Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 36 (2014) 373 – 380

Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014- Philadelphia, PA

Self-Managed Networks with Fault Management Hierarchy

Mehmet Toy*

T&P Product Engineering, Comcast Cable, LLC, Mount Laurel, NJ USA

Abstract

The monitoring of network resources and services, isolating and identifying problems when failures occur, and fixing them by sending technicians to the sites most of the time or downloading certain configuration files remotely to fix configuration related problems are common in the telecommunications industry. This is costly. In order to reduce the operational cost, it is necessary for a network isolating and identifying problems by itself and fixing them, and having technicians at the failure site only when there is a single point of hardware failure. This paper introduces the concept of a self-managed network to identify network problems during failures and repair them, in addition to self-configurations of network resources and services. Self-managed Network Element (sNE) architectures and Network Management System (sNMS) architectures for centrally managed networks are described. A hierarchy among repairing entities is defined. An in-band message format for Metro Ethernet networks is proposed for the fault management communication.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: Self-Managed; Networks; Fault; Hierarchy; In-Band; Metro Ethernet

1. Introduction

The industry is focused on auto-configuration [1-3] and monitoring of network resources and services, isolating problems when there are failures, and fixing them by sending technicians to the sites most of the time or downloading certain configuration files remotely for configuration related problems. The concept of network identifying problems by itself and fixing them and only sending technicians to the failure site only when there are single-point of hardware failures (i.e. there is no hardware redundancy) is not practiced [5,6]. Tools for self-

* Corresponding author. Tel.: 1-856-792-2801; .
E-mail address: Mehmet_Toy@cable.comcast.com.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.008

managed networks are not developed either. On the other hand, auto-configuration of network elements (NEs) such as cable modem (CM) and cable modem termination system (CMTS) is being practiced by Multiple System Operators (MSOs) using Data Over Cable Service Interface Specification (DOCSIS) back-office systems. Similar procedures are also used by DPoE networks [4] for auto-configuration of NEs and services. This paper does not discuss the auto-configuration, but focuses on fault management aspects of self and centrally managed networks.

In the proposed self-managed network concept, each self-managed NE (sNE) in a network monitors its hardware and software resources periodically, runs diagnostics tests during failures in a hierarchical fashion and identifies problems if they are local to the sNE and fixable by the sNE, and reports failures and fixes to a centralized network management system (sNMS) to be accessed by network operators, field technicians, customers, and other sNEs in the network. If the problem is not locally fixable by the sNE, the sNMS runs its own rule-based logic to determine if the problem is fixable remotely by the sNMS. If it is not, a notification is sent to a network operator or field technician to fix the problem.

Failure type, if the problem is fixable locally by sNE, remotely by sNMS, or remotely by a technician, and estimated fix time are communicated with a newly defined message format. The hierarchy of fixing failures is network architecture dependent, as discussed in section 5.

It is expected that the concept of self-managed networks as described here to change how networks operate today, reduce the cost of operation (OPEX), and network provisioning and maintenance intervals dramatically.

2. sNE and sNMS Architectures for Self and Centrally Managed Networks

A self and centrally managed network architectures consisting of self-managed NEs and self-managing NMS are depicted in Figure 1. An sNE (Figure 2) consists of intelligent agents. Each self-managing agent (i.e. intelligent agent) monitors the entity that it belongs to, runs diagnostic tests to identify problems during failures, initiates a failure message, fixes problems, and initiates fix reporting to the central self-managing system, sNMS. The message indicating that the fixing entity is sNE is communicated to other sNEs, sNMS, field technicians and customers (if desired). If the problem is determined to be not fixable locally after two-three tries or without a try, depending on the problem, a message is sent to the sNMS by the sNE indicating that the fixing entity is unidentified.

An sNE consists of an intelligent NE (iNE) and one or more intelligent agents of each type. The agents are one or more intelligent Hardware Maintenance Agent(s) (iHMA(s)), intelligent Operating System Maintenance Agent (s) (iOMA (s)), intelligent Application Maintenance Agent (s) (iAMA (s)), and intelligent Capacity Management Agent (s) (iCMA (s)), depending on the implementation.

The iHMA periodically monitors hardware entities such as CPU, memory, physical ports, communication channels, buffers, backplane, power supplies, etc., and initiates pre-defined maintenance actions during hardware failures. iOMA periodically monitors operating system and initiates pre-defined maintenance actions during Operating System failures. The iAMA periodically monitors application software and protocol software, and initiates pre-defined maintenance actions during application and protocol software failures. The iCMA periodically monitors system capacity, load and performance, and collects measurements. During failures, the iCMA initiates pre-defined maintenance actions.

In addition to the intelligent agents above, the sNE needs to be designed as an intelligent NE (iNE) with redundant hardware and software components as depicted in Figure 2, where each hardware and software component is capable of running its own diagnostics and identifying faulty subcomponents.

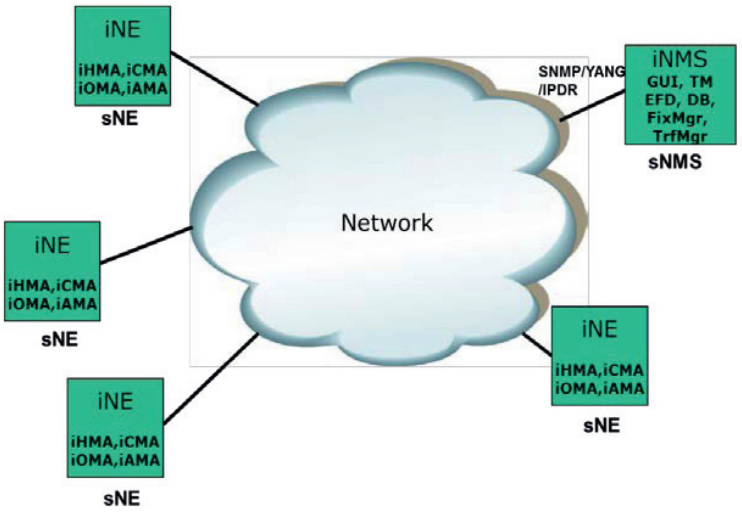


Figure 1: Self and Centrally Managed Network Architecture

On the other hand, the sNMS consists of an intelligent NMS (iNMS) that mainly deals with remote fixes, a Task Manager (TM) to manage tasks to be executed, copies of software modules for each type of sNE, a Traffic Manager (TrfMgr) to deal with network level traffic management issues such as routing policies, load balancing, connection admission control, congestion control, Event Forwarding Discriminator (EFD) to forward failures and fixes to network operators and customers, data base(DB) to store data, and Graphical User Interface (GUI) (Figure 3).

The web-based GUI provides human and machine interfaces, opens and closes sessions with its clients, performs initial authentication, validates all submitted data from the clients, processes chart data generated by parser, generates flash based charts and graphs for its clients. The parser processes GUI templates. A DB stores GUI events and data collected. A Task Manager prioritizes and schedules execution of the tasks including repair and configuration of activities that can be performed remotely using a Rule Based Logic module. A Data Handler collects end-to-end connection level measurements and sNE level capacity measurements, and stores them in the DB to support the TrfMgr.

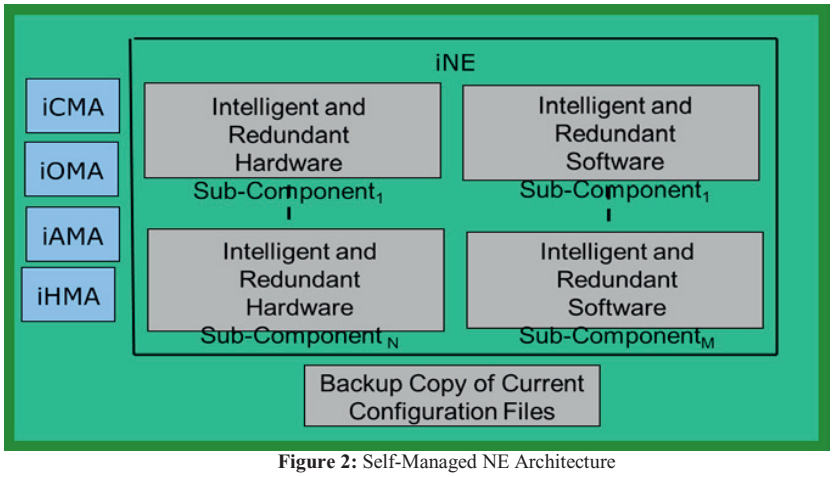


Figure 2: Self-Managed NE Architecture

There are no changes introduced to interfaces between the management systems and the network for self-management. The well-known protocols such as SNMP, IPDR (IP Detail Record) for usage information, Network Configuration (NETCONF) for manipulating configuration data and examining state information, and YANG modeling can be employed.

The intelligent agent architecture is depicted in Figure 4. Its Rule Based Logic module determines problems and initiates fixes if the problems are local to sNE, initiates tests for the fixes, determines if the fix procedure or a step or some of the steps are to be repeated, and initiates a message to all related parties about the fix. If the problem is not local to the sNE, the sNE informs all related parties including the sNMS for its conclusion which is that the fixing entity is unidentified. If the result of diagnostics cannot identify the failed component which is inconclusive, that will be conveyed as well.

The Scheduler module determines the priority and order of the tasks for each functional entity within an sNE. The Application programming interface (API) provides an interface to various types of Software and Hardware entities within the sNE. The Data Handler module collects necessary data (such as appropriate measurements) for the sNE, performs the fix, and keeps the data associated with the task. The AUTH module authenticates local user access and remote user access from the sNMS interface to sNE agents. The Utilities module supports various file operations.

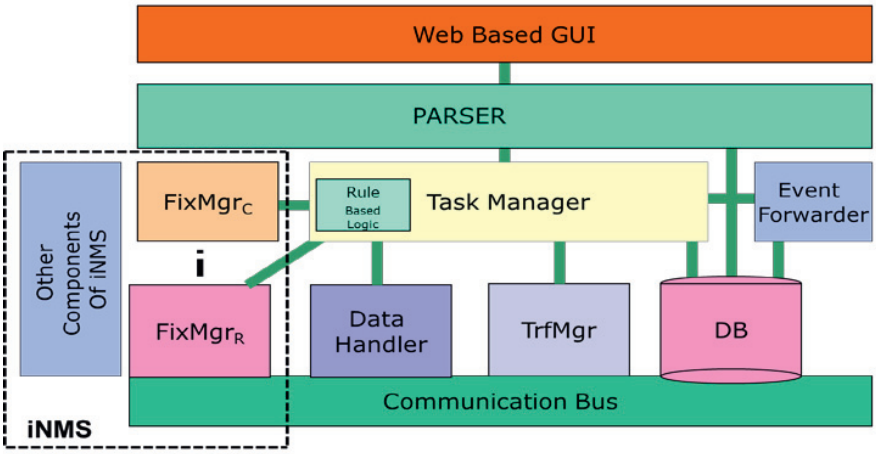


Figure 3: Architecture of Self Managing NMS

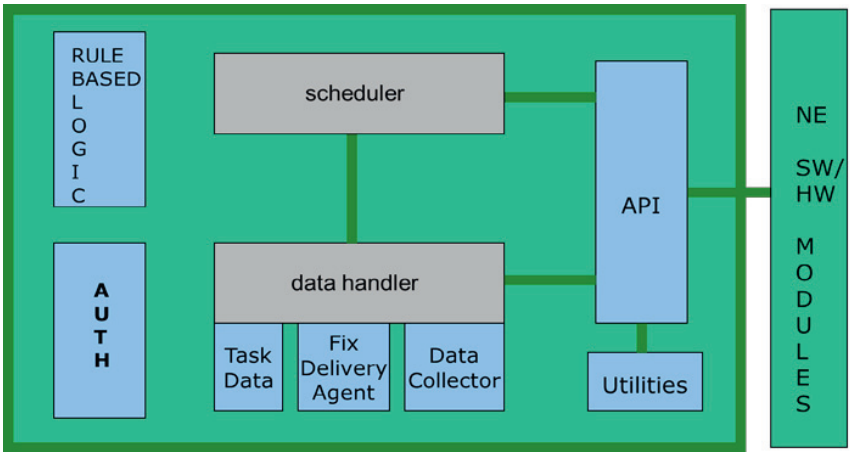


Figure 4: Intelligent Agent Architecture

3. Intelligent NE and intelligent NMS Architectures

Intelligent NE (iNE) (Figure 5) consists of intelligent sub-components such as chips, operating system and protocol software that are capable of periodic self-checking, declaring a failure when it is unable to perform its functions, running diagnostics and identifying whether the faulty entity is within a subcomponent or not, escalating the diagnostics to the next level in the hierarchy when the diagnostics are inconclusive.

When there is a failure, if failed entity is unidentified as a result of the diagnostics tests run by the intelligent subcomponents, the iNE is able to run diagnostics for a pre-defined set of sub-components that are collectively performing a specific function. A pre-defined set of sub-components can be a collection of components that are contributing to the realization of a main function such as packet forwarding, deep packet inspection, event forwarding, etc.

If the diagnostics tests ran for a pre-defined set of subcomponents cannot identify the failed entity, the iNE is able to run diagnostics at NE level to determine the failure. After the failure is identified to the smallest replaceable hardware (e.g. chips, wires connecting chips, backplane, etc.) and/or software entity (e.g. kernel, log, protocol software, event forwarding discriminator, etc.), the responsible agents determine if the failure is fixable and initiates a message to related parties with estimated fix time to repair. If the iNE diagnostics are inconclusive, then that will be communicated as well (see Section 4).

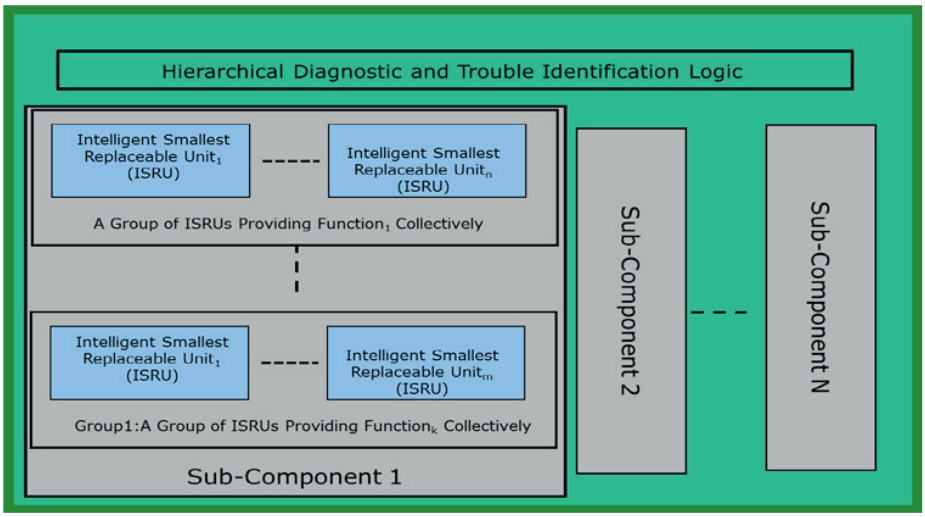


Figure 5: Intelligent NE Architecture

On the other hand, an intelligent NMS (iNMS) periodically monitors the network that sNMS is managing, identifies network level failures, estimates and communicates the fix time to related parties, and fixes them. When the sNE reports that the failure is not local (i.e. either tests are inconclusive or sNE is not capable of fixing it), the Rule Based Logic of the sNMS verifies if the sNE failure is not local.

The sNMS is redundant where the active sNMS is protected by a stand-by sNMS. The iNMS in active and stand-by units perform periodic self-checking. When the active sNMS fails, the stand-by sNMS takes over the responsibilities.

The Task Manager (TM) of sNMS manages tasks to be executed by the sNMS. The Rule-Based Logic determines if the problem is remotely fixable by the iNMS.

The iNMS includes a Fix Manager (FixMgr) for each sNE type to fix sNE problems remotely, stores software modules specific to sNEs and network level traffic management algorithms such as routing policies, load balancing, connection admission control and congestion control, and executes the algorithm when needed. Furthermore, the iNMS holds a copy of each sNE agent and remotely loads into sNEs when needed.

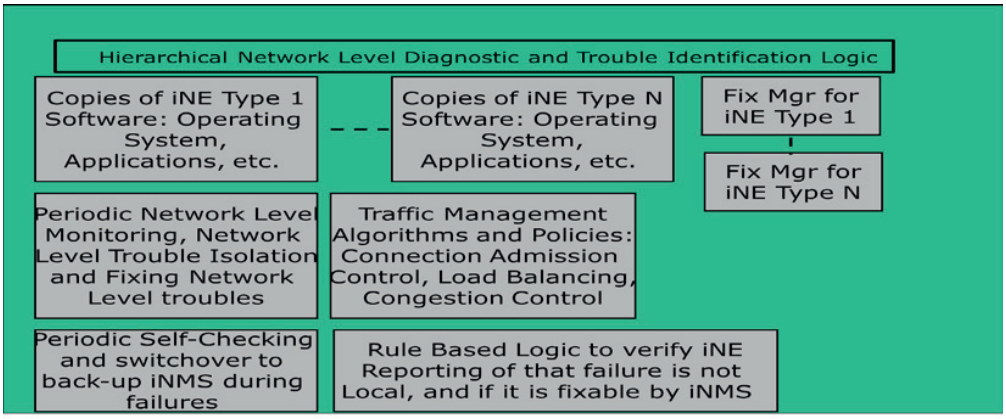
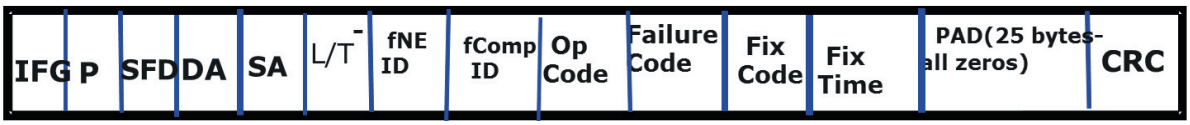


Figure 6: Intelligent NMS Architecture

4. In-band Communications of Failure types, Estimated Fix Time and Fix

In today's networks, failures related to ports and connections are mostly reported to an NMS via SNMP traps or in-band communications to NEs via AIS (Alarm Indication Signal), RDI (Remote Defect Indicator), Connectivity Check Message (CCM) related events such as Loss of Continuity (LoC) [7], etc. These alarms and traps identify the failed NE, port, or connection, but don't identify the component contributing to the failure. For self-management, it is necessary to identify faulty components, estimate the time for fix, and communicate that to all parties involved (i.e. sNEs and sNMS and field technicians, customers), so that working sNEs can store (if desired) data routed to the failed sNE (s) for the duration of fix and re-route traffic around the failed sNE (s) or port (s). This paper introduces a concept of informing sNEs, sNMS, field technicians and users (if desired) about type of failures, hierarchy for fixing the failures (i.e. whether it is locally fixable by sNE or remotely fixable by sRNMS or sNMS or locally fixable by a field technician), and proposes a frame format for in-band communications.

Figure 7 depicts a proposed Ethernet frame for Ethernet networks to carry all the information described above. Similar messages are to be created for other types of networks such as IP, MPLS and IMS.



IFG: Interframe Gap, 12 bytes
P/SFD (Preamble/Start of Frame Delimiter)-8 Bytes (P-7 bytes, SFD-1 byte)
L/T (Length/Type) : Length of frame or data type , 2 bytes (0x8808)
CRC: 4 bytes
DA: 01:80:C2:00:00:02 (6 bytes)-Slow protocol multicast address
fNE ID: 6 bytes, Failed sNE Identifier
fComp ID: 4 bytes, Failed Component Identifier
Op Code: 2 bytes-0x0202 for Disabled and 0x0303 for Enabled status
Failure Code : 4 bytes

Fix Code: 1 byte identifying fixing entity, NE (x00), sNMS (x01), sRMS (x02), sNMS-v (x03), RNMS-v (x04), sNMS-s (x05), sRNMS-s (x06), field technician (x07), unidentified entity or inconclusive diag(x08)
Fix Time: 4 bytes indicating fix time in seconds by NE, NMS, or field technician

Figure 7: Frame format for Self-managed Ethernet networks

.

For Ethernet networks, slow protocol multicast address is used to inform sNEs, sNMS, and field technician devices connected to the network. **fNE ID** indicates MAC address of the failed sNE. **fComp ID** indicates the failed component identifier within the sNE. **Op Code** indicates whether the sNE or port is operationally disabled or enabled. This operational status is disabled during failures and becomes enabled after the failure is fixed. **Failure Code** indicates failure type. If failure type is unidentified thru diagnostics, Failure Code will be unidentified or inconclusive or not-local to sNE. **Fix Code** identifies repairing entity whether it is sNE, sNMS, or a field technician.

It is possible to allocate six bytes to Fix Code field to indicate MAC address of the fixing entity. It is also possible to identify the failure type and not able to fix it. In this case, fixing entity is unidentified. It is also possible that both failure code and fix code are unidentified. **Fix time** indicates the estimated time in seconds for repair which is filled by the repairing entity. In order for sNE and sNMS to provide the estimated fix time, the fix time for each type of failure needs to be stored in sNE and sNMS.

Given the sNMS interface uses network management protocols such as SNMP, the information in the message (Figure 7) needs to be conveyed to sNMS via an SNMP trap . Similarly the SNMP trap from sNMS needs to be converted into an in-band message to convey the information to self-managing NEs.

5. Failure Fixing Hierarchy in Centralized Networks

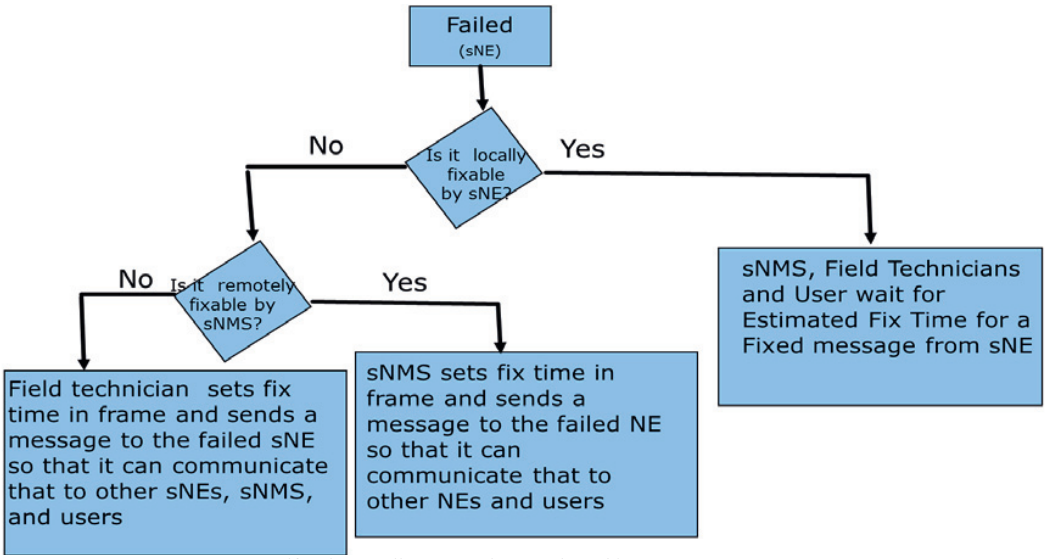


Figure 8: Self and Centrally Managed Network Architecture

In a centrally managed network, when there is a failure, sNE determines if the failure is local to the sNE or not. If the failure is local, then the sNE informs other sNEs, sNMS, field technicians and customers about failure type and fix time. If NE decides that the failure is not local to sNE, then sNE escalates the problem to the sNMS. The sNMS verifies that it is not local to the sNE and determines if it can fix the problem. If the sNMS can fix the problem, the sNMS communicates the failure type and fix time to sNEs, field technicians and customers. If the sNMS determines the failure is not fixable, the sNMS escalates the problem to field technicians. The field technician communicates fix time to sNEs, the sNMS and customers. After the fix is completed, the fixing entity initiates a self-managed

notification with Enabled status (i.e. Opcode is set to Eanabled) to other sNEs, the sNMS, and customers. Both sNMS and field technicians use one of the sNEs to send notifications to the remaining interested parties.

The sNMS and field technician communicates failures and fixes via a message from the sNMS. If there is a node failure (i.e. sNE completely fails due to a power failure for example), neither the sNMS nor field technicians is able to communicate with the sNE. Therefore, the sNMS and field technicians would use another sNE to communicate the failure.

6. Conclusion

Self-managed network concept, sNE and sNMS architectures and a fault management communication mechanism for centrally self-managed networks are introduced. However, provisioning of self- managed networks is not described. Self-provisioning of Cable Modem (CM), cable modem termination system (CMTS) and services; and DPoE systems and services are partially addressed by DOCSIS and DPoE standards. Further work is necessary to enhance these procedures.

References

1. ETSI GS AFI 002 V1.1.1 : Autonomic network engineering for the self-managing Future Internet; Generic Autonomic Network Architecture, 2013-04
2. ETSI GS AFI 001 V1.1.1 Group Specification Autonomic network engineering for the self-managing Future Internet (AFI); Scenarios, Use Cases and Requirements for Autonomic/Self-Managing Future Internet, 2011-06
3. Keller, Alexander; et al. (Eds.), “Self-Managed Networks, Systems, and Services Second IEEE International Workshops”, SelfMan 2006, Dublin, Ireland, June 16, 2006, Proceedings
4. E. Malette and M. Hajduczenia, “Automating provisioning of Demarcation Devices in DOCSIS® Provisioning of EPON (DPoE™)”, IEEE Comm. Magazine, September, 2012
5. M. Toy, Self-Managed Networks, Comcast internal document, November, 2012.
6. M. Toy, Self-Managed Carrier Ethernet Networks, April 2014, MEF Meeting in Budapest, self-managed-networks-comcast-mtoy.pdf., https://wiki.metroethernetforum.com/display/OWG/New+Work
7. ITU-T Y.1731, “OAM functions and mechanisms for Ethernet based networks”, 2008