Selective Truncating Internetwork Protocol: experiments with explicit framing

Mathias Engan, Pierre Fransson, Olov Schelén
{engan, pierre, olov}@cdt.luth.se
Luleå University of Technology, Luleå, Sweden

Abstract— **Many emerging real-time applications generate layered data streams, which can be used effectively to adapt multicast real-time transmissions to heterogeneous bandwidth environments. However, these applications have not changed with regard to being sensitive to transient congestion and cannot wait a full round-trip time for sender-initiated adaptation.**

In this paper we propose the Selective Truncating Internetwork Protocol (STRIP) supporting layered data transfer, and capable of handling congestion at a finer level of granularity by truncating packets, i.e., stripping off less important data. STRIP inter-operates with the traditional IP infrastructure and can be introduced in a step-by-step fashion, starting with routers where the benefits are obvious, such as routers connected to bandwidth constrained links with a surplus of processing capacity.

We describe the design and architecture of STRIP and compare it with solutions for differentiated forwarding on a per-packet basis. STRIP provides a simple mechanism that can meet the demands for real-time flows effectively by supporting low delay forwarding, avoiding data-unit reordering, and supporting various drop priorities at the same time.

I. Introduction

The desire to connect wireless thin clients, e.g., mobile phones and PDAs (Personal Digital Assistants), to the wired Internet is introducing an increasing amount of heterogeneity to the Internet. This heterogeneity applies both to available bandwidth resources and presentation capabilities. The intensified use of wireless communication will also likely increase the transient congestion experienced by wireless receivers, due to mobility and properties inherent to radio wave propagation (fading, absorption, etc.). In this context transient congestion refers to short term congestion that fluctuates too rapidly to be resolved through feedback mechanisms between sender or intermediate nodes and receivers. These developments coincide with the awaited breakthrough of streaming real-time data, e.g., audio and video, on the Internet arena, which will probably stand for a substantial amount of traffic in the future.

Since streaming real-time data both is capable of consuming vast bandwidth resources and is sensitive to jitter (caused by transient congestion), it is obvious that the above mentioned developments do not go hand in hand with each other. Especially noteworthy is that transient congestion can cause several consecutive packets to be dropped, thereby producing glitches in the playback of the real-time stream. The occurrence of such glitches has been noted to substantially mar end-users perception of the quality of a transmission[18].

When several receivers are interested in simultaneous transmission of streaming data it is possible to gain in bandwidth utilization through multicast. Unfortunately the use of multicast is hampered by bandwidth heterogeneity. To resolve this problem several networking solution have been proposed. Amongst these Receiver-driven Layered Multicast (RLM)[2] and Layered Video Multicast with Retransmissions (LVMR)[17] can be noted. These proposals utilize layered application data in conjunction with multicast to handle bandwidth heterogeneity. But while these solutions handle bandwidth heterogeneity they do not handle transient congestion conditions (due to usage of a feedback loop) and in some cases even cause it. Note that by transmitting several layers, using one packet for each unit of layered data, the amount of transmitted packets increase drastically. This behavior increases the processing overhead in core routers and may therefore prove detrimental to network efficiency.

In this paper we propose an alternative to the above mentioned solutions that is able to handle both bandwidth heterogeneity and transient congestion without incurring extra processing overhead in core routers and without the need to maintain any flow state.

Selective Truncating Internetworking Protocol (STRIP) is based on segmentation of the data carried in a single network layer frame, into two or more sections, chunks. Each chunk contains a single application layer, encoded using any preferred scheme. Every chunk is assigned a drop-priority, the closer a chunk is to the network layer frame-header the higher its priority. This assignment of drop-priorities allows the networking layer to drop chunks, instead of entire packets, when congestion is experienced. Allowing both for finer granularity in congestion control, capable of handling transient congestion conditions, and the ability to adapt multicast real-time flows to the requirements of a heterogeneous bandwidth environment. The priority scheme (where chunks closer to the network layer header have higher priority) also allows for efficient and fast removal of chunks, by simply dropping chunks at the end of a packet, i.e., truncating the tail of the packets.

II. Existing Approaches

This section presents alternative solutions that could be used to provide the same or part of the functionality that STRIP does. The main difference between STRIP and the presented solutions is that STRIP implements a priority scheme and truncating capabilities at the application data object level, while the alternatives implement priorities and discard on a per packet level.

Many interactive media applications such as, audio and video conferences, games or IP telephony have specific requirements[14]. They periodically transmit their data (such as 160 bytes every 20 ms for 64 kbit/s voice). They want a constant and preferably low delay with bounded variations. They can tolerate some packet loss – but not many consecutive lost packets. The output from an generalized conferencing application could be briefly characterized as a stream of many equally sized, small packets spaced out evenly over time.

Receiver-driven layered multicast[2] is a multicast based scheme in which applications transmit different data layers on different multicast groups. Receivers request data layers by joining multicast groups carrying the desired layers. Increased measured packet loss is a signal to a receiver that it might be oversubscribed, that it or the network is not currently capable of receiving the requested amount of data. The receiver can attempt

0-7803-6494-5/00/$10.00 © 2000 IEEE

to reduce packet loss (possibly enhancing the quality) by unsubscribing from a high volume layer.

Carpooling and gathercast[3], [4] is a method to aggregate small packets going to the same destination. The purpose is to reduce the overhead caused by small packets. Aggregators in the network detect and intercept small datagrams which are buffered and delayed waiting for more small datagrams sent to the same destination. A collection of small packets are bundled in a large datagram and forwarded toward their final destination.

On average, carpooling does not change the inter-packet delay. But, the buffering will increase the jitter. Carpooling reduces the number of packets in the network but does not provide mechanisms to prioritize small datagrams in a bundle.

Differentiated services [8], [7] (diffserv) introduces network level mechanisms to specify and enforce a certain service for individual datagrams. Datagrams are marked when they enter a network. Routers along the path of a datagram enforce the requested service within the bounds of a profile. Diffserv allows senders to assign priorities to transmitted data on a per packet basis.

With diffserv, applications using layered encodings send different layers in separate packets and mark them for appropriate treatment in the network. For the generalized conferencing application, the consequence of employing layered transmission in a diffserv capable network, is many small packets each containing an application data object from one layer. Each one of these small packets incur a fixed cost in every router. Routers are mainly limited in the amount of packets they can forward (not the amount of bytes) and the Internet's core routers are already heavily taxed, therefore future networking solutions should not increase this strain. It should also be noted that a scheme utilizing diffserv would incur a higher bandwidth overhead due to extra packet-headers, which becomes important in bandwidth constrained areas such as wireless links.

Low delay is important for real-time applications. Diffserv forwarding disciplines supporting delay-sensitive traffic, Expedited Forwarding [16], rely on strict policing (i.e., dropping of excess traffic) at edges and on isolating queuing in the network.

This model is not in itself appropriate for providing drop differentiation in the network, as packets of different isolated classes go in different queues in the routers and consequently may arrive at the receiver out of order and separated in time. Instead, diffserv provides forwarding disciplines that perform drop differentiation without packet reordering, Assured Forwarding [15], which are more appropriate for layered data. Drop differentiation occurs when average queue lengths grow to certain thresholds (as in RED, WRED, RIO, etc.) and therefore transient delay may be quite high, especially if drop precedence queuing is configured for good TCP throughput. On the other hand, if the drop precedence queuing is tuned for layered real-time flows, diffserv is feasible. However, the problem with many small packets still apply.

RLM and Diffserv combined allow media applications to inform the network of the relative importance of data layers. However, the overhead of many small packets would still apply.

In a heterogeneous Internet, with links ranging from high bandwidth wired links to low bandwidth wireless links, potential advantages from combining STRIP and diffserv can be won. Drop differentiation with low delay can be obtained by using single class diffserv forwarding with STRIP handling transient congestion. Admission control can be tuned to admit traffic rates closer to maximum link capacity if STRIP acts as a safety valve at times of transient congestion (adapting to temporary link-capacity changes). STRIP would also provide better bandwidth efficiency on wireless links due to less header overhead.

III. Background

The purpose of this section is to present work that has influenced and inspired the design of STRIP: application level framing, chunks and explicit framing.

Application level framing [1] proposes an end-to-end scheme where application data objects are framed and named. The purpose is to improve receiver performance and functionality. The application level framing information allows the network subsystem in the receiver to process an incoming application data object in one step. If the naming of application data objects provide temporal information, implicit priorities can be deduced by the receiver. For instance, an interactive audio tool can frame and label the application objects with a time-stamp. If objects are delayed in the network, the time-stamps are used to discard objects that are no longer useful to the application.

Feldmeier describes a method for explicit framing of application level data[5]. Chunks are objects where framing information is explicit and exposed to the network. The chunks are self containing, they carry all information required by the receiver and can be individually processed and delivered to the application. Datagrams (frames) consists of several chunks. Frames can be fragmented on chunk boundaries and chunks can be reassembled into frames. Chunks are named objects and can be delivered in any order. Ensuring the correct order are the receivers responsibility.

Exposing the application level framing information to the network enables improvements in the service offered by the network. Explicit framing, application level framing visible to the network layer, combined with gateway mechanisms extends the network service set. It provides the application with mechanisms to increase its control over the network, and the network can improve the processing and handling of a flow.

IV. Architecture

This section describes the architecture of STRIP in detail. First an overview of STRIP is presented, after which the design of STRIP is reviewed and finally the protocol specifics are presented.

The basic idea behind STRIP is to provide a networking layer that is capable of handling congestion at a finer level of granularity than today, i.e., only dropping a part of a network layer frame instead of having to drop an entire packet.

The advantage of utilizing such a protocol arises when layered application data embedded in the payload of a packet are partitioned into two or more logically independent segments , that are each associated with a priority level. When congestion is encountered, less prioritized segments in the packet can be dropped while still allowing higher prioritized segments to be delivered. The benefits of this approach are two-fold.

First, transient congestion can be handled without discarding entire packets. Although this means that the received data stream will vary in quality, it allows for uninterrupted reception of a media stream even in the face of congestion (e.g., the number of glitches in the reception is reduced). This becomes especially important in mobile scenarios where transient congestion is likely to fluctuate more rapidly than on the wired Internet, due to mobility and the inherent properties of the radio medium.

A second benefit is automatic adaption of multicast real-time data to receivers on links with heterogeneous bandwidth capabilities. This is possible since routers (upstream to receivers on low-bandwidth links) can discard excess layers when encountering congestion. Thus providing receivers on low-bandwidth links access to the same content at a lower level of quality.

A. Design

This section reviews the main design issues: generality, flexibility, efficiency and applicability, and tries to illuminate their impact on the properties of the protocol.

In order to ensure generality a networking protocol should assume as little as possible about the capabilities of the underlying link layer(s) and transport layer(s) above. It is difficult to foresee future developments and assumptions valid today may prove detrimental tomorrow.

Another aspect of generality is the ability of protocols to coexist with other protocols. This implies that protocols should ensure fairness towards other protocols with respect to consumed resources, e.g., bandwidth.

By implementing generality, a network layer usually also provides flexibility, i.e., different transport layers are able to implement diverse strategies and still utilize the same network layer. Flexibility is important when the network layer provides truncating capabilities as is shown in the following two examples: 1) Considering that there is a plethora of layered coding strategies, and new strategies are being developed, the segmentation policy of a truncating protocol has to be flexible enough to handle todays codecs and provide support for tomorrows. 2) It is common for transport layers to provide protection, e.g., CRCs and checksums, of the data to be transmitted. If a truncating protocol is used it is obvious that these protection schemes will indicate that a package is damaged if it has been truncated. A truncating network layer therefore has to provide means for the transport layer to reconstruct the protection scheme even when a packet is truncated without resorting to layer violations. In other words, by providing new and extra functionality, it becomes increasingly important to ensure that the network layer still provides sufficient flexibility.

It is of course important that a protocol is efficient with respect to all resources it utilizes, i.e., bandwidth and end-node and router processing resources. Since STRIP introduces truncating capabilities it is essential that gateways are able to drop chunks with a minimum of effort, which is the reason for STRIPs priority assignment. By assigning lower priorities to chunks at the end of a packet, gateways are able to remove chunks *by simply changing the length of incoming packets*. Although the change of length is a minor one, it has impact on the efficiency of truncation. This is due to the (minimal) protection that network layers usually offer.

Note that although STRIP is a network layer protocol, it does not imply that all routers have to actively support truncation. In fact, a likely scenario is that routers connected to bottleneck links (where bandwidth and not processing power is the scarce resource) would perform truncation. Core routers connected to high-speed backbones would however not perform truncation as the scarce resource in this case is processing power, not bandwidth. Since STRIP does not require all routers to support truncation it provides for efficiency by utilizing the network resources that are most abundant at any location: processing power or bandwidth.

Even though a network layer protocol should provide generality and flexibility, it is not certain that the truncating capabilities of STRIP are equally useful to data from all types of transport layers. For example, while real-time streaming data transmitted using UDP can easily benefit from truncation, it is not straightforward to take advantage of truncation in segment and window based feedback protocols, such as TCP.

A.1 Chunk Characteristics

There are several chunk characteristics that have impact on the design issues stated above. The following section analyses the effect of different design choices with regard to each characteristic.

Chunk quantities. A fixed number of chunks in each packet provide potential for greater efficiency while a dynamic number provide a more flexible solution. We chose the dynamic setting, since we believe that flexibility is the more important quality for a truncating network layer.

Chunk positioning. For efficiency reasons we chose to place chunks back-to-back, placing chunks relative to each other according to their priority, with lower priorities at the end of the packet.

Chunk addressing. Since we have chosen to accommodate a dynamic number of chunks, it is not possible to have a fixed placement for them. Each chunk is therefore addressed using an offset, indicating its position relative to the network layer header.

Chunk information. Each chunk has to be associated with information from the transport layer to provide a means for repair of the transport layers protection scheme.

A.2 Queuing Scheme

An integral component of STRIP is the queuing scheme in routers, that decides when and to what extent a packet should be truncated. The main objective of the queuing scheme is to try to ensure that no single stream or group of streams is given preferential treatment. It is also essential that the queuing scheme does not impact adversely on non-STRIP traffic, e.g., TCP, as this would violate the generality requirement. However, the queuing scheme should still provide an incentive for using STRIP for data traffic that can be layered, e.g., multimedia streams. Otherwise it would be possible for a sender to acquire an unfair share of bandwidth by simply sending its multimedia stream in monolithic packets (i.e., packets that cannot be truncated). Devising such a queuing scheme is a general problem, common to all architectures that provide service differentiation, and not a problem unique to STRIP.

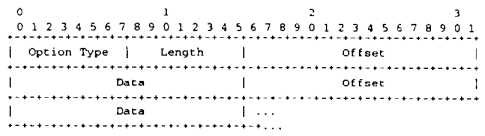

Fig. 1. STRIP IP option syntax

B. STRIP

This section describes the operation of STRIP. STRIP is unidirectional and the overview follows the dataflow in the protocol.

The sending application generates application data objects and assigns them to different layers according to their importance for the quality of transmission.

The transport protocol receives the data objects from the application. Each data object is processed and transport protocol specific error control information is generated. A transport protocol header is generated for every transport protocol data unit (TPDU) composed of application data objects belonging to several layers. The TPDU is handed down to the network layer together with error control information.

The network protocol creates a STRIP header, containing STRIP framing and error control information (see section IV-B.1). The STRIP framing information is available to the network layer.

A gateway detecting problems such as congestion, responds by dropping layers from STRIP datagrams. The gateway monitors the current situation in its local environment, for example by paying attention to interface queue lengths. The congestion detection algorithm decides on a current maximum size for STRIP datagrams, the current maximum size is compared to the size of an incoming datagram and layers are dropped until the datagram is small enough. Only the application data objects are dropped, the STRIP header is retained for use by the receiver when reconstructing the TPDU. In addition to the *IP TTL* decrease required by the IP forwarding algorithm, a STRIP gateway changes the *IP length* (and of course the *IP checksum*) after discarding a layer.

A receiver uses the information in the STRIP header to restore a truncated frame to a correct TPDU. The network layer passes STRIP header information on to the transport protocol, and the transport protocol compares the size of the received PDU with the length field in the transport layer header. If the PDU is smaller than the field indicates, one or more layers must have been dropped, then the header must be patched using information from the STRIP header.

B.1 Protocol Syntax

The network layer part of the STRIP protocol is currently implemented as an IP option[9] (see figure 1). The first two fields (*Option Type* and *Length*) are required by the IP option framework and is not a part of STRIP. The STRIP header is composed of a sequence of layer headers, each layer header contains the *offset* and *data* field. The first layer header corresponds to the least important chunk.

Offset The *offset* field is an index to the first byte of the corresponding data in the following PDU.

Data The *data* field contain error control information. The semantics of the *data* field is determined by the transport protocol used (the value of the *IP Protocol* field).

B.2 STRIP for UDP

The User Datagram Protocol[10], is an unreliable connectionless transport protocol often used for real-time data transmission. The main modifications required to adapt UDP to the IP/STRIP networking layer are: a) independent processing of separate application data objects, and, b) calculation of partial checksums.

UDP is modified to accept separate application data objects. When handing data objects to UDP the application also indicates which layers they belong to. UDP then processes the objects individually and combines them to form one TPDU. The main task during processing of data objects is checksum calculation. UDP uses the standard IP checksum[11], [12], [13] for end-to-end error control.

The properties of the IP checksum allow partial computation of checksums and they can be combined to form the UDP checksum, basically by addition.

V. Conclusion

With the awaited increase of streaming real-time data and added heterogeneity, the Internet faces some problems. The main problems that we address are: enabling receivers with heterogenous bandwidth requirements to receive multicast real-time data flows and coping with transient congestion.

As a solution to these problems, we propose a new internetworking protocol that enhances todays Internet service model to include truncating capabilities and a priority scheme to be used for transmission of layered real-time data. By allowing the network layer to drop parts of a packet, where each part contains information from a single layer, it becomes possible both to adapt to heterogenous bandwidth requirements and resolve transient congestion thereby lessening the probability of creating glitches.

Although earlier proposals have tackled parts of the problem, e.g., RLM and LVMR, the problem of transient congestion has not been dealt with in a satisfactory manner. In some cases proposals may even initiate and create transient congestion.

As part of an undergraduate project, a STRIP enabled router, running on FreeBSD 4.0, has also been created. The router is capable of truncating packets according to preset levels. The router does not yet support a queuing scheme.

In the nearest future we will implement STRIP in the networking simulator NS and examine the performance of STRIP compared to alternative solutions. We will also examine the effect STRIP traffic has on non-STRIP traffic. A live implementation of STRIP, including sender, receiver and router, will be implemented later and will provide an opportunity for real-life tests.

References

[1] Clark, David D., Tennenhouse, David L., *Architectural Considerations for a New Generation of Protocols*, Proceedings of ACM SIGCOMM '90, Sept. 1990, pp 201-208

[2] McCanne, S., Jacobson, V., Vetterli, M., *Receiver-driven Layered Multicast*, Proceedings of ACM SIGCOMM'96, pp. 117–130, Aug. 1996.

[3] Badrinath, B. R., Sudame, P., *Car pooling on the Net: Performance and Implications*, Tech report, Rutgers University, May 1999, also in ACM SIGCOMM'99 New Research Session, Sept. 1999.

[4] Badrinath, B. R., Sudame, P., *Gathercast: An efficient multi-point to point aggregation mechanism in IP networks*, Work in progress (submitted for publication).

[5] Feldmeier, D. C., *A data labelling technique for high-performance protocol processing and its consequences*, Proceedings of ACM SIGCOMM'93, pp. 170–181, Sept. 1993.

[6] Degermark, M., Pink, S., *Issues in the Design of a New Network Protocol*, Proceedings of the 3rd COST 237 Workshop on Multimedia, Telecommunications and Applications, Vol. 1185 of Lecture Notes in Computer Science, pp. 169–182, Springer-Verlag, 1996.

[7] Clark, D., Fang, W., *Explicit Allocation of Best Effort Packet Delivery Service*, IEEE/ACM Trans. on Networking, Vol. 6, No. 4, pp. 362–373, Aug. 1998.

[8] Carlson, M., Weiss, W., Blake, S., Wang, Z., Black, D., Davies, E., *An Architecture for Differentiated Services*, RFC 2475, IETF Network Working Group, Dec. 1998.

[9] Postel, J. ed., *Internet Protocol*, RFC 791, IETF Network Working Group, Sept. 1981.

[10] Postel, J. Ed., *User Datagram Protocol*, RFC768, IETF Network Working Group, Aug. 1980.

[11] Braden, R., Borman, D., Partridge, C., *Computing the Internet Checksum*, RFC 1071 IETF Network Working Group, Sept. 1988.

[12] Rijsinghani, A., *Computation of the Internet Checksum via Incremental Update*, RFC 1624, IETF Network Working Group, May 1994.

[13] Mallory, T., Kullberg, A., *Incremental Updating of the Internet Checksum*, RFC 1141, IETF Network Working Group, Jan. 1990.

[14] Clark, D. D., Shenker, S., Zhang, L., *Supporting Real-Time Applications in an Integrated Packet Services Network: Architecture and Mechanism*, Proc. ACM SIGCOMM'92, 1992.

[15] Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., *Assured Forwarding PHB Group*, RFC 2597, IETF Network Working Group, June 1999.

[16] Jacobson, V., Nichols, K., Poduri, K., *An Expedited Forwarding PHB*, IETF Network Working Group, June 1999.

[17] Xue, L., Sanjoy, P., Mostafa, A., *Layered Video Multicast with Retransmissions (LVMR): Evaluation of Hierarchical Rate Control*, Proceedings of IEEE INFOCOM'98, 1998.

[18] Partridge, C., *Gigabit Networking*, Addison-Wesley Publishing Company, 1994.