감사 추적 시스템 설계 및 구현을 통한 안전한 QoS 빌링 시스템

박 우 출[†] · 김 정 녀[††] · 이 병 호[†††]

요 약

본 논문에서는 VoIP나 멀티미디어 서비스와 같은 실시간 트래픽을 위하여, IETF에서 인터넷 QoS 제공 방안으로 제시된 IntServ와 DiffServ 혼합 모델을 사용해서 Delay Sensitive 트래픽과 넓은 대역폭을 가지는 QoS 서비스를 제공하였다. 제안한 QoS 서비스를 Best, Good, Default 서비스로 정의하고, 제안된 IntServ와 DiffServ의 혼합 모델을 사용한 End-to-End QoS의 3가지 서비스 모델을 시뮬레이션을 통하여 성능을 분석하였다. 제안된 QoS 모델에 따른 서비스 요금 지불을 위하여 RADIUS 프로토콜의 Accounting, Authentication, Authorization (AAA) 기능을 이용한 IntServ와 DiffServ의 혼합 모델을 사용한 빌링시스템을 제안하였다. 안전한 빌링 시스템에서는 RADIUS 프로토콜의 감사 추적 기능을 강화하기 위하여, IEEE POSIX.1E 표준안에 의한 감사 추적 시스템을 설계 및 구현하였다.

Secure QoS Billing System Using Audit Trail Subsystem Design & Implementation

Woo Chool Park[†] · Jeong-Nyeo Kim[††] · Byung Ho Rhee[†††]

ABSTRACT

In this paper, we propose the delay sensitive traffic and a high bandwidth QoS service in order to supply real-time traffic such as VoIP, multimedia service. We use IntServ over DiffServ network to supply end-to-end QoS service in the IETF. We define the proposed QoS services which are Best, Good, Default service. We analyze the performance using NS simulator with end to end QoS service in IntServ over DiffServ network. The proposed billing system uses the Accounting, Authentication, Authorization (AAA) functions of RADIUS protocol and proposes the dynamic pricing method according to network usage state using end-to-end QoS of IntServ over DiffServ network. In order to secure billing system, we design and implement audit trail system by the IEEE POSIX.1E standard.

키워드 : QoS, 감사 추적(Audit Trail), RADIUS, IntServ, DiffServ, 빌링 시스템(Billing System), 실시간 서비스(Real-time Service)

1. 서 론

본 논문에서는 인터넷의 발전에 따라 다양한 QoS 서비스가 요구가 증가하여 해결책으로 IETF에서 제안한 IntServ와 DiffServ 모델을 사용하여, VoIP나 멀티미디어 서비스와 같은 실시간 서비스에 맞는 delay-sensitive 트래픽과 넓은 대역폭을 가지는 서비스를 제공하고자 한다.

IntServ는 Resource Reservation Protocol(RSVP)에 의존하며, 네트워크에서 각 흐름을 위해 원하는 QoS를 예약하고, 시그널을 보내는데 사용된다. RSVP는 IP 네트워크에서 실시간 전송을 향상시킬 수 있도록 고안된 프로토콜이며, 인터넷 상에서 제공되는 여러 가지 서비스를 통합적으로 이용할 수 있도록 해주는 프로토콜이다. 일반적으로 RSVP는 네트워크에 특정 QoS를 요청하는데 사용되며, 요청된 요구 사항에 따라 응용 프로그램의 데이터 스트림에 통보된다. 이때 요구된 QoS에 해당되는 대역폭의 QoS를 요구하는 송신측이 아니라 수신측에서 먼저 계산되어 허용 가능한 범위 내에서 제시하게 되고, 최초 요구한 송신측에 이르기까지 이러한 과정이 계속되어 대역이 예약된다. 따라서 RSVP 는 인터넷상의 여러 호스트들과 네트워크의 다양한 특성들을 적절히 수용하고 이들의 확장성을 높일 수 있는 인터넷 신호 메커니즘이다[1, 2].

그렇지만 IntServ의 문제점은 확장성에 있다. 네트워크 노

† 준 회 원 : 한양대학교 대학원 전자공학과
†† 정 회 원 : 한국전자통신연구원 보안운영체제연구팀장(선임연구원)
††† 정 회 원 : 한양대학교 정보통신학부 교수
논문접수 : 2001년 8월 1일, 심사완료 : 2001년 9월 28일

드 수가 많은 규모가 큰 경우에는 자원 예약을 위한 절차 및 과정에 오버헤드가 많고, 효율성 문제가 발생한다. IETF에서는 이 문제를 해결하기 위하여 DiffServ 모델을 제안하였다[3].

DiffServ는 큰 규모의 망을 대상으로 고안되었으며, 구조는 흐름의 집합을 한 단위로 양종단간에 신호 프로토콜을 사용하지 않고 차별화된 서비스를 제공한다. 국내와 전세계적으로 DiffServ의 도입이 이루어지고 있고, 여러 부문에서 DiffServ를 이용한 서비스가 추진되고 있는 점을 감안할 때 IntServ와 DiffServ 혼합 모델은 VoIP나 멀티미디어 서비스와 같은 실시간 서비스의 QoS의 향상을 위해서 상당한 효과를 가져올 수 있을 것이다. 제안된 IntServ 와 DiffServ의 두 모델을 사용한 End-to-End QoS의 3가지 서비스 모델을 시뮬레이션 하여 성능을 분석하였다[4].

실시간 서비스의 end-to-end QoS를 위한 서비스 요금을 계산할 빌링 시스템의 개발도 중요한 문제이다. 본 논문에서는 이러한 문제를 해결하기 위하여 RADIUS 프로토콜[5]의 Accounting, Authentication, Authorization (AAA) 기능을 이용한 네트워크 상태에 따른 IntServ와 DiffServ의 혼합 모델을 사용한 빌링시스템을 제공하였다. 제안된 네트워크 상태에 따른 동적인 요금 결정 방식은 네트워크 사용량 단위 시간당 변화 요소와 제안된 세 가지 서비스 모델(Best, Good, Default)중 하나의 선택된 서비스와 실제 패킷 사용량의 곱을 통하여 요금을 결정한다. 요금 정책을 위하여 사용자와 호스트에서의 RADIUS 서버와 에이전트 사이에 연결 설정 방법을 제시하였다. 제안한 VoIP을 위한 효율적인 빌링시스템을 위하여 Metering Policy, Collecting Policy, Accounting Policy, Charging Policy와 최종적으로 Billing Policy를 이용한 계층적인 Policy들의 관리를 사용하였다[6].

RADIUS 프로토콜의 감사 기능이 부족하여, 본 논문에서는 IEEE POSIX.1E 표준안[7,8]에 의한 감사 추적 시스템을 설계 및 구현하였다. IEEE POSIX.1E 표준안의 감사 이벤트에는 시스템 관련 이벤트들과 응용 프로그램 관련 이벤트들로 크게 나눌 수 있다. 시스템 관련 이벤트에는 시스템에서 프로세스가 커널 영역에 접근할때 사용하는 시스템 호출 함수들에 대한 정보를 기록하는 것을 말한다. 응용 프로그램 관련 기록에는 사용자 영역에서 발생하는 로그온, 로그온 - ID, 서비스 종류, 디바이스 ID, 사용량, 자원 할당, 호(call) 지속시간, 대역폭 접근, 호(call) 시작시간, 거리, 호의 수(number of calls)에 관련한 보안 관련 이벤트들을 기록하는 것을 말한다. 감사 추적 레코드를 형성하기 위해서 감사 추적 이벤트를 객체, 주체, 헤더, 이벤트 정보로 나누어서 해당항목들을 감사 추적 레코드에 추가하는 방식을 사용하였다.

본 논문에서는 2장에서 제안한 IntServ와 DiffServ 혼합 모델에서 QoS 서비스를 기술하고, 3장에서 QoS 서비스의 빌링 시스템을 제안하고, 4장에서 안전한 빌링 시스템을 위한 감사 추적 시스템에 대하여 기술하였다. 5장에서는 제안된 QoS 서비스 모델과 성능 분석에 대하여 기술하고, 6장에서는 결론 및 향후 연구과제를 제시한다.

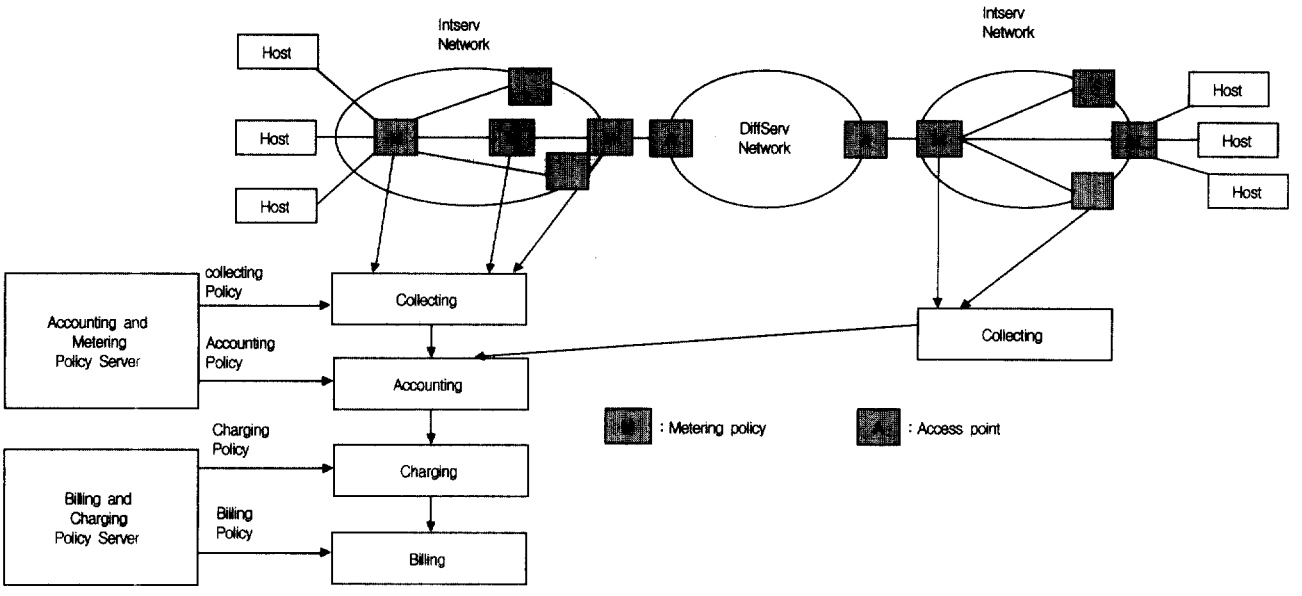
2. 제안된 IntServ와 DiffServ 혼합 모델에서 QoS 서비스

IntServ 구조의 핵심은 RSVP 신호 프로토콜을 이용하여 양 종단(End-to-End) 전 경로에 걸친 네트워크 자원(버퍼 또는 대역폭)을 사전에 예약하는 기능에 있다. 이 구조에서 자원의 예약은 개개의 패킷 흐름별로 이루어지므로 큰 규모의 망에 적용하기에는 확장성의 문제가 있다. 하지만, 이 구조는 규모가 작은 망에서는 각 흐름마다 요구하는 QoS를 보장할 수 있는 능력을 갖는다. 반면에, 큰 규모의 망을 대상으로 고안된 DiffServ 구조는 흐름의 집합을 한 단위로 양 종단간에 신호 프로토콜을 사용하지 않고 차별 화된 서비스를 제공한다.

이와 같이 볼 때, IntServ 구조를 갖는 망과 DiffServ 구조를 갖는 망은 (그림 1)와 같이 상호 보완하여 연동될 수도 있다. (그림 1)은 소규모 망에서는 IntServ/RSVP 구조가 적용되고, 다수의 ISP 망으로 구성될 수 있는 인터넷 백본망에서는 DiffServ 구조를 적용함으로써, 양종단의 호스트간에 QoS의 제공이 가능함을 나타낸다. 이 구조에서 DiffServ 망은 전달 망으로, IntServ 망은 DiffServ망의 사용자의 관계를 갖는 것으로 가정한다.

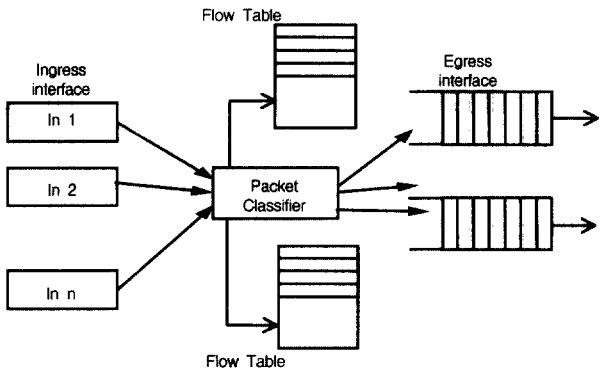
두 망의 구체적 연동에서는 다음과 같은 세부 과제들이 고려하였다. 호스트에서 발생된 RSVP 메시지가 어떻게 IntServ망과 DiffServ망을 통한 전달 과정, 양쪽 망에서 연결 수락 제어의 동작 구성 방법, IntServ망에서 정의된 서비스 유형의 패킷 흐름을 DiffServ 망의 PHB에 매핑 문제 및 이에 따른 IntServ망과 DiffServ 망의 정적 혹은 동적인 서비스 기준 협상 문제, 그리고 호스트에서 RSVP 신호 기능과 DiffServ 바이트 기록 기능 지원 문제 등을 고려하였다.

본 논문이 제안한 VoIP 서비스를 위한 delay sensitive 트래픽 특성과 넓은 대역폭 할당 특성을 가지는 트래픽을 위하여 IntServ와 DiffServ의 혼합 모델을 사용한 End-to-End QoS 서비스의 제공 방안은 다음과 같다. DiffServ영역에서는 IntServ의 RSVP 예약 요구가 명백히 실행되며, policy-based PHBs 제공을 통하여 QoS가 제공된다. IntServ영역에서는 RSVP를 사용한 QoS를 제공한다. IntServ에서는 표준인 Class Based Queue(CBQ)를 사용하였다.



(그림 1) 계층화된 빌링 정책들과 IntServ와 DiffServ의 전체 구성도

CBQ는 같은 물리적인 링크를 사용해서 에이전시 사이에 링크 공유를 제공하는 스케줄링 메커니즘이다. RSVP는 크게 패킷 분류기와 패킷 스케줄러로 나눌 수 있다. 우선 CBQ의 패킷 분류기를 이용하여, 본 논문에서 제공하고자 하는 real-time 서비스를 위한 delay sensitive 트래픽 특성과 넓은 대역폭 할당 트래픽을 위하여, Best, Good 서비스에 우선 순위를 부여하여 패킷을 분류한다. 패킷 scheduler에서는 패킷 분류기에 의하여 우선 순위가 부여된 패킷들에 대하여 우선 순위를 가지는 scheduling을 실행한다. 다음 (그림 2)는 RSVP 내부에서 사용하는 CBQ의 패킷 분류기이다.



(그림 2) CBQ의 패킷 분류기

DiffServ 영역에서는 IntServ의 RSVP 예약 요구가 명백히 실행되며, policy-based PHBs 제공을 통하여 QoS가 제공하였다. IP 헤더의 TOS 비트를 이용하여, 이 필드에서 6비트의 DSCP에 의하여 3가지 클래스(Best, Good, Default Service Class)를 제안하고자 한다. <표 1>에서는 TOS 비트의 PHB와 본 논문에서 제안하는 DSCP 비트 설정 방식을 나타내었다.

<표 1> PHB와 제안한 3가지 서비스를 위한 DSCP

PHB	Recommended DSCP Value		
Default	000000		
Class Selector	XXX000(X=0 or 1)		
EF	101110		
AF			
Drop Prob	Class 1	Class 2	Class 3
Low	001010	010010	011010
Medium	001100	010100	011100
High	001110	010110	011110

3. 제안된 QoS 서비스의 빌링 시스템

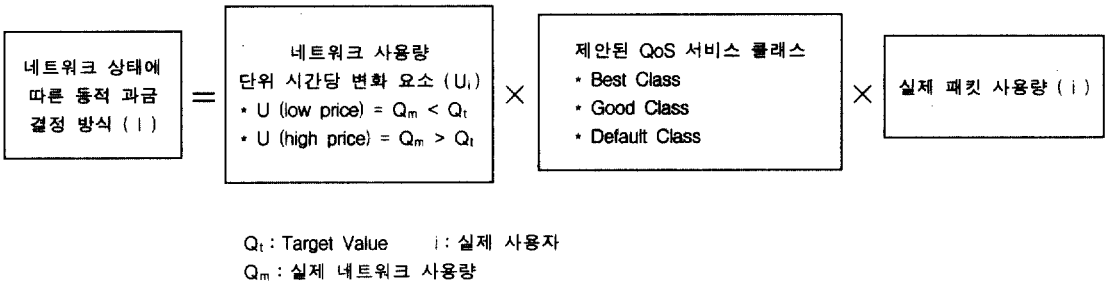
3.1 네트워크 상태에 따른 동적인 요금 결정 방식

네트워크 상태에 따른 동적인 요금 결정 방식은 네트워크 자원의 사용량이나, 사용자들의 요구에 따라서 가격이 달라진다. 네트워크 가격 결정 방식은 실제적인 대역폭의 요구를 고려하여 가격을 결정한다. 가격은 실제적인 네트워크 용량(K) 보다 실제적인 대역폭 사용량이 작을 때 가격이 상승하며, 반대로 실제적인 대역폭이 클 때 가격이 하락하는 방식을 제안하고자 한다. 사용량 중심적 요금 부과 방식은 다음과 같은 식으로 표현될 수 있다[9-11].

가격 : p, 사용량 변수 : U

네트워크 상태에 따른 동적인 요금 결정 방식 = $\sum_{i=0} p_i U_i$ 로 표시된다.

가격 결정 방식은 효과적인 네트워크 자원 사용을 위하여, 다양한 사용자들의 개인적인 트래픽양을 측정하며, 네트워크 자원의 상태에 따라서 가격을 결정하는 방식을 제안하였다



(그림 3) 네크워크 상태에 따른 동적인 요금 결정 방식

[12]. 본 논문이 제안한 네트워크 상태에 따른 동적인 요금 결정 방식은 (그림 3)과 같다. 우선 네트워크 사용량의 단위 시간당 변화 요소를 이용하여 네트워크 사용이 많을 때에 적을 때를 구분하여 차등 요금을 제공하고자 한다. 제안된 QoS 서비스 클래스는 사용자의 사용한 서비스 등급에 따라 차등 요금을 부과하고자 하며, 마지막으로 실제 패킷 사용량을 계산하여 요금을 결정하고자 한다.

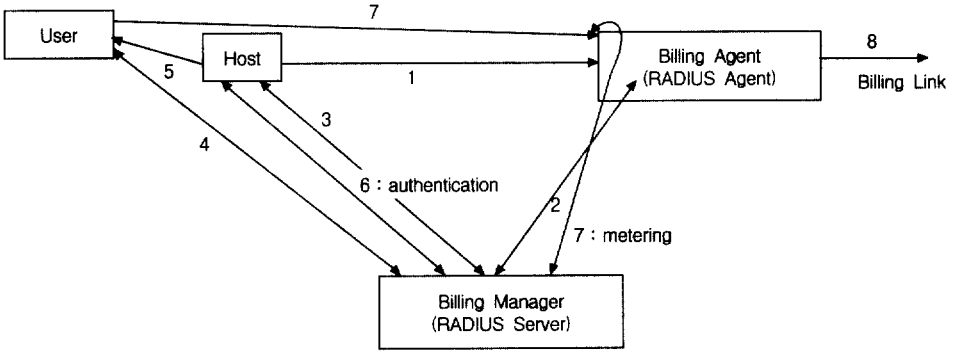
3.2 RADIUS Agent와 Server간의 연결 설정 방안

IntServ망의 끝단에 빌링 시스템을 구현하였으며, (그림 1)에 나타나 있다. 다음 (그림 4)는 초기 연결 설정 방식에 대한 빌링 시스템의 구성요소이며, 숫자는 연결 순서를 나타낸다.

빌링 시스템의 구성은 다음과 같은 3부분으로 이루어져 있다. User, Billing Agent & Sever, Internet Connectivity로 이루어져 있다. Billing Agent 기능에는 동적인 요금을 부과하는 모듈이 구현되어있다. 본 시나리오에서는 3단계의 서비스를 제공하는데, Best, Good, Default 서비스로 부른다. 단위 시간당 패킷 전송수를 계산하여, RADIUS Agent가 RADIUS Sever에게 정보를 제공한다.

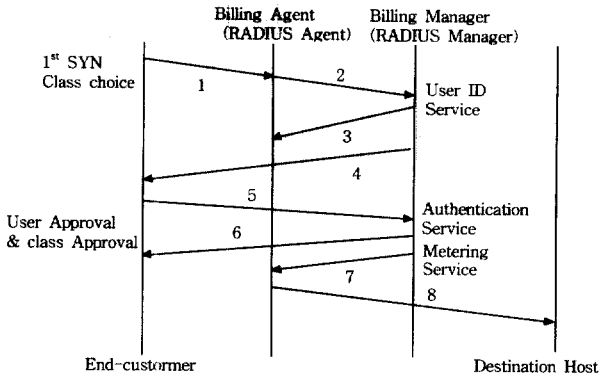
사용자가 인터넷을 연결하고자 TCP 연결을 설정하면, RADIUS Agent가 사용자의 신원을 확인하는 과정을 거친다. 사용자가 연결이 승인되고 난 후에 요금을 지불을 준비가 되어 있으면, 정상적인 연결이 설정되고, RADIUS Agent가 패킷 사용량을 계산하기 시작한다. 제안된 빌링 시스템은 각 사용자에 대하여 요금을 부과한다. 사용자가 사용한 만큼의 요금을 지불하는 것을 승인해야 한다. 따라서 정보 보안 정책이 필요하게 되는데, 본 시스템은 정확하고, 신뢰할 수 있는 온라인으로 사용자와 RADIUS Sever 사이에 피드백이 존재한다. RADIUS Sever에는 감사 추적 시스템을 구현하여, 감사 추적 이벤트 관련 정보들을 감사 추적하여 로그 기록으로 생성한다. 다음 (그림 5)는 사용자가 실제적으로 제안된 빌링 시스템을 사용시의 연결 설정 방법이다.

1. 호스트는 TCP SYN 메시지를 보내서, 연결을 시작한다. Billing Agent는 TCP SYN 메시지를 새로운 네트워크 설정으로 인식한다. 연결은 항상 고유 ID를 발급한다.[source (address, port), destination (address, port)]
2. Billing Agent는 Billing Sever가 연결을 허락할 것인가를 위한 접촉을 한다.
3. Billing Sever는 RADIUS을 이용하여, 사용자를 인증한다.
4. Billing Sever는 사용자에게 이 연결을 설정하면 요금을 지불할 것인가 확인 과정을 거친다.
5. 호스트는 RADIUS을 이용하여, Billing Sever의 인증에 응답을 한다.
6. 사용자가 이 연결에 대하여 요금을 지불할 것인가를 승인하면, Billing Sever가 RADIUS을 이용하여 인증과정을 거친다.
7. Billing Sever는 Billing Agent에게 연결된 링크를 metering 한다.



(그림 4) RADIUS 연결 설정을 위한 빌링 시스템의 구성 요소

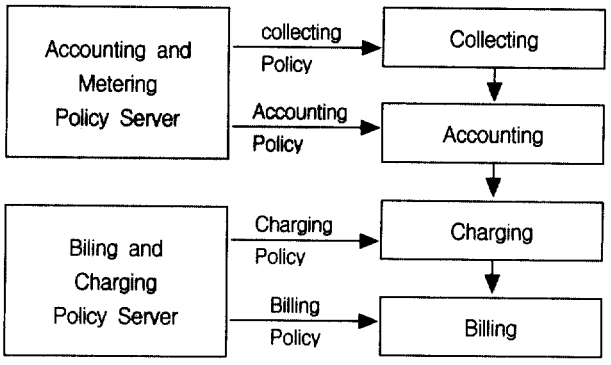
8. Billing Agent가 연결 테이블 안에 연결에 대한 연결 정보를 만들어서, 외부 호스트에 대하여 TCP SYN 메시지를 보낸다.



(그림 5) RADIUS Agent, Sever 사이의 연결 설정 방식

3.3 효율적인 빌링시스템을 위한 계층적인 Policy의 관계

효율적인 빌링시스템을 위하여 계층적인 Policy 관계를 (그림 6)에 나타냈다. Metering Policy는 트래픽의 흐름에 대하여 기록, 관리하는데 사용된다. Collecting Policy는 Metering Policy로 모여진 데이터를 수집하는 Policy를 말하며, 이러한 데이터를 상위 Policy인 Accounting Policy에 제공하는 역할을 한다. Accounting Policy는 Collecting Policy로 모여진 데이터를 가지고 네트워크 회계 데이터를 작성한다. 또한 Charging Policy에 데이터를 제공하여, 요금 결정 방식을 정하는데 데이터로 사용된다. Billing Policy는 Charging Policy에서 얻어진 데이터를 가지고, 각 사용자에게 요금을 부과하는 Policy를 담당한다.



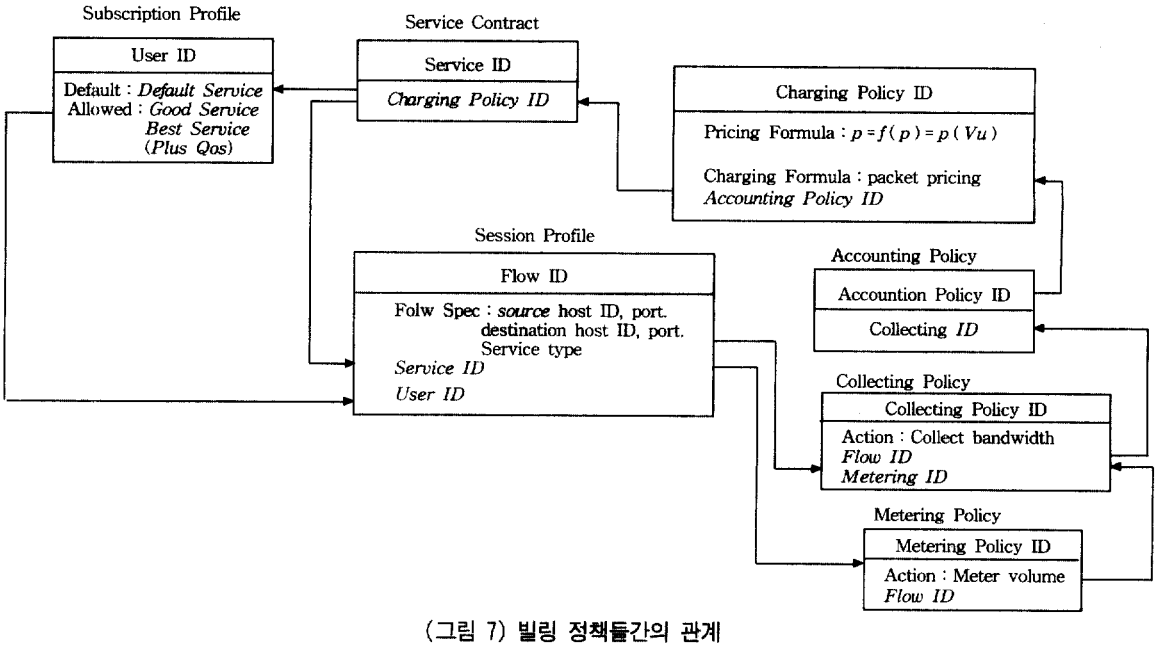
(그림 6) 효율적인 빌링시스템을 위한 계층적인Policies

다음 (그림 7)은 Policy 사이의 관계를 나타낸다. 본 논문에서 사용자는 3가지 서비스(Best, Good, Default service class)를 이용할 수 있다. 각 서비스는 Charging, Accounting, Collecting, Metering Policy와 관련이 있으며, Service Level Agreement(SLA) 또는 사용자 서비스 요구에 따라서 설정된다. (그림 7)에서 이탤릭 문자는 데이터 기록이 서로 연관됨을 나타낸다. 예를 들면, Charging Policy ID에서는 Accounting Policy ID와 서로 연관됨을 나타낸다. 다음 Policy들은 RADIUS 서버에 의하여 관리된다. RADIUS의 인증 과정에 의하여 초기 연결 설정시 User ID 와 Service ID 에 따라서, 연결을 허락할 것인지 아닌지가 결정이 된다.

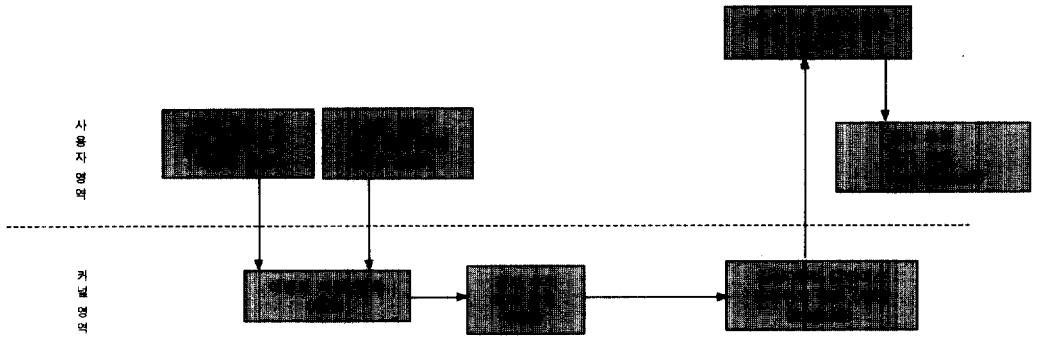
4. 안전한 빌링 시스템을 위한 감사 추적 시스템

4.1 감사 추적 시스템의 전체 구조

본 논문에서 사용한 RADIUS 프로토콜의 보안 기능으로



(그림 7) 빌링 정책들간의 관계



(그림 8) RADIUS 서버에서 감사 추적 시스템

서 감사 기능이 취약한 관계로 인하여, IEEE POSIX.1E 표준안에 의한 감사 추적 시스템을 설계 및 구현하였다. IEEE POSIX.1E 표준안의 감사 이벤트에는 시스템 호출 관련 이벤트들과 응용 프로그램 관련 이벤트들로 크게 나눌 수 있다. 시스템 호출 관련 이벤트에는 시스템에서 프로세스가 커널 영역에 접근할 때 사용하는 시스템 호출 함수들에 대한 정보를 기록하는 것을 말한다. 응용 프로그램 관련 기록에는 사용자 영역에서 발생하는 로그온, 로그온 - ID, 서비스 종류, 디바이스 ID, 사용량, 자원 할당, 호(call) 지속시간, 대역폭 접근, 호(call) 시작시간, 거리, 호의수(number of calls)에 관련한 보안 관련 이벤트들을 기록하는 것을 말한다[13, 14].

본 시스템은 감사 추적 이벤트 체킹 블록, 감사 추적 버퍼 블록, 감사 추적 디바이스 드라이버 블록, 감사 추적 데몬 블록을 사용하여 수행된다. 다음 (그림 8)은 본 논문에서 구현한 감사 추적 시스템에 전체 구성도이다. RADIUS 서버에서 시스템 호출 관련 이벤트나 응용 프로그램 관련 이벤트 발생 시 감사 추적 시스템에서 진행 과정을 나타낸다.

감사 추적 레코드를 형성하기 위해서 감사 추적 이벤트를 객체, 주체, 헤더, 이벤트 정보로 나누어서 해당항목들을 감사 추적 레코드에 추가하는 방식을 사용하였다. 다음 <표 2>, <표 3>, <표 4>, <표 5>는 감사 추적 이벤트를 위한 객체, 주체, 헤더, 이벤트 정보에 대한 항목들을 설명하였다.

〈표 2〉 감사 추적 헤더 정보

종 류	설 명	item_id
AUD_TYPE_SHORT	감사 레코드의 형식	AUD_FORMAT
AUD_TYPE_SHORT	레코드의 버전 번호	AUD_VERSION
AUD_TYPE_AUD_ID	프로세스의 감사 ID	AUD_AUD_ID
AUD_TYPE_INT	레코드의 이벤트 종류	AUD_EVENT_TYPE
AUD_TYPE_STRING	레코드의 이벤트 종류	AUD_EVENT_TYPE
AUD_TYPE_AUD_TIME	이벤트가 발생한 시간	AUD_TIME
AUD_TYPE_AUD_STATUS	이벤트의 감사 상태	AUD_STATUS
AUD_TYPE_INT	이벤트를 위해 반환된 값	AUD_ERRNO

〈표 3〉 감사 추적 객체 정보

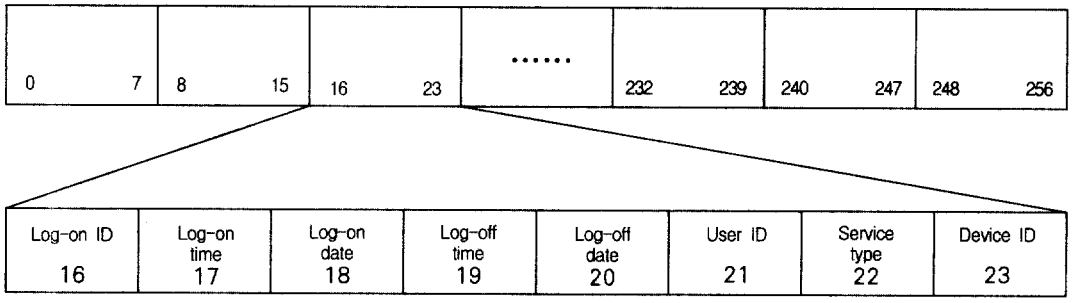
종 류	설 명	item_id
AUD_TYPE_AUD_OBJ_TYPE	객체의 종류	AUD_TYPE
AUD_TYPE_UID	객체사용자의사용자ID	AUD_UID
AUD_TYPE_GID	객체 사용자의 그룹ID	AUD_GID
AUD_TYPE_MODE	객체의 모든 비트들	AUD_MODE
AUD_TYPE_STRING	객체의 이름	AUD_NAME

〈표 4〉 감사 추적 주체 정보

종 류	설 명	item_id
AUD_TYPE_PID	프로세스 ID	AUD_PID
AUD_TYPE_UID	실제적인 사용자 ID	AUD_EUID
AUD_TYPE_GID	실제적인 그룹 ID	AUD_EGID
AUD_TYPE_GID	보충적인 그룹 ID	AUD_SGIDS
AUD_TYPE_UID	실제 사용자 ID	AUD_RUID
AUD_TYPE_GID	실제 그룹 ID	AUD_RGID

〈표 5〉 감사 추적 이벤트 정보

종 류	설 명	item_id
AUD_TYPE_STRING	path의 이름	AUD_PATHNAME
AUD_TYPE_INT	파일 서술자	AUD_FILE_ID
AUD_TYPE_AUDIT_STATE	감사 추적 상태	AUD_AUDIT_STATE
AUD_TYPE_INT	반환된 PID/FD	AUD_RETURN_ID
AUD_TYPE_STRING_ARRAY	명령어의 인자들	AUD_CMD_ARGS
AUD_TYPE_STRING_ARRAY	환경 인자들	AUD_ENVP
AUD_TYPE_UID	실제 UID	AUD_UID_ID
AUD_TYPE_GID	실제 GID	AUD_GID_ID
AUD_TYPE_INT	프로세스 exit 코드	AUD_EXIT_CODE
AUD_TYPE_INT	Signal 번호	AUD_SIG
AUD_TYPE_STRING	링크의 이름	AUD_LINKNAME
AUD_TYPE_INT	플래그들을 열기	AUD_OFLAG
AUD_TYPE_INT	파일 ID 읽기	AUD_RD_FILE_ID
AUD_TYPE_INT	파일 ID 쓰기	AUD_WD_FILE_ID
AUD_TYPE_STRING	과거 path의 이름	AUD_OLD_PATHNAME
AUD_TYPE_STRING	현재 path의 이름	AUD_NEW_PATHNAME
AUD_TYPE_TIME	접근 시간	AUD_ATIME
AUD_TYPE_TIME	변경 시간	AUD_MTIME



(그림 9) 감사 추적 시스템에서 이벤트 마스크

감사추적 레코드를 형성 할 때에는 우선 <표 2>의 헤더에 있는 모든 항목들을 포함한다. 나머지 주체, 객체, 이벤트 정보의 항목들을 이벤트가 필요한 항목에 대해서만 선택적으로 첨가한다. 이 사항에 대한 IEEE POSIX.1E에서는 발생한 시스템 호출 관련 이벤트에 대하여 최소 항목들을 정의해 주었다. 예를 들면 chdir 시스템 호출이 발생 시에 <표 5>에 있는 이벤트 항목중 item_id가 AUD_PATHNAME을 최소 포함 항목으로 정의하였다.

4.2 감사 추적 시스템 블록 기능

4.2.1 이벤트 체킹 블록(Event Checking Block : ECB)

이벤트 체킹 블록에서는 256 비트로 이루어진 데이터 구조를 구진 이벤트 체킹 마스크를 가진다. 각 1비트마다 할당된 감사 추적 이벤트가 있으며, 해당된 감사 추적 이벤트가 ON으로 설정되면, 그 이벤트는 감사 추적 레코드가 로그로 기록된다. 다음 (그림 9)는 감사 추적 시스템에서 이벤트 체킹을 위한 마스크를 나타낸다.

- 입력
  - a. 시스템 호출 관련 이벤트가 발생한다.
  - b. 응용프로그램 관련 이벤트가 발생한다.
- 처리
  - a. 발생한 시스템 호출이 감사 추적 이벤트와 관련이 있는지를 검사한다.
  - b. 발생한 응용 프로그램 관련 이벤트가 이벤트 마스크와 관련이 있는지를 검사한다.
- 출력
  - a. 발생한 시스템 호출이 감사 추적 이벤트와 관련이 없으면, 0을 반환한다.
  - b. 발생한 시스템 호출이 감사 추적 이벤트와 관련이 있으면, 해당되는 번호를 반환한다.
  - c. 발생한 응용 프로그램 관련 이벤트가 이벤트 마스크와 관련이 없으면, 0을 반환한다.
  - d. 발생한 응용 프로그램 관련 이벤트가 이벤트 마스크와 관련이 있으면, 해당되는 번호를 반환한다.

4.2.2 감사 추적 버퍼 블록(Audit Trail Buffer Block : ABB)

감사 추적 관련 이벤트는 로그 파일 형태로 쓰여진다. 감사 추적 관련 데이터를 직접 하드디스크에 저장시키면, 현저한 시스템 성능 저하 현상을 일으킨다. 시스템의 효율성을 위하여, 커널 내에 감사 추적용 버퍼를 사용하여 감사 추적 레코드를 저장한다.

- 입력
  - a. 감사 추적 이벤트 체킹 블록으로부터, 해당되는 감사 추적 이벤트의 번호를 받는다.
- 처리
  - a. 해당되는 감사 추적 이벤트를 위한 버퍼를 할당한다.
- 출력
  - a. 감사 추적 디바이스 드라이버의 레코드 요청에 따라 자료를 출력한다.

4.2.3 감사 추적 디바이스 드라이버 블록(Audit Trail Device Driver Block : ADB)

감사 추적 유틸리티들이 직접 커널 내에 위치한 감사 추적 버퍼에 접근할 수 없다. 따라서 본 감사 추적 서브 시스템에서는 외부 프로세스와의 소통점으로서 디바이스를 생성하고, 해당 디바이스 드라이버를 제공하여, 외부 프로세스가 감사 추적 데몬을 이용하여, 감사 추적 이벤트 정보를 읽어 낼 수 있다.

- 입력
  - a. 감사 추적 데몬 블록으로 감사 추적 레코드 읽기 명령이 수행된다.
- 처리
  - a. 해당 디바이스 dev/audit로부터 명령을 수행한다.
- 출력
  - a. 외부 프로세스가 감사 추적 데몬을 이용하여, 감사 추

적 이벤트를 정보를 읽어 낸다.

4.2.4 감사 추적 데몬 블록(Audit Trail Demon Block : ATB)

감사 추적 데몬은 커널 내에 위치한 감사 추적 로그의 임시 기억 장소에 저장된 감사 추적 이벤트 관련 레코드 정보를 접근해야 한다. 그러나 커널 내부에 직접 접근할 수 없기 때문에 감사 추적 디바이스 드라이버(dev/audit)를 통해 감사 추적 이벤트 관련 레코드를 읽어 로그 파일에 저장한다.

- 입력
  a. 사용자 프로세스가 감사 추적 이벤트 레코드를 요청한다.
- 처리
  a. 감사 추적 디바이스 드라이버로부터 감사 추적 이벤트 관련 레코드를 읽어 온다.
- 출력
  a. 감사 추적 디바이스 드라이버(dev/audit)를 통해 감사 추적 이벤트 관련 레코드를 읽어 로그 파일에 저장한다.

4.3 감사 추적 시스템의 커널 설정 및 초기화 과정

감사 추적 서브 시스템을 위해서는 커널 부분에 kernel_audit.c 을 첨가하였으며, kernel_audit.c 에는 /dev/audit의 디바이스 드라이버 코드와 시스템 호출 구현 코드, 사용자 영역에 감사 추적 레코드들 전달하는 코드, 커널에서 감사 추적 이벤트 데이터 형성 코드로 구성된다. 또한 커널에 audit.h 파일을 첨가하였으며, audit.h파일에는 감사 추적 이벤트와 인터페이스 변수 설정, 감사 추적 데이터 종류 변수 설정, 감사 추적 라이브러리에서 사용될 내부 데이터 구조를 정의하였다.

이러한 커널 영역에 코드 첨가로 인하여 커널 패치 작업을 하였다. 커널을 설정한 후 빌드 과정을 거친 후에 새로운 커널을 형성하였다. 새로운 시스템 호출 함수의 첨가로 인하여, libc 라이브러리도 다시 빌드하였다. 감사 추적 코드들 위한 기능들을 지원하기 위한 POSIX.1e 라이브러리도 빌드하였다.

POSIX.1e 라이브러리를 의존하는 감사 추적 데몬과 감사 추적 명령어는 시스템 초기화 파일을 설정하였다. 감사 추적 디바이스 드라이버인 /dev/audit을 위하여, /dev/MAKEDEV 을 위한 패치를 하였으며, 이를 위하여, mknod audit c 120 0 ; chmod 400 audit를 사용하였다. 이 디바이스를 위한 파일 시스템이 포함되며, 따라서 /dev/audit가 자동적으로 나타난다. 시스템 초기화 파일에 감사 추적 데몬을 부팅 시에 활성화시킨다.(auditd : audit trail daemon)

```
# Turn audit trail on.
if [ -x /sbin/auditd ]
then
        /sbin/auditd /var/log/audit
        echo "Auidt Trail daemon turned on."
fi
```

시스템이 부팅 된 후부터 감사 추적시스템이 작동된다. 감사 추적시스템을 정지하고 싶으면, 감사 추적 데몬을 종료하는 명령어를 사용한다. 감사 추적 로그 파일은 /var/log/audit 에 디폴트로 저장되어 진다. 감사 추적 로그 파일은 보안 관리자의 소유이며, 읽기/쓰기 권한이 주어지며, 다른 일반 사용자에게 읽기 권한만 주어진다.

5. 제안된 QoS 서비스 모델과 성능 분석

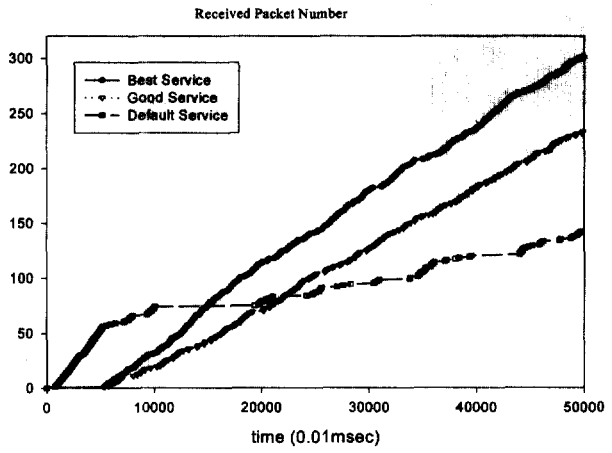
본 논문에서는 Best, Good, Default 서비스인 3개의 서비스 등급을 가진다. 다음 <표 6>은 제시한 3개의 서비스 등급에 대한 내용을 설명한 것이다. Best 서비스는 최우선 순위의 QoS 서비스이며, 특징은 Real-time 트래픽 특성에 맞도록 delay sensitive 트래픽의 특성을 가지며 더 넓은 대역폭 할당을 기능을 가진다. 서비스 요금은 가장 비싸다. Good 서비스는 Best 서비스에 비하여 우선 순위는 낮지만, Real-time 트래픽의 특성에 맞게 delay sensitive 트래픽의 특성을 가지며 더 넓은 대역폭 할당을 가지며, 가격은 Best 서비스보다는 싸다. Default Service는 현재 인터넷에서 서비스되고 있는 Best effort 서비스 형태와 비슷하며, 가장 낮은 서비스 요금이다.

〈표 6〉 제안된 서비스 클래스

	Best Service Class	Good Service Class	Default Service Class
Workload	Realtime traffic	Realtime Traffic	FTP, Telnet Simple Traffic
Priority	Highest	Higher	Low
Pricing	Highest	Higher	Low

(그림 10)에서는 제안된 Best, Good, Default 서비스의 트래픽 특성을 시간에 따라 Source 와 Destination사이의 데이터 전송을 나타내며, NS[15]를 이용하여 시뮬레이션을 하였다. (그림 10)에서 Best, Good, Default 서비스들은 각 패킷들이 도착된 시점을 나타낸다. 현재 인터넷 서비스의 Best-effort 형태인 Default Service(□)는 도착 간격이 일정치 않다는 것을 알 수가 있다. 즉 Delay에 민감하지 않은 트래픽의 특성을 나타낸다. 그렇지만 본 논문이 제안한 Best, Good 서비스는 VoIP와 같은 Real-time 서비스를 위하여 delay sen-

sitive 트래픽의 특성을 가지며 더 넓은 대역폭 할당을 가진다는 것을 알 수 있다.



(그림 10) 제안된 3가지 서비스 모델의 성능 분석

6. 결론 및 향후 연구 과제

본 논문에서는 제안한 것은 크게 세 가지로 나눌 수 있다. 첫째 VoIP와 같은 real-time 서비스를 위한 delay sensitive 트래픽 특성과 넓은 대역폭 할당 트래픽의 QoS 서비스 제공하기 위해서 IntServ와 DiffServ의 혼합 모델을 사용한 End-to-End QoS 서비스를 제공하였다. 둘째 제안된 End-to-End QoS 서비스의 요금을 결정하기 위하여, RADIUS 프로토콜의 Accounting, Authentication, Authorization(AAA) 기능을 이용하여 IntServ와 DiffServ의 혼합 모델을 사용한 빌링시스템을 제안하였다. 셋째 제안된 빌링 시스템의 안전성을 위하여, RADIUS 프로토콜의 감사 기능을 IEEE POSIX.1E 표준안에 의거해 감사 추적 시스템을 설계 및 구현하였다.

또한 IEEE POSIX.1E 표준안에 의해 구현한 감사 추적 서비스는 성능에 대한 부하문제가 발생하는데, 이를 위하여 불필요한 감사 추적 이벤트 제거 및 감사 추적 시스템의 성능의 효율성 증대의 문제를 향후 연구과제로서 해결하고자 한다. IETF의 로밍 작업 그룹에서 RADIUS가 inter-domain간에 적당하지 않다고 결론을 내려서, Diameter가 이러한 RADIUS의 알려진 결점들을 해결해 주고, NASREQ, ROAMOPS와 Mobile IP 응용 분야에서 점차로 이용이 증가할 추세이기 때문에, RADIUS에서 Diameter로의 변환을 추진할 것이다.

참 고 문 헌

[1] Seaman, M., Smith, A., Crawley, E. and J. Wroclawski, "Intergrated Service Mapping on IEEE 802 Networks," RFC 2815, May. 2000.

[2] J. Wroclaswski, "The Use of RSVP with IETF Integrated Services," RFC 2210, September, 1997.

[3] S. Blake, et al., "An Architecture for Differentiated Services," RFC 2475, December, 1998.

[4] Y. Bernet, P. Ford, R. Yavatkar and et al., "A Framework for Integrated Services Operation over Diffserv Networks," RFC 2998, November, 2000.

[5] Rigney, C., Rubens, A., Simpson, W. and S. Willens, "Remote Authentication Dial In User Service (RADIUS)," RFC 2138, April, 1997.

[6] Rinney, C., Willats, W. and Calhoun, P., "RADIUS Extensions," RFC 2869, June, 2000.

[7] IEEE Std 1003.1e - Draft standard for Information Technology-Portable Operating System Interface(POSIX) Part 1 : System Application Program Interface(API)- Protection, Audit and Control Interfaces.

[8] IEEE Std 1003.2c - Draft standard for Information Technology-Portable Operating System Interface(POSIX) Part 2 : Shell and Utilities : Protection and Control Interfaces.

[9] C. Mills, D. Hirsch and G. Ruth, "Internet Accounting : Background," RFC 1272, 1991.

[10] H. W. Braun, K.C. Claffy and G.C. Polyzos, "A Framework for Flow-Based Accounting on the Internet," Proc. of the Singapore intl Conference on Networks, 1993.

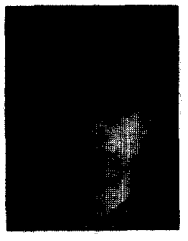
[11] M. Honig and K. Steiglitz, "Usage-based Pricing of Packet Data Generated by a Heterogeneous User Population," Proc. IEEE INFOCOM, Vol.2, Boston, MA, pp.867-874, Apr. 1995.

[12] J. Mackie-Mason and H. Varian, "Pricing Congestible Network Resource," IEEE JSAC, Vol.13, No.7, pp.1141-1148, Sept. 1995.

[13] Lunt, T. Automated Audit Trail Analysis for Introduction Detection, Computer Audit Update. pp.2-8, April, 1992.

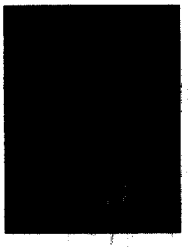
[14] National Computer Security Center. A Guide to Understanding Audit in Trusted Systems, NCSC-TG-001, Version-2. Ft. Meade, MD, 1988.

[15] The UCB/LBNL/VINT NetworkSimulator (NS). URL "http : //www-mash.cs.berkeley.edu/ns/."



박 우 출

e-mail : wcpark@ihanyang.ac.kr,
1995년 한양대학교 전자공학과 학사
1997년 한양대학교 전자공학과 석사
2002년 박사학위 취득 예정
관심분야 : 네트워크 프로토콜 설계 및 성능 분석, 빌링시스템



김 정 녀

e-mail : jnkim@etri.re.kr

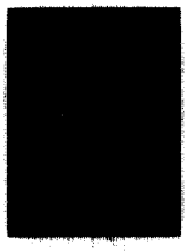
1987년 전남대학교 전산통계 학과 졸업 (학사)

1995년~1996년 Open Software Foundation Research Institute 공동연구 파견(미국)

2000년 충남대학교 대학원 컴퓨터공학과(석사)

1988년~현재 한국전자통신연구원 보안운영체제연구팀장(선임연구원)

관심분야 : 운영체제, 분산 처리, 고장 감내, 시스템 보안



이 병 호

e-mail : bhrhee@hanyang.ac.kr

1975년 한양대학교 전자공학과 학사

1977년 한양대학교 전자공학과 석사

1993년 일본 국립 지바대학 박사학위

1980~1981년 한국 전자통신연구원

현재 한양대학교 정보통신학부 교수

관심분야 : 컴퓨터 네트워크, 네트워크 보안, 신경회로망