Institute for the Wireless Internet of Things
at Northeastern University

EECE5155: Wireless Sensor Networks and the Internet of Things

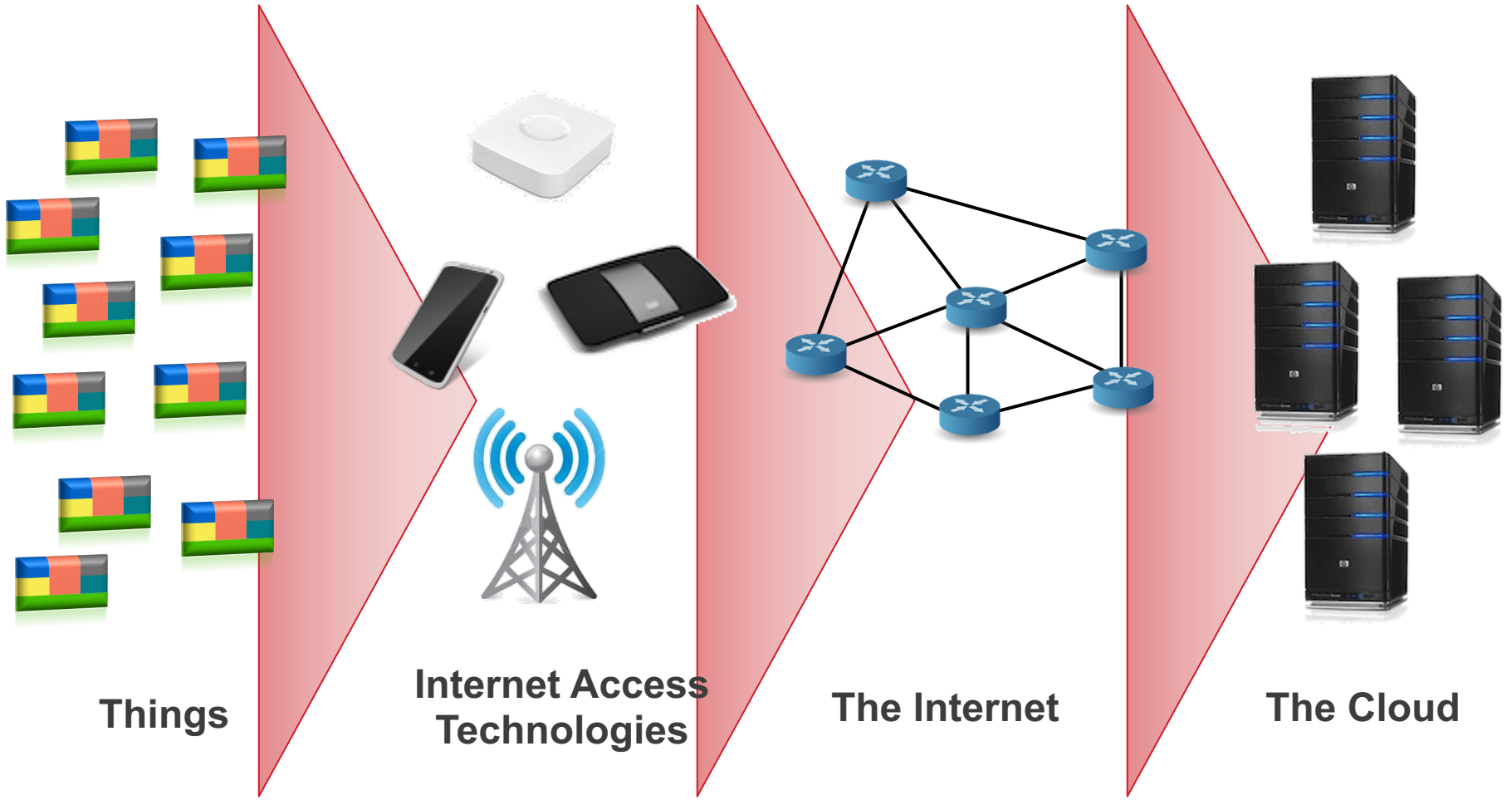
Josep Miquel Jornet
Associate Professor, Department of Electrical and Computer Engineering
Director, Ultrabroadband Nanonetworking Laboratory
Member, Institute for the Wireless Internet of Things
Northeastern University
jmjornet@northeastern.edu
www.unlab.tech

Institute for the Wireless Internet of Things at Northeastern

Module T5: Data Streaming

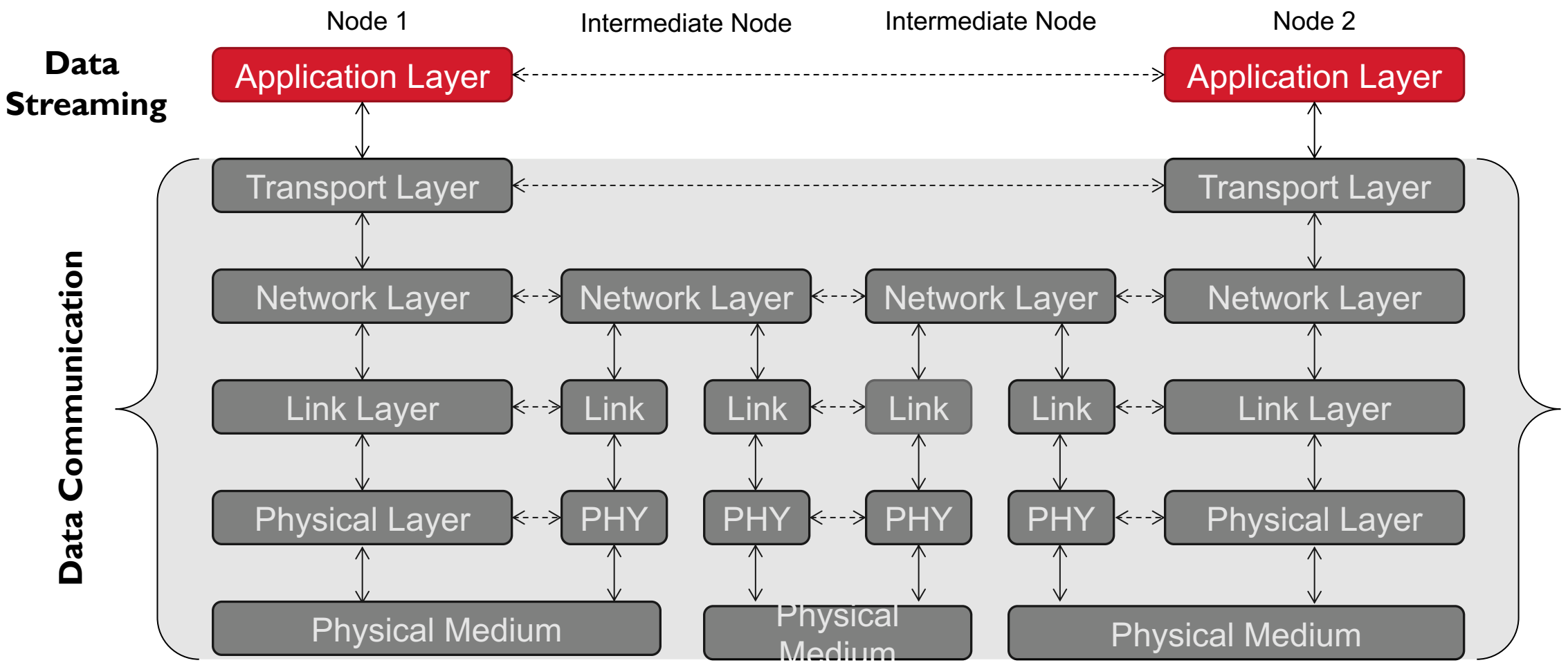
The Internet of Things

Institute for the Wireless Internet of Things at Northeastern



The Protocol Stack

Institute for the Wireless Internet of Things at Northeastern



Data Streaming Protocols

Institute for the Wireless Internet of Things at Northeastern

- At the application layer, we need to effectively transmit data:
  - From nodes to server
  - From server to nodes
  - Between nodes

- Why don't we use:
  - **FTP/SFTP** → Transfer files
    - Too complicated, unlikely needed
  - **POP3/SMTP/IMAP/Exchange** → Send emails
    - Too much burden for some simple data

Data Streaming Protocols

Institute for the Wireless Internet of Things at Northeastern

- MQTT
- CoAP
- HTTP
- XMPP
- AMQP
- …

MQTT

Institute for the Wireless Internet of Things at Northeastern

- A client-server **publish/subscribe** messaging protocol
  - It is lightweight, open, simple, and designed to be easy to implement
- Designed to handle high volumes of data in **low bandwidth networks**
  - Reduce operation cost
- **Small code footprint**
  - Reduce device cost
- Runs on top of **TCP/IP**
  - Some variations like MQTT- SN run over non-TCP/IP networks
- **Examples:**
  - Facebook Messenger
  - Supported by Amazon Web Services (AWS)
  - LoRaWAN gateways
  - Many others

MQTT History

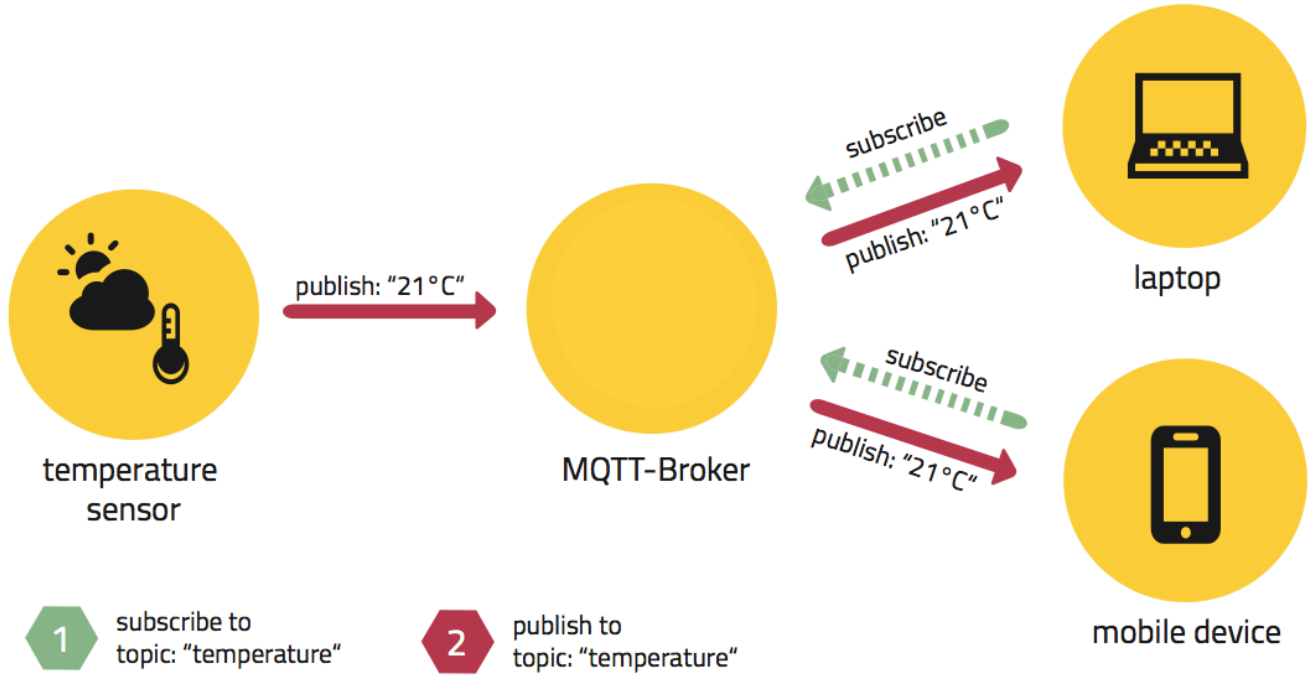
Institute for the Wireless Internet of Things at Northeastern

- Invented in 1999 by Andy Stanford-Clark (IBM) and Arlen Nipper
  - The original problem was to send sensor data from oil pipelines through a satellite link
  - The original acronym was **MQ Telemetry Transport**
    - MQ Series was an IBM product type for message queueing
      → But MQTT does not need to support message queueing
    - IBM released MQTT implementation in 2010
    - In 2013, **OASIS** standardized MQTT 3.1
      - *"A nonprofit consortium that drives the development, convergence and adoption of open standards for the global information society"*
      - http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
    - In 2016, **ISO** standardized MQTT 3.1 as ISO/IEC PRF 20922
    - In 2018, **MQTT 5.0** is released.

MQTT Publish/Subscribe Model

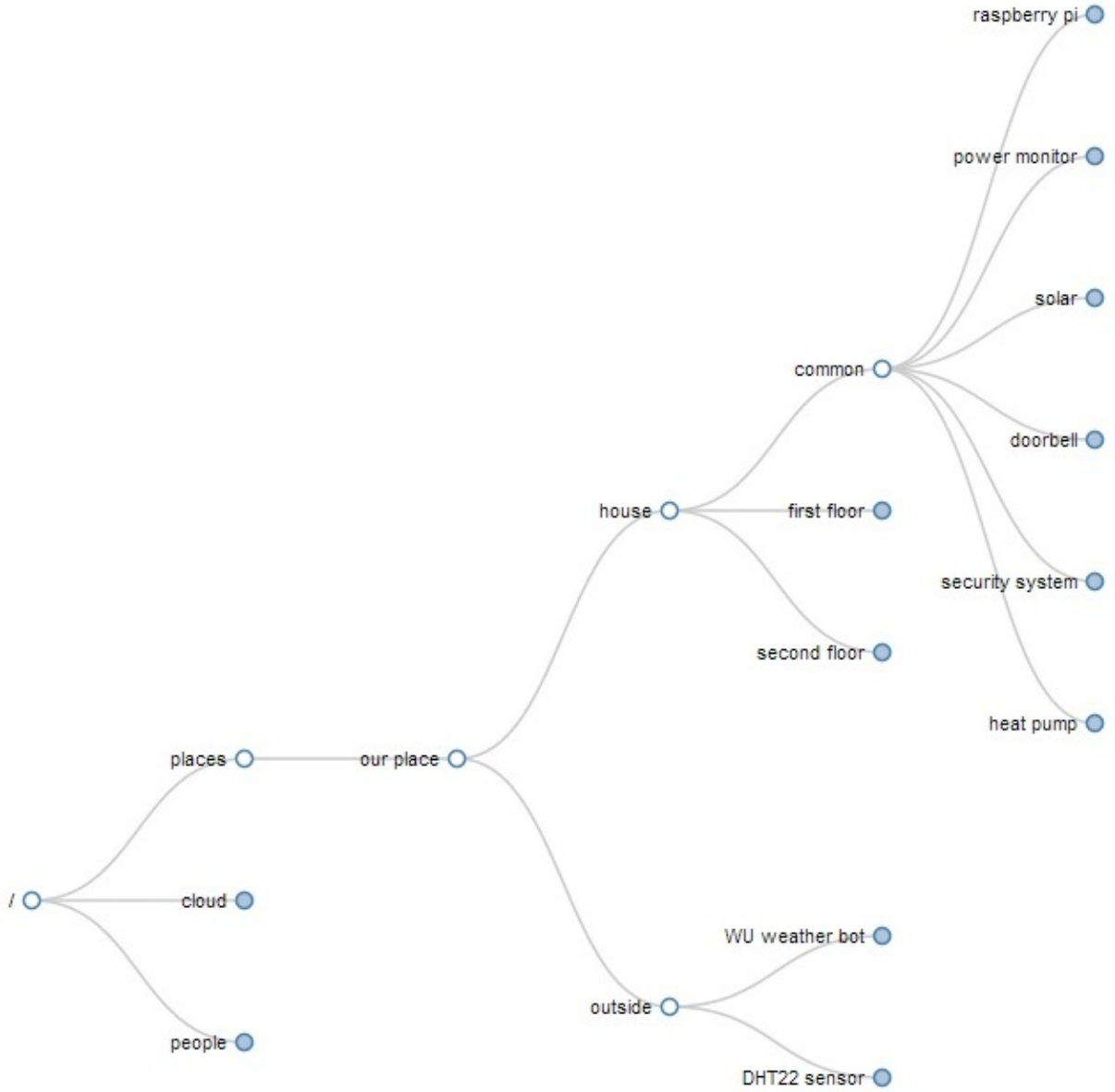
Institute for the Wireless Internet of Things at Northeastern

- MQTT allows devices to send (**publish**) information about a given topic to a server (MQTT message **broker**)
- The broker then **pushes** the information out to those clients that have previously **subscribed** to the topic
- A topic looks like a hierarchical file path:
  - Clients can subscribe to a specific level of a topic's hierarchy or
  - Use a wildcard character to subscribe to multiple levels



MQTT Topic Structure

Institute for the Wireless Internet of Things at Northeastern



MQTT Protocol

Institute for the Wireless Internet of Things at Northeastern

- An MQTT session is divided into four stages:
  1. **Connection:** CONNECT/CONNACK
     - A client starts by creating a TCP/IP connection to the broker by either using a standard port or a custom port defined by the broker's operators
       - 1883 for non-encrypted communication
       - 8883 for encrypted communication using SSL/TLS
  2. **Authentication:**
     - Without SSL/TLS:
       - Clear-text username and password are sent by the client to the server as part of the CONNECT/CONNACK packet sequence
     - With SSL/TLS handshake:
       - The client validates the server certificate to authenticate the server
       - The client may also provide a client certificate to the broker during the handshake which the broker can use to authenticate the client

MQTT Protocol

Institute for the Wireless Internet of Things at Northeastern

- An MQTT session is divided into four stages:
  3. **Communication:** Three types of service
     - **Unacknowledged Service (*"at most once"*):** PUBLISH
       - The publisher sends a message one time to the broker and the broker passes the message one time to subscribers
       - There is no mechanism in place to make sure the message has been received correctly and the broker does not save the message
     - **Acknowledged Service ("*at least once*"):** PUBLISH/PUBACK
       - Packet/ACK packet sequence between the publisher and its broker, as well as between the broker and subscribers
       - Retransmit otherwise
     - **Assured Service (*"Exactly once"*):**
       - Two pairs of packets:
         - PUBLISH/PUBRECeived
         - PUBREL/PUBCOMPlete

MQTT Protocol

Institute for the Wireless Internet of Things at Northeastern

- An MQTT session is divided into four stages:
  3. **Communication:** A client can perform 4 operations:
     - **Publish:**
       - The publish operation sends a binary block of data (the content) to a topic that is defined by the publisher
       - MQTT supports messages of up to 256 MB in size, 2 byte header
       - The format of the content is application specific
     - **Subscribe:** Topic subscriptions are made using a SUBSCRIBE/SUBACK packet pair
     - **Unsubscribe:** Un-subscription is similarly performed using a UBSUBSCRIBE/UNSUBACK packet pair
     - **Ping:** PINGREQ/PINGRESP is used to maintain the TCP connection alive
  4. **Termination:** DISCONNECT

Challenges for MQTT

Institute for the Wireless Internet of Things at Northeastern

- **Scalability:** Topic structure can easily form a huge tree and there's no clear way how to divide a tree into smaller logical domains that can be federated:
  - This makes it difficult to create a globally scalable MQTT network, because as the size of the topic tree grows, complexity increases
- **Interoperability:** Message payloads are binary, with no information as to how they are encoded
  - Problems can arise in open architectures where different applications from different manufacturers are supposed to work seamlessly with each other
- **Security:** MQTT has minimal authentication features built into the protocol:
  - Username and passwords are sent in clear text and any form of secure use of MQTT must employ SSL/TLS, which unfortunately, is not a lightweight protocol
  - Authenticating clients with client-side certificates is not a simple process and there's no way in MQTT, except by proprietary out-of-band means, to control who owns a topic and who can publish information on it

HTTP: Hyper Text Transfer Protocol

Institute for the Wireless Internet of Things at Northeastern

- Even though HTTP is old, it is still a perfectly valid protocol for transferring data between clients and servers:
  - HTTP is the most compatible with existing network infrastructure
    - If a web browser works, then nodes in this scenario should be able to send data to the cloud
  - Stateless HTTP will traverse firewalls and NAT systems easily

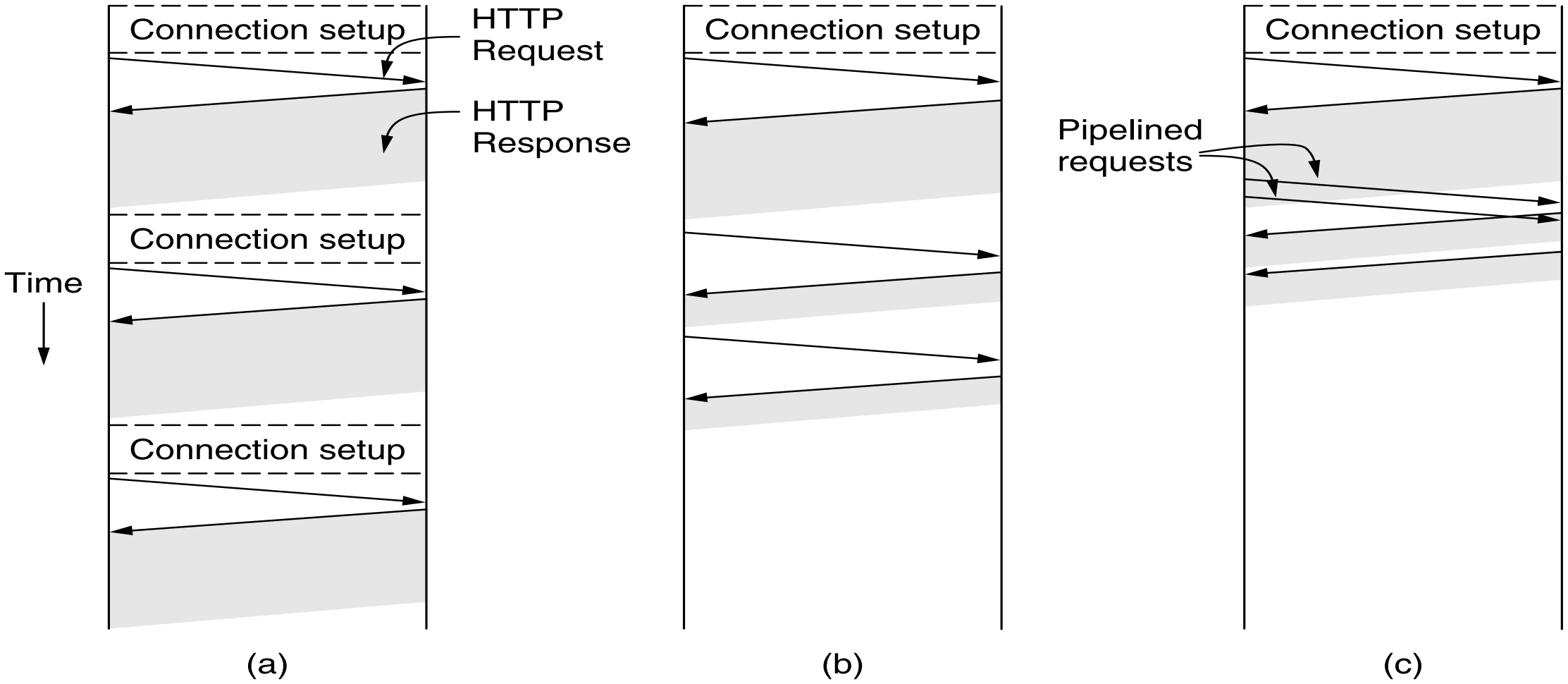
- Because the nodes initiate the connection to the server, the machines are much easier to secure
  - They don't even need any open ports
  - Most of the security effort in this system only needs to be focused in one place (cloud server)

- HTTP provides well understood authentication and encryption mechanisms

Reminder: HTTP

- HTTP is the protocol used to transport the web content from the web servers to the clients' browser
  - RFC 2616 (1999)
- The usual way for a browser to contact a server is to establish a TCP connection to port 80 on the server's machine
- The way in which TCP connections are handled in HTTP have changed since its creation:
  a) Multiple sequential TCP connections, each containing a single HTTP request
     - Too much overhead (connection establishment and release)
  b) Single persistent TCP connection, over which multiple HTTP requests are sent one after the other, sequentially
     - No need to go again through the slow-start process
  c) Single persistent TCP connection, over which multiple HTTP requests are sent simultaneously
     - The server is idling for a shorter time

HTTP: TCP Connections



HTTP: Methods

- Type of requests that can be sent over an HTTP connection

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

HTTP: Message Headers

Institute for the Wireless Internet of Things at Northeastern

- The “parameters” of the request

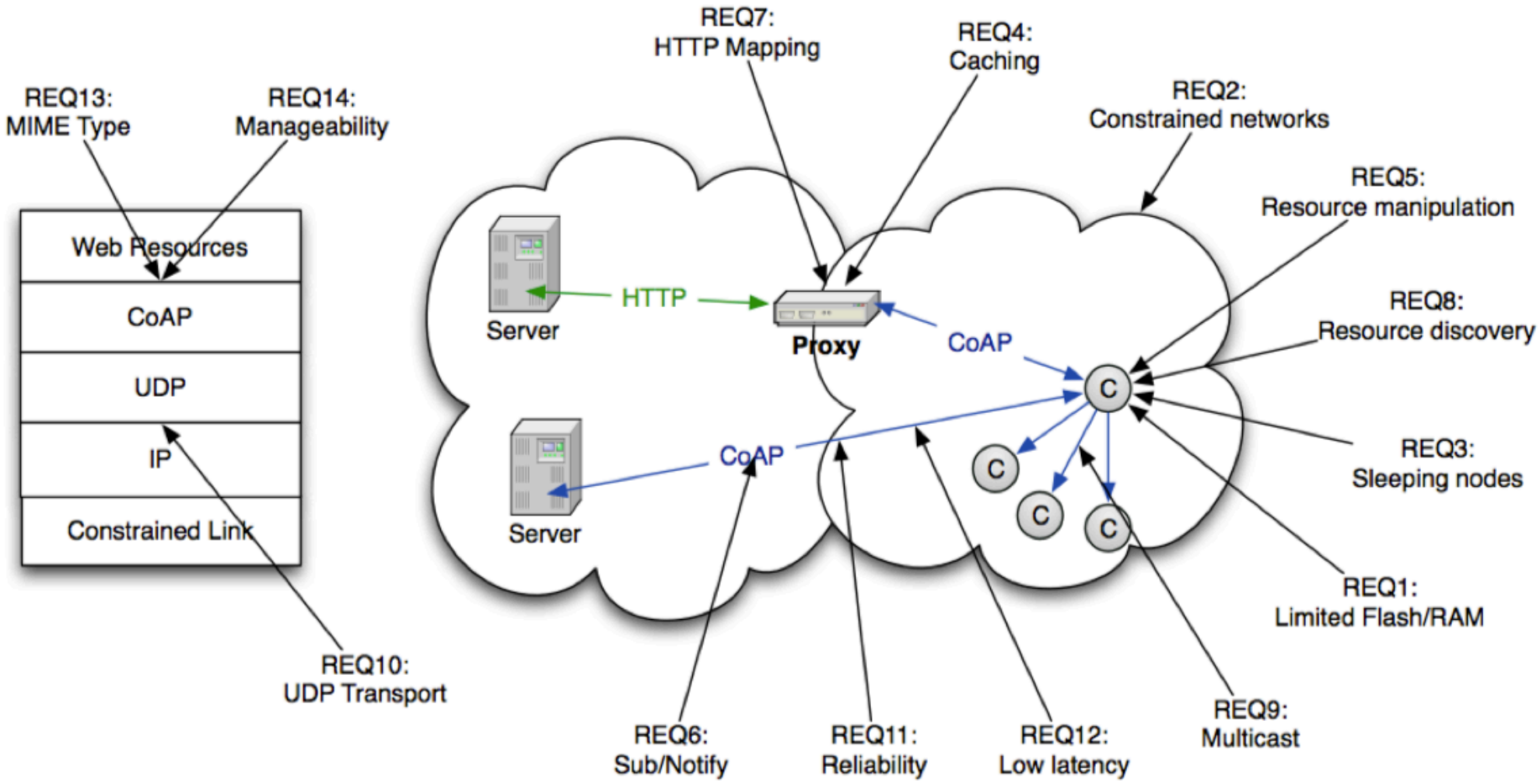
Header	Type	Contents
Accept	Request	The type of pages the client can handle
Accept-Encoding	Request	The page encodings the client can handle
If-Modified-Since	Request	Time and date to check freshness
Authorization	Request	A list of the client’s credentials
Host	Request	The server’s DNS name
Last-Modified	Response	Time and date the page was last changed
Content-Type	Response	The page’s type
…		

CoAP: Constrained Application Protocol

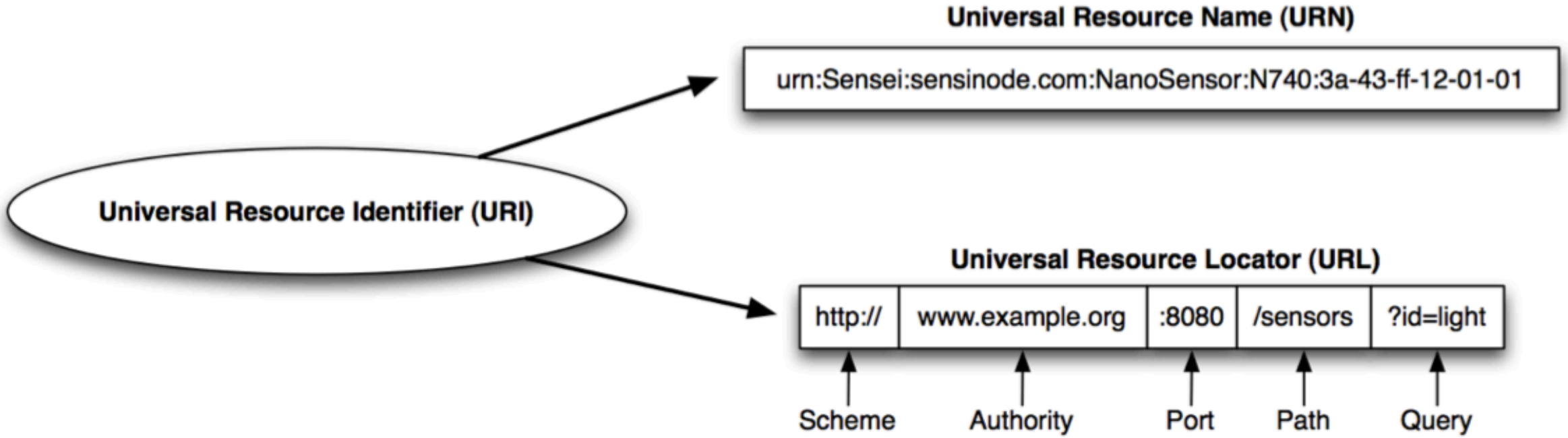
Institute for the Wireless Internet of Things at Northeastern

- Described in the RFC 7252, developed by the **IETF Constrained RESTful Environments (CoRE) working group**
- A specialized web transfer protocol that is suitable for:
  - **Constrained nodes** (low processing power, limited memory)
  - **Constrained networks** (low bandwidth)
- Like HTTP, servers make resources available under a **URI**…
  - … clients can access these resources using methods such as GET, PUT, POST, and DELETE
- The close resemblance with HTTP makes it very **user-friendly**…
  - … and also makes it easier to connect CoAP and HTTP using application-agnostic **cross-protocol proxies**
- As opposed to HTTP, CoAP is built on top of UDP

CoAP


REQ13:
MIME Type
REQ14:
Manageability
Web Resources
CoAP
UDP
IP
Constrained Link
REQ10:
UDP Transport
REQ7:
HTTP Mapping
REQ4:
Caching
REQ2:
Constrained networks
REQ5:
Resource manipulation
REQ8:
Resource discovery
REQ3:
Sleeping nodes
REQ1:
Limited Flash/RAM
REQ9:
Multicast
REQ12:
Low latency
REQ11:
Reliability
REQ6:
Sub/Notify
Server
HTTP
Proxy
CoAP
Server
CoAP
C
C
C
C

URI: Universal Resource Identifier



CoAP Messaging Model

Institute for the Wireless Internet of Things at Northeastern

- Simple message exchange between end-points
  - 4 bytes header, max payload 8 kB
  - Includes Message ID to detect duplicates, increase reliability

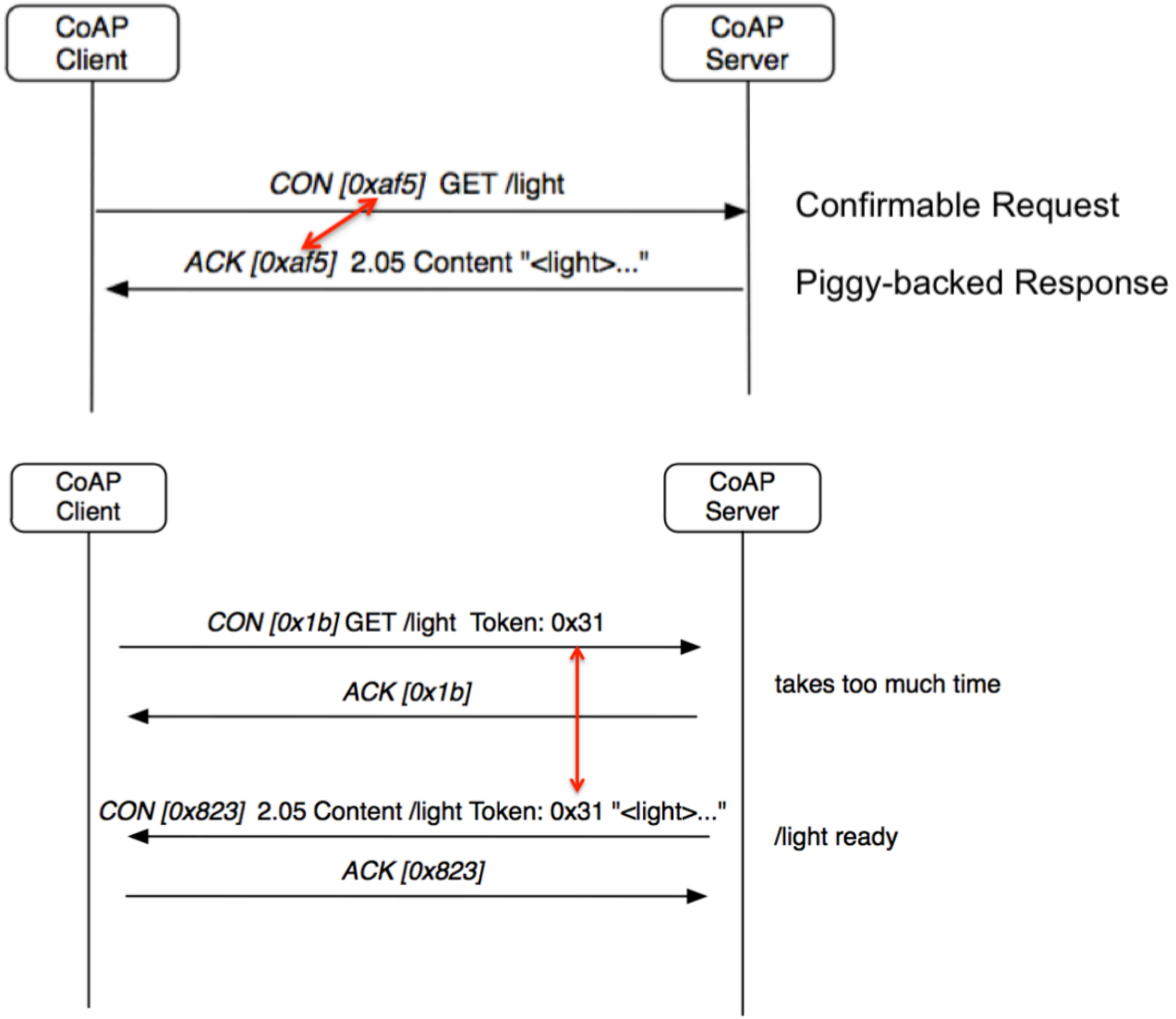
- Types of Message:
  - **CON** (Confirmable)
    - Message is retransmitted using a default timeout and exponential back-off between retransmissions, until the recipient sends an ACK with the same Message ID
  - **NON** (Non-confirmable)
    - Not acknowledged, but still have a Message ID for duplicate detection
  - **ACK** (Acknowledgement)
  - **RST** (Reset)

CoAP Request/Response Model

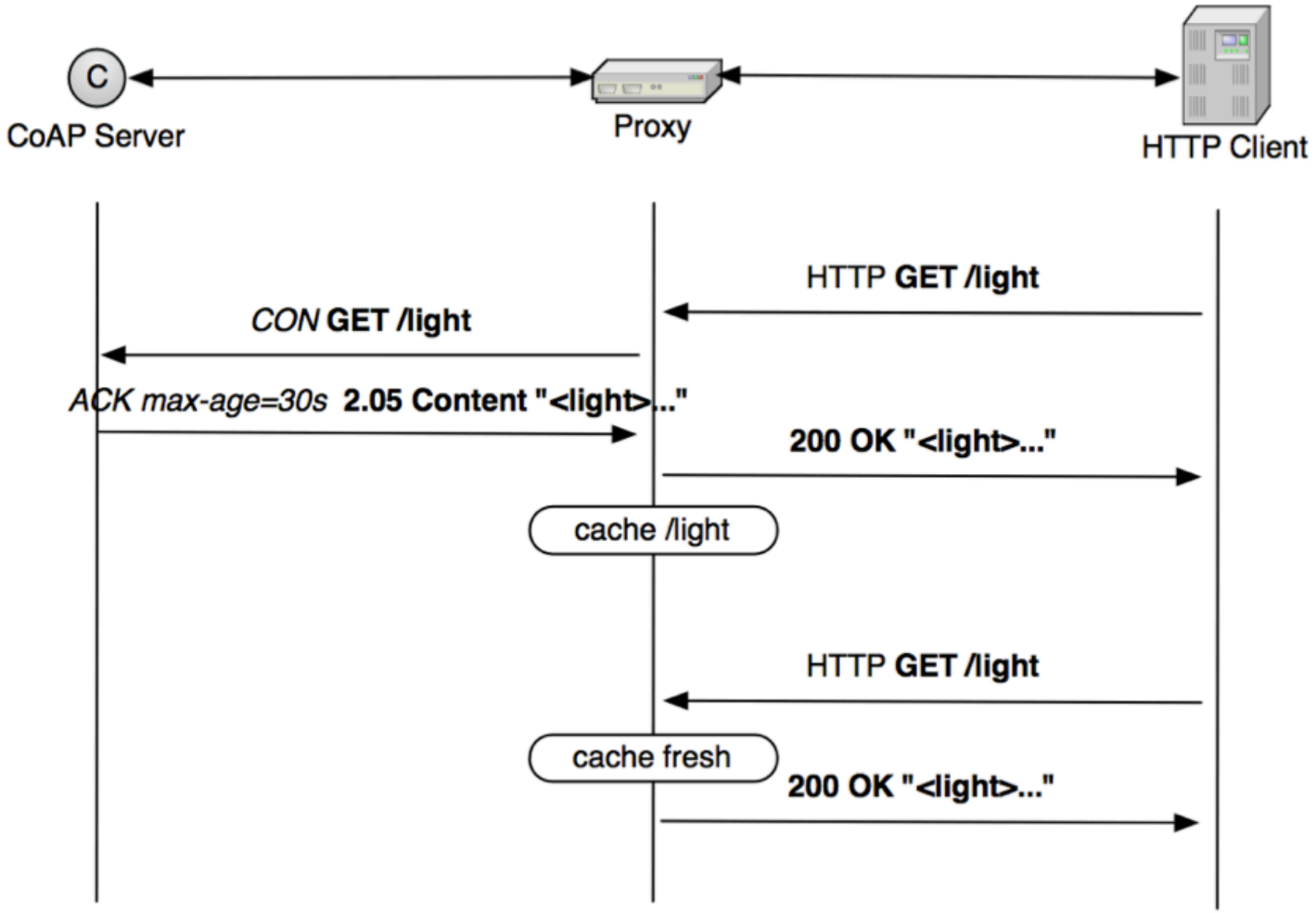
Institute for the Wireless Internet of Things at Northeastern

- CoAP request and response semantics are carried in CoAP messages, which include either a Method Code or Response Code, respectively
- If a request is carried in a CON message
  - If immediately available, the response to a request carried in a Confirmable message is carried in the resulting Acknowledgement (ACK) message
  - If the server is not able to respond immediately to a request carried in a Confirmable message, it responds with an Empty Acknowledgement message so that the client can stop retransmitting the request
    - When the response is ready, the server sends it in a new Confirmable message (which then in turn needs to be acknowledged by the client)
- If a request is sent in a NON message, then the response is sent using a new Non-confirmable message

CoAP Request/Response Model



CoAP/HTTP Proxy



XMPP: Extensible Messaging and Presence Protocol

- An application profile of the **Extensible Markup Language** (XML) that enables the near-real-time exchange of structured yet extensible data between any two or more network entities

- Standardized by **IETF**:
  - RFC 6120 (used to be 3920): Core
  - RFC 6121 (used to be 3921): Instant Messaging and Presence
  - RFC 7622 (used to be 6122): Address Format
  - RFC 3922: Mapping to CPIM
  - RFC 3923: End-to-end Signaling

- Developed originally by the **Jabber** open-source community (1999)
  - One of the ancient instant messaging platforms
    - At the basis of text messaging with Google Talk, Hangouts, etc.

XML

- A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable
  - (Reminder: HTML: Hypertext Markup Language, you use it to write websites…)

- XML is not a replacement for HTML
  - XML was designed to **describe data and to focus on what data is**
    - HTML was designed to **display data and to focus on how data looks**
  - XML is about describing information
    - HTML is about displaying information

XMPP: Model

Institute for the Wireless Internet of Things at Northeastern

- **Decentralized client–server architecture:**
  - Clients do not talk directly to one another
  - There is no central authoritative server
  - Anyone can run a server
  - Anyone may run their own XMPP server on their own domain

- **Implementation:**
  - Originally: XMPP on top of TCP
  - Currently: the majority of XMPP messages on top of HTTP
  - Alternative: WebSockets

WebSockets

Institute for the Wireless Internet of Things at Northeastern

- An application-layer communication protocol that provides bidirectional full-duplex channels over a single TCP connection
  - **Bidirectionality:**
    - There are no pre-defined message patterns such as request/response
    - Either client or server can send a message to the other party
  - **Full-duplex:**
    - Client and server can talk independent of each other
  - **Single TCP:**
    - Client and server communicate over that same TCP connection for the lifecycle of WebSocket connection
  - **Efficient implementation**

WebSockets: Protocol

Institute for the Wireless Internet of Things at Northeastern

- Designed to work well with the existing Web infrastructure (i.e., HTTP)
  - The connection starts its life as an HTTP connection, guaranteeing full backwards compatibility
  - The protocol switch from HTTP to WebSocket is referred to as a the WebSocket handshake
    - The browser sends a request to the server, indicating that it wants to switch protocols from HTTP to WebSocket
    - If the server understands the WebSocket protocol, it agrees to the protocol switch through the Upgrade header

XMPP: Addressing

Institute for the Wireless Internet of Things at Northeastern

- Every user on the network has a unique XMPP address, called JID (for historical reasons)
  - The JID is structured like an email address with a username and a domain name (or IP address) for the server where that user resides, separated by an at sign (@) → username@example.com

- Since a user may wish to log in from multiple locations, they may specify a resource:
  - A resource identifies a particular client belonging to the user (for example home, work, or mobile)
  - This may be included in the JID by appending a slash followed by the name of the resource → username@example.com/mobile

- Each resource may have specified a numerical value called priority:
  - Messages simply sent to username@example.com will go to the client with highest priority
  - Those sent to username@example.com/mobile will go only to the mobile client

XMPP: Services

Institute for the Wireless Internet of Things at Northeastern

- Channel Encryption
- Authentication
- Presence
- Contact lists
- One-to-one messaging
- Multi-party messaging
- Notifications
- Service discovery
- Capabilities advertisement
- Structured data forms
- Workflow management
- Peer-to-peer media sessions

XMPP: XML Stanzas

Institute for the Wireless Internet of Things at Northeastern

- The basic unit of communication, similar to a packet or message in other network protocols
- Defined by:
  - The stanza element name, which is **message**, **presence**, or **iq**
    - Each kind of stanza is routed differently by servers and handled differently by clients.
  - The value of the type **attribute**, which varies depending on the kind of stanza in question
    - This value further differentiates how each kind of stanza is processed by the recipient.
  - The **child element(s)**, which define the payload of the stanza
    - The payload might be presented to a user or processed in some automated fashion as determined by the specification that defines the namespace of the payload.

Some thoughts about XMPP

Institute for the Wireless Internet of Things at Northeastern

- XMPP is an IM protocol, and has a much, much higher overhead in handling presence messages between all the clients

- If you have a small memory footprint constraint, then having to handle the XML parser may make the use of XMPP impossible

- XMPP is an IM protocol which:
  - carefully defines all the message formats and requires that all messages be in XML
  - can be extended to support publish / subscribe procedures

AMQP: Advanced Message Queueing Protocol

Institute for the Wireless Internet of Things at Northeastern

- An open internet protocol for business messaging:
  - It defines a **binary wire-level protocol** that allows for the reliable exchange of business messages between two parties
    - Utilized by many companies, e.g., banking and finance, defense, manufacturing, etc.
  - Developed by OASIS: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf
  - Standardized as ISO/IEC 19464
- Used to transfer messages between two nodes:
  - Peer-to-peer
  - Peer-to-broker (server)
  - Broker-to-peer
- Usually on top of TCP

AMQP Frames

Institute for the Wireless Internet of Things at Northeastern

- The basic unit of data in AMQP is a *frame*:
  - There are nine AMQP frame bodies defined that are used to initiate, control and tear down the transfer of messages between two peers:
    - open
    - begin
    - attach
    - transfer
    - flow
    - disposition
    - detach
    - end
    - close

AMQP Links

Institute for the Wireless Internet of Things at Northeastern

- The link protocol is at the heart of AMQP.
  - An *attach* frame body is sent to initiate a new link
  - A *detach* frame is used to tear down a link
- Messages are sent over an established link using the *transfer* frame
- Messages on a link flow in only one direction
- Transfers are subject to a credit based flow control scheme, managed using flow frames
  - This allows a process to protect itself from being overwhelmed by too large a volume of messages
- Each transferred message must eventually be settled:
  - Settlement ensures that the sender and receiver agree on the state of the transfer, providing reliability guarantees.
- Multiple links, in both directions, can be grouped together in a session:
  - A session is a bidirectional, sequential conversation between two peers that is initiated with a begin frame and terminated with an end frame
  - A connection between two peers can have multiple sessions multiplexed over it, each logically independent

Some Thoughts

- **Which one is "the best"?**
  - Depends on your application
    - All these protocols provide general solutions, adaptable to your specific application
      - Widely adopted → Choose based on what your cloud supports
      - You might want to learn them ;)
    - Many times, you might want to define your data streaming protocol own..s

Next Steps

- Research paper:
  - Due on November 20

- First Quiz (up to Module T4):
  - December 1, 2020
    - On Canvas, open-book, multiple-choice questions
    - The exam will be available for 24h, but you will only have 60 minutes to complete it

- Coming soon:
  - Homework assignment (numerical)
  - Third (and last) laboratory assignment

Course Contents

Institute for the Wireless Internet of Things at Northeastern

- **Module T1:** Introduction to the Internet of Things ✓
- **Module T2:** Data Acquisition ✓
- **Module T3:** Local Data Processing ✓
- **Module T4:** Data Communication ✓
- **Module T5:** Data Streaming ✓
- **Module T6:** Data Storage & Cloud
- **Module T7:** Data Analytics