Web Proxy Server의 시뮬레이션 환경을 위한 SSFNet의 확장

이재혁[0] 조규찬 박승규 최경희
아주 대학교 정보통신전문대학원
{maestus, netopia, sparky, khchoi}@madang.ajou.ac.kr

Extension of SSFNet for Web Proxy Server Simulations

Jae-Hyuk Lee[0] Kyoo-Chan Cho Seung-Kyu Park Kyung-Hee Choi
Graduate School of Information and Communication, Ajou University

요 약

대규모 네트워크 환경에서의 Web Proxy 서버의 효과와 유용성을 알아보기 위하여 방대한 규모의 인터넷을 실제로 구축하고 구축된 환경에서 다양한 알고리즘을 실험하는 것도 우수한 전략이나 대규모 네트워크를 모델링한 후 다양한 환경에 대하여 시뮬레이션을 수행하는 것도 여러 점에서 장점을 가지고 있다. SSFNet(Scalable Simulation Framework Net)은 대규모 네트워크의 행동을 효과적으로 시뮬레이션할 수 있는 기반을 제공하고 있다. 그러나 SSFNet을 이용하여 Web proxy의 효과를 측정하기에는 여러 가지 한계를 가지고 있다. 이에 본 연구에서는 SSFNet을 확장하고 SSFNet상에서 구동되는 Web Proxy 서버와 사용자 모델을 구축하였다. 본 연구의 결과로 구축된 시뮬레이션 기반은 다양한 Web proxy 서버의 연구에 유용하게 사용될 것으로 본다.

1. 서 론

인터넷에서 어떠한 알고리즘이나 기법을 채택한 Web Proxy 서버가 웹 서버나 클라이언트에 미치는 영향을 현존하는 네트워크 상에서 실험하고 연구하는 것은 현실적으로 어려움이 있을 뿐만 아니라 가능하다고 하더라도 많은 비용과 시간이 소모된다. 또한, Web Proxy 서버의 효용과 유용성에 대한 정확한 자료도 현존하는 네트워크의 부하나 문제점 또는 서버의 부하나 기타 클라이언트의 문제점들로 인해 실제 Web Proxy 서버에 기인하기 때문에 정확한 결과를 얻기가 대단히 힘들다.

시뮬레이션은 이러한 단점을 극복하는 훌륭한 대안이며 다양한 시뮬레이션 기반이 개발되었다. NS-2 시뮬레이션 툴은 USC(University of Southern California)에서 제작되었다. 이 시뮬레이션 툴은 이산 사건 중심 시뮬레이션이며, TCP, routing, multicast protocols에 대한 시뮬레이션을 유,무선 환경에 의거하여 수행할 수 있다.[1] 다른 시뮬레이션 툴과의 비교에서 multicast를 지원하는 점이라든지, 지역적이기는 하나 무선환경에서의 시뮬레이션을 할 수 있다는 점은 NS-2가 가진 큰 장점 중 하나이다. 그러나 100,000개 이상의 대규모 인터넷을 모델링하고, 그 환경에서 Web proxy 서버의 효과와 성능을 시뮬레이션하기 위해서는 몇 가지 단점을 가지고 있다.

SSFNet는 프로세스 기반 이산 사건 중심 시뮬레이션 커널(Process-based Discrete Event-oriented Simulation Kernel)이다. 시뮬레이션 커널인 SSF의 소스는 공개되지 않았으나 그 중에서 네트워크의 시뮬레이션을 지원하는 SSFNet은 라우터, 링크, 네트워크 인터페이스 카드 등 대부분의 인터넷 서브 시스템들을 시뮬레이션하는 데 필요한 다양한 객체들이 Java로 구현되어 있어 시뮬레이션 특성에 맞추어 그들의 특성을 변경할 수 있다는 장점을 가지고 있다.[2] 또한 SSFNet은 이를 기반으로 상위단계의 10만개 이상의 노드로 구성된 대규모 네트워크까지도 표현하도록 허용하고 있으며, 네트워크상의 실존하는 특정 행동을 따라 구현이 가능하다. 그러나 SSFNet은 네트워크 시뮬레이션의 scalability만을 강조하여 다양한 네트워크 프로토콜이나 시뮬레이션 프로세스간의 데이터의 교환이 어려운 등 단점을 가지고 있다.

이에 본 논문에서는 이러한 SSFNet의 장점은 유지하면서 단점을 보완하여 Web Proxy 서버의 성능과 효과를 시뮬레이션 가능하도록 SSFNet을 확장하였다. 또한, Web Proxy 서버와 Web 서버, Web client를 구현하여 Web Proxy에 관한 연구의 기반을 확보하였다. 본 연구의 결과 얻어진 시뮬레이션 기반은 Web proxy 서버의 행동 시뮬레이션을 통한 다양한 알고리즘의 개발을 하기 위한 도구로 유용하게 사용될 것이다.

2. Web Proxy 서버 설계 및 구현

본 연구에서는 Web Proxy 서버 모듈 구현에 따른 네트워크 특성을 연구하는 기반을 확보하기 위한 SSFNet의 확장 모델을 제안한다. 이를 위하여 우선 Web Proxy 서버의 기능을 살펴 본 후, 이의 구현을 통한 SSFNet 확장 방안을 기술하고자 한다.

2.1. Web Proxy 서버 모델

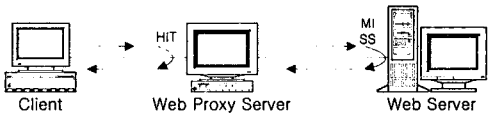

그림 1 Web Proxy Server 모델

위 <그림1>은 client, Proxy 서버, Web 서버 사이의 관계를 보여주고 있다. Client가 브라우저 상에서 Web Proxy 서버에 HttpRequest를 하며, Web Proxy 서버는 요청한 문서가 자신이 Cache 한 데이터이면, Web Proxy 서버에서 Client로 HttpRequest 에 응답을 한다. 그러나, 만약 Cache 되어 있지 않다면, Web 서버로 HttpRequest를 넘겨주며, Web 서버가 Client의 HttpRequest에 응답을 하게 된다. 그리고, Web Proxy 서버는 client에게 전달되는 문서를 자신의 선택에 따라 Cache에 저장한다. 본 연구에서는 SSFNet을 확장하고, 위 서비스 모델에 의거한 Web Proxy 서버를 SSFNet상에서 구현한다.

2.2. Web Proxy 서버 캐싱 정책

Web Proxy 서버를 위한 다양한 캐싱 정책이 제안되었으며, 그들의 성능은 다양한 환경에서 다양한 성능을 보이고 있다.[3,4] LRU(Least Recently Used)는 Relative Cache Size가 5%이상 일 경우 평균 30%의 Hit Ratio를 보이고 있으며, Greedy-Dual Size 알고리즘은 Relative Cache Size가 5%이상일 경우 평균 40%의 Hit Ratio를 보이고 있다. 본 논문에서는 가장 보편적인 캐쉬 정책인 LRU정책을 사용하였다. 30%의 Hit Ratio를 사용하여, Web Proxy 서버로 Http Request가 들어오는 패킷 중 30%의 확률로 이 패킷이 캐쉬에 있는 것인지, 아닌지를 결정토록 하였다. 예를 들면, <그림 1>에서 보는 바와 같이 Client가 Web Proxy 서버로 HTTP Request를 보내면, 우선 Web Proxy 서버가 Hit Ratio에 의한 캐쉬가 요구한 데이터를 가지고 있는지에 대해 30%의 확률을 사용하여, 캐싱을 결정하게 된다. 이 때, Hit Ratio에 의해 Hit가 된 것으로 결정된다면, 패킷을 Web Proxy 서버는 HttpRequest에 의한 응답을 자신이 하고, 만약 Miss로 결정된다면, Proxy Client를 하나 생성하여, Web 서버로 Web Proxy 서버 자신이 Http Requqest하게 된다. 이후, Web 서버에게서 Response한 데이터 메시지를 HttpRequest한 Client에게 전송한다.

2.2. Web Proxy 서버 구현

client가 요청한 패킷을 Proxy 서버 자신이 처리하거나 Web 서버로 패킷을 넘겨주기 위해서 Web Proxy 서버의 특성을 구현토록 SSFNet의 httpServer 모듈을 수정하였다. httpServer가 사용하는 httpServer 클래스는 <그림 2>에서 보는 바와 같이 우선 NIC로부터 Push되어 IP를 지나고, TCP를 지나면서 TCP header를 벗겨낸 후 httpServer에게 오게 되고, Hit Ratio에 의해 30%의 확률을 가지고, Hit인지 Miss인지를 결정하게 된다. 이때 Hit가 된다면, httpServer는 Client 에게 바로 Response해 준다.

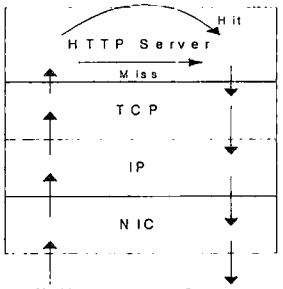

그림 2 Web Proxy Server 모듈에서 Packet 처리 동작

```
proxyGraph 1Gb [
  interface [id 0 bitrate 1000000000 latency 0.000001]
  graph [
    ProtocolSession [
      name server use SSF.OS.WWW.httpProxy
                        :
    ]
    ProtocolSession [
      name socket use SSF.OS.Socket.socketMaster]
    ProtocolSession [
      name tcp use SSF.OS.TCP.tcpSessionMaster
      _find .dictionary.tcpinit]
    ProtocolSession [name ip use SSF.OS.IP]
    ProtocolSession [name icmp use SSF.OS.ICMP] ]]
```
그림 3 Web Proxy 서버를 구현하기 위한 DML 파일

만약, Miss가 된다면, proxyClient라는 모듈을 생성하고, 이 모듈은 Web 서버에게 HttpRequest를 요청하게 되는데, Web 서버로부터 Response된 dataMessage. 즉, object size를 실제 HttpRequest한 Client에게

Proxy 서버에 대한 응답으로 돌려주게 된다. 이 dataMessage는 다시 TCP header를 붙이고, IP를 거쳐 다시 NIC를 통해 응답 받고자 했던 Client에게로 보내지게 된다. 이와 같은 Proxy 서버 클래스는 Web Proxy 서버를 구현하기 위한 <그림 3>의 DML파일에서 같은 세션에 포함되어 진다.

3. 테스트 네트워크 환경 및 동작

구현된 Web Proxy 서버를 테스트하기 위하여 <그림 4>와 같은 네트워크 환경을 구현하였다. 위 그림의 네트워크 환경은 autonomous system으로 SSF에서 기본적인 Local Network의 예를 제시한 모델이다.[2] 각각 End Node는 100개의 Client를 가지고 있고, 이 모든 클라이언트의 수는 1300개이다. 각 클라이언트 군집을 연결하고 있는 부분은 Router모듈이며, 이 네트워크의 맨 끝단에는 BGP Router 모듈이 있다. 그리고, 오른쪽 상단 위의 4개의 HTTP 서버가 있으며, Web Proxy 서버는 이 4개의 Web 서버의 내용을 캐쉬하고 있다. 끝으로, 이 네트워크의 Client들은 Web 서버와 Zipf 분포에 의해 일반적인 Http 요청과 응답이 행해지고 있게 설정하였다.

구현된 Web Proxy 서버의 동작 테스트는 Web Proxy 서버가 존재하지 않는 Client 그룹의 호스트 2개와 <그림 4>에서 보이는 Web Proxy 서버가 표시된 하단 2개의 Client 그룹에 대해 5000초 동

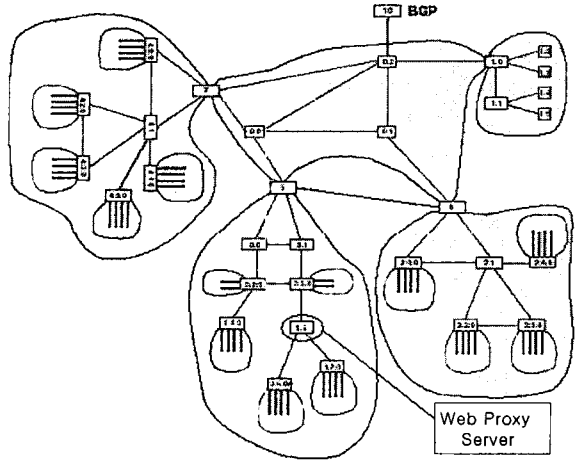

그림 4 Web Proxy Server 구현을 위한 네트워크 환경

안 시뮬레이션 하였다. <그림 5>는 그룹들이 전송 받은 데이터와 응답시간의 결과 값 중 일부이다. Web Proxy 서버가 없는 Client 그룹의 평균 초당 전송 받는 데이터들의 크기는 40016.27 Byte/초

Web Proxy 없는 PC에서 측정		Web Proxy 있는 PC에서 측정	
전송 받은 데이터(byte)	응답 시간	전송 받은 데이터(byte)	응답 시간
2073	0.259096	6229	0.097256
4793	0.173174	2976	0.393399
3433	0.173032	2775	0.072997
3764	0.259293	2441	0.072985
7063	0.259091	6851	0.473473
5082	0.259395	2340	0.361124
3325	0.173053	4993	0.393585
2119	0.172954	3283	0.073045
3034	0.259225	7485	0.07297
3701	0.173083	4270	0.073127
2958	0.173021	11227	0.488143
8109	0.345875	2045	0.361081
2104	0.259097	22712	0.073246
5573	0.17324	4024	0.361316
2871	0.259201	4407	0.369392
5445	0.259424	3026	0.369267
9065	0.345754	2749	0.048858
2488	0.172983	3167	0.36928
2961	0.173021	7159	0.48777
6756	0.173337	2013	0.369111
3288	0.173049	3196	0.385381
5902	0.173288	3898	0.073094
11673	0.173747	3389	0.361257
2343	0.172972	2459	0.369179
2289	0.172957	7248	0.09735
3047	0.259226	2140	0.048809
4526	0.259347	3582	0.073069
27210	0.345766	2097	0.048808
2828	0.265237	2632	0.393341
4448	0.177176	3401	0.073054

그림 5 클라이언트들의 전송량과 응답시간 테이블

였으며, Web Proxy 서버가 존재하는 Client 그룹의 Client들의 평균 초당 전송 받는 데이터들의 크기는 117938.5 Byte/초로 나타났다. 그 결과, Web Proxy 서버가 있는 Client 그룹들이 단위 시간당 더 많은 Object양을 전송 받는 다는 것을 알 수 있다.

4. 결론 및 향후과제

본 논문에서는 Web Proxy 서버 모듈의 구현으로 SSFNet을 확장하여 대규모 인터넷에서 Web Proxy 서버의 성능을 시뮬레이션 가능한 기반을 구축하고, 그의 정상적 구동을 확인하였다. 앞으로, 본 연구 결과를 응용하여 보다 더 대규모의 네트워크를 이용한 Web Proxy 서버의 효율적인 배치나 Cache Size의 변화, 그리고, Web Proxy 서버의 실제적인 다양한 Replacement 정책 모듈 구현, Coherency 정책 구현 등을 통한 보다 정확한 자료로 사용될 수 있는 모델 등에 대하여 시뮬레이션 하고자 한다.

5. 참고 문헌

[1] "NS-2 The Network Simulator" ,http://www.isi.edu/nsnam/ns/

[2] "SSF Simulator implementation", http://www.ssfnet.org/ssfImplementations.html

[3] Digital Equipment Corporation, "Proxy cache log traces", September 1996

[4] Jeffrey C.Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. "Potential benefits of delta-encoding and data compression for HTTP", In ACM SIGCOMM'97 Conference, September 1997.