Active Protocol Label Switching (APLS)

William Lau and Sanjay Jha
Network Research Laboratory
School of Computer Science and Engineering,
The University of NSW, Kensington Sydney 2052 Australia
wlau,sjha@cse.unsw.edu.au

UNSW-CSE-TR-0207

July 2002

THE UNIVERSITY OF
NEW SOUTH WALES

SYDNEY • AUSTRALIA

Abstract

Modern layer 3 networking technologies have mainly been designed for performance and for network providers. This report proposes a new network architecture called Active Protocol Label Switching (APLS) that combines the performance of current label switching technology with novel concepts that cultivate service provisioning. Novel features such as Virtual Label Space, APLS micro-instruction architecture, and micro-policy based forwarding provide a more powerful network model, facilitate better network level service engineering, and give tremendous flexibility to both network and service providers. The thrust of our study is to construct an APLS test-bed using open hardware and software and later use this test-bed for experimenting various features/options available with APLS.This report also describes our prototype implementation of APLS under Linux.

1 Introduction

Information, the origin of knowledge and the heart of decision making. The world is now in the age where having the right information at the right time means money, power, and even to the point of saving life. The past decade has seen a rapid change in the way information flows and one of the most influential seen today is Computer Networks.

The Information Technology (IT) boom over the past decade has changed the way Internet is structured and used. The Internet now connects millions of computers together comprising a very diverse range of networking technologies, computer technologies, and wide geographical coverage. The advances in technologies have seen great improvement in bandwidth with Cable, ADSL, Satellite, and Optical solutions. Wireless (WLAN) and mobile technologies (GPRS, 3G) have also gathered paste, and it will not be long before ubiquitous access to the Internet becomes a reality. The consequence of these technological advancement is the change in applications of the Internet. Recent new applications include Virtual Private Networks (VPNs), file sharing, instant messaging, and video broadcast etc. The future will see integration of more variety of sophisticated services into the Internet, therefore, the next generation of Internet applications will be service-oriented.

This project puts faith in giving the network a larger role to play in service provisioning and focuses on changing the IP model to become more service-oriented. This requires introduction to a new label switching technology called Active Protocol Label Switching (APLS). APLS is designed on top of IP for large scale service provisioning and delivers a model where the network providers delegate label space and traffic engineering controls to service providers. The result is a more flexible label switching architecture that highly promotes service provisioning inside the network itself thereby creating a powerful basis for value-added facilities.

The road map for this paper is as follows: First, the basis for a service-oriented network will be analyzed in Section 2, followed by the design of APLS in Section 3. The implementation of a prototype on Linux is explored in Section 4 and related works are discussed in Section 5. Finally, the conclusion is given in Section 6.

2 Service-Oriented Network

The modern Internet comprises of hundreds of thousands of services. These services range from community services to commercial services and may include a hybrid of free and fee based. Examples of services are: stock trading, online banking, online shopping, online gaming, audio/video streaming, file sharing, IP telephony, and more recently Video On Demand (VoD). What

has made Internet such a successful and attractive medium for services? Lets analyse what the Internet means to the businesses and to consumers.

2.1 Businesses and Consumers

The virtual world that Internet creates provides many opportunities to existing and new businesses. These opportunities sprung from:

- **Business reach**: Location on a virtual world is not important anymore, the only restriction is location of the real world where red tape (legislation, tax etc.) exists. It is easy for business to reach customers on all corners of the Internet provided they successfully advertise to targeted consumer base (knowledge of their existence). Internet has made Internationalisation easier.

- **Relatively low operational costs**: Internet provides many opportunities for businesses to lower their operational costs. One example would be self-service business websites where services can be automated and utilised 24 by 7. Business can also lower resources cost as evidenced in financial sites that provide electronic copies of prospectus and forms. The savings here is the reduced print, mail, and staff costs. For Internet based businesses, lower operational costs are even more apparent. These businesses do not need to have branches in different geographical locations to reach consumers. The result is lower office lease cost, less redundancy and better utilisation of employees, and lower inventory cost because stock does not need to be piled up in different locations.

- **Location Flexibility**: The position the business resides in the virtual world is tied to the Internet domain name rather than physical location or host. Therefore, it would be relatively simple to move physical location without affecting consumers' reachability of the business on the Internet[1]The change in IP address mapping to Domain Name space is relatively fast and a temporal redirection server for the old IP address can be arranged to smooth out the transition

- **B2B communication**: The reachability of Internet also means potential for business to business dealings on the Internet. Suppliers can automate up-to-date electronic price quotes while buyers can also place orders electronically. One example would be the mobile business where mobile dealers can activate new phone numbers through the Internet.

- **Low cost and efficient information distribution**: The Virtual Private Network (VPN) [1] concept provides opportunities for business

[1] .

to improve information distribution within the organisation and also potentially lowering the cost for such a information system. VPN allows secure communication between business sites and also for secure access to business sites through employee's home. Mobile employees (e.g. sales staff) can access up-to-date and secure information on the move by using the wide accessibility of the Internet.

From the consumers' perspective, the Internet provides certain values:

- **Easy access**: It is now easy to get access to the Internet. Access can be from the home computer, office computer, or the popular trend of Internet Cafe.
- **Easy reach**: Location on a virtual world is not important anymore, the only restriction is location of the real world where red tape (legislation, tax etc.) exists. Therefore, it is easy for business to reach customers and easy for customers to reach the business in this virtual world.
- **Service speed**: Electronic transactions and queues are relatively faster than personal transactions in the real world.
- **Convenience** is the key to a lot of successful services on the Internet. Easy access and easy reach unites to give a convenience not achievable by other mediums. Consumers can now access services from at a location that suit them and the only restriction is that the location must have access to the Internet. With next generation of mobile and wireless technology already being deployed, this restriction will become more and more insignificant.

The potential for Internet services is restricted by the technology that makes up the Internet. Conventional philosophy is to fit the services around the Internet but the philosophy in this project is to analyse the opposite, that is, how the Internet should evolve into a service provisioning internetwork.

First, a new term called *Service Engineering* is introduced to define the area of formalising the way in which services are developed and implemented over the Internet. Service Engineering consists of two components:

- **Application Level**: This area focuses on improving services support at the application level. This includes OS support, third party software support, and the service application itself.
- **Network Level**: This area focuses on improving services within the network. This can range from better QoS support to Active Programmable Network [2] (APN) support.

Current Service Engineering on the Internet only have Application Level Support since the current network model offers only best-effort forwarding. This paper focuses on Network Level support only.

In the following sections, the characteristics of service applications will be investigated and then various new service provisioning network models will be presented and analysed. The results will give the basis for forming an improved network model for service engineering at the network level.

2.2 Service Application Characteristics

Investigating the characteristics of service applications on the Internet will show a broader picture of what are the requirements for a network to become service-oriented.

1. Simple versus Complex transaction: Simple transaction consists of one request/response transaction and only spans for a very short duration (a few milliseconds). In other words, it is a stateless transaction. A simple example is the credit card payment.

   Complex transaction consists of multiple requests that are dependent on each other. The transaction may involve more than two parties e.g. third party certificated transactions. Other examples are secure financial transactions, video streaming, and interactive services.

2. Session-oriented versus non-session-oriented: Session-oriented service requires a session to be established between the client and the service Provider. Examples of such services are ones that require client log in, like Internet Related Chat (IRC) or online magazine subscription. Each session consists of multiple transactions. Transactions are operations that make up a service. Complex transaction based services are normally but not necessarily session-oriented. Non-session oriented services normally consist of a one-off simple transaction. Examples are credit card payment or form submission.

3. Low bandwidth versus High bandwidth transaction: Depending on the complexity of the service, transactions may require relatively low bandwidth or high bandwidth. Note that the term low/high can be in relation to percentage against network capacity.

4. Real-Time sensitive versus Real-Time insensitive: Real Time (RT) services are sensitive to delays or jitters in the network. Bad response time of the server in RT sensitive services can make data obsolete or may affect the outcome of the transaction. Examples include video streaming, IP telephony, and online trading. RT insensitive services do not have stringent requirements for response and data transmission time. However, clients will have certain expectation of the service's response time e.g., online banking.

5. Static versus Dynamic Content: Static contents refer to information that remain the same for a relatively long period of time. Some examples of static contents are magazine and video on demand. Dynamic contents are information that changes consistently over a short period of time (seconds to minutes). Examples include video conferencing, live broadcast, online trading.

6. Unicast versus Multicast: Almost all of the services available on the Internet are unicast services. Unicast means transactions that involve one client and one Server at a time. In Network overlay model, it is possible for the main server to redirect the request to another server but this is still considered as unicast since the rest of the transaction is still one to one. Multicast refers to transactions that involve many receivers and one sender. The clients are normally the receivers who join a multicast group. The multicast group gets served by one server who sends data. In more complex multicast protocols, multicast groups allow more than one sender hence multiple servers. Examples are online gaming and live broadcasting.

2.3 Service Provisioning Models

Service Provisioning Models show the different ways that services can be delivered on the Internet and what role the stakeholders have in the process. The stakeholders are listed below followed by introduction to the four existing types of service provisioning model: Client/Server, QoS, Network Overlay, and Active Networks.

2.3.1 Service Provisioning Stakeholders

There are three main parties in service provisioning: the Client, the Service Provider, and the Network Provider. The definitions of these parties will be given here and used thereafter in the rest of this report. The Stakeholders' relationship and expectation of each other are also put into perspective.

- **The Client**: The Client is the consumer that pays for using a particular service over the Internet. Client will expect services to come with low setup delay and minimum setup complexity. The service provided should meet the minimum level of expectation of the consumers in terms of performance, accessibility, and service support.

- **The Service Provider**: The Service Provider provides a particular service to the client. Service Providers do not provide raw networking services to the Client even though it is possible for service provider to be involved in reselling of some networking services.

Service Provider would like to have the ability to deploy their services faster for new clients and to be able to scale the services to a growing client base. The deployment, scaling, and operational costs are a big factor to realise a feasible service. The rapid change in consumer trend, taste and fashion will require Service Providers to be robust and dynamic. They must be able to respond to the client's demand or even create new markets.

- **Network Provider**: The Network Provider is considered as the owner of the network for which the clients or service providers use to access the Internet. It also applies to the owner of the network that the service's data transit through. Network Providers have two types of client. The first type is the end-host consumer that requires access to connectivity to use the Internet services. The second type is the Service Provider that requires networking resources to service its clients in the network or to transit through the network.

  Network Providers need to have a network technology that can meet the expectations of the Client and Service Providers. Network provisioning is a commodity business where revenue from raw networking resources such as bandwidth is lower with increasing competition as evidenced in the current industry. Therefore, Network Providers need to use technologies that offer not just increases in capacity, but also reductions in the operational costs and generation of new revenue sources by supporting new services.

2.3.2 Client/Server Model

This model is the most widely used on the Internet. The model basically contains a client end-host that requests for a service from a server end-host. The Internet is just a medium to transports the request and response data. This service model is illustrated in Figure1. In this model, the Internet does not offer any special functionality that the service can utilise. The Internet only provides best effort forwarding (no differentiation of traffic) and requires servers to adapt to the Internet environment between itself and the client. Therefore the Internet is considered orthogonal to the service.

This model is primarily designed for services that are simple in transaction (request-response) or are RT insensitive.

2.3.3 QoS Model

A lot of research has gone into extending the IP model with Quality of Service (QoS). This QoS model requires the network providers to provide some form of QoS mechanisms that the service providers can incorporate into their services. This introduces dependency on the network and thus Internet

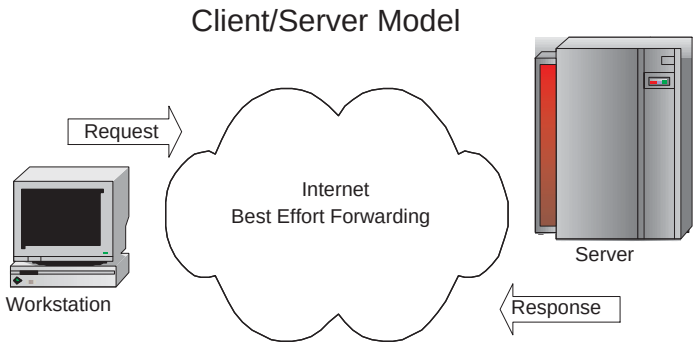

Figure 1: Client/Server Model

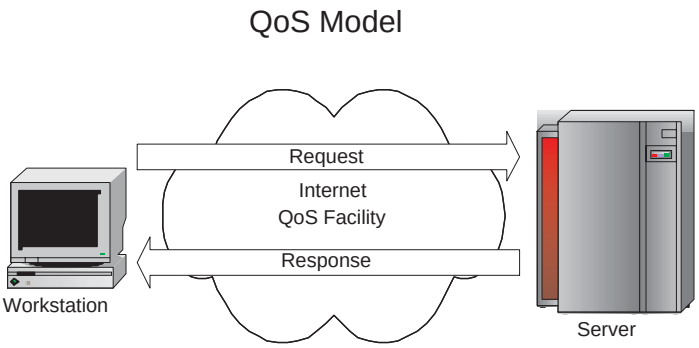

Figure 2: QoS Model

now becomes part of the overall service. This service model as illustrated in Figure 2. In this model, support for services is generalised in classes or Type of Service (ToS). Each class/ToS will have a distinct behaviour and support within the network. Service providers have some control over the behaviour through specification of parameters.

This model can enhance services that use the client/server model and also gives support for real-time sensitive traffic. QoS mechanisms are mostly session-oriented and thus are more suitable for session-oriented services. The most promising QoS approaches have been the flow-based Integrated Service Architecture (IntServ)[3] and the flow-aggregation-based Differentiated Service Architecture (DiffServ)[4, 5].

2.3.4 Network Overlay Model

This model contains three components: the client who requests for the service, the Main Server that the client initially sends the request to, and the

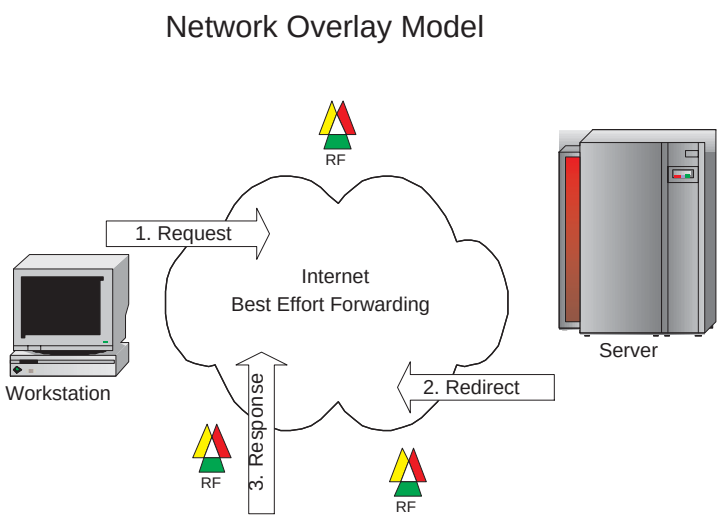

Figure 3: Network Overlay Model

Resource Facilities (RF) that are strategically allocated at different edge points of the Internet. Normally, the RF will contain servers dedicated to respond to the client. The RFs are usually maintained and deployed by the service provider. In this model, the Internet is not part of the service since it acts as the best effort transport medium.

The intention is to improve QoS provided to the client. This network overlay can provide some form of load balancing and content caching. If RF is allocated physically closer to the client then response time can be improved and bandwidth overhead in the network can be reduced. Services with high bandwidth, static content, or multicasting will benefit from this model. Network Overlay is also a good model to support multiple-party complex transactions.

Figure 3 shows an example of a video streaming service using this model. The client requests to the main service center (main server) which then redirects the request to a server that will best service the client. The chosen RF's server starts the video stream to the client.

The QoS model can be combined with the Network Overlay model to realise a more powerful service provisioning model. A more sophisticated technology that uses the network overlay model is Grid Computing [6, 7].

2.3.5 Active Programmable Networks Model

APN [2] allows service provider to utilise special computation inside the network. APN also aims to give restricted forwarding and routing control to the service provider. The result is the potential to enhance all types of

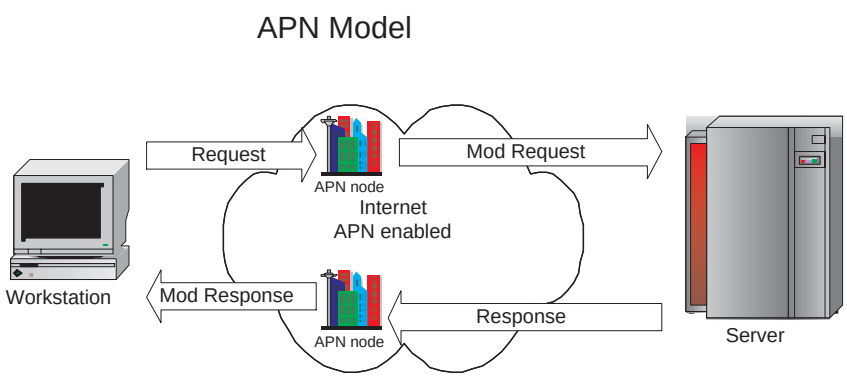

Figure 4: APN Model

service as well as better support for deployment of new services.

Figure 4 illustrates a service provider using value-added processing in strategic locations in the Internet. An example of value-added processing could be modifying the request for load balancing and content caching.

Safaei et al. [8] proposes an APN model where the network providers provide resources inside the network. These resources can in turn form a more efficient network overlay structure than the Network Overlay model. In this approach, the network provider owns and maintains the network overlay resources. This means that service provider will not have to bear the cost to maintain or deploy the resources thus making the model more affordable to a larger range of services. Multiple service providers can use the same set of resource thus increasing utilisation while receiving benefits from economy of scales. Since the network providers deploy the resources for general service use, they will deploy at more locations than if deployed by the service provider. This means that the service providers can take advantage of the increased flexibility (use what they need only) and coverage (use where it is needed).

2.3.6 The Ideal Model

The ideal model for service provisioning must focus on criteria such as: flexibility, control, scalability, and performance.

1. Flexibility allows the network to rapidly adopt to the latest trends of services that comprise of different combination of characteristics. The future beholds the unexpected.

2. Control is in respect to controlling network resources. There are two aspects of control. The first aspect is in perspective of the network provider where control is needed for traffic engineering and providing

differential service. The second aspect is in perspective of the service provider. The ideal model should provide some level of delegated control of the network to the service provider to increase their responsiveness to their clients. By giving control to the people with the right knowledge about the traffic, the network has the potential to be better utilised.

3. Scalability and performance goes hand in hand. While providing mechanisms for flexibility and control, the service model should not sacrifice performance. The service model should also perform in all network sizes including interoperation in the Internet. As the number of supported services in the network increases, the level of performance should not deteriorate significantly. Performance deterioration should be a by-product of supply (network capacity) versus demand (number of users) function.

Future networks should also provide value in the users' context rather than just in the traditional context of the network provider.

The simple client/service model offers performance and scalability but it lacks flexibility and control. The QoS model offers little flexibility (flexible in QoS parameters only) but provides excellent resource control to provide differential services. However, it does not offer (in fact not designed for) traffic engineering control nor does it delegates control to service providers. Scalability of QoS architectures is yet to be proven.

The network overlay model is more of a supplementary model that can enhance scalability and performance to certain applications. The APN model offers greatest flexibility and has good potential for control but the concept still needs to be proven for scalability and performance.

The complete solution for an ideal model is too large to be proposed, designed, or implemented by a single researcher or small group of institutions. What has made the Internet today is the specialisation of researchers and developers from different fields combining their efforts to form the big picture. Thus the approach taken in this project is to design a flexible core service provisioning network architecture that can be extended to incorporate new technological improvements. This core architecture forms the basis for scalable performance while promoting mechanisms to extend the network model for flexibility and control.

3 Design of APLS

The objective of this project is to propose a network architecture that resembles the ideal model for service provisioning. This will involve not just the conventional issues of performance and scalability, but also flexibility and control. Label switching technologies have shown great potential in terms

of scalable performance as well as ability to support QoS mechanisms [5, 9]. Numerous research on traffic engineering over MPLS [10, 11, 12, 13, 14] show promising use of label switching for controlling traffic. Therefore, label switching was chosen to form the base for the proposed network architecture called Active Protocol Label Switching (APLS).

ATM was designed with integration of both traffic and QoS in mind thus resulted in a complex and inflexible architecture. In contrast, MPLS promoted a simpler approach that gained flexibility by separating QoS mechanisms from the base forwarding core. Instead of designing a completely new label switching architecture from scratch, the general label switching concepts from MPLS are retained while several new concepts are introduced into the architecture which together creates a service-oriented network. The novel concepts proposed for the architecture are: APLS over IP, Virtual Label Space, Micro-Instructions Architecture, and Micro-Policy Based Forwarding.

3.1 APLS over IP

Label switching has traditionally been seen as a separate technology from IP but one that can support the IP protocol. MPLS is designed as a shim layer between layer 2 (L2) and layer 3 (L3). The motivations of this approach are:

1. Support for converging multiple network protocol traffic onto the same network. This is one of the major selling point for MPLS and for ATM in its early days. Converging traffic into one network will eliminate the need to maintain several networks in parallel thereby reducing operational and infrastructure costs [15, 16] for network providers.

2. Frame relay and ATM operates at the same layer, therefore, migrating from these legacy networks are relatively easier technically. Many works [16, 17, 15] are already done on achieving migration and interoperation.

During the design of APLS, this fundamental attribute of MPLS was challenged and an alternative position was proposed. The proposed position for the shim layer is between the network layer and the transport layer as illustrated in figure 5. The direct consequence of this radical change is that APLS is now dependent on the network layer. Although APLS can be generalised to work over different network layers, this project will focus specifically on APLS over IP. The reasons are that IP is the protocol expected to dominate networking traffic in near future and that IP is a subnet independent protocol layer which is already running on many high performance subnets.

Fig. 6 shows an abstract APLS packet format.

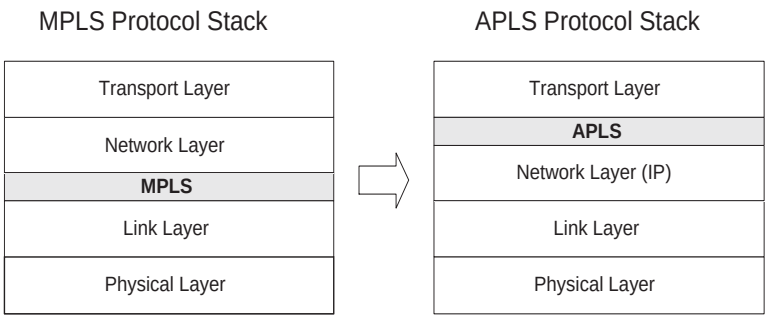

Figure 5: MPLS and APLS Protocol Stack

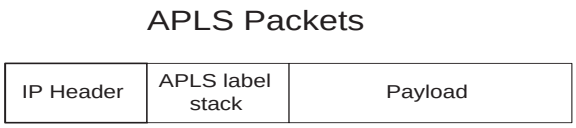

Figure 6: Abstract APLS packet format

First, the advantages for using this alternative approach in APLS:

1. Being on top of IP, APLS can exploit the IP header fields that are usually redundant for IP over MPLS (or IP over ATM). Two fields that APLS utilises are Time-To-Live (TTL) and Differentiated Services Code Point (DSCP). The result is a reduction of label size or an increase in the label space. It is also possible to use the checksum field in IP if per hop checksum in APLS is desired.

2. APLS network supports normal IP forwarding in parallel with label switching. This is made possible because APLS labels are placed after the IP header. The Label Switch Routers (LSRs) will determine from the IP header (via a flag in the IP header) whether a packet requires APLS forwarding or the default IP forwarding. Packets requiring label switching will bypass normal IP forwarding components and are handled by APLS.

3. Network providers can incrementally deploy APLS within an IP network. Label switching architectures that work below the network layer, like MPLS, requires setting up an entirely new separate network in parallel with existing IP network. This can lead to substantial cost for upgrade and for maintaining two networks. The results are higher cost and complexity. APLS on the other hand, allows the LSR to be diffused into the network incrementally. Following the traditional upgrade process (Ring Effect in figure 7) of network providers, new

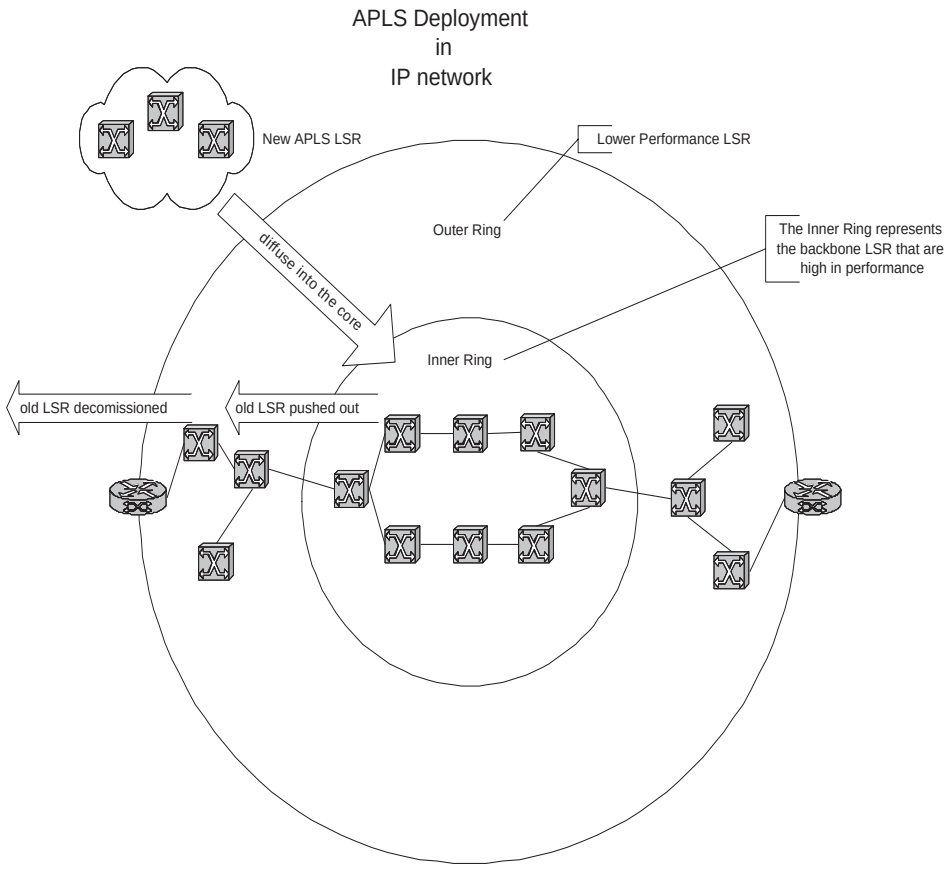

Figure 7: Traditional/APLS Deployment Process

high speed APLS LSRs will be deployed in the core and the replaced IP routers will be pushed towards the edge. At this point, the APLS LSRs will only support high speed normal IP forwarding. Once enough LSRs are deployed, network providers can start setting up APLS LSPs to utilise the performance advantage of label switching.

4. APLS can support simple and safe migration from IP to label switching. Incremental deployment and coexistence with IP means that network providers can experiment with APLS before moving commercial traffic to label switching. The process involved in migrating traffic from IP forwarding to APLS forwarding is simple. Once the ingress LSR are setup to map a FEC to a LSP, all packets in that class will start being label switched. If problems occur, the FEC can be removed and the traffic for that FEC will revert back to normal IP forwarding. This simple recovery process makes migration relatively safer. Simple and safe migration give network providers motivation to speed up adoption of APLS.

5. If IP network and APLS network are left to coexist then network providers can take advantage of the robust IP network for backup during fault scenarios. This allows traffic to continue to flow while affected and problematic LSPs are re-established.

The major disadvantage of APLS is that it supports only IP protocol. However, this downside may not be too significant for the following reasons:

1. IP currently have already dominated network traffic and the growth of the Internet will see IP completely dominate the traffic in just a few years time [18]. IDC research [19] indicates that Internet traffic will surpass voice traffic in 2002 and will be eight times more than voice traffic by 2005.

2. Different network protocol traffic can be transported over APLS. The cost of this method is the addition of a dummy IP header. Considering the percentage of the traffic that requires this additional overhead and the gradual phasing out of legacy networks, the network cost is minimal and temporary.

3. APLS can also interoperate with popular legacy networks like Frame Relay and ATM easily due to the label similarity between these technologies.

4. Amdahl's Law:

   Optimize the Common Case.

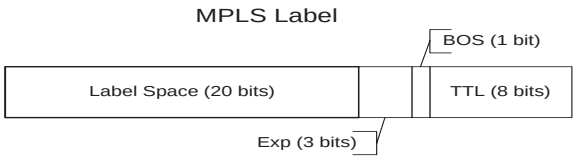

Figure 8: MPLS Label

In the short-medium term, MPLS's approach will take advantage over APLS for supporting the still significant non-IP networks. However, in the medium-long term, APLS's approach takes advantage of MPLS.

APLS will need to be transparent to the user therefore the method that distinguishes an IP packet from APLS packet must not require end-host involvement. The best solution is to use a dedicated bit flag in the main IP header and a good candidate is a unused bit in the TOS/DSCP field. An alternative is to use the IP options, however, this becomes problematic when firewalls and encryption mechanisms (IPSec) are added to the equation.

When LSR receives a packet, it checks for version of IP and then checks whether it is an APLS packet. At this point, the packet should be either passed on to the APLS layer for label switching, or is forwarded by normal IP.

3.2 Virtual Label Space

The MPLS label [20] is shown in Fig.8. MPLS uses a flat label space, that is, the label space is just an identifier with no embedded structure.

ATM has a hierarchical label space consisting of two parts: VPI and VCI. The VPI component can be used for aggregate forwarding of a group of VCIs. The *label* by definition is the label switching architecture's protocol header. Like any other protocol, it may contain various fields and one of the most important one is the *label space*. Label space refers the field that is used as the *key*[2] for looking up forwarding information inside the LSRs. It is also the field that gets swapped with another identifier that is the key to the next LSR.

The current label switching network providers all inherit high operational costs to setup and maintain label paths (circuits). Service providers that leases label paths not only will get passed part of these costs, they will also incur further cost for interacting with the network provider. This interaction introduces longer delays to realise the requests.

Due to business confidentiality or security reasons, service providers may not be able to give network providers sensitive traffic information needed to

[2]QoS fields may also be used

customise the network to better meet their demands. The black box nature of modern network limits service providers' knowledge on the current conditions of the network. As a result, no one has the complete knowledge required to better utilise the network or reduce cost. As mentioned in section 2.3.1, Service Providers need the responsiveness from network providers and the lower operational costs to make the service feasible.

A novel concept proposed here is called **Virtual Label Space** (VLS). VLS aims to solve this problem by making the network more open to the service provider. This will not only reduces cost for both parties but also results in better utilisation of the network. VLS introduces a hierarchical label space consisting of 3 components:

- **Service Identifier(SID)**: For each service provider registered with the APLS network provider, a unique service identifier is allocated to the service provider. The owner of the service identifier has control of the Effective Virtual Space (EVS) consisting of the rest of the VLS. The SIDs are global within the APLS domain and will not be switched between LSRs. SID is the means for identifying the service provider and the associated privileges. SID 0 is allocated for the network provider's general traffic.
- **Aggregate Identifier(AID)**: This aggregate identifier works in a similar way to that of the VPI in ATM for the purpose of aggregate forwarding.
- **Flow Identifier(FID)**: This identifies a unique data flow within the aggregate flow.

From perspective of the service provider, the whole label space of the network consists of the AID and the FID while the SID is just a number for identification. The service provider is exposed to a virtual label space while the network provider may control the real label space that consists of SID, AID, and FID.

Figure 9 illustrates the VLS components and the generic lookup method for VLS. SID is used as an index into the table of EVSs and the entry points to the table of aggregates. The AID is use to index the aggregate table to get the AID entry. If this AID is activated for aggregate forwarding, the entry will contain the forwarding information. Otherwise, the entry will point to the table of flows. The FID then indexes into that table and the entry contains the forwarding information.

The purpose for the SID is to allow delegated label switching control to the privileged service provider. Consequently, this potentially opens up every LSR for service provider to control (unless control is disabled by the network provider). Following are the advantages of this approach:

- Service Providers can do its only traffic engineering, that is, manipulating LSPs.

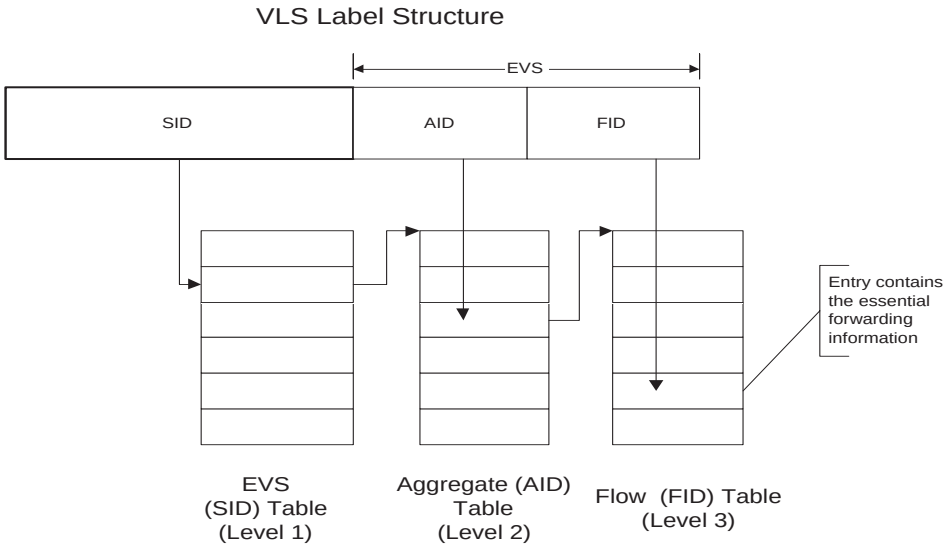

Figure 9: VLS Label and lookup process

- The granularity of control is per VLS label in each LSR, therefore, offers unprecedented flexibility. Examples of flexibility such as QoS and APN support are described in section 3.3.2.

However, the flexibility has some disadvantages as discussed below:

- Security issues in respect to delegating control. This involves general security issues such as authentication and access control, and new issues that come into play e.g. limiting damage that non-conformance use of VLS can do to the network.
- The need for complicated resource management. There is limited resource in each LSR hence some way of allocating resource are needed.
- SID increases label overhead.

VLS offers new flexibility into label switching and the only restrictions come down to security and resources. Both issues are beyond the scope of this report.

3.2.1 Aggregate Forwarding

APLS uses a field inside the label space called AID for aggregate forwarding. This concept is basically the same as VPI in ATM. The idea is to logically group flows together (flows with same AID and SID) and switching them as an aggregate flow. LSR that does aggregate switching only manipulates the SID and AID fields thereby reducing the working set of labels and also reduces the lookup time. Fig. 10 illustrates how aggregate forwarding works.

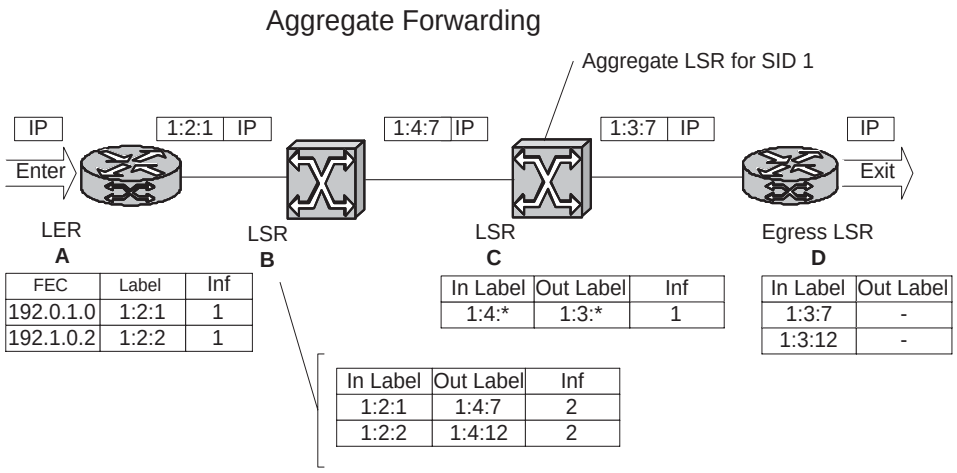

Figure 10: APLS Aggregate Forwarding using AID

In this example, an IP packet heading to network 192.0.1.0 enters APLS at Label Edge Router A (LER A). LER A maps the packet to the FEC that specifies the LSP to use. The label in APLS is represented by 3 decimal numbers separated by semicolons. The 3 numbers represent SID, AID, and FID respectively. LER A adds the label 1:2:1 to the packet and forwards it to LSR B which then performs lookup and swaps the label with 1:4:7 and forwards the packet off towards LSR C. Consider LSR C as a hot spot for traffic belonging to SID 1. Therefore, the service provider of SID 1 configures LSR C to do aggregate forwarding for SID 1. LSR C will only use the SID and AID part of the label for lookup and only the AID field will be swapped. Since 1:4 is mapped to 1:3 in the aggregate table, the packet will have label 1:3:7 upon leaving LSR C. Once reached the LER D, the packet exits the APLS network. Note that none of the LSRs in the network change the SID in the label and all non-aggregate LSRs swap both the AID and the FID.

In APLS, aggregate forwarding is an optional mechanism where the interpretation of the AID field can be optional. If network provider chooses to a deploy APLS implementation that supports AID then it will be able to take advantages of AID aggregate flow. Otherwise, the AID will be interpreted as part of the FID space.

VLS allows aggregate forwarding with subtle flexibility in contrast with ATM. In respect to aggregate forwarding in ATM, a LSR is either dedicated for aggregate forwarding (does it for all labels) or does normal label forwarding only. VLS offers aggregation to be done at the granularity per SID, that is, per service provider per LSR. Therefore, depending on the service's traffic pattern, the service provider can form customised aggregate flow paths. Service providers can setup an edge to edge aggregate path if a

traffic highway is desired. For example, in Figure 10, LSR C does not need to be an aggregate LSR for other service providers, it is only the decision made by service provider of SID 1 that labels of SID 1 are to be aggregate switched at LSR C.

MPLS supports aggregate forwarding via the use of label stacking. Label stacking can be used to establish LSP tunnels in some part of the network. By directing multiple LSPs to use the same LSP tunnel, aggregate forwarding can be achieved. If a MPLS packet's top label (the label at top of the stack) is mapped to a LSP tunnel, then this LSR is the entry point of a tunnel and the tunnel's label will be pushed onto the packet's label stack. As the MPLS packet traverses through the tunnel, the intermediate LSR will treat this packet as a normal MPLS packet and performs label switching on the top label only. Once the MPLS packet reaches the LSR at the end point, the top label is popped off and the next label on the stack is used for label switching.

In contrast with ATM's approach, label stacking does not require dedicated aggregate switches and have control at the granularity of per label per LSR. ATM's approach increase complexity since only the labels with common AID can be aggregated together. This will require careful label space planning otherwise aggregating will be difficult to achieve in the future. However, the ATM approach is more efficient then using LSP tunnels. LSP tunnels require at least one additional label overhead per packet that traverses the tunnel.

In MPLS, labels used in different hierarchical levels belong to the same label space. That is, incoming labels must be unique in the LSR regardless of which hierarchical level it belongs to. The reason behind this behaviour is that the operations done on the labels are independent of the hierarchical level. Therefore, the scalability of aggregating in MPLS is limited by the label space size.

Label stacking is a mandatory feature of APLS while AID support is optional. APLS domains that support AID leave the decision to the service provider to make i.e., which approach best suits its requirement.

Another possibility is to explore controlling aggregate forwarding at granularity of per AID. This offers finer control to the service provider at the expense of higher complexity.

3.2.2 APLS Label Stack

APLS's label structure is different to that of MPLS. The positioning of APLS over IP allows APLS to utilise the IP header to its advantage. The APLS label consists of two components, the label space and the auxiliary space. Figure 11 shows the APLS label makeup.

The label space is the VLS that identifies the LSP and it is hierarchically structured into SID, AID, and FID. SID determines the number of service

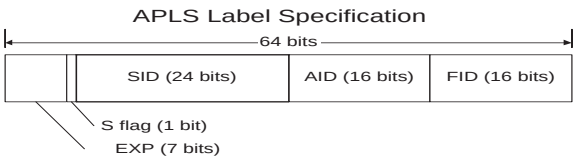

Figure 11: APLS Label Specification

provider that APLS can lease VLS thus the size of SID defines the scalability of the network in terms of service provisioning. The combination of AID and FID gives the effective label space that service providers can work with. This determines both the scalability of the service using VLS and the APLS network. With the rate that Internet is growing and the potential it offers, it would be a disaster for any network technology to under estimate the scale the Internet and services in the future (e.g. IPv4 address problems). Therefore, APLS label is designed more generously to assure future scalability. SID is defined to use 24 bits which means theoretically, APLS supports up to 16777215 VLSs. The effective label space given to service providers to control is defined to be 32 bits. With the support of LSP-tree, 12 bits more than MPLS's, and 4 bits more than ATM's, service providers can be assured that scalability is not an issue of the network architecture, but rather the hardware itself.

The auxiliary space currently consists of two sub-components: the S flag and the EXP bits. Like MPLS, APLS supports label stacking. The one bit S flag resembles the bottom of stack status field in a MPLS label. The EXP field consists of the remaining 7 bits that are reserved for potential use by future enhancements.

Time-To-Live (TTL) field as seen in the MPLS label is not needed in APLS since the IP header already supply this. DSCP field in the IP header can be used to implement QoS on APLS without the need to use the EXP field. EXP can be used to enhance the limitation of the small DSCP field.

Using the IP header field will have consequences and the most significant of these is the invalidation of the checksum when TTL or DSCP is modified. One choice is for every LSP hop to redo the checksum of the IP header but this will increase processing overhead of label switching. The other choice that is chosen by APLS is to delay re-validating the checksum until the packet exits from the APLS domain. The advantage is that core LSRs do not do checksums and these checksums are only done at the less critical edge area of the network.

To reduce label overhead when using label stacks, SID does not need to be included in labels other than the label at the top of the stack. This is because SIDs are global within the APLS domain (or Autonomous System). Thus non-top label entries are 40 bits each. Switching SID within the APLS

domain will not be allowed due to the need to isolate VLS from each other. If the service provider has VLSs in two adjacent APLS domains, then it is desirable for APLS to allow SID switching across APLS domains.

3.2.3 Multiple Data Structure Support

This is another optional feature of APLS. It is possible for network providers to offer service providers control of their own label data structures. The main advantage to the service provider is in terms of performance. Separating data structure reduces the average lookup times for packets belonging to the SID. This is achieved through breaking away from the large default data structure. Network provider can offer a range of data structures with each data structure catering for different conditions. E.g. a large service provider will use a larger part of the EVS thus it will choose a data structure that performs the best for that scenario.

Further research needs to be done in analysing what effect supporting multiple data structure will have on the LSR as a whole.

3.3 APLS Micro-instructions Architecture (AMIA)

Micro-Instructions concept has been around for a long-time, mainly used in computer architectures. The goal is to offer a set of simple instructions that can be combined to form a more complex and higher level set of instructions. What Micro-Instructions can offer to APLS is flexibility to reconfigure label operations. The application of Micro-Instructions to label switching architecture is not new. A MPLS implementation in Linux[21] uses a form of instruction set label operation architecture. However, Micro-Instructions is not part of the MPLS architecture design and is not generalised offer more then just implementation flexibility.

APLS Micro-instructions Architecture (AMIA) is novel in its generalised application to label switching. The objective of AMIA is to give flexibility to customise label operations and the basis to integrate complementary technologies onto APLS networks.

3.3.1 AMIA Core Micro-Instruction Set

APLS core functionality is label switching thus the core instruction set in AMIA involves supporting the following basic label operations:

- **POP**: pops a label from the top of the label stack.
- **PUSH**: pushes a label in the top of the label stack.
- **FWD**: forwards the packet to the network interface.
- **DLV**: delivers packet to the IP layer.

Using the micro-instructions to form the basic label operations:

- **Core Switching**: This is the operation of the core LSRs. First, the LSR pops the top label off the packet's label stack and then uses the label to do the forwarding lookup. The resulting forwarding entry contains the outgoing label, and the network interface to use. The LSR then pushes the outgoing label onto the label stack and switches the packet to the network interface. Thus the sequence of micro-instructions are POP, PUSH, and FWD.
- **Ingress Switching**: This is the operation for packets entering APLS networks. The ingress LER decides the FEC for this incoming IP packet (routing decision). The FEC indicates the LSP to use which includes the outgoing label and the network interface information. The LER will push the outgoing label onto the packet's label stack and switch the packet to the network interface. Thus the sequence is PUSH and FWD.
- **Egress Switching**: This is the operation for packets reaching the end of the LSP. First, the LER pops the top label off the packet's label stack and then delivers the packet to the IP layer for IP forwarding/routing. Thus the sequence contains POP and DLV.
- **Penultimate Hop Popping**: Penultimate LSR refers to the LSR before the egress LER for the LSP. The Penultimate Hop Popping[22] operation is the same as that of Core Switching except no label will be pushed onto the packet's label stack. Thus the sequence consists of POP and FWD.

For performance, frequently used pattern of micro-instructions can be combined into one complex micro-instruction. For example, ingress switching can be done with a micro-instruction and does both PUSH and FWD.

Sequence of micro-instructions is mapped to the FEC and Incoming Label entry to define the label operation. IP packet entering the APLS domain through a LER is mapped to a FEC that is associated with a sequence of micro-instructions. The micro-instructions determine the forwarding behaviour of the IP packet which normally consists of pushing a label into the packet and forwarding the modified packet to the next hop LSR. Upon receiving an APLS packet, LSR finds the entry for the Incoming label using the top label. The entry contains a sequence of micro-instructions that is executed to forward the packet.

3.3.2 AMIA Extended Micro-Instruction Set

These micro-instructions are considered optional and is designed to support complementary technologies. The only micro-instructions that are proposed at this stage are for supporting QoS architectures and Active Programmable Networks (APN).

The micro-instructions i-APN and i-QoS instruct APLS that the packet for this LSP need special processing. APLS will then invoke the handler code that interacts with the complementary technology. For example, APN normally requires the use of more general processor to realise the programmability thus the use of a separate processors for APN is highly possible. APLS's i-APN is an interface to invoke a handler to communicate with the APN. The handler is implementation specific.

In this way, it is possible to enable and disable complementary technologies at a very fine granularity of per LSR per incoming label. This offers another degree of flexibility into APLS. For example, it is possible to configure selective LSRs within each LSP to use APN. If the last part of the LSP have high error rate, APN that does Forward Error Correction (FEC) can be setup at LSRs on that last part only. It is also possible to select which LSP should use QoS and what type of QoS.

A discussion on supporting QoS and APN follows.

3.3.3 QoS on APLS

An example of how QoS can be integrated with APLS using AMIA is given here. Per LSP QoS can be setup easily by using CR-LDP [23]. For each LSR in the LSP, CR-LDP injects the i-QoS micro-instruction along with QoS architecture specific parameters. Executing the i-QoS micro-instruction invokes the QoS Handler with arguments such as the QoS parameters, EXP bits, and the DSCP field of the IP header etc. The QoS Handler in turn determines which output queue the packet lines up to.

3.3.4 APN on APLS

The standard problems when designing APN architectures is to identify *active packets*(packets that needs special processing) and also identifying which Execution Environment (EE) to serve the active packet. Various prominent solutions are shown below.

- **Reference approach**: ANEP Protocol[24] uses Protocol Identifier field in IP header to identify ANEP packet. The ANEP is a protocol designed for APN support. It includes a Type ID field to demultiplex between different EEs (different APN architecture). Nygren et al. [25] and Wetherall et al. [26] use a similar approach which uses some header to identifier the EE.
- **Tag approach**: Simple Active Packet Forwarding (SAPF)[27] introduces a shim layer on top of the link layer. A tag is injected at the front of the packet to demultiplex between EEs. The tag becomes a global identifier for the EE in the network domain. In SAPF, IP is just another network architecture implemented in a EE. Although SAPF

reduces overhead and increase forwarding efficiency for active packets, but it requires every packet within the SAPF network to use labeling.

- **Connection-oriented approach**: In this approach, a loose connection between the source and destination needs to be set up. Loose connection means that the packet must traverse a selected set of APN-enabled routers before reaching the destination. The loose connection involves setting up states in the APN routers such that the packet will be forwarded to the next determined place until reaching the destination node.

  After the loose connection is established, the source sends IP packet to the first APN router in the connection. When the IP packet reaches the APN router, the router will lookup its connection state to see what special processing is required and what is the next hop destination which can be another APN router or the ultimate destination. When forwarding to next hop, the destination address is switched with the next hop's destination address. The packet is then IP routed to the next hop destination.
- **Filter approach**: PRONTO[28] and ORE[29] use filters such as UDP/TCP port number or destination address to retrieve packets for the APN on a per hop basis. Note that when the filter includes the source address then it becomes very similar to the connection-oriented approach, but the difference is that Filter approach does not include connection setup mechanisms since it is per-hop.

The ANEP protocol resolve identification problems by introducing a header on every active packet while SAPF introduce a smaller header (label) for every packet in the network. SAPF is also very hard to deploy since it requires every router in the network to be changed. The Connection-oriented approach introduces complexity with connections including setup overhead, but eliminates the need for a per active packet overhead. However, it has scalability problems for supporting connections in large networks. The filter approach does not require any header or control overhead as seen in other approaches, but the downside is that granularity of control is very coarse. Filter approach is also less intrusive since it offers transparency to the end-host i.e. the end-host's network software does not need to be modified.

APLS extends IP to become LSP-oriented and AMIA can control APN behaviour at the granularity of per LSR within each LSP. The result is that when setting up LSP, it is possible to also setup the use of APN for that LSP. This incurs additional setup overhead but eliminates the need for per packet overhead. Although granularity is not as fine as connection-oriented approach, however, there is no scalability issue for LSP on large network[3]. On top of this, APN on APLS offers transparency to the end-

[3]Scability issue for APN in general still exists e.g. when special computation speed is slower then forwarding speed.

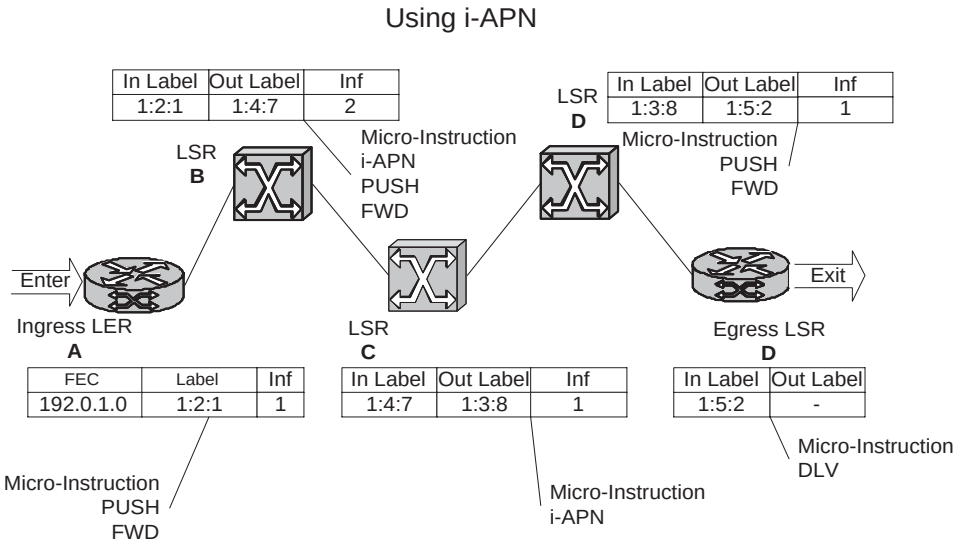

Figure 12: APN support in AMIA

host but without sacrificing control.

Figure 12 illustrates APN on APLS. For each LSR on the LSP that is selected to do special APN computation, i-APN is injected into the label's sequence of micro-instructions for that LSR. The sequence for the core LSR B would be: i-APN, PUSH, FWD. This sequence means that APN will do some content manipulation but would not override APLS forwarding behaviour. If the sequence is like LSR C: i-APN. This means that the APN will override the forwarding behaviour and thus forwards the packet itself.

To interoperate with other APN networks, APLS can also work with ANEP. This will only require the i-APN to use a handler that understands ANEP. The general concepts of APN on APLS remains the same.

3.4 Micro-Policy Based Forwarding

APLS like MPLS supports multi-path routing and multi-path forwarding.

- **Multi-path routing**: occurs when multiple LSPs are mapped to the same FEC. This can be useful for traffic engineering, setting up LSPs for backup, and load balancing.
- **Multi-path forwarding**: occurs when multiple outgoing labels for same or different network interface are mapped to the same incoming label. This can be useful to setup alternative partial paths for sections of the LSP that frequently goes down, or for load balancing at critical points in the network. Multicasting protocols can use this for creating multicast LSP-tree[30].

The MPLS standard does not define how these multi-path mechanisms are used. Therefore, a Micro-Policy (MP) based approach is introduced here. A list of labels is assumed to be associated with the FEC entry or the incoming label entry. A MP function must be associated with this entry if more than one outgoing label are mapped to the entry. The MP function selects the outgoing label to use on per-packet basis. Due to the per-packet execution nature, MP function must be small enough to run at wire speed.

Although it is possible to put some programmability to the network through customisable MP functions, however, programmability is better achieved with the Micro-Instruction Architecture introduced in section 3.3. For high performance, MP functions would most probably be placed in firmware in the routers. Thus a well-defined set of MP functions should be offered to ensure flexibility. However, if Field Programmable Gate Arrays (FPGAs) are used to implement the MP functions then it is possible for network providers to customise the MP functions.

Examples of MP functions are:

- **First in List**: This is useful when backup LSPs exist but only one LSP is ever used at a time.
- **Round Robin**: This is for a simple load balancing scheme.
- **QoS**: using part of the EXP bits in the APLS label to determine which LSP to use.
- **Statistical**: using certain statistics of the LSR or the LSPs to determine which path to use. This can be a power basis for dynamic forwarding adaptation.
- **Broadcast**: use all the outgoing labels. This is for multicasting.

It is also possible to use MP functions that look into the IP header for additional information for better decision making. For example, the DSCP field can be used by the QoS MP function.

Requirements for Traffic Engineering (RFC2702)[10] suggest the use of preference rules for multi-paths. In this approach, preference rules are applied to select a path during infrequent events such as path establishment or failure scenarios. MP-based forwarding focuses on per-packet granularity of control. In comparison, preference rules can be more heavyweight but with a coarse grain of control whereas MP-based forwarding are more lightweight but has finer grain of control. Rather than perceiving the two approaches as competing with each other, these two approaches can complement each other. A possible scenario is to use preferences rules to control MP-based forwarding e.g., preference rules can determine which path goes to the front of the path list when First in List policy function is in place. If QoS MP function is in use, preference rules can determine the EXP bits to LSP mappings.

3.5 Controlling APLS

APLS is a forwarding architecture that needs to cooperate with other technologies to form a complete network architecture. The forwarding information that APLS uses is controlled and distributed by the control architecture. The control architecture can consists of multiple protocols that specialise in different functions of control. For example, LDP may be used for distribute labels, and RSVP-TE[31] may be used for resource reservation and traffic engineering.

The introduction of service provider control into the forwarding architecture opens up several control-model possibilities:

1. All service providers use the same standard LDP. This LDP functions as the automatic and dynamic distributor of labels. A RSVP-TE like protocol will coexist to allow service provider to explicitly create LSPs and use other traffic engineering functionalities. Manual logins to LSR allows service providers to manually modify label behaviours on a particular LSR. Manual change will be necessary only if given protocols cannot achieve certain requirements needed by the service provider.

2. All service providers use the same LDP but the LDP is designed for APLS. This LDP should allow service providers to customise the decision making process that involves their labels. This can be achieved through the introduction of decision policy mechanism in the LDP. For example, when LDP is attempting to distribute labels for the VLS, the VLS's decision policy should be used to determine what labels to distribute. When receiving distributed labels information of the VLS, the VLS's decision policy should be used to determine what to do with the labels.

   It is possible to define a standard set of policies that service providers can choose from and customise through parameters. More flexible approaches like service provider defined policies are also possible but raise some security issues and complicate the LDP significantly.

3. Merwe et al.[32] introduced the idea of multiple control architectures running on the same LSR. This allows service providers to deploy their own control architecture for their label space. This requires heavy processing (memory and CPU) and networking (bandwidth) resources thus is not expected to scale. Therefore, only privileged service providers should be allowed to deploy their own control architecture. An alternative is for network providers to support a set of control architectures with each service provider choosing the one they want their VLS to use. Thus in this model, multiple service providers share a set of control architectures.

4 Linux Implementation of APLS

We have built a proof-of-concept prototype in Linux kernel 2.5.2. Linux was chosen as the platform because it is open source and is supported by a vast amount of related research and developments that APLS may exploit in the future.

An abstract implementation design is discussed first followed by the main components design.

4.1 APLS Abstract Implementation Design

The abstract implementation design of APLS can be see in 3 dimensions. The control flow can be shown in the first two dimensions while the third dimension shows the data flow. Figure 13 shows the 3 dimensions APLS design flatten into a 2 dimensional layout. The vertical flow is for control while the horizontal flow is for data forwarding.

The core of the APLS implementation is the forwarding engines:

- **Ingress Forwarding Engine**: This engine is for the ingress LER nodes that are required to map packets to FEC and map FEC to LSPs. The sequence of micro-instructions associated with the FEC/FTN is executed here.
- **Core Forwarding Engine**: This engine is for the core LSR nodes that perform the normal label switching. Egress LER node functionalities are also built into this engine. The sequence of micro-instructions associated with the incoming label (ILM) is executed here.

For a normal software implementation, it is possible to integrate these engines into one but if multi-processors or ASICs are used then parallelising these operations for performance would be desirable.

If router supports Micro-Policy Based Forwarding then whenever multi-path exists, the forwarding engine must execute the associated MP function to choose a path to use. MP functions are stored in the MP Database and invoking these functions are done through the MP Interface.

IP packet entering an APLS domain goes through the ingress forwarding engine and becomes a APLS packet if a FEC to LSP mapping is found for the IP packet, otherwise, the IP packet will be IP forwarded to the next hop. If special processing is required, like APN, then the packet control is passed vertically to the APN handler and from there to the execution environment. If the APN processing overrides the forwarding behaviour then the packet control is totally given to the Execution Environment i.e., the Execution Environment explicitly does the label operations and forwards the packet, or the packet gets dropped. If the APN processing does not override then the control is later passed back to the forwarding engine and the rest of the

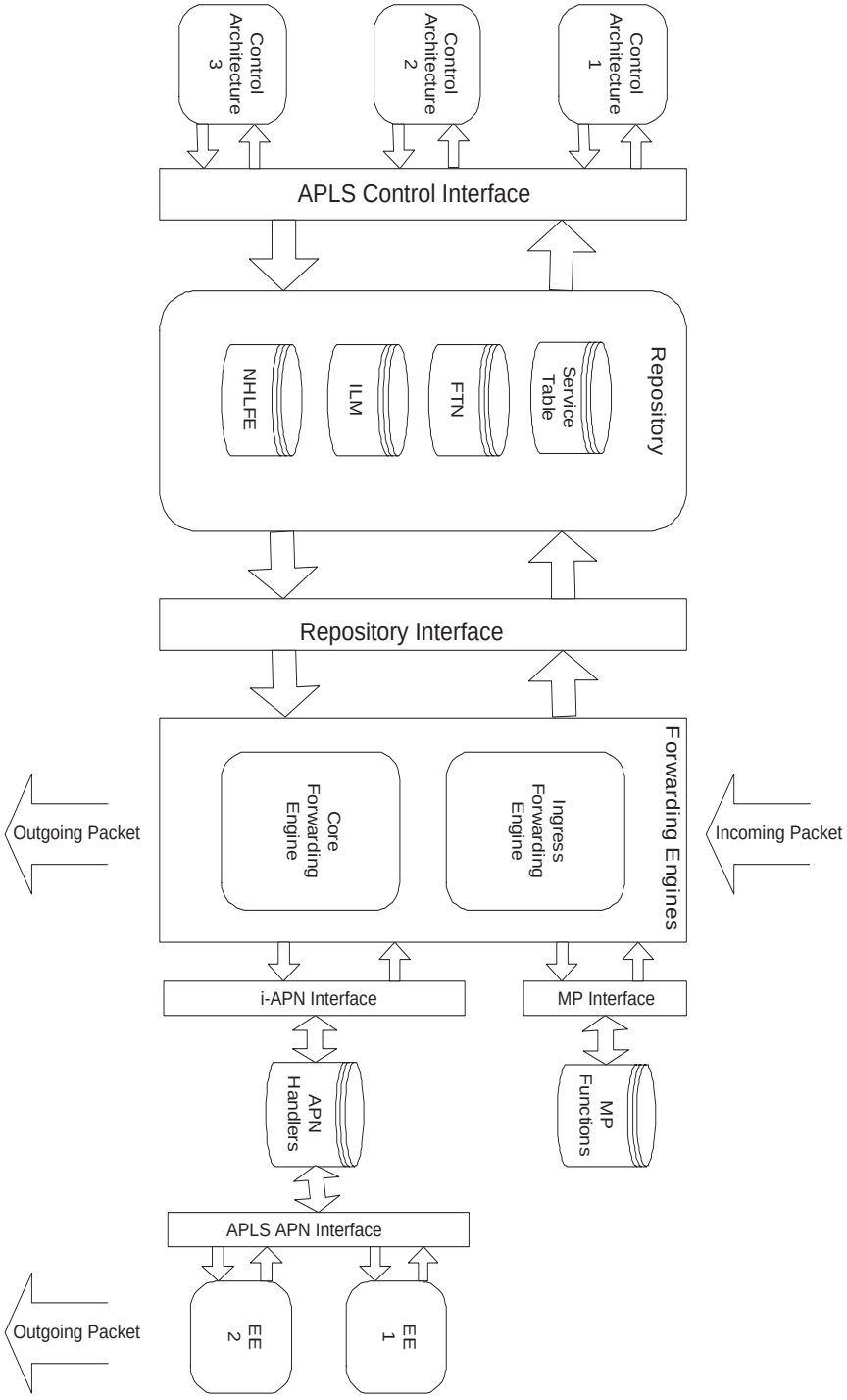

Figure 13: APLS Implementation Design.

label operations continue to be executed. The forwarding engine should not be blocked while this packet is being processed by APN.

The main information base are stored in the Repository which consist of:

- **Service table**: Every SID in use will have an entry in this table. Each entry will have a reference to the incoming label table and the outgoing label table for this SID.
- **Incoming label table (ILM)**: Each SID will be allocated its own Incoming label table. The Incoming label table uses the AID and FID as the key to find the entry.
- **Outgoing label table (NHLFE)**: Each SID will be allocated its own Outgoing label table. Since outgoing labels need to be unique if the labels are from the same network interface hence the key for this table consists of the AID, FID, and a unique identifier for the network interface.
- **FEC table (FTN)**: When IP packet enters the APLS domain, the packet must be mapped to a FEC. The FEC table contains an entry for each FEC in used by APLS. The key for this table depends on how FEC is defined in the implementation. A common FEC is the use of the longest match destination network prefix.

The forwarding engines access the Repository through a well-defined Repository Interface.

On the other side of the control spectrum is the Control Architectures. APLS only supports the interaction with multiple control architectures simultaneously, other kind of support are outside the scope of this project (refer to Section 5.6 for more information). Control architectures control the data forwarding through manipulating the control information used by the forwarding engines. This includes, the label mappings, micro-instructions, and micro-policy. The control architectures access the Repository through the APLS Control Interface.

4.2 APLS Implementation

An introduction to the implementation of the three main components: Ingress Forwarding Engine, Core Forwarding Engine, and the Repository is given along with the Repository and APLS Control Interfaces.

The implementation of the core concepts: AMIA and the Micro-Policy Based Forwarding are also discussed.

4.2.1 APLS Micro-Instruction Architecture (AMIA)

The APLS Micro-Instruction Architecture (AMIA) is used in two places The first place is the FEC table where each entry will contain a sequence

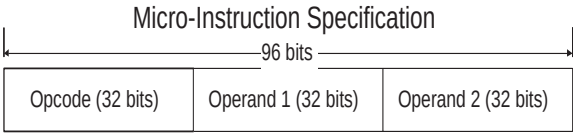

Figure 14: Micro-Instruction Specification

of micro-instructions that must be executed by the Ingress Forwarding Engine. The second place is the Incoming Label table where each entry will contain a sequence of micro-instructions that must be executed by the Core Forwarding Engine.

Each micro-instruction will have the format as shown in Figure 14. The first part of the instruction is the opcode which consists of 32 bits and the last two parts are the operands that consists of 32 bits each. The hardware used is a 32 bits architecture hence the opcode is rounded up for alignment. Operands used in this implementation are normally pointers hence 32 bits are the minimum required. Note that this is a software implementation hence there is no notion of registers. Both operands are for arguments of the micro-instruction. Output is assumed to be store in some shared variable that persists over the execution of the micro-instructions.

The POP micro-instruction (i-POP) removes a label from top of the label stack and puts it in the shared variable *label*. This i-POP will also implicitly retrieve the entry associated with this incoming label and reference it in the shared variable *entry*.

The PUSH micro-instruction (i-PUSH) pushes a label onto the label stack. There is no operand for i-PUSH because it depends on the previous i-POP. That is, the outgoing label is mapped to the incoming label entry, therefore, i-PUSH will use the shared variable *entry* to get the outgoing label to push. This micro-instruction assigns the network interface associated with the outgoing label into the shared variable *netintf*.

The FWD micro-instruction (i-FWD) forwards the packet onto a network interface. One operand is used to indicate the network interface to use but this operand is optional. If this operand is not NULL, then the specified network interface will be used. Otherwise, i-FWD assumes dependency on i-PUSH. That is, the network interface specified in the shared variable *netintf* is used.

The DLV micro-instruction (i-DLV) delivers the packet to the IP layer for IP forwarding.

4.2.2 Ingress Forwarding Engine

On receiving an IP packet, a LER will lookup the FEC table to determine if the packet should use normal IP forwarding or the use APLS forwarding.

```
int apls_ingress_engine(struct apls_fec *fec_entry,){
                    struct ip_packet packet){

        for each micro-instruction in fec_entry do{
                case micro-instruction is PUSH:
                    label = choose_outgoing_label(fec_entry);
                    if (first_label_pushed)
                        set_S_flag(label);
                    push_outgoing_label(packet,label);
                    mark_packet_APLS(packet);                          10
                case micro-instruction is FWD:
                    decrement_TTL(packet);
                    if(packet_not_apls(packet))
                        do_checksum(packet);
                    forward_packet(fec_entry,packet);
        }

        return SUCCESS;
}
```

Figure 15: Ingress Forwarding Engine

If APLS forwarding is indicated then a LSP is mapped to the FEC entry. It is possible that multi-path routing is used where multiple LSP is mapped to one FEC entry but this will be discussed later in Section 4.2.5.

If APLS forwarding is indicated then the APLS Ingress Forwarding Engine will gain control of the IP packet. The simplified operation for the Ingress Engine is shown as pseudo code in Fig. 15

For the Linux implementation, the IP layer's FIB table is used as the FEC table and each entry in the table contains the function to call when a FEC match occurs. The function can point to the normal Linux IP forwarding function or the APLS Ingress Engine function. By default all, FIB entries use the IP forwarding function unless explicitly overrided by the control architecture. For example, when BGP receives a new network route it may decide to use IP forwarding or to use APLS forwarding. If APLS forwarding is decided then the LER will associate a label to this route and distribute it using the LDP.

In effect, APLS and IP share the FEC table but each FEC can only use one type of forwarding. If FEC uses APLS then the FEC entry will reference a APLS data structure. This data structure contains information like LSP, sequence of micro-instructions, and the network interface.

4.2.3 Core Forwarding Engine

When an APLS packet is received by the IP layer, the packet control will be passed on to the APLS Core Forwarding Engine. The Core Engine will

execute the micro-instructions associated with the top label of the packet.

The simplified operation for the Core Engine is shown as pseudo code in figure 16.

4.2.4 Repository

There are three data structures in the repository:

- **Service table**: The prototype is not expecting the use of a large number of SID and hash table supports this scenario well. A hash table with overflow chain is used for this table. Best case requires one memory access to get the entry. Worst case is $O(n)$ where n is the number of SID in use.
- **Incoming label table (ILM)**: The prototype expects to support a large number of incoming label per SID thus need a data structure that can grow with usage. A multiple-level tree is used with 8 bits per level. For all cases, it requires four memory access to get the entry.
- **Outgoing label table (NHLFE)**: The prototype expects to support a large outgoing label table thus the same strategy for Incoming label table is used.
- **FEC table (FTN)**: Linux IP's FIB table is used also for the APLS FEC table thus FECs are identified by network prefix. Linux FIB table uses multiple hash tables.

More research needs to be done to choose the best performing data structure for the tables in the repository. The current prototype chooses simple and effective data structures as an interim solution. More complex data solutions like LC-trie [33] and Patricia tree [34] will be explored in the future. For the FEC table, the current well-implemented FIB table from Linux is in used but for more edge on performance, more complex structures [35, 36] need to be investigated.

The two interfaces for accessing the Repository are described below:

- **Repository Interface**: The Repository Interface is an interface for internal components of APLS such as the forwarding engines. Internal components only need read access to Repository's information while control protocols that are external to APLS will need both read and write access. External access interface is discussed in the next section.

  The Repository Interface is designed to be independent of the data structures in use, therefore, changes in the Repository can be made transparent to other components. The list of the current Repository Interface functions are listed in Fig.17.

  There is no interface function for accessing the Outgoing Label Table since the FEC table and Incoming Label Table entries have direct reference to the Outgoing Label Table entries. This referencing is done

```
int apls_core_engine(struct apls_packet packet){

        label = pop_top_label(apls packet);
        entry = get_incoming_label_entry(label);

loop:
        for each micro-instruction in entry do{
            case micro-instruction is PUSH:
                    out_label = choose_outgoing_label(entry);
                    if (packet_stack_empty(packet))                          10
                        set_S_flag(out_label);
                    push_outgoing_label(packet,out_label);

            case micro-instruction is FWD:
                    decrement_TTL(packet);
                    if(packet_stack_empty(packet)){
                        do_checksum(packet);
                        unmark_packet_APLS(packet);
                    }
                    forward_packet(entry,packet);                            20

            case micro-instruction is DLV:
                    if(packet_stack_empty(packet)){
                        do_checksum(packet);
                        unmark_packet_APLS(packet);
                    }
                    else
                        /*do not deliver APLS packet to IP*/
                        return ERROR;
                                                                             30
                    deliver_ip(packet);

            case micro-instruction is POP:
                    if(packet_stack_empty(packet))
                        return ERROR;
                    label = pop_label(packet);
                    /*control is transfer to the popped label*/
                    entry = get_incoming_label_entry(label);
                    goto loop;
        }                                                                    40

        return SUCCESS;
}
```

Figure 16: Core Forwarding Engine

```
/*Given a SID, this function returns a reference to the entry,
 *locks the entry, and returns a reference to the lock associated
 *with the entry. The lock is separate to the entry since the lock
 *does not have to be at granularity of one lock per entry.
 */
int asrv_getEntryLock(unsigned int sid,
                struct apls_service* entry,rwlock_t** lock);

/*Given pointer to the Incoming Label Table and the label
 *to lookup for, returns the entry corresponding to the label
 */
void* apls_ilm_retrieve_entry(void** tree, struct apls_label label);
```

Figure 17: Repository Inteface

when an outgoing label is mapped to the FEC/Incoming Label Table by the controlling architecture.

The FEC table is not used by any internal components. The FEC table is consider to be at the IP layer where it maps incoming IP packets to a FEC and the FEC entry will indicate whether normal IP forwarding is used or APLS forwarding is used. When APLS forwarding is used, the APLS Forwarding Engine is invoked with the FEC entry as the parameter.

The current implementation does not support the AID aggregate forwarding which will be reflected in the Repository Interface.

- **APLS Control Interface**: This is the interface to the Repository for external components, for example, the control architecture. The APLS Control Interface contains both read and write access to the Repository for manipulating forwarding behaviour.

  The APLS Control Interface consists of four sub-interfaces, one for each type of table in the Repository: Service Interface, Incoming Label Interface, Outgoing Label Interface, FEC Interface. These interfaces are shown in Figures 18, 19, 20, 21 respectively.

  Note that Linux's FIB table is used as the FEC table, hence, creating and deleting FEC entries is done through the Linux's interface. APLS Control Interface only modifies the FEC entry.

4.2.5 Micro-Policy Based Forwarding

Micro-Policy Based Forwarding applies only when multi-path exists. That is, when FEC is mapped to more than one outgoing label or when Incoming label is mapped to more than one outgoing label. Therefore, Micro-Policy (MP) functions are associated with the FEC and Incoming label entries only. The associated MP function is executed by the corresponding for-

```
/*********Service Interface**************/

/*Creates a new service entry for the given SID*/
int apls_service_create(int sid);

/*Deletes an existing service and clear all allocated resources
 *to the service
 */
int apls_service_del(int sid);

/*Set the service to be available (online) or unavailable (offline)
int apls_service_avail(int sid, int avail);
```

Figure 18: Service Inteface

warding engine. For example, when a LSR receives an APLS packet, the Core Forwarding Engine uses the top label to retrieve the Incoming Label entry. When deciding on the outgoing label to use, the engine will execute the MP function which makes the decision and the engine will then push the chosen outgoing label into the APLS packet.

The current implementation uses a link list for the outgoing label mapped to the entries and the standard MP function interface is shown in figure 22.

The top label's auxiliary bits are used for QoS related functions. Reference to the APLS packet is used for QoS functions that want additional information from IP header.

5 Related Works

5.1 MPLS and ATM

APLS is not a completely new label switching architecture. It uses the general label switching concepts from MPLS as the base thus can inherit many of the enormous research and developments done on MPLS. The greatest difference between APLS and MPLS would be the approach of positioning within the networking stack. MPLS runs between layer 2 and 3 whereas APLS runs between layer 3 and 4. The arguments for and against have been put investigated in section 3.1. The new concepts VLS and AMIA introduced in APLS, pushes for better Network Level Service Engineering that are not seen in any existing label switching architecture.

Due to similarity of APLS and MPLS, the differences between ATM and APLS would be very similar to the differences between ATM and MPLS. The major differences is the use of variable size packet, LSP-tree, and label stacking in MPLS. ATM has dominated the core networks in most carriers

```
/*********Incoming Label Interface*******/

/*Adds a new incoming label into the system. The argument is
 *the VLS label that includes the SID.
*/
int apls_ilm_add(struct apls_label label);

/*Deletes an incoming label from the system*/
int apls_ilm_del(struct apls_label label);

/*Associate a given set of micro-instructions to the specified
 *incoming label.
 */
int apls_ilm_set_instr(struct apls_label label,
                       struct apls_instr instr);

/*Maps an outgoing label to an incoming label. They must
 *both have the same SID.
 */
int apls_ilm_map(struct apls_label in, struct apls_label out);

/*Remove the mapping of an outgoing label from an
 *incoming label*/
int apls_ilm_unmap(struct apls_label in, struct apls_label out);

/*Remove all mappings from an incoming label*/
int apls_ilm_unmap_all(struct apls_label in);

/*Makes the incoming label online or offline*/
int apls_ilm_avail(struct apls_label in, int avail);
```

Figure 19: Incoming Label Inteface

```
/*********Outgoing Label Interface*******/

/*Add a new outgoing label with association to a particular
 *network interface.
 */
int apls_nhlfe_add(struct apls_label label, struct net_intf intf);

/*Deletes an outgoing label that is associated with the given
 *network interface.
 */
int apls_nhlfe_del(struct apls_label label, struct net_intf intf);
```

Figure 20: Outgoing Label Inteface

```
/*********FEC Interface*****************/


/*Modifies the FEC entry to start using APLS forwarding
int apls_ftn_create(struct apls_fec fec);

/*Modifies the FEC entry that is using APLS forwarding to start
 *using IP forwarding.
 */
int apls_ftn_delete(struct apls_fec fec);

/*Maps an outgoing label to the FEC. This effectively maps
 *a LSP to the FEC.
 */
int apls_ftn_map(struct apls_fec fec, struct apls_label label);

/*Removes a given outgoing label mappings from the FEC*/
int apls_ftn_unmap(struct apls_fec fec, struct apls_label label);

/*Removes all outgoing label mappings from the FEC*/
int apls_ftn_unmapall(struct apls_fec fec);

/*Associate a set of micro-instructions to the FEC entry*/
int apls_ftn_set_instr(struct apls_fec fec,struct apls_instr instr);

/*Makes the FEC online or offline*/
int apls_fec_avail(struct apls_fec fec,int avail);
```

Figure 21: FEC Inteface

```
/*Returns the chosen outgoing label*/
/*Parameters: top label's auxiliary bits, reference to APLS
 *packet, the list of outgoing labels.
 */
struct apls_label mp_function(unsigned char aux,
        struct apls_packet* packet, struct apls_label_ele* list);
```

Figure 22: MP Function Interface

but its complexity and inflexibility will hinder the future growth of services on the Internet.

5.2 CISCO's Universal Transport Interface (UTI)

UTI [37] was designed for tunneling layer 2 or 3 packets within a normal IP network. A special UTI header is inserted between layer 3 and layer 4. This header consists of two components: Tunnel ID and the Tunnel Key. The Tunnel ID is used to identify by the end point to identify the tunnel and thus what actions need to be taken thereafter. The Tunnel Key acts as the signature between the tunnel end points. UTI resembles APLS in the positioning it takes in the network stack and the use of special header to control forwarding. However, there are very subtle differences. UTI header is only meaningful at the tunnel end points (edge routers) thus is not a label switching architecture. It uses something similar to label switching to bridge external networks to interoperate with the IP network. UTI does not provide any mechanisms to improve the IP network in general.

5.3 ATM's Virtual Label Space

Wang et al. [38] informally introduces ATM's Virtual Label Space as a proposed potential solution to tackle the VC Space scalability issue. This involves using parallel network interfaces with each interface having its own label space. The recommendation was to localise the use of virtual label space to the hot spots within the network where it is needed. MPLS also allows similar approach to implementations where it is defined as the *scope and uniqueness of labels* [22]. APLS's VLS approach is totally different. VLS's virtualness is directed towards the service provider and is structured within the label itself. Thus the virtual space exists across the whole APLS domain. Implementation may also provide different scope for VLS per interface to increase the effective label space, but this is orthogonal to APLS's VLS concept.

5.4 BGP/MPLS VPNs

BGP/MPLS VPNs [39] is a standardised approach for realising VPN on MPLS domains. The approach uses dedicated FTN tables for each VPN at the LERs and uses a modified Border Gateway Protocol (BGP) to distribute the private addresses of the VPN. APLS provides optional support of a separate label data structure for service providers. The objective is not for realising VPNs but for offering potential performance advantages over the use of a single shared label data structure for all service providers. Separate label data structure support is not restricted to LERs as in BGP/MPLS VPN. Any LSRs within APLS network can support separate label data structures.

Performance issues in APLS's multiple label data structures support are very similar to those found in Virtual Memory [40, 41] support in OS. However, OS characteristics are different to that found in networks, therefore, further research needs to be done to analyse the feasibility of multiple data structures.

5.5 APN

APN [2] is seen as a complementary technology to APLS thus many concepts introduced in APN can be integrated with APLS. However, some APN uses the approach of positioning network layer architectures on top of APN (NAoAPN). NAoAPN contradicts with the APLS model and is considered as a separate network architecture in itself. The advantages of NAoAPN approach is that it gives flexibility to the Network Providers to support multiple network architectures on the same equipment. It also allows network architectures to be customised and upgraded easily. However, there are some inherent problems with NAoAPN:

1. High performance router usually uses ASIC rather than General CPUs hence whether it is possible to realise a high performance router for NAoAPN is questionable.
2. Supporting multiple network architectures increase the complexity of the network and raises the operational costs. This may be attractive for transitioning network architectures, but what is desired by the network/service providers is one unified network that has performance, scalability, and flexibility.
3. NAoAPN approaches require special protocol like SAPF [27] to demultiplex between the architectures. Thus the flexibility comes at an increase in overhead per packet.

Therefore, the major advantages of NAoAPN are the ability to customise and upgrade the network without the need to change the base equipment. NAoAPN will only be readily accepted when NAoAPN routers that can rival the performance of IP/Label Switching routers are developed. The Genesis Project [42] is a good example of the NAoAPN approach.

5.6 Tempest

Tempest [32] is very similar to APLS in the network model that it proposes. The major difference between Tempest and APLS is the approach taken to realise the network model.

Merwe et al. attempts to modify the ATM architecture into a service-oriented one by separating the control from the switching fabric. Instead of using the standard PNNI control architecture, Tempest inserts an additional layer of control. This layer controls and partitions the ATM switch's

resources and allocates the partitions to a logical entity called *switchlets*. Tempest offers multiple control architectures, called controllers, to run simultaneously on one switch. Each controller will be allocated a switchlet that offers a small virtual but complete switch to the controller.

Tempest delegates control to service providers through the control plane of the switch. In contrast, APLS's approach is to delegate control through mechanisms inside the forwarding plane. By using the ATM forwarding plane, Tempest inherits a lot of limitations that already exist. For example, ATM's VC scalability issue will only be deteriorated by the introduction of partitioning of the space. The inflexibility, complexity and the high operational cost of the ATM architecture will also be passed on to the service provider.

The use of multiple control architecture as the sole basis for service provisioning has several critical problems. Firstly, the existence of multiple controllers is transparent to the switch fabric hence Tempest will have to manage VC partitioning and allocation. VC space partitioning and allocation is a complicated issue and gets harder to manage when different service provider size is taken into account. Second, each service providers that want control are required to own separate controllers. This approach is just not scalable in terms of services it can support.

As discussed in section 3.5, APLS offers three types of control model and one of them is multiple control architectures. VLS simplifies the operational and deployment of control architectures by implicitly partitioning the label space. The SIDs can be mapped globally to the chosen control architecture. APLS can increase scalability by offering a balance of control. That is, small service providers share the default control architecture while a small set of large service providers can deploy private control architectures. Thus APLS's approach is more flexibility and scalability.

Tempest and APLS can also complement each other since they focus on different aspect of the switch. It is possible to use Tempest to realise multiple control architectures on APLS. Due to the fact that the ATM fabric is not designed for service support, Tempest inherits a lot of the limitations and needs to work around the inflexible switching fabric. Therefore, Tempest will find a better home on a service-oriented switch fabric like APLS. The combination of VLS and AMIA opens up more of the switching fabric for Tempest to control.

6 Conclusion

APLS's primary focus is to offer new mechanisms that promote service provisioning while retaining the performance advantages of label switching. This results in a label switching architecture that offers higher degree of control, flexibility, and scalability.

APLS's label is 32 bits higher overhead than that of MPLS thus it is expected that APLS will perform slightly slower than MPLS. The trade-offs for this overhead are better service provisioning support and higher scalability.

The concepts introduced in this paper such as AMIA and Micro-Policy Based Forwarding are independent and can be migrated to existing label-switching technologies such as MPLS. However, it is with the complete combination including VLS and Label Switch over IP, that APLS can realise a superior service-oriented network model.

References

[1] R. Venkateswaran, "Virtual Private Networks," *IEEE Potentials*, February/March 2001.

[2] W. Lau, S. Jha, and M. Hassan, "Current directions in Active Programmable Network," in *IEEE International Conference on Networks ICON*, Bangkok, October 2001.

[3] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," *IETF RFC 1633*, 1994.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *IETF RFC 2475*, 1998.

[5] I. Andrikopoulos and G. Pavlou, "Supporting differentiated services in MPLS networks," *Proceedings of IWQoS'99*, May 1999.

[6] I. Foster and C. Kesselman, "Chapter on Computational Grids," *The Grid: Blueprint for a New Computing Infrastructure*, 1998.

[7] R. Wolski, J. Brevik, C. Krintz, G. Obertelli, N. Spring, and A. Su, "Running EveryWare on the computational grid," in *Proceedings of 1999 conference on Supercomputing*, Oregon, US, 1999.

[8] F. Safaei, I. Ouveysi, M. Zukerman, and R. Pattie, "Carrier-scale programmable networks: Wholesaler platform and resource optimization," *IEEE Journal on selected areas in communications, Vol 19, No. 3*, March 2001.

[9] P. Bhaniramka, B. Sun, and R. Jain, "Quality of service using traffic engineering over MPLS: An analysis.," in *Proceedings of the 25th IEEE Conference on Local Computer Networks*, Florida, US, November 2000.

[10] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, "Requirements for Traffic Engineering Over MPLS," *IETF RFC 2702*, 1999.

[11] G. Ash, "Traffic Engineering & QoS Methods for IP-, ATM-, & TDM-Based Multiservice Networks," *IETF Internet Draft*, April 2002.

[12] T. Nadeau, J. Boyle, K. Kompella, W. Townsend, and D. Skalecki, "Protocol extensions for support of Diff-Serv-aware MPLS traffic engineering," *IETF Internet Draft*, August 2002.

[13] K. Kompella, Y. Rekhter, J.P. Vasseur, and T.W. Chung, "Multi-area MPLS Traffic Engineering," *IETF Internet Draft*, December 2002.

[14] G. Swallow, "MPLS advantages for traffic engineering," *IEEE Communications 37(12)*, pp. 54–57, 1999.

[15] C. Metz, "Layer 2 over IP/MPLS," *IEEE Internet Computing*, July/August 2001.

[16] Equipe Communications Corporation, "ATM/MPLS interworking transition strategy: Bringing MPLS to the layer 2 core," *Whitepaper*, 2001.

[17] R. Jaeger, "Transitioning from IP-over-LANE/ATM to IP/MPLS networks," *Juniper Networks Whitepaper*, 2001.

[18] K. Coffman and A. Odlyzko, "Growth of the Internet," *AT&T Labs Research*, July 2001.

[19] IDC, "More is not enough: Bandwidth end use forecast and analysis," *IDC Report*, 2002.

[20] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "MPLS Label Stack Encoding," *IETF RFC 3032*, 2001.

[21] J. R. Leu, "MPLS for linux," *http://mpls-linux.sourceforge.net/*, 2002.

[22] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *IETF RFC 3031*, 2001.

[23] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, R. Gray, J. Heinanen, T. Kilty, and A. Malis, "Constraint-Based LSP Setup using LDP," *IETF RFC 3212*, 2002.

[24] D. Alexander, B. Braden, C. Gunter, A. Jackson, A. Keromytis, and G. Minden, "Active Network Encapsulation Protocol," *http://www.cis.upenn.edu/~switchware/ANEP/docs/ANEP.txt*, 1997.

[25] E. Nygren, S. Garland, , and M. Kaashoek, "PAN: A high-performance active network node supporting multiple mobile code systems," *IEEE Communications*, pp. 78–89, 1999.

[26] D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: a toolkit for building and dynamically deploying network protocols," *IEEE Open Architectures and Network Programming*, pp. 117 –129, 1998.

[27] D. Decasper and C. Tschudin, "Simple Active Packet Format (SAPF)," *http://abone.ifi.unizh.ch/sapf/sapf-0.7.html*, August 1998.

[28] G. Hjalmtysson, "The Pronto platform - a flexible toolkit for programming networks using a commodity operating system," in *The Third IEEE Conference on Open Architectures and Network Programming (OPENARCH'2000)*, Israel, 2000.

[29] P. Wang, R. Jaeger, R. Duncan, T. Lavian, and F. Travostino, "Enabling active networks services on a gigabit routing switch," in *The 2nd Workshop on Active Middleware Services*, Pittsburgh, Pennsylvania, August 2000.

[30] D. Ooms, B. Sales, W. Livens, A. Acharya, F. Griffoul, and F. Ansari, "Framework for IP Multicast in MPLS," *draft-ietf-mpls-multicast-08.txt*, 2002.

[31] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," *IETF RFC 3209*, 2001.

[32] J. Merwe, S. Rooney, I. Leslie, and S. Crosby, "The Tempest - A practical Framework for Network Programmability," *IEEE Network*, pp. 20–28, May/June 1998.

[33] A. Andersson and S. Nilsson, "Improved Behaviour of Tries by Adaptive Branching," *Information Processing Letters: 46*, pp. 295–300, 1993.

[34] D. Morrison, "PATRICIA-Practical Algorithm to Retreive Information Coded In Alfanumeric," *Journal of the ACM 15(4)*, pp. 514–534, October 1968.

[35] E. Filippi, V. Innocenti, and V. Vercellone, "Address lookup solutions for gigabitswitch/router," in *Globecom*, Sydney, November 1998.

[36] S. Nilsson and G. Karlsson, "Fast address lookup for Internet routers," in *Proc. IFIP 4th International Conference on Broadband Communications*, 1998, pp. 11–22.

[37] Cisco, "Universal Transport Interface (UTI)," *Cisco IOS Release*, 2002.

[38] Z. Wang and G. Armitage, "Scalability Issues in Label Switching over ATM," *IETF Internet Draft*, December 1997.

[39] E. Rosen and Y. Rekhter, "BGP/MPLS VPNs," *IETF RFC 2547*, 1999.

[40] Y. Khalidi, M. Talluri, M. Nelson, and D. Williams, "Virtual Memory Support for Multiple Page Sizes," in *Workshop on Workstation Operating Systems*, 1993, pp. 104–109.

[41] K. Elphinstone and G. Heiser, "Page Tables for 64-bit Computer Systems," in *Australasian Computer Architecture Conference, Auckland, New Zealand*. 1999, pp. 211–226, Springer-Verlag, Singapore.

[42] M. Kounavis, A. Campbell, S. Chou, F. Modoux, J. Vicente, and H. Zhang, "The Genesis Kernel: A Programming System for Spawning

Network Architectures," *IEEE Journal on Selected Areas in Communications (JSAC), Vol. 19:3*, pp. 49–73, March 2001.