Национальный исследовательский ядерный университет «МИФИ»

Кафедра № 42 «Криптология и кибербезопасность»

Отчёт о выполнении полусеместровой лабораторной работы

24 октября 2020 г. Группа Б17-505

Бычков Георгий

Матвеев Владислав

Бондаренко Павел

Евсеенков Даниил

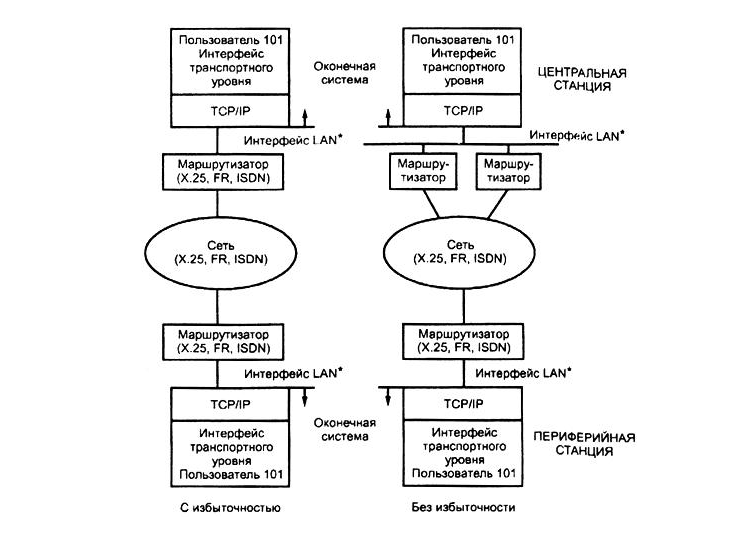
**Введение**

В конце восьмидесятых годов, в сфере телемеханики стали активно разрабатываться унифицированные открытые протоколы для устройств и систем автоматизации. Связано это было с тем фактом, что к тому времени существовало и активно использовалось большое количество несовместимых проприетарных протоколов от различных производителей. Что стало причиной возникновения серьезных проблем с совместимостью при проектировании и сопровождении систем. Кроме того, многие из существующих на тот момент протоколов имели большое число ограничений по емкости обрабатываемых данных, по типу передаваемых данных, отсутствовали средства диагностики качества передаваемых данных, не было возможности для расширения и пр. За рубежом процесс стандартизации начался раньше, и когда на наш рынок постепенно начали приходить зарубежные поставщики решений и оборудования, в качестве основы передачи телемеханической информации был взят стек протоколов IEC 870-5-101 и в 2001 году появился его отечественный перевод ГОСТ Р МЭК 870-5-101. А в 2004 году был переведен IEC 60870-5-104, получивший название ГОСТ Р МЭК 60870-5-104, являющийся расширением протокола 101 и регламентирующего использование сетевого доступа по протоколу TCP/IP. Данный протокол передачи данных, в настоящий момент, де-факто является стандартным протоколом диспетчеризации для предприятий электроэнергетического сектора.

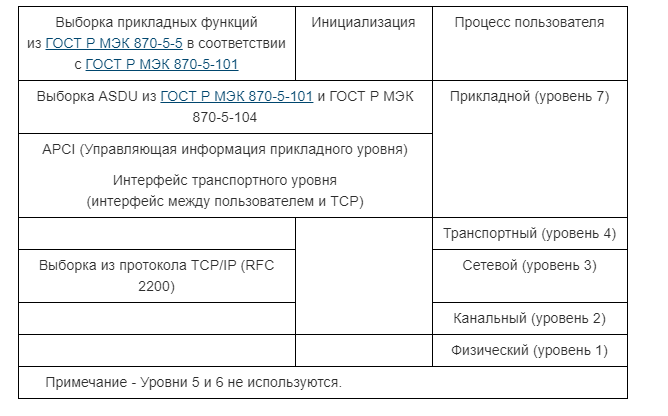
**Описание**

**Общая архитектура**

Обобщающий стандарт ГОСТ Р МЭК 60870-5-104 формализует инкапсуляцию блока ASDU из документа ГОСТ Р МЭК 870-5-101 в стандартные сети TCP/IP. Поддерживается как Ethernet, так и модемное соединение с использованием протокола PPP. Криптографическая безопасность данных формализована в стандарте МЭК 62351. Стандартный порт TCP 2404.

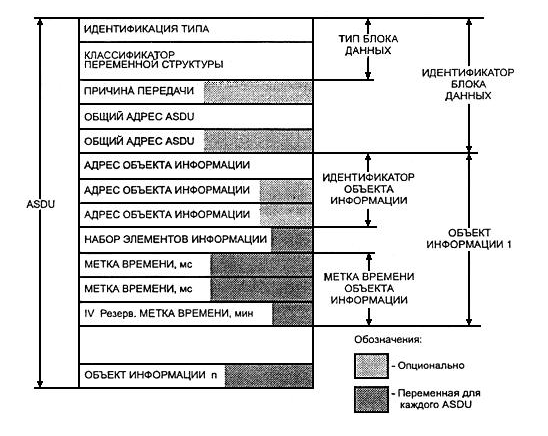
Данный стандарт определяет использование открытого интерфейса TCP/IP для сети, содержащей, например, LAN (локальная вычислительная сеть) для устройства телемеханики, которая передает ASDU в соответствии с МЭК 60870-5-101. Маршрутизаторы, включающие маршрутизаторы для WAN (глобальная вычислительная сеть) различных типов (например, Х.25, Фрейм реле, ISDN и т.п.), могут соединяться через общий интерфейс ТСР/IР-LAN. 

**Структура протокола**

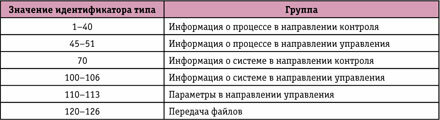
Протоколы серии 60870-5 основаны на трехуровневой модели, называемой “Архитектура повышенной производительности ”, определенной в МЭК 60870-5-3 и разработанной с целью получения более быстрого времени реакции для критической информации. Протоколы же в рамках МЭК 60870-5-104 дополнительно включают сетевой и транспортный уровни, основанные на выборке из TCP/IP в соответствии с RFC2200 и имеют следующую структуру: 

**Структура пакета данных**

ASDU – Блок данных, обслуживаемый прикладным уровнем. Он состоит из идентификатора блока данных и одного или более объектов информации. Идентификатор блока данных имеет всегда одинаковую структуру для всех ASDU. И все объекты информации, входящие в один ASDU, всегда имеют одинаковую структуру и тип, которые определены в поле “идентификация типа”.

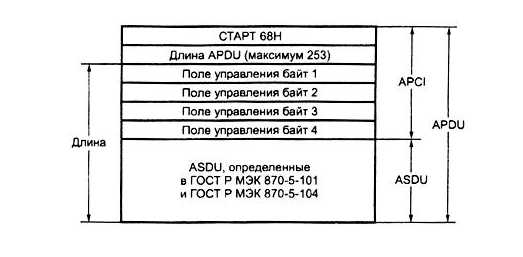


При приеме ASDU со значениями поля “идентификация типа”, не входящими в заранее определенный перечень, посылается отрицательная квитанция. Такие ASDU игнорируются как на ПУ, так и на КП. Стандарт МЭК 870-5-101 определяет диапазон значений от 1 до 127, также имея резервы для дальнейших совместимых определений. Диапазон от 128 до 255 не определяется. Значения “идентификация типа” от 136 до 255 могут быть определены независимо друг от друга пользователями. Однако возможность взаимодействия может быть получена только при использовании ASDU, имеющих значения поля “идентификация типа” в диапазоне от 1 до 127.



APCI – управляющая информация прикладного уровня.

ASDU – Блок данных, обслуживаемый прикладным уровнем.

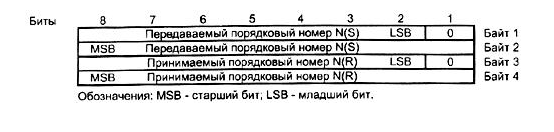
APDU – протокольный блок данных прикладного уровня. 

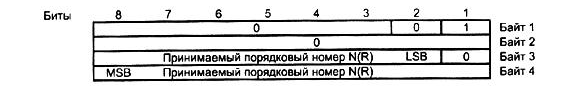
В протоколе МЭК 60870-5-104 процессы прикладного уровня взаимодействуют посредством обмена APDU – состоящего из двух элементов – ASDU и APCI. Для задания начала и конца ASDU каждый заголовок APCI включает следующие маркировочные элементы:

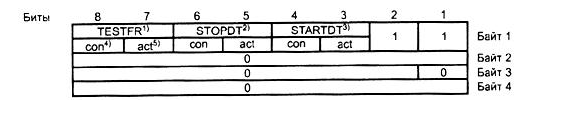
* стартовый символ – определяет точку начала внутри потока данных;
* указание длины APDU – определяет длину тела APDU, которое состоит из 4 байтов APCI плюс ASDU (максимум 249 байтов);
* поля управления – определяют управляющую информацию для защиты от потерь и дублирования сообщений, а также контроля транспортных соединений.

Может быть передан либо полный APDU, либо только APCI (для целей управления).

Возможны три формата поля управления: передача информации с нумерацией (формат I), контроль с нумерацией (формат S) и управление без нумерации (формат U).

Формат I определяется значением "0" первого бита первого байта поля управления. APDU формата I всегда содержит ASDU. Формат для передачи данных телеметрии.

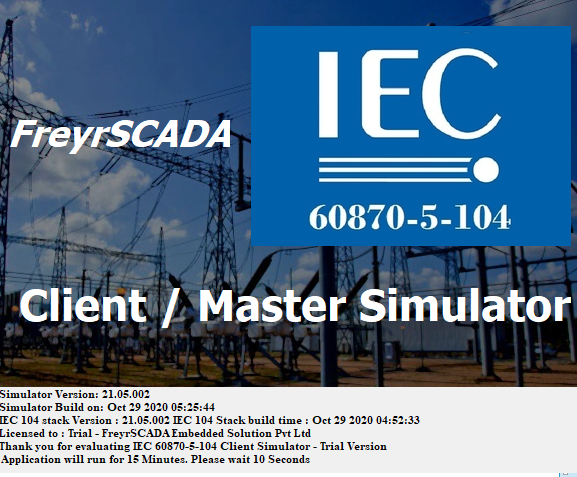
Формат S определяется значениями Бит 1 = 1 и бит 2 = 0 для первого байта поля управления. APDU формата S состоит только из APCI. Формат для передачи квитанций.

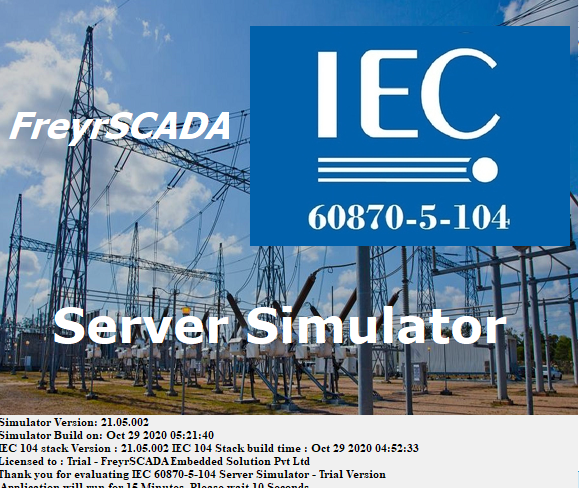
Формат U определяется значениями Бит 1 = 1 и бит 2 = 1 первого байта поля управления. APDU формата U состоит только из APCI. формат для передачи посылок установления связи и тестирования канала связи. Только одна из функций - TESTFR, STOPDT или STARTDT - может быть активной в данный момент.

1. TESTFR - Тестовый блок.
2. STOPDT - Прекращение передачи данных.
3. STARTDT - Старт передачи данных.
4. con - подтверждение.
5. act - активация.

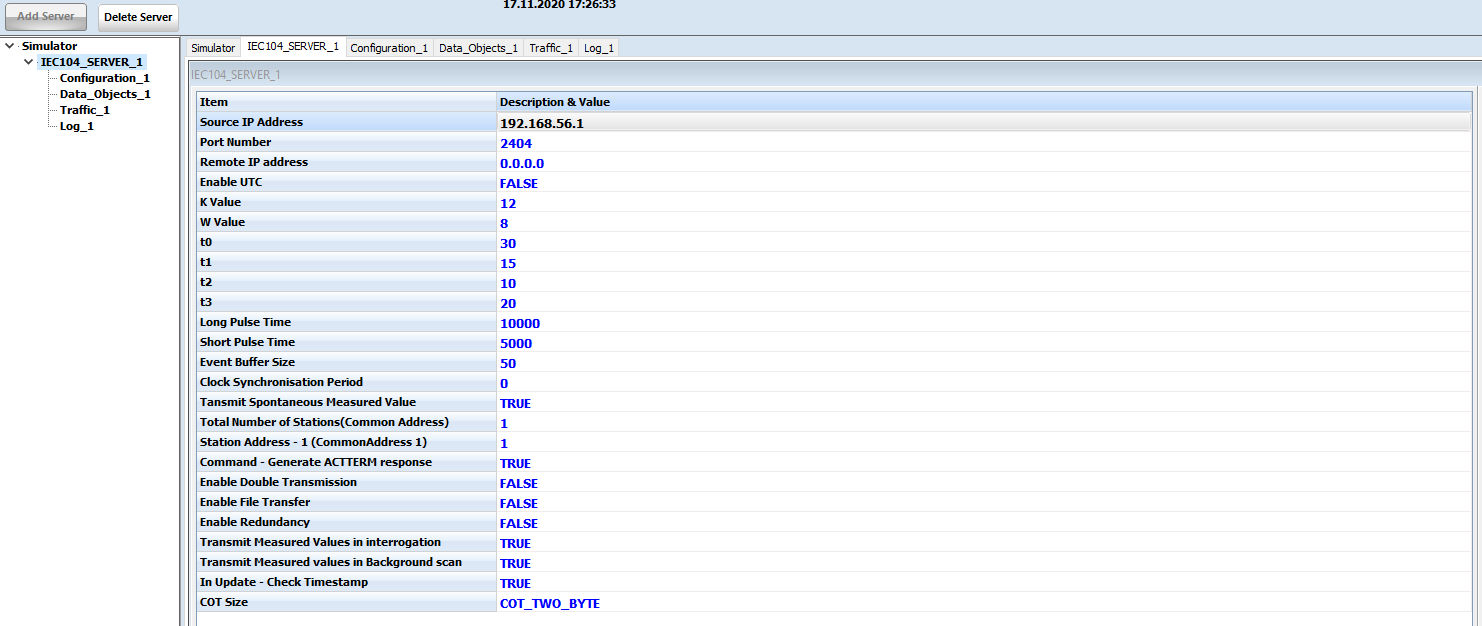
**Симультр клиент-серверного общения по протоколу IEC 68070-5-104**

Для создания структуры клиент-сервер воспользуемся уже готовым решением “FreyrSCADA IEC 68070-5-104 Server / Client simulator”

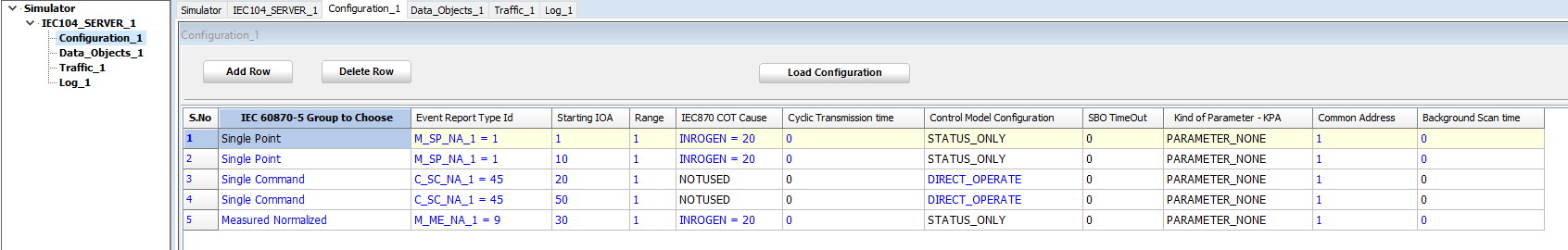




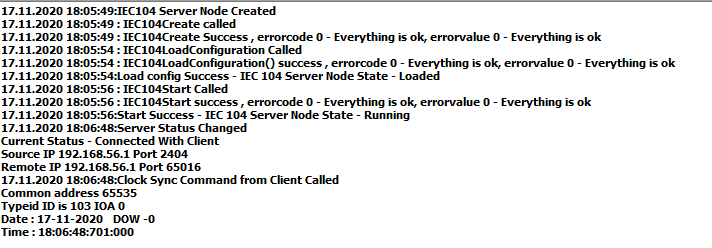
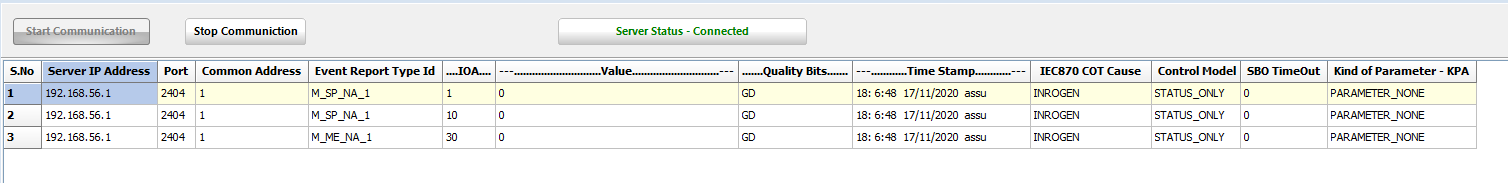
На экране создания сервера мы задаем его адрес. Остальные количественные и временные характеристики передачи можно оставить по умолчанию



В конфигурациях сервера указываем, какими объектами информации происходит обмен. Single point – команды на включение/выключение. Single command – передача команд. Measured Normalized – передача нормализованных измерений.



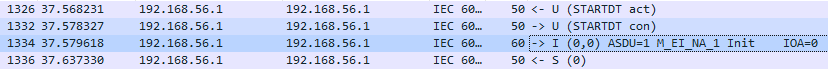
На экране создания клиента мы задаем только адрес запущенного сервера и активируем соединение.

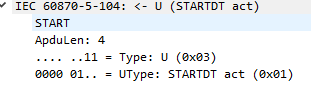


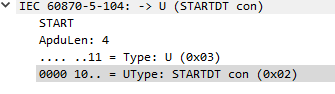
**Разбор пакетов в wireshark**

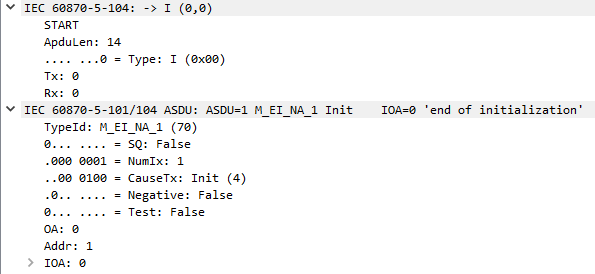
1. Подтверждение запроса на передачу данных (STARTD act/con).

* I-формат для передачи данных телеметрии;
* S-формат для передачи квитанций;
* U-формат для передачи посылок установления связи и тестирования канала связи.





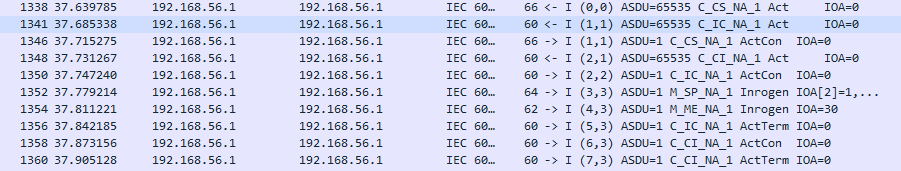




Первые три скрина – это APCI. На последнем кроме APCI также есть APDU.

1. Команды опроса - C\_IC ACT запрашивают полный объем или заданный определенный поднабор опрашиваемой информации. Команда опроса станции требует от контролируемых станций передать актуальное состояние их информации. Опрос станции используется для синхронизации информации о процессе на контролирующей станции и контролируемых станциях. Он также используется для обновления информации на контролирующей станции после процедуры инициализации или после того, как контролирующая станция обнаружит потерю канала (безуспешное повторение запроса канального уровня) и последующее восстановление его. Ответ на опрос станции должен включать объекты информации о процессе, которые запомнены на контролируемой станции.

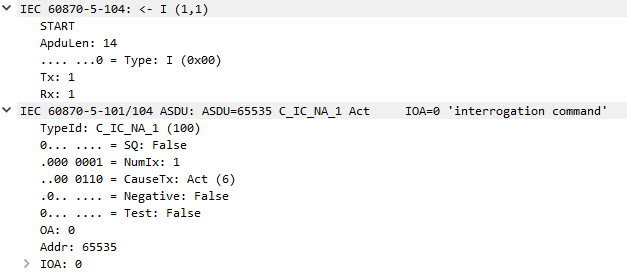




В ответ на <100> C\_IC\_NA\_1 необходимо ответить подтверждением на запрос, передать объекты информации о процессе, которые запомнены на контролируемой станции и завершить активацию.

Для отправки подтверждения необходимо в ASDU <100> C\_IC\_NA\_1 записать в байт указывающий на причину передачи (CauseTX) значение равное 7, для оправки завершения активации необходимо записать в байт указывающий на причину передачи (CauseTX) значение равное 10.

APDU блок формата S, состоящий только из APCI предназначен для подтверждения принятого APDU I формата. Для S-формата 7 старших бит служебного поля байта 1 и байт 2 не задействованы, а байт 3 (7 старших бит) и байт 4 определяют текущий номер принятой посылки



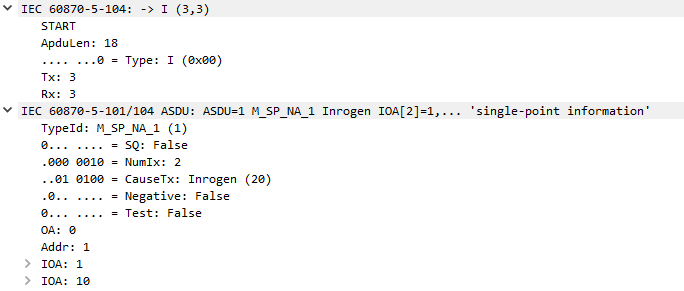
Вкратце можно сказать, что блок APCI определяет тип блока APDU и его длину. **APCI** состоит из следующих шести байтов:  
  
**1.** Признак инициализации блока APDU переменной длины, начинающийся байтом START2 68h;  
**2.** Длин APDU, в данном примере равна 14 байтам;  
**3.** Байт управления в котором определяется тип APDU  
**4,5,6** Не используются. Если пакет формата «I» то 3 байт в APCI будет также содержать значение младшего слова счетчика принятых пакетов,

**APCI:**

* В первом байте указан тип 0 означающий, что это команда опроса;
* Во втором длина APDU 14 байт;

**ASDU:**

* Первый байт в блоке ASDU определяет тип объекта информации, в данном случае <100> C\_IC\_NA\_1 (общий опрос станции);
* Второй структуру блока данных;
* Третий причину передачи (CauseTx), значение шесть означает запрос на активацию;
* Четвертый общий адрес стануии;
* Пятый адрес контролируемой (slave) станции;
* С шестого по восьмой адрес объекта информации равен нулю;
* Девятый информационный байт — QOI — описатель запроса, имеющий следующие значения:

Получив APDU подтверждение S или I формата от контролирующей станции можно начать передавать имеющиеся в распоряжении данные. 

**TypeId** — вид информации.  
**SQ** — классификатора переменной структуры.

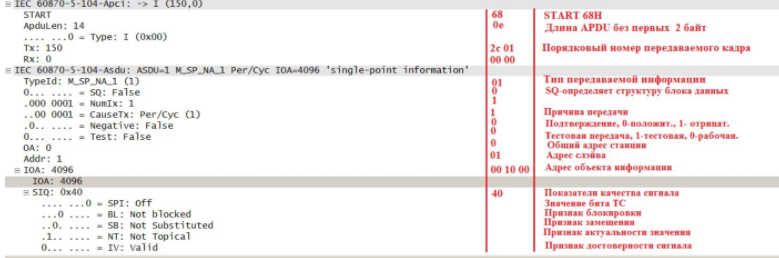
Предусматриваются две структуры блоков данных:  
  
1. Блок, содержащий i объектов информации, каждый из которых содержит по одному элементу информации (или по одной комбинации элементов); старший бит классификатора переменной структуры SQ (single/sequence) равен 0, остальные 7 битов задают число i.  
  
2. Блок, содержащий один объект информации, который содержит j элементов либо одинаковых комбинаций элементов информации; старший бит (27 = 80h) классификатора SQ равен 1, остальные 7 битов задают число j.

**CauseTx** — причина передачи.

**Negative –** подтверждение. False = 0 – положительная. True = 1 - отрицательная

**Test –** тестовая передача. 1 – тестовая, 0 - рабочая

**Addr** — адрес слэйва (указывается при конфигурировании мастера).  
**IOA** — адрес объекта информации, по этому адресу контролирующая станция будет привязывать свой тэг  
**SIQ** — показатель качества передаваемого сигнала.



**Установка сурикаты**

Установка сурикаты осуществлялась на систему Ubuntu LTS 18.04, из открытой репозитории гитхаба, с использованием следующих команд:

1. cd ~
2. sudo apt-get update && sudo apt-get upgrade -y
3. sudo apt-get install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev libcap-ng-dev libcap-ng0 make libmagic-dev libjansson-dev libnss3-dev libgeoip-dev liblua5.1-dev libhiredis-dev libevent-dev liblz4-dev m4 autoconf autogen cargo python3-pip cbindgen
4. sudo pip install python-snap7
5. sudo pip install --upgrade suricata-update
6. git clone https://github.com/OISF/suricata.git
7. cd suricata
8. git clone https://github.com/OISF/libhtp
9. sudo ./autogen.sh
10. sudo ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var
11. sudo mkdir /var/log/suricata
12. sudo mkdir /etc/suricata
13. sudo make && sudo make install && sudo make install-conf
14. sudo cp suricata.yaml /etc/suricata
15. sudo suricata-update -D /etc/suricata
16. sudo ifconfig lo mtu 1522

**Настройка сурикаты**

Для включения и работы с системой использовались следующие команды:

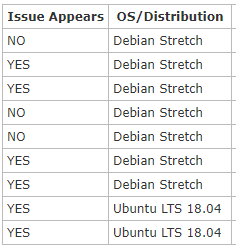
1. sudo systemctl status suricata
2. sudo vim /etc/suricata/suricata.yaml (Изменить интерфейл на lo)
3. sudo touch /etc/suricata/rules/custom.rules
4. sudo tail /var/log/suricata/suricata.log
5. sudo cat /var/log/suricata/fast.log
6. sudo suricata -c /etc/suricata/suricata.yaml -s /etc/suricata/rules/custom.rules -i lo

В конфигурационном файле ‘/etc/suricata/suricata.yaml’ был установлен интерфейс, в котором происходит отлов пакетов – lo. А в директории с правилами ‘/etc/suricata/rules/’ созданы файлы правил, используемых в рамках работы:

1. alert tcp any any <> any any (msg: "TEST any") – тестовое правило, реагирует на все пакеты;
2. alert tcp any any <> any any (msg: "Cause = Act"; content: "|06|"; offset: 52; depth: 53; rawbytes;) – предупреждение о пакете с определенной причиной передачи;
3. alert tcp any any <> any any (msg: "Detect single command"; content "|01|"; offset: 51, depth: 52; rawbytes;) – предупреждение о пакете определенного типа;
4. alert tcp any any <> any any (msg: "START"; content: "STARTDT";) - предупреждение о стартовых пакетах;
5. alert tcp any any <> any any (msg: "Working"; content: "TESTFR";) – предупржедение о тестовых пакетах, которыми обмениваются клиент-сервер;
6. alert tcp any any <> any any (msg: "type S"; content: "|01|", offset: 46; depth: 47; rawbytes;) – предупреждение о пакетах определенного формата
7. alert tcp any any <> any any (msg: "type id": content: "|01|", offset: 50; depth: 51; rawbytes;) - предупреждение о пакетах определенного формата
8. alert tcp any any <> any any (msg: "strange command"; content: "command";) – предупреждение о нестандартном пакете
9. alert tcp any any <> any any (msg: "valuev"; content: "|d0 05|"; offset: 59; depth: 61; rawbytes;) – предупреждение о пакете с определенным содержимым

**Работы сурикаты**

Во время выполнения работы с сурикатой, наша группа столкнулась с проблемой потери **части (не всех)** alertov в результате ошибки ‘stream/packet on wrong thread’. После проведенного исследования было выяснено, что эта ошибка связана с операционной системой, на которой производилась работа, и с режимами работы самой программы. Самостоятельно устранить данную ошибку нашими силами не представляется возможным.



**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

<http://docs.cntd.ru/document/gost-r-mek-60870-5-104-2004>

<http://docs.cntd.ru/document/1200019789>

<http://lapshinvr.ru/articals/sobytiynye-protokoly.html>

<https://yandex.ru/turbo/isup.ru/s/articles/4/618/>

<https://habr.com/ru/post/167327/>

<https://m.habr.com/ru/post/280818/>

<https://redmine.openinfosecfoundation.org/issues/2725>

<https://suricata.readthedocs.io/en/suricata-4.1.2/performance/runmodes.html>