서버측에서의 Downstream 측정을 이용한 중첩서버 선택 시스템의 모델 및 알고리즘

유기성*, 이원혁*, 안성진**, 정진욱**

Server selection system model and algorithm for resolving replicated server using downstream measurement on server-side

Ki-Sung Yu, Won-Hyuk Lee, Seong-Jin Ahn, Jin-Wook Chung

Abstract

In distributed replicating server model, providing replicated services is able to improve the performance of providing a service and efficiency for several clients. And, the composition of the server selection algorithm is efficiently able to decrease the retrieval time for replicated data. In this paper, we define the system model that selects and connects the replicated server that provides optimal service using server-side downstream measurement and proposes an applicable algorithm.

Key Words: Replicated server, server selection, one-way measurement, network, retrieval time, directory service

* 한국과학기술정보연구원

** 성균관대학교

1. 서론

오늘날 전 세계적으로 인터넷을 이용한 파일 송수신과 웹 서핑이 업무, 연구, 학업 그리고 레저 등을 수행하는 일상 생활의 수단이 되었다. 따라서 인터넷을 이용하여 파일을 다운받고, 웹서비스를 받는 행위가 빈번하게 이루어지게 되었다. 또한 서비스 제공의 범위도 확대되어 전 세계 어디서나 접속이 이루어진다. 따라서 같은 종류의 서비스를 원하는 사용자가 전세계 곳곳에 분포되어 있을 수도 있으며, 이에 따라 서버의 구성을 여러 대로 나누어 지역적으로 분산된 서버 시스템의 형태로 구축하는 경우가 많이 있다[1][2].

분산된 서버 시스템에서 클라이언트가 원하는 서비스를 제공하기 위해서는 서비스를 위한 적절한 서버와의 연결이 필요한데, 이러한 방법들에 대하여 많은 연구가 진행되어 있는 상황이다. 분산된 서버 모델에서, 중첩된 서비스의 제공은 여러 클라이언트에게 서비스의 제공 성능과 효율성을 높일 수 있다. 대개의 경우 단순히 지역적으로 가까운 서버로 연결해 주는 방법이 사용되고 있으나, 지역적으로 가깝거나 연결된 홉 수가 가깝다고 해서 응답시간이나 전체적인 서비스의 검색시간이 빠른 것은 아니다. 따라서 각각의 사용자가 원하는 위치에서 최적의 서버를 선택하여 서비스를 제공하는 것이 중요하며, 서버 선택을 수행하는 메커니즘에 여러 가지 기법들이 사용된다.

또한, 효율적으로 구성된 서버 선택 알고리즘은 다른 서버들에 중첩되어 있는 데이터들에 대한 검색시간(Retrieval Time)을 줄일 수 있다. 기존에는 특정 클라이언트에서 서버 선택을 위한 측정을 수행하고, 이를 바탕으로 선택된 서버에 연결하여 원하는 서비스를 제공받았다. 하지만 이러한 방법은 서버 선택을 위한 측정이 실제 서비스를 받기까지 시간 지연을 유발할 수 있으며, 이러한 시간 지연은 오히려 서비스 제공성능에 악영향을 미칠 수가 있다.

또한 현재의 방법은 RTT를 근간으로 하여 서비스의 성능 측정을 특정 클라이언트에서 수행하므로, 비대칭구조의 IP망에서 실제 서비스 제공 경로와 다를 수가 있으므로 단순한 RTT로는 실제 서비스를 제공하는 서버관점에서 정확한 테스트를 할 수 없다. 따라서 본 논문에서는 서버측 다운스트림 측정을 기반으로 하여 중첩된 서비스를 제공하는 분산 시스템들의 입장에서 최적의 서비스를 제공할 수 있는 시스템의 모델에 대하여 정의하고, 적용할 알고리즘에 대하여 연구할 것이다. 또한 기존에 연구된 클라이언트측에서의 실험 측정에 의한 서버 선택과는 달리 서버 측에서의 다운스트림측정을 통하여 보다 정확한 서버 선택을 수행하는 모델의 성능비교를 통해 더 나은 모델의 정립에 대하여 논의할 것이다[3][4][5].

2장에서는 기존에 수행되었던 서버선택 모델들의 문제점과 보완해야할 사항들을 살펴볼 것이다. 3장에서는 본 논문에서 제안하는 중첩 서버선택 시스템의 모델과 알고리즘을 제시한다. 4장에서는 실험을 통하여 본 모델과 알고리즘의 성능에 대하여 증명하고, 마지막으로 5장에서 본 논문에서 제시하는 모델의 의미를 살펴보고 결론을 맺는다.

2. 관련 연구

서버 선택을 수행하는 알고리즘의 특성은 정적, 통계적, 동적 모델로 구분할 수 있다 [6] [7].

정적 모델은 연결된 홉 수나, 연결 대역폭과 같은 하드웨어 자원이나 배치 등에 기반하여 구성된다. 이 모델에서 클라이언트는 사전에 이러한 사항들을 알고 있어서 자신에게 최적의 서비스를 제공할 수 있는 서버가 어떤 서버인지 알고 있다. 이 모델은 NNTP (Network News Transfer Protocol)를 사용하는 네트워크 뉴스 등의 서비스에서 사용된다[4].

통계적 모델은 과거의 측정 데이터를 이용하여 서버 선택을 결정하는 방법이다. 현재의 시간 t 이전에 측정되었던 지연(Delay), 대역

폭(Bandwidth), 그리고 응답시간(Response Time)등의 측정 데이터를 이용하여 현재의 서버 선택 결정에 적용하는 방법이다. 정적 모델에 비하여 네트워크 자원이나 성능 항목에 대하여 능동적인 고려가 가능하지만, 고려되어야 할 항목들이 다양해질수록 정확한 서버 결정에 어려움을 겪는다.

이러한 사항들을 고려하여 동적 모델에서는 현재의 네트워크나 서버의 상태를 고려하여 여러 가지 항목들을 측정하기 위한 에이전트를 사용한다. 에이전트는 현재 가용 자원의 측정을 보다 정확하게 고려하여 동적으로 변하는 여러 가지 사항들을 토대로 서버 선택에 정확한 데이터를 제공할 수 있다. 그러나 동적 모델은 에이전트 사용으로 인하여 네트워크에 부하를 가중시킬 수 있다는 단점을 가진다.

분산되어 중첩서버가 운영되는 환경에서 사용자는 최적의 서비스를 제공받을 수 있는 서버를 선택하기를 원하고, 서비스 제공자 입장에서도 최적의 서비스를 제공할 수 있는 서버가 서비스를 제공해야 자원의 효율적인 운용과 관리가 가능하다. 이러한 최적의 서버를 선택하는 방법 중에서 지역적인 거리를 서버 선택의 요소로 사용하는 방법이 제시되었다. [9]

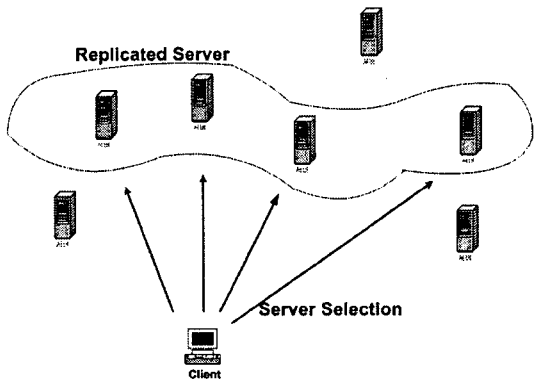
이 방법에서 서버들은 많은 요구가 있는 소스 근처에 중첩서버를 위치시켜야 하는 책임을 가진다. 클라이언트는 홈서버로부터 자신과 가장 가까운 중첩서버의 주소를 얻게된다. 이것은 앞에서 언급한 정적 모델을 사용하는 것으로 서버들은 방대한 지리적 정보를 구축하고 유지해야 한다. 정적 모델을 근간으로 한 이러한 방법은 새로운 서버에 대한 정보 습득과 유지가 어렵고, 많은 양의 정보를 가지고 있어야하는 단점을 가진다.

[8]에서는 중첩된 서버들을 웹서버에 국한하여 여러 가지 알고리즘의 제시와 함께 중첩서버 선택에 대한 방법을 제안하였다. 저자는 서버까지의 홉수나 핑을 이용한 RTT는 HTTP 요청에 대한 응답시간을 측정하는 좋은 인자로 보지 않고, HTTP 요청에 대한 지연을 웹서버 선택에 대한 중요한 인자로 제시하였다. 하지만 현재와 같이 음성과 영상으로 구성된 멀티미디어 데이터가 많은 비율을 차지하는 지금은 이러한 가정에 문제가 있다.

또한, [3]에서는 tcping라는 제작 툴을 사용하여 동적 모델을 이용한 방법이 제안되었다. 제작된 툴을 이용하여 동적 측정 방법을 제시하고, 파일 크기의 일반화도 제안하였다. 파일의 사이즈에 따른 측정뿐만 아니라 밤과 낮에 따른 시간적인 요인과 그때의 대역폭 사용에 따른 영향도 고려하여 동적으로 서버측정을 위한 성능 측정을 시행하였다.

서버 선택에 관하여 기존에 수행된 많은 연구들은 대부분 클라이언트 측에서 서버 선택을 수행하는 것이다[3][4][8].

측정 방법에 있어서는 정적 모델과 동적 모델로 구분되지만, 모든 연구가 클라이언트측에서의 성능 측정에 기반한다. 즉, 특정 클라이언트에서 자신이 정적으로 가지고 있는 서버의 리스트들을 대상으로 서비스 제공에 대한 성능 측정을 수행하는 것이다.



<그림 1> 기존의 서버선택 과정

이러한 모델에서 클라이언트는 <그림 1>과 같이 자신이 원하는 서비스를 제공하는 서버들에게 직접 테스트를 수행하여 적절한 서버를 선택하는 방법을 취하는데, 이 방식에는 다음과 같은 제약사항이 존재한다.

1) 클라이언트는 서비스를 제공하는 서버들의 목록을 사전에 가지고 있어야 한다.
2) 서버 정보의 투명성(transparent) 제공이 불가능하다.
3) 서버측의 부하(Server-side load)는 알 수 없다.

위와 같은 제약사항으로 인하여 실제 환경에서 적용할 때 클라이언트에게 서버의 목록을 최신의 것으로 유지시키기가 어렵고, 서버의 시스템 부하가 많은 경우에는 오히려 수행능력의 저하를 가져오며, 서버 선택의 정확한 결정에 어려움을 가진다.

IP네트워크는 비대칭(asymmetric) 구조를 가지고 있다. 따라서 이렇게 서비스를 제공받을 클라이언트측에서의 측정보다는 실제로 서비스를 제공하는 서버측에서의 측정을 통하여 계산하는 것이 실제 네트워크의 상황을 고려하여 적용하는 방법일 것이다. 따라서 본 논문에서는 서비스를 제공받기를 원하는 클라이언트가 디렉토리 서비스를 이용하여 동적으로 서버의 리스트를 확보함으로써 서버 정보에 대한 투명성(transparent)을 제공하고, 서비스를 실제 제공하는 서버 입장에서 정확한 분석이 가능한 모델을 제시하고자 한다. 이 모델에서는 서버측의 측정을 기반으로 하여 클라이언트가 선택하므로 서버의 현재 부하량이나 상태, 네트워크 상황 등도 혼합하여 고려할 수 있는 요인이 될 것이다.

3. 중첩 서버 선택 모델

본 논문에서는 지역적으로 널리 분산되어 서비스를 제공하는 분산 서버들 중에서 클라이언트에게 최적의 서비스를 제공할 수 있는 서버를 선택하는 서버 선택 모델에 대하여 제안한다. 이는 클라이언트에서 직접 각 서버들로의 성능 테스트를 수행하는 것이 아니라, 실제 서비스를 제공해 주는 서버들의 입장에서 서비스를 제공하는 성능 테스트를 수행하므로, 보다 최적의 서비스를 제공할 수 있는 서버의 선택이 이루어 질 수 있다.

3.1 서비스 구성 요소

1) 디렉토리 서비스 : 서비스를 제공하는 서버들의 리스트를 가지며, 클라이언트의 요청을 받아들여 현재 등록되어 있는 서비스 서버들의 리스트를 반환
2) 대행자 : 각각의 분산 서버에서 동작하며 클라이언트의 요청과 함께 전송된 데이터를 이용하여 클라이언트에 서비스를 제공하기 위한 항목을 측정하고 결과값을 반환하는 기능을 수행
3) 클라이언트 : 서비스를 제공받기 원하는 개체로서, 디렉토리 서비스를 이용하여 중첩서버들의 리스트를 얻은 후에 측정을 요청
4) 사건 : 클라이언트와 에이전트간의 메시지 흐름

3.2 서버 선택의 알고리즘

정의 1)
디렉토리서비스에서 제공하는 서비스 제공 서버들의 총 집합을 다음과 같이 정의한다.

$$T = \{ t_1, t_2, \cdots, t_k \}$$

(k는 service server의 총 수)

정의 2)
클라이언트가 제공받기를 원하는 서비스를 제공할 수 있는 서버들의 집합을 다음과 같이 정의한다.

$$R = \{ r_1, r_2, \cdots, r_k \}$$

(k는 replicated server의 개수)

정의 3)
전체 집합을 이루는 서비스 서버중에서 replicated server에 속하지 않는 서버의 집합은 다음과 같이 정의한다.

$$N = T - R = \{ n_1, n_2, \cdots, n_k \}$$

즉, 전체 집합 T와 R, N은 다음과 같은 속성

을 갖는다.

$$T = R \cup N, \quad R^c \cap N^c = \varnothing$$

정의 4)

클라이언트에서 특정 서버로 메시지를 전송하고 받는 시간을 다음과 같이 정의한다.

$\mathcal{G}(r_k)$: 클라이언트에서 r_k노드로 메시지 전송시 걸리는 시간

$\mathcal{F}(r_k)$: r_k노드에서 클라이언트로 메시지 전송시 걸리는 시간

정의 5)

클라이언트에서 각 서버로 메시지를 전송하고 응답을 받는 동안 걸리는 총 시간은 다음과 같이 정의한다.

$$\mathcal{T}(r_k) = \mathcal{G}(r_k) + \mathcal{F}(r_k)$$

클라이언트에서 각 서버 r_k에 메시지를 전송하고 응답받는 시간 $\mathcal{T}(r_k)$중에서 가장 작은 값을 선택하여 최소 서버측정 시간 MIN_Svr_Time으로 정의한다.

$$MIN\_Svr\_Time = Min(\mathcal{T}(r_1), \mathcal{T}(r_2), \ldots, \mathcal{T}(r_k))$$

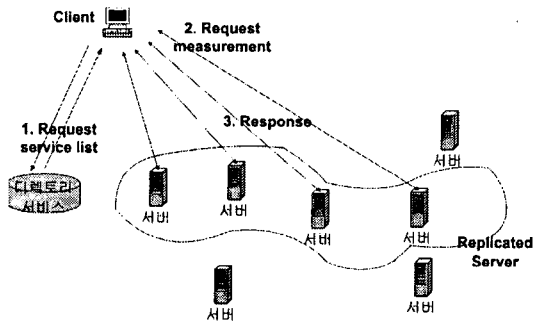
정의 6)

MIN_Svr_Time이 결정되는 서버 r_k가 클라이언트가 원하는 서비스를 가장 최단 시간에 제공할 수 있는 서버이다.

3.3 시스템의 동작 과정

시스템은 디렉토리 서비스와 클라이언트, 그리고 다수의 중첩서버들로 구성된다. 클라이언트는 서비스를 제공받기를 원하는 사용자 컴퓨터이고, 중첩서버는 이를 제공할 수 있는 서비스를 가지고 있는 서버들이다. 또한 중첩서버들의 목록은 디렉토리 서비스를 통하여 얻을 수 있다. 전체적인 동작과정은 다음 <그림 2>와 같다.

1. 클라이언트가 LDAP을 사용해서 디렉토리 서비스에 연결하여 원하는 서비스의 항목을 요청(Request)한다.
2. LDAP Server는 요청된 서버들의 리스트를 반환한다.
3. 클라이언트는 중첩서버의 리스트를 바탕으로 각각의 중첩 서버(Replicated Server)에 측정을 요청한다.
4. 중첩 서버는 테스트용 메시지를 전송한다.
5. 클라이언트는 각각의 중첩 서버가 보내온 정보를 분석하여 최적의 다운스트림을 제공하는 서버를 선택한다.
6. 클라이언트는 선택된 서버로 접속을 수행하여 원하는 정보를 얻는다.



<그림 2> 시스템의 동작과정

3.4 서비스 요청

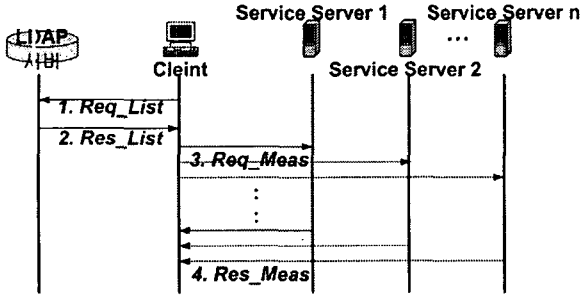
① ***Req_List*** : 클라이언트는 원하는 서비스를 제공하는 서버를 알기 위해 LDAP서버에 Req_List메세지를 보낸다.

② ***Res_list*** : 서비스 서버들의 리스트 요청 메세지를 받은 LDAP서버는 등록되어 있는 서버의 리스트를 반환한다.

③ ***Req_Meas*** : 클라이언트는 대행자에게 원하는 서비스의 종류를 명시하여 이 메시지를 보낸다.

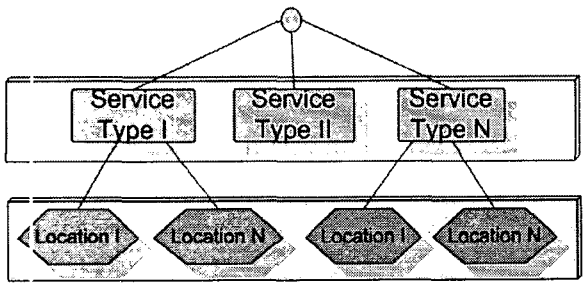
④ ***Res_Meas*** : 각 서비스 서버들에 탑재된 대행자들이 각각의 성능 평가 데이터를 클라이언트에 전달한다.

<그림 3>은 클라이언트가 디렉토리 서비스를 거쳐서 서비스 서버들의 리스트를 얻어와서 각 서비스 서버들에게 서비스를 제공하는 성능 측정의 테스트를 요청하고 응답받는 과정을 나타낸다.



<그림 3> 사건 흐름도

3.5 서비스 서버를 위한 디렉토리 서비스 설계



<그림 4> 서버를 위한 디렉토리 서비스

서비스 서버를 위한 디렉토리 서비스는 위의 <그림 5>와 같이 구성된다. 사용자가 원하는 서비스 서버를 선택하기 위한 정보를 그림과 같이 구성하고 요청시에 LDAP에 의하여 정보를 검색하여 응답한다.

디렉토리 서비스의 각 엔트리의 오브젝트 구성은 다음 <표 1>과 같다.

<표 1> 디렉토리 서비스의 오브젝트 구성

Entry	Attribute Type	OID	Value
rdn	OrganizationName	2.5.4.13	
Service_Type	ServerType	1.3.6.1.4.1.13418.500	MUST
Server_Name	ServerName	1.3.6.1.4.1.13418.501	MUST
File_Name	FileName	1.3.6.1.4.1.13418.502	MUST
File_Size	FileSize	1.3.6.1.4.1.13418.503	MUST
Description	description	1.3.6.1.4.1.13418.504	MAY

objectclass = servObject

Entry	Attribute Type	OID	Value
dn	ou	2.5.4.14	
IP	ipAddr	1.3.6.1.4.1.13418.505	MUST
Port	portNum	1.3.6.1.4.1.13418.506	MUST
Desc	description	1.3.6.1.4.1.13418.507	MUST

objectclass = locObject

```
procedure
   GetReplicatedServerList()
 1. ld := ldap에 대한 연결 객체;
   /* LDAPConnection ld*/
 2. R := ∅;
    /* replicated server의 리스트, 테스트를
       수행할 서버의 목록을 작성 */
 3. s := test를 원하는 서비스 서버;
 4. res := ldap검색 후 결과 저장을 위한
    리스트;
    /* LDAPSearchResults res */
 5. ld.connect(host, port);
    /* host = LDAP server ip, port =
     LDAP server port */
 6. res :=ld.search(( MY_SEARCHBASE
SCOPE_ONE, filter, attrs, false );
     /* filter의 검색조건에 찾고자 하는
      서비스 제공 서버의 특성 명시 */
 7. for Si ∈res에 대해  R := R ∪ {Si};
    /* Si 를 R에 추가 */
 8. return R  /* 구성된 Replicated server list */
endprocedure
```

<그림 5> LDAP으로부터 중첩서버 리스트 생성

3.6 최적의 서버 선택을 위한 단계별 방법

단계 1) 사용자가 원하는 파일의 이름을 얻는다.

단계 2) 중첩서버의 리스트인 req_list를 구성한다.

클라이언트는 LDAP서버에 연결하여 사용자가 입력한 서비스를 제공할 수 있는 서버의 리스트를 요청한다. 후에 LDAP서버로부터 디렉토리 서비스의 응답으로써 리스트를 받으면, req_list를 구성하고 연결을 종료한다.

단계 3) 적절한 서버를 선택하기 위하여 테스트 메시지를 요청한다.

Req_list의 서버 이름을 로딩하여 req_list의 끝까지 순회하면서 테스트 메세지를 요청한다.

단계 4) 서버로부터 도착한 응답시간의 리스트를 작성한다.

서버로부터 도착한 응답시간을 체크하여 계산한 후에 이의 결과를 res_list라는 응답시간 리스트로 구성한다.

```
procedure ServerSort()
 1. array[] := 다운스트림 응답 시간의 배열;
 2. left := array의 왼쪽 boundary index;
 3. right := array의 오른쪽 boundary index;
 4. pivot := array[left] /* array[]의 value의 기준값 */
 5.  i := left;
 6.  j := right+1;
 7.  while (array[i] < pivot)
         i++;
 8.  While (array[j] > pivot)
         j--;
 9.  if ( i < j )
         SWAP(array[i], array[j]);
 10.  ServerSort array, left, j-1);
      /* j를 기준점으로 recursive 서버정렬 */
 11.  ServerSort(array, j+1, right);
 12. return min(Array[])
endproedure
```

<그림 6> 각 서버별 측정시간을 정렬

단계 5) 최적의 서버를 선택한다.

res_list의 데이타를 읽어들여 다른 res_list의 서버의 응답시간과 비교하여 res_list에서 가장 빠른 응답시간을 가지는 서버를 선택한다. <그림 6>은 서버의 응답시간들을 배열에 담아 정렬하여 원하는 서버를 선택하기 위해 처리하는 알고리즘이다.

단계 6) 최적의 서버에 접속한다.

선택된 최적의 서버에 접속하여, 원하는 서비스를 제공받는다.

4. 실험 및 고찰

4.1 구현 환경 및 사용 언어

본 서버측에서의 서버선택 모델을 실험하기 위하여 다음과 같은 시뮬레이션 환경을 구축하였다. visual C++ 5.0 이상의 라이브러리와 Windows API를 지원할 수 있는 윈도우 상에서 시뮬레이션을 수행하였으며, 그 환경과 시스템 사양은 다음 <표 2>와 같다.

<표 2> 시스템 사양 및 테스트 환경

구 분	종 류	사 양
Test Systems	CPU	Intel Pentium III 733 / 900 MHz
	Main Memory	512 MBytes
	OS	Windows 2000 Professional
	Network Interface Card	3Com EtherLink XL 10/100 PCI Tx NIC (3C905B-TX)
	IP Address	203.252.53.46 / 203.255.252.144
Server Agent	CPU	Intel Pentium III 733 / 900 MHz
	Main Memory	512 MBytes
	OS	Windows 2000 Professional
	Network Interface Card	3Com EtherLink XL 10/100 PCI Tx NIC (3C905B-TX)
Test Network	Domain A	203.252.53 network
	Domain B	203.255.252 network
LDAP Server	OS	Solaris 2 .7

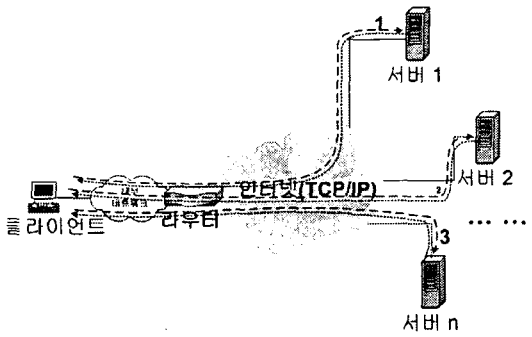
4.2 테스트 환경구축 시나리오

1) 영역은 크게 두 개로 구성하였으며 203.252.53.40부터 203.252.53.59까지의 IP주소를 사용하는 호스트 20대로 영역 A를 구성하였고, 203.255.252.141부터 203.255.252.151까지의 IP주소를 사용하는 호스트 10대로 영역 B를 구성하였다.
2) 영역 B는 203.255.252 네트워크에 서버용

시뮬레이션 에이전트를 설치하였다.

3) 구성된 두 개의 영역 중에서 203.252.53.46의 주소를 갖는 시스템에 클라이언트를 설치하였으며 203.252.53.41의 주소를 갖는 Solaris 시스템에 LDAP을 구축하였다.

4) 실험은 <그림 7>과 같은 방법으로 서버선택 측정을 수행한다. 클라이언트에서 각 서버로 서버선택 측정을 요청하는 메시지를 동시에 보내고, 이후에 각 서버들이 보내는 응답메시지 중에서 다운스트림 측정이 가장 빠른 서버를 선택하게 된다.



<그림 7> 클라이언트의 요청 후 서버측의 다운스트림 측정 수행

4.3 기존 모델과 서버측 다운스트림 측정모델의 성능 비교

4.3.1 기존 모델에서 측정의 문제점

기존 모델에서 클라이언트에서 서버 선택을 위한 측정 시험을 수행하는 모델의 경우에 다음과 같은 문제점이 발생된다. [10]

1) 현재 네트워크 상의 부하가 생겨서 호스트와 서비스를 제공하는 서버간에 지연이 생기게 되면, 일시적인 현상임에도 불구하고 서버선택에 있어서 잘못된 측정을 행하게 된다.
2) 네트워크 상의 불규칙적인 상황으로 인하여 지연의 비정상적인 변이가 발생되면, 여러 서비스 서버들에 대한 측정에 대하여 신뢰가 떨어진다.
3) 또한 이러한 상황에서 선택된 서버는 전송시, 측정시에 얻어진 정상적인 대역폭을 유지하여 제공하기가 어렵다.
4) RFC2679에서도 IP네트워크의 비대칭(asymmetric) 구조로 인하여 소스에서 목적지까지의 경로가 역방향과 다를 수 있으므로, 각각의 방향을 고려한 트래픽 측정의 필요성을 역설하고 있다. 또한 경로가 대칭(symmetric) 구조라고 하더라도 비대칭적인 큐잉 때문에 다른 성능 특성이 나타날 수 있다고 명시하고 있다.

4.3.2 서버측 다운스트림을 이용한 측정의 특성

각 서버마다 측정의 요청과 응답의 쌍으로 구성되는 측정방법과 달리 본 논문에서는 기존에 수행한 클라이언트측에서의 측정대신에 실제 서비스를 제공할 서버의 입장에서 측정하기 위하여 서버측 다운스트림 측정을 수행한다. 동작 절차와 내용은 다음과 같다.

1) 클라이언트는 각 서버로 n번의 요청을 수행한다.
2) 각각의 요청을 수신한 서버들은 측정 데이타를 클라이언트로 송신하여 다운스트림 측정을 수행한다.
3) 클라이언트는 각 서버로부터 들어오는 응답의 속도를 바탕으로 서비스를 제공할 최적의 서버를 선택한다.
4) 특별한 과부하(overhead)가 없는 측정 요청 메시지를 동시에 전송하므로, 특정 순간의 네트워크 상황에 영향을 받지 않는다.
5) 실제 서비스를 제공할 서버입장에서 측정을 수행하여 최적의 서버 선택이 가능하다.

4.4 클라이언트측 측정과 서버측 측정의 비교

클라이언트에서 서비스를 요청할 때와 실제 데이터를 전송받을 때의 경로가 달라질 수 있다. <그림 8>과 같이 중첩서버1이 있다고 할

때, 클라이언트에서 서비스를 요청할 때의 경로는 경로2로 설정되어 있지만, 네트워크의 혼잡이나 기타의 이유로 인하여 서버에서 서비스할 수 있는 경로는 경로1로 설정될 수 있다. 또한 <그림 9>에서 클라이언트에서 서버로 서비스를 요청하고 제공받는 경로가 각각 경로2와 경로1로 설정되었다고 하자. 클라이언트에서 서버 선택을 위하여 테스트 메시지를 전송했을 때 중첩서버1과 중첩서버2에 대하여 각각 <그림 8>과 <그림 9>와 같이 설정되었다고 할 때, 전체 걸리는 시간을 T1, T2라고 하면 각각은 다음과 같이 시간이 소요된다.

$$T1 = T1_{path1} + T1_{path2}$$

$$T2 = T2_{path1} + T2_{path2}$$

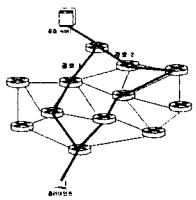
T1의 경우 경로2가 우회하여 설정됨으로써 요청하는 시간이 커지게 되어 T2가 T1보다 서비스 응답시간이 빠르게 될 수 있다. 그러나 실제 서비스를 제공하는 시간인 $T1_{path1}$ 가 $T2_{path1}$ 보다 빠를 수도 있으나 기존의 방법으로는 그 차이를 구분할 수 없다. 즉, 기존의 방법에서는 전체 응답시간만 고려하여 클라이언트에서 서버로의 왕복시간(Round Trip Time)을 고려하여 짧은 시간에 응답한 서버를 선택하는 반면, 본 논문에서 사용하는 알고리즘을 이용하면 왕복시간은 다소 느리더라도 실제 서비스를 제공하는 서버측의 다운스트림의 시간을 측정하여 선택하게 되므로, 실제 데이터를 전송받을 때의 상황에 맞는 최적의 서버를 선택할 수 있다.

또한 특정 클라이언트에서 서버k로 테스트를 요청하는데 걸리는 시간을 Tk_{path1}이라고 하고, 서버k에서 클라이언트로 응답하는데 걸리는 시간을 Tk_{path2}라고 하면, 클라이언트에서 서버선택을 위한 측정시간 T_k은 다음과 같다.

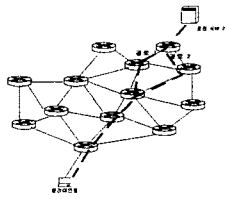
$$Tk = Tk_{path1} + Tk_{path2}$$

따라서, 전체 n개의 서버에서 측정시 걸리는 시간 T는 다음과 같다.

$$T = \sum_{k=1}^{n} Tk_{path1} + \min(T1_{path2}, T2_{path2}, \ldots, Tn_{path2})$$



<그림 8>
중첩서버1의
제공경로



<그림 9>
중첩서버2의
제공경로

그러므로 본 논문에서 제시하는 서버측 다운스트림 측정에 의한 모델의 경우에는 단순히 클라이언트측의 요청시에 RTT에 근거한 서버선택이 아니라 클라이언트에서 측정을 요청할 때의 시간과 서버측에서 응답할 때의 시간을 근거로 하여 실제 서비스를 제공받을 다운스트림 측정을 수행하므로 보다 정확한 서비스 서버를 선택할 수 있다. 전체 n개의 서버에서 m개의 서버표본을 조사하여 결정한다고 할 때 , 본 모델에서 서버 선택을 수행하기 위하여 소요되는 시간 T는 다음과 같다.

$$T = \sum_{k=1}^{n} Tk_{path1} + \sum_{k=1}^{m} \min(T1_{path2}, \ldots, Tk_{path2})$$

4.5 기존의 방법과 제안 방법의 시뮬레이션

먼저 파일 크기에 따른 응답 시간을 알아보기 위하여 각각 다음 조건에 해당하는 세 개의 서버를 선택하여 홉수와 응답시간과의 관계를 실험하였다.

조건)

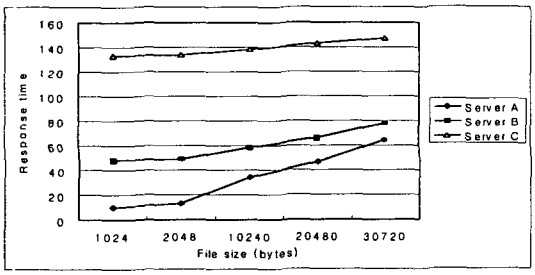
1. Server A는 우리나라 대전에 위치한 네트워크에 속한 서버이다. 테스트 지역에서 Server A까지는 7홉(hops)으로 구성되어 있다.
2. Server B는 일본에 위치한 네트워크에 속한 서버이다. 테스트 지역에서 Server B까지는 12홉(hops)으로 구성되어 있다.
3. Server C는 미국에 위치한 네트워크에 속한 서버이다. 테스트 지역에서 Server C까

위의 조건에 해당하는 Server A, B, C에 대하여 파일의 크기를 증가시키면서 응답시간을 측정하였으며, 각각의 서버에 대한 결과는 <표 3>과 같다.

<표 3> 파일 크기의 증가에 따른 응답시간 변화

server / size(bytes)	Server A	Server B	Server C
1024	9.79	48.256	132.817
2048	12.726	48.867	133.695
10240	34.104	58.033	138.272
20480	46.498	65.797	142.677
30720	63.999	77.623	147.253

또한 <그림 10>은 각 서버에서의 파일 크기별 응답시간의 변화 추이를 나타낸 그림이다. 파일의 크기가 증가함에 따라 응답시간에는 변화가 있으나 파일 크기에 따라 선형적으로 증가하지는 않는 사실을 알 수 있다.

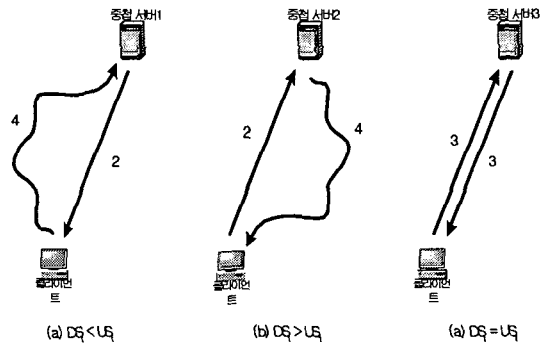


<그림 10> 테스트 서버에서 파일 크기에 따른 응답시간의 관계

Server B와 Server C는 홉수가 12로 같음에도 불구하고 응답시간에서 많은 차이가 있으며, 파일의 크기가 늘어남에 따라 응답시간의 증가율도 차이가 난다. 따라서 단순히 홉수를 가지고 서버를 선택하는 것은 현실에 맞지 않는다. 또한 파일 크기에 따른 응답 시간의 증가율은 선형적으로 증가하지 않는다. 중간 네트워크의 path MTU를 고려하지 않더라도 IP패킷의 크기가 65535 bytes를 넘지 못하므로 파일 크기의 증가에 따른 응답시간의 변화를 고려하는 것은 크게 의미가 없다.

따라서 본 논문에서는 서버에서 테스트용 메시지로서 2~3kbytes의 메시지를 사용할 것이며, 간단하게 호스트에서 원하는 서버를 선택하고, 서비스를 제공받기 위하여 신속하게 연결될 수 있도록 하기 위해 앞에서 제시한 알고리즘을 이용하여 기존의 방법과 비교할 것이다.

특정 클라이언트에서 서버로 향하는 Upstream traffic의 소요시간을 US_t라고 하고, 다운스트림 트래픽의 소요시간을 DS_t라고 할 때, 선택된 서버의 유형은 다음 <그림 11>과 같이 구분된다. 업스트림과 다운스트림에서의 숫자는 가중치(weight)를 나타낸 것이다.



<그림 11> DS_t와 US_t에 따른 서버 선택의 유형

(1) $\frac{DSt}{USt} > 1$

기존의 방법대로 단순히 RTT로만 계산하여 선택한 경우 이런 유형의 서버가 선택될 수 있다. 이 유형은 (그림 11)의 (b)와 같은 형태로서 실제 서비스를 받는 관점에서 본다면 서비스가 시작될수록 좋지 않은 성능을 유발하게 될 것이다.

(2) $\frac{DSt}{USt} = 1$

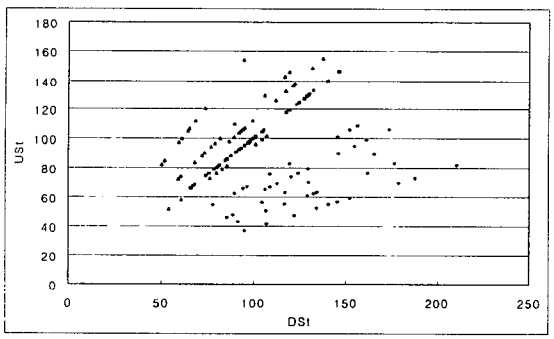
$USt = DSt = \frac{1}{2}RTT$로서 이 경우에는

어느 방법을 사용하여 서버 선택을 수행해도 거의 동일한 성능을 나타낸다.

(3) $\frac{DSt}{USt} < 1$

<그림 11>의 (a)와 같은 경우처럼 서버로 테스트 메시지를 보낼 때의 시간은 혼잡이나 기타의 네트워크 상황으로 인하여 오래 걸리지만, 실제 서비스를 받는 다운스트림의 트래픽은 최단의 경로를 취하여 서비스 제공 시간이 가장 좋은 경우이다.

중첩 서버들에 대하여 Upstream과 다운스트림의 시간측정을 수행한 결과는 다음 <그림 12>와 같다.



<그림 12> DS_t와 US_t의 서버 측정시간

<그림 12>에서 (■)기호는 US와 DS의 소요시간이 같은 경우로서 (그림 12)의 (c)유형에 해당한다. 이 값은 (a)와 (b)유형의 비교를 하기 위한 경계를 위하여 설정했을 뿐, 실제 실험에서는 거의 나타나지 않는 특성이다. $\frac{DSt}{USt}$의 값이 1보다 작은 경우인 (▲)기호일 때가 위에서 살펴본 (그림 11)의 (a)이며, 이곳에 분포하는 서버를 선택하는 것이 실제 서비스를 제공받을 때에 최적의 서비스 성능을 나타낼 것이다.

5. 결론

본 논문에서는 분산되어 설치되어 있는 여러 대의 중첩서버들 중에서 서비스를 원하는 클라이언트에게 최적의 서비스를 제공할 수 있는 서버를 선택하는 모델에 관하여 제시하였다. 즉, 대용량, 고속의 서비스를 제공하기 위하여 단일 서버에서 멀티 서버로 확장하여 구성되어 있는 환경에서 서버선택을 수행할 때 새로운 방법을 기반으로 한 모델에 대하여 제안하였다.

기존의 방법들은 클라이언트측에서 여러 가지 성능 항목(factor)들을 고려하고 RTT측정하여 서버선택을 수행하였다. 하지만, IP네트워크의 특성상 네트워크의 부하나, 일시적인 혼잡 등에 의하여 테스트를 요청할 때의 경로와 응답하는 경로가 달라질 수 있다. 만약 이렇게 경로가 달라질 때에 요청할 때의 경로가 응답할 때의 경로보다 우회하여 많은 시간이 걸리거나, 그 반대의 경우도 가능해진다.

이러한 경우에 보다 정확하게 서버를 선택하기 위해서는 실제 서비스를 제공하는 서버에서의 트래픽 방향을 고려한 서버측 다운스트림측정이 중요하다. 서버가 선택되고 서비스를 제공받을 때는 서버측 다운스트림에 의하여 성능이 결정되므로 이러한 단방향측정이 보다 현실에 적합하다고 할 수 있다.

또한 기존의 방법에서는 서버의 리스트를 정적으로 가지고 있음을 전제로 하였으나 본 모델에서는 디렉토리 서비스를 사용하여 동적으로 중첩서버들의 위치 정보를 갱신하도록 하여 항상 최신의 정보를 사용할 수 있는 모델을 마련하였다.

본 논문에서는 앞에서 제시한 여러 가지의 서버 유형들 중에서 다운스트림이 우회하는 경우를 배제하고, 서버로부터의 단방향 측정에 의하여 최적의 성능을 제공할 수 있는 서버를 선택하도록 하는 방법에 대하여 제시하였고, 이 경우에 성능의 우수성을 보였다.

현재 P2P(Peer-to-Peer)가 보급되는 추세이

며 이를 이용한 여러 가지 솔루션이나 응용프로그램이 등장하고 있다. 본 본문의 중첩서버들을 P2P의 각각의 서버들이라고 고려하면 중첩 서버에 관한 연구모델을 P2P로 확장하여 적용할 수 있을 것이다. 점차 개인컴퓨터(Personal Computer)의 성능이 개선되고, 계산파워(Computing Power)가 우수해 짐에 따라 각각의 개인 컴퓨터는 서버의 역할을 하기에 충분해 질 것이다. 따라서 보다 폭넓게 적용되는 중첩 서버 선택에 관한 연구가 필요할 것이다.

참고 문헌

[1] R.Tewari, M. Dahlin, H. M. Vin, and J. S. Kay, "Design considerations for distributed caching on the Internet" , Proceeding of the Intl. conference on Distributed Computing Systems , 1999

[2] L. Zhang, S.Floyd, and V.Jacobson, "Adaptive web caching", Proceeding of the 2nd Web Cache Workshop, 1997. 6

[3] Jiahai Yang, Peiyu Wang, Jianping Wu, "A scalable, web-based architecture for hierarchical network management" , Global TelecommunicationConference, GLOBECOM '99 , Volume: 3, page 1882 -1888 vol.35, 1999

[4] K. Ikehara, K. Ikeda, K. Motomura, "Proposal of a hierarchical network management method based on network management protocol monitoring", 2000 IEEE/IFIP Network Operations and Management Symposium, 2000

[5] Adarshpal S.Sethi, "A Hierarchical Management Framework for Battlefield Network Management", MILCOM 97 Proceedings, 1997

[6] Sandra G. Dykes, Kay A. Robbins, Clinton L. Jeffery, "An Empirical Evaluation of Client-side Server Selection Algorithm", Proceedings of IEEE Infocom, Tel-Aviv, Israel, March 2000

[7] Robert L. Carter, Mark Crovella, "Server Selection Using Dynamic Path Characterization in Wide-Area Networks", Proceedings IEEE INFOCOM '97, The Conference on Computer Communications, April 7-12, 1997

[8] Mehmet Sayal, Yuri Breitbart, Peter Scheeuermann, Redek Vingralek, "Selection Algorithm for Replicated Web Servers", Proceedings of the Workshop on Internet Server Performance, 1998

[9] James Gwertzman and Margo Seltzer, "The case for geographical push-caching", Proc of the 1995 workshop on Hot Operating Systems (HotOS-V), pp51-55, 1995

[10] G. Almes, S. Kalidindi, M. Zekauskas, "A Round-trip Delay Metric for IPPM", RFC 2681, september 1999

주 작 성 자 : 유 기 성

논문투고일 : 2004. 02. 19

논문심사일 : 2004. 10. 20(1차), 2004. 11. 01(2차), 2005. 01. 31(3차)

심사판정일 : 2005. 01. 31

● 저자소개 ●

유기성
1991. 8. 시스템공학연구소 입소
1992..1 ~ 현재 연구전산망 프로젝트수행/KISTI 슈퍼컴퓨팅센터
1998.5 ~ 1999.8 한국전자통신연구원 선임연구원
1999.9 ~ 현재 한국과학기술정보연구원 근무 (선임연구원)
2003년 성균관대학교 정보공학 석사
2004년~ 성균관대학교 컴퓨터공학 박사 과정
관심분야 : 네트워크 관리, 보안, 네트워크 성능측정

이원혁
2001년 성균관대학교 공과대학 컴퓨터공학과 학사
2003년 성균관대학교 공과대학 컴퓨터공학과 석사
2003년~현재 한국과학기술정보연구원 연구원
관심분야 : 네트워크 관리, 시스템 관리, 컴포넌트, 성능측정

안성진 (e-mail : sjahn@comedu.skku.ac.kr)
1988년 성균관대학교 정보공학과 졸업 (학사)
1990년 성균관대학교 대학원 정보공학과 졸업(석사)
1990년~ 1995년 한국전자통신연구원 연구전산망 개발실 연구원
1996년 정보통신 기술사 자격 취득
1998년 성균관대학교 대학원 정보공학과 졸업(박사)
2005년~ 현재 성균관대학교 컴퓨터교육과 부교수
관심분야 : 네트워크 관리, 트래픽 분석, Unix 네트워킹

정진욱 (e-mail : jwchung@songgang.skku.ac.kr)
1974년 성균관대학교 전기공학과 학사
1979년 성균관대학교 대학원 전자공학과 석사
1991년 서울대학교 대학원 계산통계학과 박사
1982년~1985년 한국과학기술 연구소 실장
1981년~1982년 Racal Milgo Co. 객원연구원
1985년~현재 성균관대학교 전기전자 및 컴퓨터공학부 교수
관심분야 : 컴퓨터 네트워크, 네트워크 관리, 네트워크 보안