University of El-Oued

Faculty of Exact Sciences — 18/01/2026
Department of Computer Science — Duration: 90 minutes
2nd Year Master's (IoT&CS) — Module: Networks for IoT

Exam

Q1: Select the correct answer(s)? (4 points)

1. **What is the primary difference between Active and Passive RFID tags?** b) Active tags are battery-powered and can initiate communication, while passive tags lack an internal power source and rely on the reader's energy.	2. **Which of the following are true about beacon-enabled mode?** a) Superframe structure defines active and inactive periods b) Nodes sleep during inactive periods to save energy d) PAN coordinator transmits periodic beacons for synchronization
3. **What are the functions of the IoT gateway in a network?** a) To manage communication between IoT devices and external networks b) Protocol translation d) handle raw sensor data from devices	4. **In the context of IoT network protocols, which is considered a lightweight protocol for constrained devices?** b) CoAP
5. **An IoT Communication model can be** a) Device-to-Cloud (D2C) **b)** Device-to-Device (D2D)	6. **The IETF defines "Low-Power and Lossy Networks (LLNs)" as networks with resource-constrained nodes using communication technologies that are** b) short-range, c) low-data-rate
7. **IoT devices can be** a) Controllers and actuators b) Sensors **d)** gateways	8. **In the RPL routing protocol, which algorithm is used to control the timing and frequency of control message transmissions?** b) Trickle Timer Algorithm

Q2: Answer with True or False? (3 points)

1. Low Power Wide-Area Networks (LPWAN) encompasses wireless technologies built for large-scale IoT deployments, enabling long-distance connections with minimal power use. **T**	2. Device-to-Device (D2D) is the IoT Communication model that enabling direct interaction between IoT devices **T**
3. In an IEEE 802.15.4 network, a Full Function Device (FFD) is capable of acting as a coordinator, a router, or an end device, whereas a Reduced Function Device (RFD) is limited to being an end device. **T**	4. Proximity detection systems are primarily concerned with establishing a precise, long-distance communication link between two devices. **F**

5. EPC code is used for addressing IoT devices while IPv6 is used for identifying them in an IoT network. **F**	6. IoT Networking focuses on how devices are organized, connected, and the physical range they cover. It establishes the infrastructure that allows devices to exist on a shared system. **T**

Q3: Answer the following questions? (8 points)

1) Compare MAC addresses and IPv6 addresses. Give an example of a MAC address.

Feature	MAC Address	IPv6 Address
OSI layer	Data Link (Layer 2)	Network (Layer 3)
Length	48 bits (or 64 bits)	128 bits
Scope	Local network only	Global (Internet-wide)
Assignment	Manufacturer	ISP / Network administrator
Routable	No	Yes

Example of a MAC address : 00:1A:2B:3C:4D:5E

1) **Determine the Link-Local and Global IPv6 addresses (using the EUI-64 format) for the following MAC address**: **39:A7:94:07:CB:D0**

To convert a MAC address to an IPv6 Interface ID (EUI-64), we follow a specific 3-step process.

MAC Address: `39:A7:94:07:CB:D0`

Step 1: Split and Insert

Split the MAC address into two halves (3 bytes each) and insert the reserved 16-bit value **FFFE** in the middle.

- `39:A7:94` + **FF:FE** + `07:CB:D0`
- Result: `39A7:94FF:FE07:CBD0`

Step 2: Flip the 7th Bit (Universal/Local Bit)

We must invert the 7th bit of the first byte (from the left) to indicate that this is a "Universally/Locally Administered" address.

- First byte (Hex): **39**
- Convert to Binary: 0011 10**0**1
- **Flip the 7th bit:** Change **0** to **1**.
- New Binary: 0011 10**1**1
- New Hex Value: **3B**

New Interface ID: `3BA7:94FF:FE07:CBD0`

Step 3: Formulate the Addresses

1. Link-Local Address

The Link-Local prefix is always FE80::/10. We combine this prefix with the Interface ID calculated above.

- **Answer:** `FE80::3BA7:94FF:FE07:CBD0`

2. Global Unicast Address

A global address depends on the network prefix assigned by the router (e.g., provided by an ISP). Since no prefix was given in your question, we use the standard documentation prefix (2001:DB8::/64) as an example.

- Format: `[Global Prefix] + [Interface ID]`
- **Example Answer:** `2001:DB8::3BA7:94FF:FE07:CBD0`

2) A sensor node wants to send **300 bytes** of application data using UDP over IPv6 in a 6LoWPAN network.

1. Total size of the uncompressed UDP/IPv6 datagram
* Application data: 300 Byte
* UDP header: 8 Byte
* IPv6 header: 40 Byte
UDP payload+header = (300+8=308) Byte
IPv6 datagram = (308+40= 348 Byte)
2) After 6LoWPAN compression, header size is 27 bytes
2a) Which 6LoWPAN header compression scheme is used?
Stateful 6LoWPAN header compression:IPHC (for IPv6) + NHC (for UDP).
2b) Total size of the compressed datagram
Payload: 300 B, Compressed headers: 27 B → [300 + 27 = 327 Byte]
3) Fragmentation with 118 Byte IEEE 802.15.4 MAC payload
6LoWPAN fragmentation headers:
* First fragment header (FRAG1): 4 Byte
* Subsequent fragment header (FRAGN): 5 Byte
Offset unit is 8 bytes, so all non-final fragments should carry a multiple of 8 bytes of datagram data.

- Per-fragment datagram-data capacity

* First fragment data max: (118-4=114) → largest multiple of 8 is 112 Byte
* Next fragment data max: (118-5=113) → largest multiple of 8 is 112 Byte
So use 112 Byte datagram-data for each "full" fragment.

3a) Number of fragments for a 327-byte datagram

After 2 full fragments: (112+112=224) → Remaining: (327-224=103)
A third fragment can carry at most 112, so it fits → **3 fragments**

3b) Size of each fragment (datagram data bytes + frag header)
* **Fragment 1:data 112 B + FRAG1 4 B = 116 Byte
* **Fragment 2:data 112 B + FRAGN 5 B = 117 Byte
* **Fragment 3: data 103 B + FRAGN 5 B = 108 Byte
 All are ≤ 118 B MAC payload, and offsets are valid:
* Frag2 offset = (112/8=14)
* Frag3 offset = (224/8=28)

Q5: Use case (5 points)

Figure below shows an end-to-end IP network that combines a traditional IPv6 core with a 6LoWPAN access network.

1) For the IoT access network indicate:
   a) The suitable technology (PHY and Link layers)?
      - **Physical (PHY) & MAC: IEEE 802.15.4**.

- *Standard Details:* This defines the **LR-WPAN** (Low-Rate Wireless Personal Area Network). It operates typically in the **2.4 GHz ISM band** using **O-QPSK** modulation.
- *Device Roles:* The Edge Router acts as the **FFD (Full Function Device)** and **PAN Coordinator**, managing synchronization. The sensor (Device 2) acts as an **RFD (Reduced Function Device)** or a leaf node.
  - **Adaptation Layer: 6LoWPAN**.
- *Why it's needed:* It resolves the mismatch between the IPv6 MTU (1280 bytes) and the IEEE 802.15.4 frame size (127 bytes) via **Fragmentation** and **Header Compression (IPHC)**.

b) The network topology and communication model.

Mesh or or a **Star topology and Device-to- Device (D2D) or Device-to-Gateway (D2G): Sensor sends data to the Edge Router**

c) Calculate the 64-bit EUI-64 Interface Identifier (IID) for the d**evice "2"** considering **16-bit short addresses** and **a** 16-bit PAN ID **(1)**

According to RFC 4944 (6LoWPAN), when using 16-bit short addresses, the pseudo-48-bit address is created by appending the PAN ID.

The formula for the Interface ID is: [PAN ID (16 bits)] : [00FF] : [FE00] : [Short Address (16 bits)]

- **PAN ID:** 1 → `0x0001`
- **Short Address (Device 2):** 2 → `0x0002`
- **Middle constants:** `00FF:FE00`

Resulting IID: 0001:00FF:FE00:0002

2) Assign a global unicast IPv6 addresses to: **remote server, IPv6 core** and **access networks**?

Remote Server: The IPv6 address for the remote server can be `2001:10::2105`.

IPv6 Core: The **IPv6 Core** network has the prefix `2001:10::/64`, and an address could be `2001:10::/64` for the network interface.

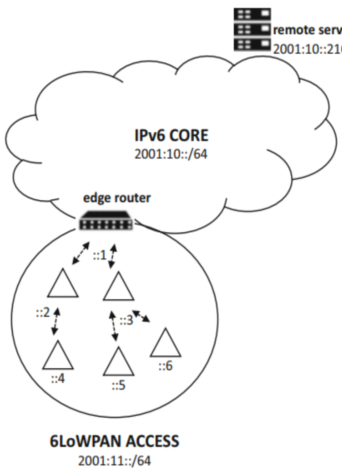
Access Network: The 6LoWPAN Access Network has the address prefix `2001:11::/64`. Devices in this network would have addresses such as:

* Device 1: `2001:11::1`

* Device 2: `2001:11::2`

* Device 3: `2001:11::3`

* etc.



3) **Develop an NS3 script to simulate the access network.**

Here is a simplified C++ script for **NS-3** that simulates a basic **6LoWPAN access network** (IEEE 802.15.4) with 2 nodes (1 Server/Gateway node and 1 Client/Sensor node) sending UDP packets.

C++
```
#include "ns3/core-module.h"
#include "ns3/csma-module.h"
```

```
#include "ns3/internet-module.h"
#include "ns3/sixlowpan-module.h"
#include "ns3/lr-wpan-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"

using namespace ns3;

int main (int argc, char *argv[])
{
  // 1. Create Nodes
  // Node 0 = Gateway/Coordinator, Node 1 = Sensor Device
  NodeContainer nodes;
  nodes.Create (2);

  // 2. Install Mobility (Static positions)
  MobilityHelper mobility;
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (nodes);

  // Set positions manually for visualization
  nodes.Get (0)->GetObject<MobilityModel> ()->SetPosition (Vector (0, 0, 0));
  nodes.Get (1)->GetObject<MobilityModel> ()->SetPosition (Vector (10, 0, 0));

  // 3. Configure IEEE 802.15.4 (PHY and MAC)
  LrWpanHelper lrWpan;
  NetDeviceContainer lowpanDevices = lrWpan.Install (nodes);

  // Associate devices to PAN (PAN ID 1) - Simplified manual association
  // Note: specific NS3 versions usually handle association automatically or via
bootstrapping
  // This step assumes the devices are ready to communicate on the same channel.

  // 4. Install 6LoWPAN Layer
  SixLowPanHelper sixlowpan;
  NetDeviceContainer sixDevices = sixlowpan.Install (lowpanDevices);

  // 5. Install Internet Stack (IPv6)
  InternetStackHelper stack;
  stack.Install (nodes);

  // 6. Assign IPv6 Addresses
  Ipv6AddressHelper ipv6;
  // Assigning prefix 2001:db8:2000::/64 (as defined in Q2)
  ipv6.SetBase (Ipv6Address ("2001:db8:2000::"), Ipv6Prefix (64));
  Ipv6InterfaceContainer interfaces = ipv6.Assign (sixDevices);

  // 7. Install Applications (UDP Echo)

  // -- SERVER APPLICATION (On Node 0 - Gateway) --
  uint16_t port = 9;
  UdpEchoServerHelper echoServer (port);
  ApplicationContainer serverApps = echoServer.Install (nodes.Get (0));
  serverApps.Start (Seconds (1.0));
  serverApps.Stop (Seconds (10.0));

  // -- CLIENT APPLICATION (On Node 1 - Sensor) --
  // Target is Node 0's IPv6 address
  UdpEchoClientHelper echoClient (interfaces.GetAddress (0, 1), port);
  echoClient.SetAttribute ("MaxPackets", UintegerValue (5));
  echoClient.SetAttribute ("Interval", TimeValue (Seconds (1.0)));
  echoClient.SetAttribute ("PacketSize", UintegerValue (50)); // Small payload
  ApplicationContainer clientApps = echoClient.Install (nodes.Get (1));
  clientApps.Start (Seconds (2.0));
  clientApps.Stop (Seconds (10.0));

  // 8. Run Simulation
  Simulator::Run ();
  Simulator::Destroy ();

  return 0;
}
```