The Internet: A Global Telecommunications Solution?

Laurent Mathy, Christopher Edwards, and David Hutchison, Lancaster University

Abstract

The provision and support of new distributed multimedia services are of prime concern for telecommunications operators and suppliers. Clearly, the potential of the latest Internet protocols to contribute communications components is of considerable interest to them. In this article we first review some of the new types of application and their requirements, and identify the need to support applications that have strict QoS requirements, the so-called critical applications. We review two proposals for enhancing the Internet service architecture. In addition to the integrated services work of the IETF, we look at the more recent proposals for differentiated services in the Internet. We then individually review recent protocol developments proposed to improve the Internet, and to support real-time and multimedia communications. These are IPv6 (the new version of the Internet Protocol), Resource reSerVation Protocol, and Multiprotocol Label Switching, respectively. In each case, we attempt to provide critical reviews in order to assess their suitability for this purpose. Finally, we indicate what the basis of the future infrastructure might be in order to support the full variety of application requirements.

One of the key requirements in deploying a pervasive and ubiquitous information superhighway is the development of a global integrated telecommunications infrastructure capable of physically moving users' information among geographically separated points. This global communication network will not only have to cope with the amount of traffic generated by the tremendous number of anticipated users, but will also have to deal with a wide spectrum of traffic characteristics. This is because the network will have to support, simultaneously, applications that have a wide range of expectations and requirements.

Among the networks available today, those based on the concept of packet switching offer the highest potential degree of flexibility to meet the different application requirements, and therefore offer the best available technology on which the global telecommunications infrastructure could rely. Because it already connects millions of users, the Internet is the uncontested prime candidate to constitute the core of a global infrastructure.

The Internet was originally designed to move files among computers as well as to allow remote access to computers. Such tasks have somewhat loose time requirements, and have led to a simple and scalable network design that offers a *best-effort* service, in which the network does not guarantee anything, not even delivery of the data. This best-effort service stems from a clear design policy to trade everything possible for simplicity. Because the major applications that first used the network could cope with a wide range of provided service quality (such applications are often said to be *elastic*), the network's simplicity allowed wide deployment of the technology.

However, with time, the Internet has perhaps become a victim of its own success. Now that more and more people realize the tremendous potential represented by a global communication network, they expect it to be able to provide support for a huge range of applications. We contend that, in order to do this successfully, the time has come for the Internet to evolve. This evolution has to follow along several lines. First, it must accommodate a fast-increasing number of users, and must be upgraded to cope with an even faster-increasing traffic load. Furthermore, in a commercial context, the flat best-effort service provided by the network may not be the most appropriate any longer: customers requiring the assurance of better quality of service (QoS), and willing to pay more, should be able to get better service than customers paying the basic rate. Therefore, even without fundamentally modifying the sorts of service provided, we already see a need for some sort of quality discrimination within the service. Also, new types of distributed real-time applications are expected to use the global communication infrastructure. Such applications have more stringent time requirements than elastic ones, and consequently will require new classes of service from the network. Somehow, if it is to become the core technology in the global infrastructure, the Internet must be able to provide such services.

We first briefly outline a taxonomy of applications and their requirements. Then we individually review recent developments and research results proposed to improve the Internet. We analyze two service architectures that have been proposed to extend the set of communication services in the Internet (currently limited to a "flat" best-effort service). We then present

0890-8044/00/$10.00 © 2000 IEEE

what we believe to be the most prominent protocols and techniques that have been designed for the implementation of the service architectures described later, placing them in the context of the two previous sections. Then we try to give a "global" picture of the evolution trends in the Internet as the potential core of a global communication infrastructure.

The reader should note that many important developments, other than those discussed in this article, are also being carried out within the Internet community (e.g., multicast QoS routing, security, and reliable multicast, to name but a few). These have been left out because we aim to focus the reader's attention on a "communication service" argument.

We also wish to emphasize that the views presented in this article are the authors' only and do not represent any consensus within the Internet research community (such a consensus does not exist).

Finally, it should be noted that the material presented within this article uses what the authors believe to be an accurate snapshot of the status of the various areas discussed at the time of writing. It is clear that as time progresses, various "work in progress" references used within the article will be enhanced, although the impact of this is expected to be minimal given that fundamental issues are likely to remain unchanged.

Application Requirements

In packet-switched networks, the bulk of traffic has so far been generated by *elastic* applications. Because the pieces of data exchanged by such applications are carried over the network as packets that have few time constraints, we call such traffic *discrete media*. The lack of time constraints on the packets allows the network to view them as loosely coupled. In fact, in the Internet, the basic service is provided by considering the data packets independent of each other (such packets are called *datagrams*). As already mentioned, elastic applications are essentially composed of file transfers.

However, some elastic applications exhibit a degree of interactivity, and therefore have additional performance requirements. Although these requirements may in some cases be quite loose and dependent on the variable demands of human users, minima can nevertheless be determined. For instance, the World Wide Web, the most important application on the Internet today, makes extensive use of on-demand file transfers between servers and clients. Because the perceived "quality" of the transfers depends on the users, their moods, the purpose of the transfers, as well as many other factors, strict performance requirements cannot easily be identified. On the other hand, it is generally recognized that any user would "give up" browsing if each page transfer took several minutes. Such time constraints may also be linked with economic reasons, when users pay for their access links in proportion to the time of use (as is the case when connecting via a modem in most European countries).

In addition to discrete media, applications increasingly make use of *continuous media*, thanks to advances in coding technology and the availability of multimedia computers. In continuous media, specifically video and audio, the data have intrinsic temporal and spatial relationships that must be respected for these forms of data to make sense. The performance requirements of continuous media are closely linked to their perceived quality.

Techniques have been devised whereby the playout quality of continuous media is adjusted to match the instantaneous capabilities of a system, and in particular of a network. Such techniques, used by *adaptive applications*, allow the use of multimedia applications on best-effort networks such as the Internet. However, even the best adaptive techniques are powerless when facing the poorest conditions in a network; and, as a consequence, guarantees cannot generally be given as to the quality delivered by adaptive applications on best-effort networks. On the other hand, in order that distributed multimedia applications become ubiquitous, especially in a commercial environment, there is a need for a communication platform that is able to provide better control and guarantees over performance. Indeed, for commercial use of applications such as teleteaching, teleconferencing, medical telediagnostics, video/entertainment on demand, and distributed games, starting a communication session may be worthwhile only if some minimum performance can be guaranteed throughout its duration. These can collectively be called *critical applications*. Critical applications can further be classified into *intolerant applications*, which do not tolerate any deviation from their expressed requirements (e.g., interactive games, some control applications), and *tolerant applications*, which essentially have nominal requirements but use adaptive techniques to deal with occasional violations of these requirements (e.g., interactive voice applications).

Typically, the requirements of critical applications can be expressed as a (sub)set of values representing bandwidth, delay, jitter, and loss rate constraints for the network. In order to be able to meet these constraints throughout the lifetime of a communication session, the components of the network (or at least the subset representing those "manipulating" the packets of the session) must be aware of their values and must cooperate in taking actions to enforce these bounds. To that end, a network component has to take part in some or all of the following general activities: admission control, resource reservation, packet scheduling, traffic policing, and signaling, in order to enforce the traffic constraints of a session.

It should be clear that because of their very different characteristics, elastic, adaptive, and critical applications[1] cover separate regions of a wide spectrum of demand. In other words, these different types of application complement, rather than compete with, each other. A global telecommunications infrastructure will therefore have to support all of them simultaneously and, if possible, seamlessly. This calls for the design of a network where the overhead associated with each type of application is no greater than necessary. In the next section we review potentially promising approaches to providing such support in the Internet. Our conjecture is that these, in combination, may provide the sort of support required by the spectrum of applications.

Proposals for an Enhanced Internet Service Architecture

In this section we present and discuss the integrated services (IntServ) and more recent differentiated services (DiffServ) architectures as proposals for enhancing service provision within the Internet.

Integrated Services

An Overview of IntServ — In the IntServ architecture [1], three classes of service are proposed based on applications' delay requirements. These are the guaranteed-service class, which provides for delay-bounded service agreements; the controlled-load service class, which provides for a form of statistical delay service agreement (nominal mean delay) that will not be violated more often than in an unloaded network; and the well-known best-effort service, which is further partitioned

[1] *This categorization of applications is an approximation. Applications certainly make up a continuous spectrum, but we believe the categories described in this section represent the majority of the application spectrum.*

into three categories: interactive burst (e.g., Web), interactive bulk (e.g., FTP) and asynchronous (e.g., e-mail).

The main point is that the guaranteed service and controlled load classes are based on quantitative service requirements, and both require signaling and admission control in network nodes. These services can be provided either per-flow or per-flow-aggregate, depending on flow concentration at different points in the network. Although the IntServ architecture need not be tied to any particular signaling protocol, Resource Reservation Protocol (RSVP), described below, is often regarded as *the* signaling protocol in IntServ. Best-effort service, on the other hand, does not require signaling.

Advantages of IntServ — The major advantage of IntServ is that it provides service classes which closely match the different application types described earlier and their requirements. For example, the guaranteed service class is particularly well suited to the support of critical, intolerant applications. On the other hand, critical, tolerant applications and some adaptive applications can generally be efficiently supported by controlled load services. Other adaptive and elastic applications are accommodated in the best-effort service class.

A major characteristic of IntServ it that it leaves the existing best-effort service class mostly unchanged (except for a further subdivision of the class), so it does not involve any change to existing applications. This is an important property since IntServ is then capable of providing this class of service as efficiently as the current Internet. IntServ also leaves the forwarding mechanism in the network unchanged. This allows for an incremental deployment of the architecture, while allowing end systems that have not been upgraded to support IntServ to be able to receive data from any IntServ class (with, of course, a possible loss of guarantee).

IntServ provides a very interesting set of service classes that, although maybe not ideal, represent an excellent approximation of the kind of services required in a global telecommunication platform since it does not discriminate against any applications.

Disadvantages of IntServ — End-to-end service guarantees cannot be supported unless all nodes along the route support IntServ. This is obviously so because any "pure" best-effort node along any route can treat packets in such a way that the end-to-end service agreements are violated.

Although it is recognized that the support of per-flow guarantees in the core of the Internet will pose severe scalability problems [2], it is our belief that the problem of flow aggregation has not been thoroughly studied yet and needs more research effort. It is crucial to solve this problem in advance of widespread deployment of the IntServ architecture, especially in the core of the network.

Finally, the subclassing of best-effort service, although already a significant improvement on the flat best-effort service currently provided in the Internet, may be considered somewhat "rough" in a commercial network. We believe it could be profitable to have finer-grained subclassing of the best-effort service class.

Differentiated Services

An Overview of DiffServ — To provide QoS support, a network must somehow allow for controlled unfairness in the use of its resources. As we will show later, controlling to a granularity as fine as a flow of data requires advanced signaling protocols. By recognizing that most of the data flows generated by different applications can be ultimately classified into a few general categories (i.e., traffic classes), the DiffServ architecture [3] aims at providing simple and scalable service differentiation. It does this by discriminating and treating the data flows according to their traffic class, thus providing a logical separation of the traffic in the different classes.

In DiffServ, scalability and flexibility are achieved by following a hierarchical model for network resource management:

- Interdomain resource management: unidirectional service levels, and hence traffic contracts, are agreed at each boundary point between a customer[2] and a provider for the traffic entering the provider network.
- Intradomain resource management: the service provider is solely responsible for the configuration and provisioning of resources within its domain (i.e., the network). Furthermore, service policies are also left to the provider.

At their boundaries, service providers build their offered services with a combination of traffic classes (to provide controlled unfairness), traffic conditioning (a function that modifies traffic characteristics to make it conform to a traffic profile and thus ensure that traffic contracts are respected), and billing (to control and balance service demand). Provisioning and partitioning of both boundary and interior resources are the responsibility of the service provider and, as such, outside the scope of DiffServ. For example, DiffServ does not impose either the number of traffic classes or their characteristics on a service provider.

Although traffic classes are nominally supported by interior routers, DiffServ does not impose any requirement on interior resources and functionalities. For example, traffic conditioning (i.e., metering, marking, shaping, or dropping) in the interior of a network is left to the discretion of the service providers.

If each packet conveyed across a service provider's network simply carries in its header an identification of the traffic class (called a DS codepoint) to which it belongs, the network can easily provide a different level of service to each class. It does this by appropriately treating the corresponding packets, say, by selecting the appropriate per-hop behavior (PHB) for each packet. In both IPv4 and IPv6, the traffic class is denoted by use of the DS header field.

It must be noted that DiffServ is based on local service agreements at customer/provider boundaries. Therefore, end-to-end services will be built by concatenating such local agreements at each domain boundary along the route to the final destination. The concatenation of local services to provide meaningful end-to-end services is still an open research issue.

The net result of the DiffServ approach is that per-flow state is avoided within the network, since individual flows are aggregated in classes.

Benefits of DiffServ — The DiffServ architecture is an elegant way to provide much needed service discrimination within a commercial network. Customers willing to pay more will see their applications receive better service than those paying less. This scheme exhibits an "auto-funding" property: "popular" traffic classes generate more revenues, which can be used to increase their provisioning.

A traffic class is a predefined aggregate of traffic. Compared with the aggregate of flows described earlier, traffic classes in DiffServ are accessible without signaling, which means they are readily available to applications without any setup delay. Consequently, traffic classes can provide *qualitative* or *relative* services to applications that cannot express their requirements

[2] *A customer can be either a host (directly connected to the provider network) or a network (e.g., an Internet service provider network connected to a transit network). Note that at any boundary point between two domains, each network is in turn customer and provider, depending on the direction of traffic.*

quantitatively. This conforms to the original design philosophy of the Internet. An example of qualitative service is "traffic offered at service level A will be delivered with low latency," while a relative service could be "traffic offered at service level A will be delivered with higher probability than traffic offered at service level B." *Quantitative* services can also be provided by DiffServ. A quantitative service might be "90 percent of in-profile traffic offered at service level C will be delivered."

Since the provisioning of traffic classes is left to the provider's discretion, this provisioning can, and in the near future will, be performed statically and manually. Hence, existing management tools and protocols can be used to that end. However, this does not rule out the possibility of more automatic procedures for provisioning.

The only functionality actually imposed by DiffServ in interior routers is packet classification. This classification is simplified from that in RSVP because it is based on a single IP header field containing the DS codepoint, rather than multiple fields from different headers. This has the potential of allowing functions performed on every packet, such as traffic policing or shaping, to be done at the boundaries of domains, so forwarding is the main operation performed within the provider network.

Another advantage of DiffServ is that the classification of the traffic, and the subsequent selection of a DS codepoint for the packets, need not be performed in the end systems. Indeed, any router in the stub network where the host resides, or the ingress router at the boundary between the stub and provider networks, can be configured to classify (on a per-flow basis), mark, and shape the traffic from the hosts. Such routers are the only points where per-flow classification may occur, which does not pose any problem because they are at the edge of the Internet, where flow concentration is low. The potential noninvolvement of end systems, and the use of existing and widespread management tools and protocols allows swift and incremental deployment of the DiffServ architecture.

Shortcomings of DiffServ — Simultaneously providing several services with differing qualities within the same network is a very difficult task. Despite its apparent simplicity, DiffServ does not make this task any simpler. Instead, in DiffServ it was decided to keep the operating mode of the network simple by pushing as much complexity as possible onto network provisioning and configuration. Of course, network provisioning and configuration have been performed since the creation of the very first communication networks, and thus they benefit from long experience, and available tools and traffic models. However, so far, large networks have mainly offered a single type of service (best-effort service in the Internet, interactive voice in telephone networks, etc.). The provisioning of networks providing multiple classes of service at the same time is therefore a rather new area which requires much research to study the added complexity due to possibly adverse interactions between different classes of service. The construction of end-to-end services by concatenating local service agreements is also a nontrivial research issue.

The key to provisioning is the knowledge of traffic patterns and volumes traversing *each* node of the network. This also requires a good knowledge of network topology and routing. The problem with the Internet is that provisioning will be performed on a much slower timescale than the timescales at which traffic dynamics and network dynamics (e.g., route changes) occur. This problem can be illustrated with the simplest case of a single service provider network whose service agreements with customers are static. Although the amount of traffic entering the domain is known and policed, it is impossible to guarantee that overloading of resources will be avoided. This is caused by two factors:

- The entering packets can be bound to any destination in the Internet, and may thus be routed towards any border router of the domain (except the one where it entered). In the worst case, a *substantial proportion* of the entering packets might all exit the domain through the same border router
- Route changes can suddenly shift vast amounts of traffic from one router to another.

We therefore see that unless resources are massively overprovisioned in both interior and border routers, traffic and network dynamics can cause momentary violation of service agreements, especially those relating to quantitative services. On the other hand, massive overprovisioning results in a very poor statistical multiplexing gain, and is therefore inefficient and expensive.

To increase resource usage in their network, service providers can trade generality and robustness for efficiency. For example, to limit the amount of expensive resources dedicated to the support of quantitative services, service providers can limit quantitative service contracts to apply between any pair of border routers in the domain. In such a case, the service would apply only to packets entering the domain at a designated ingress router and leaving the domain at a designated egress router.[3] This helps solve the first problem described above at the cost of generality, since only packets bound for destinations "served" through the egress router can benefit from the service. Of course, to ensure that the egress router is in the route to any given destination, the interdomain routing entry for that destination must be statically fixed in the ingress router. Even for a fixed ingress-egress pair, intradomain routing dynamics can still occur. This means that the set of internal routers visited by the packets travelling between the ingress and egress routers can still suddenly change. However, the "directionality" of the traffic considered here is such that the number of possible routes is considerably reduced compared with the general case, and so is the resulting and necessary overprovisioning. A service provider could, however, reduce to a minimum the overprovisioning of quantitative services offered between pairs of border routers by "pinning" the intradomain route between those routers. Fixing the egress router for a given destination and/or pinning internal routes between border routers nevertheless incurs a loss of robustness. In multicast, where receivers can join and leave the communication at any time, the problem of efficient provisioning will be even worse.

Alternatively, a service provider might wish to use dynamic logical provisioning and configuration (i.e., sharing of resources between classes) as an answer to the problems of network and traffic dynamics. However, depending on the type of service agreement (qualitative, relative, or quantitative) and the QoS parameters involved in the agreement, dynamic logical provisioning might require signaling and admission control.

From the point of view of a flow, the class bandwidth is not a meaningful parameter. Indeed, bandwidth is a class property *shared* by all the flows in the class, and the bandwidth received by an individual flow depends on the number of competing flows in the class as well as the fairness of their respective responses to traffic conditions in the class. Therefore, to receive some quantitative bandwidth guarantees, a flow must "reserve" its share of bandwidth along the data path, which involves some form of end-to-end signaling and admission control (at least among logical entities called *bandwidth brokers*). This end-to-end signaling should also track network dynamics (i.e., route changes) to enforce the guarantees, which can prove

[3] *Depending on the type of guarantees offered by the service agreement, the constraint of a single egress router can be relaxed to a set of well-defined egress routers.*

very complex. Furthermore, even qualitative bandwidth agreements require end-to-end signaling and admission control. This is because even if one class is guaranteed to have more bandwidth than another, the number and behavior of flows in the latter class may result in smaller shares of bandwidth for these flows than for the flows in the other class. Hence, in this case, end-to-end signaling would also be required to ensure that in every node along the path, the bandwidth received by a flow in a high bandwidth class is greater than the bandwidth received by a flow in a smaller bandwidth class.

On the other hand, delay and error rates are class properties that *apply* to every flow of a class. This is because in every router visited, all the packets sent in a given class share the queue devoted to that class. Consequently, as long as each router manages its queues to maintain a relative relationship between the delay and/or error rate of different classes, relative service agreements can be guaranteed without any signaling. However, if quantitative delay or error rate bounds are required, end-to-end signaling and admission control are also required.

End-to-end signaling and admission control would increase the complexity of the DiffServ architecture. The idea of dynamically negotiable service agreements has also been suggested as a way of improving resource usage in the network [3]. Such dynamic service-level agreements would require complex signaling, since the changes might affect the agreements a provider has with several neighboring networks. The timescale on which such dynamic provisioning could occur would be limited by scalability considerations, which in turn could impede its usefulness.

We therefore believe that in its simplest and most general form, DiffServ can efficiently provide pure relative service agreements on delay and error rates among classes. However, unless complex signaling and admission control are introduced in the DiffServ architecture, or generality and robustness are sacrificed to some extent, guarantees on bandwidth as well as quantitative bounds on delay and error rates cannot be provided. The reader should bear in mind that signaling could only be used by applications which can at least give a quantitative estimate of their requirements, and reside on hosts that have been modified to support signaling, thus limiting the immediate usability of DiffServ.

It should be noted that from a complexity point of view, a DiffServ scenario with dynamic provisioning and admission control is very close to an IntServ scenario with flow aggregation. The difference is that precise delay and error rate bounds might not be computed with DiffServ, since the delays and error rates introduced by each router in the domain may not be available to the bandwidth broker.

We therefore conclude that although DiffServ will undoubtedly improve support for a number of applications and is urgently needed in the Internet, it does not represent the ultimate solution for QoS support for all types of applications. The suitability of DiffServ for a given application could also depend on the context in which that application is being used. If we take the example of Internet telephony, we see that DiffServ is suitable for providing a cheap solution for internal calls between remote sites of a company by emulating leased lines between these sites. However, DiffServ may prove unsuitable for the support of telephony over the Internet for the general public, because people do not usually restrict their calls to only a few destinations.

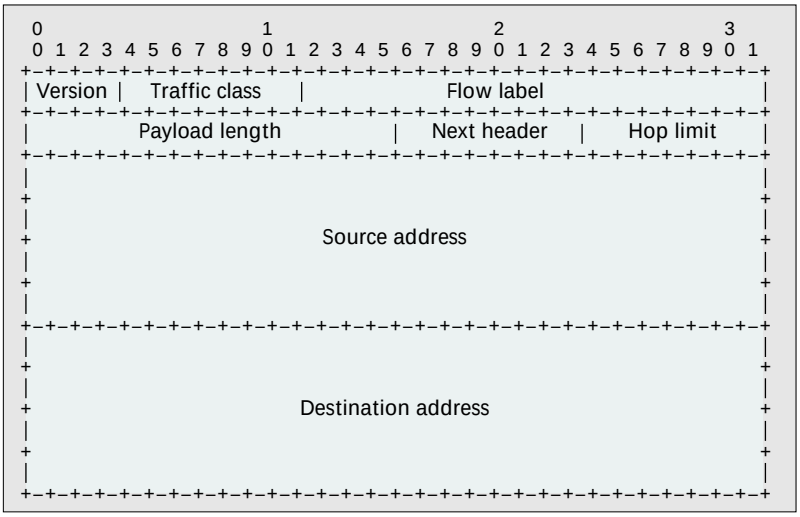

Figure 1. *The IPv6 header format.*

Major New Internet Protocols

In this section we present and discuss IPv6, RSVP, and multiprotocol label switching (MPLS) as potential candidates for improving the utility of the Internet.

Internet Protocol Version 6

An Overview of IPv6 — The 1990s witnessed an explosion in the growth of the Internet. The number of connected networks and users has increased dramatically, posing the threat of address space exhaustion. Meanwhile, a new sort of application appeared, characterized by real-time constraints and driven by the emergence of cheap digital multimedia technology, which puts significant new communication demands on the underlying networks. This has set a considerable challenge for the Internet, which was designed to support non-time-critical applications. Besides requiring an increase in network capacity (i.e., the communications bandwidth), some significant architectural enhancements were prompted as well.

This has triggered the birth of new protocols such as multicast (leading to the creation of the MBone) [4–6], Classless Interdomain Routing (CIDR) [7], RSVP [8], and Real-Time Transport Protocol (RTP) [9]. The features of the current version of the IP layer, IPv4, have been able to underpin these protocols for the past few years. However, it became evident that a more uniform and fundamentally better adapted solution to the new requirements was needed to provide a smooth evolution of the network.

IPv6 was then designed and adopted as the successor to IPv4, to occupy, of course, the central place in the entire Internet protocol architecture. As with IPv4, IPv6 [10] is still based on the key concept of a datagram. On the other hand, IPv6 exhibits the following changes from its predecessor:

- Expanded address capabilities: IPv6 uses 16-byte-long addresses; these extend the CIDR addressing hierarchy strategy and definitively overcome the scaling problem of IPv4 (which uses 4-byte-long addresses).
- New packet format: The packets are based on a simple header (Fig. 1). Also, the way header options are encoded has been totally rethought.
- Multicast support: IPv6 supports multicast as a native communication mode. Moreover, the addition of a scope field to multicast addresses improves the scalability of multicast routing.
- Flow labeling capability: This allows a sender to identify packets as being related to one another (i.e., belonging to the same traffic flow).
- Anycast support: Anycast is used to send a packet to any one member of a group of receivers (supposedly the "closest" in terms of the routing algorithm's unit of measurement).

• Authentication and privacy capabilities.

It is interesting to note how optional Internet-layer information is encoded in IPv6. Indeed, those options are encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in the packet. Such headers are called extension headers, each identified by a distinct "next header" value (Fig. 1). As a consequence, the "complete" header of an IPv6 packet can actually be seen as a "chain" of headers starting with the basic IPv6 header (Fig. 1) and followed by some (i.e., zero or more) extension headers.

So far, six extension headers have been defined:

- Hop-by-hop options header: Carries information that has to be examined and processed by each node on the packet's path, with the source and destination included.
- Routing header: Used by a source to list one or more routers to be visited by the packet en route to its destination. This is very similar to IPv4's source route option.
- Fragment header: Used when fragmentation is required. In IPv6, fragmentation can only be done at the source.
- Destination header: Used to carry information that needs to be examined and processed by the destination of the packet.
- Authentication header and encapsulation security payload header.

New extension headers can be defined if required.

Finally, we note that the traffic class field in Fig. 1 is labeled the Differentiated Services (DS) field in the DiffServ work.

IPv6 Improvements — The experience gained during the years of operating and evolving IPv4 is reflected in IPv6 through the incorporation of some very interesting design features.

By supporting multicast as a native communication mode, IPv6 takes a decisive step toward avoiding the problems of bandwidth consumption exhibited by the MBone [6]. Indeed, due to its implementation using tunnels between multicast routers (i.e., encapsulation of multicast packets in IPv4 packets), the MBone tends to replicate, several times, the same packet on its links. This is because a packet is replicated a number of times equal to the number of tunnels using the link. By avoiding the use of tunnels for multicasting (because multicast is "integrated" in the protocol), IPv6 will improve the support of multicast in the Internet. This will only happen when most of the routers in the Internet are based on IPv6, avoiding the need for tunneling IPv6 through IPv4 routes.

Besides better multicast routing, IPv6 also offers a very useful feature to improve control of multicast traffic: the scope field added to the multicast addresses. This allows precise control of where multicast packets are sent, and is very useful to avoid "leaks" of multicast packets on the Internet as well as ensure some security (e.g., making sure that a remote site cannot eavesdrop on a local teleconference). The same functionality can be obtained on the MBone today, but at the price of a difficult tuning of the IPv4 time to live (TTL) field. Also, the native support for anycast makes resource discovery a lot easier.

The IPv6 (basic) header format is simpler than that in IPv4. Indeed, some of the IPv4 header fields have been dropped (e.g., the header checksum) or made optional (e.g., fragmentation). This can only be beneficial to the performance achieved by IPv6, since the common-case processing cost (i.e., CPU instructions) of packet handling is reduced. In other words, packet forwarding is more efficient with IPv6.

Because of the simplification of the IPv6 header, a new way to deal with options had to be devised. This resulted in the extension headers already described. The good point about extension headers is that they allow less stringent limits on options than those imposed by IPv4 due to a lack of space in the IPv4 header. This is the case, for instance, with the source routing option which, in IPv4, limits the specification of the route to a maximum of nine hops. In IPv6 the limitation is 24 hops.

Fragmentation is a costly operation. Not only does it increase the overhead associated with the fragmented packet (because of the "replication" of the header in each fragment), but it also requires significant processing time at the fragmenting/reassembling nodes. By restricting fragmentation to being performed by source nodes only, IPv6 thus offloads the burden of fragmenting from potentially heavily loaded routers. It is worth noting that, given the widespread use of Ethernet LANs, this can lead to a network where it is unlikely to encounter packets bigger than 1500 bytes.

The flow labeling capability allows flow identification without layer violation (i.e., without combining fields from different protocol headers, e.g., IP and UDP headers). This not only enables fine-grained flow identification, but also simplifies the treatment of flows where security issues may require payload encryption at the network (i.e., IP) layer. Flow labeling also has an important role in QoS support since QoS in a network should be closely linked to the concept of flow (at least at the edge of the network). This is because QoS as required by multimedia communications cannot be defined on a datagram basis.

Shortcomings of IPv6 — Although we acknowledge that extension headers were introduced for flexibility when dealing with options as well as getting rid of most of IPv4's limitations, the whole idea of extension headers nonetheless introduces concerns. Since new extension headers as well as new options for each of the existing headers can be specified at any time, it seems almost impossible to design routers that will process all the extensions efficiently. Indeed, when a new extension or option is introduced, the only reasonable way to make an existing router understand it is by means of a software update. This, however, contradicts the trend to build high-speed routers using specialized hardware.

Furthermore, since each new extension or option will impose updates on routers' and end systems' software, we may find there is disparity from one node to another, since some will be updated and others not. Such disparity in the functions supported in the nodes may well render some of those extensions or options totally unusable (because the nodes will either discard or misprocess packets containing options or extensions they do not understand). At best, we may evolve toward a situation where most of the nodes in the Internet will only support the core extension headers and options specified in today's version of the IPv6 specification.

A summary of the two previous paragraphs is perhaps to say that extended systems are probably better than extensible ones, especially when considering large-scale networks.

The IPv6 specification says that "with one exception (the hop-by-hop option header), extension headers are not examined or processed by any node along a packet's delivery path, until the packet reaches the node identified in the destination address field of the IPv6 header." We would like to mention that when routing options are used (i.e., source routing), the destination of a packet changes several times on the way to the final destination, which will trigger the processing of the extension headers in the "intermediate" destination (i.e., routers). This may cause the corresponding packet to experience very long transmission delays.

It is worth noting that the concept of flow seems to somewhat contradict the concept of datagrams, and that simultaneous support for flows and datagrams by the same packet type inevitably leads to unnecessary overhead. When forwarding a packet in "datagram mode," the routers ignore the flow label. When forwarding a packet in "flow mode" (by ensuring that the different packets of a flow receive the same treatment from the network), a router should ignore the destination and source addresses of the packet.[4] This is illustrated by the

example of a telephone-quality audio flow (64 kb/s) where one packet is sent every 20 ms. Assuming the now typical use of RTP/UDP/IP for media transfer in the Internet, the header "information" would comprise 60 bytes (40 bytes of IPv6 header, 8 bytes of UDP header, and 12 bytes of RTP header); the packet payload would be 160 bytes. We would therefore have a per-packet overhead of about 27 percent, of which 53 percent is IPv6 addresses and 5 percent is the flow label.

With regard to the treatment packets of a given flow should receive, the IPv6 specification is rather elusive, stating that routers may (and thus may not) use flow labels to treat packets. It therefore appears that a source using a flow label will have no guarantee that the corresponding packets will receive special attention from the network. In other words, it cannot be assumed that IPv6 provides a communication scheme where the concept of flow has significance throughout the network (i.e., for nodes other than the source and destination).

The features of IPv6 (address size, packet format, etc.) render it incompatible with IPv4. In other words, in spite of the similarities in their name, IPv6 and IPv4 are two different protocols unable to interoperate. Therefore, a transition mechanism has had to be devised to facilitate and encourage the deployment of IPv6 [11]. This transition mechanism roughly specifies that:

- Each IPv6 router or end system has to provide a complete implementation of both versions (IPv4 and IPv6). This leads to a dual-stack architecture in every IPv6 node.
- IPv6 traffic will, when necessary, be carried over the IPv4 routing infrastructure using a tunneling technique (i.e., encapsulation of the IPv6 packets into IPv4 ones). This is likely to ruin the advantages of native support for multicast communication.

Finally, current Internet transport protocols (i.e., TCP and UDP) include the addresses from the IP header in their checksum computation. Since the size of IPv6 addresses is different from those of IPv4, these transport protocols have to be modified.

The Resource Reservation Protocol

Overview — RSVP is based on the concept of *session* [12]. A session is composed of at least one data flow and is defined in relation to a "destination" — more precisely as the triplet (destination address, destination port, protocol id). Since the destination address can be a multicast address, the destination can thus be either a group of receivers or a single receiver.

In RSVP, a flow is defined as any subset of the packets in a session, or, in other words, as a subset of the packets sent to a given destination. A flow is therefore simplex. Theoretically, the subset of packets making up a flow may be arbitrary, but in the current state of the RSVP specification, a flow is defined as the set of packets emitted from a given "source" (identified by the pair (source address, source port)).[5]

RSVP works as follows [12, 8].

Path messages are periodically[6] sent toward the destination and establish a "path state" per flow in the routers. Resv messages are periodically sent toward the sources, and establish the required reservations along the path followed by the data packets. The style of reservation in RSVP is thus receiver-oriented, since it is the receivers that initiate the requests for resources to be reserved.

In order to reduce the overhead associated with RSVP, any Path or Resv message that does change the states held by a router is not forwarded immediately by that router. Instead, each router periodically issues its own Path and Resv messages carrying information about the flows it holds.

A lifetime L is associated with each reserved resource. This timer is reset each time a Resv message confirms the use of the resource. If the timer expires, the resource is freed. This principle of resource management based on timers is called *soft state*. Soft state is also applied to the path state in the routers (in this case, the timer is reset upon reception of a Path message). By default, L is 2 min 37.5 s [12].

To improve RSVP responsiveness to network dynamics, a mechanism called *local repair* has been introduced. When an RSVP entity detects a change of route, it sends Path messages down the new route for the flows whose route has changed. When the downstream RSVP entity, situated at the junction of the old and new routes, receives these Path messages, it updates its path states accordingly and immediately sends a Resv message upstream along the new segment of the route for the corresponding flows.

Teardown messages (PathTear and ResvTear) are available for immediate release of the corresponding states (path state and reservations). Teardown requests can be initiated by a sender or receiver, or any intermediate RSVP router (upon state timeout or service preemption).

It is worth noting that all the messages described above are delivered unreliably: because of the protocol reliance on soft states, the concept of acknowledgment is not used in RSVP.

RSVP allows several styles of reservation: distinct resources may be assigned to given flows, while several flows may share some resources. The selected style for a given flow is expressed in the filter spec associated with that flow. There are three filter types:

- Wild-card filters: every flow of a session shares the associated resource
- Shared-explicit filters: explicitly identified flows share the same resource
- Fixed filters: ensure that one flow is granted the exclusive use of a resource

It is worth noting that a flow for which no resource has been reserved gets best-effort service from the routers.

Finally, in its current specification stage, RSVP does not influence the routing of packets (which are therefore routed by IPv4 or v6).

Positive Features of RSVP — RSVP has been designed to be able to operate across non-RSVP networks. It is extremely difficult (if not impossible) to guarantee end-to-end service in such a case. Nevertheless, this allows for a progressive deployment of the protocol associated with a steady improvement of the end-to-end best-effort service seen by flows exploiting RSVP in the part of the Internet where it is supported.

The soft state mechanism is a very simple self-stabilizing mechanism to keep the nodes of the network in a consistent state. It provides "natural" recovery form node crashes, as

[4] Ignoring both addresses when in flow mode implies hop-by-hop flow label translation to ensure a globally unique flow identification scheme throughout the network. Another way to achieve such a unique identification scheme is to let the source locally, and independently, select the flow label and use it in conjunction with the source address. It is important to note that in such a case, the source address is not used for any location information purposes, but rather as a unique bit pattern to build a globally unique flow identifier.

[5] This definition of a flow could, and should, be updated in future versions of the protocol to exploit the possibilities offered by the flow label field in the IPv6 header.

[6] Each period is chosen randomly in [R/2, 3R/2], with R = 30 s by default [12].

well as preventing resource leaks by reclaiming resources made obsolete by various conditions (e.g., route changes, loss of teardown messages, users leaving a multicast group without explicitly releasing resources).

Local repairs take care of establishing resources on new portions of a route after a route change. However, these cannot be relied on across a non-RSVP cloud, because routing changes that affect the egress path from such a cloud may not be detected by an RSVP node located just before the non-RSVP area.

Therefore, soft state (and its associated periodic messages) also provides an easy solution to route changes in non-RSVP areas of the network.

RSVP as a receiver-driven protocol scales to large numbers of participants in multicast groups. The reservation request from a receiver does not have to propagate all the way to the sender in most situations. If the reservation request encounters an existing reservation in one of the RSVP routers along the route which is equal to or greater than its own reservation request, it merges with the existing reservation at that router and does not travel any further.[7] This also allows for "incremental" reservations whereby some receivers behind a bottleneck can hold partial reservations and then regularly poll the network, hoping for completion of full reservations.

The different reservation styles proposed in RSVP tend to improve resource usage efficiency in the nodes of the network. For instance, shared reservations are well suited to scenarios where multiple sources are unlikely to transmit simultaneously (e.g., audio sources in conferencing applications), because, in such a case, the size of the shared reservation is essentially independent of the number of sources. It should be noted that shared reservations provide for an overall gain in reservation efficiency. Unless all the sources transmit with the same requirements, and the shared reservation is exactly equal to the requirements of a single source, resource waste may occur near the sources, while a resource gain occurs everywhere else in the network (compared to having simultaneous individual reservations for each source).

Shortcomings of RSVP — The operation of RSVP is based on periodic messages exchanged unreliably. This can result in possibly long establishment latencies, if RSVP messages are lost during the establishment phase of a reservation. This is because the average time to recover from such losses is equal to the message refresh period whose value is several tens of seconds. Several solutions have been proposed to improve the performance of RSVP at reservation establishment *without* increasing the subsequent steady-state overhead caused by the soft state mechanism on the flows [13, 14] (some of these proposals are being considered for inclusion in the specification of RSVP).

RSVP was designed to scale in terms of the size of the groups of receivers it can support on individual flows. However, the reservation model in RSVP can itself represent a threat to scalability of the protocol, especially in parts of the network where flow concentration is high (i.e., in the core of the Internet). It is so because this reservation model induces both state overhead and message overhead that are, at best, linear in terms of the number of sessions established. The state overhead resides in both the slow path of routers (reservation states in the RSVP daemon) and, more important, the fast (data) path of routers (filtering/classification, scheduling, and possibly policing states). Flow aggregation has been proposed to solve this state scalability problem [15, 16]. However, the techniques proposed for flow aggregation are often designed to work only within a single aggregation area, with the state overhead due to the aggregates proportional to the size of the aggregation area. We therefore believe that more research is needed to design aggregation techniques that could work efficiently across several (smaller) aggregation domains.

Most aggregation techniques do not attempt to reduce the message overhead caused by the periodic refresh of soft states. The message overhead consumes bandwidth and, in most cases (even with aggregation), CPU time in the routers. Furthermore, although soft state is a very simple mechanism, it has proved slow to react to some network conditions (e.g., node failures, route changes). Solutions have recently been proposed to these message overhead problems [13, 14]. These proposals are based on the principle of *hierarchical soft states* based on a soft state per node, while reverting to a per-flow soft state technique only when some conditions have been detected by the per-node soft state mechanism. Because an RSVP node usually has far fewer neighbor RSVP nodes than reservations, much smaller refresh periods can be afforded for the per-node soft state, which can improve the response of the protocol to specific network conditions such as node failures/reset. It should also be noted that, contrary to common belief, soft state is not a fault tolerance mechanism, because the automatic "reestablishment" it provides can fail (unless standby reservations have been put in place). We thus prefer the term *fault-resilient* to qualify the soft state mechanism.

The receiver-based approach, as used in RSVP (i.e., where reservation request and specification travel *upstream* toward the source), may not be well suited to all types of application. Indeed, some applications fit a model where it is the sender(s) that fix(es) the quality of transmission (e.g., Internet telephony, digital TV/VoD servers). This does not mean that with such applications the senders must be involved or aware each time a receiver joins the communication. Indeed, consider a scheme where the receiver triggers a request for a reservation (but without specifying the actual reservation) on the branch of the multicast tree on which it resides, and where the reservation message travels downstream from the closest router up the multicast tree knowing the requirements. This is an acceptable receiver-based approach since it does not directly involve the sender, who simply specified the communication requirements in its initial establishment message. We note that the sender could periodically retransmit the QoS requirement, which becomes a necessity if the routing protocol does not support "reverse path routing." Furthermore, the receiver-based scheme of RSVP leads to fairly static reservations, which in some cases can be wasteful. For instance, consider the case where sources of a multicast session use different coding schemes (giving different communication requirements) and are unlikely to be simultaneously active. A receiver that wants to receive the different data flows without any loss of quality will have to request a reservation matching the characteristics of the most demanding coding scheme, because it never knows when each sender is going to be active. In contrast, if the sender could propagate reservation messages downstream, the reservations could be updated whenever the sources are active.

Because data packets are routed in the network in the same way whether or not a reservation has been established for the corresponding flow, receivers (in a multicast group and not using RSVP) can get a "free ride" on the reservations established by their peers. Indeed, a receiver close (in a network sense) to another one which has requested a reservation can enjoy that reservation all the way from the source down to the point where the routes to the two receivers split. If the final subbranch of the multicast tree toward the first receiver follows lightly loaded links, that first receiver can see an acceptable end-to-end quality without having requested any reservation itself. Such "free riders" can only be prevented if the number of downstream receivers is available in multicast routers [17]. However, most multicast

[7] *Merging rules depend on reservation style; see [12] for details.*

routing protocols do not provide such a receiver count facility. Although this is not a significant technical problem, it could represent a challenge as soon as billing is considered [17].

Multiprotocol Label Switching

MPLS Background and Overview — The growth of the Internet has prompted the IT industry to look at mechanisms that improve the efficiency of packet forwarding. The bus architecture found within traditional routers fails to scale beyond a maximum load of about 1 Gb/s [18]. Gigabit routers have been developed to achieve speeds far greater than this by replacing the bus architecture with a switch fabric to interconnect various components within the router. Here, the switching fabric is used as a very fast interconnect, and is essentially "hidden" from the outside world, with the IP processing functionality maintained within the interfaces to the fabric.

The term *multilayer routing* covers approaches to the integration of layer 3 datagram forwarding and layer 2 switching that go beyond the use of the techniques found within gigabit routing/switching. The approach uses label lookups to allow more efficient packet classification, and the potential to engineer the network and manage the impact of data flows. A flurry of vendor-specific approaches to multilayer routing appeared between 1994 and 1997, including IP Switching [19], Cell Switch Router (CSR) [20], ARIS [21], Tag Switching [22], and IPSOFACTO [23]. The fact that these approaches were proprietary, and in the main produced incompatible solutions, led to the formation of the Internet Engineering Task Force (IETF) Multi Protocol Label Switching working group.

The MPLS working group is addressing the issues of the scalability of routing, the provision of more flexible routing services, increased performance, and more simplified integration of layer 3 routing and circuit-switching technologies, with the overall goal of providing a standard label-swapping architecture [24, 25].

Each MPLS packet has a header that is either encapsulated between the link layer and the network layer, or resides within an existing header, such as the virtual path/channel identifier (VPI/VCI) pair within asynchronous transfer mode (ATM). At most, the MPLS header will contain a label, TTL field, Class of Service (CoS) field, stack indicator, next header type indicator, and checksum.

MPLS defines a fundamental separation between the grouping of packets that are to be forwarded in the same manner (the forwarding equivalence classes, or FECs), and the labels used to mark the packets. This is purely to enhance the flexibility of the approach. At any one node, all packets within the same FEC could be mapped onto the same locally significant label (given that they have the same requirements). However, there are instances where one may wish to engineer the network in such a way that several different labels are used (e.g., when wishing to explicitly differentiate between streams). The assignment of a particular packet to an FEC is done once, at the entry point to the network. MPLS-capable routers (*label-switched routers*, LSRs) then use only the label and CoS field in order to make packet forwarding and classification decisions. Label merging is possible where multiple incoming labels are to receive the same FEC.

MPLS packets are able to carry a number of labels, organized in a last-in first-out stack. This can be useful in a number of instances, such as where two levels of routing are taking place across transit routing domains. Regardless of the existence of the hierarchy, in all instances the forwarding of a packet is based on the label at the top of the stack. In order for a packet to travel through a tunnel, the node at the transmitting side of the tunnel pushes a label relating to the tunnel onto the stack, and sends the packet to the next hop in the tunnel.

A collection of LSRs go together to make a *label-switched path* (LSP). Two options are defined for the selection of a route for a particular forwarding class. Hop-by-hop routing defines a process where each node independently decides the next hop of the route. Explicit routing is where a single node (often the ingress node of a path) specifies the route to be taken (in terms of several or all of the LSRs in the path). Explicit routing may be used to implement network policies, or allow traffic engineering in order to balance the traffic load.

There are two approaches to label path control. Independent path control means that LSRs are able to create label bindings and distribute these bindings to their peers independently. This is useful when bindings relate to information distributed by routing protocols, and means that nodes can begin to label switch before the completion of a path. Ordered path control means label binding only takes place if the node is the egress node for the particular FEC, or has received a label binding for that FEC from its next hop. This approach is used to ensure that a particular traffic class follows a path with a specified set of QoS properties.

There are three main approaches for identifying traffic to be switched. First, path creation can be control- or topology-driven, where labels are preassigned in relation to normal routing control traffic. Here, the network size dictates the load and bandwidth consumed by the assignment and distribution of label information. Second, request-based control traffic from protocols such as RSVP can trigger path creation relating to individual flows or traffic trunks. Here, the number of labels and computational overhead will depend entirely on the number of flows being supported. Finally, data-traffic-driven label assignment is where the arrival of data recognized as a flow activates label assignment and distribution on the fly. This approach implies that there will be latency while path setup takes place. Overheads in this case will be directly proportional to traffic patterns.

MPLS is able to work in an environment that uses any data link technology, connection-oriented and connectionless. MPLS also provides the potential for all traffic to be switched, but this depends on the granularity of label assignment, which again is flexible and depends on the approach used to identify traffic (discussed above). Labels may be assigned per address prefix (e.g., a destination network address prefix) or set of prefixes, and can also represent explicit routes. On a finer-grained level, labels can be defined per host route and also per user. At the lowest level, a label can represent a combined source and destination pair, and in the context of RSVP can also represent packets matching a particular filter specification.

MPLS needs a mechanism for distributing labels in order to set up paths. The architecture does not assume that there will be a single protocol (known as a *label distribution protocol*, LDP) to complete this task, but rather a number of approaches that can be selected depending on the required characteristics of the LSPs. Where paths relate to certain routes, label distribution could be piggybacked onto routing protocols [26]. Where labels are allocated to the packets of a specific flow, distribution can be included as part of the reservation protocol [27]. New protocols have been developed for general label distribution [28] and the support of explicitly routed paths [29]. MPLS label distribution requires reliability and the sequencing of messages that relate to a single FEC. While some approaches use protocols that sit directly over IP (thus implying they are unlikely to be able to meet these reliability requirements), a number of the defined LDPs solve this issue by operating over TCP.

Within the MPLS architecture, label distribution binding decisions are generally made by the downstream node, which then distributes the bindings in the upstream direction. This implies that the receiving node allocates the label. However, there are also instances (especially when considering multicast communications) where upstream allocation may also be useful. In terms of

the approach to state maintenance used within MPLS, a soft state mechanism is employed, implying that labels will require refreshing in order to avoid timeouts. Approaches to this include the MPLS peer keep-alive mechanism, and the timeout mechanisms inherent within routing and reservation protocols (in instances where they are used to carry out label distribution).

In terms of support for QoS, MPLS provides the CoS field which enables different service classes to be offered for individual labels. For more fine-grained QoS provisioning, the CoS field could be ignored, using a separate label for each class. In this instance, the label would represent both the forwarding and service classes. As noted earlier, MPLS is able to provide QoS support on a per-flow basis using either flow detection or request-based control traffic from protocols such as RSVP to trigger label assignment. More general QoS differentiation can be achieved by such means as label assignment on a per-user basis, and using more general traffic engineering techniques.

Positive Features of MPLS — MPLS and multilayer routing techniques in general allow efficient packet forwarding to enable high-speed data transfer. Although in the case of MPLS the link layer is not specified, the approaches all provide a scenario where it is possible to fully integrate and couple traditional datagram routing concepts with link-layer switching devices supported within the telecommunications industry. MPLS functionality is now being supported directly within hardware, with routing and switching mechanisms combined at the chip level in order to provide integration at high speeds, thus increasing its viability.

MPLS-capable devices are able to provide additional functionality beyond the best-effort packet forwarding found within a gigabit router. This flexibility means that in principle it is possible to support ideas such as QoS differentiation. The fundamental separation between forwarding class and label assignment provides a great deal of flexibility. While packets within a class are to be processed in the same way, this approach means that traffic can be engineered to varying extents.

Alone, IP does not lend itself to the idea of traffic engineering, that is, the ability to manage bandwidth and routes in order to provide equal loading of resources within the network. Until now, it has been reliant on other technologies (e.g., ATM) and associated encapsulation techniques in order to offer this functionality. MPLS provides support for traffic engineering through the deployment of constraint-based routing. Stemming from the idea of QoS routing, constraint-based routing not only provides routes that are able to meet the QoS requirements of a flow, but also considers other constraints including network policy and usage. Label distribution protocols supporting label switching for end-to-end constraint-based paths [29] allow traffic characteristics to be described in terms of peak rate and committed rate bandwidth constraints, along with a specified service granularity (which can be used to define the delay variation constraint).

Explicit routing (a subset of constraint-based routing) allows the specification of the route to be taken across the network. This is enabled within MPLS by allowing a label to represent a route, without the overhead of source routing found within normal IP forwarding (making it too resource-intensive for use in most circumstances). Different paths can be selected in order to allow traffic engineering to be carried out effectively, allowing network load to be balanced in a far more flexible manner than manually configuring virtual circuits (as with other primitive approaches to engineering IP traffic). The engineering of paths in such a way implies a simple mechanism for measuring traffic between edge network devices making use of an LSP.

In Internet service provider (ISP) environments where service differentiation is likely to mean users will be charged in terms of the network QoS exploited, the ability within the MPLS architecture to specify per-host and per-user label assignment is likely to be very useful for billing purposes.

Shortcomings of MPLS — MPLS essentially attempts to overlay connection-oriented concepts onto connectionless technologies. While providing several advantages, in a number of instances this approach reduces the overall flexibility of the IP protocol, and could be branded somewhat heavyweight.[8] Some of the conclusions that led to the research into multilayer routing, such as tthat routers are too slow or routing tables becoming too large, have been weakened by the appearance of fast and powerful gigabit routers.

The MPLS framework [25] and architecture [24] define a base-level label swapping technology. As shown within the previous sections, MPLS allows for traffic to be switched under different circumstances (topology-driven, flow-driven etc.), using different LDPs depending on the circumstances. While this implies that MPLS is flexible, it is likely to be applicable only within well-managed networks, where all components are able to provide support for MPLS and the individual distribution protocols in use.

While the label stack concept provides benefits, the idea of having packets carry a number of labels is likely to increase overheads, certainly in terms of making the MPLS header larger.

With topology-driven label assignment (where labels are allocated and distributed without reference to the traffic), a full mesh of labels will be established. The overhead of this approach is essentially relative to the size of the network, and has the potential to use a vast number of labels. This can be a large overhead in instances when labels are allocated to routes where very little traffic is flowing.

The current MPLS architecture and framework specifications have left the topic of multicast as an area for further study.

In terms of the provision of varying levels of QoS, MPLS poses a number of issues.

Label assignment based on support for traffic flows will require a path to be put in place the moment the flow is detected, therefore implying that there will be some latency prior to a full path being in place. In this instance, the overhead will increase in relation to the number of flows being supported and the duration of the flows. Label assignment in order to support short flows implies a large overhead. When label distribution is included as part of a reservation protocol (e.g., RSVP), the overheads and scalability of such a protocol must also be considered.

The ordered and independent control of labeled paths (described earlier) are said to be compatible approaches to path setup. However, when they interoperate the overall behavior can only be described as independent because, to ensure QoS, ordered control must be used entirely from ingress to egress node.

LDPs must work in a reliable manner given that the loss of a control message in this instance could cause a delay in the establishment of a label path. This constitutes a serious impediment to the support of critical applications. As mentioned earlier, the use of TCP with a number of LDPs offers the necessary reliability. In the case of flow-based label assignment and the use of RSVP, reliable transmission of the LDP information is not guaranteed due to the use of UDP.

The ability of MPLS to support a number of link-layer technologies provides a high degree of flexibility. However, in terms of the provision of connections with a level of associated QoS, mechanisms are required to ensure that the QoS specified for an LSP is maintained by the underlying link

[8] *The authors recognize that IP is not the only network layer protocol supported by MPLS.*

layer. This may not be possible in some instances (e.g., with an Ethernet) where firm guarantees cannot be made (because of the inherent nature of the technology). Where ATM technology is used with MPLS, in most instances the LDP acts as the ATM signaling protocol. This implies that a low-level control protocol is required which is able to configure connections with defined levels of QoS. While work is progressing in this area within the IETF GSMP Working Group [30, 31], widescale support for this type of protocol by major switch vendors is not yet evident.

Discussion and Conclusion

We now discuss several possible ways in which the different proposals presented earlier can be deployed and integrated in the Internet. We look at the different scenarios from an end-to-end service perspective, noting that the spectrum of quality provided by these services will ultimately determine the Internet's suitability as the core of a future global telecommunications infrastructure.

IPv6 does not fit well in such a discussion, because the motivations to develop a new Internet Protocol do not directly stem from the need to develop and support new services. The main factors that will drive and pace the deployment of IPv6 are, in our opinion, the "pressures" on current Internet address space and to a lesser extent the increasing demand for multicast communications.

From the proposals discussed in this article, DiffServ appears to be the most appropriate and thus the most attractive. Our discussion earlier led us to the conclusion that in its simplest and most general form, DiffServ can provide relative differentiation of service on delays and/or error rates without explicit bounds. DiffServ is therefore able to provide a range of increasingly better best-effort services. Since no bounds are guaranteed in such services, traffic conditioning becomes optional (even at network boundaries), which could further simplify the operation of the network and thus help retain the Internet's original simple philosophy. We believe that such a range of differentiated services would not need to be broad to be extremely useful since many applications can react to a wide range of network conditions. The idea here is that if the traffic generated by these applications were segregated into a few classes, the service provided by each class could be tailored to the support of a given type of application[9] (e.g., delay-sensitive, error-sensitive, or insensitive). The application would perform "better" than if competing with all others. Of course, insensitive (i.e., "pure" best-effort or asynchronous) applications would probably see worse performance than if the service space remained flat, but this does not really matter as the level of satisfaction a user gets from these applications is not related to performance. Pricing would have to be used to balance traffic over the different classes.

This idea of a range of best-effort services is re-enforced by a recent study [32], which suggests that most of the congestion in the Internet occurs at the edges, while the core is fairly underused. Indeed, as long as a dramatic shift in traffic mix does not occur, a simple DiffServ network could also probably support most adaptive multimedia applications in a satisfactory way.

We also concluded that under some traffic restrictions, the simple DiffServ architecture could provide quantitative services by essentially emulating leased-lines. These services are particularly well suited to business applications such as Virtual Private Networks (VPNs). Because the restrictions applicable to these services are "directional," the use of MPLS in supporting them promises to facilitate traffic engineering and hence improve resource usage while possibly improving robustness. Therefore, although the use of MPLS is not mandatory, it is likely that the combination of DiffServ/MPLS will play an important role in the provision of some form of quantitative service guarantees in the Internet.

The way that a DiffServ leased-line is shared between users (i.e., the hosts of the stub-network on the ingress side of the line) is left to the discretion of these users. A very flexible and efficient way to provide end-to-end guarantees is by using IntServ in the stub-networks on each side of the DiffServ leased-line. Doing so allows the users to signal their requirements locally, which enables admission control at the entrance of the DiffServ leased-line, whose characteristics are known. It should however be noted that applications not requiring any specific guarantees may still benefit from the leased-line by sharing any non-reserved part of the bandwidth.

The restrictions of directionality of a leased-line allow the previously described integration of IntServ and DiffServ to provide end-to-end guarantees in a simple way. As soon as these restrictions are relaxed, we believe that such an integration model can no longer enforce guarantees, unless complex signaling is used within the DiffServ area. We also believe that introducing such complex functions in DiffServ would spoil the major attractive features of DiffServ, namely its simplicity and close adhesion to the Internet's original philosophy. Furthermore, even though RSVP (the likely signaling mechanism in IntServ) is receiver-oriented, while DiffServ is sender-oriented, application users may not be willing to accept that their end-to-end requirements cannot be fully met. This could be the case when the bandwidth brokers in the DiffServ area do not have a precise view of their network. If DiffServ signaling can ensure that such deficiencies will be avoided, we believe its complexity will match that of IntServ signaling, and therefore we do not see any point in "reinventing the wheel." IntServ (with aggregation in the core) seems adequate to fulfill the needs of general applications with quantitative requirements and was indeed designed to provide end-to-end QoS. We therefore advocate the use of IntServ with aggregation for general applications (i.e., applications without directionality limits) that have quantitative requirements (e.g., some interactive real-time applications such as tele-education). This leads to a model where DiffServ and IntServ are both supported in the routers, especially in the core. In this model, the flexibility of the best-effort services provided by DiffServ seems ideal to provide extended and extensible best-effort services for the IntServ architecture. Furthermore, this use of DiffServ to provide a range of differential best-effort services (in terms of delay and error rates) lends itself to incremental deployment. Indeed, as long as best-effort service differentiation has been provided in parts of the network, it is guaranteed to be retained end to end. This is so because delays and error rates have additional and multiplicative properties, respectively, and the delay and error rate experienced by packets in a node providing a flat best-effort service are the same.

Because of the strict guarantees, end-to-end guaranteed-service IntServ should command a higher price than controlled load services, which, in turn, should command more than the "best" best-effort provision of simple DiffServ. This leads to a model where applications, depending on the severity of their requirements, could try successive DiffServ classes until they get acceptable performance, only using an end-to-end IntServ path in cases where the best DiffServ class is not sufficient. The important point of this model is that while trying successive best-effort services provided by simple DiffServ, no performance indication (never mind guarantee) is provided. Many applica-

[9] *The type of an application depends on the context in which it is being used. For instance, commercial Web browsing could be considered delay-sensitive, while Web browsing could be considered insensitive, depending on users' willingness to pay for improved services.*

tions can cope with such a lack of performance information, but some applications will require better guarantees from the network. These latter applications will have to use either guaranteed service or controlled load service to operate satisfactorily.

In this article we have discussed the Internet as a candidate to provide the global telecommunication infrastructure of the future. A major requirement for such an infrastructure is the ability to support new distributed, interactive multimedia applications (the critical applications) that require strict QoS guarantees. A second key requirement is the ability to cope with hugely increased traffic demands, which will be generated by both current best-effort applications and new multimedia applications. The elastic applications, where the burden lies with the applications themselves to adapt to variations in network service quality, raise a further requirement, namely the provision of a better best-effort service from the network where the QoS is more predictable and consistent. For best-effort and elastic applications, the current evolution of the Internet in terms of more efficient routing and allowing users to pay more for a differentiated service is highly welcome. However, for critical applications further development of a guaranteed service within the IntServ architecture continues to be elusive.

In this article we have discussed how the combination of IntServ and DiffServ can contribute to strategic approaches for providing appropriate compromise levels of QoS for mixtures of applications. We have also analyzed new members of the IP family, specifically IPv6, RSVP, and MPLS, and deduced that they have both significant benefits and deficiencies, the deficiencies being largely that critical applications are not ultimately supported. Telecommunication operators and suppliers should understand the benefits as well as the flaws of the new-generation IP family. If they are interested in fully supporting critical applications in the future, they should be aware that this support will not be realized until further research and development have been undertaken. Meanwhile, they might wish to adopt the Internet as the basis for a global information infrastructure, but accept that for now, and for a while to come, their applications will have to continue to adapt to the deficiencies of the network and its protocols.

Acknowledgments

We would like to thank Jon Crowcroft, Domenico Ferrari, Paul Kirkby, and Peter Newman for their comments on an earlier version of this article. We also thank the anonymous referees whose comments have substantially improved this final version.

References

[1] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
[2] A. Mankin et al., "Resource ReSerVation Protocol (RSVP) Version 1 Applicability Statement: Some Guidelines on Deployment," RFC 2208, Sept. 1997.
[3] D. Black et al., "An Architecture for Differentiated Services," RFC 2475, Dec. 1998.
[4] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Comp. Sys.*, vol. 8, no. 2, May 1990, pp. 85–110.
[5] S. Deering *et al.*, "An Architecture for Wide-Area Multicast Routing," *ACM Comp. Commun. Rev.*, vol. 24, no. 4, Oct. 1994, pp. 126–35.
[6] H. Eriksson, "MBONE: The Multicast Backbone," *Commun. ACM*, Aug. 1994, vol. 37, no. 8, pp. 54–60.
[7] V. Fuller, T. Li, and J. Yu, "Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy," RFC 1519, Sept. 1993.
[8] L. Zhang *et al.*, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, vol. 7, Sept. 1993, pp. 8–18.
[9] H. Schulzrinne *et al.*, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, Jan. 1996.
[10] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 1883, Dec. 1995.
[11] R. Gillian and E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers," RFC 1933, Apr. 1996.
[12] R. Braden *et al.*, "Resource ReSerVation Protocol (RSVP) — Version 1 Functional Specification," RFC 2205, Sept. 1997.
[13] L. Mathy *et al.*, "Improving RSVP for Better Support of Internet Multimedia Communications," *Proc. IEEE Multimedia Sys. '99*, Florence, Italy, June 1999, pp. 102–6.
[14] L. Berger, D.-H. Gan, and G. Swallow, "RSVP Refresh Reduction Extensions," IETF, RSVP Working Group, draft-berger-rsvp-refresh-reduct-00, Mar. 1999, work in progress.
[15] J. Schmitt *et al.*, "Aggregation of Guaranteed Service Flows," *Proc. IWQoS '99*, London, U.K., June 1999, pp. 147–55.
[16] S. Berson and S. Vincent, "Aggregation of Internet Integrated Services State," IETF, RSVP Working Group, draft-berson-rsvp-aggregation-00, Aug. 1998, work in progress.
[17] S. Herzog, S. Shenker, and D. Estrin, "Sharing The "Cost" of Multimedia Trees: An Axiomatic Analysis," *IEEE/ACM Trans. Net.*, 1997, vol. 5, no. 6, pp. 847–60.
[18] P. Newman *et al.*, "IP Switching and Gigabit Routers," *IEEE Commun. Mag.*, 1997, vol. 35, no. 1, pp. 64–69.
[19] P. Newman, G. Minshall, and T. Lyon, "IP Switching: ATM Under IP," *IEEE/ACM Trans. Net.*, vol. 6, no. 2, Apr. 1998, pp. 117–29.
[20] Y. Katsube, K. Nagami, and H. Esaki, "Toshiba's Router Architecture Extensions for ATM : Overview," RFC 2098, Feb. 1997.
[21] A. Viswanathan *et al.*, "Aggregate Route-Based IP Switching (ARIS)," IBM tech. rep. TR29.2353, Feb. 1998.
[22] Y. Rekhter *et al.*, "Tag Switching Architecture Overview," *Proc. IEEE 1997*, vol. 85, no. 12, pp. 1973–83.
[23] A. Acharya, R. Dighe, and F. Ansari, "A Framework for IP Switching over Fast ATM Cell Transport (IPSOFACTO)," *Conf. Broadband Networking Tech.*, vol. 3233, ch. 36, Dallas, TX, Nov. 1997, pp. 20–28.
[24] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture" IETF, MPLS Working Group, draft-ietf-mpls-arch-05.txt, Apr. 1999, work in progress.
[25] R. Callon *et al.*, "A Framework for Multiprotocol Label Switching," IETF, MPLS Working Group, draft-ietf-mpls-framework-03.txt, June 1999, work in progress.
[26] Y. Rekhter and E. Rosen, "Carrying Label Information in BGP-4," IETF, MPLS Working Group, draft-ietf-mpls-bgp4-mpls-02.txt, Feb. 1999, work in progress.
[27] B. Davie *et al.*, "Use of Label Switching with RSVP," IETF, MPLS Working Group, draft-ietf-mpls-rsvp-00.txt, Mar. 1998, work in progress.
[28] L. Andersson et al., "LDP Specification," IETF, MPLS Working Group, draft-ietf-mpls-ldp-5.txt, June 1999, work in progress.
[29] B. Jamoussi "Constraint-Based LSP Setup using LDP," IETF, MPLS Working Group, draft-ietf-mpls-cr-ldp-01.txt, Feb. 1999, work in progress.
[30] T. Worster, F. Hellstrand, and A. Doria, "A QoS Model for GSMP," IETF, GSMP Working Group, draft-worster-gsmp-qos-00.txt, Feb. 1999, work in progress.
[31] C. Adams, A. Lazar, and M. Nandikesen, "The qGSMP Protocol, IETF, GSMP Working Group, draft-adam-gsmp-qgsmp-00.txt, June 1999, work in progress.
[32] A. Odlyzko, "The Internet and Other Networks: Utilization Rates and their Implications," *Proc. 26th Telecommun. Policy Res. Conf.*, Oct. 1998.

Biographies

LAURENT MATHY (laurent@comp.lancs.ac.uk) graduated in electrical engineering from the University of Liege, Belgium, in June 1993, and was awarded his Ph.D. in computer science from Lancaster University, England, in January 2000. He is currently a research lecturer in the Computing Department at Lancaster University. He spent the 1995–1996 academic year at the University of British Columbia, Vancouver, Canada, as a visiting scholar. His research interests include integrated communication architectures, services and protocols for multimedia communications requiring QoS and group communication support, programmable networks, and e-commerce security.

CHRISTOPHER EDWARDS (ce@comp.lancs.ac.uk) graduated from Liverpool John Moores University in 1995 with a degree in computer studies. Following this, he joined the Computing Department of Lancaster University as a research student. During 1998 and 1999, he worked as a research assistant on the European ACTS Project Provision of an Enhanced Transport by Exploiting Reservation in IP and ATM Networks (PeterPan), looking at various issues concerning IP/ATM integration. He was awarded his Ph.D. in computer science in March 2000 in the area of open network control. He is currently working as a research assistant on the Lancaster and Microsoft Active Research Collaboration (LandMARC) project. His research interests include multimedia communications, future approaches to signaling and network control, and support for QoS within the Internet.

DAVID HUTCHISON (dh@comp.lancs.ac.uk) is professor of computing at Lancaster University and has worked in the areas of computer communications and distributed systems for the past 15 years. He has completed many U.K. and European-funded research contracts and published over 120 papers as well as written and edited books in these areas. He assists the U.K. EPSRC on several aspects of their research program planning and is a technical expert adviser to the Office of Telecommunications (OFTEL). He is a technical expert and auditor for the European Commission in the areas of telecommunications and telematics, and represents the United Kingdom on the management committee of the COST264 project on multimedia group communication, and the COST263 project on quality of Internet services. The main theme of his current research is architecture, services, and protocols for distributed multimedia systems, including QoS for the Internet, and multimedia content description and extraction.