Link-Layer Discovery Protocol (LLDP) Agent

In this project your goal is to create a simple LLDP agent, which should be able to:

- announce itself by sending LLDP messages to the network and
- parse LLDP messages received from the network and print them on `stdout`.

Along with this task sheet, you have received a framework for the LLDP agent, which you will have to complete, as well as an extensive set of unit tests to verify all the parts of your implementation.

Usage

All components that open raw sockets need **root priviliges**. This includes the LLDP agent itself as well as the provided test suite.

You can start the provided LLDP agent by running the following command from the project root, where `<interface>` represents the network interface the agent uses to send and receive LLDPDUs.

```
cargo run --release <interface>
```

The tests can be executed by running the following command in the project root folder:

```
cargo test --release
```

And the Documentation can be opened using:

```
cargo doc --open
```

Passing

The following requirements have to be fulfilled to pass the project:

- When receiving packets from the network, only LLDP frames should be parsed.
- Your agent should only handle frames which are directed to one of the LLDP multicast addresses.
- Ensure that frames with TLVs which are not supported by the agent don't crash the agent.
- Ensure that your agent does not parse and output LLDP frames it sent itself.
- Generated frames should include all *mandatory* TLVs in the *correct order*.

You only need to send your messages to the nearest-bridge MAC address (`01:80:c2:00:00:0e`).

To pass this project **all tests we provide have to be passed**. They check most of the conditions above. There are **additional tests not visible to you**, which check whether you correctly implemented the specification. You are not allowed to use any libraries/crates in the project, except for the Rust Standard Library and the crates already included in the provided Cargo.toml file.

We encourage you to write your own tests while developing. You can use the tests provided to you as a reference.

Tip: Use an IDE (e.g. VSCode with the rust-analyzer plugin) on your host system and have your working folder as a Shared Folder in the Lab-VM. Run the tests inside the VM to make sure that your environment is the same as ours so that you don't accidentially fail the project.

Submission

We **require** you to hand in the code as a `.zip` archive, using **Moodle**. The structure should be **the same we provided**. You may include additional files (such as additional tests), however this is not required to pass the project.

We will **deduce points** for not adhering to the folder structure, so do not risk failing the project by not providing the files in the correct structure.

Framework

We provide you with a framework for writing your agent. You have to fill in all code sections:

- marked with "`// TODO: Implement`",
- where `todo!()` is returned.

`lldp/agent.rs` contains the main agent struct. You need to implement:

- opening a raw datalink channel in `new(...)`.
- announcing yourself by composing a valid LLDP message.

`lldp/lldpdu.rs` contains the data type that allows you to store a LLDPDU. You need to implement generating it `from_bytes`, appending new TLVs and checking for validity.

`lldp/tlv` contains one module for each TLV your agent has to support. You will have to implement all of their methods (in particular `len`, `bytes` and `from_bytes`).

Note that for the `ManagementAddressTLV` you only need to support IPv4 and IPv6 addresses.

See the comments in the class files for detailed information about specific methods.

Protocol Specification

For details on LLDP refer to:

- HON Unit 11: "Link Layer (MAC)"
- *IEEE 802.1ab* (`http://ieeexplore.ieee.org/document/7433915/`) (free after registration)