Detecting Embedded Devices using Network Discovery

Rajula Aravinda Reddy, J C Narayana Swamy, R Govardhana Reddy

Abstract— Modern heterogeneous networks present a great challenge for network operators and engineers from a management and configuration perspective. The Network Discovery System is a network management framework that addresses these challenges. NDS offers centralized network configuration management functionality, along with providing options for extending the framework with additional features. The devices managed by NDS are stored in its Configuration Database (CDB). However, currently there is no mechanism for automatically adding network devices to the configuration of NDS, thus each device's management parameters have to be entered manually. The goal of this paper is to develop a software module for NDS that simplifies the process of initial NDS configuration by allowing NDS to automatically add network devices to the ND CDB

Apart from developing the software module for discovery, this paper aims to summarize existing methods and to develop new methods for automated discovery of network devices with the main focus on differentiating between different types of devices. A credential-based device discovery method was developed and utilized to make advantage of known credentials to access devices, which allows for more precise discovery compared to some other existing methods. The selected methods were implemented as a component of NDS to provide device discovery functionality. Python language is used as tool to develop code.

Index Terms— Discovery Protocols, Embedded Device Discovery, Embedded Device Configuration, Network Discovery System, Network Management.

I. INTRODUCTION

Network management became a non-trivial task, as networks grew and incorporated different types of devices. Manual network management of large scale networks is unfeasible due to the need for engineers specialized in different aspects and types of network devices and their management, limited time, need to define a strategy for configuration management, and the effort to track the configuration state of large number of different devices. These factors obviously increase the costs and effort required for network management. To overcome these difficulties Network Discovery system (NDS) were developed. An NDS is a tool for network operators and engineers. Such a tool enables centralized configuration management of many different network devices, consolidates the storage of device configuration and state information, pushes and pulls the configuration changes to and from the devices.

Additionally, an NMS may provide visualizations of the network topology and state, provide an alert system with notifications, and automate network service deployment. However, designing and developing such a system is a non-trivial task. The reasons for this are the presence of different vendors in the network equipment market, the existence of different configuration interfaces to these network devices, and the lack of a single generic interface for configuration management. Designing and developing such interfaces for every possible device type becomes impractical due to both the number of different devices and the evolution of their interfaces. A possible solution is to define a model for the device configuration structure which serves as a structured representation of the device configuration and a protocol which provides an interface to access this model and to perform configuration changes.

The rest of the paper is originated as follows: section II discusses relevant background and summarizes the results of our literature study; section III describes the implementation of the network device discovery component; and section IV concludes the thesis project, discusses the goals which have been reached, suggests future work, and describes the ethical considerations related to this project.

II. BACKGROUND OF NETWORK DISCOVERY

Several topics covered in our literature study are discussed in this chapter. Protocols, as these are essential to understand any work in the area of network management. As one of the goals of this thesis project is to develop a device discovery component for NDS

A. Network Management Protocols

Network Management Protocols define a means to access the configuration data of a managed device, as well as a means to change this data. This section will focus on the protocols which are essential in the context of working with NDS, namely: ICMP, SNMP, NetBIOS, and TCP, Full
These methods actively scan predefined networks and probe IP addresses.

1) ICMP

The Internet Control Message Protocol (ICMP) is one of the main protocols of the Internet Protocol Suite. It is used by network devices, like routers, to send error messages indicating, for example, that a requested service is not available or that a host or router could not be reached. ICMP can also be used to relay query messages. ICMP differs from transport protocols such as TCP and UDP in that it is not typically used to exchange data between systems, nor is it regularly employed by end-user network applications (with the exception of some diagnostic tools like ping and trace route). This method detects active hosts on a network by sending ICMP echo request packets and listening for ICMP echo responses. The ICMP discovery is a simple and fast discovery that detects whether an IP address exists or not. It returns only the IP address and MAC address of a detected host.

Manuscript received April, 2014.

Rajula Aravinda Reddy, M.tech in VLSI Design and Embedded Systems, Bangalore Institute of Technology, Bengaluru, Karnataka, India.

J C Narayana Swamy, Associate Professor, Department of Electronics and Communication Engineering, Bangalore Institute of Technology, Bengaluru, Karnataka, India.

R Govardhana Reddy, Senior Technical Lead, Communication Business unit, TATA-Elxsi, Bengaluru, Karnataka, India

The ICMP packet format is shown in below figure

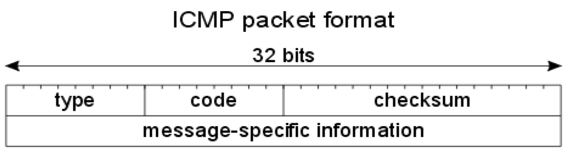


Figure 1:- ICMP packet Format

2) *SNMP*

SNMP is a protocol for network management developed in late 1980s. The standard describes a protocol for exchanging management data between a management station and a managed device, the structure of a MIB, and a simple network management system architecture.

a) Understanding of SNMP

You can use SNMP (Simple Network Management Protocol) to manage network devices and monitor their processes. An SNMP-managed device, such as a NIOS appliance, has an SNMP agent that collects data and stores them as objects in MIBs (Management Information Bases). The SNMP agent can also send traps (or notifications) to alert you when certain events occur within the appliance or on the network. You can view data in the SNMP MIBs and receive SNMP traps on a management system running an SNMP management application, such as HP OpenView, IBM Tivoli NetView, or any of the freely available or commercial SNMP management applications on the Internet.

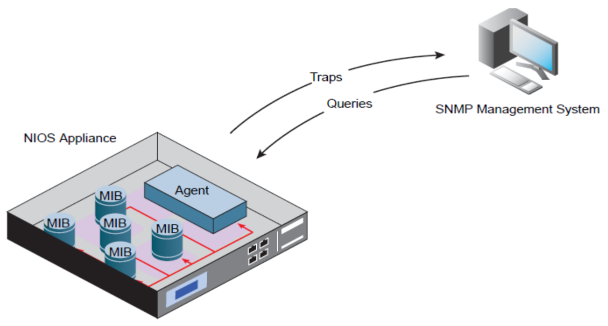


Figure 2: - SNMP Overview

b) About SNMPv1 & SNMPv2

SNMPv1 is the initial implementation of SNMP. It operates over protocols such as UDP (User Datagram Protocol) and IP (Internet Protocol). SNMPv2 includes improvements in performance and security. It adds new protocol operations such as GetBulk and Inform, which allow the management system to request larger blocks of data from the agent. Both SNMPv1 and SNMPv2 use common strings that are sent in clear text to authenticate clients.

The NIOS appliance supports SNMPv1 and SNMPv2 in which the SNMPv2 agent acts as a proxy agent for the SNMPv1 management systems. When a SNMPv1 management system sends a query to the appliance, the SNMPv2 proxy agent forwards the request to the SNMPv1 agent. The proxy agent maps the SNMPv1 trap messages to the SNMPv2 trap messages, and then forward the messages to the management system.

3) NetBIOS

NetBIOS (Network Basic Input/output System) is a program that allows applications on different computers to communicate within a local area network (LAN). It was created by IBM for its early PC Network, was adopted by Microsoft, and has since become a defacto industry standard. NetBIOS is used in Ethernet and Token Ring networks and, included as part of NetBIOS Extended User Interface (NetBEUI), in recent Microsoft Windows operating systems. It does not in itself support a routing mechanism so applications communicating on a wide area network (WAN) must use another "transport mechanism" (such as Transmission Control Protocol) rather than or in addition to NetBIOS

The NetBIOS method queries IP addresses for an existing NetBIOS service. This method detects active hosts by sending NetBIOS queries and listening for NetBIOS replies. It is a fast discovery that focuses on Microsoft hosts or non-Microsoft hosts that run NetBIOS services. This method returns the following information for each detected host such as IP address, MAC address, OS, and NetBIOS name (A NetBIOS name is a 16-byte address that is used to identify a NetBIOS resource on the network)

4) TCP

The Transmission Control Protocol (TCP) is one of the core protocols of the Internet protocol suite (IP). TCP provides reliable, ordered and error-checked delivery of a stream of octets between programs running on computers connected to a local area network, intranet or the public Internet. In this discovery it return the IP address, MAC address, and OS

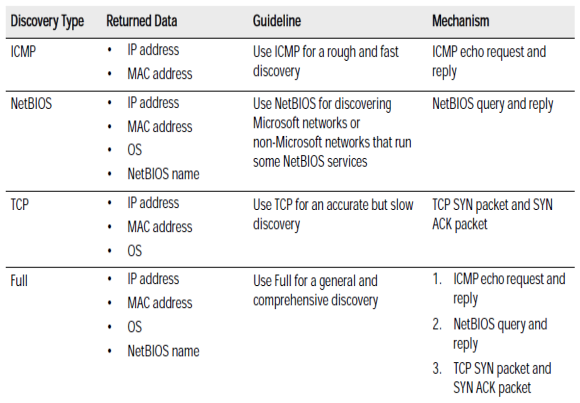
TCP is known as a connection-oriented protocol, which means that a connection is established and maintained until such time as the message or messages to be exchanged by the application programs at each end have been exchanged. TCP is responsible for ensuring that a message is divided into the packets that IP manages and for reassembling the packets back into the complete message at the other end. In the Open Systems Interconnection (OSI) communication model, TCP is in layer 4, the Transport Layer.

5) Full

The full discovery method is a combination of an ICMP discovery, a NetBIOS discovery, a TCP discovery, and a UDP scan. This method starts by sending an ICMP echo request. If no IP address on the network responds to the ICMP request, the discovery ends. If there is at least one response to the ICMP echo request, a NetBIOS discovery starts. A TCP discovery then follows by skipping through the active hosts that the NetBIOS discovery detects. The TCP discovery also handles the NetBIOS-detected hosts that have no MAC addresses. This method also performs a UDP scan to determine which UDP ports are open. The full discovery method returns the following information for each detected host:

• IP address • MAC address • OS • NetBIOS name

The following is a summary of the supported IP discovery methods:



Discovery Type	Returned Data	Guideline	Mechanism
ICMP	• IP address • MAC address	Use ICMP for a rough and fast discovery	ICMP echo request and reply
NetBIOS	• IP address • MAC address • OS • NetBIOS name	Use NetBIOS for discovering Microsoft networks or non-Microsoft networks that run some NetBIOS services	NetBIOS query and reply
TCP	• IP address • MAC address • OS	Use TCP for an accurate but slow discovery	TCP SYN packet and SYN ACK packet
Full	• IP address • MAC address • OS • NetBIOS name	Use Full for a general and comprehensive discovery	1. ICMP echo request and reply 2. NetBIOS query and reply 3. TCP SYN packet and SYN ACK packet

Figure 3:- Table for Different Discovery Protocols

B. Overview of Network Discovery

The appliance provides discovery tools for detecting active hosts on predefined networks and on specified servers. You can use the discovery feature to obtain and manage information about your network hosts. Depending on which discovery method you use, the appliance returns information, such as IP addresses, MAC addresses, and operating systems, about the detected hosts and virtual entities

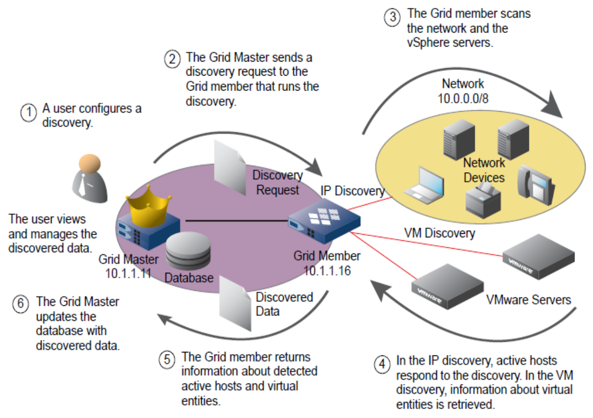

③ The Grid member scans the network and the vSphere servers.
② The Grid Master sends a discovery request to the Grid member that runs the discovery.
① A user configures a discovery.
Network 10.0.0.0/8
Network Devices
Discovery Request
IP Discovery
The user views and manages the discovered data.
VM Discovery
Grid Master 10.1.1.11
Database
Grid Member 10.1.1.16
Discovered Data
VMware Servers
⑥ The Grid Master updates the database with discovered data.
⑤ The Grid member returns information about detected active hosts and virtual entities.
④ In the IP discovery, active hosts respond to the discovery. In the VM discovery, information about virtual entities is retrieved.

Figure 4: - Overview of Discovery Process

NDS is comprised of three major parts: the service manager, the device manager, and the Configuration Database (CDB). Figure 3 provides a representation of NDS overview. The device manager functions as an interface to managed devices. It provides a means to add a new device to the CDB (a device may be added manually from a device template or from an existing device configuration); to deploy configuration changes to managed devices in a fail-safe way, so that the changes can be applied simultaneously to any number of devices; to perform a configuration rollback and to synchronize the NDS running the CDB with the actual configuration of the managed devices, which works in two ways (both to and from the devices).

III. NETWORK DEVICE DISCOVERY IMPLEMENTATION

The first task for this master's thesis project was to develop a module for NCS which performs network device discovery. The module should be able to identify which devices in the specified address space are alive and be able to provide some information about the device, such as device vendor, device model, and OS running on the device. This chapter will describe the requirements defined for the device discovery module and the process of developing this module.

A. Network Device Discovery Module Description

An important part of the requirements for the module was to focus on the devices that can be managed by NDS, as opposed to finding all devices that are online. Also, as the module is a legitimate search tool, it was assumed that network security systems are properly configured to allow the operation of this tool, therefore the solution proposed should not contain any techniques of firewall, or other network security systems bypass. Since the discovery process is assumed to be run by the network administrator or in coordination with the network administrator, it is also assumed that the user of the network device discovery module knows the credentials to access the legitimate devices.

B. Data Model Description

The data model of the network device discovery component augments the NDS data model and adds a module that defines a set of actions with associated input and output parameters as well as a structure for storing the data collected when a specific action is executed. An action, in this context, is an interface to call and execute specific code defined for that action.

The input parameter for the discovery launch actions is a definition of the discovery target, which may be a single IP address (e.g. 10.0.0.24) an IP subnet specification (e.g. 10.0.1.0/28). The output parameter for these actions is a status line which indicates if the run was successful or if there were any errors. During the discovery process reports the discovery status, such as Completed, Running, Paused, Stopped, or Error. The output parameter for each of the actions is a status line which is used to inform the user as to whether the operation was successful or some error occurred.

C. Implementation based on NetMRI

The first version of the network device discovery module was essentially a wrapper for NetMRI. This module operates as follows: receive the input parameters, execute NetMRI with predefined flags and the specified target, wait for NetMRI to finish, parse the results of the NetMRI scan, and store the results in the NetMRI database according to the model. The output generated by the module contains informational messages, such as the number of devices found and the success or failure of the execution. The execution is considered successful when there are no errors while running NetMRI and no errors while processing the results or loading them into the database, even if no devices are found in the specified address space.

After NetMRI finishes the scan, the device discovery module processes the XML file produced by NetMRI and fetches the scan results. These results are then stored in memory for further processing. For this purpose a Device class was defined which consists of a set of properties describing a network device found during the device discovery process. Additionally, this class contains methods for storing and loading the information about the device into the NDS database according to the defined data model.

When the list of discovered devices is fetched, the latest-run sub-tree of the data model is replaced with the new list of discovered devices. The information is stored as operational data, which means that it will be reset when NDS is restarted. This information is displayed to the user and can be further processed by another application

1) *Description of NetMRI*

As networks become larger and more dynamic, the need for detailed, up-to-date network device information becomes more critical. Many organizations are hindered by manual documentation and the lack of detail provided with simple ping sweeps. The NetMRI Discovery Module automates normal manual tasks and provides detailed views lacking in other tools.

The NetMRI Discovery Module also ensures you have the most current information with continuous discovery, monitoring and updating. In addition, NetMRI automatically identifies critical network information and components such as VLANs, ports, device properties, OS versions, topology, layer 3 subnets and routes. The days of out-of-date spreadsheets are over with NetMRI

D. Device Discovery Engine

The device discovery engine is implemented in Python as a separate action within the network device discovery module. It provides discovery of network devices and provides information about the device (make, model, and OS the device is running) in a way that is more efficient, more precise, and better satisfies the requirements of NDS than a NetMRI based solution.

1) *General Logic Device Discovery Engine*

The general logic of this engine is the following. First, the input parameter, which is a target specification, is processed by the Parser object, which forms a list of Device objects with only the IP address or addresses selected. Then a Scanner object is created for each of the devices in a separate thread. This Scanner object performs the scanning process, i.e. applies the scanning techniques to the defined IP address(es). If a device is discovered to be offline, it is removed from the list of devices, otherwise the Scanner populates the data fields in a Device object with all the information that it is able to discover about the device. Lastly, the list of discovered devices is stored in the NCS database as operational data for further processing

a) IP Discovery Process

When an IP discovery starts, the appliance divides the IP addresses in a network into chunks, with each chunk containing 64 contiguous IP addresses. The discovery process probes each IP address in parallel and in ascending order, reports the detected information, updates the progress report, and then moves on to the next chunk until it hits the last chunk of IP addresses. The appliance then updates the database with the discovered data.

An IP discovery scans the selected networks in the order the networks appear in the Discover Manager wizard

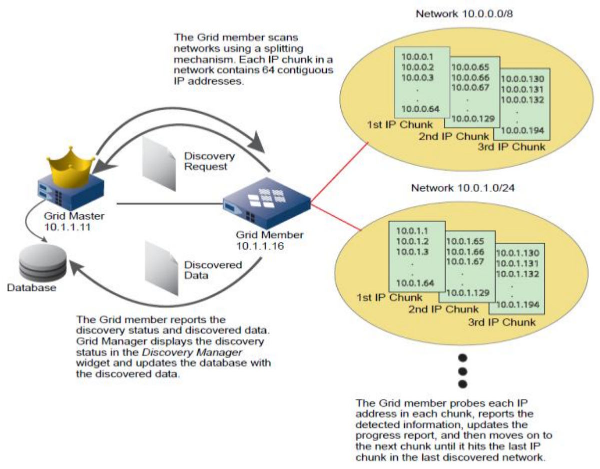

Network 10.0.0.0/8
The Grid member scans networks using a splitting mechanism. Each IP chunk in a network contains 64 contiguous IP addresses.
10.0.0.1 10.0.0.2 10.0.0.3 . . 10.0.0.64
10.0.0.65 10.0.0.66 10.0.0.67 . . 10.0.0.129
10.0.0.130 10.0.0.131 10.0.0.132 . . 10.0.0.194
1st IP Chunk
2nd IP Chunk
3rd IP Chunk
Discovery Request
Grid Master 10.1.1.11
Grid Member 10.1.1.16
Network 10.0.1.0/24
10.0.1.1 10.0.1.2 10.0.1.3 . . 10.0.1.64
10.0.1.65 10.0.1.66 10.0.1.67 . . 10.0.1.129
10.0.1.130 10.0.1.131 10.0.1.132 . . 10.0.1.194
1st IP Chunk
2nd IP Chunk
3rd IP Chunk
Database
Discovered Data
The Grid member reports the discovery status and discovered data. Grid Manager displays the discovery status in the Discovery Manager widget and updates the database with the discovered data.
The Grid member probes each IP address in each chunk, reports the detected information, updates the progress report, and then moves on to the next chunk until it hits the last IP chunk in the last discovered network.

Figure 5: - IP Discovery Process

b) TCP Port Scanning

The Scanner object takes a Device as an input. The Scanner has a set of predefined TCP ports to scan. Some of the TCP ports are assumed to run specific services, for example the Scanner expects SSH to be on port 22 and HTTP on port 80. Initially the Scanner performs a port scan iterating through the set of expected TCP ports. To scan a TCP port the Scanner tries to initiate a connection using a socket whose remote endpoint is defined by the Device's IP address and the port to be scanned

c) SNMP Scanning

In contrast to TCP port scanning, UDP port scanning is different due to the connectionless nature of the UDP protocol. However, in the context of our NDS discovery module we are most interested in only one UDP based protocol, namely SNMP (as NCS can manage SNMP enabled devices when certain conditions are met). For the purposes of the device discovery module, we are particularly interested in obtaining the system description string, which is available in MIB-II and is usually present in most devices that are SNMP enabled.

d) Sorting Device Type and Description

After the scanning is completed and a list of Device objects (which contains devices that were discovered along with their associated information which the Scanner was able to retrieve) is created, then the type, description, and vendor of the device are determined. This information is extracted from the data that was discovered during port scanning. Each port information field in a Device object is processed and matched to a regular expression set that matches expected SSH Communicator module output as well as some of the well-known SNMP system description formats; in this way the hardware platform and the OS information are obtained and stored in the respective fields of the object. Once these fields are set, the device type is determined. Determining the device type and setting appropriate management port and management protocol are essential requirements for a device to be successfully stored in the NDS CDB, as well as to enable the NDS to be able to subsequently manage that device.

E. Loading Devices into NDS

The pick action implements the functionality of fetching the full set of information related to a discovered device from the operational data and storing the device into the NDS configuration database. Once a device is stored in the CDB as a managed device, NDS can connect to the device and manage it. The pick action is defined within the device list structure of the discovery component data model. Thus, the action is bound to a specific device in the list of discovered devices. Calling the action from the context of a device allows us to offer an NDS operator the option to provide additional parameters when they select a device, such as assigning a new name or authentication group

F. Results and Analysis

The outcome of the device discovery component design and implementation is an optional package for NDS. This package provides automated network discovery functionality and the possibility to add newly discovered devices to the NDS CDB. The package's functionality minimizes the effort of the network operator when adding a device by automatically loading the set of parameters that were obtained during the discovery process (device type, management port, management protocol, etc.). The device discovery module implementation utilizes NetMRI based device discovery module approach

The NetMRI based implementation uses the following techniques to identify the type of the scanned device:

- Port scanning;
- Service probing and banner grabbing; and
- OS fingerprinting.

A significant condition of the NetMRI based module is the need for an extended license for NetMRI usage, as the default NetMRI license does not allow using any of the NetMRI source code nor executing NetMRI and parsing the results in a non-General Public License (GPL) product.

IV. CONCLUSIONS AND FUTURE WORK

This chapter concludes the project. The following sections give a general summary of the work; discuss the initial goals that were defined for the project and compare them with the outcome of this work. The chapter also discusses some ethical aspects related to this project, and the future work that could built upon the results of this project.

A. Project summary and Results

The result of this project is two optional NDS packages that provide network device discovery and network topology discovery functionality. This result reflects the goals originally defined for this project.

The device discovery package provides network device discovery of devices in the specified IP address range, while also determining the type (software and hardware platform) of the devices whenever possible and collecting all the additional parameters required to add the discovered device to NDS's managed devices tree. This package minimizes the effort required by an NDS operator by allowing the operator to automatically save the discovered device(s) into the NDS configuration database together with the relevant parameters and provides an option to add multiple devices with a single command. The device discovery package meets the requirements initially set for the package. The package includes both requested NetMRI based device discovery engine which utilizes a credential based approach to device discovery and is better suited for use by an NDS operator. The package's data model and the structure of the package meets the requirements for an NDS package.

The following section describes the ethical considerations behind this work. The section also discusses the authors' view of the intended use of software which was developed and the use of the material presented in this Project

B. Ethical Considerations

Network scanning is a highly debated topic from an ethical perspective and especially from a legal perspective. This project covers some aspects and methodologies used for network scanning and discovery, additionally the software developed during this project provides network scanning functionality. The content of this Project is intended to provide research based insight into network scanning and should be useful for other research, educational, or other legitimate purposes.

The usage of components is controlled by NDS's access control and security mechanisms; the components themselves do not implement any additional usage restrictions. Thus, in multiuser NDS environments, the NDS administrator should configure suitable usage policies for these new components. The only security feature implemented for these new components is the encryption of the credentials stored in the configuration database and the valid credentials that worked on a specific device when stored into the operational database. Encrypting strings is a built-in feature of NDS, which is controlled by the NDS configuration. The NDS administrator can specify the encryption algorithm to be used and the associated parameters (encryption keys, initialization vectors, etc.). The NDS administrator is advised to select an appropriate encryption algorithm and associated parameters, while adhering to the general practice of securing access to the encryption keys.

The following section describes some potential future work related to this project. It also discusses some possible future work in the area of network discovery from the perspective of this thesis project.

C. Future Work

The device and topology discovery packages provide the required discovery functionality - as this functionality was defined within the scope of this thesis project. However, these packages are still an experimental feature and do not provide the functionality which might be ultimately required. Future work should make these new discovery components more complete in order to provide a feature rich network discovery solution to NDS customers.

REFERENCES

1. J. Case, R. Mundy, D. Partain, and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework," RFC 3410 (Informational), Internet Engineering Task Force, Dec. 2002
2. K. McCloghrie, D. Perkins, and J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)," RFC 2578 (INTERNET STANDARD), Internet Engineering Task Force, Apr. 1999.

3. R. Stadler, lectures notes for the course EP2300 Management of Networks and Networked Systems, KTH Royal Institute of Technology, Aug.-Oct. 2012, unpublished.
4. J. Yu and I. Al Ajarmeh, "An Empirical Study of the NETCONF Protocol," in Sixth International Conference on Networking and Services (ICNS), March 2010.
5. R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network Configuration Protocol (NETCONF)," RFC 6241 (Proposed Standard), Internet Engineering Task Force, Jun. 2011
6. J. Schönwälder, M. Björklund, and P. Shafer, "Network configuration management using NETCONF and YANG," in IEEE Communications Magazine, vol. 48, no. 9, September 2010.
7. M. Wasserman, "Using the NETCONF Protocol over Secure Shell (SSH)," RFC 6242 (Proposed Standard), Internet Engineering Task Force, Jun. 2011
8. M. Badra, "NETCONF over Transport Layer Security (TLS)," RFC 5539 (Proposed Standard), Internet Engineering Task Force, May 2009.
9. T. Goddard, "Using NETCONF over the Simple Object Access Protocol," RFC 4743 (Historic), Internet Engineering Task Force, Dec. 2006.
10. E. Lear and K. Crozier, "Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)," RFC 4744 (Historic), Internet Engineering Task Force, Dec. 2006.
11. M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)," RFC 6020 (Proposed Standard), Internet Engineering Task Force, Oct. 2010.
12. J. Schönwälder and H. Langendörfer, "How to Keep Track of Your Network Configuration," in Proceedings of the 7th USENIX conference on System administration (LISA 1993), 1993.
13. G.G. Richard, "Service advertisement and discovery: enabling universal device cooperation," Internet Computing, IEEE, vol. 4, no. 5.
14. "IEEE std 802-2001, Standard for Local and Metropolitan Area Networks: Overview and Architecture," February 2002.
15. Luo Junhai, Fan Mingyu, and Ye Danxia, "Research on Topology Discovery for IPv6 Networks," in Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007, SNPD 2007. 8th ACIS International Conference, vol. 3, 2007.
16. U. Blumenthal and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," RFC 3414 (INTERNET STANDARD), Internet Engineering Task Force, Dec.
17. S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Draft Standard), Internet Engineering Task Force, Sep. 2007.