State Model Diagrams – A Universal Runtime Network Management Tool

S P Maj (Corresponding author)
School of Computer Science and Security Science, Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia, 6050, Australia
Tel: 61-8-9370-6277 E-mail: p.maj@ecu.edu.au

Woratat Makasiranondh
School of Computer Science and Security Science, Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia, 6050, Australia
Tel: 61-8-9370-6670 E-mail: wmakasir@our.ecu.edu.au

D Veal
School of Computer Science and Security Science, Edith Cowan University
2 Bradford Street, Mount Lawley, Western Australia, 6050, Australia
Tel: 61-8-9370-6295 E-mail: d.veal@ecu.edu.au

Abstract

Modern networks are complex systems made up of diverse devices such as routers, wireless access points, and firewalls running many interacting protocols. Networks can be managed by a range of command-line interface (CLI) tools and/or Graphical User Interfaces (GUIs). While CLI tools are often preferred by network administrators, they do have some inherent disadvantages. Text based CLI outputs can be verbose. Furthermore the commands are sequential and it is therefore more difficult to navigate between different devices and protocols. GUIs assist but it can be problematic to concurrently display runtime data from multiple sources. In response to these problems State Model Diagrams (SMDs) were developed and implemented as a run-time network management tool.

Keywords: Run-time model, State Model Diagrams, Network management

1. Network Management

Network management is concerned with ensuring networks operate according to the defined Service Level Agreement (SLA) and hence underpin the operational expectations of an organization. An SLA is a description of the quality of services provided. SLA key performance indicators include network downtime, response time, end-to-end delay, packet loss and jitter. Network management has different definitions that depend upon the operational circumstance, however typical activities include: provisioning, operations, administration and maintenance. Provisioning is associated with configuring networked resources to provide a given service; administration involves resource tracking; operations are concerned with ensuring continuation of networked services; maintenance provides mechanisms for system recovery and upgrades. It is recognized that network management costs are significantly higher than equipment purchase costs (Broek & Looijen, 1997).

Network management models employ a layered architecture consisting of managed objects and managing devices. Alerts from the managed objects may result in managing devices, informing network administrators via e-mail notification, event logging or automatic system repair. There are two important network management protocols: Simple Network Management Protocol (SNMP) and Common Management Information Protocol (CMIP). Despite known limitations, such as the use of connectionless User Datagram Protocol (UDP) with un-acknowledged messages, SNMP is widely employed.

There are a range of simple and advanced Windows and Linux based tools for manually testing network connectivity that include: ping, traceroute, whois, nslookup, dig, netsat and nbstat, all of which are typically based on the Command Line Interface (CLI) with associated pull down menus. The syslog protocol allows network devices to automatically send unacknowledged notification messages to event message collectors called syslog servers. Windows and Linux based syslog servers typically provide a CLI with associated pull down menus. In addition, GUIs are available for displaying time dependent information such as Syslog statistics.

An operation network must be monitored to determine the availability of network resources, create trending reports and identify and correct issues in real time. Network monitoring is an essential aspect of an SLA. There are a variety of Windows and Linux based network monitoring tools all based on the Internet Control Message Protocol (ICMP) Echo request-reply (i.e. PING) that include: Spong, Nagios/Netsaint, Open NMS and NIMIS. These tools typically provide a CLI with associated pull down menu and also web based GUI. A variety of SNMP based tools, both Windows and Linux based are available that include: Multi-Router Traffic Grapher (MRTG); Round Robin Database Tool (RRDTool), Cacti and Cricket. These tools typically provide a CLI with associated pull down menu and also web based GUI.

Vendor based products for device configuration and management are also available such as Ciscoworks, Cisco netManager, Security Device Manager, WhatsUp Gold, OpenNMS, IBM Tivoli and Novell Zenworks. Such products are typically designed to provide entire systems solutions and hence address the problems associated with the convergence of different technologies such as Voice, wireless, data, public, private etc and also the persistent problem of network security.

Each of the different network management tools has strengths, weaknesses and limitations. Some of the more complex tools are immature and loosely bundled with integration being left to the customer (Paquet & Strovink, 1997). However, these tools are typically lacking in three important areas:

– It is difficult to concurrently display runtime data from multiple devices and protocols
– It is difficult to configure multiple devices at the same time
– The CLI is inherently sequential and requires multiple commands

Some complex security protocols such as IPSec require two (or more) devices to be configured in a coordinated manner. For example, both devices must use the same encryption/decryption method. The ability to concurrently display both devices is therefore important. Any non-trivial network will be a disparate collection of dedicated devices, each running its own protocols some of which interact with protocols on other devices. For example a router will have the Address Resolution Protocol (ARP) which automatically maps an IP address to an associated hardware address. ARP communicates with the routing protocol, such as the Routing Information Protocol (RIP), in order to redirect and encapsulate IP packets. Some protocols, such as RIP, are implemented on all routers in a network so they can share information to populate their individual routing tables. When a device is configured it must be done in the context of the other devices on the network in order to avoid conflicting parameters. Also a problem that occurs on a single critical device can quickly propagate through the entire network. In effect a problem on a device may be due to a fault on a remote device.

Efficient network management therefore means administrators need to be able to select either specific devices and/or protocols and concurrently view both the associated configuration and runtime data.

2. Command Line Interface/Graphical User Interface

Does the clock on your microwave oven always blink 12:00? Do you know how to set your DVD player in order to record your favorite television program when you are not home? If this sounds familiar, then you are probably experiencing what Donald A. Norman calls the gulf of execution and evaluation. In The Psychology of Everyday Things, Norman (Norman, 1988) applied cognitive psychology to the human/machine interfaces of many everyday things— telephones, doors, televisions, software— and found many of these things made it hard for users to do what they intended (execution); and these things often gave little feedback (evaluation) about what was actually done. So, the user is left confused: did it work? should I give up and just let the clock blink 12:00? For Norman, the best designs keep the gulf between execution and evaluation as small as possible and often this simply means:

– Make sure the user can figure out what to do.
– Tell the user what is going on.

By Norman's guidelines, and those of others (Furnell, 2007), (Johnston, Eloff, & Labuschagne, 2003), the venerable command line interface (CLI) is an exemplar of a wide gulf of execution and evaluation.

For example, using a CLI to configure and monitor a network essentially means selecting a specific device, such as a router, switch, or printer, and entering a sequence of commands. Each type of device may need its own complex and idiosyncratic incantation (Bartal, Mayer, Nissim, & Wool, 2004). Then, when the command has been executed, the user must interpret the results and decide what needs to be done next. How successfully this next course of action is selected will depend on the experience of the user; the skill of the device interface writer;

and the logging level (too much feedback can be as bad as too little). Also, the device-specific nature of many commands makes it difficult to concurrently display runtime data from multiple sources. The end result: it is easy to innocently misconfigure a device (Wool, 2004).

As an alternative to the CLI, some vendors offer web-based consoles that provide runtime visualization and monitoring of either parts or the whole of a network. There are relative advantages and disadvantages in both interface methods (table 1). The CLI is less resource intensive allowing fine-grained control however the CLI command output can be verbose. The major disadvantage of the CLI is that it is not possible to concurrently display protocol device status. However many administrators still consider the CLI to be more reliable, robust, accurate and trustworthy than a GUI (Takayama & Kandogan, 2006).

3. State Model Diagrams

In response to some of the shortcomings of existing network management tools, like the gulf of execution and evaluation when using a CLI, a novel method has been developed for looking at and monitoring networks: State Model Diagrams (SMDs) (Maj, Kohli, & Murphy, 2004; Maj & Tran., 2006). However, 'model' is an overused word in computer science, so some clarification is needed first. Tom DeMarco gives a good explanation when he calls a model:

> A miniature representation of a complex reality. A model reflects certain selected characteristics of the system it stands for. A model is useful to the extent that it portrays accurately those characteristics that happen to be of interest at the moment. (DeMarco, 1982)

A model is almost guaranteed to be incomplete because it is an abstraction of the system it represents. This incompleteness is not a bad thing if the model includes only the details that are relevant to the task at hand. Then, the model concentrates our thinking and forces our ideas out into the open where we can at least talk about and share them.

So, a model is a convenient representation (in words, numbers, or other symbols) of some real-world system. When modeling networks and their communications protocols, this convenient representation is usually a finite state machine described using state transition diagrams, extended event-state tables, or high-level structured programs. These types of models are low-level, solution-oriented ways of showing the way network protocols interact. However, to see the critical system-wide perspective, we need more.

SMDs provide this bifocal view of a network, while also narrowing the execution and evaluation gulf. For example, a single SMD consolidates the command line output from multiple commands into tables on a single diagram. Each table has a dedicated purpose: to do this, or to do that (Maj, Kohli, & Fetherston, 2005). The tables are modular, meaning a network model can be built in a top-down fashion by proceeding from the known to the unknown, down to the level of detail that makes most sense.

For instance, the helicopter-view, 'level zero diagram' represents the topology map of an entire network showing how the different network devices are connected along with their individual IP addresses. Administrators can check the state of a specific device without sacrificing altitude. It is then possible to obtain further detail, as needed, by opening subsequent SMD levels. In essence, these diagrams present a network as an easily-digestible abstraction as opposed to the unfiltered, raw view offered by CLI. By means of hierarchical top down decomposition it is possible to partition a complex network into units of an amenable size. It is therefore possible to view the entire network or obtain increasing levels of detail whilst maintaining links and interfaces to each level.

A level zero SMD consists of the overall topology of the network. A given device can be selected and expanded to show protocols arranged according to the Open Systems Interconnection Reference Model (OSI) (Figure 1).

– OSI layer 1 (physical): shows the status of the interfaces – the line may be up or down.

– OSI layer 2 (data link): shows the details of each interface hardware address and line protocol – the line protocol may be up, down or manually disabled.

– OSI layer 3 (network): shows what protocols (ARP, routing, Security) the device is using. Any of the protocols can be interrogated to show more details. For example, expanding the ARP table shows that the routing protocol (RIP) has been configured (Figure 2).

Significantly it is possible to concurrently monitor interactions between all the different protocols. For example, it is possible to observe: interface Fa0/0 is operational; Fa0/0 MAC address; Fa0/0 IP address; and that the router has learnt about two remote networks (192.168.2.0 and 192.168.20.0) by means of RIP that are reached via interface Fa0/0. To obtain this data would require at least four different CLI commands. In effect a single SMD

diagram may be used instead of multiple, sequential and often verbose text-based CLI commands. This is particularly important because it is then possible to concurrently observe the state of every protocol and the associated protocol interactions not only within a single device but all devices on the network. Significantly the SMD method represents a standard universal template that may be used for all network devices and protocols.

Both configuration and run-time validation may be considerably improved by viewing concurrent images of devices and protocols. SMDs are intrinsically concurrent allowing the user to view intra-device protocols and their relationships. Using SMDs it is also possible to concurrently view the same protocol, such as RIP, operational across populations of devices. A device may then be configured in the context of the associated configurations of other devices thereby significantly reducing the possibility of conflicting parameters such as address duplication. Furthermore run-time validation may be considerably improved.

3.1 Network Security protocols and devices

Managing network security devices and protocols is typically accomplished by tools in the Internet Protocol Security (IPsec) suite. This suite provides authentication, confidentiality, and data integrity by allowing network peers, such as routers, firewalls, or hosts, to cooperatively exchange cryptographic keys so that traffic between them can be encrypted. However, the CLI output from IPsec is notoriously verbose and complex. In Norman's terms, the gulf of execution and evaluation is wide.

Even so, it is possible to model IPsec using SMDs (C Nuangjamnong, Maj, & Veal, 2007). SMDs capture IPSec configurations in a series of tables each of which can be expanded or collapsed as needed. A router may be configured for the IPSec protocol. There are four main aspects to this configuration: keys; encryption algorithms; Access Control Lists (ACLs) and interface mapping (Figure 3) the details associated with other protocols (routing and ARP) have been minimized in this diagram. It is then possible to selectively open each of the four IPSec functions and minimize other functions such as the interface details associated with OSI layers 1 and 2. Thus, on a single screen it is possible to observe the configuration details of the IPSec protocol.

For IPSec to operate correctly, the peer IPSec router must be configured so that on both routers the encryption algorithms are identical; the keys and key exchanges are complimentary and also that the ACLs are mirror images. Using SMDs it is possible to concurrently display both devices side-by-side so their configuration details can be cross-checked. This is especially important in fault diagnosis.

Based on a heuristic evaluation method Nielson developed criteria for a successful human interface (Nielsen & Molich, 1990). These criteria may be used to evaluate security related interfaces. To address issues specific to interfaces for security purposes Johnston proposed criteria for a security Human Computer Interface (HCI-S) (Johnston et al., 2003). HCI-S criteria are defined as: convey features; visibility of system status; learnability; aesthetic and minimalist design; errors; satisfaction and trust. SMDs meet these criteria. The SMD user interface represents an intermediate between the CLI and the new Security Device Manager (SDM) GUI (S. P. Maj, Makasiranondh, W., Veal, D., 2010). The use of the SMD method has been favorably received by practicing professionals (S. P. Maj, Veal, D., 2010a). Unlike the Security Device Manager GUI, the SMD representation can also be used to manage secure layer 2 switches (S. P. Maj, Veal, D., 2010b).

3.2 Low-level SMDs

Unlike other GUIs, the SMD method may also be used to represent network protocol behavior down to the detailed level of a simplified state-machine or automaton. It is therefore possible to capture protocol entity states not only for an entire network system in equilibrium but also the transitions between defined protocol states. For example ARP automatically obtains the mapping between an IP address and a hardware address. When ARP is modeled as an automaton, it is an entity that can exist in three possible states - free, pending and resolved. In the free state there is no mapping between the IP and MAC address. In the pending state the device has an IP address and, for communication to occur, this must be mapped to an associated physical address. The pending state is maintained until either a reply is received or a time out occurs. When a reply is received the entity changes from the pending to the resolved state.

State changes are typically observed, especially during fault diagnosis, using the CLI command called debug. The debug command is problematic for a number of reasons: debug has a high CPU priority and can potentially therefore render the entire system unusable; debug output can be extremely verbose, which makes it difficult to track state transition changes. It is strongly recommended that debug commands are only used during periods of low network traffic. It is possible to use SMDs to monitor some transitions in runtime and hence represent an alternative method of fault diagnosis.

4. SMD run-time software

The original SMD diagrams, paper based and completed by hand, were evaluated as a network management tool

and found to be as useful as the CLI for all aspects of network management, and significantly, more useful than the commercial product Ciscoworks used in these trials (Maj & Tran., 2006). Extensive trials have demonstrated significant pedagogical advantages to using SMDs (paper based) as a teaching tool for both novices and practicing professionals. The SMD model has been converted to a run-time model, written in C#, called Sopwith that builds and populates the SMDs using PING and SNMP requests.

SNMP is a client-server application layer protocol with three main components: managed devices (a switch, router, firewall or PC), software agents on the managed devices, and a network manager. The software agents on each device will gather specific information about the state of the device into data structures called Management Information Bases (MIBs), which the network manager can interrogate. Sopwith starts with a list of known SNMP network devices. This list includes the IP address and the SNMP authentication information for each device. (The file that stores this information must be protected for security reasons). SNMP is a standard method for network devices management and therefore does not represent an undue computational overhead. With this initial information, Sopwith proceeds as follows:

– The IP addresses in the interface tables of each device are used to build a list of the sub-networks.
– A PING is sent to each address in the sub-networks. From the responses, Sopwith builds a list of attached devices and hosts.
– Sopwith fetches the freshly populated ARP table from each device and uses this to construct a list of network connections.

There are four possible actions with the current version of Sopwith, namely – load network, network status update, retrieve device information and modify SMD tables (Figures 4, 5). In this way, Sopwith builds a topology map of the entire network of devices and connections. Users can then drill up and down through the map to display just those operational protocols and interactions needed for the task at hand. Sopwith updates its data on a one second cycle. This period could be changed if the user felt a need to change the balance between network usage and timeliness of data. The network traffic generated by the PING commands and SNMP requests is kept to a minimum because Sopwith only fetches data that will be displayed and this is limited by the user hiding data that is not relevant. The current implementation uses SNMP version 1, which can fetch only one row of a table per fetch cycle. This would be a problem with large tables that could be solved by changing to SNMP version 2 and fetching each table with one request.

One of the key uses of Sopwith is fault diagnosis. When a network interface in a device develops a fault, it's on-screen representation changes from black to red. If a device drops off the network, the failed link is shown in red along with the output of the PING command that discovered the problem. These types of errors are flagged at each level of the SMD model so that users can drill down to the source of a problem.

Sopwith allows the user to add sub diagrams and tables to the SMD so that groups like the IPSec group can be constructed within Sopwith as it runs. An interface is provided to enable configuration of the tables. The user is able to specify:

– The SNMP Object Identifier (OID) for an SNMP table.
– Which table columns to display.
– How to name the columns.
– Conditions indicating that a row should not be shown.
– Which, cell values for a column represent error conditions so that network device errors can be color coded in the live diagram.

The diagram and table specifications are stored in an XML file so that the user may create an SMD that is relevant to their own needs. This means that any device that supports SNMP can be investigated via an SMD. One of the main difficulties is in finding the OID's to use when specifying the tables. Sopwith includes a MIB file browser to help the user with this.

Unlike other network management tools, Sopwith lets users decide what is relevant for their circumstances. For example, they can create customized SMD tables and diagrams that group information, display or hide particular fields, and they can decide what data values imply an error condition. The control object (ctrl) sends a request to the SNMP object for specific device information. SNMP object then responds to the request. The control object updates and populates the SMD tables in XML format.

This implementation of Sopwith was written in C# using an SNMP library put together and wrapped in C#. SNMP libraries also exist for C, C++ and Java. However, the SMD could have been implemented in any

language giving support to TCP/IP for communicating with the SNMP devices and a GUI API for drawing the diagram

5. Conclusion and further work

Previous work has clearly demonstrated that it is possible to model all network devices and protocols with SMDs in the original paper form. Sopwith has demonstrated that the SMD principles can be implemented in software, however further software development is needed. Evaluations of Sopwith indicate:

– Easy to use and make the interactions between devices and protocols explicit.
– Modular structure is intuitive and allows the depth of view into the network to be adjusted to the task at hand.
– Complexities of the CLI are abstracted away.

Sopwith is still under active development with the following major enhancements planned. At present the data displayed in the SMD tables is read-only but in later versions users will be able to configure the devices they see. To better show the interactions between devices, later versions will allow populations of different devices and protocols to be displayed on multiple screens. There are considerable pedagogical advantages to using SMD based software not only for novices but also for managing complex security protocols (S. P. Maj, Veal, D., Yassa, L., 2010). SMD modeling principles are in the process of being incorporated into the Cisco Network Academy Program (CNAP) network simulation and visualization tool, Packet Tracer. CNAP is the world's largest network curriculum program with over 600,000 students in 11,000 Cisco Networking Academies in 162 countries. CNAP offer a comprehensive range of 18 different awards suitable for both novices and practicing professionals. The SMD software will run as an external application to Packet Tracer using the Java Application Program Interface (API).

References

Bartal, Y., Mayer, A., Nissim, K., & Wool, A. (2004). Firmato: A novel firewall management toolkit. *ACM Trans. Comput. Syst., 22*(4), 381-420.

Broek, F., & Looijen, M. (1997). Mangement of International Networks. *International Journal of Network Management, 7*(5), 342-250.

DeMarco, T. (1982). *Controlling software projects*. New York: Yourdon Press.

Furnell, S. (2007). Making security usable: Are things improving? *Computers & Security, 26*, 434-443.

Johnston, J., Eloff, J. H. P., & Labuschagne, L. (2003). Security and human computer interfaces. *Computers & Security, 22*(8), 675-684.

Maj, S. P., Kohli, G., & Fetherston, T. (2005). *A pedagogical evaluation of new state model diagrams for teaching internetwork technologies*. Paper presented at the 28th Australasian Computer Science Conference (ACSC2005).

Maj, S. P., Kohli, G., & Murphy, G. (2004). *State models for internetworking technologies*. Paper presented at the IEEE 34th annual conference on Frontiers in education.

Maj, S. P., Makasiranondh, W., Veal, D. (2010). An Evaluation of Firewall Configuration Methods. *IJCSNS International Journal of Computer Science and Network Security, 10*(8), 1-7.

Maj, S. P., & Tran., B. (2006). *State Model Diagrams - a systems tool for teaching network technologies and network management*. Paper presented at the International joint conferences on Computer, Information and Systems sciences, and Engineering.

Maj, S. P., Veal, D. (2010a). An Evaluation of State Model Diagrams for Secure Network Configuration and Management. *IJCSNS International Journal of Computer Science and Network Security, 10*(9), 66-72.

Maj, S. P., Veal, D. (2010b). Using State Model Diagrams to Manage Secure Layer 2 Switches. *IJCSNS International Journal of Computer Science and Network Security, 10*(9), 141-144.

Maj, S. P., Veal, D., Yassa, L. (2010). A Preliminary Evaluation of the new Cisco Network Security Course. *IJCSNS International Journal of Computer Science and Network Security, 10*(9), 183-187.

Nielsen, J., & Molich, R. (1990). *Heuristic evaluation of user interfaces*. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people.

Norman, D. A. (1988). *The psychology of everyday things*. New York: Basic Books.

Nuangjamnong, C. (2009). *An Investigation into Network Management.* Edith Cowan University, Perth.

Nuangjamnong, C., Maj, S. P., & Veal, D. (2007). *Network security devices and protocols using state model diagrams*. Paper presented at the 5th Australian information security management 2007.

Paquet, R., & Strovink, K. (1997). *The risks of network and systems management*.

Takayama, L., & Kandogan, E. (2006). *Trust as an underlying factor of system administrator interface choice*. Paper presented at the CHI '06 extended abstracts on Human factors in computing systems.

Wool, A. (2004). The use and usability of direction-based filtering in firewalls. *Computers & Security, 23*, 459-468.

Table 1. CLI versus GUI (Source: (C. Nuangjamnong, 2009))

	CLI	**GUI**
Characteristics	It is a simple, quick, and functional interface. The only tools necessary are a shell prompt, some service specific commands, and a simple text editor.	Interface is easier for the novice administrator. The GUI tool often lacks completeness or efficiency.
Ease of Use	Demands skill and experience to master all the possible command and option choices.	Most users are familiar with GUIs and navigating using a mouse.
Control	More fine-grained control of the file system and operating system are possible.	May require extra authorizations to access some functions.
Multitasking	Multitasking is possible but only a single view is possible at any one time.	Overlapping windows allow users to view, control, and manipulate multiple operations.
Speed	A command line interface often only needs to execute a few lines to perform a task. For an advanced command line interface users, CLI would be able to get something done faster than GUI.	Using a mouse and/or keyboard to navigate and control the operating system is much slower than working in a command line environment.
Low resources	Uses less system resources.	Requires more system resources because of elements that need to be loaded such as icons, fonts, and drivers.
Scripting	A user can easily script a sequence of commands to perform a task or execute a program.	Users need to create shortcuts, tasks, or other similar actions to complete a task.
Remote access	Provides low-bandwidth access to remote systems.	All computers and particularly network equipment have this capability under web-based GUI.

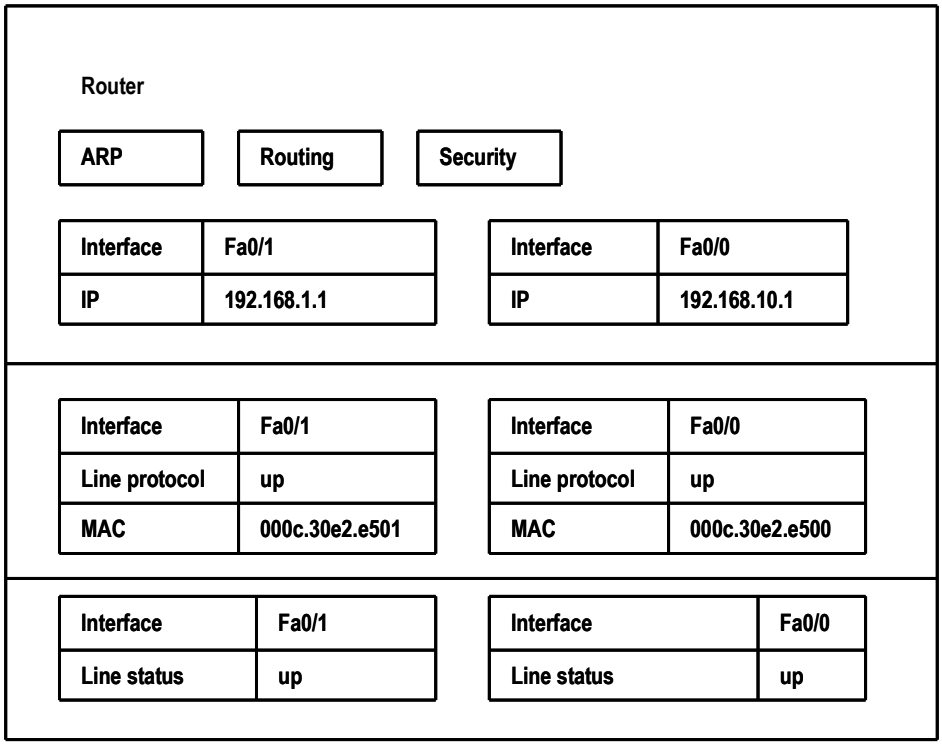

Figure 1. Level 1 State Model Diagram

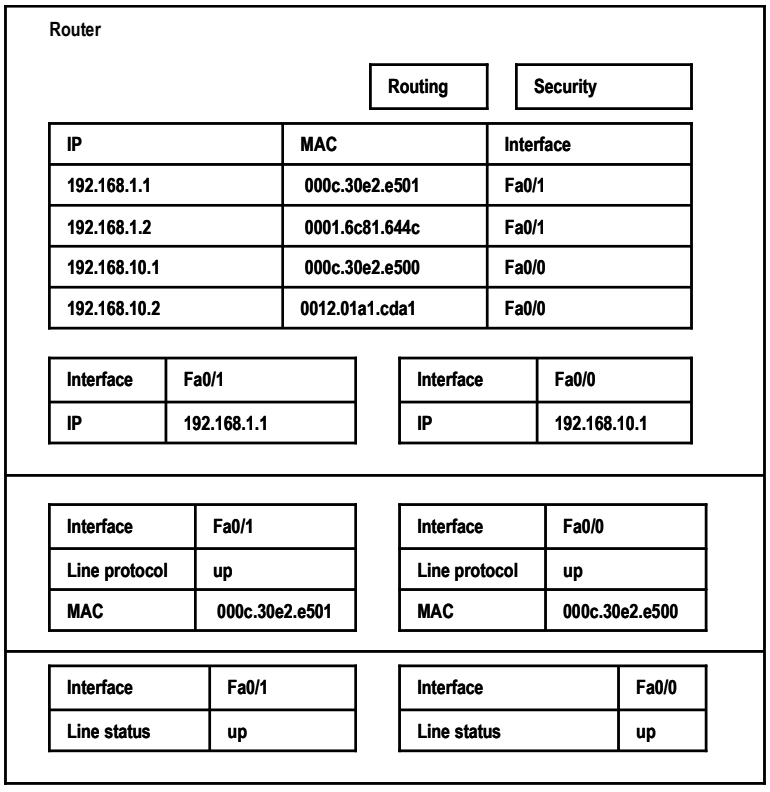

Figure 2. ARP Table expanded

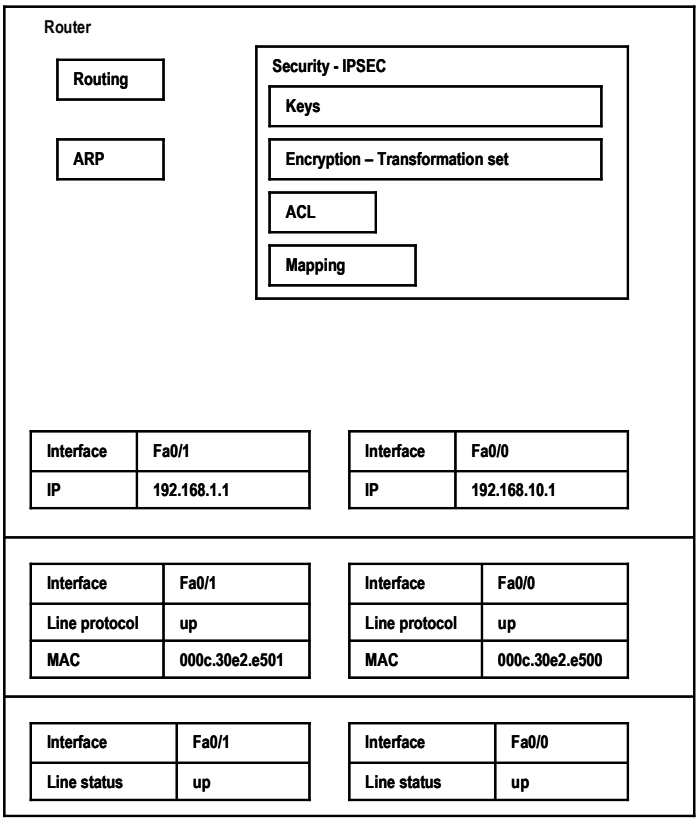

Figure 3. IPSec on router

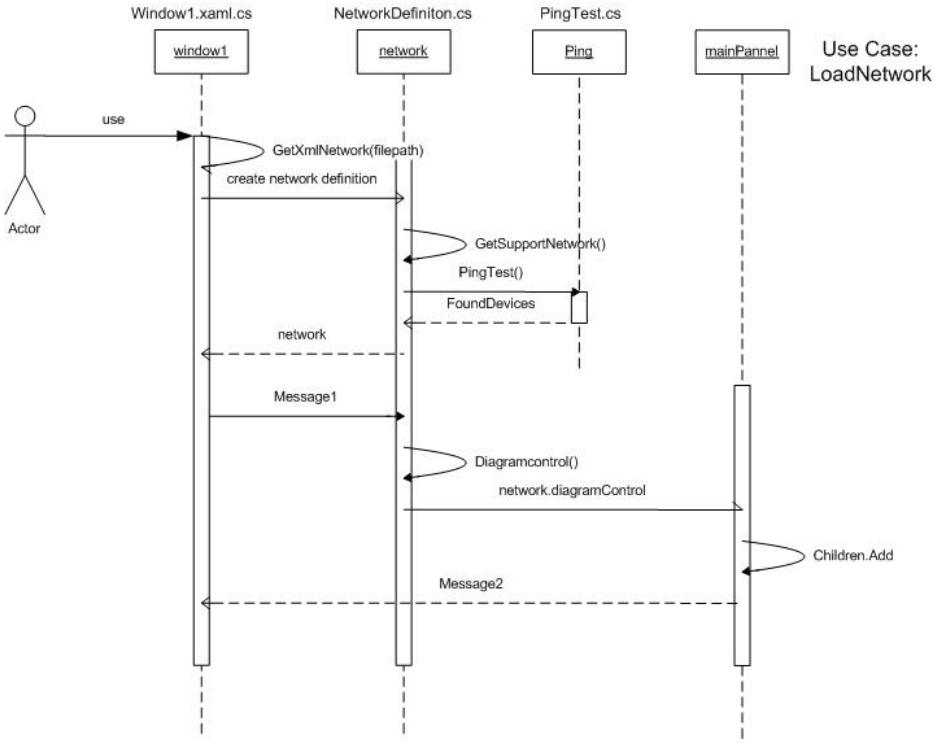

Figure 4. UML shows step when Sopwith explore the network

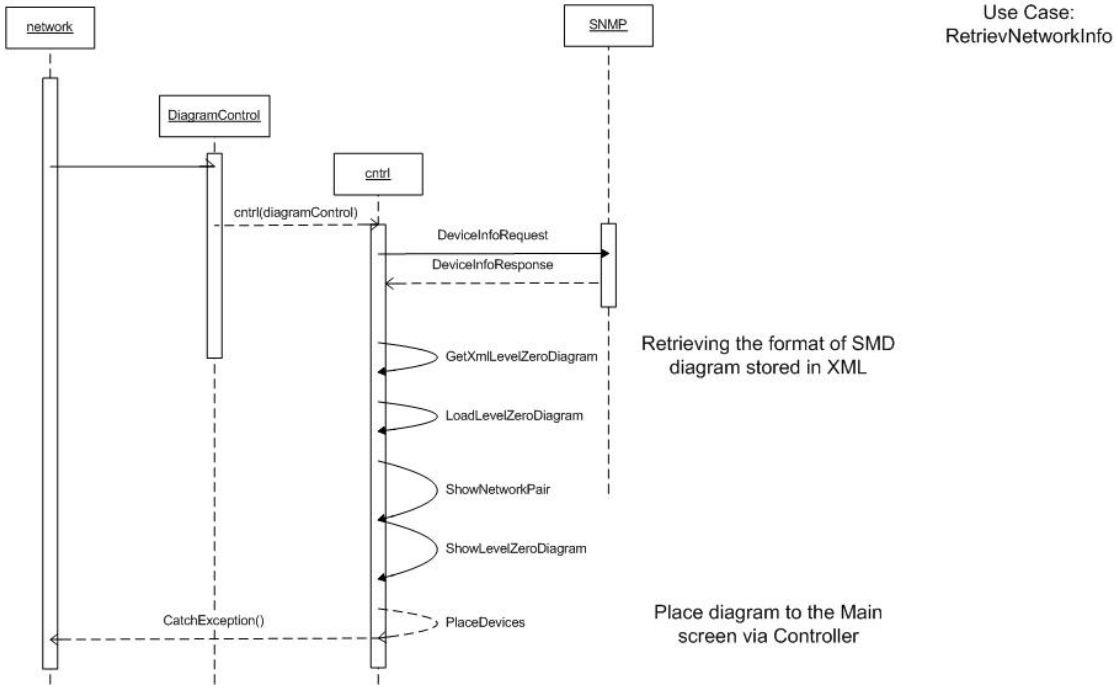

Figure 5. UML shows steps when Sopwith retrieve and layout device information on SMD