ITSA Scenarios

July 8th, 2005

Jane Curry

Skills 1st Ltd

Jane Curry
Skills 1st Ltd
2 Cedar Chase
Taplow
Maidenhead
SL6 0EU
01628 782565

jane.curry@skills-1st.co.uk

Synopsis

IBM offers IBM Tivoli NetView as its IP network management solution. Whilst providing a good IP Layer 3 solution, NetView alone provides limited functionality to monitor ports on switches.

This paper looks at IBM Tivoli Switch Analyzer 1.3, released in March 2005, which delivers a graphical representation of a networking layer 2 topology. It also introduces Port Status Monitoring which performs active monitoring on ports of switches.

NetView will deliver a number of events from both IP Layer 3 and switch port Layer 2, with ITSA integrated. NetView can selectively forward events to a Tivoli Enterprise Console (TEC) for further correlation with other system, application and middleware events. This paper also looks at the out-of-the-box correlation and root-cause-analysis that TEC provides for networking events.

1 Introduction

IBM Tivoli Switch Analyzer (ITSA) is a product that is installed on the same machine with a distributed version of IBM Tivoli NetView; this includes AIX, Solaris, Linux and Windows operating systems. NetView provides full network management at the IP level, or layer 3 in a networking stack. NetView alone can detect and manage switches to the extent that a switch can be pinged and, if it supports SNMP, information can be obtained about the switch.

The Web Console of NetView offers reports on switches that support the Bridge MIB (RFC 1493) to the extent that reports can be displayed showing what NetView-discovered devices are attached to what ports on what switches; however there is no graphical display and there is no concept of monitoring the switch ports themselves, as opposed to monitoring the devices connected to those ports.

This paper uses a mixture of real, physical networking devices and emulated, virtual devices to explore a number of scenarios involving ITSA. The physical switches are Cisco 1900 and 2900XL devices. The emulated network is provided by **raddle**, an open-source network emulator that is available from http://sourceforge.net/projects/raddle/ . The emulated network comprises 3 Cisco routers, 2 Cisco switches and a number of networks and nodes, some of which support SNMP.

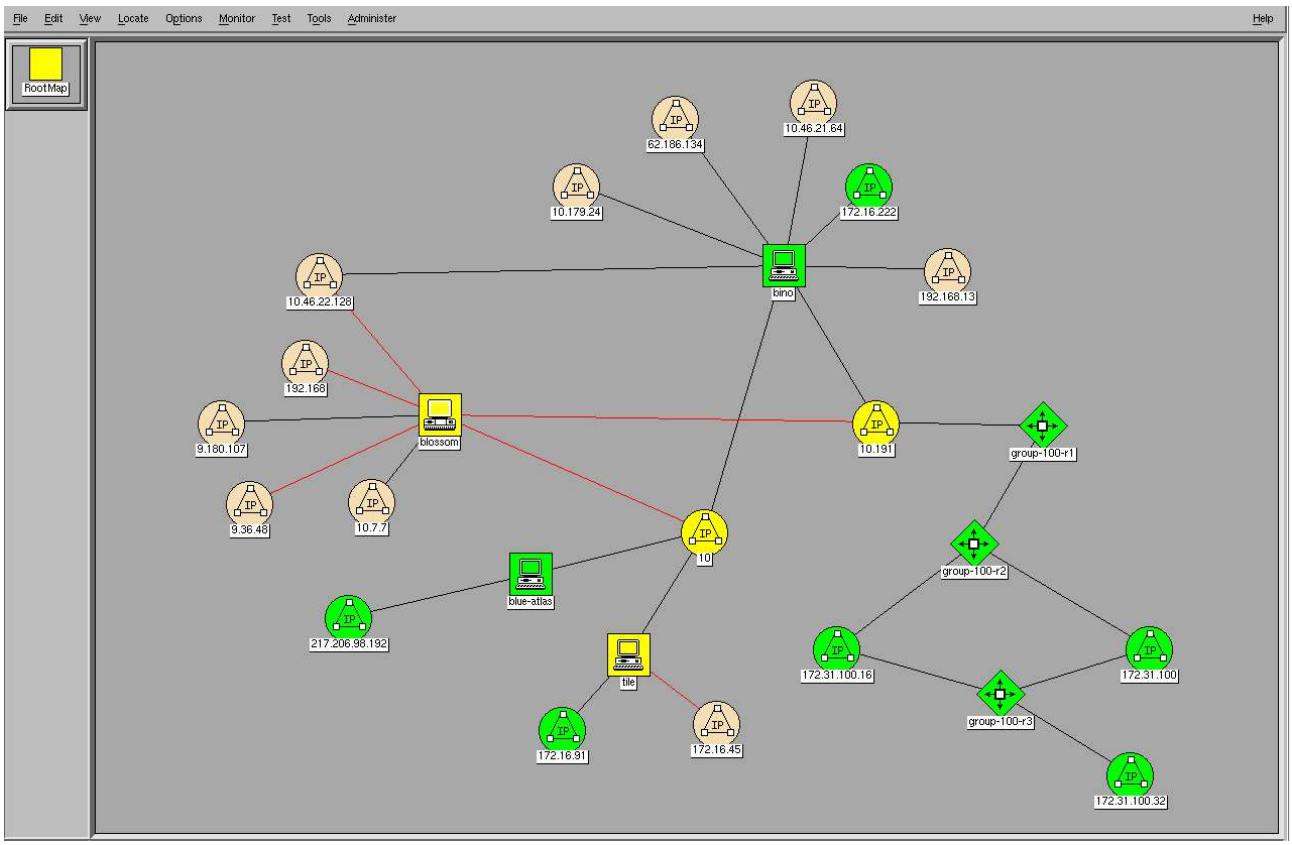

Figure 1 IP network discovered by NetView

The bottom right-hand part of figure 1, from the group-100-r1 router, is raddle emulated network. Three real switches, ***switch***, ***switch2*** and ***switch3***, exist as part of network 10. Two emulated switches, group-100-s1 and group-100-s2, exist in the 172.31.100.32 and 172.31.100.16 networks, respectively. The environment runs with Domain Name Server (DNS) active so real devices will be in the skills-1st.co.uk domain and emulated devices will be in the class.example.org domain. All **Selection Name** fields in the NetView object database will be fully-qualified domain names.

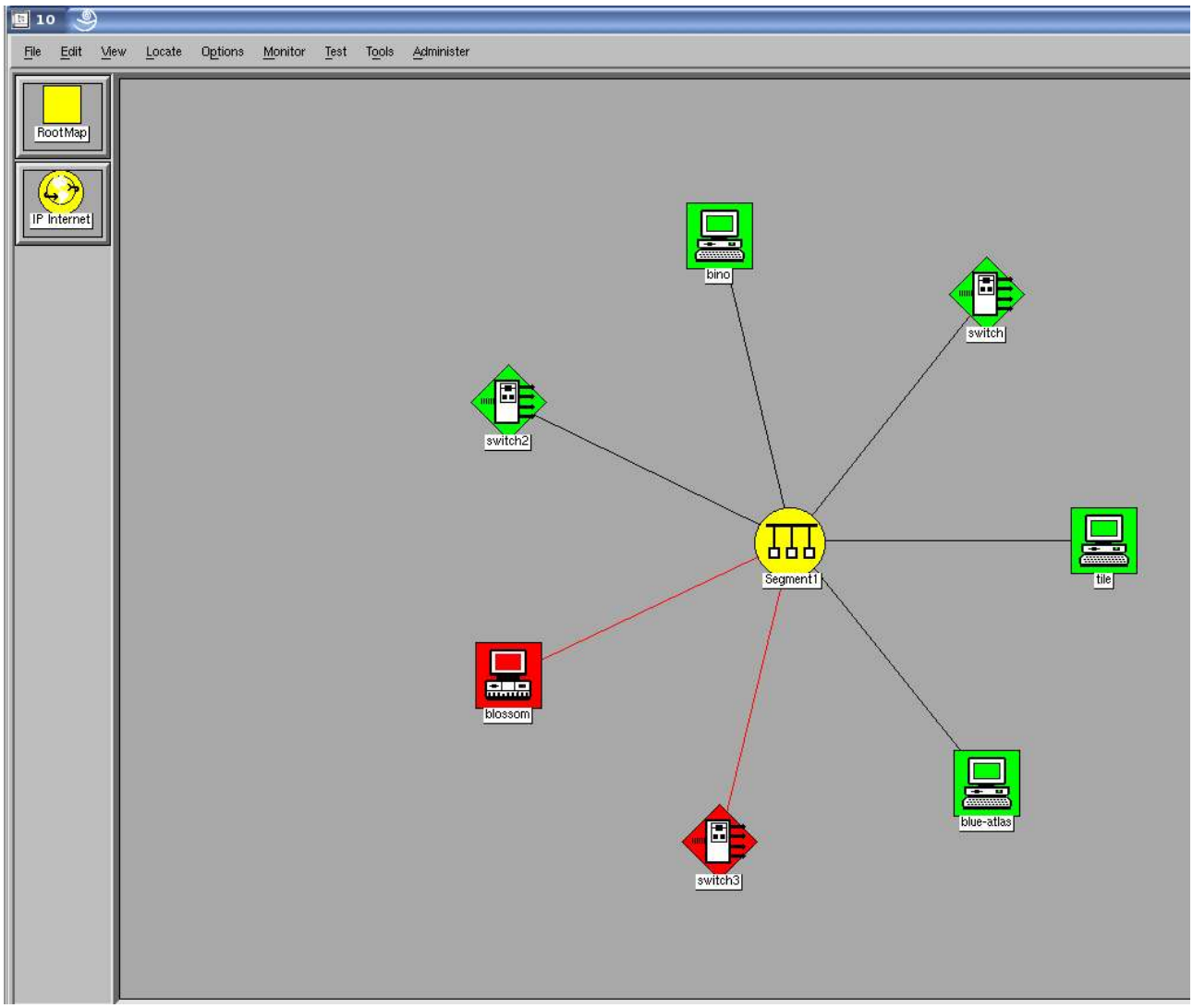

Figure 2 Network 10 showing switch, switch2 and switch3

The network management software used in this paper is NetView 7.1.4 Fixpack3 with ITSA 1.3, hosted on a SuSE 9.1 Professional system. Tivoli Enterprise Console (TEC) is also installed at version 3.9 Fixpack 2. This system is itself a virtual machine running under VMware Workstation 4.5.

2 IBM Tivoli Switch Analyzer 1.3 functionality

ITSA 1.3 was released in March 2005. It has two very significant enhancements over version 1.2.1:

- Graphical display of layer 2 topology
- Active monitoring of ports on switches

2.1 Functionality already available from ITSA 1.2.1

Previous versions of ITSA offered the ability to discover the Layer 2 topology of a network, using SNMP to get Bridge MIB information from switches. A correlation function performs root-cause analysis to work out whether a problem is indeed at Layer 2 or at Layer 3. If the problem is at Layer 2, ITSA can evaluate, amongst cascaded switches, where the real root-cause lies.

2.1.1 ITSA Correlator functionality

The correlator process is the heart of ITSA 1.2.1 problem analysis. **It is triggered by events from Layer 3 NetView**. It is important to understand that the correlator functionality is not proactive – it depends on base NetView to detect nodes and interfaces down. ITSA receives events directly from NetView's trapd daemon so immediately starts its own polling and correlation process of the Layer 2 network, once triggered by an event from NetView. This is also referred to as ITSA's **passive** mode of fault detection.

2.1.2 Reports and actions

ITSA 1.2.1 provides a number of reports and actions available from the NetView GUI (both native console and web console) and from a command line.

- Rediscover — Forces ITSA to perform rediscovery of Layer 2 for selected switch
- Discovery — Display table of ports on switch with devices connected to them
- Impact Analysis — Displays all nodes affected if selected switch goes down
- Impact Analysis (Connectors) — Shows routers and switches affected by outage of switch

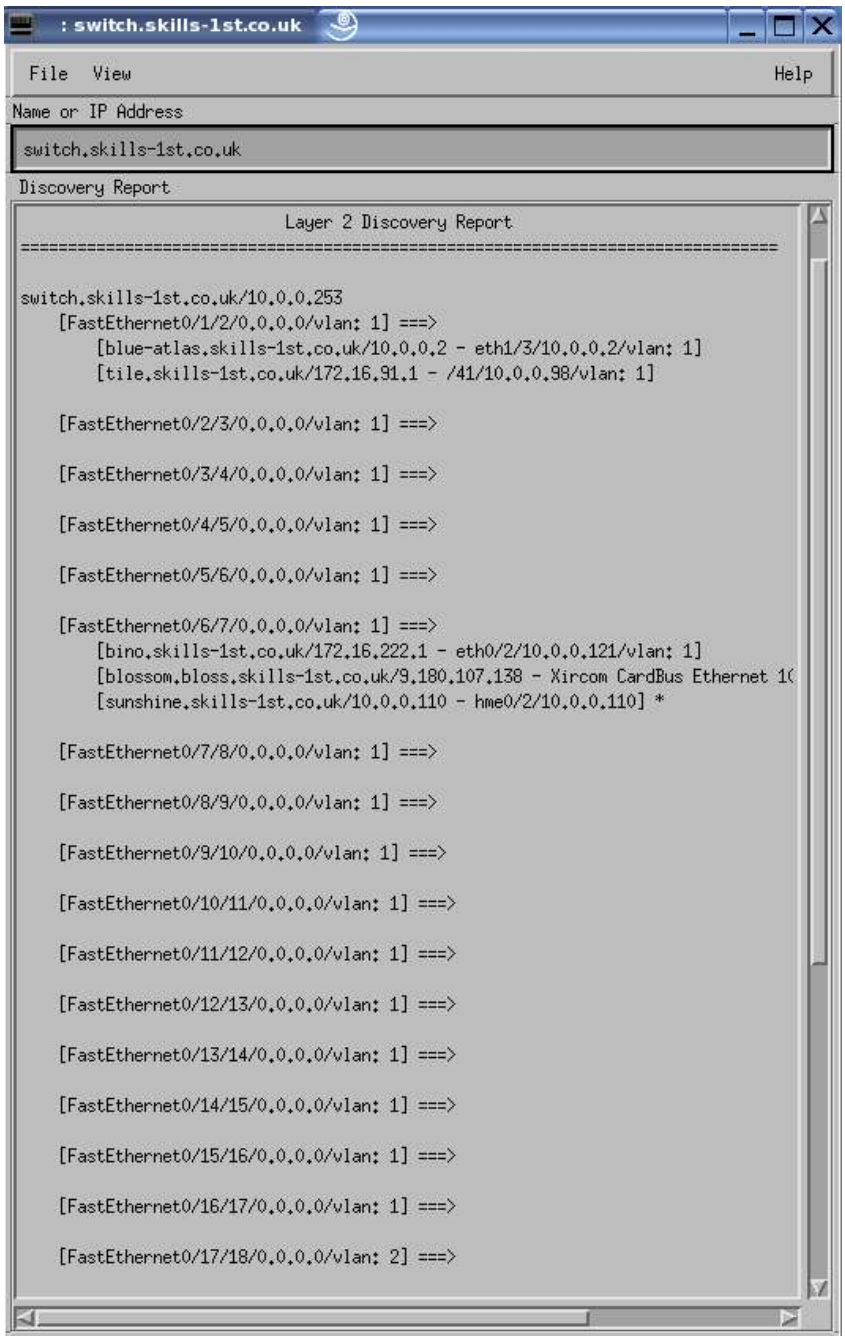

Figure 3 Discovery report for switch

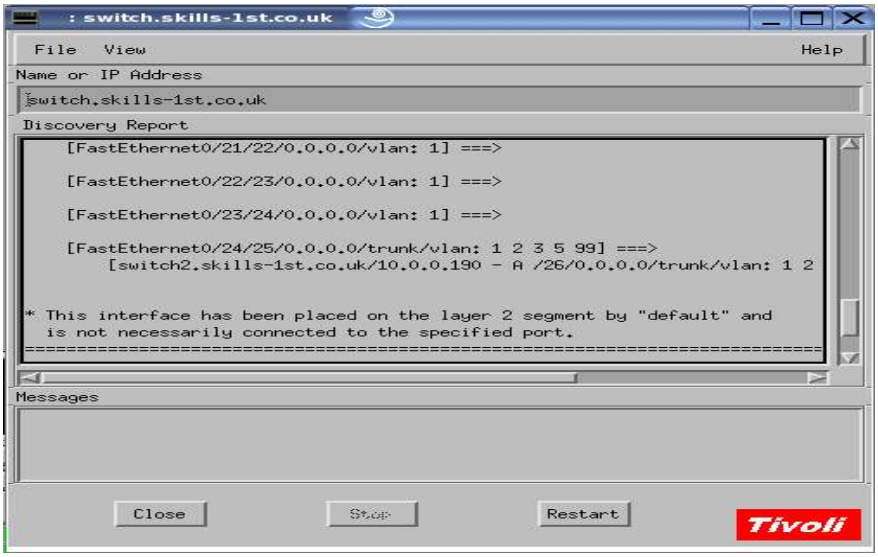

Figure 4 Discover report for switch (bottom half)

Note in the discovery report that there may be several devices apparently attached to one port. Port 1 is actually attached to a non-managed wireless router which supports ***blue-atlas*** and tile; thus port 1 on the switch sees traffic for both these devices.

Port 6 is actually attached to ***bino***.skills-1st.co.uk. ***blossom*** and sunshine both have asterisks against them indicating that the ITSA discovery algorithm cannot determine exactly which port the devices are attached to currently. This is because these two nodes are currently down.

Note that in the discovery report, there is nothing connected to port 16. There is a device physically connected to this port but it is not known to NetView.

Note that port 17 shows correctly that it is part of vlan 2 whereas the rest of the ports are connected to vlan 1.

Port 24 shows that it is a trunk and is part of vlans 1, 2, 3, 5 and 99. ***switch2*** is cascaded from ***switch***.

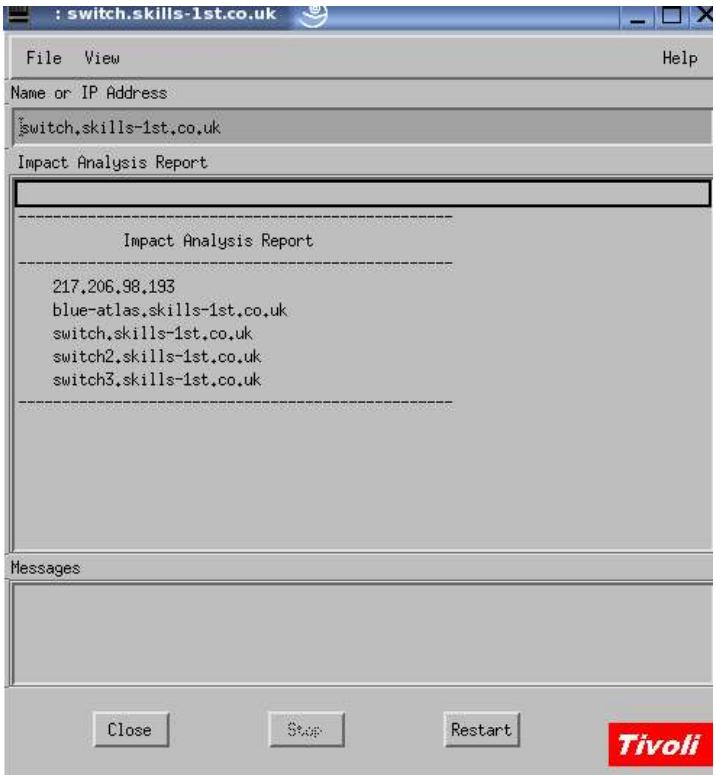

Figure 5 Impact report for switch

Note that the impact report not only includes boxes that directly attach to the switch; it also includes the node 217.206.98.193 which is routed to by ***blue-atlas***.skills-1st.co.uk. ITSA has determined that the loss of ***blue-atlas*** will result in the loss of connectivity to 217.206.98.193.

Note that there appears to be a problem with this report. ***bino*** is also directly connected to this switch and shows in the discovery report and in the physical view GUI. It would seem that this device should also show in the impact analysis; however the NetView server, ***tino***, can reach ***bino*** without going via the switch (***tino*** is actually a virtual machine on the physical machine called ***bino***). Thus ITSA has decided that there is a redundant path to ***bino*** and hence does not include it in the impact analysis for ***switch***.

2.2 New functionality in ITSA 1.3

ITSA 1.3 introduces two new significant pieces of functionality:

- Graphical representation of Layer 2 topology
- Active monitoring of switch ports with Port Status Monitoring (PSM)

The new GUI Layer 2 topology displays can only be seen from the NetView Web Console, not from the native console.

2.2.1 Physical View of Layer 2 topology

The Physical View shows which devices are connected to a switch as in Figure 6. ***switch*** is connected to ***switch2***, ***bino*** and ***blue-atlas***.

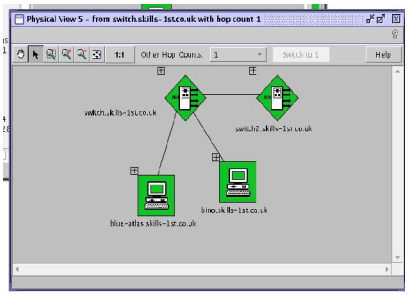

Figure 6 Physical View of switch

To see which ports devices are connected to, click on the "+" symbol in the top left corner of the switch – see Figure 7.

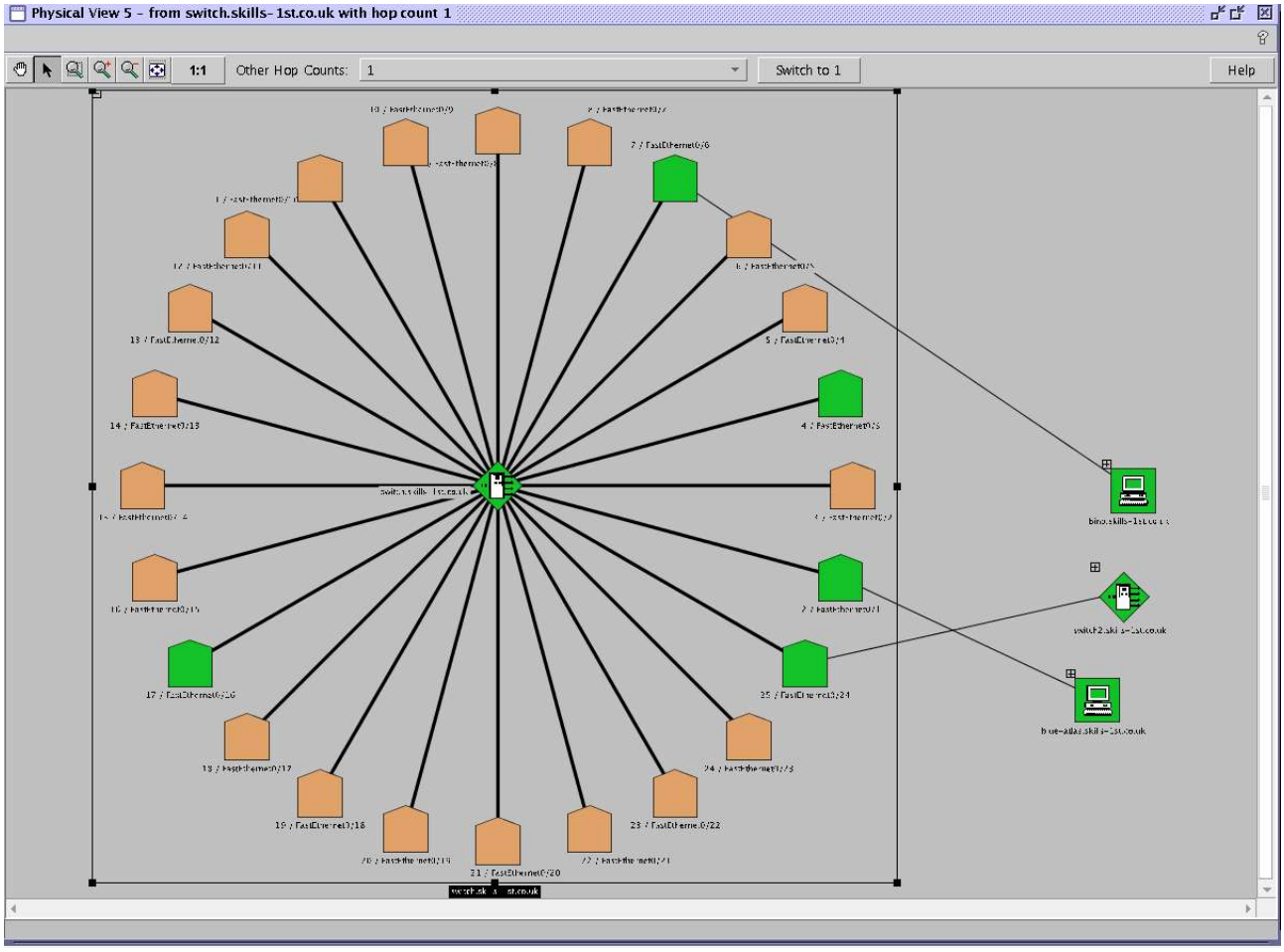

Figure 7 Physical view of switch with port detail

Although it is often impossible to read labels under ports and nodes, the label will be displayed clearly if the mouse cursor is placed over the label. By default, all ports are displayed – if nothing is attached to them, they will show light brown, unmanaged. Note that port 16 shows green, rather than brown. This port has a device connected that is unknown to NetView.

Using the "Hop Count" box at the top of the display, the layer 2 connectivity can be shown at 1, 2, 3 and 4 hop counts away from the node selected. In our network, this is better demonstrated by looking at the emulated portion of the network. Figure 8 shows that group-100-s2 has router group-100-r2 and nodes group-100-b1 and b2 attached; similarly switch group-100-s1 has group-100-c1, c2 and c3 attached. The router group-100-r3 is attached to both switches.

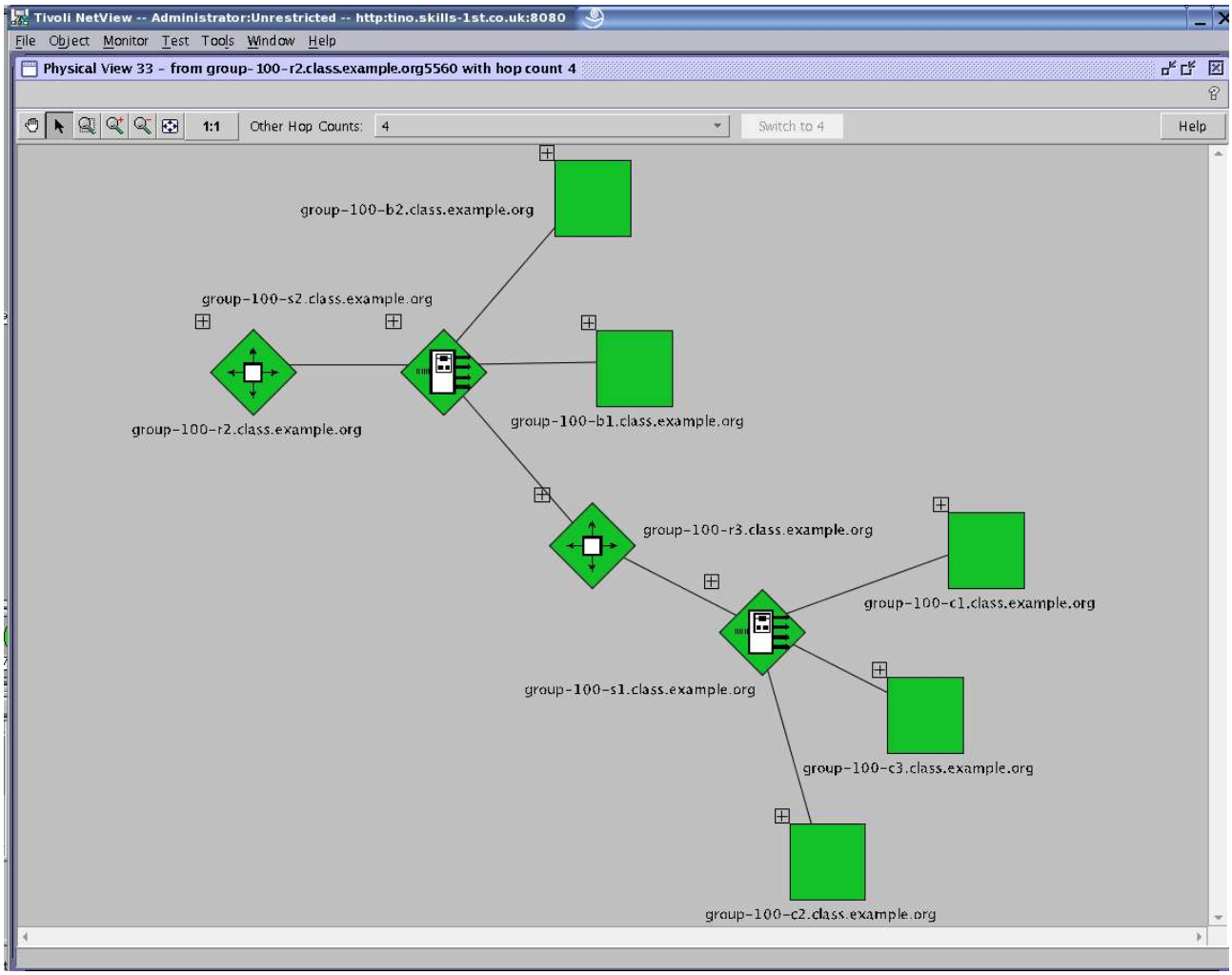

Figure 8 Physical View with Hop Count 4

To see what devices are connected to what ports, the "+" symbols can be explode the devices but this can rapidly become messy!

Note that, due to the emulation, these switches always report that something is physically connected, hence the ports are green here where as on the real ports shown for ***switch***, unconnected ports were light brown.

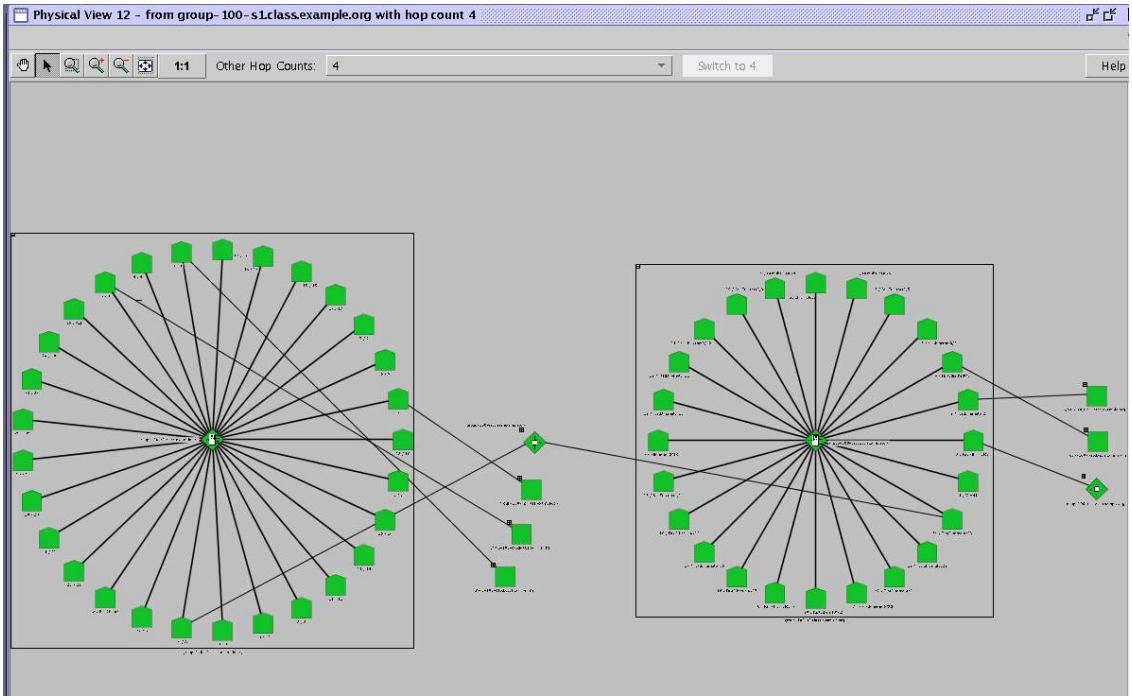

Figure 9 Physical View with Hop Count 4 and port details

2.2.2 Point-to-point View of Layer 2 topology

The Point-to-point view allows the user to select two nodes, one of which must support SNMP and to display the physical route between the two devices, through switches and routers. This is dependent on the routers supplying router table information via SNMP as well as the switches providing Bridge MIB Layer 2 connectivity information via SNMP. If one of the source / destination nodes does NOT support SNMP, the route will be shown from the nodes that DOES support SNMP.

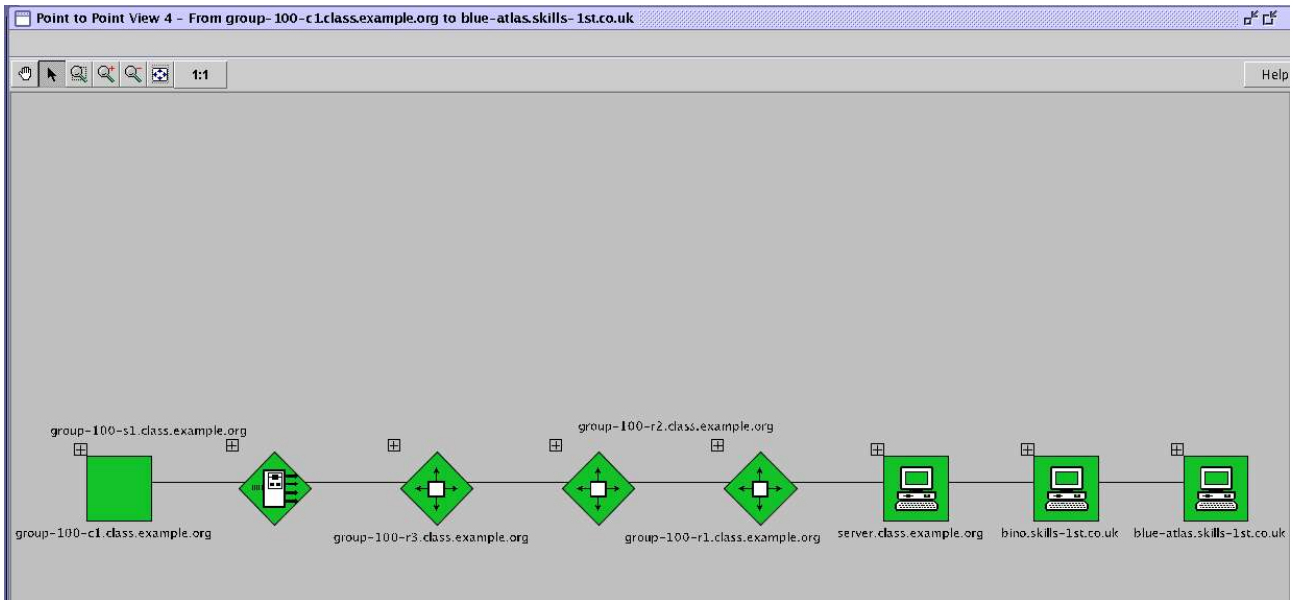

Figure 10 Point-to-point view from group-100-c1 to blue-atlas

Note that I think this should include ***switch*** between ***bino*** and ***blue-atlas***.

2.2.3 VLAN View of Layer 2 topology

The third new topology view shows, for a switch, which ports are allocated to which VLANs.

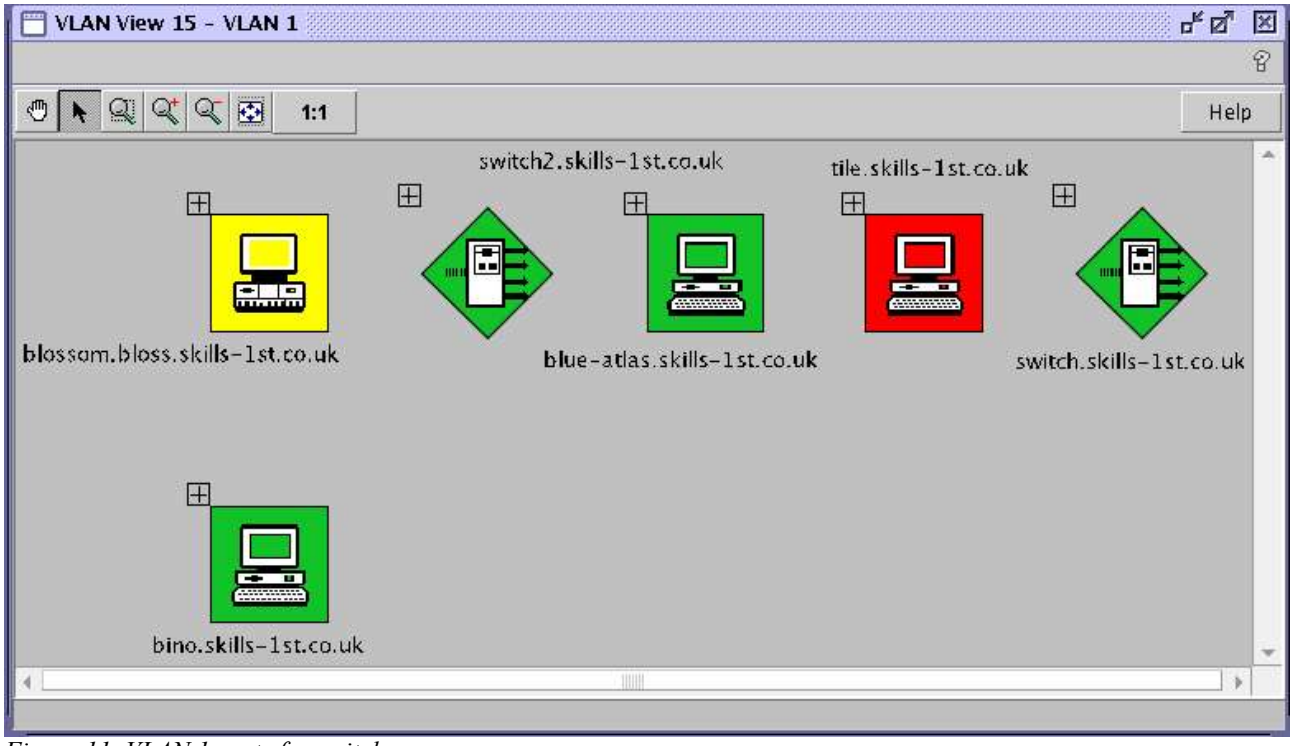

Figure 11 VLAN 1 ports for switch

Note that this VLAN view still appears to show ***blossom*** and tile which are, in fact down. Some ITSA displays and reports seem to maintain history of devices connected to switch ports, even though a Rediscover is initiated for a switch. There is a parameter in / usr/OV/ITSL2/conf/l2_topo_adapter.ini, **discovery_interval**, which is 24 hours by default. Any devices that have been down beyond this discovery interval do finally seem to get cleaned away.

Exploding switch devices using the "+" symbol shows the ports for this VLAN and also provides the opportunity to look at other VLANs.

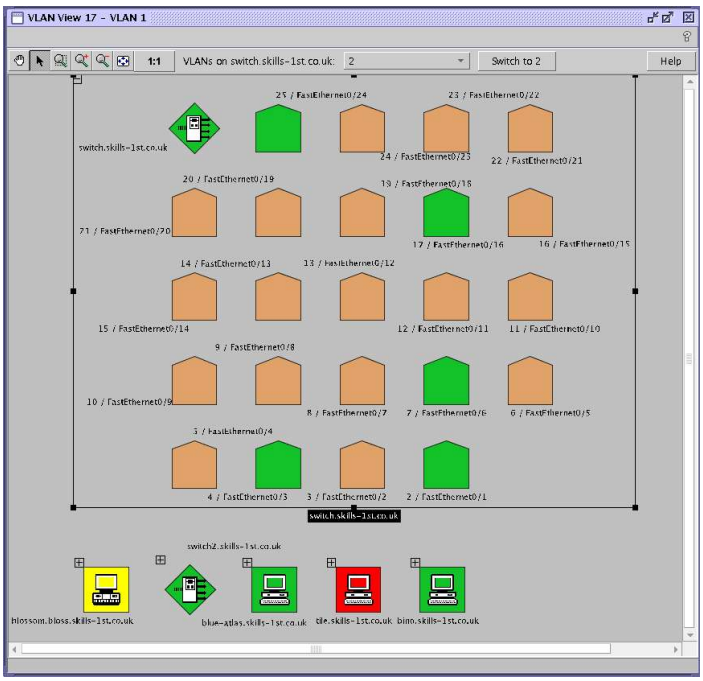

Figure 12 VLAN 1 ports for switch

2.2.4 Layer2Status icons in NetView nodes

When ITSA 1.3 is installed on a NetView system, switch devices have an extra icon created inside the Node submap (alongside the interface icon). This icon will initially be **Unset** (blue) but will change colour to red, green or yellow depending on whether the overall Layer 2 status of the switch is Critical, Normal or Marginal respectively.

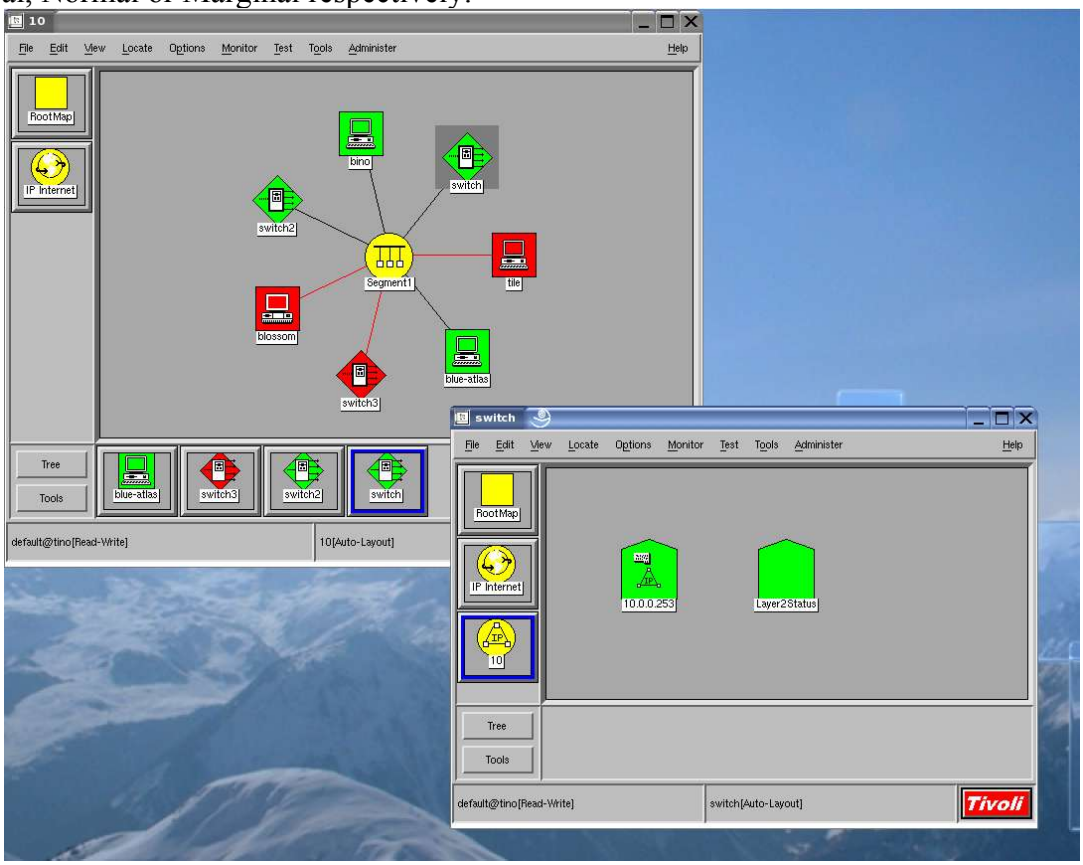

Figure 13 Layer2Status icon for node switch

2.2.5 Monitoring switch ports with Port Status Monitoring (PSM)

ITSA 1.2.1 could only perform "passive" monitoring, triggered by Layer 3 Node Down or Interface Down events from NetView. Once these events are received by ITSA, it will start its own polling to determine the Layer 2 status and the root cause of any problem.

Port Status Monitoring (PSM) is new with ITSA 1.3 and allows ITSA to **actively** monitor ports on switches, regardless of whether NetView has discovered the device attached to a particular port. By default, all ports are managed on all switches that have been discovered by NetView and hence are managed by ITSA.

The Status Report has also been introduced which displays status of switches **both** from the passive polling perspective of the correlator **and** from the active polling perspective of PSM.

Note in the status report for ***switch*** in Figure 14, that ports 1, 6, 16 and 24 show "Interface Up" and a "Correlated Up" status. The previous discovery report shown in Figures 3 and 4 did **not** show anything connected on port 16. This demonstrates the benefit of the active PSM which, by default, will monitor *all* ports, regardless of whether NetView has discovered the node attached to a particular port. The anomaly between status and discovery reports is caused by a device which is physically connected to the switch but it is not known to NetView.

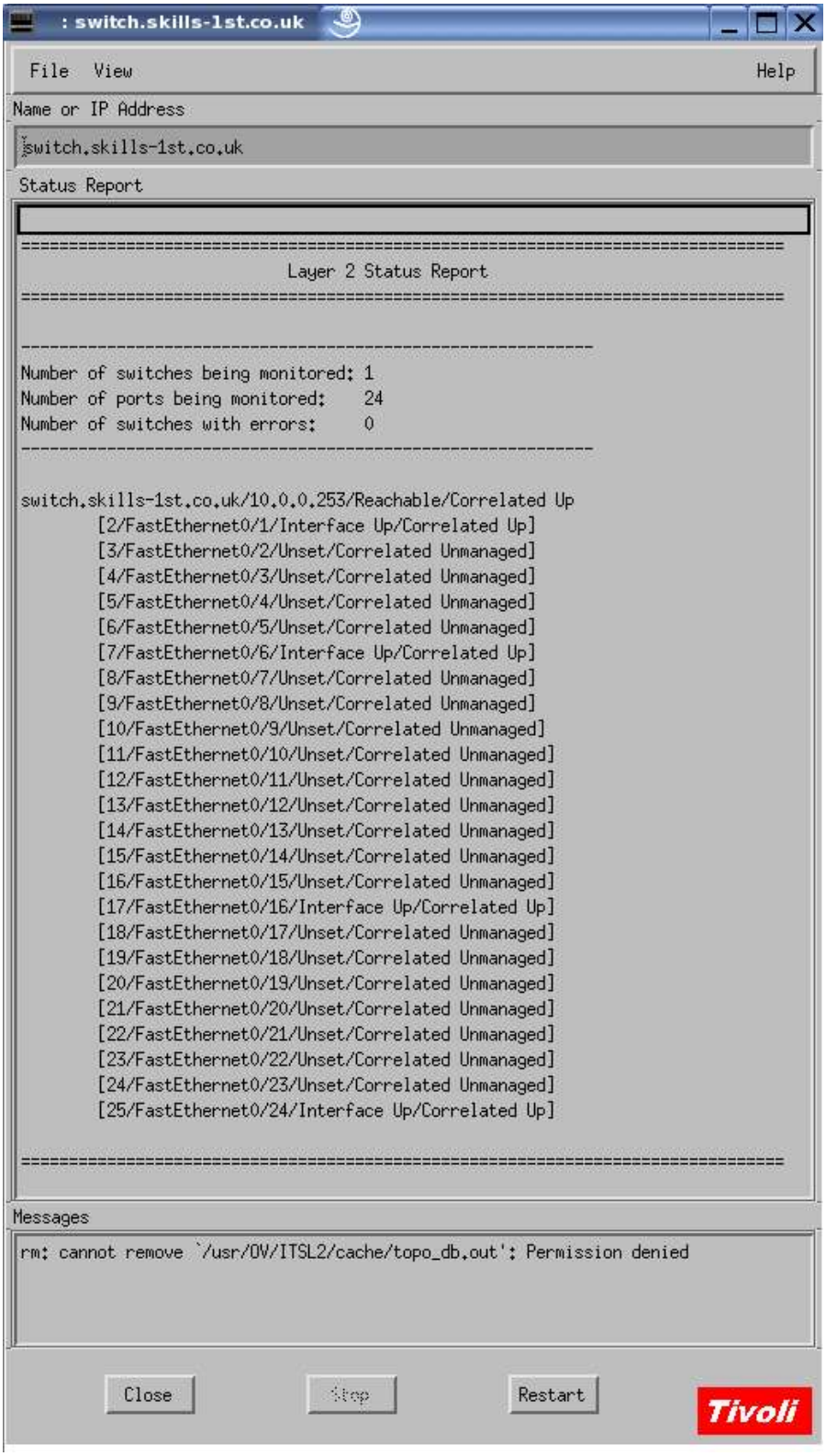

Figure 14 Status report for switch

The fields of this report are:

<interface index> / <interface type> / <port number> / <PSM status> / <correlator status>

3 Configuring ITSA

There are a number of parameters in a number of ITSA configuration files that help control exactly what is monitored.

3.1 Controlling Port Status Monitoring (PSM)

There are 2 main parameters that control the behaviour of PSM:

- /usr/OV/ITSL2/conf/files/l2_polling.cfg - the **Manage** and **Ports** fields
- /usr/OV/ITSL2/conf/correlator.ini - the **poll_all_ports** field

Note that there is **no**, repeat **no** way currently to uniquely specify polling on or off purely on a per-port basis.

3.1.1 /usr/OV/ITSL2/conf/files/l2_polling.cfg

This file specifies which switches should be polled by the active PSM polling mechanism; it is known as the **switch table**. By default, it has a single uncommented line which specifies all ports on all NetView-discovered switches, should be PSM polled. Entries can be added to this file to specify different polling characteristics either by IP address, host name or SNMP OID. These specifications can include wildcards.

Each switch table entry has eight fields separated by vertical bar characters (|):

type|description|layer_2|OID|IP_address|hostname|manage|ports|

The default entry is:

type	description	layer2	OID	IP address	hostname	manage	ports
switch	monitor all layer 2 switches	Y	*	*	*	Y	A

The **manage** field specifies whether these switches should be PSM-polled at all.

The **ports** field specifies whether all ports should be polled (**A**). If this field is set to **C** then **only** ports connected to devices that have been discovered by NetView at Layer 3 will be polled.

Setting the ports field to **C** has a wider connotation than Port Status Monitoring. A Physical View will only show ports that are connected to NetView-discovered nodes (none of those light-brown, unmanaged ports in the exploded Physical View). A Status Report and Discovery Report will only show ports that are connected to NetView-discovered nodes. Ports that **are** physically connected to devices but those devices have not been discovered by NetView, will not be shown anywhere or managed at all.

Note that to be managed by ITSA, the ports must only be connected to NetView Layer 3 **discovered** nodes; ports will still be managed by ITSA if the end nodes have been unmanaged by NetView at Layer 3.

Note that if this file is changed, the itsl2 daemon must be recycled with ovstop / ovstart .

3.1.2 /usr/OV/ITSL2/conf/correlator.ini – poll_all_ports field

There are a large number of parameters in the correlator.ini file, many of which are inter-related. Some of them will be discussed later. The **poll_all_ports** field comes under the PortStatus category and by default, is set to **y**. The ITSA 1.3 User Guide has the following description of poll_all_ports:

To specify whether all configured ports managed by the port status monitor are polled during each polling cycle, modify the poll_all_ports option in the /usr/OV/ITSL2/conf/correlator.ini file. This option can be used to optimize port status monitoring in large network environments. If this option is set to n, the port status monitor polls only those ports for which it cannot rely upon the correlation process to detect outages. This would include the following:

- Any port that is not connected to a managed node. No Tivoli NetView event would result from a layer 2 outage that does not affect a managed node.

- Any port that is part of a redundant path to a managed node. No Tivoli NetView event would result if the connected node does not become unreachable.

Setting poll_all_ports to n can help to reduce excessive network traffic in large environments. However, this optimization should be used with caution, because it relies upon the accuracy of the discovered layer 2 topology information. By default, this option is set to y (all ports are polled in each polling cycle)

This description raises a question:

- Define a *configured port* ? In practise, this means any port as specified by the A/C value of the Ports field in l2_polling.cfg.

A value of **n** in the poll_all_ports field **prevents** PSM polling of any port to which a NetView Layer 3 **discovered AND managed** node is attached. This means that the following ports **will** be status polled by PSM:

- Switch ports in a redundant layer 2 path
- Switch ports on switches that are in a remote campus
- Switch ports where the connected device (downstream) to that port is unmanaged by NetView Layer 3
- Switch ports connected to devices undiscovered by NetView Layer 3

A value of **n** in the poll_all_ports field does **not** mean that any ports are totally unmonitored by ITSA.

- Ports connected to devices not discovered by NetView will be actively monitored by PSM.
- Ports connected to devices that NetView has discovered but unmanaged will also be actively monitored by PSM.
- Ports connected to devices that NetView has discovered and is managing at Layer 3, will be passively monitored. This means that an event will be required from NetView Layer 3 (a Node Down or Interface Down from the netmon source (N)) to trigger the ITSA passive **correlator** polling at Layer 2 for those ports that are connected to devices for which a layer 3 event has been received.

In other words, you can turn PSM off for ports where ITSA knows NetView can trigger the outage detection.

3.2 Scenarios with l2_polling.cfg and poll_all_ports

To better understand the relationship between these two customisation features, here are some scenarios and outcomes. A single switch, ***switch*.skills-1st.co.uk**, is used for testing. It has attached:

- At least 1 NetView discovered, NetView Layer 3 Managed device

- At least 1 NetView discovered, NetView Layer 3 Unmanaged device
- At least 1 active device, totally undiscovered by NetView Layer 3

3.2.1 Default settings for l2_polling.cfg and poll_all_ports

The starting point is the default settings, that is:

- /usr/OV/ITSL2/conf/files/l2_polling.cfg **ports** field set to **A**
- /usr/OV/ITSL2/conf/correlator.ini **poll_all_ports** field set to **yes**

Ensure that the itsl2 daemon is stopped and restarted to pick up configuration changes to these files and to perform rediscovery of Layer 2.

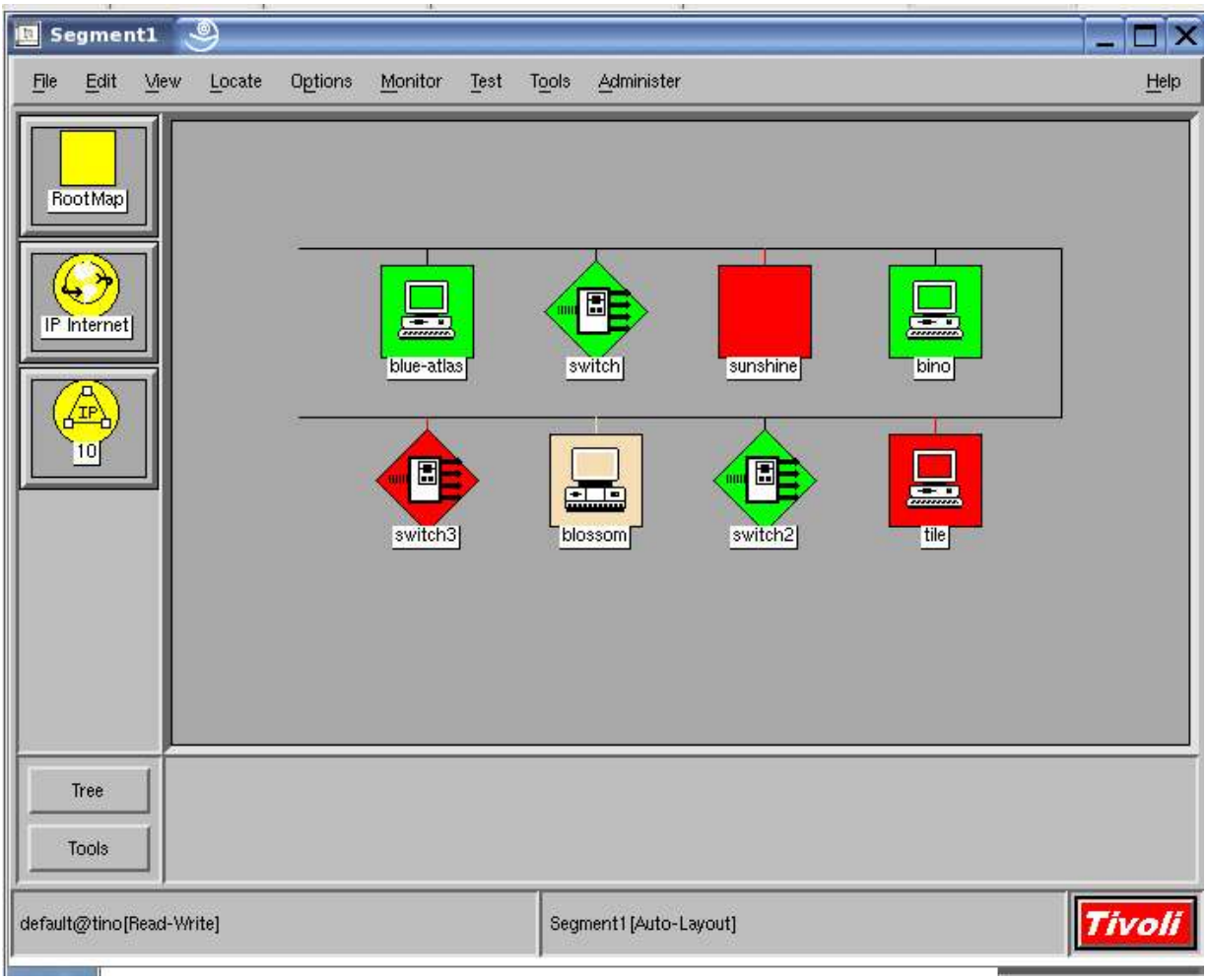

Figure 15 Devices on network 10, some connected to switch

In Figure 15 above, ***blossom*** is unmanaged by NetView (light brown); ***bino***, ***blue-atlas*** and ***switch2*** are managed by NetView; all four are connected to ***switch***. A fifth device (***poppet***) is plugged into ***switch*** but it is not discoverable by NetView.

Figure 16 Physical view of switch

Figure 16 shows the Physical View of ***switch***. The connected devices on the right, from top to bottom, are ***blossom***, ***switch2***, ***bino*** and ***blue-atlas***. Note that ***blossom*** has a status of Normal (green) on this display because it is managed by Port Status Monitoring and is connected at Layer 2.

Note also that port 16 (centre bottom port) is green but shows nothing connected. This port is physically connected but the attached device, ***poppet***, is totally undiscovered by NetView at Layer 3.

The status for ITSA views comes from ITSA's view of the world, not NetView's. They both have different status polling engines and may be momentarily out of sync. Notably in the ITSA Physical View when a port is down, ITSA will apply red to all the downstream nodes on non-redundant paths.

For easier reference, here is a table showing the port numbers, attached devices and their status for the device called ***switch***:

Port	***Attached Node***	***Node Status***
1	blue-atlas	Discovered & managed by NetView L3
6	bino	Discovered & managed by NetView L3 (NetView system)
9	blossom	Discovered & unmanaged by NetView L3
16	poppet	Undiscovered by NetView L3
17	nothing attached	Configured as part of VLAN2

Port	*Attached Node*	*Node Status*
25	switch2	Discovered & managed by NetView L3 – port defined as a trunk

If the device on port 16 is taken down and ***blossom*** is disconnected from the switch, the following Physical View results.

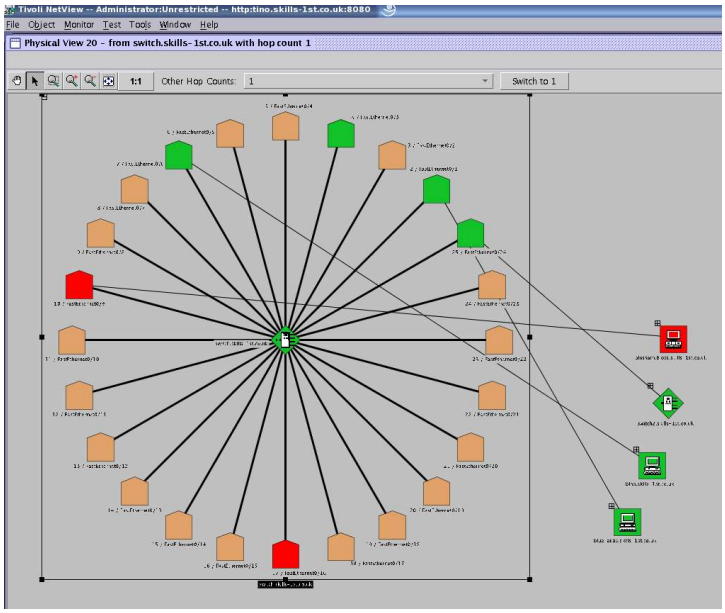

Figure 17 switch with 2 ports in Layer 2 fault condition

Note that events on port 9 (connected to ***blossom***, which is unmanaged at NetView Layer 3) nor port 16 (connected to the undiscovered ***poppet***), will have events generated by NetView's netmon. Problems with these ports can **only** be detected by ITSA's Port Status Monitoring (PSM).

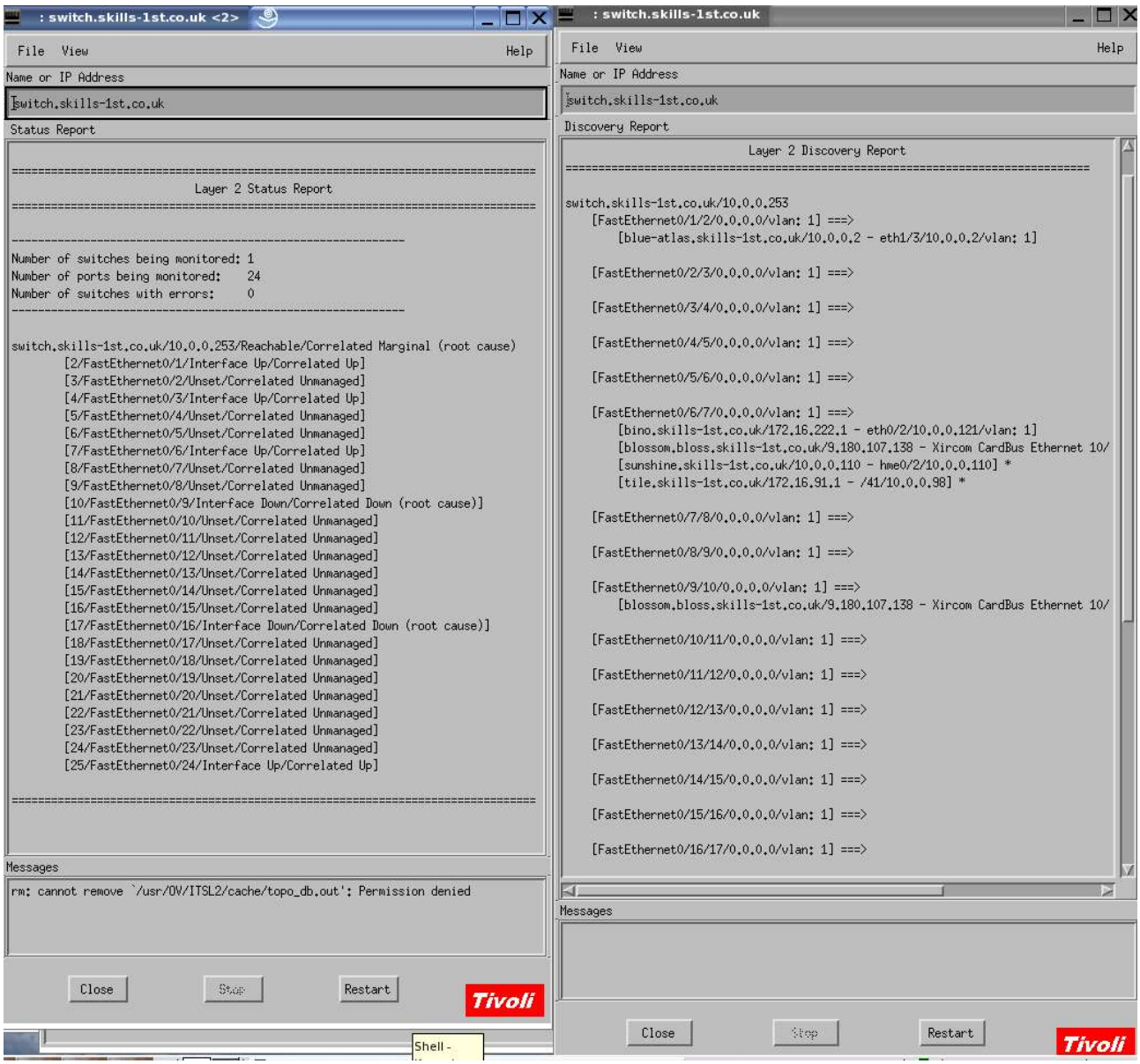

Figure 18 Status and Discovery Reports during 2 ports disconnected

Note in Figure 18 that the Discovery Report has nothing on port 16 but the Status Report has PSM status of Interface Down and Correlated Status of Correlated Down (root cause).

Figure 19 shows the sequence of events that arrive in the NetView event log.

3.2.1.1 Events generated by ITSA

Events will appear in the NetView event log for both Layer 3 events and Layer 2 events. NetView's netmon daemon detects layer 3 outages and events will show as being from the source netmon (N). Events generated by ITSA will have the vendor source character (V).

Remember that ITSA has two ways of detecting problems; PSM **actively** monitors ports (all ports by default) and generates events when a problem is seen. The **passive** correlator process is triggered by events from Layer 3 NetView and it then starts its own polling and correlation process

of the Layer 2 network, once triggered by an event from NetView. Both types of event generated by ITSA will come from the V source.

For the scenario in Figure 17 where there are issues with two nodes, neither managed by NetView Layer 3, events will only be generated by the active PSM polling. The PSM polling interval is controlled by the following parameters defined in **/usr/OV/ITSL2/conf/l2_event_adapter.ini** :

- poll_cycle default is 300 seconds, I have it set to 120 seconds
- retry_cnt default is 3, I have not changed this

If the 2 nodes are disconnected in separate PSM polling intervals then each problem will result in the following event sequence for each node:

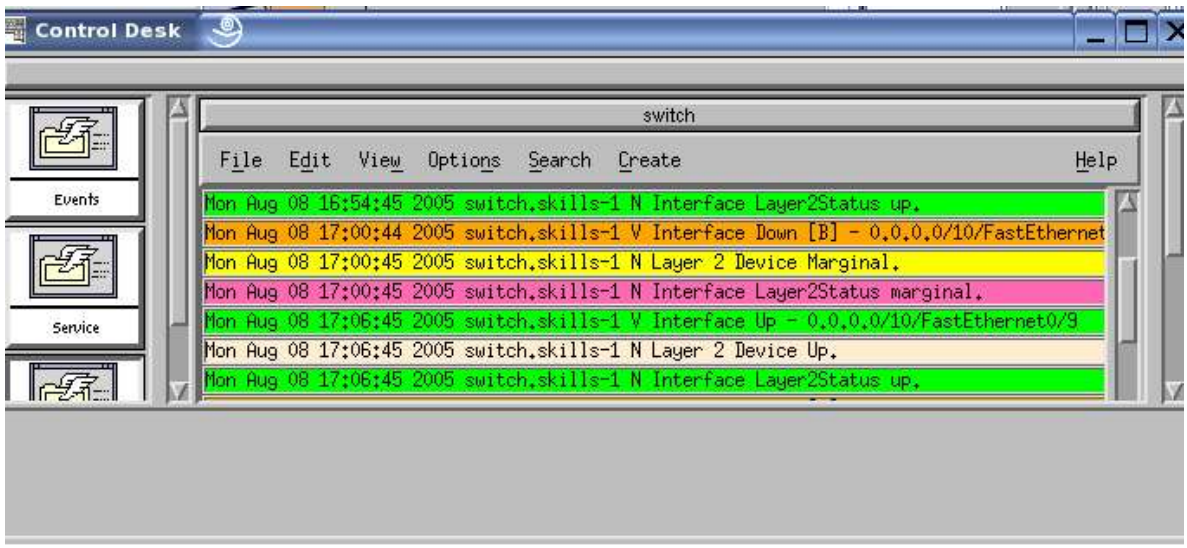

Figure 19 Events when single node disconnected

Note that the origin for the events is the switch, not the end node. The "V" interface down event reports on the **index** of the port that is down, which, in this case of these switches, is one higher than the actual port number. The events here are for the unmanaged ***blossom*** node which is physically connected to port 9 of ***switch***.

Remember that when ITSA is installed and switch nodes are discovered by NetView and ITSA, an extra icon is added to a switch at the node level, which represents Layer2Status. It is this "interface" icon that changes colour to marginal in the NetView Layer3 topology hierarchy, resulting in the switch node also changing to marginal. These node and interface changes are reported by netmon (N), as shown above.

A major possible confusion arises if more than one PSM problem is detected in the same polling interval. In the next test, the undiscovered ***poppet***, the unmanaged ***blossom*** and the managed ***switch2*** nodes were all disconnected from ***switch*** in quick succession. The following results were received:

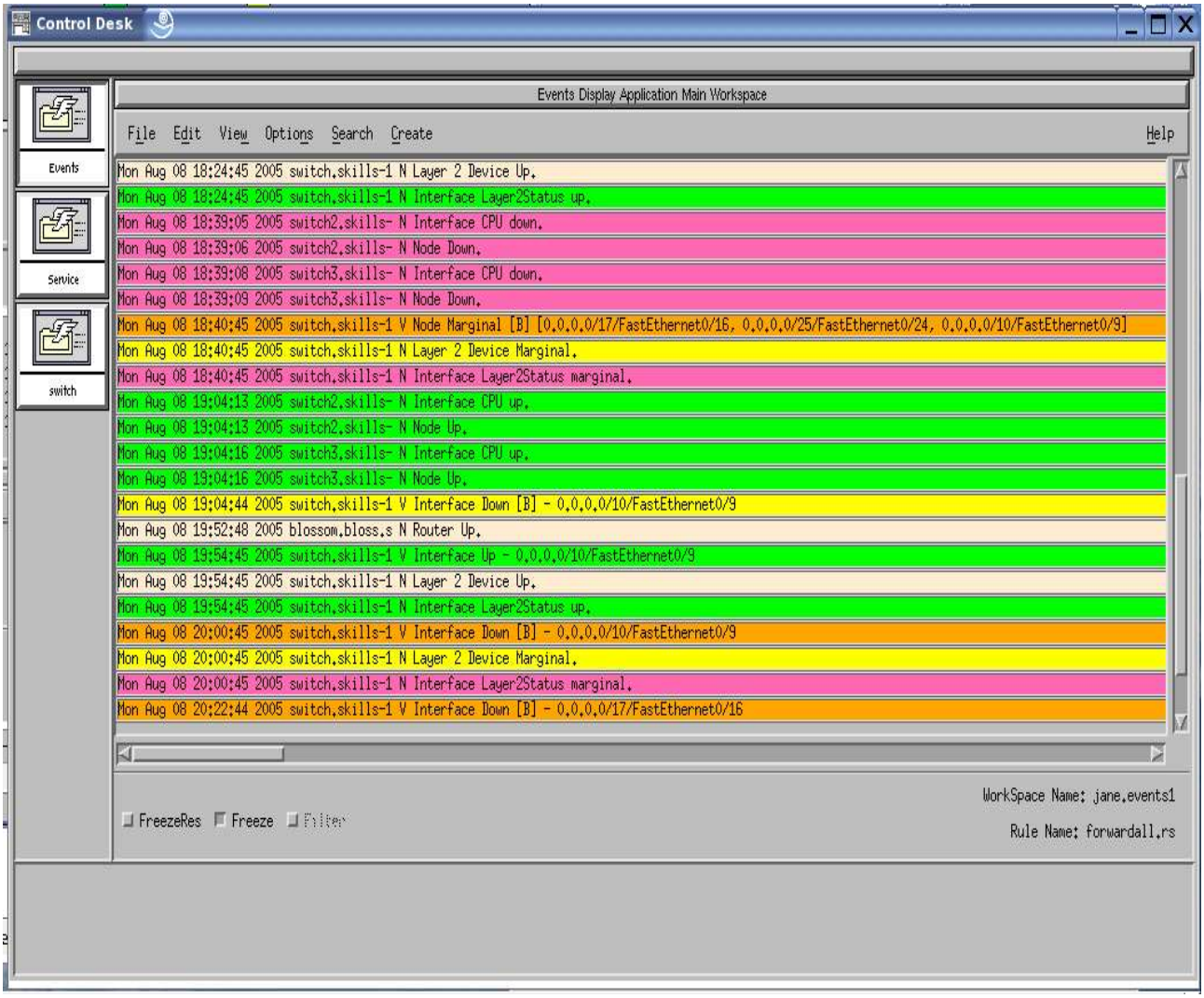

Figure 20 Events when 3 nodes disconnected simultaneously and reconnected separately

The first events detected were by netmon (N), noticing that ***switch2*** had gone. ***switch3*** is connected off ***switch2*** so that could not be pinged either, resulting in the node and interface down messages from ***switch3***.

About 90 seconds later, ITSA generated a "V" event with an origin of ***switch***. Rather than generating three Interface Down events, a single Node Marginal event is generated with the message varbind listing the disconnected interfaces. I find this is far less useful than having three separate interface events.

More confusingly, the nodes were brought back up separately, at greater than 2 minute intervals (the PSM polling interval). The undiscovered ***poppet*** was brought back first – no events were generated although a Status Report and a Physical View show the port active again. The second node to be reconnected was ***switch2***. This resulted in netmon (N) Layer 3 events when the netmon polling interval noticed both ***switch2*** and ***switch3*** were now pingable again. There is still no "V" event to do with either of these restored nodes. However, once the layer2 problems are reduced to a single remaining issue, a "V" Interface Down event is generated, originating from ***switch***, for the port index that is still down (***blossom*** on port 9, index 10). When ***blossom*** is reconnected, a "V" Interface Up event is generated.

Note in the event log shown above that the node called ***blossom*** was temporarily remanaged by NetView Layer3 which explains the (N) event from ***blossom*** of "Router Up".

3.2.2 l2_polling.cfg, ports=A and correlator.ini, poll_all_ports=no

This configuration has:

- /usr/OV/ITSL2/conf/files/l2_polling.cfg **ports** field set to **A**
- /usr/OV/ITSL2/conf/correlator.ini **poll_all_ports** field set to **no**

Ensure that the itsl2 daemon is stopped and restarted to pick up configuration changes to these files and to perform rediscovery of Layer 2.

The **l2_polling.cfg**, **ports=A** setting is the Port Status Monitoring (PSM) Switch Table and defines that (PSM) **is** active on all ports. From the point-of-view of the **poll_all_ports** parameter in correlator.ini, the **A** setting defines all ports to be "configured".

A value of **n** in the **poll_all_ports** field of correlator.ini **prevents** PSM polling of any "configured" port to which a NetView Layer 3 **discovered AND managed** node is attached. This means that the following ports **will** be status polled by PSM:

- Switch ports in a redundant layer 2 path
- Switch ports on switches that are in a remote campus
- Switch ports where the connected device (downstream) to that port is unmanaged by NetView Layer 3
- Switch ports connected to devices undiscovered by NetView Layer 3

Ports connected to devices that NetView has discovered and is managing at Layer 3, will be **passively** monitored. This means that an event will be required from NetView Layer 3 (a Node Down or Interface Down from the netmon source (N)) to trigger the ITSA passive correlator polling at Layer 2, for those ports that are connected to devices for which a layer 3 event has been received.

Discovery reports are exactly the same for this configuration as for the previous one and the Physical Views should also be exactly the same – it is only PSM polling that is affected by this change.

The Status reports differ slightly. The fourth field for interfaces on a Status Report is defined as showing the PSM status and the fifth field shows the correlated status. The correlated status should show the same values for identical situations, regardless of whether **poll_all_ports** is set to "yes" or "no"; the PSM status, however, will be "Unset" for ports whose end nodes are discovered and managed by NetView. The only ports in our scenario that should now be actively managed by PSM are ports 9 (with the unmanaged node ***blossom***) and 16 (with the undiscovered ***poppet***).

Do ensure that the itsl2 daemon has been recycled and that a rediscovery has been completed after changing configuration parameters and before running Discovery and Status reports. The parameter **topo_cache_freq** in correlator.ini determines how frequently the cache is updated from which the Status and Discovery reports are generated. By default, this is only updated every 900 seconds but I have modified this to 200 seconds. Even so, it often takes longer than this before changes are seen, especially to see the correct PSM polling status.

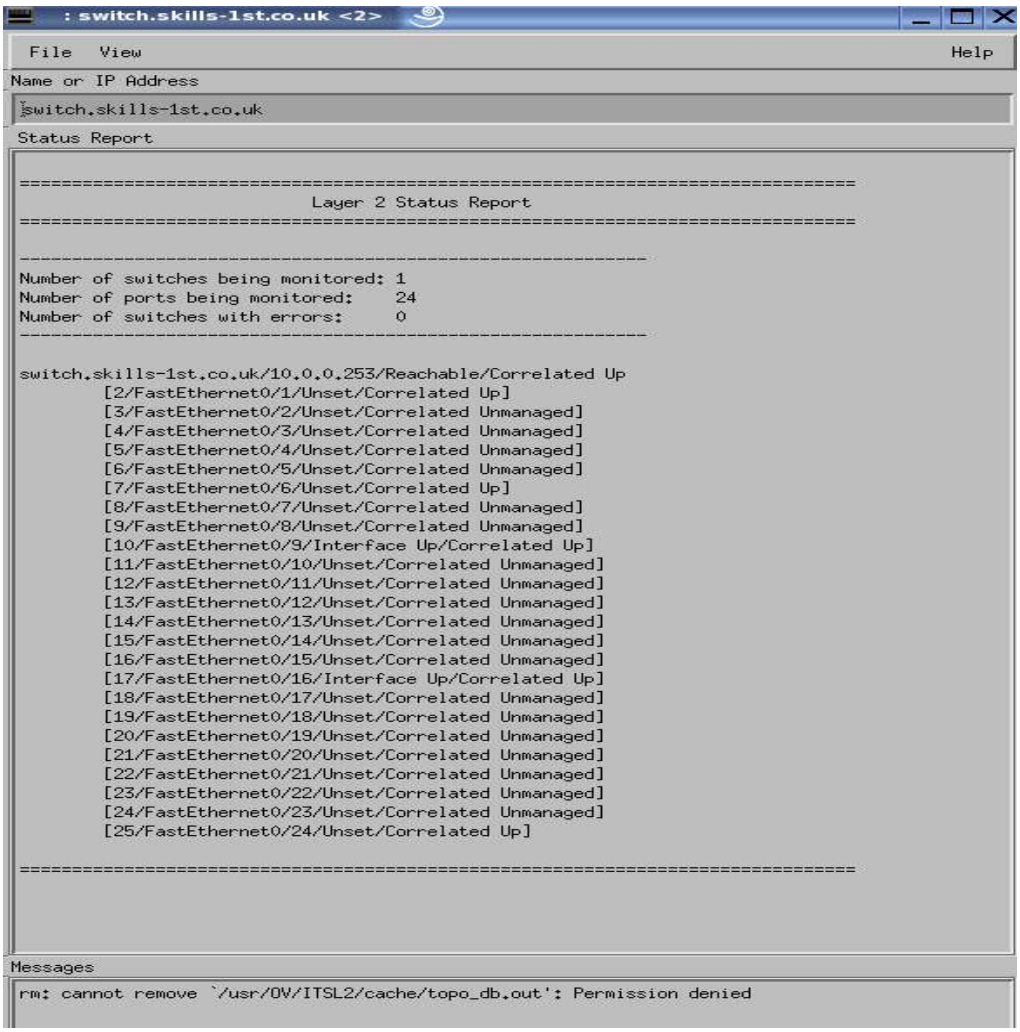

Figure 21 Status Report with poll_all_ports=no

3.2.2.1 Events generated for this scenario

Events appearing in the NetView event log when switch-attached nodes are lost, should be similar for this scenario as for the first one. If nodes are lost that are either NetView Unmanaged or NetView undiscovered then a “V” event will be generated by the PSM polling function of ITSA. If a NetView Layer 3 Managed node is lost, a netmon (N) event will be seen first, followed by an ITSA “V” event. Either way, the ITSA correlator function will be triggered to determine the root-cause of the problem. There is no way from the NetView Event Log to tell the difference between a “V” event generated by PSM and a “V” event generated by the correlator root-cause function.

The only difference between these 2 scenarios is that the first scenario may produce a “V” event followed by an “N” event, depending on the polling cycles of netmon and PSM respectively. This second scenario will never produce a “V” event first for NetView Managed nodes, as the event will be generated by a passive correlator poll which is triggered by a netmon event.

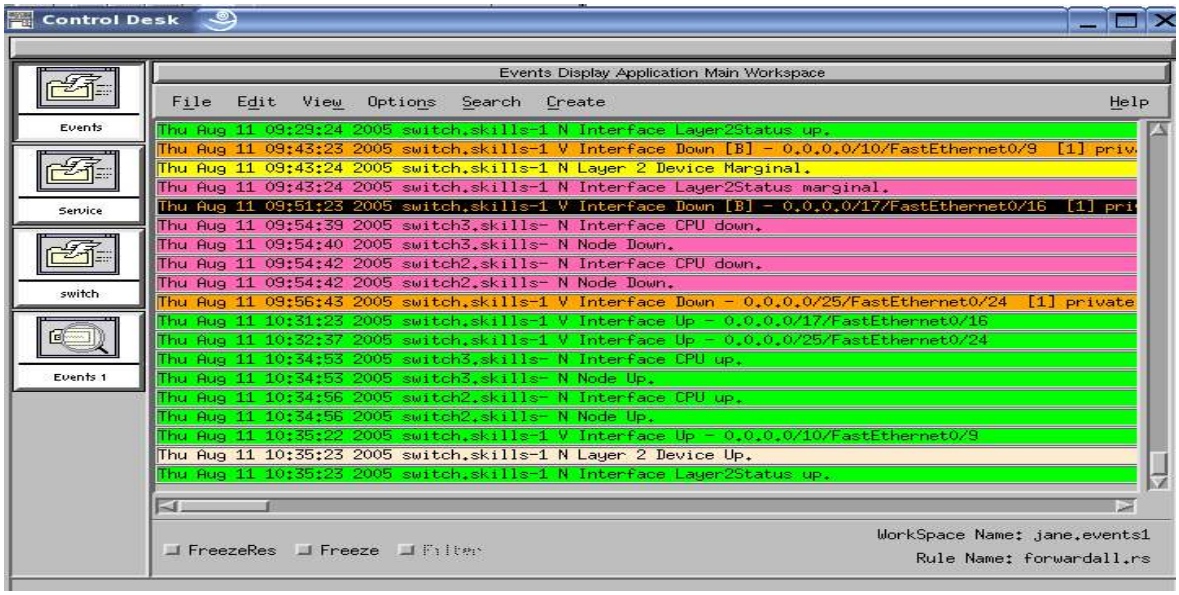

Figure 22 An unmanaged, an undiscovered and a managed node are lost

Note in Figure 22 that an Interface Down event with the origin of ***switch***, is received when each port loses contact with its end node. The first event is the loss of port index 10 (actual port 9), attached to the unmanaged ***blossom***; this event also triggers the netmon Layer 2 marginal events. The second event 8 minutes later is the loss of the undiscovered node ***poppet*** on port index 17 (actual port 16). The third sequence of events (approximately 3 minutes later) is when ***switch2*** is disconnected from ***switch*** port index 25 (actual port 24). This problem is detected by netmon (it must be as PSM is not monitoring this port); 2 minutes after the netmon (N) events the ITSA (V) event is generated by the correlator poll. Note that these problems have deliberately been spaced at greater than 120 second intervals (the configured PSM polling interval) in order to generate separate ***switch*** Interface Down events rather than a combined ***switch*** Node Marginal event.

The **interface_timeout** parameter in correlator.ini controls how many seconds elapse between discovery of a problem (either by netmon or by PSM), and the correlator root-cause algorithm starting. This is in case the problem is a "bounce" and rapidly self-heals. The default value of **interface_timeout** is 300seconds; I have 120 seconds configured, which means that "V" events will lag behind "N" events by approximately 2 minutes.

If an interface goes down and comes back up (bounces) within the interface_timeout period, correlation is stopped and no correlated root cause trap is issued. The interface that bounced is monitored and compared to the bouncing threshold that is defined by the **interface_bounce_count** parameter and the **interface_bounce_interval** parameter. By default, if the interface goes down three times within one hour, a problem exists and a correlated root cause trap is issued. When the interface stays up for the same interface_bounce_interval, an up trap is issued. I have changed these default values, for testing, to have **interface_bounce_count** set to **1** and **interface_bounce_interval** set to **120** seconds.

3.2.3 l2_polling.cfg, ports=C and correlator.ini, poll_all_ports=yes

This configuration has:

- /usr/OV/ITSL2/conf/files/l2_polling.cfg **ports** field set to **C**
- /usr/OV/ITSL2/conf/correlator.ini **poll_all_ports** field set to **yes**

Ensure that the itsl2 daemon is stopped and restarted to pick up configuration changes to these files and to perform rediscovery of Layer 2.

The **l2_polling.cfg**, **ports=C** setting is the Port Status Monitoring (PSM) Switch Table and defines that PSM is **only** active for ports to which devices are connected that have been **discovered** by NetView at Layer 3. Note that the **ports=C** parameter does not care whether the node is Managed or Unmanaged by NetView Layer 3 – the only criterion is that it has been discovered. From the point-of-view of the **poll_all_ports** parameter in correlator.ini, the **C** setting defines that only these ports are "configured". The **poll_all_ports=yes** parameter denotes that all "configured" ports will be PSM-polled.

l2_polling.cfg, **ports=C**, has a much wider effect than simply Port Status Monitoring; it also affects what is shown in the ITSA topology views and the Discovery Report.

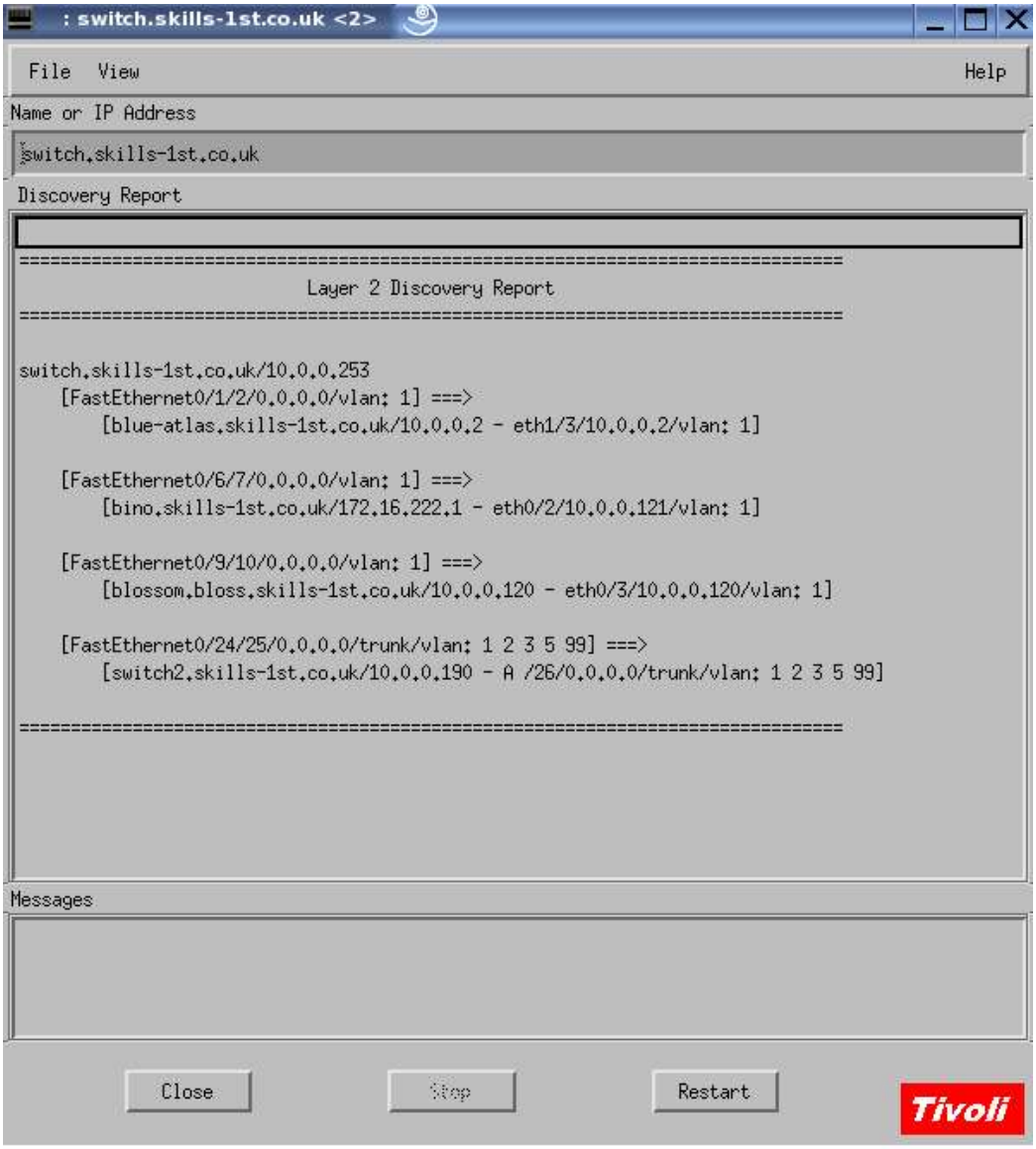

Figure 23 Physical View with l2_polling.cfg, ports field set to C

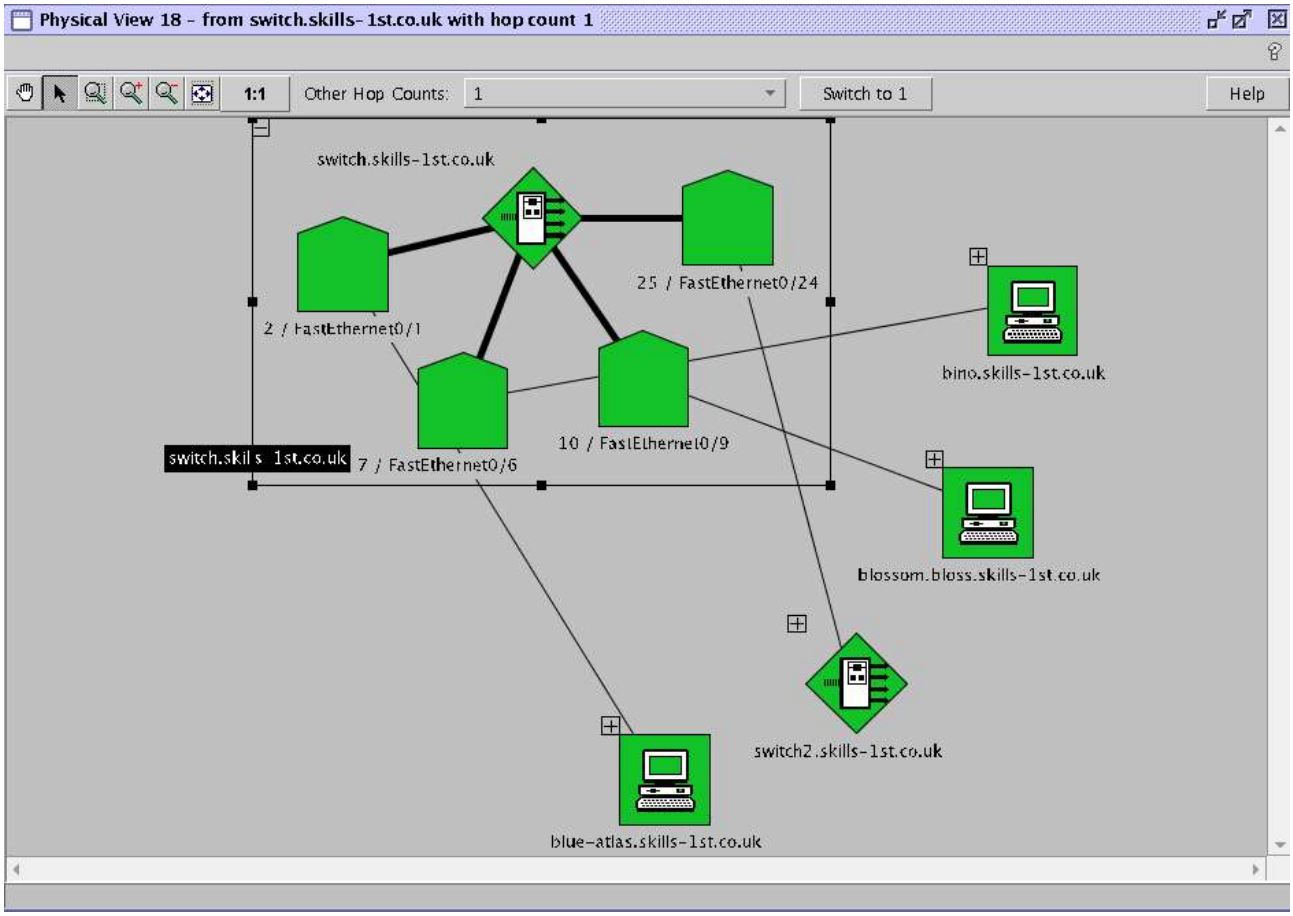

Figure 24 Physical view of switch with l2_polling.cfg, ports field set to "C"

Note in the Physical View, when the ports parameter is set to "C", that ports with nodes undiscovered at NetView Layer 3 do not display at all. You no longer see buff-coloured unmanaged ports representing disconnected ports.

Note that with this configuration, there is no way to see ports with nodes undiscovered by NetView – port 16 is not shown at all even though it still has the active but NetView-undiscovered node, ***poppet***, attached to it. Switch ports in a redundant layer 2 path will also **not** show ????or will they????. Switch ports in a Remote Campus LAN are unaffected by the A / C setting as ITSA does not discover connections in remote campuses.

Note that ***blossom*** **does** appear as a Normal node attached to a Normal status ***switch*** port, even though ***blossom*** is Unmanaged at NetView Layer 3. The node (correctly) has a Normal status at Layer 2.

An issue does arise if these parameters are changed when a node has already been discovered by NetView Layer 3 and has already been Unmanaged. When the Layer 2 topology rediscovery takes place, it does not find NetView Unmanaged nodes correctly. The node shows in a Discovery Report followed by an asterisk and associated with the wrong port. This means that the node does not appear in a Physical View or in a Discovery Report; hence it is also not included in a Status Report or in a PSM poll. This issue can be circumvented by ensuring that Layer 2 discovery is performed **before** a node is Unmanaged by NetView. Sometimes, it appears to need two ovstop / ovstart cycles of the itsl2 daemon to move nodes from asterisk status to correct discovery status.

3.2.3.1 Events generated for this scenario

For this scenario, events will appear in the NetView event log in a similar fashion to previous scenarios. The major difference here is that there will never be any events for the NetView-undiscovered node, ***poppet***.

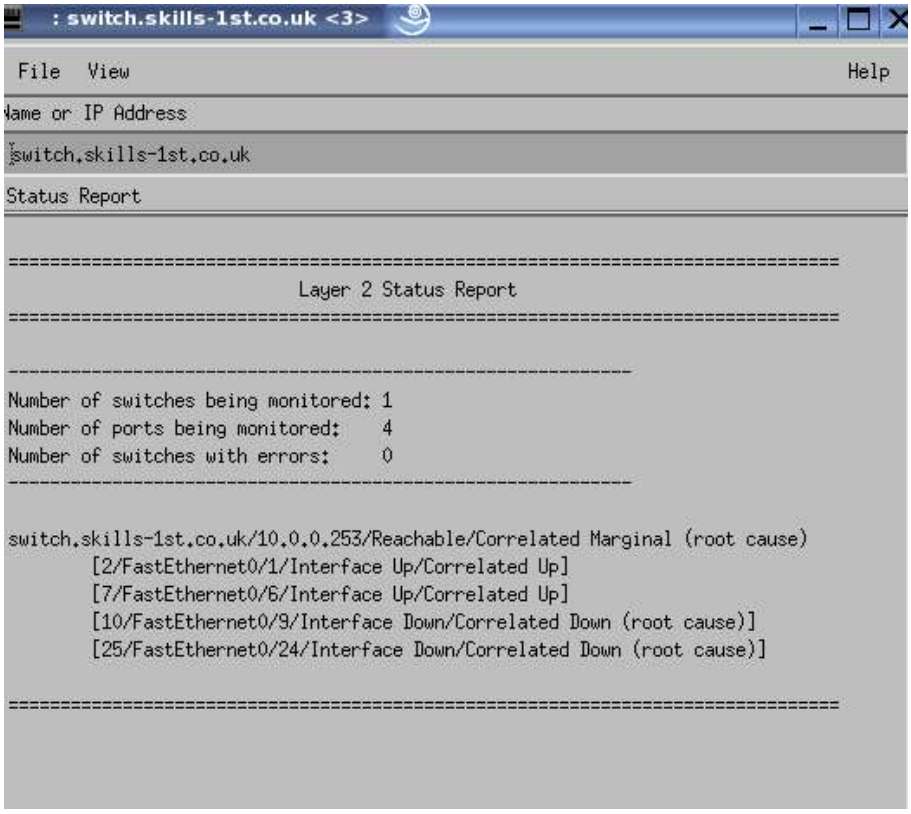

Figure 25 Status Report with poppet, blossom and switch2 disconnected, ports parameter set to "C"

Note in Figure 25 that there is no information for port 16 to which the undiscovered node, ***poppet***, is attached. There **is** information for port 9 to which the NetView-Unmanaged node, ***blossom***, is attached. Port 24 has the NetView-managed device, ***switch2***, attached.

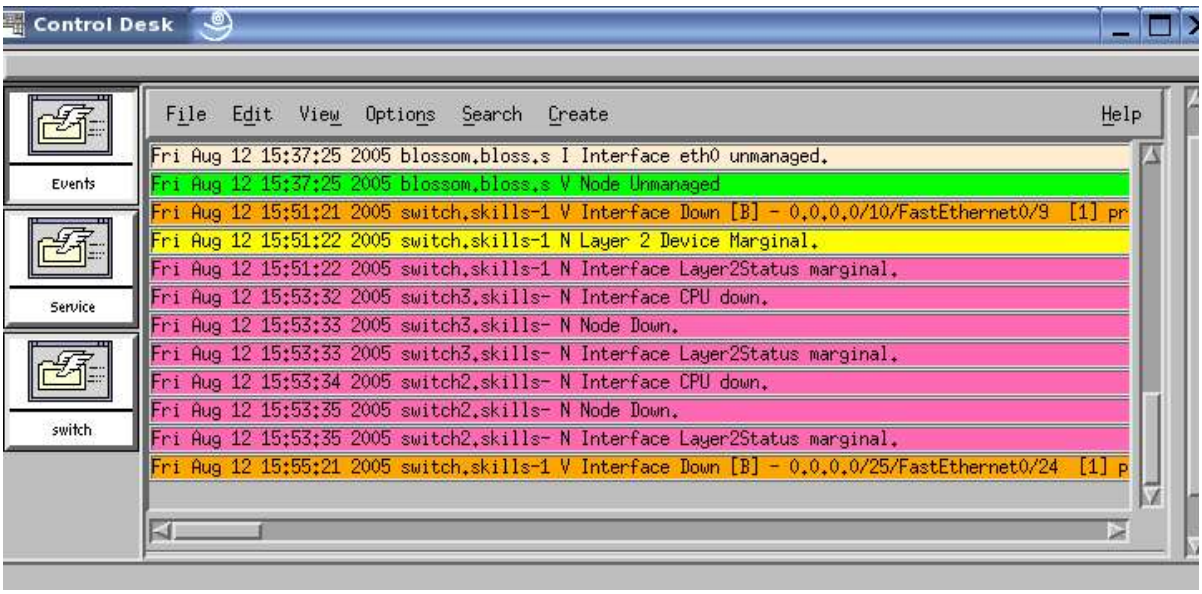

Figure 26 Event log when poppet, blossom and switch2 are disconnected

Note in Figure 26 that there is no event when the undiscovered node, ***poppet***, is disconnected. ***blossom*** (on port 9) and ***switch2*** (on port 24) are disconnected with more than 120 seconds between these two disconnections to ensure that separate "V" Interface Down events are received (remember that the PSM polling interval is configured to be 120 seconds by the **poll_cycle** parameter in /usr/OV/ITSL2/conf/l2_event_adapter.ini).

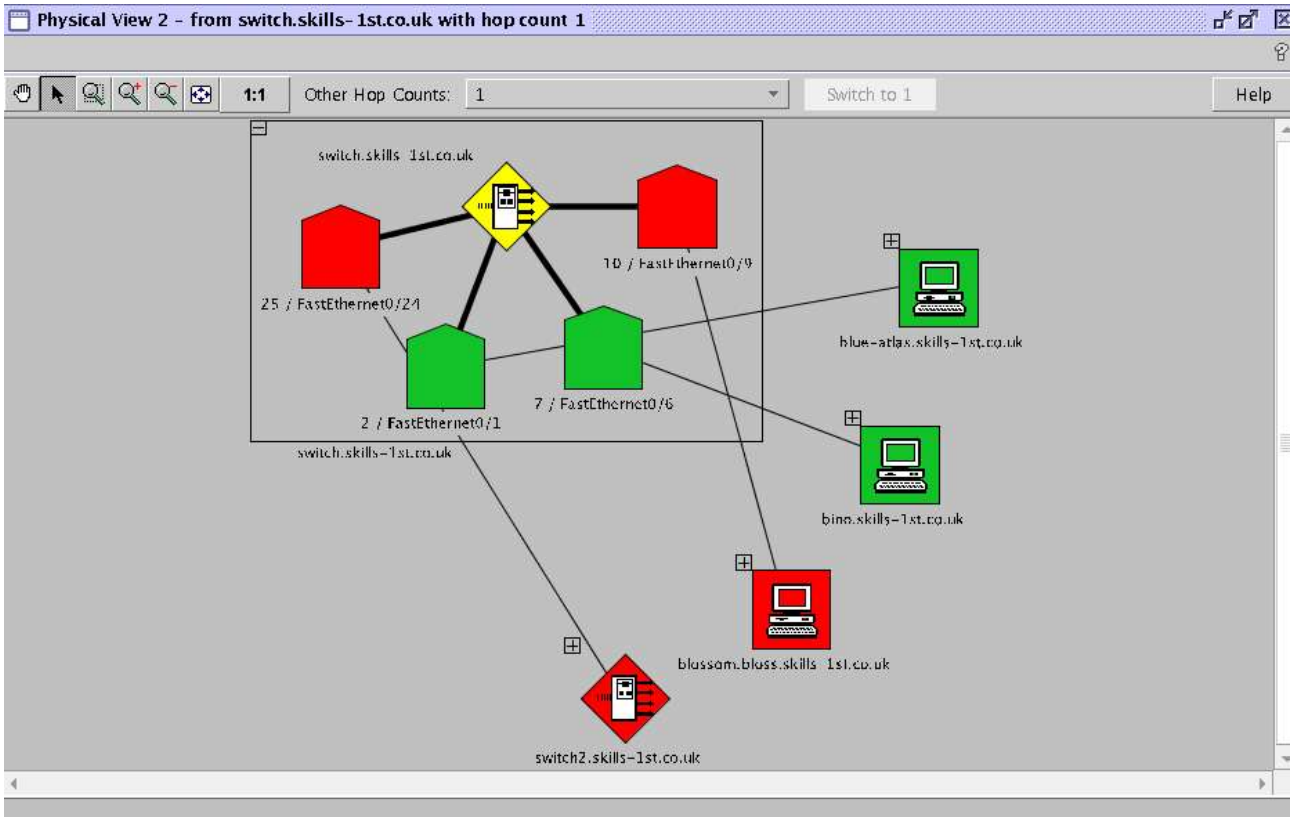

Figure 27 Physical View with poppet, blossom and switch2 down, ports parameter set to "C"

Note in Figure 27 that port 16, connected to the undiscovered node ***poppet***, does not display at all.

3.2.4 l2_polling.cfg, ports=C and correlator.ini, poll_all_ports=no

Setting the ports field to **C** in l2_polling.cfg determines that only ports that have NetView-discovered nodes attached to them, will be PSM-polled. Setting poll_all_ports=no in correlator.ini denotes that only ports with nodes that have **not** been discovered by NetView Layer 3 or are Unmanaged at NetView Layer 3, will be PSM-polled.

Thus, this configuration should only PSM-poll ports with NetView-discovered but Unmanaged nodes attached.

Ports to which nodes that are discovered and managed by NetView Layer 3 are attached, will still participate in Layer 2 correlator root-cause polls when triggered by a netmon, Layer 3 event.

Ports attached to NetView undiscovered nodes will be invisible at Layer 2.

To test this scenario, the NetView status polling interval for ***switch2*** and ***switch3*** was changed from 5 minutes to 25 minutes. The PSM poll interval is still set at 2 minutes.

The “netmon -a 12” and “netmon -a 16” commands were run and /usr/OV/log/netmon.trace was inspected to check that no Layer 3 ping or SNMP poll was due for at least 10 minutes. “netmon -a 12” causes netmon to dump his internal ping list showing, in the first column, the number of seconds to when an interface will next be pinged. “netmon -a 16” dumps the equivalent SNMP list.

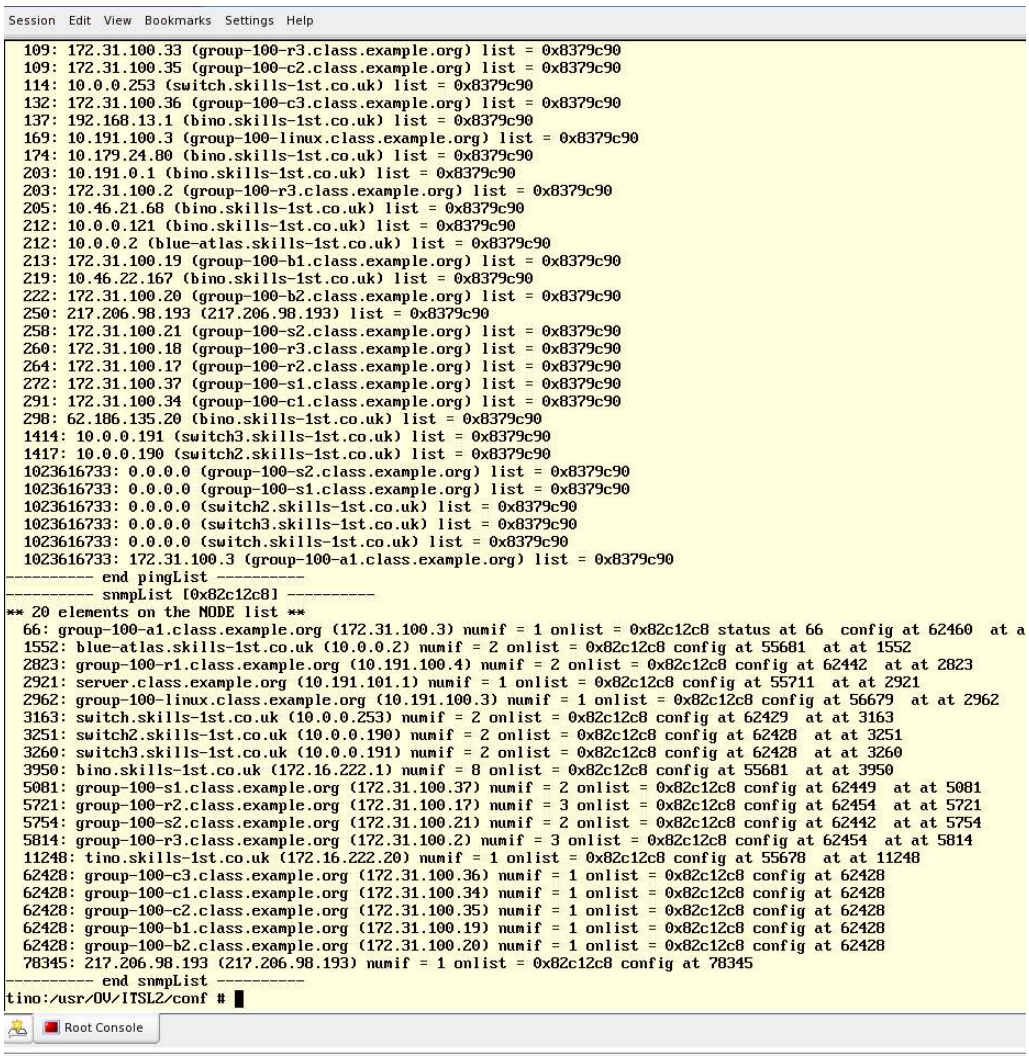

Figure 28 tail of /usr/OV/log/netmon.trace after "netmon -a 12" and "netmon -a 16"

Figure 28 shows that no ping poll of ***switch2*** or ***switch3*** is expected for 1414 seconds. The next SNMP poll of either is in 3251 seconds. With the current combination of settings in l2_polling.cfg and correlator.ini, there should be no PSM poll to either of these devices so no event should be generated in the NetView event log for at least 23 minutes.

3.2.4.1 Events generated for this scenario

For this scenario, two devices were disconnected – the NetView discovered but unmanaged, ***blossom*** and the NetView discovered and managed node, ***switch2***. The objective was to demonstrate that there were no PSM polls to ***switch2*** with this combination of parameters.

An unexpected result occurred initially when both devices were disconnected simultaneously. When the PSM poll that detected ***blossom*** connected to port 9 was down, it **also** detected that port 24 was down so a "V" Node Marginal was generated, rather than the expected "V" Interface Down against port 9.

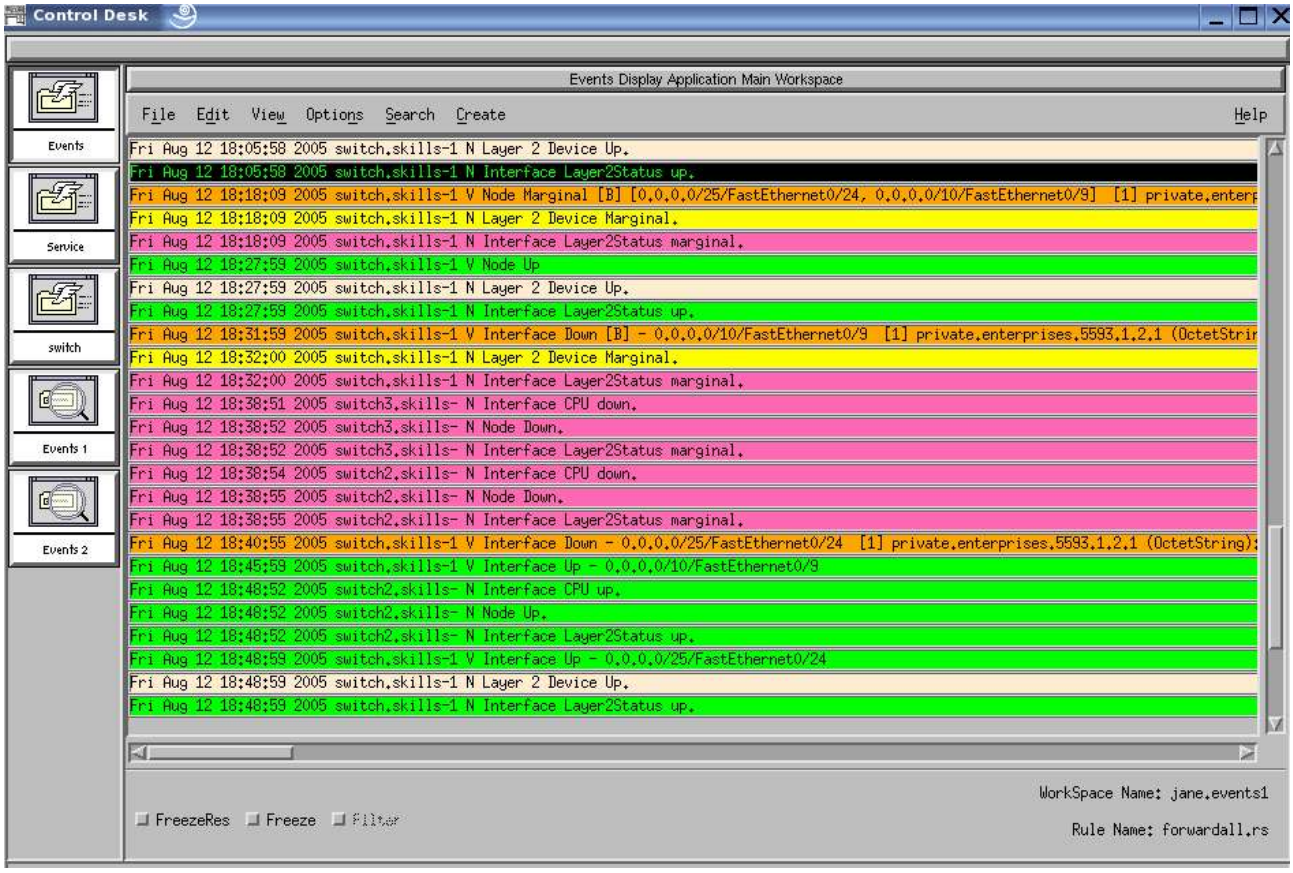

Figure 29 Demonstration of l2_polling.cfg, "ports" set to "C", and poll_all_ports=no

Both devices were reconnected and then ***blossom*** was disconnected first with a 2 minute interval before ***switch2*** was disconnected. The "V" Interface Down event for port 9 arrived 2 minutes after disconnection, as expected. By this time, "netmon -a" commands showed that the next Layer 3 poll of ***switch2*** was due in about 6 minutes. The Layer 3 netmon "N" event arrived on cue, with the usual 2 minute delay before the correlator root-cause "V" event arrived for port 24 (remember the 2 minute delay between Layer 2 problem detection and event generation is controlled by **interface_timeout** in correlator.ini).

3.2.5 Summary of PSM polling parameters

The summary of the setting for the ports field in l2_polling.cfg and the poll_all_ports field in correlator.ini, is as follows:

l2_polling.cfg	*correlator.ini*	*Result*
ports=A	poll_all_ports=yes	All ports configured to be PSM-polled. Correlation poll triggered by Node / Interface down found either by PSM or by netmon. This is the default.
ports=A	poll_all_ports=no	All ports configured to be PSM-polled; however PSM polling will only occur for ports with nodes undiscovered or discovered-and-unmanaged by NetView Layer 3. Ports with nodes managed by NetView Layer 3 will rely on correlator polls, triggered by netmon events, to determine Layer 2 problems.

l2_polling.cfg	*correlator.ini*	*Result*
ports=C	poll_all_ports=yes	Only NetView Layer 3 discovered nodes will appear in a Layer 2 Discovery Report and Status Report. These nodes are “configured” for Layer 2. All ports connected to these devices will be PSM-polled and those that are not in Remote Campus LANs will also participate in correlator root-cause polls. Ports connected to nodes undiscovered by NetView Layer 3 will be invisible. This is equivalent to ITSA 1.2.1.
ports=C	poll_all_ports=no	The only ports that will be PSM-polled will be NetView Layer 3 discovered but unmanaged nodes. Ports connected to nodes discovered by NetView Layer 3 will be polled by the passive correlator root-cause mechanism when triggered by a netmon Layer 3 event.

In conclusion regarding Port Status Monitoring, the *primary* purpose for PSM is to enable users to rely on ITSA to monitor active ports on switches for failures, without regard for complications relating to what NetView has discovered and/or the accuracy of the connections ITSA has discovered (given known information). The passive, correlator polling in the connected region simply adds a real time layer to outage detection, a bonus if you will. Correlator polling cannot occur in Remote Campus LANs.

The **poll_all_ports=n** moves away from this purpose and should only be considered when you are willing to sacrifice some possible accuracy in return for polling efficiency in large networks.

The **Ports "C"** option also moves away from this purpose and should only be considered when you are willing to sacrifice some possible accuracy for the benefit of reducing the noise of "uninteresting ports" (for example, ports on an access switch for desktop farms). This option preserves the ITSA 1.2.1 behaviour where only ports connected to discovered nodes will have V Interface Down events issued.

By "possible accuracy" I am referring to the reality of ITSA building topology from known information. When it cannot get all the information, it does the best it can for topology.

3.3 Configuration parameters for ITSA

A number of configuration parameters have been referred to throughout this document. They are all collated in this section. If any of these parameters are changed then the itsl2 daemon should be stopped and restarted.

For further information on these parameters, consult the ITSA 1.3 Admin Guide Chapter 7 and the ITSA 1.3 Troubleshooting Guide.

3.3.1 Parameters in /usr/OV/ITSL2/conf/correlator.ini

Name	*Default*	*Configured to..*	*Notes*
topo_cache_freq	900 s	200 s	Time between topology cache refreshes (reports are generated based on this cached data)
interface_timeout	300 s	120 s	Time between netmon / PSM event and correlator process starting
interface_bounce_count	3	1	Number of times an interface can go down within the interface bounce interval before it is correlated as down
interface_bounce_interval	3600 s	120 s	Time against which the accumulated bounce count is measured before an interface is correlated as down
polling_wait_time	0	0	Time to wait after an interface goes down before starting the correlator polling process. The polling_wait_time is included within the interface_timeout interval
corr_timeout	10 s	10 s	
status_poll_interval	15 s	15 s	Time internal polling process waits between interface polls. (This option refers to polling related to the root cause analysis engine and not port status monitoring.)
man_poll_interval	60 s	60 s	How often an ITSA-managed interface is polled. (This option refers to polling related to the root cause analysis engine and not port status monitoring.)

3.3.2 Parameters in /usr/OV/ITSL2/l2_topo_adapter.ini, Layer2 section

Name	*Default*	*Configured to..*	*Notes*
discovery_interval	1440 m (1 day)	200 s	The discovery poll forces a rediscovery of each switch
retry_interval	900 s	900 s	How frequently the correlator should retry a failed layer 2 request
retry_cnt	3	3	Maximum number of times the correlator should retry a failed layer 2 request

3.3.3 Parameters in /usr/OV/ITSL2/l2_event_adapter.ini

Name	*Default*	*Configured to..*	*Notes*
poll_cycle	300 s	120 s	Time between port status monitoring polling cycles
retry_cnt	2	2	How many times the port status monitor attempts to query a device before setting its status to unreachable

4 Event sequence for router root-cause problem

NetView and ITSA work in collaboration to determine the root-cause of a problem.

NetView's netmon daemon has the Router Fault Isolation (RFI) algorithm which determines whether a node has been lost or whether the problem is the loss of one or more routers. RFI quickly coalesces the situation to a critical or marginal router and one or more *Unreachable* (white) networks. Once a network is determined to be Unreachable then node status polling is suspended into such networks. The root-cause of the problem is easily displayed in the NetView topology (by a red or yellow router) and the NetView event log should also only show the root-cause problem, along with one or more "Network Unreachable" events.

In parallel with RFI, the ITSA correlation process determines whether the root-cause of a problem is the loss of a node or the loss of a switch or switch blade. The overall result is a rapid determination whether the root-cause is a node, a switch or a router.

In the previous section all the problems were actually caused by individual nodes being disconnected. This section examines the outcome when the problem is with a router.

..... to be continued .....

5 TEC integration of Layer 2 and Layer 3 events

NetView and ITSA integrate well together to present, at the NetView event log and in the NetView topology, a combined root-cause analysis. The Tivoli Enterprise Console (TEC) adds an extra layer of root-cause problem determination and automation.

TEC also permits correlation between *network* events from NetView and ITSA, and *system, middleware* and *application* events generated by IBM Tivoli Monitoring (ITM), Distributed Monitoring (DM), TEC adapters such as the Unix logfile adapter and the Windows event adapter, ITM add-ons for monitoring WebSphere, databases, WebSphere Message Queueing (MQ), and from third party TEC adapters for monitoring applications.

TEC 3.9 ships with a ruleset, **netview.rls**, that automatically correlates a number of NetView and ITSA events out-of-the-box. NetView 7.1.4 ships with a NetView ruleset, **TEC_ITS.rs**, that is designed to forward a subset of NetView and ITSA events from NetView to TEC. By default, this forwarding is disabled but it is a simple matter to enable it.

5.1 NetView / TEC Architecture

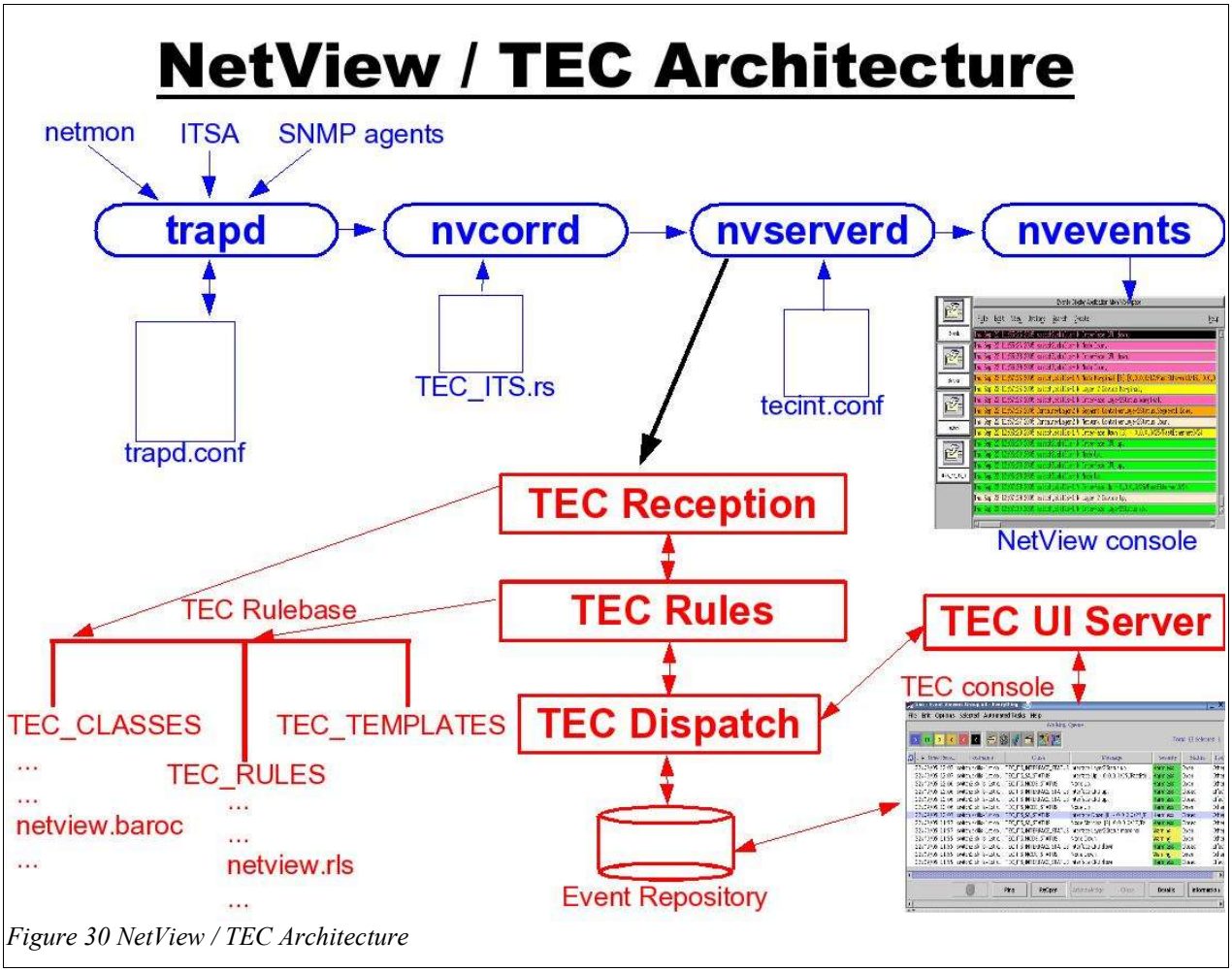

Figure 30 NetView / TEC Architecture

Figure 30 shows a simplified version of both NetView and TEC architecture – note that other processes and configuration files exist but are not relevant to this discussion.

In Figure 30, the top 4 daemons (in blue for those with colour) are part of NetView. The lower half of the diagram show TEC processes and structures (in red). It is not part of this paper to discuss either TEC or NetView architecture in detail; however, the areas of each product relevant to receiving and customising events from NetView and ITSA, will be discussed, along with their configuration files. Roughly speaking, the configuration files are:

- NetView, /usr/OV/conf/C/trapd.conf "How to forward events to TEC"
- NetView, /usr/OV/conf/rulesets/TEC_ITS.rs "Which events to forward to TEC"
- NetView, /usr/OV/conf/tecint.conf "Where to forward events to TEC"
- TEC, netview.baroc in active rulebase "Is this a legal NetView event?"
- TEC, netview.rls in active rulebase "What to do with this NetView event"

5.1.1 NetView's TRAP handling architecture

NetView's **trapd** is the fundamental receiver of SNMP TRAPs, whether they are from netmon, ITSA or from SNMP agents out in the network. trapd can forward TRAPs to a number of NetView daemons, including the NetView correlation engine, **nvcorrd**.

trapd is configured through **/usr/OV/conf/C/trapd.conf**. This file is generally manipulated from the NetView GUI (**Options -> Event Configuration**), or by the command line **xnmtrap**. This file defines each SNMP TRAP to NetView and determines what customisation is appropriate (such as severity, status, category, message displayed) and what action should be taken (such as don't display, run a script, produce a popup).

trapd.conf can only make simple decisions - "*if* the TRAP is of this type, *then* do that". Events are passed from trapd to nvcorrd. nvcorrd can take much more complex decisions on TRAP processing by applying NetView rulesets. This can include greater examination of the incoming TRAP (like TRAP varbind values) and can also take decisions by checking various other factors (such as whether the TRAP originates from a member of a SmartSet, has a certain NetView attribute set, whether the TRAP is part of a sequence of TRAPs, etc.).

NetView rulesets are created using the GUI Ruleset Editor (from the **Tools** menu).

nvcorrd can apply different NetView rulesets to different "output streams". These "output streams" are:

- the NetView event GUI (**nvevents**)
- background processing (by placing a ruleset name in **/usr/OV/conf/ESE.automation**)
- the NetView TEC adapter (**nvserverd**)

NetView ships with a number of sample rulesets, including **TEC_ITS.rs** which is designed for use with the nvserverd TEC adapter and selects a subset of NetView and ITSA events for forwarding to TEC. This ruleset can obviously by modified by users or a different ruleset can be used for event forwarding.

The nvserverd TEC adapter has a typical ".conf" configuration file that can either by modified using the NetView **serversetup** utility, or can be modified directly. If modified directly, nvserverd must be recycled using **nvtecia -reload**. **/usr/OV/conf/tecint.conf** is the mandatory path for this TEC adapter configuration file. It specifies, among other things, where the TEC Server is and which NetView ruleset is used to filter events to TEC. If you wish to send *all* events, the sample **forwardall.rs** can be used.

5.1.2 TEC architecture

An event from NetView arrives at TEC at the **Reception engine**. The first thing that happens is the event is timestamped and checked for correctness. A TEC Server keeps a "dictionary" of known events in a TEC rulebase. A rulebase is simply a directory structure with three main subdirectories:

- TEC_CLASSES contains the event "dictionary" in files ending **.baroc**
- TEC_RULES contains correlation and automation rules in files ending **.rls**
- TEC_TEMPLATES contains TEC action primitives in files ending **.wic**

By default, TEC 3.9 provides a definition file for NetView events, called **netview.baroc** and a rules file called **netview.rls**. For an event from NetView to be successfully received and processed by TEC, the format of the NetView event *must* match a definition in the active TEC rulebase; otherwise it will be set aside by the TEC reception engine and will not be processed by the TEC **Rules engine.** (Use the TEC **wtdumprl** command and look for **PARSING FAILED** messages if you suspect that a NetView event is not being correctly parsed at TEC).

It is the TEC rules engine that processes the incoming event against each of the rules in the active rulebase and determines what correlation and automation will take place.

The TEC **Dispatch** process is responsible for taking processed events and storing them in the TEC **Event Repository** database. Dispatch can also send events to a **Task engine** (not shown in Figure 30) for long-running automation (such as running scripts or Tivoli Tasks).

The TEC **UI Server** process is responsible for communicating with TEC Consoles.

5.2 Scenarios with ITSA events at TEC

A number of series of networking events were generated in order to demonstrate the correlation at TEC between "good news" and "bad news".

5.2.1 Configuration files for TEC / ITSA scenarios

5.2.1.1 NetView trapd.conf configurations relating to TEC

The IBM Tivoli NetView for Unix Administrator's Guide V7.1.4, Appendix B, documents all the TRAPs that are configured to be converted to TEC events, out-of-the-box (Table 36). This includes most events generated by netmon and by ITSA.

All these NetView and ITSA events are converted into TEC classes that start with **TEC_ITS** and a number of variables from the SNMP TRAP are mapped into **attributes** (or **slots**) on the TEC event (for example, hostname, NetView node, message, interface name, ...).

There are a category of events generated by NetView that are "Layer 2" events, generated as a result of correlation between NetView and ITSA. These events also appear in Table 36.

5.2.1.2 NetView ruleset TEC_ITS.rs

A subset of the events configured for TEC conversion in trapd.conf, appears in the default TEC_ITS.rs NetView ruleset. The subset in TEC_ITS.rs is documented in the same appendix in the Admin Guide, in Table 35. This subset includes:

- Router / Node / Interface Up / Down / Marginal / Unreachable
- Node / Interface Added / deleted / Managed / Unmanaged
- Subnet connectivity Reachable / Unreachable
- ISDN status Active / Dormant
- SNMP collection Threshold / Rearm
- ITSA events

Note that with NetView 7.1.4, the NetView "Layer 2" events do **not** appear in TEC_ITS.rs; with NetView 7.1.3 these events **were** included.

5.2.1.3 nvserverd TEC adapter configuration file, /usr/OV/conf/tecint.conf

For these scenarios, the default configuration of tecint.conf has been used which uses non-TME communications between NetView and TEC, has State Correlation enabled and uses the NetView ruleset, TEC_ITS.rs.

5.2.1.4 TEC rulebase files - netview.baroc

The default class file with TEC 3.9, **netview.baroc** was used, along with the default rules file, **netview.rls**.

netview.baroc contains TEC definitions to match all the TRAPs generated by NetView and ITSA and configured in NetView's trapd.conf. TEC classes are built in hierarchies such that more specific events inherit characteristics from more generic events. The base event, **EVENT**, defines attributes that **every** TEC event has.

There is a superclass of events for NetView events, **TEC_ITS_BASE,** which adds on extra attributes such as generic trap number, specific trap number, trap variable slots, etc. Leaf node NetView TEC event examples would be **TEC_ITS_INTERFACE_STATUS, TEC_ITS_ROUTER_STATUS**, and so on.

ITSA event classes have a superclass, **TEC_ITS_SA_EVENT** which inherits from **TEC_ITS_BASE** but also includes two extra attributes:

- sastatus
- saticketnumber

5.2.1.5 TEC rulebase files - netview.rls

netview.rls is a large and complex rules file, a detailed examination of which is beyond the scope of this paper. The main features that it implements, relevant to ITSA, are:

- If NetView “Node Down” and “Interface Down” events arrive from the same node, within 10 minutes, then the “Node Down” will be the root-cause event and the “Interface Down” event will be CLOSED.
- If NetView “Router Down” and “Interface Down” events arrive, from the same node, within 10 minutes, then the “Router Down” will be the root-cause event and the “Interface Down” event will be CLOSED. Any “Subnet Unreachable” events will be effect events of the “Router Down”.
- If NetView “Router Marginal” and “Interface Down” events arrive, from the same node, within 10 minutes, then the “Interface Down” will be the root-cause event and the “Router Marginal” event will be CLOSED. Any “Subnet Unreachable” events will be effect events of the “Interface Down”.
- Any Router / Node / Interface / Subnet event will clear a similar event of the same class, from the same element, within 10 minutes.
- If Router events and ITSA events appear for the same node, within 10 minutes, the Router event will be causal and the ITSA events will be CLOSED.
- If ITSA events and NetView “Layer 2” events appear for the same node, within 10 minutes, then the ITSA event will be causal and the “Layer 2” event will be CLOSED.
- If Node events and ITSA events appear for the same node, within 10 minutes, then the ITSA event will be causal and the Node event will be CLOSED.
- If ITSA events appear with the same saticketnumber number, from the same node, then previous ITSA events from this node, within 10 minutes, will be CLOSED.

netview.rls was included in the active TEC rulebase. The active rulebase also included the default ruleset **cleanup.rls**, which is useful for automatically clearing “good news” events.

5.2.2 Scenario 1 – Node undiscovered by NetView goes down

In this first scenario, the node ***poppet,*** which is attached to port 16 of the switch called ***switch***, is disconnected. ***poppet*** is completely unknown to NetView at layer 3 but ITSA understands the Layer 2 connectivity.

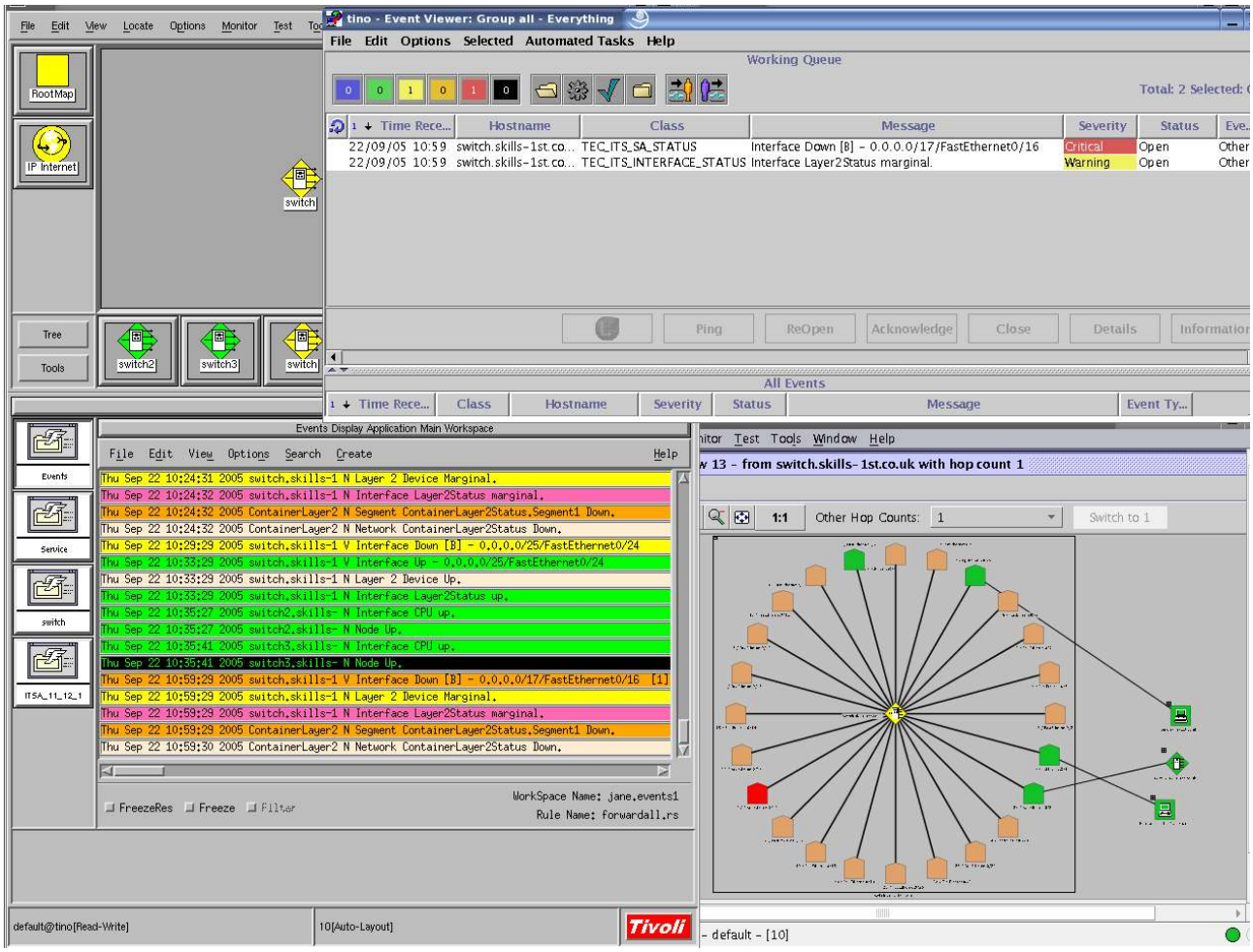

Figure 31 poppet disconnected

Figure 31 shows the NetView event log when ***poppet*** is disconnected. A "V" "Interface Down" event is received from ITSA for ***switch***, detailing the problem on port 16. This is followed immediately by a NetView "Layer 2" event of "Device Marginal" for ***switch***. The third event in immediate succession is a standard NetView "Interface Down" event for ***switch*** – this refers to the extra Layer2Status icon inside the Node submap of ***switch***, that is created for any switch when ITSA is installed. This interface will be yellow inside the ***switch*** node submap and is responsible for the yellow, Marginal status of the whole node, ***switch*** .

The last two events from an origin of **ContainerLayer2** for Segment and Network Down, appear to be redundant and may be better removed from the NetView event log.

The Physical View from ITSA for ***switch*** shows that port 16 has turned Critical (red). The node ***poppet*** is not shown as it is undiscovered by NetView at Layer 3.

The TEC Console shows two events:

- The TEC_ITS_SA_STATUS event has been translated from the "V" "Interface Down" TRAP and contains the failing port in the **msg** message slot. This event is customised at NetView to send a TEC severity of CRITICAL.
- The TEC_ITS_INTERFACE_STATUS event is the standard NetView "Interface Down" event for ***switch*** for the Layer2Status interface. As NetView is customised to send all standard NetView Interface Down events to TEC, this event is obviously included but is probably redundant to TEC.

Note that "Layer 2" NetView events and Network and Segment events are not included n TEC_ITS.rs so these events are not forwarded to TEC.

When ***poppet*** is reconnected, the scenario in Figure 32 is depicted.

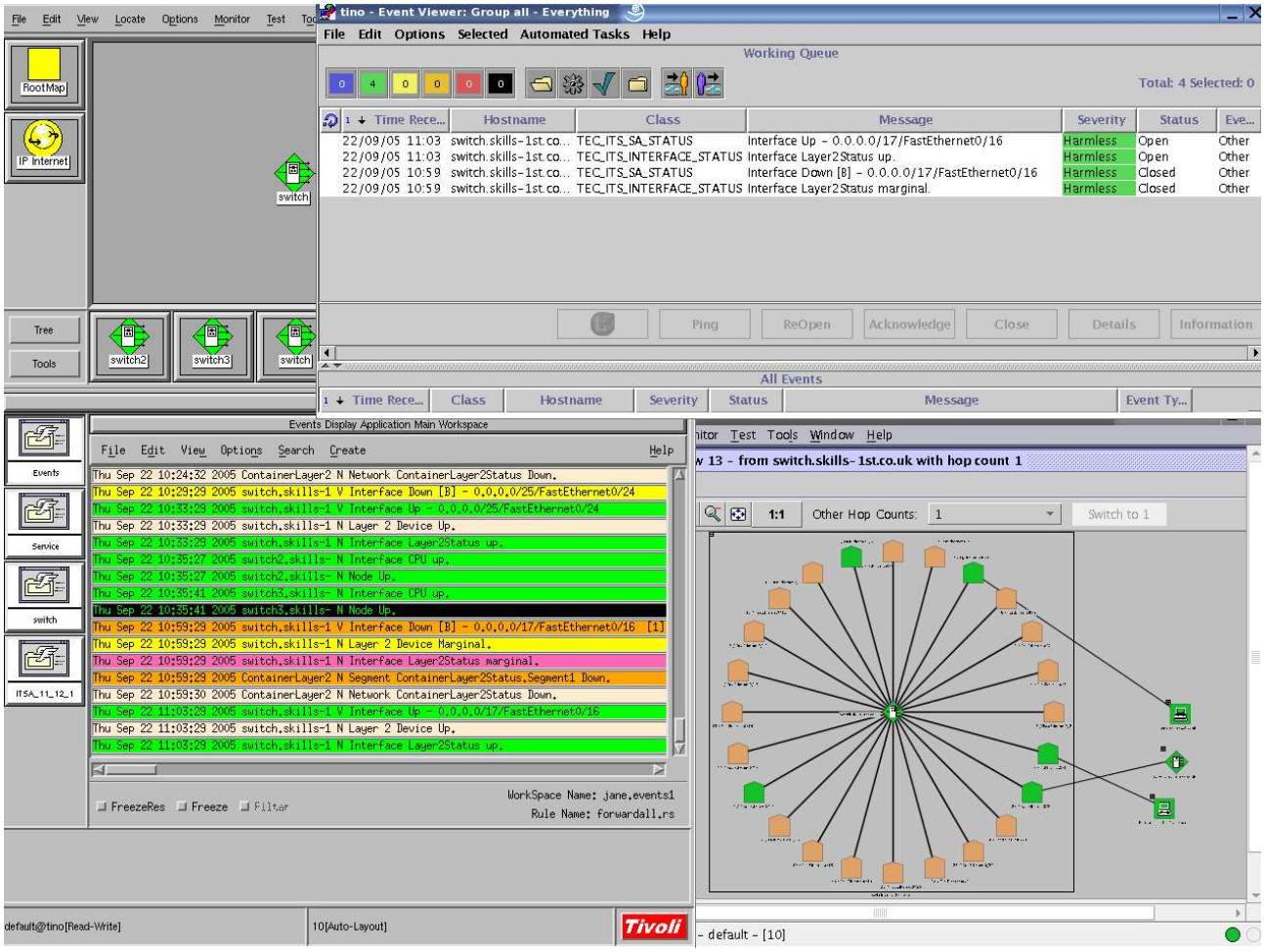

Figure 32 poppet reconnected

The NetView event log shows the corresponding "good news" TRAPs. The ITSA Physical View shows port 16 restored to Normal (green) status.

At TEC, two events are received – the corresponding "good news" events to the "bad news" events received earlier. The rule in netview.rls that clears TEC_ITS_INTERFACE_STATUS events with any other TEC_ITS_INTERFACE_STATUS event from the same node, within 10 minutes, is responsible for closing the bottom event seen in the TEC console.

Similarly, the rule in netview.rls that clears TEC_ITS_SA_STATUS events with any other TEC_ITS_SA_STATUS events, from the same node, with the same saticketnumber, within 10 minutes, is responsible for closing the next-to-bottom line in the TEC console.

Note that the "good news" events are left with a status of OPEN and a severity of HARMLESS. The ruleset **cleanup.rls**, shipped as standard with TEC, will close any HARMLESS or UNKNOWN events after 48 hours. This ruleset could be modified to suit local procedures.

5.2.3 Scenario 2 – 2 nodes down, one NetView-discovered, one not

In this scenario, two nodes were disconnected. ***blossom*** is a node attached to port 9 of ***switch*** and ***switch2*** is connected to port 24 of ***switch*** . ***blossom*** is Unmanaged at NetView Layer 3 whereas ***switch2*** is Managed. The nodes are disconnected more than 2 minutes apart so that the ITSA polling interval detects these events as separate "Interface Down" events rather than a single "Node Marginal" event.

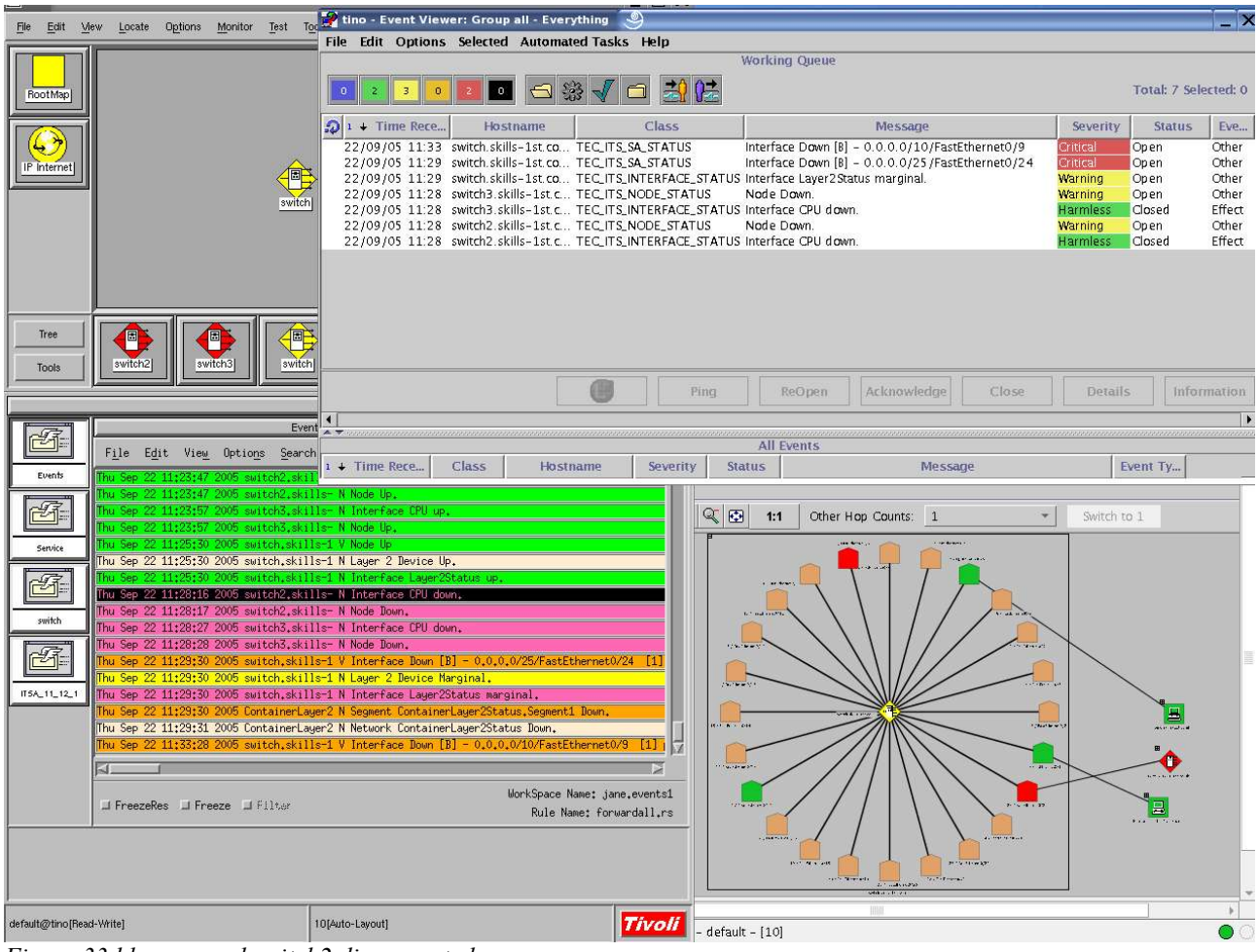

Figure 33 blossom and switch2 disconnected

Figure 33 shows the NetView event log. netmon was the first process to discover problems by detecting a failed ping to ***switch2*** and to ***switch3*** (which is cabled from ***switch2***). A minute later, the ITSA correlation process also reports the problem with an ITSA "V" "Interface Down" event, reporting a problem with port 24. The next 4 events, all received at the same time, are similar to the previous scenario.

2 minutes later, ***blossom*** was disconnected. As this node is Unmanaged by NetView Layer 3, there are no NetView "Node Down" or "Interface Down" events but the port to which it is attached (port

9 on ***switch***) is being monitored by the Port Status Monitoring (PSM) function of ITSA. Thus the problem is detected and reported in the last event shown in the NetView event log.

The ITSA Physical View shows the problem ports as Critical (red). Since ***switch2*** has been discovered and managed by NetView Layer 3, it also shows on the Physical View (as Critical). ***blossom*** is discovered but Unmanaged at NetView Layer 3 so the node is not displayed here.

At the TEC console, the standard NetView "Node Down" and "Interface Down" events can be seen for ***switch2*** and ***switch3*** . The rule in netview.rls that correlates node and interface events, determines that the Node events are causal and the Interface events are effects. The effect events are thus closed.

A minute later, the TEC_ITS_SA_STATUS event is received for the failing port 24 on ***switch***. As in the previous scenario, a standard TEC_ITS_INTERFACE_STATUS event is also received from ***switch*** for the interface Layer2Status marginal. Note that there is no correlation at TEC between the Layer 3 events for ***switch2*** and the Layer 2 events for port 24 on ***switch;*** however, this is not an issue.

4 minutes later, the second problem is reported to TEC when ITSA detects problems with port 9 on ***switch***.

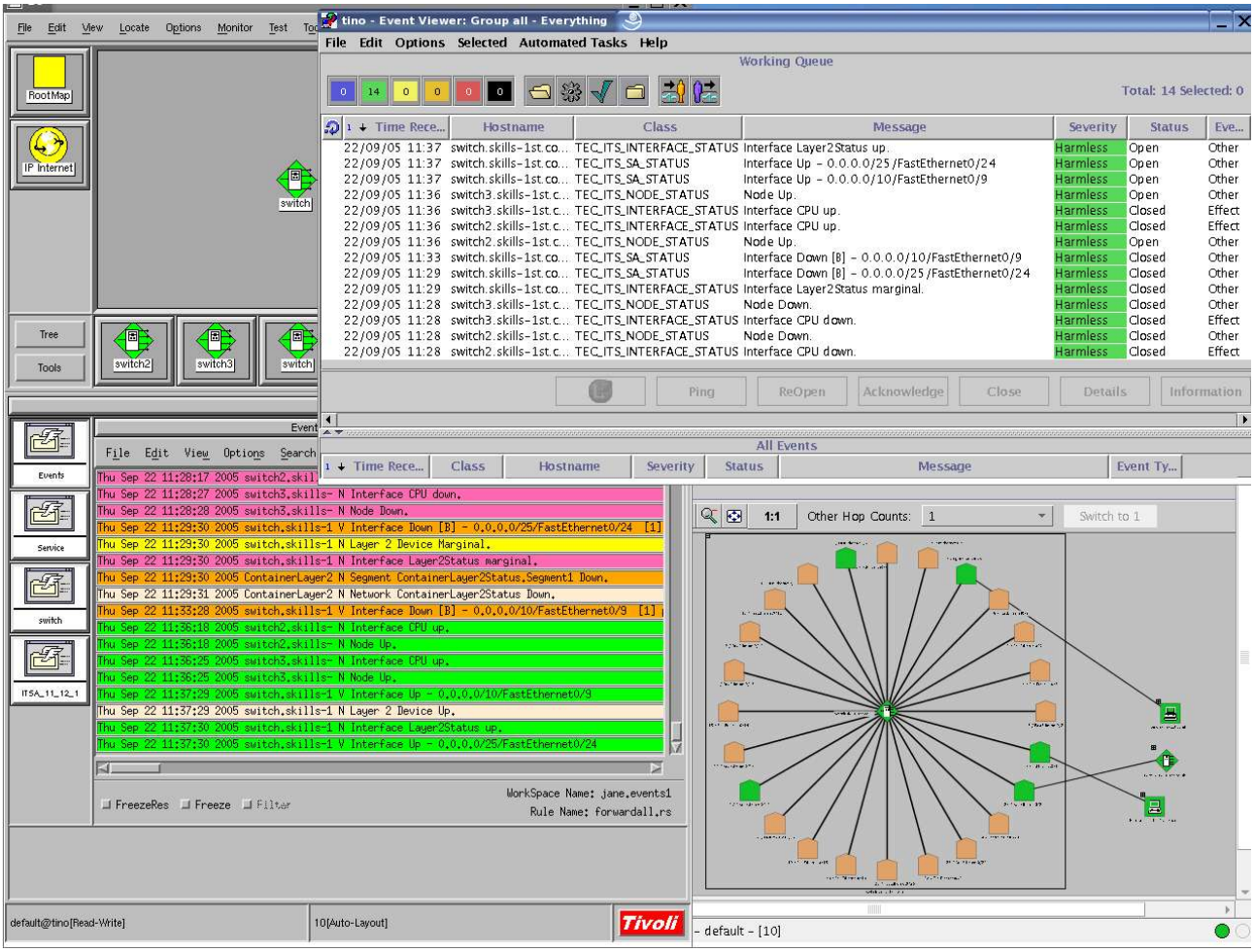

Figure 34 blossom and switch2 reconnected

switch2 and ***blossom*** were reconnected at the same time. The NetView event log shows that, again, it was netmon that first discovered the problem was solved with ***switch2*** and ***switch3***, followed by an ITSA event for port 24 on ***switch*** and an ITSA event for port 9 on ***switch*** .

At the TEC console, the “Interface Up” events for ***switch2*** and ***switch3*** are correlated with the corresponding “Node Up” events, and the Interface effect events are closed. The “Node Up” events are also correlated with their respective “Node Down” events as they have arrived within 10 minutes, so the “Node Down” events are closed. This means that the Layer 3 events for ***switch2*** and ***switch3*** are all dealt with.

Similarly, the top event showing in the TEC console is the standard TEC_ITS_INTERFACE_STATUS event for ***switch*** for the Layer2Status interface, with the “good news” status. This event automatically closes the corresponding “bad news” event.

The two TEC_ITS_SA_STATUS events can be seen showing the “good news” events for each of the ports affected on ***switch***; these events close the earlier corresponding bad news events as they have arrived within 10 minutes and have the same saticketnumber.

This completes the TEC correlation, leaving only causal “good news” events left open. Again, the rules in the **cleanup.rls** ruleset can tidy up these events.

5.2.4 Scenario 3 – 3 switch-connected nodes down at the same time

This third scenario, shows what happens when three nodes (but not all nodes) on the same switch, go down within one ITSA polling interval. If all the connected nodes went down then ITSA would correlate the problem to a single root-cause problem with the switch itself, rather than a number of root-cause problems on ports.

The three problem nodes are the same already described above, ***poppet***, ***blossom*** and ***switch2***. They are all disconnected simultaneously.

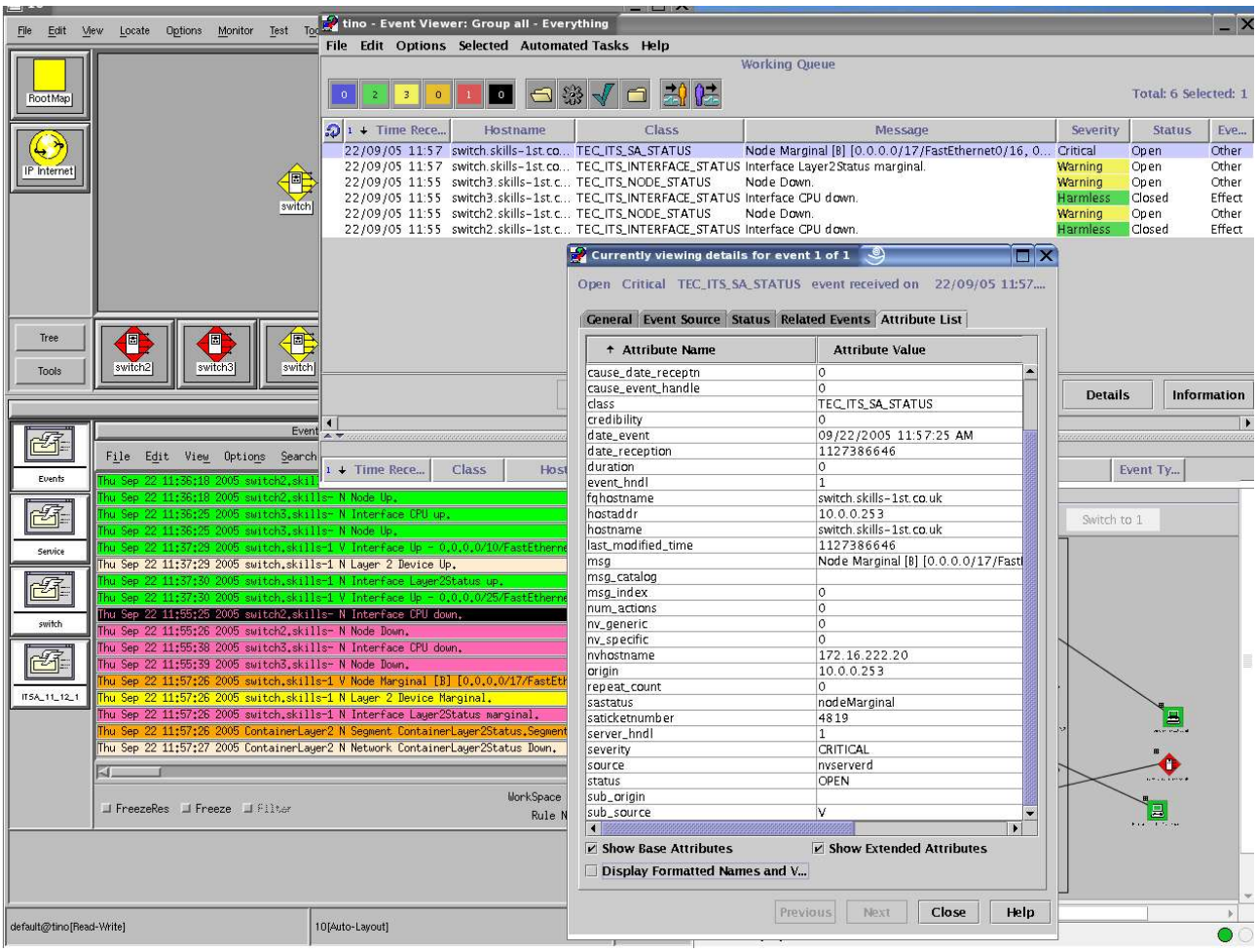

Figure 35 poppet, blossom and switch2 disconnected

The NetView event log shows a similar pattern as in the previous scenario. The difference here is that a single "V" "Node Marginal" event is received against ***switch***, rather than separate "V" "Interface Down" events. This is because a number of problems have been discovered within one ITSA polling interval. The single SNMP varbind with the message documents each of the failing ports. Events in the ITSA Physical View and at the TEC console are similar to the previous scenario; however the event detail for the TEC_ITS_SA_STATUS Node Marginal event is included in Figure 35, to highlight the saticketnumber attribute (4819) on the event.

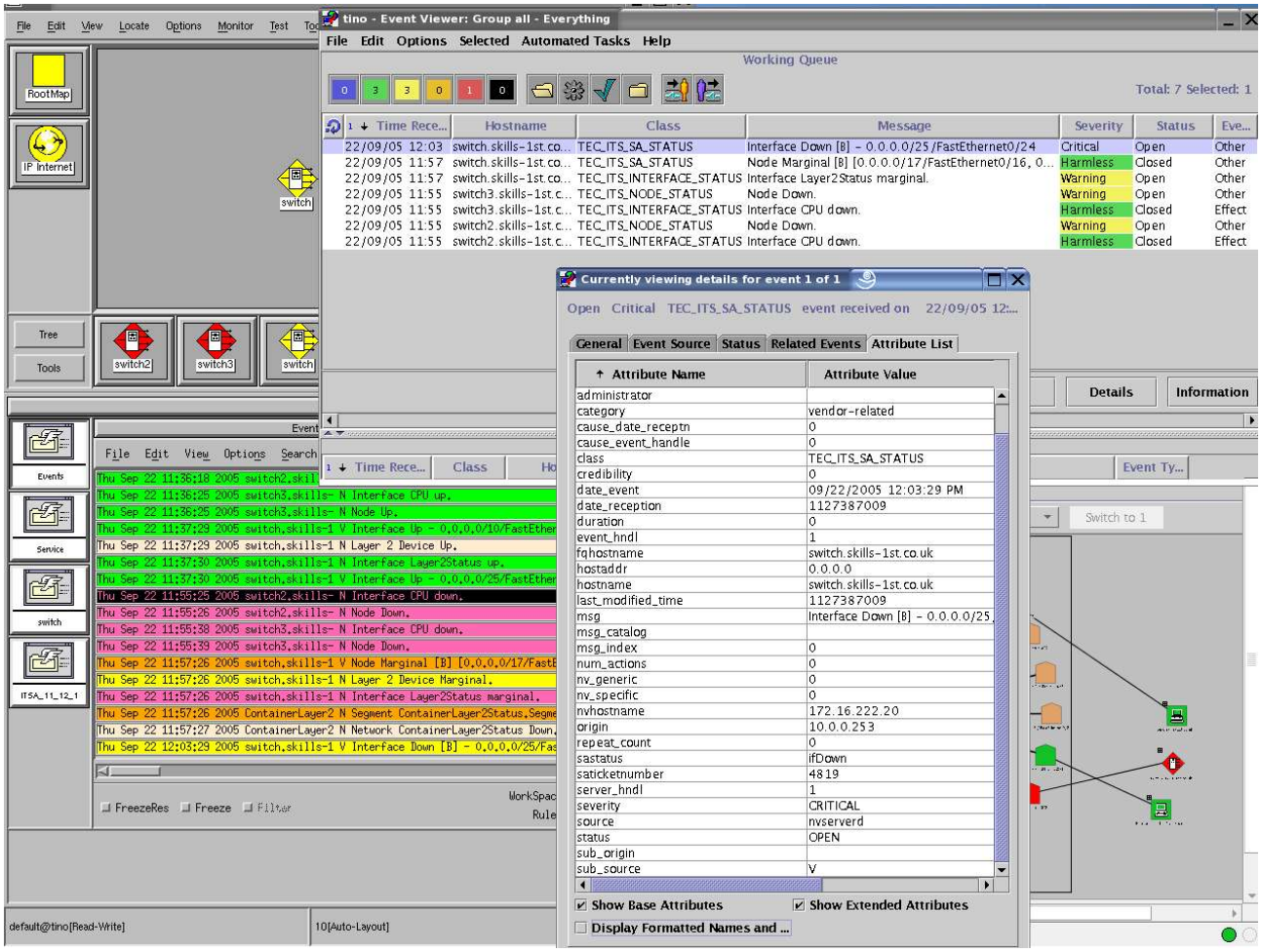

Figure 36 poppet and blossom reconnected, switch2 still disconnected

The nodes ***poppet*** and ***blossom*** were reconnected. Remember that ***poppet*** is undiscovered by NetView Layer 3 and ***blossom*** is unmanaged by NetView Layer 3 so neither reconnection generates Layer 3 events. However, both are connected to ports on ***switch*** that are being polled by PSM so their recovery is noted by ITSA and can be seen in the ITSA Physical View.

The slightly strange effect from the point-of-view of ITSA events, is that no "V" "Interface Up" events are generated for these reconnections. What *does* happen is that when there is only one outstanding problem left on ***switch***, a "V" "Interface Down" event is generated for that outstanding problem on port 24. Effectively, in the NetView event log, the problem on ***switch*** has been downgraded from a "Node Marginal" to an "Interface Marginal".

The "V" "Interface Down" event can be seen at the TEC console. Interestingly, the logic in netview.rls checks the saticketnumber attribute on this event and, finding it the same as the earlier TEC_ITS_SA_STATUS Node Marginal event, the earlier event is automatically closed. Thus events at TEC also reflect that the problem with ***switch*** has been downgraded from a "Node Marginal" to an "Interface Marginal" and, better than the event log in NetView, the "Node Marginal" event has been closed.

When the last problem on ***switch*** is resolved as ***switch2*** is reconnected, the usual succession of Layer 3 and Layer 2 events arrive at the NetView event console and at TEC.

In this case, the "Interface Up" event for the Layer2Status interface on ***switch*** and the "Node Up" events for ***switch2*** and ***switch3*** arrived more than 10 minutes after their corresponding "bad news" events so the automatic closure rule in netview.rls did not apply.

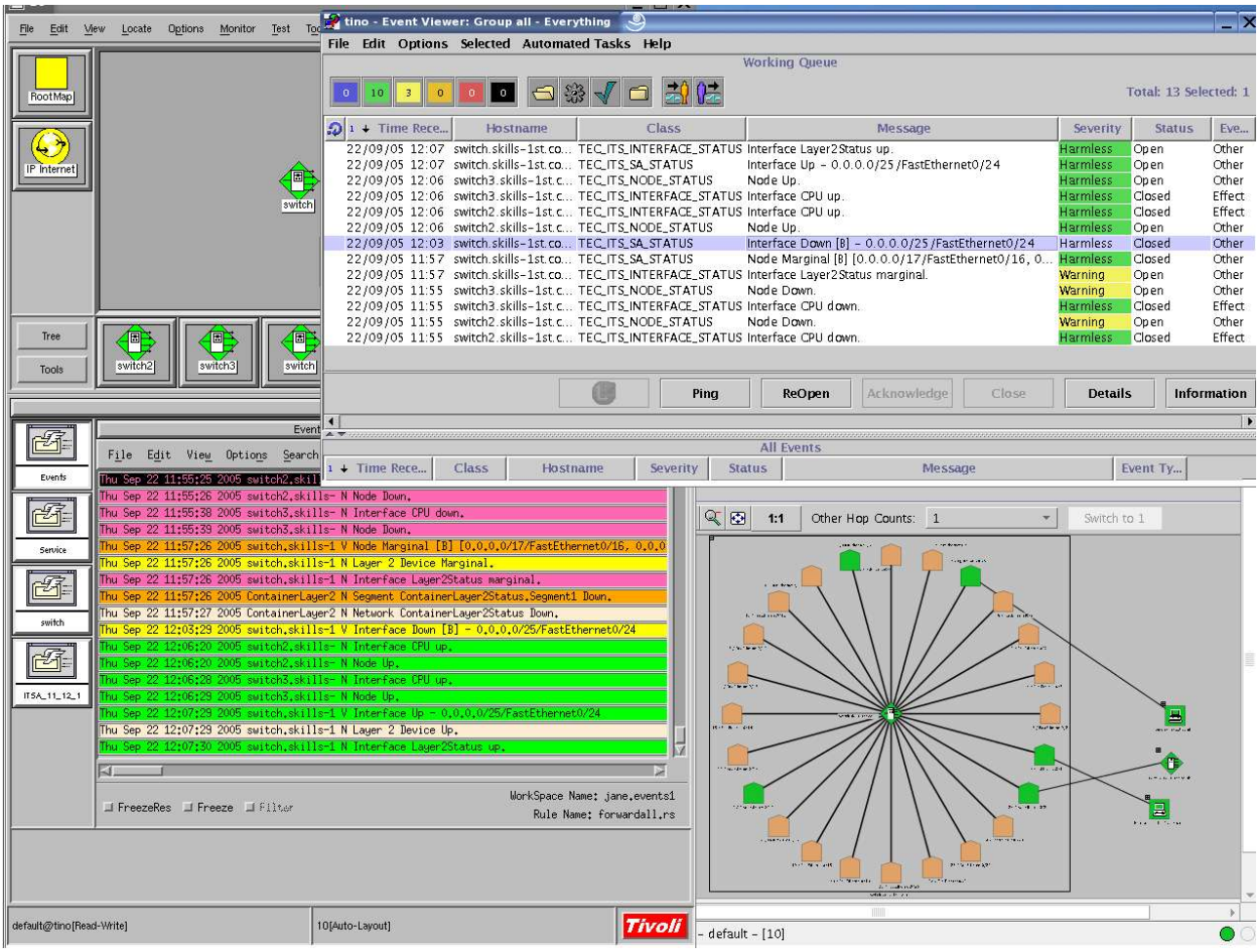

Figure 37 poppet, blossom and switch2 reconnected

In conclusion to this section, the rules delivered out-of-the-box in netview.rls, handle the correlation of NetView Layer 3 events and ITSA Layer 2 events very well.

6 Other useful ITSA commands

Here is a collection of useful commands for testing and checking ITSA, that have been used in the course of building this paper.. Consult the ITSA 1.3 Admin Guide for detailed information.

6.1 L2_lookup

Command resides in /usr/OV/ITSL2/bin. You can use it either to check what nodes are attached to what ports of a switch (ie. *switch* focused) or you can use it to check what port of a switch a particular node is connected to (ie. *node* focused).

L2_lookup

-s <switch> -o <switch OID> -p <port i/face index> -i <IP addr> -m <MAC addr> -M

Figure 38 L2_lookup examples

6.2 L2_topo_req.sh

/usr/OV/ITSL2/bin/L2_topo_req.sh is a command line utility to request a rediscover of a specific switch. It is the equivalent of the NetView menu item from Monitor -> Layer2 -> Rediscover

L2_topo_req.sh -s <NetView Selection Name> | -o <Object Id in NetView database>

Note that the ITSA 1.3 Admin Guide incorrectly documents this utility as being in /usr/OV/ITSL2 rather than in /usr/OV/ITSL2/bin .

6.3 /usr/OV/ITSL2/bin/ITSL2_reports

/usr/OV/ITSL2/bin/ITSL2_reports is a command line interface to producing the ITSA reports that can be accessed from the NetView menu items under Monitor -> Layer2. In addition to the Discovery Report and Status Report, there is also a Summary Report. These reports (whether generated by GUI or command) are populated from /usr/OV/ITSL2/cache/topo_cache. This cache file is updated from /usr/OV/ITSL2/cache/topo_db.out periodically – the interval is controlled by the **topo_cache_freq** parameter in correlator.ini, and defaults to 900 seconds.

A root user can add the "-d" parameter to ITSL2_reports to force an update of topo_cache. Note that the GUI menus include this "-d" parameter which is why a non-root user of the NetView GUI receives an error message at the bottom of the panel saying that /usr/OV/ITSL2/cache/topo_db.out cannot be removed – permission denied.

/usr/OV/ITSL2/bin/ITSL2_reports

-r summary	(summary report)
-r layer2	(discovery report)

-r status (status report)

-d (updates cache file)

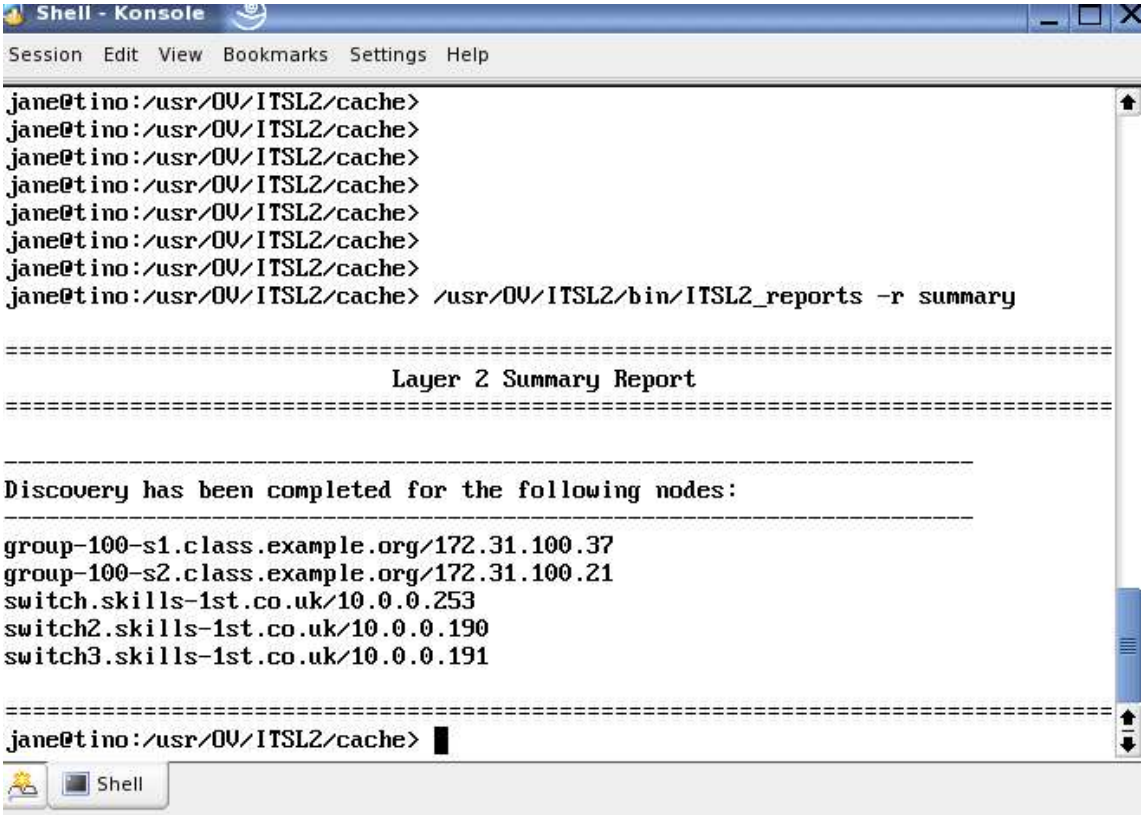

Figure 39 "ITSL2_reports -r summary" report

7 References

"IBM Tivoli Switch Analyzer 1.3 Admin Guide "– obtain from:
http://publib.boulder.ibm.com/tividd/td/SwitchAnalyzer1.3.html

"IBM Tivoli Switch Analyzer 1.3 Troubleshooting Guide" – obtain from:
http://www-306.ibm.com/software/sysmgmt/products/support/Field_Guides_Technical.html#SSGMPW7006128

IBM Tivoli NetView for Unix Administrator's Guide V7.1.4
http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?toc=/com.ibm.itnetview.doc/toc.xml

Acknowledgements

Several people have contributed advice and comments for this paper, to whom I am most grateful:

Rob Clark, IBM
Becky Anderson, IBM