Vehicular Networking [C2X]

# EDCA Performance Evaluation

Agon Memedi, Gurjashan Pannu

17.05.2019, due on 06.06.2019 at 23:55

## 1 Introduction

IEEE 802.11p adopts the Enhanced Distributed Channel Access (EDCA) mechanism from IEEE 802.11e for ensuring Quality of Service (QoS). In this exercise, you will be doing performance evaluation of the system in terms of end-to-end packet delays via different access categories and the goodput observed under simulated conditions.

**Attention**

Carefully follow the instructions given in the sheet.

Solution to the *Tasks* have to be handed in as PDF via PANDA[a], hence make sure you are registered in the Vehicular Networking course in PANDA and you also have a team.

Note down the name, surname and matriculation number of all *contributing* group members in the PDF solution and submit it until the due date. Late or wrong submissions will not be considered.

This is a mandatory exercise sheet and the submissions will be graded!

[a] https://panda.uni-paderborn.de/

## 2 EDCA Performance Evaluation

To help you getting started with the project easily, we provide a skeleton project which you will be extending further. The skeleton project can be found here[1]. Please make sure that you already have Veins in your OMNeT++ IDE workspace as the project uses models defined in Veins.

1. After downloading the skeleton code from the course website, import it to the workspace:
   From the menu bar select *File* → *Import* → *General* → *Existing Projects into Workspace* → *Select archive file* radio button → *Browse* for the downloaded skeleton code, select it and *Finish* importing.
2. Compile the code for the *edca* project by right-clicking on the project and selecting *Build project*.

   *Note:* If the project compilation fails, make sure that the veins project is linked to the edca project (see 3.1 of TicToc Tutorial). The project should also have the same build configuration as used in Veins. To

[1] http://www.ccs-labs.org/teaching/c2x/2019s/edca-skeleton.tar.gz

make sure, confirm this in project *Properties* → *C/C++ Build* → *Manage Configurations* → choose the same build configuration as in Veins → *Set Active* → *OK* → *Apply and Close*.

In the skeleton project, you can find the following directories:

**simulations** contains the files required for setting up the scenario in which we conduct the performance evaluation.

**src/application** contains the `C2XApplication` C++ and NED files. This is the application which will be running on the RSUs and generating a network traffic for all 4 access categories.

**src/message** contains the message description file for `C2XMessage` which is used for communication between the RSUs which run the `C2XApplication`.

## 2.1 C2XApplication

The application provided in the skeleton code does not specify a particular behavior – no message exchange takes place. As a first step, let us implement the application layer. You can read the documentation in the header file of the application for the provided functions. We want the application to be generating packets of all priorities mapping to 4 different access categories, namely, `AC_VO`, `AC_VI`, `AC_BE` and `AC_BK`. We will use fixed packet size of 500B. Implement the inter-arrival time of packets for the different access categories such that it is exponentially distributed with mean value of 1ms.

*Note:* Referring to volatile[2] keyword will be helpful for configuring inter-arrival packet times.

There are several `Todos` and other informative comments in the `C++`, `.ned` and `.ini` files which will guide you through the implementation process.

## 2.2 Metrics

In this performance evaluation, we will be studying end-to-end delay and goodput. End-to-end delay is defined as the time difference between the packet reception and packet generation in the application layer. Goodput is defined as the application data that is transferred per second. You will have to think of a way to record these metrics and whether it should be recorded as a scalar or vector.

## 2.3 Understanding EDCA model in Mac1609_4

The RSUs are using *MAC1609_4* implementation from Veins in the simulations. If you want to look into some additional metrics within EDCA, you can also record them in MAC. For this, you will have to make changes in Veins. You can also check out some functions defined in nested class `EDCA`, e.g., `initiateTransmit()`, `startContent()`, `stopContent()`, `backoff()` for understanding the system model.

[2] https://doc.omnetpp.org/omnetpp/manual/#sec:ned-lang:volatile

## 2.4 Simplest Scenario

In the simplest scenario, we will consider only 2 RSUs. Out of them, only one RSU acts as a sender. In other words, the channel is completely available for the sender to transmit, i.e., there will be no interference.

Setup up the desired scenario with only 2 RSUs and configure the parameters in `omnetpp.ini` file for the following simulation configurations:

1. Packet generation for `AC_VO` only.
2. Packet generation for `AC_VI` only.
3. Packet generation for `AC_BE` only.
4. Packet generation for `AC_BK` only.
5. Packet generation for all access categories.

*Note:* Carefully read `todo` notes in `.ini` file.

## 2.5 Scalar and Vector Parsing Scripts

In *simulations/results/*, we also provide 2 scripts namely *opp_sca2csv.pl* and *opp_vec2csv_v2.pl*. As an alternative to *scavetool*, you can also use these scripts to parse the desired metrics from generated scalar and vector files. An example is given below:

Using the given perl scripts for parsing sca and vec files

```
# For parsing a scalar named SNIRLostPackets
perl opp_sca2csv.pl -m '^Scenario\.rsu\[(?<module>[0-9]*)\]' SNIRLostPackets -f ↘
    abc.sca > abc.csv
# For parsing a vector named goodputVO
perl opp_vec2csv_v2.pl -A repetition -A experiment -F goodputVO:vector abc.vec > ↘
    abc.csv
```

For further details, you can read the documentation using `perl opp_sca2csv.pl` or `perl opp_vec2csv_v2.pl`.

**Task 1**

Submit 2 plots:
1) end-to-end delay comparing each of the access categories (config 1-4 in Section 2.4) against the case when all 4 access categories are active (config 5 in Section 2.4).
2) goodput comparing each of the access categories (config 1-4 in Section 2.4) against the case when all 4 access categories are active (config 5 in Section 2.4).

Discuss the results briefly in less than 50 words.

*Note:* To understand certain patterns, you may also look at additional metrics relevant to EDCA which help you support your arguments.

### 2.6 Larger Scenario

In the next step, let us do a parameter study on the RSU count and the inter-packet arrival rate. You can study the behavior with RSU count of 10, 25 and 50, and the inter-arrival time for packets should be distributed exponentially with mean of 10ms and 5ms.

**Task 2**

Using the aforementioned parameter values, study the end-to-end delay and the application goodput in a larger and realistic scenario. Identify *another EDCA relevant metric* which is playing a crucial role affecting the trend as observed in plots from Task 1.

In your submission, include the plots for end-to-end delay, goodput and the *metric* that you identify. Discuss the results briefly.

*Note:* You may record the identified metric yourself in the MAC.

## Contact


Agon Memedi <memedi@ccs-labs.org>
Gurjashan Pannu <pannu@ccs-labs.org>

Course Website: http://www.ccs-labs.org/teaching/c2x/2019s/