

A Technique for improving  
the scheduling of network  
communicating processes  
in MOSIX

Rengakrishnan Subramanian  
Masters Report, Final Defense

Guidance by Prof. Dan Andresen

# Agenda

- **MOSIX**
- Network communicating processes
- Breaking it down
- Timing
- Implementation
- Test
- Results
- Conclusion

# MOSIX - purpose

- Software tool for cluster computing
- Multiple servers work as if single to achieve high performance
- Automatic work distribution
- Load balancing
- Adaptive management (processes v/s resources)

# MOSIX – tools

- Create a process (or more processes)
- Distribute (and redistribute)
- Algorithms respond to variation
- Works on Linux x86 platforms
- Kernel patch
- System Admin tools

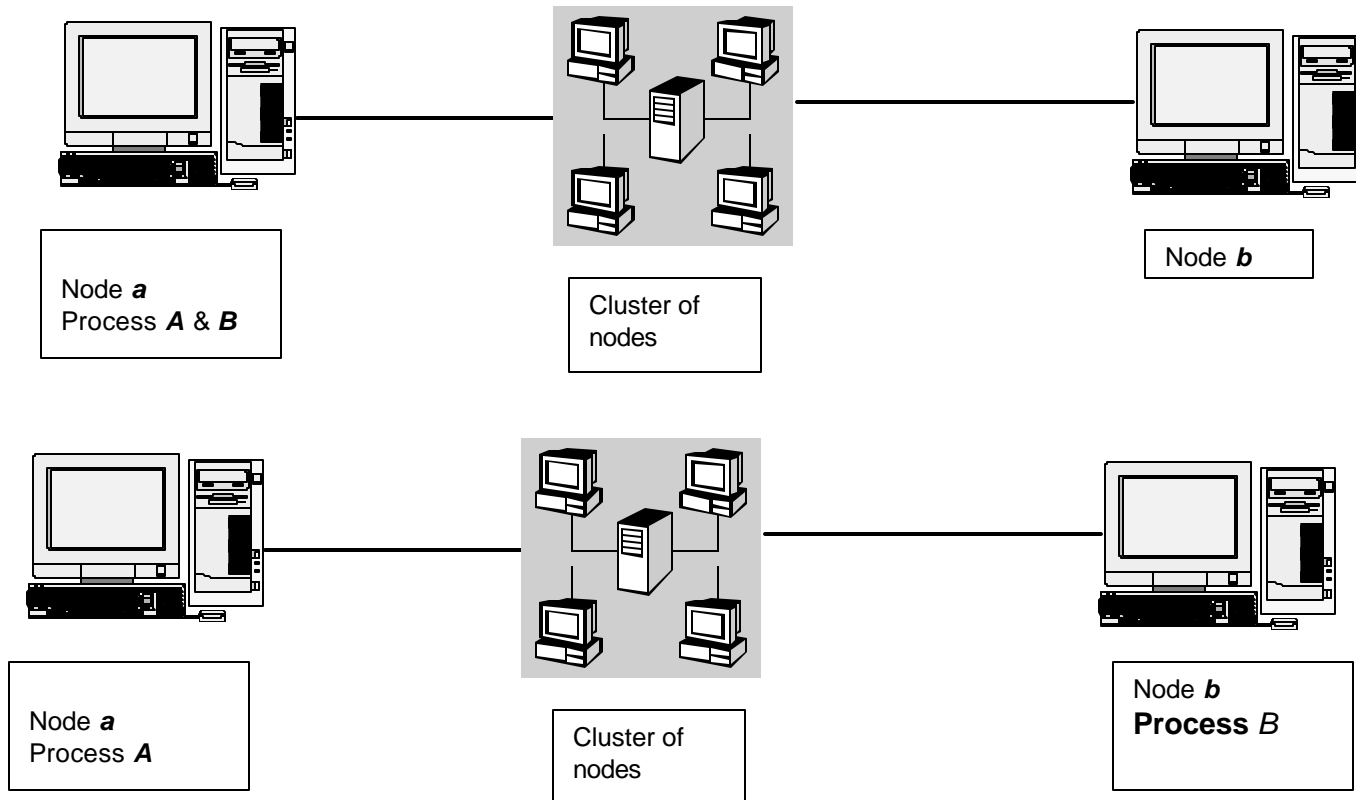
# Agenda

- MOSIX
- **Network communicating processes**
- Breaking it down
- Timing
- Implementation
- Test
- Results
- Conclusion

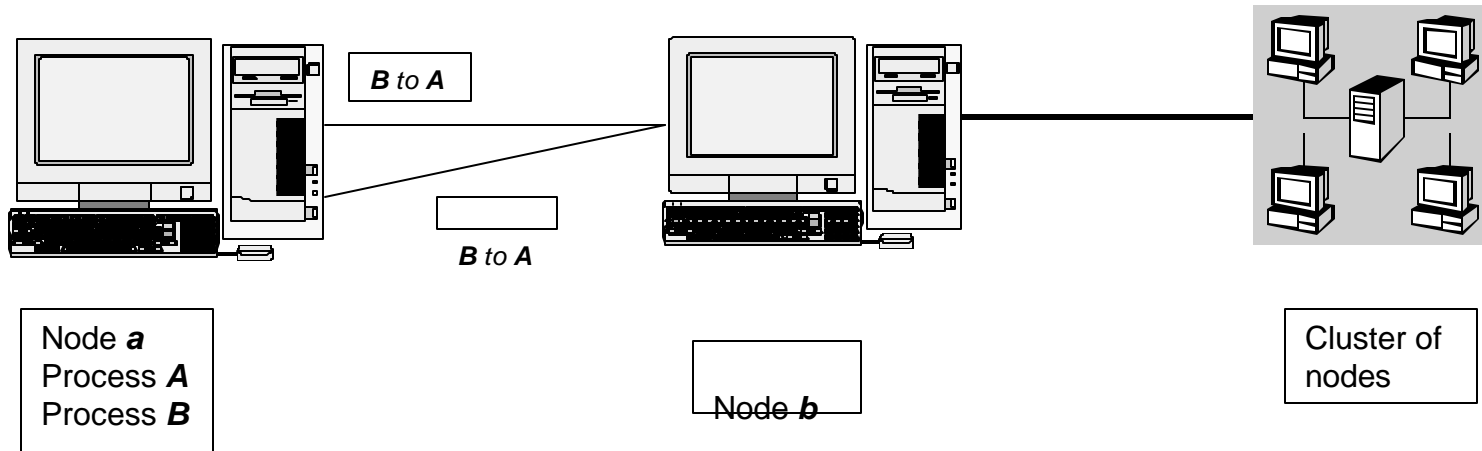
# Network Processes

- Preemptive process migration
- Interaction with environment after migration
- User context, System / UHN context
- Remote migrates, deputy stays at UHN
- Deputy has kernel resources (sockets too!)

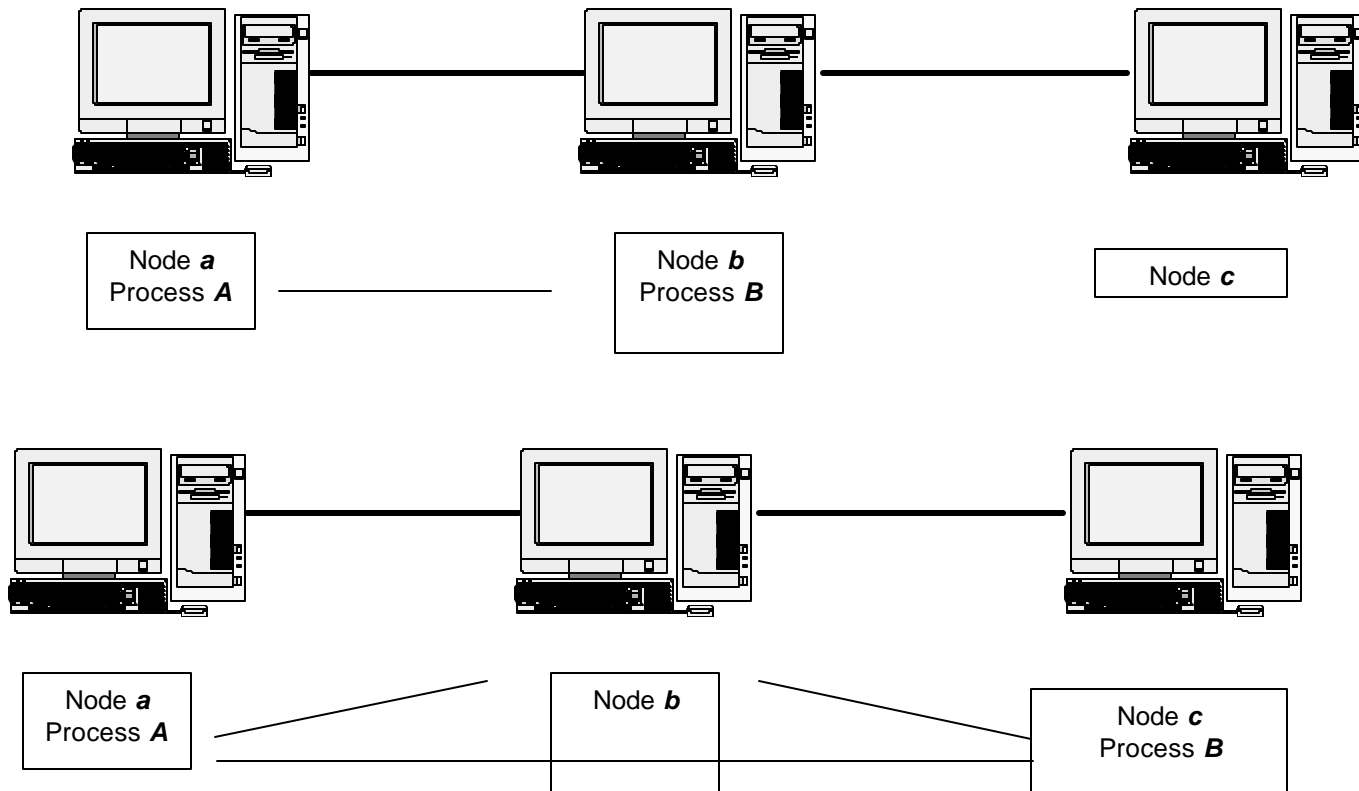
# Network Processes - origin



# Network Process - migrated



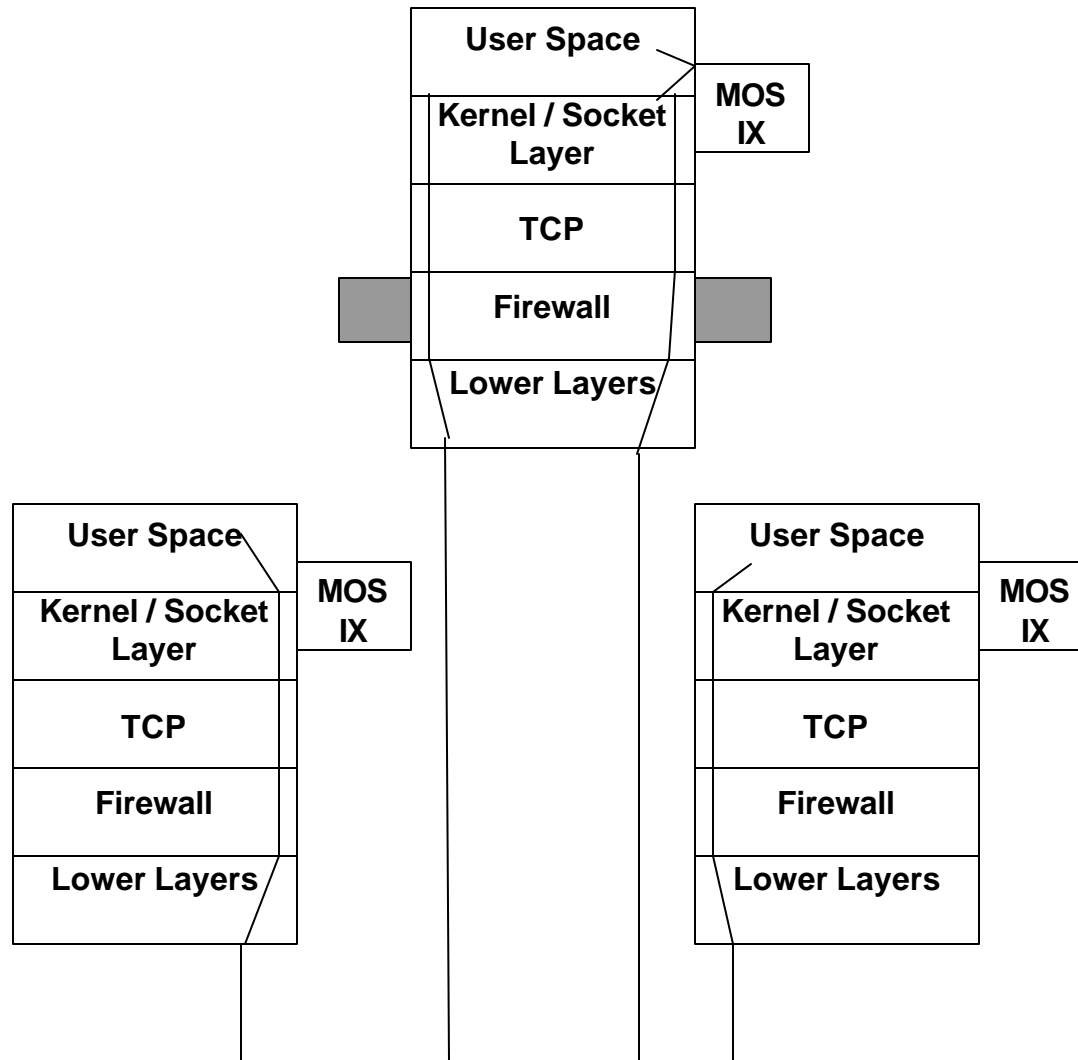
# Network Process - migrated



# Agenda

- MOSIX
- Network communicating processes
- **Breaking it down**
- Timing
- Implementation
- Test
- Results
- Conclusion

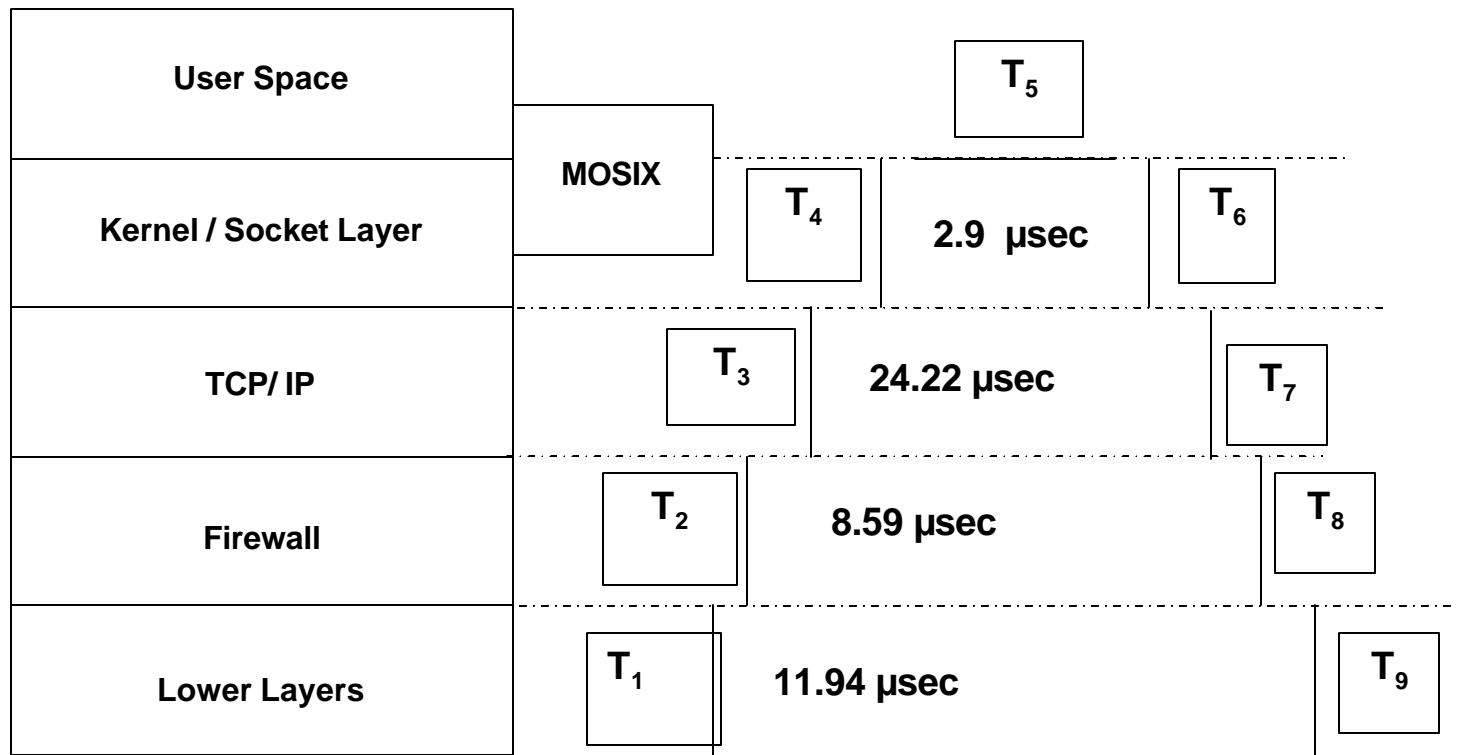
# Breaking down – Microscopic



# Agenda

- MOSIX
- Network communicating processes
- Breaking it down
- **Timing**
- Implementation
- Test
- Results
- Conclusion

# Timing - picture



# Timing - equation

- The amount of time that can be saved if the packets are redirected at the firewall layer =  
*(Time taken at the TCP layer, 24.22  $\mu$ sec)*  
*+ (Time taken at the socket layer, 2.9  $\mu$ sec)*  
*+ (Time taken by MOSIX to decide on the fate of the packet,  $M$   $\mu$ sec).*

# Timing – equation 2

- *Recalculated Time =*  
 $2 \times \{$   
*(Time taken at the TCP layer, 24.22  $\mu$ sec) +*  
*(Time taken at the socket layer, 2.9  $\mu$ sec)*  
 $\} +$   
*(Time taken by MOSIX to decide on the fate*  
*of the packet, M  $\mu$ sec).*

# Timing - Inference

- Yes, time can be saved
- Yes, time can be saved at the firewall layer
- The packet can be identified
- The packet can be redirected
- Because, the firewall can do NAT

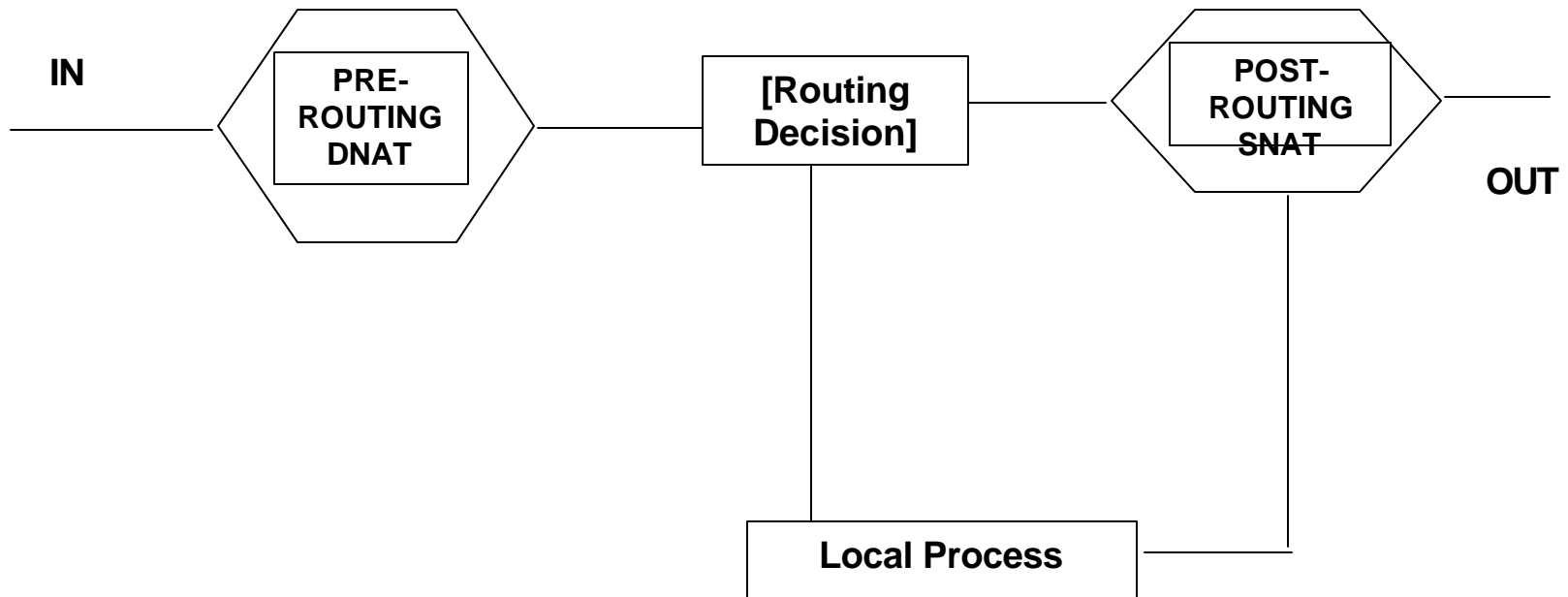
# Agenda

- MOSIX
- Network communicating processes
- Breaking it down
- Timing
- **Implementation**
- Test
- Results
- Conclusion

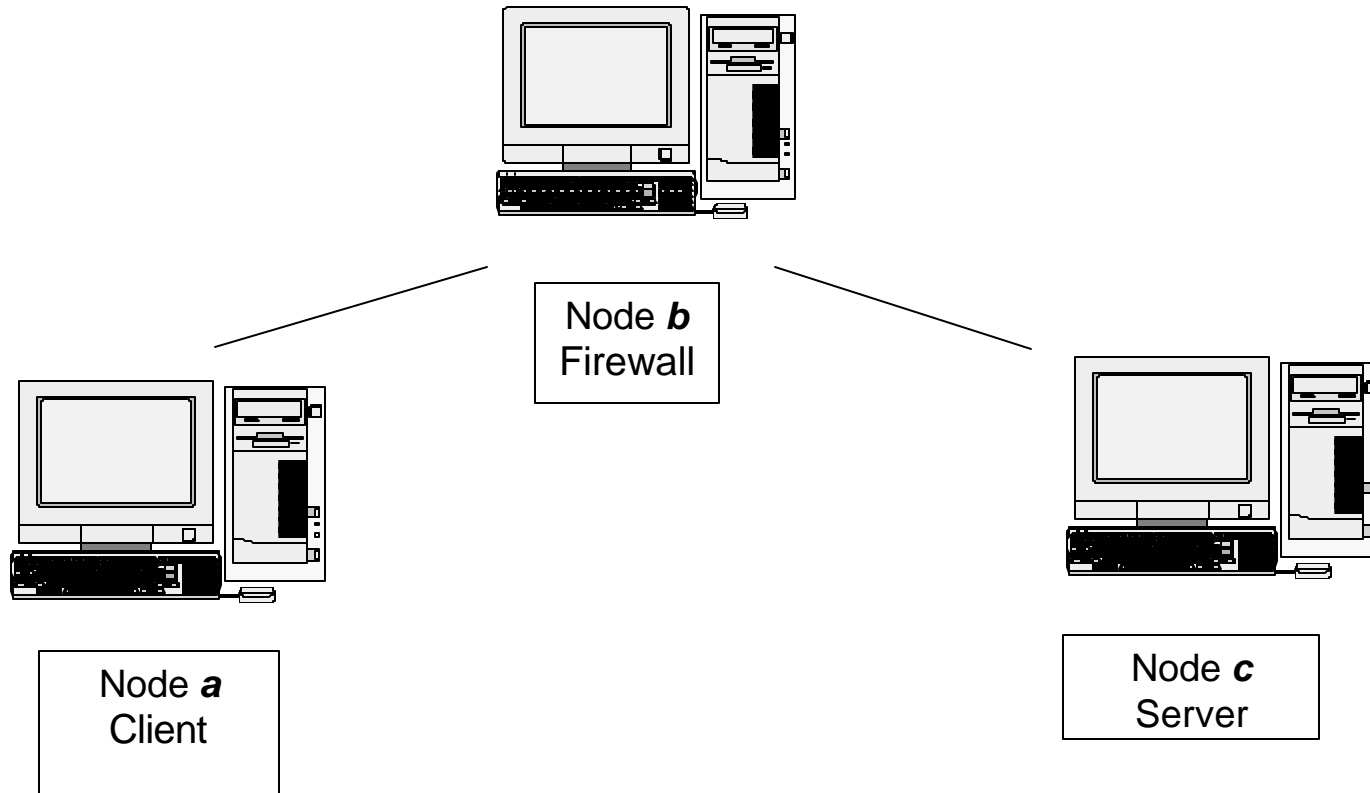
# Implementation - IPTables

- Built-in firewall tool in Linux kernel
- Successor of IPChains and IPFWadm
- MOSIX works on Linux x86
- Timing available for IPTables

# Implementation - NAT



# Implementation - picture



# Implementation - rules

- ***# iptables -t nat -A PREROUTING -s \$CLIENT -d \$FIREWALL\_SYSTEM -p tcp --dport \$SERVER\_PORT -i eth0 -j DNAT --to-destination \$NEW\_DESTINATION***
- ***# iptables -t nat -A POSTROUTING -s \$CLIENT -p tcp --dport \$SERVER\_PORT -o eth0 -j SNAT --to-source \$FIREWALL\_SYSTEM***
- ***# iptables -t nat -A PREROUTING -s \$SERVER -d \$FIREWALL\_SYSTEM -p tcp --sport \$SERVER\_PORT -i eth0 -j DNAT --to-destination \$CLIENT***

# Agenda

- MOSIX
- Network communicating processes
- Breaking it down
- Timing
- Implementation
- **Test**
- Results
- Conclusion

# Testing – what ?

- MOSIX communication through redirection
- IPTables communication through redirection
- Direct communication

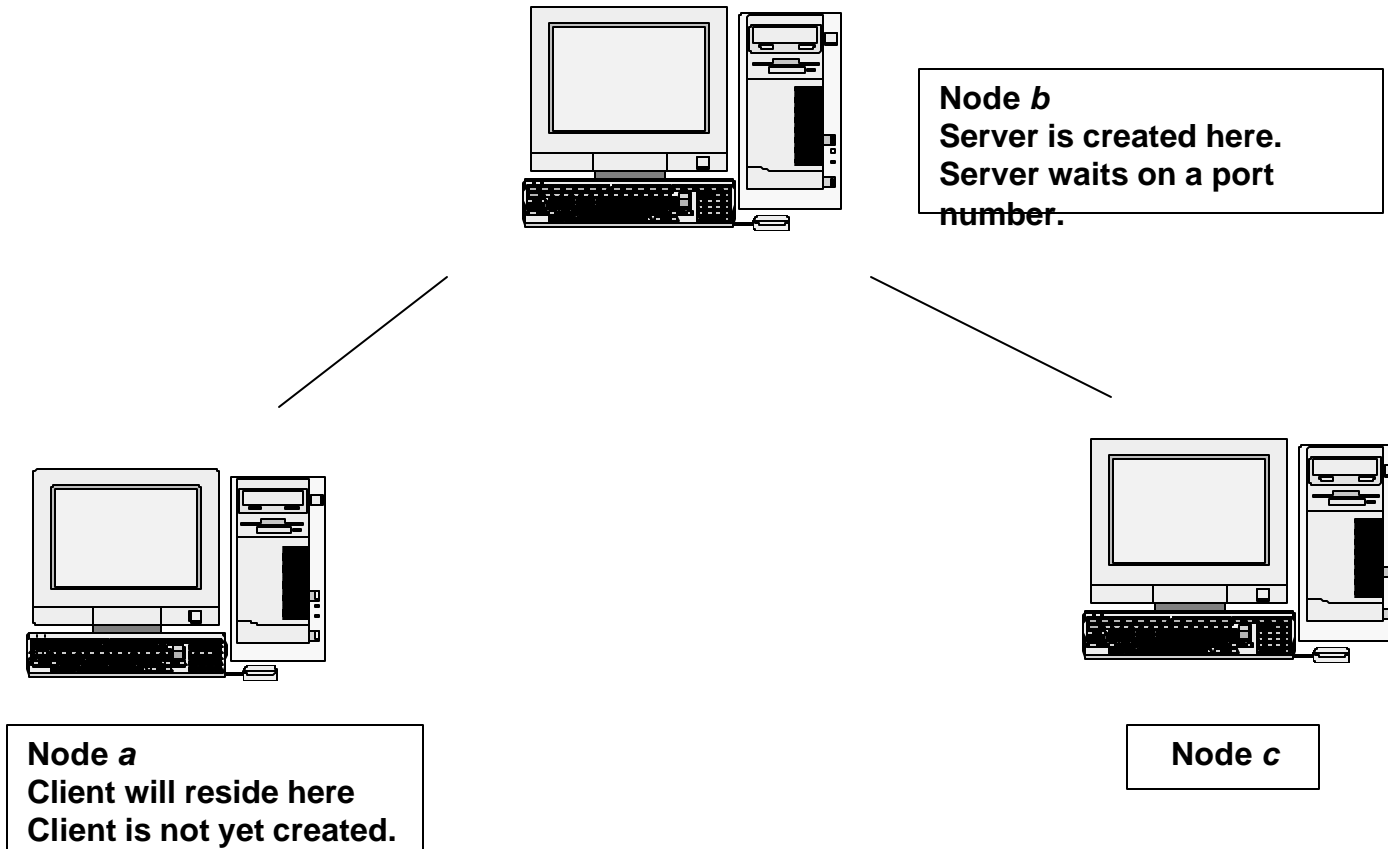
# Testing - Environment

- Pentium P4 CPU
- 1.6 GHz Processor speed
- Intel Ether express Network card
- 100 Mbps LAN
- Two Red Hat 7.2 Linux boxes with Kernel 2.4.19
- One Debian Linux box with Kernel 2.4.18
- All nodes were connected on to the same LAN switch

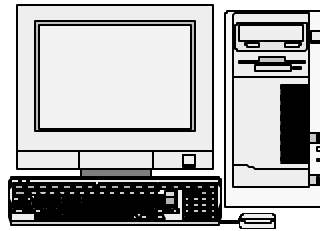
# Testing – how?

- Server-Client communicating pair
- Parameterized
- Buffer size
- Amount of data → time
- Number of such communicating pairs
- Port numbers
- Prints out time taken for data transfer

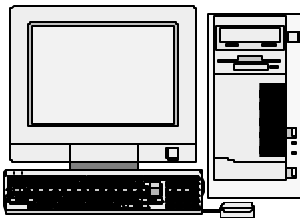
# Testing – MOSIX 1



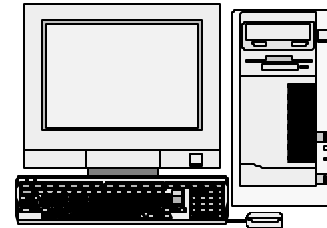
# Testing – MOSIX 2



**Node *b***  
Server is migrated manually  
using MOSIX admin tools to  
node *c*  
All processes think server is  
still in Node *b*

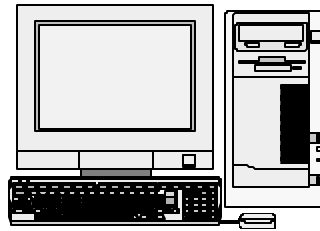


**Node *a***  
Client is created.  
Contacts server in Node  
*b*.  
Is unaware that server  
is in Node *c*

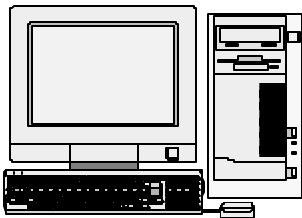


**Node *c***  
Server is now here.  
But, it goes to Node *b* for  
system calls. Node *b* is  
its UHN

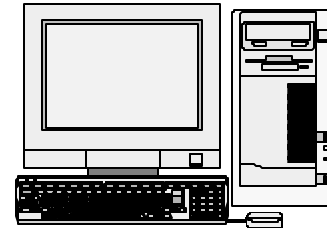
# Testing - IPTables



**Node *b***  
IPTables rules are written here.  
Forwards packets from Node *a* to Node *c* and vice-versa

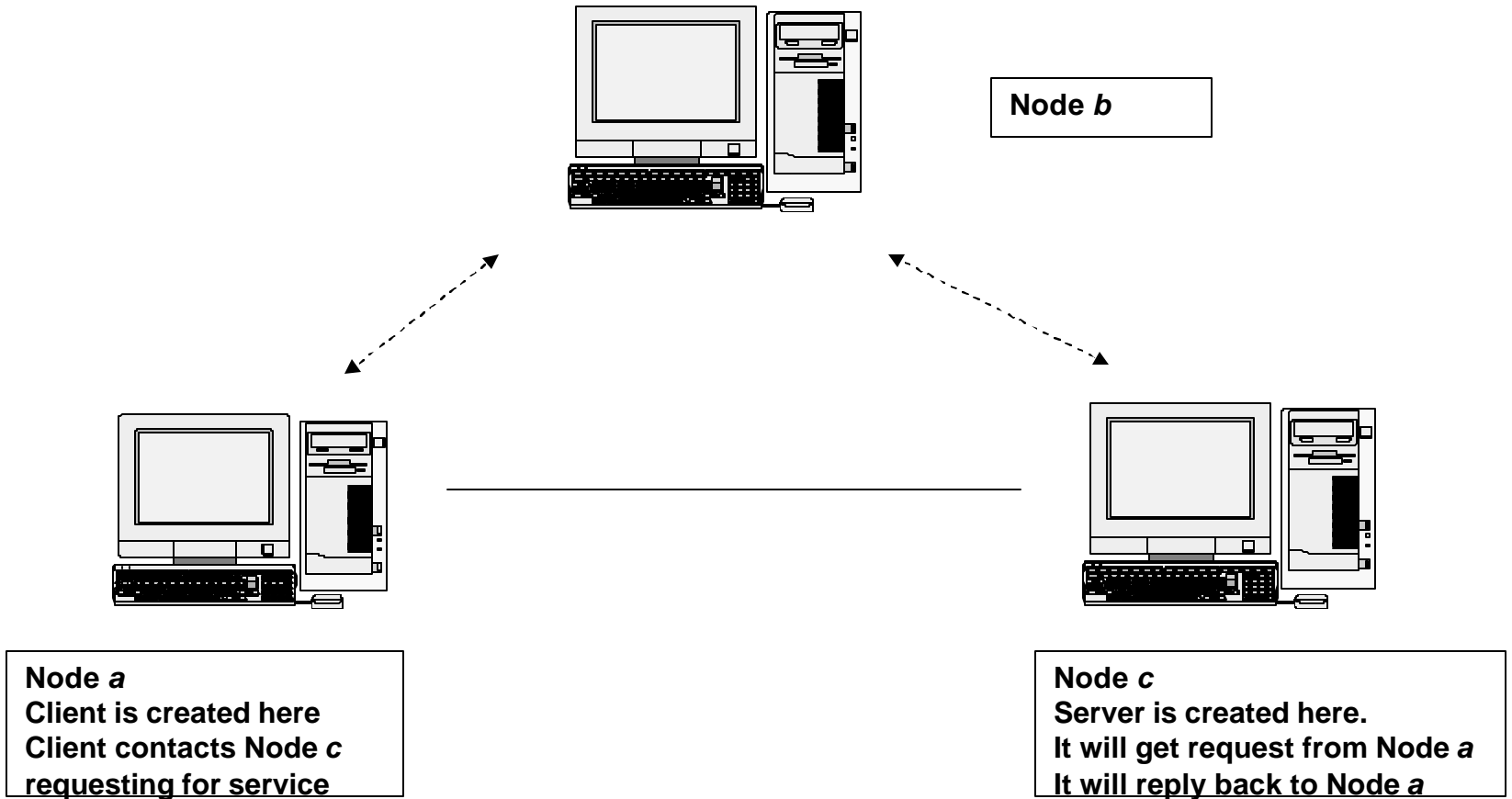


**Node *a***  
Client is created here  
Client contacts Node *b*  
requesting for service



**Node *c***  
Server is created here.  
It will get request from Node *b*  
(which is in reality from Node *a*).  
It will reply back to Node *b*

# Testing - Direct



# Agenda

- MOSIX
- Network communicating processes
- Breaking it down
- Timing
- Implementation
- Test
- **Results**
- Conclusion

# Results – Execution time

No end-to-end connections		LATNENCY (sec)		
		MOSIX	IPTABLES	NORMAL
3		203.73	109.79	103.51
6		275.56	219.58	212.61
9		390.28	328.89	316.60
12		513.64	437.77	424.55
15		640.92	552.14	529.11
25		1063.21	913.89	882.72
50		2130.70	1840.89	1746.64

# Results - Bandwidth

No end-to-end connections		BANDWIDTH (Mbps)		
		MOSIX	IPTABLES	NORMAL
3		15.71	29.15	30.92
6		11.61	14.57	15.05
9		8.19	9.73	10.11
12		6.23	7.31	7.54
15		4.99	5.79	6.05
25		3.01	3.50	3.63
50		1.50	1.74	1.83

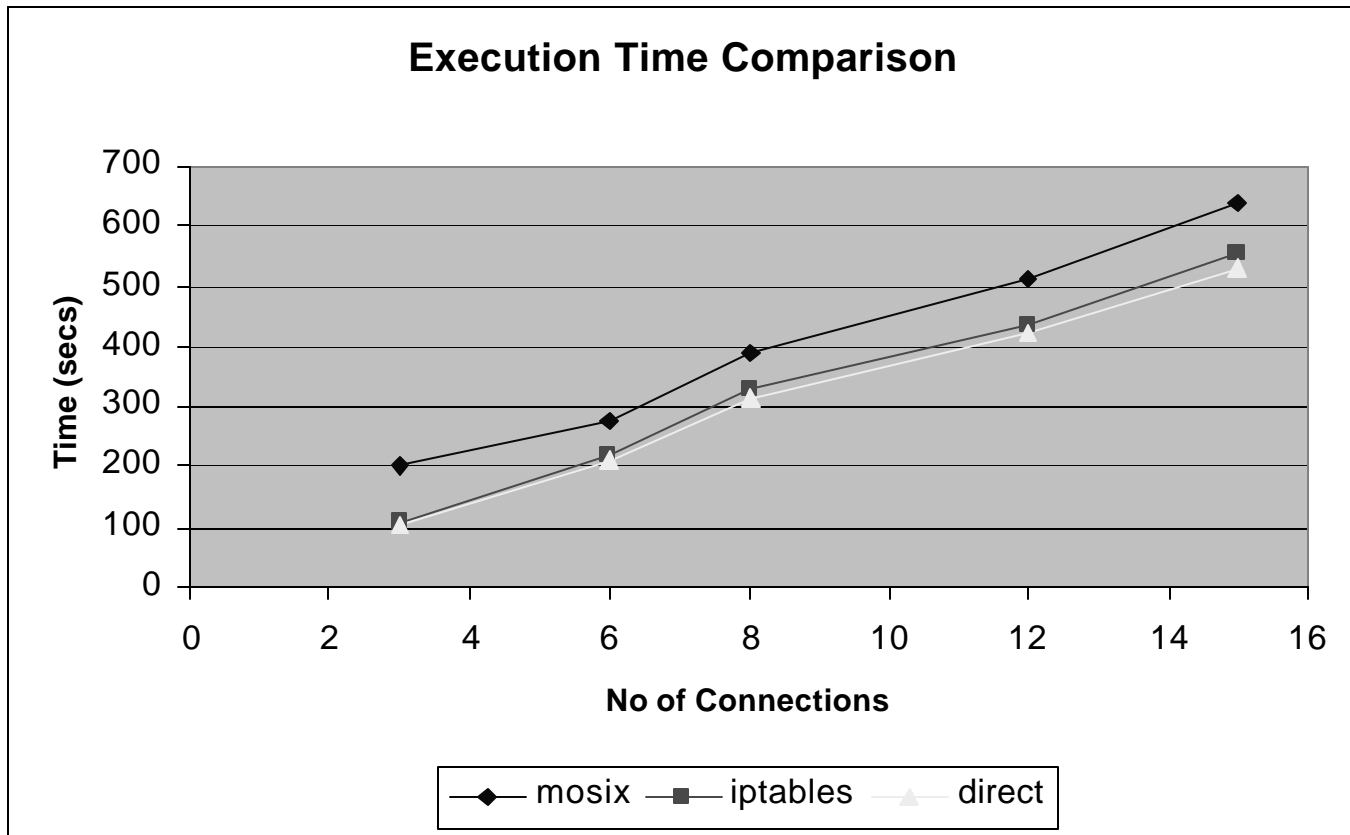
# Results – CPU Utilization

No end-to-end connections		%cpu utilization	
		MOSIX	IPTABLES
3		58.8	27.1
6		85.7	27.1
9		85	25.5
12		87.3	27.5
15		87.5	27.9
25		90	24.5
50		90	28

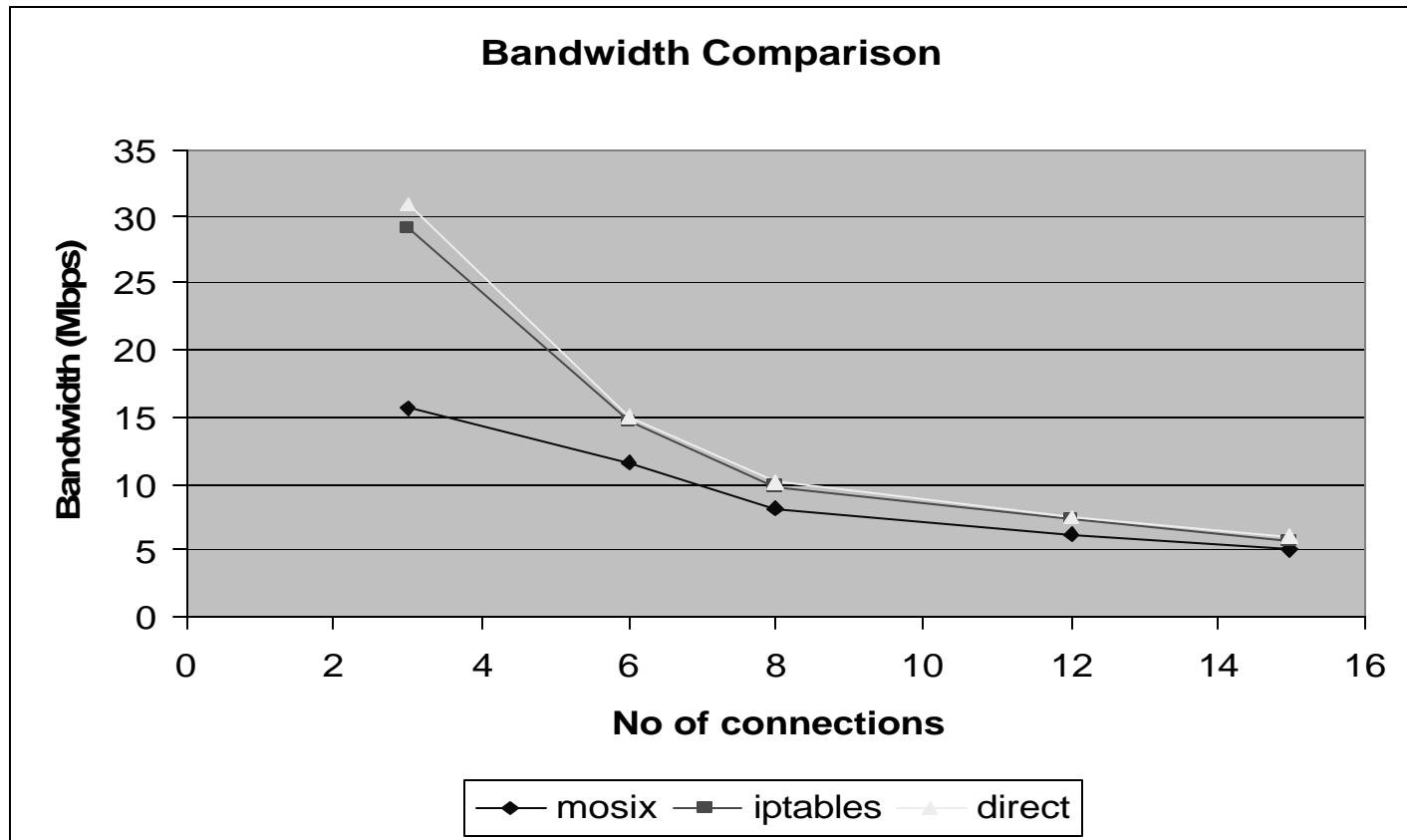
# Results – Load Average

No end-to-end connections		load average	
		MOSIX	IPTABLES
3		0.52	0.02
6		1.34	0.01
9		1.73	0.01
12		1.83	0.02
15		1.8	0.01
25		3.55	0.02
50		4.42	0.02

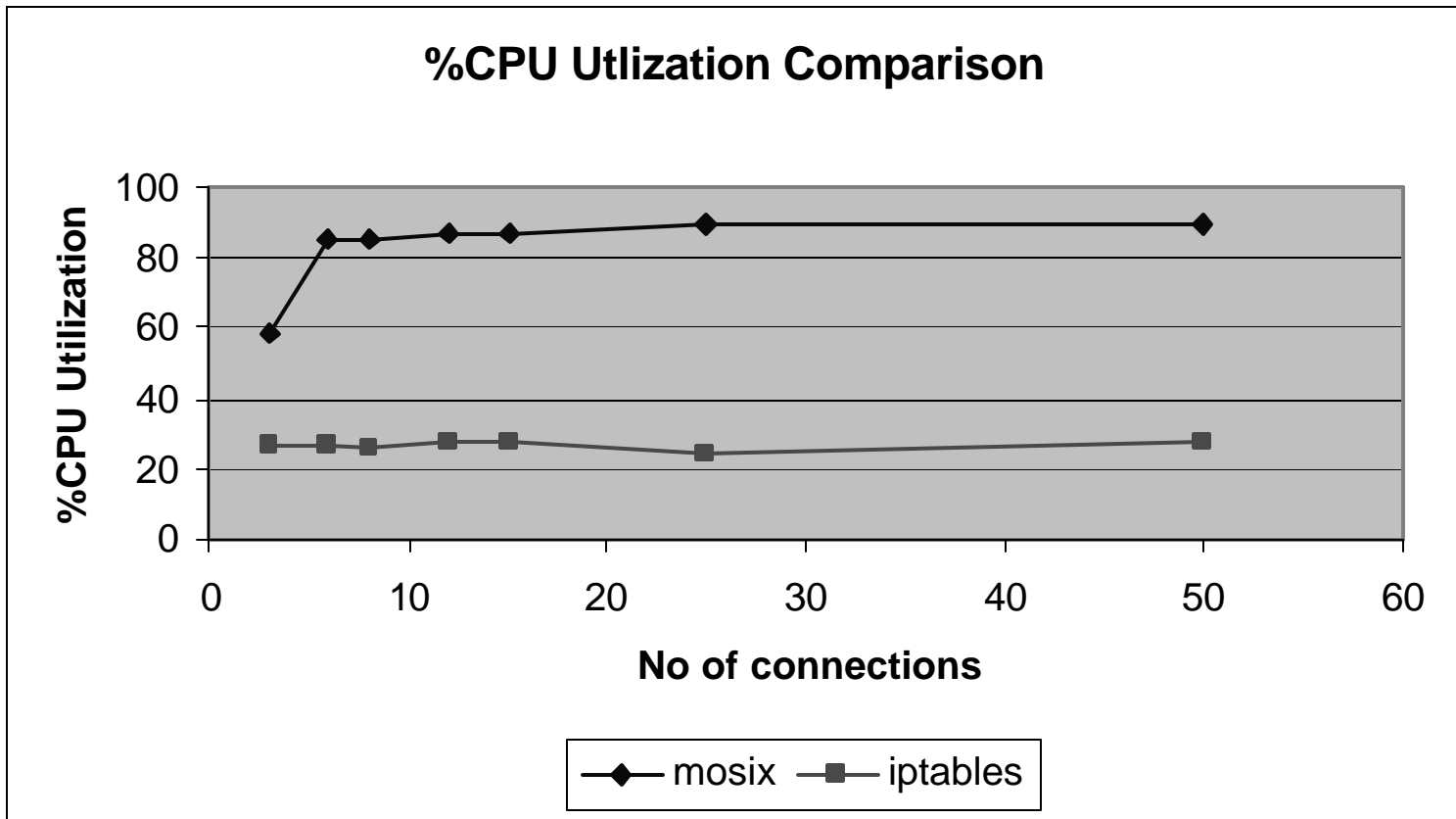
# Results – Execution Graph



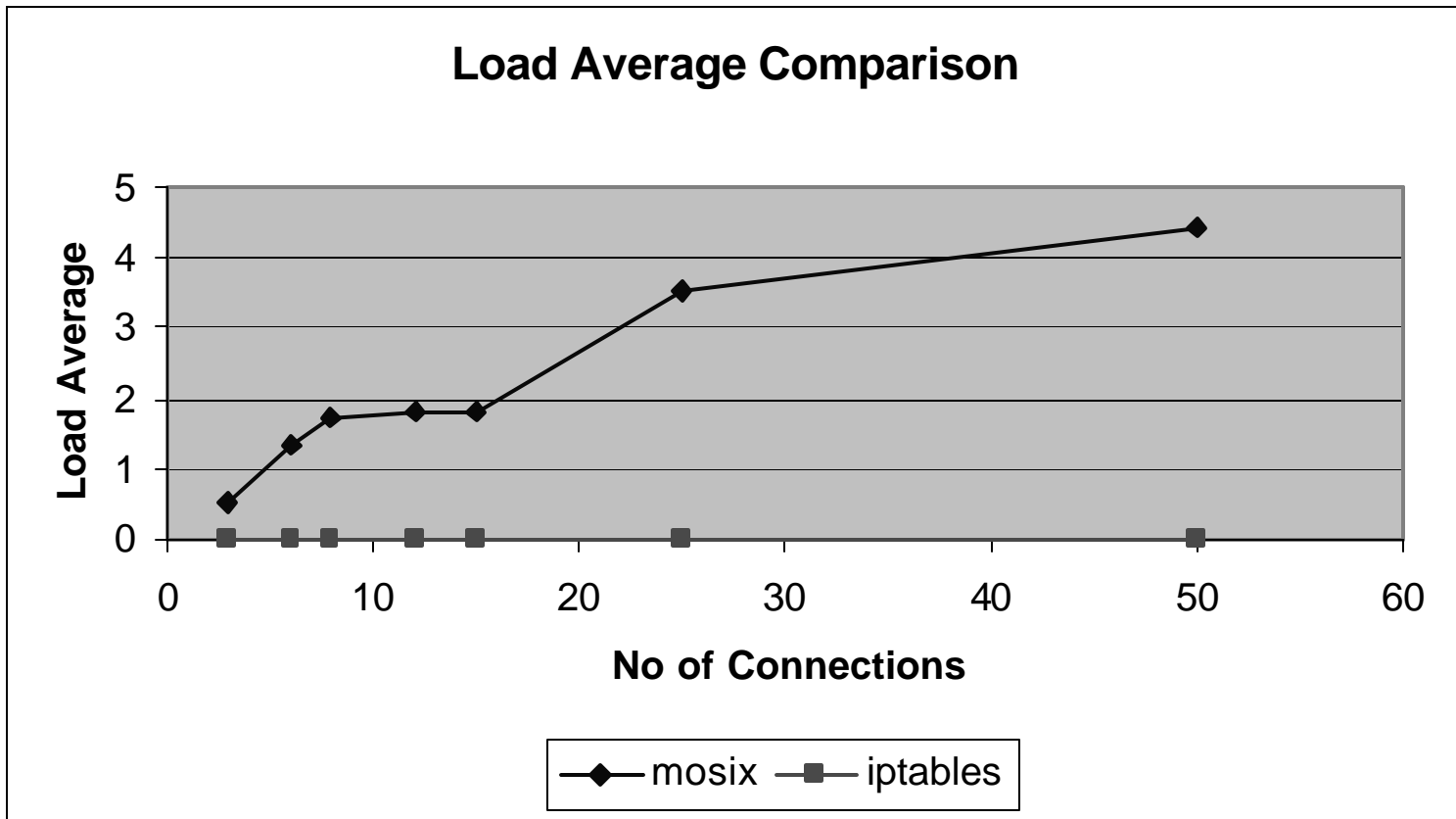
# Results – Bandwidth Graph



# Results – CPU Graph



# Results – Load Graph



# Agenda

- MOSIX
- Network communicating processes
- Breaking it down
- Timing
- Implementation
- Test
- Results
- **Conclusion**

# Conclusion - 1

- MOSIX takes 33% more time on execution than direct communication
- IPTables takes only 4% more
- MOSIX takes 28% more time on execution than IPTables

## Conclusion - 2

- Bandwidth of MOSIX is 20% less than IPTables
- MOSIX, on an average, takes 212% more CPU utilization
- MOSIX takes at least 138 times more load average than IPTables

## Conclusion – 3

- Yes, better performance can be achieved
- Better Execution time
- Better Bandwidth utilization
- Better load average
- Better CPU utilization

[

Questions

]