyond that is outside the scope of our examples.

In the example, we show how to implement the call request and the bearer request using the FastConnect option of H.323. When FastConnect is used, the H.225.0 setup message carries H.245 information elements, thus combining steps 1 and 4 of the information flow of Figure 4. If FastConnect were not used, the H.225.0 setup message would implement the call request (step 1), and an H.245 open logical channel request would implement the bearer request (step 4). A similar story holds for the subsequent messages, “Alerting” and “Connect,” on which the bearer information is piggybacked in the form of H.245 information elements.
SIP Flows

Figure 8 shows the mapping of the information flows 1 to 14 of Figure 4 to SIP messaging flows. The BC, CC, and SC layers are implemented as one SIP proxy server. The MC layer is implemented as the media gateway, and the services layer is implemented as the back-end SIP server. In this example as well, the protocol used to communicate with the IP transport is RSVP. SIP does not allow separate signaling for bearers but includes SDP session descriptions in the SIP messages; hence, in SIP, steps 1 and 4 are always performed simultaneously. The same holds for the “Ringing” and “OK” messages—bearer information is piggybacked in these messages in the form of SDP structures.

BICC Flows

Figure 9 shows the mapping of the information flows 1 to 14 of Figure 4 to BICC messaging flows. The BC and CC layers are implemented as one BICC call service function (CSF). The MC layer is implemented as the BICC bearer interworking function (BIWF), and the SC and services layers are implemented as a back-end service platform (presumably IN). Since the only stable version of BICC, Capability Set 1 (CS1), supports ATM and not IP, ATM signaling is used in the communication with the transport network. We chose to implement the CallRequest and BearerRequest using an initial address message (IAM) so that steps 1 and 4 are performed at the same time. The mapping of the other flows is straightforward.
Interworking These Protocols

Interworking between SIP, BICC, and H.323 is an emerging topic at the moment. Thus, the only available literature is in drafts and proposals that have been submitted to standards bodies for consideration. In this section, we show that the architecture model and flows described in this paper can be used for this interworking.

From the flows shown in Figures 7, 8, and 9, one can see that the flows of H.323, SIP, and BICC can be rather similar. Interworking may therefore be done at an entity with two faces. On one side, it could look like an H.323 gatekeeper—on the other, like a SIP proxy server or a BICC CSF. The states of the protocols would look the same (they all conform to the Q.931/IN model).
Note that interworking between these protocols is possible given an appropriate implementation of the specifications, but it is not straightforward given any two implementations of the protocols. Interworking may be hampered by the following issues:

- H.323 allows the separate exchange of call request and bearer request information while SIP does not. In fact, the H.323 base mode of operation (as defined in ITU-T Recommendation H.323\(^3\)) is to send these exchanges separately.
Some details of the information exchanged in the H.323 and SIP messages do not necessarily map. This mapping is beyond the scope of this paper and will be included in further study.

BICC for IP transport, BICC Capability Set 2 (CS2), is still under development. The ease or difficulty of interworking with other protocols will be affected by decisions made during the standardization process of BICC CS2.

From this exercise, one can see that interworking
between SIP, BICC, and H.323 can be aided by the high-level flows described in the “Functional Architecture” section. One may define compliant H.323 and SIP entities that will be able to interwork using these flows. Note that this does not imply that any implementation of the two specifications may interoperate, because they may not implement the same subset of the specifications.

Conclusions and Future Work

In this paper, we have described how creating a functional model helps in understanding and modeling the transition from SCNs to packet telephony as well as an end-to-end packet telephony deployment. The model described here abstracts from transport issues while retaining aspects of QoS and security so that it will work for all kinds of transport, including third-generation mobile networks. In the “Applications” section, we have shown how the model can be used for the basic call and a number of (supplementary) services. This gives us confidence that the addition of new services does not break the model but merely refines it.

In the “Implementation” section, we have shown that the model can be translated back to the real world and can be mapped to existing protocols, giving us confidence that the model can be applied in the real world. Because the model was created with requirements for service provider networks, this exercise also shows that these protocols can satisfy the basic requirements for service provider networks. As a side effect, we have seen how, through this model, existing protocols can be made to interwork, but we have also shown that not any common implementation of these protocols will satisfy the requirements. Our model offers directions for refinements of these protocols.

The model described here has been brought to the standards bodies, especially ETSI/TIPHON, and it has received acceptance. Other bodies in which similar work is being done include the Multiservice Switching Forum (MSF) and the BICC architecture body. These two architectures, however, are still more or less implementation specific and can therefore be regarded as subsets of this model.

Although the model described in this paper is receiving acceptance in the industry, much work remains to be done, and the model must withstand extensive peer review. In the “Applications” section, we explored supplementary services and interactive services. This work has not been discussed elsewhere and is quite new. Going through this exercise has given us confidence that the model works, but more services need to be explored. Experience with implementation of these services will also provide more insight into the work.

The architecture model we presented is one step away from physical implementations in equipment. Our model can be used to model existing telephony equipment and protocols. Unlike the H.323 and SIP specifications, our specifications do not define such service entities as the H.323 gatekeeper or the SIP proxy server. The elements of our model can be combined to create such entities as appropriate. The model can also be used to design new ones.

The work in this paper has laid the foundation for migration toward multiservice packet networks. We believe that revenue-generating services will be the most profitable if they integrate other services and are offered transparently. The next steps in this work should include a more comprehensive model for creating and integrating services from multiple providers, deployments, and technologies.

Acknowledgments

We would like to thank a number of people who have been working with us in creating this model, firstly, the active participants in the TIPHON architecture interim meetings. We would also like to thank Kumar Vemuri of Lucent’s Service Provider Networks organization for reviewing the drafts and Mike Buckley of the OPENet Solutions unit for the initial discussions on the planes. Finally, we would like to acknowledge Pieter Veenstra and Oskar van Deventer of Royal KPN N.V. of the Hague, the Netherlands, for their insightful discussions on BICC.

References
3. International Telecommunication Union, “Packet-Based Multimedia Communications


(Manuscript approved August 2000)

PAUL G. A. SIJBEN is a distinguished member of technical staff in the Forward-Looking Work Department at Lucent Technologies’ Service Provider Networks organization in Huizen, the Netherlands. His responsibilities include research and development of multimedia service architectures and voice over IP. Mr. Sijben received an M.S. degree in computer science from the University of Twente in the Netherlands.

JOHN P. L. SEGERS, a member of technical staff in the Forward-Looking Work Department at Lucent Technologies’ Service Provider Networks organization in Huizen, the Netherlands, holds M.Sc. and Ph.D. degrees in computer science from Eindhoven University of Technology in Eindhoven, the Netherlands. Currently, Dr. Segers is working on voice
over IP architecture and media gateway control (H.248/MEGACO).

LOUISE F. A. SPERGEL is a distinguished member of technical staff in the End-to-End Architecture Department at Bell Labs in Warren, New Jersey. She holds B.S. and M.S. degrees in electrical engineering from Cornell University in Ithaca, New York, and the Massachusetts Institute of Technology in Cambridge, respectively. Ms. Spergel works on architecture and standards for Internet protocol telephony and is vice-chair of the ETSI/TIPHON project.

JACK KOZIK is director of the Next-Generation Architecture and Evolution Department in Lucent Technologies’ e-Services organization in Naperville, Illinois. His department is responsible for planning the evolution of IN and messaging platforms and applications, focusing on key customer needs for enhanced services in packet-based fixed and mobile networks. Mr. Kozik holds a B.S. in computer science and an M.S. in electrical engineering from Purdue University in West Lafayette, Indiana. ♦