Evaluation of Dynamic TCP Congestion Control Scheme in the ENABLE service

Mahesh Akarapu

M.S. Thesis Defense

Date: 5th Nov 2002

Committee

Dr. Victor Frost (chair)

Dr. Joseph B. Evans

Dr. Jerry James

Overview

- Introduction & Motivation
- Enable service architecture
- Dynamic TCP Congestion Control Scheme
- Evaluation
- Conclusions and future work

Introduction

- Increased interest in the TCP performance issues with the considerable increase in the Internet backbone speeds.
- Distributed applications often fail to take the full advantage of these high-speed networks.
- Reason: Improper TCP parameters (e.g., buffer sizes)
- Applications need to be “network-aware”
- Network-Aware: Adjusting the networking parameters and the resource demands to the network conditions.

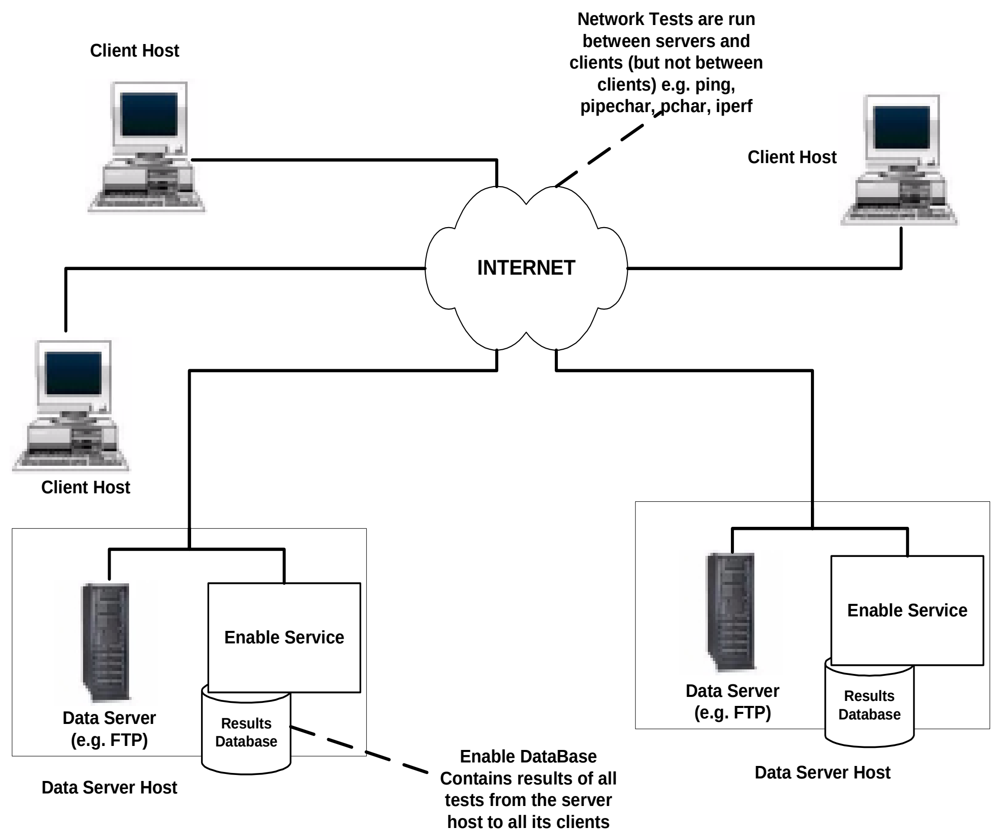
Introduction (Continued…)

- Existing work using tuning techniques.
    - Optimal TCP buffer sizes and the use of parallel sockets.
- Difficult task of determining the correct tuning parameters to use.
- Parameters are different for different networks and vary over time.
- Need for a mechanism which provides the clients with the correct tuning parameters to use.
- The solution: Enable service

Overview of ENABLE

- **E**nhancement of **N**etwork-Aware **A**pplications and **B**ott**L**eneck **E**limination
- Enables applications to optimize their use of network and achieve the highest possible throughput.
- A main component: Enable network advice server
- Associated with a data server (e.g., FTP)
- Provides clients with the correct TCP parameters to use for a network path

ENABLE Service Architecture



TCP Congestion Control limitations

- Congestion Control algorithms in TCP have a performance bottleneck on high latency links
- On links with high RTT, the time TCP spends in the slow start phase is high
- For a protocol like HTTP, the response times can be disastrous.
- A random loss of a packet in a TCP connection causes the TCP congestion control algorithms to slow down the sender.

New scheme in the Enable service

- Experimental modifications made to the TCP stack
- Enables an application to turn off the congestion control in TCP
- Taking advantage of this mechanism to improve the application performance
- Cannot turn off the congestion control totally
- Dynamically adjust the congestion control state of TCP based on the network conditions.

Overview of TCP Dynamic Congestion Control Scheme (DCCS)

- Gives the input on the congestion control state to use and the next advice time by modeling the changes in the available bandwidth on a network path.

- Estimate if the path is congested or not based on the available bandwidth on the network path.
  - NOCC if ABW > Bandwidth threshold
  - CC if ABW < Bandwidth threshold

- Next advice time is calculated based on the historical data of the available bandwidth such that the change in the available bandwidth is not greater than the threshold(5% in our case).

Application of the TCP DCCS

- FTP server (ProFTPD) as the data server
- The bandwidth on the network path can change drastically for large file transfers
- Modified to interact with the Enable advice server during the file transfers.
- Changes the congestion control state in TCP based on the Enable server input
- Advice server is contacted repeatedly during the file transfer and the congestion state is changed dynamically

How do we evaluate the TCP DCCS?

- Tests to evaluate the performance of FTP with the Enable service
  - To test if the mechanism is working properly and if it provides any performance gains
  - Tests in networks with different levels of congestion
- Tests to see the effect of the Enable service on the background traffic
  - Background (Iperf) flow is run during the FTP transfers
  - Tests with multiple background flows
- Tests with different history databases
  - How the decision process of the advice server is effected
  - Tests in networks with different levels of congestion

Test Environment

- Emulated Network Environment
- Emulate the conditions of a WAN in a lab-environment network.
- Controlled, reproducible environment for running real code.
- Valid estimation of the performance of the transfer protocols.
- NISTNet, a network emulator is used.
  - Tool for emulating performance dynamics in IP networks
  - Packet delay, congestion loss, packet reordering or duplication

Emulating network congestion

- Limit the bandwidth available to traffic by applying the packet drops
- The long-term throughput of a TCP flow and the packet drop rate is approximated by the following equation:

  $T_{TCP} = (C * S) / (RTT * \sqrt{P})$

  C is a constant, *S* is the packet size, *P* is the packet loss rate RTT is the round trip time

  $P = [(C * S) / (RTT * T_{TCP})]^2$

- Used to generate the packet drop rates for a bandwidth value.

Creation of the Test Environment

- Obtained the traffic data, i.e., available bandwidth as a function of time, on the router interface

  ks-2-a10-52.r.greatplains.net
- Router connecting the KU network to the Internet2.
- Stored in a database as a set of records with the fields of the timestamp and available bandwidth.
- Used this data as the history data for the advice server decision process and also to emulate the WAN conditions during the test time

Evaluation of the TCP DCCS

- Tests to evaluate the performance of FTP with the Enable service
- Tests to see the effect of the Enable service on the background traffic
- Tests with different history databases

Performance of FTP with the Enable service

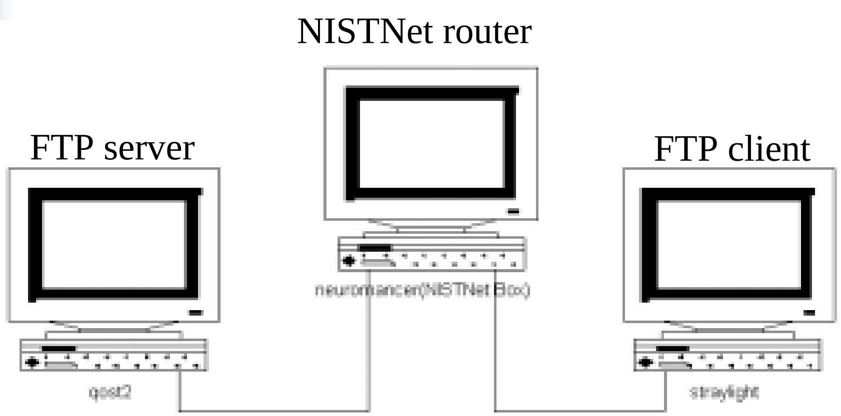
- Performance of FTP as a function of network load.
- FTP transfers of 16GByte files.
- Tests in networks with different congestion levels

  a) slightly congested network

  b) moderately congested network

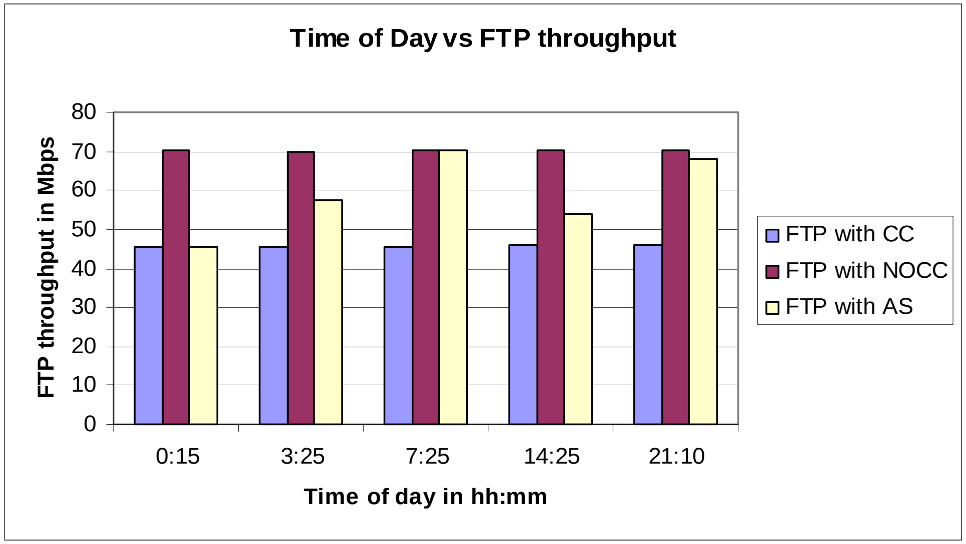
  c) highly congested network

Tests in a slightly congested network



TCP buffer size	1Mbyte
NISTNet delays	50ms
FTP transfer size	16Gbyte
NISTNet drops	YES
Available Bandwidth (ABW) Threshold	42Mbps (58% used bandwidth)
History data used	Whole database

Tests in a slightly congested network(continued…)



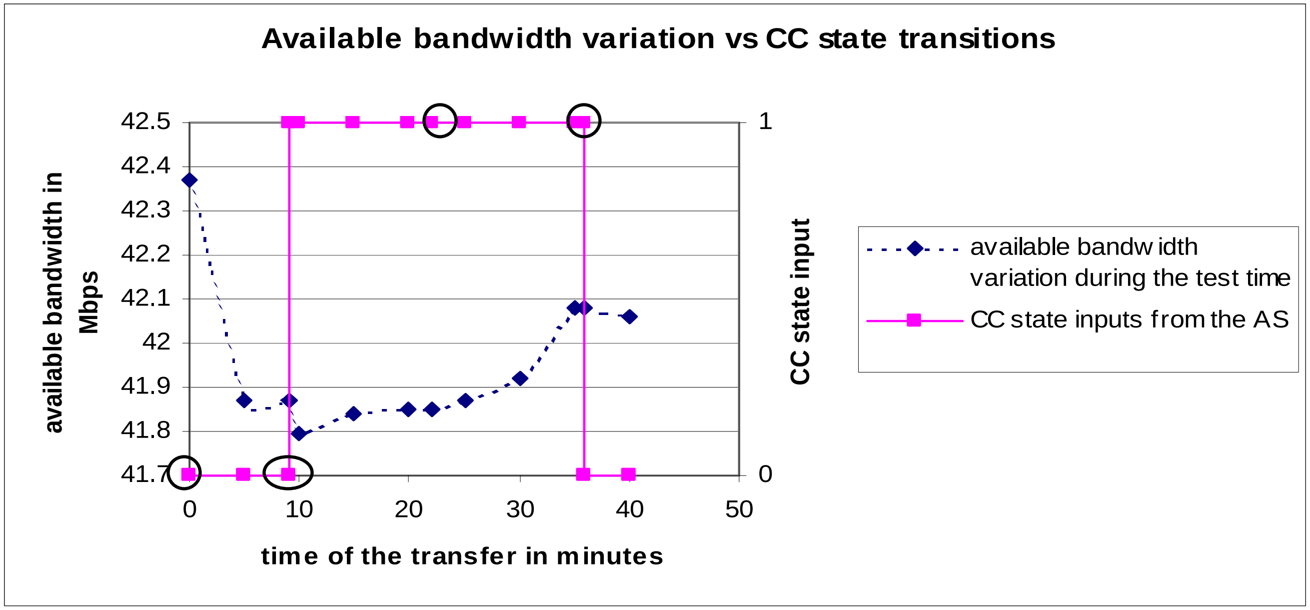
Time of day Vs FTP throughput

Time of Day	CC State inputs
00:15	1(4)
03:25	1,0(3)
07:25	0(2)
14:25	0,1(2),0
21:10	0(3),1

CC state inputs

Note: ‘0’ indicates NOCC and ‘1’ indicates CC

Tests in a slightly congested network(continued…)

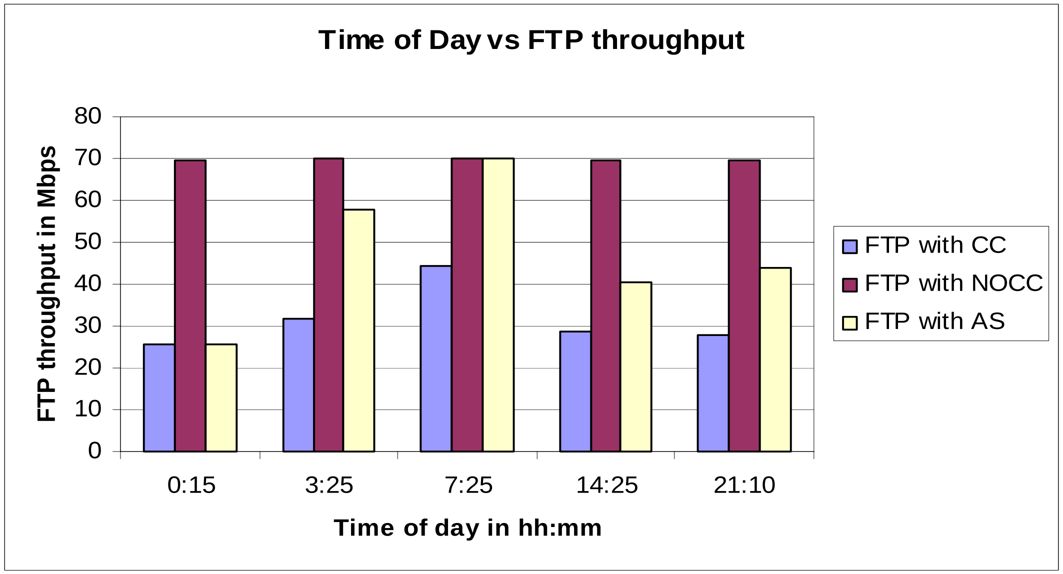


Available bandwidth variation vs. CC state transitions at test time 14:25

- Enable service gives the correct inputs about the congestion state
- Avg. Percentage improvement in the FTP throughput = 29.07%

Tests in a moderately congested network

Available Bandwidth(ABW) threshold = 27Mbps (73% used)

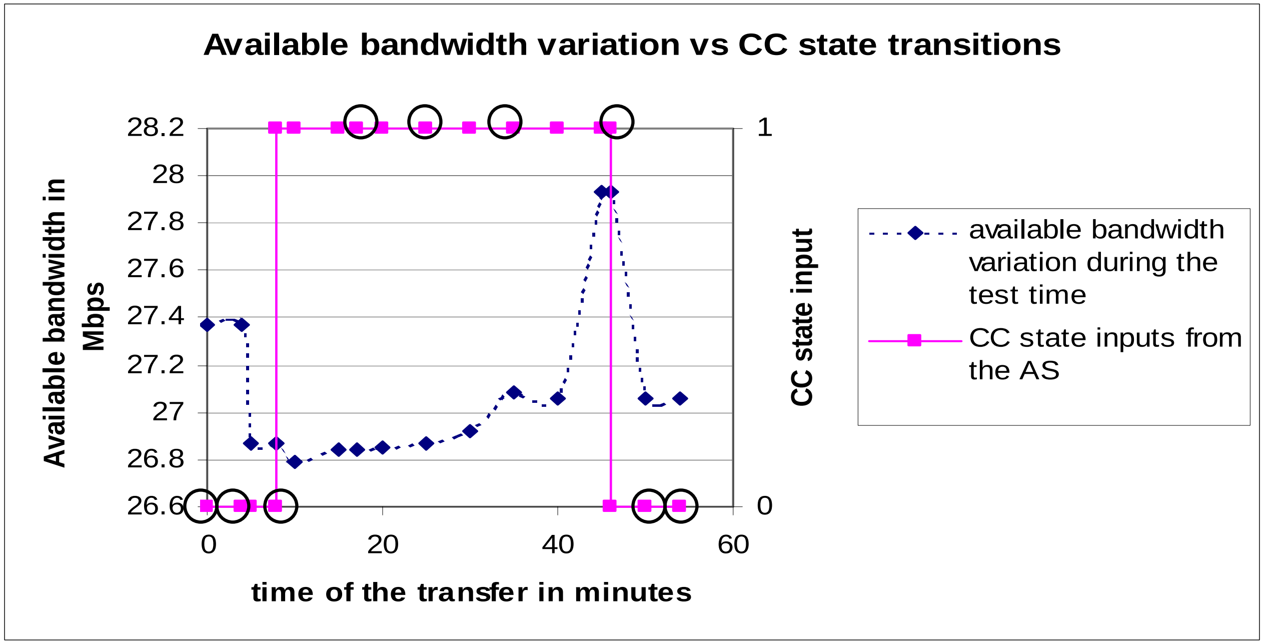


Time of day Vs FTP throughput

Time of Day	CC State inputs
00:15	1(12)
03:25	1,0(9)
07:25	0(5)
14:25	0(2),1(4),0(3)
21:10	0(5),1(4)

CC state inputs

Tests in a moderately congested network (continued…)

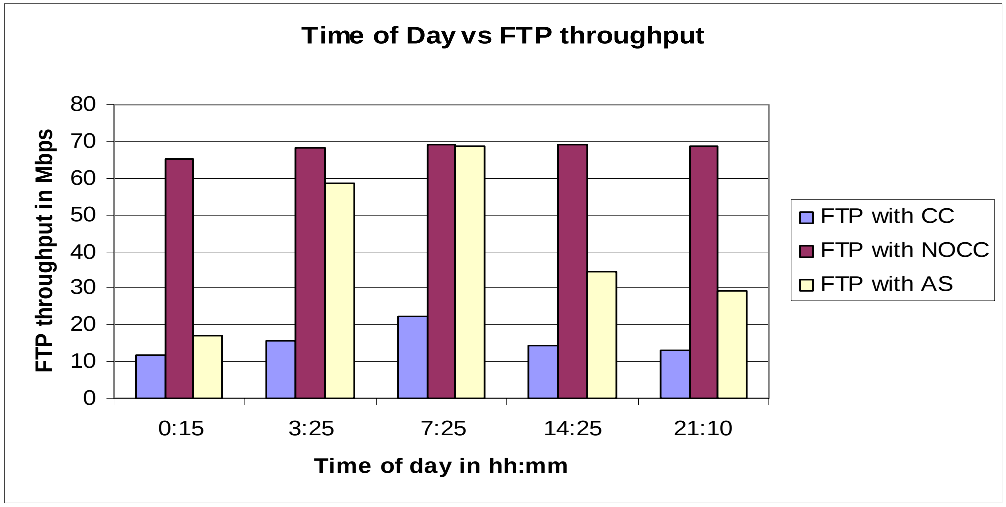


Available bandwidth variation vs. CC state transitions at test time 14:25

- Avg. Percentage improvement in the FTP throughput = 47.112%

Tests in a highly congested network

Available Bandwidth(ABW) threshold = 12Mbps (88% used)

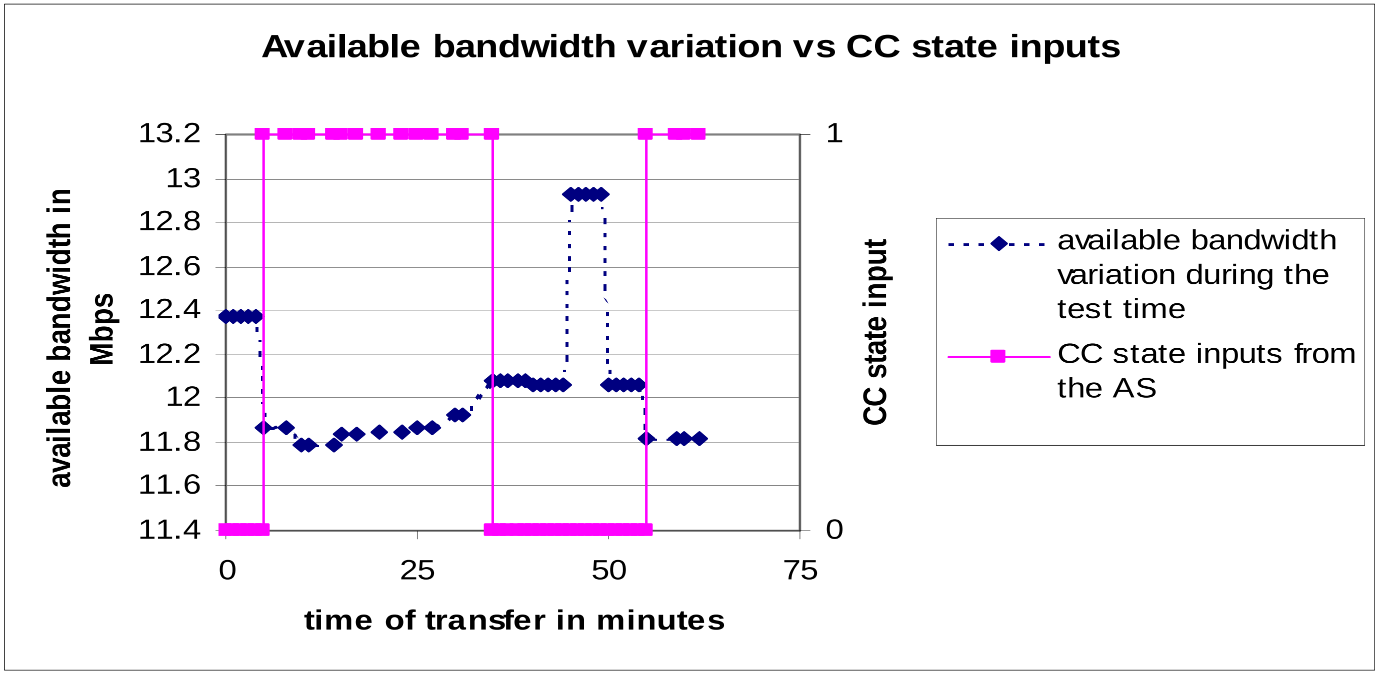


Time of day Vs FTP throughput

Time of Day	CC State inputs
00:15	1(46),0(23)
03:25	1,0(48)
07:25	0(18)
14:25	0(7),1(9),0(21),1(3)
21:10	0(24),1(6),0(4),1(9)

CC state inputs

Tests in a highly congested network (continued…)



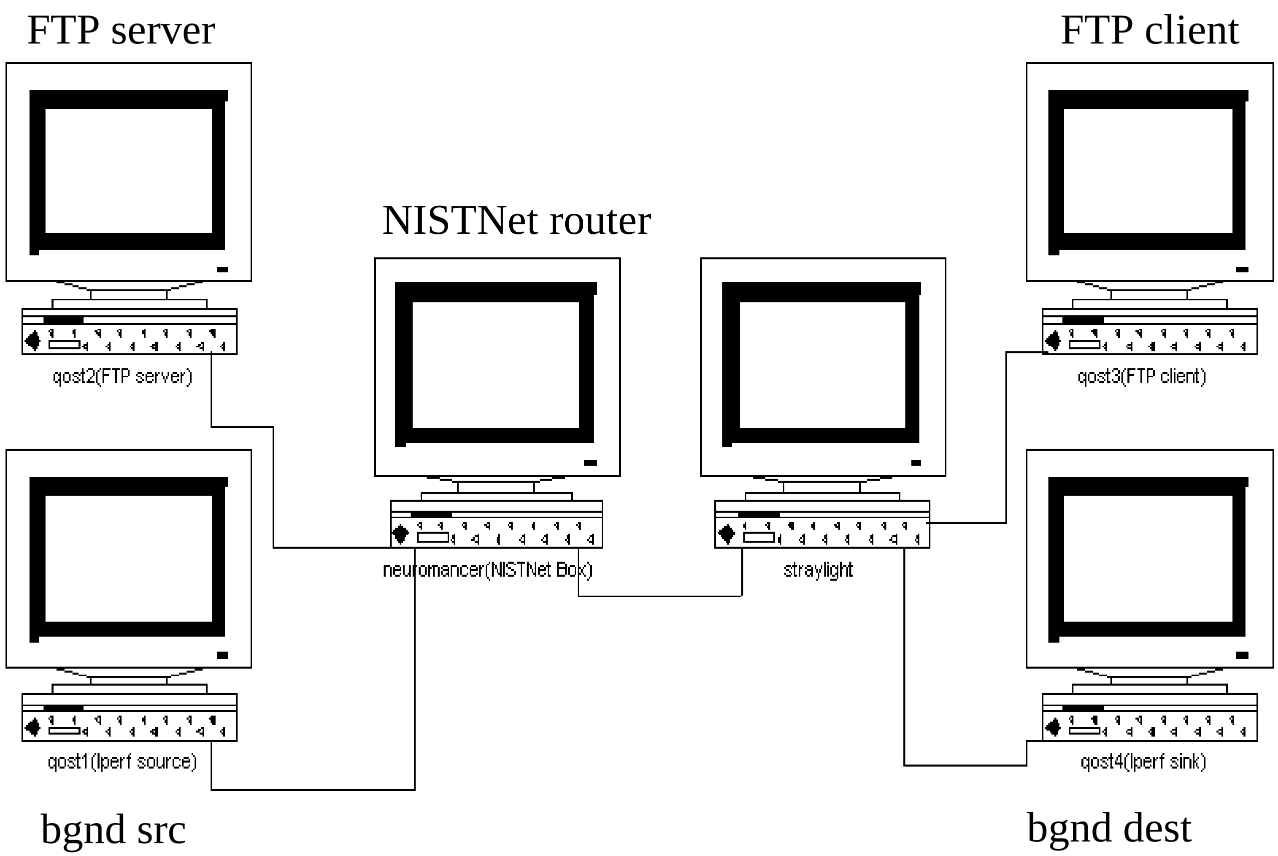
Available bandwidth variation vs. CC state inputs at test time 14:25

- Avg. Percentage improvement in the FTP throughput = 158.72%

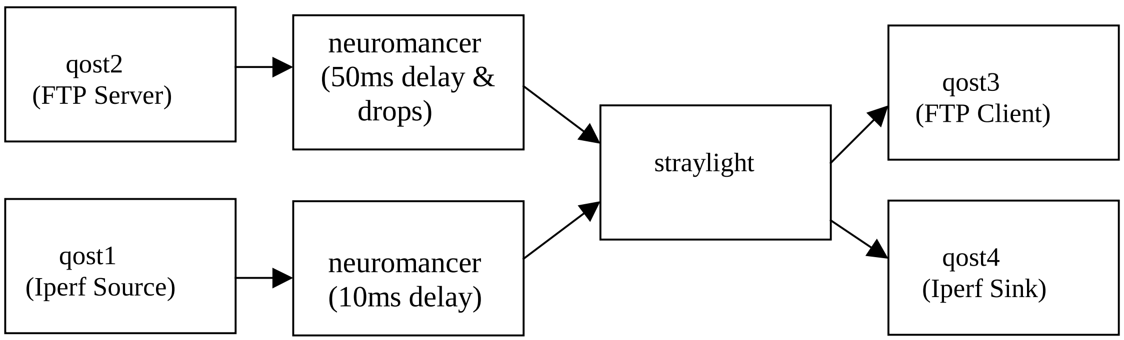
Performance of FTP with the ENABLE service

- Tests to evaluate the performance of FTP with the Enable service
- Tests to see the effect of the Enable service on the background traffic
- Tests with different history databases

Tests with a single background flow



Tests with a single background flow



Logical network topology

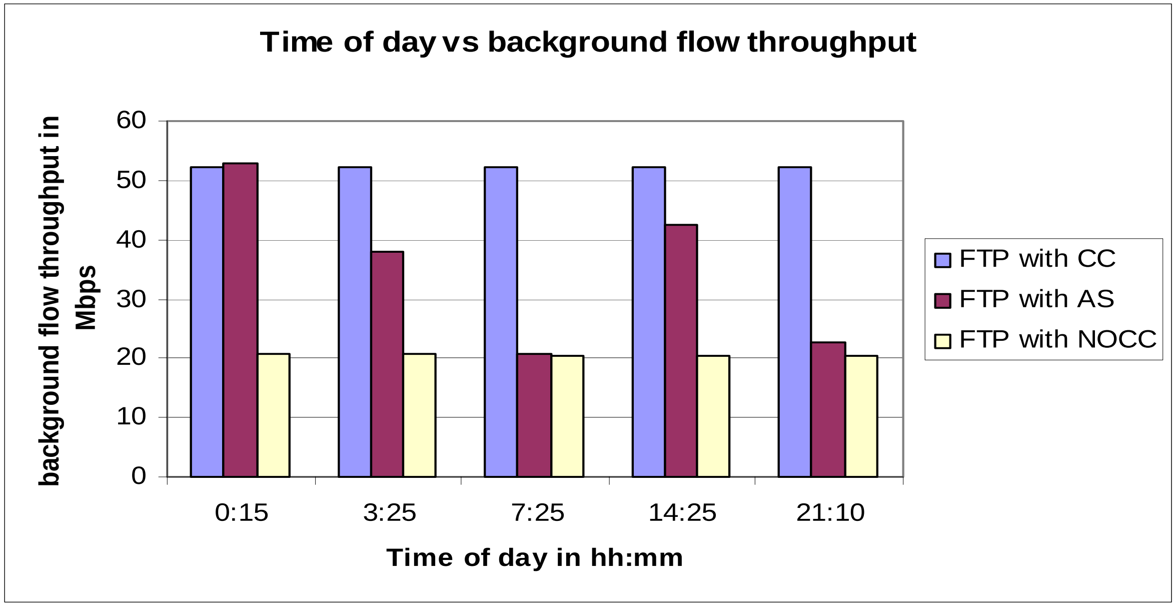
TCP buffer size	1Mbyte
NISTNet delays	50ms
FTP transfer size	16Gbyte
NISTNet drops	YES
Minimum used bandwidth	45%
Available Bandwidth (ABW) Threshold	42Mbps (58% used bandwidth)
History data used	Whole database

FTP Parameters

TCP buffer size	128Kbyte
NISTNet delays	10ms
NISTNet drops	NO

Iperf Parameters

Tests with a single background flow

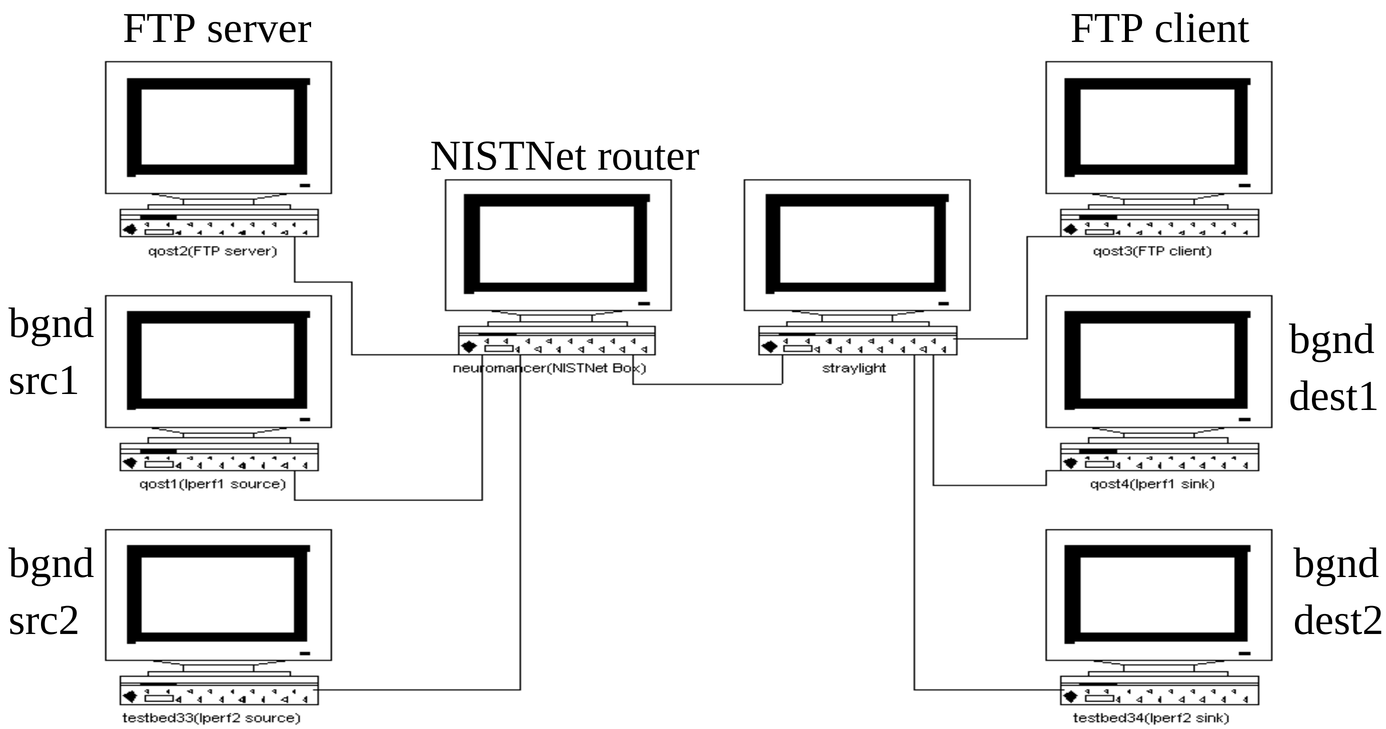


Avg.decrease in the throughput when run with FTP with CC = 37.46%

Avg.decrease in the throughput when run with FTP with NOCC = 75.30%

Avg. decrease in the throughput when run with FTP with AS = 57.65%

Tests with multiple background flows



Tests with multiple background flows

Time of day	Throughput when FTP is run with AS		
	FTP	Background flow1	Background flow2
12:15am	38.64	25.83	27.60
3:25am	52.24	23.04	24.17
7:25am	69.20	10.74	10.28
2:25pm	47.68	21.24	22.96
9:10pm	66.48	12.03	11.29

Throughput of FTP with AS when run with multiple background flows

- Advice server reduces the effect on the background flows
- Has similar effect on multiple background flows.

Performance of FTP with the ENABLE service

- Tests to evaluate the performance of FTP with the Enable service
- Tests to see the effect of the Enable service on the background traffic
- Tests with different history databases

Tests with different history databases

- 03/01/2002 Whole database 05/03/2002
- 03/01/2002 04/28/2002 One week 05/03/2002
- 03/01/2002 Friday Friday .... Friday Friday 05/03/2002

Tests in a slightly congested network

- Not much difference in the throughput
- No. of CC state inputs is highest when we use the whole database and least with previous weeks data
- No.of CC state transitions is also high with whole database

Time of Day	Throughput of FTP when run with AS in Mbps		
	Whole dB	Previous Fridays	Previous week
00:15	45.44	45.52	45.36
03:25	57.60	52.96	51.36
07:25	70.08	70.00	70.08
14:25	53.76	57.84	70.08
21:10	68.08	70.08	70.08

Time of day	CC states received when FTP is run with AS		
	Whole dB	Previous Fridays	Previous week
00:15	1(4)	1(4)	1(3)
03:25	1,0(3)	1,0	1,0
07:25	0(2)	0(2)	0
14:25	0,1(2),0	0,1	0
21:10	0(3),1	0(2)	0(2)

Tests in a moderately congested network

- Not much difference in the throughput
- The No. of CC state inputs is highest when we use the whole database and least with previous weeks data
- The No. of CC state transitions is same

Time of Day	Throughput of FTP when run with AS		
	Whole dB	Previous Fridays	Previous week
00:15	25.60	25.68	25.68
03:25	57.68	53.60	51.76
07:25	70.00	69.92	70.00
14:25	40.24	42.00	43.92
21:10	43.84	45.84	55.28

Time of day	CC states received when FTP is run with AS		
	Whole dB	Previous Fridays	Previous week
00:15	1(12)	1(8)	1(5)
03:25	1,0(9)	1,0(5)	1,0(4)
07:25	0(5)	0(4)	0(2)
14:25	0(2),1(4),0(3)	0,1(2),0(2)	0,1,0
21:10	0(5),1(4)	0(3),1(2)	0(3),1

Tests in a highly congested network

- Not much difference in the throughput

- The No. of CC state inputs is highest when we use the whole database and least with previous weeks data

- The No. of CC state transitions is same

Time of Day	Throughput of FTP when run with AS		
	whole dB	Previous Fridays	Previous week
00:15	16.88	17.68	17.52
03:25	58.40	57.04	56.88
07:25	68.56	69.44	69.52
14:25	34.72	30.72	28.08
21:10	29.28	34.16	38.08

Time of day	CC states received when FTP is run with AS		
	Whole dB	Previous Fridays	Previous week
00:15	1(46),0(23)	1(27),0(10)	1(22),0(15)
03:25	1,0(48)	1,0(26)	1,0(24)
07:25	0(18)	0(10)	0(6)
14:25	0(7),1(9),0(21),1(3)	0(3),1(6),0(7),1,0(3)	0(2),1(3),0(5),1(2),0(3)
21:10	0(24),1(6),0(4),1(9)	0(14),1(5),0(4),1(4)	0(12),1(3),0(2),1(2)

Summary of the results

- The No. of redundant inputs received is highest with whole database
- The No. of redundant inputs received is lowest with previous week's data
- More pronounced in highly congested networks
- Using the most recent data is useful

Conclusions & Future work

Conclusions

- Successfully made the FTP server to interact with the Enable service during large file transfers.
- Tested the performance of FTP with Enable service under different network congestion conditions.
- Determined the effect on the performance of the background traffic
- Performance with different history databases

Future work

- Real WAN environment
- Improve the Advisory Server decision process
- Network monitoring tools to collect the data about the network state

Questions