UNIT – II

IOT & M2M: Introduction, M2M, Difference between IoT and M2M, SDN and NFV for IoT, Need for IoT Systems Management, Simple Network Management Protocol (SNMP), Limitations of SNMP, Network Operator Requirements, NETCONF, YANG, IoT Systems Management with NETCONF-YANG, NETOPEER.

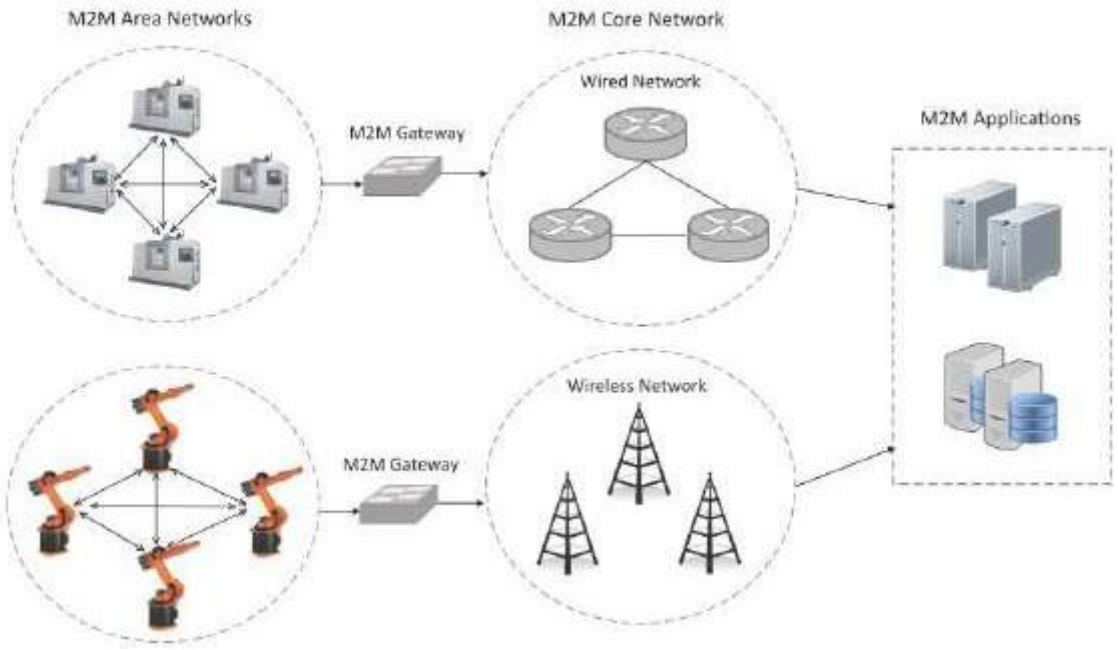
IOT & M2M

M2M:

Machine-to-Machine (M2M) refers to networking of machines (or devices) for the purpose of remote monitoring and control and data exchange.

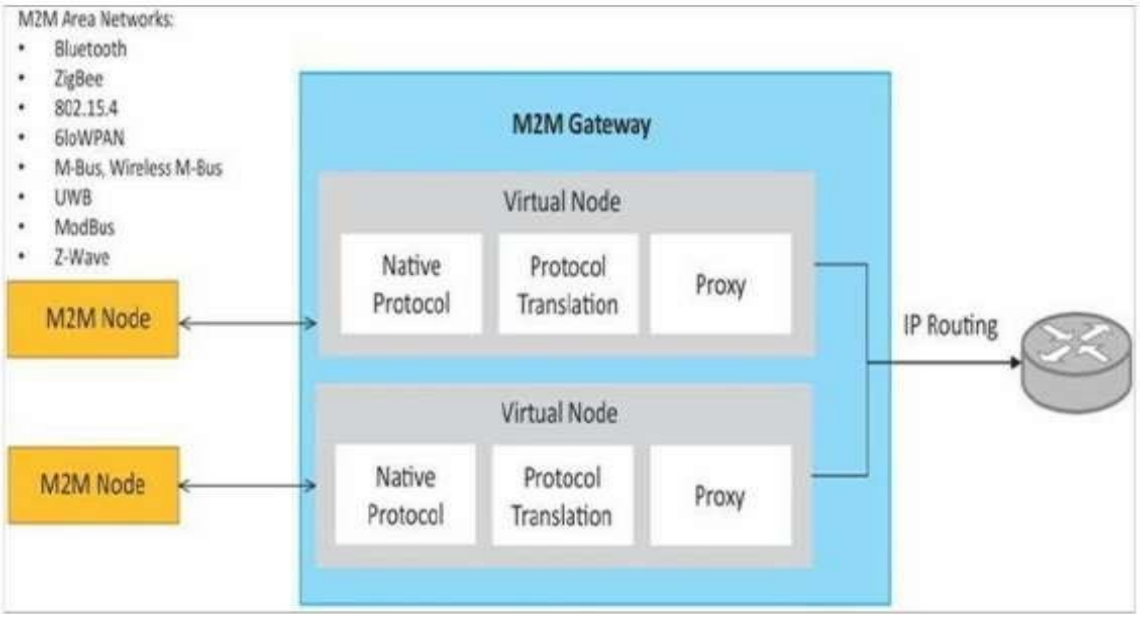
1. Term which is often synonymous with IoT is Machine-to-Machine (M2M).
2. IoT and M2M are often used interchangeably.

The following figure shows the end-to-end architecture of M2M systems comprises of M2M area networks, communication networks and application domain.



- An M2M area network comprises of machines (or M2M nodes) which have embedded network modules for sensing, actuation and communicating various communication protocols can be used for M2M LAN such as ZigBee, Bluetooth, M-bus, Wireless M-Bus etc., These protocols provide connectivity between M2M nodes within an M2M area network.

- The communication network provides connectivity to remote M2M area networks. The communication network can use either wired or wireless network (IP based). While the M2M are networks use proprietary or non-IP based communication protocols, the communication network uses IP-based network. Since non-IP based protocols are used within M2M area network, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M are network, M2M gateways are used.



The above figure shows a block diagram of an M2M gateway. The communication between M2M nodes and the M2M gateway is based on the communication protocols which are native to the M2M are network. M2M gateway performs protocol translations to enable Ip-connectivity for M2M are networks. M2M gateway acts as a proxy performing translations from/to native protocols to/from Internet Protocol (IP). With an M2M gateway, each mode in an M2M area network appears as a virtualized node for external M2M area networks.

Differences between IoT and M2M

1) Communication Protocols:

- Commonly uses M2M protocols include ZigBee, Bluetooth, M-Bus,Wireless M-Bus etc.,
- In IoT uses HTTP, REST, WebSocket , MQTT ,XMPP ,DDS ,AMQP etc.,

2) Machines in M2M Vs Things in IoT:

- Machines in M2M will be homogenous whereas Things in IoT will be heterogeneous.

3) Hardware Vs Software Emphasis:

- The emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.

4) Data Collection &Analysis:

- M2M data is collected in point solutions and often in on-premises storage infrastructure.
- The data in IoT is collected in the cloud (can be public, private or hybrid cloud).

5) Applications:

- M2M data is collected in point solutions and can be accessed by on-premises applications such as diagnosis applications, service management applications, and on-premisis enterprise applications.
- IoT data is collected in the cloud and can be accessed by cloud applications such as analytics applications, enterprise applications, remote diagnosis and management applications, etc.

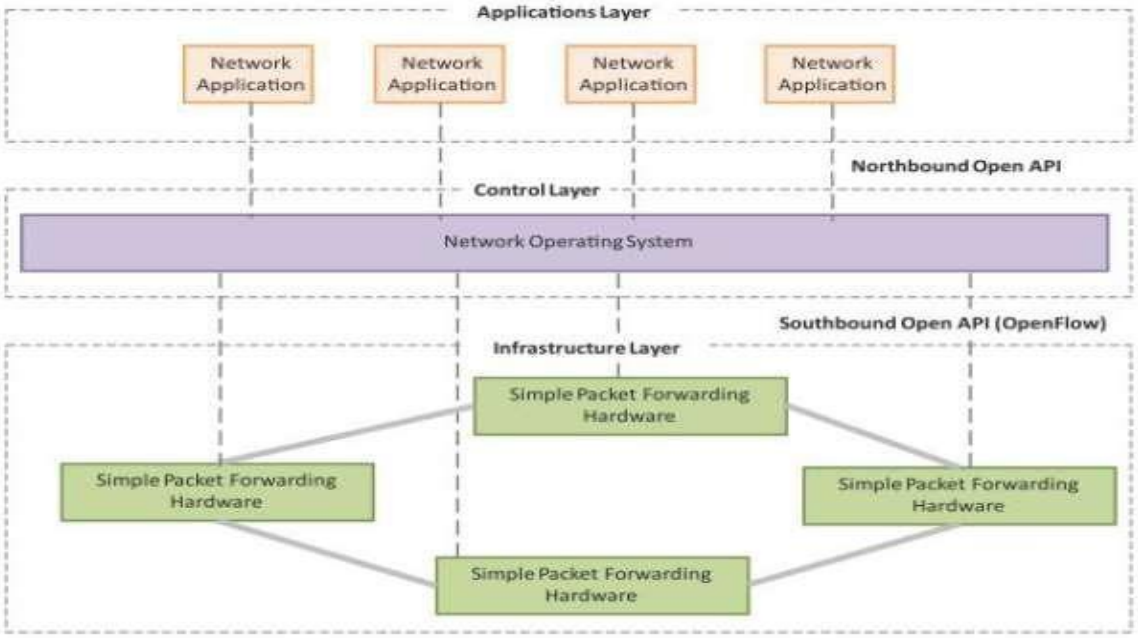
SDN and NVF for IoT:

Software Defined Networking (SDN):

a) This is a networking architecture that separates the control plane from the data plane and centralizes the network controller.

b) Software-based SDN controllers maintain a united view of the network

c) The underlying infrastructure in SDN uses simple packet forwarding hardware as opposed to specialized hardware in conventional networks.



SDN Architecture:

Key Elements of SDN:

1) Centralized Network Controller

With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network.

2) Programmable Open APIs

SDN architecture supports programmable open APIs for interface between the SDN application and control layers (Northbound interface).

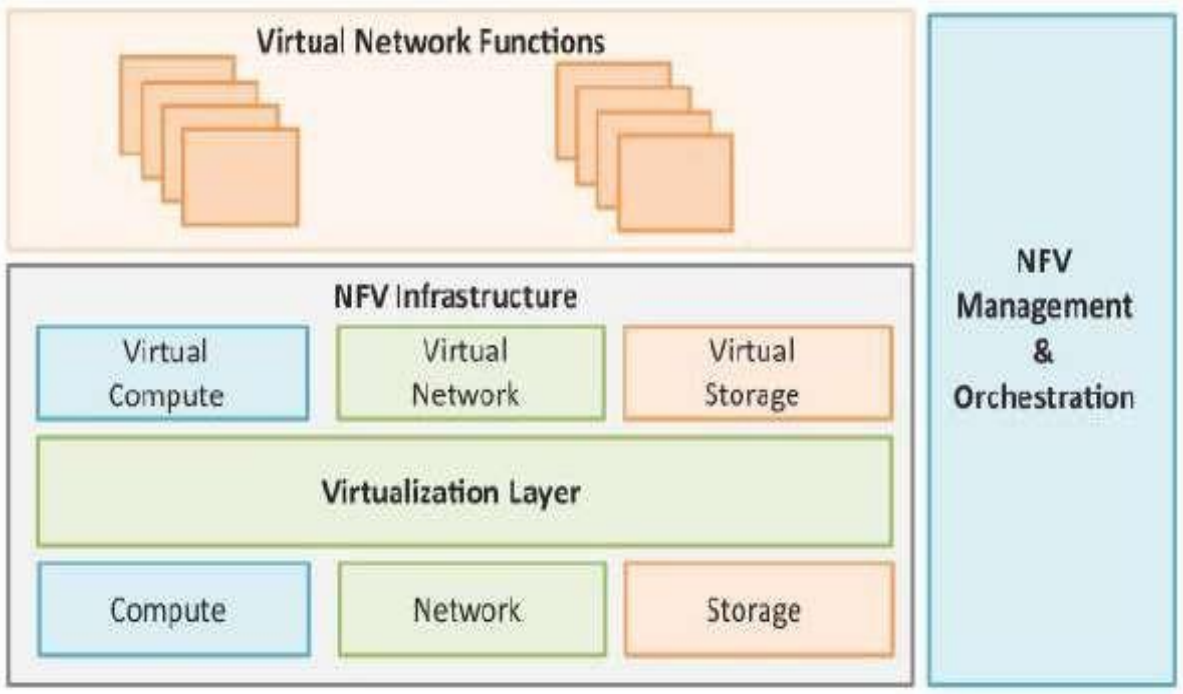
3) Standard Communication Interface (OpenFlow)

SDN architecture uses a standard communication interface between the control and infrastructure layers (Southbound interface). OpenFlow, which is defined by the Open Networking Foundation (ONF) is the broadly accepted SDN protocol for the Southbound interface.

Network Function Virtualization (NFV):

a) Network Function Virtualization (NFV) is a technology that leverages virtualization to consolidate the heterogeneous network devices onto industry standard high volume servers, switches and storage.

b) NFV is complementary to SDN as NFV can provide the infrastructure on which SDN can run.



Key Elements of NFV:

NFV Architecture:

1) Virtualized Network Function (VNF):

VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).

2) NFV Infrastructure (NFVI):

NFVI includes compute, network and storage resources that are virtualized.

3) NFV Management and Orchestration:

NFV Management and Orchestration focuses on all virtualization-specific management tasks and covers the orchestration and life-cycle management of physical and/or software resources that support the infrastructure virtualization, and the life-cycle management of VNFs.

Need for IoT Systems Management:

Managing multiple devices within a single system requires advanced management capabilities.

1) Automating Configuration: IoT system management capabilities can helpin automating the system configuration.

2) Monitoring Operational & Statistical Data: Management systems can help in monitoring operational and statistical data of a system. This data can be used for fault diagnosis or prognosis.

3) Improved Reliability: A management system that allows validating the system configurations before they are put into effect can help in improving the system reliability.

4) System Wide Configurations: For IoT systems that consists of multiple devices or nodes, ensuring system wide configuration can be critical for the correct functioning of the system.

5) Multiple System Configurations: For some systems it may be desirable to have multiple valid configurations which are applied at different times or in certain conditions.

6) Retrieving & Reusing Configurations: Management systems which have the capability of retrieving configurations from devices can help in reusing the configurations for other devices of the same type.

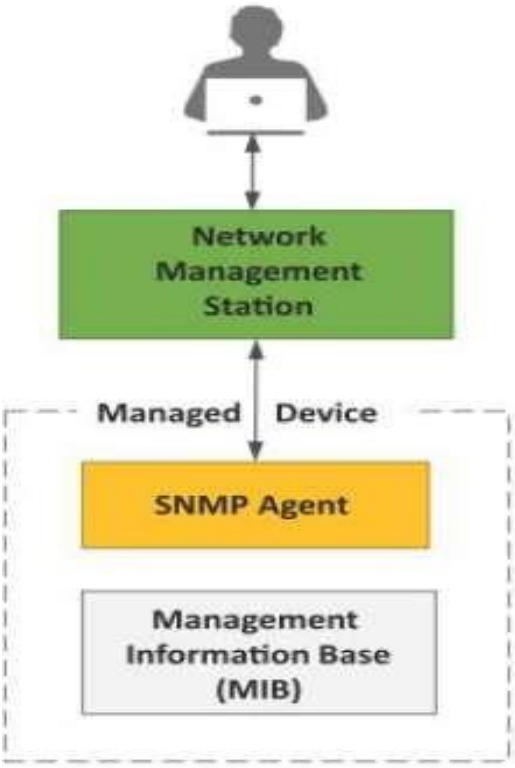
Simple Network Management Protocol (SNMP):

SNMP is a well-known and widely used network management protocol that allows monitoring and configuring network devices such as routers, switches, servers, printers, etc.

Limitations of SNMP:

a) SNMP is stateless in nature and each SNMP request contains all the information to process the request. The application needs to be intelligent to manage the device.
b) SNMP is a connectionless protocol which uses UDP as the transport protocol, making it unreliable as there was no support for acknowledgement of requests.
c) MIBs often lack writable objects without which device configuration is not possible using SNMP.
d) It is difficult to differentiate between configuration and state data in MIBs.
e) Retrieving the current configuration from a device can be difficult with SNMP.
f) Earlier versions of SNMP did not have strong security features.

The components of SNMP are Network Management Station (NMS), Managed Device, Management Information Base (MIB), and SNMP Agent that runs on the device and is described in the following diagram:



Network Operator Requirements:

The following point provides a brief overview of the operator requirements.

1) Ease of use: From the operator point of view, ease of use is the key requirement for any network management technology.

2) Distinction between configuration and state data: Configuration data is the set of writable data that is required to transform the system from its initial state to its current state.

State data is the data which is not configurable. State data includes operational data which is collected by the system at runtime and statistical data which describes the system performance. For an effective management solution, it is important to make a clear distinction between configurations and sate data.

3) Fetch configuration and state data separately: It should be possible to fetch the configuration and state data separately from the managed device. This is useful when the configuration and state data from the different devices needs to be compared.

4) Configuration of the network as a whole: This is important for systems which have multiple devices and configuring them within one network wide transaction is required to ensure the correct operation of the system.

5) Configuration transactions across devices: Multiple devices should be supported to configure transactions across the devices.

6) Configuration deltas: It should be possible to generate the operations necessary for going from one configuration state to another. The device should support configuration deltas with minimal state changes.

7) Dump and restore configurations: It should be possible to dump configurations from devices and restore configurations to devices.

8) Configuration validation: It should be possible to validate configurations.

9) Configuration database schemas: There is a need for standardized configuration database schemas or data models across operators.

10) Comparing configurations: Devices should not arbitrarily reorder data, so that it is possible to use the text processing tools such as ***diff*** to compare configurations.

11) Role-based access control: Devices should support role-based access control model, so that a user is given the minimum access necessary to perform a required task.

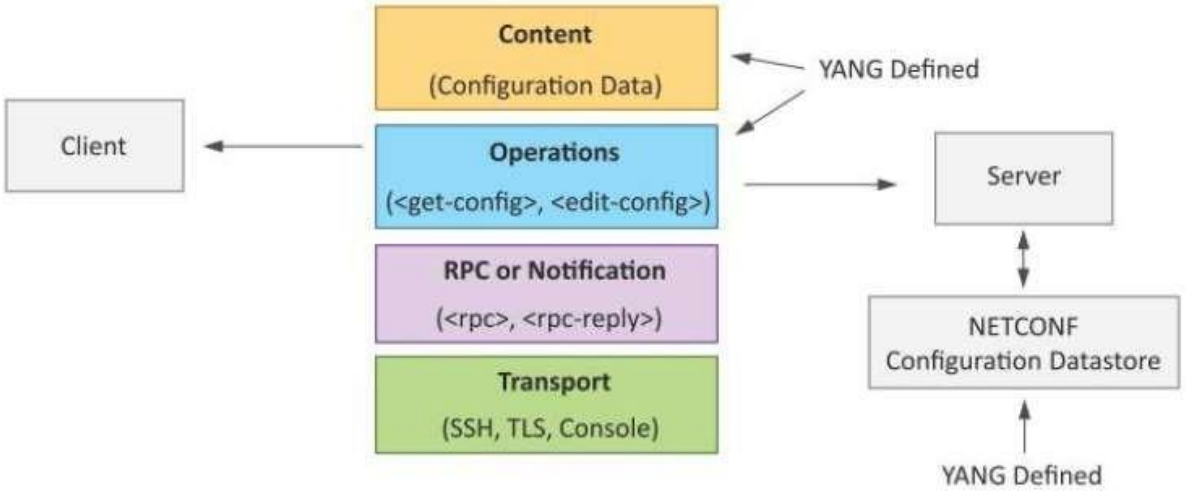
12) Consistency of access control lists: It should be possible to do consistency checks of access control lists across devices.

13) Multiple configuration sets: There should be support for multiple configuration sets on devices. This is the way to differentiate between candidate and active configurations.

14) Support for both data-oriented and task oriented access control: While SNMP access control is data oriented, CLI access control is usually task oriented. There should be support for both types of access control.

NETCONF:

Network Configuration Protocol (NETCONF) is a session-based network management protocol. NETCONF allows retrieving state or configuration data and manipulating configuration data on network devices.



1) NETCONF works on SSH transport protocol.
2) Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.
3) NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.
4) The RPC layer provides mechanism for encoding of RPC calls and notifications.
5) NETCONF provides various operations to retrieve and edit configuration data from network devices.
6) The Content Layer consists of configuration and state data which is XML-encoded.
7) The schema of the configuration and state data is defined in a data modeling language called YANG.
8) NETCONF provides a clear separation of the configuration and state data.
9) The configuration data resides within a NETCONF configuration datastore on the server.

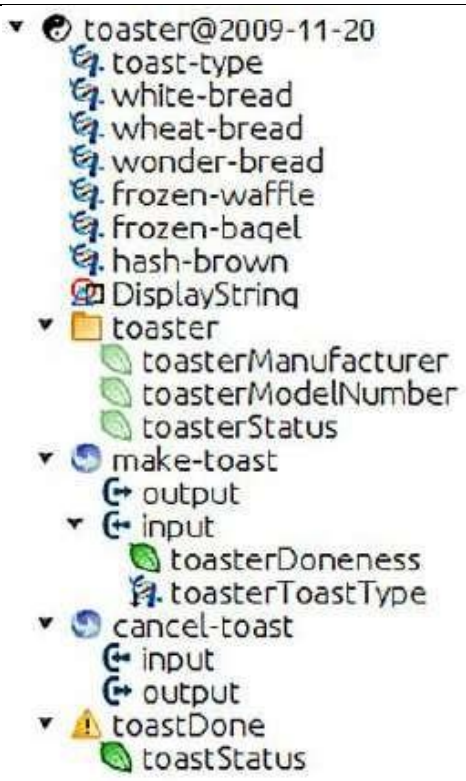
YANG:

1) YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol
2) YANG modules contain the definitions of the configuration data, state data, RPC calls that can be issued and the format of the notifications.
3) YANG modules defines the data exchanged between the NETCONF client and server.

4) A module comprises of a number of 'leaf' nodes which are organized into a hierarchical tree structure.
5) The 'leaf' nodes are specified using the 'leaf' or 'leaf-list' constructs.
6) Leaf nodes are organized using 'container' or 'list' constructs.
7) A YANG module can import definitions from other modules.
8) Constraints can be defined on the data nodes, e.g. allowed values.
9) YANG can model both configuration data and state data using the 'config' statement.

YANG Module Example (Visual Representation of the Toaster Yang Module)

a. This YANG module is a YANG version of the toaster MIB
b. The toaster YANG module begins with the header information followed by identity declarations which define various bread types.
c. The leaf nodes ('toasterManufacturer', 'toasterModelNumber' and toasterStatus') are defined in the 'toaster' container.
d. Each leaf node definition has a type and optionally a description and default value.
e. The module has two RPC definitions ('make-toast' and 'cancel-toast').



IoT Systems Management with NETCONF-YANG:

YANG is a data modeling language used to model configuration and state data manipulated by the NETCONF protocol. The generic approach of IoT device management with NETCONF-YANG. The roles of various components are:

1) Management System
2) Management API
3) Transaction Manager
4) Rollback Manager

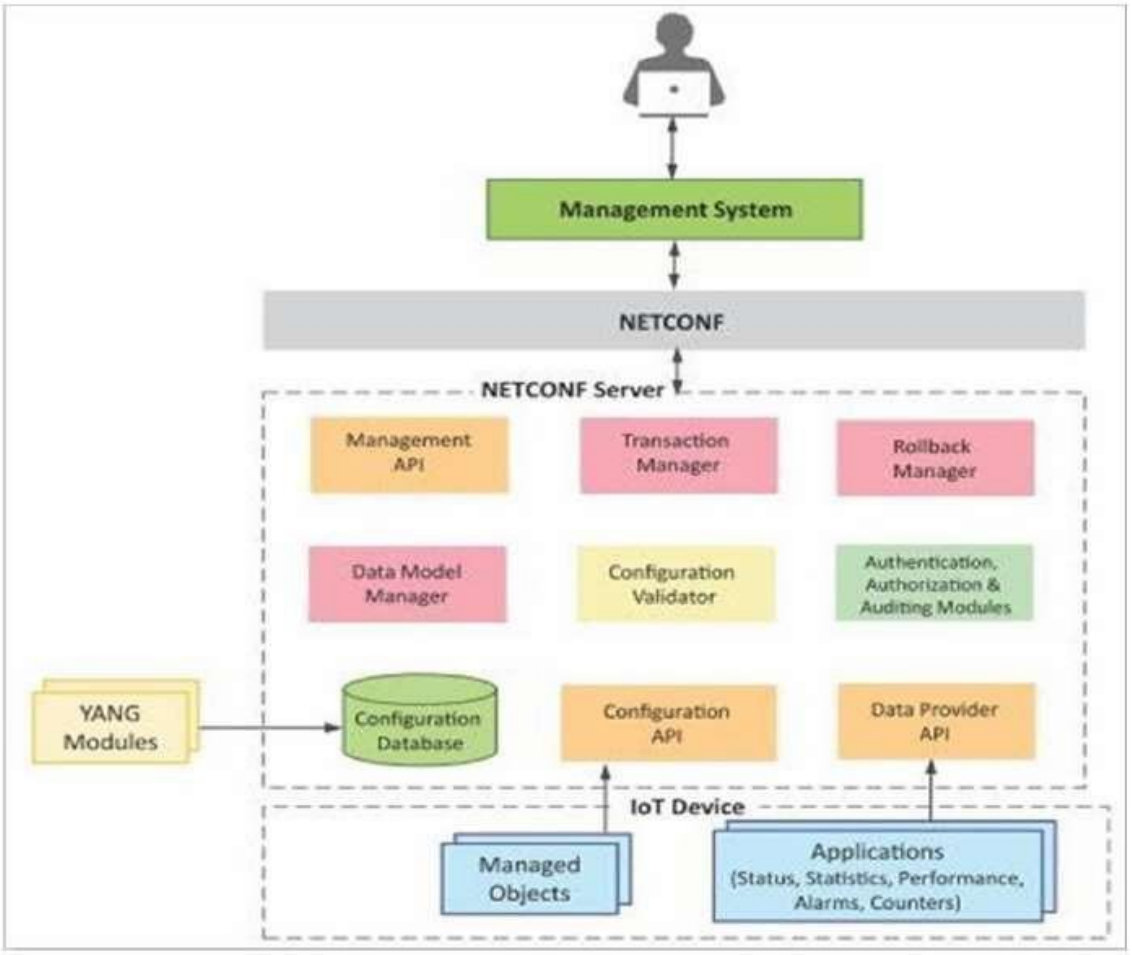
5) Data Model Manager

6) Configuration Validator

7) Configuration Database

8) Configuration API

9) Data Provider API



1) Management System: The operator uses a management system to send NETCONF messages to configure the IoT device and receives state information and notifications from the device as NETCONF messages.

2) Management API: Allows management application to start NETCONF sessions.

3) Transaction Manager: Executes all the NETCONF transactions and ensures that ACID properties hold true for the transactions.

4) Rollback Manager: This is responsible for generating all the transactions necessary to rollback a current configuration to its original state.

5) Data Model Manager: Keeps track of all the YANG data models and the corresponding managed objects. Also keeps track of the applications which provide data for each part of a data model.

6) Configuration Validator: checks if the resulting configuration after applying a transaction would be a valid configuration.

7) Configuration Database: Contains both configuration and operational data.

8) Configuration API: Using the configuration API the application on the IoT device can be read configuration data from the configuration datastore and write operational data to the operational datastore.

9) Data Provider API: Applications on the IoT device can register for callbacks for various events using the Data Provider API. Through the Data Provider API, the applications can report statistics and operational data.

Steps for IoT device Management with NETCONF-YANG:

1) Create a YANG model of the system that defines the configuration and state data of the system.

2) Complete the YANG model with the 'Inctool' which comes with Libnetconf.

3) Fill in the IoT device management code in the TransAPI module.

4) Build the callbacks C file to generate the library file.

5) Load the YANG module and the TransAPI module into the Netopeer server using Netopeer manager tool.

6) The operator can now connect from the management system to the Netopeer server using the Netopeer CLI. (Command Line Interface)

7) Operator can issue NETCONF commands from the Netopeer CLI. Command can be issued to change the configuration data, get operational data or execute an RPC on the IoT device.

NETOPEER:

Netopeer is a set of open source NETCONF tools built on the Libnetconf library. The following figure shows how to manage an IoT device using the Netopeer tools. The Netopeer include:

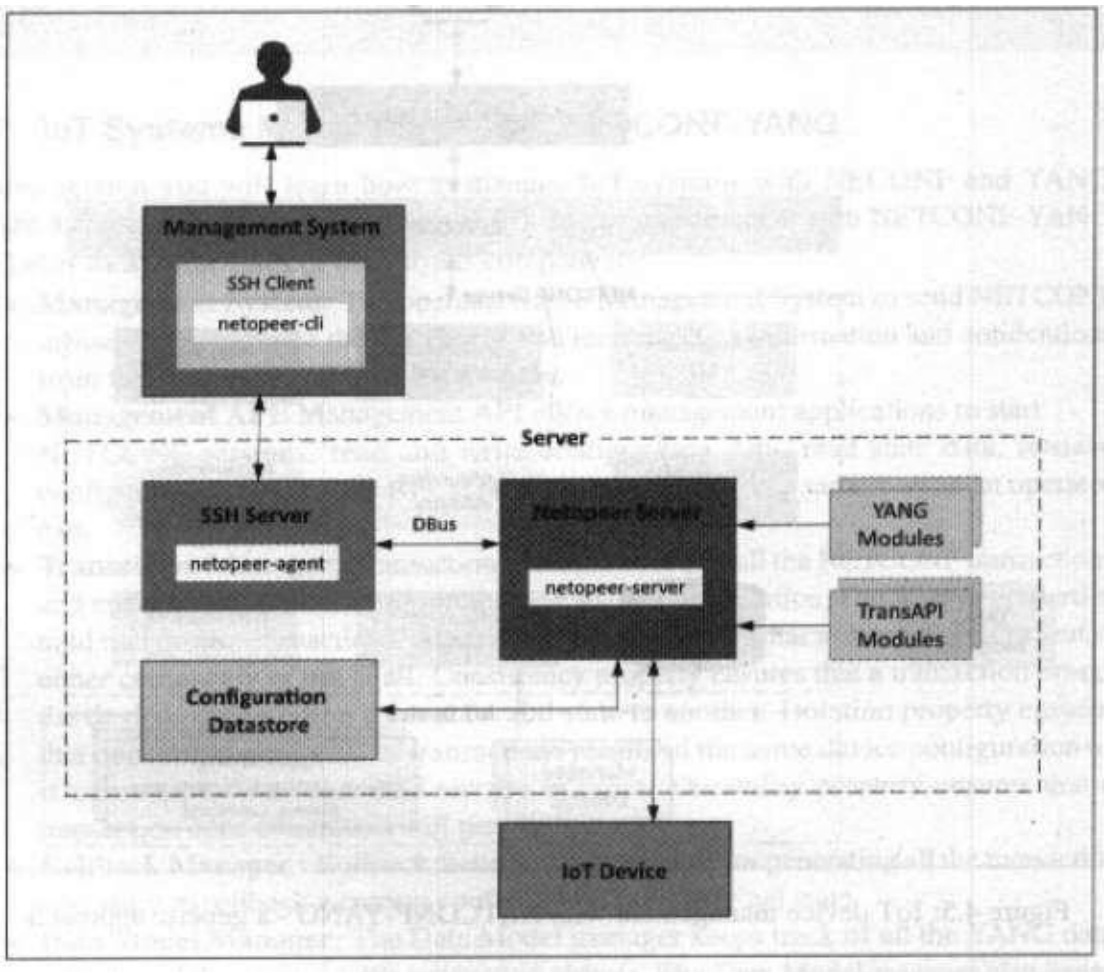
Netopeer-server: This is a NETCONF protocol server that runs on the managed device. Netopeer-server provides an environment for configuring the device using NETCONF RPC operations and also retrieving the data from the device.

Netopeer-agent: This is the NETCONF protocol agent running as a SSH/TLS subsystem. Netopeer-agent accepts incoming NETCONF connection and passes the NETCONF RPC operations received from the NETCONF client to the Netopeer-server.

Netopeer-cli: This is NETCONF client that provides a command line interface for interacting with the Netopeer-server. The operator can use the Netopeer-cli from the management system to send NETCONF RPC operations for configuring the device and retrieving the state information.

Netopeer-manager: This allows managing the YANG and Libnetconf Transaction API (TransAPI) modules on the Netopeer-server. With Netopeer-manager modules can be loaded or removed from the server.

Netopeer-configurator: This is a tool that can be used to configure the Netopeer-server.



****