Transparent QoS Support of Network Applications using Netlets

Kalaiarul Dharmalingam and Martin Collier

Research Institute in Networks and Communications
Dublin City University, Dublin
Republic of Ireland
{arul, collierm}@eeng.dcu.ie

Abstract. This paper describes a method for providing QoS support to legacy (non-QoS aware) network applications. This facility allows such applications to request desired QoS levels from QoS supporting networks and thus helps to increase their lifespan and performance levels. We use mobile-agent components called Netlets for this purpose. Netlets are nomadic components that roam in a network providing predefined network services. The solution we propose is not restricted to any particular QoS model or signalling protocol, and thus can readily accommodate emerging networking standards. This approach to QoS support, by decoupling applications from the specifics of network QoS support, which may differ among networks, or evolve over time, should greatly facilitate the introduction of new real-time services.

1 Introduction

The current Internet has been one of the greatest successes of the past century in the field of networking. With the explosive growth in the population of network users, the Internet has been used to support a varied range of applications, workloads and environments. This has generated a need to introduce QoS mechanisms in the Internet. Quality of Service (QoS) [1] refers to the capability of a network to provide priority including dedicated bandwidth, controlled jitter and latency, and improved loss characteristics to selected traffic classes.

IntServ/RSVP, DiffServ, MPLS and Constraint-based routing [2] are some of the major approaches proposed to retrofit the Internet with QoS capabilities. To enable applications to request reservations from QoS provisioned networks, different APIs like the RAPI for RSVP, the QoS API from the Internet2 community [3], and the generic QoS API integrated in WinSock2 from Microsoft [4] have been developed.

In contrast, there exists a large pool of non-QoS aware applications (hereafter referred to as legacy applications) that are unable to exploit and benefit from the QoS support available in networks. The technology to support QoS in networks is not yet fully mature. Thus, developing an application to interact with a specific QoS protocol carries the danger that the application may become obsolete if the QoS protocol is modified or superseded.

The task of providing QoS support to non-QoS aware applications has been studied by other research groups [5–8]. Many of the proposals were based on using middleware

architectures [5, 6] or modifying the underlying operating system for QoS support [7] or to use signalling protocol specific software modules at user nodes [8].

In this paper, we present a novel approach to provide QoS support to legacy and emerging network applications based on mobile agent-components called Netlets [9, 10]. Netlets are nomadic components that roam in a network providing predefined network services. The solution we propose is not restricted to any particular QoS model or signalling protocol. Thus it may be continue to be used even if the QoS support provided by the underlying network changes. Adapting this approach insulates future network applications required to be aware of the specifics of the network support for QoS.

The rest of this paper is organised as follows. In section 2 we begin by describing the environment in which the Netlets operate to provide QoS support. In section 3, we describe the Netlets approach in providing QoS support to legacy applications. We describe the implementation of the prototype model in section 4 and conclude the article in section 5.

2 IntServ Over DiffServ QoS Model

IntServ [11] is a per-flow based QoS framework with dynamic resource reservation. The reservation is accomplished by using a signaling protocol such as RSVP (discussed in section 2.2). End-to-end QoS provisioning in the IntServ network is based on soft state storage [11] along the "network elements" present in a flow's data path. Each network element implements admission control, resource reservation and traffic policing in a per-flow basis, resulting in a dynamic and efficient network resource management.

In the IntServ network model, a flow descriptor is assigned to individual flows in the network. The flow descriptor defines the traffic and QoS characteristics for a specific flow. The flow descriptor consist of a filter specification (FilterSpec - to identify the packets that belong to a specific flow) and a flow specification (FlowSpec includes traffic specification -TSpec, of the application and reservation parameters -RSpec, required by the application to ensure accurate service levels) parameters.

The soft-state storage at every network node along the data path for each flow leads to a poor scalable QoS model when adapted for large network domains. This scalability problem has lead to the poor acceptance of this model as a solution for providing QoS in the Internet.

To address the problems associated with the IntServ model, the differentiated services (DiffServ) [12] architecture has been proposed with scalability as the main goal. In this scheme, the traffic is classified, marked, policed and shaped (if required) by the edges routers, while the core routers only implement the treatment provided for the different classes of traffic. The traffic differentiation inside the network is achieved by the marking of a field (Differentiated Service field or DS field) in the header of IP packets: packets with the same value in the DS field receive the same treatment inside the network. Packets in different flows can have the same value in the DS field, providing the aggregation of traffic. The major problem with the DiffServ model is the lack of signalling mechanisms that would enable end applications to request desired QoS levels from the QoS provisioned network.

IntServ over DiffServ framework [13] provides a scalable end-to-end QoS model for the Internet. This approach is currently one of the most valuable solutions for end-to-end QoS provisioning, since it tries to conjugate benefits of both the IntServ and the DiffServ architectures. This model provides QoS signalling capabilities for resource reservation by end applications and also provides good scalability properties when working in core networks. The reference network which we have used to describe our approach for providing QoS support to legacy applications is based on this QoS model (see Fig. 1).

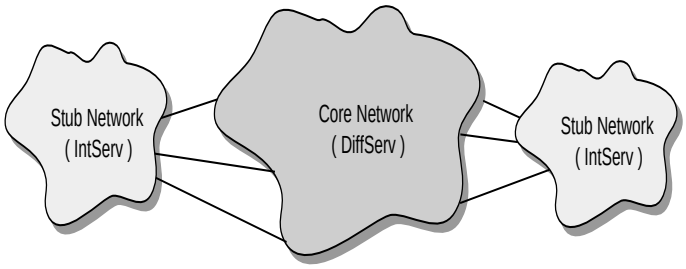

Fig. 1. End-To-End QoS Model

In our approach we use the Netlet services to enable QoS support to end applications operating over such a network environment. Netlets are nomadic mobile agent-components that roam in the network providing predefined network services. We use Netlet Nodes to support operation of the Netlet services. Before we describe the complete solution, we present an overview of the Netlets Network architecture and the QoS signalling protocol (RSVP) used.

2.1 The Netlets Network

The Netlets network architecture is a hybrid approach based on Active Networks [14] and mobile agents [15]. It is an Active Network architecture realised using the mobile agent paradigm. The concept of Active Networks [14] is relatively new, where a network is not just a passive carrier of bits but a more general computational model. A mobile agent [15] is an active program that acts on behalf of a user or another program but under its own control. The difference between conventional mobile code and a mobile agent lies in the inclusion of states i.e. data state and execution state. This state can be transported encapsulated within the mobile agent, whereas conventional mobile code entities are not able to transport state. This feature allows the mobile agent to have the property of autonomous operation. The methodology of mobile agent technology supports encapsulation, program interposition and execution, which make a mobile agent a suitable building block for the construction of Active Networks.

The Netlets network architecture follows the mobile agent paradigm for implementing an Active Network infrastructure and uses the software component technology for building incremental network services [10]. The goal of the Netlets architecture is to

build an open, intelligent, customisable, secured network architecture with autonomous, persistent mobile software components- the Netlets. These Netlets encapsulate service-provisioning code that persist and roam in the network independently, providing network services. Netlet components are exchanged on demand by the Netlet nodes to implement customised network services at required points in the network

The component based model of the Netlets architecture lends itself to dynamically compose tailored network services at runtime in the Netlets network. A custom Netlet network service is created by putting multiple Netlets together using component composition methods [16]. The service composition model adapted in the Netlets architecture is similar to the widely used dataflow programming model.

The Netlet Node offers support for composition and execution of Netlet based network services. The core of a Netlet Node is the Netlet Run-time environment (NRE) which supports execution of the Netlet services. A virtual machine representation is used to allow portability and mobility of service code. A more detailed discussion of the Netlets approach and the architectural overview of the Netlet node can be found in [9, 10].

2.2 Resource Reservation Protocol (RSVP)

The Resource Reservation Protocol (RSVP) [17] was invented as a signalling protocol for applications to reserve resources in the IntServ network. RSVP identifies a communication session based on flows. RSVP is not a routing protocol; it is only used to reserve resources along the existing route set up by the underlying routing protocol. The primary messages used by RSVP are the PATH message which originates from the traffic sender and RESV message which originates from the receiver. The PATH message collects information of the path characteristics along the route to the receiver and also includes the TSpec parameter contents of the application. The RESV messages in turn reserve resources based on PATH message content. A complete discussion of IntServ/RSVP networks has been presented in [18].

3 QoS Support using Netlets

In this section we describe the mechanism based on Netlets to couple legacy network applications with QoS support features. The QoS support to end applications are based on user requests to a manager Netlet node present in the stub network. This manager node processes and coordinate the QoS-support requests from end users and also performs the deployment of Netlet services for QoS signalling. The deployed Netlet services interact with the IntServ based stub network on behalf of end applications to provide an end-to-end QoS support. The details of the complete process involved to enable signalling support are presented below.

It would be highly inefficient to modify the existing end applications or to install QoS signalling specific software at end nodes that require QoS support. Furthermore, such an approach will not be readily realisable and scalable in networks accommodating large user groups. Additionally, developing an application to interact with a specific QoS protocol carries the danger that the application may become obsolete if the QoS

protocol is modified or superseded. Due to this, a remote service invocation method is proposed by the Netlets approach to provide signalling support for end applications.

In order to invoke QoS signalling support from other than the end host running the application, the primary tasks involved are: (a) identify the occurrence of the flow belonging to the session requesting QoS support; and (b) to know the lifetime for which the signalling service has to be in place. Hence flow monitoring and signalling support services are mandatory. In the Netlets approach these services are handled by Netlet components.

3.1 The QoS-Support Manager Netlet Node

The general framework of the Netlets based approach to enable QoS support to legacy applications in the Internet is shown in Fig. 2. The manager node functions to inter-

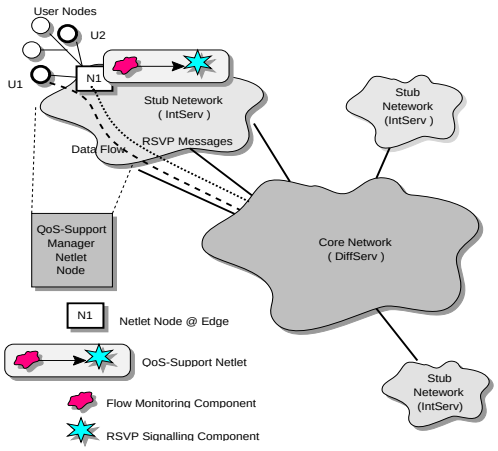

Fig. 2. QoS Support Using Netlets

act with and receive QoS-support requests from end users through a web-based utility. This node is assumed to be a well known node in the network. The QoS-support request generated by the web-based utility on behalf of the session requiring service contains the following information: the sender and destination addresses, the *application type* (data, multimedia, groupware) and its *operational mode* (either as a sender, receiver or both). The information about the *application type* helps the manager node to make an initial estimate of the source's traffic characteristics. In the case of the IntServ network, this information will allow an initial choice of the TSpec parameters. The *operational mode* of the application indicates the signalling features required to support the application's traffic. A prototype version of the web-based utility used in our implementation is shown in Fig. 3.

The manager node contains and deploys (based on user requests) the Netlet components for providing QoS support. On receiving a QoS-support request for an end

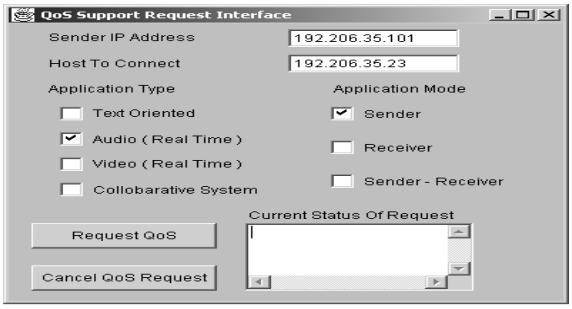

Fig. 3. QoS Support Request Interface

application (for example from U1 as in Fig. 2), the manager node runs a trace-path program to identify the edge node of the stub network to which the requesting end host is attached (Node N1 as is Fig. 2). On identifying the connecting edge node, the manager node deploys a QoS-Support Netlet (discussed in the next section) at that edge node to enable QoS signalling for the requesting end application. Here we assume all the edge nodes present in the stub network are Netlet enabled active nodes. A case for the presence of non-active segments along the stub network edge is discussed in later part of section 4.

The purpose of deploying the Netlet component at the edge node is to identify the occurrence of the flow pertaining to the session requesting signalling support and to provide online traffic modelling of the flow. As in the case of an IntServ based network, traffic modelling allows the accurate calculation of the TSpec parameters. Source traffic in the Internet typically follows a Variable Bit Rate (VBR) pattern and thus continuous traffic estimation of the flow will have to be performed for accurate resource reservation.

A QoS-Support Information Table (QiT) is used at the manager node to manage the Netlet components operating in the stub network. This table records the details of the end users requesting service and the address of the Netlet edge nodes at which the Netlets corresponding to the requests resides to provide QoS-Support.

To avoid multiple copies of the same Netlet being present at a node to serve individual flows, the QoS supporting Netlet can be designed to handle multiple flows simultaneously. For example, if in the case of another user U2 (see Fig. 2), connected to the edge node N1 (N1 already hosts Netlet service for U1) requires QoS support, the manager node requests additional service from the Netlet at N1 (based on information available in QiT) instead of deploying a new Netlet service at N1.

3.2 The QoS-Support Netlet

This Netlet encapsulates flow monitoring and signalling components for QoS service support. On initialisation, the manager node feeds this Netlet with the session details (obtained through the web-utility) of the flow for which the QoS support has to be enabled. This input also includes an initial estimate of the TSpec parameters. This estimate aids the Netlet in starting the reservations immediately and also allows a short

convergence period during which the traffic measurement process of the signalling component gets stabilised. On migration to the edge node to which the end host is attached, this Netlet autonomously starts the monitoring component to identify flows belonging to the session requesting QoS signalling. The flow monitoring is based on the pair of source and destination addresses, communication ports and protocols used.

Based on the flow information collected by the monitor component, the FilterSpec parameter of an application's packet streams can be accurately obtained. The monitor component triggers the signalling component on occurrence of the flow pertaining to the requesting session. The signalling component then uses the FilterSpec information along with the TSpec parameters of the application's flow to reserve resources. Traffic measurement capabilities present in the signalling Netlet component allows to model the application's source traffic pattern online.

The operational support offered by the signalling component is based on the *operational mode* of the application (obtained through the web-based utility). If the application is either sending/receiving packets into the network as in the case of a video server/media player receiving video packets, then the key function of the signalling element is to send PATH/RESV messages respectively before timeout periods to confirm reservations. In the case of an application being both sender and receiver as in the case of video conferencing and collaborative system applications the signalling component performs both reservation requests and actual reservations in the network.

The QoS-Support Netlet is designed to handle multiple flows simultaneously. This reduces the need for multiple Netlet services required to be present at a single node for serving individual flows. The ability of the Netlet to clone and relocate itself to a new node avoids the need for the manager node to host and deploy services in a centralised fashion.

3.3 Advantages of Using Netlet Services

Introducing New Network Services: Introducing new services in the Netlets network is performed dynamically. For example emergence of a new/modified version of the signalling protocol for the IntServ model will only require removing the existing Netlet services and introducing new services which implement that protocol.

Mobility and Autonomy: The service code in the Netlets architecture is mobile and autonomous which avoids manual intervention for service deployment. For example, to introduce signalling support at multiple edge points, the service Netlet component is informed with the address list of edge nodes requiring service. This Netlet then autonomously migrates to each node and installs service thus avoiding centralised deployment schemes and generates less network traffic.

Demand-Driven Population of Network Services: The life of a particular type of Netlet will purely be based on the user demand for that species. On increase of demand for a particular type of Netlets, the required set of species are cloned and dispersed into the network for service deployment. The least used Netlet components would be purged by agent bodies [10] as the demand falls. This leads to a demand-driven population of services in the network.

Migratory Path: The Netlets approach to provide QoS support to legacy applications offers a reliable and flexible approach without actually making the end applications QoS

aware. This approach can be used to migrate from the current application model which either provides restricted or no QoS support to a model which provides QoS support of choice on demand.

Network Intelligence using Netlets: It will be highly advantageous to avoid continuous traffic measurement of an application's flow every time a QoS-support request is issued. Most users habitually use a small set of applications. The manager node can maintain record of *"traffic-patterns"* associated with different applications when providing signalling support and later can use these patterns to select reservation parameters thus bypassing the traffic measurement procedure. This learning strategy can be further extended to automatically identify applications/users in the access network requiring QoS support. The user involvement in generating QoS-support requests (through web-based utility) can be completely avoided thus incorporating a higher level of intelligence and autonomous decision making in the network.

4 Implementation

In this section we describe the prototype model that we have built to couple QoS support to legacy network applications. The QoS support that is required by end applications operating over a IntServ based stub network (see Fig. 2) will require RSVP signalling.

We implemented the QoS-Support Netlet using Java so as to support portability and mobility of service code in the Netlets network. A web utility (see Fig. 3) was used to receive QoS support requests from users. A packet capture library [19] was used to identify flows corresponding to the end application requesting service. We used the Linux port of the RSVP package [20] to test the RSVP signalling for the QoS-Support requests. Java Native Interface (JNI) was used to interface native code that supported packet capturing and RSVP signalling with the Netlet service.

RSVP APIs were combined under function sets to reduce: (a) the number of calls the QoS-Support Netlet had to place between Java and native domains; and (b) the number of JNI stubs required to interface with the RSVP API. We developed three distinct function sets for our purpose: the sender, receiver and general set. The sender set encapsulated the QoS parameter initialisation (based on parameters received from the Netlet), socket creation for communication and RSVP session start-up methods for both unicast and multicast sessions which also included reservation checks and confirmations. Depending on the session type (unicast/multicast), the sender set implemented the set of interfaces required for signalling. The receiver set contained APIs that allowed receiving reservation requests, sending reservations messages and their corresponding error and confirmation messages. The general set encapsulated functions that checked for errors and confirmations during the reservation lifetime. We used the C language for building these function sets.

We performed tests to check the mobility of Netlet components offering signalling support and to confirm the connectivity established between Netlets and requesting session's receiver nodes. The test environment had two Netlet nodes serving as edges of a stub network and a manager Netlet node to process and coordinate QoS support requests. For the purpose of testing we used two end applications working in sender mode.

The end nodes hosting the application were configured to start the communication session through the two edge nodes of the stub network.

The manager node then launched a Netlet (based on user request) with address of the two edges nodes as the points for service installation. The Netlet states also included the FilterSpec and Tspec parameter (initial) of the flows requiring QoS support. The Netlet migrated to the edge nodes and installed the QoS support service. This confirmed the ability of the Netlet component to migrate and autonomously install service at desired points in the network.

On identifying the occurrence of a flow matching the FilterSpec parameter the signalling component of the QoS-support Netlet initiated RSVP signalling messages using the sender set and the general set function calls. The receiver node replied with acceptance messages for the reservation requests, thus confirming the validity of the session initiated by the QoS-support Netlet. This implementation running on a small network, proves the validity of the concept.

Deployment in Large Networks

Practical issues need to be addressed before this application can be deployed on a large scale. These issues include the following:

Multiple Manger Nodes: Depending on the size of the stub network, multiple manager Netlet nodes for QoS support can be used to process end user requests. This will prevent user requests from overloading a single node, thus avoiding a single point of failure in the network.

Non-Active Stub Network Edges: When non-active segments are present along the stub network edge, Gateway Netlet Nodes (GNN) can be used to enable signalling support to users attached to this segment of the network. In this case, the web-based utility can be enhanced to provide the user with a list of GNNs through which the end users can choose to route their traffic into the QoS aware network. Based on the GNN selected by the end user, Netlets can be deployed by the manager node to invoke signalling support for end applications.

Netlet Purging: On service completion (based on flow timeout periods or messages from the manager node), the QoS-Support Netlet can be either purged [9] or can autonomously travel back to the manager node which deployed it. The decision followed is based on the local policies being employed to manage the mobile components. This feature will prevent network services from holding onto network resources when not required.

Wireless Subnets: Large scale deployment will require the Netlets architecture to function in wireless subnets. Wireless networks have unique QoS requirements. The Netlets architecture is well suited to supporting QoS in such networks, since it provides a method to offer tailored network services at the interface between the wired and wireless segments of the network.

5 Conclusions and Future Work

In this paper, we presented a novel approach based on mobile agent-components called Netlets to transparently retrofit QoS support to legacy network applications. The Netlets

approach is not restricted to a single QoS model or signalling protocol. Thus it may be continue to be used even if the QoS support provided by the underlying network changes. This approach can be used as a permanent long-term solution to interface network applications with emerging QoS models on demand. Adapting this approach insulates future network applications required to be aware of the specifics of the network support for QoS. A proof of concept implementation has been deployed in a laboratory environment. Future work will address scalability issues in the deployment of these services and their extension to service provision in wireless networks.

References

1. C. Aurrecoechea, A. Campbell, and L. Hauw, “A Survey of QoS Architectures”, Multi-media Systems Journal, vol. 6, no. 3, pp. 138-151, May 1998
2. X. Xiao and L. Ni, “Internet QoS: A Big Picture”, IEEE Network Mag., March 1998
3. B.Riddle and A.Adamson, “A Quality of Service API Proposal”, http://apps.internet2.edu/qosapi.htm
4. The WinSock Group, “Windows Sockets 2 Protocol Specific-Annex”, http://apps.internet2.edu/winsock2.htm, 1996
5. P. Wang, Y. Yemini, D. Florissi, J. Zinky, “A Distributed Resource Controller for QoS Applications”, NOMS 2000, Hawaii, April 2000
6. K. Nahrstedt, J. Smith, “Design, Implementation and Experiences with the OMEGA Endpoint Architecture”, IEEE Journal on Selected Areas in Communication, Vol. 17, No. 7, September 1996, pp. 1263-1279
7. Jose Brustolonis et al., “Quality of Service Support For Legacy Applications”, International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV) 1999
8. Nicola Ciulli et al., “QoS provision to QoS-unaware applications on IntServ networks”, INET 2000 Posters, Stockholm, Sweden
9. Martin Collier, “Netlets: the future of networking?”, First IEEE Open Architectures and Network Programming Conf., San Francisco, CA, Apr. 1998
10. Kalaiarul Dharmalingam and Martin Collier, “Netlets: A New Active Network Architecture”, First IEI/IEE on Telecommunications Systems Research Symposium, Ireland, Nov. 2001
11. Braden R, et al., “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, Jun. 1994
12. S. Blake et al., “An Architecture for Differentiated Services”, RFC 2475, Dec. 1998
13. Y.Bernet et al., “A Framework for Use of RSVP with diff-serv Networks”, Internet-draft, June 1988, <draft-ietf-diffserv-rsvp-00.txt>
14. DARPA Active Network Program, http://www.darpa.mil/ato/programs/activenetworks/actnet.htm
15. Nwana H.S, Software agents : “An Overview, Knowlegde Engineering Review”, vol 11, no. 3, pp.205-244, Oct/Nov 1996
16. Clemens Szyperski Component Software, “Beyond Object-Oriented Programming. ACM Press and Addison-Wesley”, Reading, MA, 1998
17. R. Braden, et al., “Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification”, RFC 2205, Sept. 1997
18. P. White, “RSVP and integrated services in the Internet: A tutorial”, IEEE Commun. Mag., pp.100–106, May 1997
19. V. Jacobson et al., “pcap - Packet Capture library”, UNIX man page.
20. RSVP Port Under Linux, http://www.isi.edu/div7/rsvp/release.html