IMS 네트워크 기반 이종 다중 AS 환경에서의 부하 분산 기법

정회원 류 영 준*, 조 윤 상*, 송 민 도*, 김 무 현*

Load Balancing Scheme in Heterogeneous Multiple AS Environment based on IMS Network

Yung-Jun Yoo*, Yoon-Sang Cho*, Min-Do Song*, Moo-Hyun Kim* ***Regular Members***

요 약

본 논문에서는 IMS(IP Multimedia Subsystem) 기반의 네트워크에서 이종의 다중 AS(Application Server)를 사용할 때의 부하 분산 기법을 제안한다. IMS 기반 네트워크에서 서비스 등록 패턴이 서로 다른 AS를 사용하는 경우 부하 분산을 위해서는 AS마다 서로 다른 비중 값을 두어야 하는데 기존의 시스템은 운용자가 결과 값을 보고 비중 값을 수동으로 설정해야 하는 불편함을 가지고 있다. 본 논문에서는 이상적인 비중 값을 자동으로 계산하고 동시에 부하분산을 시킬 수 있는 기법을 제안한다. 또한 특정한 상황에 대해 계산량을 줄일 수 있는 간편화 알고리즘을 제안하여 상황에 따라 유동적으로 제안 기법을 사용할 수 있는 방안을 제시하였다. 시뮬레이션 결과를 통해 기존의 기법들과 비교하여 제안하는 기법이 부하 분산을 하는데 있어 더 우월한 성능을 보이며 환경의 변화에도 자동적으로 빠르게 수렴한다는 점을 밝힌다.

Key Words : Load Balancing, IMS, Application Server, Service Trigger, S-CSCF

ABSTRACT

In this paper we propose a load balancing scheme for heterogeneous multiple AS's (Application Server) in IMS (IP Multimedia Subsystem) based network. In IMS network, to perform load balancing among multiple ASs with different registration pattern, different weight value should be set for each AS. In previous systems, there exists an inconvenience that the weight value should be set manually by the operator after monitoring the result. In this paper we propose a method to calculate optimal weight in automatic manner and to perform load balancing simultaneously. We also propose a simplified algorithm to reduce calculation in specific situation and present a way to apply our proposed scheme in adaptive manner according to the situation. Through simulation result, we verify that our proposing scheme outperforms previous schemes in load balancing and adjusts well to the change of the system in automatic manner with fast convergence.

I. 서 론

IMS는 최근 VoIP(Voice Over Internet Protocol)와 멀티미디어 서비스의 융합 기술인 SoIP(Service on Internet Protocol) 서비스를 지원하기 위해 가장 많이 사용하고 있는 시스템으로서 3GPP 표준에 멀티미디어 통신의 핵심 네트워크로서 정의되어 있다[1~3]. IMS 네트워크에서 AS는 서비스 트리거 패킷을 처리하고

* KT IP/Platform 운용센터 ({yjryu83,yscho84,smd,jkjewel01}@kt.com)
논문번호 : KICS2010-06-262, 접수일자 : 2010년 6월 10일, 최종논문접수일자 : 2011년 1월 28일

다양한 응용 서비스 로직들을 수행하는 데 있어 핵심적인 역할을 하고 있다.

가입자의 수가 적었던 시절에는 한 대의 AS만으로 서비스 수행이 가능하였다. 그러나 가입자 수와 서비스 수가 점점 늘어남으로 인하여 점점 더 많은 부하가 AS에 걸리게 되었고 하나의 IMS 네트워크에서 다수의 AS를 사용하는 방법이 연구되었다[7]. 최근에는 신규 서비스 등록 등의 이유로 기존의 시스템에 새로운 AS를 추가하는 경우가 많아지고 있는데 이 경우AS의 벤더가 기존의 것과 다르거나 혹은 신규 서비스가 등록됨으로 인하여 AS별로 서로 다른 서비스가 등록되어 있는 경우가 많다. 이러한 이종의 AS를 사용하는 경우 서버 별로 등록되어 있는 서비스 종류가 다르기 때문에 기존의 라운드 로빈 방식 등을 사용했을 때 부하를 분산 시키는데 어려움이 있으며 특정 AS로 트래픽이 몰릴 경우 시스템 장애의 원인이 된다. 이런 문제를 해결하기 위해서는 효과적인 서비스 트리거링을 통하여 부하 분산을 시켜주어야 한다.

대부분의 서비스 트리거 관련 연구들은 단일 AS혹은 동종의 AS를 사용하는 환경에서 지연을 줄인다거나 성능을 높이는 방법이 대해 주로 이루어졌으며 이종 AS 간의 부하 분산에 대한 연구는 미흡하다[4-6]. 기존의 부하 분산 방법으로는 라운드 로빈 방식, 비중 값을 주는 방식 그리고 서비스별로 AS를 지정하여 트리거를 하는 방식이 있다. 현재 상용화 되어 있는 시스템에서는 비중 값을 줄 때 운용자가 결과값을 보고 직접 수동으로 설정하는 방식을 사용하고 있다. 수동으로 비중 값을 선택하게 되면 정확한 부하분산이 힘들어지고 장애가 나거나 새로운 AS 장비를 들여오는 등 시스템의 변화가 생겼을 경우 비중 값을 새로 설정해야 한다는 번거로움이 있다. 본 논문에서는 이러한 문제점을 해결하기 위해 각 AS의 서비스마다의 이상적인 비중 값을 자동으로 계산하고 이를 통하여 정확하게 부하분산을 시킬 수 있는 기법을 제안한다. 또 특정한 상황에 대하여 알고리즘 수행을 위한 계산을 줄일 수 있는 방법을 제시하여 상황에 따라 운용자가 두 가지의 알고리즘을 선택적으로 운용할 수 있도록 한다.

본 논문의 순서는 다음과 같다. 2장에서는 본 논문에서 사용하는 시스템 모델 및 주요 파라미터 들을 설명하고 3장에서는 이종의 멀티 AS를 사용하는 경우 부하 분산을 수행할 수 있는 알고리즘과 특정한 상황에 대하여 계산량을 줄일 수 있는 간편화 알고리즘을 제시한다. 그리고 4장에서는 시뮬레이션 결과를 통해 제안하는 기법과 기존의 기법들의 부하 분산 성능을 비교하고 마지막으로 5장에서는 결론을 내린다.

Ⅱ. 시스템 모델

N개의 AS가 있고 M개의 iFC 기반 서비스가 존재하는 시스템을 고려한다. 모든 서비스는 iFC 기반 서비스이며 S-CSCF(Serving-Call Session Control Function)는 iFC 정보를 통하여 각 AS마다 수용되어 있는 서비스와 그렇지 않은 서비스가 무엇인지 알고 있다고 가정한다.

S-CSCF는 SPT(Service Point Trigger)를 통해 서비스 트리거가 발생하면 HSS(Home Subscriber Server)로부터 iFC(initial Filter Criterion) 정보를 받아오게 되고 이것을 기준으로 적합한 AS의 후보군을 결정하게 된다. AS의 후보군이 결정되면 DNS (Domain Name Server)에 설정된 비중 값에 따라 후보군에 속해 있는 각AS로의 트리거 전송 확률이 정해지고 이 전송 확률에 의거하여 S-CSCF는 해당 AS로 서비스 트리거를 전송하게 된다.

대부분의 상용화된 IMS 장비에서는 운용의 편리성을 위해 EMS(Element Management Server)라고 하는 통계서버를 사용한다. EMS의 정보를 이용하면 사용자들이 어떤 서비스를 어떤 빈도로 사용하는 지를 알 수 있는데 이를 통해 서비스 트리거 패킷의 확률 분포를 알 수 있다. 서비스 트리거 패킷의 확률 분포를 정확하게 알면 운용자가 원하는 비율로 정확하게 부하 분산을 시킬 수 있다. 본 논문에서는 EMS에서 SOAP (Simple Object Access Protocol) 연동 방식을 사용하여 S-CSCF에 서비스 트리거 패킷의 확률 분포를 주기적으로 업데이트 해줌으로써 S-CSCF가 이를 정확하게 알고 있다고 가정한다.

본 논문에서는 시스템 모델을 표현하기 위해 3가지의 행렬을 정의한다. 첫 번째 행렬은 비중 행렬(Weight Matrix)이다. 비중 행렬은 $N \times M$ 행렬로서 행은 AS, 열은 서비스 종류를 나타내며 $\omega_{i,j}$ 를 i 번

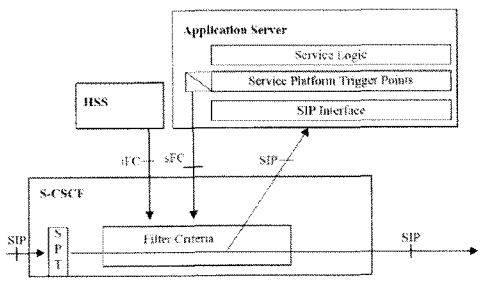


그림 1. 서비스 트리거를 위한 IMS 네트워크 구조

째 행의 j번째 열 구성요소로 갖는다. $\omega_{i,j}$는 AS_i가 서비스 j 를 올릴 확률로서 조건부 확률이다. 따라서 $\sum_{i=1}^{N}\omega_{i,j}=1$ 이 되는데 즉 하나의 행 혹은 하나의 열의 구성요소를 모두 더하면 1이 되는 특징을 가진다. 만약 i 번째 AS에 서비스 j 가 등록되어 있지 않은 경우 $\omega_{i,j}$의 값은 0이 된다. 대부분 운용단에서는 HSS에 들어가 있는 iFC 정보를 통하여 AS에 올라가 있는 서비스 정보를 모두 알기 때문에 해당 AS에 서비스가 올라가 있지 않은 경우 $\omega_{i,j}$의 값은 미리 0으로 초기화 되어 다른 구성 요소들의 값 할당이 끝날 때까지 변하지 않는다. 즉 S-CSCF는 서비스 트리거를 위한 비중 행렬의 값을 계산하기 전에 구성 요소 중 0인 것과 아닌 것을 미리 구분할 수 있다. 본 논문에서는 비중 행렬을 W 라 정의한다.

두 번째는 확률 행렬(Probability Matrix)이다. 확률 행렬은 각 서비스 별 수행 확률의 분포를 나타내며 $M\times 1$행렬이다. 확률 행렬의 구성요소 p_j는 서비스 트리거 패킷이 서비스 j일 확률이 되고 j 값이 커질수록 우선순위가 낮아지게 된다. 또한 $\sum_{j=1}^{M}p_j=1$이 되는 성질을 지닌다. 확률 행렬을 이루는 모든 구성 요소들의 값은 EMS를 통해 주어진 값이며 이 값이 정확하기만 하면 부하 분산을 하는 데 있어 이상적인 비중 행렬을 구할 수 있게 된다. 본 논문에서는 이 확률 값을 정확하게 추정 할 수 있다고 가정한다.

마지막은 자원 행렬(Resource Matrix)이다. 이는 실제 AS에 부하분산 되는 비율을 의미하며 비중 행렬의 값을 계산하는데 있어 제약 조건(constraint)의 역할을 하게 된다. 자원 행렬의 구성요소 r_i는 AS_i로 서비스가 트리거될 확률이며 $\sum_{i=1}^{N}r_i=1$이 되는 특성을 지닌다. r_i의 값을 설정함으로 인해 운용자는 각 AS 별로 부하를 자신이 원하는 비율로 조정을 할 수 있게 된다. $N\times 1$행렬이며 본 논문에서는 R로 정의한다.

위에서 정의한 3가지 행렬을 통해 시스템을 표현하면 $WP=R$의 형태가 된다. 위의 시스템에서 P 행렬은 EMS로부터 주어지는 값이고 R 행렬은 운용자가 직접 미리 설정한 값이다. 따라서 이에 맞춰서 W 행렬을 계산하고 S-CSCF에서 해당 서비스 열을 검색하여 구해진 W 행렬의 조건부 확률에 의거하여 서비스 트리거를 올리면 R 행렬에 운용자가 지정한 비율대로 부하 분산을 수행할 수 있다. 따라서 본 논문은 경험을 통해 축적한 P 행렬의 정보를 가지고 주어진 R 행렬에 적합한 W 행렬을 계산하는 방법을 찾아내는 것을 목적으로 한다. 기존에는 호처리 결과를 보고 수동으로 비중 값을 잡았는데 이러한 모델링을 통하면 비중 값을 자동으로 잡을 수 있다.

Ⅲ. 멀티 AS 환경에서의 부하 분산 기법

3.1 기본 알고리즘

제약조건인 R 행렬 값은 W 행렬의 요소와 P 행렬의 요소의 곱들의 합으로 이루어져 있다. 이럴 경우 두 요소의 곱을 고려해야 하기 때문에 주어진 R 행렬과 P 행렬을 가지고 W 행렬을 구하는 것이 상당히 복잡하다. 따라서 단일 요소들의 합만을 고려하기 위해서 비중 행렬 W를 다음과 같이 두 행렬의 곱으로 분리할 필요가 있다.

$$W=XQ \tag{1}$$

여기서 Q 행렬은 $M\times M$ 정사각행렬로서 대각선 행렬이다. Q 행렬의 구성요소는 다음과 같이 표현할 수 있다.

$$\begin{cases} q_{i,j}=\dfrac{1}{p_j} & (i=j) \\ q_{i,j}=0 & (i\neq j) \end{cases} \tag{2}$$

위와 같이 Q 행렬을 정의해주게 되면 QP 는 모든 구성요소가 1인 $M\times 1$ 행렬이 되고 이는 원래 시스템을 다음과 같이 변형시켜준다.

$$WP=XQP=X\begin{pmatrix}1\\1\\\vdots\\1\\1\end{pmatrix}=R \tag{3}$$

위 식에서처럼 W 행렬을 X 행렬과 Q 행렬의 곱으로 분리하게 되면 X 행렬에 곱해지는 값들이 모두 1이 되기 때문에 R 행렬 값에 맞추어서 X 행렬 하나만 정해주면 되고 따라서 계산을 한결 간편하게 할 수 있게 된다. 여기서 X 행렬을 보조 행렬(auxiliary matrix)라 정의한다. X 행렬의 특징은 각 행의 모든 구성요소의 합은 R 행렬의 해당 행의 값과 같고 각 열의 모든 구성요소의 합은 해당 서비스의 트리거 확

률(p_j)가 된다. 또 $\sum_{j=1}^{M} p_j = 1$이라고 했으므로 X 행렬의 구성요소를 모두 더하면 1이 된다는 특성을 지닌다. 이를 수식으로 표현하면 다음과 같이 나타낼 수 있다.

$$\sum_{j=1}^{M} x_{i,j} = r_i \quad (4)$$

$$\sum_{i=1}^{N} x_{i,j} = p_j \quad (5)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} = 1 \quad (6)$$

위와 같은 제약조건을 가지고 X 행렬을 구하기만 하면 EMS 정보를 통해 이미 알고 있는 Q 행렬을 사용하여 W행렬을 구할 수 있고 이에 기반하여 트리거를 올리면 R 행렬과 같은 비율로 부하분산을 시킬 수 있게 된다. X 행렬을 구하기 위해서는 다음과 같은 알고리즘을 적용한다.

1. AS_i에 서비스 j가 등록되어 있지 않은 경우에 해당하는 X 행렬의 모든 구성 요소 $x_{i,j}$를 0으로 미리 할당한다.
2. 할당이 완료되지 않은 열 중에서 0이 가장 적은 열을 선택한다. 만약 열에서 0이 아닌 구성요소의 수가 같으면 iFC 정보에 기반하여 서비스 우선 순위가 높은 열이 선택된다.
3. 선택된 열에서 각 구성요소마다 속해있는 열을 검색하여 해당 열에서 할당되지 않은 0이 아닌 구성 요소의 수($N_{i,j}$)가 몇 개인지 파악한다. $x_{i,j} = 0$인 경우 $N_{i,j}$는 ∞로 세팅된다. 선택된 열에서 모든 구성 요소의 $N_{i,j}$가 파악됐으면 4단계로 진행한다.
4. $x_{i,j} = \frac{p_j}{N_{i,j}} \times \frac{r_i}{\sum_{i=1}^{N} \frac{r_i}{N_{i,j}}}$로 할당 후 해당 행의 자원 행렬 값에서 할당한 양만큼을 차감한다. 열의 모든 구성요소의 할당이 끝나면 그 열은 할당이 끝난 열로 간주되며 차후에 이루어지는 계산에 대해서는 관여하지 않는다.
5. 해당 열에서 $x_{i,j} > r_i$를 만족하는 구성요소가 있으면 그 요소에 대해 $N_{i,j} = \infty$로 세팅한다. $K_j = \sum_{k=1}^{N} (x_{k,j} - r_k)^+$라고 정의하면 모든 $x_{i,j}$를 다음과 같이 할당한다.

$$x_{i,j} = \begin{cases} r_i & (x_{i,j} > r_i) \\ x_{i,j} + \frac{K_j}{N_{i,j}} \times \frac{r_i}{\sum_{i=1}^{N} \frac{r_i}{N_{i,j}}} & (otherwise) \end{cases}$$

해당 열에서 모든 구성요소가 $x_{i,j} \le r_i$를 만족할 때까지 이 과정을 반복한다.

6. 더 할당할 열이 있으면 2단계로 가고 없으면 알고리즘을 종료한다.

위 알고리즘을 통하여 X 행렬을 구할 수 있고 계산된 X 행렬에 미리 알고 있는 Q 행렬을 곱하면 W 행렬을 구할 수 있다. 여기서 구해진 W 행렬에 의거하여 서비스 트리거를 수행하면 R 행렬에 정해진 비율대로 부하 분산이 가능하다. 이에 대한 성능은 4장에서 보이도록 한다.

3.2 간편화 알고리즘

위의 알고리즘을 사용하면 매번 할당을 할 때 마다 $N_{i,j}$값과 r_i값을 매번 확인해야 한다. 서비스의 개수 혹은 AS의 개수가 늘어나면 W 행렬을 구하기 위한 계산 역시 이것의 곱에 비례해서 증가하게 된다. 따라서 많은 수의 서비스를 여러 개의 AS로 지원하기 위해서는 기본 알고리즘에서 계산량을 줄인 간편화 알고리즘이 필요하다.

일반적으로 서비스의 개수는 AS의 개수보다 훨씬 많다. 그리고 대부분의 회사들은 비용 절감을 위해 적은 수의 AS로 최대한 많은 수의 서비스를 지원하기를 원한다. 예를 들어 KT의 경우 총 4대의 AS를 사용하여 27개의 서비스를 수용하고 있다. 따라서 동일한 AS에 동일한 패턴으로 등록되어있는 서비스의 그룹이 존재할 확률이 높다. 이러한 일련의 그룹을 본 논문에서는 동종 집합(homogeneous set)이라고 하고 특정 서비스와 같은 동종 집합에 속해있는 열을 동종 열(homogeneous column)이라 정의한다. 예를 들어 표1에서 살펴보면 {서비스 1번, 서비스 3번}은 동종 집합이 되고 서비스 1번과 서비스 3번은 동종 열의 관계에 있다고 할 수 있다. 동종 집합이 존재한다는 의미는 다시 말해서 X 행렬의 열 중에서 같은 구조를 가진 (값이 0인 구성요소의 위치가 같은) 열의 집합, 즉 동

종 열끼리의 집합이 존재한다는 의미이다. 특히 2^M이 N보다 작을 경우 비둘기집의 원리(pigeon hole theorem)[8]에 의해 최소한 한 쌍의 동종 열이 존재한다.

이러한 동종 열이 존재하는 경우 동종 열끼리는 서로 같은 양의 $x_{i,j}$를 할당 받을 것이라는 것을 추측할 수 있다. 따라서 이와 같은 점에 착안했을 때 하나의 열이 할당이 끝나면 이 열과 동종 집합에 속해 있는 열들은 별도의 추가적인 계산 없이 같은 값으로 할당할 수 있다는 결론이 나온다. 이와 같은 점을 반영하여 본 논문에서는 동종 집합이 존재할 경우 기본 알고리즘에서 계산량을 줄일 수 있는 간편화 알고리즘을 제시한다.

간편화 알고리즘에서는 할당된 열에서 할당된 값들을 복사해서 동종 열이 존재하면 같은 값으로 할당하는 방법을 사용하여 계산량을 줄인다. 간편화 알고리즘에 대한 자세한 내용은 다음과 같다.

1. AS_i에 서비스 j가 등록되어 있지 않은 경우에 해당하는 X 행렬의 모든 구성 요소 $x_{i,j}$를 0으로 미리 할당한다.
2. 할당이 완료되지 않은 열 중에서 0이 가장 적은 열을 선택한다. 만약 열에서 0이 아닌 구성요소의 수가 같으면 iFC 정보에 기반하여 서비스 우선 순위가 높은 열이 선택된다.
3. 선택된 열에서 각 구성요소마다 속해있는 열을 검색하여 해당 열에서 할당되지 않은 0이 아닌 구성 요소의 수($N_{i,j}$)가 몇 개인지 파악한다. $x_{i,j}=0$인 경우 $N_{i,j}$ 는 ∞로 세팅된다. 선택된 열에서 모든 구성 요소의 $N_{i,j}$가 파악됐으면 4단계로 진행한다.
4. $x_{i,j}=\frac{p_j}{N_{i,j}}\times\frac{r_i}{\sum_{i=1}^{N}\frac{r_i}{N_{i,j}}}$ 로 할당 후 해당 행의 자원 행렬 값에서 할당한 양만큼을 차감한다. 열의 모든 구성요소의 할당이 끝나면 그 열은 할당이 끝난 열로 간주되며 차후에 이루어지는 계산에 대해서는 관여하지 않는다.
5. 해당 열에서 $x_{i,j}>r_i$를 만족하는 구성요소가 있으면 그 요소에 대해 $N_{i,j}=\infty$로 세팅한다. $K_j=\sum_{k=1}^{N}(x_{k,j}-r_k)^+$라고 정의하고 모든 $x_{i,j}$를 다음과 같이 할당한다.

$$x_{i,j}=\begin{cases} r_i & (x_{i,j}>r_i) \\ x_{i,j}+\frac{K_j}{N_{i,j}}\times\frac{r_i}{\sum_{i=1}^{N}\frac{r_i}{N_{i,j}}} & (otherwise)\end{cases}$$

해당 열에서 모든 구성요소가 $x_{i,j}\le r_i$를 만족할 때까지 이 과정을 반복한다.

6. 모든 미할당 열 중에서 5단계에서 할당된 열의 동종 열을 모두 찾아서 동종 열이 존재할 경우 서비스 순위가 높은 순서대로 5단계에서 구했던 값과 동일하게 할당한다. $x_{i,j}>r_i$를 만족하는 구성요소가 하나라도 있으면 5단계로 돌아가 다시 수행한다. 할당이 모두 끝나면 동종 집합 내 모든 열들은 할당이 된 것으로 간주되며 차 후에 다른 요소를 할당하는 계산에서 제외된다. 만약 더 할당할 열이 남아있다면 2단계로 돌아가고 더 이상 할당할 열이 없다면 알고리즘을 종료한다.

간편화 알고리즘은 동종 집합이 존재하는 경우에 사용해야 한다. 왜냐하면 동종 집합이 없는 경우 동종 열을 검색하는 것은 의미가 없는 동작이며 이것이 오히려 간편화 알고리즘으로 하여금 기본 알고리즘보다도 계산량을 많게 하는 요인이 되기 때문이다. 기본 알고리즘은 0이 아닌 요소들의 개수를 세는 데 있어 $\theta(MN)$의 계산이 필요하고 할당을 하는 데 있어서 $O(\frac{MN(M+1)}{2})$의 계산이 필요하다. 따라서 X 행렬을 구하는데 있어 $\Theta(MN)+O(\frac{MN(M+1)}{2})$ 의 계산이 필요하다. 반면 간편화 알고리즘을 사용하게 되면 동종집합의 개수를 j, i번째 동종 집합에 속하는 동종 열의 개수를 K_i라고 했을 때 동종열을 찾는데 $O(M-\sum_{l=0}^{i}K_l)$의 계산이 필요하므로 $x_{i,j}$값들을 할당을 하는 데 있어서 $O(\sum_{i=0}^{j-1}(M-\sum_{l=0}^{i}K_l)+(M-\sum_{l=0}^{i}K_l)N)$의 계산이 필요하다. 이를 정리해서 좀더 간단히 하면 $O((N+1)(2Mj-\sum_{i=1}^{j}iK_i))$가 된다. 간편화 알고리즘에서도 0이 아닌 요소들의 개수를 세는 데 있어 $\theta(MN)$의 계산이 필요하므로 X 행렬을 구할 때 계산량은 $\theta(MN)+O((N+1)(2Mj-\sum_{i=1}^{j}iK_i))$이 필요

하다. 따라서 $j < \frac{1}{2M}\sum_{i=1}^{j} iK_i + \frac{N(M+1)}{4(N+1)}$일 때 간편화 알고리즘이 기본 알고리즘보다 적은 계산량을 필요로 하게 된다. 일반적으로 시스템 운용자는 동종 집합의 존재 여부를 알고 있기 때문에 운용자가 미리 알고 있는 정보에 기반해서 기본 알고리즘과 간편화 알고리즘 중 더 효과적인 알고리즘을 찾아서 적용할 수 있다.

3.3 알고리즘의 정당성 검증

본 장에서는 앞에서 제시한 기본 알고리즘과 간편화 알고리즘의 정당성을 검증한다. 알고리즘의 정당성을 검증하기 위해서는 알고리즘이 끝나기 전에 자원을 모두 소모하는 경우가 없다는 점과 마지막 열의 할당이 끝나며 알고리즘이 종료된 후 남는 유휴 자원이 없다는 점을 밝혀야 한다. 먼저 알고리즘이 끝나기 전, 즉 마지막 열의 할당이 끝나기 전에 자원이 모두 소모되는 것은 불가능하다. 왜냐하면 식 (5)에 의하면 서비스 j에 해당하는 열의 구성요소의 합은 p_j가 되고 이는 한 열에서 사용할 수 있는 자원의 총 합이 p_j라는 의미가 된다. 알고리즘이 끝나기 전에 자원을 모두 사용하려면 지금까지 할당한 p_j들의 합이 1보다 커야 하는데 이는 식 (5)와 식 (6)에 위배된다. 따라서 알고리즘이 끝나기 전에 모든 자원이 고갈되는 경우는 없다고 볼 수 있다.

알고리즘 종료 후 남는 유휴자원이 존재하는 것도 불가능하다. 기본 알고리즘과 간편화 알고리즘 모두 4단계에서 $x_{i,j} = \frac{p_j}{N_{i,j}} \times \frac{r_i}{\sum_{i=1}^{N} \frac{r_i}{N_{i,j}}}$로 할당을 하는데 다른 모든 열의 할당이 끝나고 마지막 열을 할당하는 경우 $N_{i,j}$의 값은 1이다. 마지막 열에서 $\sum_{i=1}^{N} r_i = p_j$이므로 $x_{i,j}$는 r_i와 값이 같아지게 된다. 따라서 마지막 열에서 $x_{i,j}$는 남아있는 자원을 모두 다 할당 받는다는 사실을 확인할 수 있고 이에 따라 알고리즘 종료 후 남는 유휴자원이 존재할 수 없다는 것을 보일 수 있다.

이와 같은 이유로 기본 알고리즘과 간편화 알고리즘 모두 알고리즘이 주어진 제약조건에 맞게 정상적으로 작동한다는 점을 확인할 수 있다.

Ⅳ. 성능 평가

본 장에서는 여러 시뮬레이션 결과를 통해 본 논문에서 제안하는 부하 분산 기법의 성능을 검증하도록 한다. 복수의 이종 AS를 사용하는 IMS 기반 네트워크에서 모의실험을 수행하였다. 시뮬레이션 도구는 MATLAB을 사용하였고 총 서비스의 개수는 8개이며 AS의 개수는 4개로 설정하였다. 각 AS의 서비스는 모두 다른 배열을 지니고 있으며 S-CSCF의 개수는 편의상 1개로 가정하였다. S-CSCF는 항상 보낼 패킷이 있다고 가정하였고 HSS로부터 받은 iFC 정보에 기인하여 해당 서비스가 등록되어 있는AS의 위치 정보를 받게 되고 제안하는 기법에 의해 계산된 W매트릭스에 지정된 확률에 따라 AS로 트리거를 올리도록 하였다. 각 AS 별 서비스 등록 패턴은 표 1에 나타나 있으며 해당 AS에 서비스가 등록되어 있으면 O, 등록되어 있지 않으면 X로 표시하였다. 부하 분산의 척도는 Jain's Fairness Index[9]를 사용하였으며 이 값이 1에 가까울수록 부하 분산이 잘 된 것이라고 판단하였다. 각 실험에 대한 Jain's Fairness Index는 표 3에 정리되어 있다. 그림 2는 시뮬레이션에 사용한 네트워크의 모델을 간략하게 그림으로 나타낸 것이다.

그림 3은, 모든 서비스의 수행 확률이 같은 경우 라운드 로빈 방식과 지정 AS 트리거 기법 그리고 제안하는 기법을 사용했을 때 AS별 호 처리 수를 비교해서 보여주고 있다. 라운드 로빈 방식은 해당 서비스가 수용되어 있는 AS를 향해 동일한 확률로 트리거를 올리는 방식이고 지정 AS 트리거 방식은 각 서비스마다

표 1. AS 별 서비스 등록 패턴

서비스	#1	#2	#3	#4	#5	#6	#7	#8
AS #1	O	O	O	O	O	X	X	O
AS #2	O	X	O	X	O	O	O	X
AS #3	X	X	X	O	X	O	O	O
AS #4	O	X	O	O	X	O	X	O

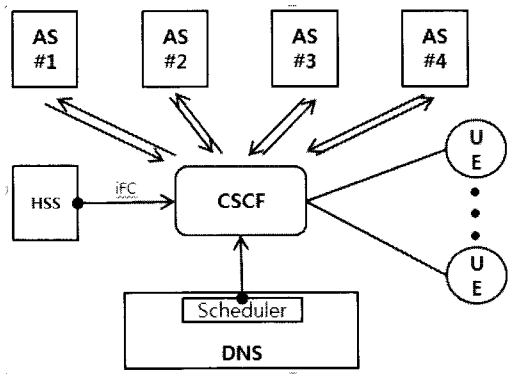


그림 2. 실험에 사용한 네트워크 모델

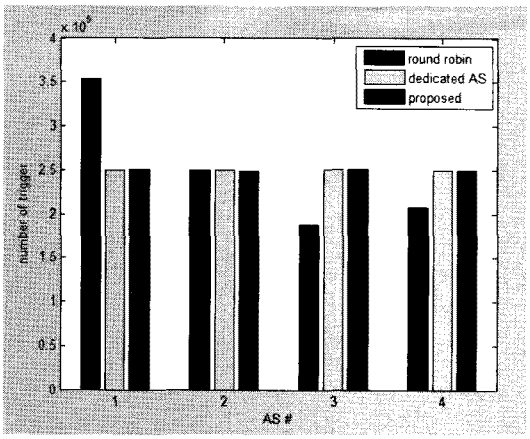


그림 3. 동일 서비스 확률에서 AS 별 호 처리 수

AS를 지정하여 서비스 별로 지정된 AS로만 트리거를 올리는 방식이다. 이 실험은 그림 4에서의 결과를 보기 위한 실험군으로서 AS 1대 당 서비스 2개씩 지정하여 지정 AS 트리거 방식을 사용했을 때 부하 분산이 고르게 되도록 설정하였다.

그림에서 보다시피 제안하는 기법은 라운드 로빈 기법에 비하여 훨씬 좋은 부하 분산을 보여주고 있다. 라운드 로빈으로 트리거를 올리게 되면 가입되어 있는 서비스 수의 비율로 호가 분배되기 때문에 서비스 등록 패턴이 다른 이종의 AS를 사용하는 경우 부하 분산에 문제가 있다는 것을 알 수 있다. 반면 제안하는 기법이나 지정 AS 트리거 기법을 사용하면 부하 분산 적인 측면에서 좋은 성능을 낼 수 있다는 점을 확인할 수 있다.

그림 4는, 서비스의 수행 확률을 다르게 했을 경우이다. 그림 3의 경우는 모든 서비스가 0.125의 확률로 균등하게 수행된 반면 이번 실험에서는 표 2에 나와

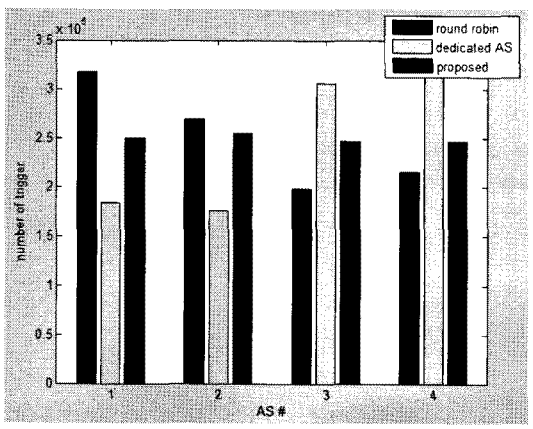


그림 4. 이종 서비스 확률에서 AS 별 호 처리 수

표 2. 서비스별 수행 확률

서비스	#1	#2	#3	#4
수행확률	0.130	0.123	0.200	0.122
서비스	#5	#6	#7	#8
수행확률	0.045	0.135	0.185	0.065

표 3. 각 실험 별Jain's Fairness Index

	라운드로빈	지정 AS	제안기법
그림3	0.9264	1.0000	1.0000
그림4	0.9633	0.9260	1.0000
그림5	0.9714	0.9923	0.9995

있는 데로 각 서비스 별로 서로 다른 서비스 수행 확률을 지닌다. 그림 3에서는 부하 분산이 잘 되었던 지정 AS 방식도 서비스의 확률이 달라지면서 부하분산에 실패하는 모습을 보여주고 있다. 반면 제안 하는 기법은 서비스 확률이 달라지더라도 별다른 영향 없이 서비스 부하 분산에 성공하는 모습을 보여주고 있다.

그림 5는 시스템에 장애가 발생했을 경우의 AS 별 호 처리 수를 보여준다. 1000,000개의 호가 트리거 되는 동안 AS 4번에서 임의의 시간 동안 장애가 일어나 서비스를 못하게 되고 장애가 일어난 동안에는 해당 AS로 트리거를 올리지 못하도록 하였다. 장애는 0~1000,000 사이에서 유니폼하게 발생시켰으며 장애로 인하여 AS 4번이 동작을 하지 못하는 동안에는 AS 1,2,3 번이 트리거를 나누어서 가져가도록 하였다. 그림에서 살펴보면 시스템 장애로 인하여 4번 AS의 트리거 개수가 줄어들고 제안하는 기법을 상용하였을 때 나머지 1,2,3번이 고르게 패킷을 가져가는 모습을

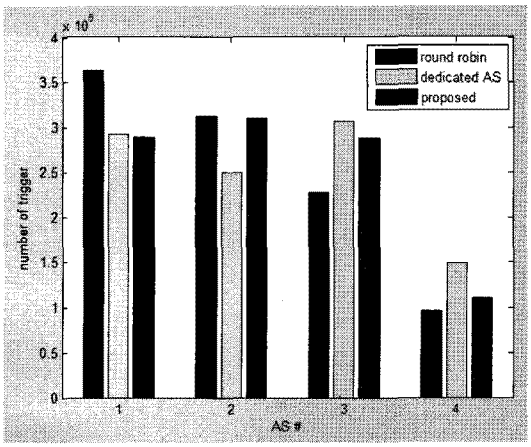


그림 5. 장애가 있을 경우 AS 별 호 처리 수

볼 수 있다. 제안하는 기법은 라운드 로빈이나 지정 AS 트리거 기법보다 좋은 부하 분산 성능을 보여주었음을 한 눈에 확인 할 수 있다.

그림 6은 서비스 수행 확률이 변화하는 환경에서 고정된 비중 값을 사용하는 경우와 제안하는 기법을 사용하는 경우의 부하 분산 성능을 비교해서 보여주고 있다. 그림을 살펴보면 고정된 비중 값을 사용하는 경우에는 부하 분산에 실패한 반면 제안하는 기법은 서비스 수행 확률의 변화에도 불구하고 부하 분산에 성공하는 모습을 보여주고 있다. Jain's Fairness Index를 구해봤을 때 고정된 비중 값을 사용 할 때는 0.9486, 제안하는 기법을 사용했을 때는 1.0000이 나왔다. 고정된 비중 값을 사용할 때 실패한 부하 분산을 바로 잡기 위해서는 운용자가 결과를 보고 수동으로 비중 값을 재설정 해주어야 한다. 반면 제안하는 기법은 비중 값을 자동으로 계산하여 부하 분산을 시켜주고 시스템 환경의 변화에도 빠르게 잘 적응한다. 이러한 결과를 통해 제안하는 기법이 시스템의 변화에 대해 기존의 고정 비중 값을 사용하는 기법보다 빠른 수렴 시간을 가진다고 결론을 내릴 수 있다.

실제로 운용을 하는 데 있어서 시스템에 장애가 나는 일은 비일비재하며 각 서비스 별 가입자 수의 변동 등에 의해 서비스 확률이 변화하는 경우도 많이 있다. 이런 경우 부하 분산에 실패하게 되면 특정 시스템에 부하율이 상승하여 연쇄 장애를 일으키는 원인이 된다. 제안하는 기법에서는 시스템에 변화가 생기더라도 이상적인 비중 값을 자동으로 계산해 주기 때문에 시스템 변화에 대한 적응성이 뛰어나며 시뮬레이션 결과에서 살펴 본 바와 같이 기존의 기법들에 비해서 뛰어난 부하 분산 성능도 보여준다.

또한 서비스 수행 확률 그 자체도 가입자가 증가한다던가 서비스의 등록 형태가 변화하는 등의 이유로 변화할 수 있다. 이런 경우 새로운 시스템에 빠르게 적응할 수 있느냐 없느냐는 매우 중요한 이슈가 될 수 있다. 고정된 비중 값을 쓰는 경우에는 시스템의 변화에 빠르게 반응하기 어렵다. 반면 제안하는 기법은 비중 값을 자동으로 계산하고 시스템의 변화에 빠르게 적응할 수 있기 때문에 기존의 기법들보다 운용을 하는데 있어 더욱 효과적이라고 볼 수 있다.

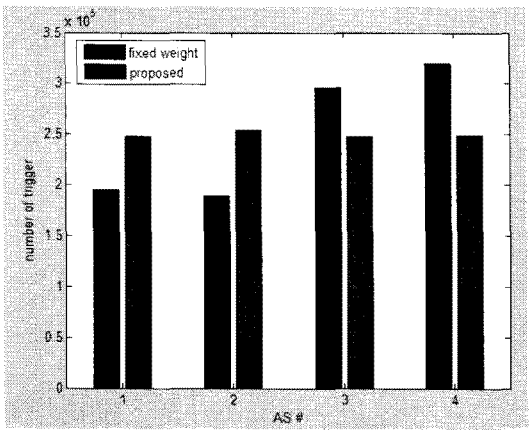


그림 6. 서비스 확률이 변하는 환경에서 AS별 호 처리 수

V. 결 론

본 논문에서는, IMS 기반의 네트워크 환경에서 이종의 AS를 사용할 때 서비스 트리거 패킷을 부하 분산하는 기법을 제안하였다. EMS의 서비스 별 호 처리 통계를 통해 서비스 별 수행 확률을 알 수 있고 HSS로부터 iFC 정보를 받으면 어느 AS에 어떤 서비스가 수용되어 있는 지를 알 수 있다. 위의 두 가지 정보를 알면 이것을 기반으로 AS별 서비스 트리거 확률을 조정하여 부하 분산을 수행할 수 있다.

제안한 부하분산 기법은 기존의 방식과는 달리 자동으로 비중 값을 설정할 수 있다. 따라서 장애가 생기거나 신규 서비스가 추가 되는 등 시스템의 변화가 생겼을 때 빠르게 적응할 수 있다. 또한 본 논문에서는 동종 집합이 존재하는 경우 계산량을 줄일 수 있는 간편화 알고리즘을 제시하였다. 시뮬레이션 결과를 통해 제안한 부하분산 기법이 기존의 라운드 로빈 방식이나 지정 AS 트리거 방식보다 훨씬 우수한 부하 분산 성능을 보여주었음을 확인 할 수 있었다

참 고 문 헌

[1] 3GPP TS 23.002. Network architecture; release 7. September 2007

[2] 3GPP TS 23.218. IP Multimedia (IM) session handling; IM call model;Stage 2; release 7. June 2007

[3] 3GPP TS 23.228. IP multimedia subsystem; Stage 2; release 7. June 2007

[4] Z. Xun, J. Liao, X. Zhu, C. Wang, and Y. Cao "A Group Based Service Triggering Algorithm for IMS Network," in *Proc. Communications, 2009. ICC '09. IEEE International Conference on..*

[5] W. Fangyi, L. Xia, and Z. Hua, "SSTM: A State Based IMS Service Triggering Mechanism,"

Communications and Mobile Computing (CMC), 2010 International Conference on.

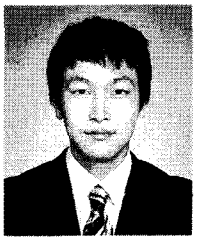
[6] Z. Xun, J. Liao, and X. Zhu, "On Performance of 3GPP Service Triggering Mechanism in IMS Network." *Software Engineering and Advanced Applications, 2008. SEAA '08. 34th Euromicro Conference.*

[7] J. Jeon, J. Lee, S. Woo and J. Son, "A Distributed Architecture of Application Server for SoIP services" *KT R&DZine, 2009. Vol 2.*

[8] Grimaldi, Ralph P. *Discrete and Combinatorial Mathematics: An Applied Introduction.* 4th edn. 1998. pp. 244 - 248.

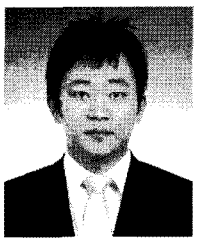
[9] Jain, R., Chiu, D.M., and Hawe, W. (1984) A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems. DEC Research Report TR-301

송 민 도 (Min-Do Song) 정회원



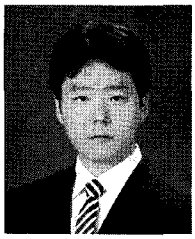
2010년 2월 숭실대학교 정보통신전자공학부 학사
2010년 1월~현재 KT 근무
<관심분야> 무선통신, cognitive radio, SBC

김 무 현 (Moo-Hyun Kim) 정회원



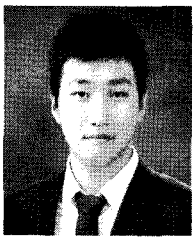
2006년 2월 단국대학교 전기전자컴퓨터공학과학사
2009년 2월~12월 ICRAFT
2010년 1월~현재 KT 근무
<관심분야> 라우팅 프로토콜, All IP 네트워크

류 영 준 (Yung-Jun Yoo) 정회원



2008년 2월 서울대학교 전기공학부 학사
2010년 2월 서울대학교 전기공학부 석사
2010년 1월~현재 KT 근무
<관심분야> 무선통신, IMS

조 윤 상 (Yoon-Sang Cho) 정회원



2010년 2월 한양대학교 전자통신컴퓨터공학부 학사
2010년 1월~현재 KT 근무
<관심분야> 차세대 이동 통신 기술, 근거리 무선 통신