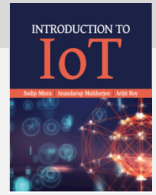
Communication Technologies

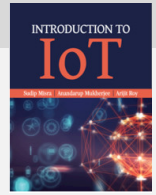
Computing for Internet of Things

Dr. Arijit Roy



Constrained Nodes

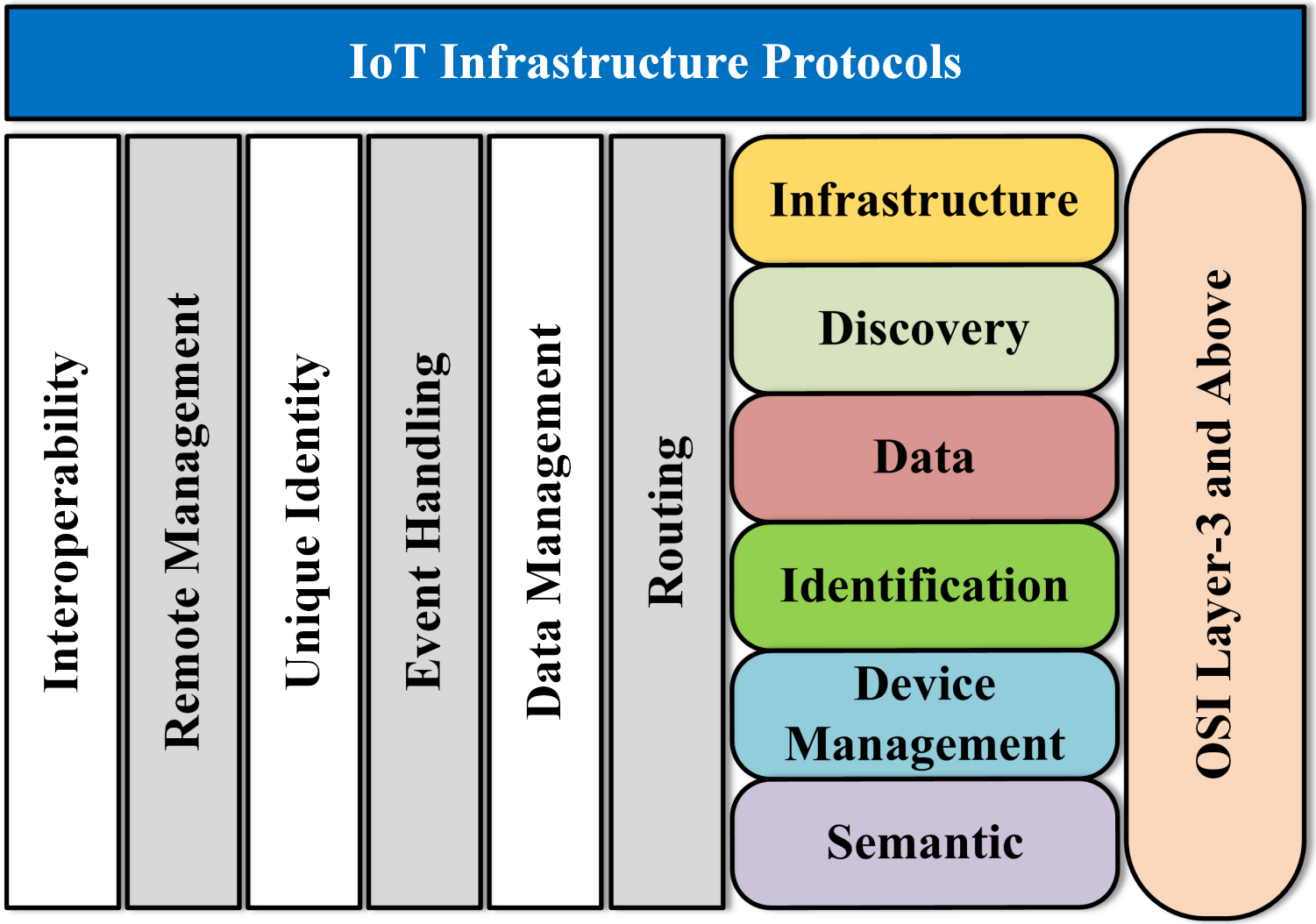
- Constrained nodes is a term associated with those nodes where regular features of Internet-communicating devices are generally not available.
- These drawbacks are often attributed to the constraints of costs, size restrictions, weight restrictions, and available power for the functioning of these nodes.
- The resulting restrictions of memory and processing power often limit the usage of these nodes.
- For example, most of these nodes have a severely limited layer-2 networking capability and often lack full connectivity features and broadcasting capabilities.
- These nodes require special design considerations while architecting their use in networks and networked applications.
- The issues of energy optimization and bandwidth utilization are dominant work areas associated with these nodes.

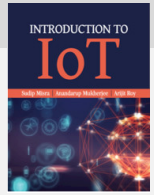


Constrained Networks

- limited processing power resulting in restrictions on achieving smaller duty cycles.
- low data-rates and low throughput.
- asymmetric links and increased packet losses.
- restrictions on supported packet sizes due to increased packet losses.
- lack of advanced layer-3 functions such as multicasting and broadcasting.
- limited temporal device reachability from outside the network due to the inclusion of sleep states for power management in the devices.

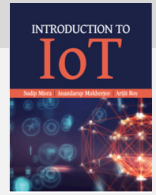
IoT Communication Protocol Groups





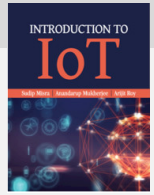
Types of Constrained Devices I

- **Class 0**: these devices are severely constrained regarding resources and capabilities. The barely feasible memory and processing available to these classes of devices do not allow for direct communication to the Internet. Even if the devices manage to communicate to the Internet directly, the mechanisms in place for ensuring the security of the device are not supported at all due to the device's reduced capabilities. Typically these class of devices communicates to the Internet through a gateway or a proxy.



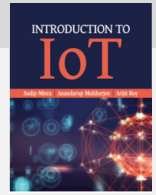
Types of Constrained Devices II

- **Class 1**: these devices are constrained concerning available code space and processing power. These devices can primarily talk to the Internet, but cannot employ a regular full protocol stack such as HTTP. Specially designed protocols stacks such as CoAP and other such protocols can be used to enable Internet-based communication with other nodes. Concerning Class 0 devices, Class 1 devices have a comparatively increased power budget, which is attributed to the increased functionalities it supports over Class 0 devices. These class of devices does not need a gateway for accessing the Internet and can be armed with security features for ensuring safer communication over the Internet.



Types of Constrained Devices III

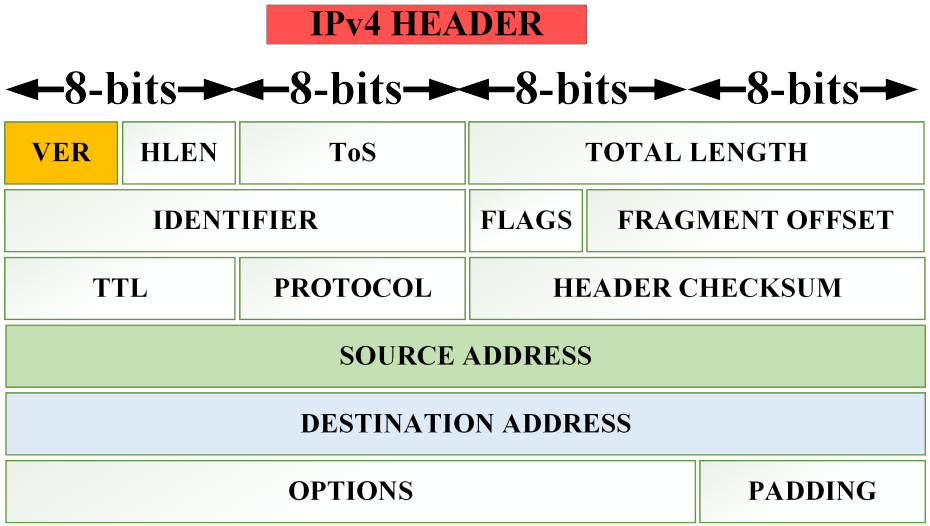
- **Class 2**: these devices are functionally similar to regular portable computers such as laptops and PDAs. They have the ability and capacity to support full protocol stacks of commonly used protocols such as HTTP, TLS, and others. However, as compared to the previous two classes of devices, these devices have a significantly high power budget.

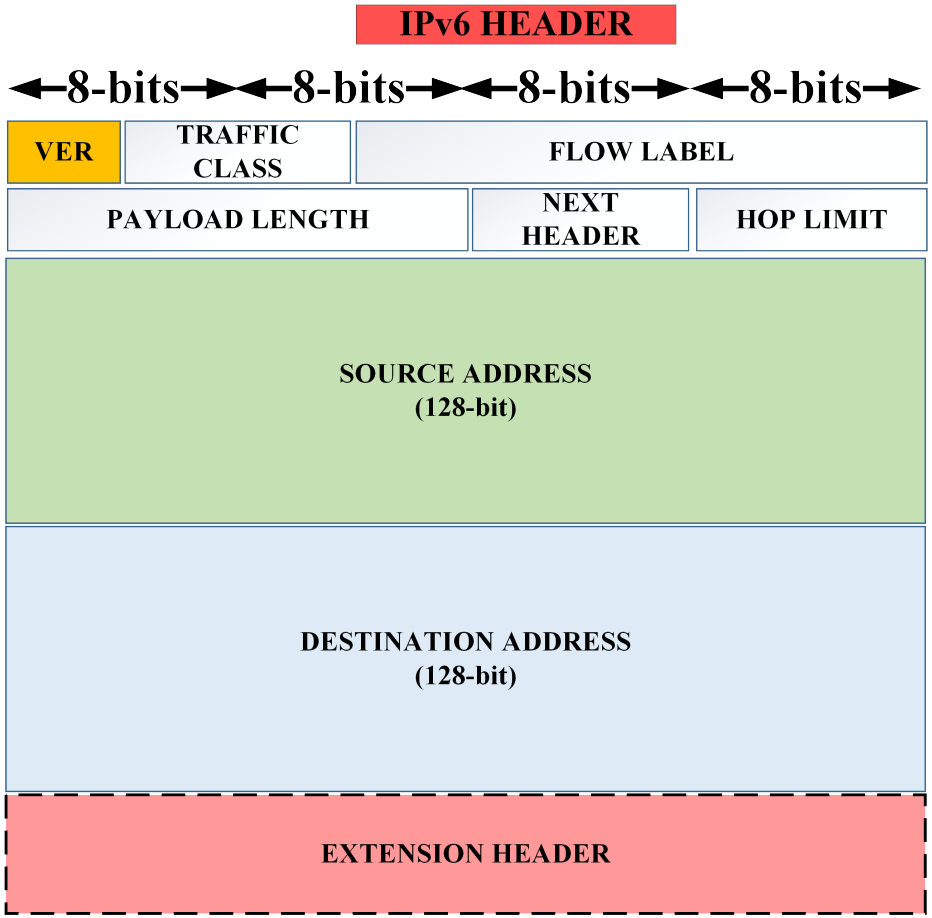


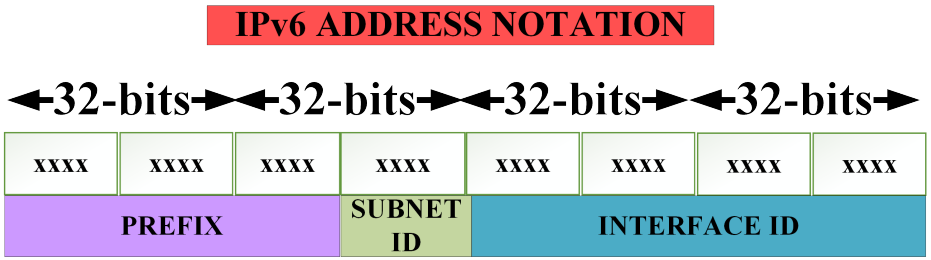
Low-Power and Lossy Networks

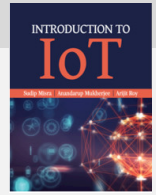
- Low-Power and Lossy Networks (LLNs), typically comprise of devices or nodes with limited power, small usable memory space, and limited available processing resources.
- The network links between the devices in this network may be composed of low-power Wi-Fi or may be based on the IEEE 802.15.4.
- The physical layers of the devices comprising LLNs are characterized by high variations in delivery rates, significant packet losses, and other similar behavior, which makes it quite unreliable, and often compromises network stability.
- However, LLNs have found extensive use in application areas such as industrial automation and monitoring, building automation, smart healthcare, smart homes, logistics, environment monitoring, and energy management.

Internet Protocol Version 6 (IPv6) I





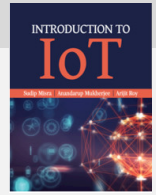




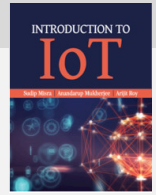
Internet Protocol Version 6 (IPv6) II

1. **Larger Addressing Range**: IPv6 has roughly four times more addressable bits than IPv4. This magnanimous range of addresses can accommodate the address requirements for any number of connected or massively networked devices in the world.
2. **Simplified Header Structure**: Unlike IPv4, the IPv6 header format is quite simple. Although much bigger than the IPv4 header, the IPv6 header's increased size is mainly attributed to the increased number of bits needed for addressing purposes.
3. **End-to-End Connectivity**: Unlike IPv4, the IPv6 paradigm allows for globally unique addresses on a significantly massive scale. This scheme of addressing enables packets from a source node using IPv6 to reach directly to a destination node without the need for network address translations en route (as is the case with IPv4).

Internet Protocol Version 6 (IPv6) III



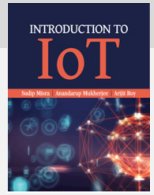
4. **Auto Configuration**: The configuration of addresses is automatically done in IPv6. It supports both stateless and stateful autoconfiguration methods and can work even in the absence of DHCP servers. This mechanism is not possible in IPv4 without DHCP servers.
5. **Faster Packet Forwarding**: As IPv6 headers have all the seldom-used optional fields at the end of its packet, the routing decisions by a router are taken much faster, only by checking the first few fields of the header.
6. **Inbuilt Security**: IPv6 supports inbuilt security mechanisms (IPSec), which is not directly supported in IPv4. IPv4 security measures were attained using separate mechanisms in conjunction with IPv4. The present-day version of IPv6 has security as an optional feature.



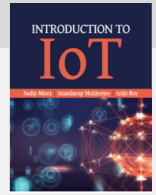
Internet Protocol Version 6 (IPv6) IV

7. **Anycast Support**: Multiple networking interfaces globally, are assigned the same IPv6 addresses, which are known as anycast addresses. This mechanism enables routers to send packets to the nearest available destination during routing.
8. **Mobility Support**: IPv6 has one of the essential features of mobility support, which is crucial for IoT and the modern-day connected applications. The mobility support of IPv6 allows for mobile nodes to retain their IP addresses and remain connected, even while changing geographic areas of operation.
9. **Enhanced Priority Support**: The priority support system in IPv6 is entirely simplified as compared to IPv4. The use of traffic classes and flow labels determine the most efficient routing paths of packets for the routers.

Internet Protocol Version 6 (IPv6) V

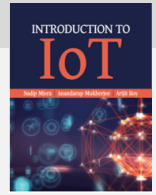


10. **Extensibility of Headers**: The options part of an IPv6 header can be extended by adding more information to it, and is not limited in size. Some applications may require quite large options filed, which may be comparable to the size of the packet itself.



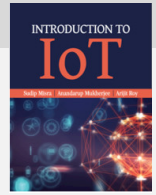
IPv6 Addressing I

- The IPv6 addressing scheme has a crucial component – the interface identifier (IID).
- IID is made up of the last 64-bits (out of the 128-bits) in the IPv6 address.
- IPv6 incorporates the MAC address of the system for IID generation.
- As a device's MAC address is considered as its hardware footprint and is globally unique, the use of MAC makes IID unique too.
- The IID is auto-configured by a host using IEEE's Extended Unique Identifier (EUI-64) format.
- IPv6 supports three types of unicasting – Global Unicast Address (GUA), Link-Local Address (LL), and Unique-Local Address (ULA).
- The GUA is synonymous with IPv4 static addresses (public IP), is globally identifiable, and uniquely addressable.



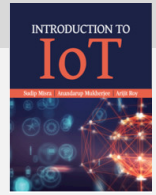
IPv6 Addressing II

- The global routing prefix is designated by the first (most significant) 48-bits.
- The first three bits of this routing prefix is always set to 001, which are also the most significant bits of this prefix.
- In contrast, LLs are auto-configured IPv6 addresses, whose communication is limited to within a network segment only (under a gateway or a router).
- The first 16-bits of LL addresses are fixed and equals to FE80 in hexadecimal. The subsequent 48-bits are set to 0.
- As these addresses are not routable, the LLs' scope is restricted within the operational purview of a router or a gateway.



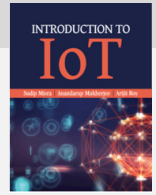
IPv6 Addressing III

- Finally, ULAs are locally global and unique and are meant for use within local networks only. Packets from ULAs are not routed to the Internet.
- The first half of ULA is divided into four parts and the last half is considered as a whole.
- The four parts of the first part are – Prefix, Local Bit, Global ID and Subnet ID, whereas the last half contains the IID. ULA's prefix is always assigned as FD in hexadecimal (1111 110 in binary). If the least significant bit in this prefix is assigned as 1, it signifies locally assigned addresses.



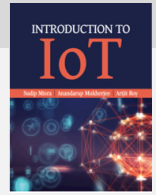
IPv6 Communication

- Any node in an IPv6 network is capable of auto-configuring its unique LL address.
- Upon assigning an IPv6 address to itself, the node becomes part of many multicast groups, which are responsible for any communication within that segment of the network.
- The node then sends a neighbor solicitation message to all its IPv6 addresses.
- If no reply is received in response to the neighbor solicitation message, the node assumes that there is no duplicate address in that segment, and its address is locally unique.
- This mechanism is known as duplicate address detection (DAD) in IPv6.
- Post DAD, the node configures the IPv6 address to all its interfaces and then sends out neighbor advertisements informing its neighbors about the address assignment of its interfaces. This step completes



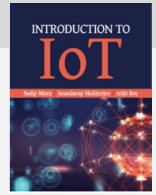
IPv6 Mobility I

- A mobile IPv6 node located within its home link uses its home address for routing all communication to it.
- However, whenever the mobile IPv6 node goes beyond its home link, it has to first connect to a foreign link for enabling communication.
- A new IPv6 address is acquired from the foreign link, which is also known as the mobile node's care-of-address (CoA).
- The mobile node now binds its CoA to its home agent (a router/gateway to which the node was registered in its home segment). This binding between the CoA and the home agent is done by establishing a tunnel between them.
- Whenever the node's home agent receives a correspondence message, it is forwarded to the mobile node's CoA over the established tunnel.



IPv6 Mobility II

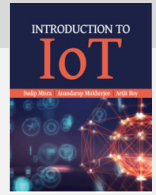
- Upon receiving the message from a correspondent node, the mobile node may choose not to reply through its home agent but can communicate directly to the correspondent node by setting its home address in the packet's source address field. This mechanism is known as route optimization.



LOADng I

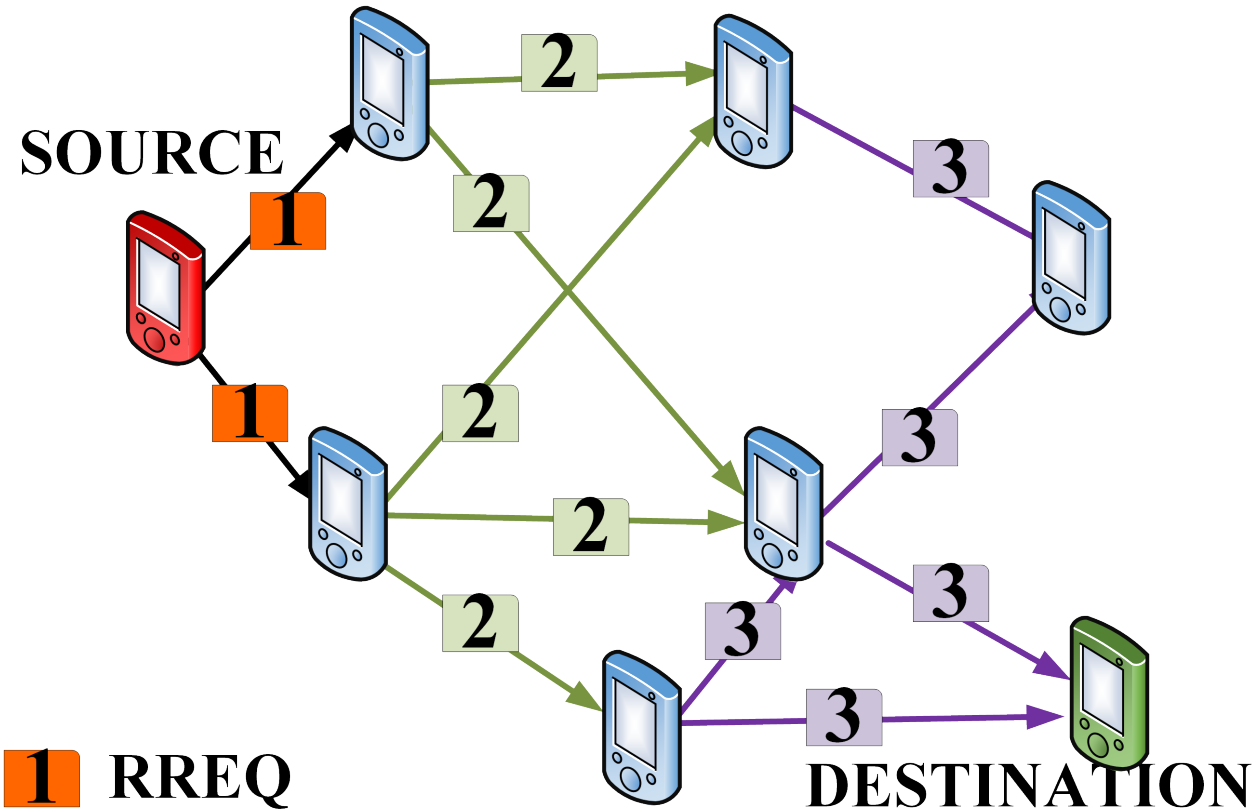
- LOADng stands for Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation.
- The LOADng is inspired by the AODV routing protocol, which is primarily a distance-vector routing scheme.
- Fig. 2 illustrates the LOADng operation. However, unlike AODV, LOADng was developed as a reactive protocol by taking into consideration the challenges of Mobile Ad-hoc Networks (MANETs).
- The LOADng process starts with the initiation of the action of route discovery by a LOADng router through the generation of Route Requests (RREQs).
- The router forwards packets to its nearest connected neighbors, each of which again forward the packets to their one-hop neighbors.
- This process is continued until the intended destination is reached.

LOADng II

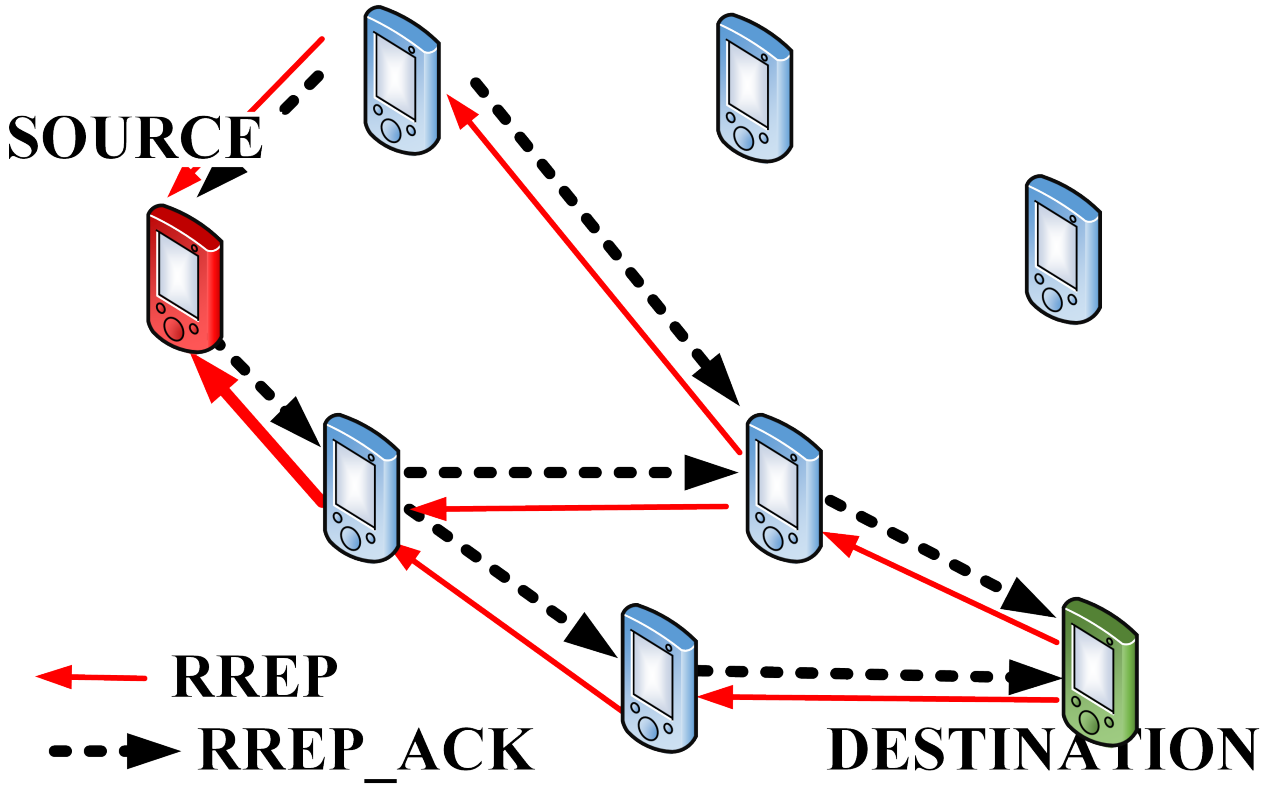


- Upon receiving the RREQ packet, the destination sends back a Route Reply (RREP) packet towards the RREP originating router.
- In continuation, Route Error (RERR) messages are generated and sent to the origin router if a route is found to be down between the origin and the destination.

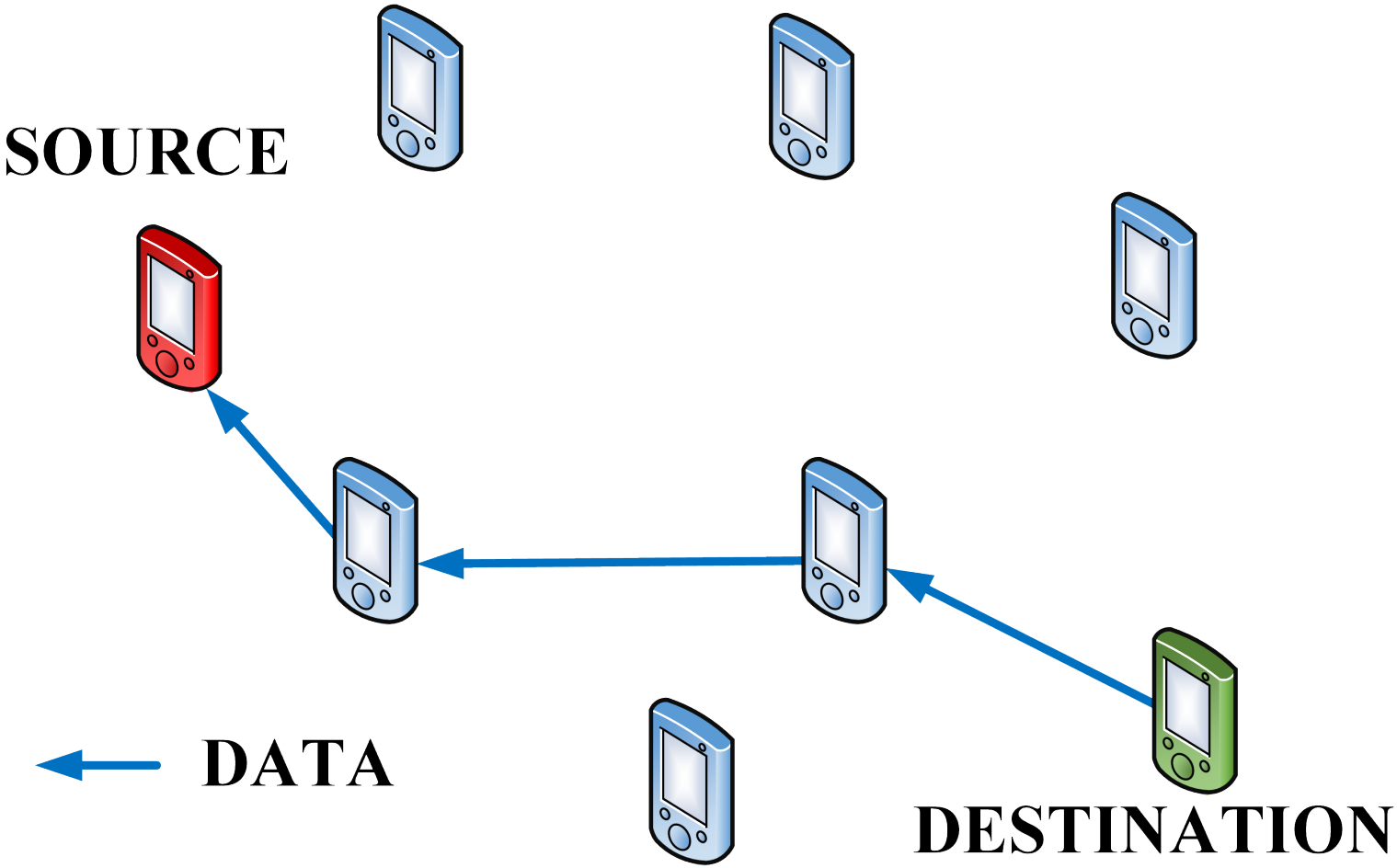
LOADng III

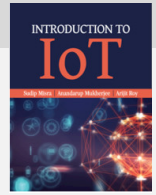


LOADng IV



LOADng V

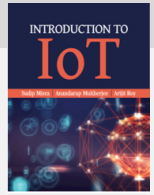




LOADng VI

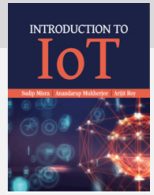
The summarize the operation of LOADng, a router performs the following tasks:

- Bi-directional network route discovery between a source and a destination.
- Route establishment and route maintenance between source and destination only when data is to be sent through the route.
- Generation of control and signaling traffic in the network only when data is to be transferred or a route to the destination is down.



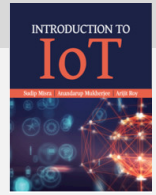
RPL I

- RPL stands for Routing Protocol for Low-Power and Lossy Networks (LLN) and is designed for IPv6 routing.
- RPL follows a distance vector-based routing mechanism.
- RPL aims to achieve a destination-oriented directed acyclic graph (DODAG). The network DODAG is formed based on an objective function and a set of network metrics.
- The DODAG built by RPL is a logical routing topology, which is built over a physical network.
- The logical topology is built using specific criteria set by network administrators.
- The most optimum path (best path) is calculated from the objective function, a set of metrics, and constraints.



RPL II

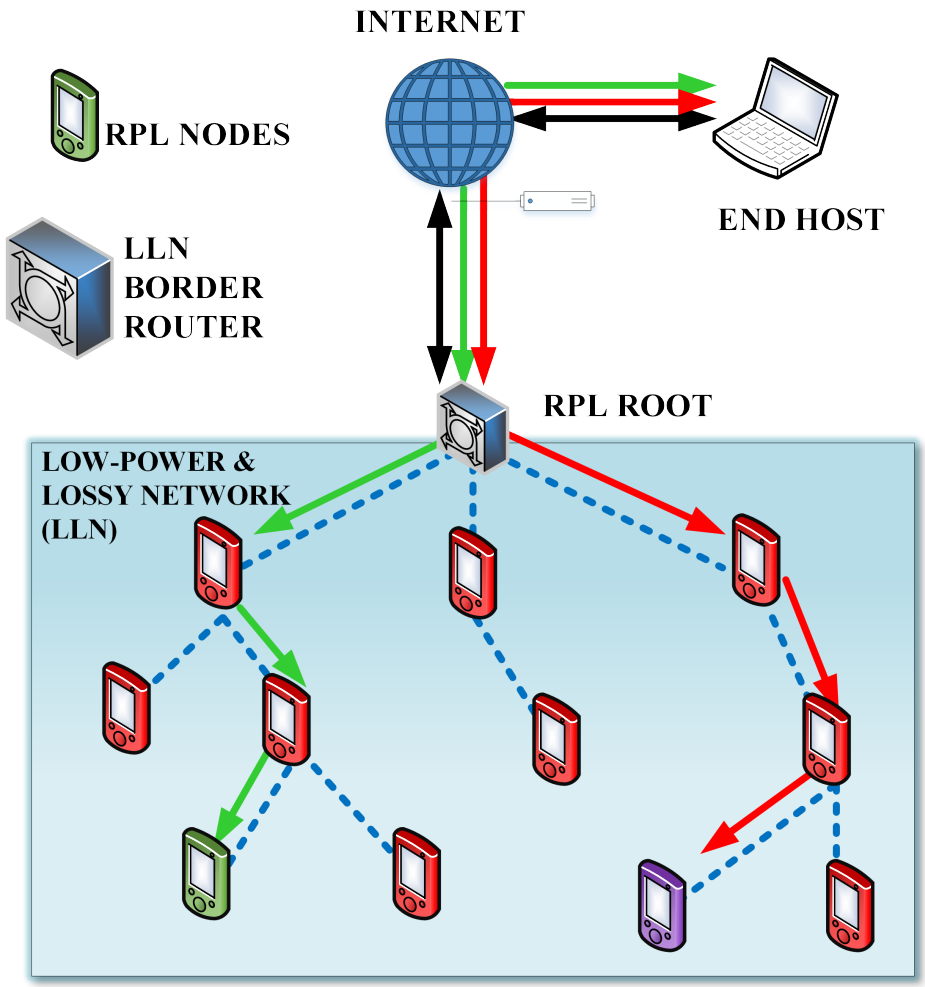
- The metrics in RPL may be expected transmission values (ETX), path latencies, and others.
- Similarly, the constraints in RPL include encryption of links, the presence of battery-operated nodes, and others. In general, the metrics are either minimized or maximized, whereas the constraints need to be minimized.
- The objective function dictates the rules for the formation of the DODAG. Interestingly, in RPL, a single node in the mesh network may have multiple objective functions. The primary reason for this is attributed to the presence of different network-traffic path-quality requirements that need separate addressal within the same mesh network.

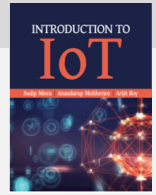


RPL III

- Using RPL, a node within a network can simultaneously join more than one RPL instance (graphs). This enables RPL to support QoS aware and constraint-based routing. An RPL node can also simultaneously take on multiple network roles – leaf node, router, and others.

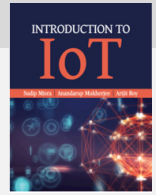
RPL IV


INTERNET
RPL NODES
END HOST
LLN
BORDER
ROUTER
RPL ROOT
LOW-POWER &
LOSSY NETWORK
(LLN)



6LoWPAN I

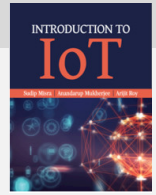
- 6LoWPAN allows low-power and constrained devices/nodes to connect to the Internet.
- 6LoWPAN stands for IPv6 over Low-Power Wireless Personal Area Networks.
- As the name suggests, it enables IPv6 support for WPANs, which are limited concerning power, communication range, memory, and throughput.
- 6LoWPAN is designed to be operational and straightforward over low-cost systems, and extend IPv6 networking capabilities to IEEE 802.15.4-based networks.
- Popular uses of this protocol are associated with smart grids, M2M applications, and IoT. 6LoWPAN allows constrained IEEE 802.15.4 devices to accommodate 128-bit long IPv6 addresses.



6LoWPAN II

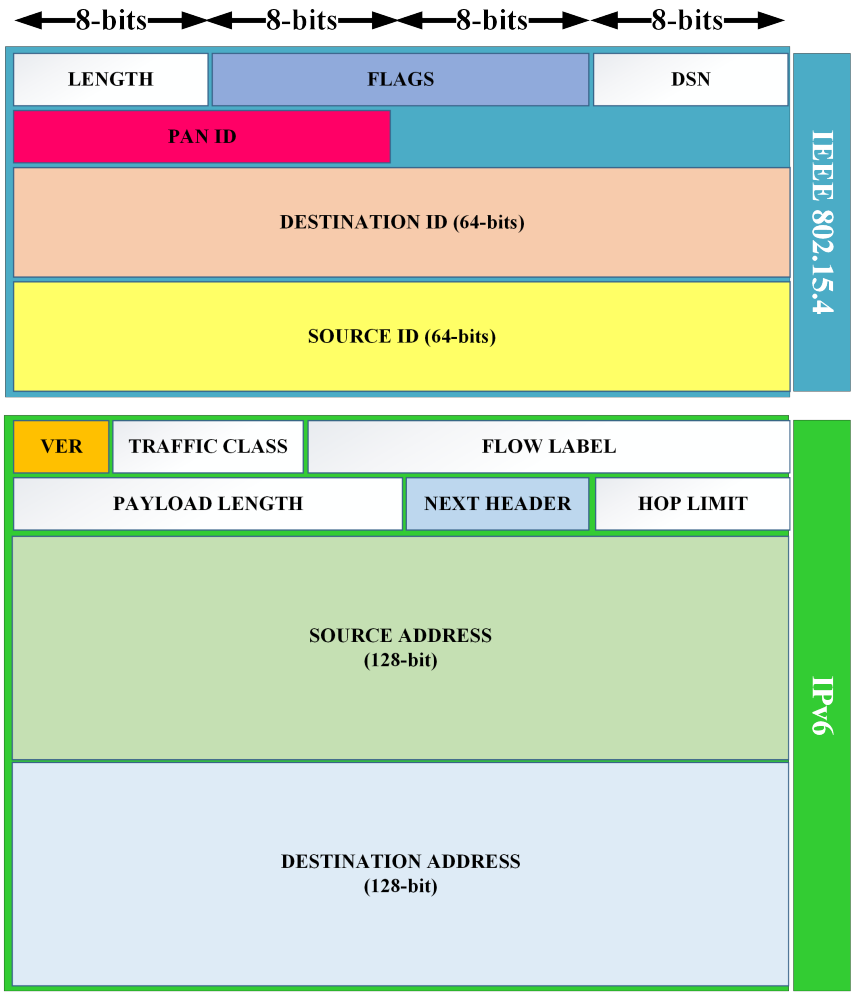
- This is achieved through header compression, which allows the protocol to compress and retro-fit IPv6 packets to the IEEE 802.15.4 packet format.
- 6LoWPAN networks can consist of both limited capability (concerning throughput, processing, memory, range) devices – called reduced function devices (RFD) – and devices with significantly better capabilities – called full-function devices (FFD).
- The RFDs are so constrained that for accessing IP-based networks, RFDs have to forward their data to FFDs in their Personal Area Network (PAN).
- The FFDs yet again forward the forwarded data from RFD to a 6LoWPAN gateway in a multi-hop manner.

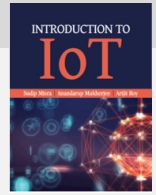
6LoWPAN III



- The gateway connects the packet to the IPv6 domain in the communication network. From here on, the packet is forwarded to the destination IP-enabled node/device using regular IPv6-based networking.

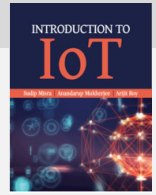
6LoWPAN IV


8-bits
8-bits
8-bits
8-bits
LENGTH
FLAGS
DSN
PAN ID
DESTINATION ID (64-bits)
SOURCE ID (64-bits)
IEEE 802.15.4
VER
TRAFFIC CLASS
FLOW LABEL
PAYLOAD LENGTH
NEXT HEADER
HOP LIMIT
SOURCE ADDRESS
(128-bit)
DESTINATION ADDRESS
(128-bit)
IPv6



6LoWPAN V

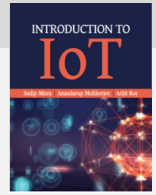
- **PHY and MAC layers**: The PHY layer consists of 27 wireless channels, each having their separate frequency band and varying data rates. The MAC layer defines the means and methods of accessing the defined channels and use them for communication. The 6LoWPAN MAC layer is characterized by the following:
  1. Beaconing tasks for device identification. Beaconing tasks include both beacon generation and beacon synchronization.
  2. Channel access control is provided by CSMA/CA.
  3. PAN membership control functions. Membership functions include association and dissociation tasks.



6LoWPAN VI

- **Adaptation layer**: As mentioned previously, 6LoWPAN accommodates and retro-fits the IPv6 packet to the IEEE 802.15.4 packet format. The challenge presented to 6LoWPAN is evident from the fact that IPv6 requires a minimum of 1280 octets for transmission. In contrast, IEEE 802.15.4 can support a maximum of 1016 octets (127 bytes) – 25 octets for frame overheads and 102 octets for payload. Additional inclusion of options in the IEEE 802.15.4 frame, such as security in the headers, leaves only 81 octets for IPv6 packets to use, which is insufficient. Even out of these available 81 octets, the IPv6 header reserves 40 octets for itself, 8 octets for UDP, and 20 octets for TCP, which are added in the upper layers. This leaves only 13 octets available at the disposal of the upper layers and the data itself. The 6LoWPAN adaptation layer between the MAC and network

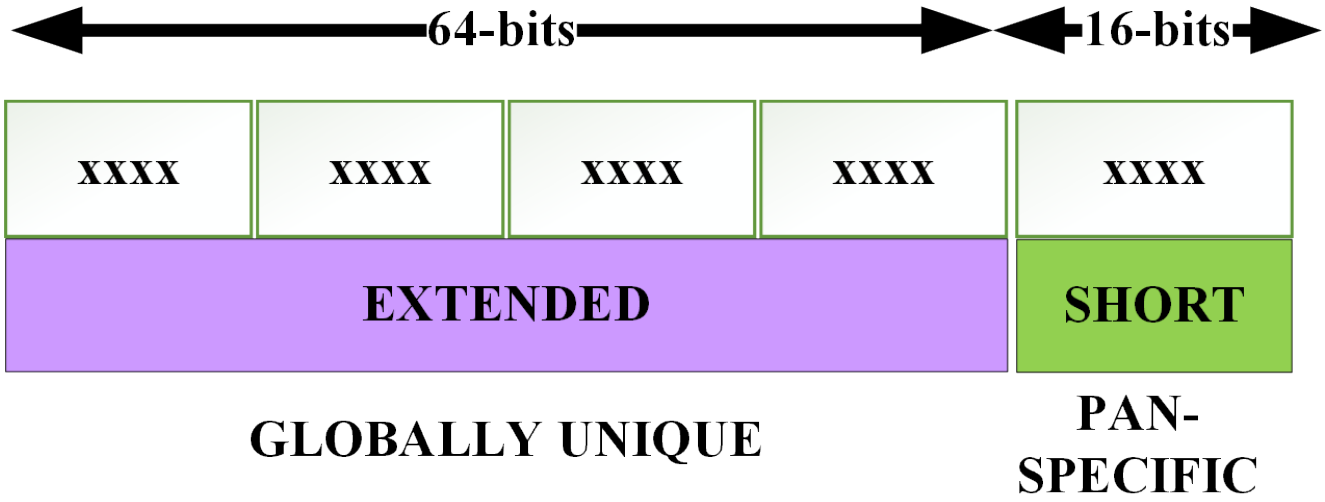
6LoWPAN VII

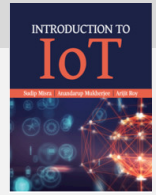


layers takes care of these issues through the use of header compression, packet forwarding, and packet fragmentation.

- **Address Format**: The 6LoWPAN address format is made up of two parts – 1) the short (16-bit) address and 2) the extended (64-bit) address. The short address is PAN specific and is used for identifying devices within a PAN only, which makes its operational scope highly restricted and valid within a local network only. In contrast, the globally unique extended address is valid globally and can be used to identify devices, even outside a local network uniquely. Fig. **??** illustrates the 6LoWPAN address format.

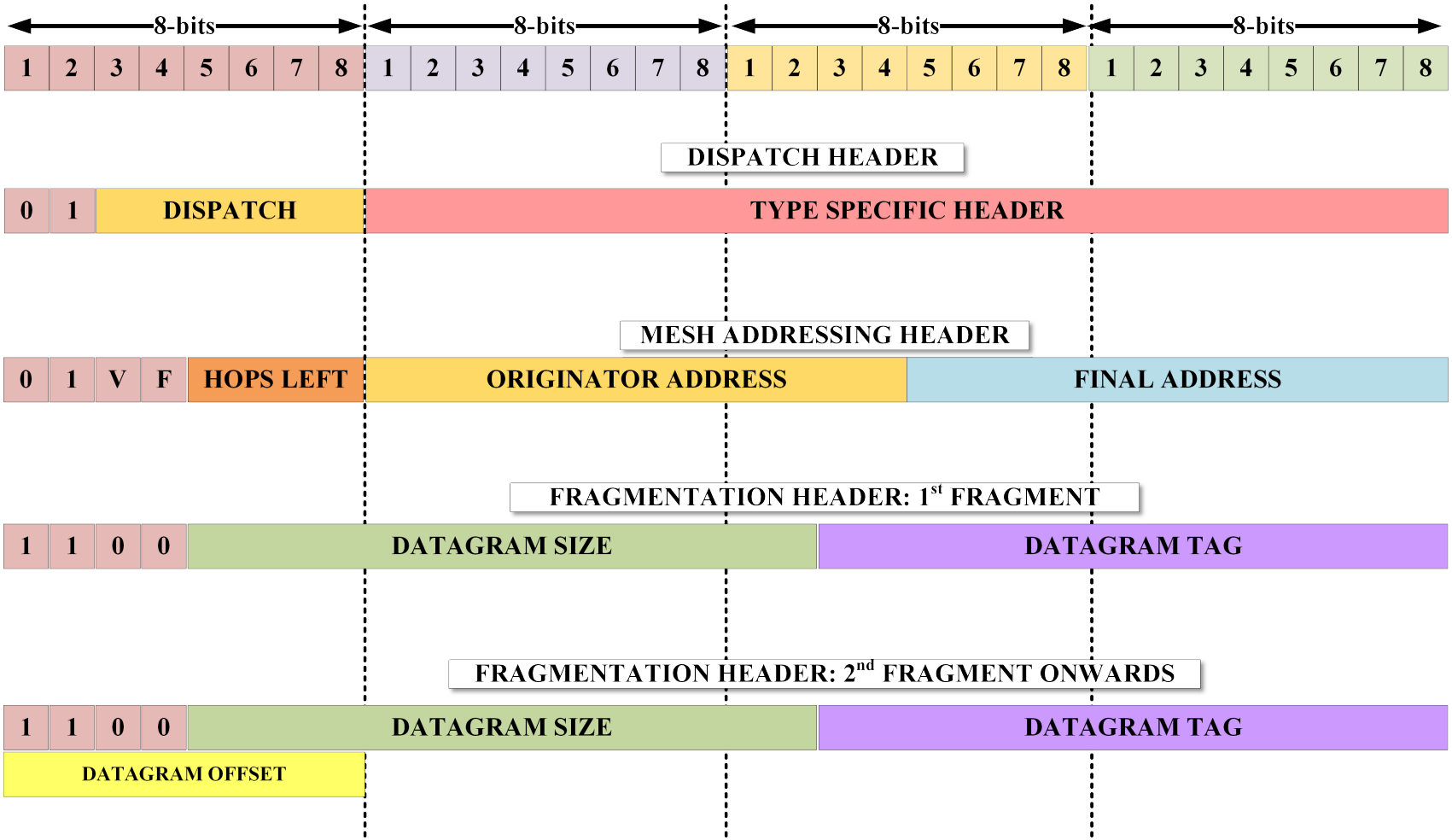
6LoWPAN VIII

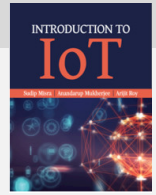




6LoWPAN Encapsulation Header Formats

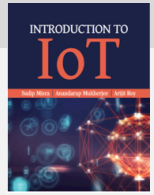
- The encapsulation headers, as the name suggests, defines methods and means to encapsulate the IPv6 payloads within IEEE 802.15.4 frames by 6LoWPAN.
- 6LoWPAN has three encapsulation header types associated with it – dispatch, mesh addressing, and fragmentation.
- This system is similar to the IPv6 extension headers.
- The headers are identified by a *header type* field placed in front of the headers.
- The dispatch header type is used to initiate communication between a node and a destination node.
- The mesh addressing header is used for multi-hop forwarding by providing support for layer-two forwarding of messages.
- Finally, the fragmentation header is used to fit large payloads to the IEEE 802.15.4 frame size.


8-bits
8-bits
8-bits
8-bits
1 2 3 4 5 6 7 8
DISPATCH HEADER
0 1 DISPATCH
TYPE SPECIFIC HEADER
MESH ADDRESSING HEADER
0 1 V F HOPS LEFT
ORIGINATOR ADDRESS
FINAL ADDRESS
FRAGMENTATION HEADER: 1st FRAGMENT
1 1 0 0
DATAGRAM SIZE
DATAGRAM TAG
FRAGMENTATION HEADER: 2nd FRAGMENT ONWARDS
1 1 0 0
DATAGRAM SIZE
DATAGRAM TAG
DATAGRAM OFFSET



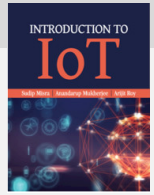
QUIC I

- Quick UDP Internet Connections (QUIC) was developed (and still undergoing developments) to work as a low-latency and independent TCP connection.
- The aim behind the development of this protocol is to achieve a highly reduced latency (almost zero round-trip-time) communication scheme with stream and multiplexing support like the SPDY protocol developed by Google.
- The connection latency in QUIC is reduced by reducing the number of round trips incurred during connection establishment in TCP, such as those for handshaking, data requests, and encryption exchanges.
- This is achieved by including session negotiation information in the initial packet itself.



QUIC II

- The QUIC servers further enhance this compression by publishing a static configuration record corresponding to the connections.
- Clients synchronize connection information through cookies received from QUIC servers.
- QUIC uses advanced techniques such as packet pacing and proactive speculative retransmission to avoid congestion.
- Proactive speculative retransmission sends copies of most essential packets, which contain initial negotiation for encryption and error correction.
- The additional speedup is achieved using compression of data such as headers, which are generally redundant and repetitive.



QUIC III

- This feature enables QUIC connections to make multiple secured requests within a single congestion window, which would not have been possible using TCP.
- The use of UDP and multiple transmission paths significantly speeds-up the performance of streaming over QUIC as compared to regular HTTP-based packet streaming.

QUIC IV

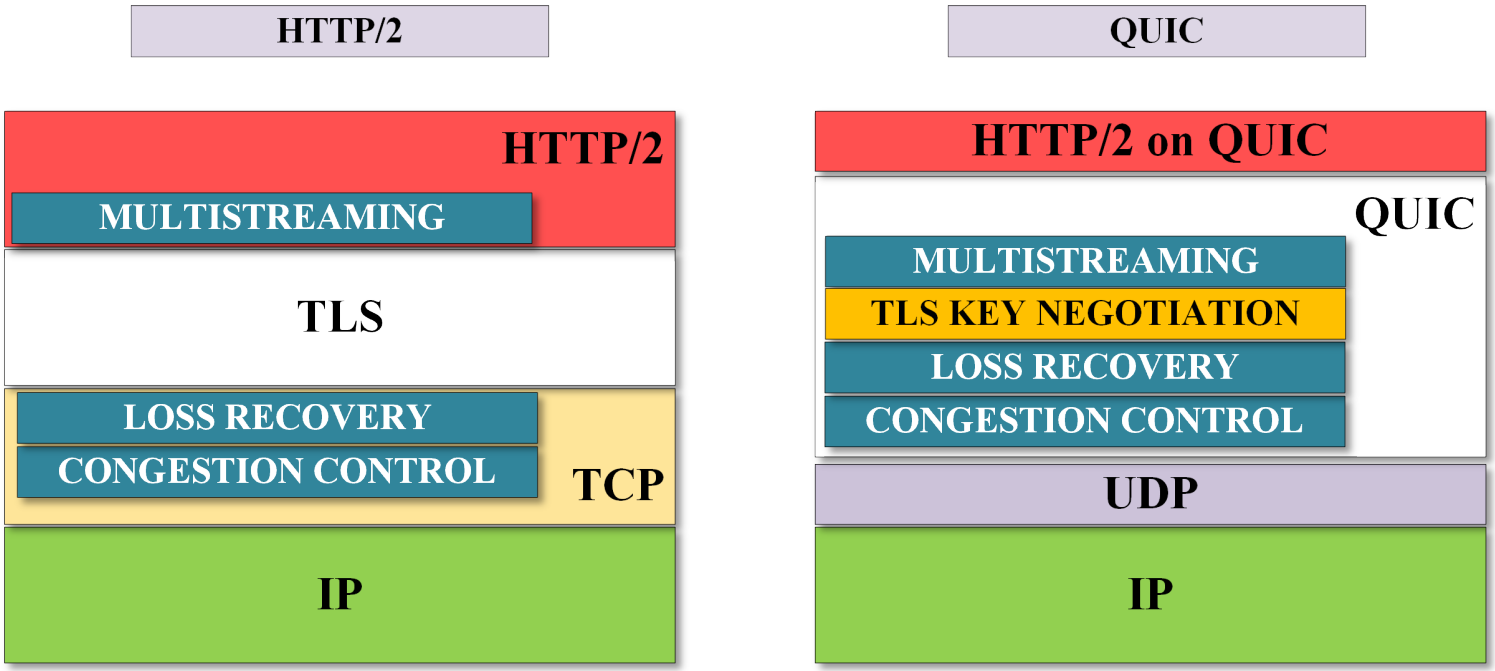


Figure: Differences between HTTP and QUIC protocols

QUIC V

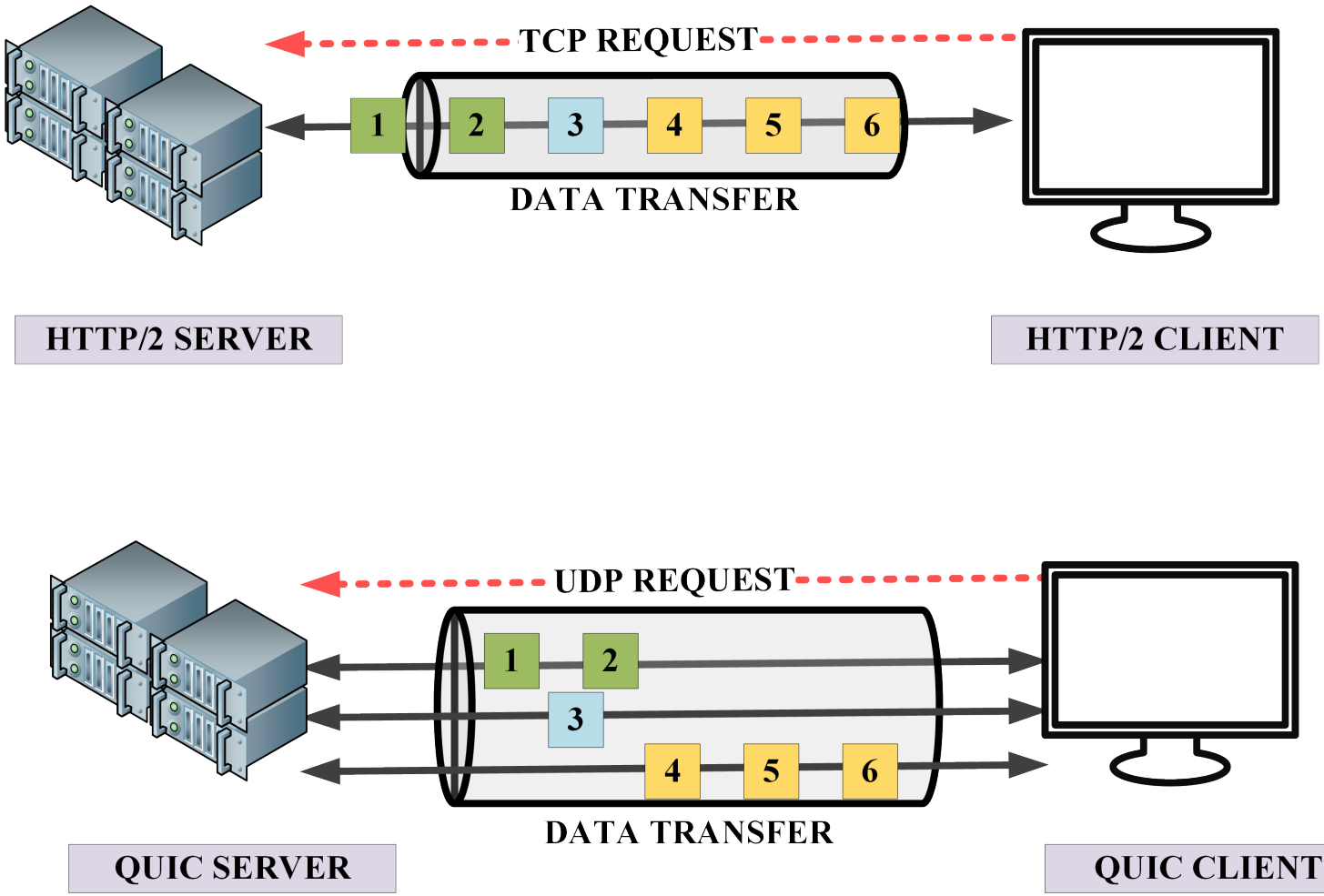
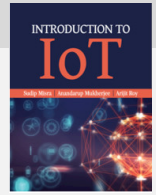
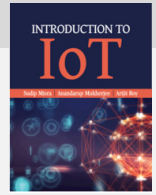


Figure: Differences between stream of packets over HTTP and QUIC protocols



Micro Internet Protocol (uIP) I

- The Micro-IP (uIP) protocol is developed to extend the TCP/IP protocol stack capabilities to 8-bit and 16-bit microcontrollers.
- uIP is an open-source protocol and developed by the Swedish Institute of Computer Science (SICS).
- The low code-space and memory requirements of uIP make it significantly useful for networking low-cost and low-power embedded systems.
- uIP now features a full IPv6 stack, which was developed jointly by Atmel, Cisco, and SICS.
- The software interface of uIP does not require any operating system for working, making it quite easy to integrate with small computers.
- When called in a timed-loop within the embedded system, it also manages all the network behavior and connection retries.



Micro Internet Protocol (uIP) II

- The hardware driver for the uIP is responsible for packet builds, packet sending, and also may be used for response reception for the packets sent.
- uIP uses a minimal packet buffer (packet buffer =1) in contrast to normal IP protocol stacks. This makes uIP suitable for low-power operations.
- The packet buffer is used in a half-duplex manner so that the same buffer can be repurposed for use in transmission and reception.
- Unlike regular TCP/IP protocols, uIP does not store data in buffers in case there is a need for retransmission. In the event of retransmission of packets, the previous data has to be reproduced and is recalled from the application code itself.

Micro Internet Protocol (uIP) III

- Unlike conventional IP-based protocols, where a task is dedicated for each connection to a distant networked device/node, uIP stores connections in an array, and serves each connection sequentially through subroutine calls to the application for sending data.

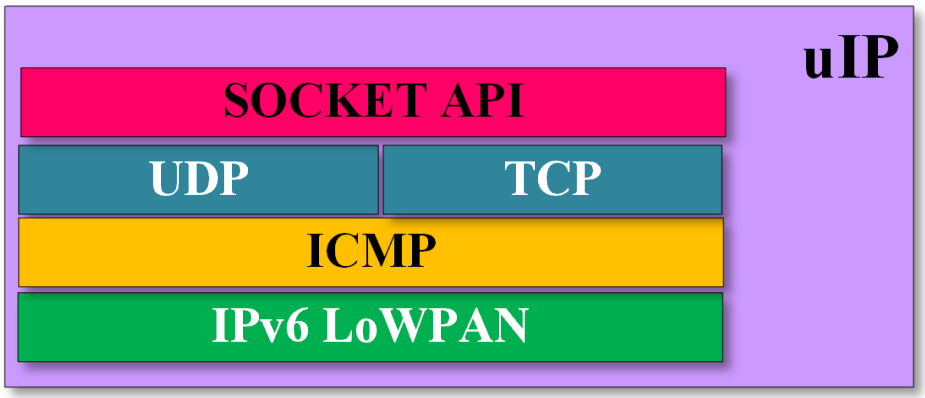
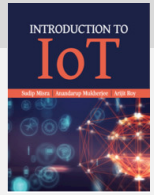


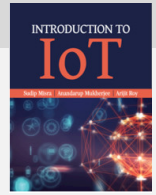
Figure: The uIP protocol



Nano Internet Protocol (nanoIP) I

- The Nano Internet Protocol or NanoIP was designed to work with embedded devices and specifically sensor devices by enabling internetworking amongst these devices.
- The concept of nanoIP enables wireless networking among low-power sensor devices, which is address-based but without incurring the overheads associated with the TCP/IP protocol stacks and mechanisms.
- NanoIP is made up of two two transport mechanisms – nanoUDP and nanoTCP. These two transport mechanisms are analogous to the conventional UDP (unreliable transport protocol) and TCP (reliable transport protocol), respectively.
- NanoTCP even supports packet retransmissions and flow-control, just like regular TCP.

Nano Internet Protocol (nanoIP) II



- Instead of logical addressing, nanoIP uses hardware (MAC) addresses of devices for networking.
- The supported port range is 256 each for source and destination nodes, which is the allowable limit for an 8-bit port representation. Because of the nanoIP, several associated protocols have also come up, such as nHTTP and nPing.

Nano Internet Protocol (nanoIP) III

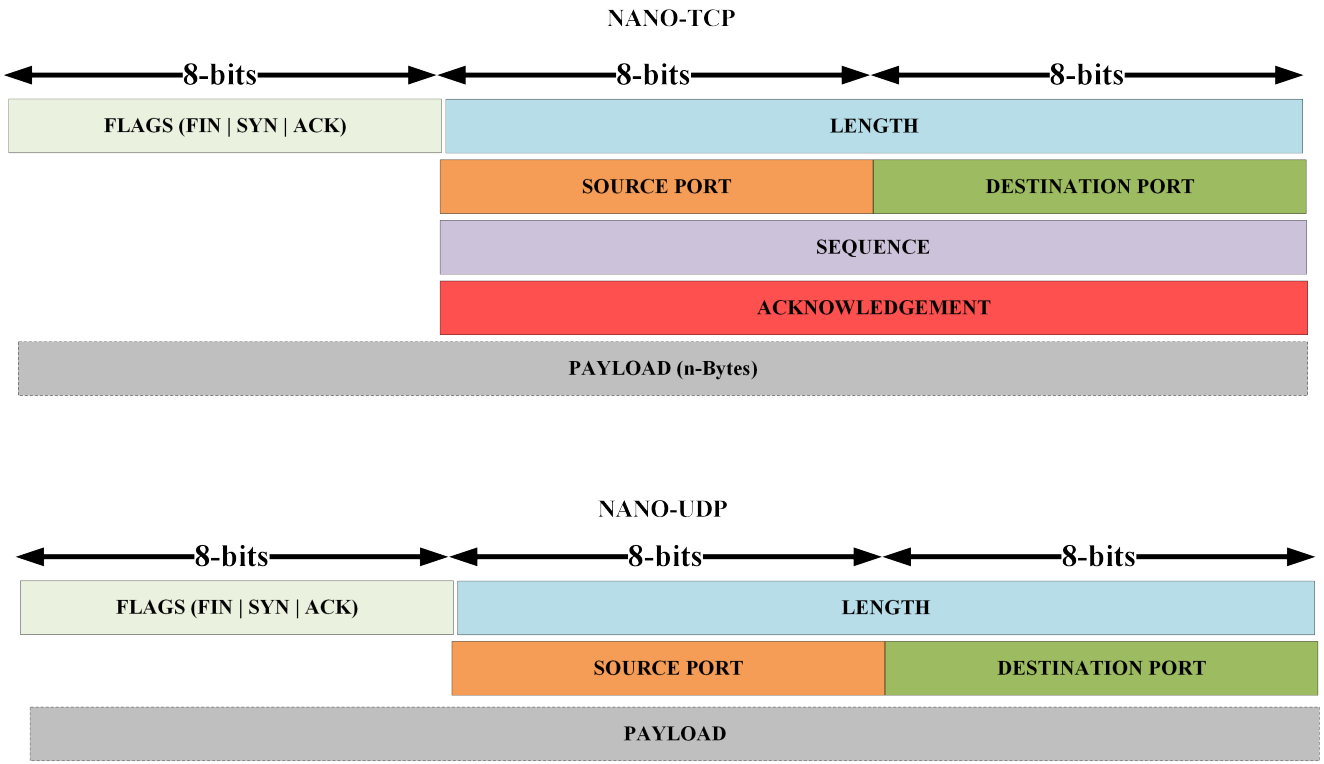
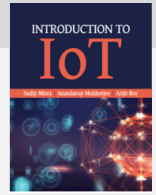


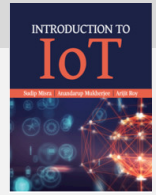
Figure: The nano-IP TCP and UDP protocols



Content Centric Networking (CCN) I

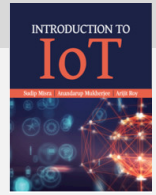
- The Content-Centric Networking (CCN) paradigm is majorly known as Information-Centric Networking (ICN).
- Other names associated with this paradigm are Named Data Networking (NDN) and Publish-Subscribe Networking (PSIRP).
- CCN enables communication by defining and adhering to the concept of uniquely named data.
- This networking paradigm, unlike conventional networking approaches, is independent of location, application, and storage requirements.
- CCN is anchorless, which enables mobility and focuses on in-network caching for operations.
- These measures extend the features of data and communication efficiency, enhanced scalability, and robustness, even in constrained and challenged networks.

Content Centric Networking (CCN) II



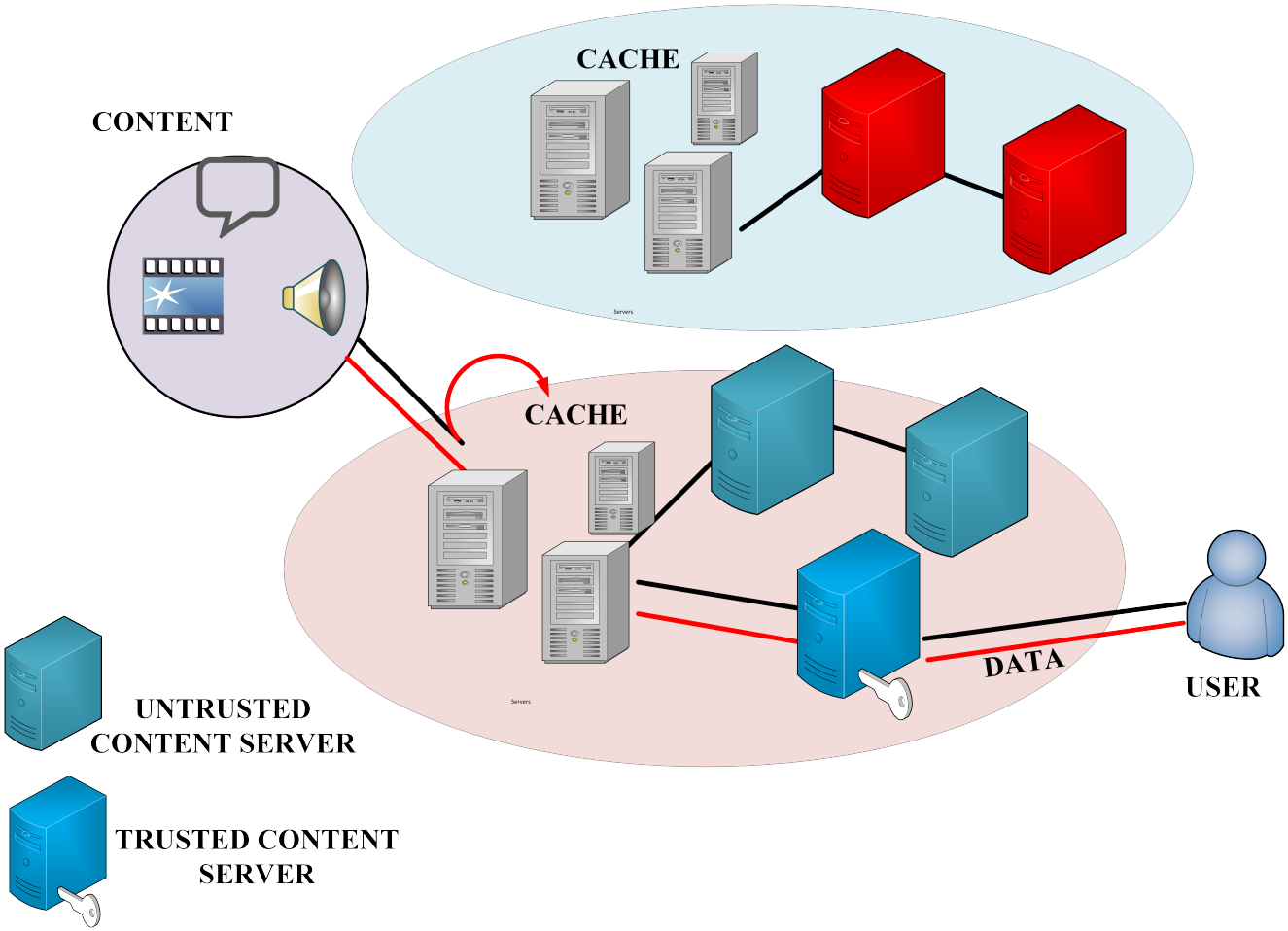
- Users can access cached content from multiple content-generating sources by accessing data from trusted content servers, which also enable security of the data (not the communication channel).
- In CCN, a forwarder checks a named request through hierarchical prefix matching (typically, longest prefix match) with a forwarding information base (FIB).
- A binary comparison is performed for prefix matching. The CCN request is a hierarchical sequence of network path segments.
- The FIB matching is then used to forward the named request to the appropriate network or network segment, which can respond to the issued request.

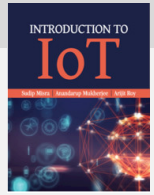
Content Centric Networking (CCN) III



- The forwarder has to additionally determine the reverse path from the responder to the requester. All these operations are carried out without specifically binding a request to a network end-point.
- The FIB at each CCN router stores information in a table, which is updated by a routing mechanism.
- Although the path segments and names are theoretically unbounded, however, due to practical reasons, these are restricted by the routing protocol being used.

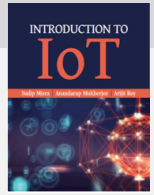
Content Centric Networking (CCN) IV





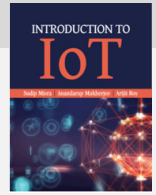
Physical Web I

- The physical web was designed to provide its users with the ability to interact with physical objects and locations seamlessly.
- The information to the users can be in the form of regular web pages or even dynamic web applications.
- Some examples in the context of the physical web include user-friendly buses, which can alert its users about various route-related information, smart home appliances that could teach new users how to use them, self-diagnostic robots and machinery in industries, smart pet tags which could provide information about the pet's owner and its home location, and many more.
- The main takeaway of this concept is the seamless integration of several standalone smart systems through the web to provide information on demand to its users.



Physical Web II

- The physical web broadcasts a list of URLs within a short radius around it so that anyone in range can see the available URLs and interact with them.
- This paradigm primarily is built upon Bluetooth Low Energy (BLE), which is used to broadcast the content as beacons.
- The primary requirement of any supporting beacons to function in the physical web, and broadcast URLs, is its ability to support the Eddystone protocol specification.
- BLE was primarily chosen for the physical web due to its ubiquity, efficiency, and extended battery life of several years.
- The URLs are one of the core principles of the web and can be either flexible or decentralized.



Physical Web III

- These URLs allow any application to have a presence on the web and enables the digital presence of an object or thing.
- As of now, physical web deployments have been undertaken in public spaces, and any device with a physical web scanner can detect these URLs.
- The use of URLs extends the benefits of the web security model to the physical web. Features such as secured login, secure communication over HTTPS, domain obfuscation, and others can be easily integrated with the physical web.

Physical Web IV

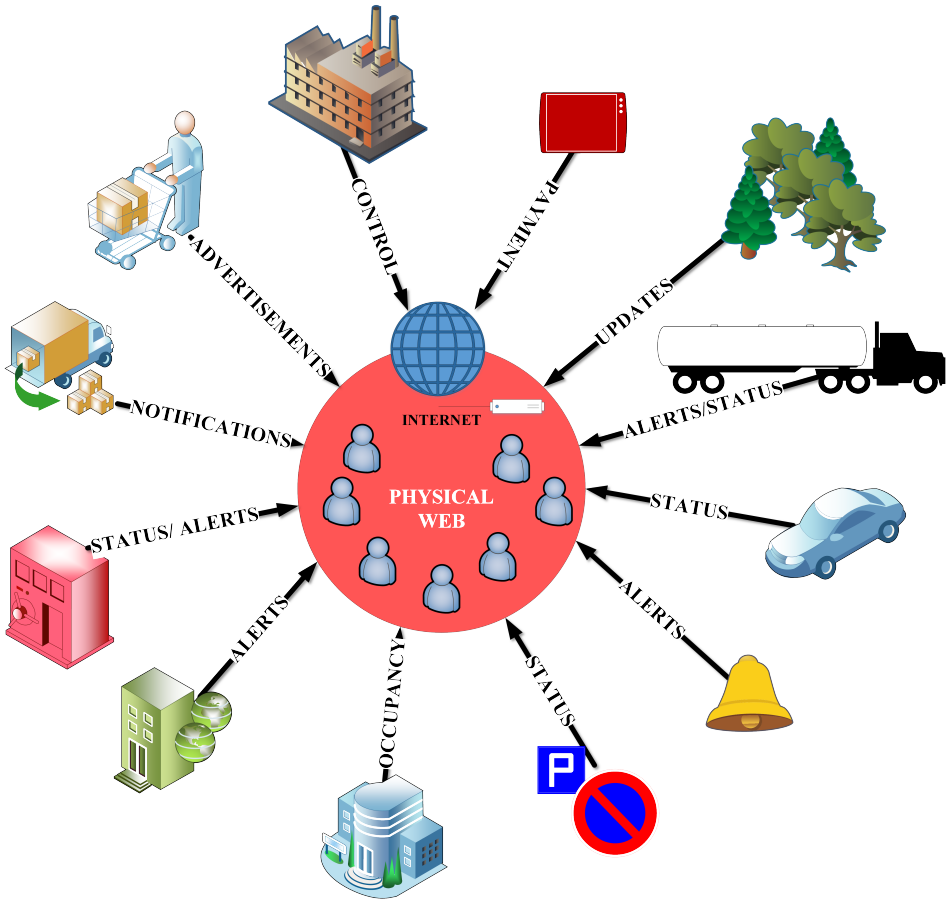
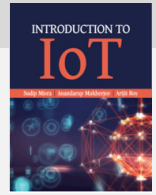


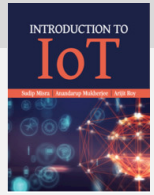
Figure: The physcial web model



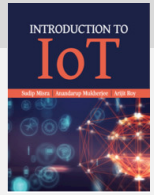
Multicast DNS (mDNS) I

- The multicast domain name system or mDNS is explicitly designed for small networks and is analogous to regular DNS, which is tasked with the resolution of IP addresses.
- Interestingly, this system is free from any local name server from an operational point of view.
- However, it can work with regular DNS systems as it is a zero-configuration service. It uses multicast UDP for resolving hostnames.
- An mDNS client initiates a multicast query on the IP network, which asks a remote host to identify itself.
- The mDNS cache in the associated network subnet is updated with the multicast response received from the target.

Multicast DNS (mDNS) II

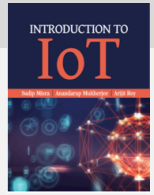


- A node can give-up its claim to a domain name by setting the time-to-live (TTL) field to zero in its response packet to an mDNS query.
- Some popular implementations of mDNS include the Apple Bonjour service and the networked printer discovery service in Windows 10 operating system from Microsoft. The main drawback of mDNS is its hostname resolution reach to a top-level domain only.



Universal Plug and Play (UPnP) I

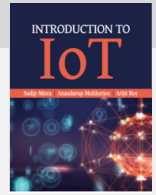
- Universal Plug and Play or UPnP encompasses a set of networking protocols aimed at service discovery and the establishment of functional network-based data sharing and communication services.
- Summarily, it is mainly used for enabling dynamic connections of devices to computing platforms.
- This paradigm is termed plug and play as the devices attaching to a computer network can configure themselves and update their hosts about their working configurations over a network.
- The UPnP is backed by a forum of many consumer electronics vendors and industries and is managed by the Open Connectivity Foundation.



Universal Plug and Play (UPnP) II

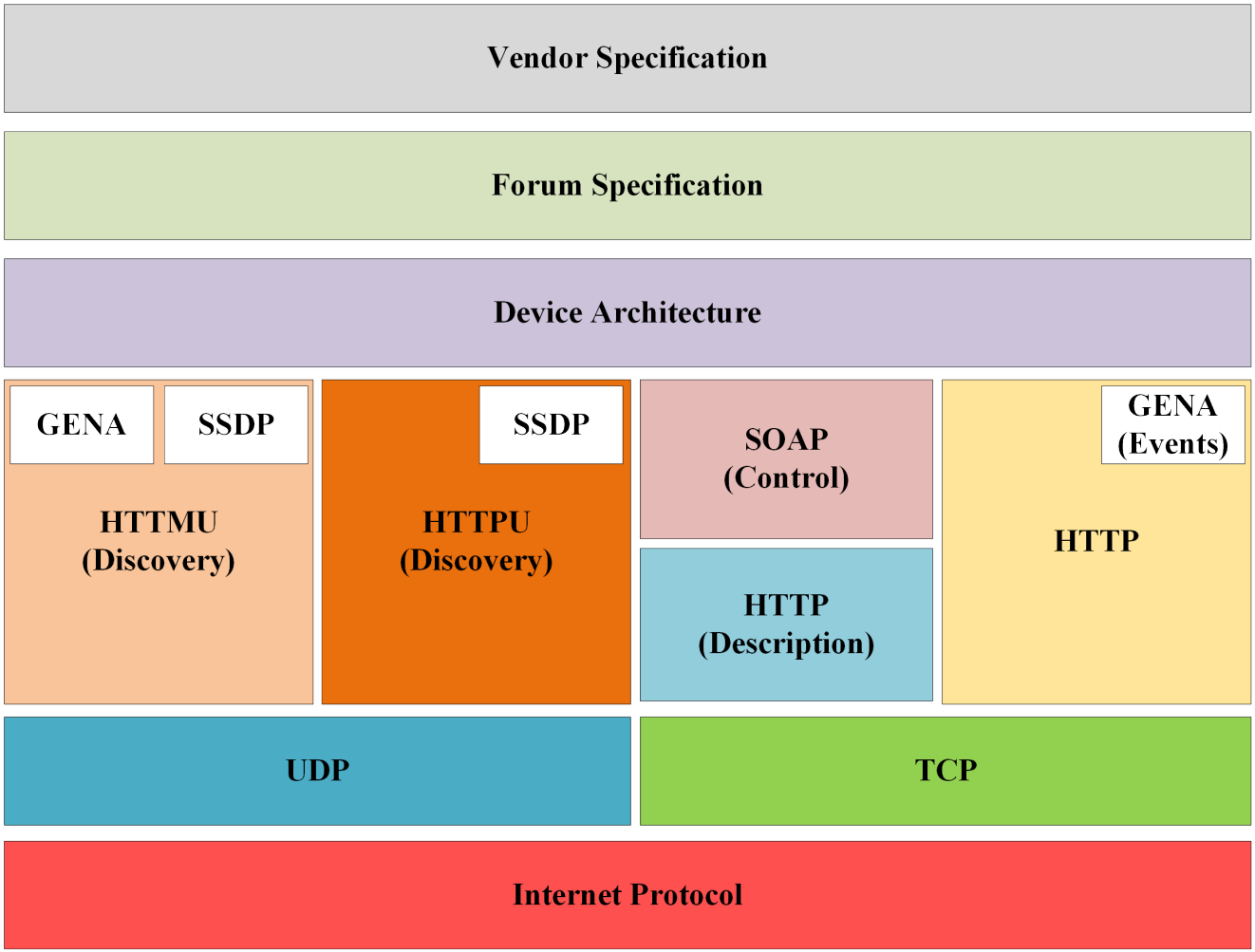
- As UPnP is primarily designed for non-enterprise devices, its scope includes the discovery and intercommunication between networked devices such as mobiles, printers, access points, gateways, televisions, and other regular commercial systems enabled with IP capabilities.
- The present-day UPnP has been designed to run on IP enabled networks, and makes use of networking services of HTTP, XML, and SOAP for data transfer, device descriptions, and event generation and monitoring.
- UPnP enables UDP-based HTTP device search requests and advertisements using multicasting. The responses to device requests are necessarily unicast.

Universal Plug and Play (UPnP) III

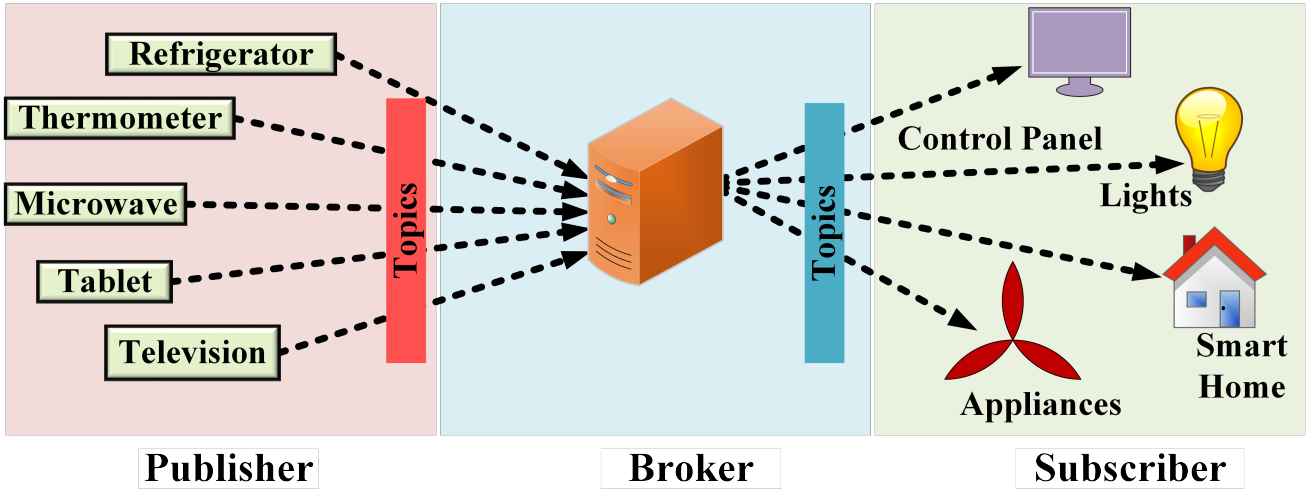


- UPnP advertisements use UDP port 1900 for multicasting. The unnecessary overheads and traffic generated by UPnP systems and its multicast behavior make them unsuitable for enterprise systems.
- The main reason for this is because, on a large scale, the cost of this solution would be infeasible from an operational point of view.

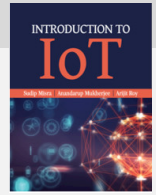
Universal Plug and Play (UPnP) IV



MQTT I

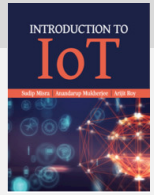


- Message Queue Telemetry Transport or MQTT is a simple, lightweight publish-subscribe protocol, designed mainly for messaging in constrained devices and networks.
- It provides a one-to-many distribution of messages and is payload-content agnostic.



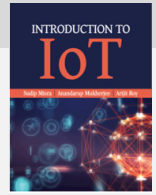
MQTT II

- MQTT works reliably and flawlessly over high latency and limited bandwidth of unreliable networks without the need for significant device resources and device power.
- The MQTT paradigm consists of numerous clients connecting to a server – referred to as a *broker*.
- The clients can have the roles of information publishers (sending messages to the broker) or information subscribers (retrieving messages from the broker).
- This allows MQTT to be largely decoupled from the applications being used with MQTT.
- MQTT is built upon the principles of hierarchical topics and works on TCP for communication over the network.
- Brokers receive new messages in the form of topics from publishers.



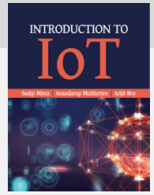
MQTT III

- A publisher first sends a control message along with the data message. Once updated in the broker, the broker distributes this topic's content to all the subscribers of that topic for which the new message has arrived.
- This paradigm enables the publishers and subscribers to be free from any considerations of the address and ports of multiple destinations/subscribers or network considerations of the subscribers, and vice-versa.
- In the absence of any subscribers of a topic, a broker normally discards messages received for that topic unless specified by the publisher otherwise.



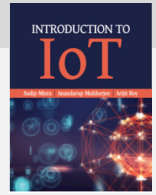
MQTT IV

- This feature removes data redundancies and ensures that maximally updated information is provided to the subscribers. This also reduces the requirements of storage at the broker.
- The publishers can set up default messages for subscribers in agreement with the broker if the publisher's connection is abruptly broken with the broker.
- This arrangement is referred to as *Last will and testament* feature of MQTT. Multiple brokers can communicate in order to connect to a subscriber's topic if it is not present directly with the subscriber's primary broker.
- MQTT control message sizes can range between 2 bytes to 256 megabytes of data, with a fixed header size of 2 bytes. This enables the MQTT to reduce network traffic significantly.



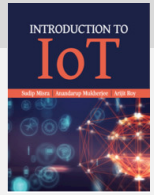
MQTT V

- The connection credentials in MQTT are unencrypted and often sent as plain text. The responsibility of protection of the connection lies with the underlying TCP layer.
- The MQTT protocol provides support for fourteen different message types, which range from connect/disconnect operations to acknowledgments of data. The following are the standard MQTT message types:
  1. CONNECT: Publisher/subscriber request to connect to the broker.
  2. CONNACK: Acknowledgement after the successful connection between publisher/subscriber and broker.
  3. PUBLISH: message publish between a publisher to a broker or a broker to a subscriber.
  4. PUBACK: Acknowledgement of the successful publishing operation.
  5. PUBREC: Assured delivery component message upon successfully receiving publish received.



MQTT VI

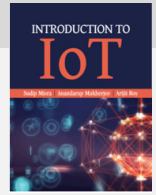
6. PUBREL: Assured delivery component message upon successfully receiving publish release signal.
7. PUBCOMP: Assured delivery component message upon successfully receiving publish completion.
8. SUBSCRIBE: Subscription request to a broker from a subscriber.
9. SUBACK: Acknowledgement of successful subscribe operation.
10. UNSUBSCRIBE: Request for unsubscribing from a topic.
11. UNSUBACK: Acknowledgement of successful unsubscribe operation.
12. PINGREQ: Ping request message.
13. PINGRESP: Ping response message.
14. DISCONNECT: Message for publisher/subscriber disconnecting from the broker.



MQTT VII

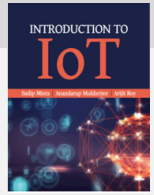
- The MQTT protocol provides support for fourteen different message types, which range from connect/disconnect operations to acknowledgments of data. The following are the standard MQTT message types:
  1. CONNECT: Publisher/subscriber request to connect to the broker.
  2. CONNACK: Acknowledgement after the successful connection between publisher/subscriber and broker.
  3. PUBLISH: message publish between a publisher to a broker or a broker to a subscriber.
  4. PUBACK: Acknowledgement of the successful publishing operation.
  5. PUBREC: Assured delivery component message upon successfully receiving publish received.
  6. PUBREL: Assured delivery component message upon successfully receiving publish release signal.

MQTT VIII



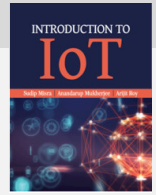
7. PUBCOMP: Assured delivery component message upon successfully receiving publish completion.
8. SUBSCRIBE: Subscription request to a broker from a subscriber.
9. SUBACK: Acknowledgement of successful subscribe operation.
10. UNSUBSCRIBE: Request for unsubscribing from a topic.
11. UNSUBACK: Acknowledgement of successful unsubscribe operation.
12. PINGREQ: Ping request message.
13. PINGRESP: Ping response message.
14. DISCONNECT: Message for publisher/subscriber disconnecting from the broker.

- MQTT's features and content delivery mechanisms were primarily designed for message transmission over constrained networks and through constrained devices. However, MQTT supports three QoS features:



MQTT IX

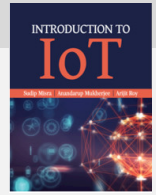
- At most once: This is a best-effort delivery service and is largely dependent on the best delivery efforts of the TCP/IP network on which the MQTT is supported. This may result in message duplication or loss.
- At least once: This delivery service guarantees assured delivery of the messages. However, message redundancy through duplication is a possibility.
- Exactly once: This delivery service guarantees assured message delivery. Additionally, this service also prevents message duplication.



MQTT-SN I

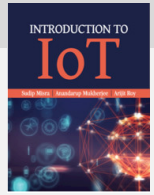
The primary MQTT protocols heavily inspire MQTT for Sensor Networks or MQTT-SN, however is robust enough to handle the requirements and challenges of wireless communications networks in sensor networks [**?**]. Typical features of MQTT-SN include low bandwidth usage, operating under high link failure conditions, and is suitable for low-power, low-cost constrained nodes and networks. The major differences between the original MQTT and MQTT-SN include:

- The CONNECT message types are broken into three messages in which two are optional and are tasked with the communication of the testament message and testament topic to the broker.



MQTT-SN II

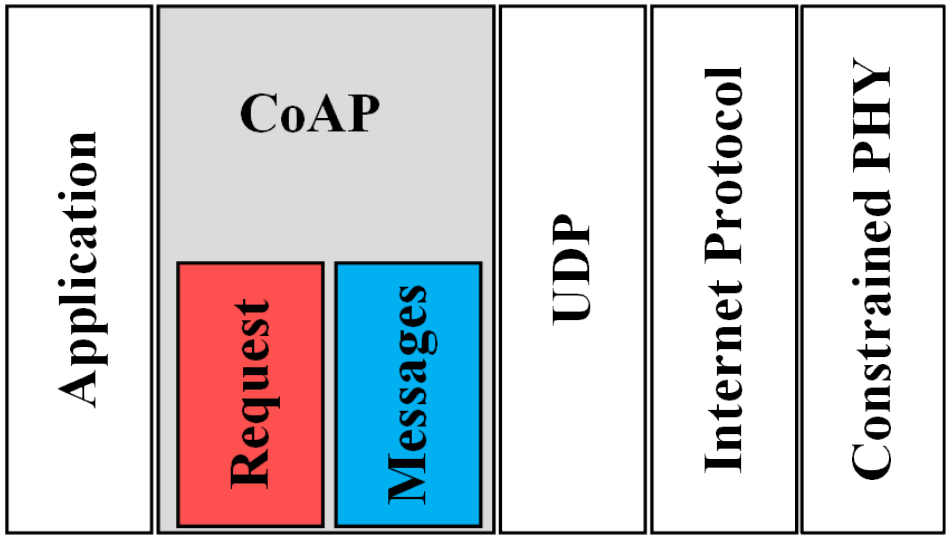
- The topic name in the PUBLISH messages are replaced by topic identifiers, which are only 2 bytes long. This reduces the traffic generated from this protocol and enables the protocol to operate over bandwidth-constrained networks.
- A separate mechanism is present for topic name registration with the broker in MQTT-SN. After a topic identifier is generated for the topic name, the identifier is informed to the publisher/subscribers. This mechanism also supports the reverse pathway.
- In special cases in MQTT-SN, pre-defined topic identifiers are present, which need no registration mechanism. The mapping of topic names and identifiers are known in advance to the broker as well as the publishers/subscribers.



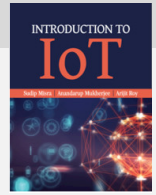
MQTT-SN III

- The presence of a special discovery process is used to link the publisher/subscriber to the operational broker's network address in the absence of a preconfigured broker address.
- The subscriptions to a topic, Will topic, and Will message are persistent in MQTT-SN. The publishers/subscribers can modify their Will messages during a session.
- Sleeping publishers/subscribers are supported by a keep-alive procedure, which is offline, and which helps buffer the messages intended for them in the broker until they wake up. This feature of MQTT-SN is not present in regular MQTT.

CoAP I

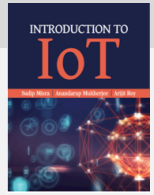


- The Constrained Application Protocol or CoAP as it is more popularly known as is designed for use as web transfer protocol in constrained devices and networks, which are typically low-power and lossy.



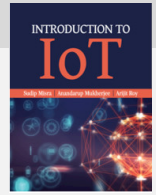
CoAP II

- The constrained devices typically have minimal RAM and an 8-bit processor at most. CoAP can efficiently work on such devices, even when these devices are connected to highly lossy networks with high packet loss, high error rates, and bandwidth in the range of kilobits.
- CoAP follows a request-response paradigm for communication over these lossy networks.
- Additional highlights of this protocol include the support for service discovery, resource discovery, URIs, Internet media handling support, easy HTTP integration, and multicasting support – that too while maintaining low overheads.
- Typically, CoAP implementations can act as both clients and servers (not simultaneously).



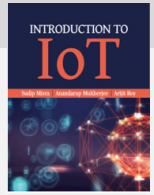
CoAP III

- A CoAP client's request signifies a request for action from an identified resource on a server, which is similar to HTTP.
- The response sent by the server in the form of a response code can contain resource representations, as well. However, CoAP interchanges are asynchronous and datagram-oriented over UDP.
- Packet traffic collisions are handled by a logical message layer incorporating the exponential back-off mechanism for providing reliability.
- The reliability feature of CoAP is optional. The two seemingly distinct layers of messaging (which handle the UDP and asynchronous messaging) and request-response (which handles the connection establishment) are part of the CoAP header.
- The CoAP is characterized by the following main features:



CoAP IV

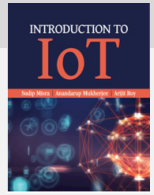
1. Suitable web protocol for integrating IoT and M2M services in constrained environments with the Internet.
2. Enables UDP binding and provides reliability concerning unicast as well as multicast requests.
3. Message exchanges between end-points in the network, or between nodes is asynchronous.
4. The limited packet header incurs significantly lower overheads. This also results in less complexity and processing requirements for parsing of packets.
5. It has provisions for URI and other content-type identifier support. CoAP additionally provides DTLS binding.
6. It has a straightforward proxy mechanism and caching capabilities, which is responsible for overcoming the effects of the lossy network without putting extra constraints on the low-power devices. The caching is based on the concept of the maximum age of packets.



CoAP V

7. It provides a stateless mapping with HTTP. The server or receiving node does not retain information about the source of the message, and rather it is expected from the message packet to carry that information with it. This enables CoAP's easy and uniform integration with HTTP.

- CoAP defines four messaging types – 1) Confirmable (CON), 2) Non-confirmable (NON), 3) Acknowledgement (ACK), and 4) Reset. The method codes and the response codes are included in the messages being carried.
- These codes determine whether the message is a request message or a response message. Requests are typically carried in confirmable and non-confirmable message types.



CoAP VI

- However, responses are carried in both of these message types, as well as with the acknowledgment message. The transmission of responses with acknowledgment messages is known as piggybacking and is quite synonymous with CoAP.

CoAP VII

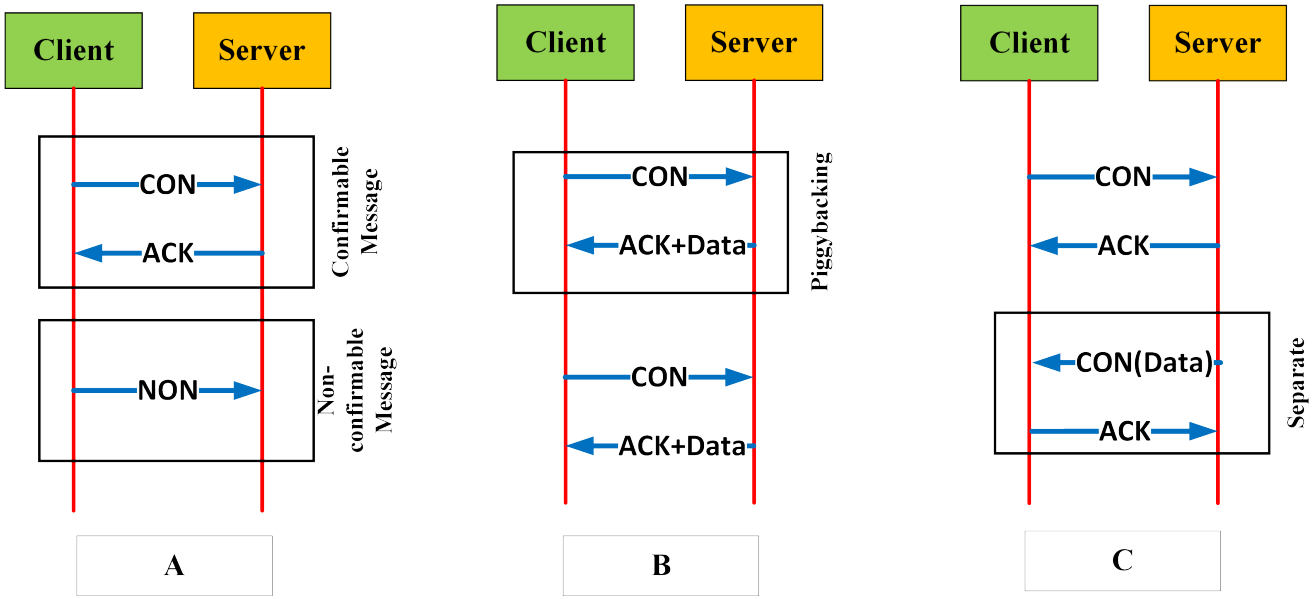
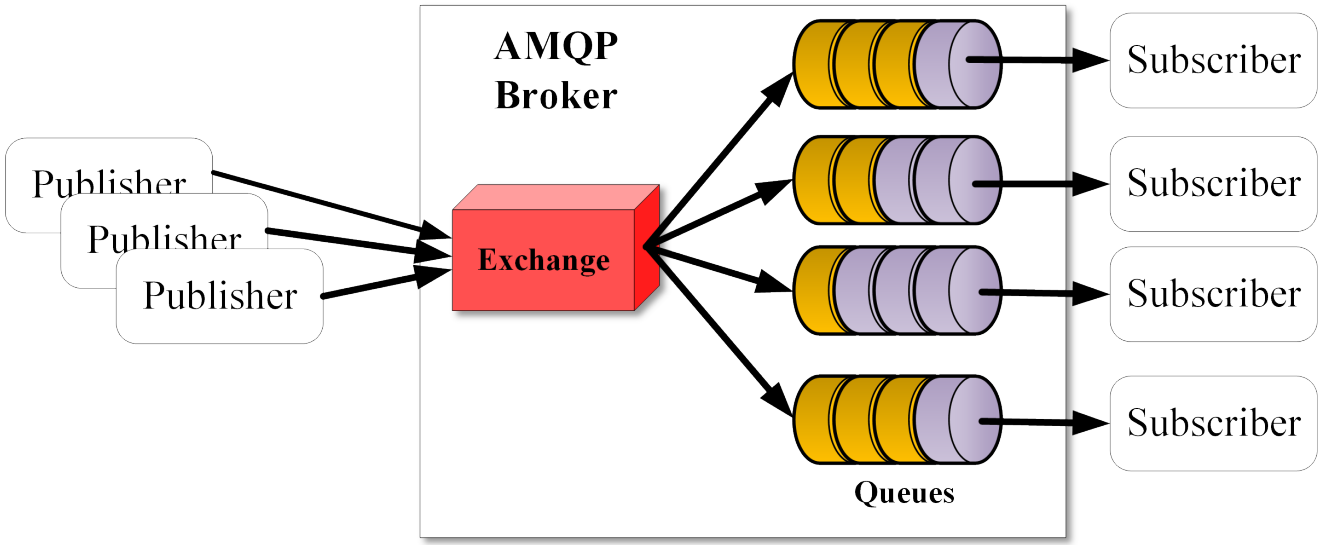
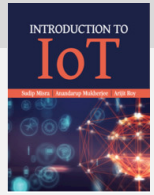


Figure: Various CoAP response-response models. (A): CON and NON messages, (B): Piggyback messages, and (C): Separate messages

AMQP I

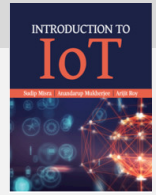


- AMQP or the Advanced Message Queuing Protocol is an open standard middleware at the application layer developed for message-oriented operations.
- AMQP tries to bring across the concept of interoperability between the clients and the server by enabling cross-vendor implementations.



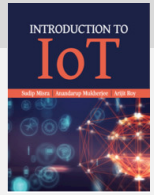
AMQP II

- An AMQP broker is tasked with maintaining message queues between various subscribers and publishers.
- AMQP is armed with the features of message orientation, queuing, reliability, security, and routing. Both request-response and publish-subscribe methods are supported.
- AMQP is considered as a wire-level protocol. Here, the data format description is released on the network as a stream of bytes. This description allows AMQP to connect to anyone who can interpret and create messages in the same format. This results in a level of interoperability where anyone with the compliant or supporting means can make use of this protocol without any need for a specific programming language.



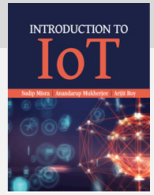
AMQP III

- AMQP is built for the underlying TCP and is designed to support a variety of messaging applications efficiently.
- It provides a wide variety of features such as flow-controlled communication, message-oriented communication, message delivery guarantees (at-most-once, at least once, and exactly once), authentication support, and an optional SSL or TLS-based encryption support.
- The AMQP is specified across four layers – 1) type system, 2) process to process asynchronous and symmetric message transfer protocol, 3) extensible message format, and 4) set of extensible messaging capabilities.



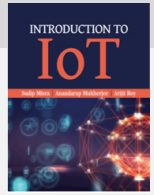
AMQP IV

- The primary unit of data in AMQP is referred to as a frame. These frames are responsible for the initiation of connections, termination of connections, and control of messages between two peers using AMQP.
- There are nine frame types in AMQP:
  1. Open: responsible for opening the connection between peers.
  2. Begin: responsible for setup and control of messaging session between the peers.
  3. Attach: responsible for link attachment.
  4. Transfer: responsible for message transfer over the link.
  5. Flow: responsible for updating the flow control state.
  6. Disposition: responsible for updating of transfer state.
  7. Detach: responsible for detachment of link between two peers.
  8. End: responsible for truncation of a session.
  9. Close: responsible for closing/ending a connection.



XMPP I

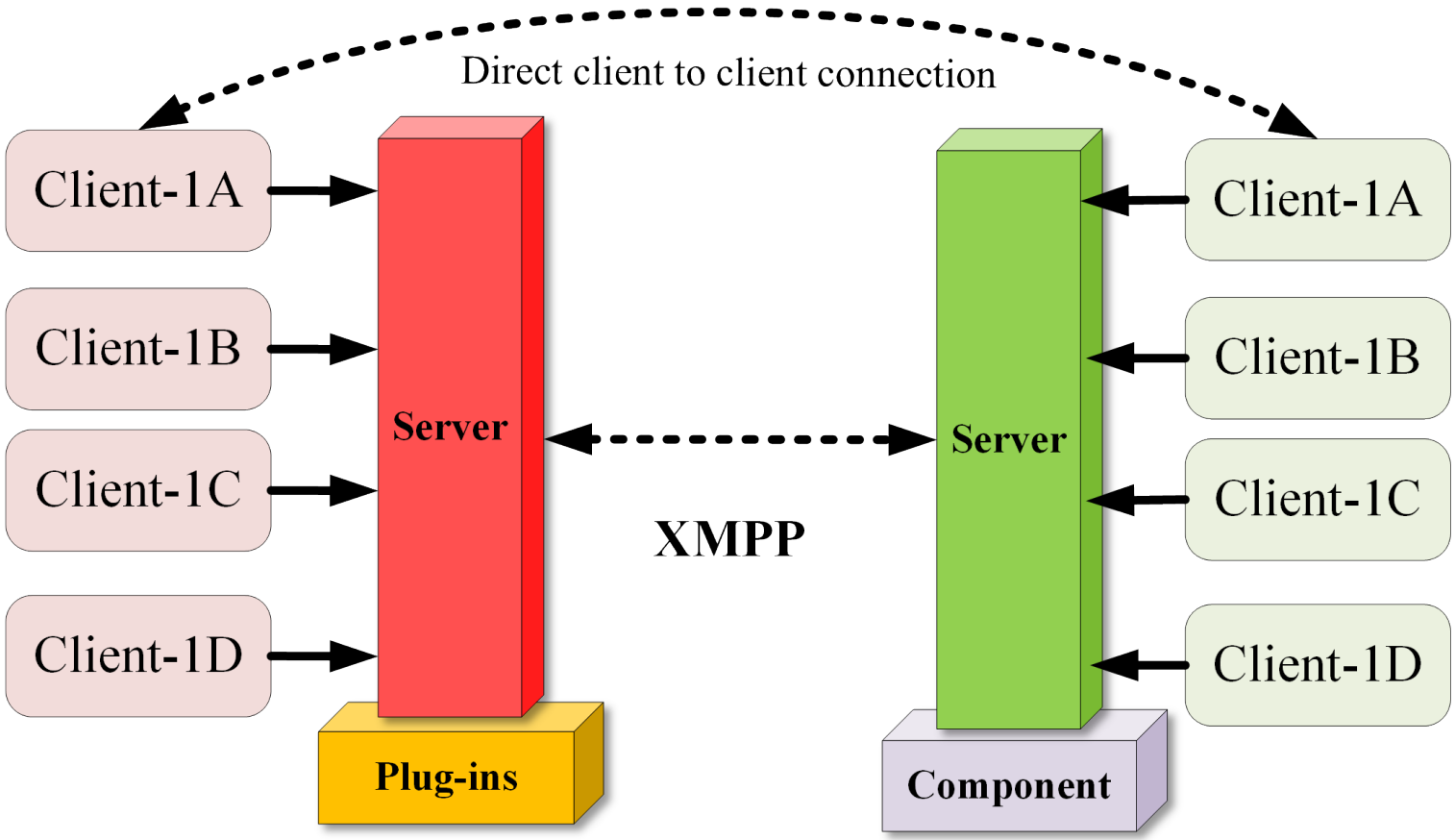
- The Extensible Messaging and Presence Protocol, or XMPP, which was initially named as Jabber, is designed for message-oriented middlewares based on the Extensible Markup Language (XML).
- XMPP was developed for instant messaging, maintenance of contacts, and network presence information.
- Structured and extensible data between two networked nodes/devices can be exchanged in near real-time using this protocol.
- XMPP has found use in VOIP presence signaling, video and file transfers, smart grid, social networks, publish-subscribe systems, IoT applications, and others.
- XMPP, being open-source, has enabled the spurt of developments in various freeware as well as commercial messaging software.



XMPP II

- As XMPP follows a client-server architecture, peers in a network cannot talk directly to one another through XMPP.
- All communication between peers has to be routed through an XMPP server. The XMPP model is considered to be decentralized as anyone can host an XMPP server to which various clients can subscribe.

XMPP III


Direct client to client connection
Client-1A
Client-1B
Client-1C
Client-1D
Server
Plug-ins
XMPP
Server
Component
Client-1A
Client-1B
Client-1C
Client-1D

XMPP is an extensible, flexible, and diverse protocol, and has resulted in the development of a significant number of technologies based on it. Some key XMPP technologies include:

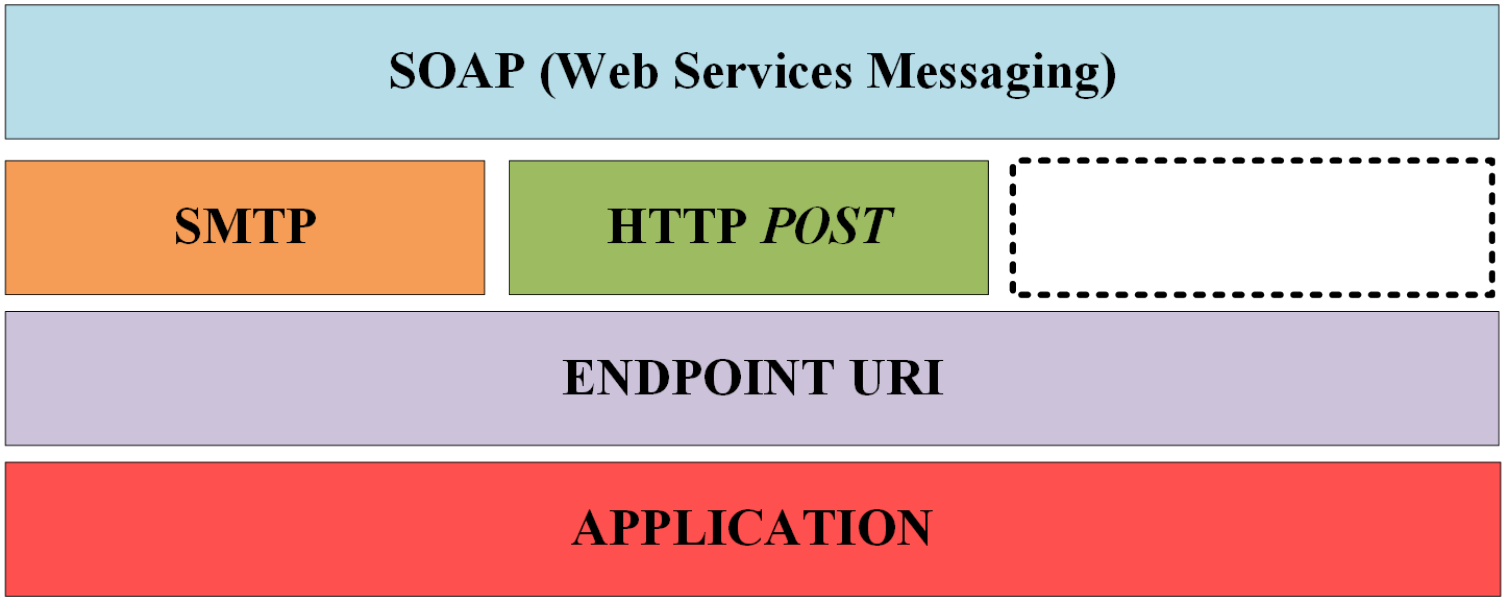
- **Core**: it deals with information about the core XMPP technologies for XML streaming over a network. The core includes the base XML layer for streaming, provides TLS-based encryption, provides Simple Authentication and Security Layer (SASL) based authentication, informs about the availability of a network, provides UTF-8 support, and contact lists, which are presence enabled.
- **Jingle**: provides session initiation protocol (SIP)-compatible multimedia signaling for voice, video, file transfer, and other applications. Various media transfer protocols such as TCP, UDP, RTP, or even in-band XMPP is supported. The Jingle session initiation signal is sent over XMPP, and the media transfer takes place in a peer-to-peer manner or over media relays.
- **Multi-user chat**: or MUC is a flexible, multi-party communication exchange extension for XMPP. Here multiple users can exchange information in a chat room or channel. The support for strong

chat-room controls is also provided, which enables the banning of users and updation of chat room moderators.

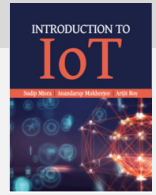
- **Pubsub**: provides publish-subscribe functionality to XMPP by proving alerts and notifications for data syndication, vibrant presence, and more such features. It enables XMPP clients to create topics at a pub-sub service and publish/subscribe to them.

- **BOSH** — it stands for Bidirectional-streams Over Synchronous HTTP. This is an HTTP binding for XMPP (and other) traffic. BOSH incurs lower latencies and lesser network bandwidth usage by doing away with HTTP polling. It is mainly used for the XMPP traffic exchange between clients and servers.

SOAP I

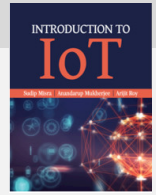


- SOAP or Simple Object Access Protocol is used for exchanging structured information in web services by making use of XML information set formatting over application layer protocol (HTTP, SMTP) based transmission and negotiation of messages.



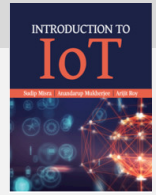
SOAP II

- This allows SOAP to communicate with two or more systems with different operating systems using XML, making it language and platform-independent.
- The use of SOAP facilitates the messaging layer of the web services protocol stack.
- A SOAP application can send a request with the requisite search parameters to a server with web services enabled.
- The target server responds in a SOAP response format with the results of the search. The response from the server can be directly integrated with applications at the requester's end, as it is already in a structured and parsable format.



SOAP III

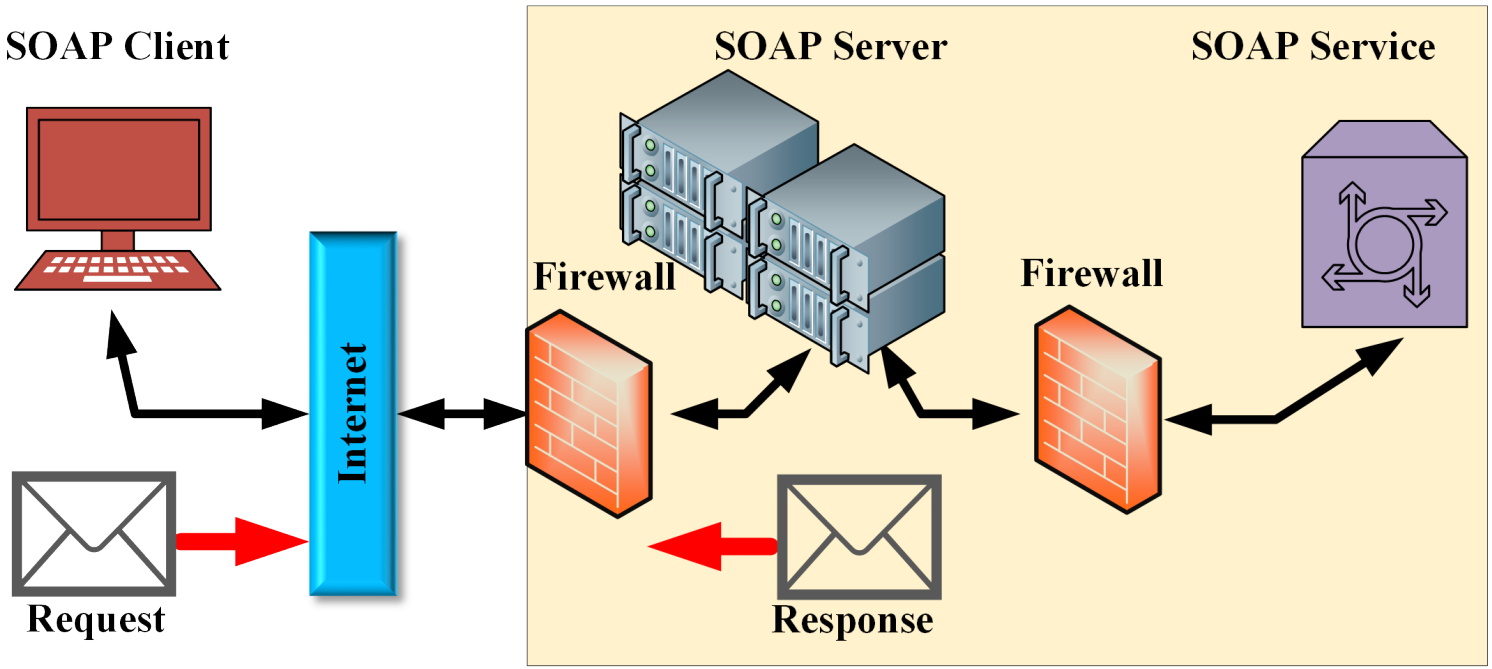
- SOAP is made up of three broad components – 1) envelope (which defines the structure of the message and its processing instructions), 2) encoding rules (which deal with handling of various datatypes arising out of the numerous applications), and 3) convention (which is responsible for web procedure calls and their responses).
- This messaging protocol extends the features of neutrality (can operate over any application layer protocol), independence (independent of programming models), and extensibility (features such as security and web service addressing can be extended) to its services.
- The use of SOAP with HTTP-based request-response exchanges does not require the modification of the communication and processing infrastructures.



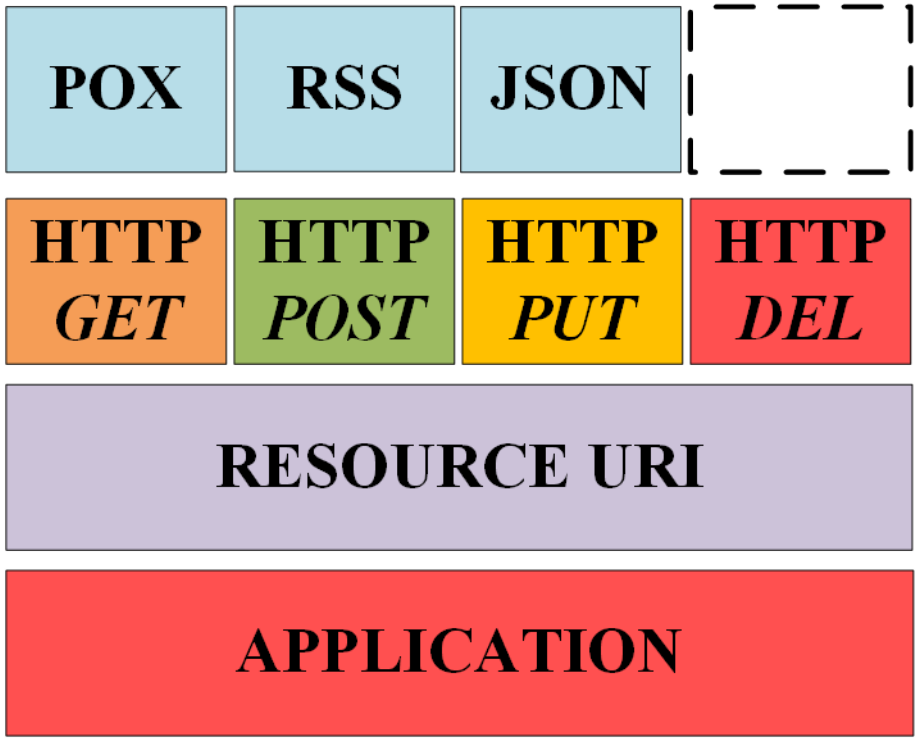
SOAP IV

- It can easily pass through network/system firewalls and proxies (similar to tunneling).
- However, the use of XML affects the parsing speed and hence, the performance of this protocol.
- Additionally, the verbose nature of SOAP is not recommended for use everywhere. The specifications of the SOAP architecture are defined across several layers, such as – message format layer, message exchange patterns (MEP) layer, transport protocol binding layer, message processing model layer, and protocol extensibility layer.

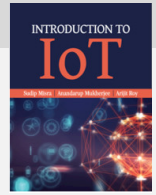
SOAP V


SOAP Client
SOAP Server
SOAP Service
Firewall
Firewall
Internet
Request
Response

REST I

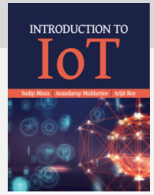


- Representational State Transfer or REST encompassed a set of constraints for the creation of web services, mainly using a software architectural style.



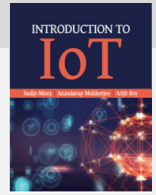
REST II

- The web services adhering yo REST styles are referred to as RESTful services and enable interoperability between various Internet-connected devices.
- RESTful systems are stateless – the web services on the server do not retain client states. The use of stateless protocols and standards makes RESTful systems quite fast, reliable, and scalable.
- The reuse of components can be easily managed without hindering the regular operations of the system as a whole.
- Requesting systems can manipulate textual web resource representations by making use of this stateless behavior of REST.
- RESTful web services, in response to a request made to a resource's URI, mainly responds with either an HTML, XML, or JSON formatted payload.



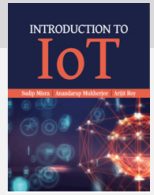
REST III

- As RESTful services use HTTP for transfer over the network, the following four methods are commonly used – 1) GET (read-only access to a resource), 2) POST (for creating a new resource), 3) DELETE (used for removing a resource), and 4) PUT (used for updating an existing resource or creating a new one).
- REST offers several advantages over regular web-based services. Enhanced network efficiency through the use of REST is ensured by an increase in the performance of interaction between components.
- Its use also enables a uniform and simple interface, easy live operational modification capabilities, reliability against component and data failures, portability of components, robust scalability, and support for a large number of components.



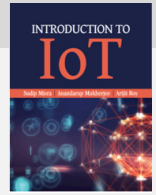
REST IV

- In REST, the requests are used for identifying individual resources. As the resources can be represented in a variety of formats such as HTML, XML, JSON, and others, RESTful services can identify the individual resources from their representations, which allows them to modify, update or delete these resources.
- The REST messages contain sufficient information in them to direct a parser on how to interpret the messages.
- It is expected from REST client to dynamically discover all web resources and actions associated with an initial URI.
- This enhances the dynamicity of applications using REST by avoiding the need to hard-code all clients with the information of the proper structure or dynamics of the web application.



REST V

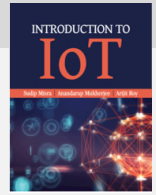
- RESTful systems are guided by six general constraints, which define and restrict the process of client-server interactions and requests-responses. These guidelines increase system performance, scalability, reliability, modifiability, portability, and visibility.
- All RESTful systems have to adhere to these six guidelines strictly:
  1. **Statelessness**: The statelessness of the client-server communication prevents the storage of any contextual information of the client on the server. Each client request has to be self-sufficient in informing its responders about its services and session state. This is done by including the possible links for new state transitions within the representation of each application state. Generally, upon detecting pending requests, the server infers that the client is in a state of transition.



REST VI

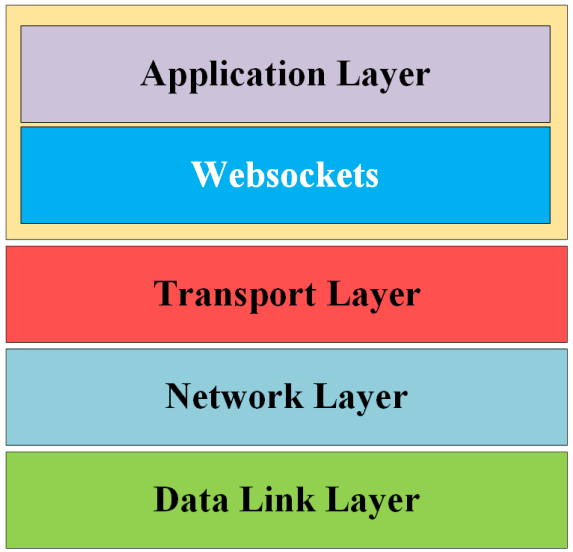
2. **Uniform interface**: Each part or component of a RESTful system must evolve independently as a result of the decoupling of architectures and its simplification.
3. **Cacheability**: The responses have to be implicitly, or in cases, explicitly clear on whether they have to be cached or not. This helps the clients in retaining the most updated data in response to requests. Caching also reduces the number of client-server interactions, thereby improving the performance and scalability of the system as a whole.
4. **Client-server architecture**: The user-interface interactions should be separate from data storage ones. This would result in enhanced portability of user interfaces across multiple platforms. This separation also allows for the independent evolution of components, which would result in scalability over the Internet across various organizational domains.

REST VII

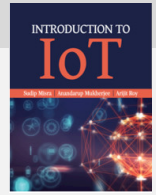


5. **Layered system**: The client in RESTful services is oblivious to the nature of the server to which it is connected – an end-point server or an intermediary server. The use of intermediaries also helps in improving the balancing of load, enhancing security measures, and system scalability.
6. **Code on demand**: This is an optional constraint. Here, the functionality of clients can be extended for a short period by the server. For example, the transfer of executable codes from compiled components.

Websocket I

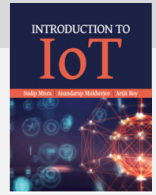


- WebSocket is an IETF-standardized full-duplex communication protocol.
- WebSockets (WS) – an OSI layer seven protocol – enable reliable and full-duplex communication channels over a single TCP connection.
- The WS relies on the OSI layer 4 TCP protocol for communication.



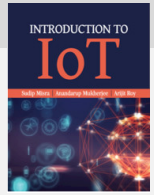
Websocket II

- Despite being different from the HTTP protocol, WS is compatible with HTTP and can work over HTTP ports 80 and 443, enabling support for network mechanisms such as the HTTP proxy, which is usually present during organizational Internet accesses through firewalls.
- WS enables client-server interactions over the Web.
- Web servers and clients such as browsers can transfer real-time data between them without incurring many overheads.
- Upon establishment of a connection, servers can send content to clients without the clients requesting them first.
- Messages are exchanged over the established connection, which is kept open, in a standardized format.



Websocket III

- Support for WS is present in almost all modern-day browsers; however, the server must also include WS support for the communication to happen.
- The full-duplex communication provided by WS is absent in protocols such as HTTP. Additionally, the use of TCP (which supports byte stream transfers) is also enhanced by enabling it to provide message stream transfers using WS.
- Before the emergence of WS, Comet channels were used for attaining full-duplex communication over port 80.
- However, the Comet systems were very complicated and incurred significant overheads, which made their utility insignificant for constrained application scenarios mainly associated with IoT.



Websocket IV

- The WebSocket (WS) and WebSocket Secure (WSS) have been specified as a uniform resource identifier (URI) schemes in the WS specification, which are meant for unencrypted and encrypted connections, respectively.
- The WS handshake process and the frames can be quickly inspected using browser development tools.