Issues in the Design of a New Network Protocol*

Mikael Degermark[1] and Stephen Pink[1,2]

{micke, steve}@cdt.luth.se

[1] CDT/Department of Computer Science, Luleå University, S-971 87 Luleå, Sweden
[2] Swedish Institute of Computer Science, PO Box 1263, S-164 28 Kista, Sweden

Abstract. We describe some of the issues in the design of a new packet switched network protocol. Adaptation to various network technologies along the dimensions of speed, error model, robustness, etc., is a goal for this new protocol. We look at the adaption in size of the packet header to the speed and robustness of the underlying network to allow efficient communication on low-speed wireless networks, for example. We also explore issues in resource reservation and multicast for real-time multimedia, the notion of a network "flow", a hybrid of datagrams and virtual circuits, and suggest common solutions for both mobile and multicast routing. The authors are engaged in the design of a network protocol, NP++, whose goal is flexibility over a wide dynamic range of speeds and varying kinds of hardware switching elements.

1 Introduction

The last quarter century has produced mainly two sorts of packet-switched networks: virtual circuit and datagram. Virtual circuit networks emulate circuit switching by setting up a permanent path through a network for packets to travel during the time of a connection. Datagram networks allow packets to be routed along any available path such that no two packets have to follow the same path in the network. There are clearly advantages and disadvantages to both of these schemes. In an effort to provide the most flexible protocol for next-generation networks, we are developing a new network protocol that we call NP++. This paper will describe some of the issues in the design of this new protocol.

A new network protocol must be flexible enough to operate over a large variety of communication technologies. It must be designed for flexibility in the face of networks with widely different speeds, latencies, error characteristics, packet formats, abilities to provide quality of service, etc. A new network protocol should build on the core properties of both datagram and virtual circuit networks, as well as on the the properties of their hybrids.

* This work was supported by a grant from the Centre for Distance Spanning Technology (CDT), Luleå, Sweden, and a grant from Ericsson Radio Systems AB

2 High *and* Low Speed Networks

The new generation of computer networks can be characterized as having a wide dynamic range of properties. Speed, latency, robustness in the face of errors, and other properties, vary much more among today's networks than last generation's. New high speed fiber optic networks are being developed at the same time as relatively low speed radio channels. Such high speed networks exhibit very low error rates especially in comparison with lossy wireless links.

Today's network protocols are being designed mainly with high speed links in mind. This is generally a good guiding principle since even low speed radio channels will be getting faster. But protocol designers should also realize that the *dynamic range* across the set of new networks will also be increasing. Thus, flexibility to adapt to both high and low speed networks, high as well as low delay paths, and high and low error rates, should be the design principle for a next generation network protocol.

A new network protocol should adapt to such a wide dynamic range of computer networks. Borrowing from a number of traditions in the history of packet switching protocols, the new protocol should take advantage of the most successful features of both datagram virtual circuit protocols with the goal of minimizing their disadvantages. For example, where connections and state in the network make sense for communication, the new protocol should maintain a virtual circuit path between sender and receivers; in other situations, it should use datagrams as the main mode of packet transmission. Similarly, on networks with low bandwidth–delay products, a new protocol should rely on end-to-end mechanisms at the transport layer to control congestion. On high bandwidth–delay product networks, hop-by-hop congestion control should be provided. This is because the minimum granularity for the end-to-end control of congestion in a network is one round-trip time, and the amount and rate of change of data in a high bandwidth–delay product network may be too large to control during this time.

A key idea in NP++ is that a packet header (or trailer) is seen as a set of independent fields which can all be sent with each piece of data, or can be sent separately from other header fields more or less frequently. Thus, by removing large addresses from packets and assigning small, temporary connection identifiers, header compression can be achieved on a low speed link with soft state that is refreshed by the arrival of address field chunks sent less frequently than the data [10, 12]. On a high speed link the small connection identifier may speed up forwarding as no routing lookup is necessary.

3 Network Hardware Architectures

Another dimension in which NP++ is designed to be flexible is in its adaptability to various network hardware architectures. Network switches and routers are being developed in rapid succession to provide high performance switching, quality of service such as bounded delay, etc. New hardware switch designs often

require changes to packet formats, forcing redesign of the protocol when new switch architectures are deployed. Advances in VLSI design methods, for example, have enabled the production of some low-cost, high speed switching fabrics for small fixed length packets (cells). Thus, it has been argued that today's ATM cell switches offer the best price/performance for high speed packet switching. On the horizon however are new designs and prototypes for IP routers [19] that will provide cost-effective routing of variable-length packets at multi-gigabits per second, where large packets increase the total throughput.

The point is that new network hardware architectures are being developed at an increasing rate and we should not have to specify, negotiate, deploy or even have to *choose* new network protocols each time next-generation network hardware is introduced. So, it is important for a flexible network protocol to adapt to new hardware in the network. We are designing a network protocol that can be either *switched* as fixed size cells or *routed* as variable length packets depending on the hardware of choice in the network. For this to be the case, the new protocol must be able to fragment and reassemble itself in the network when necessary [15].

4 Datagrams and Virtual Circuits

This section describes the two dominant ways of designing protocols for packet switched networks: *datagrams* as in the IP protocol used in the Internet, and *virtual circuits* as in ATM.

When a source is sending packets into the network, it needs to specify the source and destination of the packets. With datagrams this is done on a per-packet basis and with virtual circuits it is done once per connection. The source address is needed for error reporting, some forms of multicast forwarding, and perhaps cost allocation. The destination address is needed for locating and/or specifying (as in the case of a multicast address) the intended receiver(s).

4.1 Datagrams

Datagram protocols place full addresses in every packet, which means that routers do not have to keep any *inter-packet state* for routing purposes[3]. This has obvious advantages in terms of robustness and simplicity since packet forwarding state need not be maintained in the network. For example, if the path a stream of packets is following through the network fails, packets will automatically detour around the damaged part as soon as the dynamic routing protocols find a viable path.

If the address space is large, however, the lookup required to find the forwarding information can be costly. Today the size of a backbone Internet routing

[3] Some datagram networks use source routing, where each packet contains the complete list of routers the packet should pass through on its way through the network. We do not consider such networks here.

table is around 35,000 routing entries.[4]

Moreover, routing information is not the only kind of information that may be needed to forward a packet. If some packets need special treatment, for example, to get acceptable delay through the network for real-time voice and video, routers may have to maintain and access forwarding state for those packets. For each packet, a router must determine if the packet needs special treatment and, if it does, the router must access the forwarding parameters for that kind of packet.

Current work in the IETF on Controlled Load Service [28] is attempting to reduce the amount of forwarding state needed by lumping packets together into *classes* of packets that are treated in the same way. Such aggregation does not reduce the lookup time, however, since the size of the lookup domain is not decreased. Classification of packets is typically done on the basis of addresses, transport protocol, and port numbers. Aggregation of the form discussed in [28] reduces the amount of forwarding state by sharing it, but does not help to simplify the procedure of deciding what class a packet belongs to. In a general classification scheme, where it is possible to specify special treatment by arbitrary combinations of addresses and port numbers, classification can be an expensive operation.

4.2 Virtual Circuits

Network protocols built on the virtual circuit model do not put full addresses in each packet. Instead, a connection phase occurs before data is sent in which addresses are given to the network and routing and forwarding state is established in the network switches. The switch normally uses a small integer per-hop to access the forwarding state, and this index is made consistent at switches on both sides of the hop. This index is sometimes called a *virtual circuit identifier, (VCI),* and is placed in each data packet belonging to the connection. In addition, there has to be an end-to-end virtual circuit identifier that uniquely identifies the circuit across all of its hops, as in ST-2 [26]. These two identifiers, along with the per-switch identifiers that must be present in order to negotiate the VCI between any two hops, mean that virtual circuit networks need at least three levels of naming. An advantage of virtual circuits, on the other hand, is that it reduces forwarding state lookups to simple table lookups based on the VCI. This advantage works especially well when there are many packets sent per connection.

One problem for virtual circuits is that one must establish a connection before transmitting any data, even in the case where only a single packet needs to be sent. This adds complexity and delay. To alleviate this problem for virtual circuits, various caching schemes to reuse existing connections and reduce setup delay have been developed. Another problem is that the forwarding state is typically *hard* and must be torn down by explicit control messages, which adds

[4] Statistics of the routing table at the Mae East NAP are available from `http://compute.merit.edu/stats/mae-east/routing-table` . Its size (prefix count) on Aug 20th 1996 was 34617 entries, and on Oct 10th it was 38515 entries.

further complexity. *Soft-state* [5], where timers and refresh messages keep state in place avoiding tear-down messages, has been developed for networks where hard-state is a problem.

In large networks, routing tables are usually maintained by dynamic routing protocols. Changing network conditions can cause such protocols to modify the routing table. With virtual circuits such changes do not affect the quality of service since the path through the network is pinned down. With datagrams, however, changes in the routing table causes packets to follow different paths through the network, possibly resulting in changes in service quality. Routing changes are not uncommon in the Internet. A recent study [20] shows that the routing instability in the Internet has increased slightly between December 1994 and December 1995. When the network is actually damaged, on the other hand, virtual circuit networks require special mechanisms if connections through the damaged part are to be recreated on paths circumventing the damage. If such special mechanisms are not used, the failure of a single router along the path of a connection will destroy the connection. With datagrams, all paths between sender and receiver must fail before communication fails.

4.3 Datagrams or Virtual Circuits?

With both datagrams and virtual circuits a router needs to access a routing table to be able to figure out where to forward packets. In the datagram case, this routing table may have to be accessed once per packet. For virtual circuits, the routing table is accessed when a connection is established. If all IP addresses were given out systematically such that all addresses in a certain part of the Internet had identical prefixes [16, 24], the number of entries in routing tables would be smaller than today, probably in the hundreds instead of in the tens of thousands.

The cost of a routing lookup would thus be significantly reduced and there would be one less advantage of virtual circuits over datagrams. The routing lookup would cease to be the bottleneck of router processing. Changing IP addresses in the current Internet to achieve such small routing tables is not an option. The required changes to millions of computers and routers around the globe are administratively and politically impossible. However, new datagram network protocols, for example IPv6, are designed to allow simple renumbering of addresses and can achieve such fast routing lookups. In such networks the virtual circuit model offers no advantages to traditional elastic data traffic and it is clear that such traffic is best served by the datagram model.

To support real-time traffic, future routers are likely to maintain many entries of forwarding state related to quality of service. Thus, it is packet classification that is the processing bottleneck instead of the routing lookup. The feature of virtual circuits to associate forwarding state with a small identifier in each packet can be important to allow fast lookup of such forwarding state. Traditional bulk data traffic, however, does not need quality of service handling and can be forwarded with less processing as only the fast routing lookup is needed.

In a flexible network protocol all packets should be able to carry complete addresses. They can also carry an identifier to find appropriate forwarding state quickly if such state has been established. The addresses do not need to be present if forwarding state is present. In this manner the network protocol syntax, exemplified by the fields contained in the packet header or trailer, can be used for datagrams, virtual circuits or an intermediate form such as soft state-based "flows", where addresses are sent occasionally to refresh forwarding state that might otherwise time out.

5 Flows and Packets

The concept of a network "flow" has been introduced into contemporary discourse. Generally, a network flow is a stream of packets from one source to one or many (in the case of multicast) destinations. Packet streams that require different qualities of service are normally distinguished into different flows even if they have the same source and destination(s); the idea is that packets belong to the same flow if they are to be forwarded along the same path with the same service from the network. Obviously, a virtual circuit is also a network flow. In the Internet, unicast or multicast IP packets from source to destinations have also been termed "flows," especially when these packets contain real-time continuous media like voice and video.

In the next-generation IP protocol, IPv6 [9], there is a new field called the Flow Label that provides at least syntactical support for flows in the Internet. One proposed use for the IPv6 Flow Label is to provide at least one part of a name (along with the source IP address) for an Internet resource reservation made by the RSVP resource reservation protocol [29]. Thus, even in the Internet, one may be able to make reservations on behalf of flows. These reservations are *not* made using classical hard-state based signaling protocols (although they could be). RSVP uses soft-state in hosts and routers to provide flow-based resource reservations from the receiver upstream towards the source of the flow.

The IPv6 flow label is a 24-bit field that is autonomously chosen by the source and is guaranteed to be unique only in combination with the source address. All packets that should receive the same service from the network are labeled with the same flow label by the source. The flow label is meant to simplify classification of IP packets and avoid the need to parse the whole packet to look at port numbers in the transport protocol header. Such parsing can be costly for IPv6 because the header is composed of a list of sub-headers that must be traversed before the transport protocol header can be examined.

The way the flow labels are designed is typical for the Internet architecture; it avoids complexity in the network. Since sources autonomously choose flow labels, no signalling is needed for this purpose. The drawback is that flow labels are not unique unless combined with the source address, and thus cannot be used by themselves to access forwarding information. In virtual circuit networks the VCI is used instead of flow label/source address, but then a signalling protocol is needed to select VCIs.

With some caveats, such as the requirement that the source address must be included with the flow label to individuate a flow, it should be clear that the Internet has borrowed concepts from classical virtual circuit networks to integrate new real time services with traditional Internet services. It is arguable then that IPv6 and RSVP are steps along the way to a hybridization of the Internet. The design of a new network protocol should benefit from the experiences of this hybridization. An aim of a new protocol should be to incorporate both hard and soft-state resource reservation into the network as well as providing support for pure datagram routing.

6 Adapting Headers to Networks

A dominant assumption of packet-switched protocols is that the header (or trailer of a packet) contains fields that are always sent together with each packet. If we look at the fields of a header as separate pieces of control information that can be sent with varying frequencies, we can derive a more flexible network protocol. This principle could also be applied to protocols that are *now* being specified as standards such as IPv6.

For example, as address size grows larger, as in IPv6, or as the need increases to include more than just the source and destination addresses in a packet (for tunneling, etc.), a good use of resources might be to take the large addresses out of some of the packets and substitute a smaller identifier. Soft-state could be established at a downstream switch, with the larger addresses sent downstream periodically to refresh the state. This can provide a new kind of header compression for links in a path that have relatively low-bandwidth. Our design for IPv6 header compression [13], uses this principle to achieve soft-state-based header compression for UDP/IP. On links that have high bandwidth, i.e., where bandwidth is not the dominant factor in the delay of the packet through the switch, the whole header can be reassembled and sent as a unit with the data to provide higher robustness. Indeed, if it is the processing power of the switches that is the forwarding bottleneck, one could add more fields to the header, as suggested in [4], to decrease the processing time in the switch. This is, in effect, the inverse of header compression.

Another example of adapting headers to networks is the use of *header implication* such as can be done when layering IPv6 over ATM. Since the flow label in IPv6 is the same size of the ATM Virtual Circuit Identifier (VCI), it is plausible to map a flow label onto a single VCI. For the common case, when the IPv6 header does not change between the packets in a flow, the IPv6 header need not be sent at all, since it is implied by the ATM VCI and can be recreated on the other side of the IP hop. IP would then present very little protocol overhead from the point of view of the ATM network. This would be especially useful when sending IP traffic over wireless ATM. Moreover, the scheme easily allows quality of service to be preserved over the hop for real time multimedia, if the ATM network provides the quality needed by the IP flow.

7 Switching at one layer only

The Ipsilon flow switching system [18] for switching IP packets in an ATM network is similar to the header implication scheme outlined in the previous section, except that the IP header is not removed. There, a switch that detects an incoming IP network flow[5] that should be switched and forwarded in the same way tells the upstream node to use a separate VCI for packets in that flow. The ATM cells that arrive on that VCI can then be forwarded directly without reassembling the IP packet and examining the IP header. The Ipsilon technique could be used to speed up datagram forwarding over any virtual circuit network technology. For it to work well, it is important that the recognition of network flows is quick and reliable. In the network protocol, the IPv6 Flow Label provides the needed syntactic support.

Ideas similar to Ipsilon's approach for speeding up IP datagram forwarding are now under development in other parts of the Internet community [23, 14, 1, 6]. These techniques are aimed at utilizing ATM switching hardware but do not use ATM connection establishment protocols.

In these schemes, once a packet flow has been detected a VCI is allocated for it and subsequent packets belonging to that flow are sent over that VCI. In this manner the routing lookups and forwarding state lookups need only be done once or at most a few times per packet flow instead of once per packet. Which VCIs are used for what flow is established by a separate protocol.

We believe that the basic idea behind these techniques are general enough to warrant support in the network protocol itself. Having a VCI-like field in network packet headers would allow all switching/routing to occur in one layer only. In this manner, the overall complexity of the network architecture would be reduced. The combination of datagram and virtual-circuit ideas, as exemplified above with flow and tag switching, provides the simplicity and flexibility of datagrams and the fast forwarding capability of virtual circuits.

A new network protocol should recognize this and provide syntactic and signaling support for virtual circuit switching and datagram routing. A new protocol should be able to do one or the other, or a combination of both, as the packet travels from source to destination. This implies that the packets of the new network protocol should be able to be fragmented and reassembled in the network; something that IPv6 forbids. In this way, the new protocol can be used over a variety of network hardware elements, and avoids the necessity of network protocol layering, as seen in IP/ATM architectures with all of their complexities.

8 Hierarchy and Scalability

An important property of a network protocol is its ability to scale to a large number of end systems and network elements. A factor that limits the size of networks

[5] The technique can be used even if the IPv6 flow label is zero or if the switch forwards IPv4 packets. Packet streams that are not normally considered flows in the IPv6 sense, for example large file transfers, will also benefit from this technique.

is the amount of routing/forwarding information that routers or switches in the network need to maintain.

For datagram networks, one bottleneck is the size of routing tables. With CIDR [16] a hierarchical structure is imposed on subnet identifiers by assigning subnet identifiers in the same "area" a common prefix. Several routing table entries can thus be collapsed into a single entry consisting of the common prefix. When combined with the "longest-match" strategy for searching routing tables, the original flexibility of the routing table is maintained and the routing topology need not be totally hierarchical.

Virtual circuit network protocols also use hierarchy to decrease the size of forwarding tables. For example, ATM virtual circuit identifiers exhibit a two-level hierarchy. In the backbone of an ATM network, only the first part of the identifier, called the virtual path identifier (VPI), is used when forwarding. Cells that follow the same path through the backbone have the same VPI. This decreases the size of the forwarding table in backbone switches and allows faster switching.

A potential problem with the Flow Labels of IPv6, as currently defined, is that sources pick them autonomously. This makes it difficult to combine several flows that should be forwarded in the same way into a single forwarding entry. If the number of flows passing through a switch is large, the size of the forwarding table can reduce the forwarding speed.

If we consider IPv6, it would be desirable if all flows that are forwarded along the same path and are classified in the same way for quality of service could use the same forwarding information. With the current specification of IPv6 flow labels this is not possible, this is likely to limit the scalability of resource reservation mechanisms in an IPv6 Internet.

9 Reservation Architectures

In traditional connection-oriented virtual circuit protocols, if resource reservations need to be made, they must be made during or after connection establishment. In protocols such as ST-2 or ATM, resources are reserved for real time traffic as the path is being setup in the network, i.e., at connection establishment time. There are some provisions for making or changing reservations during the life of the connection. But there are no provisions for making the reservation in advance of the actual communication. Unfortunately, where network resources are low and time pressures high, e.g., in global teleconferencing where participants reside in widely differing time zones, advance reservations may be necessary for efficient communication. To compare, the airline reservation system would be inefficient for customers and airlines if one could not make reservations in advance.

The case for advance reservations is somewhat better with the proposed resource reservation protocol for the Internet, RSVP. Even though RSVP is designed to provide reservations at the time of communication, reservations in advance of communication can be made. The source of a flow in a multi-party tele-

conference, for example, sends *PATH messages* downstream periodically through multicast routers to receivers who then send *Reservation messages* upstream towards the source. Reservations for the flow are then made in the routers where Path and Reservation messages meet. If a receiver receives a Path message in advance of a scheduled teleconference, a reservation could then be made and refreshed periodically before as well as during the teleconference. The burden on the network by advance reservations could be minimized by having the sender send Path messages in advance infrequently, increasing the frequency as the time of the teleconference approaches [11].

The problem with this scheme for RSVP is that the sender must be present and sending Path messages for the receiver to be able to make a reservation; and this may be impractical. Also, it should be possible for nodes other than receivers to make advance reservations on behalf of receivers who plan to be part of future communication. A promising approach for providing advance reservations is to see it as an administrative task, i.e., a task for network management. Such a scheme would allow reservations to be made by proxies, without the presence of the actual senders or receivers.

Our conclusion is that the most flexible approach to making resource reservations is to use all three mechanisms mentioned above. When resources are scarce enough so that quality could degrade during a session, but not scarce enough to demand advance reservation, then it may make sense to reserve resources by setting up hard state in a connection establishment phase. When senders and receivers are available before a session starts, soft-state based reservation could be used. Finally, when resources are so scarce as to demand reservations far enough in advance that senders and receivers are not yet in place, it makes sense for network management protocols to administer advance reservations in the network. A reservation architecture for a new network protocol should be prepared to make all three kinds of reservations.

10 Multicast

At least two models of multicast are available in today's communication networks: the group broadcast model used in datagram networks and the point-to-multipoint model used in connection-oriented virtual circuit networks. As usual, each model has its advantages and disadvantages.

The point-to-multipoint model builds a tree from each sender to a group of receivers. This model is exemplified in one version of ATM signaling and in the ST-2 protocol. The model is centralized in that the sender must know the identities of all the receivers, and new receivers can only join the tree by consent of the sender. This centralized approach provides the most security since the sender has almost total control (with the exception of network tapping) over who receives data packets. A disadvantage of this centralized approach, however, is that since the sender must keep track of every receiver, the number of receivers per-sender is limited.

The group broadcast model of multicast, used in Multicast IP [7], builds trees from senders to receivers without the sender necessarily knowing the identities of the receivers. The DVMRP multicast routing protocol for IP, for example, uses flooding and pruning to reach potential receivers on the edges of the network. In addition, local grafting of new nodes onto a pre-existing tree prevents senders from being bothered by changes to the multicast receiver group. The sender does not even have to be a member of the group it is sending to.

On the group broadcast model of multicast, a sender only has to send a packet to a group address and the routers conspire to forward the packets to all the members of the group. This means, however, that a receiver may become a member of a group simply by listening on the multicast address for the group and no consent of the sender is needed. To achieve privacy and security on the broadcast model of multicast, transmissions must be encrypted and receivers may sometimes need to be authenticated at a protocol layer higher than the network.

11 Mobility

An increasing number of users wish to be mobile while accessing the network. This is a problem for some protocols as addresses usually encode topological information, i.e., an address encodes a point of attachment to the network. For example, unicast IP addresses encode a particular subnet and an interface on that subnet.

Better support for mobility is possible if addresses denote a particular end system, regardless of where it happens to be connected to the network. Mobile IP [21, 22] solves the problem by having a *home agent*, located at the mobile host's home network. The home agent captures packets sent to the mobile host and forwards them to where the mobile host happens to be at the time. Forwarding is done by encapsulating packets with an additional IP header whose destination address specifies the mobile's current location. The mobile host reports back to the home agent when it moves again. When the mobile host and its correspondent have established contact, packets can take a shortcut around the home agent and follow the direct route from the correspondent to the mobile host.

In this manner, the address of the mobile host loses its topological significance. The destination address is only used for finding the home agent, after which it is no longer used by the network. There is another kind of IP address that also lacks topological significance: an IP multicast address. At a sufficiently high level of abstraction, the address of the mobile host is like a multicast address that is restricted to having only a single receiver.

The similarity between Multicast and Mobile IP becomes clear when comparing the latter to the PIM multicast routing protocol [8]. In PIM, a *rendez-vouz point, RP,* acts as a place where senders and receivers in a multicast group can meet. Receivers inform the RP that they want packets sent to the multicast group, and senders send their packets to the RP for further delivery to receivers. So, an RP and a home agent have very similar tasks. The principal difference is

that there is a single receiver in the Mobile IP case and that authentication of a receiver is vital for Mobile IP but not necessarily for multicast. A network protocol should support both multicast and mobility. It would be surprising if the routing support needed for multicast could not be used to simplify or improve mobility mechanisms.

12 Related Work

Another way to achieve a flexible network protocol is the *ActiveNet* or *Active IP* approach described in [30, 25, 27]. There, packets carry code to be executed by network elements. This provides an easy way to add control or monitoring functions to the network without having to wait for standardization bodies and updating of code in network elements. There are obvious security threats with such schemes and consequently much effort is expended to make the code safe. [27] uses Tcl[2], [25] also mentions Java [17] and various forms of safe platform-dependent binary code.

We agree that the idea of having packets contain the code that network elements should use to process them is very flexible. However, the approach seem to be most useful at moderate speeds; for high speeds, where the processing power of network elements is the bottleneck, it is questionable if such code can be interpreted or compiled and executed fast enough. On low speed networks it is questionable if the increase in packet size can be justified; adding as little as 10 bytes of code to each packet in a flow of interactive audio traffic can increase the required bandwidth beyond what is available.

13 Conclusion

To summarize, a new network protocol should combine the best features of datagram and virtual-circuit packet switching paradigms. To achieve a high degree of flexibility, fields in packet headers can be seen as independent pieces of control information that can be sent with varying frequency in a flow of packets. When a packet does not belong to a flow, all fields can be present. As a packet travels through the network, it might be either switched or routed, or a combination of both, on the path towards the destination(s). Such flexibility allows adaption to various current and future hardware. The new protocol should incorporate both hard and soft-state resource reservation mechanisms, possibly for making reservations in advance. Scalability is an important goal for a network protocol, and thus network addresses as well as flow labels should be hierarchical. Since the routing support for multicast and mobility has some overlap and the problems are similar it might be fruitful to merge the mechanisms.

In this position paper we have tried to lay out some of the fundamental assumptions behind traditional packet switched protocols. We have identified a number of network protocol philosophies such as datagram and virtual circuit models, connectionless and connection-oriented methods of communication, and

talked about hybrid models such as network "flows.". In addition we have mentioned different views of resource reservation, hard-state and soft-state, multicast and mobility. We have suggested that a new network protocol incorporate the best features of the foregoing and avoid their pitfalls. We are designing a new protocol, NP++, whose aim is to provide flexibility across many different kinds of networks and hopes to combine the best features of its predecessors.

References

1. F. Baker, Y. Rekhter: *Use of Flow Label for Tag Switching.* Internet Draft (Work in progress), October 8, 1996.
   `draft-baker-flow-label-00.txt`
2. Nathaniel Borenstein: *Email with a Mind of its Own: The Safe-Tcl Language for Enabled Mail.* In Proc. IFIP International Conference, Barcelona, Spain, June, 1994.
3. Robert Braden, Lixia Zhang, Steve Berson, Shai Herzog, Sugih Jamin: *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.* Internet Engineering Task Force, Internet Draft (work in progress), August 13, 1996.
   `draft-ietf-rsvp-spec-13.{txt,ps}`
4. Girish P. Chandranmenon, George Varghese: *Trading Packet Headers for Packet Processing.* Proc. SIGCOMM '95, Computer Communication Review Vol. 25, No. 4, October, 1995, pp. 162–173.
5. David D. Clark: *The Design Philosophy of the DARPA Internet Protocols.* Proc. SIGCOMM '88, Computer Communication Review Vol. 18, No. 4, August, 1988, pp. 106–114. Also in Computer Communication Review Vol. 25, No. 1, January, 1995, pp. 102–111.
6. B. Davie, P. Doolan, J. Lawrence, K. McCloghrie: *Use of Tag Switching With ATM.* Internet Draft (Work in progress), October 8, 1996.
   `draft-davie-tag-switching-atm-00.txt`
7. Steve Deering: *Host Extensions for IP Multicasting.* Request For Comment 1112, August, 1989.
   `ftp://ds.internic.net/rfc/rfc1112.{ps,txt}`
8. Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ching-Gung Liu, Liming Wei: *An Architecture for Wide-Area Multicast Routing.* Proc. SIGCOMM '94, Computer Communication Review Vol. 24, No. 4, October, 1994, pp. 126–135.
9. Steve Deering, Robert Hinden: *Internet Protocol, Version 6 (IPv6) Specification.* Request For Comment 1883, December, 1995.
   `ftp://ds.internic.net/rfc/rfc1883.txt`
10. Mikael Degermark, Mathias Engan, Björn Nordgren, Stephen Pink: *Low-loss TCP/IP Header Compression for Wireless Networks.* To Appear in Proc. MobiCom '96, Rye, New York, November 11-12, 1996.
    `ftp://cdt.luth.se/micke/low-loss-hc.ps.Z`
11. Mikael Degermark, Torsten Köhler, Stephen Pink, Olov Schelén: *Advance Reservations for Predictive Service.* Proc. 5th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95), Durham, New Hampshire, April 1995.
12. Mikael Degermark, Björn Nordgren, Stephen Pink: *Header Compression for IPv6.* Internet Engineering Task Force, Internet Draft (work in progress), February, 1996.
    `draft-degermark-ipv6-hc-01.txt`

13. Mikael Degermark, Stephen Pink: *Soft-state Header Compression for Wireless Networks*. Proc. 6th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'96), Zushi, Japan, May 1996.
14. P. Doolan, B. Davie, D. Katz: *Tag Distribution Protocol*. Internet Draft (Work in progress), September 16, 1996.
`draft-doolan-tdp-spec-00.txt`
15. David C. Feldmeier: *A Data Labelling Technique for High-Performance Protocol Processing and Its Consequences*. Computer Communications Review (SIGCOMM '93), vol. 23, no. 4, October, 1993, pp. 170–181.
16. V. Fuller, T. Li, J. Yu, K. Varadhan: *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*. Request For Comment 1519, September, 1993.
17. James Gosling: *The Java Language Environment: A White Paper*. 1995. Sun Microsystems.
18. Ipsilon, Inc.:*IP Switching: The Intelligence of Routing, the Performance of Switching*. Available from:
`http://www.ipsilon.com/productinfo/techwp1.html`.
19. Guru Parulkar, Douglas C. Schmidt, Jonathan Turner: *IP/ATM: A Strategy for Integrating IP With ATM*. Proc. SIGCOMM '95, Computer Communication Review Vol. 25, No. 4, October, 1995, pp. 49–58.
20. Vern Paxon: *End-to-end Routing Behaviour in the Internet*. To Appear in Proc. SigComm '96, Stanford University, August 26–30, 1996. A longer version of the paper is available at
`ftp://ftp.ee.lbl.gov/papers/routing.SIGCOMM.ps.Z`
21. Charlie Perkins, ed: *IP Mobility Support*. Internet Engineering Task Force, Internet Draft (work in progress), May 31, 1996.
`draft-ietf-mobileip-protocol-17.txt`
22. Charles Perkins, David B. Johnson: *Mobility Support in IPv6*. Internet Engineering Task Force, Internet Draft (work in progress), June 13, 1996.
`draft-ietf-mobileip-ipv6-01.txt`
23. Y. Rekhter, B. Davie, D. Katz: *Tag Switching Architecture Overview*. Internet Draft (Work in progress), September 17, 1996.
`draft-rfced-info-rekhter-00.txt`
24. Yakov Rekhter, Tony Li, Editors: *An Architecture for IP Address Allocation with CIDR*. Request For Comment 1518, September, 1993.
25. David L. Tennenhouse, David J. Wetherall: *Towards An Active Network Architecture*. Computer Communication Review, Vol 26, No 2, April, 1996, pp. 5–18.
26. C. Topolcic, Editor: *Experimental Internet Stream Protocol, Version 2 (ST-II)*. Request For Comment 1190, October, 1990.
27. David J. Wetherall, David L. Tennenhouse: *The ACTIVE IP Option*. Proc. 7th ACM SIGOPS European Workshop, Connemara, Ireland, September, 1996.
28. John Wroclawski: *Specification of the Controlled-Load Network Element Service*. Internet Engineering Task Force, Internet Draft (work in progress), August 14, 1996. `draft-ietf-intserv-ctrl-load-svc-03.txt`
29. Lixia Zhang, Stephen Deering, Deborah Estrin, Scott Shenker, Daniel Zappala: *RSVP: A New Resource ReSerVation Protocol*. IEEE Network Magazine, pp. 8-18, September, 1993.
30. *ActiveNets Home Page*. `http://www.tns.lcs.mit.edu/activeware/`

This article was processed using the LaTeX macro package with LLNCS style