Rongqiang Zhang

Bringing the OpenBMC for Platform Management System in Telco Cloud

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

30 Apr 2019

Metropolia
University of Applied Sciences

Author(s) Title Number of Pages Date	Rongqiang Zhang Bringing the OpenBMC for Platform Man-agement System in Telco Cloud 88 pages + 0 appendices 30 Apr 2019
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Networking and Services
Instructor(s)	Ville Jääskeläinen, Head of Degree Program Zinaida Grabovskaia, PhL, Senior Lecturer Antti Koivumäki, Senior Lecturer Ari Helminen, Business Manager

The current platform management system in Telco cloud infrastructure is based on closed firmware stack. With the upcoming 5G, this closed firmware stack has created several technology and business problems. The major problems are hardware-software vendor lock-in, long lead time for feature development and bug fixing, and security risks.

The objective of this study is to evaluate the possibility to bring an Open Source software stack for platform management system and baseboard management controller in Telco cloud.

The study was divided into 3 parts. First part is to analyse the current state and project specification. Second part is to introduce and evaluate the OpenBMC, an open source software stack for the objective of this study. Third part is Proof of Concept to run OpenBMC on Telco.

Keywords	BMC, 5G, NFV, Redfish, Security

Table of Contents

List of Abbreviations

4G	The Fourth Generation of Broadband Cellular Network Technology
5G	The Fifth Generation of Broadband Cellular Network Technology
AHB	Advanced High-performance Bus
AMT	Active Management Technology
API	Application Programming Interface
AR	Augmented Reality
ASF	Alert Standard Format
BIOS	Basic Input Output System
BMC	Baseboard Management Controller
BSP	Board Support Package
C-RAN	Cloudification of Radio Access Network
CCLA	Corporate Contributor License Agreement
CD	Continuous Delivery
CI	Continuous Integration
CIM	Common Information Model
CNI	Container network interface
CLI	Command Line
COTS	Commercial off-the-shelf
CP	Control Plane
CPRI	Common Public Radio Interface
CVE	Common Vulnerabilities and Exposures
DASH	Desktop and Mobile Architecture for System Hardware
DC	Data Centre
DCMI	Data Centre Management Interface
DevOps	Development and Operations
DMTF	Distributed Management Task Force
DoS	Denial-of-Service
DU	Digital Unit
ESW	Embedded Software
ETSI	The European Telecommunications Standards Institute
FW	Firmware
HPC	High-Performance Computing
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
I^2C	Inter-Integrated Circuit
IB	In-Band
IETF	Internet Engineering Task Force

IoT	Internet of Things
IPC	Inter-Process Communication
IPMI	Intelligent Platform Management Interface
JSON	JavaScript Object Notation
LTS	Long-Term Support
MANO	Management and Orchestration
MC	Management Controller
MCTP	Management Component Transport Protocol
MD	Management Device
MEC	Mobile Edge Computing
ML	Machine Learning
NC	Network Controller
NC-SI	Network Controller Sideband Interface
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
Nf-Vi	NFVI - Virtualized Infrastructure Manager
NFVO	NFV Orchestration
NIC	Network Interface Controller
OBSAI	Open Base Station Architecture Initiative
OCP	Open Compute Project
OData	Open Data Protocol
OMSA	OpenManage Server Administrator
OpenBMC	Open Baseboard Management Controller
OoB	Out-of-Band
OS	Operating System
OSF	Open System Firmware
PCI-E	Peripheral Component Interconnect Express
PLDM	Platform Level Data Model
PoC	Proof of Concept
QEMU	Quick Emulator
RAN	Radio Access Network
REST	Representational State Transfer
RMC	Rack Management Controller
RU	Radio Unit
SCSI	Small Computer System Interface
SDDC	Software Defined Data Centre

SDN	Software Defined Network
SDS	Software Defined Storage
SES	SCSI Enclosure Services
SIM	Systems Insight Manager
SMASH	System Management Architecture for Server Hardware
SMBIOS	System Management BIOS
SMBus	System Management Bus
SNIA	Storage Networking Industry Association
SNMP	Simple Network Management Protocol
SoC	System on Chip
SOL	Serial Over Lan
SSH	Secure Shell
SW	Software
Telco	Telecommunication
TLS	Transport Layer Security
TPM	Trusted Platform Modul
TSC	Technical Steering Committee
UEFI	Unified Extensible Firmware Interface
UDP	User Datagram Protocol
UP	User Plane
URLLC	Ultra-Reliable and Low Latency Communications
USB	Universal Serial Bus
VI-Ha	Virtualization Layer-Hardware Resource
VIM	Virtualized Infrastructure Manager
VM	Virtual Machine
VNF	Virtualized Network Functions
vRAN	virtualized RAN
WDM	Wavelength Division Multiplexing
YANG	Yet Another Next Generation

1 Introduction

Management Controller (MC) or Baseboard Management Controller (BMC) is a specialized microcontroller embedded on the motherboard of a server, manages the interface between system-management software and platform hardware, provides the platform management functionality towards Data Centre (DC) administrator.

This thesis concentrates on three parts. Part one is current state analysis of platform management subsystem on the BMC-based server, the analysis will focus on Telecom industry perspective. Part two is feasibility study of Open Baseboard Management Controller (OpenBMC), including introduction and evaluation as an open source software stack for platform management system from Telecommunication (Telco) Cloud perspective. Part three is Proof of Concept to run OpenBMC on Telco hardware.

1.1 Industry Background

The Fifth Generation of Broadband Cellular Network Technology (5G) is the latest technology that is happening in Telecom industry. 5G will provide higher data rate, ultra-low latency, energy saving, cost reduction, higher system capacity, massive device connectivity, and better security. With these characteristics and performance of 5G, it's expected many applications, such as autonomous driving, Industrial Internet of Things (IoT), Smart City, Smart Factory, Augmented Reality (AR), Cloud-based Machine Learning (ML), Cloud-based Gaming, will be benefited and boosted [1].

To achieve the high data rate (up to 20 Gbps) and ultra-low (<1ms) latency target, 5G Core Network will be divided into User Plane (UP) in charge of bearer delivery, and Control Plane (CP) in charge of control functions. The CP will stay in the central cloud, but UP will be distributed closer to the cell nodes and installed in edge cloud [2]. Below figure is KT's 5G Network Architecture:

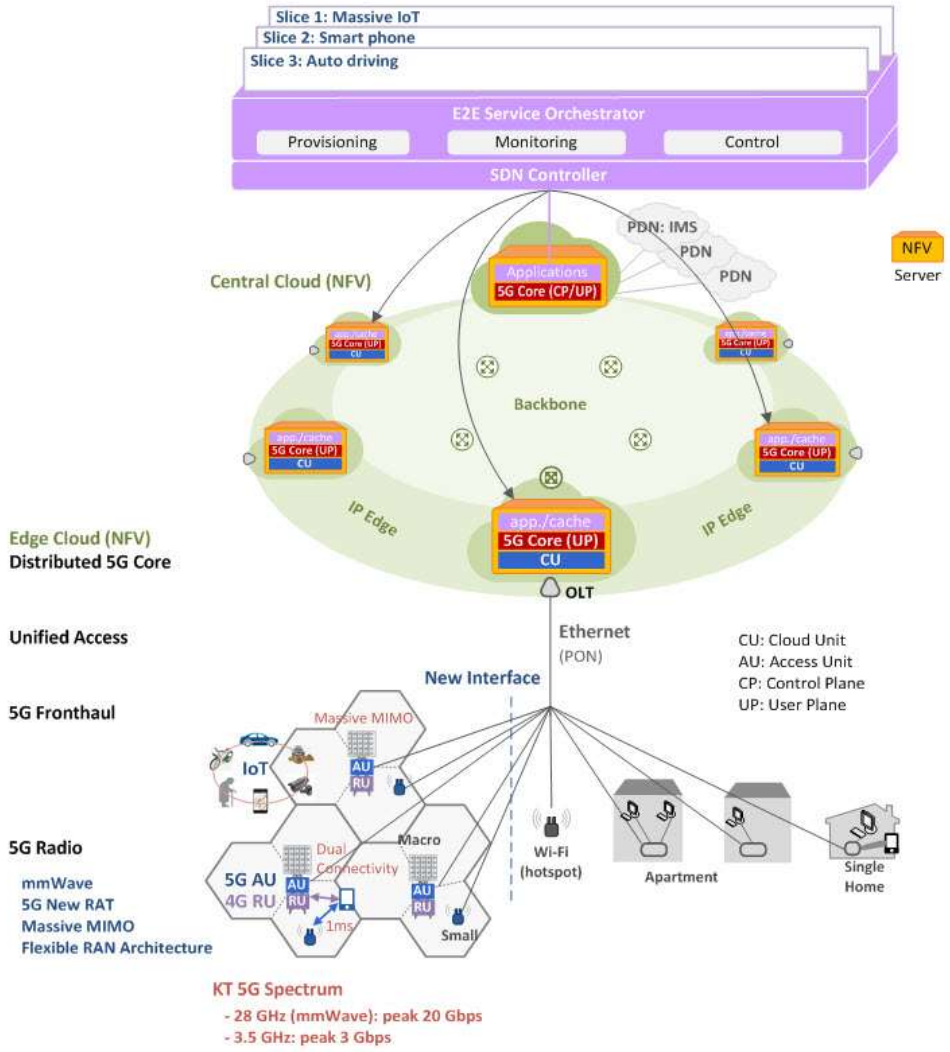


Figure 1 KT's 5G Network Architecture

Because of 5G new network architecture, user plane related application servers will be close to user as well and is possible to bring ultra-low latency services. It will also help to balance the fronthaul traffic and backhaul traffic – there will be significant user plane traffic processed within the edge cloud, therefore the user data rate will be much higher than before while backhaul traffic will be decreased.

To achieve the energy saving, cost reduction, and better security target, 5G Telco Cloud will be designed as software-centric cloud architecture [3]. Virtualized Network Functions (VNF) will be implemented as software and can be installed on Virtual Machines (VM) [4]. This will bring the opportunity to deploy 5G network function on Commercial off-the-

shelf (COTS) IT hardware and help to reduce the infrastructure cost significantly [5]. The software-centric cloud architecture will also bring better elasticity, better security and service continuity. Below figure is SK Telecom's software-centric 5G platform:

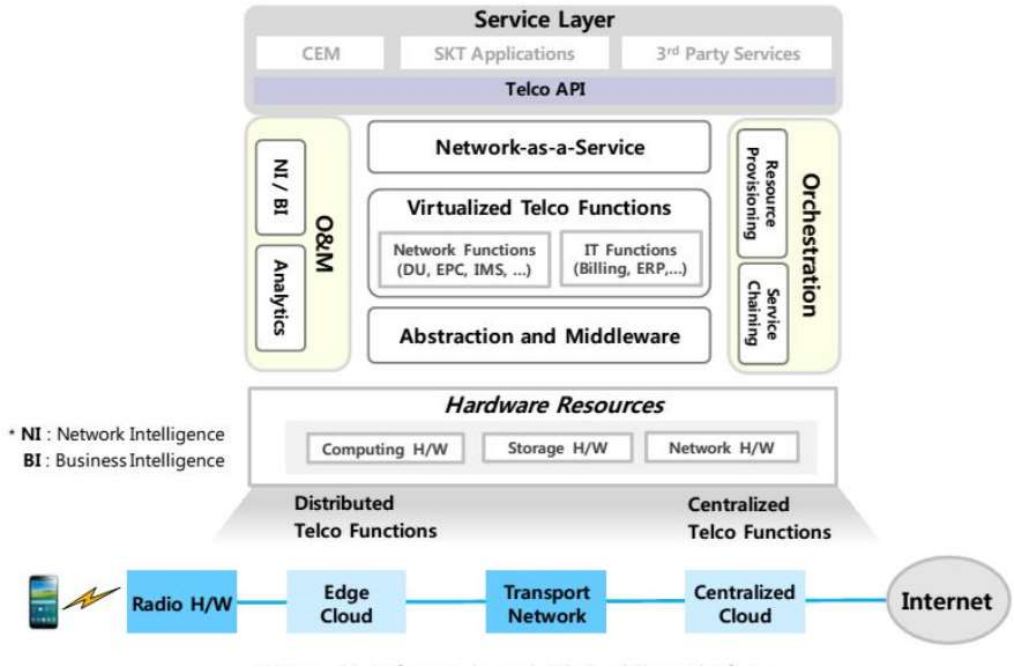


Figure 2 SK Telecom's software-centric 5G platform

Software-centric will bring the opportunity to deploy 5G network function on Commercial off-the-shelf (COTS) IT hardware and help to reduce the infrastructure cost significantly [5]. The software-centric cloud architecture will also bring better elasticity, better security and service continuity.

1.2 Technology Problem

The current platform management system and BMC firmware in Telco Cloud Infrastructure are mainly a closed ecosystem. The baseboard chip selection is limited within very few vendors, the baseboard firmware stack is closed and proprietary. There are several major problems which will become impediments on the path to 5G revolution.

The first major problem is vendor lock-in, both baseboard hardware and firmware highly depend on the specific vendor(s) [6]. The cost of switching vendor would be very high

and even prevents such possibility. As a result, the middleware needs to compromise and offer vendor adaption layer for VNF. This will violate the target of software-centric Telco Cloud.

The second major problem is the lead time for new development and bug fixing is usually long. Since the baseboard management firmware stack is closed and proprietary, the developer is lack of support from community, the community is lack of software upstream from developer neither [7]. The whole ecosystem mainly relies on a few key companies, who mainly serve the IT industry, to contribute and moves forward slowly. Telecommunication (Telco) industry specific requirements are likely compromised when there is confliction with IT industry requirements.

The third major problem is security. Telco Cloud has very high requirement on the security, the external interfaces must be protected with encryption protocol and each software stack must be kept-to-update with latest security patch. Same reason as previous problem, vendor doesn't have the motivation to abandon the existing mature implementation even if there is known security vulnerability.

1.3 Objective and Outcome

The objective of this study is to evaluate the possibility to bring an Open Source software stack for platform management system and baseboard management controller in Telco cloud.

This thesis concentrates on three parts. Part one is current state analysis of platform management subsystem on the BMC-based server, the analysis will focus on Telecom industry perspective. Part two is introduction and evaluation of Open Baseboard Management Controller (OpenBMC). Part three is Proof of Concept (PoC) of bringing OpenBMC for baseboard management software in Telco cloud.

1.4 Project Methodology

To achieve the research goal, this study will be conducted using literature review and Proof of Concept (PoC) approach [8]. The literature review will be used to collect the requirement and specification for this study. The PoC is used quite widely in software development to verify the feasibility of new technology or new method.

1.5 Scope and Structure

The project includes analysis between current platform management subsystem and OpenBMC, and feasibility to bring the OpenBMC in future Telco Cloud.

This thesis consists of 7 sections. Section 1 gives a brief introduction of the thesis project. Section 2 describes on current state analysis and gap analysis of Telco Cloud platform management software. Section 3 studies the project specification which is mainly 5G requirement in platform management system. Section 4 describes OpenBMC technical framework. Section 5 evaluates OpenBMC and analysis how to bring it for platform management in Telco Cloud. Section 6 gives a proof of concept on emulation and real hardware. Section 7 summarizes the study result and outcome.

2 Current State Analysis

This section first introduces the concept of Platform Management Subsystem and Management Controller (MC). Secondly, describes current technology of Platform Management Subsystem used in the industry. Thirdly, describes the current system solution, including hardware solution and software solution in the industry. Finally, gives a summary of current state analysis.

2.1 Concept of Platform Management Subsystem and BMC

This chapter will introduce the basic concept and definition of Platform Management Subsystem and BMC.

The term “Platform Management” initially defined by Intelligent Platform Management Interface (IPMI) specification:

“The term “Platform Management” is used to refer to the monitoring and control functions that are built in to the platform hardware and primarily used for the purpose of monitoring the health of the system hardware. This typically includes monitoring elements such as system temperatures, voltages, fans, power supplies, bus errors, system physical security, etc. It includes automatic and manually driven recovery capabilities such as local or remote system resets and power on/off operations. It includes the logging of abnormal or ‘out-of-range’ conditions for later examination and alerting where the platform issues the alert without aid of run-time software. Lastly it includes inventory information that can help identify a failed hardware unit.” [9 pp. 28].

Distributed Management Task Force (DMTF), which is a specification working group focuses on creating open manageability standards spanning diverse emerging and traditional IT infrastructures [10], has defined the Platform Management Subsystem in the following way:

“A Platform Management Subsystem in today’s enterprise computing platforms is comprised of a set of components which communicate to perform management functions within the platforms. In many cases, these communications and interfaces are specialized and adapted to each individual platform, installation and component in the environment. A Platform Management Subsystem provides hardware management services such as platform environmental monitoring functions (for example, temperature probing,

voltage monitoring, fan speeds, hardware error status, etc.) and control functions (for example, platform power-on/off, reset, watchdog timer, etc.).” [11 pp.9]

Management Controller (MC) or Baseboard Management Controller (BMC), is the key component in Platform Management Subsystem. BMC is a microcontroller or processor that manages the information from all attached Management Devices (MD) and provides the information to local or remote software, operates independently of the Operating System (OS). BMC receives control message from local or remote software and perform control function on the Management Devices. BMC also monitors Management Devices' health status, including Host Operating System, and performs prevention actions or recovery actions to avoid of system failure.

A modern Platform Management Subsystem Component is defined by DMTF [11 pp.10]:

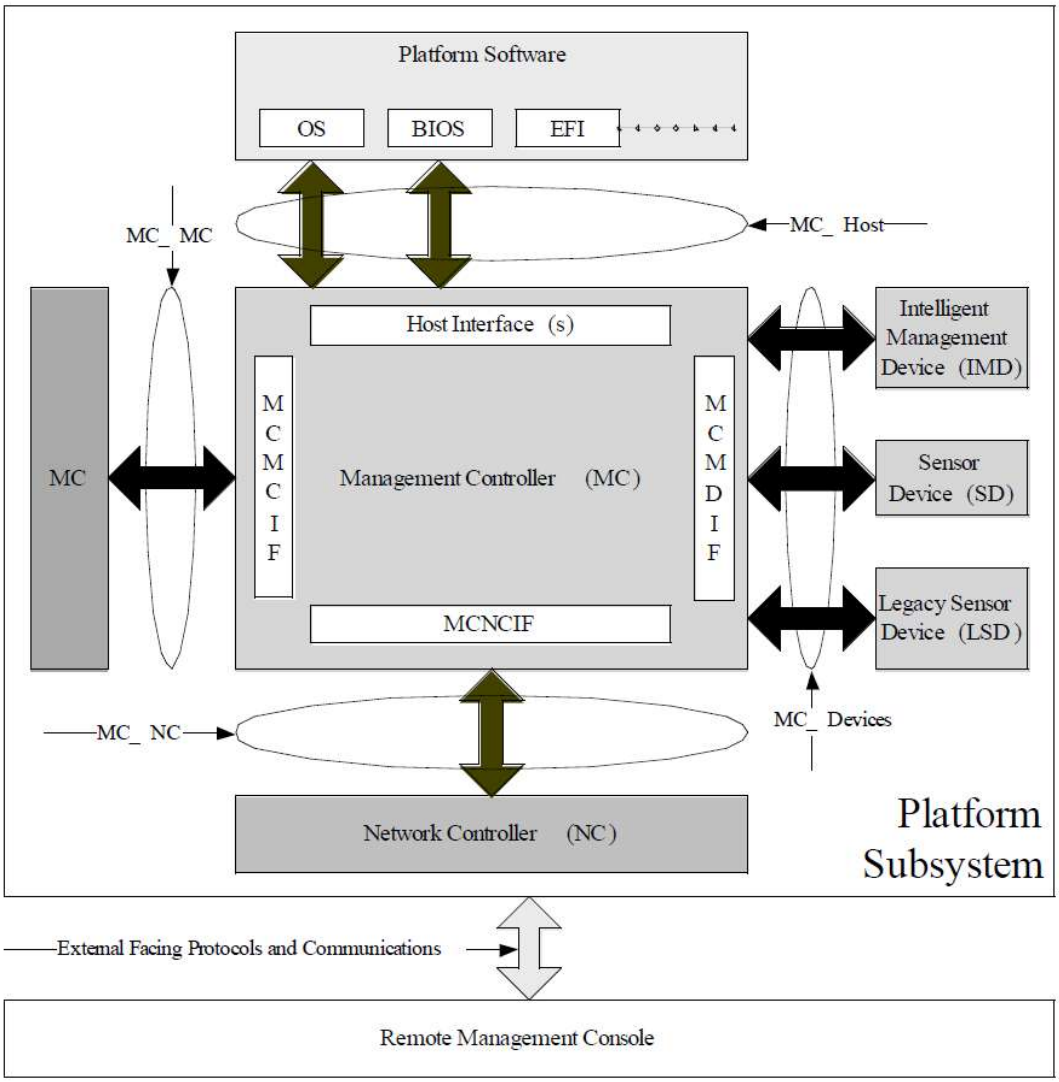


Figure 3 Platform Management Subsystem Component

BMC also play a key role in the Network Function Virtualization (NFV) Virtualized Infrastructure Management (VIM) architecture [12 pp. 43]. Below figure is an overview of NFVI domains and interfaces:

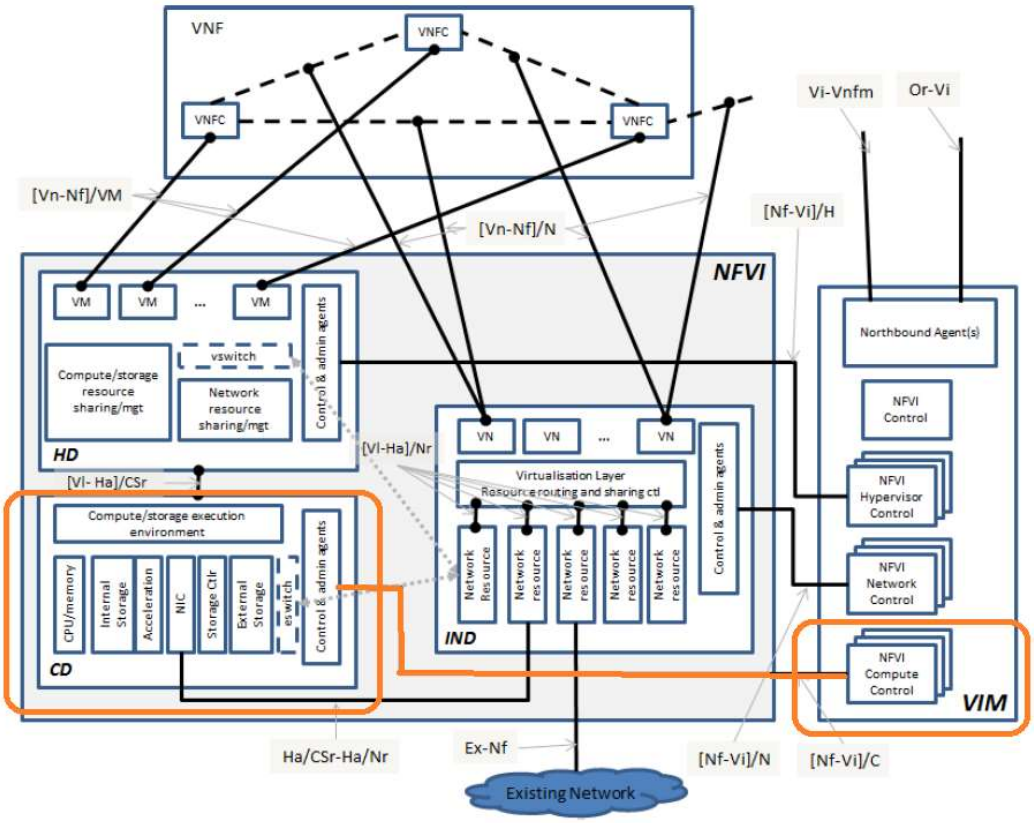


Figure 4 Overview of NFVI domains and interfaces

BMC can provide hardware platform information and receive control message to the VIM via the Out-of-Band (OoB) interface, which enables the datacenter admin to manage massive amount of hardware resources remotely and automatically.

2.2 Current Technology of Platform Management Subsystem and BMC

This chapter will introduce the mainstream technology in Platform Management Subsystem and BMC area. The feature of each technology will be analyzed, the advantages and disadvantages of will be compared.

Historically, the technology of Platform Management Subsystem and BMC started from industry leading hardware and firmware companies, to provide In-band (IB) and Out-of-Band manageability of the hardware resources.

In the recent years, “Cloudization” and “Open” are the key factors that drive the technology innovation in Platform Management Subsystem and BMC. There are several task force groups and standard groups who are actively contributing on the new technology and enhancement of existing technology on this area.

2.2.1 IPMI

Intelligent Platform Management Interface (IPMI) was a specification led by Intel and first published in 1998. It is still widely used in IT industry and Telco industry today. IPMI is a message based, hardware-level interface specification. IPMI operates independently of the operating system (OS) to allow administrators to manage a system remotely in the absence of an operating system or of the platform management software.

The advantage of IPMI includes:

- Installed by default: almost every server computer supports IPMI when it leaves the factory.
- Hardware native: IPMI command structure is designed with a hardware native style, the parameters is like “<hardware object> <action> <options>”. It’s straightforward to manage to manage hardware resource with IPMI command.
- Easy scripting: IPMI is a line by line message and can be easily implemented as a script.
- Good ecosystem: there are a lot of tools and solutions built on IPMI, most of them are opensource and supported by mainstream Linux distribution.

The disadvantage of IPMI includes

- Security risk: remote IPMI message is based on User Datagram Protocol (UDP) protocol and encrypted with a cypher. However, the cypher is not strong enough and there is already known critical vulnerabilities.
- Hardware dependency: each server computer supplier has own proprietary IPMI OEM extension command set. The syntax of IPMI command may vary between different vendors, and even vary between different hardware from the same vendor. It creates vendor-lock, as well as the complexity to manage computing resource.

- Not Cloud native: A Cloud native API data model is usually a REST (Representational State Transfer) message with data format including XML, HTML and JSON. But IPMI is a Command Line (CLI) style message, its data format is hexadecimal string which has dependence on the machine state and vendor's OEM extension.

2.2.2 SMASH

System Management Architecture for Server Hardware (SMASH) is a suite of specifications defined by DMTF and was first published in 2006. SMASH specifications describe an architectural framework, interfaces in the form of protocols, an addressing scheme, and profiles for server platforms [13 pp. 2]. The goal of SMASH is to enable the same interfaces regardless of server state, to enable the same tools, syntax, semantics, and interfaces to work across a full range of server products – standalone systems, rack-mounted servers, blades, Telco servers, and partitionable as well as virtual and redundant servers [13 pp. 12].

The advantage of SMASH includes:

- Same interface with unified object model: DMTF has defined two interfaces for SMASH, one is Command Line (CLI) and another is web, with unified syntax and object model. This has minimized the dependency on the hardware and simplified the remote access.
- Better security: SMASH is running on the secure shell (SSH) or Hypertext Transfer Protocol Secure (HTTPS), and SMASH specification includes provisions for authentication and authorization.

The disadvantage of SMASH includes:

- Complex to use: The operations, schema and profiles are complex to understand and perform even a simple task.
- Lack of acceptance: In the early years after SMASH was born, it was treated by most of server hardware manufactures as a threat in the beginning, since the purpose of SMASH is to unify the interface and reduce vendor-lock. In 2014, DMTF published Redfish specification which is widely accepted by industry.

- Poor ecosystem: due to lack of acceptant, the ecosystem of SMASH is very poor and it's very hard to get any tool built on it.

2.2.3 REST

Representational State Transfer (REST) is a software architectural style that provides interoperability between computer systems on the Internet, which allows the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations [14]. The term was introduced by Roy Fielding in his doctor thesis in 2000 [15 pp. 17].

By using a stateless protocol and standard operations, RESTful systems aim for fast performance, reliability, and the ability to grow, by re-using components that can be managed and updated without affecting the system, even while it is running.

The advantage of REST includes:

- Stateless protocol: in REST communication protocol, no session information is retained by the server. A stateless protocol does not require the server to retain session information or status about each communicating partner for the duration of multiple requests. The stateless design simplifies the server design because there is no need to dynamically allocate storage to deal with conversations in progress [16].
- Unified operations: the most common REST operations are GET, HEAD, POST, PUT, PATCH, DELETE, CONNECT, OPTIONS and TRACE.
- Cloud native: the data format of REST is usually HTML, XML or as JSON, which is self-descriptive and widely used in most popular cloud stack.
- Better security: REST can be run over Hypertext Transfer Protocol Secure (HTTPS), and REST specification includes provisions for authentication and authorization.
- Good ecosystem: REST is widely accepted by the IT industry and Telco industry, there are abundant of open source tools built on it.

The disadvantage of REST includes:

- Overhead in communication: stateless communication protocol requires client to include additional information in every request, and this extra information will need to be interpreted by the server.
- Lack of unified schema and profile: REST has unified operations and data format; however, REST doesn't have unified schema and profile definition. It's still very difficult to interoperate between different APIs and unable to break the vendor-lock.

2.2.4 Redfish

Redfish is a standard API published by DMTF in 2014. Redfish is a management standard using a data model representation inside of a hypermedia RESTful interface. The model is expressed in terms of standard, machine-readable Schema, with the payload of the message being expressed in JSON. The protocol itself leverages OData v4. Since it's a hypermedia API, it's capable of representing a variety of implementations via a consistent interface. It has mechanisms for managing data center resources, handling events, long lived tasks and discovery as well [17 pp. 3].

Redfish is designed to deliver simple and secure management for converged, hybrid IT and the Software Defined Data Center (SDDC). Both human readable and machine capable, Redfish leverages common Internet and web services standards to expose information directly to the modern tool chain.

The advantage of Redfish includes:

- It's a RESTful protocol: Redfish API is a RESTful API; thus, it has all the advantages of REST.
- Unified data format: DMTF has defined JavaScript Object Notation (JSON) as unified data format for Redfish. JSON is inherently human readable, is more concise than XML, has a plethora of modern language support and is the most rapidly growing data format for web service APIs [17 pp. 4].

- Unified schema and profile: DMTF has defined unified schema, URL conventions, and naming and structure of common properties which follow Open Data Protocol (OData) conventions, this provides for interoperability between different APIs [17 pp. 4].
- Good ecosystem: by adopting common OData conventions in a JSON payload, Redfish does not only encapsulate best practices for RESTful APIs which can be used in traditional and scalable environments but also further enables Redfish services to be consumed by a growing ecosystems of generic client libraries, applications, and tools [17 pp. 4].
- Hardware native as well as Cloud native: Redfish has a large scope of hardware architecture support as IPMI can do. But the data format of Redfish is usually JSON plus OData convention, which is human-readable as well as machine-readable and widely used in most popular cloud stack.
- Widely acceptance by industry: Redfish is widely accepted by the industry and is expected to be the future of hardware platform management.

The disadvantage of Redfish includes:

- Overhead in communication: Redfish is a RESTful API, so it's also stateless and must add extra information in each request message.
- Performance compared with IPMI: Below figure is a typical software stack for Redfish and IPMI.

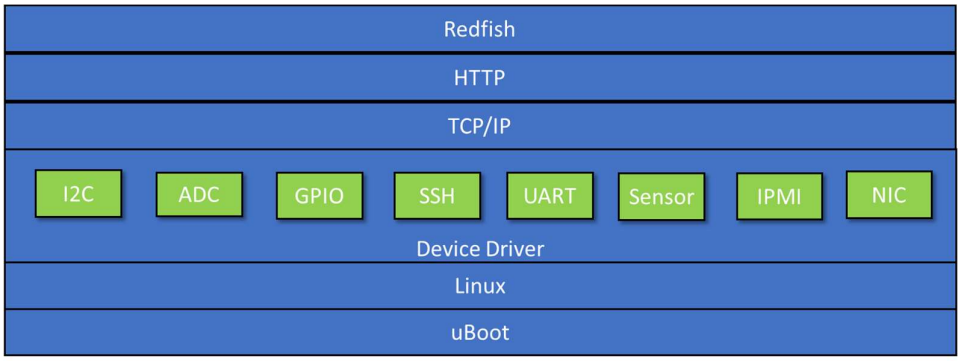


Figure 5 Redfish and IPMI software stack

Redfish software stack has more layers since Redfish is running over HTTPS, which will need more CPU time. Redfish also requires user authentication for each session, this

also slowdown Redfish response performance. While IPMI is initialized as device driver, which consumes less CPU resource and response speed is very fast.

2.2.5 SMBIOS

System Management BIOS (SMBIOS) is a specification published by DMTF in 1996. SMBIOS addresses how motherboard and system vendors present management information about their products in a standard format by extending the BIOS interface on Intel architecture systems [24].

For OS-present, OS-absent, and pre-OS environments, SMBIOS offers motherboard and system vendors a standard format to present management information about their products. By extending the system firmware interface, SMBIOS can be used with management applications that use the DMTF's Common Information Model (CIM) or another technology, such as SNMP. It eliminates the need for error-prone operations, such as probing system hardware for presence detection.

The advantage of SMBIOS includes:

- Simplify the management work from OS: SMBIOS contains the hardware inventory information. So, there is no need for host operating system to probe hardware directly to discover what devices are present in the computer.
- Widely support: DMTF has estimated that more than two billion client and server system implement SMBIOS [25].

The disadvantage of SMBIOS includes:

- Only static information: SMBIOS provides a snapshot of system inventory information, but doesn't support dynamic information, such as sensor information, on the fly.
- Unable to control hardware resource: SMBIOS just provide the inventory information to host OS. But host OS is not possible to control the inventory via SMBIOS.

2.2.6 NC-SI

Network Controller Sideband Interface (NC-SI) is an electrical interface and protocol published by the DMTF in 2010. NC-SI enables a common interface connection of a BMC to a set of Network Interface Controller (NICs) in server computer systems for enabling out-of-band remote manageability. [26].

The NC-SI is defined as the interface (protocol, messages, and medium) between a BMC and one or multiple Network Controllers (NC). This interface, referred to as a Sideband Interface in below figure, is responsible for providing external network connectivity for the Management Controller while also allowing the external network interface to be shared with traffic to and from the host.

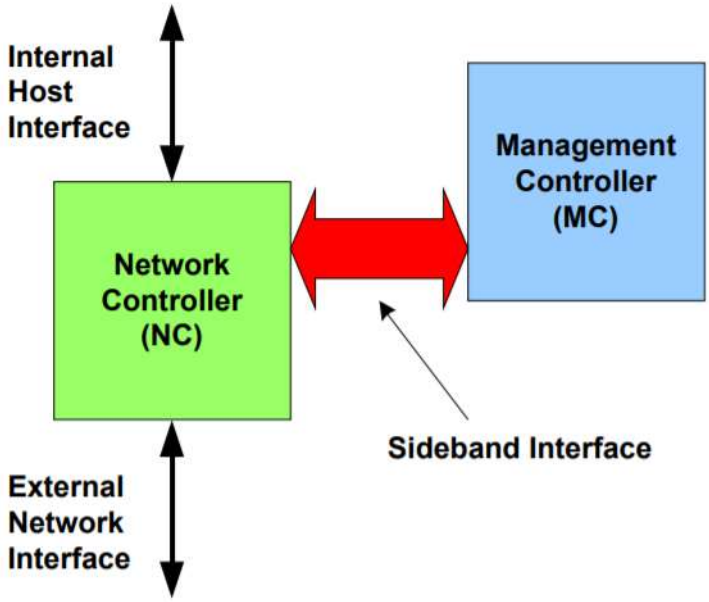


Figure 6 NC-SI Functional Block Diagram

The advantage of NC-SI includes:

- Reduce hardware cost: the sideband interface is a shared interface between BMC and host, which can help to reduce one ethernet port from the server as well as external switch port.
- Simplify the cabling: the shared interface also helps to reduce and simplify the external cabling.

The disadvantage of NC-SI includes:

- Reduced network interface capacity: Since the NC-SI is a shared interface between BMC and host, the capacity will inevitably fall into race condition as well. In some cases, this will cause delay and slow response to remote manageability.
- Risk to lose remote manageability to BMC and host at the same time: in case the shared interface is disconnected due to faulty or incorrect configuration, there is risk that remote manageability to BMC and host will be lost at the same time.

2.2.7 PLDM

Platform Level Data Model (PLDM) is an effective interface & data model published by DMTF in 2009. PLDM is designed to be an effective data and control source for mapping under Common Information Model (CIM). PLDM is targeted to provide an efficient access to low-level platform inventory, monitoring, control, evening, and data/parameters transfer functions such as temperature, fan, voltage, event logging, and boot control [11 pp. 14].

PLDM is a set of specifications, the PLDM specification work includes:

- PLDM Base Specification [27].
- PLDM for Firmware Update Specification [28].
- PLDM for FRU Data Specification [29].
- PLDM State Set Specification [30].
- PLDM for Platform Monitoring and Control Specification [31].
- PLDM for BIOS Control and Configuration Specification [32].
- PLDM for SMBIOS Transfer Specification [33].
- PLDM IDs and Codes Specification [34].
- PLDM Over MCTP Binding Specification [35].

- PLDM for Redfish Device Enablement Specification [36].

2.2.8 MCTP

The Management Component Transport Protocol (MCTP) is a protocol published in 2010. MCTP is designed for intercommunications among intelligent devices within a platform management subsystem. This protocol is independent of the underlying physical bus properties, as well as the "data-link" layer messaging used on the bus. [11 pp. 14].

MCTP's underlying buses include System Management Bus (SMBus), Inter-Integrated Circuit (I^2C), serial links, Peripheral Component Interconnect Express (PCI-E) and Universal Serial Bus (USB). Simplified nature of the protocol and reduced encapsulation overheads make MCTP suitable for implementation and processing within system firmware and integrated BMCs, on a wide range of platforms – including servers, workstations and embedded devices.

MCTP is a set of specifications, the MCTP specification work includes:

- MCTP Overview White Paper [37].
- MCTP Packets and NC-SI over MCTP Overview [38].
- MCTP Base Specification [39].
- NVMe over MCTP Binding Specification [40].
- MCTP SMBus/I2C Transport Binding Specification [41].
- MCTP PCIe VDM Transport Binding Specification [42].
- MCTP IDs and Codes [43].
- MCTP Serial Transport Binding Specification [44].
- MCTP KCS Transport Binding Specification [45].
- MCTP Host Interface Specification [46].
- NC-SI over MCTP Binding Specification [47].

Below is a simple view of firmware stack of platform management components intercommunication:

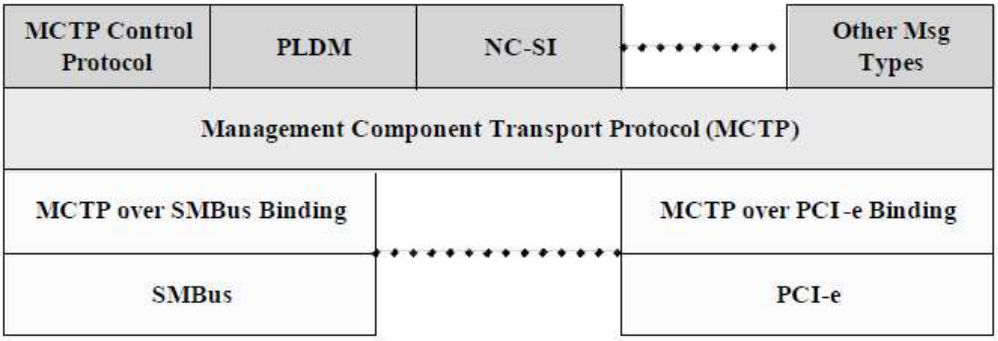


Figure 7 Platform Management Components Intercommunication stack [11 pp. 13]

MCTP specification and PLDM specification work together and enable a complete solution for platform management components intercommunication.

2.2.9 Swordfish

Swordfish is a standard API published by the Storage Networking Industry Association (SNIA) in 2016. Swordfish specification helps to provide a unified approach for the management of storage and servers in hyperscale and cloud infrastructure environments, making it easier for IT administrators to integrate scalable solutions into their data centers. SNIA Swordfish is an extension of the DMTF Redfish specification, so the same easy-to-use RESTful interface is used, along with JSON and OData, to seamlessly manage storage equipment and storage services in addition to servers [18].

Swordfish is very similar as Redfish. Swordfish is designed as an extension of Redfish and mainly focus on storage device management.

2.2.10 SES-2

SCSI Enclosure Services (SES-2) is a storage device management protocol published by T10 Technical Committee in 2002 [19]. SES-2 is the most widely used protocol for Small Computer System Interface (SCSI) enclosure-based storage device management. It permits the management and sense the state of power supplies, cooling devices, LED displays, indicators, individual drives, and other non-SCSI elements installed in an enclosure. SES2 alerts users about drive, temperature and fan failures with an audible alarm and a fan failure LED [20].

The advantage of SES-2 includes:

- Efficient: SES-2 utilizes the SCSI command which is embedded by most of SCSI devices. So, the SES-2 command response is very fast and efficiently.
- Low cost: there is almost no additional cost to support SES-2, since the storage device already supports SCSI by default.
- Hardware native: SES-2 command structure is designed with a hardware native style; the parameters is like “<page> <actions> <hardware address> <options>”. It’s straightforward to manage to manage storage device with SES-2 command.

The disadvantage of SES-2 includes:

- Hardware dependency: each storage enclosure has own proprietary SES-2 OEM extension command set. The syntax of SES-2 command may vary between different vendors, and even vary between different hardware from the same vendor. It creates vendor-lock, as well as the complexity to manage storage device resource.
- Not human readable: partial SES-2 commands have a human-readable format, but most of SES-2 commands are just hexadecimal string and not human-readable.
- Lack of remote manageability: SES-2 protocol cannot be transmitted over the Ethernet and IP packet; thus, it can’t be used remotely. User must use a bridge server and own adaptation layer to convert the SES-2 protocol.
- Not Cloud native: A Cloud native API data model is usually a REST (Representational State Transfer) message with data format including XML, HTML and JSON. But SES-2 is a CLI style message, its data format is hexadecimal string which has dependence on the machine state and vendor’s OEM extension.

2.2.11 YANG

Yet Another Next Generation (YANG) is a data modeling language used on network devices, published by Internet Engineering Task Force (IETF) as RFC6020 in 2010 [21]. YANG is a data modeling language used to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF), NETCONF remote procedure calls, and NETCONF notifications. YANG is used to model the operations and content layers of NETCONF.

YANG structures the data definitions into tree structures and provides many modeling features, including an extensible type system, formal separation of state and configuration data and a variety of syntactic and semantic constraints. YANG data definitions are contained in modules and provide a strong set of features for extensibility and reuse.

The advantage of YANG includes:

- Programmable Interface: YANG uses structured data format (XML or JSON) and REST-like APIs, which enables user to programmatically access different network devices.
- Security: the communication channel of YANG could be encrypted by Transport Layer Security (TLS), the security level is much higher than traditional SNMP protocol.
- Scalability: the grouping, refine, augment, and typedef statements are supported for users to extend YANG models and data types.
- Performance: in large scale of network management, YANG has outweighed the traditional SNMP in terms of performance [22].
- Interoperability with Redfish: Both YANG and Redfish can support CSDL annotation, DMTF has defined a specification for YANG to Redfish Mapping [23].

The disadvantage of YANG includes:

- Data structure limitation: the types of container data structures that YANG supports are not enough to model complex data types. For example, defining a recursive data structure is not possible with YANG. Authors are often forced to rethink the model in terms of lists and containers.

2.2.12 Other Technologies

There are other technologies specified by DMTF and other standard for computer hardware platform management, such as Desktop and Mobile Architecture for System Hardware (DASH), Alert Standard Format (ASF), Simple Network Management Protocol (SNMP). This study will not cover them since these technologies will be depreciated gradually.

There are other proprietary technologies for server hardware platform management, such as Active Management Technology (AMT) from Intel, OpenManage Server Administrator (OMSA) from Dell and Systems Insight Manager (SIM) from HP. This study will not cover them since these technologies are proprietary.

2.3 Current System Solution of Platform Management Subsystem and BMC

This chapter will summarize the current system solution of Platform Management Subsystem and BMC. The feature of each solution will be analyzed, the advantages and disadvantages of will be compared.

System solution includes hardware solution and software solution, as well as the long-term serviceability.

2.3.1 Current HW solutions

The Platform Management Sub-system is composed of BMC shipset and a bunch of management devices. The BMC is a key component in the platform management subsystem, it's usually a highly integrated System on Chip (SoC) and supports a bunch of SuperI/O functionalities in one chip.

There are several vendors provide BMC SoC. Currently ASPEED is dominating this market with AST2400 & AST2500 SoC, and ASPEED is actively developing the next generation BMC SoC PILOT 4. Nuvoton NPCM7xx, Renesas 775x and Emulex Pilot series are alternative options. Below table shows the major features among these HW solutions.

Table 1. major features among the ASPEED and Nuvoton SoCs.

	ASPEED AST2400	**ASPEED AST2500**	**ASPEED PILOT 4**	**Nuvoton NPCM7xx**
Embedded CPU	ARM9, 400MHz	ARM11, 800MHz	Dual ARM Cortex A9 500MHz	ARM Cortex A9, 800MHz
Second Processor	200MHz RISC	200MHz RISC		200MHz RISC
SDRAM Speed	DDR3 800Mbps	DDR4 1600Mbps	DDR4 1600Mbps	DDR4 1600Mbps

SDRAM Bus Width	16-Bit	16-Bit	16-Bit	16-Bit
Max. SDRAM Capacity	512MB	1GB		
System Bus Interface	PCIe 1x	PCIe 1x, Gen-2		PCIe 1x, Gen-2
PCIe Host	No	Yes	Yes	Yes
VGA/2D Controller	Yes	Yes	Yes	Yes
Flash Memory Type	SPI x2	SPI x3	eSPI	SPI x2
Virtual Media	CD, DVD, USB, Floppy	CD, DVD, USB, Floppy, NFS	CD, DVD, USB, Floppy, NFS	
KVM Re-Direction	Yes	Yes	ClearKVM	Yes
Ethernet MAC	Dual MAC	Dual MAC	Dual MAC	Dual MAC
Ethernet MAC Speed	10 / 100 / 1000M	10 / 100 / 1000M	10 / 100 / 100M	
USB over IP	Yes	Yes	Yes	Yes
VGA Resolution	1920 x 1200	1920 x 1200		
KVM Resolution	1920 x 1200	1920 x 1200		
UART	x5	x5		
Virtual UART	Yes	Yes		
MCTP over PCIe	Yes	Yes	Yes	Yes
Secure Boot	Yes	Yes	Yes	Yes
FW Support	AMI / Insyde / OpenBMC	AMI / Insyde / OpenBMC	AMI / Insyde / OpenBMC	OpenBMC

Table 1 above lists the major features supported by AST2400, AST2500, and Nuvoton NPCM7xx SoC. Due to limited information availability, some of the PILOT 4 and Nuvoton features remain unknown during this study. Renesas and Emulex are not analysed in this study due to lack of information.

2.3.2 Current SW solutions

The software solution for Platform Management Sub-system is mainly the BMC firmware stack.

There are several vendors provide the BMC firmware solution. AMI is dominating this market with its MegaRAC firmware solution. Supervyse firmware solution from Insyde, Avocent and ATEN are alternative options. Below table shows the major features among these BCM firmware solutions.

Table 2. major features among the AMI, Insyde and ATEN BMC firmware solutions.

	AMI MegaRAC	**Insyde Supervyse**	**ATEN**
Platform Support	Intel / AMD / ARM	Intel / AMD	Intel
BMC SoC Support	AST2400 / AST2500 / PILOT 4 / Nuvoton / Renesas	AST2500 / PILOT 4 / Nuvoton	AST2400 / Renesas
Out-of-Band Management Support	Yes	Yes	Yes
In-Band Management Support	Yes	Yes	Yes
Power Management	Yes	Yes	Yes
Cooling Management	Yes	Yes	Yes
IPMI Support	1.1 / 2.0	1.1 / 2.0	1.1 / 2.0
Redfish Support	Yes	Yes	No
RESTful Support	Yes	Yes	No
HTTPS Web GUI Support	Yes	Yes	Yes
SNMP Support	Yes	Yes	Yes
SMASH Support	Yes	Yes	Yes
SSH Support	Yes	Yes	No
NC-SI Support	Yes	Yes	Yes
Remote KVM Support	Yes	Yes	Yes

Remote Media re-direction	CD / DVD / USB / HD / NFS	CD / DVD / USB	CD / DVD
Serial over Lan	Yes	Yes	Yes
Remote BIOS configuration	Yes	Yes	No
IPv6 Support	Yes	No	No
Log Support	Sensor reading / SNMP trap / Email alert	Sensor reading / SNMP trap / Email alert	Sensor reading / SNMP trap / Email alert
MCTP over PCIe Support	Yes	No	No
MCTP over I2C Support	Yes	No	No
Remote BMC Firmware Up-grade	Yes	Yes	Yes
Remote BIOS Firmware Up-grade	Yes	Yes	Yes
3rd Party Authenti-cation	LDAP / RADIUS / Active Directory	LDAP / RADIUS / Active Directory	LDAP / RADIUS / Active Directory

Table 2 above lists the major features supported by AMI RegaRAC, Insyde Supervyse and ATEN BMC firmware solutions. Due to limited information availability, Avocent BMC firmware solution is not analysed in this study.

2.4 Summary

From the study analysis, a clear trend of the platform management sub-system and BMC technology innovation can be concluded. In the past, the technology in this industry was mainly defined by biggest companies and there was strong dependency on the hard-ware. In the recent years, the technology innovation is driven by standard organization with open specifications to define unified protocol, unified interface and unified data for-mat.

The hardware solution is mainly ARM based SoC, which is an optimized solution and can leverage the cost, performance and stability.

The software solution is mainly provided by 3rd party companies with propriety firmware stack. These solutions are compatible with most popular management technology such as IPMI. Some solutions also embrace the open specifications defined by standard organizations.

3 Project Specifications

This section first analyzes the 5G Telco Cloud requirements from Platform Management Subsystem and BMC's extent. Secondly, compares and analyzes the gap between 5G Telco Cloud requirements and current technology. Finally, gives a summary of project specification.

3.1 Requirement from 5G Telco Cloud

The most significant characters of 5G use cases are high data rate, scalability, ultra-low latency, ability to support a massive number of concurrent sessions and ultra-high reliability and security.

In this study, 5G Telco cloud requirement will be analyzed from below domains: Network Slicing, Cloud-native design principles, End-to-end Service Management, Edge Computing, Cloudification of the Radio Access Network (RAN), Security, and Reliability.

3.1.1 Network Slicing

Network slicing is key feature provided by 5G network to support diverse 5G use case. Network slicing is the embodiment of the concept of running multiple logical networks as virtually independent business operations on a common physical infrastructure in an efficient and economical way. This is a radical change of paradigm compared to 4G network implementations. With network slicing the 5G network can adapt to the external environment rather than the other way around [48].

Simply saying, Network Slicing is the ability to provide APIs that, using virtualization and quality of service, partition a network into diversified slices across a set of heterogeneous domains. Effectively, the same concepts used in Infrastructure as a Service are applied to the network. By creating network slices, the physical networks will be capable of carrying diverse traffic in a secure, multi-tenant fashion.

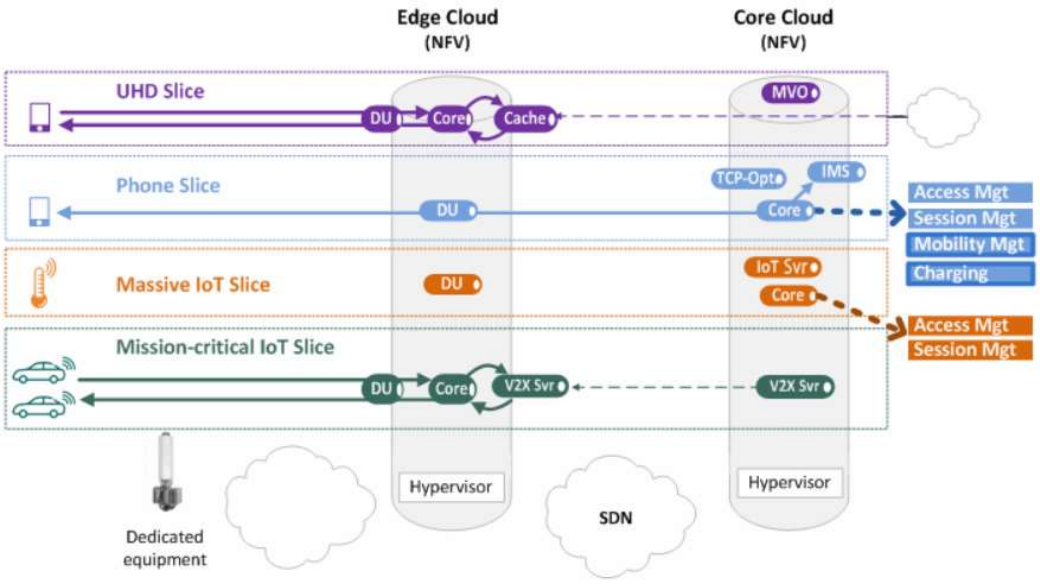


Figure 8 Network Slicing Example [49]

To enable full automation of network slice management, NFV Management & Orchestration functions (NFV-MANO) need to be complemented and interwork with slice management functions. These slice management functions can consume the Application Programming Interfaces (APIs) that fulfill the NFV reference point requirement.

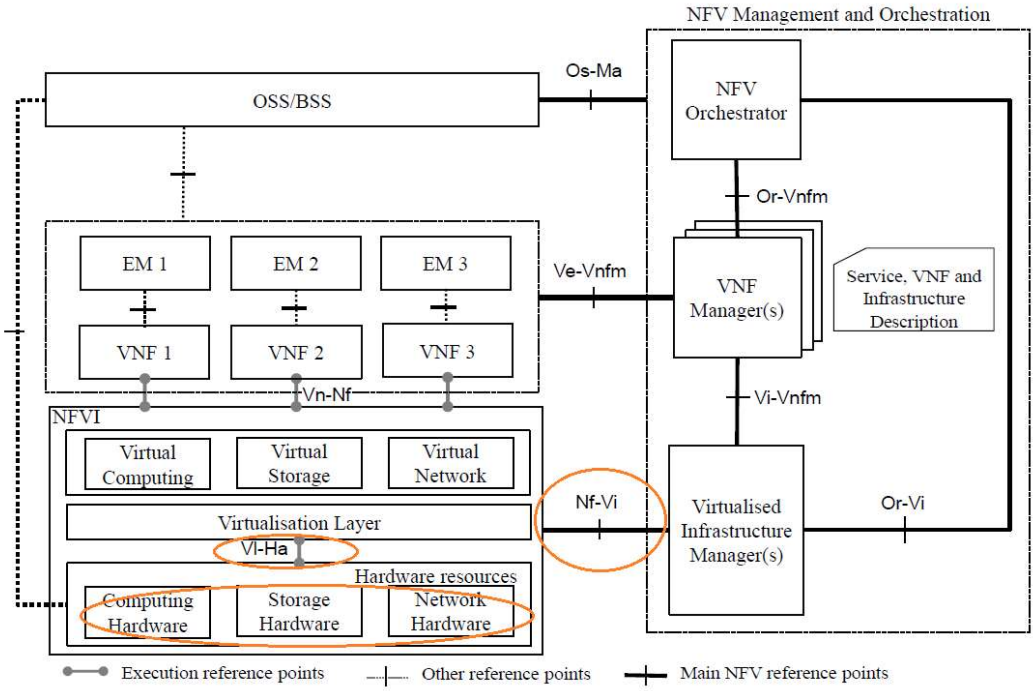


Figure 9 NFV Reference Architectural Framework [50]

NFVI - Virtualized Infrastructure Manager (Nf-Vi) and Virtualization Layer-Hardware Resource (VI-Ha) are the major reference points that will be analyzed in this study.

- [Nf-Vi]/C: This is the reference point between the management and orchestration agents in compute domain and the management and orchestration functions in the Virtual Infrastructure Management (VIM). It is the part of the Nf-Vi interface relevant to the compute domain [52 pp. 21].
- [VI-Ha]/CSr: This is the interface between the compute domain and the hypervisor domain. It is primarily used by the hypervisor/OS to gain insight into the available physical resources of the compute domain [51 pp. 21].

Table 3. major requirements for [Nf-Vi]/C and [VI-Ha]/CSr interfaces.

	[Nf-Vi]/C	**[VI-Ha]/CSr**
Band	Out-of-Band	In-Band
Hardware Layer	Ethernet	Internal host interface
Compatibility	A common information model over the standard IP packet	A common information model over different hardware bus
Interoperability	The API and tools should be able to interoperate with all management devices (MD).	The API and tools should be able to interoperate with all management devices (MD).
Portability	The API and tools should be able to reuse on other hardware resources.	The API and tools should be able to reuse on other hardware resources.
Cost Efficiency	• Obtaining, parsing and formatting of information will not generate excessive CPU utilization • NC-SI shared interface between Out-of-Band management & Host to reduce hardware cost • Serial Over Lan (SOL) support to reduce hardware cost. • KVM over IP support to reduce hardware cost	• Obtaining, parsing and formatting of information will not generate excessive CPU utilization
Current Protocol	Redfish, Swordfish, SMASH, IPMI, SNMP, YANG	MCTP, SMBIOS, Redfish, IPMI, SES-2

Table 3 above lists the major requirements for [Nf-Vi]/C and [VI-Ha]/CSr interfaces. Today, there are quite many APIs and tools available but most of them are vendor unique. There is a need for a common information model supported by non-vendor unique APIs and tools, to support fully automated network slicing in 5G deployment.

3.1.2 Cloud-Native Design

Cloud-native network functions are network functions implemented using generic IT cloud techniques beyond virtualization (e.g. functions composed from re-usable components rather than monolithic implementations of functions). The goal is to maximize efficient use of resource through bare metal infrastructure, and to embrace the advanced cloud orchestration techniques as used in IT environment [52].

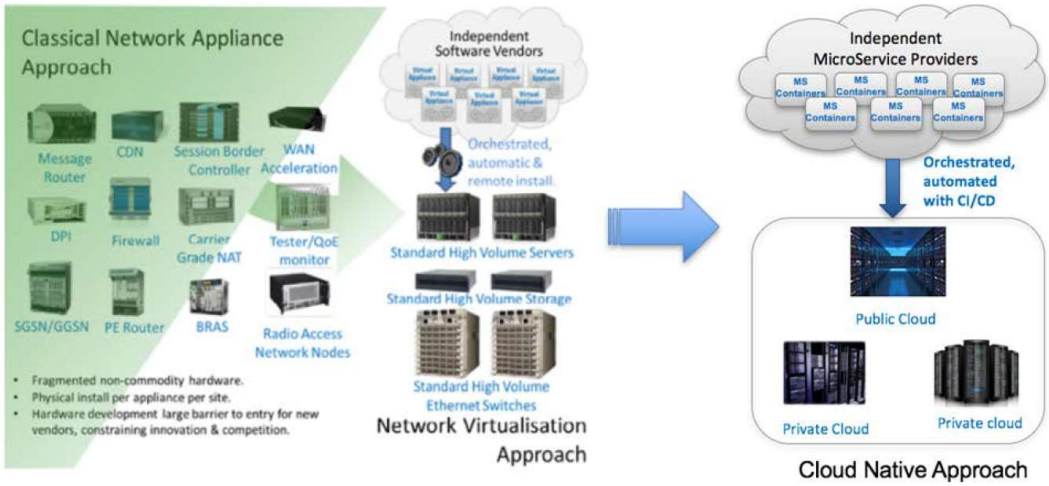


Figure 10 The Trend of Cloud Native NFV [53]

Compared with virtualization, Cloud-native design has below feature:

- Containerization: Containers are a lightweight virtualization alternative to virtual machines, it's an isolated user space in which computer programs run directly on the host operating system's kernel but have access to a restricted subset of its resources [54]. Docker and Kubernetes are most widely used container technologies.
- DevOps: DevOps is a set of software development practices that combines software development (Dev) and information technology operations (Ops) to

shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives [55].

- NFV Orchestration (NFVO): The NFVO performs resource orchestration and network service orchestration, including hardware resource and virtualized resources [50].
- Observability & Analysis: Monitoring, logging and tracing are three crucial requirements of any service deployment in general, and particularly for VNFs. They enable observability and allow the analysis of a service state.
- Networking: Traditional IP routing-like networking cannot cope with the requirements of cloud-native systems where instances of micro-services are created, moved, and stopped very quickly. Container network interface (CNI) is widely used solution for container networking.
- Messaging: Containers are not deployed in isolation. They communicate and exchange message with each other very often. REST is usually used and provides the required performance.
- Software Distribution: When installing any piece of software on any system, its origin must be authenticated, and its integrity must be verified to avoid any compromise of the system by malware or other means. TLS and digital signature are widely used technologies for secure communication and verification of the software's integrity.

From platform management subsystem and BMC perspective, a Cloud-native design would fulfill below requirements:

- The decoupling of a VNF from the underlying hardware resources.
- High availability of management capability.
- Reduce the on-site maintenance work to the minimum level.
- BMC firmware stack can support Continuous Integration (CI) [57] / Continuous Delivery (CD) type of development model [58].
- Hardware resource orchestration support.
- Monitoring & logging on the management device healthy status and system event.
- IPv6 support.
- REST type of API support.
- Digital signature inside of the BMC firmware stack.

- Transport Layer Security (TLS) cryptographic protocol over the IP packet [59].

3.1.3 End-to-End Service Management

Historically, Telco customers are used to manage the network by using Network Management tool that has been based on Element Management supported by intelligent strategies such as alarm filtering and correlation. The Network Element in legacy network is mainly static.

5G network is designed to enable different service offerings for different customers. Telco customers may be able to compose their proprietary service by selecting basic network service components to meet their specific need. Thus, an end-to-end service management model needs to provide a complete solution across a mixture of NFV functions, infrastructure, and legacy interconnected network systems, in a highly dynamic NFV environment.

NFV Management and Orchestration (MANO) has addressed the below requirements for end-to-end service management [56 pp. 19]:

- Open and standard interfaces.
- Programmable interfaces which can be automated.
- Common information models.
- Mappings to different data models.
- Ability to incorporate real-time data analytics.
- Ability to process huge amount of data.
- Ability to support run-time integration of management functions and processes.

3.1.4 Edge Computing

Support of services with ultra-low latency requirements is one of the key differentiators for 5G over previous technologies. This implies deploying 5G networks as highly distributed systems, where network functions that are the most sensitive to latency run on servers located as close as possible to end-user devices or even within such devices.

ETSI has defined Mobile Edge Computing (MEC), which be a cloud server running at the edge of a mobile network and performing specific tasks that could not be achieved with traditional network infrastructure [60].

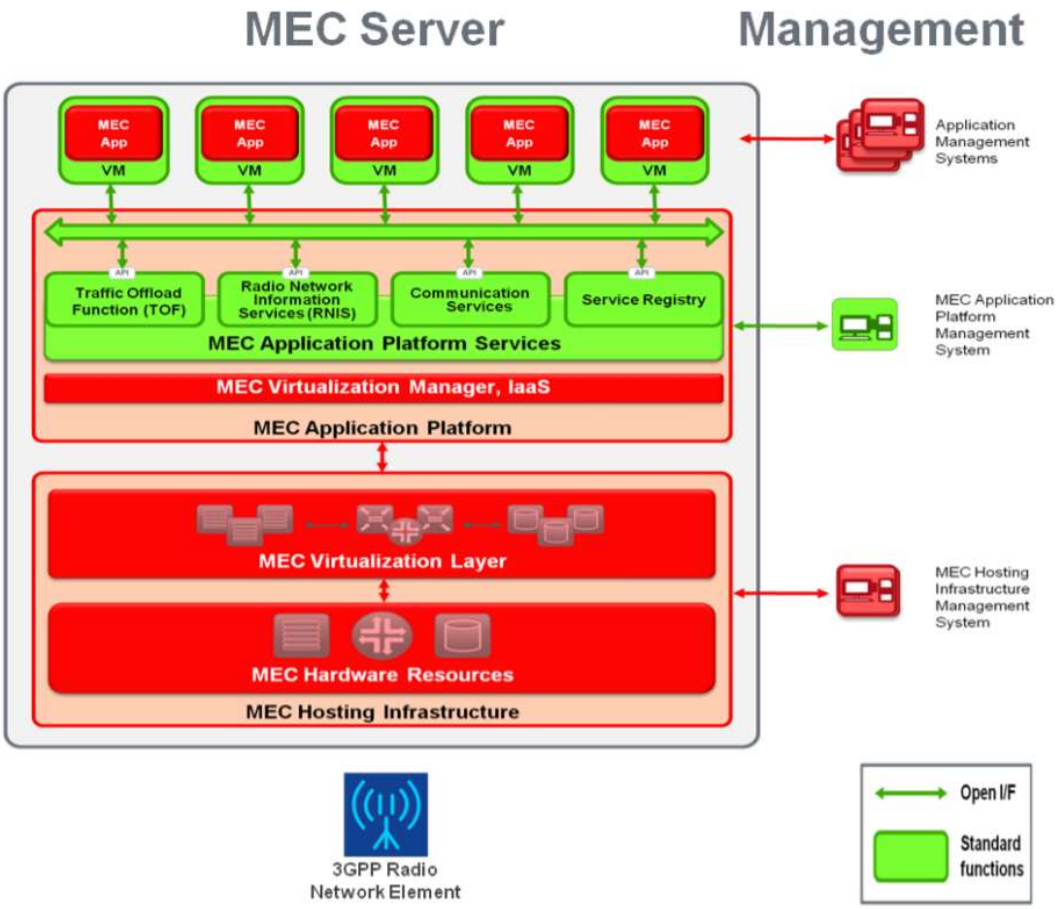


Figure 11 MEC Server Platform Overview [60]

A MEC hosting infrastructure needs to support below features to fulfil 5G Edge computing:

- Management & Orchestration: 5G network design should leverage the same Management & Orchestration (MANO) interfaces for Edge network infrastructure and Core network infrastructure.
- Remote Manageability: Edge cloud might be distributed in multiple sites; thus, all the management operations can be done remotely.
- Remote Deployment: It should be possible to automate the deployment for the service which is running on Edge cloud remotely.

- Application Portability: The MEC application should be able to port from one host to another host. Thus, the hardware-software decoupling is important requirement from Edge cloud.
- Distributed Database: A typical use case in Edge cloud is application might be running on the host server that is close to user geographically, but the storage database is distributed in other places.
- Security: there are several aspects of security concerns need to be taken into consideration.
    - Hardware security: Edge cloud might be close to end-user, so it should take advantage of the hardware security, such as Trusted Platform Modul (TPM) as much as possible.
    - Digital signature and boot guard: It's crucial to protect the edge host infrastructure with digital signature and boot guard, to ensure the bootable firmware won't be replaced by hacker.
    - Encrypted communication: The communication between Edge cloud and Core cloud might be carried over public network connection, thus the communication must be encrypted with a strong enough algorism.
    - Strong Authentication: user access to the Edge could must be authenticated with a strong enough rule.
- Acceleration: to achieve the real-time latency target, the acceleration technology will be integrated into Edge host infrastructure.

3.1.5 Cloudification of Radio Access Network (C-RAN)

Cloudification started and focused on the Core Network to underpin growth of existing systems. After showing enough success in the Core Network, cloudification is now also moving into the Radio Access Network (RAN) area. The concept of RAN cloudification includes, Radio functions virtualization, Cloud RAN (C-RAN), elastic RAN, virtualized RAN (vRAN), etc.

C-RAN is a new RAN architecture with centralized Digital Unit (DU) nodes in which all computation/processing resource could be virtualized and orchestrated, and therefore allocated on demand among different nodes [61].

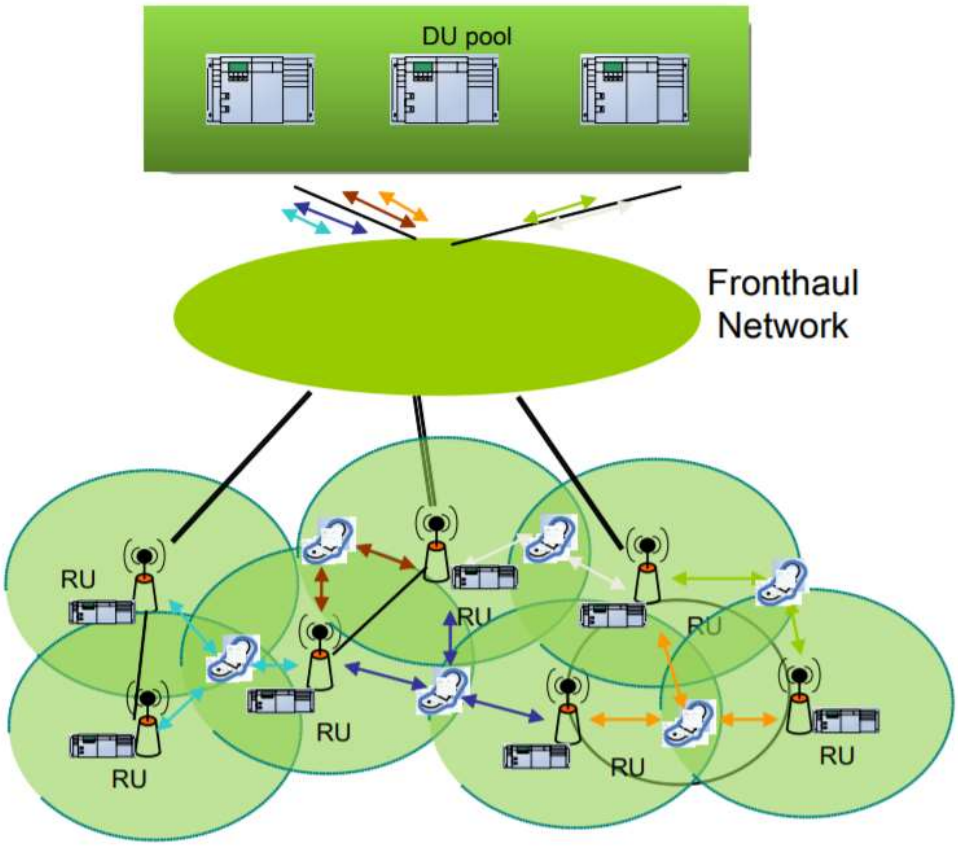


Figure 12 Cloud RAN Concept [61]

To enable the Cloud RAN implementation, the infrastructure should support below requirements:

- Network Function Virtualization on DU cloud, as defined by ETSI NFV specification.
- Management and Orchestration on DU cloud, as defined by ETSI MANO specification.
- Mobile-Edge Computing on DU cloud, as defined by ETSI MEC specification.
- Low energy consumption and dynamic power management.
- Reduce the infrastructure cost by using Commercial off-the-shelf (COTS) IT hardware.
- Support fronthaul conversion in Cloud RAN infrastructure.

Fronthaul is the intermediate links between the DU cloud and Radio Unit (RU) node. A fronthaul network can be based on different layer 1 medias, such as Wavelength Division Multiplexing (WDM) optical fiber and Microwave. The data carried on fronthaul network is usually based on Common Public Radio Interface (CPRI) or Open Base Station Architecture Initiative (OBSAI) protocols [61].

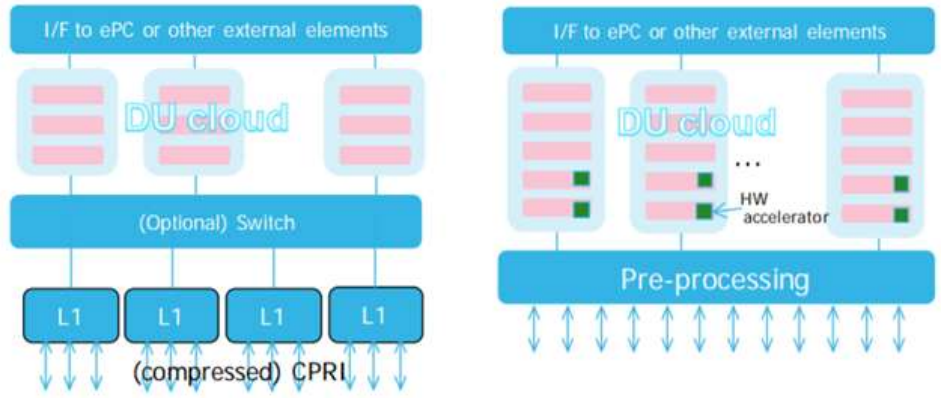


Figure 13 Fronthaul conversion standalone architecture and inside of DU cloud architecture [62]

The above figures show two different fronthaul conversion architectures for C-RAN. One is standalone architecture, where the fronthaul conversion is implemented on a standalone specific hardware which converts the CPRI or OBSAI protocol to standard Ethernet protocol. Another architecture is to convert the fronthaul inside of DU cloud, by a hardware accelerator which is an add-in card in the compute server [62].

3.1.6 Security

Today there are more and more public and private services running on top of the telecommunication networks, this makes the telecommunication networks an important pillar of economic and social development in a modern society. 5G technology is expected to have wider range of impact to the existing applications, and also enable new applications such as smart city, remote health care, etc. In overall 5G technology will continue increasing the human being productivity in many aspects. Many countries treat the telecommunication networks as fundamental infrastructure and request very high security requirement.

This study has analyzed the security content from below white papers and use cases which are defined by several standard organizations:

- NGMN Alliance 5G white paper [65].
- ETSI NFV ISG 5G white paper [52-1].
- ETSI NFV ISG Security Problem Statement [64].
- NGMN Alliance 5G Security Recommendations Packages #1 [65].
- NGMN Alliance 5G Security Recommendations Packages #2: Network Slicing [66].
- NGMN Alliance 5G Security Recommendations Packages #3: Mobile Edge Computing / Low Latency / Consistent User Experience [67].

Below requirements are application to Telco Cloud the infrastructure platform management subsystem:

- Hardware-rooted security technologies, such as Secured Boot or Trusted Boot, can enable the protection against root kits and reset attacks, and detection of unauthorized changes to BIOS, the boot sequence, and the hypervisor or OS.
- User Plane data and Control Plane data should be isolated.
- User Plane data and Control Plane data should be encrypted.
- The encryption should be performed with a strong cipher algorism.
- Weak cipher, such as linear cipher, should be avoided.
- For these use cases, multiple-hop security, where intermediate nodes need to decrypt and re-encrypt data, should be avoided.
- All the remote interfaces should be based on secure protocol and protected with strong cipher.
- User authentication of User Plane and Control Plan should be isolated. A normal subscriber user or mobile application user shall not be possible to gain authentication to the control plan.
- User privacy of User Plane should be protected, so a system administration from control plane shall not be possible to access the data of normal subscriber user or mobile application user.
- User authentication should be based on central authentication service, such as LDAP, RADIUS and Active Directory.
- Local authentication should be limited as minimum as possible. One user case is it might be needed during the deployment period only.
- The self-trusted certification should be avoided.
- User password should be stored as hash format.

- User password must fulfill a set a rule with adequate complexity and length.
- In the design of 5G, in case there will be tradeoffs between security and performance, security should be given high priority over the performance.
- Different 5G network slices will require different security polices and security protocols.
- A security protocol should not be accepted for one 5G network slice if it would be considered unacceptably weak for 5G as a whole.
- In network configuration, security parameters (such as authentication frequency) should not be set lower in any one slice than would be considered acceptably secure for the network as a whole.
- 5G latency targets can pose quite severe constraints on security mechanisms. However, operators should not compromise generic 5G security in order to reach very low latencies that are only required for specific use cases.
- For most anticipated 5G services, a target latency of 30-50ms for security mechanisms seems acceptable. In most cases, this would also be compatible with <10ms latencies on the user plane traffic itself e.g. once keys are established, basic crypto-operations can all be completed in <<10ms.
- Data exchanges between the mobile edge cloud and core cloud should be encrypted so that security standard is not compromised. The security level of the entities securing the data both at the core and the edge should ideally be assured according to a recognized scheme.
- The security level of the entities securing the mutual authentication framework both at the core and the edge should ideally be assured according to a recognized scheme.
- The authentication from mobile edge cloud should include an integrity check, to compare it with that of the original blueprint installed; this is particularly recommended in the case of premises not controlled by network operators.
- Open Source code used in the implementation should be monitored via Common Vulnerabilities and Exposures (CVE), which is a list of common identifiers for publicly known cybersecurity vulnerabilities. Software used in 5G network should be kept updating and minimizing the vulnerabilities that might pose 5G network with security risks.
- Debug and diagnostic interfaces added to network functions at design and implementation time are sometimes left accessible at run-time. These interfaces should be disabled by default and protected with strong encryption.

3.1.7 Reliability

Achieving five 9s reliability has been an important concept in telecommunications for a long time. Five 9s concept is defined as the computing infrastructure working 99.999% of the time [70]. A telco system has to be highly reliable and provides carrier-grade or 5 9s availability, corresponding to a downtime of about 5 minutes per year. Such high availability is critical as telco systems are used for emergency calls.

Ultra-Reliable and Low Latency Communications (URLLC) is an important use case of 5G. It has strict requirements for capabilities such as throughput, latency and availability. URLLC will enable several new use cases such as industrial automation, remote medical surgery, autonomous driving, etc.

To achieve five 9s availability and ultra-reliability of URLLC, the Core Network and RAN network will be cloudified and network function will be virtualized according to ETSI NFV specification. This study has analyzed the reliability, stability and resilience content from below NFV specifications:

- ETSI Network Functions Virtualization (NFV); Resiliency Requirements [70]
- ETSI Network Functions Virtualization (NFV); Reliability; Report on Models and Features for End-to-End Reliability [71]
- ETSI Network Functions Virtualization (NFV); Assurance; Report on Active Monitoring and Failure Detection [72]
- ETSI Network Functions Virtualization (NFV); Accountability; Report on Quality Accountability Framework [73]
- ETSI Network Functions Virtualization (NFV); SW Upgrade Spec [74]
- ETSI Network Functions Virtualization (NFV); Service Quality Metrics [75]

Below requirements are applicable to Telco Cloud infrastructure platform management subsystem:

- Platform management subsystem shall support the hardware status monitoring, such as temperature and voltage monitoring.
- Platform management subsystem shall support hardware fault detection.

- Platform management subsystem shall support critical hardware component management, such as cooling management, power consumption management, throttling management, to avoid of system over-heat or over-load situation.
- BMC software shall be able to provide five 9s capable stable service over years.
- BMC software shall avoid of over consuming CPU resource, memory and storage space.
- BMC software shall be able to sustain from max concurrent sessions attack or Denial-of-Service (DoS) attack.
- All services provided by BMC shall be remote accessible.
- BMC software shall support updates and upgrades.
- BMC software upgrade shall not interrupt the VNF software running on the same infrastructure.
- BMC shall support interactions with MANO or user during software upgrade.
- It shall be possible to initiate the BMC software upgrade from MANO.
- It shall be possible to save and restore the stable states of BMC software.
- It shall be possible to restore to the last known stable state automatically after BMC software upgrade.
- The deployment and modification of VNF software shall not cause interrupt to BMC serviceability.
- System event log should be properly stored and can be reported remotely.
- Quick bug fixing cycle. Bug found with a cloud-scale environment can be reproduced in lab environment and fix can be delivered quickly.

3.2 Gap Analysis

This chapter will compare the features supported by existing vendor software solutions and requirements from 5G Telco Cloud. After the analysis, the output will be the supported requirements and gaps.

Table 4. Gap analysis between existing software solutions and 5G Telco Cloud Requirements

Areas	**Supported requirements**	**Gaps**
Network Slicing	Out-of-Band interface: • Common ethernet layer.	Out-of-Band interface:

	• Common protocols such as IPMI, Redfish, SNMP. • Redfish protocol has good support on compatibility, interoperability and portability. • NC-SI support • SOL support • KVM support In-Band Interface: • Different host interface layers. • Common protocols such as IPMI, SMBIOS, SES-2	• Redfish still has feature gap compared with IPMI/SNMP in some use cases. • Lack of support to Swordfish and YANG protocols. In-Band Interface: • Lack a common host interface layer. • Lack of support on MCTP protocol.
Cloud-Native	• High availability • Reduce on-site maintenance work • Redfish has a good support on decoupling VNF from hardware • Hardware resource orchestration. • Monitoring and logging. • IPv6 • RESTful type of API • Digital signature. • TLS encryption over Redfish/SNMP	• IPMI/SNMP don't have a good support on decoupling VNF from hardware. • CI / CD due to closed software stack. • Lack of TLS encryption over IPMI.
E2E Service	• Redfish is an open, standard, and programmable interface. • Redfish has a common information model and can map to different data models. • Ability to incorporate real-time data analytics. • Ability to process huge amount of data.	• IPMI & SNMP are partial open, standard, and programmable interfaces. • IPMI & SNMP don't have a common information model, hard to map to different data models. • The Redfish firmware stack was iterated on IPMI firmware stack, which causes delay in Redfish real-time data analytics. • Partial support on run-time integration.
Edge Computing	• Redfish can offer good support to MANO on edge cloud with common data model. • Good remote manageability support. • Redfish has good support on Portability due to hardware-software decoupling. • Good hardware security support. • Digital signature and boot guard security support. • Encryption on the communication.	• IPMI/SNMP can also support MANO but lack of common data model. • Remote deployment option is limited, and performance is not satisfied. • IPMI/SNMP don't have good support on Portability. • Lack of support to distributed database. • IPMI encryption cipher is not strong, with known vulnerability. • Lack of strong encryption on remote media.

	• Strong authentication.	• There is risk that vendor default credential might be well-known in this industry. • Long lead time to fix security vulnerability due to closed software stack. • Lack of support to acceleration devices.
Cloud RAN	• Network Function Virtualization • Management and Orchestration • Low energy consumption and dynamic power management • Redfish can help to reduce the infrastructure cost	• Lack of support to fronthaul conversion devices. • Optimization for Mobile-Edge Computing is needed. • Redfish still has feature gap compared with IPMI/SNMP in some use cases.
Security	• Hardware-rooted security. • UP & CP data isolation. • UP & CP data encryption. • Some interfaces (Redfish & HTTPS) are based on secure protocol and protected with strong cipher. • Central authentication service.	• Weak cipher is still in use. • Local authentication is in use and might use vendor default credential. • Some interface (IPMI) is not based on secure protocol. • Some interface (SNMP) is based on secure protocol but usually has a weak cipher. • Self-trusted certification is still in use. • Not all user passwords are stored as hash format. • Not all user passwords have a complex rule. • Performance always have high priority over security. • Some of data exchanges between edge cloud and core cloud is not encrypted. • Long lead time to fix security vulnerability due to closed software stack. • Debug and diagnostic interface are left accessible at run-time.
Reliability	Existing software solutions have a good support on reliability: • Hardware status monitoring. • Hardware fault detection. • Cooling management. • Power consumption management. • Throttling management. • Five 9s reliability service. • Avoid of over consuming resources. • Sustain from attack.	• Existing software solution vendors intend to use old but stable software versions for reliability. • Hard to reproduce the cloud-scale bug in a lab environment and deliver fix quickly.

	• Remote accessibility. • Software upgrade support will not interrupt NVF and remote service. • Save and restore the stable state support. • System event log.	

Table 4 above lists the supported requirements and gaps between existing software solutions and 5G Telco cloud requirements.

3.3 Summary

From the study analysis, it can be concluded that existing software solutions have supported most of 5G Telco cloud requirements.

The area that has best support is reliability. It shows the existing software solution vendors are very conservative in taking new technology or new software version into use. Instead, vendors intend to use old but stable software versions to ensure the system reliability.

The area that has worst support is security, where a long list of gaps is concluded. One of reason is when there is tradeoff between security and reliability, existing software solution vendors intend to give priority to reliability over security.

The other major gaps include:

- Long lead time for software fixing and security patching.
- Hard to reproduce the cloud-scale bug in a lab environment and deliver fix quickly.
- Lack of support to fronthaul conversion device and acceleration device.
- Redfish is a great option to decouple hardware-software dependency, but Redfish still has feature gap compared with IPMI/SNMP in some use cases.
- IPMI/SNMP has hardware-software dependency, but it's still widely supported and used.
- Unsecure protocols are still in use.
- Closed software stack has limitation on CI / CD software model.
- Lack of common host interface layer and MCTP support on In-Band interface.

4 OpenBMC Framework

This section first introduces the history and goal of OpenBMC project. Secondly, gives a brief explanation of OpenBMC software architecture and repository layer structure. Thirdly, introduces OpenBMC release management and features for each release, as well as feature roadmap. Finally, gives a summary of OpenBMC framework.

4.1 OpenBMC Introduction

OpenBMC is an open software framework to build a complete Linux image for a standard Board Management Controller (BMC) firmware stack and provide full support for platform management subsystem [83].

Originally OpenBMC project was a prototype firmware stack created by Facebook during its internal hackathon event in 2014. Later, IBM and Rackspace created an in parallel project also named as OpenBMC in 2015, and with similar concept with Facebook's OpenBMC project. Started from 2018, OpenBMC became an open-source project in Linux Foundation and converged based on IBM stack [77].

IBM, Facebook, Google, Intel, Microsoft are 5 founding companies of OpenBMC project and main contributors. The Technical Steering Committee (TSC) is chaired by them together.

The goal of OpenBMC project is to define a highly extensible open-source BMC firmware stack which can work for different industries that include enterprise, High-Performance Computing (HPC), telecommunications, and cloud-scale data centers. OpenBMC project has identified 3 major reasons which make the scale of cloud deployment with traditional commercial BMC firmware impractical:

- *"Reproduce-debug-fix-deployment cycle: When an issue arises with a cloud-scale deployment, it is difficult to reproduce the same issue in a controlled lab environment. This makes it tough to find and fix the problem quickly. An open BMC stack allows faster debugging.*

- *Security models: Modern, open BMC implementations allow end users to leverage their own security models rather than forcing them to use older models with known vulnerabilities.*
- *Configuration and monitoring: Traditional BMCs required use of their own tooling. Linux, the host OS for most systems in data centers, provides standard tools that can be used to configure and monitor BMCs". [78]*

Below figure shows the traditional BMC development mode vs OpenBMC development mode.

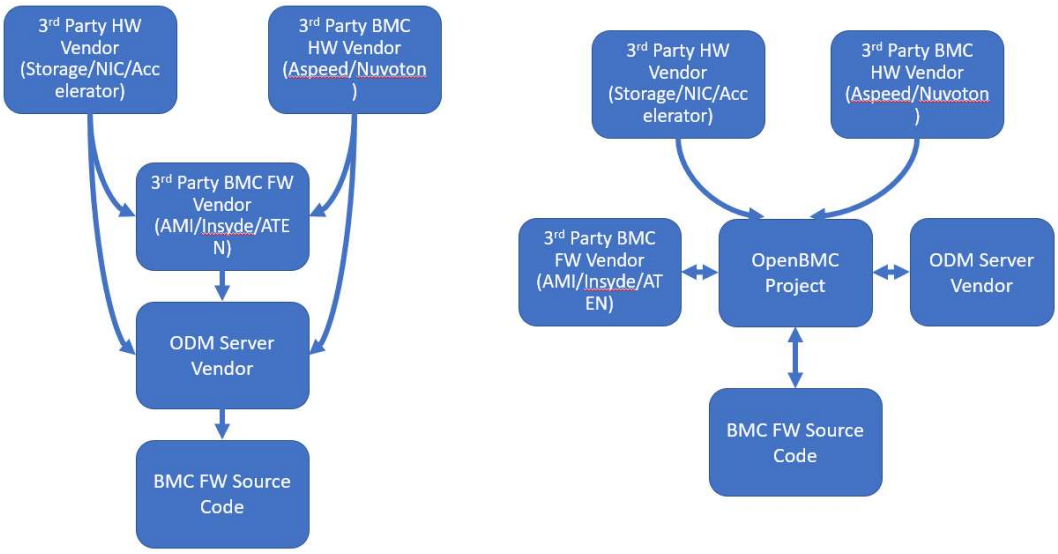


Figure 14 Traditional BMC vs OpenBMC

OpenBMC project is also a sub-project in Open Compute Project (OCP) Hardware Management project. OCP is a collaborative community led by Facebook and focused on redesigning hardware technology to efficiently support the growing demands on compute infrastructure [79].

4.2 OpenBMC Architecture

OpenBMC software architecture is designed to embrace an open source development model, as well as to support multiple SoCs and hardware boards. OpenBMC uses Yocto project as the underlying building and distribution generation framework [80]. There are 3 major blocks in OpenBMC software architecture:

- u-Boot: the boot loader which is installed in the board boot ROM, used to initialize and test the hardware.
- Linux Kernel: kernel provides the Board Support Package (BSP) which contains drivers for each hardware modules, most of drivers are SoC specific.
- Application: services, protocols and user interface applications. OpenBMC uses systemd for service management and D-Bus for Inter-Process Communication (IPC) [81].

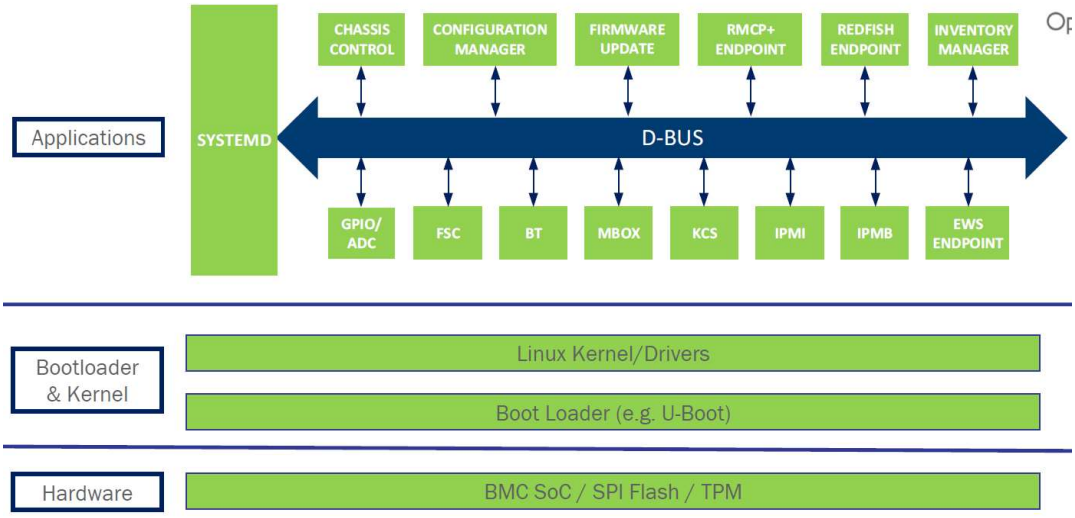


Figure 15 OpenBMC Architecture [82]

OpenBMC repository is stored in GitHub [83], which includes 3 set of layers:

- OpenBMC Common Layer - Common packages and recipes for all SoCs and hardware boards.
- SoC Layer - SoC specific drivers and tools. This layer includes the u-Boot and the Linux kernel.
- Board Specific Layer - Board specific drivers, configurations, and tools.

4.3 OpenBMC Releases & Features

After it became a Linux Foundation Project in 2018, OpenBMC project has decided to make biannual stable releases like Yocto and adopt same release naming as Yocto. The first official OpenBMC stable release (v2.6) was announced in Feb 2019. Next one (v.2.7) is planned at Aug 2019 and followed by v2.8 at Feb 2020.

The release planning & feature roadmap is based on the information from OpenBMC GitHub at 10th Apr 2019 when this chapter is written. This might be changed by OpenBMC community dynamically.

4.3.1 Prior to Stable Release

Before it became a Linux Foundation Project, OpenBMC project had an informal release v1.0 in Jun 2016 and it was not considered as a stable release. After that, OpenBMC project just tagged new releases periodically. The model at this period was like an incubation mode, even quite many applications were written by Python with the purpose of fast prototyping. The goal in this period was to design the basic framework and support fundamental features.

Table 5. OpenBMC basic framework and fundamental features list

Feature	**Description**
Linux Kernel	Stable version 4.4
SoC Support	ASPEED: AST2400, AST2500 Nuvoton: NPCM7XX Raspberry Pi
File System	Support both JFFS2 and UBI file system
Hardware Simulation	QEMU for AST2500
Automated Testing	Automated compilation & test by Jenkins CI framework. Automated system testing on simulation by Robot framework.
D-Bus Based IPC	All D-Bus interfaces defined in GitHub
Host Interface	Virtual UART Local tty mirroring
IPMI Interface	Full IPMI 2.0 compliance with DCMI
REST Interface	All REST APIs have a direct mapping to all defined D-Bus interfaces.
Web Interface	Chassis control, Event and Sensor views, Firmware Updates, Server configuration
Power Management	Power control to the payload Power capping and measurements

Cooling Management	Temperature sensor and FAN speed monitoring
Event Notification	Create dynamic web sockets to get call backs on critical events
Event Log	System event log
LEDs	LEDs status control
Boot Option	Boot option setting via IPMI, REST
Inventory Information	Host system inventory information over IPMI
Watchdog	Host watchdog support
Firmware Update Inter-faces	Support TFTP, SCP, (REST) PUT
Brick Protection	Only for UBI file system
Digital Signature	Signed image during build.
User Configuration	Support REST
Network Configuration	LAN, VLAN, DNS (Avahi), NC-SI
Time Configuration	NTP, Host/BMC control management
Remote Host Console	SSH based SOL

Table 5 above lists the basic framework and fundamental features supported by OpenBMC prior to stable release [84] [85].

4.3.2 Stable Release v2.6

In Aug 2018, the OpenBMC basic framework design was completed and was ready for production. In Feb 2019, the first formal stable release v2.6 was announced by OpenBMC project with quite a lot of new features. The feature list planned in v2.6 was quite long but just partial of them have been completed on time [85] [86] [87].

Table 6. OpenBMC Release v2.6 Complete Feature List

Feature	**Description**
Yocto refresh (2.6 Thud)	Linux 4.19 LTS
User Management	Adding local user accounts, permission

Secure Boot	Secure boot of the BMC, verification
Serial Over Lan	OBMC console, Host OS Serial port access via LAN
IPMI 2.0 Support	Refactor current IPMI 2.0 stack
Partial Redfish Support	• 2018.3 schema pack updates • Fan control configurability • Role and RoleCollection support • NetworkProtocol schema • AccountService • Managers • Systems • Chassis • Memory • Processors
GUI Enhancements	SNMP and Date/Time

Table 7. OpenBMC Release v2.6 incomplete Feature List

Feature	**Description**
KVM over IP	keyboard, video, mouse
Remote Media	File or USB stick, mass storage device
In band Firmware Update	update via host
SNMP Traps	
LDAP	LDAP backend for authentication
GUI Enhance-ments	Power, Network, LDAP config
IPv6 support	Update to Busybox 1.29 includes better IPv6 support
Security	daemons running as root, process isolation where not running as root, etc. Looking at basic security requirements, document features as a starting point, need feedback for priorities
KCS Host Inter-face	SMS and SMM host interfaces
PECI	Platform Elemental Control Interface

IPMB Support	Intelligent Platform Management Bus - protocol specification on top of i2c for multiple intelligent controllers to communicate - controllers must support multi-master i2c protocol.
Dynamic Sensor Configuration & Scanning	Defining sensor topology, platform configuration - multiplatform support
Fan Speed Control	Phosphor PID control, default fan control, pluggable framework
Remote Host JTAG Debug	JTAG debug interface via BMC agent (Intel At-Scale-Debug)
Out of Band Host State Dump	Out of band CPU crash dump via BMC. The Intel contribution is being discussed internally.
SSL Certificate Generate/Upload	

Table 6 and Table 7 above list the completed features and incomplete features in OpenBMC v2.6 respectively.

4.3.3 Stable Release v2.7

OpenBMC stable release v2.7 is planned at Aug 2019. It will include the incomplete features from release v2.6 by default, as well as support a bunch of new features [87] [88].

Table 8. OpenBMC Release v2.7 Candidate Feature List

Feature	**Description**
Yocto refresh (2.7 Warrior)	Linux 5.x
Python-Free	Re-write the Python coded applications with C/C++ code.
Refactoring for Reference Platforms	Split out platform meta from OpenBMC common and move to different repos
Redfish Remaining Schema	• LDAP, CertificateService • Firmware Update • Power & Thermal • Mutual TLS authentication • Storage & Drives Schemas • Virtual Media
MCTP Binding Support	• MCTP Serial Transport Binding • MCTP SMBus Transport Binding

PLDM Over MCTP	PLDM Over MCTP binding
NVMe Over SMBus	NVM Express Basic Management Command over SMBus
Thermal Zone Configuration	Host can fetch thermal zone configuration information from BMC.
Configuration Backup and Restore	Support by Redfish
ECC error log	ECC error logging for SEL
ASPEED P2A Secure Driver	AST2400 and AST2500 SoC driver. Disable the access from host to BMC P2A bridge by default. Only enable it when needed and limited to the smallest region.

Table 8 above lists the candidate features in OpenBMC v2.7 roadmap.

4.3.4 Stable Release v2.8 and Feature Backlog

OpenBMC stable release v2.8 is planned at Feb 2020. The content of v2.8 will be selected from OpenBMC feature backlog [86].

Table 9. OpenBMC Feature Backlog

Feature	**Description**
Yocto refresh (2.8 Zeus)	Kernel update
Documentation	Data model for state management Process for separating API distinct from implementation Admin guide Product usage guide Fan speed control
PMBus	Power Management Bus - protocol layer on top of i2c. Defined set of commands for retrieving power telemetry data, manufacturer information, enabling SMBus Alert interrupt, and power supply firmware updates.
MCTP Binding Support	• MCTP PCIe VDM Transport Binding • MCTP KCS Transport Binding • MCTP I2C Transport Binding
SSIF	SMBus System Interface

HBA Management Library	
PCIe Switch Management	NVMe Drive switching
NVMe Management	Potentially same thing as PCIe switch above
GPU Management	
Filesystem Hardening	Wrapper library for file handling that is more fault resilient
Component Recovery	FPGA, CPLD recovery over JTAG
Restricted Shell	SSH? bash restricted shell?
Power Capping	DCMI support, set power limit DCMI supported today, IPMI refactor
Controllers Console Redirection	PCIe Switch, Storage Expander
OCP LCD Debug Card Support	OCP LCD Debug card to display system info, critical logs, sensor info, POST code etc.
IPMI SEL Logger	

Table 9 above lists the features in OpenBMC backlog at the time of 10th Apr 2019. The backlog is managed by the OpenBMC release planning work group and is growing all the time.

4.4 Summary

OpenBMC framework has an extensible architecture and layer structure, it has already supported the most popular SoCs and most required common applications. Now it just has the first stable release and is ready for production in some use cases.

OpenBMC can support a new hardware board by plugging in the board specific layer. This helps to enable fast prototype and fast go-to-market.

The maturity of OpenBMC is still quite behind of traditional BMC, from both functionality and stability perspective. But OpenBMC community has offered a backlog type roadmap and the community is working actively on deliver the features step by step. Nowadays there are more and more developers from different companies are contributing the OpenBMC project.

5 Evaluation of OpenBMC for Telco Industry

This section first evaluates whether OpenBMC can satisfy 5G Telco requirements. Secondly, evaluates the OpenBMC' s ecosystem from Telco perspective. Thirdly, gives proposal of bringing OpenBMC into Telco. Finally, gives a summary of OpenBMC evaluation.

5.1 Evaluation of OpenBMC for 5G Requirement

This chapter will evaluate the features supported by OpenBMC and requirements from 5G Telco Cloud. The feature support status represented in the evaluation is based on Chapter 4.3 OpenBMC Release & Features which is written around 10th Apr 2019, it might be changed by OpenBMC community dynamically. After evaluation, the output will show how OpenBMC supports the 5G requirements, and classified as:

- Supported in OpenBMC existing releases (v2.6 or earlier releases)
- Planned in OpenBMC future releases (v2.7 or backlog)
- Not planned in OpenBMC backlog yet

Table 10. Evaluation between OpenBMC features and 5G Telco Cloud Requirements

Areas	**Supported**	**Planned**	**Not Planned**
Network Slicing	Out-of-Band interface: • Ethernet interface for OoB management. • Full IPMI 2.0. • SNMP v2/v3. • Partial Redfish support with schema 2018.3. • REST API • SSH • Basic GUI	Out-of-Band interface: • Full Redfish support with schema 2018.3 • GUI enhancement • SNMP trap • KVM over IP • Obtaining, parsing and formatting of infor-	Out-of-Band interface: • Swordfish and YANG protocols. • Reuse OpenBMC on Switch and Rack Management Controller (RMC). In-Band Interface:

	• OpenBMC can provide good support on compatibility, interoperability and portability. • NC-SI • Host console mirroring. In-Band Interface: • Virtual UART host interface • IPMI 2.0 • SMBIOS	mation will generate excessive CPU utilization In-Band Interface: • KCS host interface • MCTP-host binding • Controller console redirection • NVMe management • PCIe switch management • GPU management • HBA management • In-Band firmware update	• Ethernet-over-USB interface
Cloud-Native	• High availability • Reduce on-site maintenance work • Native support on decoupling VNF from hardware • Native CI / CD support • Hardware resource orchestration. • Monitoring and logging. • RESTful type of API • Digital signature. • TLS encryption over Redfish/SNMP	• IPv6 • Log improvement	
E2E Service	• Open, standard and programmable interfaces such as IPMI, SNMP and Redfish	• PLDM for common information model. • Performance improvement to incorporate real-time data analytics. • Performance improvement to process huge amount of data.	
Edge Computing	• Good support to MANO on edge cloud with common data model	• PLDM for common information model	• Acceleration devices management

	• Good remote manageability • Remote deployment • Good support on Portability due to hardware-software decoupling. • Digital signature and boot guard security • Encryption on the communication. • Short lead time to fix security vulnerability. • Possibility to support distributed database. • Possibility of strong encryption on remote media.	• LDAP authentication. • Remote media • Security enhancement	• Option to disable IPMI
Cloud RAN	• Network Function Virtualization • Management and Orchestration • Low energy consumption and dynamic power management • OpenBMC can help to reduce the infrastructure cost	• Redfish is not fully supported yet. • Full Redfish support with schema 2018.3	• Fronthaul conversion device management • Optimization for MEC
Security	• UP & CP data isolation. • UP & CP data encryption. • Some interfaces (Redfish & HTTPS) are based on secure protocol and protected with strong cipher. • Possibility to disable insecure interfaces. • Possibility to disable weak cipher. • Possibility to adopt strong hash encryption. • Possibility to define complex credential rule. • Possibility to decide the priority with trade-off between	• Central authentication service	• Option to disable IPMI • Option to disable SNMP v2 • Remove default password

	performance and security. • Short lead time to fix security vulnerability. • Possibility to disable debug and diagnostic interface after running. • Possibility to define better security policy for hardware interface.		
Reliability	• Hardware status monitoring. • Hardware fault detection. • Cooling management. • Power consumption management. • Throttling management. • Five 9s reliability service. • Sustain from attack. • Remote accessibility. • Software upgrade support will not interrupt NVF and remote service. • System event log. • Possibility to reproduce the cloud-scale bug in a lab environment and deliver fix quickly.	• Power Capping • Performance is compromised due to Python coded application. • It will grow with bugs and after first stable release • Configuration backup & restore • Log improvement • File system hardening • Documentation improvement	

Table 10 above list the status of OpenBMC feature roadmap from 5G Telco requirement perspective. From the roadmap of OpenBMC, the latest stable release v2.6 already has a good basic support and the future stable release v2.7 and v2.8 would be able to cover most of 5G Telco requirements. Several Telco specific features, such as fronthaul conversion device management and acceleration device management, are not yet planned in OpenBMC roadmap.

5.2 Evaluation of OpenBMC Ecosystem

This chapter will evaluate OpenBMC' s ecosystem from Telco industry's perspective. The evaluation will cover hardware ecosystem, software ecosystem and Telco cloud

ecosystem. The evaluation is based on the OpenBMC GitHub repository status at 18th Apr 2019 when this chapter is written [83].

5.2.1 Hardware Ecosystem Evaluation

The hardware ecosystem will be evaluated from 3 different aspects: system architecture, SoC and hardware board.

Firstly, from system architecture perspective, OpenBMC has supported below architectures:

- Intel x86 and x86_64 architecture
- ARM-based architecture, including Qualcomm Centriq 2400
- OpenPOWER architecture

Secondly, from SoC perspective, there is a SoC layer in OpenBMC software architecture which contains SoC specific drivers, u-Boot and tools. This extensible architecture allows OpenBMC to be used on different type of SoCs. Currently OpenBMC has supported below SoCs:

- ASPEED AST2400 SoC
- ASPEED AST2500 SoC
- The next generation of ASPEED SoC, codename PILOT 4
- Nuvoton NPCM7XX SoC
- Enclustra Mars ZX3 SoC
- Xilinx MicroBlaze, Zynq and ZynqMP SoCs
- Raspberry Pi 3 (good option for study and demo purpose)

Thirdly, from hardware board perspective, there is a board layer in OpenBMC software architecture which contains board specific drivers, configurations, and tools. This extensible architecture allows OpenBMC to be used on different hardware boards, such as server board, switch board, storage board and power board. Currently OpenBMC has supported below hardware boards:

- ASPEED evaluation board
- Nuvoton evaluation board

- OpenPOWER P8 and P9 reference boards
- Portwell Neptune development kit
- IBM Witherspoon, Romulus, Palmetto, FSP2 servers
- Intel S2600WF server
- Facebook Yosemite, Tioga Pass, YosemiteV2 servers [89]
- Facebook Wedge, Wedge100, Backpack LC/FC, Backpack CMM switches [89]
- Facebook Lightning JBOF, Bryce Canyon JBOD [89]
- Google / Racksapce / Ingrasys Zaius server
- Quanta F0B, GSJ and Q71L servers
- HXT StarDragon4800 server
- INSPUR ON5263M5 server
- Inventec LANYANG server
- Mellanox MSN switch

The architecture of OpenBMC creates a healthy ecosystem for hardware support. First it creates the opportunity for SoC vendors and hardware board vendors to upstream the own meta layer into OpenBMC GitHub. Secondly, SoC vendors and hardware board vendors will also have strong motivation to contribute on OpenBMC, since this will help to benefit their marketing and sales. Thirdly, OpenBMC solution can also be used on different system architectures and different types of hardware, which is very attractive to customers. Supporting OpenBMC will become a de facto in certain IT industries.

From Telco industry perspective, most of BMC SoCs are supported by OpenBMC already and Telco can take advantage of it directly. To move forward, Telco should work together with hardware board vendors to create the board meta layer support. One of feasible starting point is to evaluate the upstreaming code from different hardware vendors on OpenBMC GitHub, since this will give direct evidence of the compatibility and competence of hardware vendors.

5.2.2 Software Ecosystem Evaluation

The software ecosystem will be evaluated from 3 different aspects: Linux kernel, application layer, supporting frameworks.

Firstly, from Linux kernel perspective, OpenBMC is based on Linux Foundation Yocto project. By following same release cycle as Yocto update, OpenBMC will receive the latest Long-Term Support (LTS) kernel update twice a year.

Secondly, from application layer perspective, OpenBMC uses D-Bus for Inter-Process Communication (IPC) which simplifies application layer design such as information-sharing, modularity and privilege separation. This resulted a highly modular software architecture in OpenBMC and allows developer to flexible select different applications for own need. Currently OpenBMC has more than 80 application repositories in GitHub [83]. Below list gives some examples of application repositories available currently:

- D-bus interface
- Host-ipmid
- Net-ipmid
- Bmcweb
- WebUI
- Networkd
- Power-control
- Rest-server
- User-manager
- Inventory-manager
- Snmp
- Watchdog
- Certificate-manager

Most of contribution for OpenBMC application layer comes from 5 founding companies: IBM, Facebook, Google, Intel, Microsoft. As the OpenBMC ecosystem grows, traditional BMC firmware solution vendors are also attracted and started to contribute. One of example is AMI, it has active contribution to OpenBMC stable release v2.6 for IPMI 2.0 and Redfish features.

Thirdly, OpenBMC also has good ecosystem in its supporting frameworks:

- GitHub is the most important framework, it's used as code repository, community, bug reporting tool, document and wiki page for OpenBMC.

- Code review framework is based on Gerrit and running on https://gerrit.openbmc-project.xyz/q/status:open. It has a good integration with Git and very friendly HTML user interface.
- Continuous Integration (CI) framework is based on Jenkins and running on IBM server (https://openpower.xyz/job/openbmc-build/). It can build and test OpenBMC images for different targets (Palmetto, Qemu, Witherspoon, evb-ast2500, Zaius, Romulus) on a daily base automatically.
- Test Automation framework is based on Robot. All test automation scripts are also upstreaming to GitHub (https://github.com/openbmc/openbmc-test-automation).

From Telco industry perspective, the software ecosystem of OpenBMC mainly serves the 5 founding companies' interest at this moment. There are common interests shared between the 5 founding companies and Telco industry, and Telco can take big advantage from the common interests directly. But Telco requirements can't get high priority or even are not planned in the roadmap, if they are not in the priority list of 5 founding companies. To move forward, Telco should try to leverage the common interests with 5 founding companies, also start to contribute on the source code and test automation script on own interested areas.

There are two approaches that Telco can contribute OpenBMC software ecosystem. The first approach is indirect contribution, by working together with OpenBMC solution vendor and upstreaming via vendor. The second approach is direct contribution, by developing own OpenBMC stack and upstreaming via own.

5.2.3 Security Ecosystem Evaluation

The security ecosystem will be evaluated from 3 different aspects: security feature, security vulnerability, security test.

Firstly, the security feature is managed by a dedicated Security Working Group that maintains a list of security features as well as associated priorities [90]. Currently the major contribution for security features implementation and testing comes from 5 founding companies. The security working group also implements best practices from upstream projects and Open Source Software Security. This helps the security feature is implemented in an effective and standard approach.

Secondly, from security vulnerability perspective, there is a dedicated Security Response Team who follows CVE database and works on fixing vulnerabilities before public disclosure. The security response team also works together with SoC vendors and hardware board vendors, to provide security advisory to help end user the impact of security vulnerability and mitigation suggestions. One of example is very detailed security advisory for AST2400 / AST2500 specific Advanced High-performance Bus (AHB) arbitrary access via host, nickname "PantsDown" (https://github.com/openbmc/openbmc/issues/3475).

Thirdly, from security testing perspective, there are different test levels for different contributors. Developer can contribute on the scanning source code for security vulnerabilities by code inspection tool such as SonarQube [91]. Tester can contribute on the automation test script and configuration, and different kinds of penetration test. OpenBMC CI framework can be used to run security test automation and functional test automation to check the side effect.

In overall OpenBMC has a very healthy security ecosystem, covering upstreaming and patching, planning and testing. Telco industry has a very strict security requirement and different security use cases. However, the current OpenBMC security ecosystem is mainly contributed by the 5 founding companies and mainly serves their use cases. To move forward, Telco should contribute on own security use case definition and source code upstreaming, security vulnerability reporting and testing. Telco should also actively involve in the security working group discussions and meetings, to bring influence on the Telco use case during security feature planning.

5.2.4 Collaboration with other Communities

In an open source software world, the ecosystem of one project will also have strong connections with other communities, including standard organizations, forums, foundations and open source software projects. The evaluation will cover below communities: Yocto, OpenEmbedded, OCP Foundation, OpenPOWER Foundation, DMTF forum, Open System Firmware (OSF) Project, OpenStack Project.

- **Collaboration with Yocto.** Yotco is an open source collaboration project that provides templates, tools and methods to create custom Linux-based systems for embedded products regardless of the hardware architecture. OpenBMC uses

the Yocto tools to manage configuration and creation of BMC images. OpenBMC follows same release cycle as Yocto project, so the Linux kernel and tools will be refreshed from Yocto in every OpenBMC release.

- **Collaboration with OpenEmbedded.** OpenBMC uses BitBake recipes build system, which was created by OpenEmbedded project and upstreaming to Yocto project. BitBake is a make-like build tool with the special focus of distributions and packages for embedded Linux cross compilation [92]. BitBake is one of best practices for embedded Linux development.
- **Collaboration with OCP Foundation.** OpenBMC project has strong connection with OCP Foundation historically and currently. There were two different OpenBMC incubation projects in the beginning before they merged into one project. One of incubation OpenBMC project was created by Facebook and aimed to create an open software stack for Open Compute Project (OCP) hardware. Today OpenBMC is a separate project but still has strong collaboration with OCP Foundation and is a key speaker in OCT summit. For example, Open RMC is a sub-project in OCP Hardware Management Project, which aims to create similar software stack as OpenBMC on Rack Management Controller (RMC) hardware [93].
- **Collaboration with OpenPOWER Foundation.** OpenBMC project has strong connection with OpenPOWER Foundation historically and currently. The OpenPOWER Foundation is an open technical community based on the POWER architecture, enabling collaborative development and opportunity for member differentiation and industry growth [94]. One of incubation OpenBMC project was created by IBM and aimed to create an open software stack for OpenPOWER architecture servers. Today OpenBMC is a separate project, but OpenPOWER Foundation has officially marked OpenBMC as the reference firmware stack. OpenBMC has been productized and deployed in several POWER P8 and P9 servers already. OpenBMC is also a key speaker in OpenPOWER summit.
- **Collaboration with DMTF forum.** DMTF is the international standard leader of many platform management specifications such as Redfish, MCTP, PLDM, PMCI, NC-SI, etc. DMTF and OCP Foundation have alliance partner relationship. OpenBMC project is actively planning and implementing DMTF standards into its roadmap.
- **Collaboration with Open System Firmware Project.** Open System Firmware (OSF) project is a sub-project in OCP Foundation which is still at incubation

phase. OSF project is aimed to create and deploy an open source hardware platform initialization and OS load firmware, which can be used to replace current closed firmware stack such as BIOS or partial opened firmware stack such as EFI [95]. OpenBMC and OSF will support each other as a “default option” under OCP Foundation and OSF is incubated as the same way that OpenBMC has done in the past. With OpenBMC and OSF together, it’s expected to reach a truly open firmware solution in the future.

- **Collaboration with OpenStack Project.** OpenStack is the most popular cloud operating system based on open source software platform [96]. OpenStack Ironic module is a sub-program which aims to provision bare metal machines instead of virtual machines. The collaboration between OpenBMC and OpenStack Ironic mainly focus on integration with each other. For example, OpenStack Ironic has added specific patches for OpenBMC RESTful API support at early days when OpenBMC didn’t have fully IPMI 2.0 feature. Similarly, OpenBMC has put Ironic Integration as a feature in the planning.

In overall OpenBMC project has a good collaboration with the leading communities from system architecture, Linux distro, firmware solution, cloud stack and platform management system areas. The collaboration helps OpenBMC and these communities to share the best practices, achieve better compatibility with each other and create a healthy ecosystem together.

5.3 The Proposal of Bringing the OpenBMC into Telco

This chapter will evaluate the proposals of bring the OpenBMC into Telco product, from marketing, roadmap, investment, development, productization and serviceability perspective.

- **From Marketing Perspective.** Supporting OpenBMC would bring customer’s strong interest in Telco industry. Most of Telco operators would like to reuse same BMC software on different hardware for different purposes, this will help operators to reduce cost on infrastructure. One of good starting point is to show a demo OpenBMC running on Telco hardware at different conferences or summits.
- **From Roadmap perspective.** It’s proposed that Telco can follow OpenBMC stable release cycle, which is twice a year in current practice. There are two ap-

proaches to bring Telco interested features into OpenBMC roadmap. First approach is acting as a leading role for Telco interested features, this means Telco needs to implement the feature and upstream to OpenBMC trunk. By this approach, Telco will be a contributor and need sign the Corporate Contributor License Agreement (CCLA) with OpenBMC community. Second approach is acting as a follower for Telco interested features, this means Telco mainly rely on OpenBMC Technical Steering Committee (TSC) to prioritize the feature and wait the implementation from other contributors.

- **From Investment Perspective.** Bringing OpenBMC into Telco can help to reduce the investment on the hardware since the development could be done on simulator and existing hardware. But it is not cost free, there are 3 areas which require investment for this target. First investment area is supplier cost, since the board meta layer from hardware board supplier is mandatory for OpenBMC. It's also expected that traditional BMC solution vendor will provide OpenBMC solution sooner or later. Second investment area is R&D resource, mainly for the software, testing and customer documentation. The size of R&D resource would depend on whether Telco plans to build commercial OpenBMC software by own or by solution vendor. Third investment area is R&D tools, such as CI framework and Test Automation Framework.
- **From Development Perspective.** Brining OpenBMC into Telco also means changes in development mode. Although the development based on open source software is very common in Telco, but firmware development is always based on closed stack for many years. To move firmware development to open source software engineering, there are 3 areas might to be changed. The first, development model must be based on agile approach, DevOps is a good option. The second, the development tools must support code review, CI, test automation, bug reporting in an efficient manner. The third, the development team must learn how to collaborate with the community and how to utilize the developer / tester from other companies.
- **From Productization Perspective.** To bring OpenBMC into Telco for productization and commercial usage, there are 4 aspects need to be considered. The first is performance, quite many OpenBMC applications are written by Python for fast prototyping purpose but it's slow since BMC SoC doesn't have a strong processor. OpenBMC project has a plan to rewrite all Python applications with C/C++ in Release v2.7. The second is stability, compared with traditional BMC, the OpenBMC just has limited deployment and is mainly in database for web-scale

use case. The Telco use case, which has much strict stability requirement, hasn't been piloted yet. The third is quality, since the first stable Release 2.6 was just announced in Feb 2019, it's not surprised that many bugs will be reported, and it will need time to fix and improve the quality. The last is the customer document, current documents provided by OpenBMC community are mainly for developer, but not for end user.

- **From Serviceability Perspective.** Bringing OpenBMC into Telco will enhance the serviceability. The OpenBMC stack allows fast installation, configuration and debugging. It's also possible to implant the debug code on a live running OpenBMC machine, without re-compilation and re-installation, to help to find the root cause quickly.

5.4 Summary

After the evaluation of OpenBMC for Telco industry, below is the summary of evaluation:

- OpenBMC has a healthy ecosystem and good collaboration with other communities. It's growing fast and can replace traditional BMC in the future.
- Supporting OpenBMC will become a de facto in the future.
- The most popular SoC chips already support OpenBMC. SoC vendors also have the plan to support OpenBMC in next generation of SoC chips.
- Many hardware board vendors are offering hardware meta layer into OpenBMC repository.
- It's not surprised that some companies will offer commercial OpenBMC solution in the future.
- In the first stable release v2.6, a basic set of features is supported by OpenBMC. This release could be used for pilot purpose in Telco industry.
- In planned stable release v2.7, quite many important features and security fixes are planned, performance is expected to be improved after Python-based applications are rewritten to C/C++ based. This release could be used for limited deployment in Telco industry.
- The quality of OpenBMC is expected to reach at productization level at planned stable release in v2.8 or later. This release could be used for mass deployment in Telco industry.
- The documentation of OpenBMC community is mainly for developer, not ready for productization yet.

- There are still gaps in OpenBMC roadmap for Telco use case. Telco's contribution will be helpful to implement the gap features.
- Together with Open System Firmware (OSF), it's expected to reach a truly open firmware solution in the future.

6 OpenBMC Proof of Concept

This section firstly provides the OpenBMC PoC on emulation. Secondly, provides OpenBMC PoC on real Telco hardware. Finally, gives a summary of OpenBMC PoC.

6.1 OpenBMC PoC on emulation

In this chapter, the OpenBMC software will be running on emulation. OpenBMC has a good support with emulator, and the application layer in its software architecture is isolated with hardware layer. This allows the application layer can be developed with the emulation rather than real hardware. This is helpful to the open source software's ecosystem.

The first step is to build the emulator. OpenBMC uses Quick Emulator (QEMU) to perform hardware visualization. The code can be downloaded from GitHub and compiled in host operating system which is CentOS or Ubuntu.

```
# git clone https://github.com/openbmc/qemu.git
# cd qemu
# git submodule update --init dtc
# mkdir build
# cd build
# ../configure --target-list=arm-softmmu
# make
```

Listing 1. Building QEMU.

The second step is to build the OpenBMC image. OpenBMC use BitBake recipes build tool. In this example, the pre-built image from OpenBMC CI server will be used. There are several images built for different hardware, the AST2500 evaluation board image is used in this example.

```
#     wget     https://openpower.xyz/job/openbmc-build/distro=ubuntu,la-
bel=builder,target=evb-ast2500/lastSuccessfulBuild/artifact/deploy/im-
ages/evb-ast2500/flash-evb-ast2500
```

Listing 2. Downloading flash-evb-ast2500.

The third step is to boot the OpenBMC emulator. In this example, the guest is running on bridge subnet which gives benefits to avoiding of port mapping between host and guest.

```
# ./arm-softmmu/qemu-system-arm -m 256 -M ast2500-evb  -nographic -drive
   file=./flash-evb-ast2500,format=raw,if=mtd       -net       nic,mac-
   addr=C0:FF:EE:00:00:02,model=ftgmac100                          -net
   bridge,id=net0,helper=/usr/libexec/qemu-bridge-helper,br=virbr0
```

Listing 3. Booting emulator

After booting up, a lot of services will be seen on the console screen. After logged in, the system state could be checked.

```
Phosphor OpenBMC (Phosphor OpenBMC Project Reference Distro) 0.1.0 evb-ast2500 ttyS4

evb-ast2500 login: root
Password:
root@evb-ast2500:~# obmcutil state
CurrentBMCState     : xyz.openbmc_project.State.BMC.BMCState.Ready
CurrentPowerState   : xyz.openbmc_project.State.Chassis.PowerState.Off
CurrentHostState    : xyz.openbmc_project.State.Host.HostState.Off
BootProgress        : xyz.openbmc_project.State.Boot.Progress.ProgressStages.Unspecified
OperatingSystemState: xyz.openbmc_project.State.OperatingSystem.Status.OSStatus.Inactive
```

Figure 16 OpenBMC console on emulator

SSH, IPMI, REST and Redfish are supported by emulator. Several examples are given below:

```
# ssh root@${bmc}
# ipmitool -H ${bmc} -U ${user} -P ${pw} -I lanplus mc info
# curl -k -X GET https://${user}:${pw}@${bmc}/xyz/openbmc_project/list
#  curl  -k  -H  "X-Auth-Token:  $bmc_token"  -X  GET  https://${bmc}/red-
fish/v1/Managers/bmc
```

```
[root@server47 ~]# curl -k -X GET https://${user}:${pw}@${bmc}/xyz/openbmc_project/list
{
  "data": [
    "/xyz/openbmc_project/Ipmi",
    "/xyz/openbmc_project/certs",
    "/xyz/openbmc_project/certs/server",
    "/xyz/openbmc_project/certs/server/https",
    "/xyz/openbmc_project/control",
    "/xyz/openbmc_project/control/host0",
    "/xyz/openbmc_project/control/host0/TPMEnable",
    "/xyz/openbmc_project/control/host0/auto_reboot",
    "/xyz/openbmc_project/control/host0/boot",
    "/xyz/openbmc_project/control/host0/boot/one_time",
    "/xyz/openbmc_project/control/host0/power_cap",
    "/xyz/openbmc_project/control/host0/power_restore_policy",
    "/xyz/openbmc_project/control/host0/restriction_mode",
    "/xyz/openbmc_project/control/host0/turbo_allowed",
    "/xyz/openbmc_project/control/minimum_ship_level_required",
    "/xyz/openbmc_project/control/power_supply_attributes",
    "/xyz/openbmc_project/control/power_supply_redundancy",
    "/xyz/openbmc_project/dump",
    "/xyz/openbmc_project/dump/internal",
    "/xyz/openbmc_project/dump/internal/manager",
    "/xyz/openbmc_project/events",
    "/xyz/openbmc_project/inventory",
    "/xyz/openbmc_project/inventory/system",
    "/xyz/openbmc_project/led",
```

Figure 17 OpenBMC REST example on emulator

```
[root@server47 ~]# curl -k -H "X-Auth-Token: $bmc_token" -X GET https://${bmc}/redfish/v1/Managers/bmc
{
  "@odata.context": "/redfish/v1/$metadata#Manager.Manager",
  "@odata.id": "/redfish/v1/Managers/bmc",
  "@odata.type": "#Manager.v1_3_0.Manager",
  "Actions": {
    "#Manager.Reset": {
      "ResetType@Redfish.AllowableValues": [
        "GracefulRestart"
      ],
      "target": "/redfish/v1/Managers/bmc/Actions/Manager.Reset"
    }
  },
  "DateTime": "2019-05-01T04:53:44+00:00",
  "Description": "Baseboard Management Controller",
  "EthernetInterfaces": {
    "@odata.id": "/redfish/v1/Managers/bmc/EthernetInterfaces"
  },
  "FirmwareVersion": "2.7.0-dev-530-g1553ba857",
```

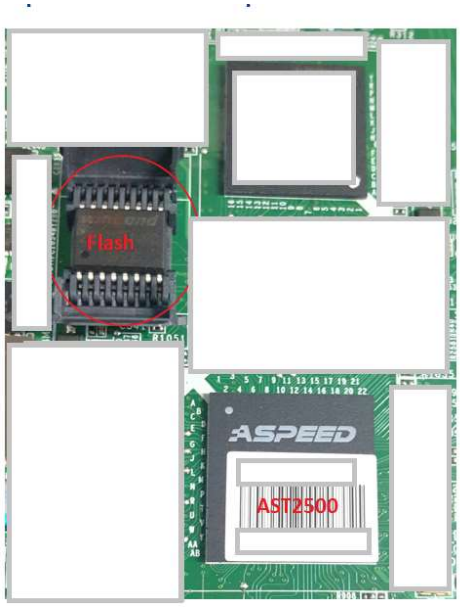
Figure 18 OpenBMC Redfish example on emulator

6.2 OpenBMC PoC on Telco hardware

In this chapter, the OpenBMC software will be running on real Telco hardware. The target hardware is based on AST2500 SoC and installed with traditional BMC originally. The target hardware is a full functional Telco infrastructure. Running OpenBMC on existing hardware will allow Telco to move from traditional BMC to OpenBMC without swapping the underlaying hardware.

The first step is to burn OpenBMC image into the BMC's flash chip by using IC programmer device. The IC programmer device must be compatible with the flash chip used on target hardware.

The second is to boot OpenBMC on real hardware. The boot loader can be the default one from OpenBMC meta-aspeed repository or can be replaced by own.




```
U-Boot

       Watchdog enabled
DRAM:
Flash:
*** Warning - bad CRC, using default environment

In:    serial
Out:   serial
Err:   serial
Net:   aspeednic#1
Error: aspeednic#1 address not set.

Hit any key to stop autoboot:  2 □□□ 1 □□□ 0
## Loading kernel from FIT Image at 20080000 ...
   Using 'conf@    openbmc.dtb' configuration
   Trying 'kernel@1' kernel subimage
     Description:  Linux kernel
     Type:         Kernel Image
     Compression:  uncompressed
     Data Start:
     Data Size:
     Architecture: ARM
     OS:           Linux
     Load Address:
     Entry Point:
```

Figure 19 Boot OpenBMC on real hardware

The third step is to access the BMC via SSH, IPMI, REST and Redfish, the same method as shown in previous chapter on emulation. With the real hardware, the valid hardware monitoring information and inventory information could be fetched.

The fourth step is to access the BMC via web application. Unlike the traditional BMC which contains a light web application by itself, OpenBMC just contains a backend web-service. The frontend must be installed on a separate machine which can communicate with OpenBMC machine via REST interface.

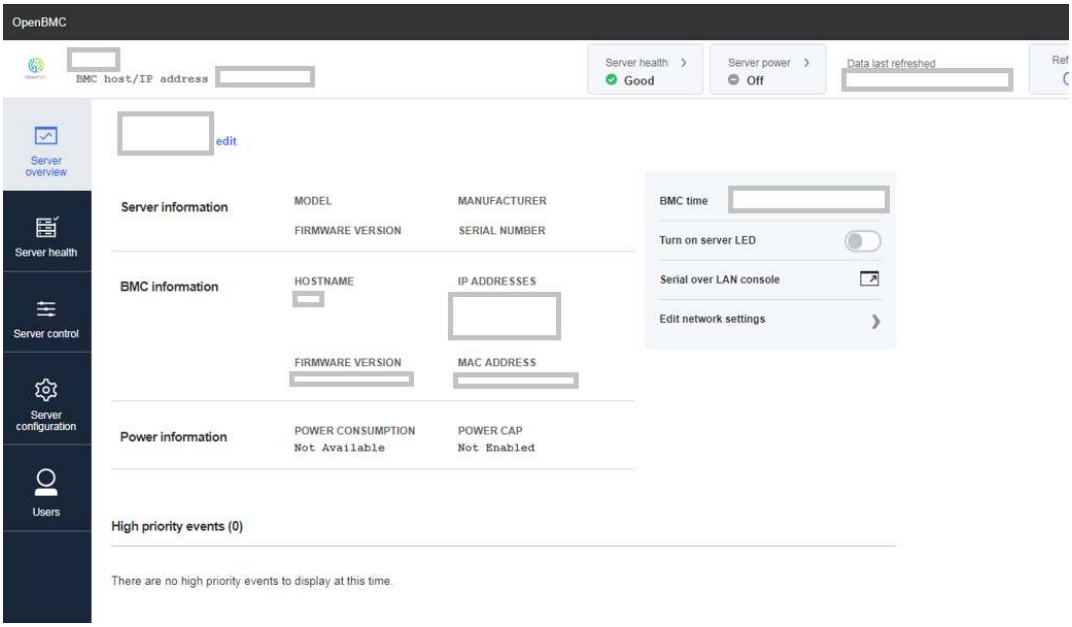


Figure 20 OpenBMC Web UI

The last step is run the KVM interface, this step can only be done on real hardware since it utilizes the integrated graphic chip inside of BMC SoC. The KVM session is over IP so it could bring the host video, keyboard and mouse remotely.

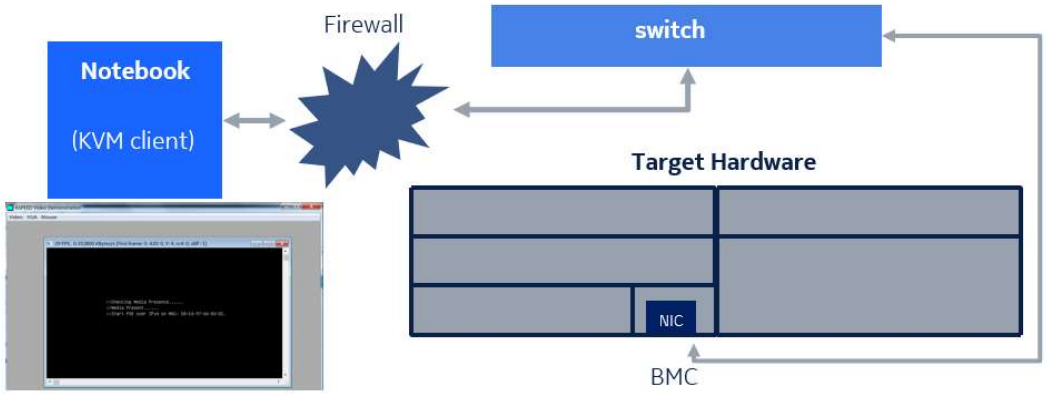


Figure 21 OpenBMC KVM

6.3 Summary

The achievements from the PoC includes:

- Better understanding to OpenBMC architecture and features with more details.
- A basic development environment was established
- Several OpenBMC emulators were created.
- Pilot on real hardware gave a positive feedback of bringing OpenBMC into Telco.
- Simple test on basic features and covering multiple interfaces.
- Many issues and imperfection were found as well.

In overall the PoC result shown a positive technical path about moving from traditional BMC to OpenBMC, as well as exposing the imperfections and protentional improvements.

7 Discussions and Conclusions

This section makes the conclusion of the study. Firstly, gives the summary of the study. Secondly, gives the evaluation of the study. Lastly, provides suggest for future steps.

7.1 Summary of the Study

The objective of this study is to evaluate the possibility to bring an Open Source software stack for platform management system and baseboard management controller in Telco cloud. This study was supposed to find solution for these technology and business problems:

- Vendor lock-in.
- Lead time for new development and bug fixing is usually long.
- Security in platform management system and BMC.

The current technology and system solution in platform management system were analyzed for this study. The analysis was made to ease understanding the status and limitations. The next step was project specification review, which was made to collect platform management system related requirement in different 5G domains and used as a base of this study. During the review, most of requirements were produced from ETSI NFV, NGMN and GSMA specification groups. After the requirements was collected for this study domain, the gaps between current solution and requirement was analyzed.

The gap analysis gave proof and details on the current technology and business problems.

OpenBMC, which is an open source BMC firmware stack and a solution to tackle current technology and business problems, was analyzed in this study. The analysis started with an introduction to OpenBMC ‘s history, objectives, development model and system architecture. Then the releases and features were reviewed and summarized, which provides a better visibility on the OpenBMC roadmap. The next was an evaluation of OpenBMC for Telco cloud. The evaluation started from gap analysis between 5G requirements and OpenBMC features, and followed by ecosystem evaluation which covered hardware, software, security and collaboration with other communities. The next step was the proposal of brining OpenBMC into Telco. The evaluation and proposal concluded the study objectives, that OpenBMC is a solution to the current technology and business problems.

The last part of this study was a Proof of Concept for OpenBMC. The PoC was made on emulation at first, to have a try on the application features which don’t rely on real hardware. Then it was made on the real Telco hardware, to have a try on the booting and some features which can’t be done on emulation. The PoC gave proof on bringing OpenBMC into Telco.

Summing up, it can be considered that the study has identified and evaluated problems, objectives, specifications and solutions.

7.2 Evaluation of the Study

The evaluation of the study compares the research objectives and technology/business problems defined in the beginning of this study against its outcome.

The objective of this study is to evaluate the possibility to bring an Open Source software stack for platform management system and baseboard management controller in Telco cloud. The outcome of this study includes the evaluation and proposal of how to bring OpenBMC into Telco cloud, from different perspective. It also includes the proof of concept of running OpenBMC on emulation and real Telco hardware.

There are 3 technology and business problems identified in the beginning of this study. The first question is vendor lock-in. Solution was studied and addressed in OpenBMC architecture chapter and ecosystem evaluation chapter. The second question is long lead time for new development and bug fixing. Solution was studied and addressed in evaluation of OpenBMC introduction for 5G requirement chapter and software ecosystem evaluation chapter. The third question is Security. Solution was studied and addressed in OpenBMC security ecosystem evaluation chapter.

Summing up, it can be considered that the outcome of this study has answered the research questions and fulfilled the objectives.

7.3 Future Steps

In the evaluation of OpenBMC for Telco industry chapter, some concerns and proposals for the future steps shown up.

The first is step is quality. It's not surprised to find many bugs in current OpenBMC since it just had the first stable release. And it would need contribution from different players, including Telco industry, to devote on the testing and integration to help OpenBMC reach a stable quality level. This will take one ~ two years.

The second step is the documentation. The current documentation from OpenBMC community is mainly for developer, not ready for commercial use yet. It would need contribution from different players, including Telco industry, to devote on the documentation creation and updating to a commercial usage level. This will take one ~ two years.

The third step is the 5G specific requirement. The current feature planning of OpenBMC focuses on the 5 Founding companies' need. It would need contribution and influence from Telco industry itself, to create the features for 5G specific requirement and devote on the development.

The fourth step is preparation for production. OpenBMC has a very healthy ecosystem, it's growing fast and supporting OpenBMC will become a de facto in the future. Even it's not mature in today, but it's time to start the production preparation.

References

1 Wikipedia, 5G. <https://en.wikipedia.org/wiki/5G>. Accessed Jan 18, 2017.

2 Dr. Michelle M. Do, KT 5G Network Architecture. <https://www.netmanias.com/en/post/oneshot/11146/5g-kt-sdn-nfv/kt-5g-network-architecture-update>. Accessed January 10, 2017.

3 SK Telecom, SK Telecom 5G White Paper. <https://www.sktelecom.com/img/pds/press/SKT_5G%20White%20Paper_V1.0_Eng.pdf>. Accessed October 20, 2014.

4 Wikipedia, Network Function Virtualization. <https://en.wikipedia.org/wiki/Network_function_virtualization>. Accessed Jan 18, 2017.

5 Wikipedia, Commercial off-the-shelf (COTS). <https://en.wikipedia.org/wiki/Commercial_off-the-shelf>. Accessed Jan 18, 2017.

6 Wikipedia, Vendor Lock-in. <https://en.wikipedia.org/wiki/Vendor_lock-in>. Accessed Jan 18, 2017.

7 Wikipedia, Software Upstream. <https://en.wikipedia.org/wiki/Upstream_(software_development)>. Accessed May 11, 2017.

8 Wikipedia, Proof of Concept. <https://en.wikipedia.org/wiki/Proof_of_concept>. Accessed Jan 20, 2017.

9 Intel, Intelligent Platform Management Interface specification v2.0 rev1.1. < https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/ipmi-intelligent-platform-mgt-interface-spec-2nd-gen-v2-0-spec-update.pdf>. Accessed Jan 20, 2017.

10 DMTF, about DMTF. < https://www.dmtf.org/about>. Accessed Mar 16, 2017.

11 DMTF, DSP2015-Platform Management Component Intercommunications (PMCI) Architecture White Paper. < https://www.dmtf.org/sites/default/files/standards/documents/DSP2015.pdf>. Accessed Mar 16, 2017

12 ESTI, Network Function Virtualisation (NFV) Infrastructure Overview. < https://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/001/01.01.01_60/gs_NFV-INF001v010101p.pdf>. Accessed Mar 25, 2017

13 DMTF, Systems Management Architecture for Server Hardware (SMASH) White Paper. < https://www.dmtf.org/sites/default/files/standards/documents/DSP2001_2.0.0.pdf>. Accessed Mar 26, 2017

14 Wikipedia, Representational state transfer. < https://en.wikipedia.org/wiki/Representational_state_transfer>. Accessed Mar 19, 2017

15 R. T. Fielding, Architectural Style and the Design of Network-based Software Architectures. < https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>. Accessed Mar 19, 2017

16 Wikipedia, Stateless Protocol. < https://en.wikipedia.org/wiki/Stateless_protocol>. Accessed Mar 19, 2017

17 DMTF, Redfish White Paper. < https://www.dmtf.org/sites/default/files/standards/documents/DSP2044_1.0.4.pdf>. Accessed Sep 30, 2018

18 SNIA, Swordfish Scalable Storage Management API Specification. < https://www.snia.org/sites/default/files/SMI/swordfish/v107a/Swordfish_v1.0.7a_Specification.pdf>. Accessed Dec 12, 2018

19 T10, Introduction to T10. < http://www.t10.org/intro.htm>. Accessed Dec 12, 2018

20 Wikipedia, SES-2 Enclosure Management. < https://en.wikipedia.org/wiki/SES-2_Enclosure_Management>. Accessed Dec 12, 2018

21 IETF, RFC6020 YANG YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). < https://tools.ietf.org/html/rfc6020>. Accessed May 21, 2018

22 Brian Hedstrom, Akshay Watwe, Siddharth Sakthidharan, Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions < https://pdfs.semanticscholar.org/5664/44aa2023ac8cf9910cc33ead8582ace4c9c4.pdf>. Accessed May 21, 2018

23 DMTF, DSP0271 YANG to Redfish Mapping Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0271_0.5.6.pdf>. Accessed May 22, 2018

24 DMTF, System Management BIOS (SMBIOS) Reference Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0134_3.2.0.pdf> Accessed Aug 28, 2018

25 DMTF, System Management BIOS. < https://www.dmtf.org/standards/smbios> Accessed Aug 28, 2018

26 DMTF, Network Controller Sideband Interface (NC-SI) Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0222_1.2.0a.pdf>. Accessed Dec 10, 2018

27 DMTF, Platform Level Data Model (PLDM) Base Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0240_1.0.0.pdf>. Accessed Apr 14, 2018

28 DMTF, Platform Level Data Model (PLDM) for Firmware Update Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0267_1.0.1.pdf>. Accessed Apr 14, 2018

29 DMTF, PLDM for FRU Data Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0257_1.0.0.pdf>. Accessed Apr 14, 2018

30 DMTF, PLDM State Set Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0249_1.0.0.pdf>. Accessed Apr 14, 2018

31 DMTF, Platform Level Data Model (PLDM) for Platform Monitoring and Control Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0248_1.1.1.pdf>. Accessed Apr 14, 2018

32 DMTF, PLDM for BIOS Control and Configuration Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0247_1.0.0.pdf>. Accessed Apr 14, 2018

33 DMTF, PLDM for SMBIOS Transfer Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0246_1.0.1.pdf>. Accessed Apr 14, 2018

34 DMTF, PLDM IDs and Codes Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0245_1.2.0.pdf>. Accessed Apr 14, 2018

35 DMTF, PLDM Over MCTP Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0241_1.0.0.pdf>. Accessed Apr 14, 2018

36 DMTF, Platform Level Data Model (PLDM) for Redfish Device Enablement Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0218_0.9.0a.pdf>. Accessed Feb 9, 2019

37 DMTF, Management Component Transport Protocol (MCTP) Overview White Paper. < https://www.dmtf.org/sites/default/files/standards/documents/DSP2016.pdf>. Accessed Apr 19, 2018

38 DMTF, MCTP Packets and NC-SI over MCTP Overview. < https://www.dmtf.org/sites/default/files/standards/documents/DSP2037_1.0.0.pdf>. Accessed Apr 19, 2018

39 DMTF, MCTP Base Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0236_1.3.0.pdf>. Accessed Apr 19, 2018

40 DMTF, NVMe™ (NVMe Express™) Management Messages over MCTP Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0235_1.0.1.pdf>. Accessed Oct 23, 2018

41 DMTF, Management Component Transport Protocol (MCTP) SMBus/I2C Transport Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0237_1.1.0.pdf>. Accessed Apr 19, 2018

42 DMTF, Management Component Transport Protocol (MCTP) PCIe VDM Transport Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0238_1.1.0.pdf>. Accessed Feb 9, 2019

43 DMTF, MCTP IDs and Codes. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0239_1.5.0.pdf>. Accessed Apr 19, 2018

44 DMTF, MCTP Serial Transport Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0253_1.0.0.pdf>. Accessed Apr 19, 2018

45 DMTF, MCTP KCS Transport Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0254_1.0.0.pdf>. Accessed Apr 19, 2018

46 DMTF, MCTP Host Interface Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0256_1.0.0.pdf>. Accessed Apr 19, 2018

47 DMTF, NC-SI over MCTP Binding Specification. < https://www.dmtf.org/sites/default/files/standards/documents/DSP0261_1.2.1.pdf>. Accessed Oct 23, 2018

48 GSMA, An Introduction to Network Slicing. < https://www.gsma.com/futurenetworks/wp-content/uploads/2017/11/GSMA-An-Introduction-to-Network-Slicing.pdf>. Accessed Apr 20, 2018

49 Dr. Harrison J. Son and Chris Yoo, E2E Network Slicing - Key 5G technology: What is it? Why do we need it? How do we implement it? < https://www.netmanias.com/en/post/blog/8325/5g-iot-network-slicing-sdn-nfv/e2e-network-slicing-key-5g-technology-what-is-it-why-do-we-need-it-how-do-we-implement-it>. Accessed January 10, 2017.

50 ETSI, Network Function Virtualisation (NFV) Architectural Framework. <https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV%20002v1.2.1%20-%20GS%20-%20NFV%20Architectural%20Framework.pdf>. Accessed Mar 28, 2017

51 ETSI, Network Function Virtualisation (NFV) Infrastructure Compute Domain. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-INF%20003v1.1.1%20-%20GS%20-%20Infrastructure%20Compute%20Domain.pdf>. Accessed Mar 28, 2017

52 ETSI, Network Function Virtualisation (NFV) 5G White Paper. < https://portal.etsi.org/nfv/nfv_white_paper_5g.pdf>. Accessed Mar 1, 2017

53 Chen YuLing and Alon Bernstein, Bridging the Gap Between ETSI-NFV and Cloud Native Architecture. < https://www.nctatechnicalpapers.com/Paper/2017/2017-bridging-the-gap-between-etsi-nfv-and-cloud-native-architecture>. Accessed Feb 4, 2018.

54 Wikipedia, Container (virtualization). < https://en.wikipedia.org/wiki/Container_(virtualization)>. Accessed Feb 3, 2018.

55 Wikipedia, DevOps. < https://en.wikipedia.org/wiki/DevOps>. Accessed Feb 3, 2018.

56 ETSI, Network Function Virtualisation (NFV) Management and Orchestration. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-MAN%20001v1.1.1%20-%20GS%20-%20Management%20and%20Orchestration.pdf>. Accessed Mar 27, 2017

57 Wikipedia, Continuous Integration. < https://en.wikipedia.org/wiki/Continuous_integration>. Accessed Feb 8, 2018.

58 Wikipedia, Continuous Delivery. < https://en.wikipedia.org/wiki/Continuous_delivery >. Accessed Feb 8, 2018.

59 Wikipedia, Transport Layer Security. < https://en.wikipedia.org/wiki/Transport_Layer_Security>. Accessed Feb 8, 2018.

60 ETSI, Mobile-Edge Computing – Introductory Technical White Paper. < https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf>. Accessed Nov 30, 2018

61 NGMN, FRONTHAUL REQUIREMENTS FOR C-RAN. < https://www.ngmn.org/fileadmin/user_upload/NGMN_RANEV_D1_C-RAN_Fronthaul_Requirements_v1.0.pdf>. Accessed Nov 30, 2018

62 NGMN, FURTHER STUDY ON CRITICAL C-RAN TECHNOLOGIES. < https://www.ngmn.org/fileadmin/user_upload/NGMN_RANEV_D2_Further_Study_on_Critical_C-RAN_Technologes_v1.0.pdf>. Accessed Nov 30, 2018

63 NGMN, 5G White Paper. < https://www.ngmn.org/fileadmin/ngmn/content/images/news/ngmn_news/NGMN_5G_White_Paper_V1_0.pdf>. Accessed Sep 19, 2018

64 ETSI, NFV Security Problem Statement. < https://www.etsi.org/deliver/etsi_gs/NFV-SEC/001_099/001/01.01.01_60/gs_NFV-SEC001v010101p.pdf>. Accessed Sep 19, 2018

65 NGMN, 5G Security Recommendations Package #1. < https://www.ngmn.org/fileadmin/user_upload/160506_NGMN_5G_Security_Package_1_v1_0.pdf>. Accessed Sep 20, 2018

66 NGMN, 5G Security Recommendations Package #1: Network Slicing. < https://www.ngmn.org/fileadmin/user_upload/160429_NGMN_5G_Security_Network_Slicing_v1_0.pdf>. Accessed Sep 20, 2018

67 NGMN, 5G Security Recommendations Package #3: Mobile Edge Computing / Low Latency / Consistent User Experience. < https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2018/180220_NGMN-5G_Security_MEC_ConsistentUExp_v2.0.pdf>. Accessed Sep 20, 2018

68 The MITRE Corporation, About CVE. < https://cve.mitre.org/about/index.html>. Accessed Sep 22, 2018

69 Wikipedia, High Availability. < https://en.wikipedia.org/wiki/High_availability>. Accessed Mar 17, 2018

70 ETSI, Network Functions Virtualisation (NFV); Resiliency Requirements. < https://www.etsi.org/deliver/etsi_gs/NFV-REL/001_099/001/01.01.01_60/gs_NFV-REL001v010101p.pdf>. Accessed Sep 24, 2018

71 ETSI, Network Functions Virtualisation (NFV); Reliability; Report on Models and Features for End-to-End Reliability. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-REL%20003v1.1.2%20-%20GS%20-%20E2E%20reliability%20models%20report.pdf>. Accessed Sep 24, 2018

72 ETSI, Network Functions Virtualisation (NFV); Assurance; Report on Active Monitoring and Failure Detection. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-REL%20004v1.1.1%20-%20GS%20-%20Active%20monitoring%20and%20failure%20detection%20report.pdf>. Accessed Sep 24, 2018

73 ETSI, Network Functions Virtualisation (NFV); Accountability; Report on Quality Accountability Framework. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-REL%20005v1.1.1%20-%20GS%20-%20Quality%20Accountability%20Framework.pdf>. Accessed Sep 24, 2018

74 ETSI, Network Functions Virtualisation (NFV); Reliability; Maintaining Service Availability and Continuity Upon Software Modification. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-REL%20006v3.1.1%20-%20GS%20-%20SW%20Upgrade%20spec.pdf>. Accessed Sep 24, 2018

75 ETSI, Network Functions Virtualisation (NFV); Service Quality Metrics. < https://docbox.etsi.org/ISG/NFV/Open/Publications_pdf/Specs-Reports/NFV-INF%20010v1.1.1%20-%20GS%20-%20NFV%20Service%20Quality%20Metrics.pdf>. Accessed Sep 24, 2018

76 GitHub, Facebook OpenBMC project. < https://github.com/facebook/openbmc>. Accessed Aug 14, 2018

77 Wikipedia, OpenBMC. < https://en.wikipedia.org/wiki/OpenBMC>. Accessed Feb Aug, 2018

78 Jim Zemlin the Executive Director at The Linux Foundation, OpenBMC Project Community Comes Together at The Linux Foundation to Define Open Source Implementation of BMC Firmware Stack. < https://www.linuxfoundation.org/blog/2018/03/openbmc-project-community-comes-together-at-the-linux-foundation-to-define-open-source-implementation-of-bmc-firmware-stack/>. Accessed Aug 28, 2018

79 Open Compute Project, About OCP. < https://www.opencompute.org/about>. Accessed Aug 28, 2018

80 Yocto Project, About the Yocto Project. < https://www.yoctoproject.org/about/>. Accessed Sep 1, 2018

81 Havoc Pennington, David Wheeler, John Palmieri, Colin Walters, D-Bus Tutorial. < https://dbus.freedesktop.org/doc/dbus-tutorial.html#whatis>. Accessed Aug 21, 2018

82 James Mihm, Sai Dasari, OCP18 OpenBMC Status Update. < https://f990335bdbb4aebc3131-

b23f11c2c6da826ceb51b46551bfafdc.ssl.cf2.rackcdn.com/images/6b289067c77f7bbac31d67104326e6e270fd27aa.pdf>. Accessed Sep 1, 2018

83 GitHub, OpenBMC Repository. < https://github.com/openbmc>. Accessed Sep 1, 2018

84 Andrew Geissler, OCP18 OpenBMC State of Development. < https://www.opencompute.org/files/OCP18-OpenBMC-State-of-Development.pdf>. Accessed Mar 21, 2019

85 OpenBMC Project, OpenBMC Release Notes. < https://github.com/openbmc/docs/blob/master/release/release-notes.md>. Accessed Mar 22, 2019

86 OpenBMC Project, OpenBMC Release Priorities. < https://docs.google.com/spreadsheets/d/1lts-YX8J_AnS2dKfhW649OQHijzJRiJWHTDpKY62Hk/edit#gid=0>. Accessed Apr 10, 2019

87 Brad Bishop, OCP19 OpenBMC Status Update. < https://github.com/openbmc/openbmc/wiki/files/OCPGLO19_OpenBMC_Project_Update.pdf>. Accessed Apr 1, 2019

88 OpenBMC Project, OpenBMC Release Labels v2.7. < https://github.com/openbmc/openbmc/labels/Release2.7>. Accessed Apr 1, 2019

89 GitHub, Facebook OpenBMC Repository. < https://github.com/facebook/openbmc>. Accessed Apr 3, 2019

90 GitHub, OpenBMC Security Working Group. < https://github.com/openbmc/openbmc/wiki/Security-working-group>. Accessed Apr 5, 2019

91 SonarQube, About SonarQube. < https://www.sonarqube.org/about/>. Accessed Apr 5, 2019

92 Wikipedia, BitBake. < https://en.wikipedia.org/wiki/BitBake>. Accessed Apr 14, 2019

93 OCP, Hardware Management/Open RMC. < https://www.opencompute.org/wiki/Hardware_Management/Open_RMC>. Accessed Apr 14, 2019

94 OpenPOWER Foundation, About OpenPOWER Foundation. < https://openpowerfoundation.org/about-us/ >. Accessed Apr 15, 2019

95 OCP, Open System Firmware. < https://www.opencompute.org/projects/open-system-firmware>. Accessed Apr 15, 2019

96 OpenStack Foundation, What is OpenStack. < https://www.openstack.org/software/>. Accessed Apr 15, 2019