# Visualization of Node Interaction Dynamics in Network Traces

Jürgen Schönwälder

JACOBS
UNIVERSITY

AIMS 2009, Enschede, 2009-07-01

Co-authors: Petar Dobrev, Sorin Stancu-Mara
Support: EU IST-EMANICS Network of Excellence (#26854)

# Outline of the Talk

# Background and Motivation

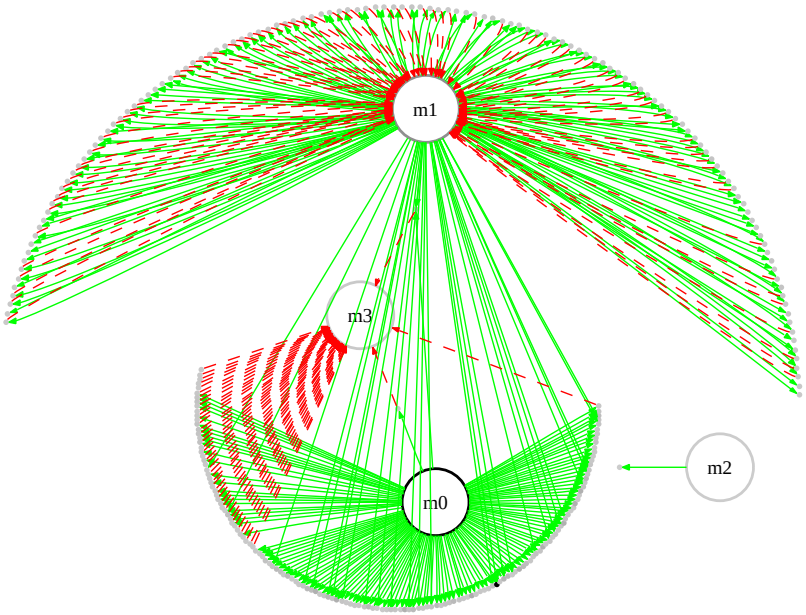
# Background: SNMP Trace Collection and Analysis

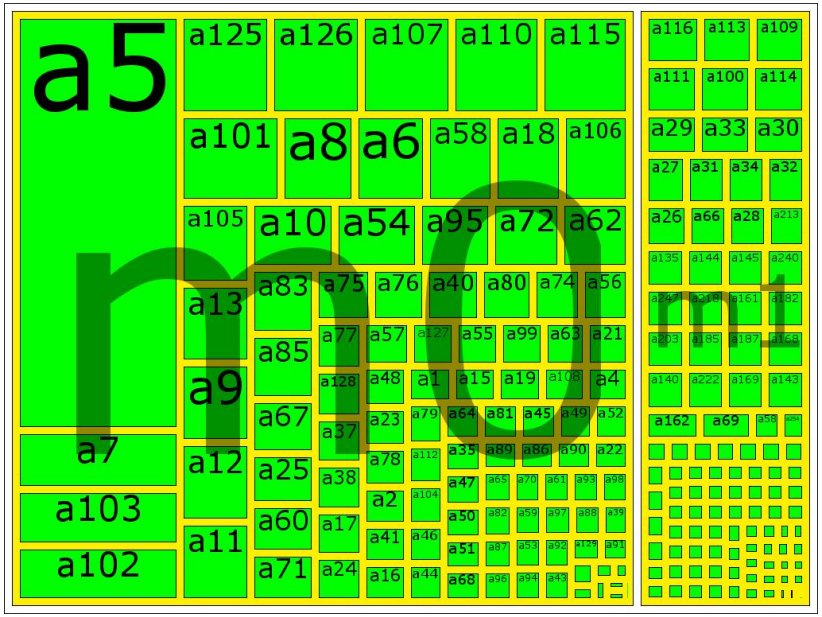
- SNMP trace collection and analysis developed since 2006 by the NMRG of the IRTF and the EMANICS project

- Insights to be gained from the analysis of traces:
  - basic trace statistics
  - periodic versus aperiodic traffic
  - message size distributions and latency distributions
  - manager and agent concurrency levels
  - table retrieval algorithms
  - popular MIB modules and MIB objects
  - usage of protocol specific features
  - ...

- For more background information, consult [1, 2, 3].

# Early Analysis Results (trace l06t01, IM 2007)




- Analysis covering the whole trace capture period
- Traffic broken down into flows ($\neq$ NetFlow/IPFIX flows)
- Static visualizations using flow graph layouts (Graphviz) and flow traffic treemaps

# Visualization of Node Interaction Dynamics

## Some Research Questions

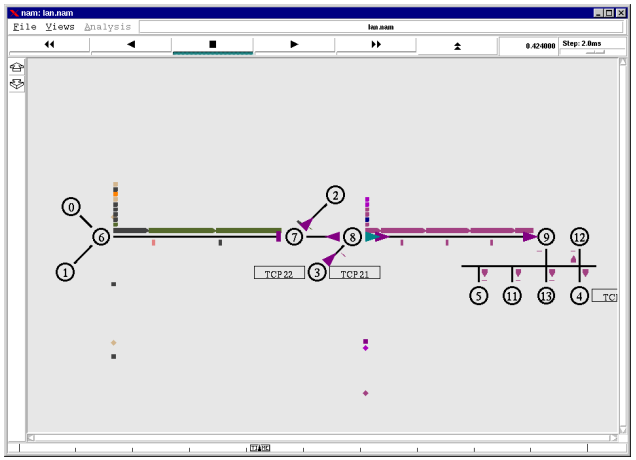
- Does a data collection engine spreads the data retrieval traffic over a polling cycle or does it send a burst of polling requests at the start of each cycle?
- Are there any topology changes or even pattern of topology changes?
- Do notifications change the traffic pattern generated by management applications?

## Approach

- Visualize node interaction dynamics in order to gain insights into the short-term properties of SNMP traffic.

# Experiment #1: NAM (C, C++, Tcl/Tk)

# Network Animator NAM



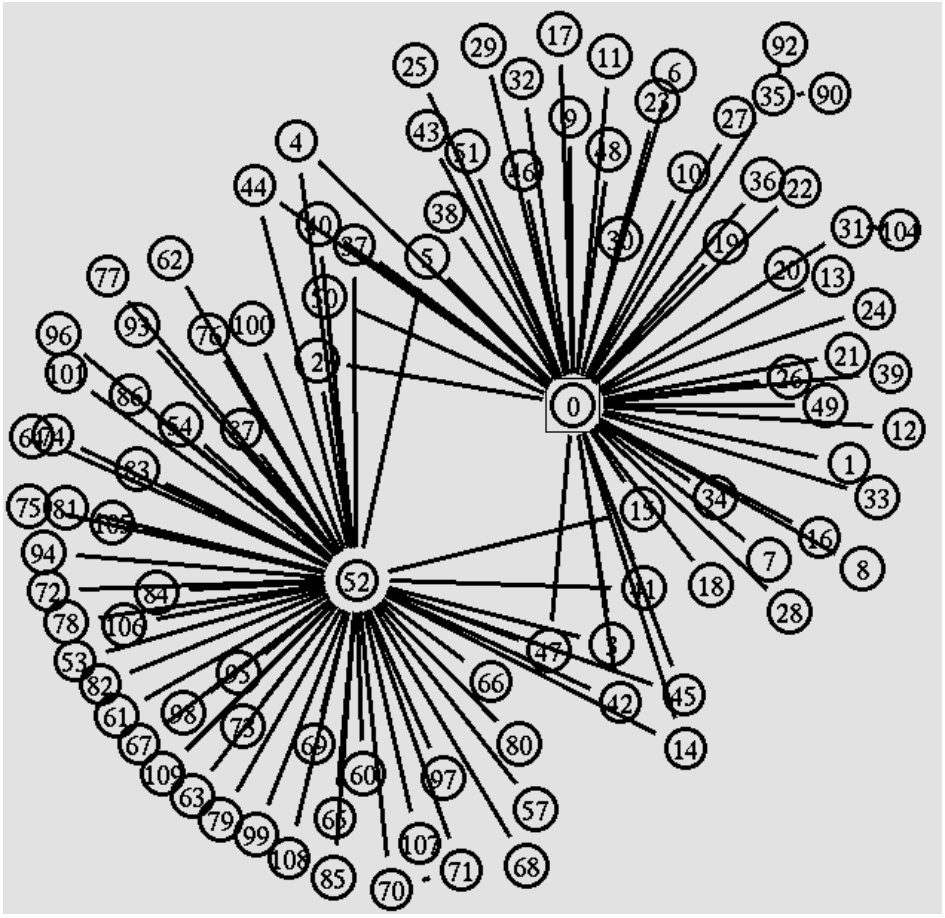

- Animation tool for viewing network simulation traces and real world packet traces
- Created in 1990 at LBL and further developed at ISI
- Includes an editor to create network diagrams
- Supports the visualization of queues and packet drops

# Visualizing SNMP Traces with NAM

## Conversion Algorithm

- Read an SNMP trace file in the CSV format
- Generate a NAM trace file header defining nodes and links
- Generate send/receive events for each SNMP message
- Set the link delay to 60 seconds (parameter)
- Set the link data rate to 1kbps (parameter)

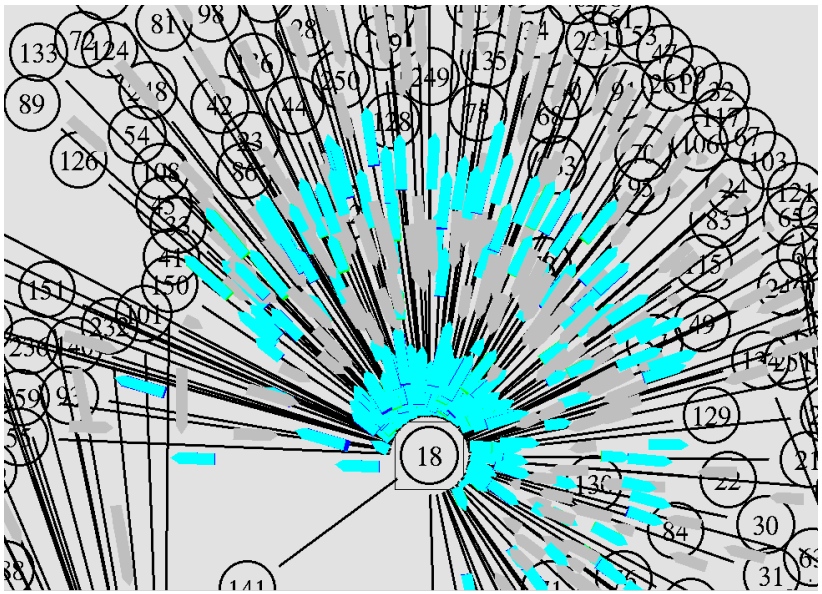
# NAM Results



The graph layouts produced by NAM are of relativly poor quality (overlapping nodes, overlapping links)

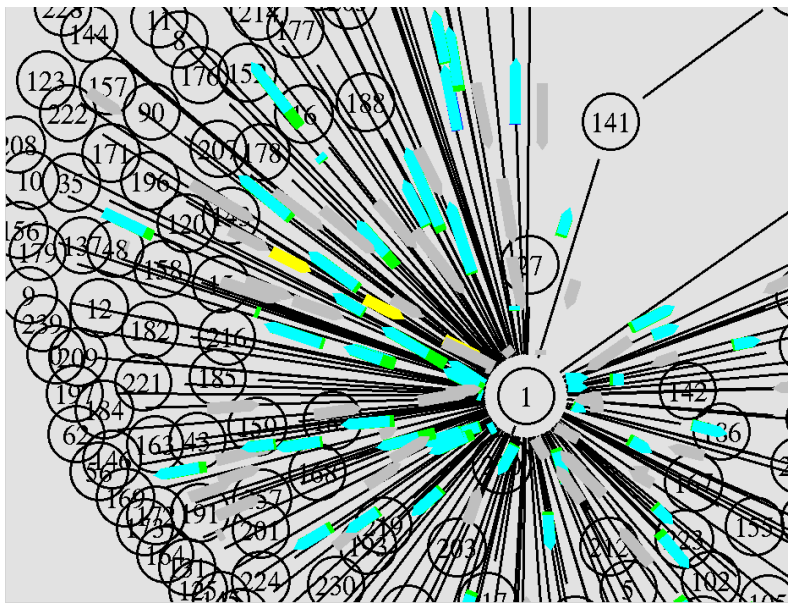
The builtin zooming capabilites of NAM are only of little help

# NAM Results (trace l06t01)




A manager polling devices in “waves”, causing bursts of SNMP traffic

A manager distributing polling requests well over time plus a notification originating from one agent



# NAM Discussion

## Positive NAM Experience

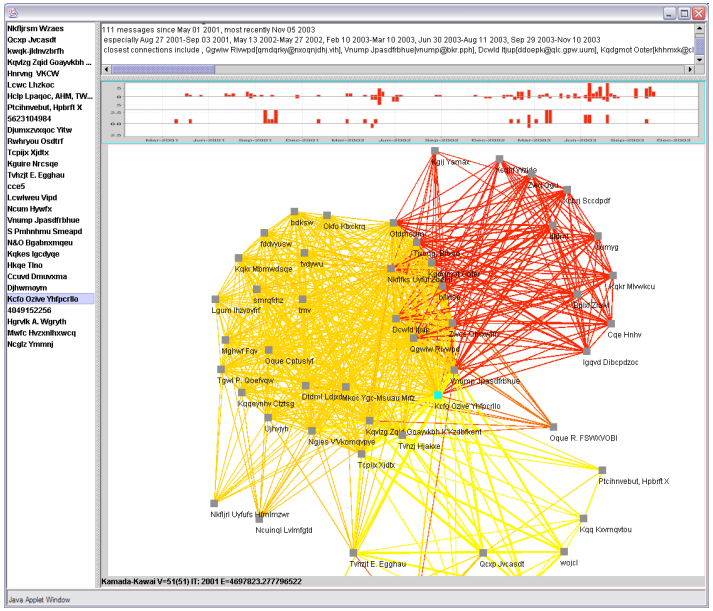
+ Availability (part of the popular ns-2 simulation toolkit)
+ Easy to get started (textual intermediate file format)

## Negative NAM Experience

- Difficult to extend mixture of C, C++, and Tcl/Tk
- Weak deeply embedded graph layout algorithm
- Graphics quality poor according to today's standards
- Performance problems when dealing with larger traces
- GUI controls optimized for short timescales

# Experiment #2: JUNG (Java)

# JUNG Graphics Library



- Java Universal Network / Graph Framework (JUNG)
- Written in 2003 by PhD students (University of California at Irvine)
- Analysis of social networks

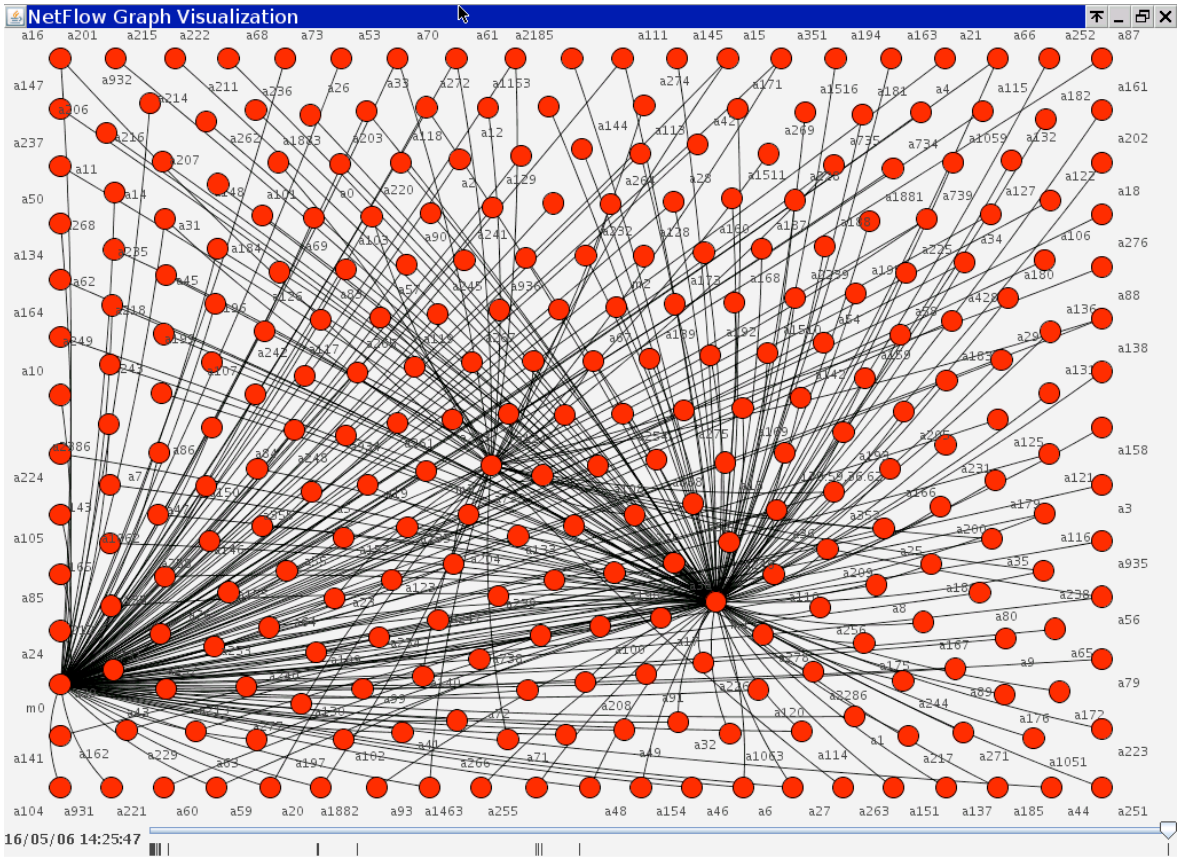
- Our goal: visualize topology changes over time (nodes / links appearing / disappearing)
- Generate an intermediate file with graph change events
- Visualize the changing graph using JUNG

# Visualizing SNMP Traces with JUNG

## Conversion Algorithm

- Read an SNMP trace file in the CSV format
- Generate vertex add (Va) and edge add (Ea) events when a new node appears or a node starts communication with some other node it has not communicated with recently
- After $t_{link}$ seconds of inactivity, an edge is removed by generating an edge remove (Er) event
- A node without any links is removed after $t_{node}$ seconds by generating a vertex remove (Vr) event

# JUNG Results




The timeline shown at the bottom of the window indicates when changes happen in the topology

# JUNG Discussion

## Positive JUNG Experience

+ Much better graph layout algorithms (compared to NAM)
+ Very good documentation of the JUNG library
+ Many code examples make it easy to get started

## Negative JUNG Experience

- Performance problems with the animation of larger graphs
- Topology change timeline is essential due to change blindness of the human visual perception system
- Controlling graph “stability” is complicated

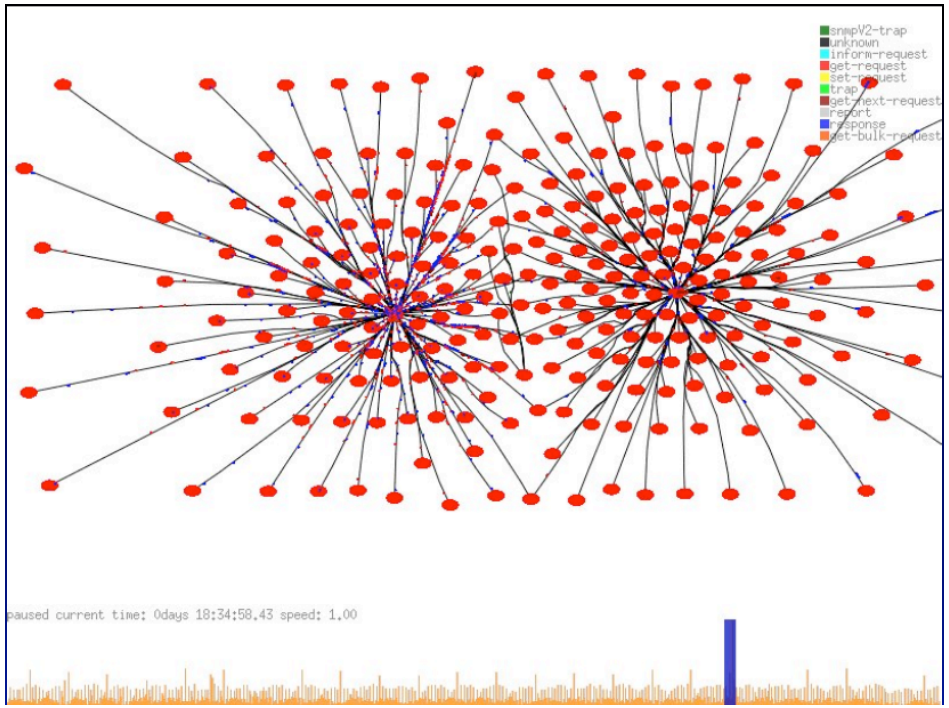
# Experiment #3: SNAM (OpenGL)

# SNAM

## Features

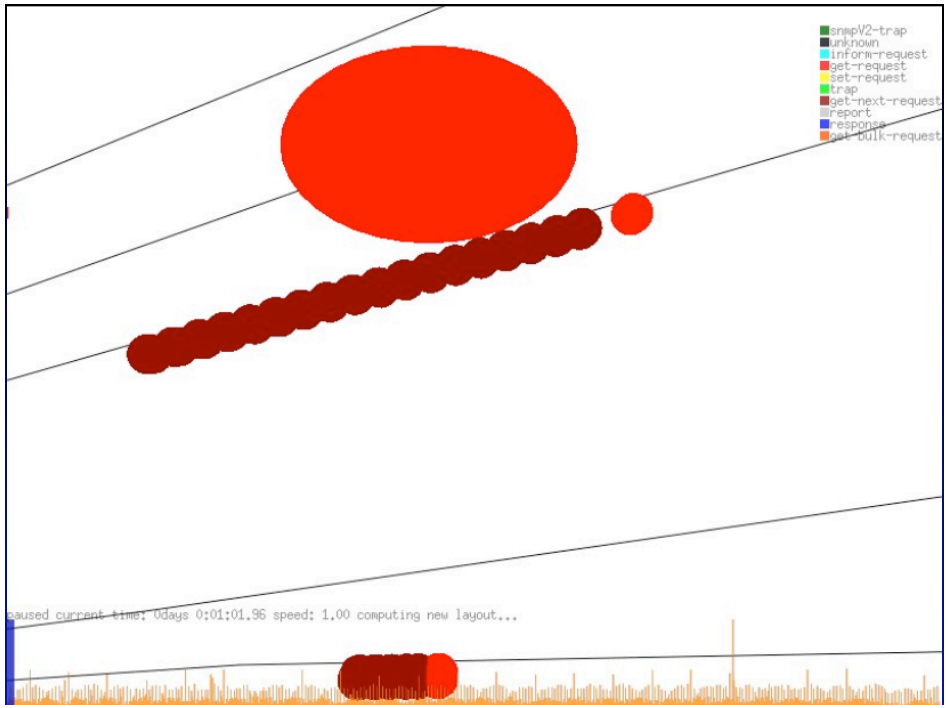
- Written in C++ utilizing fast graphics hardware
- Graph layout outsourced to Graphviz
- Layout recalculations run as a separate thread
- Links can be represented as splines
- Supports fast forward/backward animations and jumps
- Inspection of annotations of nodes, links, messages
- Uses an “extended subset” of the NAM file format

# SNAM Results #1



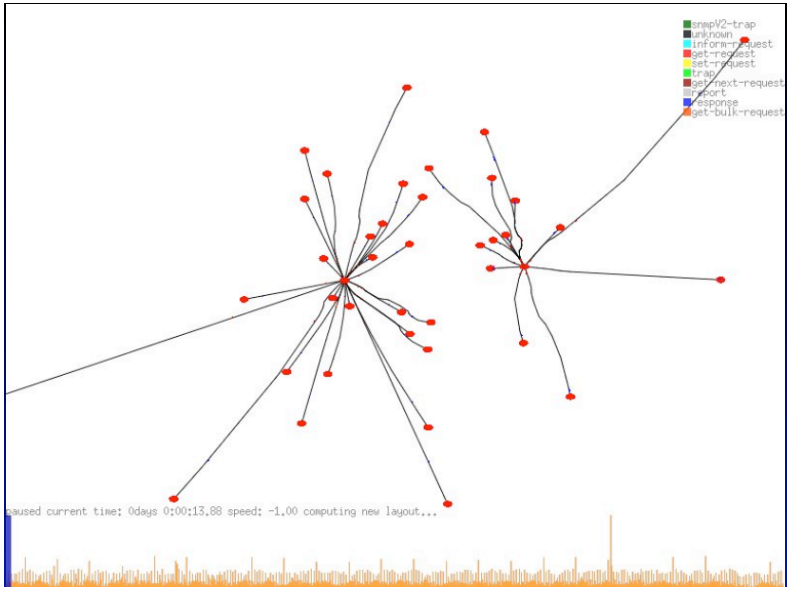

- Topology, color coded messages, activity timeline
- Textual information displayed in the background
- Controls to zoom, change speed, jump in the timeline, ...

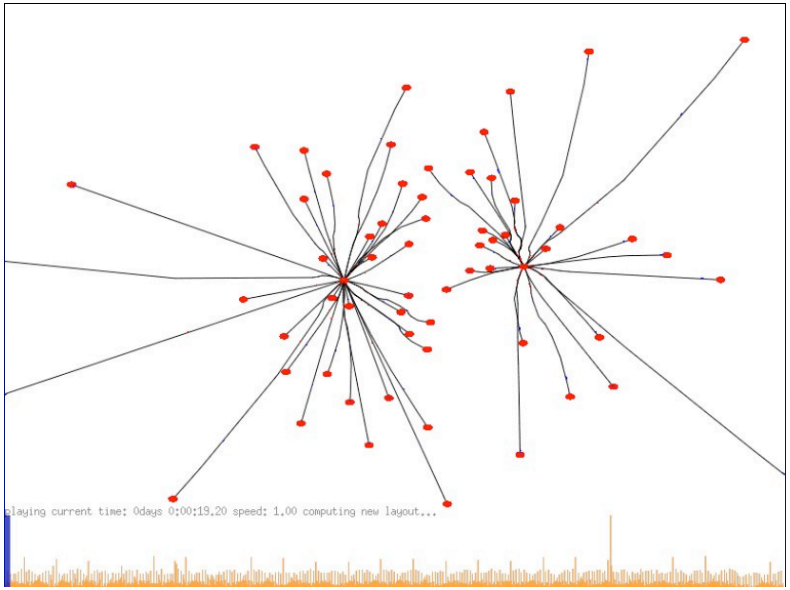
# SNAM Results #2

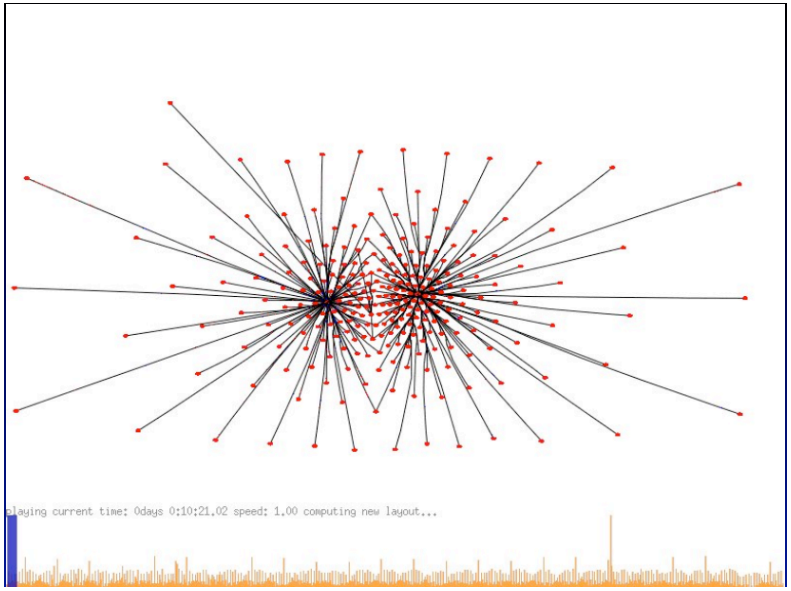



- A manager trying a get-bulk-request and then falling back to a series of get-next-requests
- Obviously not caching learned SNMP agent capabilities

# SNAM Results #3



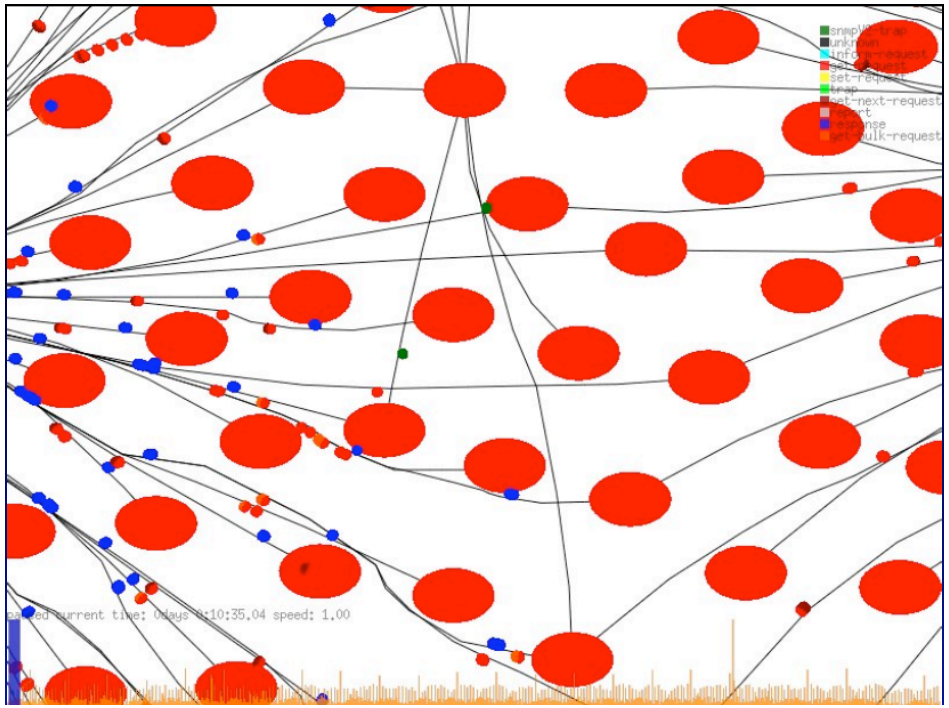




The graph topology changes dynamically (similar to the JUNG prototype)

New layouts are calculated in the background

# SNAM Results #4



- Graph layout can use splines to avoid links crossing nodes
- Messages travel along the splines
- Zooming is cool — lenses might be even cooler...

# SNAM Discussion

## Positive SNAM Experience

+ OpenGL is crucial for scalable fast visualizations
+ Using Graphviz allows us to focus on the main problem
+ Simple installation and high fun factor
+ Intermediate file format decouples visualization from the data source

## Negative SNAM Experience

- Selecting suitable Graphviz options sometimes difficult
- Significant startup time (no streaming of trace data)
- Portable user controls take effort to implement
- Incomplete feature set (only subset of NAM supported)

# Conclusions

# Conclusions

## Contribution

- Created three tools to visualize node interactions
- Scalability is important for dealing with larger traces
- Reuse graph drawing libraries, use multi-threading
- OpenGL solution (SNAM) has the highest fun factor

## Future Work

- SNAM will be open sourced (its pretty small)
- We like to encourage people to try and improve it
- More visualizations (multiple alternate views)
- More interactive controls (select, highlight, filter, . . . )

# References


J. Schönwälder.

SNMP Traffic Measurements and Trace Exchange Formats.
RFC 5345, Jacobs University Bremen, October 2008.

J.G. van den Broek, J. Schönwälder, A. Pras, and M. Harvan.

SNMP Trace Analysis Definitions.
In *Proc. of the 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2008)*, number 5127 in LNCS. Springer, July 2008.

J. Schönwälder, A. Pras, M. Harvan, J. Schippers, and R. van de Meent.

SNMP Traffic Analysis: Approaches, Tools, and First Results.
In *Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007)*, pages 323–332, May 2007.

P. Dobrev, S. Stancu-Mara, and J. Schönwälder.

Visualization of Node Interaction Dynamics in Network Traces.
In *Proc. of the 3rd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2009)*, number 5637 in LNCS, pages 147–160. Springer, June 2009.