Technical Report 2002-03-01
Internetworking and Media Communications Research Laboratories
Department of Math & Computer Science, Kent State University
http://medianet.kent.edu/technicalreports.html

FAST STREAM INTERCEPT FOR HIGH PERFORMANCE FILTER APPLIANCES

Javed I. Khan & Yihua He

Media Communications and Networking Research Laboratory
Department of Computer Science, Kent State University, 233 MSB, Kent, OH 44242
javed@kent.edu

March 15, 2002

1. Introduction

Stream interception is rapidly becoming a common and important task in Internet appliances. Beginning from cache, proxy, filters, firewalls, and gateways, there are now host of new services including content adaptation, content personalization, location-aware data insertion, to security filters -- all are fundamentally stream interception machines requiring some form of intermediate access inside transiting traffic's content. By some very conservative estimate almost 40% percent of the delivered Internet traffic is now 'touched'. Yet new rush of intercepting applications will arrive with the advent of emerging technologies such as programmable or active networks.

However, if we look into the current protocol design, particularly the protocol packet structures we will see that it has little facility which enables efficient stream intercept. While, fixed end-to-end applications can do away with marginal treatment of this issue, indeed, we believe right placement of protocol element inside data stream and some form of random access will be one of the most important factor for high performance stream data processing appliances.

1.1 Potential Pay-off: Active traveler with a jumbled up suitcase?

The problem can be illustrated by a traveler's analogy. Today, our traveler is packing his suitcase by almost randomly throwing in objects, and closing the overfilled suitcase by sitting on it (that's compression)! These intercepting appliances at stops are asking him/her to open his overfilled suitcase to show specific things. At every stop he has to take all the things out, put them back and recompress. To make the matter even worse this poor traveler has not just one but many suitcases, and the demanded items are scattered all over! A performance meltdown is inevitable. Imagine the jam in airport security check. Few hints about the items to be demanded- and some small organization in packing can enormously speedup the process.

Most real-life content, particularly high performance networked multimedia transport data carried over a network packet are multi-level hierarchically encapsulated (that means bags within suitcases). For example access to a Dolby AC-3 component in a standard audio stream may require as much other 5 different data elements before a particular fields can be accessed in current arrangement. An H.261 video element may require access to 16 pre-dependent elements before the element of interest can be read! The situation is also difficult for recent tag-delimited content protocol standards (such as XML, HTML), where this dependency is formally infinite. The problem has enormous implication on the CPU cycle, memory size, and overall performance of any intercepting appliance system's architecture. The

stream is the working data structure of these capsules. It is perhaps as salient to appliance's overall architecture as the design of disk scheduling algorithm or multilevel memory/cache organization is to the conventional machine architecture.

In this research we focus on this important yet little explored problem and demonstrate a novel content indexing scheme which can facilitate dynamic index based random access into streams and provide serious performance boost to intercepting filter like appliances. Though conceptually the mechanism can be implemented in layers above IP, we present an IPV6 based protocol called *Embedded Data Indexing Protocol* (EDIP). It is an IPv6 extension header based content indexing mechanics, which defines the how a Content Provider (CP)'s serverlet can add special marks into the data stream, and how an Active Router (AR) can decode those marks from the data streams and gain pattern dependent random access into the elements of required data stream. In this extended abstract we provide a brief summary of the system and its performance. The full report will provide further details of the issues.

1.2 Few Related Work

S. Blake discussed about the differentiated services by adding marker field DS in IPv4 and IPv6 headers [1]. Packets marked by this field will receive a particular per-hop forwarding behavior on nodes along their path. It is a close approach as this report's. However, they didn't investigate the possibility to add indexing information into IP headers and utilize it in value-added service to make random access of the data stream available. In other extreme Akamai [3] and ESI [4] use a full application level XML-like markup language to define web page components for dynamic assembly and delivery of web applications at the edge of Internet. Spatscheck et. al. [7] and D. Maltz and E Bhagwat have separately presented two TCP splicing mechanisms which would allow a filter

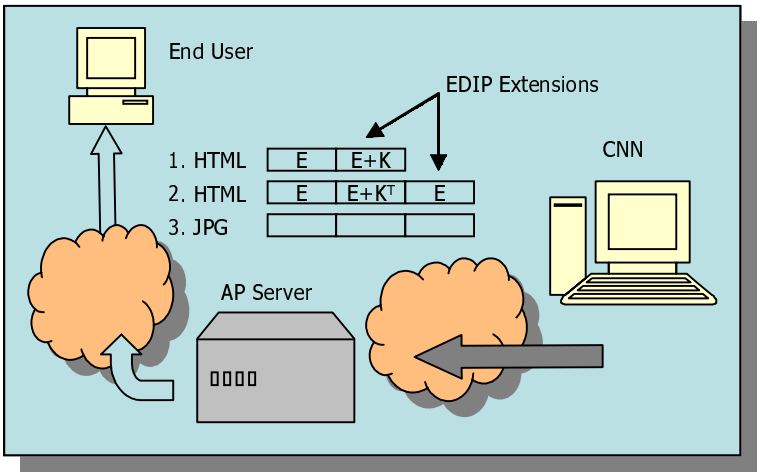

Fig 1: Example Service with EDIP Header

(connected by two TCP links at two ends), to shed-off some TCP window maintenance functions, for passive filters by splicing the two TCP stream at two end-points.

The technique we propose accelerates the actual filtering operation and applications, as much as it helps the networking layers. Also, the gain is not restricted for passive mode of operation. It uses network layer markup mechanisms to avoid decapsulation of non essential application data (stream segments). Also, a key difference is that we include the case of cooperative application processing in the service model where server side help may also be available.

2. In Route Application Service

First we explain the service model. In the service model a *content stream* from *content provider's server* (CP) flow to the *end-user* (EU). However it may also be processed in an ISP *application processing* (AP) server in between during transit. The end user initiates the content delivery by requesting content

from the content provider via Internet. The Application Service Provider (ASP) modifies the content and adds value to the communication by application level intercept processing at strategically and/or topologically located AP servers. In special cases the CP and AP can be collocated in application service provider's AP.

A special case of AP intervention is the *passive filtering* service where AP server only monitors the stream without changing it. A further special case is

the *stealth filters* where servers or end-users are not aware of the intercept service (and thus also not helping). If the content provider is also willing to help we call it *co-operative filtering*[1].

The AP server additionally can provide "content cache". The cache can connect at both 'pre' or 'post' AP stage. Conceptually, caching is just another piped service which AP can provide. AP server can be configured to provide multiple services piped on a specific request/response stream-- caching can be one of them. The *piping sequence* is soft configurable. Complex application service can be composed from simpler services by *service piping*. The connection between EU, CS, and AP servers are provided by point-to-point separate TCP/IP or UDP connections.

3. EDIP Indexing Mechanism

The operation of application processing is expedited by two techniques. The first is pre-marking the content stream and allowing fast access into to the stream. Second is the selective decapsulation reencapsulation of only the pertinent data segments. Finally, we also define a language to express and carry the marks between the parties involved.

The actual content intercept processing is performed by a program called the application *filter capsule*, and it runs on AP server. The application service provider generally also sends a *marking servlets* to the CP server for marking of the content stream. Every Application Service Processing have a specified "*scope segment*" and a "*key segment*" in it. Generally a service is conditional. The data element which contains the condition or key is always intercepted and is decapsulated and delivered to the application capsule. The stream segment which is within the scope of an active key is intercepted and buffered However, its decapsulation and delivery can be

deferred based on the key evaluation result. If the evaluation is false, it is directly forwarded. Fig-2 is the schematics of the enhanced network layers that we have designed for the appliances machine.

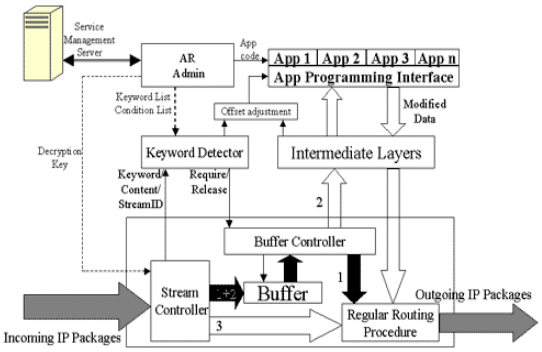

Fig-2 EDIP Selective Decapsulation System

4. EDIP Header Format

EDIP uses IPV6 extension header for content marking. It contains two parts: the General Field (GF) and Key Blocks (KB). The General Field (GF) identifies that it is an EDIP header, and contains general information in how to process the header. Each Key Block (KB) represents a keyword in this IP package, with positions of the keyword indexed by the offsets. Not every EDIP header has one or more KBs. Sometimes, an EDIP header may only have a GF, representing that the current IP package belongs to an indexed stream, while there is no key word appearance in this package. The total number of KBs

[1] For non-co-operative filtering some extra fast string matching operations are needed at the AP server.

that an EDIP can have is only limited to the maximum size of an IP package.

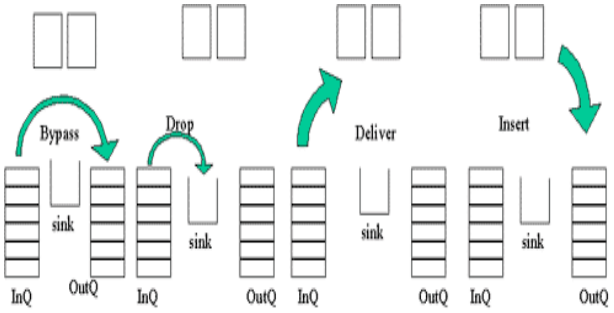

Fig 3 Stream Edit API Operations.

4.1 EDIP Encapsulation by Servelets

After capsulated by TCP/UDP, data stream can pass through multiple servelets in the servelet pipe. Each filter is associated with exactly one keyword and it examines the passing stream to see if there is any keyword appearance inside. If there are one or more appearances, the filter generates a key block containing the offset information about where the keyword is in the stream. Later, these key blocks join the original data stream in the general field generator, where a GF, as well as the key blocks, will be added at the beginning of each package.

4.2 Decapsulation and Services APIs

EDIP decapsulation and value-added services are executed in the ISPs Filter Server (FS), which sit on the edge of Internet backbone. There are several tasks that an FS must do. (1) Differentiate the IP packages that need special processing from those normal IP packages. (2) Retrieve the offset information from the special-marked streams to the corresponding applications, which may use the information for value-add-in service. (3) Negotiate with SMS and maintain the service statistics. The main components include a stream controller, a keyword detector and a buffer controller.

API	comment
Associate (inQ, outQ)	Associate two streams
Bypass(aid, a, b)	Forward bytes from a to b
Drop(aid, c, d)	Drop bytes c to d (into trash sink)
Deliver(aid, e,f, &*msgbuffer*)	Deliver bytes from e to f with *newcontent*
Insert (aid, *msgbuffer*)	Insert the messagebuffer content to the stream.

Table-1 Stream Edit API Subset

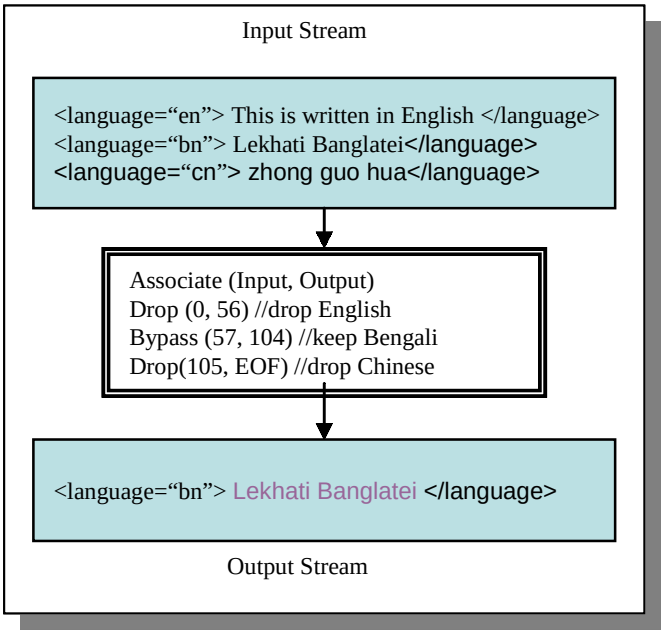

Fig-4 Example of a content processing using stream edit API

4.3 Application Processing:

The filter programs are armed with a set of special services APIs to take advantage of the marking process. With a scope data it can edit a stream using the stream edit APIs. Fig-3 explains them. The application program can use them to edit, bypass, drop, or insert bytes with a sequence stream of

incoming data. The buffers are application buffers. Each of these operations is performed within the context of an incoming and outgoing TCP socket stream pair. Fig-4 shows an example of a stream-edit capsule and its edit operation on a stream. The stream offsets are algebraically calculated from key indexes supplied by EDIP. There are additional setup APIs, with which it can enable/disable the tracking of keys by activating/deactivating the servelets and the intercept mechanism beneath. It can request for the next offset for a particular key. If the key test is successful (or unsuccessful), it can request (or release) delivery of the scope data. The AP capsules are also given a set of fast string search and protocol parsing routines (with potential hardware accelerators).

5. Benefits

The proposed mechanism accelerates the application level intercept process. The advantage is derived essentially by three principal sources -- (i) Only the byte segments carrying 'keys' are unconditionally decapsulated. (ii) The byte segments carrying 'scope' are conditionally decapsulated only when the key conditions are true. (iii) Rest of the bytes is never decapsulated.

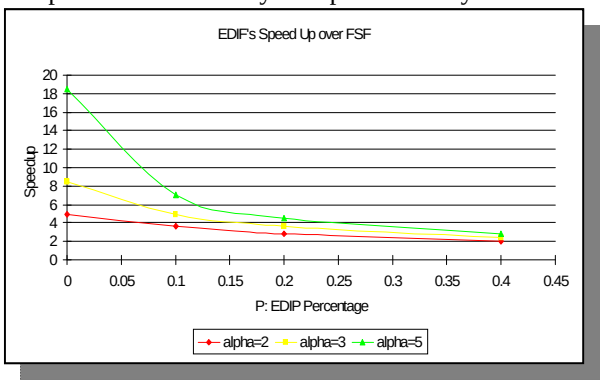

Fig-5 EDIP Speedup

This is also another source of run-time performance boost. Stream is marked by the serve let processes running at the content source. In cases, it is sometime possible to mark with direct content knowledge by the content generator without any string search. Otherwise, the marking can still be performed by sting search/ or parsing of the original content as preprocessing. It still therefore can drastically reduce the run time cost. To compare—current filters have to perform run-time full search and/or full parsing. The scheme however has cost. It is the extra data that will be needed by the EDIP markers. The actual saving therefore is the function of *key density*, and the *key success probability* in the stream. Though, apparently it may seem that high key density can offset the performance gain, but in practice always the EDIP key density can be controlled, by using a gross key in EDIP and then using application level processing to find the real keys. This is benefit of application level soft key definition ability. In practice, only a small part of data stream is generally modified. Consequently, the expensive part is way too inconsequential compared to the saving made by bypassing the costly decapsulation/ reencapsulation of the rest. In fig-5 we share a plot showing the speedup in EDIF filter with respect to a conventional full search filter (FSF). The speedup is plotted with respect to the percentage of packets which are carrying the key (p) for three different relative decapsulation costs (α). The saving from decapsulation increases as the speed differential in the decapsulation increases. As shown with 10-20% EDIP density about 200-500% times speedup is possible my EDIP marking. The full report will provide more detail space cost and speedup analysis under various network, application, and service scenarios.

6. Comments

Fast intercept of streamed data is a growing concern in networking. The application level embedded processing is rapidly increasing and can be a potential bottleneck in Internet traffic carriage. The network protocols and packet data structures have been

designed mostly for end-to-end processing. In this report we have presented a part of our research that looks into mechanisms which can provide random access in a stream. In [9] we present a protocol for deployment of such filtering service on an application services network.

7. Reference:

[1] S. Blake, D. Black and etc., RFC 2475 “An Architecture for Differentiated Services”, 1998

[2] Christian Huitema, “IPv6, the new internet protocol”, second edition, Prentice Hall, 1998

[3] Akamai, http://www.akamai.com

[4] ESI, http://www.esi.org

[5] S. Deering, R. Hinden, “Internet Protocol, version 6 specification”, RFC 2460, 1998

[6] Wei-Ying Ma, Bo Shen and Jack Brassil, “Content Services Network: The Architecture and Protocols”, International workshop on web caching and content distribution, June 2001.

[7] Oliver Spatscheck, J. S. Hansen, J. H. Hartman and L. L. Peterson; Optimizing TCP forwarder performance, IEEE/ACM Trans. Networking 8, 2 (Apr. 2000), Pages 146 - 157.

[8] D. Maltz and E Bhagwat, "TCP splicing for application layer proxy performance," IBM, ftp://ftp.cs.cmu. edu/user/dmaltz/Doc/splice-perf-tr.ps, Mar. 1998.

[9] 1. Javed I. Khan & Yihua He, Ubiquitous Internet Application Services on Sharable Infrastructure, Technical Report: 2000-03-02, Kent State University, [available at URL http://medianet.kent. edu/ technicalreports.html, also mirrored at http:// bristi.facnet.mcs.kent.edu/medianet]

The work is currently being funded by the DARPA Research Grant F30602-99-1-0515 under its Active Network initiative.