ANMAC: A Novel Architectural Framework for Network Management and Control using Active Networks

Authors: Samphel Norden

In this paper, we propose a new framework called Active Network Management and Control (ANMAC) for the management and control of high speed networks. The software architecture in ANMAC allows routers to execute dynamically loadable kernel plug-in modules which perform diagnostic functions for network management. ANMAC uses mobile probe packets to perform efficient resource reservation (using our novel reservation scheme), facilitate feedback-based congestion control, and to provide "distributed debugging" of complex anomalous network behavior. ANMAC also provides security measures against IP spoofing, and other security attacks. The network manager has the flexibility to install custom scripts in routers for tracking down anomalous network faults.
ANMAC: A Novel Architectural Framework for Network Management and Control using Active Networks

Samphel Norden

WUCS-99-16

May 1999

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130
ANMAC: A Novel Architectural Framework for

Network Management and Control using Active Networks

Samphel Norden

Applied Research Lab

Department of Computer Science

Washington University in St.Louis

Abstract

In this paper, we propose a new framework called Active Network Management and Control (ANMAC) for the management and control of high speed networks. The software architecture in ANMAC allows routers to execute dynamically loadable kernel plug-in modules which perform diagnostic functions for network management. ANMAC uses mobile probe packets to perform efficient resource reservation (using our novel reservation scheme), facilitate feedback-based congestion control, and to provide “distributed debugging” of complex anomalous network behaviour. ANMAC also provides security measures against IP spoofing, and other security attacks. The network manager has the flexibility to install custom scripts in routers for tracking down anomalous network faults.

Key words: Active Networks, Network management, QoS, congestion control, router plug-ins, resource reservation

1 Introduction

Technological advances in the network infrastructure (ATM, IPng) have led to the development of high performance local and wide area networks. Although advanced networking technology has increased the range of potential applications to include video conferencing, remote collaboration, and metacomputing,
it has also increased the potential for chaos. For example, most of us have been on the receiving end of a network-wide denial-of-service attack that could have been initiated from some anonymous, distant host. Even more prevalent is the helpless feeling of waiting for a web page without knowing the reason for the delay. Although network management alone will not solve these problems, these are two examples of problems that may be examined through network management functions and in some cases, corrected. Network management becomes even more critical as user expectations and network usage rise with the increasing availability of high bandwidth, QoS based network technology.

As networking technology becomes more complex, there will be a need for management tools that can utilize dynamic functionality in the network itself. Complexity in networking systems naturally leads to an increased level of errors in the system implementation itself. Furthermore, the growth in the scale and number of networks present additional problems. Network growth can make manual inspection impractical and increases the chance for strange network behavior caused by misconfigured devices. It is the unpredictable, anomalous behavior that is the most difficult to identify, characterize, and repair. For example, the BGP route flapping problem [15] has plagued the Internet for years and is still not completely understood. Identifying and resolving unpredictable scenarios is often akin to program debugging because it is targeted at the rare cases.

In this paper, we explore the use of active networking ideas in managing networks. In particular, we examine how the addition of dynamic functionality to network devices (specifically routers) can aid the network manager and how these same features can be leveraged to provide value-added network services to the user in the form of service setup, inspection and verification. We describe a novel architectural framework, the Active Network Management and Control Architecture (ANMAC). ANMAC uses the concept of dynamically loadable plug-ins [6] to install modules at active routers. The active routers incorporate functionality to perform congestion control, packet shaping/policing, network reconfiguration, packet filtering, admission control and a variety of management functions. We also propose novel mechanisms for provisioning network resources.
The rest of this paper is organised as follows. Section 2 describes traditional approaches in network management and discusses the need for an active networking approach in network management. It highlights some of the typical problems with current network management tools and describes the active networking features that can be exploited in network management and control. Section 3 presents ANMAC, our active network framework for network management. We then describe the important plug-in modules that are used in ANMAC for providing dynamic functionality in Section 4. Section 5 presents scenarios that illustrate the working of the ANMAC, while Section 6 describes internal details for our implementation environment. Finally we make some concluding remarks in Section 7.

2 Motivation and Related Work

Network management functions fall into two broad categories: 1) monitoring (determining the operating point\(^1\)), and 2) control (moving the operating point). The purpose of these functions is to identify performance bottlenecks and unusual behavior and handle faults. Monitoring involves observing events and identifying those events that indicate important phenomena to the manager. Inspecting configuration settings can be viewed as a degenerate case of monitoring in which the target is a static variable which should have a constant value. Typically, some low level, continuous, structured monitoring is used throughout a network to establish a temporal baseline (defined as normal behavior) and to watch for gross bottlenecks. Some of this data is often logged in a history database for subsequent analysis and reporting.

Traditional approaches to network management are built on simple approaches such as SNMP and RMON that are built bottom-up [17]. SNMP is in wide use due to its operational simplicity requiring very little prior knowledge to begin gathering interesting information. An SNMP managed device has a set of status and control variables structured in a MIB (Management Information Base) and encapsulated in a MIB agent which resides on the device. SNMP operations consist of MIB variable inspection (GET) and update (SET) and event notification (TRAP). Even the initiation of an SNMP action is performed

---

\(^1\)A baseline reference for stable network behaviour
by setting a MIB variable. SNMP is used to manage nodes (e.g., network devices) and has no concept of a network of nodes (e.g., LAN). However, user-written scripts of SNMP operations which poll multiple devices can give the illusion of a LAN.

RMON gives a network-wide view and is concerned mainly with dealing with data from network probes (e.g., LAN analyzers). Thus, it complements SNMP which is concerned more with individual devices than the network as a whole. RMON is based on the same paradigm as SNMP: 1) the manager has a relatively static set of management functions; and 2) control is centralized at the NOC (Network Operations Center). The manager sits at a console and issues primitive operations in the hope of finding a reason for poor network performance. A script might be written to aid the manager, but again, the basic control structure is a master-slave one with communication to devices as slaves and from the NOC as master. Coordination, especially real-time coordination, is difficult in this approach, and MIB agents have no autonomy to communicate with other peer agents. Remote proxy agents add some degree of coordination flexibility by acting on behalf of the NOC by coordinating several devices but is more a mechanism for off-loading work from the NOC or providing functionality when the device doesn't have the resources to house its own agent. These agents are statically configured when they are created and cannot be dynamically altered on-the-fly.

Commercial tools which are based on this traditional approach (e.g., HP OpenView [7] and IBM NetView [8]) use status-based polling mechanisms to detect network failures and performance polling which poll the network for performance parameters (e.g., link throughput). As the number and the complexity of nodes increase, management stations become sinks for huge amounts of redundant information and this solution does not scale. Another major problem with polling is that a component can suffer several state changes in less than a round trip time and can even fluctuate on a per-packet basis [21]. In both cases, the result is higher detection and monitoring latency. These commercial tools tend to be more oriented toward managing discrete entities than the global network state. In general, we see that the traditional approach is static, uses a centralised approach that could lead to severe scalability problems, and avoids
direct coordination between distributed entities that should interact.

Active Networking (AN) is an emerging area of research that enables the entire network to be a fully programmable computational environment. Traditionally, there have been two approaches for code injection in active networks: capsules and programmable switches. Capsules [18] are miniature programs that are transmitted in-band and executed (through interpretation rather than direct execution) at each node along the capsule’s path. The result of the computation can determine what happens to subsequent packets. Programmable switches [9] are programmed out-of-band by the network manager by injecting code into the network.

While the former approach is flexible in terms of adapting to the user requirements, the code fragments are limited in size and therefore their functionality. Also, virtual machines restrict the address space that a capsule can access for security purposes limiting the applicability of the capsule. For example in [18], the code that is contained in the capsule is of a size less than a kilobyte. Furthermore, security constraints degrade the performance further by requiring complex authentication mechanisms. The programmable switch approach is static and requires the manual configuration of the router by the network manager. As explained in a subsequent section, we propose to use a hybrid approach that brings to network management increased efficiency, scalability, and flexibility.

Scalability: Typical solutions to scaling the network are to provide more management stations. However, if there is an increase in the number of users, then adding more stations does not suffice. There needs to be some way of dynamically supporting an increase in traffic. As the number and complexity of nodes increase, management stations become points of implosion and can face large amounts of redundant information. A scalable solution would be to tailor the amount of information that is returned to the management center. This is done by a two-level hierarchical event gathering mechanism that allows the routers themselves to perform low-level management of their local devices, while providing specific non-redundant information to the NOC. In ANMAC, flow-specific state information (to specify QoS) stored at routers are aggregated using generalized filters [11] preventing state explosion at the routers.
Performance: There are several potential performance enhancements that can be realised in ANMAC. These can be categorised into pipelined and parallelised operations. For example, if an application requires a complex operation to be performed, then we can fragment this operation across different active routers in the QoS route chosen for the packets of the application. Each router along the path does a part of the computation in a pipelined fashion for all packets of the flow. The other possible enhancement is one of parallelism. If packets belonging to different flows require different operations to be executed at the router, then the router can handle several flows simultaneously given sufficient processing power. This requires efficient matching of the packet to the required function.

Dynamic functionality: Many opportunities in network management can be derived from the dynamic functionality offered by ANMAC. Some examples are:

- **Fine-grained monitoring and QoS down to the flow level**: [2] has already shown in their router plug-in software framework that packets can be efficiently classified and steered toward dynamically installed functions. This same framework has been extended to form an AN node [1] that can be used for fine grain monitoring when coupled with our high-performance hardware monitor [5].

  The network manager has the capability to remotely install packet filters in routers to allow specialized treatment to flows depending on their QoS requirements, which is crucial during congestion when the minimum QoS must be guaranteed to flows. This aspect has an interesting parallel in Differentiated services (DiffServ) where flow requirements are mapped to per-hop-behaviors in routers. Essentially, a per-hop-behaviour can be mapped to a special packet filter in ANMAC. The NOC can coordinate with the edge-routers in DiffServ and install packet specifiers for aggregated flows that would provide the QoS. Thus, ANMAC is easily extensible to future technologies such as DiffServ.

- **Hot-spot probing with mobile probes**: Mobile probes are special control packets can be sent by a network manager to probe congestion points. The mobile probes trigger the download and execution of code that can facilitate detection of network anomalies. The probes are also used to reserve resources on links belonging to QoS routes. The probes are also used to find new QoS
routes when the network experiences congestion and QoS renegotiation is performed. The active routers themselves initiate the probes to reserve resources in a departure from traditional resource allocation. We propose a novel resource reservation protocol called Deferred Reservation (DRES) for ANMAC, that uses mobile probes for resource provisioning.

- **Handling security attacks:** In active networks that use the capsule approach, security is a major issue. How does the router trust that the code sent by the user is authentic and non-malicious? Thus additional authentication and verification mechanisms are required. However, in ANMAC, this is avoided by using the NOC to send control information on behalf of the users. Only the NOC can install code at the routers, preventing any malicious user from disrupting the network. Actually, either the network manager can install custom scripts or code that is cached in a code server [2] can be downloaded to the routers. We note that it is also possible for the user to send code for a specific application that cannot be provided by the network. However, this approach will require authentication of third-party code. The NOC will then assign a special code that the user needs to include as an identifier in the packet that will allow the router to recognize that a special application needs to be executed for that packet. An additional feature of ANMAC is the collaboration of active routers to prevent denial-of-service attacks using effective filtering mechanisms. We provide examples later as to how we can use ANMAC to handle security attacks such as IP spoofing.

- **Distributed debugging through stepping:** Each active router can have some special storage where it can store its past actions and state. If a network fault occurs, this information is transmitted to the NOC. The NOC can then obtain a snapshot of the network state up to the fault point from the router. The NOC can now use the logged information to simulate network requests and recreate the network fault in a step-wise manner facilitating diagnosis of difficult network faults. This could be selectively enabled by the NOC by sending a special control packet that triggers monitoring functionality at active routers. The approach taken is to transfer the state information to the NOC as soon as the packet is processed. By reserving a small fraction of the link bandwidth for control
related information, this information can be transmitted back to the NOC for off-line analysis.

- Further, the network manager can install plug-in code on demand and upgrade network protocols in an incremental manner. Active networks are perfectly suited to incrementally deploy protocols such as IPv6 in a seamless manner without interrupting the network service.

Dynamic functionality can be a boon to network management. However, there is an associated overhead with this feature. There is a need to classify the extent of the functionality that one can achieve in routers. Processing power at the routers are constrained by the fact that they need to switch packets at gigabit and possibly terabit speeds. There is a fine line between how much computation can be achieved at the router before the packet is switched, and the actual benefit of the computation. Since ANMAC adopts a distributed management perspective, issues of coordination between different routers and NOCs also need to be considered. However, the key thing to note is that ANMAC is a flexible architectural framework that facilitates incremental deployment. Thus, it is possible to add functionality in a step-wise fashion, while determining if the functionality added is not offset by the introduced overhead. In essence, a network manager can customize and add those features that he/she thinks is most crucial for the network, while avoiding those features that are unnecessary and could potentially be performance bottlenecks. We will now describe the architectural framework of ANMAC.

3 The ANMAC Framework

The ANMAC framework consists of the active network subsystem (active routers), and the network management subsystem that includes components that manage and monitor the network including agents and mobile probes.

3.1 The Active Network subsystem

In ANMAC, active routers offer different services depending on the plug-ins that are locally available. Plug-ins are downloaded from centralised components called code caches [2] on demand. It is also possible
Figure 1: Network Topology

for the network manager to create custom code or scripts and install them in the active router. Routers communicate with end-systems (users) using signaling for resource provisioning. Users send packets with headers containing function identifiers which are indirect references to plug-in code modules at the router as well as input parameters.

The overall topology resembles Figure 1. There are two types of packets that are transmitted in ANMAC. The first kind of packet is a control header packet that sets up state information at the routers for the rest of the application packets. In the event that the router has the code stored locally, it will execute the code on all packets that subsequently follow the header packet. If necessary, the control packet could result in parallel downloads on all links of the route chosen for the data packets. The idea here is to do a prefetch of the requisite code in advance to reduce the latency for downloading. The other approach is one where every packet has an identifier, and state need not be reserved in routers for the application. Applications that use this approach may need a variety of functions to be applied and packets may take different routes depending on the plug-ins stored at the routers. The end-user could use the global knowledge of the network manager to query which routers have the necessary plug-ins. Since this
could be a performance bottleneck, an alternate mechanism is to allow querying of the services offered by routers using anycasting [23] and finding a QoS route that enables these services. However, there is a tradeoff in the two approaches between minimizing state information at the routers (scalability), versus the longer set-up latencies in finding appropriate routes.

We note that routers are required to maintain minimal state information of the flows that are passing through them. This is required for several reasons including admission control and for providing information about flows to the NOC. Since maintaining information for each flow will lead to state explosion at the routers, we propose the use of generalized packet filters [11] to store state at routers. These filters can be wildcarded to arbitrary degrees and a single filter can describe several application level flows. This state aggregation prevents the explosion of state caused by storing per-flow state information at the router. Thus, technologies like DiffServ that use flow aggregation can be supported by ANMAC.

We further propose an additional performance enhancement in ANMAC. Active routers possess the ability to transfer a plug-in module to other routers that do not have the module, allowing some form of load sharing. More importantly, this can have important consequences in resource allocation, when no existing QoS route is found initially, but on plug-in migration, routers adapt themselves (by acquiring plug-ins) to satisfy QoS requirements.

3.2 The Network Management subsystem

Network Operations Center: One of the key components is the Network Operations Center (NOC) which provides an interface to the network manager to comprehensively view the current network state and “debug” network faults. It transmits control packets for downloading new or updated plug-ins at the active routers. The NOC can remotely install filters at routers that modify the services that are provided to flows requiring QoS. The NOC can also upgrade protocols in an incremental fashion. It is possible to implement both the IPv6 and IPv4 protocol stacks on the same active router. Incremental deployment of IPv6 can be done by enabling a special filter on certain routers on some static route. Using filters and
device driver modifications, a packet will be processed by the appropriate protocol stack. The NOC is also responsible for efficiently correlating events to derive the correct cause of a network fault. It also provides a graphical display of the current network state as well as the possible faults in the network at multiple levels (such as IP, ATM, Transport). The NOC is graphically shown in Figure 2.

**Mobile Probe:** The next component is the mobile probe packet that can be sent by both the active routers and the NOC. This packet is a special control packet that is utilized for the following purposes.

*Resource reservation:* The probe can perform resource reservation. Essentially, routers send the probe for resource reservation when admitting a new flow. We propose to use DRES, a novel resource reservation protocol, for this purpose.

*Handling alarms:* The probe packet could be sent to congested network areas by the NOC. Once a router reports back to the NOC using signaling, the probe could be sent simultaneously by the router to find secondary routes. Also, on those links where data collection is required, the routers could send the probe to selectively gather data.

*Host administration:* The NOC can send a probe directly to the end-user to collect data. In a corporate
network, the NOC may have complete authority and could probe for information in any host for security reasons. However, this could be exploited to collect data from the end-users perspective. This could be useful in cases when a fault occurs in the end-users application program. Since the problem is with the end-user and not the network, the NOC can verify this fact by sending a probe to the end-user. A further use of this feature is to prevent the sender from transmitting malicious flows. If the sender begins to transmit at rates that have not been negotiated, the NOC could send the probe to throttle the senders transmission rate.

**Distributed Monitoring Entity:** The third key component is the set of Distributed Monitoring Entities (DME). The DMEs are responsible for actual monitoring of the links and collection of data, which can then be used for efficient computation of the network management parameters such as the overbooking ratio, and other QoS statistics such as ATM connection blocking rates, detecting malicious flows, etc. We use efficient hardware probes using the ATM Port Interconnect Controller (APIC) [5] chip for this purpose. Each chip can support one full duplex 1.2 Gbps ATM port. The APIC snoops on ATM connections logging only the frame/cell header information and other monitoring information like cell counts without the necessity of bringing the entire frame into memory. This leads to a significant saving on memory bandwidth. The NOC can install remote filters in the DME interface via the router that allows specific information to be logged or monitored.

**Packet format:** Each packet is enclosed in the ANEP header [22], though they could have their own IPv4 or IPv6 headers. In [2], control packets had a *function identifier* ($f_i$) to denote the function that needs to be executed on the data packets for the flow. In ANMAC we propose *context dependent function identifiers* ($cdf_i$) that would enable the execution of a plug-in depending on some predicate or context. The rest of the header stores QoS parameters apart from the source and the destination of the flow. This is required in order to police or reroute flows. For example, *plug-in location* could be one such predicate. In some cases, it may be necessary to enable plug-in execution only at certain locations in the network. Consider encryption algorithms that encrypt data streams when they leave a corporate
network and go on the internet. Obviously we need to perform the encryption only at the last hop and not at each router along the route. Thus, the $<cdfi>$ with a location predicate would enable the selective execution of the encryption plug-in at the right location. A further enhancement is to carry a sequence of $(<cdfi>, data)$ tuples in a nested manner. Thus, when one $<cdfi>$ is referenced and the code is executed at one router, this layer is stripped, and the next router will then reference the $<cdfi>$ that is nested one level within and so on. This approach allows pipelining the functions to be executed on a data stream improving the performance. The packet format is as shown in Figure 3.

Another important predicate we propose to use is the dependence predicate. With this predicate, an active router could store context information for transferring execution state during fragmented execution of data. Also, the router could write dependency related information in the predicate which allows parallel fragmented execution of code on different routers to be correctly integrated at the end of execution using dependency information.

4 The Plug-in Modules

As mentioned earlier, the dynamic load feature of the plug-in modules can be exploited to perform network management and control. The plug-in components of ANMAC that we plan to implement are given below.

4.1 Feedback-Congestion Control, Traffic shaping and Network Reconfiguration

Congestion is an application independent event and occurs within the network. This makes it suitable for active networks, especially since the time required for congestion notification information to propagate back to the sender limits the speed at which an application can decrease its sending rate or ramp up
depending on the current state. Thus, the approach is to use the active router and the NOC to alleviate congestion. The feedback component is activated as soon as the NOC receives an alarm message from the DMEs of the specific router indicating queue thresholds have been exceeded. This control packet indicates several parameters such as the location of congestion and flow-specific information such as the current output transmission rates for flows, and other management related information.

The NOC has the flexibility to respond to the alarm in several ways. It could send a special feedback message directly to the source of the flow using the information contained in the alarm message suggesting that the rate of the flow be reduced or change network control parameters such as the overbooking ratio. Alternately, routers could maintain a congestion profile which is calculated by the NOC using past history, and required QoS for a particular flow. The congestion control algorithm (for example: RED) in the router would use this profile for deciding which packets to drop/shape. A third option that we plan to implement is to allow the NOC to remotely install filters in the routers to reroute packets on alternate QoS routes. These alternate routes are calculated by a QoS routing plug-in that is activated at routers when they are informed about congestion.

With ANMAC, it is possible for the NOC to update the network state and reconfigure if necessary by comparing it with a stored plan. This can be done by installing appropriate filters to reroute traffic as per the target network plan. This module is also activated in cases when plug-in modules need to be updated, or there is a network protocol upgrade. Also, it may be necessary for the NOC to shape flows. Normally routers do the shaping and policing of flows. However, the NOC could override the router shaping module in cases when flows misbehave and are tracked down by the DME. In this case, the NOC could send a &lt; cdfi &gt; that targets the malicious flow by installing a special filter at the router which would enforce shaping to penalize the malicious flow.
4.2 Distributed Monitoring Entity Handler

This module collects the various data from the distributed DMEs and sends the data in a coherent coordinated fashion. This plug-in essentially polls the DME interface at periodic intervals and gathers data. The NOC can query the plug-in for information. The polling interval is quite important since a larger interval would mean more stable data whereas a shorter interval would be more sensitive to critical events. Critical events that are sporadic and could occur at any time are better captured using trap-directed polling. ANMAC uses a combination of status-based polling and trap-based alarms. Thus, this approach does not suffer from the problems of current network management systems which do status-based polling and consequently are insensitive to the critical events.

The link monitoring capabilities of the APIC are exploited for collecting data that could be used to predict congestion, as well as to find new QoS routes. ATM Cell counts can be used by the NOC to determine the state of a link and also to check the flow itself. This aids in detecting malicious flows that exceed their negotiated QoS. In certain environments, IP is implemented over ATM [12]. Permanent virtual circuits with a fixed capacity are established to carry IP data. For effective IP level resource management, peak rate allocation can be performed for the PVCs and one can isolate these using priority mechanisms in ATM. Since the allocated bandwidth can be dynamically varied (due to the ATM base), there is a need for measuring traffic and congestion conditions in the PVCs. At the IP level, the network probe can measure average PVC utilization, queue lengths, etc., while at the ATM level, PVC cell counts can be determined. The NOC can then vary the bandwidth allocated to the PVCs using the information from the probes (DME).

The NOC can remotely install a special packet filter in the DME interface for handling links that do not have a network probe directly attached. Thus, link monitoring data can be obtained from another link that has a probe or by probing the router itself. Traditional monitoring techniques could also be performed by executing applications such as traceroute to determine the response times of the network. These applications can be triggered at specific routers by the network manager by sending relevant control
packets. Another mechanism is to monitor Internet Control Message Protocol (ICMP) source quench messages. IP specifies a control message called source quench that requests a host to transmit more slowly on a particular connection to avoid congestion. The routers could be programmed (by remotely installing a special filter) to track these messages to detect impending congestion.

4.3 Resource Provisioning and QoS

In ANMAC, admission control is performed by active routers rather than any centralised arbitration logic. When a user requests a connection with a specified QoS, the nearest active router will then use the admission control plug-in to decide whether a new connection can be accepted or denied. Though, admission control is not a network management prerogative, we propose to exploit the active routers to perform resource management. Since ANMAC couples both the management and control aspects, it is easier for a network manager to modify network control parameters so as to support QoS. We propose the use of a novel resource reservation protocol in ANMAC.

Control parameters such as the Overbooking ratio as in ATM are used for admission control. The appropriate choice of the overbooking ratio is crucial. On high bandwidth links, a higher overbooking ratio would exploit the inherent statistical multiplexing advantage since it allows higher utilization. At the same time, a lower overbooking ratio could be chosen to deal with congestion. A higher value can be used when flows are flexible in their QoS requirements even during congestion, allowing more flows to be admitted to reduce blocking (more optimistic admission policy). There is a tradeoff between increased admissibility of flows versus the increased chance of congestion.

Different QoS parameters can be used such as (Bandwidth, delay, delay jitter, loss probability) in order to characterize a flow. In this context, we classify the traffic in ANMAC into three broad classes. The Real-Time (RT) class provides support for flows requiring fixed delay bounds. The Statistical Bandwidth (SB) class includes flows of delay-tolerant applications requiring a minimum bandwidth. The Best Effort (BE) class provides best effort service similar to the current Internet. Depending on the class of service,
the admission control module uses different admission tests for accepting or rejecting a connection. We will now describe our novel resource reservation protocol.

Deferred Reservation: DRES

We propose a simple scheme for resource reservation in ANMAC. As mentioned earlier, users can set up connections by communicating with active routers using signaling. The routers perform admission control and reserve resources for the users using the Deferred Reservation (DRES) protocol. DRES is a soft state, 2-phase resource reservation protocol. By soft state, we mean that the reservation state installed in the network has to be refreshed periodically by the receiver otherwise the state times out and is deleted. This protocol is a hybrid variation of RSVP and increases the admissibility of flows to the network. We note that DRES functionality can be added with suitable modifications to an existing protocol such as RSVP or ATM connection admission control.

The key idea of DRES is to use deferring (delaying) as a mechanism to accommodate flows that can tolerate initial set-up delays. When a user requests a new connection, the router first checks if the link capacity has been used up. Now there are three aspects to the link capacity. The first is the bandwidth that has been committed\(^2\) to flows, the bandwidth that has been tentatively reserved for non-committed flows, and finally the unused bandwidth. It is the second fraction that we are focusing on. Also, we are targeting applications that do not require tight delay bounds during initialization, i.e., we are considering application that belong to either the statistical bandwidth or the best effort classes. However, even real-time applications that can tolerate initial setup delays can use DRES. We note that we can bound this setup delay as shown subsequently.

If the available bandwidth is not enough to accommodate the flow, the request is not rejected. Instead this request is stored in a queue maintained in the router for deferred requests. The request is also forwarded to parallelize this operation at all routers on the QoS route. The point we are making here is that if there are some existing flows that have reserved non-committed or even committed bandwidth and

\(^2\)This means that the connection is established by DRES
are rejected at some point downstream due to reasons such as congestion, tearing down flows or rejected real-time flows, then we can free up resources and reallocate them to the deferred flows. We use a set of timers to prevent deadlocks from occurring by flushing deferred flows after a timeout period.

**DRES Algorithm**

We will now describe the admission control algorithm of DRES. We first define the following terms for the admission control process.

\( L_{tot} \): Total Link capacity;

\( L_{co} \): Committed link capacity;

\( L_{nco} \): Non committed but reserved link capacity;

\( L_{free} \): Free link capacity which is \( L_{tot} - (L_{nco} + L_{co}) \);

\( B_i \): Bandwidth requested by the user \( i \);

\( q_{defer} \): Queue storing the deferred requests.

The actual algorithm is given as follows:

1. A new user request arrives at a router.
2. Perform admission check on the bandwidth availability: \( B_i \leq L_{free} \rightarrow (1) \). Can also use overbooking ratio.
3. If the check fails,
   a. Enqueue the user request in \( q_{defer} \)
   b. Start timer \( t_{defer} \): Forward the user request to the next hop. It ultimately reaches all routers on the QoS route.
4. If the check succeeds, set up state on the link and update the parameters, \( L_{co}, L_{nco}, L_{free} \). Start a new time \( t_{ackwait} \) that is used to track a positive acknowledgement from the next hop router. If the router is the last hop router, then it sends a positive acknowledgement \( P_{ack} \) to the previous router.
5. If \( t_{defer} \) expires, dequeue appropriate entry in \( q_{defer} \) and send a negative acknowledgement \( (N_{ack}) \) to user.
   If \( t_{ackwait} \) expires then the state setup for a flow is deleted, the parameters \( L_{co}, L_{nco}, L_{free} \) are updated and a \( N_{ack} \) is sent to the previous hop.
6. If router receives a \( N_{ack} \) for a flow \( j \), then
   a. Free resources allocated to the flow. Update the parameters, \( L_{co}, L_{nco}, L_{free} \).
b. Perform the admission check (Eqn. (1)) and reserve resources accordingly for the first flow in \( q_{defer} \).

If the flow is a deferred flow, then dequeue from \( q_{defer} \) and forward the \( N_{ack} \) to the previous hop.

7. If the router receives a positive acknowledgment (\( P_{ack} \)), update the status of \( L_{co} \) and forward \( P_{ack} \) to previous hop router. However, if the state has been deleted for that flow (due to expiry of timer \( t_{ackwait} \) then, DRES requires the router to send a \( N_{ack} \) to the previous hop router.

8. If the router receives a connection.tear.down message, free resources; update the parameters, \( L_{co}, L_{sync}, L_{free} \). Attempt to reserve resources for deferred flows.

The important point to note is there are two timers in DRES. While one keeps track of the waiting time in the queue, the other keeps track of the time that a partially admitted flow can block the resources at a node until it receives an acknowledgement from the next hop. Thus, we are able to bound the maximum time \( (t_{defer} + t_{ackwait}) \) that a flow may need to wait before being admitted or rejected. Also, the protocol essentially requires two messages (one to setup state/stored in deferred queue, and the other from the next hop router to the previous router to indicate that the flow has been allocated resources from the penultimate router of the QoS route to that next hop router.

The primary contributions of DRES are: improved admissibility, overhead comparable to RSVP/ATM, and low latency (response time) by deferring in parallel across all routers. Admissibility is improved by deferring flows that would have been otherwise rejected by standard admission control procedures.

4.4 Event filtering and Correlation

There are several possible events that require the routers/NOC to take appropriate actions leading to a need for event classification. Associated with classification, correlation is an important aspect since a network fault could trigger many simultaneous events that could cause an event storm at the NOC. Thus, correlation and filtering allows a coherent view of the network events. We propose a simple correlation model in ANMAC in which there are two kinds of correlators. We propose to use event ordering algorithms as in [16]. A lightweight correlator engine is located at the active routers and allows minimal correlation to be performed by the routers themselves. For example, failure of a link may mean that DMEs on that link would trigger multiple alarms due to the several devices reporting the link failure. The router can
correlate these numerous events to the link failure instead of unnecessarily sending multiple alarms to the NOC. This reduces the complexity of filtering at the NOC by hierarchical isolation. The heavy-weight version of the correlator engine is located at the NOCs. This deals with more complex events that could potentially affect several routers in a short span of time such as a denial-of-service attack.

4.5 Security

Traditional security concerns in active networks are avoided by the NOC installing plug-in code in routers. Thus, we will concentrate on denial-of-service attacks that occur when IP address spoofing is done or unauthorised users break into the system. This would require sporadic event handlers that are triggered on such events. The NOC must periodically scan the traffic for possible signs of break-ins or other kinds of attacks. It is feasible to maintain some signature database and compare the packet contents to the signature. The next aspect is to track down the source and either modify the access control permissions to completely block the source or send a warning message. For example, IP spoofing can be handled using router collaboration. We consider a specialised attack called TCP Syn-Ack flooding [13]. Once the router determines that spoofing is being done, it could inform the NOC which can remotely install a filter in its DME interface. This will trigger the remote installation of this filter in the next hop router, so that the neighbouring router can deal with the spoofed or forged packet in a similar fashion. This could be extended to all routers in the path of the forged packet. Thus, subsequent attacks will fail due to this collaborative filtering process.

5 Illustrative Examples

We will now proceed to examine a situation that can occur requiring the network manager to use the active network in the manner described earlier. The semantics of the various components of the active router and NOC will be made clear using these examples. For all the example scenarios, we will consider a network topology as in Figure 4. All links in the figure have a capacity of 10 Mbps. Let us assume that there are
four sources, S1, S2, S3, and S4, and two receivers R1 and R2. We assume that bandwidth is the only QoS required by the flows which could be multimedia streams. Note that the focus of this example is not on how the active network could manipulate the data but more on how the NOC could manage the network using the active routers. Thus, we are not concerned with what plug-ins the routers execute on the data. For completeness, we assume that the active network is functioning as a media gateway [20] that performs transcoding on the data.

Let the flows be characterized as follows (source,QoS,destination): (S1,2 Mbps,R1); (S2,4 Mbps,R1); (S3,4 Mbps,R2); (S4,1 Mbps,R2).

5.1 Scenario 1: Congestion control and Link Monitoring

When the flows are first admitted, the admission control plug-in will use the link state information to determine a QoS route for the flow. We assume that there is a load balancing facility in the QoS routing policy. Therefore, the router AR1 communicates with the senders using signaling. AR1 then sends probes that attempt to reserve resources for each flow. We assume the flows require a peak bandwidth as mentioned
earlier. Consequently, the flows could be categorized into the statistical bandwidth class of service. Thus, the probes reserve resources at each router on the computed QoS route. Since the NOC periodically probes for the current network state, the admission of the three flows will be tracked by the NOC.

Without loss of generality, we assume that S1 and S2 will be sent through the route (AR1,AR2,AR4), while S3 and S4 will go through the route ((AR1,AR3,AR4). Let us further assume that the network management parameter of overbooking ratio is 0.8. As mentioned earlier, the overbooking ratio decides the importance of reduced connection blocking versus a lower probability for congestion. The low value for the overbooking ratio indicates that the network manager is more concerned about the occurrence of congestion. Since the applications require a transcoding plug-in at the router, the NOC uses its programmable interface to install the transcoding plug-in module at the routers. This is done by sending a control packet with a $<fi>$ that allows the code to be installed in the router. It may not be necessary to implement this at all the active routers.

The DME plug-in is activated at all routers. Note that links between (AR1,AR3) and (AR2,AR4) do not have probes. These links therefore depend on the installation of a special filter in the DME interface that allows the control packets requiring link information to obtain information about flows directly from the router itself (AR2) for (AR2,AR4), or obtain it from a combination of both router (AR1), and downstream probe (between routers AR3 and AR4) for (AR1,AR3). The NOC is responsible for ensuring that the complete network state information is captured, and hence must send control packets to the router to install special filters if necessary. This is done for routers AR1 and AR2. Thus, the network probes collect information about the output transmission rates, cell counts (in case of an ATM network) and other relevant data.

Now let us suppose that the link between AR1 and AR2 goes down. The DME plug-in will then send an alarm message to the NOC. Actually, it is possible that several alarms may be generated which could result in several events dispatched to the NOC. The event correlator module filters all irrelevant and redundant events to provide the correct picture to the NOC. The display manager would then graphically present
the problem. The network manager now has to take action using the feedback module. At this point, the manager decides to reroute the affected flows, namely those from S1 and S2. Thus, the manager can now send probe packets to reserve resources on the other route (AR1,AR3,AR4) or delegate this responsibility to the active router.

Note that the links do not have sufficient capacity to simultaneously accommodate all flows (11 Mbps) on the 10 Mbps link. When the flows are rerouted, all flows will now go through the route (AR1,AR3,AR4). This will lead to packet loss and congestion. Once again, one or multiple alarms may be sent to the NOC indicating these events. At this stage, there are multiple options for the NOC. Depending on the nature of the flows, the network manager can increase the overbooking ratio to admit all the flows. If the network manager changes the value to 1.1, all four flows will be admitted on the link. The assumption here is that the four flows transmit at different times, and if there is no overlap, all four can send data on the links. Exploiting statistical multiplexing allows more admissibility at the expense of an increasing probability of congestion.

Another orthogonal alternative is to instruct the routers to perform traffic shaping to smooth out bursty flows. It may be necessary for the NOC to send a control packet with appropriate pacing parameters that the router can use in the shaping plug-in. However, this may not work with statistical multiplexing. A third alternative is to perform selective packet discard using a congestion profile or parameters such as cell loss probability. A fourth alternative which will not always work is to direct the routers to buffer packets assuming that the overload is transient. In the above example, this will not succeed. In addition, buffering introduces delays that cannot be tolerated by real-time flows. A final alternative is to re-admit the flows. Here the router informs the user (signaling) that the flow needs to be renegotiated at a different QoS.

5.2 Scenario 2: Effect of Malicious flows

Another concern in a network is when flows begin to misbehave and transmit at rates that are not negotiated in the QoS. Let us assume that S1 begins to transmit at 20 Mbps instead of 2 Mbps. This will be detected
by the DME plug-in and an alarm will be sent to the NOC. If we used the traffic shaping plug-in feature of the router, this is automatically handled since the excess is dropped using token bucket filters. However, if no policing was performed, the router could be directed by the NOC to simply drop the flow and send a warning control message to the sender requesting a renegotiation of the QoS. The preferred alternative would be the remote installation of a special packet filter by the NOC at the router that basically gives only the requisite 2 Mbps bandwidth to the flow and drops the remainder. This provides flow isolation from misbehaving flows.

6 Implementation Environment

We plan to build a high performance software framework for ANMAC. The APIC chip will be incorporated into a line card for link monitoring. The software structure of the active router will be implemented using the following key components as shown in Figure 5. We make a distinction between control packet path and module control. By the former, we refer to the possible path of flow for control traffic that enters an active router while the control lines from one module to another indicate that one controls the other. We also provide a separate data flow path for packets that do not require active processing from the routers. These packets will bypass the active path as shown. The explanation of the components are as follows:

Plug-in loader: which loads and installs a specific plug-in on receipt of a control packet from the NOC. It can also be activated using signaling by another router for transferring plug-in code and for fragmented execution. The plug-in interface will be implemented using a similar approach as in [6].

Priority assignment: module which assigns priorities to plug-ins dynamically (under the control of the NOC) and also arbitrates between plug-ins for sharing the processor cycles. Some plug-in modules can and should be prioritized over others.

Packet classifier: that classifies incoming flows into the three classes (Real-time, statistical bandwidth and best effort) that were initially described in case of data flows, otherwise the packets are control packets.

Admission control module: determines the current resource availability and decides whether a flow can
be admitted or rejected. DRES will be used to reserve resources, depending on the admission policies for each traffic class.

**Signaling module:** uses DRES to communicate with the end-users and the NOC as well as other routers.

**Function processor:** that processes the function identifier (<code>cdfi</code>) in each header packet. It then forwards the data packets to the corresponding plug-in module in order to perform the required operation. It records all its actions in a log file that is stored in a centralised cache. This information is useful for network debugging.

**DME processor:** which handles the data from the link monitoring devices or network probes (APIC) and transmits them to the NOC for obtaining an accurate picture of the current network state.

**Management Agent Module:** deals with congestion control, traffic shaping, remote filter installation, and network state updates and is activated by the appropriate management control packet sent by the NOC or directly by the router. It also controls the operation of the admission control, plug-in modules and the DME processor. It also does basic correlation of events before transferring them to the NOC.
The key components of the Network Operations Center are described as follows:

**Event Handler module:** filters, orders and correlates the events to provide a semantically correct view of the state of the network.

**Display module:** provides a consistent view of the current network at different levels (IP, ATM).

**Feedback module:** takes the feedback action as decided by the network manager including resource provisioning, sending mobile probes and control packets.

**State debugger module:** allows the network manager to perform step debugging using the state information that is stored at the routers.

**Programmable interface:** allows the network manager to update or revise the plug-in modules, upgrade protocols, change admission control policies, install remote filters, and inject new shaping mechanisms at the router.

7 Conclusions

In this paper, we have proposed a new framework for performing network management using active networks. We show that by providing routers with dynamic functionality, this allows a customizable interface that allows monitoring and management of the network at any level of granularity. We have described several dynamically loadable plug-in modules that tackle congestion, provide support for QoS traffic and correlate network faults. We have also shown the robustness of the framework by implementing mechanisms to prevent security attacks. The plan of implementation of the framework is described. We plan to implement and evaluate ANMAC in a QoS-enabled testbed.

References


