EPOS 2.2 - Adding Ethernet Protocols

Table of contents

How To Do

This is a simple and easy tutorial on how to **properly** add a new Ethernet Protocol on EPOS 2.2.
We will cover step-by-step the existing compiler-related structures adjusting.

1. EPOS code tree.

in order to improve the code tree divisions organization, EPOS is subdivided into many intuitive directories. The ones we will need to adjust in order to add a new protocol will be *include/network* and *include/system* which contains the *.h* EPOS files, and *src/network/* which will contain the new Protocol *.cc* files.

1.1. Create the Protocol directory structure

First, we need to create two directories.

- One at *include/network*, which will contain the *.h* files of the new protocol, and usually have the protocol name. Ex.: *include/network/ipv4*. In this case create a new directory as follows: ***include/network/protocol_name***.
- And another one, with the same name at the source counterpart. Ex.: *src/network/ipv4*. In this case create a new directory as follows: ***src/network/protocol_name***.

After this step, move the new protocol files to the respective directory, remember that to use a NIC defined over Ethernet, the new protocol must privately inherit *NIC<Ethernet>::Observer*.

2. Add the protocol directory on the makefile

In order for the compiler to find the new Protocol code, we need to add to the network makefile the created directories. The network makefile can be found at *src/network*.
We only need to modify the *SUBDIRS* declaration, adding the directory name in the line, as follows:

```
# EPOS Network Makefile

include ../../makedefs

SUBDIRS := ipv4 tstp protocol_name
OBJS := $(subst .cc,.o,$(shell find *.cc | grep -v _test | grep -v _init))
INITS := $(subst .cc,.o,$(shell find *.cc | grep _init))

all:            $(SUBDIRS) $(LIBSYS) $(LIBINIT)
```

```
$(SUBDIRS):     FORCE
                (cd $@ && $(MAKE))

#$(LIBSYS):     $(LIBSYS)($(OBJS))

$(LIBINIT):     $(LIBINIT)($(INITS))

clean:
                make MAKE:="$(MAKECLEAN)" $(SUBDIRS)
                $(CLEAN) *.o *_test

FORCE:
```

3. Add the protocol makefile

In order to the *.cc* protocol files to be compiled, we also need to add a makefile in *src/network/protocol_name/* to compile them. It's quite simple, just follow the example below (IPv4) and customize as needed:

```
# EPOS IPv4 Makefile

include ../../../makedefs

OBJS := $(subst .cc,.o,$(shell find *.cc | grep -v _test | grep -v _init))
INITS := $(subst .cc,.o,$(shell find *.cc | grep _init))

all:            $(LIBSYS) $(LIBINIT)

$(LIBSYS):      $(LIBSYS)($(OBJS))

$(LIBINIT):     $(LIBINIT)($(INITS))

clean:
                $(CLEAN) *.o *_test
```

4. Create the class Type

As any new class in EPOS, in order to use a newly defined class, we need to add a definition of this class on EPOS Types. Thus, add a declaration of the new defined classes in the new Protocol at *include/system/types.h*. Tip: look for the already defined protocols and add the new classes definition right bellow. Remember that the class should have the same name as the classes your protocol defines.

5. Define the Protocol Traits

EPOS metaprogram uses Traits to configure most of the classes and to compile only the needed files for the application to run. So, each application defines its own application or inherits it from the System.
To complete the addition of the new Protocol to EPOS, you must first define its traits at the machine traits file They can be found at *include/system/traits.h* or can be defined in each application Traits file. The configuration added to the traits file should look like this (you can modify the traits as needed):

```
template<> struct Traits<Protocol_Name>: public Traits<Network>
{
```

```
    static const bool enabled = NETWORKS::Count<Protocol_Name>::Result;

    static const unsigned int RETRIES = 2;
    static const unsigned int TIME_OUT = 3;
};
```

Along with the protocol traits definition, it is also needed to enable it through the Network Traits (as the *enabled* variable depends on the result of the NETWORKS Count operation). Thus, modify Network Traits as follows adding the new protocol to the NETWORKS list:

```
template<> struct Traits<Network>: public Traits<void>
{
    static const bool enabled = (Traits<Build>::NODES > 1);

    static const unsigned int RETRIES = 3;
    static const unsigned int TIMEOUT = 10; // s

    typedef LIST<Protocol_Name> NETWORKS;
};
```