UNIT-II

IoT and M2M

Introduction-M2M-Difference between IoT and M2M-SDN and NFV for IoT-Software Defined Networking, Network Function Virtualization-IoT system Management with NETCONF-YANG. 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.

- **Provide Remote acess – IOT and M2M** provide acess to information without human intervention.

- **M2M – provides** direct communication between individual machines or devices. It is designed to communicate between devices(Machines) for a specific purpose.

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

  M2M uses non-IP based proprietary networks and IoT uses broad networks protocol based on IP.

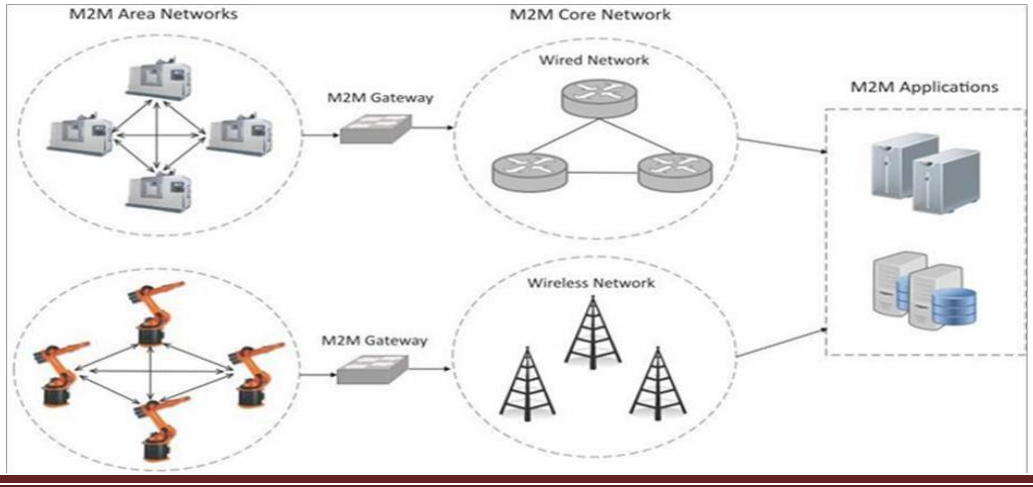
  The term which is often synonymous with IoT is Machine-to-Machine (M2M).

  IoT and M2M are often used interchangeably

- **IOT - IOT** is a broader concept for internet communication between devices. IT involves a wide range of devices, sensors, actuators and applications that communicate with the internet.

The following figure Shows the end-to-end architecture of M2M systems comprises of M2M area networks,communication networks and M2M applications.

M2M Architecture:



- 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-Busetc., 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 either proprietary or non-IP based communication protocols, the communication network uses IP-based network. Since non-IP based protocols areused within M2M area network, the M2M nodes within one network cannotcommunicate with nodes in an external network.
- To enable the communication between remote M2M are network, M2M gateways are used.

- M2M gateway acts as an intermediatory between various devices and system in a machine-to-machine setup.

  IT provides the following functionalities:

  - Connectivity between machines.
  - Data exchange between machines.
  - Compatibilities between different networks.

  The purpose or necessity of an M2M gateway

  - Within the network nodes communicate with each other.
  - To communicate with remote M2M area network,M2M gateway is required.

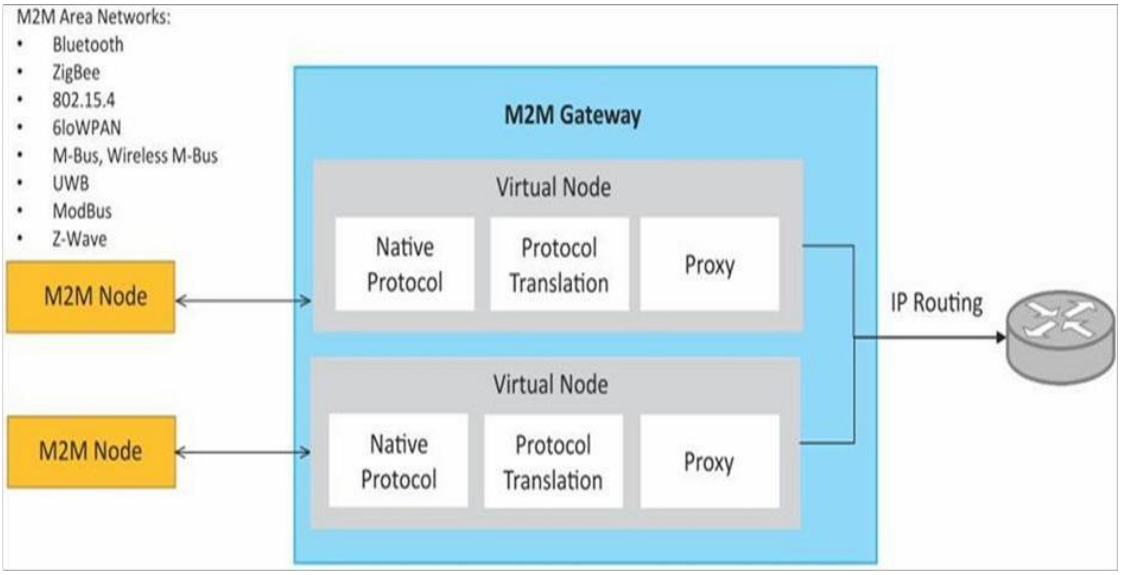


Fig. Shows a block diagram of an M2M gateway. The communication between M2M nodes and the M2M gateway is based on the communication protocols which are naive to the M2M are network. M2M gateway performs protocol translations to enable IP-connectivity for M2M are networks. M2M gateway acts as a proxy 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, ModBus, M-Bus,Wireless M-Bus etc.,
   - In IoT uses HTTP, CoAP, 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 emphasisof IoT is more on software.
4) **Data Collection &Analysis**
   - M2M data is collected in point solutions and often in on-premises storageinfrastructure.

- The data in IoT is collected in the cloud (can be public, private orhybrid 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, andon- premises 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.

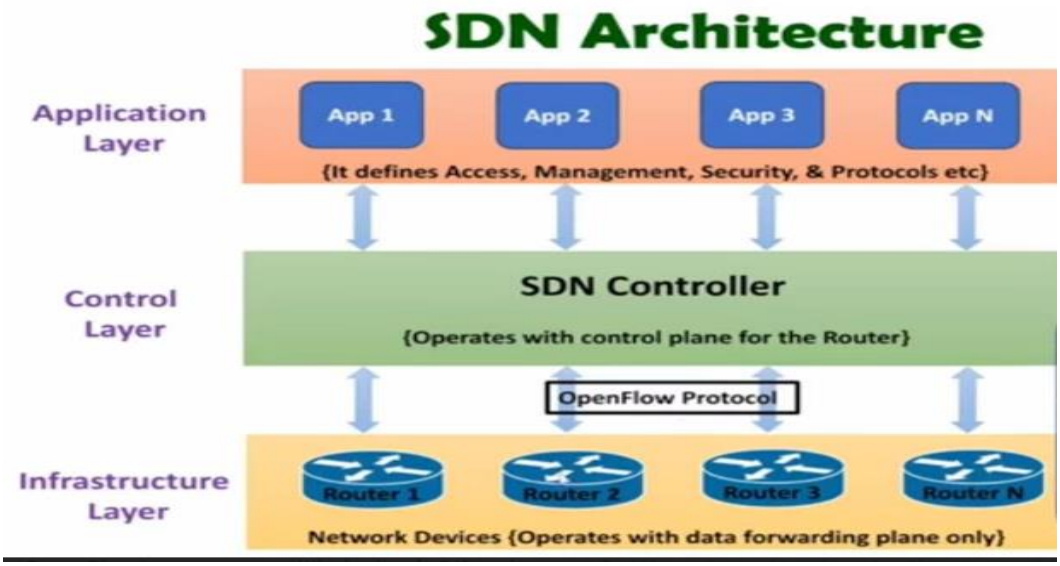
Basis of	IoT	M2M
Abbreviation	Internet of Things	Machine to Machine
Intelligence	Devices have objects that are responsible for decision making	Some degree of intelligence is observed in this
Connection type used	The connection is via Network and using various communication types.	The connection is a point to point
Communication protocol used	Internet protocols are used such as HTTP, FTP, and Telnet.	Traditional protocols and communication technology techniques are used
Data Sharing	Data is shared between other applications that are used to improve the end-user experience.	Data is shared with only the communicating parties.
Internet	Internet connection is required for communication	Devices are not dependent on the Internet.
Scope	A large number of devices yet scope is large.	Limited Scope for devices.
Business Type used	Business 2 Business(B2B) and Business 2 Consumer(B2C)	Business 2 Business (B2B)

OpenAPI support	Supports Open API integrations.	There is no support for Open Api's
Examples	Smart wearables, Big Data and Cloud, etc.	Sensors, Data and Information, etc

SDN and NFV for IoT

Software Defined Networking(SDN):

- Software-Defined Networking (SDN) is a networking architecture that separates thecontrol plane from the data plane and centralizes the network controller.
- Control plane is the part of the network that carries the signaling and routing message traffic while the data plane is the part of the network that carries payload data traffic.
- Software-based SDN controllers maintain a unified view of the network.
- SDN simplifies data communication in the network(IoT, Cloud, Computer Networks, NFV etc).
- The Router has a Control plane and a Data forwarding plane.
- Control plane does the computational task for routing and Data forwarding plane does the transfer of data packets.
- SDN simply removes the control plane task of the routers.



- Routing decisions are now taken care by the SDN controller in the SDN.
- SDN controllers maintain a unified view of the network & make configuration, management and

provisioning simpler.

The limitations of the SDN

- **Complex network devices**: More and more protocols are used to improve link speeds and reliability. Interoperability is limited due to the lack of standard and open interface. This is well suited for static traffic patterns.
- **Management Overhead:** Network managers find it increasingly **difficult to manage multiple vendors**. Upgradation of network requires configuration changes in multiple devices.
- **Limited scalability:** IoT computing environment requires **highly scalable and easy to manage network architecture**, which is becoming increasingly difficult with the conventional networks.
- SDN attempts to create **network architectures** that are simpler, inexpensive, scalable, agile and easy to manage.
- The underlying infrastructure in SDN uses simple packet forwarding hardware asopposed to specialized hardware in conventional networks.
- The following figures shows the SDN architecture and SDN Layers where the **control and data plane** are decoupled and network controller is centralized.

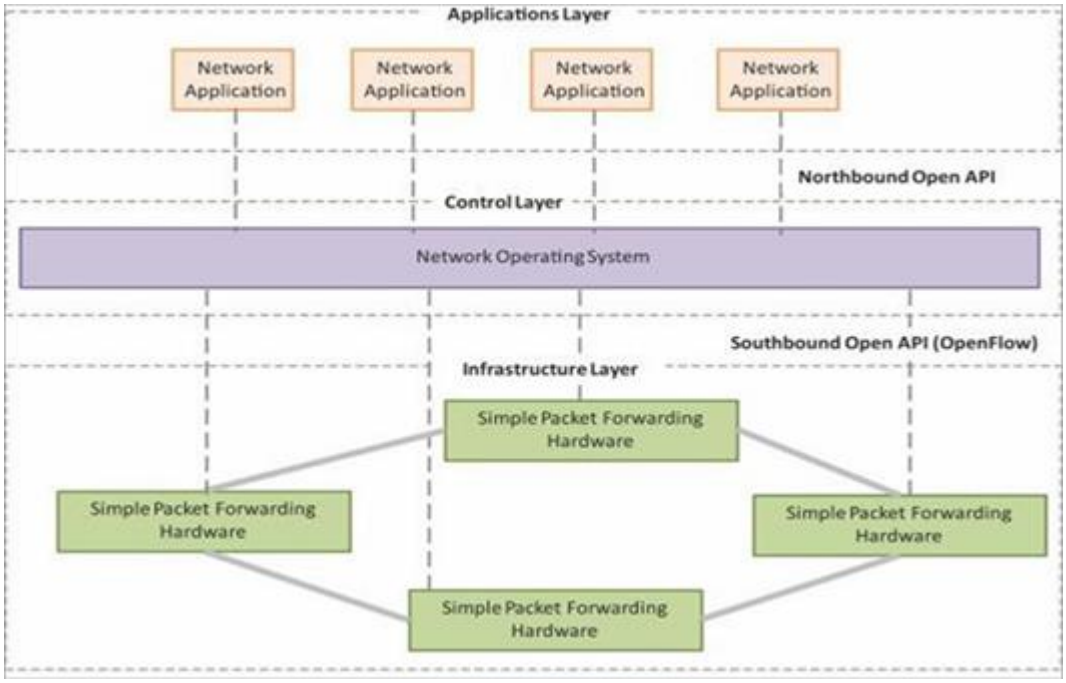


Fig. SDN Layers

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 (Open Flow)**

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

Network Function Virtualization(NFV)

- 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.
- NFV is complementary to SDN as NFV can provide the infrastructure on which SDNcan run. NFV and SDN are mutually beneficial to each other but not dependent. Network function can be virtualized without SDN, SDN can run without NFV.

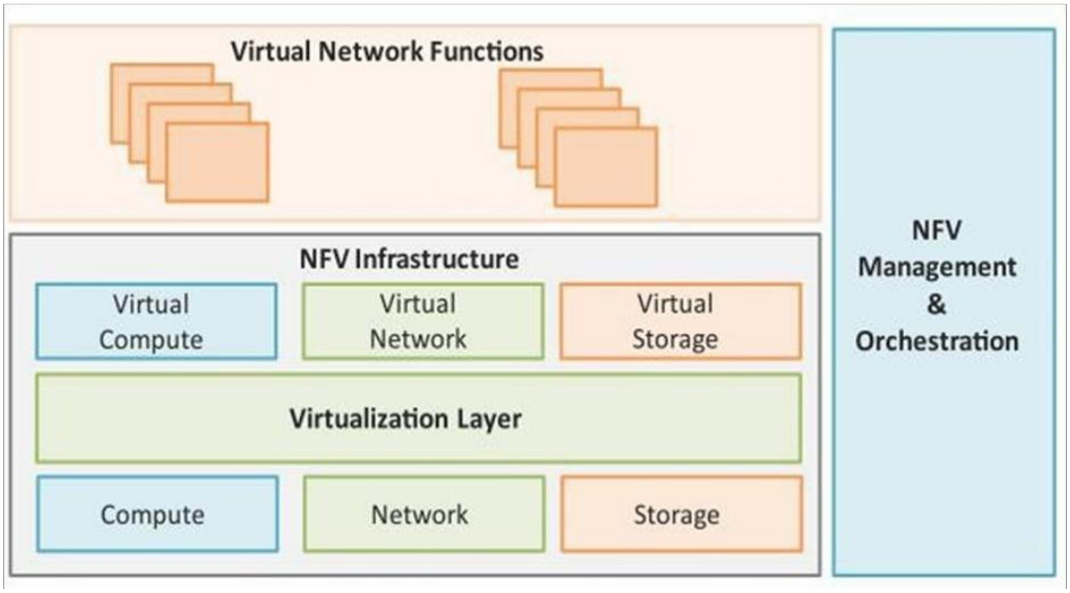


Fig. NFV Architecture

Key elements of 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).

- VNF is a software implementation of a network function.
- VNF is capable of running over the NFVI
- Examples:vFirewall,vRouter

2) **NFV Infrastructure(NFVI):**

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

- The first layer of NFVI consists of hardware resources (CPU), Storage resources(Hard disk) and network resources(Router, Switch and Firewalls).
- The Second layer of NFVI is the virtualization layer, which separates hardware and replaces it with software.
- The Third layer of NFVI is virtualized resources such as virtual computers, virtual networks and virtual storage.

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.

NFV comprises of network functions implemented in software that run on virtualized resources in the cloud. NFV enables separation of network functions from underlying hardware. Thus network functions can be easily tested and upgraded by installing new software while hardware remains the same.

- Virtualized infrastructure manager-It controls and manages network functions with NFVI resources and monitors the virtualization layer.
- VNF manager-It manages the life cycle of VNF such as initialize, update, quarry, scale, terminate etc.
- Orchestrator- It manages the life cycle of network services which includes policy, management, performance, measurement and monitoring

The following figure shows an example of how NFV can be used for virtualization of home network.

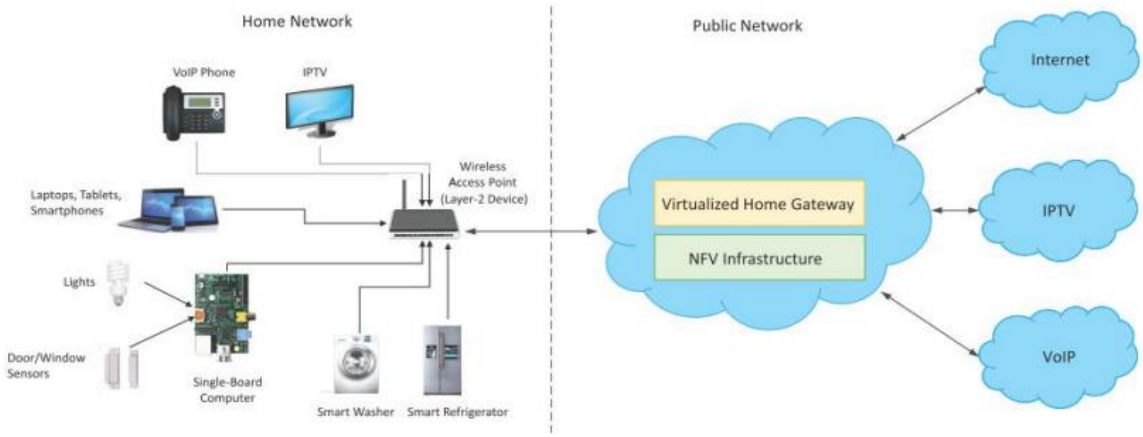


Fig: Network Architecture with virtualized home gateway

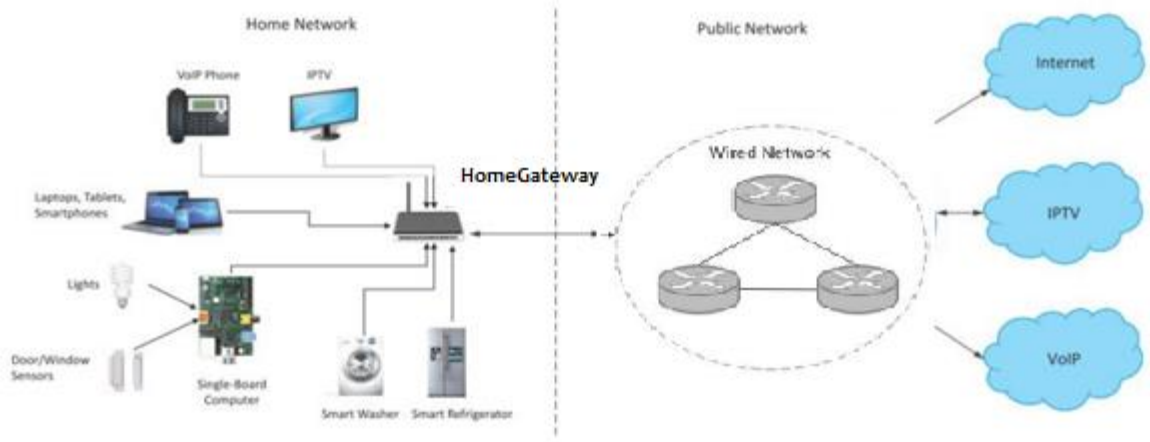


Fig: Conventional Home Network Architecture

Need for IoT Systems Management

IOT systems can have complex software, hardware and deployment designs including sensors and actuators, network resources, data collection and analysis services and user interfaces. Managing multiple devices within a single system requires advanced management capabilities. The need for managing IOT systems is described as follows.

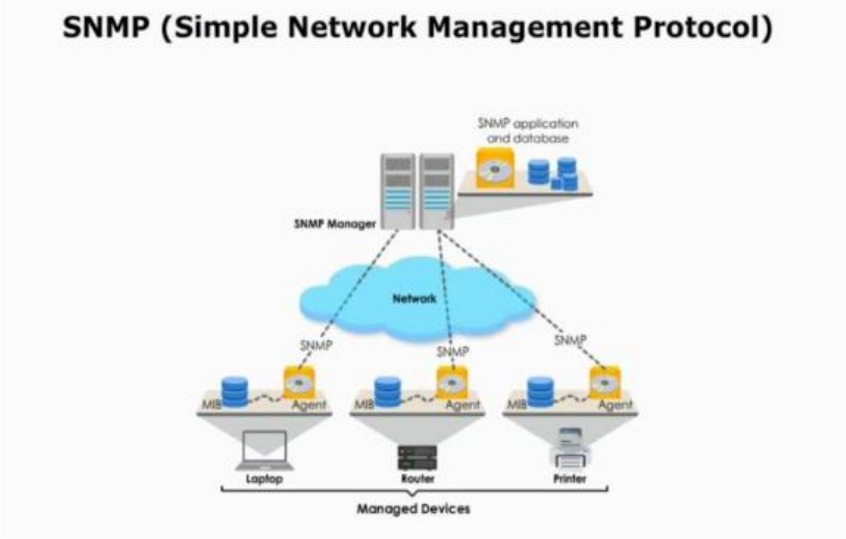
1) **Automating Configuration:** IoT system management capabilities can help inautomating the system configuration. System management interfaces provide predicate and easy-to-use management capability to automation system configuration when a system consists of multiple devices or nodes. Ensures all devices have the same configuration and variations or errors due to manual configurations are avoided.
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. Operational data is the system's operating parameters that are collected by the system at runtime. Statistical data is the system performance (e.g. CPU and memory usage) data for fault diagnosis or prognosis (forecasting).
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 consist of multiple devices or nodes, ensuring system wide configuration can be critical for the correct functioning of the system. Each device is configured separately (either manual or automated) can result in system faults or undesirable outcomes.

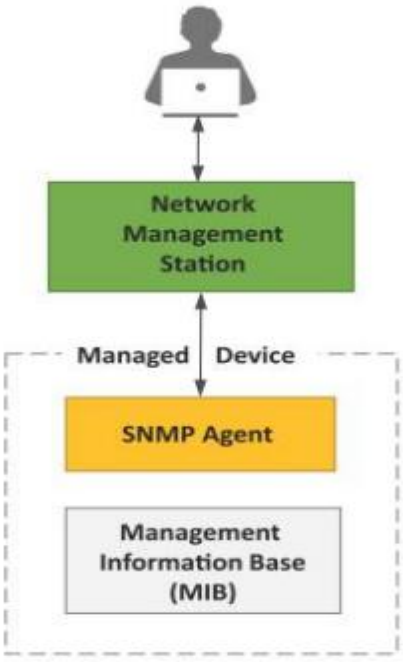
This happens when some devices are running on old configurations and some on new. To avoid this system wide configurations are required. Ensures that the configuration changes are either applied to all devices or to none. In the failure, the configuration changes are rolled back.

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 capabilityof retrieving configurations from devices can help **in reusing the configurations** for other devices of the same type. Ensure that when a new device is added, the same configuration is applied. The management system can retrieve the current configuration from a device and apply the same to the new devices.

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., SNMP is an application layer protocol that uses UDP port number 161/162. SNMP is used to monitor the network, detect network faults, and sometimes even used to configure remote devices. The following figure shows the components involved in managing a device with SNMP.





SNMP components –

There are 3 components of SNMP:

1. **SNMP Manager –**

   It is a centralized system used to monitor network. It is also known as Network Management Station (NMS), sends and receives SNMP messages to manage and monitor devices on an IP network.

   - SNMP database and application
   - Used for network configuration, security, performance and troubleshooting

2. **SNMP – managed devices:** workstations, IP phones, printers and routers

SNMP agent-

- An SNMP agent is Software running on network devices (e.g, routers, switches, servers, printers) that collects data about the device's performance, configuration and status.
- The agent maintains the MIB, a database that contains information about the devices configuration, performance statistics and other operational data.
- The agent listens for requests from the SNMP manager, processes those requests, and sends response back to the manager.

(MIB) Management Information Base –

- The MIB is a hierarchical database or structure that contains all the manageable information on the device. It defines the types of data available from the managed device (e.g., CPU utilization, memory usage, interface statistics).
- The MIB is organized into a tree-like structure, with each object in the MIB represented by a unique Object identifier(OID). Each OID corresponds to a specific piece of information or control functionality.

OID (object Identifier):

- An OID is a globally unique identifier used to access specific information in the MIB. It is represented as a series of numbers separated by dots(e.g., 1.3.6.1.2.1.2.2.1.1)
- The OID allows the SNMP manager to request specific pieces of information from the SNMP agent, such as interface status, system uptime or interface throughout.

5. **SNMP protocols operations:**

SNMP defines several operations (messages) that allow communication between the manager and the agent. These operations include:

- **GET :** The SNMP manager uses the GET request to retrieve information from the agent. The manager specifies the OIDs it wants to query and the agent returns the corresponding values.
- **SET :** The SET request is used by the manager to configure or change the values of specific parameters on the managed device. For example, an SNMP SET request can modify the device's configuration or enable/disable interfaces.
- **GETNEXT :** The GETNEXT operation is used to fetch the next available object in the MIB hierarchy. It is often used in walks of the MIB to traverse through the hierarchical data.
- **GETBULK** : This operation retrieves a large block of data in one request(used mainly in SNPv2c and SNMPv3).It is useful for fetching multiple data points from the MIB efficiently.
- **TRAP :** A TRAP is an asynchronous message sent by the agent to the manager, notifying it of certain events or status changes. For example, if an interface goes down, the agent can send an SNMP TRAP to inform the manager.
- **INFORM :** Similar to TRAP, but with a response mechanism. The manager must acknowledge the reception of the INFORM message.

- **REPORT:** Used for handling exceptions or errors in the SNMPv3 protocol, typically in response to a failure in communication or authentication.

Versions of SNMP

- SNMPv1

  It uses Community string and it is send in Clear-text
- SNMPv2c

  It also uses Community string and it is send in Clear-text It has additional GetBulk and GetNext messages.
- SNMPv1

  It has more security mechanism as compared to V1 and V2c

  Three different security modes:
  - noAutoNopriv: username authentication but no encryption
  - authNoPriv: MD5 or SHA authentication but no encryption
  - authPriv: MD5 or SHA authentication and encryption

SNMP Strength

1. It is simple to implement.
2. Agents are widely implemented.
3. Agent level overhead is minimal.
4. It is robust and extensible.
5. Polling approach is good for LAN based managed object.
6. It offers the best direct manager agent interface.
7. SNMP meet a critical need.

Limitations of SNMP Management

- **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.

- SNMP is a connectionless protocol which uses UDP as the transport protocol, making it unreliable as there was no support for acknowledgement of requests.
- MIBs often lack writable objects without which device configuration is not possible using SNMP. With the absence of writable objects, SNMP can be used only for device monitoring and status polling.
- It is difficult to differentiate between configuration and state data in MIBs.
- Retrieving the current configuration from a device can be difficult with SNMP.
- Earlier versions of SNMP did not have strong security features making the management information vulnerable to network intruders.
- Though security features were added in the later versions of SNMP, it increased the complexity a lot.

Network Operator requirement

To address the limitations of the existing network management protocols and plan the future work, the Network Architecture Board(NAB) and Internet Engineering Task Force(IETF) held a workshop on network management in 2002 with all the network operators and protocol developers based on that an operational requirements was prepared . They are listed in the following points.

- **Ease of use:** From the operator point of view, ease of use is the primary requirement for any Network management technology.
- **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 current state. State data is an operational data which is collected by the system at runtime and Statistical data which describes the system performance. For effective management solution, it is important to differentiate between configuration and state data.
- **Fetch configuration and state data separately:** It should be possible to fetch the configuration data and state data separately from the managed devices. This is useful when different devices need to be compared.
- **Configuration of the network as a whole:** This is important for systems which have multiple devices and configure them within one network wide transaction is required.

- **Configuration transactions across devices:** System must support of configuration transactions across multiple devices.
- **Configuration deltas:** It should be possible to generate the operations necessary for going from one configuration state to another
- **Dump and restore configurations:** It should be possible to dump the configurations from devices and restore configurations to devices.
- **Configuration validation:** It should be possible validate configurations.
- **Configuration database schemas:** There is a need for standardized configurations data base schemas or data models across operators.
- **Comparing configurations:** It should be possible to compare configurations.
- **Role-based access control**: Devices should support role based access, so that a user is given the minimum access necessary to perform a required task.
- **Consistency of access control lists:** Consistency checks of access control lists across devices must be possible.
- **Multiple configuration sets:** should support for multiple configuration sets on devices.
- **Support for both data-oriented and task-oriented access control:** SNMP access control is data oriented. CLI access control is task oriented. It should support both access controls.

NETCONF (NETwork CONFiguration):

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.-NETCONF" stands for "Network Configuration Protocol," which is a standardized network management protocol used to remotely configure and manage network devices by sending configuration data in an XML-based format, allowing for automated configuration changes across a

network through a client-server interaction model; it is typically used with the YANG data modeling language to define the structure of configuration data.

Key points about NETCONF:

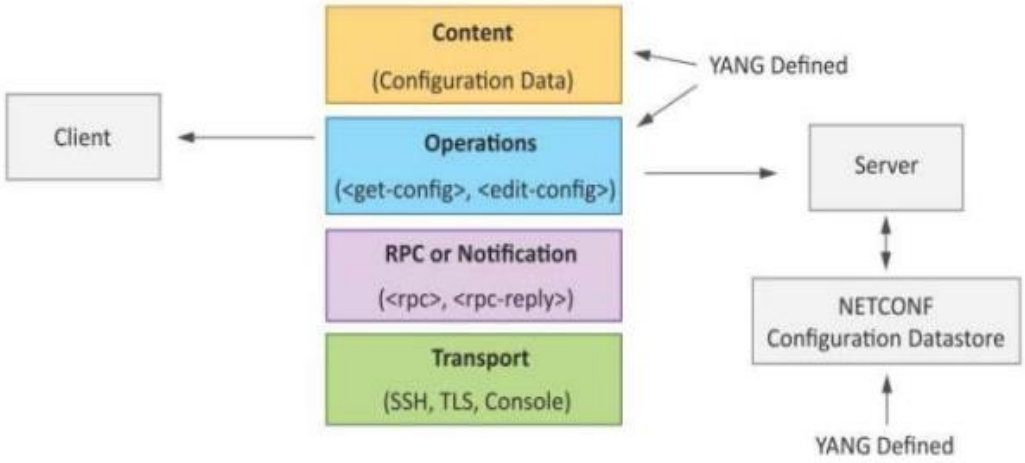
Function: Enables the retrieval, modification, and deletion of network device configurations.

Data format: Uses Extensible Markup Language (XML) to encode configuration data and protocol messages.

Communication model: Based on Remote Procedure Calls (RPC) for client-server interaction.

YANG integration: Leverages the YANG data modeling language to define the structure of configuration data on devices.

Transport protocol: Often runs over Secure Shell (SSH) for secure communication.



- NETCONF works on SSH transport protocol.
- Transport layer provides end-to-end connectivity and ensure reliable delivery of messages.
- NETCONF uses XML-encoded Remote Procedure Calls (RPCs) for framing request and response messages.
- The RPC layer provides mechanism for encoding of RPC calls and notifications.

- NETCONF provides various operations to retrieve and edit configuration data from network devices.
- The Content Layer consists of configuration and state data which is XML-encoded.
- The schema of the configuration and state data is defined in a data modeling language called YANG.
- NETCONF provides a clear separation of the configuration and state data.
- The configuration data resides within a NETCONF configuration data store on the server.

Operation	Description
connect	Connect to a NETCONF server
get	Retrieve the running configuration and state information
get-config	Retrieve all or a portion of a configuration datastore
edit-config	Loads all or part of a specified configuration to the specified target configuration
copy-config	Create or replace an entire target configuration datastore with a complete source configuration
delete-config	Delete the contents of a configuration datastore
lock	Lock a configuration datastore for exclusive edits by a client
unlock	Release the lock on a configuration datastore
get-schema	This operation is used to retrieve a schema from the NETCONF server
commit	Commit the candidate configuration as the device's new current configuration
close-session	Gracefully terminate a NETCONF session
kill-session	Forcefully terminate a NETCONF session

Table 4.1: List of commonly used NETCONF RPC methods

YANG (Yet Another Next Generation) :

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

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

Node Type	Description
Leaf Nodes	Contains simple data structures such as an integer or a string. Leaf has exactly one value of a particular type and no child nodes.
Leaf-List Nodes	Is a sequence of leaf nodes with exactly one value of a particular type per leaf.
Container Nodes	Used to group related nodes in a subtree. A container has only child nodes and no value. A container may contain any number of child nodes of any type (including leafs, lists, containers, and leaf-lists).
List Nodes	Defines a sequence of list entries. Each entry is like a structure or a record instance, and is uniquely identified by the values of its key leafs. A list can define multiple key leafs and may contain any number of child nodes of any type.

Table 4.2: YANG Node Types

YANG Module Example: A network enabled Toaster

- This YANG module is a YANG version of the toaster MIB.
- The toaster YANG module begins with the header information followed by identity declarations which define various bread types.
- The leaf nodes (‘toaster Manufacturer’, ‘toaster Model Number’ and ‘toaster Status’) are defined in the ‘toaster’ container.
- Each leaf node definition has a type and optionally a description and default value.

- The module has two RPC definitions (‘make-toast’ and ‘cancel-toast’).

Visual representation of Toaster YANG module is given bellow.

IoT Systems Management with NETCONF-YANG

YANG is a data modeling language used to model configuration and state data manipulatedby the NETCONF protocol.

Roles ofvarious 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

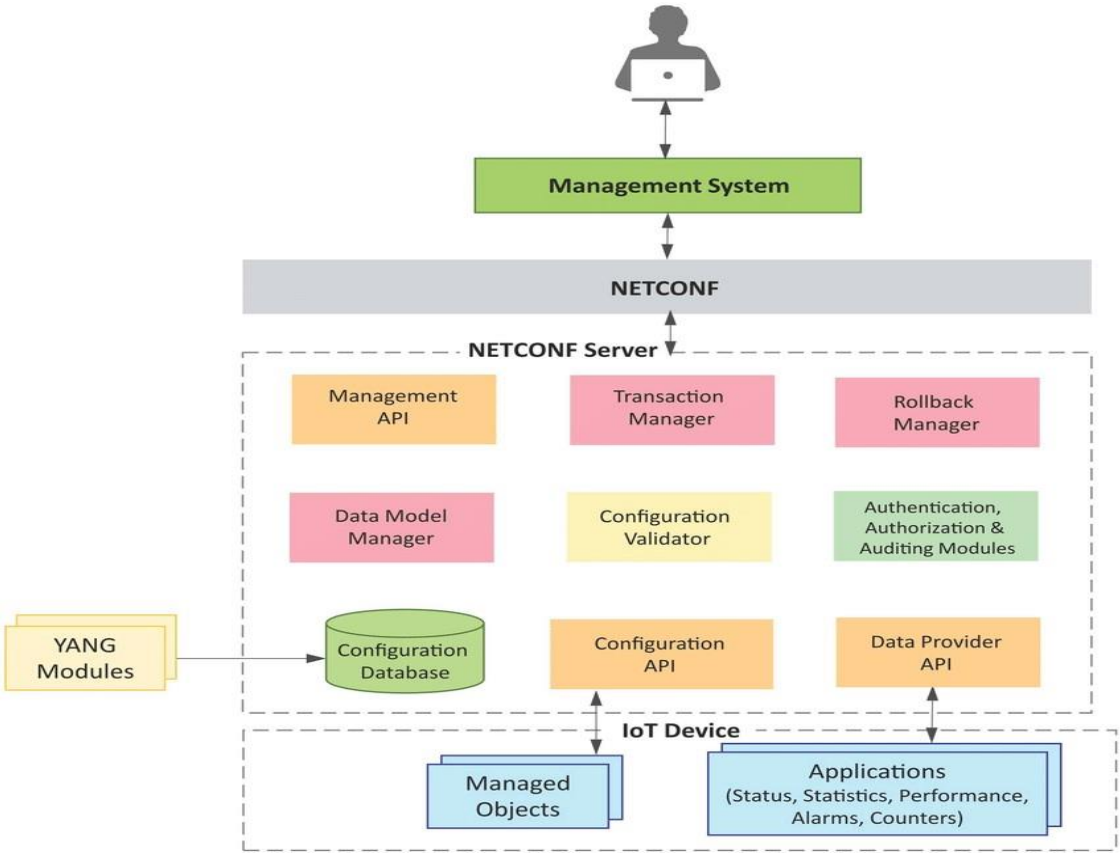


Fig. Generic approach of IoT device management with NETCONF-YANG.

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 :** It allows management application to start NETCONF sessions, read and write configuration data, read state data,retrieve configurations and invoke RPCs, programmatically in the same way as on operator.
3) **Transaction Manager:** It executes all the NETCONF transactions and ensures that ACID(Atomicity, Consistency, Isolation and Durability) properties hold true for the transactions.

	Description
Atomicity	Ensure that the transaction is executed either completely or not at all

Consistency	Ensure that the transaction brings the device configuration from one valid state to another
Isolation	Ensure concurrent execution of transaction results in same device if transaction executed serially
Durability	Ensure that once a transaction is committed will persist

4) **Rollback Manager:** Responsible for generating all the transactions necessary torollback a current configuration to its original state.
5) **Data Model Manager :** Keeps track of all the YANG data models and the correspondingmanaged objects. Also keeps track of the applications which provide data for each part ofa data model.
6) **Configuration Validator:** It checks if the resulting configuration after applying a transaction would be a valid configuration.
7) **Configuration Database :** The 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 data store and write operational data to the operational data store.
9) **Data Provider API:** Applications on the IoT device can register for callbacks for variousevents using the Data Provider API. Through the Data Provider API, the applications can report statistics and operational data.

NETOPEER

Netopeer is set of open source NETCONF tools built on the **Libnetconf** library

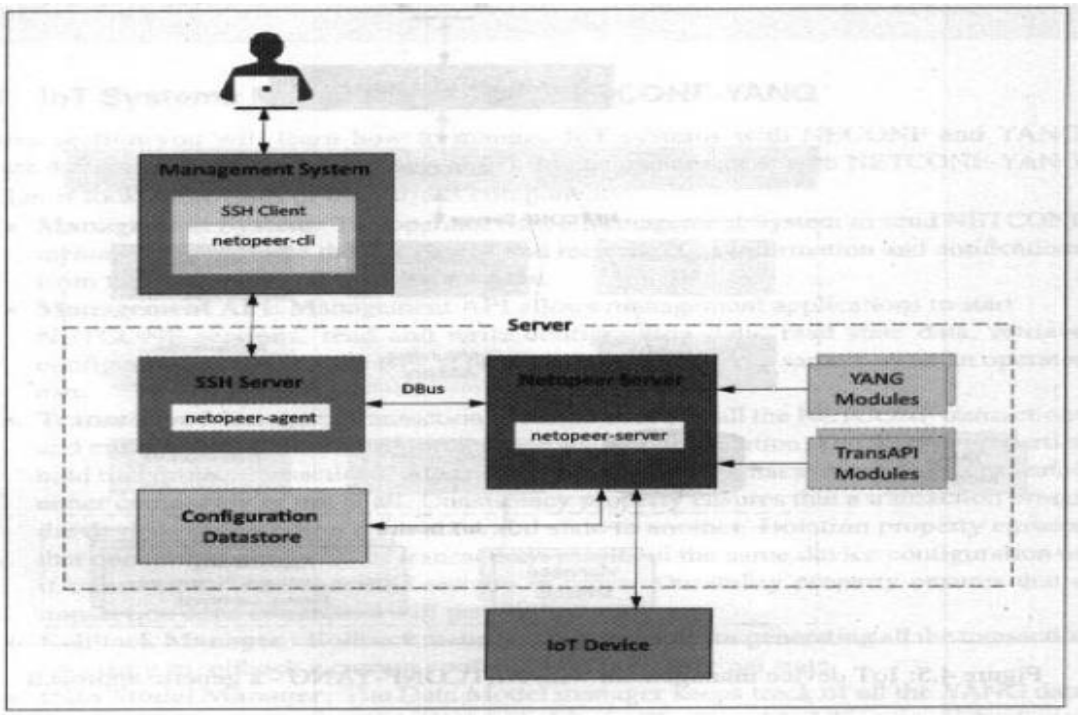


Fig. IOT device management with NETCONF- a specific approach based on Netopeer tools

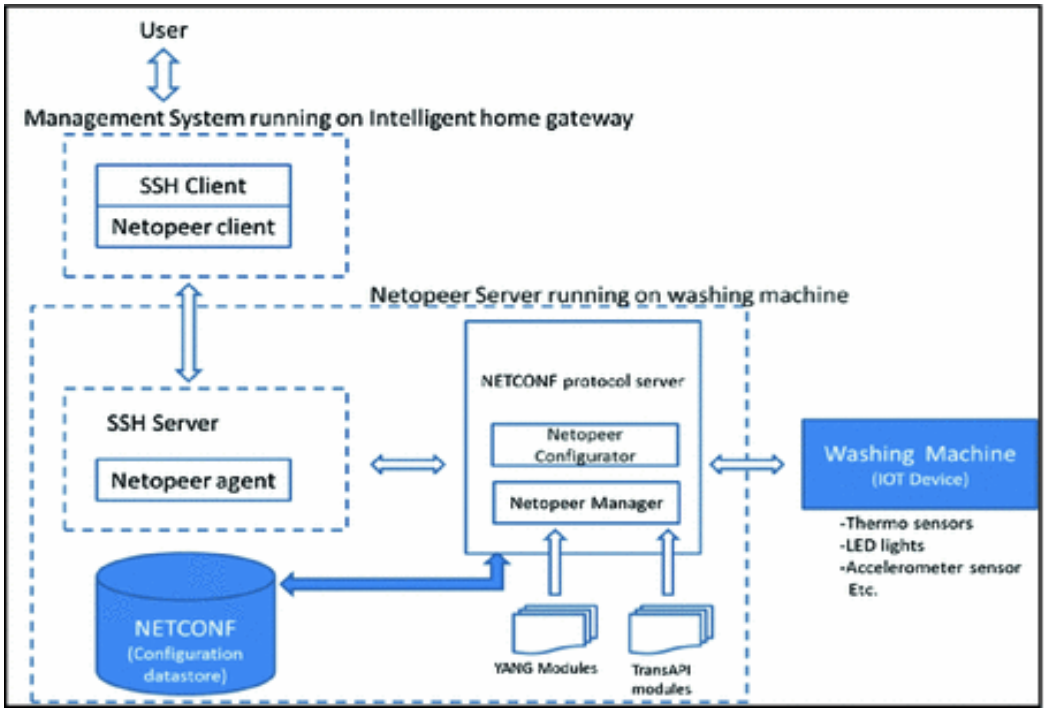


Fig. Example Washing machine management with Netopeer

- **Netopeer-server**: is a NETCONF protocol server that runs on the managed device, this server provides an environment for configuring the device using NETCONF RPC operations and also retrieving the state data from the device.
- **Netopeer-agent**: is a NETCONF protocol agent running as a SSH/TLS subsystem. This agent accepts incoming NETCONF connection and passes the NETCONF operations received from the NETCONF-client to the NETCONF-server.
- **Netopeer-cli**: is a NETCONF client that provides the command line interface for interacting with the Netopeer-server. The operator uses Netopeer-cli to from the management system to send NETCONF RPC operations for configuring the device and retrieving the state information.
- **Netopeer-manager**: allows managing the YANG and Libnetconf Transaction API modules on the Netopeer-server.
- **Netopeer- configurator**: is a tool that can be used to configure the Netopeer-server.

Steps for IoT device Management with NETCONF-YANG

1) Create a YANG model of the system that defines the configuration and state data of thesystem.
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 NetopeerCLI.
7) Operator can issue NETCONF commands from the NetopeerCLI. Command can be issued to change the configuration data, get operational data or execute an RPC on the IoT device.