Visualization of Router Statistics

Ken Schmidt

Introduction

- To compare the performance of router designs, monitor router statistics
- Two ways, raw data, graphical representation
  - Data may be generated very quickly
  - Humans evolved to quickly interpret visual information
  - Humans can not understand raw data very quickly

What Is Available

- 100s of commercial and open source packages available
- Oriented towards network managers
- Many use RMON, which we are not implementing
- All will display a router statistic vs time
  - We want the option to plot one statistic value vs another
- Some are real-time, others are static

Publications on Router Statistics Visualization

- Journal publications for router statistics visualization do not exist
- There are many web sites with commercial and open source programs for visualization
- Two were the most comprehensive:
  - http://www.caida.org/tools/taxonomy/ CAIDA: Cooperative Association for Internet Data Analysis
  - http://www.slac.stanford.edu/~cottrell/tcom/nmtf-tools.html SLAC: Stanford Linear Acceleration Center
- Of the programs that I look at, all produced some statistic vs time
- Some were static, some were active (real time)

What Statistics May Be Monitored From a Router

Many statistics are available for routers

- Response time
- Delay-Latency
- Throughput
- Bandwidth
- Queue size (length)
- Queue fill (or fill between high and low limits)
- Loss
- Packet length
- Packet rate
- Efficiency
- Utilization
- No of flows
- Among others

Plot one statistics vs another

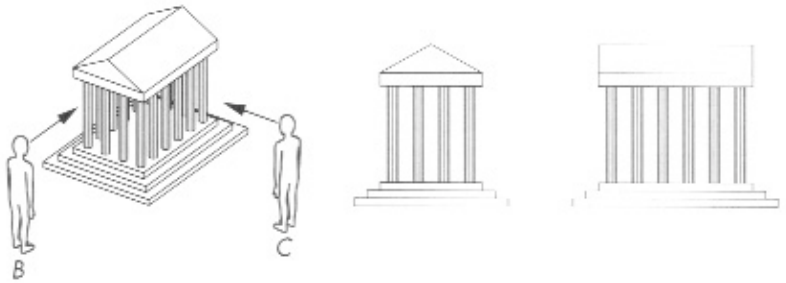
- It is desirable to plot one of these statistics vs another as well as plotting one vs time as most available software does
- It is desirable to do this in real time
- Plotting many points of a statistics pair will show how these statistics are related over time

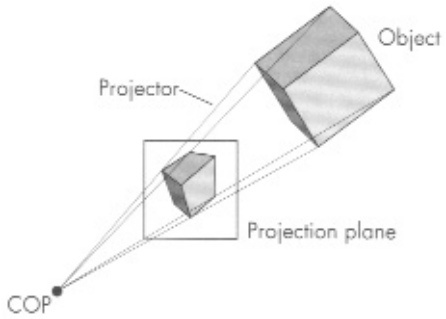
OpenGL Was Chosen

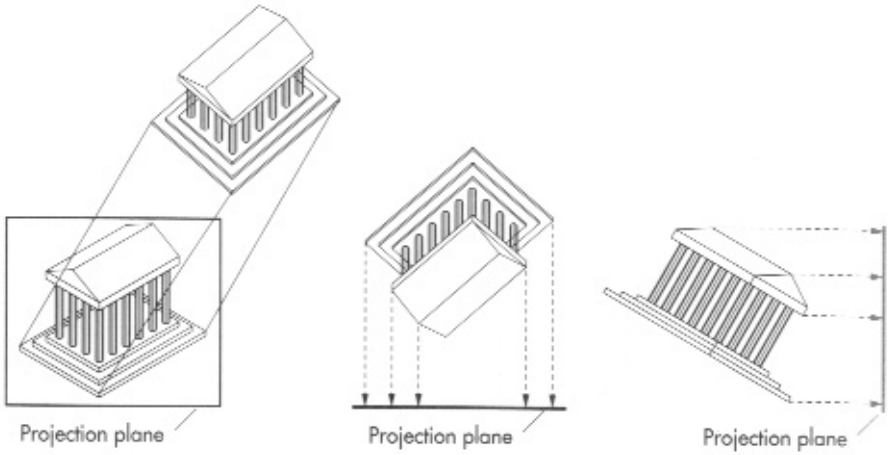
- OpenGL is relatively easy to use (with a moderate learning curve)
- OpenGL can display data in real time
- OpenGL is multi-platform
- OpenGL integrates with many languages, including C++

OpenGL Characteristics

- Rotation
- Translation
- Scaling
- Projection

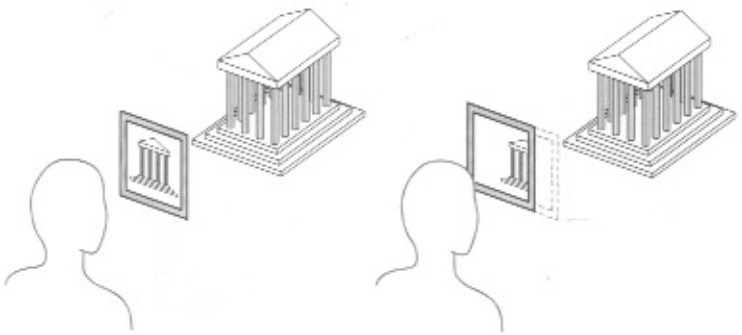






OpenGL Characteristics

- The viewing window can obscure part of the image



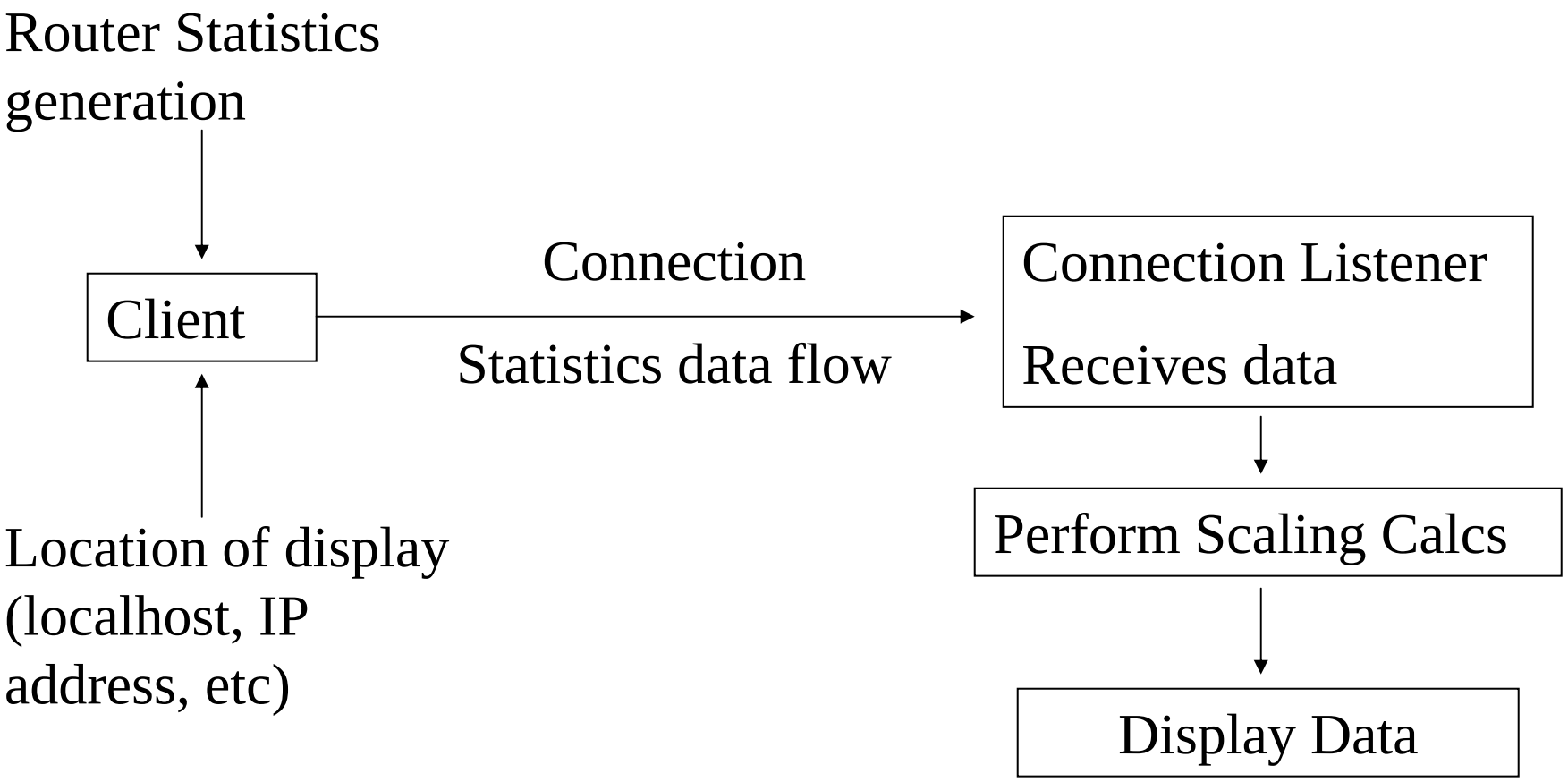
Implementation Requirements

- Must be fast – points may be generated very quickly
  - Arrays are faster than lists (but fixed size)
  - A large number of points takes longer to display
  - Scaling for each point is costly (points may go out of the viewing window)
- Enough points for a meaningful representation
  - Too few points are not representative
  - Too many points may illuminate every pixel on the screen

Implementation Characteristics

- It was decided to create a display that could be used on any inputs to graphically represent any data and link it to the router statistics generator in this instance
- To make the display more useful, it was implemented with a listener so that the router could be remotely monitored

Interface Block Diagram



Client Side Code

- Input:
  - Names of the display axes
  - Data Points from statistics generator
  - Location of display (IP address or name)
- Output:
  - A connection to the display on port 32767 at it’s address
  - Flow of data
    - Names of axes
    - Statistics points

Client Side Code

- Connect to the display
- Main loop is a while(true) loop which sends data, receives an ack
- Data format is a four-tuple x x x x where the data is a string separated by spaces; data types are (respectively):
  - 0 or 1, 0 = names follow, 1 = data follows
  - 1 through 9, the number of the destination screen (there are 9 possible screens)
  - Name of X coordinate or data value for the X coordinate
  - Name of Y coordinate or data value for the Y coordinate
  - Example: 0 1 throughput queue_length = names for screen 1 with x axis label “throughput, y axis label queue_length
  - Example: 1 2 4.5 6.7 = data for screen 2, x value of 4.5, y value of 6.7

http://www.cs.kent.edu/~kschmidt/visualization/visClient.cpp.pdf

Visualization Side Code

- Inputs:
  - A connection from a client
  - Flow of data
    - Names of axes
    - Statistics points
  - Selection of screen to be viewed from user via keyboard
- Outputs:
  - Display of data on one of 9 screens, displayed in real time as points are received, normalized to fit on the screen
    - While the windows may be moved to different areas of the screen, only the active window (the one which has focus) will display data in real time, the others will be static displays until they receive focus by keyboard selection

Visualization Side Code

- Arrays were used to store data points since they are faster than lists and data may arrive quickly from the router
- Global variables were used in lieu of message passing between threads for faster response time
- Main consists of
  - Starting a new thread for the connection listener
  - Parent thread:
    - Displays a help messagebox
    - Displays a blank window (there always has to be some window displayed and the user has not yet selected a window to view)
    - Starts the main loop which listens for an event (keyboard entry, mouse click, etc)

Listener Function (child thread)

- Listens for a connection from a client
- Upon connecting, waits for data
- Upon receiving data, determines data type (name data or numerical data)
- Name data is strcpy’ed to the variable appropriate for the intended screen

Listener Function (child thread)

- Numerical data is
  - Sent to the appropriate screen
  - Normalized to a max value of 1 to fit on the visible screen
    - Max value calculation occurs when 1%, 10% and 50% of the data points have filled the array
    - The screen is 50% larger than points normalized to 1, so that new points that scale to greater than 1 will still be on screen until the next scheduled max value calculation
  - Added to the array
    - The array is a circular list so that after it has been filled, new points overwrite the oldest points

Parent Thread

- The main loop waits for keyboard input which selects screen 1 through 9 with the number keys, h for the help screen, m key to see the max values used for scaling, or esc to kill the child thread (the listener) and exit the program
- When a screen is selected, it is displayed using:
  - A double buffer so that new points are plotted in a buffer until they are all plotted, then the buffers are switched; this eliminates jitter in the screens as they are being painted
  - The window is located on the screen and sized
  - The colors are set (just black and white, but it must be told what colors to use)
  - The size of the points is set to 2 pixels, with one pixel they are just not visible enough

Parent Thread

- The image is projected to the front plane
- The visible world is defined (-0.1 to1.5 units x direction, -0.1 to 1.5 units y direction, -1.0 to 1.0 z direction)
- The keyboard is activated
- The x and y axes are drawn on the screen with tic marks showing where the max value is located on each axis at the last time the max value was calculated
- The idle function is activated which continuously redisplays the screen of focus with all points that are currently available which may change over time

http://www.cs.kent.edu/~kschmidt/visualization/vis.cpp.pdf

Some Considerations

- Remote use of the display
  - Long times to transfer data
    - Remote display was reliable >100ms, <200ms
    - Local data was completely reliable with less than 1ms delay
- The display must be started before the router asks for a connection

Conclusions

- Nothing is currently available which is specifically designed for data displays for router developers
- Available software is oriented towards the network manager and displays data vs time
- Router statistics are more quickly understood graphically
- Graphical displays must be as fast as the router generating the data
- OpenGL is a useful tool to generate real-time graphical displays

http://www.cs.kent.edu/~kschmidt/visualizatoin/