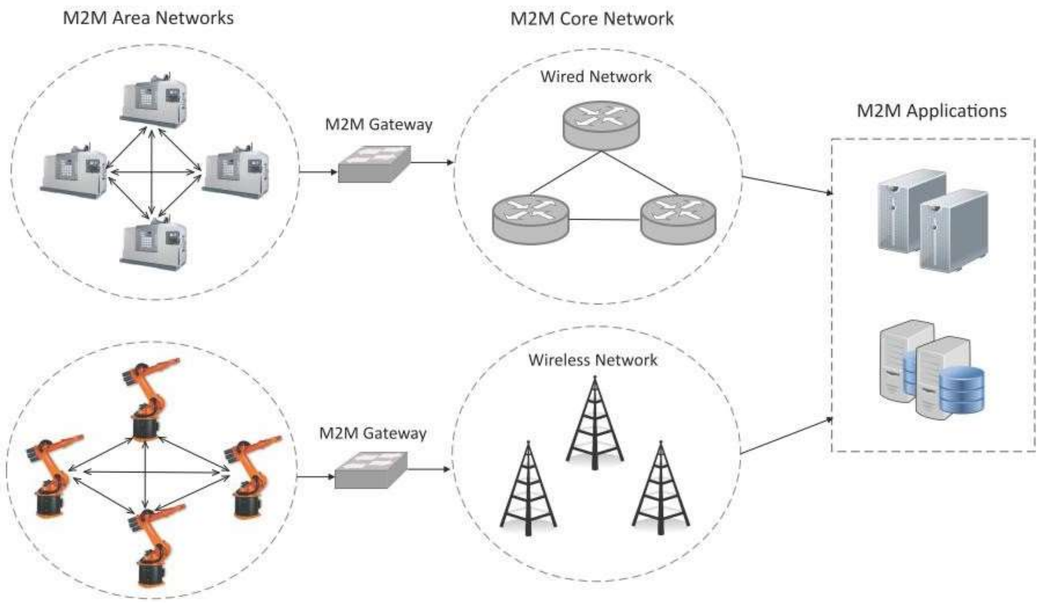
M2M

Machine-to-Machine (M2M)

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



M2M system

Each machine in embeds a smart device

- Device senses the data or status of the machine
- Performs the computation and communication functions
- A device communicates via wired or wireless systems

Machine-to-Machine (M2M) to IoT

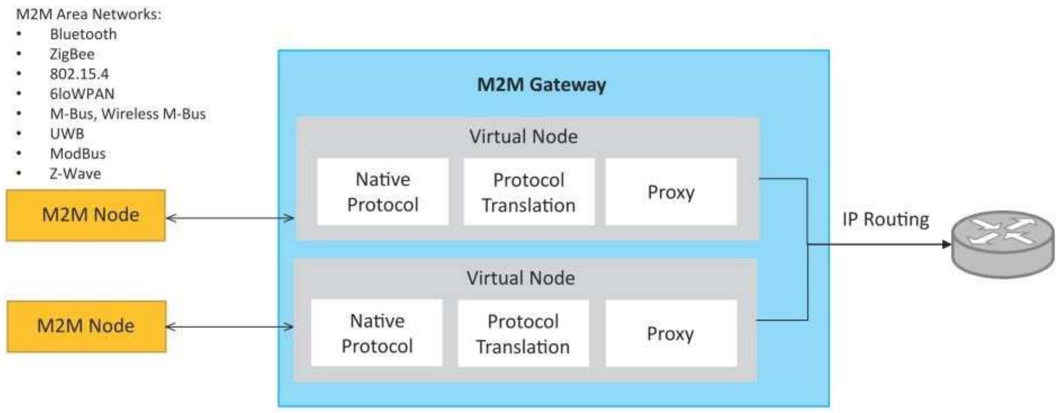
- Technology closely relates to IoT which use smart devices to collect data that is transmitted via the Internet to other devices.
- Close differences lies in M2M uses for device to device communication also for coordinated monitoring and control purposes

M2M Application Areas

- Connected Cars for Safety
- Remote Monitoring
- ATMs/Point of Sales Terminal Connected for centralized Security

M2M gateway

- Since non-IP based protocols are used within M2M area networks, the M2M nodes within one network cannot communicate with nodes in an external network.
- To enable the communication between remote M2M area networks, M2M gateways are used.



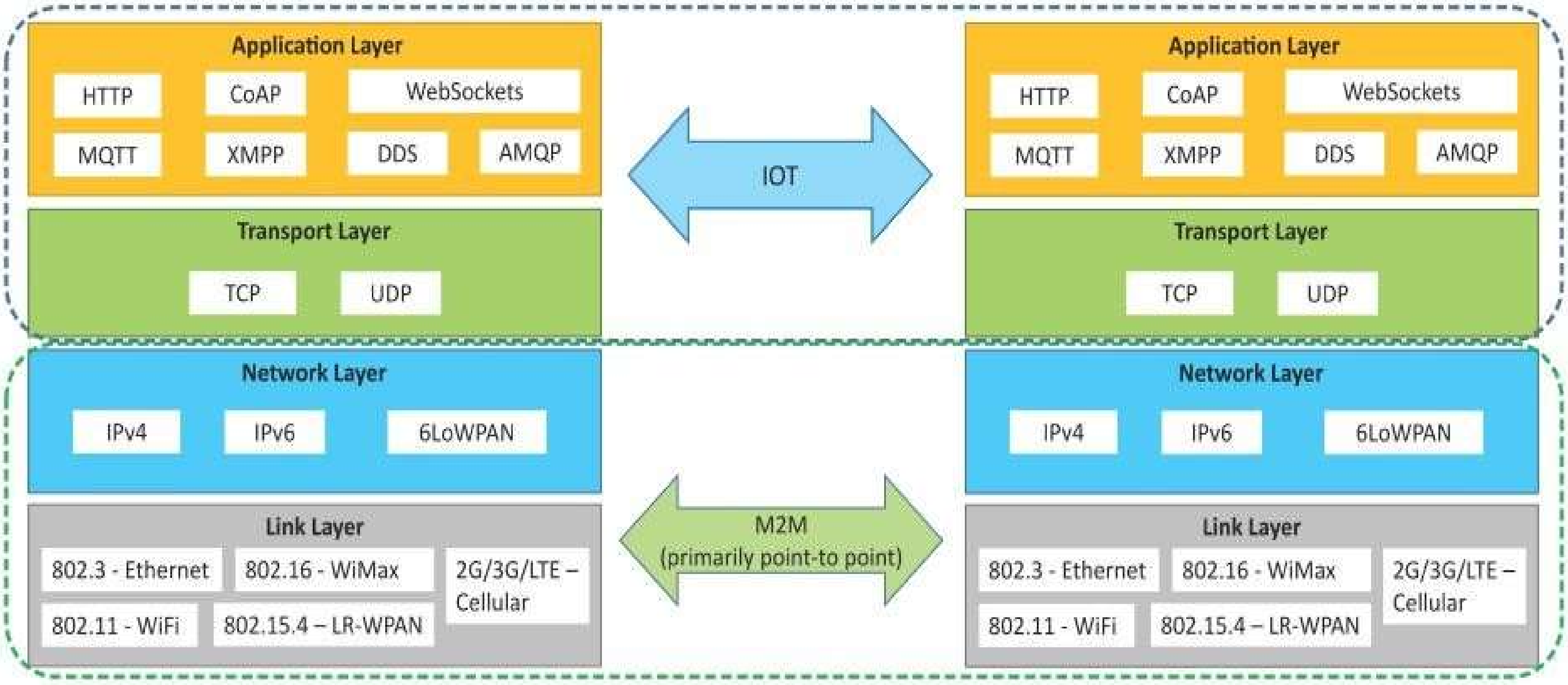
Difference between IoT and M2M

- Communication Protocols
  - M2M and IoT can differ in how the communication between the machines or devices happens.
  - M2M uses either proprietary or non-IP based communication protocols for communication within the M2M area networks.
- Machines in M2M vs Things in IoT
  - The "Things" in IoT refers to physical objects that have unique identifiers and can sense and communicate with their external environment (and user applications) or their internal physical states.
  - M2M systems, in contrast to IoT, typically have homogeneous machine types within an M2M area network.

Difference between IoT and M2M

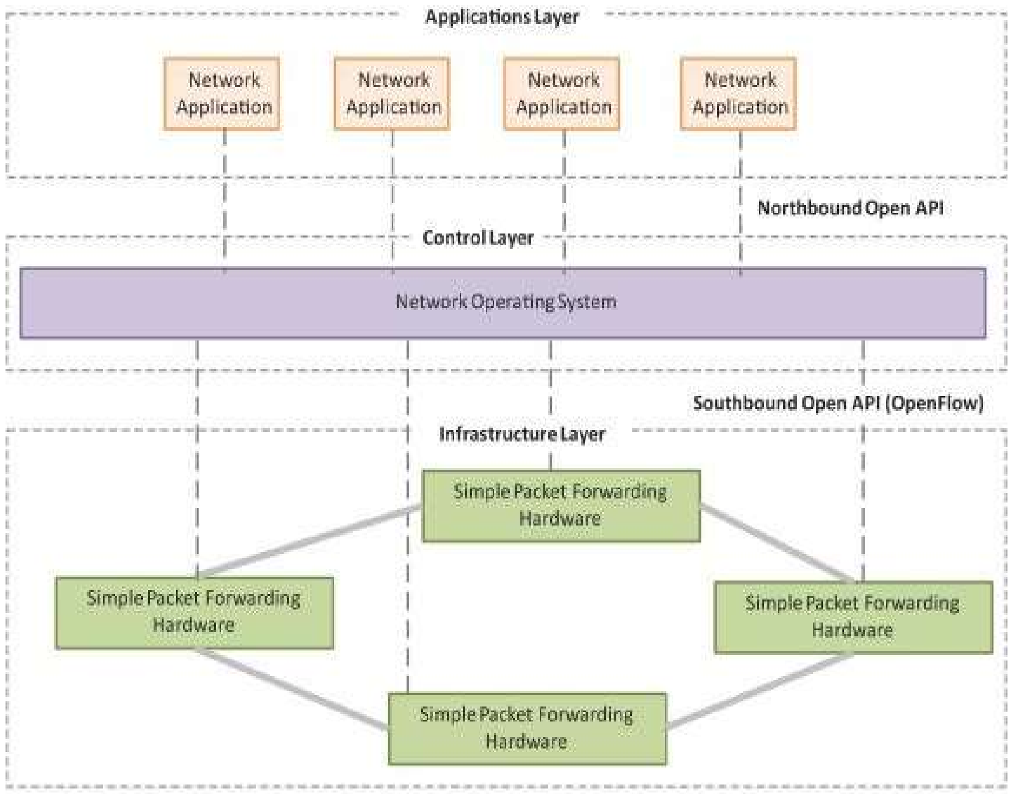
- Hardware vs Software Emphasis
  - While the emphasis of M2M is more on hardware with embedded modules, the emphasis of IoT is more on software.
- Data Collection & Analysis
  - M2M data is collected in point solutions and often in on-premises storage infrastructure.
  - In contrast to M2M, the data in IoT is collected in the cloud (can be public, private or hybrid cloud).
- 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.

Communication in IoT vs M2M



SDN

- Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and centralizes the network controller.
- Software-based SDN controllers maintain a unified view of the network and make confi guration, management and provisioning simpler.
- The underlying infrastructure in SDN uses simple packet forwarding hardware as opposed to specialized hardware in conventional networks.

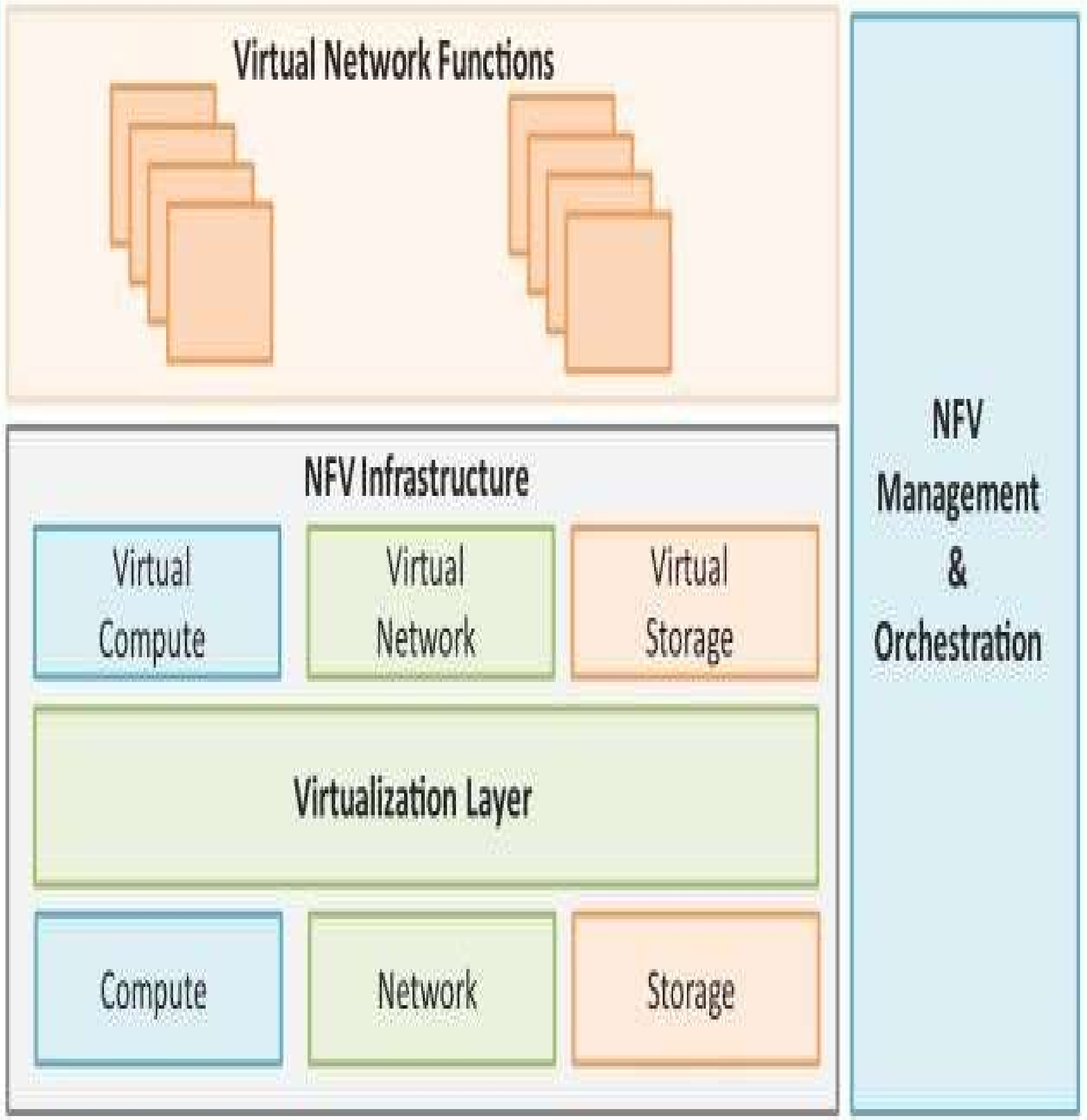


Key elements of SDN

- Centralized Network Controller
  - With decoupled control and data planes and centralized network controller, the network administrators can rapidly configure the network.
- Programmable Open APIs
  - SDN architecture supports programmable open APIs for interface between the SDN application and control layers
- 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.

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 SDN can run.

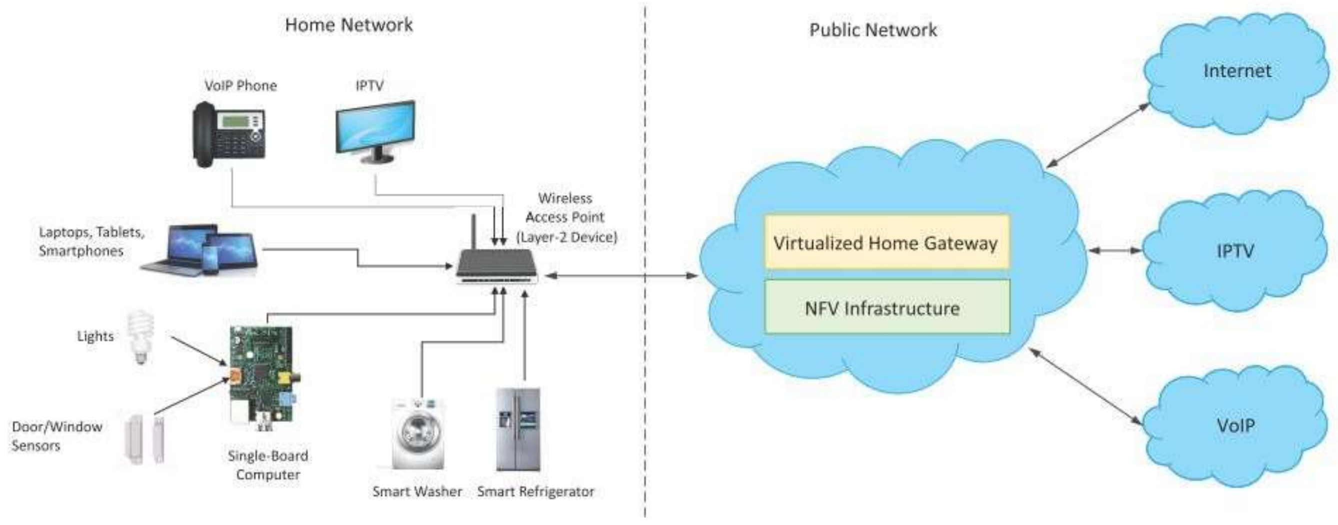


Key elements of NFV

- Virtualized Network Function (VNF):
  - VNF is a software implementation of a network function which is capable of running over the NFV Infrastructure (NFVI).
- NFV Infrastructure (NFVI):
  - NFVI includes compute, network and storage resources that are virtualized.
- 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 Use Case

- NFV can be used to virtualize the Home Gateway. The NFV infrastructure in the cloud hosts a virtualized Home Gateway. The virtualized gateway provides private IP addresses to the devices in the home. The virtualized gateway also connects to network services such as VoIP and IPTV.



Simple Network Management Protocol (SNMP)

What is it?

A Protocol that Facilitates the exchange of management information between network devices.

Why was it developed?

To **control and monitor status** of network devices

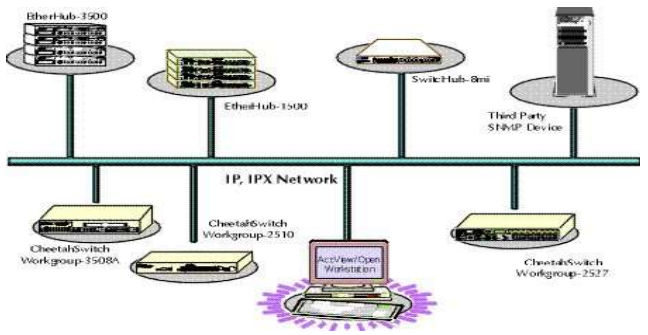
How is it beneficial?

Enables network administrators to:

Manage network performance

Find and solve network problems

Plan for network growth



SNMP Basic Components

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

Network Management station

- Collects and stores management information, and makes this information available to NMS using
- Could be a work station or PC

Network Management System (NMS)

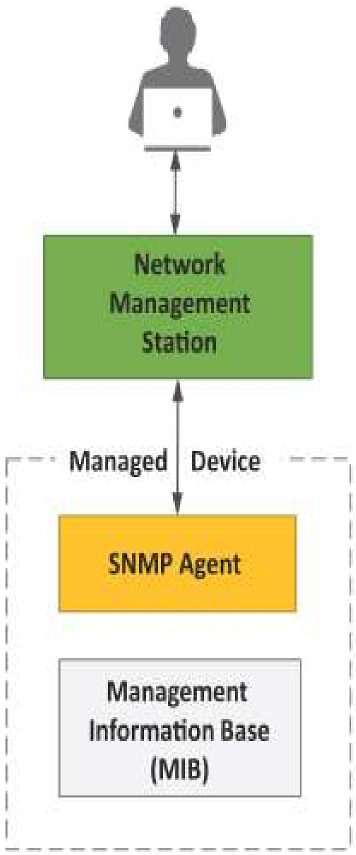
- Executes applications that monitor and control managed devices

Agent

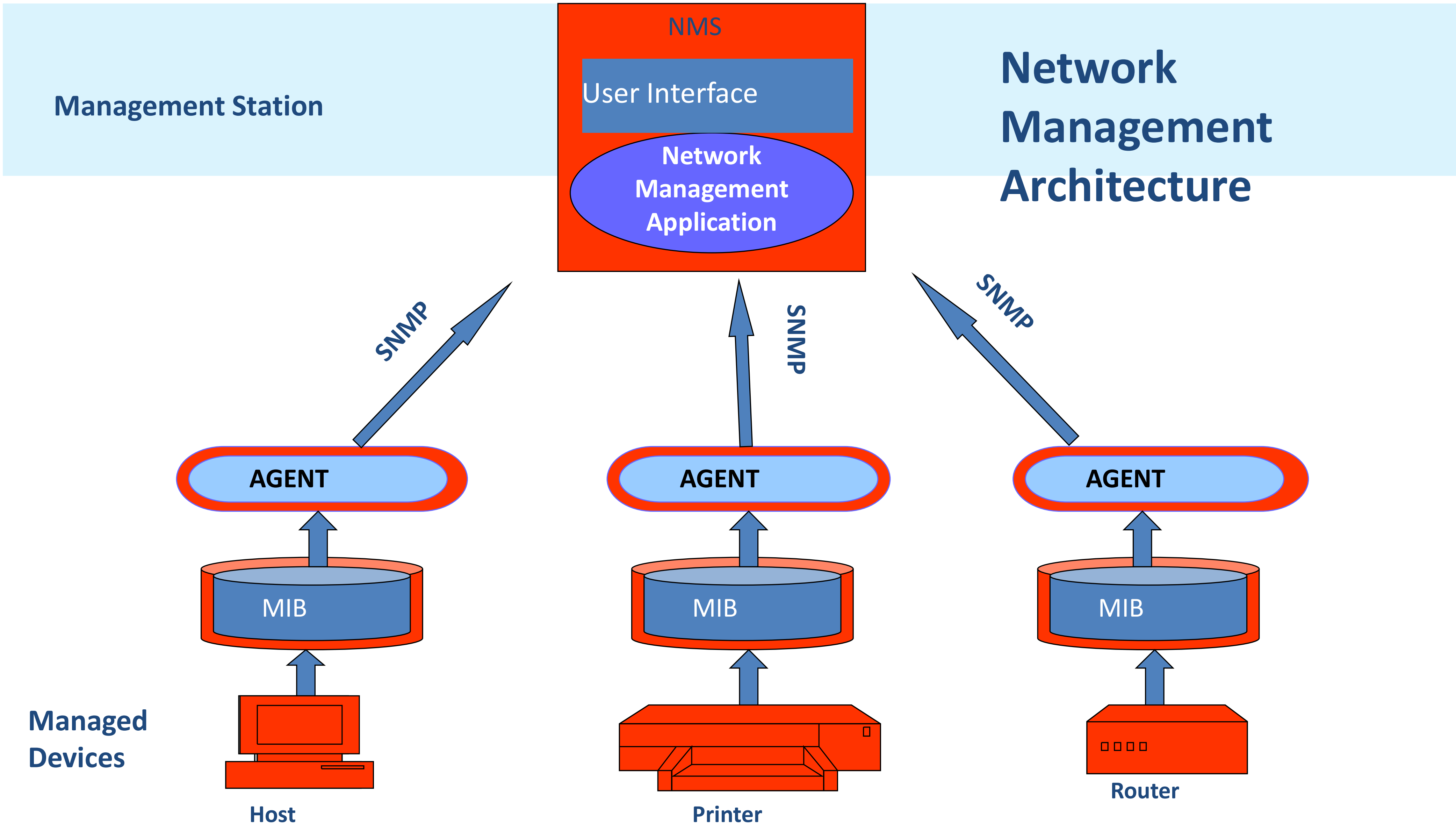
- A network-management software module that resides in a managed device

Management Information Base (MIB)

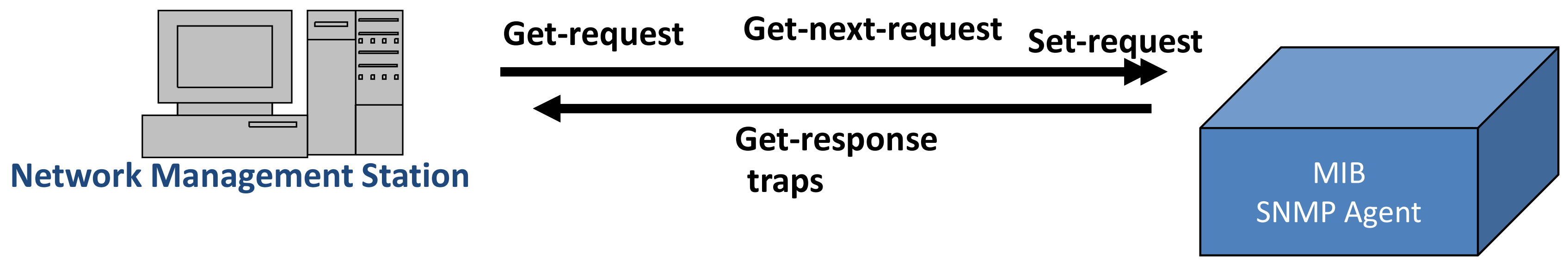
- Used by both the manager and the agent to store and exchange management information



Network Management Architecture



How SNMP Works



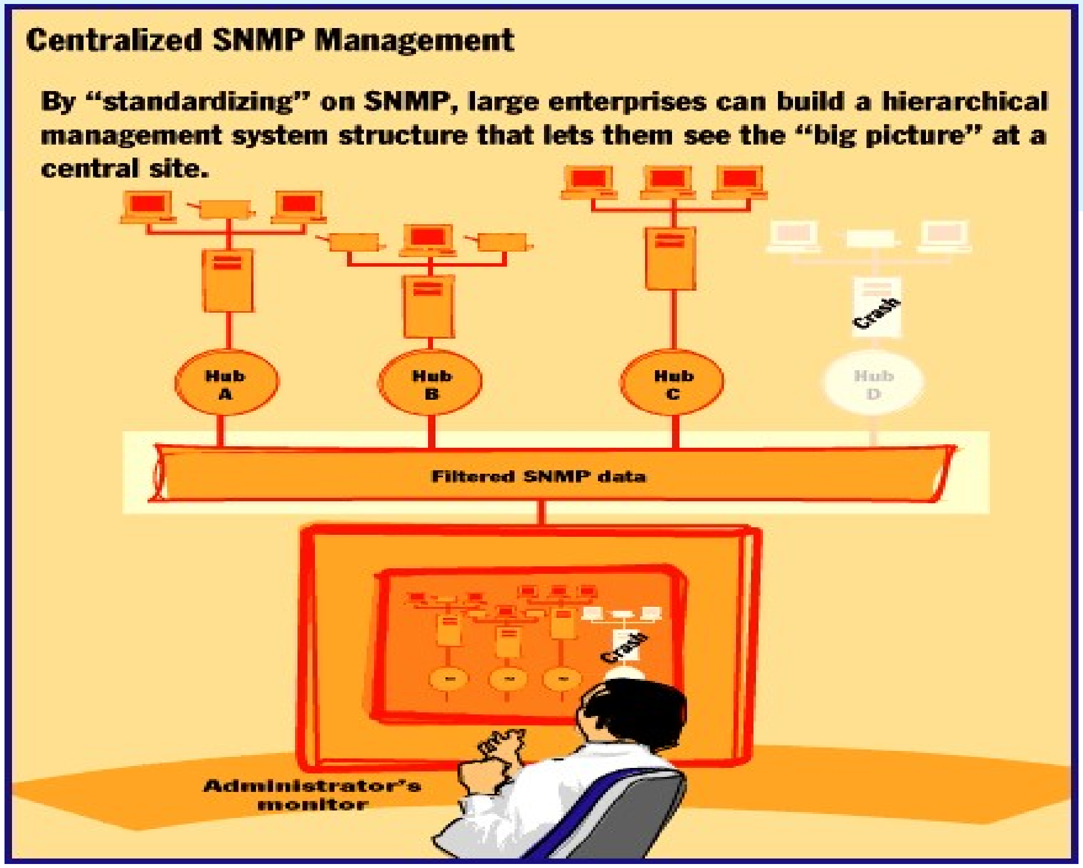
Get-request: Request value of variable.

Get-next-request: Request the value of next variable (like a table).

Set-request: update one or more variables (for reconfiguration).

Get-response: agents sends a reply to manager.

Traps: agent notifies manager about an abnormal situation.


Centralized SNMP Management
By "standardizing" on SNMP, large enterprises can build a hierarchical management system structure that lets them see the "big picture" at a central site.
Crash
Hub A
Hub B
Hub C
Hub D
Filtered SNMP data
Crash
Administrator's monitor

With a Network Management System, you can monitor a large network from a central console.

LIMITATIONS :

- SNMP is stateless in nature and each SNMP request contains all the information required 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 is no support for acknowledgement of requests.
- MIBs often lack writable objects without which device configuration is not possible using SNMP.
- 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

SNMP SECURITY :

✓ Lacks authentication. Vulnerable to the variety of security threats.

✓ Vulnerable to masquerading, modification of information, time modifications, message sequencing and disclosures.

✓Message sequence and timing modifications occurs when an entity who is unauthorized reorders, delays, or copies and later replays a message generated by an authorized entity.

Network Operator Requirements:

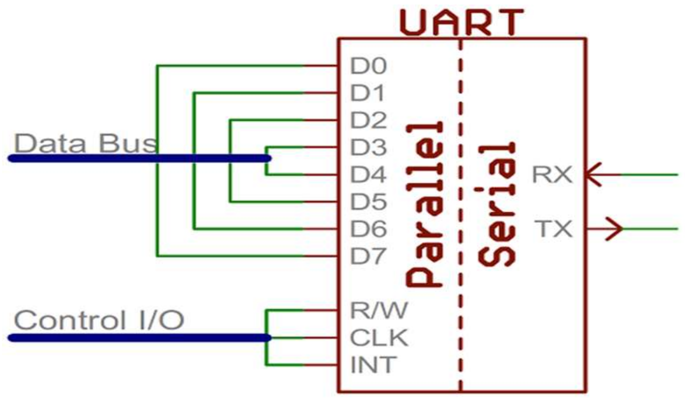
- ➢Depending on the specific devices and applications involved, an IoT network may require:
- ➢The ability to connect large numbers of heterogeneous IoT elements
- ➢High reliability
- ➢Real-time awareness with low latency
- ➢The ability to Secure all traffic flows
- ➢Programmability for application customization
- ➢Traffic monitoring and management at the device level
- ➢Low cost connectivity for large number of devices/sensors

UART

Universal Asynchronous Receiver/Transmitter

Hardware that translates between parallel and serial forms

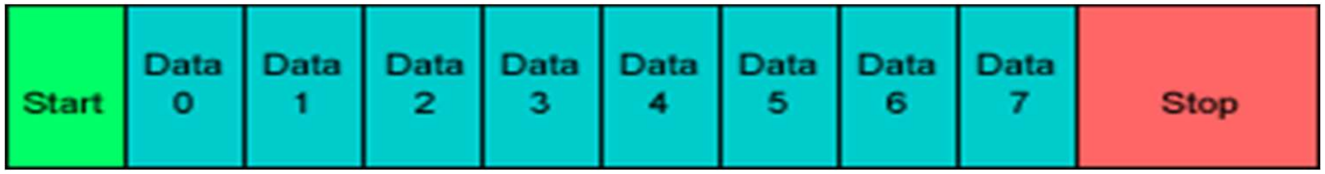
Commonly used in conjunction with communication standards such as EIA, RS-232, RS-422 or RS-485



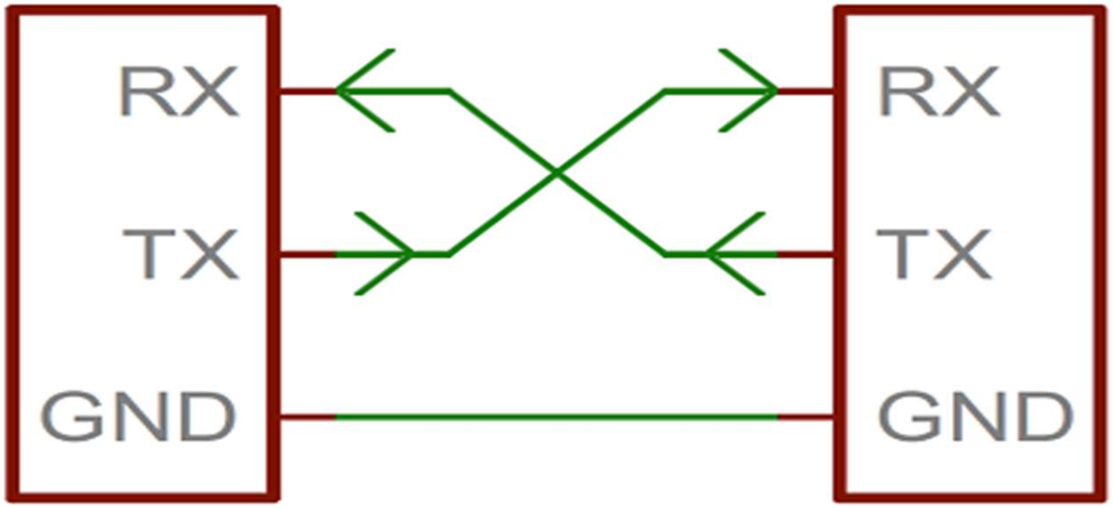
Protocol

Each character is sent as

- a logic *low* **<u>start</u>** bit
- a configurable number of data bits (usually 7 or 8, sometimes 5)
- an optional parity bit
- *one or more logic high* **<u>stop</u>** bits
- with a particular bit timing (“baud”)



UART Hardware Connection



Security Risks for using UART

Internet of Things (IoT) devices may support UART to send and transmit signals wirelessly. Manufacturers install UART interfaces on IoT boards to review serial console logs and complete any debug activity required. Since UART interacts with IoT devices, it is possible for hackers to infiltrate the UART shell and root shell. Shells manage user interaction with a computing system through an input-output interface. Hence, we have to take UART Security into consideration

If a cybercriminal gains access to the root shell, they can cause detriment to an organization. For example, hackers may:

✓Infiltrate and reverse engineer firmware to see how to exploit it further

✓Gain access to sensitive information such as certificates or API keys

✓Examine communication protocols for vulnerabilities

✓Target user devices, including company users and clients

Serial Peripheral Interconnect (SPI)

Another kind of serial protocol in embedded systems (proposed by Motorola)

Four-wire protocol

- SCLK — Serial Clock
- MOSI/SIMO — Master Output, Slave Input
- MISO/SOMI — Master Input, Slave Output
- SS — Slave Select

Single master device and with one or more slave devices

Higher throughput than I2C and can do “stream transfers”

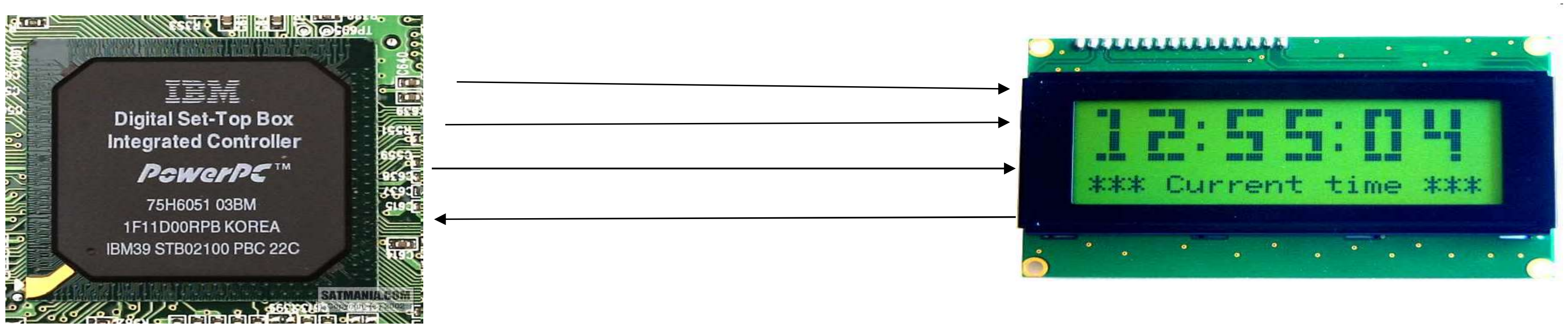
No arbitration required

But

- Requires more pins
- Has no hardware flow control
- No slave acknowledgment (master could be talking to thin air and not even know it)

What is SPI?

Serial Bus protocol
Fast, Easy to use, Simple
Everyone supports it



SPI Basics

A communication protocol using 4 wires

   Also known as a 4 wire bus

Used to communicate across small distances

Multiple Slaves, Single Master

Synchronized

SPI Capabilities

Always Full Duplex
- Communicating in two directions at the same time
- Transmission need not be meaningful

Multiple Mbps transmission speed

Transfers data in 4 to 16 bit characters

Multiple slaves
- Daisy-chaining possible

SPI Protocol

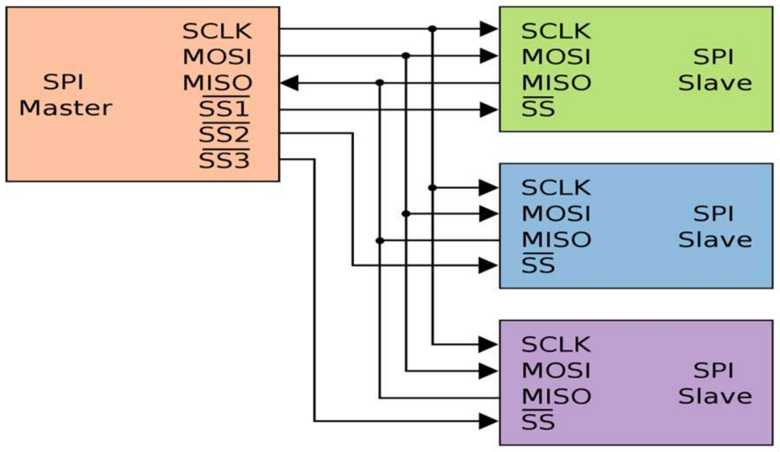
Wires:

- Master Out Slave In (MOSI)
- Master In Slave Out (MISO)
- System Clock (SCLK)
- Slave Select 1…N

Master Set Slave Select low

Master Generates Clock

Shift registers shift in and out data



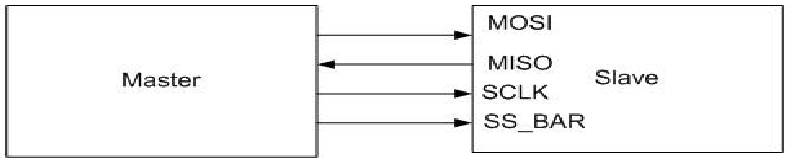
SPI Wires in Detail

MOSI – Carries data out of Master to Slave

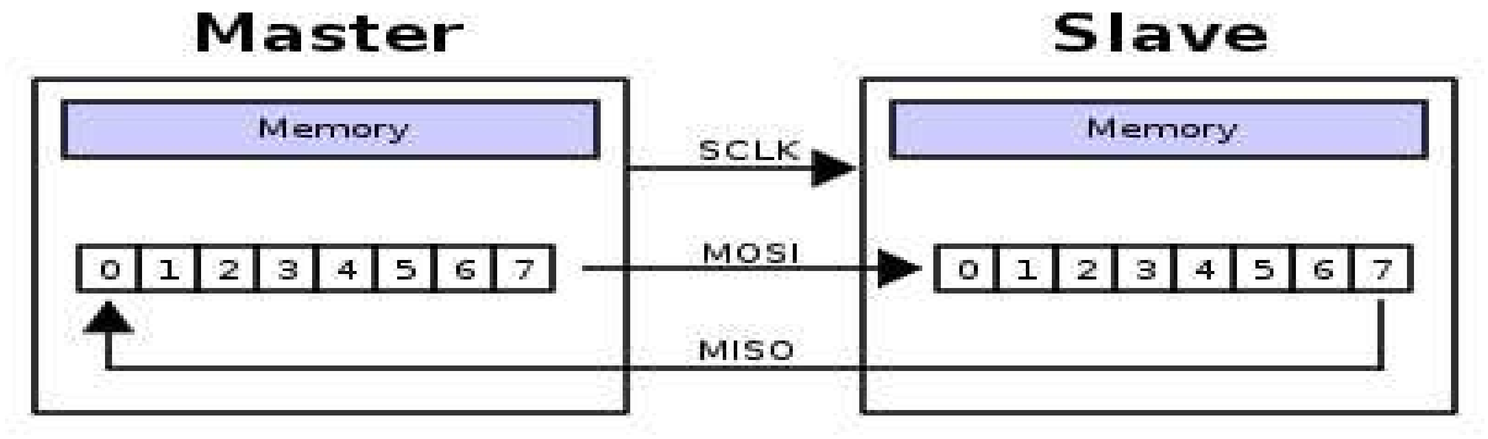
MISO – Carries data from Slave to Master
Both signals happen for every transmission

SS_BAR – Unique line to select a slave

SCLK – Master produced clock to synchronize data transfer



SPI uses a "shift register" model of communications



Master shifts out data to Slave, and shifts in data from Slave

http://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/SPI_8-bit_circular_transfer.svg/400px-SPI_8-bit_circular_transfer.svg.png

Possible Attack Scenarios

In an embedded device, the SPI protocol is used to communicate with various peripherals on the device. It could be sensors, control devices, LCD, SD Card, EEPROM, Flash memory, etc. If an attacker gets access to the device hardware, there are possibilities that he may find the SPI device as an attack surface on the hardware.

✓Sniffing the communication between an SPI device like sensors, controlling device, memory chip, etc. and the controller/processor.

✓Extracting the firmware/sensitive info from SPI memory chips.

✓Patching the firmware in the SPI flash memory chip.

Measures that can mitigate the attacks

➢Encrypt data on the memory chip to prevent leakage of sensitive info.

➢Do not hardcode sensitive data on the memory chips.

➢Store encrypted firmware/data

➢Physical hardening and protection of chips

What is I^2C

The name stands for “Inter - Integrated Circuit Bus”
A Small Area Network connecting ICs and other electronic systems
Originally intended for operation on one single board / PCB

- Synchronous Serial Signal
- Two wires carry information between a number of devices
- One wire use for the data
- One wire used for the clock

Today, a variety of devices are available with I^2C Interfaces

- Microcontroller, EEPROM, Real-Timer, interface chips, LCD driver, A/D converter

What is I^2C used for?

Data transfer between ICs and systems at relatively low rates

- "Classic" I^2C is rated to 100K bits/second
- "Fast Mode" devices support up to 400K bits/second
- A "High Speed Mode" is defined for operation up to 3.4M bits/second

Reduces Board Space and Cost By:

- Allowing use of ICs with fewer pins and smaller packages
- Greatly reducing interconnect complexity
- Allowing digitally controlled components to be located close to their point of use

I^2C Bus Characteristics

Includes electrical and timing specifications, and an associated bus protocol

Two wire serial data & control bus implemented with the serial data (SDA) and clock (SCL) lines

- For reliable operation, a third line is required:
- Common ground

Unique start and stop condition

Slave selection protocol uses a 7-Bit slave address

- The bus specification allows an extension to 10 bits

Bi-directional data transfer

Acknowledgement after each transferred byte

No fixed length of transfer

I^{2}C Bus Characteristics (cont’d)

True multi-master capability

- Clock synchronization
- Arbitration procedure

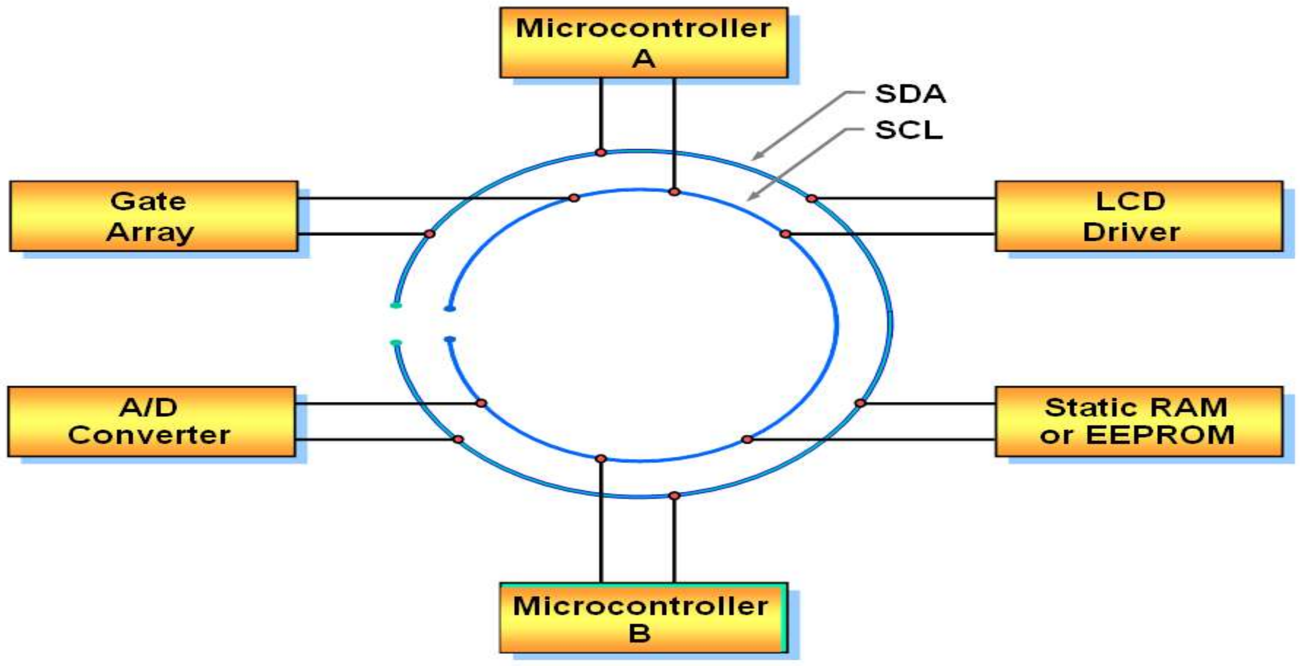
Transmission speeds up to 100Khz (classic I2C)

Max. line capacitance of 400pF, approximately 4 meters (12 feet)

Allows series resistor for IC protection

Compatible with different IC technologies

I^{2}C Bus Configuration Example



What measures can make it difficult for an attacker to attack:

✓Encrypt data on the memory chip to prevent leakage of sensitive info.

✓Do not hardcode sensitive data on the memory chips.

✓Store encrypted data

✓Physical hardening and protection of chips

✓Protection against clock glitching

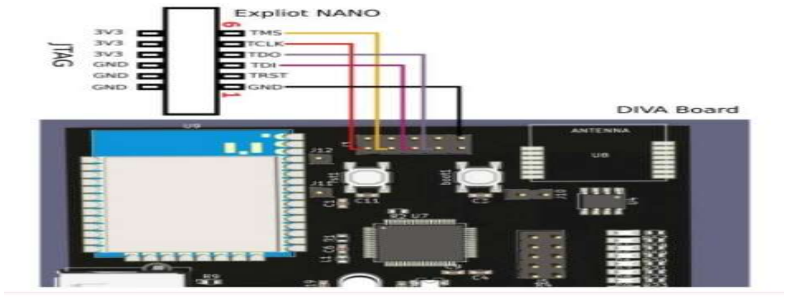
JTAG(Joint Test Action Group)

What is JTAG?
JTAG, commonly referred to as boundary-scan and defined by the Institute of Electrical and Electronic Engineers (IEEE) 1149.1, originally began as an integrated method for testing interconnects on printed circuit boards (PCBs) implemented at the integrated circuit (IC) level.

JTAG/boundary-scan presented an elegant solution to this problem: build functionality into the IC to assist in testing assembled electronic systems.

Today, JTAG is used for everything from testing interconnects and functionality on ICs to programming flash memory of systems deployed in the field and everything in-between.

JTAG and its related standards have been and will continue to be extended to address additional challenges in electronic test and manufacturing, including test of 3D ICs and complex, hierarchical systems.



Possible Attack Scenarios

Getting access to the JTAG/SWD interface on the hardware opens many possibilities for an attacker to break into the device. The following are the possible attack scenarios :

✓Attackers get access to the controller's internal memory leading to the manipulation of the register values.

✓Having access to the hardware with the JTAG/SWD interface gives the attacker the possibility of debugging the system.

✓Another most significant advantage from the attackers' perspective is that the access to JTAG debug interface opens up the possibilities to extract the firmware from the device, patch the firmware, and re-flash the modified vulnerable/malicious firmware back into the device.