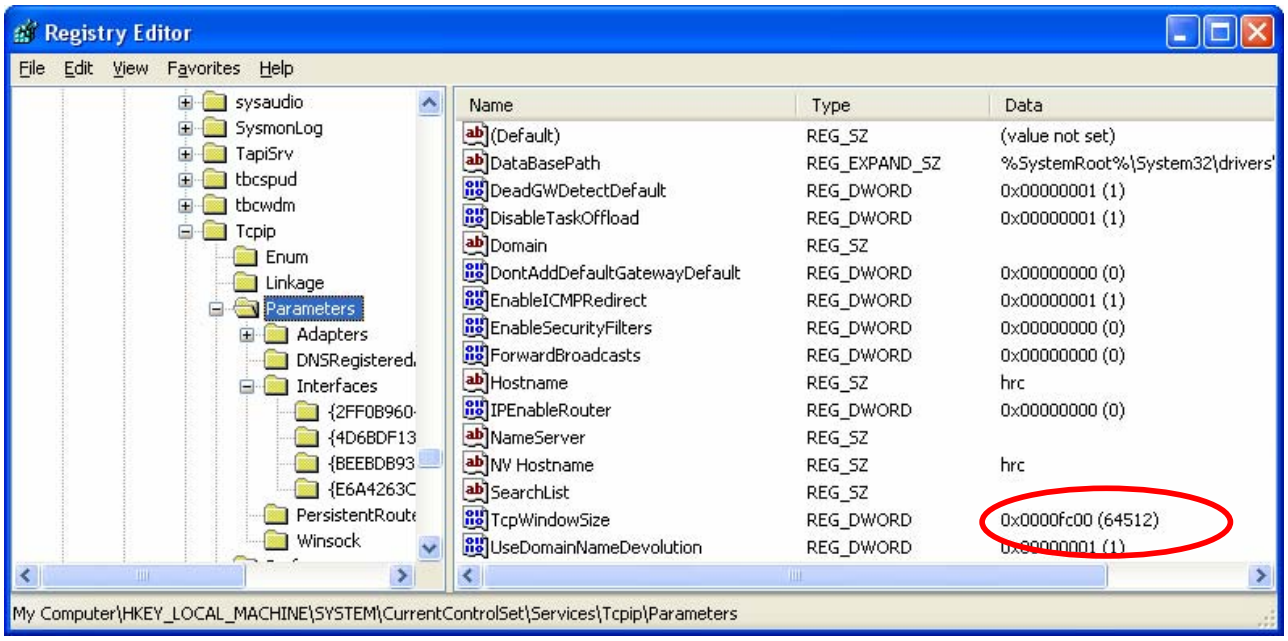
This documents describes a tool, which I use to simulate WAN latency with the purpose to demonstrate a long-haul-network and it's throughput with FTP. Doesn't matter if you are connected via a satellite connection or a long-haul-network. You may have a bandwidth of 10Mbit and you are wondering, why you only can get 25kbit/s on this link. One of the reasons is the default window size of your TCP stack on your client and the latency on your network. The basic calculation of theoretical throughput can be done as follows:

$$\text{Max. Throughput} = \frac{\text{Window-Size}}{\text{Round-Trip-Time}} \quad \text{or} \quad \text{Max. Throughput} = \frac{\text{Window-Size}}{\text{Latency} * 2}$$

What is your default window size? This depends on your operating system. Using Windows 2000 with Default Window Size of 16KB and Round-Trip-Time of 600ms will give me a theoretical throughput of 213 Kbits/s

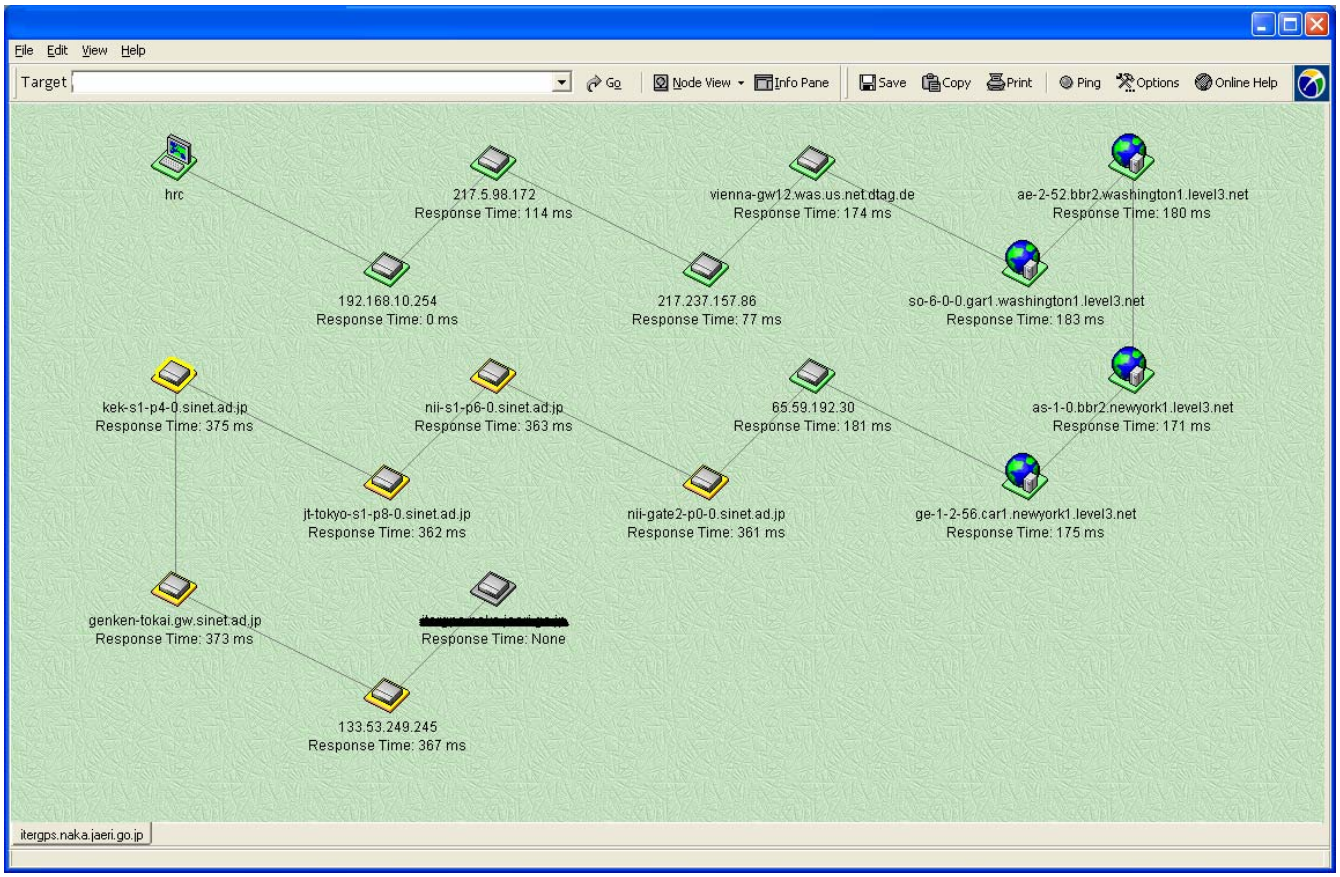
On a Window XP Professional for example, the default window size is 64kB



On other operating systems the window size is only 8kB. A table with different Operating Systems and their default Window Size can be found at www.bemsel.com/050208/window_size_matrix.jpg

On the next few pages I'm going to explain a simulated Network with high latency to "fake" a long-haul-network.

SIMULATED NETWORK



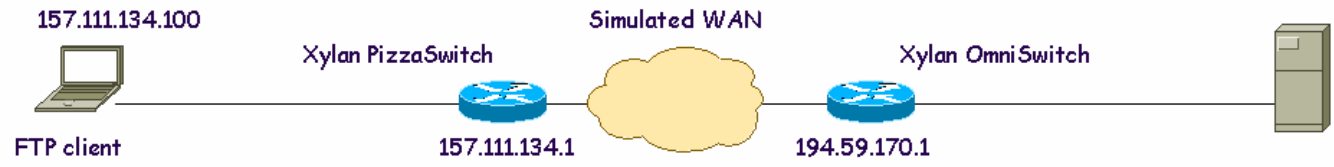
My client "HRC" did a visual trace to a host, located in Japan. Because of the long delay, the throughput on FTP is far away from the allocated bandwidth. To setup a test environment, I've used "Dummynet" to add similar latency into my test lab.

USED EQUIPMENT & SOFTWARE

- Xylan OmniSwitch & Xylan Pizzaswitch to run both routers
- Compaq Armada with WIN2000 Pro
- IBM Thinkpad T41p with WIN 2000 Advanced Server
- CuteFTP 2.0 (FTP Client)
- 3DAEMON (FTP Server)
- FreeBSD with IPFW & Dummynet on a now name Computer with AMD 1000Mhz & 2x Intel EtherExpress Pro 10/100 Network Cards

TEST LAB #1

As routers, I've used two Xylan Switches with static IP Routing Table.



OmniSwitch Info:

IP Routing Table

Network	Mask	Gateway	Metric	Group:VLAN ID	Protocol
0.0.0.0	0.0.0.0	194.59.170.1	1	1:1	STATIC
194.59.170.0	255.255.255.0	194.59.170.1	1	1:1	DIRECT

PizzaSwitch Info:

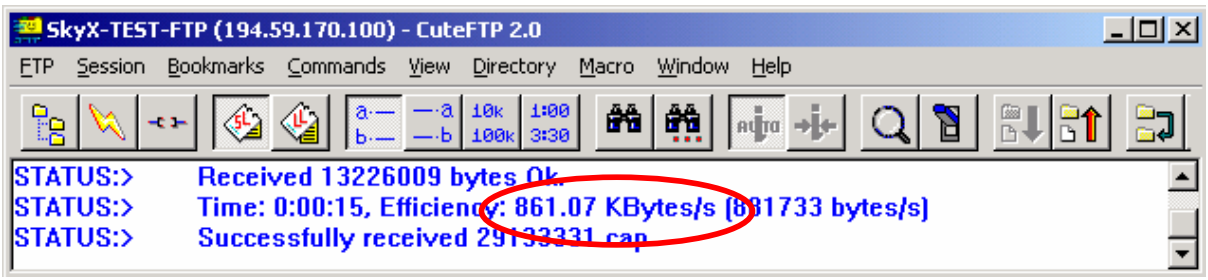
IP Routing Table

Network	Mask	Gateway	Metric	Group:VLAN ID	Protocol
0.0.0.0	0.0.0.0	194.59.170.1	1	1:1	STATIC
194.59.170.0	255.255.255.0	194.59.170.1	1	1:1	DIRECT

Round-Trip-Info (Ping from FTP Client to FTP Server - without Dummynet)

```
C:\Documents and Settings\rbemsel>ping 194.59.170.100

Pinging 192.168.10.44 with 32 bytes of data:

Reply from 194.59.170.100: bytes=32 time<1ms TTL=255
Reply from 194.59.170.100: bytes=32 time<1ms TTL=255
Reply from 194.59.170.100: bytes=32 time<1ms TTL=255
Reply from 194.59.170.100: bytes=32 time<1ms TTL=255

Ping statistics for 194.59.170.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\rbemsel>
```

Throughput FTP Client



Following next pages explain Dummynet Integration

DUMMYNET
It simulates/enforces queue and bandwidth limitations, **delays**, packet losses, and multipath effects. It also implements a variant of Weighted Fair Queueing called WF2Q+. It can be used on user's workstations, or on FreeBSD machines acting as routers or bridges.

A Floppy BOOT DISK can be found: http://www.bemsel.com/050208/pico.zip . Use included Rawrite to create the bootdisk.

In this purpose, I was trying to find a tool to simulate a Long-Haul-Network with **high latency.**

USED HARDWARE:
- AMD Processor 1000MHz
- 2x Intel EtherExpress Pro 10/100

START DUMMYNET
Insert the Floppy and boot the PC. Once you get to:

```
Please enter a hostname and IP Address for your system 00:a0:c9:xx:xx:xx
dummynet 192.168.10.40
```

It should accept your entry and prompt you with a login prompt

```
PicoBSD 0.445 (bridge)

Welcome to PicoBSD

Login as root (password "setup"

================================================================================

login: root
password: setup
#
```

VERIFY IPFW STATUS

```
# ipfw show
65535 0 0 allow ip from any to any
```

UNDERSTAND DUMMYNET

Dummynet was originally designed for running experiments on a standalone machine. The loopback interface lets you run senders and receivers on the same machine, but you should remember a few things:

Do a simple test and ping 127.0.0.1 and have a look on the delay

```
#ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.061ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=0.029ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=0.011ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=255 time=0.009ms
^C
--- 127.0.0.1 ping statistics
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.009/0.027/0.061/0.021ms
```

Add a new pipe fro 127.0.0.1 and configure a latency of 100ms

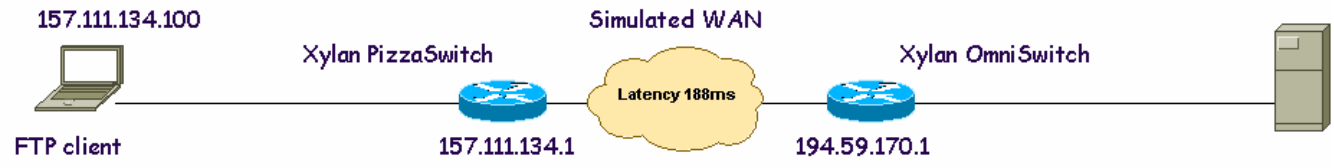
```
#ipfw add pipe 4 ip from 127.0.0.1 to 127.0.0.1
#ipfw pipe 4 config delay 100ms
```

Again, do a simple ping 127.0.0.1 you will see a delay of approximately 400ms. In fact the ICMP request goes through the pipe twice (once down, once up), and the same for the ICMP reply.

```
#ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=399.363ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=255 time=399.932ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=255 time=399.936ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=255 time=399.936ms
^C
--- 127.0.0.1 ping statistics
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 399.363/399.792/399.936/0.248ms
```

This short demonstration have shown a simple tool to add any latency you want to test. At the next section, I going to create the pipe for real IP traffic across the network.

TEST LAB #2



CONFIGURE DUMMYNET FOR TEST LAB #2

Because my host, running Dummynet is placed between the two routers, I asked the config to intercept the internet traffic and add latency. First you have to set some sysctl variables and add a new IPFW Pipe with desired delay of 188ms. This will give me a round-trip-delay of around 376ms.

```
# sysctl -w net.link.ether.bridge=1
net.link.ether.bridge: 0fxp0: promiscuous mode enabled
>> now fxp0 promisc ON if_flags 0xffff8993 bdg_flags 0x5
Fxp1: promiscuous ON if_flags 0xffff899d bdg_flags 0x5
-> 1

# sysctl -w net.link.ether.bridge_ipfw=1
net.link.ether.bridge_ipfw -> 1

#ipfw add pipe 1 ip from any to any
00000 pipe 1 ip from any to any

#ipfw pipe 1 config delay 188ms
#ipfw pipe 1 s
00001: unlimited 188ms 50sl. 1 queues (s bucket) droptail
```

Round-Trip-Info (Ping from FTP Client to FTP Server – Dummynet active)

```
C:\Documents and Settings\rbemsel>ping 194.59.170.100

Pinging 192.168.10.44 with 32 bytes of data:

Reply from 194.59.170.100: bytes=32 time=390ms TTL=255
Reply from 194.59.170.100: bytes=32 time=371ms TTL=255
Reply from 194.59.170.100: bytes=32 time=370ms TTL=255
Reply from 194.59.170.100: bytes=32 time=371ms TTL=255

Ping statistics for 194.59.170.100:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 370ms, Maximum = 390ms, Average = 375ms

C:\Documents and Settings\rbemsel>
```

Throughput FTP Client