CCIT Report #433 June 2003

A HARDWARE SIGNALING PARADIGM FOR FINE-GRAINED RESOURCE RESERVATION

Dan Gluskin and Israel Cidon

Department of Electrical Eng.
Technion - Israel Institute of Technology

ABSTRACT

Current implementation of real time service quality within converged IP networks is mainly accomplished by over-provisioning of bandwidth with limited definition of traffic classes on a network wide basis possibly enhanced by quasi-static provisioning of network elements using traffic engineering. Per-flow on-demand resource reservation is mostly unavailable in large packet networks. Examples are establishments of switched VC using PNNI signaling in ATM and establishment of LSPs using RSVP signaling in MPLS supported networks. Management complexities and limited scalability slow down this trend. While per-flow reservation is a conceptually straightforward QoS solution it is usually looked at as an impractical, non-scalable and even a higher cost solution.

Today high-end routers and switches can handle traffic volumes of many hundreds gigabit and even terabits per seconds, which can be translated to millions of simultaneous voice and video connections. However, current-signaling technologies will enable to handle only several hundreds connections that can be translated to only few thousands simultaneous short lived connections. Clearly, today call establishment mechanisms cannot scale to support per-flow reservation that is conceptually a simple QoS solution. In order to solve this limitation, there is an on-going effort to reduce the required reservation rate by developing complex hierarchical aggregation schemes and multiplexing concepts. This limits the use of connection establishment signaling to aggregate traffic engineering, which is hard to define, understand and manage and may fail to provide a required QoS solution.

This work (based on unpublished thesis [1]) is first to examine the possible solution of the above scalability problems for IP signaling by implementing them in hardware. We present a novel design termed "Keep-It-Simple" Signaling (*KISS*) which is optimized for hardware signaling of unicast connections which dominate the present world of real time services. We show, that backbone routers, can process by hardware *KISS* messages to improve their connection establishment capabilities to a level where fine grain user application-initiated resource reservations is feasible. Such hardware signaling may deliver the missing component to fulfill full QoS support in large IP networks.

1. INTRODUCTION

Despite a tremendous growth in Internet traffic, capacity and services, QoS support is still missing in the global Internet and is highly limited at smaller installments like corporate WANs. Therefore, high-quality real time services such as voice and video conferencing, VOD and real time Webcast are not currently deployed in large scale. Traditional network technologies offer few distinctive services. The IP network architecture offers scalable datagram connectivity, good resource utilization and a reasonable support of few prioritized best effort QoS classes. This fits well the bursty nature of computer communication and data services but is sub-optimal for many potential real-time applications. The circuit-switch telephone network, offers global connectivity as well as real-time QoS capabilities but is inefficient for supporting bursty or short transactions. It presents long connection setup time and a limited number of QoS services. The goal of future converged networks is to introduce a single infrastructure that offers the global connectivity and efficiency of the Internet as well as a rich array of QoS and real-time capabilities starting from established services such as peer to peer telephony, voice and video conferencing, video on demand (VOD) and on-demand bandwidth dialup for access and peer to peer connectivity. A known way to assure that connections quality meets a desired level using limited bandwidth resources is connection based network resource reservation. Given the emerging trend of rich-media, livecast and on-demand high bandwidth services such capabilities may become critical to broadband networks and providers. Current support of real time service quality in converged networks is accomplished by bandwidth over-provisioning and

network wide definition of few traffic classes. A quasi-static provisioning of network elements using DiffServ and MPLS may enhance this possibly utilizing signaling. Examples are ATM SVCs setup using PNNI signaling and establishment of MPLS LSPs using RSVP signaling. Deployment complexities, limited scalability and lack of a QoS routing limit the usability of these signaling systems to coarse traffic engineering. While per-flow reservation has been discussed in the literature as a conceptually straightforward QoS solution it is usually considered to be impractical, non-scalable and high cost solution. Consequently, today's high-end routers handle traffic volumes of hundreds gigabit and terabits per seconds, which can be translated to millions of simultaneous short-lived voice and video sessions. However, current software signaling technologies handle only hundreds (possibly thousands) connections per second and the signaling capability of routers cannot provide per-flow on-demand resource reservation. This limits the use of connection establishment to aggregate traffic engineering, which is hard to define and manage, and may not provide the full-required user on-demand support. This implies a basic gap. Today call establishment mechanisms cannot scale to support per-flow reservation that is conceptually a simple QoS solution. In order to solve this limitation, there is an ongoing effort to reduce the required reservation rate by developing complex hierarchical aggregation schemes and multiplexing concepts. This both complicates the simple end-to-end resource reservation idea and may fail to provide a valuable and sellable QoS solution.

Looking on today's most popular IP QoS mechanisms we also observe that RSVP is originated in receiver initiated multicast model and did not originally adopted the notion of connection *pinning down*. Other approaches, considered today for providing QoS, are over-provisioning and QoS quasi-static assignment (MPLS or DiffServ) using traffic engineering. The first requires global availability of low-cost very high-bandwidth IP infrastructure. The second, while being on the surface simpler than on-demand reservation is complex to define and manage, has unclear efficiency in utilizing and saving bandwidth and requires global management and traffic monitoring.

The on-demand resource reservation operation should include the following tasks. QoS routing - finding a low cost network path with sufficient network resources to support a new connection. Node and link call admission control, which are the tasks of estimating whether each network element in the path has enough residual resources (in particular bandwidth) for the new reservation request and a configuration of QoS policing and forwarding which are the tasks of providing each flow with its reserved share of the network resources. Finally, the signaling mechanism that forms the exchange of information between network nodes for establishing connections in real time. While this paper focus mainly on the signaling part which is held back by major scalability issues, the other components should also be solved and optimized in an integrated way see for example [31, 30, 2]. Moreover, certain mechanisms like multi-path reservation [32, 33] tie together the path selection and signaling mechanisms and mix the different call establishment tasks.

Current signaling protocols are designed mainly for software implementation [15, 16, 17, 18, 19, 20, 21, 24, 30, 31], were some attempts were made for high-rate software implementations [31, 3] As user population, communication volumes and connection rates increase faster than CPU performance, such software implementations cannot support on-demand, real-time fine-grain reservation. This situation is very similar to the situation faced by the early packet switched networking world in the late eighties and early nineties. Packet switches based on software processing using general purpose processors were not able to cope with the increasing levels of traffic. In order to free switching bottlenecks, packet switching in special purpose hardware was proposed [6, 7, 5, 4, ?]. At the first stage, in order to facilitate such transition to a high-speed hardware-based packet switching, new and simpler network architectures suitable for hardware switching were proposed. Most of them sacrificed some efficiency (such as giving up on hop-by-hop congestion control, large address space or a variable packet size). Eventually, (a more complex) hardware switching support was also developed for the de-facto standard architectures such as Frame Relay and IP.

Similarly, while current-signaling protocols may end up eventually in hardware implementations, their basic design was not optimized for hardware implementations and follow-on optimizations assumed a general purpose processor implementations. For example, RSVP was designed for receiver-based multicast services that are not seem to migrate to converged networks. On the other hand, most telephony, streaming and even multimedia conferencing are based mostly on unicast service (where conferencing is largely accomplished by a conference central server). Moreover, aggregation techniques (and/or getting read of soft-states) that help in scaling better software implementations of such protocols by reducing messages rate and CPU cycles actually make hardware implementations harder by adding more complexity to each aggregate operations and by separating the implementation of frequent main stream event from exception cases. Therefore, a reasonable ap-

proach toward hardware signaling is to devise signaling protocols that are optimized to unicast services signaling in hardware and that minimizes hardware complexity at the expense of bandwidth and case uniformity.

In this work we propose a signaling paradigm for fast, simple and scalable hardware implementation. Using its concepts, we developed a sample signaling suite which is optimized for MPLS based IP networks. We estimate that using current standard ASIC and memory technologies, a single *KISS* hardware unit can easily support signaling for millions simultaneous connections and tens of thousands setup and tear-down requests per second. We also show that *KISS* scales to even higher setup rates. This presents more than a 100-fold improvement over today's ATM switches and MPLS routers that are limited to few hundreds requests per second [37] Our work focuses on IP networking because of its big install base, its dominant position in the market as well as the growing availability of label-switching capable routers (MPLS) that offer new possibilities for resource reservation. Finally, this work is based on work [1] that was completed almost 3 years ago. These results were generally rejected due to being *incompatible* and a failure to *prove* that a different approach is needed. We still find that this integrated signaling paradigm may be one of several possible ways to overcome the IP convergence barriers.

2. MOTIVATION

Today's high-end routers use dedicated forwarding hardware which enables them to cope with high traffic volumes. Signaling, however, is implemented in software and executed by the router's CPU. As routers performance increase faster than CPU speed, the connection establishment capability becomes less adaquate for the task of on-demand reservation. In this section we describe the rational which leads us to develop the *KISS* signaling paradigm.

Currently, there are three main proposals for integrated network signaling protocols: IETF's RSVP [19, 20, 21], MPLS's LDP [15, 16] and ATM's PNNI [17]. It seems that RSVP signaling for MPLS networks is the leading market solution.

RSVP development correlates in time with the experimental MBONE [28]. It is also optimized for a model of broadcast hosts to which large amount of users tunes-in and out. Consequently, a major part of the RSVP standard deals with receiver initiated multicast. Multicast is important for a better usage of network resources for such applications. However, reviewing today paid for services, and the on-demand nature of most internet applications, it is likely that unicast applications will produce the vast majority of reservation requests. Consequently, unicast support should not be compromised for the support of the more complex multicast. The importance of a better unicast support was also acknowledged by the RSVP community. The RSVP-TE draft [21], suggests simplification of unicast reservations. This draft enables RSVP to open unicast simplex connections (LSP) on MPLS networks.

Robustness is an important issue for a signaling protocol. A failure in the tear-down of connections might lead to situations where resources are wasted. For that reason, RSVP has introduced the concept of soft-states. Soft states expire after a predefined timeout if not refreshed by special messages. This way "orphan" reservations are released automatically. The original RSVP proposal assumes that the route may change during the connection life time (somewhat in contradiction to a QoS guarantee). Therefore, their soft-states are refresh by retransmitting the whole connection setup messages (PATH & RESV) which are relatively large. Soft-state refresh messages are sent for all active connections. If refresh interval is short and the number of open connections is high, it results in a high signaling overhead. On the other hand, decreasing the refresh interval reduces the effectiveness of the operation.. "RSVP refresh reduction draft" [22] proposes some mechanisms to lower soft-state overhead at the expense of increasing the protocol complexity.

Connection setup latency should be as short as possible. As RSVP doesn't use acknowledgments, a loss of an RSVP setup message (first PATH or RESV) is left unnoticed, the message will be resent after a soft state refresh interval which is usually 30 seconds. Protocols, like PNNI, which use hop by hop acknowledgments can quickly identify message loss and resent the lost message.

Reservation aggregation is important for global networks. For example an ISP that needs to connect multiple customer remote sites, might setup permanent connections among the company sites and dedicate resources for them (Virtual Private Network). The customer may still need to multiplex multiple connections among sites. MPLS label-stack (and ATM virtual-path) support this feature, however currently there is no support for it in RSVP.

Parsing is the first stage of a protocol message processing. A reservation protocol has to carry information which is opaque to it. RSVP and LDP use an "object oriented" approach. their messages are made of variable size objects, each starts with its own header. The object actuall size appears as a field in that header. To find out what objects are carried in a message, the implementation must parse the whole message to find

the different object headers. RSVP extensive usage of objects and sub-objects, each with its own header enlarges the size of the signaling messages.

Our first attempt was to modify one of the existing signaling protocols (RSVP actually) for hardware implementation. However, the amount of fundamental changes required and the RSVP design model (receiver initiated multicast, changing routes, no acknowledgement, etc.) lead us to develop a new signaling protocol. *KISS* was designed with hardware implementation in mind. We later show that it fits well the structure of current high performance routers.

Although, extra work is needed before *KISS* can be deployed in commercial networks, it demonstrates that parts of the signaling processing can be offloaded to dedicated hardware. We show that it enables per-application reservation even in backbones on global public networks

2.1. Software vs. Hardware

Dedicated hardware usually performs much better than software running on a general purpose CPU. On the other hand, software solutions are usually preferable as they can enjoy new faster CPUs without a redesign. Other advantages are the availability of good design and debug tools and the ease of upgrading. In general, for almost any task, software designs are simpler and faster to create. Even if today's CPUs cannot cope with a given task, software solution can, in many cases, be chosen as it can be expected that CPU speed will rise during the development period.

Networking is one of the few exceptions as, in recent years, communication volume increases faster than CPU performance. The vast improvement in fiber-optic bandwidth, forced router designers to use dedicated hardware (ASIC). This was not a trivial evolution, as unlike ATM, which was designed for using forwarding hardware, IP was originally designed as a software overlay network. IP has features which makes dedicated hardware design difficult. For example, efficiently switching IP packets with their variable length, is harder than switching ATM fixed size cells. The need for switching bandwidth forced hardware engineers to find creative hardware solutions for this task. We believe that to enable backbone routers to support per-application reservation, signaling processing should follow the same path - hardware implementation.

Even a quick review of current converged networking signaling standards, reveals their complexity. Designing a hardware implementation for PNNI, for example, seems infeasible. We believe the signaling protocol should be designed in a way that enables hardware implementation of the most common tasks, like setting up and tearing down unicast connections. Other, less frequent, procedures can be left for software.

Beside the ease of design, software is also easier to debug and upgrade. There are additional hardware solutions which do not perform as good as ASIC but offer some of the above advantages.

Programmable hardware chips (FPGA) can acommodate large designs of millions of gates [35]. Although, they do not match todays ASIC clock rates, they offer better flexibility as their design can be changed by loading a newer configuration file.

Network Processors [36] are microprocessors which are optimized for network related tasks. Theoretically, they enable software implementation of network tasks which are competitive with hardware speeds while keeping the ease of development and upgrade of regular software. At this point this technology is still immature and does not replace hardware implementations in large scale. We plan to evaluate in the future the implementation of *KISS* using network processor technology.

In the previous subsection, we explored the limitation of current signaling protocols. In the *KISS* design we took some decisions that enable simple hardware implementations (software solutions can also make use of its simplicity). In section 5 we show that an FPGA *KISS* implementation, is expected to cope with signaling load of millions short term connections.

3. MODELS

3.1. Network Model

An internet is a collection of networks that are connected by routers that use the internet protocol suite. In the classic model, routers are machines with a number of network interface modules (NIM), each resides in a different network. In the early days of the Internet, routers were general purpose computers with standard network-interface cards that run a special routing software. Today, routers are usually dedicated units which are design to handle the growing speeds of network links. To cope with the amount of routing decisions a router has to take, functionality is gradually moved from the router's software to a dedicated hardware. In many routers each NIM is supplied with its own routing table hardware. The router's CPU updates these tables according to the routing protocol information (EIGRP, OSPF, BGP etc). In these designs, regular IP forwarding is done on-the-fly by the router hardware, without any CPU processing. Such a structure is scalable as each routing engine has to take decisions only for one incoming traffic link.

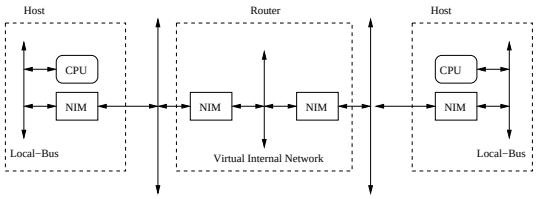

Figure 1: NIM view of the internet

The connection among the NIMs is made by a special high speed connection element typically a switch or a ring. This leads to another level of abstraction, we can think of a router's inter-NIM connection as a virtual internal networks (figure 1). Now, an internet can be viewed as networks connected by NIMs (each with only two connections). Regular hosts have only one NIM connecting their local-bus to a LAN. Routers have a virtual-internal-network that connects NIMs and maybe a CPU.

The proposed signaling protocol (*KISS*) was planned for this model. It can be viewed as a signaling protocol among NINs. Unlike traditional signaling implementation, which use the router's CPU, this structure scales well as each signaling unit is responsible for only one link. In section 5 we show that a single *KISS* unit can easily cope with the fastest fiber optic link.

3.2. Conceptual KISS-enabled Network Interface Model

Figure 2 shows a block diagram of a hardware signaling enabled NIM. Regular packets are being forwarded without software intervention using the hardware speed path. The CPU has to deal only with special packets, like IP packets that contain options.

Each router is built from NIMs connected to each other by an internal fabric. The NIM has connections to a network and to the internal fabric. The NIM speed path contains the following part: packet classifier, packet scheduler, forwarding table/engine and ARP (address resolution) tables. There are speed paths for incoming packets and for outgoing packets.

Routers that support different levels of QoS may manage several queues to provide different priorities for different flows. Modern routers can manage tens and even hundreds of thousands of queues. The packet-classifier task is to identify packets, according to predefined parameters and move them to the right queues. When working with a label-swapping paradigm like MPLS, this task becomes trivial - the labels identify the flow. Scheduling packets for transmission is done by the packet scheduler, which is responsible to grant each queue with its time share according to some predefined parameters (like token-bucket parameters).

New dedicated hardware can provide wire-speed forwarding decisions. The forwarding tables are updated by the router's CPU which executes routing software. The tables contain the next-IP hop. To translate this IP to a MAC address there are Address Resolution (ARP) tables. Note, that the incoming speed path in figure 2 contains an "internal ARP" block. We added it for the sake of symmetry. Its function is to translate the next NIM IP to the internal address that enables the fabric to move the packet to the right NIM. Another set of routing-tables is needed for MPLS routers. These tables are simple as only full-match lookup is needed.

Our proposed hardware-assist signaling protocol can be controlled using a simple hardware. The packet classifier moves packets that contain signaling messages to the signaling hardware. This identification can be done according to the IP protocol field. We defined two dedicated *KISS* logic blocks (KLB), one processes messages that arrive from the network (and can send messages back), the other processes messages that arive from the fabric. The connection between KLB on the same interface is treated as reliable, while connection among KLBs on different NIM is treated as unreliable.

Our protocol always uses labels, it holds the flow state information in simple look-up-tables (LUTs). For simplicity, the labels are selected to be the flow LUT entry number. The KLB can access the LUT directly as each signaling message carries a label. The LUT are also being accessed periodicly to implement timeouts and soft states.

The KLB communicates with the CAC and an optional policy units. The network-side KLB verifies with the CAC unit network resources availability for data flow from the NIM to the network, the internal connection side verifies resources availability from the NIM to the fabric. The Policy unit verifies that an initiator of a reservation-request has a permission to do so. This action has to be done only at the edges of the network. After the signaling controller gets a permission to reserve resources it configures the packet-classifier and scheduler accordingly.

4. DESIGN

Suggesting fast hardware implementations for tasks which were design for software is not new to the networking world, IP CIDR "longest best match" [9] is a good example. Signaling protocols, however, are quite complex. We developed the *KISS* ("Keep-It-Simple" Signaling) suite which has all the important features and is

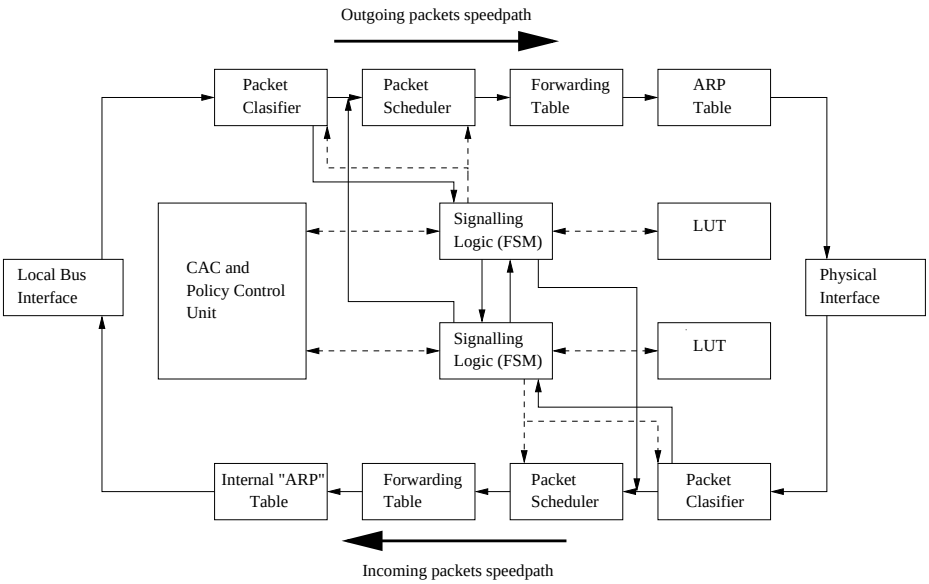

Figure 2: Enabled NIM conceptual model

simple enough for fast hardware implementation. The signaling protocol was design for IP networking (version 4) preferably with MPLS support. The source size of its software implementation is less than 1/10 of RSVP implementation [23]. This is not a reduced RSVP but a totaly new protocol. Its development was inspired by signaling protocols we reviewed including PNNI, RSVP, CR-LDP, SS7, OPENET [31] and more. In the following paragraphs we explain the *KISS* protocol design considerations.

Maintaining complex data-structures in Hardware can be a challenging task. The only data-structures required by *KISS* implementation are Lookup Tables (LUT), which are accessed either by direct entry number or periodically for maintaining soft-states. Note, that if the LUTs are implemented in DRAM, periodic accesses can replace refresh cycles and thus their overhead is low.

To simplify the task of associating the signaling messages with their specific flow, *KISS* always uses labels. These labels are the LUT entry number of the flow state. During a connection setup phase, each NIM along the path selects the incoming label which marks incoming signaling packets as related to the connection ("downstream on-demand" in MPLS termination). The signaling mechanism allocates a label when it receives the first reservation-request of a flow (SETUP message). The Network Interface Module (NIM) signaling unit sends the label back in an acknowledgment message, and forwards it to the next-hop with the SETUP message. All other flow related messages, that are sent to that NIM, carry this label. The labels are switched as messages travel from one NIM to the other. In some cases, if the regular forwarding mechanism also uses labels, the same label can be used for both data and signaling. In contrast, to associate an RSVP message to a flow, for example to tell if a received RESV message should refresh an existing state, the RSVP daemon has to check multiple fields. The *KISS* usage of labels, not only simplifies this task but also enables the use of short acknowledgments and refresh messages (RSVP designers have later introduced their MESSEGE-ID object for this task [22]).

Timeouts and soft-states are easily maintained in the proposed architecture. When an event occurs for a given flow, the time of the next-expected-event timeout is stored is a special field in that flows LUT entry. As mentioned above, the signaling controller does periodic accesses to the LUTs. During these accesses it checks for timeouts, by comparing the stored time to the current time. These event-time fields have to be accessed once every t, where t is the shortest time constant used. We believe that t in the order of one second should be enough for maintaining logical time constants.

Two kinds of propagation delays were identified, end-to-end and hop-to-hop. In connections that transverse many-hops, the end-to-end delay can be long and thus, the timeout constants used when waiting for a message must be long too. As hop-to-hop propagation delays are shorter, the protocol avoids situations where a loss of a control message causes a long wait for an end-to-end type timeout. This is achieved by the

usage of hop-by-hop acknowledgments. To reduce the acknowledgment overhead, one packet can carry multiple acknowledgments.

For robustness and self-healing, we chose to use soft-states. Unlike RSVP, the soft-states are refresh by special refresh-messages. Note, that although RSVP doesn't use hop-by-hop acknowledgments, "RSVP refresh overhead reduction" draft [22] suggests a mechanism for it (by piggy-backing of MESSEGE-ID-ACK objects).

Fast failure detection is an important feature as it enables the network to take countermesures to fix the problems, even before the users sense it. We expect that the most common network configurations, in which reservation will be done, will be point-to-point links. In this case checking the link state can be a data link layer service or be done by say a software daemon sending ICMP echo-request ("pinging"). The daemon can also calculate the Round trip time and adjust the signaling time constants accordingly. Alternatively, it can be done by the signaling unit (we defined an "hello" and "hello-ack" messages"). This task is more challenging in the "partial reservation" or multicast network case. In these configurations, it might not be a-priori known which are all the signaling enabled neighbors. The signaling unit finds the address of these neighbors when it receives acknowledgments to the connection setup messages. It can thus maintain a list of them. IP lookup is needed for checking if a new hop is already in the list. But, unlike forwarding tables IP lookup, it requires full match only. In some configurations, this information can be gained by adding extra information to the routing protocol.

KISS uses a simple encoding. All messages start with a common header which contains the message size and number of objects. The header is followed by an objects map, which holds the types of objects and their starting point related to the beginning of the message. To find out the content of a message, only the header and the map should be parsed.

To reduce the signaling messages size, parts of *KISS* message data is carried in bit-fields. In general, objects are used only for data which isn't processed by the signaling controlers themselves (like *flowspec* and MPLS lables). Further reduction is possible by placing few messages in a single packet.

Like RSVP, *KISS* is build directly over IP. But, it uses acknowledgments to enable fast retransmissions in case of packet loss.

KISS packets are protected by a 32bit checksum which is placed at the end of the packet. This eases an "on the fly" checksum generation and verification. An alternative is to use a message-digest, like MD5 [11] which can also give some protection against spoofing.

Another feature that simplifies implementation is alignment. *KISS* message objects and fields are aligned to 4-octet boundaries, which is also the alignment of IP headers. It enables easy hardware parsing with 32-bit data path. .

4.1. Multicast

As mentioned before, we don't believe that multicast support should come at the expanse of simple unicast reservations. **However, we found that it is possible to support one-to-many and many-to-one distribution trees while maintaining the simple lable/LUT design. Changing the transmitter of a multicast tree is also relatively simple**. When a transmitter wishes to add a new receiver to the session, it opens a new connection to it. If succesful, this new connection merges with the existing multicast tree. The merge is done upstream, from the new receiver towards the transmitter. Thus, unlike RSVP, the transmitter has an indication of the procedure status. This flow resembles PNNI "ADD PARTY" procedure.

For unicast sessions it is clear that source-initiation is natural. RSVP style, receiver initiation of sessions, is more complex but has some theoretical advantages for large multicast sessions. One of these advantages is the reduction of signaling load of multicast-trees cores. *KISS* , on the other hand, uses transmitter initiation. To reduce signaling load of the core, it uses PIM (Protocol Independent Multicast protocol [26]) style Rendezvous Points (RP). RP are network nodes which are already part of a multicast session. In order to join a multicast session, receivers can send join request messages to the nearest RP. The RPs initiate setup messages to add the new receivers to the session. The multicast-tree core is unaware of this signaling and thus, the signaling load is reduced. It's the core responsibility to set the RPs and deciminate their information to enable receivers to use them.

Receiver-initiation might have an advantage in cases where each receiver have different bandwidth demands. For example, one receiver might want to get the full resolution and color of a broadcasted video, while the other might settle with low-res B&W. RSVP introduced the "filter-specification" concept which, in its general form, can take advantage of this situation. Receivers can use this filter-spec to specify which of the flows-packets they wishes to receive. RSVP filters can also limit a reservation only for packets that were send by specific transmitters. The multicast tree branch-points should do the actual filtering e.g. to identify which of the flow's packets can take advantage of the reservation placed by each downstream sub-tree. This

can be quite challenging as it has to be done for all *data* packets. When reservation is placed, filter-specs from sub-tress should be merge before the resevation request can be sent upstream (for RSVP see [20]).

Building high-speed packet classifiers which can filter packets according to arbitrary packet fields is not trivial. It's even harder for IPSEC enciphered packets (see RSVP extension [13]). Label-switching can make the task of associating packets to a specific flow easy. MPLS shim header has three "CoS" bits. limiting "filter-specification" for selecting packets according to their CoS ,or even IP ToS fields, can make filtering feasible.

At first look it seems that a similar effect might be achieved by simply setting few parallel multicast distribution trees, for example, one can carry luminance data while the other color data. receiver that wishes to display only B&W video can join only to the first multicast tree. Packets that are sent on a distribution tree, maintain their original order. But, packet order is not assured when different distribution trees are used. This might cause synchronization problems for receivers that use those parallel trees.

4.2. Support for partial-deployment

The signaling protocol was planed to support partial deployment. Reservation can even be placed if not all the network nodes along the path support the reservation protocol. To assure that new reservation request messages will travel along a desired route a standard IP loose-source-route (LSRR) [8] can be use. Source route is an IP option. Many high-end routers don't process options in the hardware speedpath but move them to their CPU for processing or simply ignore them. However, *KISS* signaling units should be able to support LSRR by hardware[1]. The destination field of the SETUP message is the final destination.

An IP alert-option[2] [12] can be use to notify routers along the path that they should examine the setup messages. Before forwarding a SETUP message, each enabled NIM replaces the IP-source field with it's own interface address. When a NIM receives a SETUP message it examines the IP source field and thus, it knows

[1]LSRR support is not complex, in only involves with advancing the next hop pointer. As LSRR length is 3 bytes + 4*(Number of hops), we assume that a one-byte NULL-OPTION is always placed before the LSRR

[2]BSD raw sockets enables adding support for new IP protocols without changing the OS kernel. But, normally only packets with destination addresses that belong to the machine are passed to the raw sockets. This can pose a problem for *KISS* setup messages which should be processed by every node along the path. On the other hand the OS kernel moves every packet with router-alert-option to the raw socket, no matter its destination.

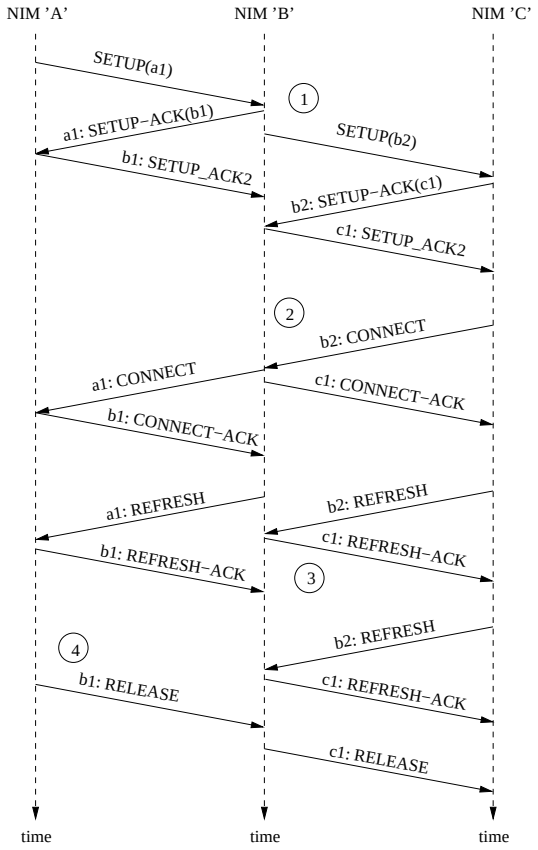

Figure 3: KISS message flow

the address of the previous *KISS* enabled hop. Routers can identify their direct neighbors, so they can tell (a) if it's a partial-reservation and (b) who is the previous enabled NIM. From this state on, whenever the NIM need to send a message downstream it sends it with the previous NIM address in the packet's IP destination field. Setup messages are also acknowledged on a hop-by-hop basis (actually, enabled NIM hop by enabled NIM hop). The senders of setup-acknowledgments place their address in the IP source field. This way, a NIM finds out which are the next-enabled NIMs. Note that this "IP destination swapping" is done by the signaling controller and thus, works only for the control messages. In order to support such a reservation the forwarding mechanism should also support this kind of "IP destination swapping". The same support is available in RSVP

4.3. KISS sample message flow

Figure 3 describes the message exchange for a typical *KISS* connection.

In the beginning of first phase (1), the initiator

(NIM 'A') sends a *setup* message toward the destination. This message can carry IP source-route to ensure it travels along a predefined route. It carries a flow identification label (FIL) which identifies the flow in NIM 'A' (a1). The message also carries a *flowspec* of the new flow. When 'B' receives the message, it verifies with its CAC unit that it has enough resources to accept the new connection. From the *setup* packet, it extracts 'A' address. 'B' is ready to accept the connection, so it sends back a *setup-ack* message to 'A'. This message carries the FIL from the *setup* message. It also contains the FIL of the flow in 'B' (b1). The *setup-ack* message is acknowledged with the *setup-ack2* message. The connection was accepted so 'B' sends the *setup* message to the next hop ('C'). This message carry 'B' FIL[3]. The CAC unit can reduce the *flowspec* if it doesn't have enough resources.

In phase (2), the destination accepted the new connection. It sends a *connect* message, which carries the adjusted *flowspec*. The message travels upstream toward the initiator (A), all the NIM along the way modify the amount of reserved resources according to this *flowspec*. The *connect* message is acknowledged with the *connect-ack* message. Note, the FIL swapping as the message travels upstream. When the message reaches the initiator the reservation is active and it can start to send data.

During the connection lifetime (3), the soft-states refreshed in relative long intervals (30 seconds). The refresh messages are initiated by the NIM, in a hop-by-hop manner.

When one party wishes to terminate the connection (4), it sends a *release* message. When a NIM receives this message, is releases all reserved resources and sends the message to the next hop.

5. IMPLEMENTATION

KISS was designed for hardware implementation but a software implementation was used for the debug and protocol development. Using this tool we tested the *KISS* protocol on large virtual networks.

As seen in figure 2, the KISS protocol implementation relies on two similar *KISS* logic blocks (KLB), one intercepts signaling messages coming from the network connection, the other intercepts messages coming from the internal connection. The two logic blocks also exchange messages with each other. Each KLB has it's own look-up table for storing the flows states. An extra CAC and policy unit is needed for managing the network resources.

[3]Note that the FIL which is sent to 'C' can be different than the one which is sent to 'A'. This may occurs due to the two KLB structure. Each can select it FIL.

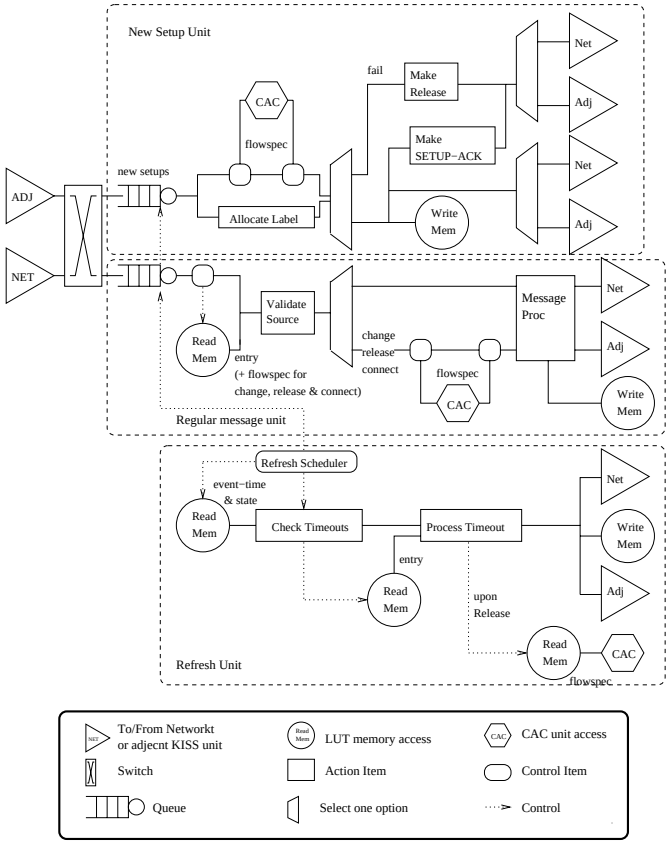

Figure 4: KISS logic block diagram

Figure 5 is a proposed KLB block diagram. It has three major units. The "setup-unit" is responsible for managing new connection setup requests (a setup request doesn't carry an incoming label). The "message unit" processes all other messages. There are separate queues for both units. The "refresh-unit" which does periodic access to the LUT, is responsible for identifying and responding to timeouts. Before a new signaling packet reaches a KLB, a special checksum unit verifies its integrity (KISS checksums are at the end of the message so this can be easily done on-the-fly). The *KISS* protocol uses 32 bit alignment which is can be easily kept in IP version 4. Thus, a 32 bit data path is natural for the KLB.

The "Setup unit" processes new setup request either from the adjacent KLB or from the network or internal connection. For each new request, the unit sends the *flowspec* to the CAC for inspection. Concurrently, it allocates new LUT entry for it. If the admission control and label allocation were successful, the unit generates an acknowledgment, forwards the setup

message and writes the flow state to the allocated LUT entry.

The "Regular message unit" processes all other messages. When it receives a new message, it first fetches its flow state from the LUT according to the message label. Using the state data, it verifies that the message was sent by the right NIM. If the message changes the flows resource allocation (e.g. tear-down, change-reservation or connect) the CAC unit has to be accessed. When the message processing ends, the unit usually sends back an acknowledgment,forwards a message and writes back the flow state to the LUT entry.

The "Refresh unit" does periodic accesses to the LUTs, it checks the flow-state and the time-tag. If a timeout occurred, it fetches the flows LUT entry, performs the required task and writes it back. In case a soft-state expired it also has to access the CAC unit for releasing the reserved resources.

Current unicast implementation, uses about 180 bits for each flow state. Some extra bits per flow are needed by the CAC unit. We believe that 512 bits per flow should be sufficient for every CAC unit. The LUT are most easily implement in standard DRAM. Current DRAM technology offers up to 256 Mbit in a single DRAM device. Standard devices have data path of 4,8 or 16 bits. To create a 32 bits data path we need 8,4 or 2 devices which hold a total of 2048, 1024 or 512 Mb accordingly. If one quarter of the available memory is reserved for temporal message storage, this fabric can handle 0.768, 1.536 or 3.072 million of connections states. Note, that two KLB units are needed in each NIM, the number should be divide by half if they share the same memory.

The high clock frequencies that can be reached in VLSI designs compare to possible clock speeds for DRAM memories or PCB traces and the simplicity of *KISS* message processing, results in memory being the major performance bottleneck. Signaling bandwidth is not expected to pose any problem as the number of connections is proportional to the links bandwidth. Another possible bottelneck is the CAC and policy control unit. For best result we recommend that these units be integrated with KLB.

Synchronous DRAM technology use pipelined synchronous logic to enhance DRAM performance. Current standard SDRAM can use a 133MHz clock (PC133) while new Double Data rate technology, which transfer data on both rising and falling clock edges, actually doubles this transfer rate. DRAM memory-matrices are divided to pages, data-per-clock is only possible within a single opened DRAM page, DRAM page switching can take many clock cycles. Standard SDRAM memory matrices are further divided to four banks which

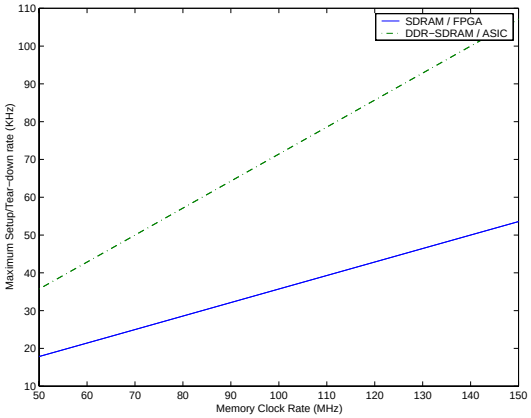

Figure 5: KISS Performance estimation (Avarage connection duration 40 seconds)

enable keeping four DRAM pages open. KLB can use this memory architecture to use the DRAM more efficiently.

Simulation shows that in low packet loss conditions, a 10000 LUT entries KLB needs 559K SDRAM cycles per second to handle 200 connection tear-downs and 200 connection setups per second These parameters are expected to scale linearly (figure 5). A FPGA implementation working with standard PC133, 133MHz SDRAM (a high but possible clock rate for FPGA) can support about 45K setup and tear down requests per second. For an average connection duration of 40 seconds, this is enough to cope with 1.8 millions of simultaneous connections. Assuming 64Kbps PCM voice channels it means support for a 115Gbps links. If a more sophisticated codec is used, for example, 5.3Kbps ITU-T G.723.1 9.54Gbps links, about OC-192, can be support. Using DDR-SDRAM can double these figures, a further enhancement is possible if few units are used in parallel.

6. DISCUSSION

The goal of our work was to find scalable techniques for micro-flow resource reservation in large networks. Note that a full solution, must address signaling, QoS routing, admission control, QoS policing, etc.

This article describes the *KISS* signaling paradigm. We showed that it offers new ideas for creating a scalable signaling mechanisms, and also raise new possibilities for fast signaling hardware that can reach adequate connection setup capabilities.

QoS routing is an active research topic and much was done on that issue (a good overview can be found at [38]). However, there are still no widely acceptable

solutions and more work is required to glue the signaling and the QoS routing to an integrated solution. This work is currently conducted at the Technion.

7. REFERENCES

[1] Dan Gluskin "Hardware Oriented Resource reservation scheme for integrated networks," Technion EE Master Thesis, March 2001.

[2] IETF's Network Working Group "RFC 2676: QoS Routing Mechanisms and OSPF Extensions ", August 1999

[3] I. Cidon, I. Gopal and A. Segall " A. Fast Connection Establishment in High Speed Networks ", ACM SIGCOMM, September 1990, pp. 287–296

[4] I. Cidon and I. Gopal " PARIS: An approach to integrated high- speed private networks", International Journal of Digital Analog Cabled Systems, April-June 1988, pp. 77–86

[5] A. Huangand S. Knauer " Starlite: a wideband digital switch ", Proc. GLOBECOM'84 (November 1984), pp 121-125

[6] J. Turner " New Directions in Communications (or Which Way to the Information Age?) " Proceedings of the Zurich Seminar on Digital Communication, pp. 25–32, March 1986.

[7] J. Turner " Design of a Broadcast Packet Switching Network," " Proceedings of Infocom 86, April 1986, pp. 667–675

[8] DARPA "RFC 791: Internet Protocol, DARPA internet program, Protocol specification" September 1981

[9] IETF's Network Working Group "RFC 1519: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy", September 1993

[10] IETF's Network Working Group "RFC 1819: Internet Stream Protocol Version 2 (ST2), Protocol Specification - version ST2+" August 1995

[11] IETF's Network Working Group "RFC 1992: The MD5 Message-Digest Algorithm", April 1992

[12] IETF's Network Working Group "RFC 2113: IP Router Alert Option", February 1997

[13] IETF's Network Working Group "RSVP Extensions for IPSEC Data Flows", September 1997

[14] IETF's Network Working Group "RFC 2210: The use of RSVP with Integrated Service" September 1997

[15] IETF's Network Working Group "LDP Specification" October 1999

[16] IETF's MPLS working group "Constraint-Based LSP Setup using LDP", September 1999

[17] The ATM Forum, Technical Committee "Private Network-Netwok Interface Specification Version (PNNI) 1.0" March 1996

[18] The ATM Forum, Technical Committee "ATM User-Network Interface (UNI) Signalling Specification Version 4.0" July, 1996

[19] Lixia Zhang, Stephan Deering, Deborah Estrin, Scott Shenker and Daniel Zappala "RSVP: A New Resource ReSerVation Protocol" , IEEE Network Magazine. September 1993

[20] IETF's Network Working Group "RFC 2205: Resource ReSerVation Protocol (RSVP), Version 1, Functional Specification", September 1997

[21] IETF's Network working group, "RSVP-TE: Extension to RSVP for LSP Tunnels", draft-ietf-mpls-rsvp-lsp-tunnels-05.txt, February 1999

[22] IETF's Network working group. "RSVP refresh Overhead Reduction Extensions", draft-ietf-rsvp-refresh-reduct-02.txt, January 2000

[23] USC Information Sciences Institute (ISI), RSVP deamon implementation. http://www.isi.edu/div7/rsvp/rsvp.html

[24] Ping Pan & Henning Schulzrinne "YESSIR: A simple reservation mechanism for the internet", August 1997

[25] Arun Viswanathan, Nancy Feldman, Zheng Wang, Ross Callon "Evolution of Multiprotocol Label Switching" IEEE communication Magazine, May 1998 pp. 165-173

[26] IETF's Network Working Group "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification" draft-ietf-pim-v2-sm-01.txt November 1999

[27] IETF's MPLS working group, "IMPROVING TOPOLOGY DATA BASE ACCURACY WITH LSP FEEDBACK VIA CR-LDP", draft-ietf-mpls-te-feed-00.txt, February 2000

[28] IETF's Network Working Group, "Introduction to IP Multicast Routing", draft-ietf-mboned-intro-multicast-03.txt, July 1997

[29] David E. McDysan, Darren L. Spohn "ATM Theory and Application", McGraw-Hill, September 28, 1998, ISBN: 0070453462

[30] Israel Cidon, Inder Gopal, Roch Guéring "Bandwidth Management and Congestion COntrol in plaNET", IEEE Communication Magazine, October 1991, pp. 54-64

[31] I. Cidon, T. Hsiao, A. Khamisy, A. Parekh, R. Rom and M. Sidi, "OPENET: An Open and Efficient Control Platform for ATM Networks," in Proceedings of the Conference on Computer Communications (IEEE Infocom), (San Francisco, California), pp. 824, March/April 1998.

[32] I. Cidon, R. Rom and Y. Shavitt, " Multi-Path Routing Combined with Resource Reservation," in Proceedings of the Conference on Computer Communications (IEEE Infocom), (Kobe, Japan), pp. 92–100, April 1997.

[33] I. Cidon, R. Rom and Y. Shavitt, "Analysis of Multi-Path Routing," IEEE/ACM Transactions on Networking, pp. 885–896, December 1999.

[34] IETF Network Working Group, "Stream Control Transmission Protocol", draft-ietf-sigtran-sctp-13.txt, July 2000

[35] FPGA white papers can be found at: Altera (www.altera.com), Xilinx (www.xilinx.com)

[36] Network processors are designed and manufactured by many companies including: MMC Network (www.mmcnet.com), Intel (http://developer.intel.com), EZchip (http://www.ezchip.com/) and more.

[37] Cisco, "LightStream 1010 Multiservice ATM Switch Overview" ,http://www.ieng.com/warp/public/cc/pd/si /lsatsi/ls1010/prodlit/ls10m_ov.htm

[38] Shigang Chen and Klara Nahrstedt "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions", IEEE Network, vol. 12, no. 6, November 1998, pp. 64-79