Network Visualization with Nodemap
Agenda

- What is *Internode Nodemap*?
- Why would I use it?
- How do I configure it?
- How do I run it?
- Where can I find out more?
What do I need first?

- Basic knowledge of SNMP, network monitoring.
- Basic Cisco knowledge -- don’t need a CCNA, but you should know what an interface is.
- A UNIX box (FreeBSD, Linux, Solaris, Mac OS X, ...)
- A network. *Duh.*
Network visualization with Internode Nodemap

- *Internode Nodemap* is a network monitoring, diagnostic, mapping and visualization application
- Displays in your web browser – no client software needed
- Near real-time visual feedback
- Expressive configuration language
- Scalability: monitors 100’s of routers, 1000’s of links, from one host
Why Nodemap?

- Status monitoring on large networks is “hard”
- Tools like MRTG are good at the “micro” view, but don’t work holistically
- Tools like Nagios and mon work holistically, but are bad at managing relationships
Why Nodemap?

- Tracking DoS attacks with thousands of MRTG graphs is Not Fun™

- Getting dozens of Nagios SMS messages about latency and reachability without knowing there’s a DoS in progress is also Not Fun™
Why Nodemap?

- Nodemap fills the gap.
- Provides a holistic visualization of an entire network
- More detailed views available on-demand
- Supplements, doesn’t replace: Other monitoring tools are still needed, but Nodemap makes interpreting their results easier.
Why Nodemap?

- Additional uses:
  - **Network mapping**
    Nodemap config file gives you automatically-maintained network diagrams for free!
  - **Fault-finding**
    Operations staff can use visual feedback to assist with performance and reachability fault resolution
Nodemap display

- Network administrators define maps
- A map consists of an image, a set of nodes, and a set of links between the nodes
Nodemap display

- Links change color depending on status
- Color changes occur with load increases, congestion and packet loss
Nodemap display

- Nodes can have additional maps “nested” within.
- Nodes change color depending on the state of links within nested maps.
Nodemap display

- Hovering the mouse over a link provides link statistics
- Stats are updated every few minutes
Nodemap display

- Hovering over a node lists all the maps nested inside.
- If any nested links are non-green, it tells you about that too.
How?

Nodemap Configuration
**Nodemap Configuration**

- Nodemap reads a config file
- Heirarchical tree of named objects

```
/* Comment text */
objecttype OBJECTNAME {
  attribute VALUE;
  ...
};
```
**Nodemap Configuration**

- Top level of the tree is a “map” object
- A map contains a source (background) image and the node and link objects to be painted on that map
- Image format is GIF (Now patent-free!)
Nodemap Configuration

map adelaide {
    /* Background image definition */
    image images/adl_ubd_mapscan.gif;

    /* nodes and links go here */
};
**Nodemap Configuration**

- Node objects minimally contain “x” and “y” coordinates.

- Coordinates count in pixels from the top left of the source image (i.e., each map has an independent coordinate system)

- Nodes optionally have other attributes
Nodemap Configuration

map adelaide {
    image images/adl_ubd_dirscan.gif;

    node richmond {
        x 240;
        y 320;
    };

    node kensington {
        x 490;
        y 310;
    };
};
Nodemap Configuration

- “Link” objects connect the nodes

- Link definitions describe the nodes the link connects, other nodes it passes through on the way, and either one or two “endpoint” object definitions

- If two endpoints are provided, their usage stats are averaged
**Nodemap Configuration**

- “Endpoint” objects describe a router / switch interface which Nodemap can query for performance stats.

- Nodemap uses SNMP – The router or switch must be able to respond to SNMP queries (RFC-1213 INTERFACES-MIB and Cisco gunk).
link r_to_k {
    between richmond kensington;
    endpoint richmond_outside {
        host richmond-router.company.com;
        interface Serial0;
        location richmond;
    };
    endpoint kensington_outside {
        host kensington-router.company.com;
        interface Serial4;
        location kensington;
    };
};
Nodemap Configuration

• Maps can be nested inside nodes:

```plaintext
map adelaide {
    image images/adelaide.gif;
    node richmond {
        x 150; y 320;
        map south_road {
            ...
        };
        map railway_terrace {
            ...
        };
    };
};
```
Nodemap Configuration

• Extra node attributes:
  ✓ terminal
  ✓ hide
  ✓ url

• Extra link attributes:
  ✓ thickness <thind|medium|thick|obese|XX>
  ✓ shaded
  ✓ bandwidth XX <kbps|mbps|k|m>
  ✓ url
**Nodemap Configuration**

- **Latency measurement:**
  ```
  ping X.X.X.X from ENDPOINTNAME;
ping Y.Y.Y.Y from @host:interface;
  ```

- **Feature is Cisco-centric.** If you don't have a Cisco and you need this to work, *send patches*. Thanks.

- **@host:interface** syntax permits ping tests from a nearby router when the endpoint is a switch
map sa {
    image images/sa.gif;

    node adelaide { x 160; y 240; };
    node mt_gambier { x 235; y 360; terminal; };

    link adl_mtg {
        bandwidth 48M;
        between adelaide mt_gambier;
        ping 192.168.2.41 from adlborder;
        endpoint mtgborder {
            host 192.168.2.41;
            interface Serial2;
            location mt_gambier;
        };
        endpoint adlborder {
            host 192.168.2.40;
            interface Serial0;
            location adelaide;
        };
    };
};
Configuration Example
Nodemap Runtime

- Once you have a config file, you need to get Nodemap to render it.
- The software has an installation directory (e.g., /usr/local/nodemap)
- Default config file is etc/nodemap.cf inside the installation directory: /usr/local/nodemap/etc/nodemap.cf
Nodemap Runtime

- You also need an output directory
  `/usr/local/www/data/nodemap/`

- Output directory needs to be populated with `.css` and `.js` files which come in Nodemap’s `webroot/` distribution directory

- Default is your current directory
Nodemap Runtime

- Command to invoke is `scheduler` in the installation directory:
  ```
  # ./scheduler [-p] [-d]
  [-c config-file]
  [-o output-directory]
  mapname
  ```

- The scheduler will render the map called `mapname` and any other maps nested within.

- `-p` disables packetloss/latency checks
Nodemap Runtime

- `scheduler` forks other processes to perform SNMP queries and rsh/ssh checks for packetloss and latency
- Resource limits are enforced to make sure it doesn’t fork enough processes to kill your system (even with large configs)
- Use `-d` to debug
Нodemap Runtime

• Three types of scheduled jobs:
  
  – *update_stats*
    Collects SNMP stats from routers
  
  – *update_pktloss*
    Uses rsh/ssh to ping across links
  
  – *update_nodemap*
    Redraws the map
**Nodemap Runtime**

- Jobs are scheduled “intelligently”

- `update_pktloss` and `update_stats` scheduled at random intervals (less than 5 minutes) to avoid load surges

- `update_nodemap` every 2 minutes

- Jobs rescheduled if system is too busy
Nodemap Runtime

• Status reflected in “ps” output:

newton@nodemap> ps ax | grep nodemap
41331 ?? Ss 12:51.90 nodemap sleeping for 1 seconds with 4 slots (perl)
98103 p5 R+ 0:00.00 grep nodemap
newton@nodemap> ps ax | grep pinger
1106 ?? S 0:00.05 pinger: 192.168.21.4 from bdr1.adl:Ethernet0/2
1121 ?? S 0:00.05 pinger: 192.168.126.118 from bdr1.bne:Serial16
1148 p5 RV 0:00.00 grep pinger (tcsh)
newton@nodemap> ps ax | grep snmp
2858 ?? R 0:00.37 /usr/bin/perl ./snmp_show_int -v 2c -h lns1.adl Gig
2875 ?? R 0:00.08 /usr/bin/perl ./snmp_show_int -v 2c -h lns1.syd Gig
newton@nodemap>
Nodemap Resources

- **Website:**
  
  http://nodemap.internode.on.net/

- **Manual**

  http://nodemap.internode.on.net/docs.html

- **Mailing list**

  nodemap-users@lists.internode.on.net
We're done!

Any questions?