ISBN:978-89-5884-891-2 98560

# 데이터 전송 프로그램(SCP) 성능 향상 관련 연구

일 자: 2007년 8월 3일
부 서: 연구망개발팀
제출자: 권윤주, 이길재, 석우진, 곽재승

한국과학기술정보연구원
305-806 대전광역시 유성구 어은동 52번지
TEL (042)869-0676 / FAX (042)869-0679
www.kisti.re.kr

# 데이터 전송 프로그램(SCP) 성능 향상 관련 연구

작성자: 권윤주, 이길재, 석우진, 곽재승

연구망개발팀, 한국과학기술정보연구원
{yulli, giljael, wjseok, jskwak}@kisti.re.kr
최종수정일: 2007년 8월 3일

## Abstract

*고속 데이터 전송을 위하여, TCP/IP 스택의 여러 계층에서 성능 개선 노력들이 이루어져왔다. 물리계층에서는 10Gbps의 속도를 수용할 수 있는 네트워크 인터페이스 카드들이 개발되었고, 전송 프로토콜 계층에서는 프로토콜 자체 한계를 극복하기 위해서 다양한 성능 향상 프로토콜들이 개발되고 있다. 그러나 아직까지 실제로 고속전송을 필요로하고 있는 응용 그룹(고에너지 물리 분야, 천체물리 분야 등)에서는 낮은 데이터 전송 속도로 원활히 협업연구를 진행하지 못하고 있다. 그 이유는 네트워크 하드웨어 기술, 전송 프로토콜 기술 등은 개발되어 왔으나 데이터 전송을 위해서 사용되고 있는 응용 프로그램의 경우 아직 개선 작업이 많이 진행되지 않았기 때문이다. 이에 본 고에서는 SCP의 고속 데이터 전송에서의 한계점에 대해서 알아보고, SCP의 한계를 개선한 HPN-SCP에 대하여 기술하고, 그 성능에 대하여 비교 분석하도록 한다.*

## Topics



## 1. 서론

데이터 전송을 빠르게 하기 위하여, TCP/IP 스택의 여러 계층에서 개선 노력들이 이루어져왔다. 물리계층에서는 10Gbps의 속도를 수용할 수 있는 네트워크 인터페이스 카드들이 개발되었고, 전송 프로토콜 계층에서는 프로토콜 자체 한계를 극복하기 위해서 다양한 성능 향상 프로토콜들이 개발되어 왔다. 특히 BIC-TCP의 경우 리눅스 커널 2.6.9이상에서 디폴트 TCP로 사용될 정도로 그 성능을 인정받았다. 이러한 전송 프로토콜들의 전송 능력은 약 7Gbps정도로 보고되고 있다.

그러나 아직까지 실제로 고속전송을 필요로하고 있는 응용 그룹(고에너지 물리 분야, 천체물리 분야 등)에서는 낮은 데이터 전송 속도로 원활히 협업연구를 진행하지 못하고 있다. 그 이유는 네트워크 하드웨어 기술, 전송 프로토콜 기술 등은 개발되어 왔으나 데이터

전송을 위해서 사용되고 있는 응용 프로그램의 경우 아직 개선 작업이 많이 진행되지 않았기 때문이다.

특히 KREONET에서 지원 대상으로 고려하고 있는 첨단과학응용 그룹에서는 보안성을 인정받고 있는 대표 응용프로그램만을 외부에 오픈하고 있어서 scp, rsync 등에 대한 성능 개선이 요구되고 있는 실정이다. 이에 본 고에서는 scp의 고속 데이터 전송에서의 한계점에 대해서 알아보고, scp의 한계를 개선한 hpn-scp에 대하여 기술하고, 그 성능에 대하여 비교 분석하도록 한다.

## 2. 기존 SCP의 한계

OpenSSH에 구현되어 있는 SCP는 TCP에서 사용하는 flow control buffer가 고정적으로 사용되고 있어, 이로 인해 데이터 전송 성능의 한계를 가져오고 있다. 이러한 버퍼들은 특히 원거리이며 고대역폭 네트워크 링크일 때 SCP 전송성능의 병목점으로 작용되고 있다.

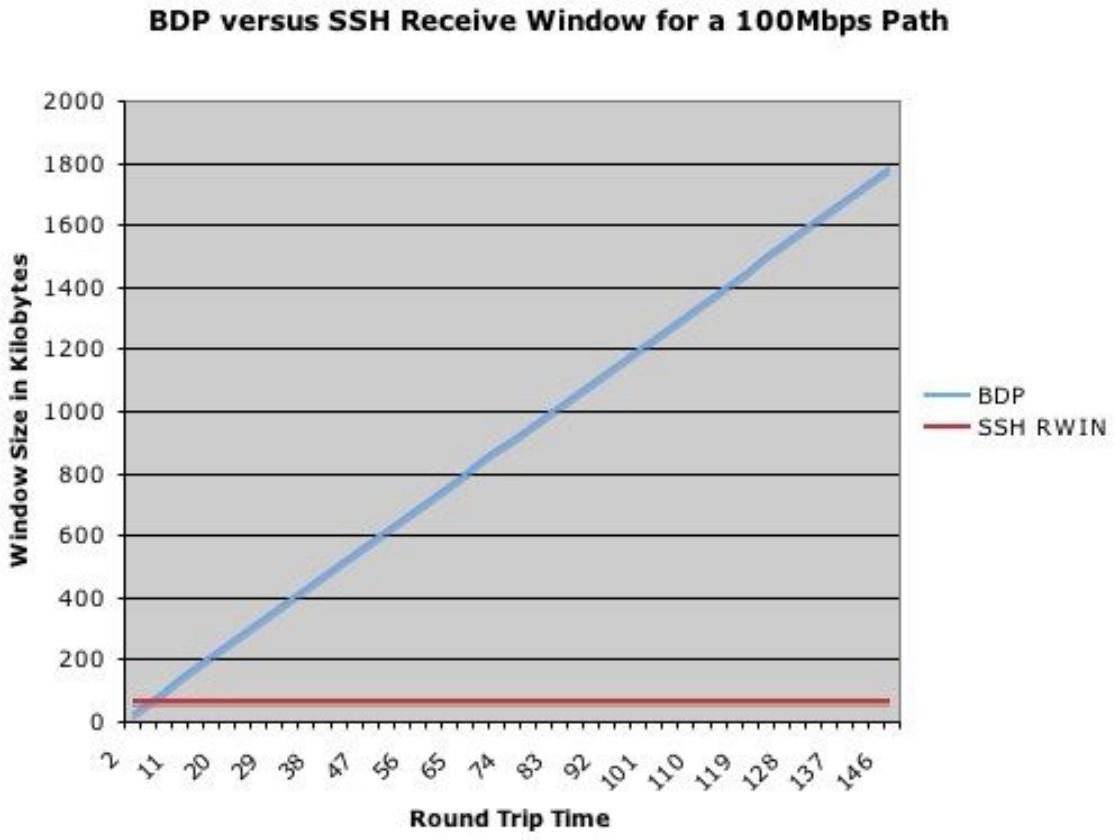


그림 1) 100Mbps에서의 BDP와 SSH 수신윈도우 크기[1]

그림 1)에서 보는 바와 같이, 전송거리가 길어지고 네트워크 대역폭이 커질수록 BDP(Bandwidth Delay Product)는 증가되는 데 반해, SSH가 사용할 수 있는 수신윈도우의 크기는 일정한 값으로 고정되어 있다. 이것은 SSH의 전송 능력과 네트워크 링크가 가지고 있는 대역폭 사이에 '성능 차이(Performance Gap)'를 야기시켜 가용 네트워크 링크를 낭비하게 만드는 결과를 초래하고 있다.

## 3. 개선된 SCP

피츠버그 슈퍼컴퓨팅센터에서는 OpenSSH의 병목 현상을 개선할 수 있는 패치 프로그램(HPN-SSH)을 개발하였다. HPN-SSH에서는 기존 SSH에서 64KB로 제한되어 있는 송수신 버퍼를 증가시킴으로써 그림2)에서 보는 바와 같이 데이터 전송속도를 높일 수 있었다. 그림 2)의 HPN-SSH 성능의 차이는 암호화 방식에 따른 결과이다.

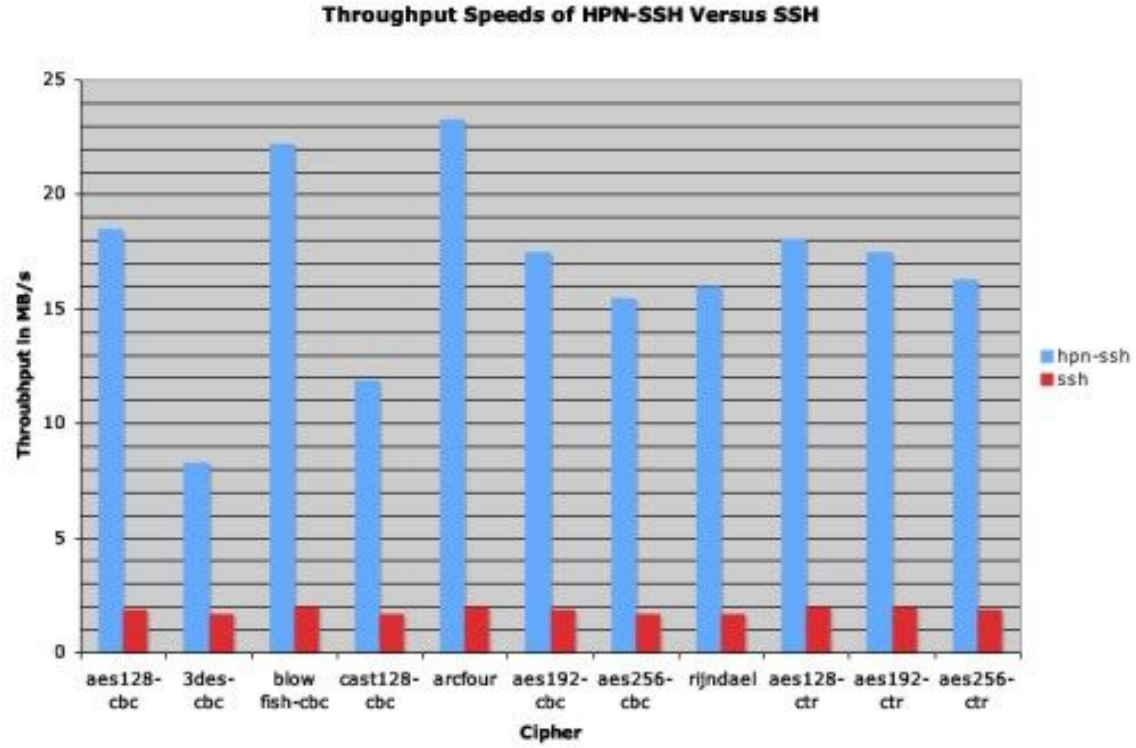


그림 2) HPN-SSH와 기존 SSH의 전송속도 비교[1]

실제 테스트 했을 때 HPN-SSH는 시스템 사양과 메모리만 보장된다면 기존의 SCP에 비하여 50~60배 정도의 전송 속도 향상을 볼 수 있었다. 이 경우, SSH 서버와 클라이언트가 모두 패치되어 있어야 하며 송수신 버퍼에 따라 전송성능의 차이가 나타날 수 있다. 만약 클라이언트만 패치하게 되는 경우, 송수신 버퍼를 충분히 세팅해주었다면 전송속도는 600KB/s 정도 수준에서 수렴될 것이다.

### 3.1 패치 방법

HPN-SSH는 OpenSSH3.9부터 현재 OpenSSH4.6까지 지원하고 있으며, 각각의 패치 파일들은 피츠버그슈퍼컴퓨팅센터(http://www.psc.edu/networking/projects/hpn-ssh) 사이트에서 다운받을 수 있다. 다운받은 패치파일은 기존 SSH 소스파일에 'patch' 유틸리티를 이용하여 소스들을 업데이트한다. 패치 및 재인스톨 과정은 다음과 같다.

```
[root@proxy2 packages]# ls
openssh-4.6p1       openssh-4.6p1-hpn12v17.diff
```

```
[root@proxy2 packages] patch  -p0 < openssh-4.6p1-hpn12v17.diff
[root@proxy2 packages]# cd openssh-4.6p1
[root@proxy2 openssh-4.6p1]# ./configure
[root@proxy2 openssh-4.6p1]# make
[root@proxy2 openssh-4.6p1]# make install
```

'configure'시 인스톨 패스를 지정하지 않았다면 hpn-ssh의 관련 프로그램들은 /usr/local에 저장된다. 따라서 sshd, ssh, scp등의 위치는 /usr/local/sbin/sshd, /usr/local/bin/ssh, /usr/local/bin/scp이며 제대로 패치된 프로그램인지를 확인할 때는

```
ssh  -V
```

옵션을 이용하여 버전을 확인한다.

ex1) 패치 전의 ssh

```
[root@proxy2 openssh-4.6p1]# ssh -V
OpenSSH_4.3p2, OpenSSL 0.9.8a 11 Oct 2005
```

ex2) 패치 후의 ssh

```
[root@proxy2 openssh-4.6p1]# /usr/local/bin/ssh -V
OpenSSH_4.6p1-hpn12v17, OpenSSL 0.9.8a 11 Oct 2005
```

### 3.2 네트워크 관련 버퍼 설정

TCP는 얼마나 많은 패킷을 한꺼번에 보낼 수 있는 지를 결정하는 CWND(Congestion WiNDow)를 사용한다. 최대 CWND는 각 소켓을 위해 커널이 할당하는 버퍼 용량과 관련이 있다. 각 시스템에는 소켓 버퍼에 대한 디폴트 값은 있지만 그것은 현재의 네트워크 상황에 비해 너무 작은 값이므로, 다음과 같은 방식을 이용하여 자신의 네트워크 상황에 맞게 소켓 버퍼를 세팅해 주는 작업이 필요하다.

① 적절한 소켓 버퍼 사이즈

소켓 버퍼 사이즈는 2배의 BDP(Bandwidth Delay Product) 값 이상으로 권장한다. BDP란 네트워크 링크의 밴드위스와 목적지까지 걸리는 시간의 곱을 의미한다. 이론적으로는 지연시간을 단방향 지연값으로 얘기하지만, 실질적으로는 양방향 지연값(RTT)값으로의 적용이 망성능향상에 효과적이다. 이때, 시스템의 메모리(RAM)사이즈를 고려해서 그 크기를 넘기지 않는 범위로 소켓 버퍼 사이즈를 세팅한다. 따라서 권장 소켓 버퍼 사이즈는

socket buffer size >= 2 × bandwidth  × delay

- ***bandwidth***는 현재 시스템과 연결되어 있는 네트워크 대역폭 (예, 1Gbps)
- ***delay***는 'ping'을 통하여 구해진 전송시스템과 목적시스템 사이의 RTT/2값 (RTT : Round Trip Time, 양방향 지연시간)

예를 들어, 1Gbps 네트워크로 연결되어 있는 시스템이 RTT가 4ms인 시스템과 파일전송을 한다면, 권장되는 소켓 버퍼 사이즈는

$$1Gb/s \times 4ms \times 1/8 = 0.5MB = 512KB = 512 \times 1024\ B = 524288B$$

이상의 소켓버퍼사이즈가 요구된다. (여기서 '1/8'은 소켓 버퍼 사이즈를 BYTE 단위로 설정해주기 때문에 'bit'에서 'Byte'로 단위변화를 해주기 위한 작업임.)

② 소켓 버퍼 설정 방법

'①'에서 구한 BDP 값 이상으로 소켓버퍼 값을 설정해주어야 한다. 소켓 버퍼를 결정하는 방식에 따라 커널에 영향을 미치는 작업이므로, 이 작업은 루트에게 권한이 있다. 따라서 이 작업을 수행할 때는 루트 계정을 이용하여 실행하여야 한다.

```
/sbin/sysctl -w net.core.rmem_max=VALUE
/sbin/sysctl -w net.core.wmem_max=VALUE
/sbin/sysctl -w net.ipv4.tcp_mem="MIN DEFAULT MAX"
/sbin/sysctl -w net.ipv4.tcp_rmem="MIN DEFAULT MAX"
/sbin/sysctl -w net.ipv4.tcp_wmem="MIN DEFAULT MAX"
```

소켓 버퍼 사이즈 설정시 BYTE 단위로 인식시켜야 하기 때문에 'VALUE' 등 이탤릭체 내용은 B(BYTE) 단위로 적어주어야 한다. '①'에서 구한 값을 예로 들면,

```
/sbin/sysctl -w net.ipv4.tcp_mem="524288 524288 524288"
```

참고로 위의 문법을 띄어쓰기에 민감한 작업이므로 'space' 사용에 주의한다.

# 4. 성능 테스트 및 결과

기존 SCP와 HPN-SSH의 성능을 비교하기 위하여 원거리 두 시스템 간 데이터 전송 성능을 비교해보았다.

### 4.1 테스트 환경

한국-미국간 패치된 SCP의 전송속도를 테스트하기 위해서, 그림 3)에서 보는 바와 같이, RTT(Round Trip Time) 228ms(한국-미국거리)인 proxy1과 proxy2에서 HPN-SCP를 통해 파일전송 테스트실험을 수행하였다. 두 시스템 간 네트워크 대역폭은 1Gbps이며, 실제 데이터 전송에 사용된 파일들은 약50MB크기의 파일 680개였으며 전체 파일 사이즈는 28GB였다. 이 실험에서는 각 시스템의 송수신 버퍼를 증가시키면서 전송성능의 변화를 측정하였다.

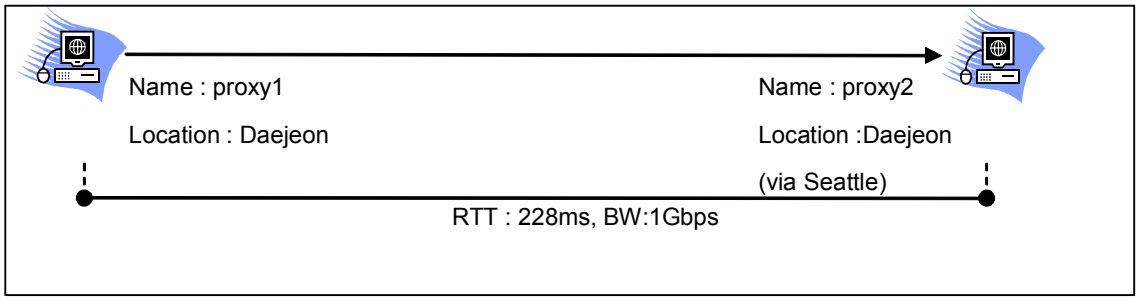


그림 3) 성능 테스트 환경

### 4.2 성능 결과

1Gbps의 대역폭, 228ms의 RTT를 가진 네트워크 환경에서 데이터 전송 성능은 기존의 SSH를 이용하였을 경우 윈도우 사이즈에 따른 성능 변화가 없었고(약 3Mbps), HPN-SSH를 이용하였을 경우에는 윈도우 사이즈에 따라 최고 평균 172Mbps(21.59MB/s)까지의 성능이 측정되었다.

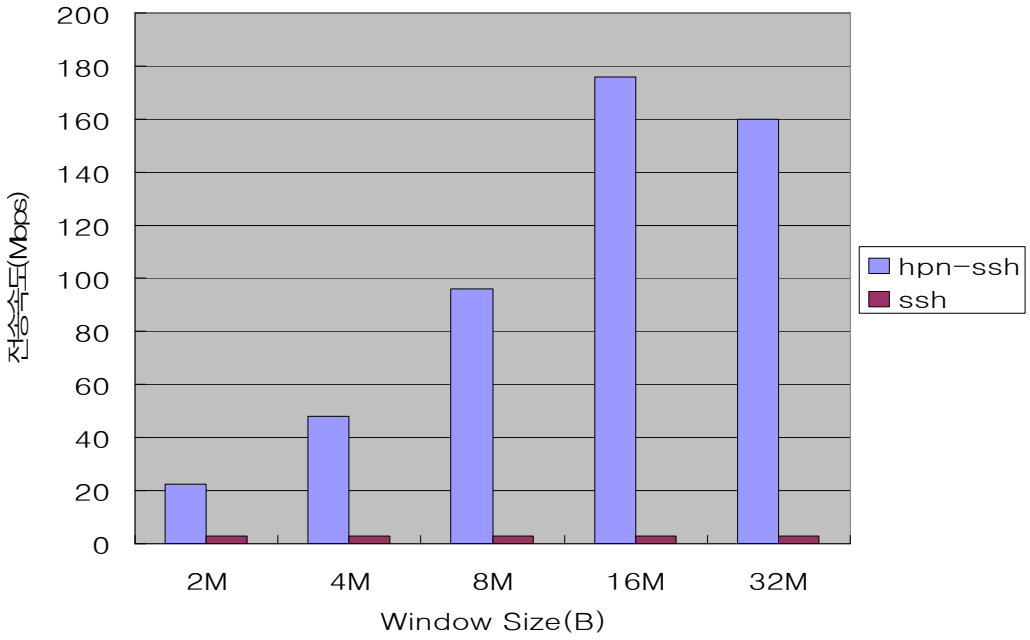


그림 4) 송수신 버퍼에 따른 데이터 전송 성능 변화

HPN-SSH의 경우 한 연결(single connection)동안 그림 5)에서 보는 바와 같이 데이터 전송 성능이 5MB/s~40MB/s이 될 정도로 변화가 심하다. 이것은 HPN-SSH가 가용 소켓 버퍼를 공격적으로 사용하여 최고 속도가 나왔다가 버퍼를 다 소진한 경우 최저 속도로 데이터 전송을 하는 것으로 추정된다.

이번 실험에서 소켓 버퍼 사이즈를 16MB로 설정했을 때 최고 전송 속도인 172Mbps(21.59MB)가 측정되었는데, 피츠버그 슈퍼컴퓨팅센터의 전송 결과 그래프인 그림

2)와 비교해 보았을 때 비슷한 성능을 보이는 것으로 보아 현재까지 HPN-SSH의 성능은 약 170Mbps인 것으로 사료된다.

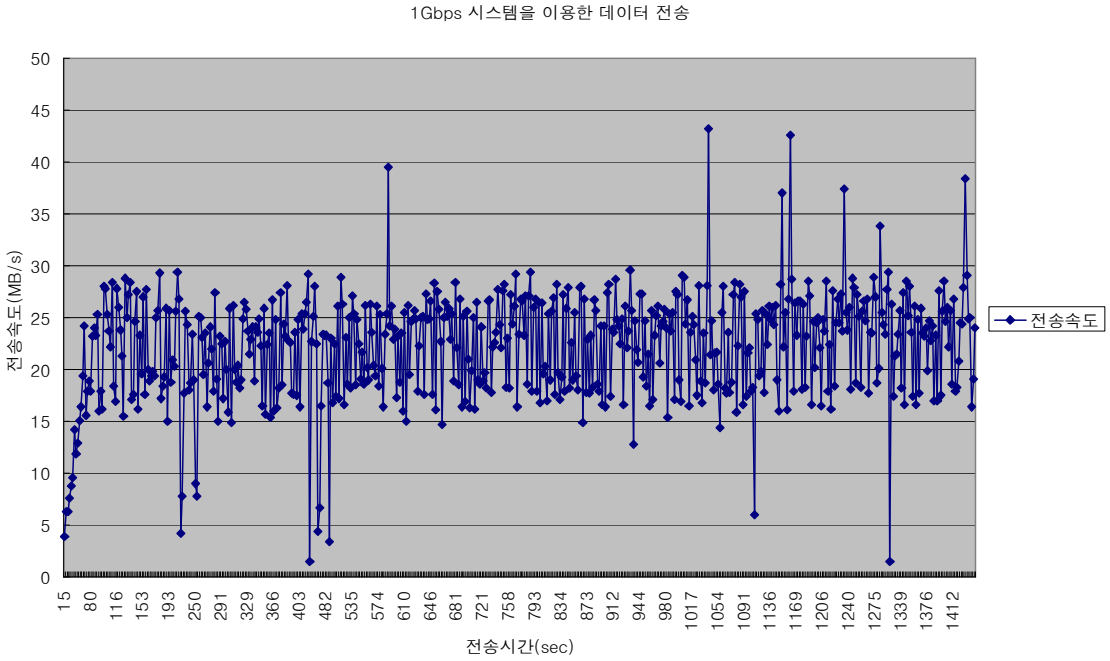


그림 5) 소켓 버퍼 사이즈를 32MB로 설정했을 때 HPN-SSH을 이용한 데이터 전송 패턴 예

## 5. 결론

데이터 전송 성능에 영향을 미치는 요소로는 전송 주체간 확보되어있는 네트워크 대역폭, 전송 프로토콜, 전송 응용 프로그램이 있다. 앞서 언급한 것처럼 네트워크 하단 요소인 네트워크 대역폭과 전송 프로토콜은 초기의 데이터 전송 성능 향상 대상이었다. 이것은 가장 큰 영향을 미치는 요소이기도 하고, 모든 데이터 전송의 공통요인이기도 하기 때문에 이 요소들의 개선을 통하여 데이터 전송 성능 향상을 기대해왔던 것이다.

그러나 하단의 요소들이 모두 성능 개선이 된 시점에서, 데이터 전송 성능 한계는 최종적으로 네트워크 이용자들이 사용하는 응용프로그램에 기인되어 있음을 볼 수 있었다. 특히 KREONET을 이용하는 첨단 과학 응용 연구자의 경우, 안정성과 보안성이 겸비된 프로그램만을 이용해야 하는 상황이기 때문에 새로운 프로그램의 개발을 통한 데이터 전송 성능 향상 보다 기존 프로그램의 문제점을 파악하여 전송 성능 향상 문제를 해결해야 했다.

이에 연구망 개발팀에서는 첨단 과학 응용 연구자들이 사용하고 있는 SCP의 한계점을 분석하고, 이를 해결하고 있는 HPN-SSH를 KREONET에 적용하고 그 성능 향상 정도를 측정하였다. 기존의 SSH에 비해서 HPN-SSH는 최대 60배정도의 성능향상이 있었고, 현재까지는 약 170Mbps 정도가 최대인 것으로 추정된다. 아직 HPN-SSH는 안정적인 프로그램이 되기 위해서 좀 더 개선해야 할 몇 가지 요인들이 발견되고 있다. 좀 더 안정적으로 HPN-SSH를 사용할 수 있는 네트워크 환경을 구축하여 KREONET 사용자들에게 서비스할 필요가 있다고 하겠다.

## 부록 : 네트워킹에 관계된 버퍼 세팅 예

```
#############################################################
# 2M 버퍼 세팅 작업 예
# setting TCP read/write buffers (min/default(pressure)/max)
#############################################################
sysctl -w net.ipv4.tcp_rmem="2097152 2097152 2097152"
sysctl -w net.ipv4.tcp_wmem="207152 2097152 2097152"
sysctl -w net.ipv4.tcp_mem="2097152 2097152 2097152"
sysctl -w net.core.rmem_max=2097152
sysctl -w net.core.wmem_max=2097152


#############################################################
# 4M 버퍼 세팅 작업 예
# setting TCP read/write buffers (min/default(pressure)/max)
#############################################################
sysctl -w net.ipv4.tcp_rmem="4194304 4194304 4194304"
sysctl -w net.ipv4.tcp_wmem="4194304 4194304 4194304"
sysctl -w net.ipv4.tcp_mem="4194304 4194304 4194304"
sysctl -w net.core.rmem_max=4194304
sysctl -w net.core.wmem_max=4194304


#############################################################
# 8M 버퍼 세팅 작업 예
# setting TCP read/write buffers (min/default(pressure)/max)
#############################################################
sysctl -w net.ipv4.tcp_rmem="8388608 8388608 8388608"
sysctl -w net.ipv4.tcp_wmem="8388608 8388608 8388608"
sysctl -w net.ipv4.tcp_mem="8388608 8388608 8388608"
sysctl -w net.core.rmem_max=8388608
sysctl -w net.core.wmem_max=8388608
```