EXPERMINT: 02

- **Title**: : CoAP protocol in Contiki OS
- **Aim:** To study and simulate CoAP protocol in Contiki OS.

Theory:

Contiki is an operating system for networked, memory-constrained systems with a focus on low-power wireless Internet of Things (IoT) devices. Contiki OS has three network stacks: IPv4, IPv6, and Rime. The lIP stack is for IPv4 and IPv6 networking. Rime stack is used for low-power wireless networks using a lightweight protocol stack. The major work is related to network layer as the proposed work mainly focuses on RPL design. Constrained Application Protocol (CoAP) is a specialized Internet application protocol for constrained devices, as defined in RFC 7252. It enables those constrained devices called "nodes" to communicate with the wider Internet using similar protocols.

CoAP a customary client-server IoT protocol. It enables clients to make requests for web transfers as per the need of the hour. On the other hand, it also let supporting servers to respond to arriving requests. In summary, devices' nodes in the IoT ecosystem are enabled to interact over through CoAP only. CoAP and HTTP follow the same working procedure. However, CoAP attains its functionality via asynchronous transactions (using UDP). It utilizes the POST, GET, PUT, and DELETE calls. That's the reason why API security is of higher grade while CoAP is active as it is an RPK and PSK-certified protocol.

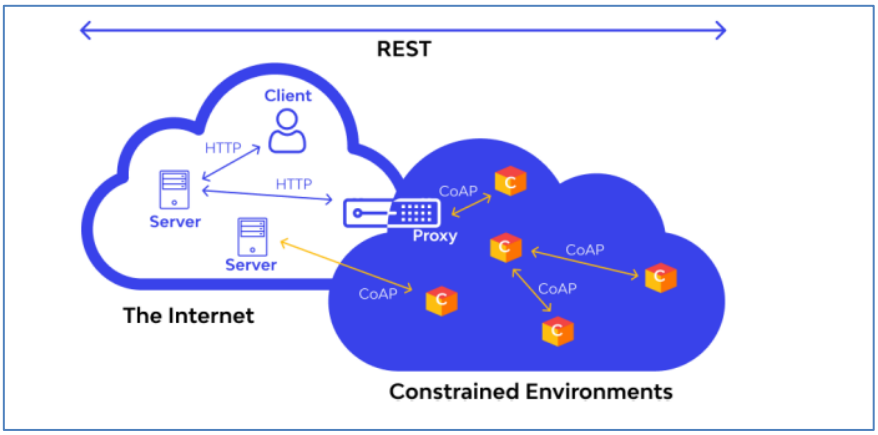
CoAP is compatible with 4 types of information exchange:

- Acknowledgments confirm the completion or failure of an event.
- Confirmable are the messages that are resent on time out until the confirmation of successful sending doesn't arrive.
- Reset messages are empty, with confirmable as their nature.
- Non-confirmable information is just sent and has no guarantee of successful delivery. There is no acknowledgment of success either.

CoAP Architecture:

The WWW and the constraints ecosystem are the 2 foundational elements of the CoAP protocol architecture. Here, the server monitors and helps in communication happening using CoAP and HTTP while proxy devices bridge the existing gap for these 2 ecosystem, making the communication smoother. CoAP allows HTTP clients (also called CoAP clients here) to talk or exchange data/information with each other within resource constraints. While one tries to understand this architecture, gaining acquaintances with some key terms is crucial:

- Endpoints are the nodes that host have knowledge of;
- Client sends requests and replies to incoming requests;
- Server gets and forwards requests. It also gets and forwards the messages received in response to the requests it had processed.
- Sender creates and sends the original message.
- Recipient gets the information sent by the client or forwarded by the server.



●CoAP Function:

The key role of CoAP is to act like HTTP wherever restricted devices are a part of communication. While filling the gap of HTTP, it enables devices like actuators and sensors to interact over the internet. The devices, involved in the process, are administered and controlled by considering data as a system's component. CoAP protocol can operate its functions in an environment having reduced bandwidth and extreme congestion as it consumes reduced power and network bandwidth. Networks featuring intense congestion and constrained connectivity are not ideal conditions for TCP-based protocols to carry out their responsibilities. CoAP comes as a rescuer at this place and supports the wen transfers. Web transfers happening using satellites and covering long distances can be accomplished with full perfection using CoAP. Networks featuring billions of nodes take the help of the CoAP protocol for information exchange. Regardless of the function handled or role played, CoAP promised security of highest grade as DTLS parameters as default security parameter; the counterpart of 128bit RSA keys. Speaking of its deployment, it's simple and hassle-free. It can be implemented from scratch for a straightforward application. For the application ecosystem where CoAP is not desirable, generic implementations are offered for various platforms. Most of the CoAP implementations are done privately while few are published in open-source libraries like MIT license.

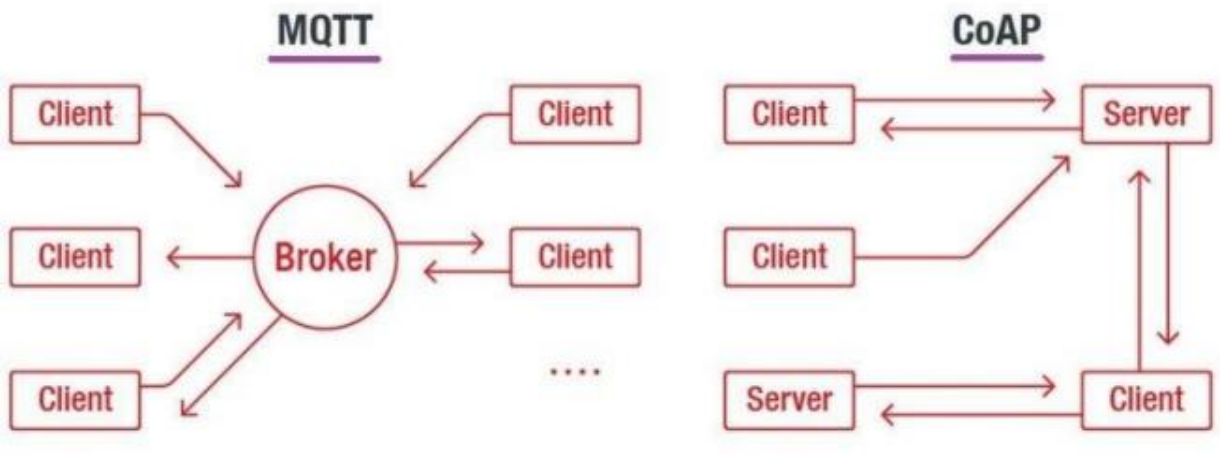
●Key Features of CoAP:

1. RESTful Architecture: CoAP uses a RESTful (Representational State Transfer) architecture. It follows a set of constraints that allow it to operate efficiently over a large, distributed network. In a RESTful system, data and functionality are considered resources, and these resources are accessed using a standard, uniform interface.
2. Built-In Discovery
3. The CoAP protocol's built-in discovery mechanism allows devices to discover resources on other devices without requiring any prior knowledge of their existence. This is especially useful in IoT networks, where devices may be constantly joining and leaving the network.
4. Asynchronous Message Exchanges
5. CoAP supports asynchronous message exchanges, which is crucial for IoT networks where devices may not always be connected or available. With asynchronous message exchanges, a device cansend a request to another device and then continue with other tasks without waiting for a response. The response can be processed once it arrives, even if delayed.
6. Optional Reliability with Confirmable Messages
7. CoAP offers optional reliability through the use of confirmable messages. When a device sends a confirmable message, it expects an acknowledgement from the recipient. If no acknowledgement is received within a certain time, the message is retransmitted.

●Advantages to CoAP:

1. Reduced power requirements
2. It operates over UDP, which requires minimal overhead for communications.
3. It also allows faster wake up times and extended sleepy states.
4. Taken together, this means batteries last longer for IoT devices.

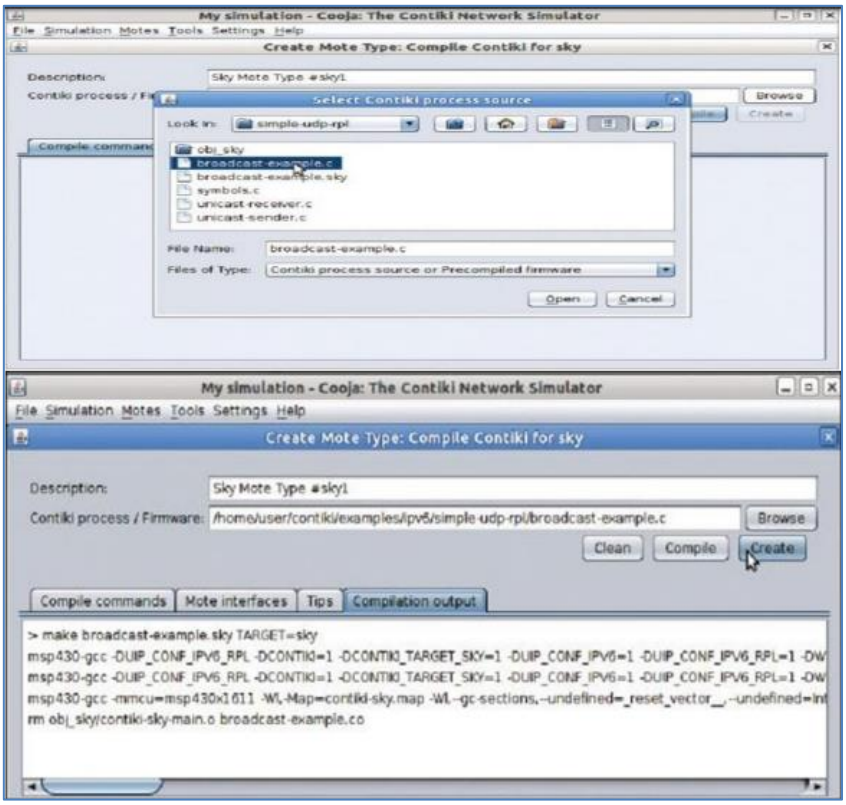
●CoAP vs. MQTT:



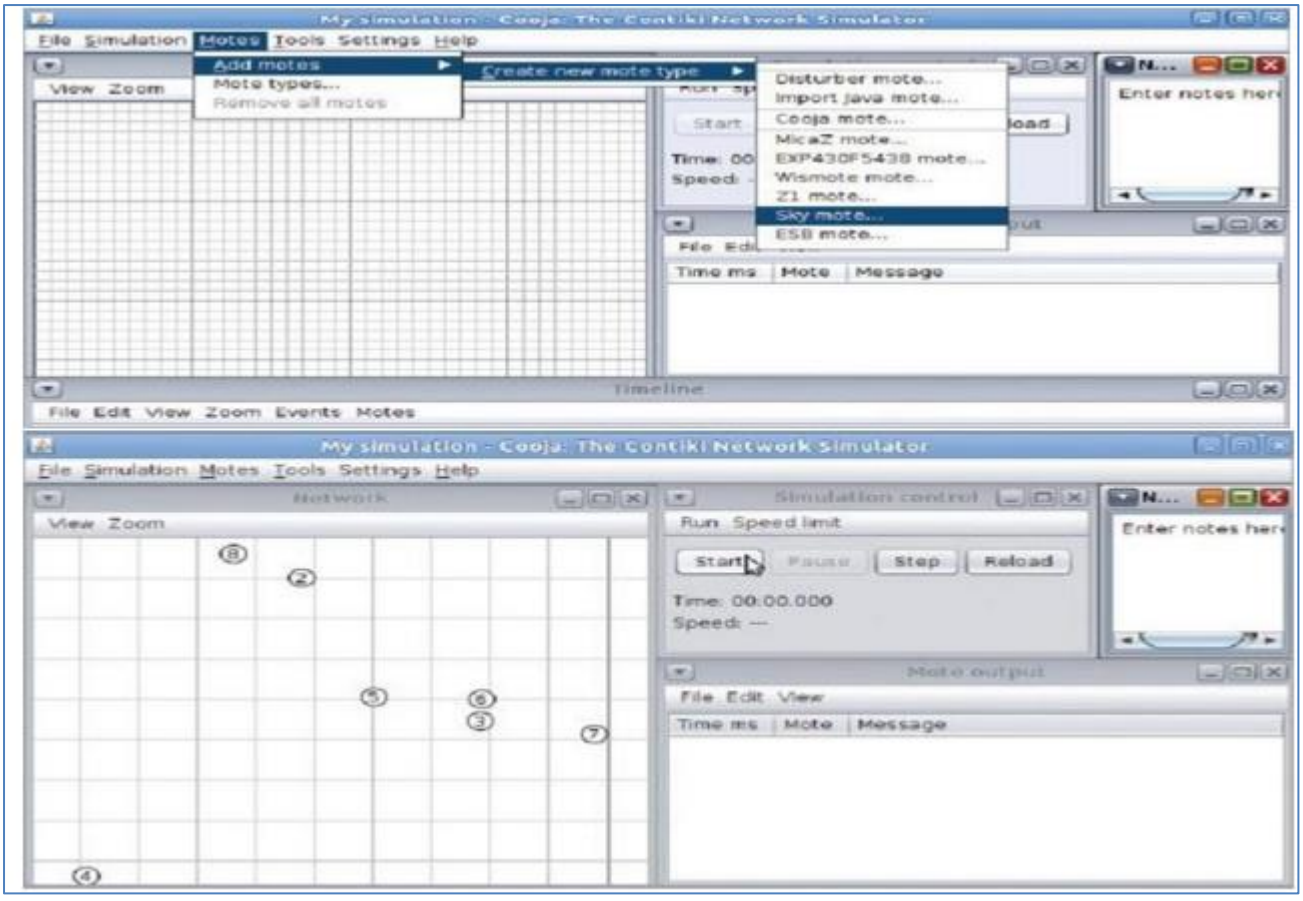
	MQTT	CoAP	AMQP	HTTP
Base protocol	TCP	UDP	TCP	TCP
Paradigm	Publish/Subscribe	Request/Response or Publish/Subscribe	Publish/Subscribe or Request/Response	Request/Response
Header Size	2 Bytes	4 Bytes	8 Bytes	Undefined
Message Size	Small and Undefined (up to 256 MB)	Small and Undefined	Negotiable and Undefined	Large and Undefined
Reliability	QoS 0 - At most once QoS 1 - At least once QoS 2 - Exactly once	CON Message NON Message	Settle Format Unsettle Format	Limited (via TCP)
Standards	OASIS, Eclipse Foundations	IETF, Eclipse Foundation	OASIS, ISO/IEC	IETF and W3C
Licensing	Open Source	Open Source	Open Source	Free

●Procedure:

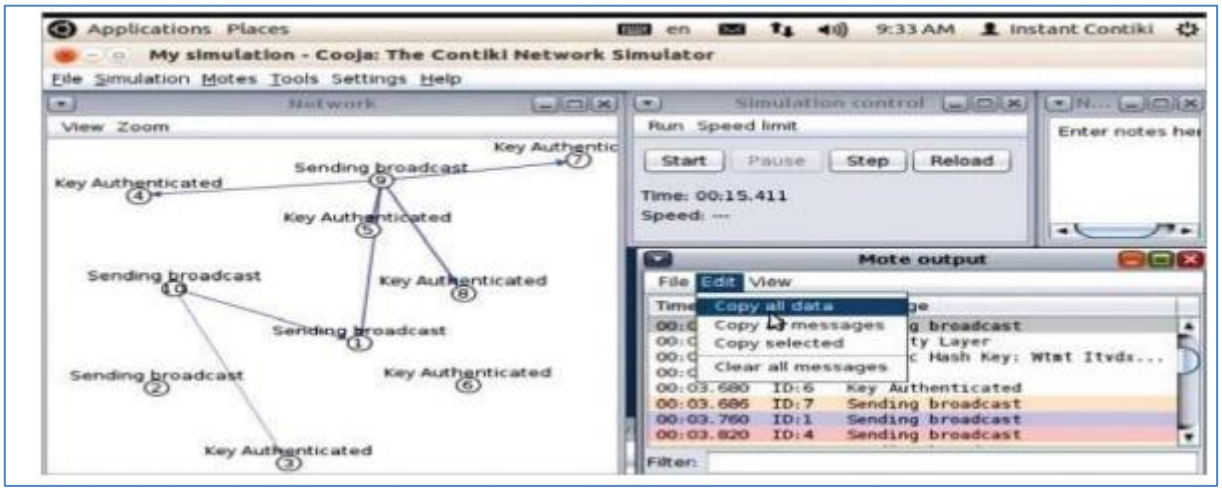
1. Start a new simulation.
2. Open and compile broadcast example from Contiki process source.

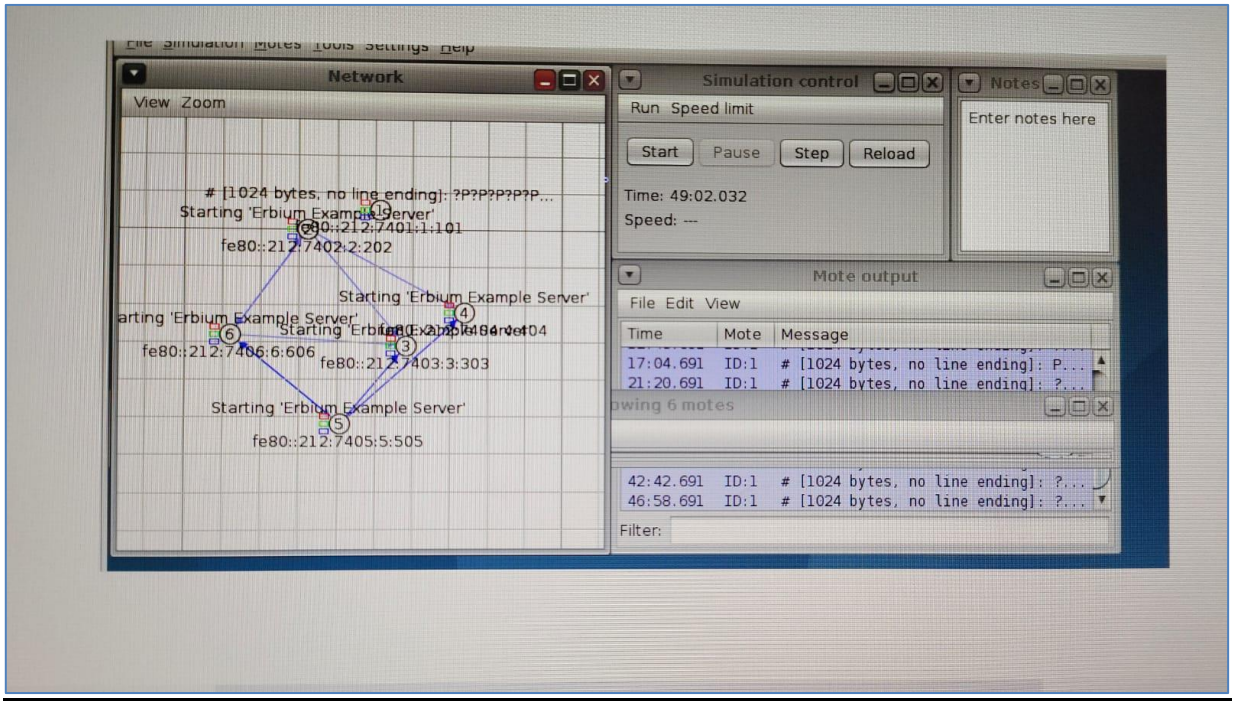


3) Add Motes and make the connection



4) Connection Established





●Result & Conclusion:

The study and simulation of CoAP in Contiki OS contribute to the broader exploration of IoT communication protocols, emphasizing the practical implementation and adaptability of CoAP for resource-constrained devices. This knowledge is valuable for developers and researchers working on IoT applications and protocols.