NAME

nam − VINT/LBL Network Animator

SYNOPSIS

nam [**−g** *geometry*] [**−t** *graphInput*][**−i** *interval*] [**−P** *peerName*] [**−N** *appName*] [**−c** *cacheSize*] [**−f** *configfile*] [**−S**] *tracefile*

DESCRIPTION

Nam is a Tcl/TK based animation tool for viewing network simulation traces and real world packet trace data.

The first step to use nam is to produce the trace file. The trace file should contain topology information, e.g., nodes, links, as well as packet traces. The detailed format is described in the TRACE FILE section. Usually, the trace file is generated by *ns(1)*. During an ns simulation, user can produce topology configurations, layout information, and packet traces using tracing events in ns. Refer to *ns(1)* for detailed information.

When the trace file is generated, it is ready to be animated by nam. Upon startup, nam will read the trace file, create topology, pop up a window, do layout if necessary, then pause at the time of the first packet in the trace file. Through its user interface, nam provides control over many aspects of animation. These functionalities will be described in detail in the USER INTERFACE section.

This version of nam is highly experimental - there will be bugs!. Please mail ns-developers@mash.cs.berkeley.edu if you encounter any bugs, or with suggestions for desired functionality.

OPTIONS

−g Specify geometry of the window upon startup. The format is described in *X(1)*

−t [Information incomplete] Instruct nam to use tkgraph, and specify input file nam for tkgraph.

−i [Information for this option may not be accurate] Specify *rate* (real) milliseconds as the screen update rate. The default rate is 50ms (i.e., 20 frames per second). Note that the X server may not be able to keep up with this rate, in which case the animation will run as fast as the X server allows it to (at 100% cpu utilization).

−N Specify the application name of this nam instance. This application name may later be used in peer synchronization.

−P Specify the application name of the peer nam instance whose execution will be synchronized with the execution of this nam instance. Refer to the above option (-N) as how to specify application names.
General usage is: (1) starting the first nam instance (slave) by:
nam -N <name #1> <trace file name #1>
Then start the second nam instance (which will be the master):
nam -N <name #2> <trace file name #2>
Then every animation control (play, stop, backward, but *exclude* other inspection and interactive operations such as monitoring) will be synchronized between the two instances.
Please note that because this mechanism uses Tcl's send command, it requires that your X server used xauth as authentication. Specifically, you should add option '-auth <authorization file name>' when you starts your X server. Without this option, X will use xhost as authentication, which is too weak and considered insecure. Refer to man page of Xsecurity, xauth and Xserver for details, and the available authentication protocols.

−c [Information incomplete] The maximum size of the cache used to store 'active' objects when doing backward animation.

−f Name of the initialization files to be loaded during startup. In this file, user can define functions which will be called in the trace file. An example for this is the 'link-up' and 'link-down' events of dynamic links in ns. (Refer to **$ns rtmodel** for detail, and tcl/ex/simple-dyn.tcl in your ns directory for example). Example initialization files can be found at ex/sample.nam.tcl and ex/dynamic-nam.conf.

–S Enable synchronous X behavior so it is easier for graphics debugging. For UNIX system running X only.

tracefile is the name of the file containing the trace data to be animated (format described in TRACE FILE section below). If *tracefile* cannot be read, **nam** will try to open *tracefile*.nam.

OBJECTS IN NAM

nam does animation using the following building blocks: node, link, queue, packet, agent, monitor. They are defined below:

node Nodes are created from 'n' trace event in trace file. It represents a source/host/router, etc. nam will terminate if there are duplicate definition for the same node. Node may have many shapes, (circle, square, and hexagon), but once created it cannot change its shape. Node may also have many colors, it can change its color during animation. Refer to *ns(1)* for related tracing events.

link Links are created between nodes to form a network topology. nam links are internally simplex, but it is invisible to the users. The trace event 'l' creates two simplex links and other necessary setups, hence it looks to users identical to a duplex link. Link may have many colors, it can change its color during animation. Refer to *ns(1)* for related tracing events.

queue Queue needs to be constructed in nam between two nodes. Unlike link, nam queue is associated to a *simplex* link. The trace event 'q' only creates a queue for a simplex link. In nam, queues are visualized as stacked packets. Packets are stacked along a line, the angle between the line and the horizontal line can be specified in the trace event 'q'.

packet Packet is visualized as a block with an arrow. The direction of the arrow shows the flow direction of the packet. Queued packets are shown as little squares. A packet may be dropped from a queue or a link. Dropped packets are shown as rotating squares, and disappear at the end of the screen. Dropped packets are not visible during backward animation.

agent Agents are used to separate protocol states from nodes. They are always associated with nodes. An agent has a name, which is a *unique* identifier of th agent. It is shown as a square with its name inside, and a line link the square to its associated node.

AUTOMATIC LAYOUT

In nam, a topology is specified by alternating node objects with edge objects. But to display the topology in a comprehensible way, a layout mechanism is needed. Currently nam provides two layout methods.

First, user may specify edges' orientations. An edge orientation is the angle between the edge and the horizontal line, in the interval [0, 2*pi). During layout, nam will honor the given edge orientations. Generally, it will first choose a reference node, then place other nodes using edge orientation and edge length, which is determined by link delay. This works well for small and manually generated topologies.

Second, when we are dealing with randomly generated topologies, be it small or large, we may want to do layout automatically. An automatic graph layout algorithm ([1] [2]) is adapted and implemented. The basic idea of the algorithm is to model the graph as balls (nodes) connected by springs (edges). Balls will repulse each other, while springs pull them together. This system will (hopefully) converge after some iterations. In practice, after a small number of iterations (tens or hundreds), most graphs will converge to a visually comprehensible structure.

There are 3 parameters to tune the automatic layout process:

Ca Attractive force constant, which controls springs's force between balls. Default value is 0.15

Cr Repulsive force constant, which controls the repulsive force between balls. Default value is 0.15

Number of iterations
Self explained. Default value is 10.

For small topologies with tens of nodes, using the default parameters (perhaps with 20 to 30 more iterations) will suffice to produce a nice layout. But for larger topology, careful parameter tuning is necessary. Following is a empirical method to layout a 100 node random transit stub topology generated by Georgia Tech's ITM internet topology modeler. First, set Ca_ and Cr_ to 0.2, do about 30 iterations, then set Cr_ to 1.0, Ca_ to about 0.01, then do about 10 iterations, then set Ca_ to 0.5, Cr_ to 1.0, do about 6 iterations.

THE USER INTERFACE

The top of the **nam** *nam* window is a *menu bar.* Two pulldown menus are on the left of the menu bar. The 'File' menu currently only contains a 'Quit' button. It has a 'Open...' button as well, but that is not implemented yet. The 'View' menu has 4 buttons:

– New view button: Creates a new view of the same animation. User can scroll and zoom on the new view. All views will be animated synchronously.

– Show monitors checkbox: If checked, will show a pane at the lower half of window, where monitors will be displayed.

– Show autolayout checkbox: If checked, will show a pane at the lower half of window, which contains input boxes and a button for automatic layout adjusts. This box may not always be enabled. When a trace file has its own layout specifications, this box will be disabled. If and only if the trace file does not have complete layout specification (i.e., each link has orientation specified in the traces), will this box be enabled.

– Show annotation checkbox: If checked, will show a listbox at the lower half of window, which will be used to list annotations in the ascending order of time.

The 'Help' menu is on the right side of the menu bar. It has two buttons. Clicking the 'Help' button will pop up a new window showing information on nam usage. Clicking the 'About' button will pop up a new window showing history and status of nam.

Acceleration Keys

ALT+'f' will pull down the 'File' menu. ALT+'v' will pull down the 'Open...' menu. ESC will abort a menu selection in progress.

Below the menu bar, there is a *control bar* containing 6 buttons, a label, and a small scrollbar (scale). They can be clicked in any order. We will explain them from left to right.

Button 1 (<<)

Rewind. When clicked, animation time will go back at the rate of 25 times the current screen update rate.

Button 2 (<)

Backward play. When clicked, animation will be played backward in time.

Button 3 (square)

Stop. When clicked, animation will pause.

Button 4 (>)

Forward play. When clicked, animation will be played in time ascending order.

Button 5 (>>)

Fast Forward. When clicked, animation time will go forward at the rate of 25 times the current screen update rate.

Button 6 (Chevron logo)

Quit.

Time label

Show the current animation time (i.e., simulation time as in the trace file).

Rate slider

Controls the screen update rate (animation granularity). The current rate is displayed in the label above the slider.

Below the first control bar, there is *Main Display*, which contains a tool bar and a main view pane with two panning scroll bars. All new views created by menu button 'File/new view' will have these three components.
The tool bar contains two zoom buttons. The button with an up arrow zooms in, the button with a down arrrow zooms out. The two scroll bars are used to pan the main animation view.
Clicking the left button on any of the objects in the main view pane will pop up a information window at the clicking point. For packet and agent objects, there is a 'monitor' button in the popup window. Clicking that button will bring out the monitor pane (if it is not there), and add a monitor to the object. For link object, there will be a 'Graph' button. Clink that button will bring out another popup window, where user can select drawing bandwidth utilization graph or link loss graph of one of the two simplex links of the duplex link clicked on. These functionalities are also available in the views created by 'File/new view'.
NOTE: These functionalities are *HIGHLY EXPERIMENTAL AND UNSTABLE* in this release (v1.0a2).

Below the gadgets we have discussed so far, there may or may not be a *Monitor pane*, depending on whether the checkbox 'View/show monitors' is set. (The default is unset). All monitors will be shown in this pane. A monitor looks like a big button in the pane. Currently only packet and agent may have monitor. A packet monitor shows the size, id, and sent time. When the packet reaches its destination, the monitor will still be there, but saying the packet is invisible.
A agent monitor shows the name of the agent, and if there are any variable traces associated with this agent, they will be shown there as well.

Below the monitor pane (or in its place if the monitor pane isn't there), there is a *Time Slider*. It looks like a scaled rule, with a tag 'TIME' which can be dragged along the rule. It is used to set the current animation time. As you drag the 'TIME' tag, current animation time will be displayed in the time label in the control bar above. The left edge of the slider represents the earliest event time in the trace file and the right edge represents the last event time.
Clicking left button on the rule (not the tag) has the same effect as Rewind or Fast Forward, depending on the clicking position.

The *Automatic Layout Pane* can be visible or hidden. If visible, it is below the time slider. It has three input boxes and one relayout button. The labeled input boxes let user adjust two automatic layout constants, and the number of iterations during next layout. When user press ENTER in any of the input boxes, or click the 'relayout' button, that number of iterations will be performed. Refer to the AUTOMATIC LAYOUT section for details of usage.

The bottom component of the nam window is a *Annotation Listbox*, where annotations are displayed. An annotation is a (time, string) pair, which describes a event occuring at that time. Refer to *ns(1)* for functions to generate annotations. Double-click on an annotation in the listbox will bring nam to the time when that annotation is recorded.
When pointer is within the listbox, clicking right button will stop animation and bring up a popup menu with 3 options: Add, Delete, Info. 'Add' will bring up a dialog box with a text input and add a new annotation entry which has the current animation time. User can type annotation string in the dialog box. 'Delete' will delete the annotation entry pointed by the pointer. 'Info' will bring out a pane which shows both the annotation time and the annotation string.

KEYBOARD COMMANDS

[Incompelete, but accurate] Most of the buttons have keyboard equivalents. Note they only function when mouse cursor is inside the nam window.
Typing a space or return will pause nam if it's not already paused. If nam is paused, space or return will step the animation one simulated clock tick. (If your keyboard autorepeats, holding down space is a good way to slow-step through some part of the animation.)

'p' or 'P'
: Pause but not step if paused.

‘c’ or ‘C’
Continue after a pause.

‘b’ or ‘B’
Descrease animation time for one screen update interval.

‘r’ or ‘R’
Rewind.

‘f’ or ‘F’
Fast Forward.

‘n’ or ‘N’
Move to next event.

‘x’ or ‘X’
Undo the last rate change

‘u’ or ‘U’
Undo the last time slider dragging.

‘>’ or ‘.’
Increase the granularity (speed up) by 5%.

‘<’ or ‘,’
Decrease the granularity (slow down) by 5%.

SPACE Toggle the pause state of nam.

‘q’, ‘Q’ or Control-c
Quit

RECORDING ANIMATIONS

To record nam animations, select the “Record Animation” option under the file menu. A series of namXXX.xwd files will be produced (where XXX is the frame number), one per time-step. These files can then be assembled into animated GIFs or MPEGs with the appropriate post-processing tools.

TRACE FILE FORMAT

The trace file events can be divided into 6 types, depending on to which object the event is associated. Below, we discuss them in detail.

Packet Basic packet events are a type character, followed by some tags:

<type> -t <time> -e <extent> -s <src_addr> -d <dst_addr> -c <conv> -i <id>

<type> is one of:

‘h’ - Hop. The packet started to be transmitted on the link from src_addr to dst_addr
‘r’ - Receive. The packet finished transmission and started to be received at the destination.
‘d’ - Drop. The packet was dropped from queue or link from src_addr to dst_addr.
‘+’ - Enter queue. The packet entered the queue from src_addr to dst_addr.
‘-’ - Leave queue. The packet left the queue from src_addr to dst_addr.

Drop here doesn’t distinguish between dropping from queue or link. This is decided by the drop time.

The flags have the following meanings:

-t <time> is the time the event occurred.
-e <extent> is the size (in bytes) of the packet.
-s <src> is the originating node.
-d <dst> is the destination node.

-c <conv> is the conversation id.
-i <id> is the packet id in the conversation.
-a <attr> is the packet attribute, which is currently used as color id.

Additional flags may be added for some protocols. This list may be extended as required:

-P <pkt_type> gives an ASCII string specifying a comma separated list of packet types. Some values are: TCP - a tcp data packet. ACK - generic acknowledgement. NACK - generic negative acknowledgement. SRM - SRM data packet.
-n <sequence number> gives the packet sequence number.

Link/Queue State

l -t <time> -s <src> -d <dst> -S <state> [-c <color>] [-r <bw> -D <delay>]
q -t <time> -s <src> -d <dst> -a <attr>

<state> gives the link state transition. It has 3 possible values: UP and DOWN marks link failure and recovery, COLOR marks link color change. If COLOR is given, a following -c <color> is expected which gives the new color value. In link event, [-r <bw> -D<delay>] gives link bandwidth and delay, respectively. It is only used when nam creates the link, i.e., loading the trace file.
<attr> specifies the queue position, i.e., the angle between the link along which queued packets are displayed and the horizontal line.

Node State

n -t <time> -s <src> -S <state> [-c <color>] [-o <color>] [-A <labels>]

Flags '-t', '-S' and '-c' have the same meaning as those in Link. Flag '-A' is used to add a arbitrary string to the label of the node. It can be used to display explainations of a node's state. Flag '-o' is used in backtracing to restore old colors of a node.

Node Mark

Node marks are colored circles around nodes. They are created by:

m -t <time> -n <mark name> -s <node> -c <color> -h <shape> [-o <color>]

and can be deleted by:

m -t <time> -n <mark name> -s <node> -X

Note that once created, a node mark cannot change its shape. The possible choices for shapes are, circle, square, and hexagon. They are defined as lower-case strings exactly as above.

Protocol State

Agents can be constructed by:

a -t <time> -n <agent name> -s <src> -d <dst>

They can be destructed by:

a -t <time> -n <agent name> -s <src> -d <dst> -X

To visualize protocol state variables associated with an agent, we use the name ‘feature’. Currently we allow three types of features: *timers*, *lists* and simple *variables*. But only the last one is implemented in *ns(1)* tracing APIs.

Features may be added or modified at any time after agent creation using:

f -t <time> -a <agent name> -T <type> -n <var name> -v <value> -o <prev value>

<type> is ‘l’ for a list, ‘v’ for a simple variable, ‘s’ for a stopped timer, ‘u’ for an up-counting timer, ‘d’ for a down-counting timer.
-v <value> gives the new value of the variable. *Variable* values are simple ASCII strings obeying the TCL string quoting conventions. *List* values obey the TCL list conventions. *Timer* values are ASCII numeric values.
-o <prev value> gives the previous value of the variable. This is to allow backward play of animation.

Features may be deleted using:

f -t <time> -a <agent name> -n <var name> -o <prev value> -X

Misc v -t <time> *TCL script string*

is used for annotation, it may includes an arbitrary tcl script to be executed at a given time, as long as the script is in one line (no more than 256 characters). The order of flag and the string is important.

c -t <time> -i <color id> -n <color name>

defines a color. The color name should be one of the names listed in color database in X11 (/usr/X11/lib/rgb.txt). After this definition, the color can be referenced using its id.

EXAMPLES

FILES

/usr/lib/X11/rgb.txt

SEE ALSO

tcpdump(1)

[1] Fruchterman, T.M.J. and Reingold, E.M., Graph Drawing by Force-directed Placement, Software - Practice and Experience, vol. 21(11), 1129-1164, (November 1991).

[2] Amir, E., Carta: A Network Topology Presentation Tool, Project Report, EECS Dept., UC Berkeley, 1993. http://http.cs.berkeley.edu/~elan/mbone.html

Mailing lists for nam users and announcements are the same as those for ns users. Send email to ns-users-request@mash.cs.berkeley.edu or ns-announce-request@mash.cs.berkeley.edu to join. Questions should be forwarded to ns-users@mash.cs.berkeley.edu, ns-announce will be low-traffic announcements only.

BUGS

This manual page is incomplete.