Tags for High Performance Active Networks

Tilman Wolf[1], Dan Decasper[1], Christian Tschudin[2]

[1]{wolf, dan}@arl.wustl.edu
Applied Research Laboratory
Washington University, St. Louis, MO, USA

[2]tschudin@docs.uu.se
Department of Computer Systems
Uppsala University, Sweden

Abstract–**We propose the use of "selectors for (active) packet flows" similar to tags employed in the IP world. Their impact on the performance of an active network node is significant, as active packets have to be demultiplexed not only to the network layer, but all the way to an application level Execution Environment. We have built an Active Network Node that implements the selector-based Simple Active Packet Format (SAPF). Our measurements show that SAPF packets can be processed 30% faster than regular IP packets that use the traditional Active Network Encapsulation Protocol (ANEP) header.**

Key Words: **active networks, high performance, tag switching, protocols**

I. Introduction

Active networking (AN) research [1] addresses how to bring flexible customized processing into the network. This is achieved by adding instructions to control and data packets that describe how to process it. Customizable packet processing requires an active router that provides Execution Environments (EE) for execution of these instructions. The EEs implement security and resource sharing schemes to allow a safe execution of the custom code.

The design space of an active network platform can be divided into three elementary components: the hardware platform, the software which consists of the node's operating system (NodeOS) [2] and a set of Execution Environments [3], and the protocol used for communication between active network nodes. This paper puts emphasis on a streamlined protocol between active nodes and the efficient forwarding of active packets through the OS and EE components on an active node.

While most AN software architectures [4], [5], [6], [7] address network functionality, security issues, and resource control, only recently there has been emphasis on the performance of these systems [8]. Designing active network systems for high performance is particularly important since active networks inherently shift complexity from the end-systems towards the more heavily loaded routers inside the network. Thus, a successful, wide-spread deployment of active network technology cannot be realized without components that can compete with the performance of today's routers.

To provide the necessary computational power, specialized hardware for active networks has been realized in form of Field-Programmable Gate Arrrays (FPGA) [9] and active router line cards [10]. Active processing chips with multiple processors, cache, and memory on a single ASIC have been proposed [11].

On the protocol level, only one solution for active networks has been proposed so far, the Active Network Encapsulation Protocol (ANEP) [12]. ANEP works on top of IPv4/IPv6 and provides the following functionally:

- *Demultiplexing* of packets to their corresponding EE,
- *Minimal default processing* for packets for which the intended EE is unavailable,
- *Storage space* in the ANEP header for information that conceptually or pragmatically does not fit into the encapsulated packet (i.e., security headers).

Fast demultiplexing is a very important criteria for high-performance active networking, since the active processing on a node requires that all active packets are demultiplexed up to the active processing engine, not only to the network layer, as in traditional networks. Though it can be argued that default processing and security are required in ANEP, we believe that this additional complexity limits efficient demultiplexing. Instead of putting complexity into the "active header", we believe that functions, like security, should be implemented in the NodeOS and the EEs. Active headers should solely identify the code handler to which a packet has to be forwarded.

We introduce the Simple Active Packet Format (SAPF) [13] which addresses these issues. The SAPF header merely indicates the flow association of a packet to the downstream node by using a "selector" or a "tag". Since the downstream node has state information associated with the flow, it can easily access this information by

This research was supported by DARPA under contract number N66001-98-C-8510.

hashing with the selector as a key. This reduces a complex flow classification and demultiplexing over several packet headers to a simple table lookup.

The paper is organized as follows: Section II introduces the SAPF packet format and shows how selectors are assigned. We elaborate on the process of how selectors are exchanged between nodes and how nodes that use SAPF interoperate with regular IP nodes. In Section III we describe our software and hardware implementation of a high speed active networking node using SAPF. A quantitative comparison between SAPF and IP/ANEP is made in Section IV. Related Work is addressed in Section V. A summary and future work conclude this paper in Section VI.

II. Tagging Active Packets

While datagram traffic caused by web clicks accounts for the largest share of traffic in the current Internet, longer sequences of packets from a sender to a destination (called flows) are becoming more common. Media applications, like real-time audio and video, are the most popular examples. Such flow-based applications have been shown to benefit greatly from the flexible processing on active routers to adapt to traffic patterns, like congestions [14]. In traditional networks, a drawback of flow-based processing was that nodes had to keep state information for every flow. But active networks are literally bound to establish state in the network nodes. Thus, active networks should explicitly support the notion of flows and allow fast demultiplexing on a per flow basis.

Traditionally, flow state is accessed by performing a flow classification on certain fields of the IP and higher layer headers. If these fields are always at the same location in the header, this flow classification can be computed very efficiently [15]. In active networks, though, the location of the classification fields in the ANEP and EE-specific header can change due to the variable length of the ANEP header and the variable number of header options. Also, it is possible for the header of the EE to change within the same flow. This makes packet classification complex and slow.

The Simple Active Packet Format aims at reducing header information to a minimum in order to make demultiplexing, which is a time critical and common task in packet forwarding, more efficient. In this section we highlight the salient features of SAPF.

A. Simple Active Packet Format

The Simple Active Packet Format [13] consists of a 64 bit header that contains a one bit field specifying the version and a 63 bit field called the "selector". The format is shown in Fig. 1. The selector is an identifier for the flow or class that the packet belongs to. The payload field contains arbitrary content designated for the handler

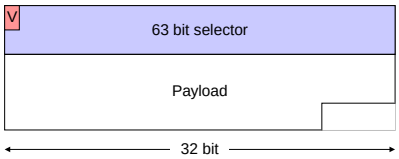


Fig. 1. Simple Active Packet Format (SAPF).

or EE that processes the packet. Other information, such as datagram length and link layer addresses are provided by the underlying protocol layer. While this extreme simplification poses some restrictions on the flexibility of the protocol (i.e., the packet's total length is limited to the the link layer maximal transfer unit), it allows extremely efficient demultiplexing: A single selector field can be used as key into a hash table.

B. The Semantics of SAPF Selectors

At a router, an incoming packet is demultiplexed based on its selector value and passed to the corresponding handler. A handler can be an Execution Environment in user space or a kernel data structure with an associated function. Selector values belong to different categories that are characterized by the way selector values are assigned. While some selector values are statically assigned, most commonly selector values are assigned dynamically, either by the upstream or the downstream node.

In any case, it is important that the selector value uniquely identifies the flow at the receiver side. For dynamically assigned selectors where a node receives transmissions from more than one entity (e.g., broadcast medium), a protocol has to be established on how senders and the receiver can agree on unique identifiers.

- *Static selectors*: these selectors have predefined, "well-known" values. These values can be assigned by an independent numbering authority, e.g., by ANANA (Active Network Assigned Numbers Authority). They are used to identify static handlers which are defined and installed, usually out-of-band, by an administrator. Examples of such handlers are EEs and protocol processing handlers like an IP stack. Static selectors can be used for non-active traffic (i.e., regular IP), for traffic that is not flow-bound (i.e., all HTTP traffic), or to set up new flows by sending active configuration messages to neighboring nodes. The initial static selectors will be replaced by dynamic selectors for subsequent packets of the same flow.
- *Receiver-assigned selectors*: these selectors are assigned by the downstream node. The selector for a certain flow is either requested by the upstream node via a control message, or the downstream node initiates the usage of the selector. With the selector assignment, a state record for the flow is created. Since the selector can be chosen arbitrarily by the downstream node, uniqueness can be assured.

• *Sender-assigned selectors*: we reserve a class of selectors to be solely chosen by the sender for registration of a handler without prior coordination with the receiver. This scheme does not guarantee unique selectors at the receiver side. To avoid complications arising from doubly used selectors, each sender can reserve a "block" of selectors at startup time from which it can choose selectors as they are needed. Applications for this type of selectors are flows that want to avoid the link round-trip delay introduced by receiver-assigned selectors.

C. SAPF Packet Processing and Forwarding

SAPF demultiplexing is done inside the kernel. As packets are delivered from the interface card, the kernel has to decide how to process them. For this, it consults the SAPF Forwarding Information Base (FIB). The FIB is similar to the tag information database described in [16]: It associates a handler instance to each SAPF identifier. Handlers are responsible for looking more deeply into the packet or for plain forwarding to another interface card. During forwarding, the kernel also performs SAPF header rewriting, if required.

D. Tags for Active Networking

SAPF requires a setup process between neighboring Tags for Active Networking (TAN) nodes. First, we consider a network consisting of only TAN nodes and show how packet handlers and associated SAPF selector values are set up and assigned. Second, we discuss interoperability between IP and TAN nodes and illustrate this with the example of a real-time video flow.

D.1 TAN Networks

The space-time diagram in Fig. 2 shows two TAN end systems connected by two TAN routers. This is an example where selectors are established before the first data packet is sent. Assume the sender on the left side wants to transmit a data stream to the receiver on the right side. Before sending any data, the sender creates a state record (an instance) with information for the active processing of the data stream and queries the downstream router for a selector. When receiving the selector request from the sender, the first router creates a state record for active video processing on the new flow. It then chooses a selector that is locally unique, replies with a control message to the sender, and forwards a selector request to its downstream router. On reception of the reply from the router, the sender associates the selector with the instance created before. The same procedure applies to the second router and the receiver.

The sender starts sending data on the negotiated selector as soon as it receives the reply from the first router. It does not have to wait for the full selector setup cycle to complete. All subsequent packets are forwarded on the

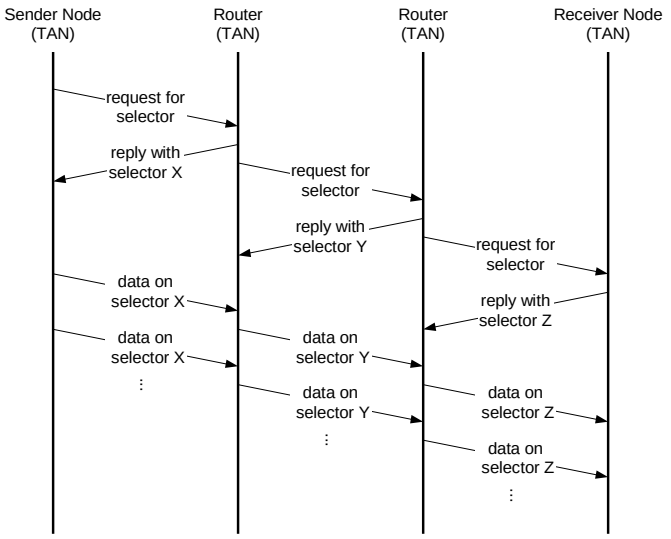


Fig. 2. Selector setup for TAN nodes and receiver-chosen selectors. The state information for the flow is set up when a node receives the request for a selector.

negotiated selectors. The binding of the selector to a state record is similar to the creation of a channel described in [3]. State records in all active devices expire if they are not used after some configurable amount of time (soft-state). This is similar to the soft-state behavior of RSVP [17] implementations. Therefore, there are no explicit tear down messages required to free instances.

D.2 IP - TAN Gateways

To illustrate interactions between regular IP routers and TAN nodes, we consider the example shown in Fig. 3. Sender and receiver communicate with IP/ANEP active packets through a cloud of TAN routers. The two plain IP routers adjacent to the TAN nodes are not capable of active processing or using selectors. This is an example where no state is setup before the first data packet is transmitted.

Since the sender is not capable of negotiating selectors, it starts sending regular IP/ANEP packets (we assume the active header can be understood by the end-system). The first IP router forwards the packet to the first TAN node. At the TAN node, the active packet is demultiplexed to the video processing code. After processing it is forwarded to the next TAN node and so on to the receiver. Both TAN nodes recognize that the packet is active and it might be more efficient to use the SAPF protocol for transmission. Therefore each TAN nodes chooses a selector for the flow and sends a control message to the upstream node informing it about the selector value. The control message reaching the first TAN node causes all following data packets to be transmitted on selector Y. The regular IP router, though, does not understand the

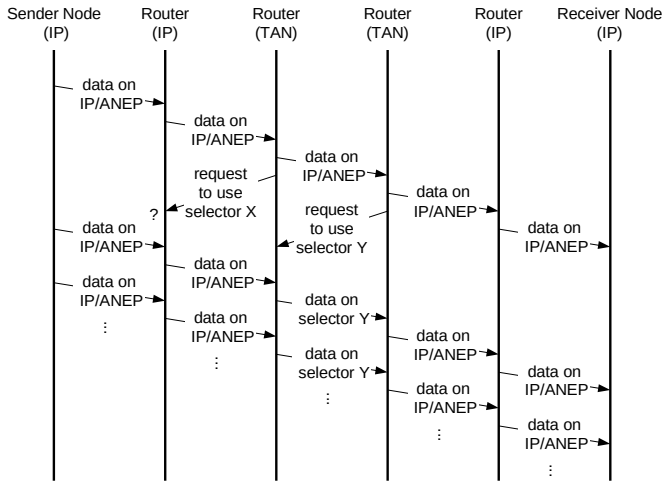


Fig. 3. Selector Setup for IP/TAN network. The initial data is sent with IP/ANEP. The TAN nodes can use SAPF between them. Control messages that go to regular IP routers are ignored by the receiver (?).

control message and discards it. Thus, the router will still transmit all packets with IP/ANEP.

The change from IP/ANEP to SAPF requires that the header of a packet is modified on each border node of a TAN cloud. On the way from IP to TAN, the IP header is discarded and the SAPF selector is used instead. From TAN to IP, an IP/ANEP header has to be put in place of the selector. This can be achieved by caching the IP/ANEP header of the first packet of the flow (assuming that the flow does not change the IP/ANEP header over time). Another, more flexible, solution is not to discard the IP/ANEP header on entering the TAN cloud, but to encapsulate it with the SAPF header. On leaving the TAN cloud, the SAPF header is stripped from the packet, and the original IP/ANEP header can be used. This method can only be used efficiently, though, when the IP/ANEP headers are fixed length for the particular flow.

This example illustrates that the SAPF protocol can easily be used together with traditional IP routers. The decision to use selectors can be made on a per-hop basis. If TAN nodes are traversed, the performance improvements from SAPF can be used. If a node does not understand the SAPF protocol, there is no adverse effect.

III. TAN Node Implementation

Our implementation of a TAN node addresses all three important components of an active network router: high-performance hardware, streamlined software, and the SAPF protocol for efficient demultiplexing. This BSD-Unix based implementation has been used to do the measurements discussed in Section 4. More details on the general ANN architecture can be found in [8]. There is another SAPF implementation available for Linux based systems which is used for the ARRCANE (Active Routing and Resource Control for Ad-hoc NEtworks) active wireless networking research project [18]. Although not aiming at high-speed routing, SAPF is beneficial in this context, too, because it allows to keep a low computational form factor. Every cycle that is not spent for data forwarding can be used for active control tasks.

A. Node Hardware

The hardware of our TAN node consists of an ATM switch fabric (Washington University Gigabit router [19]) that connects eight ports with data rates as high as 2.4 Gb/s. To provide the active processing capabilities, each port is equipped with a Smart Port Card (SPC) line card [10]. An SPC is comprised of a Pentium processor, cache, and main memory. Each SPC has a main data path that passes though the ATM Port Interconnection Controller (APIC) [20]. The APIC supports zero-copy semantics to reduce the overhead introduced by data copying.

While an SPC provides the flexibility required for active networking, its computational power is too limited to support Gb/s data traffic. To increase the processing capability of a port, several SPCs can be added to the port.

At the time when measurements where performed, the SPC had been in an experimental state. Thus, the actual system used for measurements consisted of a Pentium PC with Ethernet and ATM network interfaces runing the SPC operating system.

B. Node Software

The software basis of the active network node is the Active Router Plugins [21] research platform. Router Plugins provide a highly modular router platform that allows adding and removing extended services router functionality at run time. Software for dynamic integration in the kernel, called plugins, can be dynamically retrieved from a server using Distributed Code Caching for Active Networks (DAN) [22].

All performance critical components, especially the complete data path, are implemented in the kernel to avoid the substantial overhead introduced by forwarding data to user space. Only more complex, low bandwidth components, like management and control, are implemented in user space.

In the context of our architecture, we call handlers referenced by SAPF selectors "active plugins". Basically, active plugins can be of two types: class plugins and instance plugins. The terminology is intentionally derived from object-oriented programming, since the semantics are similar. Class plugins contain code which is downloaded and installed on the node, either on-demand on oc-

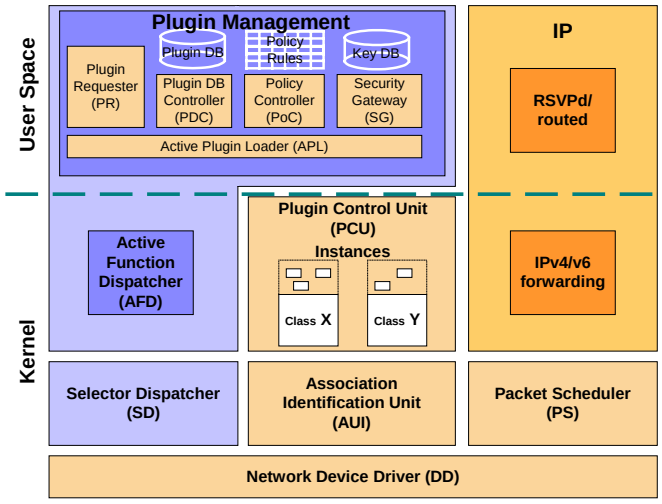


Fig. 4. The ANN node architecture. All data path components are kept in the kernel to improve performance. Plugin management and control functions are implemented in user space.

currence of a reference in a datagram, by a special configuration packet, or by an administrator. Class plugins can therefore be anything from an Execution Environment, as shown in [3], to an active router plugin. Instance plugins are generated (instantiated) by class plugins at run time. All instances are identified by the incoming selector value. They store the outgoing selector value, a pointer to a handler function for the datagram packet, and other configuration information.

In the simplest case, an instance of the IP forwarding handler could store a route and link layer information for a particular flow. This is very similar to an entry in the Tag Information Database described in [16]. On the other extreme, an instance can be a complex ANTS program [7] with state information.

The components of the TAN node are shown in Fig. 4. The following list explains the functionality of the main components:

- *Network Device Drivers (DD)*: the DDs implement standard hardware-specific send and receive functions. IP/ANEP packets are passed to the Association Identification Unit for flow classification before they enter the IP stack. SAPF packets are directly forwarded to the Selector Dispatcher.
- *Association Identification Unit (AIU)*: the AIU performs flow classification for IP packets based on source and destination addresses, incoming interface, transport layer protocol, and port numbers (if available). All state information associated with a flow is maintained by the AIU.
- *Selector Dispatcher (SD)*: the SD maintains the forwarding information base with selectors values that map to flow associations. SAPF packets entering the SD are forwarded to the appropriate plugin instances in the Plugin Control Unit according to their selector value.
- *Plugin Control Unit (PCU)*: the PCU manages the active processing engine. Plugin instances are spawned from their plugin class for each independent flow that requires the plugin. Plugin instances perform the active processing of a packet.
- *IP Stack (IP)*: the IP stack performs the usual IP processing. IP/ANEP packets are forwarded to the Active Function Dispatcher.
- *Active Function Dispatcher (AFD)*: the AFD processes the ANEP header of an IP/ANEP packet. According to the code references in the packet, the corresponding plugin in the PCU is called. If a plugin class is missing, the Plugin Management is notified, which triggers an automatic download of the code from a trusted code server.
- *Plugin Management (PM)*: the PM provides the functionality to download plugin code from a set of trusted code server. Only plugins that are in accordance with the local installation policies are downloaded. Each plugin source and developer are authenticated to ensure security. More details are given in [8].
- *Packet Scheduler (PS)*: all outgoing packets go through the PS to ensure fair sharing of bandwidth. The actual transmission of packets is again performed by the DDs.

The data path for active IP/ANEP packets goes from the Device Drivers to the packet classification in the AIU. The IP header then is processed in the IP stack and the packet is forwarded to the AFD for ANEP processing. The ANEP header is parsed to find the references to the plugins that should execute on the packet. If the corresponding plugin is already present, the packet is sent to the instance that corresponds to the flow of the current packet. After the plugin has processed the packet, it is sent out through the IP stack and the Packet Scheduler to the Device Drivers. If the plugin is not present, the Plugin Management receives a request to download the plugin. Before the plugin is installed, its source and developer are authenticated and local installation policies are checked.

The data path for SAPF packets is simpler. The Device Driver forwards the packet to the Selector Dispatcher that performs the table hash to look up the flow association of the packet. With the flow association the state information becomes available. The state information contains the pointer to the instance that has to be called (the instance is always present, since it was created when the selectors were exchanged). After processing the selector is set to the value for the outgoing link, and the packet is sent out through the Packet Scheduler and the Device Drivers.

IV. Results

The performance gains that can be achieved using selectors are measured by comparing the processing time

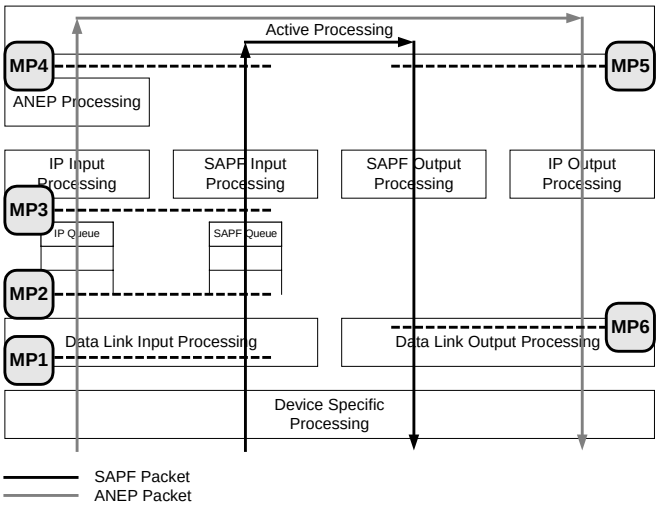


Fig. 5. Data path through kernel for IP/ANEP and SAPF packets. MP1 - MP6 correspond to measurement points for the performance evaluation.

for SAPF packets with regular IP/ANEP packets. Fig. 5 depicts the path that active packets take through the kernel. IP/ANEP packet are shown with a grey line, SAPF packets with a black line. We placed several measurement points (MP1 - MP6) within the kernel to evaluate the processing times of different network components. At each measurement point, the current CPU clock is stored. The results were obtained from a Pentium processor running NetBSD 1.3.2 with our current ANN modifications.

The measurement points are located to measure the following processing components:

- *Packet processing before enqueuing* (measurement points 1 and 2). For IP/ANEP packets, this accounts for demultiplexing the packet according to the LLC/SNAP header, performing a flow classification according to IP source and destination, layer 4 protocol type, and incoming interface. The flow classification is required for later demultiplexing to the appropriate active packet handler. For SAPF packets, demultiplexing according to the LLC/SNAP value is performed. Packets are enqueued in their respective queues. The queues are necessary, since the IP- or SAPF-processing is performed on a different interrupt level than the link layer processing. Since the dequeuing happens asynchronously, the delay introduced by the queue is not included in the measurement.
- *IP/ANEP or SAPF processing* (measurement points 3 and 4). IP/ANEP packets are checked for validity (i.e., header checksum) and parsed to obtain the corresponding active processing handler. For SAPF packets a hash is performed to associate the state record with the packet. The state records contains a pointer to the active processing handler.
- *Output processing* (measurement points 5 and 6). IP/ANEP packets go through the regular IP output processing, where the header checksum is computed and routing is performed. SAPF packets are prepended with a SAPF header containing the correct SAPF value for the next hop link. The packets are then passed to the link layer output function. Since the link layer processing is, again, media specific, it is not included in the measurement.

All network device driver related processing was intentionally left out to keep the results independent from device-specific behavior.

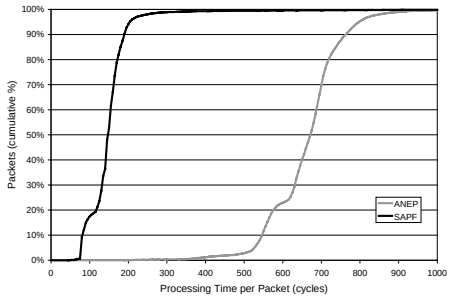
The cumulative percentage of processing time (i.e., the percentage of packets that require that much or less processing time) for the respective kernel components in CPU cycles is shown Fig. 6 (a)-(c). The measure of CPU cycles is mostly independent of the processor clock frequency, but it is highly dependent on the CPU architecture. As a reference, 400 cycles correspond to 1 μs on a 400 MHz processor. Each measurement was performed for 10,000 packets.

There is a significant difference in the data link input processing (Fig. 6 (a)). SAPF packets require only very little processing (around 150 cycles for the 50% percentile), while IP/ANEP packets need about 650 cycles. This is due to the flow classification that has to be performed on IP packets. It can be argued that this classification might be performed at a different point in the processing, but it has to be done at some point before the active processing, in order to find the processing handler. The fact that flow classification is not required for SAPF packets, since the SAPF selector inherently is a flow identifier, results in the considerably reduced processing time.

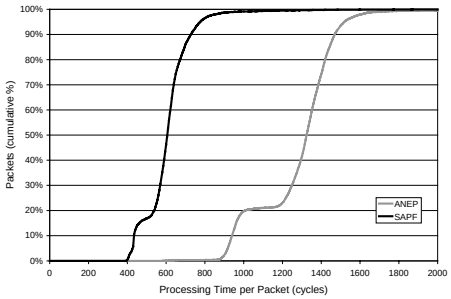
The processing time of 1300 cycles for the IP/ANEP input is about twice the processing time of 600 cycles for SAPF input (Fig. 6 (b)). This comes from the relative complex IP input processing, and the necessity to parse the ANEP header to obtain the identifier for the active processing code. SAPF, on the other hand, only performs a hash lookup in a table that contains the state information for the current flow, and forwards the packet to the appropriate handler for active processing.

The least difference between IP/ANEP and SAPF can be observed in the output processing (Fig. 6 (c)). While IP/ANEP requires about 1350 cycles to adjust the IP header of the packet, SAPF needs 1000 cycles to look up the SAPF value for the outgoing link and adjust the SAPF header. The routing lookup for IP/ANEP is relatively insignificant when the route is cached (as it is in this example). SAPF packets have the route stored in the state information table, which can also be accessed readily.

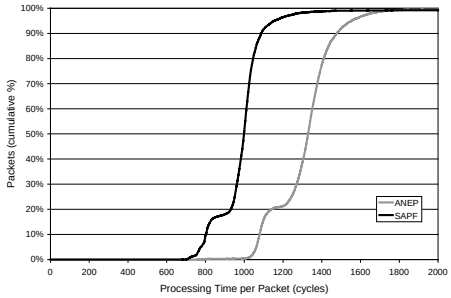
To total processing time from data link layer input to data link layer output, including queuing delays and active processing, is shown in Fig. 6 (d). It allows



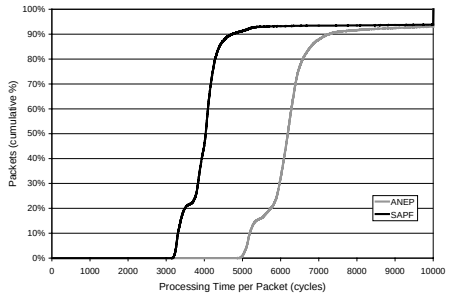
(a) Processing time for MP1 - MP2



(b) Processing time for MP3 - MP4



(c) Processing time for MP5 - MP6



(d) Processing time for MP1 - MP6

Fig. 6. Processing time measurements for IP/ANEP and SAPF packets. Fig. (a)-(c) show delays for different stages of the packet processing. Fig. (d) shows the total processing time from data link layer input to data link layer output.

to compare the overall performance of both approaches. The total time for IP/ANEP is about 6000 cycles, while SAPF requires only about 4000 cycles. These numbers include a queuing delay of about 2500 cycles, which is not actual processing time, since the system is performing other tasks during that time. Also included is an active processing time of roughly 150 cycles. This comparison shows that SAPF requires 30% less processing time over IP/ANEP.

V. Related Work

The Active Network Encapsulation Protocol (ANEP) [12] is the de-facto standard for active packets over IP. It encapsulates the active payload with a header that contains a version number, a flag field, an identifier that indicates the active processing engine, a header length, and a packet length field. ANEP also allows for multiple variable-length header options that can include security headers as well as a header checksum. The parsing of multiple header options, though, imposes considerable overhead when the packet is demultiplexed. Therefore, we believe that a simpler format, like SAPF, supports demultiplexing more efficiently. Nevertheless, the flexibility of ANEP is very useful for control purposes (e.g., establishing flows).

The SAPF protocol is conceptually similar to Tag Switching [23] and Multiprotocol Label Switching (MPLS) [24]. Tag Switching is a predecessor to MPLS, both schemes use labels on data packets to simplify routing. The label is used as a mean to aggregate forwarding information, which enables the implementation of special services for certain flows. The difference to TAN is that MPLS is only used on the network layer. The SAPF protocol is used to demultiplex up to the EEs and code handler in one step. This makes a significant difference in the context of active networks, since all active packets have to be processed in the active processing engine.

Single-step demultiplexing has also been addressed in the context quality-of-service research [25]. Migrating sockets use an 'OS handle' similar to a SAPF selector.

VI. Conclusions and Future Work

We described SAPF, a tag-based protocol for active networks, that allows very efficient demultiplexing of packets to their handler code or EE. We showed how tags are exchanged between active TAN nodes and how regular IP routers and TAN interoperate. The TAN node architecture focuses on three important aspects to achieve high-performance active processing: a powerful hardware, streamlined software that performs packet processing completely in the OS kernel, and the SAPF protocol for fast demultiplexing. Measurements on our active network node show that SAPF packets can be processed 30% faster than traditional IP/ANEP packets.

Our future research on SAPF is focusing on specifying the Selector Exchange Protocol (SXP) which is used to communicate selector values between nodes. Currently, SXP is used in our research environment for receiver-assigned selectors. Before SXP can be presented to the AN community, it has to be tested in a wider variety of scenarios and systems.

Another goal of our research is to embed data layer specific information into the selector values. Using ATM as the underlying hardware allows to map the VPI field (8 bit) and the VCI field (16 bit) into a 24 bit area of the selector. The remaining bits can be used to identify the code handler. This could reduce the number of lookups required for demultiplexing and further decrease the packet processing time.

References

[1] D. Tennenhouse, J. Smith, D. Sincoskie, D. Wetherall, and G. Minden, "A survey of active network research," *IEEE Communications*, 35, 1, January 1997, 80-86.

[2] L. Peterson (ed.), "NodeOS interface specification," AN Node OS Working Group, *http://www.cs.princeton.edu/nsg/papers/nodeos99.ps*

[3] Active Network Working Group, "Architectural framework for active networks," *http://www.cc.gatech.edu/projects/canes/arch/arch-0-9.ps*

[4] S. Alexander, W. Arbaugh, M. Hicks, P. Kakkar, A. Keromytis, J. Moore, C. Gunter, S. Nettles, and J. Smith, "The SwitchWare active network architecture," *IEEE Network Special Issue on Active and Programmable Networks*, 12, 3, July 1998, 29-36.

[5] J. Hartman, L. Peterson, A. Bavier, P. Bigot, P. Bridges, B. Montz, R. Piltz, T. Proebsting, and O. Spatscheck, "Joust: a platform for communications-oriented liquid software," *IEEE Computer*, 32, 4, April 1999, 50-56.

[6] B. Schwartz, A. Jackson, T. Strayer, W. Zhou, D. Rockwell, and C. Partridge, "Smart packets for active networks," *Proc. of OPENARCH 99*, IEEE, New York, NY.

[7] D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: a toolkit for building and dynamically deploying network protocols," *Proc. of OPENARCH 98*, IEEE, San Francisco.

[8] D. Decasper, G. Parulkar, S. Choi, J. DeHart, T. Wolf, and B. Plattner, "A scalable, high performance active network node," *IEEE Network*, 31, 1, January 1999, 8-19.

[9] I. Hadzic, W. Marcus, and J. Smith, "On-the-fly programmable hardware for networks," *Proc. of GLOBECOM 98*, IEEE, Sydney, Australia.

[10] J. DeHart, W. Richard, E. Spitznagel, and D. Taylor, "The smart port card: an embedded Unix processor architecture for network management and active networking," unpublished.

[11] T. Wolf and J. Turner, "Design issues for high-performance active routers," *Proc. of the 2000 International Zurich Seminar*, in press.

[12] S. Alexander, B. Braden, C. Gunter, A. Jackson, A. Keromytis, G. Minden, and G. Wetherall, "Active network encapsulation protocol (ANEP)," RFC Draft, *http://www.cis.upenn.edu/~switchware/ANEP/docs/ANEP.txt*

[13] C. Tschudin and D. Decasper, "Simple active packet format (SAPF)," Experimental RFC, *http://abone.ifi.unizh.ch/~sapf*

[14] R. Keller, S. Choi, D. Decasper, M. Dasen, G. Frankhauser, and B. Plattner, "An active router architecture for multicast video distribution," *Proc. of INFOCOM 2000*, in press.

[15] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast scalable algorithms for level four switching," *Proc. of SIGCOMM 98*, ACM, Vancouver, BC.

[16] Y. Rekhter, B. Davie, D. Katz, G. Swallow, and D. Farinacci, "Tag switching architecture - overview," Internet Draft, *draft-rekhter-tagswitch-arch-00.txt*

[17] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "A new resource reservation protocol," *IEEE Network*, 7, 5, September 1993.

[18] Uppsala University and Ericsson Switchlab, ARRCANE (Active Routing and Resource Control for Active NEtworks), *http://www.docs.uu.se/arrcane/*

[19] T. Chaney, A. Fingerhut, M. Flucke, and J. Turner, "Design of a gigabit ATM switch," *Proc. of INFOCOM 97*, IEEE, Kobe, Japan.

[20] Z. Dittia, J. Cox, and G. Parulkar, "Design of the APIC: a high performance ATM host-network interface chip," *Proc. of INFOCOM 95*, IEEE, Boston, MA.

[21] D. Decasper, "A software architecture for next generation routers," *Dissertation no. 13123*, ETH Zurich, April 1999.

[22] D. Decasper and B. Plattner, "DAN: distributed code caching for active networks," *Proc. of INFOCOM 98*, IEEE, San Francisco, CA.

[23] Cisco Systems, Inc., "Tag switching: uniting routing and switching for scalable, high-performance services," White Paper, *http://www.cisco.com/warp/public/cc/cisco/mkt/ios/tag/tech/tagsw_wp.pdf*

[24] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," Internet Draft, *draft-ietf-mpls-arch-06.txt*

[25] D. Yau and S. Lam, "Migrating sockets - end system support for networking with quality of service guarantees," *IEEE/ACM Transactions on Networking*, 6, 6, December 1998, 700-716.