2014년 12월 전자공학회 논문지 제 51권 제 12 호
Journal of The Institute of Electronics and Information Engineers Vol. 51, NO. 12, December 2014 http://dx.doi.org/10.5573/ieie.2014.51.12.204

논문 2014-51-12-22

MPLS망에서 Q-MOTP를 위한 자원 적응적 QoS 관리

(Resource Adaptive QoS Management for Q-MOTP in MPLS Network)

최 원 근*

(Won-keun Choi©)

요 약

본 연구는 멀티미디어 객체 스트림 전송 프로토콜을 위한 QoS지원 기법과 관리에 관한 연구를 수행하였다. 자원관리자는 시스템 자원정보를 QoS관리자에게 제공한다. QoS 및 시스템 자원의 2차원 정보를 가진 QoS 관리자는 자원 관리자와 서로 같은 시간 영역으로 강력하게 유기적으로 결합되어 효과적인 QoS관리를 수행한다. 그러므로 연결 수락에 대한 빠른 판단이 가능하고, 성능 변동(performance fluctuation)들을 실시간으로 감지하고 대응할 수 있다.

Abstract

This paper has described a QoS mechanism and management for multimedia object transport protocol. Resource manager reports the system resource information to the QoS manager. Based on the combination of two informations, the more effective QoS management can be achieved by organic combination between them on the protocol time domain. Therefore, it is possible for QoS manager to decide call admission control quickly and detect and react the performance fluctuations in protocol time domain.

Keywords : Multimedia, QoS, QoS관리자, CBQ, Q-CBQ, Q-MOTP

Ⅰ. 서 론

QoS 보장을 위해 인터넷은 DiffServ를 지원하는 MPLS[1]라우터를 백본 네트워크에 배치하여 QoS (quality of service)를 보장하고자 하는 노력이 진행되고 있고[2~3], QoS보장을 위한 또 다른 연구로서는 멀티미디어 스트림 자체 특성을 이용한 효율인 전송에 대한 연구이다[4~5].

* 정회원, 인하공업전문대학 정보통신과
(Dep. of Information & communication, Inha Technical College)
© Corresponding Author(E-mail: wkchoi@inhatc.ac.kr)
※ 본 논문은 2014년도 인하공업전문대학 교내 연구비 지원을 받아 수행되었습니다.
접수일자: 2014년11월06일, 수정일자: 2014년11월22일
게재확정: 2014년12월01일

연구[6~7]에서는 CBQ기법을 사용하는 MPLS 라우터에 QoS를 지원하는 기법(Q-CBQ)을 제안하고 성능을 평가한 연구가 진행되었고, 수정된 Q-CBQ기법을 사용하는 MPLS망에서 멀티미디어 객체 데이터를 효율적으로 전송하기 위한 프로토콜(Q-MOTP)에 대한 연구[8]와 기존 프로토콜들과의 성능을 비교하는 연구[9~10]를 수행하였다. 본 연구는 연구[8]에서 제안한 Q-MOTP에 대한 QoS를 지원하기 위한 연구로서 이전 연구[11]에서 제안한 자원관리자를 통해서 얻은 자원정보를 QoS 관리자가 갖게 함으로써, QoS 관리자는 QoS 정보 및 자원정보의 2개의 정보를 갖는다. 정보들을 이용해서 QoS 관리를 수행하게 함으로써 기존의 QoS정보만으로 QoS 관리를 수행하는 기존의 방법과 다른 기법을 사용하는 연구를 수행하였다.

연결설정 시 합의된 QoS 정보만으로 QoS를 관리하는 QoS 관리 기법들의 문제점들을 살펴보면 다음과 같다.

첫째 기존의 QoS 감쇠에 대한 대응은 실시간적 대응이라 보기 어렵다. 사용자가 QoS 위반을 알아내고 QoS 위반에 대한 조정을 요구한다. QoS 관리자가 프로토콜에게 QoS에 관련된 통계적 데이터를 수집하도록 요구하면, 프로토콜이 통계적 데이터를 수집하여 판단하고, 대응하게 되는데 이는 실시간적 대응이라 보기 어렵다.

둘째 기존의 QoS 관리기법에서는 QoS정보와 자원정보와의 관계를 고려하지 않았다. 예를 들어서 처리율이 합의된 QoS 수준이하로 떨어진 경우, 원인은 CPU나 대역폭의 과부하를 생각할 수 있으나 또 다른 QoS 감쇠현상들을 분석해서 정확한 원인을 찾아야 한다.

Ⅱ장에서는 설계된 구조에서 QoS 관리자가 수행하는 정적관리인 연결수락제어와 동적관리인 자원동조요청을 상세한 내용을 설명하고 마지막으로 Ⅲ장에서는 본 연구에 대한 결론 및 향후과제를 논할 것이다.

Ⅱ. QoS 관리

그림 1에서는 본 연구에서 설계한 QoS 관리시스템의 구조를 볼 수 있다.

제안된 QoS 관리구조는 Q-MOTP[8]와 자원관리자(RM: resource manager)[11] 그리고 QoS관리자(QM:QoS manager)로 구성된다. QoS관리자는 QoS의 규격, 번역, 협상, 관리 등에 관여한다. 특히 본 연구에서는 자원(들)의 과부하 상태가 보고되면, Q-MOTP에게 QoS 통계 데이터를 수집하도록 요구하여 사용자와 합의된 QoS값과 측정된 QoS값을 비교한다. QoS 위반이 인지되면 자원관리자에게 자원 동조를 요구하는 방법으로 QoS의 수준을 관리한다.

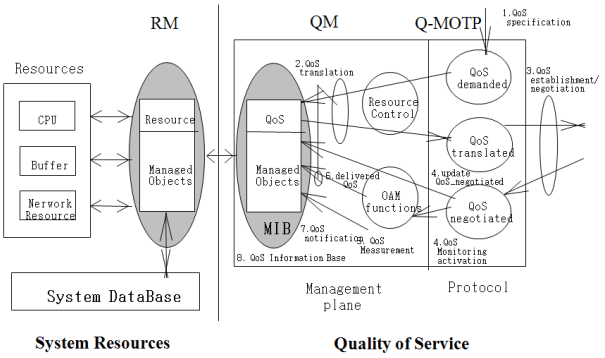

그림 1. QoS 관리구조
Fig. 1. QoS management structure.

QoS 관리관점에서 사건들의 발생은 합의된 QoS 위반으로 정의할 수 있다. 사건은 연결에 할당된 자원들의 고갈이나 병목으로 인해 발생하며 자원의 고갈이 QoS 위반보다 시간적으로 먼저 발생하게 된다. 예를 들어 낮은 처리율의 경우 CPU나 대역폭의 과부하가 이미 보고되어 있었다.

QoS관리는 일반적으로 정적관리와 동적관리 2가지이다. 정적관리에는 QoS 번역과 연결수락제어 등과 같은 관리들이며, 연결수락제어(call admission control)는 실시간 처리를 요구하지는 않지만 모든 시스템 자원들에게 영향을 미친다. 동적관리에는 QoS 감시(monitoring), 유지(maintenance), 재협상(renegotiation) 등과 같이 데이터 전송에 필요한 관리들이다. 자원동조요청(resource tuning request)은 모든 자원들에 관여하지는 않지만 실시간 처리를 요구한다.

본 연구에서는 자원관리자를 통해 자원정보를 갖게 된 QoS관리자는 2차원 정보를 이용해서 QoS 관리를 수행한다.

1. 연결 수락 제어

사용자 혹은 상위 계층으로부터 새로운 연결에 대한 요구가 있을 때, 연결수락제어는 기존의 연결들에 대해 영향을 미치지 않으면서 새로운 연결에 대한 허락 여부를 결정한다.

기존 연구들에서는 자원정보는 자원관리자만이 갖고 있으므로, 연결수락제어에 대한 판단을 자원관리자가 수행한다.

본 연구에서는 QoS관리자가 자원정보를 갖고 있으므로, 새로운 연결에 대한 연결수락 판단을 QoS관리자가 수행하도록 하였다.

사용자 혹은 상위계층으로부터 새로운 연결에 대한 요구가 있을 때 QoS관리자는 시스템 매개변수로 변환된 QoS 매개 변수들을 갖고, 다음과 같은 절차에 의해서 시도된다.

1) 시간을 점검한다.($T_{current}-T_{rep}<=T_{itv}/2$)시간 점검은 시스템 자원정보의 정확도를 표시한다. 제안된 방법에서는 자원정보 시간 간격(T_{itv})의 절반을 중심으로 하였다. 그러나 좀더 높은 정확도를 요구하려면 좀 더 작은

시간으로 줄일 수 있다(예 $T_{itv}/3$).

2) 현재시간에서 가장 최근에 보고된 시간을 뺀 시간이 보고 주기 간격의 절반보다 작다면, 가장 최근에 보고 받은 값들로써 새로운 연결 수락에 대한 판단을 시작한다.

3) 현재시간에서 가장 최근에 보고된 시간을 뺀 시간이 보고 주기 간격의 절반보다 크다면, 자원관리자에게 자원정보요청을 발생시킨다. 자원관리자로부터 새로 받은 자원정보를 갖고 연결 수락에 대한 판단을 시작한다.

4) QoS관리자는 사용자 혹은 상위 계층으로부터의 번역된 QoS 매개변수들과 자원관리자가 보고한 자원 매개 변수들과 비교해서 판단한다.

5) 새로운 연결에 대하여 자원이 충분하면, QoS관리자는 요구된 매개변수들을 자원관리자에게 전달된다.

```
typedef  struct {
int      request_type; /* CAC 요청(예:'4') */
int      request_time; /* CAC요청 시간 */
int      connect_id;   /* 연결 번호 */
float    cpu;
int      buffer;
float    bandwidth;
} cac_request;
```

그림 2. 새로운 연결에 대한 자원 요청
Fig. 2. resource request for a new connection.

```
typedef  struct {
int      response_type;      /* CAC의 응답 */
int      response_res;       /* CAC의 결과 */
int      response_time;      /* 응답 시간*/
int      connect_id; /* 연결 번호 */
         resource    cpu;
         resource    buffer;
         resource    bandwidth;
         rlscpu;
         rlsbuffer;
         rlsbandwidth;
} cac_res;
```

그림 3. 새로운 연결에 대한 응답
Fig. 3. resource response for a new connection.

6) 연결에 대하여 자원이 불충분하면, QoS관리자는 거절 메시지를 전송한다.

7) 자원관리자는 새로운 연결 요청에 대해서 자원 할당이 성공하면, 매개 변수들을 자원관리자 MIB에 저장하고 승인을 보고한다.

그림 2는 QoS관리자가 자원관리자에게 새로운 연결에 대한 요청을 하는 내용으로 CPU, 버퍼, 대역폭등 연결에 필요한 자원을 요청한다. 그림 3은 요청한 QoS관리자에 대한 자원관리자의 응답에 대한 내용이다.

QoS관리자가 자원정보를 갖고 연결수락제어를 수행함으로써 2가지 장점을 갖는다.

1) 연결수락에 대한 빠른 판단이 가능하다. QoS관리자가 이미 자원정보를 갖고 있으므로 요구된 QoS가 지원 가능한지를 바로 판단할 수 있기 때문에 연결 수락에 대한 빠른 판단이 가능하다. 최악의 경우, 연결수락제어의 판단 시간이 기존의 방법들과 같아 질 수 있다.

2) 실패 연결 요청 횟수를 줄일 수 있다는 장점을 갖는다. 연결을 요청한 최종 사용자에게 지원 가능한 시스템 자원정보를 제공함으로써, 사용자는 자신의 QoS 값들을 자원의 용량에 맞추어서 연결을 요청할 수 있다.

물론 시스템 자원정보를 QoS 값으로 역 변환하는 기능이 제공되어야 한다. 이렇게 함으로써 자원정보 상태를 모름으로써 발생할 수 있는 실패 연결요청 횟수를 줄일 수 있다는 장점을 가질 수 있다.

2. 자원 동조 요청

가. QoS 감시(monitoring)

본 연구에서는 사용자가 QoS 위반을 감지하기 이전에 자원관리자의 과부하 상태가 보고되며, QoS관리자는 QoS와 자원과의 관계를 통해서 발생될 수 있는 QoS 매개변수들을 예측할 수 있다.

QoS관리자는 Q-MOTP에게 QoS 매개변수 중 현재 전송 중인 데이터들의 QoS에 관련된 통계 정보를 모으도록 지시한다. 측정 간격은 데이터 전송률의 역 수로 계산된다.

① 지연 감시는 패킷들의 프로세싱 시간으로써 관측된다. 지연에 관련된 자원들로써는 CPU와 대역폭을 생

각할 수 있다.

② 처리율 감시는 CPU와 대역폭의 과부하를 관측하는데 사용되며, 패킷이 자원을 떠나는 것을 감지함으로써 가능하다.

③ 체증(congestion)을 감지하는 방법은 패킷의 손실율을 측정하는 것이다.

나. QoS 유지(maintenance)

Q-MOTP가 측정한 통계 정보는 QoS관리자에게 보고되며, QoS관리자는 QoS 위반이 발생했는지 점검한다. 합의된 QoS 값과 QoS 관측 값과의 비교를 통해서 차이를 계산한다. 즉 사건의 발생 유무를 판단한다.

자원의 과부하 상태가 반드시 QoS위반으로 나타나는 것은 아니다. 예를 들어서 동화상 데이터의 샘플링이나 압축과정 등의 비 실시간 처리를 프로세싱 할 때는 CPU의 이용률은 높아질 수 있으나, 실시간 처리를 우선한다면 CPU의 높은 이용률에도 불구하고 QoS위반이 일어나지 않을 수 있다.

합의된 QoS 수준을 만족하지 못할 경우, 즉 QoS 위반이 발생했다면

① QoS관리자는 합의된 QoS 요구 수준이 만족되도록 하기 위해서 동조해야 할 자원(들)을 결정한다.

② QoS 조정(adaptation)을 위해서 동조 되어야 할 자원에 필요한 값을 계산한다.

③ QoS관리자는 계산된 값을 가지고 이미 갖고 있는 자원정보와 비교한다.

④ 계산된 값을 자원들에서 지원할 수 있다면, 필요한 자원의 동조를 자원관리자에게 요청한다. 그림 4에서 자원동조요청에 필요한 시간, 연결번호, CPU, 버퍼, 대역폭등을 요구한다.

```
typedef          struct {
int     request_type; /*자원동조요청(예'5')*/
int     request_time;      /* 요청 시간 */
int     connect_id;        /* 연결 번호 */
float   req_cpu;     /* 필요한 CPU 용량 */
int     req_buffer; /* 필요한 버퍼의 용량 */
float   req_band;  /* 필요한 대역폭 용량 */
} tune_req;
```

그림 4. 자원 동조 요청
Fig. 4. resource tuning request.

```
typedefstruct {
int     response_type; /* 동조요청 응답 */
int     response_res;  /* 동조요청 결과 */
int     response_time; /* 응답 시간 */
        resource cpu;
        resource buffer
        resource bandwidth;
        rls      cpu;
        rls      buffer;
        rls      bandwidth;
} tune_res;
```

그림 5. 자원 동조 응답
Fig. 5. resource tuning response.

⑤ 자원관리자는 자원의 재할당 및 동조를 시도한다.

⑥ 자원관리자는 자원을 재할당하고, 승인을 보고한다. 그림 5에서 자원동조 요청에 대한 응답으로 관련된 내용뿐만 아니라 자원의 사용률에 대한 내용까지 보고한다.

⑦ 자원이 충분치 않다면, 자원관리자는 실패 메시지를 반환하고 동조요청은 거절된다.

다. QoS 재협상(renegotiation)

자원동조 요청이 실패한 경우 취할 수 있는 동작들은 재협상, 새로운 QoS에의 적응, 상위 계층에게 보고, 연결해제, 무응답 혹은 이러한 동작들의 부분적인 결합으로 볼 수 있다.

본 논문에서는 재협상을 기본으로 연구되었다. 그림6은 재협상을 요구한 QoS관리자가 상대편 QoS관리자에게 요구하는 형태를 보여준다.

그림 6에서 보는 바와 같이 재협상은 두개의 값으로 설정되는데, 하나는 요구된 값(des_X)이고, 다른 하나는 응용 계층에서 필요로 하는 최소의 값(min_X)이다. X는 본 논문에서 수행된 자원들을 표시하며, 자원관리자

```
Q.Request (remote, {des_cpu, min_cpu},
{des_buffer, min_buffer}, {des_bandwidth,
min_bandwidth}, appli_name, result)
```

그림 6. 재협상 요구
Fig. 6. renegotiation request.

```
if ( result not ok)
   if (ava_cpu < min_cpu or ava_buffer <
min_buffer or ava_bandwidth < min_bandwidth)
                terminate;
   else
   {
        if (ava_cpu >= min_cpu)
        {
        alternate strategy;
        des_cpu = new value;
        }
        if (ava_buffer >= min_buffer)
        {
        alternate strategy;
        des_buffer = new value;
        }
        if (ava_bandwidth >= min_bandwidth)
        {
        alternate strategy;
        des_bandwidth = new value;
        }
    }
```

그림 7. QoS 재협상
Fig. 7. QoS renegotiation.

의 구조체를 갖는다. Appli_name은 응용 이름이고, result는 Q.Request의 결과를 나타낸다. 만약에 Q.Request()가 실패하면, 사용 가능한 자원들의 용량을 보낸다. 예를 들어서, 상대편으로부터 ava_cpu, ava_buffer, ava_bandwidth를 받았다고 가정하면 보내온 값으로 QoS관리자는 그림 7과 같이 재협상을 시도한다. ava_cpu, ava_buffer, ava_bandwidth도 자원관리자의 구조체를 갖는다.

제안된 QoS 관리 구조에서의 QoS관리자의 알고리즘이 그림 8에서 가상 코드로 서술되어 있다. 그림 8에서와 같이 처음은 새로운 연결에 대한 요청에 대해 수행되는 내용을 설명하고 있고, 두 번째 경우에는 QoS관리자가 Q-MOTP 에게 QoS상태 정보를 수집하도록 요청하는 경우에 수행되는 행동과 내용이 언급되어있다. 세 번째와 네 번째 경우는 자원관리자가 QoS관리자에게 주기적으로 또는 사건이 발생했을 경우에 대한 처리 부

```
while(1) {
        if ((Event occurs) >= 0 ) {
        switch(Event number) {
        Case CAC request:
          translate QoS parameters;
          Calculate time;
         Compare QoS parameters with report
              values;
          Request to RM ; break;
        Case monitoring results from Q-MOTP:
          gather information's;
          measure informations;
         decide which of the resources have to
             be tuned;
          compute the tuning value;
          request resource tuning ; break;
        Case periodic report:
          update the values in QoS MIB; break;
        Case Event report:
          update the values in QoS MIB;
            request monitoring to Q-MOTP;
break;
        Case CAC response;
          check response result;
          update the values in QoS MIB;
          response to Q-MOTP; break;
        Case tuning response;
          check response result;
          update the values in QoS MIB break;
        }
        }
}
```

그림 8. QoS 관리자의 기본 알고리즘
Fig. 8. Basig algorithm of QoS Manager.

분을 다루고 있다. 그 다음은 새로운 연결에 대한 응답과 동적관리에서 요구한 자원동조에 대한 응답에 대한 내용으로 구성된다.

Ⅲ. 결 론

본 연구에서는 기존 연구들과는 달리 QoS 관리자는 자원관리자를 통해서 얻은 시스템의 자원정보를 갖는다. QoS정보 및 자원정보의 2가지 정보를 가진 QoS 관리자는 보다 효과적으로 합의된 QoS 요구수준을 만족하도록 QoS의 정적관리인 연결수락제어와 동적관리인 자원동조요청을 수행한다.

제안된 연구에서

1) 정적관리에서는 다음과 같은 장점을 갖는다.

▸연결 수락에 대한 빠른 판단이 가능하다.

▸실패 연결 요청 횟수를 줄일 수 있다

2) 동적관리에서 QoS 관리자는 자원 관리자와 서로 같은 시간 영역으로 강력하게 유기적으로 결합되어 프로토콜 시간 영역에서 발생할 수 있는 성능변동(performance fluctuations)들을 실시간으로 감지하고 자원동조 요청을 통해서 대응하도록 하므로 QoS 관리를 수행한다.

추후 연구과제로는 본 연구에서 제안한 QoS 관리 구조에서 수행되는 프로토콜인 Q-MOTP의 성능에 대한 수학적 분석과 시뮬레이션을 이용한 평가가 수행될 것이다.

REFERENCES

[1] G. Armitage, “MPLS: the magic behind the myths,” *IEEE Commu. Magazine,* pp. 124-131, Jan. 2000.

[2] G. M. Lee, and J. K. Choi, “Flow-based Admission Control for Multiple Service Classes in ATM-based MPLS Network,” *Proc ICATM'01* pp. 37-41, 2001

[3] C. Lin and E. C .Lim, “Dynamic Queue Length Thresholds for Scheduling Real-Time in ATM Traffic,” *Proc ICC'99* pp. 869-874, 1999

[4] Hyong W. Lee and Jon W.Mark “ATM Network Traffic Characterization Using Two Types of On-Off Sonurces”, *INFOCOM'93*, March 28-29 1993, pp.152-159

[5] W. K. Choi, S. S. Ahn “Performance Analysis of Multimedia-Oriented Error Control Mechanism Over ATM Network”, *KISS.(A)* Vol.26 No.7 1999 pp827-838

[6] W. K. Choi “Effective QoS Supporting Scheme for Multimedia Streams in MPLS Router”, *KICS*. Vol.34 No.8 2009 pp260-266

[7] W. K. Choi “Performance evaluation of Q-CBQ method for multimedia streams in MPLS Router”, *KICS*. Vol.37 No.1 2012 pp1-8

[8] W. K. Choi “Effective multimedia object data transport protocol in MPLS network using Q-CBQ method”, *KICS*. Vol.37, 2012. 8 pp180-184

[9] W. K. Choi “Performance evaluation of Q-MOTP for multimedia object data transfer in MPLS network”, *KICS*. Vol.37 No.1 2012 pp175-179

[10] W. K. Choi, “A study of transfer delay of Q-MOTP for multimedia object streams in MPLS network” *KICS* Vol38, 2013, 2 pp28-32

[11] W. K. Choi ,“Resource Manager of QoS supporting of Q-MOTP for multimedia object data transfer in MPLS network using Q-CBQ ”*KICS* Vol38, 2013, 12 pp962-966

저 자 소 개

최 원 근(정회원)

1982년 아주대학교 전자공학과 학사졸업.

1986년 고려대학교 전자공학과 석사졸업.

1999년 고려대학교 전자공학과 박사 졸업.

<주관심분야 :멀티미디어통신, QoS, 트래픽모델링>