Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.М04-мм

Разработка СПО прикладного уровня системы оплаты проезда российского производства на платных автодорогах

Пантелеймонов Андрей Радиевич

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
профессор кафедры системного программирования, д.ф.-м.н. А.Н. Терехов

Консультант:
Инженер-исследователь ООО «ЛИС», к.т.н. А.Г. Шадрин

Санкт-Петербург
2024

Оглавление

Введение

Первые платные дороги в России появились уже довольно давно - впервые на трассе М4 в 1998 году. С тех пор появились проекты, такие как ЗСД, которые требуют оплаты на всём участке пути. Постоянная оплата через оператора - человека сильно замедляет поток автомобилей и сокращает пропускную способность автодороги. Именно поэтому так важно наличие рабочей и безотказной системы оплаты проезда без участия человека.

По аналогии с банковскими картами, которые есть у любого человека, была придумана Система автоматического сбора пошлины (EFC, Electronic Fee Collenction), для работы который необходимо наличие транспондера или OBU (On-Board Unit, "устройства на борту") (аналога банковской карты) на лобовом стекле машины и RSU (Road-Side Unit, устройства на дороге), которое бы принимало и осуществляло транзакцию (аналог банкомата). Не так давно крупнейшая фирма, осуществлявшая полный цикл оплаты проезда такой системы Norbit ушла из России.

Таким образом появилась потребность в продукте, состоящем как из Программного обеспечения, так и аппаратного обеспечения, которое способно в полной объёме заменить разработки ушедшей норвежской фирмы.

Исходя из этого запроса, компании «Mobil-group» [1] и «ЛИС» [2] взялись за эту задачу, причём ответственными за ПО стала первая компания, а за создание АО — вторая.

Планируется, что продуктом будет физическое устройство с установленным на нём ПО, способным осуществить полный цикл обмена информацией и оплату с проезжающего мимо автомобиля с установленным на нём транспондером.

1 Постановка задачи

Цель работы — реализовать прикладной уровень выделенной радиосвязи ближнего действия

Задачи на осенний семестр:

- Изучить имеющуюся документацию предыдущих производителей на российском рынке
- Изучить соответствующие стандарты необходимые для разработки системы
- Реализовать выдачу информации в режиме реального времени для информирования оператора о текущем статусе в терминах описанных в документах протоколов
- Реализовать логирование работы системы для анализа ошибок и сбора статистики
- Приступить к реализации обмена информации между канальным и прикладным уровнями

Задачи на весенний семестр:

- Закончить реализацию обмена информации между канальным и прикладным уровнями
- Реализовать обмен информацией с биллинговой системой
- Провести апробацию продукта

2 Обзор

2.1 Обзор предметной области

2.1.1 Архитектура DSRC

Архитектура Выделенной Радиосвязи Ближнего действия (DSRC — Dedicated Short-Range Communication) состоит из нескольких компонентов:

- OBU (On-Board Unit) — транспондер или устройство, которое находится на транспортном средстве, служащее картой оплаты
- RSU (Road-Side Unit) — устройство, которое расположено на пути взимания оплаты проезда и осуществляющее обмен данными не только с транспондером, но и с внутренними сервисами, а также системой биллинга
- Application Layer Core — ядро прикладного уровня, которое отвечает за обмен информацией с канальным уровнем и приложением, а также ядром другого устройства (OBU или RSU)
- Data Link Layer (L2) — канальный уровень
- Physical Layer (L1) — физический уровень

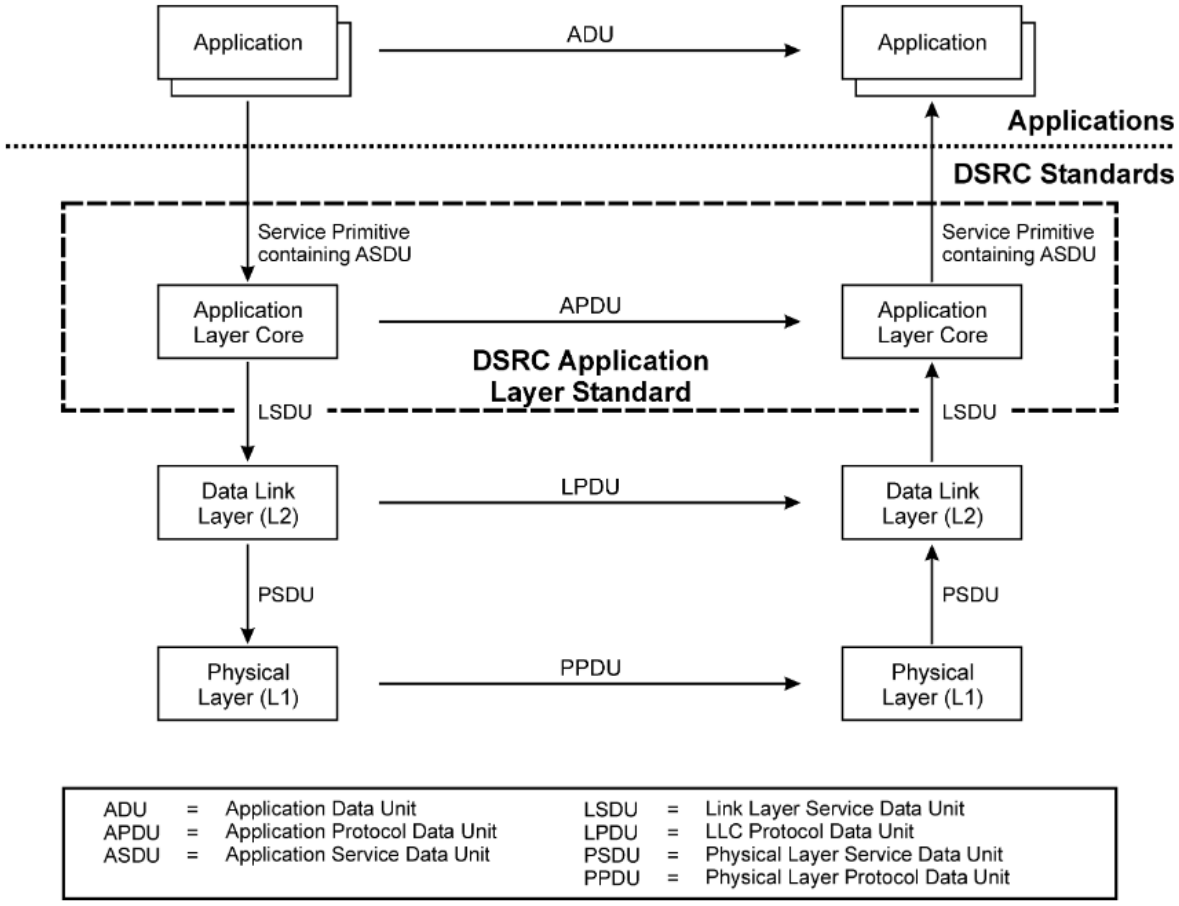

Рис. 1: Архитектура стека DSRC

2.1.2 Архитектура прикладного уровня DSRC

Архитектура прикладного уровня DSRC состоит из нескольких компонентов:

- I-Kernel — ядро инициализации, которое отвечает за инициализацию обмена информацией между RSU и OBU.
- T-Kernel — трансферное ядро, задача которого — обмен данными между канальным и прикладным уровнем и прикладным уровнем и приложением, а также с прикладным уровнем другой сущности (OBU или RSU)
- B-Kernel — ядро широколосной передачи, которое должно реализовывать сбор, широкую передачу и распространение информа-

ции для различных применений путём обмена через широковещательный пул

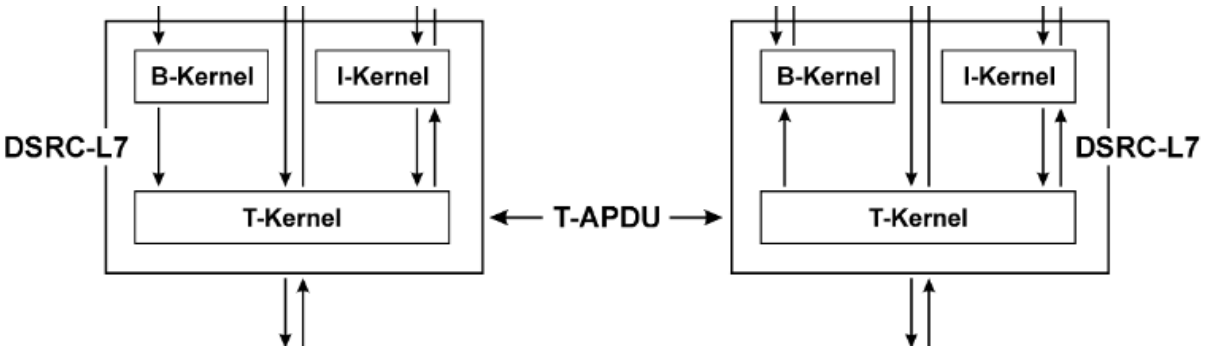

Рис. 2: Архитектура прикладного уровня DSRC

2.1.3 Обзор этапов обмена данными в ходе транзакции

2.1.3.1 BST

Первым этапом является BST - Beacon Service Table - данные, которые непрерывно посылает RSU по направлению к дороге, ожидая получения ответа от какого-либо OBU. На изображении ниже приведена его структура

Table B.11 — Initialisation request (BST) frame content

Octet #	Attribute/Field	Bits in Octet b_7 b_0	Description
1	FLAG	0111 1110	Start Flag
2	Broadcast LID	1111 1111	Link address for broadcast
3	MAC control field	1010 0000	The frame contains a command LPDU
4	LLC control field	0000 0011	UI command
5	Fragmentation header	1xxx x001	No fragmentation. PDU no shall never be set to 0000_2 or 0001_2.
6	BST SEQUENCE {	1000	INITIALISATION.request
	OPTION indicator	0	NonmandApplications not present.
	BeaconId SEQUENCE { ManufacturerId INTEGER (0..65535)	000	Manufacturer identifier- Example 1 (=Kapsch). See ISO 14816.
7		0000 0000	Register at www.tc278.eu/index.php/14816-registers for value assignment.
8		0000 1	
	IndividualId INTEGER (0..134217727)	000	27 bit ID available for manufacturer. Example: Id=1052_{10}
9		0000 0000	
10		0000 0100	
11	}	0001 1100	
12	Time INTEGER(0..4294967295)	0100 0001	32 bit UNIX real time. Example: 1103790512_{10}
13		1100 1010	
14		1000 0001	
15		1011 0000	
16	Profile INTEGER (0..127,...)	0000 0000	No extension, Profile. Example : Profile = 0

Рис. 3: Битовое представление BST

2.2 Обзор аналогов

В данном разделе приведен обзор системы компании Norbit, которая и стала отправной точкой для создания россйсеого аналога

1. ITS NORBIT (Система умного планирования дорожного трафика) [3] — программно-аппаратный комплекс, который осуществляет полную обработку данными в ходе транзакции между RSU и OBU

Norbit являлся мнополистом до ухода из России, поэтому его решения были взяты за основу для создания российского аналога

2.3 Выбор окружения для разработки

После обсуждения в команде разработчков было решено использовать язык C++ 11 версии, в виду необходимости быстрой работы с поступающими данными, а также необходимостью написания функциональности аппаратного обеспечения. В качестве ОС была выбрана Ubuntu 22.04 как ОС семейства GNU/Linux.

3 Реализация

3.1 Входные данные с канального уровня

С канального уровня на прикладной уровень поступает строка в hex формате, которую в дальнейшем необходимо преобразовать в бинарный формат, поскольку того требует спецификация стандартов, использованных при разработке. Пример реальной строки в hex формате:
100a002c 00600000 24760e71 c0039190 0001c101 02105700 01ff0070 02021dd1 0204118e 0d7bf301 00320100 00320100.
Здесь первые 2 октета - заголовок, а последний - незначащий, таким образом полезной нагрузкой являются 9 октетов. Для преобразования в бинарный формат используется следующая функция:

```
    std::string hextobin(const std::string &s)
{
    std::string out;
    for (auto &i : s)
    {
        uint8_t n;
        if (i ≤'9' and i ≥'0')
            n = i - '0';
        else
            n = 10 + i - 'A';
        for (int8_t j = 3; j ≥0; --j)
            out.push_back((n & (1 << j)) ? '1' : '0');
    }

    return out;
}
```

После этого строка преобразуется в строку из 0 и 1, после чего уже происходит её разбор согласно структуре, указаной на Рис.3

3.2 Выходные данные для протокола EARP

Протокол EARP (EFC Read Attribute Protocol - протокол чтения аттрибутов) требует специальный формат вывода, не соответствующий приходящему в него бинарному формату. Структура должна иметь следующий вид:

Part	Len	Description
Timestamp	19	Timestamp for the transaction, according to the RSUs internal clock
(space)	1	A single space character as delimiter
ContextMark	(40)	The OBU ContextMark, consisting of subfields:
CountryCode	3	The ISO-3166-1 country code of the OBU ContextMark
,	1	A single comma character as delimiter
OperatorId	5	The OperatorId part of the OBU ContextMark
,	1	A single comma character as delimiter
TypeOfContract	5	The TypeOfContract part of the OBU ContextMark
,	1	A single comma character as delimiter
ContextVersion	3	The ContextVersion part of the OBU ContextMark
:	1	A single colon character as delimiter
VST Parameter	(20)	DSRC application type dependent 'extra' information associated with the ContextMark. May be e.g. nonce value
(space)	1	A single space character
OBU-Info	(14)	Info about the OBU, as contained in the VST. consisting of the subfields:
EquipmentClass	4	The OBU EquipmentClass (in hexadecimal!)
/	1	Slash character as delimiter
ManufacturerId	4	The OBU ManufacturerId (in hexadecimal!)
/	1	Slash character as delimiter
ObeStatus	4	OBU Status (in hexadecimal!)
(space)	1	A single space character
Non-OBU Info	(28)	Information not read from the OBU, consisting of the subfields:
[	1	A left square bracket
ComputerNo	3	The number of the RSU performing the transaction
/	1	Slash character as delimiter

Part	Len	Description
Speed	6	The estimated speed of the vehicle, in km/h. Negative number means the vehicle is driving in the wrong direction. Speed is given with one decimal digit. Speed will always contain a sign (plus or minus).
/	1	Slash character as delimiter
Xpos	4	The estimated position of the OBU along the road
/	1	Slash character as delimiter
Ypos	4	The estimated position of the OBU across the road
]	1	A right square bracket
(space)	1	A single space character

Рис. 4: Структура данных в формате EARP

Пример вывода информации в протоколе EARP:

```
20100611T095008.513 578,00008,00001,001:--------------------
0000/002a/0000
[104/00645 +015.7/05.1/00.0]
AutoPASS <0EC7AC9DB0EA114C> |
099:021830C008CC3CDE140000C28422C6DF9C187A21EDF42A770F2B |
```

Рис. 5: Выходные данные

3.2.1 Время для протокола EARP

Как видно в примере выше, время необходимо не в самом типичном формате. Для создания строки, содержащее время в необходимом формате реализована следующая функция:

```
char *current_time()
{
    char *currentTime = (char *)malloc(sizeof(char) * 20);
    timeval curTime;
    gettimeofday(&curTime, NULL);
    int millisecs = curTime.tv_usec / 1000;

    char current_time_no_mill[16];
    strftime(current_time_no_mill, 16, ''%Y%m%dT%H%M%S'',
       localtime(&curTime.tv_sec));
    sprintf(currentTime, ''%s.%03d'', current_time_no_mill, millisecs);
    return currentTime;
}
```

Заключение

В результате работы были выполнены следующие задачи.

- Изучена имеющуюся документацию предыдущих производителей на российском рынке
- Изучены соответствующие стандарты необходимые для разработки системы
- Реализована выдачу информации в режиме реального времени для информирования оператора о текущем статусе в терминах описанных в документах протоколов
- Реализовано логирование работы системы для анализа ошибок и сбора статистики
- Реализована часть обмена информации между канальным и прикладным уровнями

Исходный код находится на локальном сервере Мобил-Групп на платформе Gitlab

Список литературы

[1] Mobil-Group. — URL: https://mobil-group.spb.ru/ (дата обращения: 2023-12-15).

[2] ЛИС. — URL: https://labics.ru/ (дата обращения: 2023-12-15).

[3] Норбит. — URL: https://norbit.com/its/products/ (дата обращения: 2024-01-08).