第VI部

Differenciated Services

第 6 部
Differenciated Services

第 1 章　概要

Differentiated Services (Diffserv) は、IETF (Internet Engineering Task Force) で検討が進められている、スケーラブルな通信品質 (QoS：Quality of Service) 保証を IP 網上で実現する技術である。本 WG では IETF では行われていない Diffserv の包括的な研究開発を行い、いち早いサービス展開を目指す。本 WG は主として以下の４つを柱に活動を行っている。

1. Forwarding Plane のインプリメーテーションの実ネットワークにおける検証
2. Management Plane 実現機器の研究開発
3. Diffserv に適応可能な Traffic Engineering 手法の研究開発
4. Diffserv を対象にした計測ツールの研究開発

本報告書では、上記の 1, 2 について記載する。問題点を 2 章で述べ、WIDE 合宿での実地テストの結果を 3 章で紹介した後に、4 章で Diffserv に適した API 及びネットワークアーキテクチャの提案について記載する。

第 2 章　Issues in augmenting Diffserv to meet application's CoS requirements

2.1 Introduction

The collapse of Internet fair-share model necessitates extended Internet architecture that can differentiate forwarding treatment of different types of flows. In response to such demand, Diffserv was designed as an extension of Internet fair-share model. In Diffserv, all traffic is classified into several classes, and each class may receive different forwarding treatment, according to consistent policy across Diffserv subnetwork, or Differentiated Services domain (DS domain).

Today, Diffserv specification has been standardized to the extent that it can be deployed within single DS domain. Applications can only implicitly signal their class-of-service (CoS) requirements to DS domain.

In this paper we describe various issues and possible directions in augmenting Diffserv, in order to meet application's CoS requirements.

In Section 2.2, our analysis of current Diffserv specification is presented. We point out that Diffserv cleanly separated both service models and underlying mechanisms from current set of specification, leaving enough flexibility for future development that will eventually augment Diffserv.

Section 2.3 describes pieces that will be needed by the time Diffserv is widely deployed in the Internet. We propose to decouple development of signaling protocols with application development by establishing Diffserv API. Next, we introduce the notion of service-level engineering and service-level monitoring that are necessary to meet customer's service-level specification (SLS).

In Section 2.4, we argue that Diffserv is useful in areas that are totally different from Intserv. SLS can be used to separate offered bandwidth from the bandwidth of access network. Diffserv can promote fair use by shaping and policing aggregate of greedy flows. Diffserv can more efficiently accommodate adaptive applications when applications can interact with NRB.

2.2 Analysis of Diffserv

Based on recent developments in link-sharing algorithms[51, 150] and active queue management algorithms[50, 27, 46], IETF Diffserv working group has designed Diffserv architecture. Diffserv

is a significant extension to the Internet architecture, while changes made to IP protocols are small.

The goal of Diffserv architecture is scalability. The key concept of Diffserv to achieve scalability is: 1) division of IP internetworks into DS domains, 2) traffic classification at both ingress node and egress node of DS domain, and 3) aggregated packet forwarding at interior nodes of DS domain. However, scalability does not come for free. These issues will be discussed later in Sections 2.2.4 and 2.3.

Diffserv architecture has been built around a set of mechanisms (e.g., packet classifier) and a minor redefinition of IPv4 TOS field and IPv6 traffic class field. The initial focus of Diffserv architecture is single DS domain, i.e., a portion of IP internetwork that has consistent policy on packet's class of service.

Diffserv architecture has carefully isolated service models and policies from currently specified set of standard. The standard is also independent of underlying mechanisms. This clean separation of specification is intentional; basic Diffserv specification will continue to be a convergence platform even after significant progress is made in underlying mechanisms or in service models.

2.2.1 Diffserv architecture

The core specification of Diffserv[113, 16] consists of mechanisms for packet classification at the border of DS domain, and mechanisms for differentiating packet treatment at the interior nodes of DS domain. At the ingress border of DS domain, packets are classified according to multiple fields of IP header, and the class of each packet is designated by Diffserv code-point (DSCP) in the IP header. In the current Diffserv architecture, MF (multiple field) classifier is specified for the purpose of packet classification.

In order to represent traffic conformance to particular traffic characterization parameter, meter and marker are defined. Meter and marker, together with packet-shaper and dropper, are collectively called traffic conditioners. However, none of these traffic conditioning mechanisms are specified as mandatory feature.

At DS interior nodes, packets receive different treatment according to DSCP. These differentiated treatment of packets are called Per-Hop Behaviors (PHBs) in Diffserv. Three PHBs are defined so far: EF (expedited forwarding), AF (assured forwarding) and DF (default forwarding, or best effort).

Packets marked as EF are prioritized over other packets, as long as their aggregate departure rate does not exceed configured limit. EF PHB can be implemented by variety of queueing disciplines, e.g., weighted round-robin. It may improve usability of delay-sensitive, or jitter-sensitive, applications.

AF PHB group is further divided into four classes, each of which may have three levels of drop precedence. Drop precedence bits can be marked by marker at the DS ingress node, according to traffic conformance to particular traffic characterization parameter. The combination of meter, marker and drop precedence can be used to realize the expected capacity service, which will be described later.

At DS interior nodes, packets with the same DSCP value are treated as an aggregate of flows, and they are collectively called Behavior Aggregate (BA).

2.2.2 Services

While service model is not part of current Diffserv specification, several services are described as examples in Diffserv literatures. One of those services is called "Better than Best Effort" service or BBE service; it emulates lightly-loaded best-effort network with AF PHB[15]. It seems to be almost identical with controlled-load service of Intserv architecture[166].

The key difference of BBE service to controlled-load service is the notion of Service Level Specification(SLS) and expected capacity. Under expected capacity scheme[27], subscriber's packets are marked lower drop precedence as long as they

conform to traffic profile. Packets exceeding the traffic profile are marked higher drop precedence, but they are forwarded as long as they do not experience congestion along the path.

Under expected capacity scheme, subscribers can complain to an ISP if they receive explicit congestion notification or experience packet drop when they are sending traffic within the traffic profile. Using the same mechanism, an ISP can detect under-provisioned network node when it is either dropping in-profile packets or marking CE (congestion experienced) bit on in-profile packets.

"Virtual Leased Line" service emulates leased line with predefined maximum bandwidth. It can be easily implemented with EF PHB, as described in [82].

These two services represent the initial set of service we would see in emerging Diffserv networks. They can be realized in a straightforward manner, without introducing additional mechanisms into Diffserv architecture.

In the current Diffserv architecture, offered services are limited to relatively static services such that negotiation for resource occurs rarely among human agents. It is not clear whether more dynamic services, such as video on demand, can be realized only with minor addition to the current Diffserv architecture.

2.2.3 Applicability of CoS

It should be noted that EF PHB cannot provide hard guarantee of maximum delay fluctuation to individual flow. Since flows are aggregated into BA, each flow is affected by queueing delay of other flows within the same BA, even if the BA is forwarded at highest priority. Theoretically, jitter-sensitive flows cannot be accommodated in Diffserv internetworks because of aggregation.

There are commonly used tricks to accommodate such flows. One can design bandwidth hierarchy of DS domain so that the backbone links are fast enough compared to that of access network. Queueing delay at DS interior nodes can be reduced, which will result in smaller jitter. In addition, one may buffer arriving packets if increase in end-to-end delay is not a problem.

Adaptive buffering and loss compensation are two of common methods for audio/video applications. Since Diffserv can only provide statistical delay bound, applications must anticipate packets that arrive after allowable period, even when EF PHB is used. Applications that require hard guarantee of packet delivery within bounded delay will encounter problems, although the probability may be small.

2.2.4 Scalability

Since initial focus of Diffserv architecture is deliberately limited to single DS domain and static SLS, scalability is limited within single DS domain and flexibility of service models are also limited. The problems of multiple DS domain and dynamic SLS are left for further study. The component that deal with multiple DS domains and dynamic management of bandwidth is called NRB (network resource broker) in this paper[1].

However, the presence of missing component does not preclude deployment of Diffserv across multiple DS domains. If ISPs can come up with common set of service classes, or if they can define translation between different set of service classes, Diffserv can be deployed across multiple domains just by concatenating them together. Diffserv architecture has necessary components to deal with this scenario; e.g., traffic conditioning at the DS egress node.

Dynamic SLS can be realized as well, to limited extent. If ISP backbones are over-provisioned and fast enough compared to individual flow's bandwidth fluctuation, they can accommodate certain number of dynamic flows. ISPs can detect overbooking situation by loss of in-profile packets and

[1] It is often called policy server or bandwidth broker in other literature, but since bandwidth is the interim bottleneck in the Internet architecture, NRB is introduced as a generalized concept of resource broker for bottleneck network resource.

react to the situation by reducing the number of accommodated dynamic flows. However, such methods are limited to relatively small flows. For example, in order to accommodate "HDTV on demand" traffic within an OC-12 backbone, mechanisms like NRB will be necessary.

2.3 Missing pieces

2.3.1 Application programming interface

There are variety of ways to adapt applications to Diffserv-capable internetworks. Applications can either implicitly indicate the class of service by other information in IP header (e.g., destination IP address, TCP port number) or explicitly indicate the class. Details of explicit indication approach may vary. Application can either set initial DSCP, send signaling packet to DS domain, or it might talk to NRB in the DS domain.

While initial deployment of Diffserv will most likely depend on implicit approach, provision must be made for certain types of applications that require explicit indication of service-level requirement.

While there are many ways to realize explicit indication of service-level requirement, application needs a single, consolidated programming interface for CoS capable internetworks. In order to understand the need for such a convergence platform, we need to take a look at the Socket API.

Socket API provides a single, consolidated programming interface for networked applications and network administration tools. Since it has been initially designed for 4BSD operating systems, the API has become widespread to other platforms as well, together with the growth of the Internet.

Socket API can be used to implicitly specify underlying protocol. Transport protocol for use with particular application can be selected by specifying "domain" and "type." For example, most of current implementations will select TCP if Internet domain and stream type is specified. In future versions of operating systems, the same specification may select a better transport protocol.

It is necessary to deal with signaling protocol and other explicit indication approaches via this kind of established interface, because signaling protocols will evolve when future innovations in underlying mechanisms become widespread.

While some people think RSVP will be used as the signaling protocol of Diffserv, we already see related protocol developments that may render RSVP unnecessary. For example, service location protocol[54] might be used to locate NRB, and electronic transactions may occur on top of XML with digital signature.

2.3.2 Established practice for service-level engineering

Depending on the type of service-level specification that each subscriber and ISP agreed with, it might be difficult to meet all of these SLS under certain traffic mix. For example, if the ISP has accommodated too many EF flows, it may result in excess packet loss at other classes.

ISP will be required to perform service-level engineering, in order to mitigate the impact of excess aggregation, inappropriate aggregation, unexpected mix of traffic, and so on. Service-level engineering will be an everyday task to meet service-level specifications of individual customers.

We will be in need of established practice for service-level engineering. Since many of the underlying queueing disciplines and active queue management algorithms have been well understood, it should not be difficult to digest existing work on queueing disciplines and recast them under the context of Diffserv.

Network simulator is an alternative way to assist service-level engineering. Network simulation tools, such as ns-2[12], can be used to help operators understand the problem by running simulation with setup similar to the real-world network. Possible difficulty lies in accurate workload characterization, but this potential problem could be overcome by implementing a traffic generator that conforms to SLS.

2.3.3 Established practice for aggregation

Since an application's packets must be aggregated with other application's packets under Diffserv architecture, the algorithm to form BA is of great importance. BA can be formed if individual flow can find a BA with similar traffic characterization parameters. But there is a problem in this approach.

There may be great diversity in traffic characterization parameters, whose aggregation may exceed available number of code-points. Considering the scale of current backbone ISPs, this situation is likely to happen. There must be established guidelines to deal with such situations, e.g., by exploiting use of experimental/local use code-points, or by aggregating dissimilar flows into single BA. The impact of aggregating dissimilar flows must be known as part of established practice for service-level engineering.

Application requirements will often be stated in more abstract terms than traffic characterization parameters. ISP and customer will more likely agree on the application-level usability index, from which SLS might be deduced or provided. If SLS cannot be deduced from application's service-level requirements, it must be provided either by application developers or by service-level engineers.

2.3.4 Components for service-level monitoring

In order to identify violation of SLS, it will be necessary to monitor the levels of service that are actually delivered to subscribers. Also, service-level monitoring will be useful to confirm that specific service-level engineering task has actually improved the situation. There are three methods of service-level monitoring.

The most direct and simplest approach is to put active performance probes at the border of DS domain. They can form a mesh of end-to-end performance probes, which will produce a matrix of SLS conformance.

Passive performance probes could be useful if they are co-located with active performance probes. Since both active probes and passive probes are fairly easy to build, there will be revised version of available products that are tailored to handle Diffserv packets.

Another approach to service-level monitoring is to collect data from various counters in the traffic conditioner components. For example, the number of lost packets, shaped (delayed) packets, and packets marked CE, could be maintained for each class. While these counters do not directly represent conformance to SLS, they will be helpful to identify poor aggregation policy or under-provisioned link. Unless IETF plays active role in standardization of management information base for Diffserv service-level monitoring, there will be incompatible set of proprietary management information base.

2.4 Applications of Diffserv

Diffserv is often mistaken as a scalable alternative to the Intserv[19]. Since Diffserv aggregates all traffic into a few classes, it is hard to guarantee packet delivery of individual flow. Those flows demanding hard guarantee of packet delivery or delay bounds must rely on Intserv architecture. There are rare exceptions where the same guarantee can be provided in Diffserv networks, e.g., by restricting all but one EF flow inside DS domain, but it is not discussed here.

Diffserv can be useful in areas that are totally different from Intserv. We will describe potential applications of Diffserv in this section.

2.4.1 Establishment of service level

SLS represents part of SLA (service-level agreement), which essentially is a contract between ISP and customer. Traditionally, SLA between ISP and subscriber was an implicit agreement, e.g., a right to use 56Kbps dial-up access networks anytime, for a fixed amount of fee. In the Diffserv architecture, SLA is an essential element for explicit agreement between ISP and customer.

SLA is a contract with more accurate techni-

cal characterization of traffic that a customer can send to and receive from his or her ISP. In other words, SLS isolates the specification of network performance from the speed of access network.

This isolation may have significant impact on the fundamental architecture of commercial next-generation Internet. ISPs will seek data-link technology with sufficient bandwidth for their access networks, regardless of the backbone link speed and forwarding capacity of backbone routers. High-speed data-link technologies become attractive, since ISPs can avoid frequent upgrade of access networks. Meanwhile, they can offer faster SLS to customers as backbone links and routers become faster.

SLA also has impact on the backbone provisioning policy. Since SLS is associated with a contract, failure to meet specific SLS may have negative impact on the ISP's revenue. Therefore, they will be forced to provision their backbone facility so that the sum of active SLS does not exceed the capacity of bottleneck resource.

2.4.2 Promotion of fair use

Diffserv can promote fair use of network resource, with appropriate combination of traffic conditioner elements. Greedy traffic can be classified into separate class, or can be marked with higher drop precedence at the DS ingress router. Such use of traffic conditioner will be useful to regulate unsolicited bulk e-mail, unsolicited advertisements, or malicious attempts to occupy bandwidth (e.g., directed broadcast of ICMP packets[25]).

Since the commercial sector started to develop Internet software for profit, there are growing number of software that attempt to accelerate end-to-end performance in a greedy manner. We already see prefetching web proxies, bulk e-mail transmitters, and modified TCP implementations, for example. This kind of market-driven development may necessitate deployment of Diffserv.

2.4.3 Applications with adaptive bandwidth requirements

Under Diffserv architecture, network can interact more closely with applications that have flexible bandwidth requirements. DS interior node may either drop a packet or mark CE bit in its IP header when the node experiences congestion[131]. Applications may react to congestion by detecting CE bit and then reducing the transmission rate.

There can be many alternatives for explicit congestion notification. NRB may inform applications of how much reduction in transmission rate are necessary. Applications that can respond to NRB messages will receive some pricing incentives, since these NRB-aware adaptive applications can be controlled very easily under congestion. In contrast, applications that only respond to ECN (explicit congestion notification) will be given marginal incentives, since ECN does not tell applications how much back-off is necessary.

2.4.4 Applications with adaptive scheduling

In the Diffserv architecture, applications can adapt to more flexible resource allocation policies than simple admission control. Under simple admission control scheme, end-node requests represent their instantaneous intent to use certain portion of network resource; requests are either permitted or rejected by the network. Flexible resource allocation policies include NRB-initiated admission of flow, and advance scheduling of flow. These flexible policies are particularly attractive when the requesting application does not require instant admission of flows.

NRB may initiate admission of flow if the requesting application does not have specific time constraint and when NRB finds sufficient amount of resource are available. This resource allocation policy might be useful to background application that works independently of others, e.g., background file transfer.

Advance scheduling of flow will be necessary if the requesting application must reserve other

resources, in addition to network resource. The application typically requires co-allocation of resource, i.e., it must find a time-slot where various kind of resource can be allocated at the same time. Other resources can be computational resources such as supercomputers, or scientific instruments, e.g., linear accelerator or electron microscope.

Smart middleware will be necessary to facilitate use of such flexible resource allocation policies. Grid toolkits[98], e.g., Globus[52], are ongoing efforts to provide smart middleware for resource co-allocation, as well as other essential functions for distributed computing. Other middleware efforts, e.g., NWS[164], may enable NRB-initiated admission of flow.

2.5 Possible directions of development

As we have seen in previous sections, resource allocation policies of individual DS domain are affected by the capability of NRB and end-node middleware. In other words, the algorithm of NRB and the mode of interaction between NRB and end-node middleware may change the way resources are shared among various classes.

This relationship has significant impact to ISPs, as the offered set of services may become richer depending on the NRB algorithms and protocols deployed within individual DS domain.

2.5.1 CoS middleware development

Middleware development has seldom gained interest from the Internet research community. While protocol design might be primary interest from engineering perspective, the development and consolidation of middleware are more important for application developers.

Middleware development is particularly important for extensible architecture like Diffserv, since feedback process is essential for further development of the architecture. By having initial version of CoS middleware, one can exploit application's potential CoS requirements. Some of those CoS requirements may not be met by underlying mechanisms, which will trigger development of improved NRB algorithms, improved traffic conditioner, etc. Improved underlying mechanisms may then be made available at the next version of middleware.

This feedback process will result in a CoS middleware that offer interface to rich set of services. The middleware may offer rich interaction of NRB and applications so that NRB can exploit application's adaptability.

It is desirable to have multiple research projects that attempt to develop CoS middleware through interaction with application communities and with Internet research community. Eventually, one will see consolidation process specifically for application programming interface, as we have seen in Grid Forum[71] for distributed high-performance computing.

2.5.2 NRB development

Since individual DS domain can deploy different set of NRB implementations, there will be strong demand for better NRB algorithms. ISPs may attempt to differentiate themselves from other ISPs by incorporating better NRB implementations into their networks.

NRB development may follow sophistication process similar to CoS middleware. Eventually, the protocol between NRB and CoS middleware must be standardized so that NRB development can happen independent of end-node middleware, which may be difficult to change.

2.6 Conclusion

This paper made several contributions by presenting analysis of Diffserv's current status, missing components and practices for CoS applications and ISPs, and possible application of Diffserv by both ISPs and application programmers.

We first pointed out that Diffserv cleanly separated both service models and underlying mechanisms from current set of specification. The separation leaves enough flexibility for future development that will eventually augment Diffserv.

Next, we have discussed pieces that will be

necessary for widespread deployment of Diffserv. We have proposed to decouple development of signaling protocols with application development by establishing Diffserv API. Then, we have discussed the necessity of service-level engineering and service-level monitoring. We also pointed out that aggregation of dissimilar flows may have negative impact on delivered service level.

Next, we argued that Diffserv is useful in areas that are totally different from Intserv. SLS can be used to separate offered bandwidth from the bandwidth of access network. Diffserv can promote fair use by shaping and policing aggregate of greedy flows. Diffserv can more efficiently accommodate adaptive applications when applications can interact with NRB.

Finally, we argued that NRB and CoS middleware will gain research interest. Several trials to design these components will result in standardized NRB protocol and Diffserv API. NRB's internal algorithms and CoS middleware's capability may differentiate the set of services offered to subscribers.

第 3 章　QoS ルーティングとダイナミック SLA のフィールドテストとその考察

3.1 はじめに

ベストエフォート転送が前提であったインターネット上で品質制御 (QoS) サービスを提供するために、Integrated Service、Differentiated Services、QoS ルーティングなど、様々な要素技術が提案されている。しかしその一方で、「ユーザサービスとしてのこれらの要素技術の積み上げ」という視点での研究が充分に行われておらず、これが QoS サービスの Deployment における大きな課題となっている。特にユーザがダイナミックに QoS をネットワークに要求したときの問題点については議論が充分になされていない。

このため、ユーザが QoS サービスをダイナミックに要求できる (ダイナミック SLA を要求できる) ネットワークを 2000 年秋の WIDE 合宿で構築・提供し、本フィールドテストを通して、ダイナミック SLA を提供する上での課題の抽出を行った。

3.2 合宿ネットワーク

合宿ネットワークで用意された 1.5Mbps の ATM と 2.0Mbps の衛星リンクのうち、本実験ではそれぞれ計 1Mbps を用いた (図 3.1)。また SFC (慶應大学湘南藤沢キャンパス) 経由と NAIST (奈良先端科学技術大学院大学) 経由の 2 経路に対し、OSPF のコスト値により、SFC 側のリンクをメインに、NAIST 側のリンクをバックアップとして利用した。

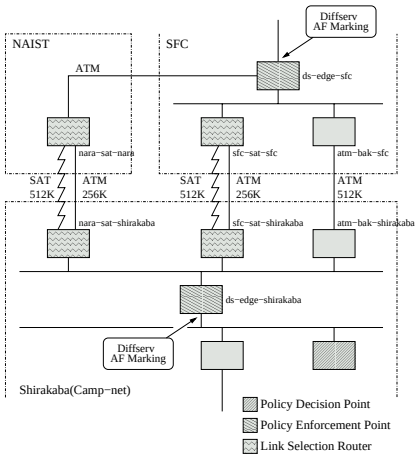

図 3.1.　合宿ネットワーク

ユーザからの要求受付とそれに伴うリンクの選択、及び帯域の予約は図 3.2 の帯域予約システムで提供した。ユーザは web インターフェース経由で、このシステムにアクセスできる。予約要求は Policy Decision Point (PDP) で受付可否が決められ、その決定は Policy Enforcement Points (PEPs)、即ちルータに送

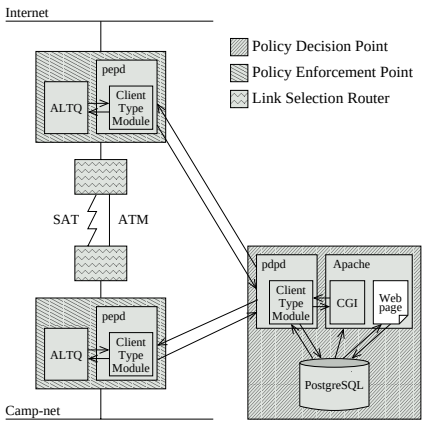

図 3.2.　帯域予約システム

られる。この PDP, PEP のプラットフォームは IPv6 スタックである KAME[72] を載せた FreeBSD3.5 であり、両者の通信には COPS (Common Open Policy Service protocol)[17] を採用した。またパケットへは、Diffserv の AF (Assured Forwarding) クラス [106] により帯域保証を提供した。

さらに本システムは “WIDE Unit” という仮想通貨による課金をサポートしており、各参加者には 10,000 WIDE Unit を初期値として与えた。今回は ATM リンクを利用するユーザに対して、以下のポリシで課金を行った。

- ATM リンク
  - リンク利用率 50%以下で、ユーザ要求 1 kbps 分につき 1 WIDE Unit
  - リンク利用率 50%以上では利用率が 5%増加するごとに、10%の値上げ
- 衛星リンク
  - ATM リンクの 1/4 の価格設定

ATM リンクと衛星リンクの価格差は、衛星リンクの RTT 値が ATM に比べ平均 11.7 倍大きいためである。

3.3 結果と評価

3.3.1 結果

2000 年秋の WIDE 合宿の出席者、252 名中 126 名が本 SLA システムを利用した。4 日の実験での予約数は延べ 605 であり、78.8%が ATM リンクへの予約であった。さらにこのうち 46.3%が 1kbps 分の要求であり、100kbps 以上を要求した参加者はわずかであった (図 3.3)。また 6〜30 分の帯域予約が 48.2%を占めた (図 3.4)。

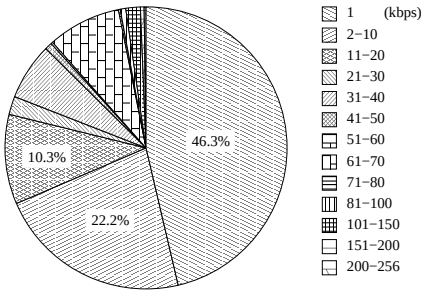

図 3.3. ATM リンクへの要求帯域値の分布

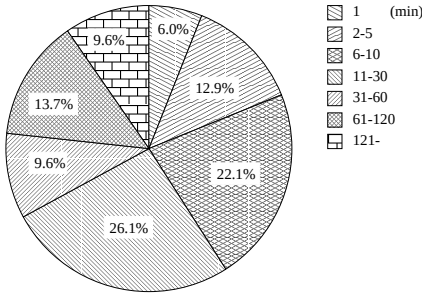

図 3.4. ATM リンクへの要求帯域時間の分布

3.3.2 評価

前回の合宿での同様な実験では 43.8%の参加者が本システムを利用したが、今回はこれが 50%にまで増加した。これによりインターネットにおける QoS サービスの普及に見込みがあると判断している。

一方、課題として下記の 2 点が抽出できた。

まず、ユーザに対するレートコントロールの管理手法の確立である。本実験では RIO[27] により各ルータにおけるバッファ管理を行ったが、RIO ではパケットの待ち行列が定常的に長くならない限り、予約帯域分以上のパケットに対しても廃棄を行わない。今回は ATM リンクの帯域幅が充分に広かったため、待ち行列が長くならず、結果としてユーザは予約分以上の帯域が利用できてしまう問題点が起きた。

次に、QoS のユーザへの見せ方の確立である。つまり QoS にも遅延、ジッタ、帯域、パケット廃棄率といったパラメータがあり、これらをユーザが直感的にわかるように見せることが重要である。本実験では当初、ATM は小さい RTT を要求するアプリケーション向けに、衛星は広い帯域幅を要求するアプリケーション向けを意図していたが、ATM のほうが RTT 値が小さく、かつ広帯域の予約のニーズが低かったために充分にユーザを収容できた。このため衛星の特長をアピールし切れなかった。即ち、ATM の RTT 値の小ささが衛星リンクの帯域におけるコストパフォーマンスの良さよりもユーザにアピールしたことになる。このように、QoS の各パラメータをユーザやアプリケーションにおける重要度に基づいて整理する必要がある。

3.4 まとめ

本報告では、2000 年 9 月の WIDE 合宿でダイナミックなユーザ要求に基づく QoS 提供ができるネットワークのフィールドテストを行い、その有効性の

確認及び課題の洗出しを行った。

今回の結果では、ユーザに公平に帯域を割当てるようなレートコントロール技術の必要性と QoS 属性のユーザへの見せ方及び課金設定が課題として明確になった。

今後、QoS の様々な属性の測定方法やユーザへのその結果のフィードバックや QoS ルーティングを含め、ダイナミックなユーザ要求に対するインターネットでの QoS 提供を研究していく予定である。

第 4 章　サービスの信頼性を拡張した QoS ネットワークアーキテクチャの研究

4.1 序論

本研究では、ユーザに QoS を意識させないでサービス品質を要求できる API を設計し、QoS ネットワークアーキテクチャの要求仕様を考える。すなわち、ユーザに必要なネットワーク品質とその提供手段とを、アプリケーション／利用目的毎に仕様として取り決め、ユーザはアプリケーションおよび利用目的のみを意識すれば良いようにする。また、ネットワーク品質が有料で提供されることへの配慮も重視する。具体的には、障害発生下での品質保証手段、品質劣化時の課金停止、品質に関する情報の提供の 3 点について検討し、これに対して充分な信頼性を持つ API とネットワークアーキテクチャの提案を行う。

4.2 アーキテクチャへの要求条件

本章ではユーザから見たネットワークアーキテクチャの要求条件を考察し、その結果から望ましい API を提案する。図 4.1 に API と前提とするネットワークの関係を示す。

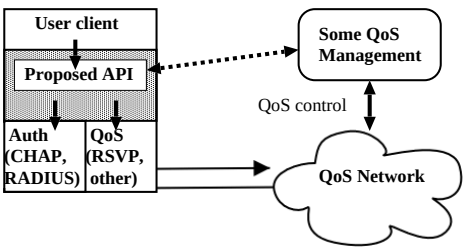

図 4.1.　API と QoS ネットワークとの関係図

なお本報告書では、実現性とユーザの利便性を考え、以下の特徴を有するネットワークを前提とする。

- Diffserv アーキテクチャに基づいたネットワークノード
- ドメイン内のネットワーク情報とユーザ情報の実時間管理
- SLA に基づくネットワーク資源や品質の監視機構

4.2.1 想定アプリケーション

QoS を要求するアプリケーションとしては、現在、下記のものが代表的である。

- ストリーム配信
- IP 電話
- データ転送
- エリア毎の帯域幅制限

「エリア毎の帯域幅制限」とは特定のネットワークもしくはクライアントからの全トラフィックの単方向帯域幅制限を行うものである。例えば、企業内の部署毎に外部へ発信するトラフィック量への帯域割当である。

これらのアプリケーションから QoS への要求事項を表 4.1 にまとめる。本表から、QoS 設定を要求するのに必要な情報として、フローを識別するためのフロー情報と、フローへの QoS 設定が有効になる開始・終了時間の時間情報、帯域幅やその属性、遅延・ジッタなどのサービス情報が挙げられる。しかし遅延とジッタの保証はインターネットでは難しいことから、API としてユーザに開示するパラメータからは除外する。

サービス番号とはサービスの属性を指定するパラメータであり、ユーザを細かな通信品質を指定する煩わしさから開放する。例えば、サービス番号 1 番はインターネット放送用のサービスであることをユーザには提示し、ユーザが利用するアプリケーションの情報、フロー情報、及び時間に関するパラメータがパッケージとして扱われる。よって必要なパラメータは以下の通りにまとめられる。

表 4.1. QoS が有効なシステムモデルと QoS に関係するパラメータ

システムモデル	帯域幅の属性	通信方向	遅延	ジッタ	その他
ストリーム配信	最大値保証	単方向	-	最小	時間指定
テレビ会議システム	最大値保証	双方向	最小	最小	時間指定
IP 電話	最大値保証	双方向	最小	最小	終了時間無指定
データ転送	最小値保証	-	-	-	パケットロス小
エリア毎の流出量制限	最小値保証	単方向	-	-	時間指定、 アドレスマスク
エリア毎の流入量制限	最大値保証	単方向	-	-	アドレスマスク

- フロー情報
  送信側 IP アドレス、アドレスマスク、ポート番号
  受信側 IP アドレス、アドレスマスク、ポート番号
  アドレスファミリー、プロトコル、DSCP、フローラベル
- 時間情報
  開始日時、終了日時または利用時間
- サービス情報
  帯域幅、サービス番号

以下、サービスとはサービス情報によって設定された QoS によって得られるネットワーク品質を指す。

4.2.2 障害発生時の処理

QoS は一般に差別化サービス提供のために、ユーザから課金を取ることが想定される。よってユーザに提供した QoS は、障害発生時においても正常に提供しなければならず、これが困難な場合には代替経路を選択したり、課金を停止するなどの適切な処理を行い、ユーザに対してその理由などを通知する必要がある。

ネットワークの障害は大きく二つに分類できる。

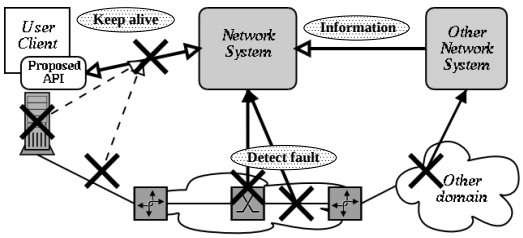

図 4.2. 管理ドメインと外部との接続と障害検出範囲図

即ち、ネットワーク管理者が修理や交換、変更をしない限り改善されない重度障害と、一定時間経過すると状況が改善されうる軽度障害である。障害の度合いによって、サービス提供維持のために採るべき対策は異なるべきである。

ネットワークの障害時には、障害発生箇所の検知とユーザ通知の二つが必要である。即ち、「ネットワーク資源とユーザとを関連づけての管理」がネットワークへの要求事項になる。

ドメイン外での障害によりサービスの提供に影響が及ぶ場合もある。図 4.2 に管理しているドメインと外部との接続を示す。

図 4.2 の Detect fault が管理ドメイン内の障害検知を、Information が他ドメインからの障害通知を示す。また、ユーザがネットワークへ接続している接続回線は管理ドメインの外にある場合も想定され、その場合には物理的・論理的な接続性をネットワークを監視している部分が把握することは困難である。よって、ユーザを管理する部分がユーザとの間で生存確認を行うことでユーザがサービスを利用できる状態であることを確認することにする。図 4.2 では Keep alive の部分にあたる。これによりユーザが利用できない状態であるにもかかわらずサービスの設定が有効になっており課金されつづけるという状況は回避できると考えられる。

上記した障害通知と生存確認、さらに QoS 要求などのメッセージはユーザとネットワークシステムのユーザを管理する部分との間に制御用の通信路 (制御コネクション) により、やりとりされるようにする。このためサーバにはサービスを利用しているユーザの数だけ制御コネクションが張られることとなる。

4.2.3 現在の状況を把握する手段

4.2.2 節で説明したように、ユーザとネットワーク

システムとの間には制御コネクションが張られていて、そこを障害通知や生存確認のためのメッセージが通過する設計とする。しかし、利用しているサービスが停止されるには障害以外にも、ユーザの料金未払いや利用可能な上限まで到達したなどである。それらネットワークシステムが持っているユーザに関連する情報もまたユーザの知りたい情報となる。また、ネットワークシステムからは障害情報以外の情報も通知することでより詳しくユーザが現在のサービス提供状況を把握することが可能になる。そこで、API として QoS 設定の要求・解除、障害報告、生存確認の他に情報を問い合わせる機能も含めることにする。

4.2.4 API の設計

前節までをまとめると、API に必要な特徴は以下のようになる。

- 複数のサービス要求が可能
- いつでも障害通知を受信できる状態
- 生存確認を行う
- ネットワークとのメッセージの送受信を行うセッションを確保
- いつでも問い合わせができる状態

ユーザはネットワークシステムのユーザを管理する部分との制御コネクションを確保することで API の重要な特徴を実現できる。この制御コネクションを確立する段階でユーザは認証される必要がある。認証後には制御コネクションは維持されるため、その制御コネクションを介して問い合わせなどをするときには再度ユーザを認証する必要はなくなる。

複数のサービス要求を可能とするには、ネットワークシステムへの接続とサービスの利用開始を一つの命令で行ってはならない。よって、ネットワークシステムへの接続とサービスの要求は別の命令とし、ネットワークシステムへの接続は認証とセッションの確保を、サービスの要求はユーザがサービスを利用するためのデータの送受信を目的とする。生存確認は定期的にユーザがネットワークシステムに対してメッセージを送ることで実現する。生存確認を行うプロセスはネットワークへの制御コネクションが確立した時点から動作し、制御コネクションを失うまで動作し続ける。

障害通知はサービス停止の理由を通知するために使用するが、それ以外の情報も同じ枠組に入れることが可能なため、サービス利用が開始されているかどうかにかかわらず受信できる状態にしなければならない。そのため、障害通知を受信できる状態にする段階も制御コネクションが確立されたときとなる。

さらに、障害通知を受信できる状態とは別に、いつでもネットワークシステムに対して問い合わせをすることが可能である状態でなければならない。そのためには確保されている制御コネクションを受信用と送信用の 2 つ用意するか、もしくは一つの制御コネクションを競合しないように共有しなければならない。それらは実装依存となるので、セッションを直接気にする必要のないインターフェースでなければならない。

また、問い合わせの内容としてユーザの各種の情報を問い合わせる場合はその種類を指定することになるが、相手先まで指定するサービスが利用可能かどうかという問い合わせはフロー情報やサービス情報を含む QoS 情報を引数にする必要があるので別の扱いとする。

これらの考察から、この API は図 4.3 の状態遷移を持つ。API としては次のような命令を用意する。戻り値の値は正常な応答のものである。

- ネットワークへのセッションの開始 (connect)：ユーザ認証、メッセージ送受信用セッション確立。
  引数：ユーザ認証情報
  戻り値：生存確認間隔
- ネットワークへのセッションの終了 (close)：セッション切断
  引数：なし
  戻り値：なし

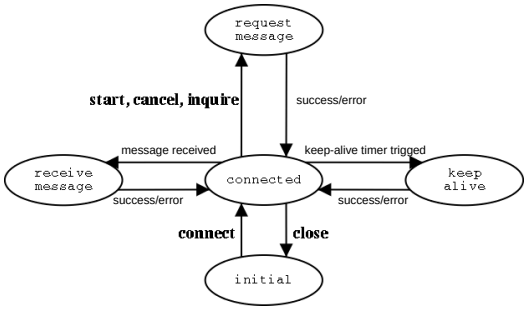

図 4.3.　API の状態遷移図

- サービスの利用開始 (start)：QoS 設定情報によるサービス開始
  引数：QoS 設定情報 (フロー情報、時間情報、サービス情報)
  戻り値：サービス利用識別番号
- サービスの利用終了 (cancel)：サービス利用識別番号に関連するサービスの終了
  引数：サービス利用識別番号
  戻り値：なし
- サービスが利用問い合わせ (inquire_service)：QoS 設定情報によるサービスが利用可能かの問い合わせ
  引数：QoS 設定情報 (フロー情報、時間情報、サービス情報)
  戻り値：単位時間当たりのサービス利用料金
- ユーザ情報の問い合わせ (inquire_status)：ユーザに関する各種情報 (e.x. ネットワーク障害情報、ユーザ契約情報) の問い合わせ
  引数：ユーザ情報の種類
  戻り値：ユーザ情報
- ネットワークからのメッセージ受信 (receive)：ネットワークシステムからのメッセージ受信・処理
  引数：ネットワークからのメッセージ
  戻り値：なし
- 生存確認 (keep-alive)：生存確認用メッセージの送受信
  引数：なし
  戻り値：なし

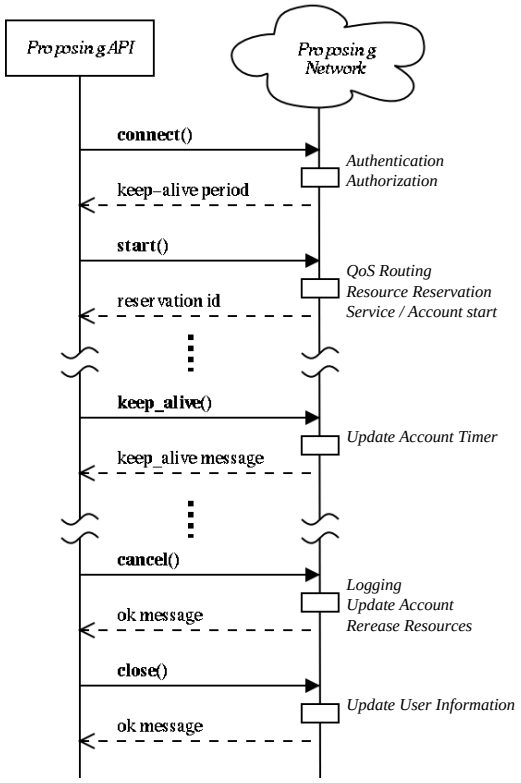

図 4.4. API のシーケンス図

この API を使用してサービスを要求して終了するまでの流れを図 4.4 に示す。

4.2.5 評価

本 API と他の API との比較結果を表 4.2 にまとめる。対象は RSVP Application Interface[18]、Winsock2 GQOS API[14]、Internet2 QoS API[134] である。

表 4.2. 既存の QoS 用 API との比較

API 名	RSVP API	GQOS API	Internet2 QoS API	提案する API
命令数	4	?	5	8
対象プロトコル	RSVP	非依存	非依存 (RSVP)	非依存
対象プラットホーム	非依存	Windows	非依存	非依存
認証	なし	なし	あり	あり
障害報告 (QoS 設定時以外)	なし	Winsock2 のサポート範囲	ライブラリに依存	常時受信可能
状態の問い合わせ	なし	なし	なし	あり
QoS 既定値解決	なし	あり	あり	あり
遅延パラメータ	なし	あり	プロファイルに依存	サービス番号に依存
時間パラメータ	なし	なし	なし	あり

4.3 ネットワークアーキテクチャの設計

本章では前節で設計した API を提供するネットワークのシステムについて考察し、API を含むネットワークアーキテクチャ全体を考察する。

4.3.1 API に対応したシステム

ネットワークシステムはユーザ管理機構、サービス管理機構、ネットワーク管理機構から構成する。これらは論理モデルであるため、物理的に分散させることが可能である。構成図を図 4.5 に示す。

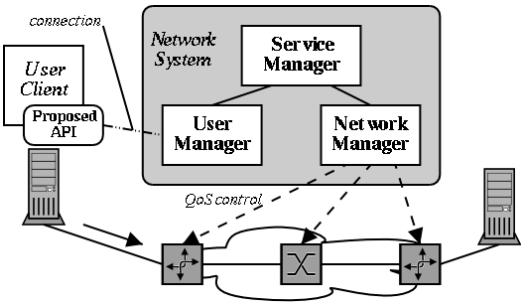

図 4.5.　ネットワークアーキテクチャの構成図

ユーザ管理機構は、ネットワークシステムがユーザに対してサービスを提供するために必要なユーザに関する下記の情報を有する。

- ユーザ識別情報
- 認証情報
- サービス利用契約情報
- 課金情報
- 制御コネクションの管理
- 現在の利用サービス情報

ユーザ識別情報や認証情報、契約情報は自動的に更新されるものではないが、課金情報は利用していたサービスが終了した時点で更新される。制御コネクションは API がメッセージの送受信に使用するものである。この制御コネクションはユーザを認証した後もそのまま維持され、API の機能であるネットワークシステムからのメッセージの通知や生存確認、サービスの要求や問い合わせを実現する。この制御コネクションの維持により信頼性のある実時間課金が可能となる。トラブルでユーザがサービスを利用しているにも関わらず制御コネクションを失ったときは、その時点で課金を停止しネットワーク資源を解放するか、ネットワーク資源の解放を少し待ち、ユーザから再び制御コネクションが確立されてサービスの終了を宣言されなければ課金を再開するなど適切な処理を行う。ユーザからのサービス要求を受信したときはユーザ管理機構はそのユーザが指定したサービスを利用可能かどうかをサービス利用契約情報に基づいて判断する。サービス管理は QoS のパラメータの設定によって決定するが、ネットワーク提供者からの情報開示の難しさ、ユーザの設定の煩わしさ、を考慮し、大部分の QoS のパラメータを予めネットワーク品質と関連付けておき、それをユーザにサービスとして提示する。そのサービスを管理するのがサービス管理機構である。具体的には以下の情報を持つ。

- サービス情報
  ネットワーク資源情報、料金体系情報
- 現在のユーザ利用状況
  ユーザ毎の予約ネットワーク資源量、ユーザ識別番号、予約番号
- 現在のサービス利用率
  サービス毎の予約ネットワーク資源量、実際の使用ネットワーク資源量

本機構がサービスを介して利用ユーザとネットワークノードとを関連づけることで、障害が発生したときやサービス終了のメッセージを受信したときにネットワークノードの QoS 設定を無効にできる。また、ユーザからのサービス要求を受信したときはサービス管理機構は指定したサービスが利用可能かどうかを他のユーザの利用状況に基づいて判断する。ネットワーク管理機構は、ネットワークノードの設定と監視を行う。ネットワークノードから得られる情報はネットワーク管理機構で扱うだけでなく、サービス管理機構にも渡す必要がある。具体的には以下の情報を持つ。

- ネットワークノード情報 (QoS パラメータ、インターフェース情報、利用サービス番号)
- 現在のネットワーク資源利用率
- ドメイン内 QoS ルーティング
- 障害管理情報

ネットワークノード情報はネットワーク管理機構が

管理しているルータやスイッチなどである。例えば、ノードの識別番号や装備しているインターフェースと割り当てられているアドレス情報、利用可能なネットワーク資源とそのノードを利用しているサービスの番号である。ネットワーク資源利用率は利用しているサービスとは関係なくクラス分けされたトラフィック毎に帯域幅や遅延時間などを測定して現在の値を監視する機能である。障害管理情報はドメイン内での物理・論理的なネットワーク障害の状況を示す。

ユーザからサービス要求を受けたときの流れは次の通りである。要求サービスを実現する QoS パラメータをサービス管理機構から受け取る。その QoS パラメータを基に QoS ルーティングを行う。このときにネットワーク資源の不足や障害によってパス検索が失敗に終ったときはエラーメッセージをサービス管理機構、ユーザ管理機構を経由してユーザへ通知する。パスが発見された場合はパスを構成するネットワークノードに QoS 設定を行う。

4.3.2 ネットワークシステムの機能要素

QoS を保証してネットワーク品質をユーザに提供するには次のような機能が必要になる。ユーザ認証、ユーザ認可、課金、ユーザ状態管理、QoS 設定制御、トラフィック監視、障害管理、QoS ルーティング、スケジュール管理、ポリシ管理、サービス状態管理、メイン調停 (インタードメインでの契約の作成・変更・削除)。

表 4.3. QoS ネットワークシステムの機能要素と対応する管理機構

機能要素	対応する管理機構
ユーザ認証	ユーザ管理機構
ユーザ認可	
課金	
ユーザ状態管理	
QoS 設定制御	ネットワーク管理機構
トラフィック監視	
障害管理	
QoS ルーティング	
スケジュール管理	
	サービス管理機構
ポリシ管理	
サービス状態管理	
ドメイン調停	

提案するネットワークシステムは、ユーザ管理機構、サービス管理機構、ネットワーク管理機構の 3 つの部分で構成することで上記、必要機能を実現する。それぞれの機能分割を表 4.3 にまとめる。

4.3.3 階層化管理によるドメイン間管理への適応

これまではユーザとネットワーク提供者との間でのサービスの提供について説明し、この関係でサービスの正常な提供が可能であることを示した。この関係を、隣接ドメイン間でのサービス提供の枠組に適応することが可能であることを示す。

ユーザとネットワーク提供者の間でサービスを提供する枠組が成立すると、その関係を下流と上流のプロバイダとの関係に置き換えることでも成立すると考えられる。これは Diffserv の特徴であるトラフィックの集約によるもので、ネットワーク提供者も 1 ユーザとみなせるからである。下流のプロバイダをユーザに、上流のプロバイダをネットワーク提供者にみなせるからである。ドメイン間へ適応させた構成図を図 4.6 に示す。

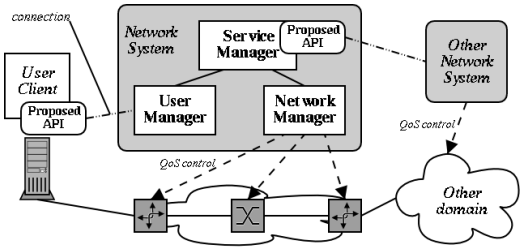

図 4.6. ドメイン間通信を含むネットワークアーキテクチャの構成図

4.3.4 評価

提案ネットワークアーキテクチャと他のモデルとの比較結果を表 4.4 にまとめる。対象は ENICOM PREM[143]、Internet2 QBone Bandwidth Broker[153]、BBN QuO/AQuA[85][32] である。なお、本研究で提案するネットワークシステムのネットワーク管理機構には PREM を使用するため、該当する部分には PREM の機能であることを記した。

4.4 実装

4.2 章、4.3 章で設計したネットワークアーキテクチャのリファレンス実装について説明する。

表 4.4.　既存のネットワークアーキテクチャとの比較

アーキテクチャ名	PREM	QPS+GARA+AAA	AQuA	提案するシステム
ユーザ認証	あり	あり (AAA)	あり (QuO)	あり
ユーザ認可	あり	あり (AAA)	あり (QuO)	あり
課金	なし	あり (AAA)	なし	あり
ユーザ状態管理	なし	あり (QPS)	あり (AQuA)	あり
QoS 設定制御	あり	あり (QPS)	あり (QuO)	あり (PREM)
トラフィック監視	なし	あり (QPS)	あり (AQuA)	あり
障害管理	なし	なし	あり (AQuA)	あり
QoS ルーティング	あり	あり (QPS)	なし	あり (PREM)
スケジュール管理	あり	あり (GARA)	なし	あり (PREM)
ポリシ管理	あり	あり (GARA)	あり (AQuA)	あり (PREM)
サービス管理	なし	あり (QPS)	なし	あり
ドメイン調停	なし	なし	なし	あり

4.4.1 概要

実装はプラットホームへの依存を避けるために Java2 version 1.3 Standard Edition にて行った。

この実装ではユーザ管理機構はサービス管理機構が生成する 1 つのオブジェクトとした。そのため、ユーザ管理機構とサービス管理機構の間でのデータの授受にはプロトコルは用意せず、関数呼び出しの際の引数や戻り値としてデータの授受を行った。

またネットワーク管理機構には PREM を使用した。PREM とサービス管理機構とのデータの授受は Java2 で用意されている CORBA/ORB によって行った。

4.4.2 実装の動作検証

実際の実験環境では、ユーザ・サービス・ネットワーク管理機構は全て同じサーバ内にて動作させた。図 4.7 に実験構成図を示す。

動作確認には DV_A から DV_B へ帯域幅を予約して、途中のルータにおいては要求された帯域幅予約が正常に設定されているかを監視することとした。また終了時間を指定して、終了時間が過ぎた場合には優先配送や帯域幅指定が解除されていることを確認した。

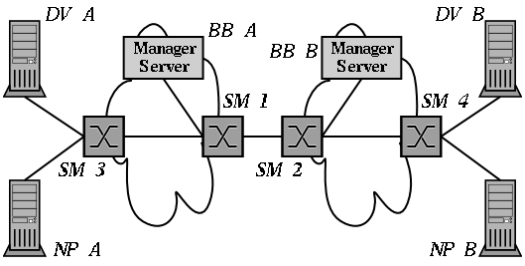

図 4.7.　実験の構成図

また、予約が有効な期間においてネットワーク障害が発生したときに正常にサービスが停止することを確認した。今回はネットワークが 1 経路で、PREM がサポートしている PHB が EF のみのため、代替サービスへの移行は行わなかった。また、PREM にネットワーク監視機構が無いため、実際には SMOperator 内で疑似的に障害情報を発生させ、サービスを停止するという契約が結ばれていると想定して、実際にサービスが停止することを確認した。

4.5 結論

本研究では、インターネット上で交換される情報のマルチメディア化が急速に進行する状況を受けて、QoS を保証することを眼目とするネットワークアーキテクチャの設計を行った。音声/画像を中心とするマルチメディア化された情報は、通信される際の帯域幅や遅延等の品質に対し厳しい要求条件を持つ。提案のアーキテクチャは、これらの通信品質を保証すること自体をネットワークサービスとして提供する観点から設計されており、回線の切断やルーティング異常などの障害状態が発生した場合にも正常にサービスを提供し続ける機能を含むものである。

本研究で提案する API およびネットワークアーキ

テクチャは、特定のプラットホームやプロトコルに非依存で移植性に富むことに加えて、的確で精度の高い課金機能を実現している。また、インターネット放送やテレビ会議のように予約を必要とするサービスにも対応しているのを始め、多様なサービスに柔軟に対応出来るよう設計されている。さらに、セキュリティ対策についても、関連する指標を QoS パラメタとして設定する方法により対応している。

さらに提案した API とアーキテクチャの下で、システムが有効に機能することを実証するため、リファレンス実装を行った。