JAVA NETWORK SIMULATOR

Rhushabh Goradia and Piyush Porwal

Computer Science and Engineering
IIT Bombay
rhushabh@cse.iitb.ac.in, porwalpiyush@cse.iitb.ac.in

November 29, 2004

Outline of Presentation

- Overview of JNS
  - Working
  - Structure
- Proposed Extension
  - 802.3 Ethernet LAN
    - Overview
    - Design
    - Modifications/Extensions required
  - 802.4 Token Bus
    - Overview
    - Design
    - Modifications/Extensions required
- Conclusion

What is JNS ?

- JNS is a Java implementation of ns-2
- Allows developers of networking protocols to simulate their protocols in a controlled environment
- Produces a trace file which can be viewed in a network animator program
- Current version available at SourceForge.org

JNS History

- Original version was from University College London (UCL) until version 1.6 by Lee Eriera, Mark Jatana, Christian Nentwich and Aleksandar Nikolic (1999)
- Current Version (1.7 as of July 2002) maintained by Einar Vollset, PhD student from Newcastle University

What is JNS ?

- JNS is a Java implementation of ns-2
- Allows developers of networking protocols to simulate their protocols in a controlled environment
- Produces a trace file which can be viewed in a network animator program
- Current version available at SourceForge.org

JNS History

- Original version was from University College London (UCL) until version 1.6 by Lee Eriera, Mark Jatana, Christian Nentwich and Aleksandar Nikolic (1999)
- Current Version (1.7 as of July 2002) maintained by Einar Vollset, PhD student from Newcastle University

Comparison Between JNS and NS-2

Parameters	JNS	NS-2
Language used	JAVA	C++, object TCL
Source Size	Less than 1 MB.	40 MB.
Recommended Animator	JAVA Visualizer (JAVIS)	Network Animator (NAM)
Operating System	All platforms where JAVA is available	UNIX and Linux Not recommended for Windows

Nodes

Components	Functionality
Zero or more Interfaces	Attaching interfaces
One IP Handler object	Adding a routing table entry
A name	Adding a default route

Interfaces

Components	Functionality
IP Address	Attaching a link
A bandwidth	Attaching a queue
A reference to an IP Handler	
A queue	
A maximum Transfer unit	

Types Of Interfaces

- Simplex
- Duplex

Nodes

Components	Functionality
Zero or more Interfaces	Attaching interfaces
One IP Handler object	Adding a routing table entry
A name	Adding a default route

Interfaces

Components	Functionality
IP Address	Attaching a link
A bandwidth	Attaching a queue
A reference to an IP Handler	
A queue	
A maximum Transfer unit	

Types Of Interfaces

- Simplex
- Duplex

Nodes

Components	Functionality
Zero or more Interfaces	Attaching interfaces
One IP Handler object	Adding a routing table entry
A name	Adding a default route

Interfaces

Components	Functionality
IP Address	Attaching a link
A bandwidth	Attaching a queue
A reference to an IP Handler	
A queue	
A maximum Transfer unit	

Types Of Interfaces

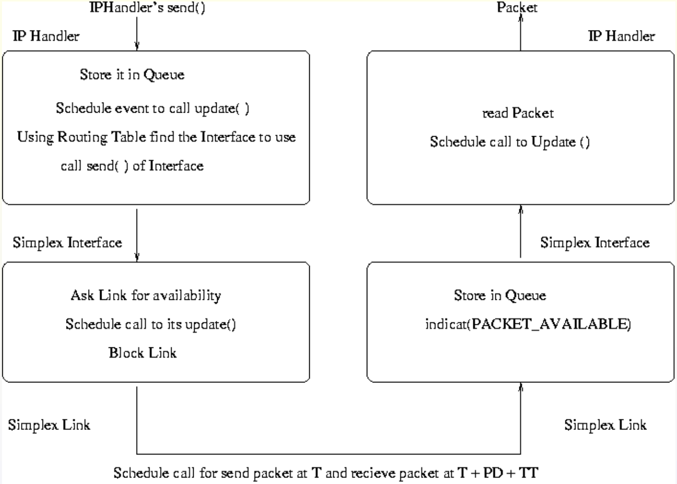
- Simplex
- Duplex

LINKS

- Characteristics
  - A Bandwidth
  - A propagation delay
  - An error rate
- Types
  - Simplex Link
  - Duplex Link

JNS Structure

Elements	Node Interface Link Queue IP Packet IPhandler
Agent	Scheduler CL Agent CO Agent
Command	To Be executed
Trace	Generate trace file
Util	IP Address Route, Route Table Preferences
Dynamic(NEW)	Dynamic Scheduler RMI Technologies



IPHandler's send()

IP Handler

Store it in Queue

Schedule event to call update( )

Using Routing Table find the Interface to use

call send( ) of Interface

Simplex Interface

Ask Link for availability

Schedule call to its update()

Block Link

Simplex Link

Schedule call for send packet at T and recieve packet at T + PD + TT

Simplex Link

Store in Queue

indicat(PACKET_AVAILABLE)

Simplex Interface

read Packet

Schedule call to Update ()

IP Handler

Packet

JNS Working Flow Diagram

Some important classes and methods

- Event, EventParameter
- JavisHandler - JavisPacketHandler, JavisLinkHandler
- Agent - Attach() method
- canSend(), indicate() method

802.3 Ethernet LAN - CSMA/CD

Introduction: Our Focus is Ethernet

- **History**
  - Developed by Bob Metcalfe at Xerox PARC in mid-1970s
  - Roots in Aloha packet-radio network
  - Standardized by Xerox, DEC, and Intel in 1978
- **LAN Standards for MAC & Physical layer connectivity**
  - IEEE 802.3 (CSMA/CD - Ethernet) standard originally 2Mbps
  - IEEE 802.3u standard for 100Mbps Ethernet
  - IEEE 802.3z standard for 1,000Mbps Ethernet
- **CSMA/CD**
  - **CS** = carrier sense : Send only if medium is idle
  - **MA** = multiple access
  - **CD** = collision detection : Stop sending immediately if collision is detected

802.3 Ethernet LAN - CSMA/CD

Introduction: Our Focus is Ethernet

- **History**
  - Developed by Bob Metcalfe at Xerox PARC in mid-1970s
  - Roots in Aloha packet-radio network
  - Standardized by Xerox, DEC, and Intel in 1978
- **LAN Standards for MAC & Physical layer connectivity**
  - IEEE 802.3 (CSMA/CD - Ethernet) standard originally 2Mbps
  - IEEE 802.3u standard for 100Mbps Ethernet
  - IEEE 802.3z standard for 1,000Mbps Ethernet
- **CSMA/CD**
  - **CS** = carrier sense : Send only if medium is idle
  - **MA** = multiple access
  - **CD** = collision detection : Stop sending immediately if collision is detected

802.3 Ethernet LAN - CSMA/CD

Introduction: Our Focus is Ethernet

- **History**
  - Developed by Bob Metcalfe at Xerox PARC in mid-1970s
  - Roots in Aloha packet-radio network
  - Standardized by Xerox, DEC, and Intel in 1978
- **LAN Standards for MAC & Physical layer connectivity**
  - IEEE 802.3 (CSMA/CD - Ethernet) standard originally 2Mbps
  - IEEE 802.3u standard for 100Mbps Ethernet
  - IEEE 802.3z standard for 1,000Mbps Ethernet
- **CSMA/CD**
  - **CS** = carrier sense : Send only if medium is idle
  - **MA** = multiple access
  - **CD** = collision detection : Stop sending immediately if collision is detected

CSMA/CD

In Aloha, decisions to transmit are made without paying attention to what other nodes might be doing

Working :

- Ethernet uses CSMA/CD listens to line before/during sending
-
  - If line is idle (no carrier sensed)
  - send packet immediately
  - upper bound message size of 1500 bytes
-
  - If line is busy (carrier sensed)
  - wait until idle and transmit packet immediately
  - called 1-persistent sending
-
  - If collision detected
  - Stop sending and jam signal
  - Try again later

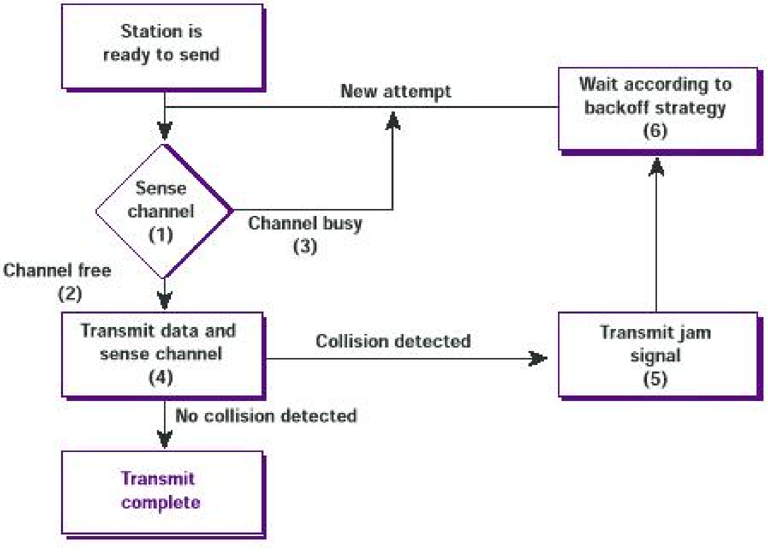
CSMA/CD

In Aloha, decisions to transmit are made without paying attention to what other nodes might be doing

Working :

- Ethernet uses CSMA/CD listens to line before/during sending
-
  - If line is idle (no carrier sensed)
  - send packet immediately
  - upper bound message size of 1500 bytes
-
  - If line is busy (carrier sensed)
  - wait until idle and transmit packet immediately
  - called 1-persistent sending
-
  - If collision detected
  - Stop sending and jam signal
  - Try again later

CSMA/CD


Station is ready to send
New attempt
Wait according to backoff strategy (6)
Sense channel (1)
Channel busy (3)
Channel free (2)
Transmit data and sense channel (4)
Collision detected
Transmit jam signal (5)
No collision detected
Transmit complete

Exponential Backoff

A collision is detected, delay and try again

- Delay time is selected using binary exponential backoff
  - 1^{st} time: choose K from $\{0,1\}$ then delay $= K * 51.2us$
  - 2^{nd} time: choose K from $\{0,1,2,3\}$ then delay $= K * 51.2us$
  - n^{th} time: delay $= Kx51.2us$, for $K = 0...2n1$
  - Note max value for $k = 1023$.
  - Give up after several tries (usually 16).Report transmit error to host.

-
  - If delay were not random, then there is a chance that sources would retransmit in lock step.
  - Why not just choose from small set for K
  - This works fine for a small number of hosts
  - Large number of nodes would result in more collisions

Exponential Backoff

A collision is detected, delay and try again

- Delay time is selected using binary exponential backoff
    - 1^{st} time: choose K from $\{0,1\}$ then delay $= K * 51.2us$
    - 2^{nd} time: choose K from $\{0,1,2,3\}$ then delay $= K * 51.2us$
    - n^{th} time: delay $= Kx51.2us$, for $K = 0...2n1$
    - Note max value for $k = 1023$.
    - Give up after several tries (usually 16).Report transmit error to host.
-
    - If delay were not random, then there is a chance that sources would retransmit in lock step.
    - Why not just choose from small set for K
    - This works fine for a small number of hosts
    - Large number of nodes would result in more collisions

10 BASE 5	10 BASE 2
Thick Co-Ax	Thin Co-Ax

Characteristics :

CABLE	Sg.Ln	N/S	Compnts.	BW
10 BASE 5	500 m.	200	Thick Co-Ax	10 Mbps.
			NIC	
			Trans-receivers	
			AUI cables	
10 BASE 2	200 m.	30	Thin Co-Ax	10 Mbps.
			NIC	
			BNC-T	

Extension/Modifications Required

- A new class for defining Frame format of 802.3
- Extend Link class.
- Extend Interface class
- Collaborate to implement the Protocol.

Extension/Modifications Required

- A new class for defining Frame format of 802.3
- Extend Link class.
- Extend Interface class
- Collaborate to implement the Protocol.

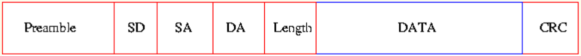
Extension/Modifications Required

- A new class for defining Frame format of 802.3
- Extend Link class.
- Extend Interface class
- Collaborate to implement the Protocol.

Extension/Modifications Required

- A new class for defining Frame format of 802.3
- Extend Link class.
- Extend Interface class
- Collaborate to implement the Protocol.

New class FRAME_3



Preamble	SD	SA	DA	Length	DATA	CRC

Ethernet 802.3 Frame Format

- Variables:
    - Preamble
    - Start Delimiter
    - Destination address
    - Source Address
    - Length
    - Data_Unit
    - CRC
- Methods:
    - To compute CRC.
    - To generate frame.
    - Set the flags like HEADER_SIZE, FRAGMENT_OR_NOT.

Extend Duplexlink Class

- To define the properties of the various types of Physical links.
- It can be
  - 10Base5 Thick cable - Duplex_10B5_Link class
  - 10Base2 Thin cable - Duplex_10B2_Link class
- Define particular values of the cable like MaxLength, Bandwidth, etc here.

Extend DuplexInterface Class

- Extend to DuplexInterface_3
- It will deals with the actual data transmission.
- All the steps required to simulate Ethernet will be defined.
- To do that, keep the interface working intact, but override SEND() method.

Extend Duplexlink Class

- To define the properties of the various types of Physical links.
- It can be
  - 10Base5 Thick cable - Duplex_10B5_Link class
  - 10Base2 Thin cable - Duplex_10B2_Link class
- Define particular values of the cable like MaxLength, Bandwidth, etc here.

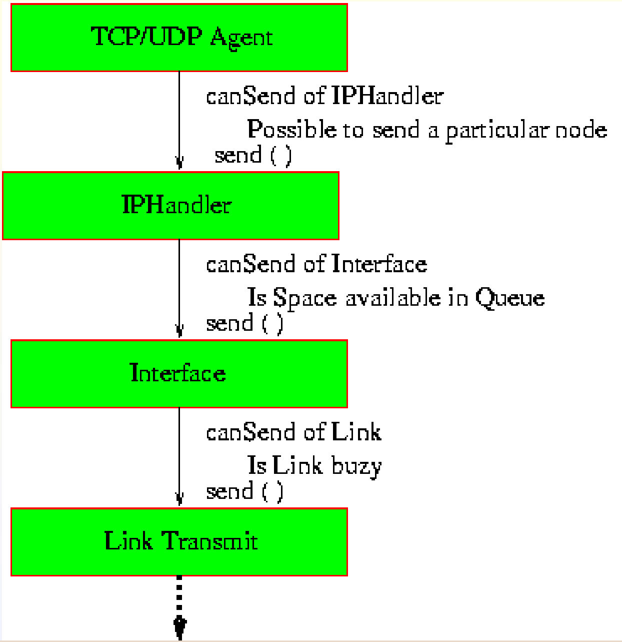
Extend DuplexInterface Class

- Extend to DuplexInterface_3
- It will deals with the actual data transmission.
- All the steps required to simulate Ethernet will be defined.
- To do that, keep the interface working intact, but override SEND() method.

Functioning

Whole implementation lies within SEND() of DuplexInterface_3
send(Frame frameObject)

- Check frame can be send or not.
- Schedule events for sending and receiving(after propagation delay expected).
- Listen to schedule, to find collision.
- if(collision)
  - Wait using exponential back-off.
  - repeat 1-4.
- Returns Happily.



TCP/UDP Agent

canSend of IPHandler

Possible to send a particular node

send ( )

IPHandler

canSend of Interface

Is Space available in Queue

send ( )

Interface

canSend of Link

Is Link buzy

send ( )

Link Transmit

802.4 Token Bus

Standard 802.4 describes a LAN called “TOKEN BUS”. Physically a token bus is a linear cable onto which stations are attached. But logically stations are organized into a **ring**.

When the ring is initialized the first station sends the data in 802.4 frame format. After it finishes it passes permissions to its logical successor by sending a special control frame called **Token**.

The token propagates around the logical ring. If station wants to send data, it must wait and capture the token. Only the token holders are permitted to transmit frame, and since only one station holds the token, collisions do not occur!!

802.4 Token Bus

Standard 802.4 describes a LAN called “TOKEN BUS”. Physically a token bus is a linear cable onto which stations are attached. But logically stations are organized into a **ring**.

When the ring is initialized the first station sends the data in 802.4 frame format. After it finishes it passes permissions to its logical successor by sending a special control frame called **Token**.

The token propagates around the logical ring. If station wants to send data, it must wait and capture the token. Only the token holders are permitted to transmit frame, and since only one station holds the token, collisions do not occur!!

802.4 Token Bus

Standard 802.4 describes a LAN called “TOKEN BUS”. Physically a token bus is a linear cable onto which stations are attached. But logically stations are organized into a **ring**.

When the ring is initialized the first station sends the data in 802.4 frame format. After it finishes it passes permissions to its logical successor by sending a special control frame called **Token**.

The token propagates around the logical ring. If station wants to send data, it must wait and capture the token. Only the token holders are permitted to transmit frame, and since only one station holds the token, collisions do not occur!!

Types Of Frames

There are some special frames besides the Token and Data frames.

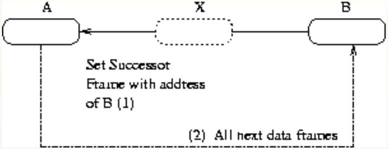
- Solicit Successor Frame: Used to add new stations.Each station periodically transmits this inviting new stations to join with addresses between itself and the next station in sequence. If there is a response it passes token to the new station thereafter.
- Who Follows Who Frame: Useful for detecting failure of any station. After passing the token to the next station sender waits for a valid frame from the next station i.e. either **token release or data release**. If still no valid frame, sender sends a **Who Follows Who** frame with address of the failed station. The station next to failed station responds with a **Set Successor Frame** with its address. Sender then passes token to this station.

Types Of Frames

There are some special frames besides the Token and Data frames.

- Solicit Successor Frame: Used to add new stations.Each station periodically transmits this inviting new stations to join with addresses between itself and the next station in sequence. If there is a response it passes token to the new station thereafter.
- Who Follows Who Frame: Useful for detecting failure of any station. After passing the token to the next station sender waits for a valid frame from the next station i.e. either **token release or data release**. If still no valid frame, sender sends a **Who Follows Who** frame with address of the failed station. The station next to failed station responds with a **Set Successor Frame** with its address. Sender then passes token to this station.

- Set Successor Frame: Required if some station wants to leave the ring



Data Frame

The actual IEEE 802.4 Data Frame :

Preamble: For Synchronization

SD: Start of Frame

FC: Used to differentiate between frames

DA: Destination Address

SA: Source Address

Data: IP Header + Packet

FCS: CRC code

ED: End of Frame

Preamble	SD	FC	DA	SA	Data	FCS	ED

Design

- Initialization Network
  - Create Nodes each having two interfaces
  - User will specify logical connections
  - Nodes will be connected physically as specified.
  - IP table will consists of only one entry, that of logical successor.
  - First node created will start transmission by generating token

- Functioning
  - Each node can keep the Token for at most T time units.
  - In that amount of time, a node can send packet
  - Release token to its successor
  - Any node want to send data, checks TOKEN_HOLDING variable.
  - If not holding will sets a variable WANT_TO_SEND
  - On receiving token, if WANT_TO_SEND variable is set, transmission of data will be started.
  - The node which releases, checks for two events - Token passed or Data transmission started
  - If none happens, it sends WHW frame.
  - The node following will reply with its IP and that IP will be stored as of logical successor.

- Network Structure modifications
  - Adding a Node
    - A node, periodically generates a SOLICIT_SUCCESSOR frame to the next node.
    - Say, a station X is added between nodes A and B, then this S.S. frame will be accepted by X.
    - X then sends its IP to A. Node A updates its IP routing table.
    - A sends its previous routing table entry to X(i.e. IP of B).X updates its routing table.
    - Now the connection flow is A-X-B, which previously was A-B.
  - Removing a node
    - Node to be removed will send SET_SUCCESSOR frame to previous node.
    - Previous node will change its IP table.

Changes and Extensions

- A new class for defining Frame format of 802.4
- Modify Simulator to initialize network
- Extend Interface class to implement functionality
- Collaborate to simulate the Protocol as a whole.

Extend DuplexInterface Class

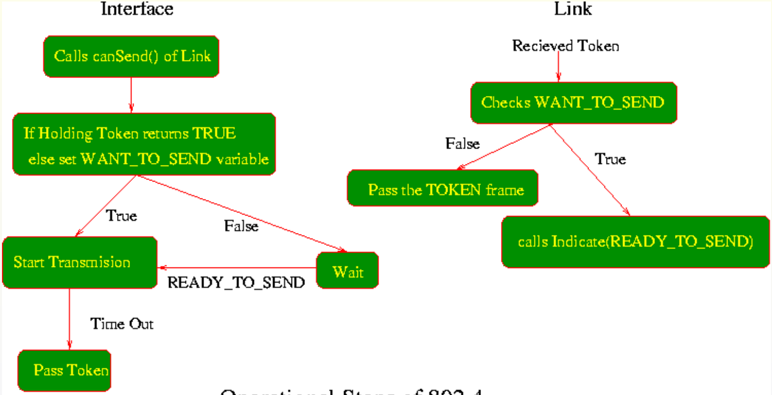
- Extend to DuplexInterface_4
- It will deals with the actual data transmission.
- All the steps required to simulate Token-Bus will be defined.
- To do that, keep the interface working intact, but override SEND() method.

Changes and Extensions

- A new class for defining Frame format of 802.4
- Modify Simulator to initialize network
- Extend Interface class to implement functionality
- Collaborate to simulate the Protocol as a whole.

Extend DuplexInterface Class

- Extend to DuplexInterface_4
- It will deals with the actual data transmission.
- All the steps required to simulate Token-Bus will be defined.
- To do that, keep the interface working intact, but override SEND() method.



Interface

Calls canSend() of Link

If Holding Token returns TRUE
else set WANT_TO_SEND variable

True

False

Start Transmision

READY_TO_SEND

Wait

Time Out

Pass Token

Link

Recieved Token

Checks WANT_TO_SEND

False

True

Pass the TOKEN frame

calls Indicate(READY_TO_SEND)

Operational Steps of 802.4

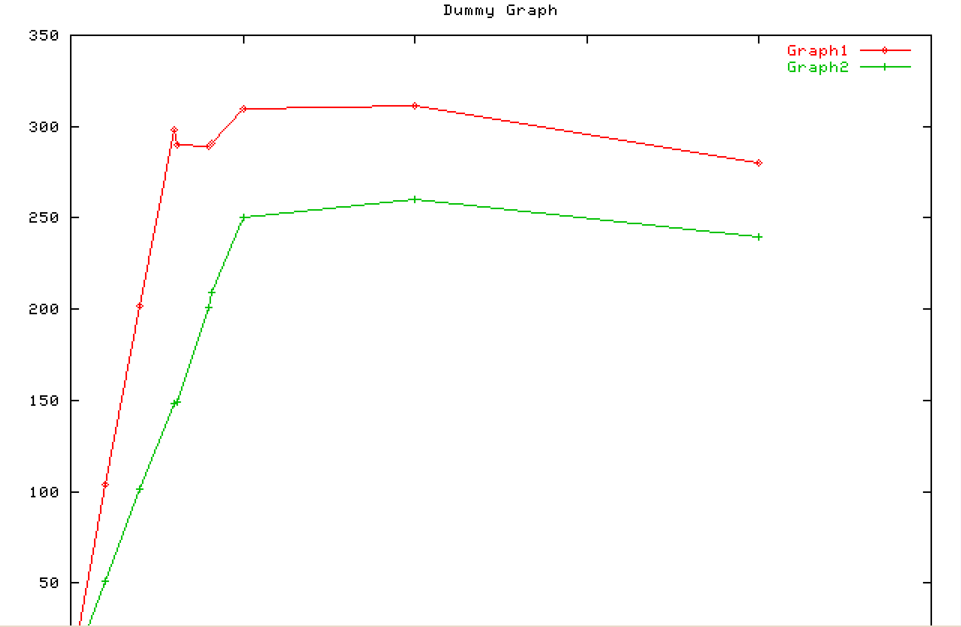
- Conclusion
  - JNS is a powerful simulator tool, which is easy to use and visualize
  - JNS is very short and simple, and therefore easy to extend.
  - It is platform independent and thus is Superior then NS-2
  - Two new protocols are suggested for implementations - 802.3 and 802.4
  - Our implementation will extend the functionality at DLL.
- Future Work
  - JNS is still lacking in various DLL protocols like 802.5(Token Ring)
  - Also, suggested work can be further extended like Fast Ethernet.
- References
  - JNS documentation/code
  - NS-2 manual
  - Computer Networks by A.S.Tanenbaum

beamericonarticle Java network simulator.
http://jns.sourceforge.net/.

beamericonarticle Network simulator 2.
http://www.isi.edu/nsnam/ns/.

beamericonarticle Andrew S. Tanenbaum.
Computer Networks.
Prentice Hall PTR, 2002.

THANK YOU !!


Dummy Graph
350
300
250
200
150
100
50
Graph1
Graph2