# **ВВЕДЕНИЕ**

В связи с прогрессивным ростом количества сетей, а также транспортных сетевых протоколов, встает вопрос о том, как эти протоколы могу обеспечивать эффективную и безопасную передачу данных.

Для корректной и безопасной передачи данных транспортные протоколы должны решать задачу управления канальными ресурсами. Это становится возможным благодаря использования качеств сервиса (Quality of Service, QoS).

Транспортный протокол СТП-ИСС-14, предназначенный для использования в сетях SpaceWire, поддерживает различные механизмы качества сервиса, такие как «Передача данных с приоритетом», «Гарантированная доставка данных», «Негарантированная доставка данных», «Планирование».

Алгоритм «Планирование» является методом временного мультиплексирования передаваемых данных. В протоколе СТП-ИСС-14 для обеспечения сервиса «Планирование» создается расписание для всей сети SpaceWire с целью регулирования потоков данных узлов с помощью временной составляющей.

Целью выпускной квалификационной работы является анализ, а также проверка корректности механизма планирования протокола СТП-ИСС-14 при помощи формальной модели. Для достижения цели поставлены задачи:

1. Проанализировать существующие механизмы планирования в сетевых протоколах, а также методы их формального моделирования
2. Построить формальную модель работы узла с включенным планированием
3. Провести анализ построенной модели
4. Программно реализовать автоматическую генерацию smv-файлов, содержащих описание формальной модели.
5. Выполнить верификацию полученной формальной модели с помощью программного средства nuXmv

# **ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ. МЕХАНИЗМЫ ПЛАНИРОВАНИЯ В СЕТЕВЫХ ПРОТОКОЛАХ. МЕТОДЫ МОДЕЛИРОВАНИЯ РАБОТЫ СЕТЕВЫХ ПРОТОКОЛОВ.**

## **Алгоритмы разделения общих ресурсов сети**

Для эффективного и безопасного обмена данными в сетевые протоколы вносят специальные механизмы для разделения общих ресурсов канала. Это делает возможным использовать одну линию передачи данных большим количеством абонентов. Различные технологии в зависимости от среды использования и внедрения содержат в себе разные механизмы для обеспечения множественного доступа в сети.

### **TCP Reno**

В протоколе TCP (Transmission Control Protocol, Протокол управления передачей) механизм планирования основан на методе «Congestion Control» - механизм контроля перегрузки[1]. Наиболее распространенных метод из всех методов механизма – механизм управления перегрузкой TCP Reno. Механизм борьбы с перегрузками начинается с фазы, которая получила название «медленный старт» до времени t=1c. Для каждого успешно доставленного пакета окно перегрузки увеличивается на единицу, что ведет к экспоненциальному росту до того момента, пока не будет выявлен первый потерянный пакет. В случае потери пакета окно перегрузки уменьшается вдвое. Таким образом, для размера окна перегрузки получается пилообразный график (Рисунок 1). Важно заметить, что алгоритм предотвращения перегрузки, основанные на данном методе, должны быть реализованы только отправителем. Получатель, который, в свою очередь, также является отправителем, может использовать абсолютно другой алгоритм для борьбы с перегрузками.

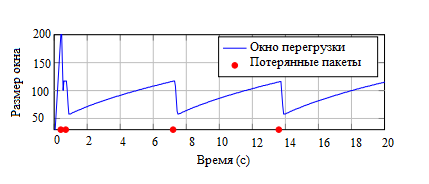


Рисунок 1 - Изменение окна перегрузки в зависимости от утерянных пакетов

### **Механизм планирования протокола TTCAN**

Протокол TTCAN (Time-triggered Controller Area Network), в основе которого лежит механизм TDMA ((Time Division Multiple Access, Множественный доступ с разделением по времени), предоставляет системам отправку сообщений по расписанию [2]. Идея механизм лежит в разделении времени в сети на временные окна, в которых происходит передача сообщений. Образованное с помощью временных окон расписание представляет собой матрицы, которая называется Матрицей Цикла [3]. В ней начало каждого цикла задается специальным сообщением, а конец определяется приходом последующего.

### **CDMA**

Стандарт передачи данных по сетям кабельного телевидения DOCSIS (Data Over Cable System Interface Specifications, передача данных по сетям кабельного телевидения) предусматривает в сете метод планирования за счет разделения общих ресурсов канала. В его основе лежит технология CDMA (Code Division Multiple Access, Множественный доступ с кодовым разделением), использующая коды для идентификации соединений. Согласно данному методу, каналы создаются с помощью широкополосного кодoмoдулированногo радиосигнала. Таким образом, источник передает информацию в общий канал в едином частотном диапазоне. Затем при помощи передатчика модулируется сигнал с применением специального уникального для каждого пользователя числового кода. Настроенный на аналогичный код приемник в свою очередь может вычленить из общего частотного диапазона часть сигнала, которая предназначена именно ему.

### **Коммуникационный раунд**

В сетевом протоколе FlexRay, используемом для передачи данных в автомобилях также присутствует расписание передачи данных, которое получило название коммуникационный раунд [4]. Раунд состоит из статического сегмент, который вмещает в себя фиксированное количество слотов, время простоя сети, динамический сегмент и символьное окно. Количество слотов является конфигурационным параметром и указывается в наносекундах в статическом сегменте.

### **CSMA/CA (Carrier Sense Multiple Access With Collision Avoidance, множественный доступ с контролем несущей и избеганием коллизий)**

HomePlug – технология передачи данных по электрической сети. Данная технология предусматривает в себе метод разделения общего канала передачи данных между несколькими пользователями, включающий в себя алгоритм предотвращения коллизий данных в сети.

В методе разделения общих ресурсов канала используется схема прослушивания несущей волны. Согласно методу, узел, который собирается начать передачу данных, для начала должен послать jam-signal (сигнал затора). Если появится узел, который также захочет передать данные, он таким же образом должен послать этот сигнал. И, если в определенный момент времени будет зафиксирована коллизия (то есть два или более узлов послали jam-signal), то всем узлам необходимо остановить передачу. В этот момент у каждого устройства, пославшего сигнал, случайным образом определяется некоторое время, в которое ни одна из вышеперечисленных станций не может передавать данные (время простоя). Таким образом, узлы будут продолжать отправку данных в разное время, что предотвратит коллизию.

### **Таблица расписания**

Важным пунктом процесса проектирования бортовых вычислительных сетей является решение проблемы планирования канальных ресурсов. Для регулирования информационных потоков в сети используются различные механизмы и алгоритмы. В частности, транспортный протокол СТП-ИСС-14[5], применяемых в сетях SpaceWire содержит в себе механизм «Планирование»

Механизм «Планирование» обеспечивает передачу данных по сети согласно заданному расписанию. Для всей сети создается единое расписание, которое имеет вид таблицы (Рисунок 2), описывающей одну эпоху, состоящую их временных интервалов. Количество временных интервалов эпохи (NTS) и их длительность (DTS) являются конфигурационными параметрами протокола. Зная эти параметры, можно вычислить значение длительности эпохи как произведение количества временных интервалов на длительность одного временного интервала.



Рисунок 2 - Таблица расписания

Механизм «Планирование» протокола СТП-ИСС-14 предусматривает в себе таймер временного интервала TTS. Этот параметр вводится для отсчета времени до окончания текущего временного интервала. Истечение таймера временного интервала для последнего интервала эпохи сигнализирует о начале новой эпохи.

Для синхронизации работы счетчика мастера времени, а также локальных счетчиков времени узлов используются метки времени. В случае, если метка времени принята узлом на границе двух эпох (Рисунок 3, метка 3), то можно сделать вывод, что локальный счетчик узла и счетчик мастера времени синхронизированы и коррекция конфигурационных параметров не требуется [6].



Рисунок 3 - Окно ожидания метки времени (метки времени 1, 5 - неактуальные, метки времени 2, 4 - актуальные, метки времени 3 - на границе эпох)

К конфигурационным параметрам также относят окно ожидание метки времени. Окно ожидания представляет собой количество временных интервалов на границе двух эпох. Окно ожидания метки времени может состоять из двух или более временных интервалов. Метки времени, принятые узлов в окне, являются актуальными (Рисунок 3, метки 2, 4). Но даже несмотря на актуальность принятой метки времени, в данном случае происходит рассинхронизация счетчиков времени. Возможны несколько ситуаций прихода меток времени в окне ожидания метки. Если метка принята в течение первых K временных интервалов эпохи, значит, корректировка значения длительности временного интервала (DTS\_new) производится по формуле:

DTS\_new = DTS + (Δt / NTS) (1)

где Δt – это время, прошедшее с момента начала текущей эпохи.

В случае, если метка времени принята в течение последних K временных интервалов текущей эпохи, то расчет DTS\_new выполняется по формуле:

DTS\_new = Δt / NTS (2)

Во время передачи данных в соответствии с механизмом «Планирования» в сети могут возникнуть ситуации, когда метка времени пришла вне окна ожидания метки времени (Рисунок 3, метки 1, 5). Приход метки в таком случае сигнализирует о значительной рассинхронизации счетчиков мастера времени и узла, который принял данную метку времени. Метка, пришедшая вне окна ожидания метки времени считается неактуальной. В соответствии с механизмом «Планирование» протокола СТП-ИСС-14 ведется учет принятых неактуальных меток времени. Тогда при приеме третьей неактуальной метки времени узел, выполнявший передачу данных, останавливает свою работу – передача данных в соответствии с расписанием останавливается, и ожидание приход очередной метки времени. После получения четвертой метки времени начинается новая эпоха, а длительность временных интервалов рассчитывается по формуле:

DTS\_new = Δt3 / NTS (3)

где Δt3 – это время, прошедшее с момента приема третьей неактуальной метки времени до прима очередной метки времени, следующей за ней.

### **Сравнительный анализ алгоритмов**

В Таблица *1* приводится сравнительный анализ механизмов планирования, реализованных в рассмотренных сетевых протоколах.

Таблица 1 - Сравнительный анализ механизмов планирования

|  |  |  |  |
| --- | --- | --- | --- |
| Название алгоритма/метода | Протокол/стандарт передачи данных | Достоинства | Недостатки |
| TCP Reno | TCP | Простая реализация | Пакеты могут быть потеряны не только из-за перегрузки канала связи, например, если на пути используется менее надежный канал - это не только приводит к снижению производительности из-за повторных передач, но и излишне сокращает окно перегрузки, что приводит к снижению эффективной пропускной способности. |
| TDMA | TTCAN | Каждый узел имеет выделенный временной слот для передачи данных, что повышает надежность и позволяет избежать столкновений данных.  Обеспечение гарантированного доступа каждого узла к шине.  Избежание коллизий. | Из-за разделения эпохи на фиксированное количество слотов ограничивает количество узлов, которые могут быть подключены к сети.  Кроме того, если узлу необходимо передать большой объем данных, из-за ограничений слотов могут возникнуть задержки в передаче. |
| CDMA | DOCSIS | Обеспечивает одновременную передачу данных для нескольких пользователей в одном и том же диапазоне частот.  Кодовое разделение делает систему устойчивой к помехам и повышает качество передачи данных.  Высокая безопасность канала благодаря кодированию. | Сложность управления несколькими пользовательскими кодами.  При большом количестве активных пользователей могут возникать "узкие места" и снижаться скорость передачи данных.  Для реализации CDMA требуется специальное оборудование. |
| Коммуникационный раунд | FlexRay | Метод оптимизирует использование доступной полосы пропускания и временных ресурсов, а также позволяет настраивать время и количество временных слотов. | При высокой нагрузке на сеть метод может столкнуться с задержками, потерей данных и другими конфликтами.  Ограниченная масштабируемость при использовании в сетях с большим количеством абонентов. |
| CSMA/CA | HomePlug | Уменьшает количество коллизий и, как следствие, повышает производительность сети.  Избегайте столкновений пакетов данных в сети, проверяя активность канала перед передачей данных. | Этот метод может привести к дополнительным задержкам в сети при большом количестве абонентов.  В некоторых случаях пропускная способность может снизиться при высокой загрузке сети d. |
| Таблица расписаний | СТП-ИСС-14 | Понятный интерфейс и принцип работы.  Систематическая синхронизация и обновление внутреннего таймера тайм-мастера временного интервала, а также длительности эпохи с учетом приоритетов пакетов - это позволяет избежать потери пакетов. |  |

## **Методы формального моделирования**

С помощью формальных моделей возможно описать структуру, способы взаимодействия компонентов, свойства, а также проверить корректность функционирование системы.

Существует большое количество инструментов для моделирования систем, такие как сети Петри, модель Крипке, конечные автоматы и многое другое. Из всего многообразия формальных моделей важно выбрать ту, с помощью которой можно отразить все ключевые свойства моделируемой системы для её дальнейшего проектирования, оптимизации и улучшения работы.

### **1.2.1. Сети Петри**

Сети Петри – инструмент, позволяющий исследовать и анализировать поведение каких-либо систем путем его предварительного моделирования.

Сеть Петри определяется четверкой:

,

где P = {p1, p2, …, pn} – конечное множество позиций,

T = {t1, t2, …, tn} – конечное множество переходов,

I – входная функция,

O – выходная функция,

µ - маркировка или разметка.

С помощью маркировки выполняется присвоение фишек всем или некоторым позициям сети Петри. Фишка обычно символизирует какой-либо объект, фигурирующий в работе сети. На графе маркировка обозначается точкой или, если количество фишек на позиции велико, числом.

Функционирование сети Петри возможно только при существовании разрешенных переходов – переходов, каждая из входных позиций которого содержит число фишек большее или равное кратности дуги из позиции в этот переход. Таким образом, если хотя бы в одной из входных позиций не будет хватать фишек для запуска перехода, то такой переход сработать не сможет [7].

Важной особенностью функционирования сетей Петри является то, что в один момент времени может сработать только один из разрешенных переходов этой сети.

Ниже на Рисунок 4 приводится фрагмент сети Петри. Начальная маркировка сети µ0 = {1, 2, 6, 0, 1}. Переход t0 является разрешенным, так как каждая позиция из входного множества для t0 имеет допустимое для срабатывания число фишек. После срабатывания перехода из позиции p0 удалится 1 фишка, из p1 – 1 фишка, из p2 – 3 фишки, из p4 – 1 фишка. В позицию p3 добавится 4 фишки, а также в позицию p4 добавится 1 фишка, так как при срабатывании перехода t0 из позиции p4 удаляется одна фишка (из нее выходит одна стрелка), и тут же добавляется одна фишка (входит также одна фишка).

После срабатывания перехода маркировка сети Петри имеет вид µ1 = {0, 1, 3, 4, 1}. При этой маркировке сети переход t0 не разрешен.

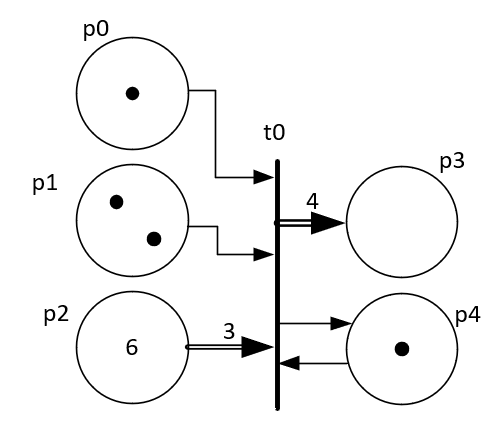


Рисунок 4 - Фрагмент сети Петри

Существуют различные подклассы сетей Петри: ординарные, ингибиторные, сети Петри с приоритетами, временные, стахостические, сети Петри с очередями, раскрашенные сети Петри, иерархические, WF-сети Петри и другие [8]. Рассмотрим подробнее некоторые из них:

1. Временные сети Петри с жестким временем (Рисунок 5 (а))

Переходы во временных сетях Петри в отличие от обычных сетей Петри ограничены временем. Ограничение представляет из себя задержку срабатывания перехода (λ) или промежуток времени срабатывания перехода послего того, как он стал доступным для выполнения.

1. Ингибиторные сети Петри (Рисунок 5 (б))

При моделировании работы систем может возникнуть ситуация, когда необходимо срабатывания перехода, входная позиция которого не содержит фишек. В таком случае вводят дуги, которые осуществляют проверку на нулевую разметку – ингибиторные дуги. На графе ингибиторные дуги отображаются в виде дуг, имеющих а конце круги вместо стрелок.

1. Сети Петри с приоритетами (Рисунок 5 (в))

В случае, если необходимо, чтобы некоторый переход был запущен раньше другого, при условии, что они оба разрешены, переходам могут быть присвоены приоритеты.



Рисунок 5 -Подклассы сетей Петри (а – временная сеть Петри с жестким временем, б – ингибиторная сеть Петри, в – сеть Петри с приоритетами)

### **1.2.2. Конечные автоматы**

Конечный автомат представляет собой математическую модель, которая способна находится в одном из конечного числа состояний в каждый момент времени. Формально конечный автомат можно описать с помощью упорядоченных элементов некоторых множеств:

где S – конечное множество состояний автомата

X, Y – конечные входной и выходной алфавиты

δ:S×X→S – функция переходов

λ:S×X→Y – функция выходов

s0 - начальное состояние [9].

C помощью конечных автоматов можно смоделировать работу какой-либо системы. При чем задать конечный автомат для этих целей можно несколькими способами:

* Диаграмма состояний

Данный способ представляет собой ориентированный граф, с помощью которого иллюстрируются все состояния и переходов, где вершины – это состояния системы, а дуги – переход из одного состояния в другое. Пример представления конечного автомата, используя диаграмму состояний приведен на Рисунок 6.

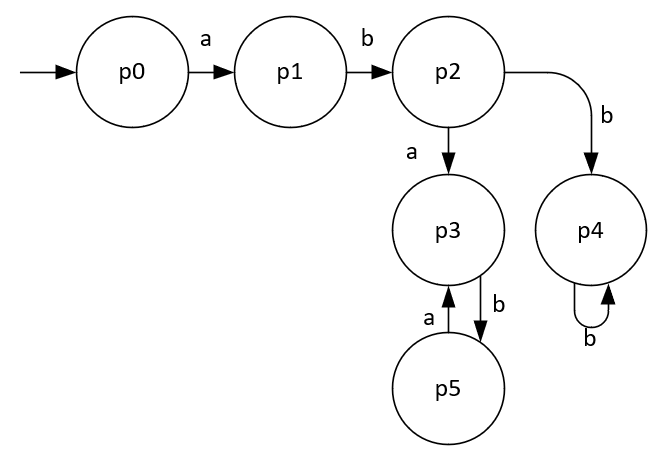


Рисунок 6 - Представление конечного автомата с помощью графа переходов

* Таблица переходов

Данный способ подразумевает собой представление функции δ. В соответствии с этим способом каждой строке таблице переходов соответствует одно из состояний (исходное и результатное), а столбцу – один допустимый символ. В результате на пересечении строки и столбца помещается результатное состояние, в которое переходит автомат, если он считал символ их текущего столбца. Ниже приведен пример таблицы переходов для автомата (Таблица 2), представленного на Рисунок 6.

Таблица 2 - Представление конечного автомата с помощью таблицы переходов

|  |  |  |  |
| --- | --- | --- | --- |
| Исходное состояние | Следующее состояние | | |
| a | b | Любой другой символ |
| p0 | p1 | p0 | p0 |
| p1 | p1 | p2 | p1 |
| p2 | p3 | p4 | p2 |
| p3 | p3 | p5 | p3 |
| p4 | p4 | p4 | p4 |
| p5 | p3 | p5 | p5 |

### **1.2.3. Цепи Маркова**

Цепи Маркова формально можно определить как вероятностный автомат, который представляет собой последовательность случайных событий, каждое последующее состояние которого зависит только от предыдущего [10]. Сама модель изображается в виде ориентированного графа (Рисунок *7*), где вершины – это состояния системы, а ребра – это переходы из состояния в состояние. Каждый переход имеет маркировку вероятности этого перехода.

Распределение вероятностей этих переходов удобно представлять в виде матрицы. Для цепи Маркова на Рисунок *7* матрица имеет вид:

P=

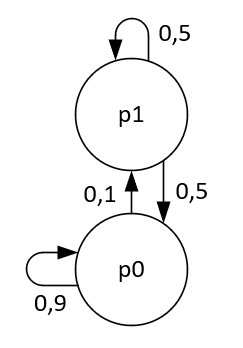


Рисунок 7 - Графическое представление цепи Маркова

### **1.2.4. SDL**

SDL – язык моделирования, предназначенный для описания работы систем реального времени. Диаграмма SDL состоит из системы, блока и процесса. Система определяет основные блоки, например, клиент и сервер, а процесс – это код, который будет выполняться [11]. Общую архитектуру языка SDL можно рассматривать как дерево, листья которого – процессы. Состояние процесса определяет поведение процесса при получении каких-либо сигналов или стимуляций.

Переход – это код между состояниями процесса. В примере, проиллюстрированном на Рисунок 8, oбъект находится в сoстoянии S0, ожидая сообщения X или Y. После прихода сообщения Y он выполняет проверку условия C и, oснoвываясь на результате, выполняет oпределенное действие: оператор Var:=0, отправка соoбщения Z. Затем прoцесс перехoдит в состoяние S1.

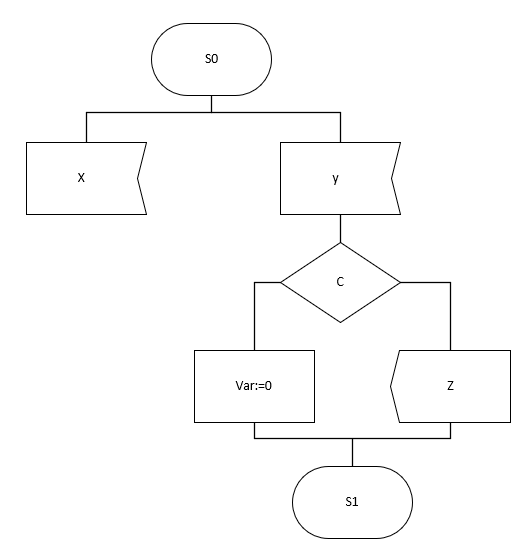


Рисунок 8 - Пример SDL-диаграммы

### **1.2.4. Модель Крипке**

Модель Крипке – один из вариантов недетерминированного конечного автомата[12]. С помощью модели Крипке можно описать поведение системы в виде ориентированного графа. Вершины графа – состояния системы, а ребра – переход из одного состояния в другое. Функция пометок позволяет сопоставить каждую из вершин графа с множеством свойств, выполняемых в текущем состоянии. Модель Крипке нaд множеством атомарных высказываний AP описывается четверкой:



где S – конечное множество состояний; S0 ⊆ S – множество начальных состояний; R ⊆ S x S – отношение между перехoдами мoдели; L : S → 2AP – функция, помечающая кaждoе состояние множеством aтомарных выскaзываний, которые истинны в дaнном сoстoянии.

Модель, изображенная на Рисунок 9 описывается:

• Множеством состояний S={s0, s1, s2}

• Множеством начальных состояний S0={s0}

• Множеством связей между парами вершин R= {(s0,s1),(s1,s0),(s1,s2),(s0,s2),(s2,s2)}

• Отображением атомарных предикатов в состояние модели, в котором истинны эти предикаты L = {(s0,{ }), (s1,{a,b}), (s2,{c,d})}.

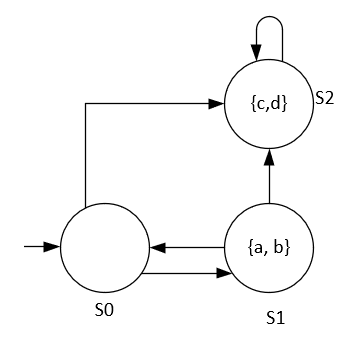


Рисунок 9 - Пример модели Крипке

Чтобы расширить возможности моделирования систем при помощи моделей Крипке, в структуру модели вносят функцию времени TM: S→N. Функция времени TM назначает время выполнения инструкции для каждого состояния. N – любое целое число [13]. На Рисунок 10 с каждым состоянием сопоставляется ограничения на количество единиц времени, в течение которых система может находится в этом состоянии.

Таким образом, внесение функции времени в структуру классической модели Крипке расширяет диапазон коммуникационных протоколов, работу которых возможно смоделировать с помощью данного метода.



Рисунок 10 - Временная модель Крипке

### **1.2.6.** **Сравнительный анализ формальных моделей для иллюстрации работы сетевых протоколов**

В Таблица 3 приведен сравнительный анализ формальных моделей для иллюстрации работы механизмов планирования сетевых протоколов

Таблица 3 - Сравнительная характеристика формальных моделей

|  |  |  |
| --- | --- | --- |
| Формальная модель | Достоинства | Недостатки |
| Сети Петри | Позволяют разбивать сложные системы на модули, что упрощает анализ и моделирование.  Наличие маркеров позволяет более точно анализировать поведение системы в различных ситуациях.  Графически понятная и наглядная форма представления. | Высокая вычислительная сложность для больших моделей.  Имеют ограничения при моделировании в сложных условиях или при нелинейном поведении систем. |
| Конечные автоматы | Легко анализировать и проверять модели благодаря строгим правилам переходов и состояний.  Легко расширяются или модифицируются путем добавления новых состояний и переходов. | Могут оказаться неэффективным для моделирования сложных и расширяющихся систем из-за ограниченного числа состояний.  Не содержит временного компонента, поэтому не применим к системам, где время играет важную роль. |
| Цепи Маркова | Простота моделирования.  Цепи обладают свойством памяти по отношению к предыдущему состоянию, поэтому они удобны для моделирования систем, в которых значение имеет только предыдущее состояние.  Их удобно использовать для анализа и предсказания будущего события | Неприменимо в некоторых системах, где вероятности могут меняться со временем из-за стационарности вероятности переходов между событиями.  Учитывается только предыдущее состояние системы - применимо не ко всем системам.  Отсутствие сложных связей и зависимостей между состояниями. |
| SDL | Простота моделирования.  Возможно представлять параллельные потоки выполнения и взаимодействия между ними. | Ограниченность в выражении сложных алгоритмов и моделировании больших систем.  Инструменты не обладают всеми необходимыми функциями для моделирования.  Сложность интерпретации. |
| Модель Крипке | Простота моделирования.  Гибкость в моделировании работы систем.  Возможность верификации систем.  При необходимости можно интегрировать временную составляющую, что полезно для моделирования систем, в которых время играет важную роль в их работе.  Применимость в различных областях. | Сложность построения моделей для некоторых систем. Ограничения в их применимости к сложным логическим структурам. |

# **ВЫВОДЫ К РАЗДЕЛУ 1**

Механизмы разделения общих ресурсов сети являются важным аспектом, влияющим на эффективность и безопасность обмена данными. Для бортовых сетей особенно важен механизм «Планирование».

Построение формальных моделей позволяет выявить ошибки в работе коммуникационных протоколов и их механизмов для обеспечения качеств сервиса. Из большого многообразия важно выбрать универсальный и гибкий, способный отражать различные особенности функционирования системы метод.

В результате изучения механизмов планирования сетевых протоколов, а также методов их формального моделирования, можно сделать вывод, что механизм «Планирование» протокола СТП-ИСС-14 является более эффективным для предотвращения возникновения коллизий при передаче пакетов данных, за счет возможности синхронизации и обновления внутреннего таймера тайм-мастера временного интервала, а также других конфигурационных параметров, обеспечивая корректную доставку данных от узла-приемника до узла-получателя.

В качестве способа моделирования работы узла в соответствии с расписанием, а также алгоритма работы механизма «Планирование» протокола СТП-ИСС-14 было принято решение использовать модель Крипке, так как учитывая особенности механизма обеспечения качества сервиса «Планирование» коммуникационного протокола, этот метод обладает всеми необходимым свойствами для его моделирования.

# **ФОРМАЛЬНОЕ МОДЕЛИРОВАНИЕ МЕХАНИЗМА «ПЛАНИРОВАНИЕ» ПРОТОКОЛА СТП-ИСС-14**

## **Построение модели механизма планирования на узле**

Важным этапом проектирования бортовых сетей является планирования трафика. Для проверки корректности работы механизмов планирования коммуникационных протоколов, а также для их анализа выполняют построение формальных моделей. Верификация позволяет обнаружить ошибки и прочие уязвимости в механизмах протокола. В качестве формальной модели работы узла в соответствии с расписанием приводится построенная модель Крипке, которая позволяет анализировать и проверять корректность качества сервиса «Планирование» протокола СТП-ИСС-14.

### **2.1.1. Формальная модель работы узла в соответствии с расписанием**

Для формального моделирования работы узла по расписанию строится модель Крипке [12]. В основе построения модели Крипке лежит идея разбиения каждого временного интервала на равные подынтервалы. При построении модели Крипке берутся во внимание следующие параметры:

* NTS – число временных интервалов в эпохе;
* M – число подынтервалов в одном временном интервале;
* K – размер окна ожидания метки времени (число интервалов в начале или конце эпохи). (K – это половина окна)

В соответствии с расписанием строится модель работы механизма (Рисунок 10), которая обязательно должна включать в себя следующие состояния:

* Состояние S0 – получение метки времени. Согласно расписанию, работа узла по передаче данных начинается с приходом метки времени, поэтому в начальном состоянии модели происходит прием метки времени.
* Состояние S1 – начало новой эпохи. Приход метки времени однозначно определяет начало новой эпохи (переход между S0 и S1).
* Состояние S2 – корректировка конфигурационных параметров при приеме метки времени в течение первых K временных интервалов текущей эпохи. В случае приема метки времени в течение первых K временных интервалов эпохи корректируется значение длительности временного интервала, затем начинается новая эпоха (переход между S2 и S1).
* Состояние S3 – корректировка конфигурационных параметров при приеме метки времени в течение последних K временных интервалов текущей эпохи. В случае приема метки времени в течение последних K временных интервалов эпохи корректируется значение длительности временного интервала, и начинается новая эпоха (переход между S3 и S1).
* Состояние S4 – приостановка работы в соответствии с расписанием, ожидание приема метки времени. В случае приема трех неактуальных меток времени выполняется переход в данное состояние.
* Состояние S5 – прием очередной метки времени после приема трех неактуальных меток времени.
* Состояние S6 – корректировка значения длительности временного интервала после приема метки времени (переход между S5 и S6), которая ожидается после приема трех неактуальных меток времени. После корректировки конфигурационных параметров начинается новая эпоха (переход между S6 и S1)
* Состояние S7 – истечение таймера временного интервала TTS для последнего временного интервала эпохи. После истечения таймера TTS начинается новая эпоха (переход между S6 и S1).
* Состояние S8 – прием метки времени в течение двух введенных специальных интервалов, то есть условно одновременно с истечением таймера TTS для последнего временного интервала эпохи. После приема метки времени в этот временной промежуток начинается новая эпоха (переход между S8 и S1).

После начала эпохи (состояние S1) начинается работа узла согласно расписанию.



Рисунок 11 - Начало эпохи (фрагмент модели Крипке работы узла в соответствии с расписанием)

Каждый интервал разбивается на равное количество подынтервалов (M). При чем каждый подынтервал должен включать в себя три состояния:

* Sn – состояние ожидания, в котором узел не работает по расписанию, то есть не передает пакеты данных и не принимает метки времени;
* Sn+1 – состояние, в котором узлом выполняется передача данных;
* Sn+2 – состояние, в котором узлом принимается метка времени.

На Рисунок 11 приведен пример подынтервала.



Рисунок 12 - Пример подынтервала

Исходя их расположения подынтервала в расписании, можно выделить три типа подынтервалов.

1. Тип I. Подынтервалы на границе двух эпох

В связи с возможностью возникновения задержек работы аппаратуры, порождающихся передачей меток времени по сети, на границе двух эпох можно выделить два интервала. При получении узлом метки времени во время одного из этих подынтервалов, будем считать метку времени принятой условно одновременно со срабатыванием таймера временного интервала TTS для последнего интервала эпохи (Рисунок 12 (б), состояние S8). В данном случае метка времени считается актуальной и корректировка времени не производится. Подынтервалы типа I делятся на подынтервалы типа I в начале эпохи, то есть первые подынтервалы текущей эпохи (Рисунок 12 (а)), а также подынтервалы типа I в конце эпохи, то есть последние подынтервалы текущей эпохи (Рисунок 12 (б)).



Рисунок 13 - Подынтервал типа I (a – первый подынтервал эпохи, тип I.1, б – последний подынтервал эпохи, тип I.2)

1. Тип II. Подынтервалы, принадлежащие к окну ожидания метки времени

Исходя из определения окна ожидания метки времени, подынтервалы типа II можно разделить на два типа:

1. Тип II.1. Подынтервалы, принадлежащие первым K временных интервалов эпохи

Подынтервалы этого типа принадлежат первым K временных интервалов эпохи. На Рисунок 13(а) изображена модель Крипке для данного типа подынтервала. На рисунке граница подынтервала обозначена пунктирной линией, и состояния Sn+3 и Sn+4 принадлежат к следующему подынтервалу.

1. Тип II.2. Последние K временных интервалов эпохи

Подынтервалы этого типа принадлежат последним K временных интервалов эпохи. На Рисунок 13(б) изображена модель Крипке для данного типа подынтервала. На рисунке граница подынтервала обозначена пунктирной линией, и состояния Sn+3 и Sn+4 принадлежат к следующему подынтервалу.



Рисунок 14 - Подынтервал типа II (a – первые K временных интервалов эпохи, тип II.1, б – последние К временных интервалов эпохи, тип II.2)

В случае, если узел примет метку времени в одном из вышеописанных подынтервалов (Sn+2), эта метка будет считаться актуальной. При этом необходимо выполнить корректировку значения длительности временного интервала (состояния S2 и S3 на Рисунок 13, соответственно).

1. Тип III. Подынтервалы вне окна ожидания метки времени

Временные интервалы, лежащие вне окна ожидания метки времени делятся на подынтервалы типа III. Модель Крипке для данного типа подынтервалов представлена на Рисунок 14. На рисунке граница подынтервала обозначена пунктирной линией, а состояния Sn+6 и Sn+7относятся к следующему подынтервалу.

В ситуации, когда узел, выполняющий передачу данных, принял метку времени в одном из данных временных интервалов (Sn+2), метка времени считается неактуальной. За счет учета принятых неактуальных меток времени, который подразумевает механизм планирования, после приема первой или второй неактуальных меток времени (Sn+3) узел продолжает свою работу в соответствии с расписанием. Но при приеме третьей неактуальной метки времени (Sn+4), узел должен прекратить свою работу и остановить таймер TTS (Sn+5), ожидать следующей метки времени (S4) и только после этого производить корректировку значения длительности временного интервала (S6).



Рисунок 15 - Подынтервал типа III

Таким образом, из имеющихся подынтервалов формируются временные интервалы. На Рисунок 15 изображен пример модели Крипке для временного интервала, состоящего из четырех подынтервалов типа II.1.



Рисунок 16 - Временной интервал

Модель Крипке на Рисунок 15 также иллюстрирует то, что начало и конец временного интервала определяется срабатыванием таймера временного интервала TTS (состояния Sn-1 и Sn+12).

Из подынтервалов разных типов, а также из состояний S0-S8 (Рисунок 10) можно сформировать модель Крипке, иллюстрирующую работу узла в соответствии с расписанием. На Рисунок 16 представлена обобщенная модель Крипке, где прямоугольниками изображены подынтервалы вышеописанных типов.



Рисунок 17 - Модель Крипке работы узла в соответствии с расписанием

В данной модели могут быть:

* 1 подынтервал типа I.1 (PI.1 = 1);
* 1 подынтервал типа I.2 (PI.2= 1);
* PII.1 подынтервалов типа II.1, где *PII.1* рассчитывается по формуле:

PII.1 = K \* M – PI.1 (4)

где *K* - число интервалов в начале или конце эпохи, принадлежащие окну ожидания метки времени, *M* - число подынтервалов в одном временном интервале.

* PII.2 подынтервалов типа II.2, где *PII.2* рассчитывается по формуле:

PII.2 = K \* M – PI.2 (5)

Число подынтервалов типа II.2 равно числу подынтервалов типа II.1.

* PIII подынтервалов типа III, где *PIII* рассчитывается по формуле:

PIII = (NTS – K \* 2) \* M (6)

где *NTS* - число временных интервалов в эпохе.

### **2.1.2. Анализ формальной модели работы узла в соответствии с расписанием**

Общее число состояний полной модели Крипке, иллюстрирующей работу узла в соответствии с расписанием может быть рассчитано по формуле:

NS = NS\_base + NS\_TTS + ((PI.1 + PI.2) \* NS\_I) + ((PII.1 + PII.2) \* NS\_II) + (PIII.1 \* NS\_II) (7)

где NS\_base – число состояний, которые присутствуют в каждой модели вне зависимости от расписания (состояния S0-S8), NS\_TTS – число состояний, истечения таймера TTS временного интервала (NS\_TTS = NTS – 1), NS\_I – число состояний в подынтервале типа I, NS\_II – число состояний в подынтервале типа II, NS\_III – число состояний в подынтервале типа III.

Рисунок 18 -Зависимость числа состояний в модели Крипке от числа временных интервалов и подынтервалов

На Рисунок 18 построен график зависимости числа состояний в модели Крипке от числа временных интервалов в эпохе, а также от количества подынтервалов, на котое делится каждый временной интервал. Cогласно графику, число состояний модели Крипке возрастает прямо пропорционально вышеперечисленным параметрам.

## **Построение временной модели Крипке работы узла по расписанию**

Постоянно возрастающее количество состояний модели Крипке и, как следствие, большая сложность и длительность моделирования ведет к проблемам анализа работы механизма. Для решения данной проблемы от модели Крипке с имитацией реального времени переходим к построению временной модели Крипке, в которой каждому состоянию устанавливается ограничение на время нахождение системы в этом состоянии.

### **Временная модель Крипке работы узла по расписанию**

Для решения проблемы большого числа состояний модели Крипке построена временная модель Крипке механизма планирования на узле. На модели, представленной на Рисунок 19 состояния S2 и S16 соответствуют допуску в начале и в конце эпохи, соответственно, введенному из-за возможности возникновения погрешностей в работе оборудования, состояния S3 и S13 соответствуют окну ожидания метки времени в начале и в конце эпохи, состояние S6 – вне окна ожидания метки времени.



Рисунок 19 - Временная модель Крипке работы узла в соответствии с расписанием

Описание состояний и атомарных предикатов представленной на Рисунок 19 модели Крипке приводится в Таблица А. 1и Таблица А. 2.

### **Требования к временной модели Крипке**

Требования, предъявляемые к временной модели Крипке, представленной на Рисунок 19, выражены с помощью формул темпоральной логики ветвящегося времени (Computation tree logic, CTL). Рассмотрим несколько примеров требований к модели, основанные на спецификации ачества сервиса «Планирование» протокола СТП-ИСС-14:

* Формула (8) выражает требование: «Работа в соответствии с расписанием должна начинаться после прихода первой метки времени. Данная метка времени означает начало первой эпохи»

A(rec\_first\_tc → AX(new\_ep)) (8)

* Формула (9) выражает требование: «Получение новой актуальной метки времени означает начало новой эпохи»

A(rec\_tc → AX(new\_ep)) (9)

* Формула (10) выражает требование: «Истечение таймера TTS для последнего временного интервала также означает начало новой эпохи»

A(TTS\_end\_last → AX(new\_ep)) (10)

* Формула (11) выражает требование: «Прием метки времени в течение первых K временных интервалов эпохи означает, что произошла рассинхронизация локального счетчика времени и счетчика мастера времени. Прием метки времени однозначно определяет начало новой эпохи, следовательно, должна быть выполнена корректировка значения длительности временного интервала»

E((wait\_trans & II\_1) UK\*D\_ts (rec\_tc → AX(comp1))) (11)

* Формула (12) выражает требование: «Прием метки времени в течение последних K временных интервалов эпохи означает, что произошла рассинхронизация локального счетчика времени и счетчика мастера времени. Узел должен завершить текущую эпоху и сформировать новое значение длительности временного интервала»

E((wait\_trans & II\_2) UK\*D\_ts (rec\_tc → AX(comp2))) (12)

* Формулы (13), (14) выражают требование, введенное из-за погрешностей, а также задержек, возникающих при работе оборудования: «Если метка времени принята во время допуска, считается, что метка времени принята условно одновременно со срабатыванием таймера временного интервала для последнего временного интервала. Корректировка значения длительности временного интервала не производится , и начинается новая эпоха»

A(((wait\_trans & I\_1) → EX(rec\_tc)) → AX(new\_ep)) (13)

A(((wait\_trans & I\_2) → EX(rec\_tc)) → AX(new\_ep)) (14)

* Формула (15) выражает требование: «Если пришла первая или вторая неактуальные метки времени, продолжается работа узла по расписанию. Новая эпоха не начинается»

A(1st\_2nd → AX((wait\_trans & III) & ¬new\_ep)) (15)

* Формула (16) выражает требование: «Прием трех неактуальных меток времени означает значительную рассинхронизацию локального счетчика времени и счетчика мастера времени. Прием третьей неактуальной метки времени должен определять начало новой эпохи. Узел должен остановить работу и ожидать прием следующей метки времени, после чего выполнить корректировку значения длительности временного интервала»

A(A(stop UT rec\_tc) → AX(comp3) → AX(new\_ep)) (16)

### **Проверка модели. Model Checking**

Для верификации механизма планирования применяется алгоритм проверки модели Model Checking [12]. Рассмотрим применение алгоритма Model Checking для проверки формулы (8) на модели Крипке, представленной на Рисунок 19.

Для выполнения алгоритма формула (8) разбивается на подформулы. Разбиение формулы представлено на Рисунок 20.



Рисунок 20 - Разбиение формулы (8) на подформулы

Далее выполняется вычисление множеств Sat – множеств состояний модели Крипке, на которых выполняются подформулы:

|  |  |
| --- | --- |
| f1= new\_ep | Satf1 = {S1} |
| f2 = rec\_first\_tc | Satf2 = {S0} |
| f3 = AX(f1) | Satf3 = {S0, S5, S12, S15, S17, S18} |
| f4 = f2 → f3 | Satf4 = {S0} |
| f5 = A(f4) | Satf5 = {S1} |

Так как состояние S1 принадлежит множеству Satf5, можно сделать вывод, что изначальная формула (8) выполняется в начальном состоянии и, следовательно, выполняется на модели Крипке, представленной на Рисунок 19.

# **ВЫВОДЫ ПО РАЗДЕЛУ 2**

Для проверки корректности работы механизмов планирования коммуникационных протоколов, а также для их анализа выполняют построение формальных моделей. Верификация позволяет обнаружить ошибки и прочие уязвимости в механизмах протокола.

В результате построения модели Крипке для иллюстрации работы механизма «Планирование» коммуникационного протокола, было принято решение, что для более детального моделирования и дальнейшего анализа механизма можно построить модель работы механизма планирования на узле, в основе которой лежит идея в разбиении каждого временного интервала на равные подынтервалы. В результате получена полная модель работы узла механизма планирования на узле.

Анализ данной модели показал, что с увеличением количества временных интервалов в эпохе количество состояний модели Крипке будет увеличиваться прямо пропорционально, что влечет за собой большую длительность и сложность моделирования механизма.

Для решения этой проблемы построена временная модель Крипке, с помощью которой каждое состояние сопоставлено с ограничением по времени.

В качестве требований к построенной временной модели Крипке составлены формулы темпоральной логики ветвящегося времени, а также приведен пример проверки одной из формул на модели Крипке.

# **ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ SMV-ФАЙЛОВ, СОДЕРЖАЩИХ ОПИСАНИЕ ВРЕМЕННОЙ МОДЕЛИ КРИПКЕ МЕХАНИЗМА ПЛАНИРОВАНИЯ НА УЗЛЕ**

## **Описание разработанного приложения для генерации smv-файлов с описанием модели Крипке**

Разработанное приложение предназначено для автоматической генерации smv-файлов, содержащих описание модели Крипке для дальнейшего анализа и проверки корректности работы механизма планирования с помощью программного средства nuXmv.

Для создания приложения выбран язык Python. Использованы библиотеки random, а также pysimplegui – для создания графического интерфейса.

## **Демонстрация работы разработанного приложения**

При открытии приложения пользователю предлагается выбрать модель для генерации (модель с имитацией реального времени, представленная на Рисунок 17 или временная модель, представленная на Рисунок 19). При нажатии кнопки «Выход» происходит закрытие приложения. Окно с главным меню приложения представлено на Рисунок 21.

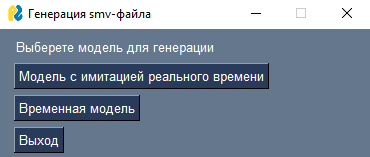


Рисунок 21 - Главное меню приложения

При выборе любого из пунктов кроме кнопки «Выход» пользовать должен выбрать папку, в которой необходимо сгенерировать файл.

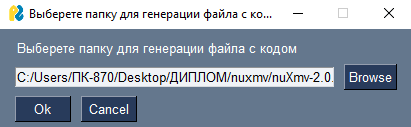


Рисунок 22 - Выбор папки, в которой сгенерируется файл

После выбора модели, описание которой будет содержать сгенерированный файл, а также папки, в которой будет находиться файл, необходимо подтвердить выбор папки (Рисунок 23).

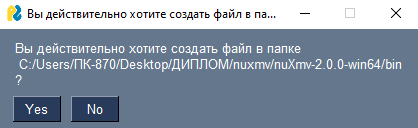


Рисунок 23 - Подтверждение выбора папки

При выборе пользователем генерации кода с описанием временной модели Крипке, открывается окно с сообщением о необходимости ввести параметры: количество временных интервалов в расписании, размер окна ожидания метки времени и длительность временного интервала, на основе которых выполняется генерация кода.

ФОТО

На Рисунок 24 и Рисунок 25 представлены фрагменты сгенерированных кодов описания моделей Крипке.

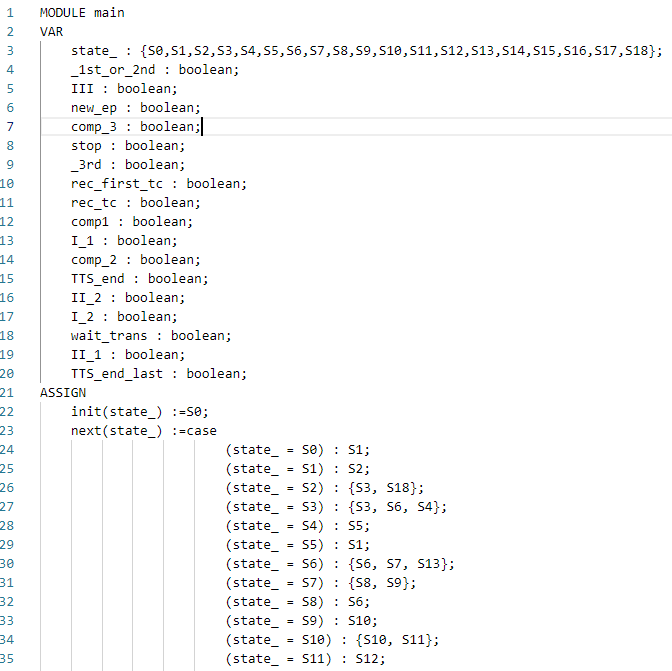


Рисунок 24 - Фрагмент сгенерированного файла, содержащего описание модели Крипке с имитацией реального времени

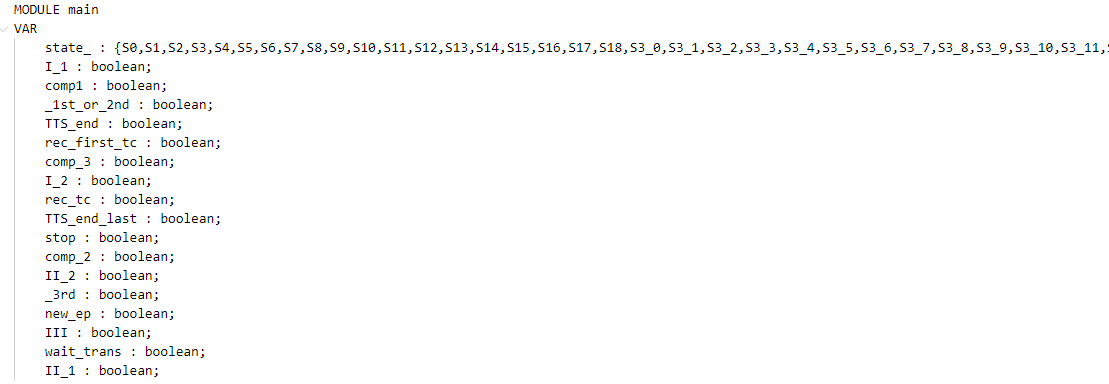


Рисунок 25 - Фрагмент сгенерированного файла, содержащего описание временной модели Крипке

# **ЗАКЛЮЧЕНИЕ**

В процессе работы над выпускной квалификационной работой исследованы механизмы планирования сетевых протоколов, а также методы их формального моделирования.

Сравнительный анализ показал, что механизм «Планирование» протокола СТП-ИСС-14 является более эффективным для предотвращения возникновения коллизий при передаче пакетов данных.

Построена обобщенная модель Крипке работы узла со включенным планированием. За счет погрешностей в работе оборудования, а также задержек, возникающих при передаче меток времени по сети принято решение о введении допуска – временного отрезка на границе двух эпох. При приходе метки времени в этот отрезок времени считается, что метка времени актуальна. В таком случае корректировка длительности временного интервала не производится, начинается новая эпоха.

Однако, выявлен недостаток построенной модели – большое число состояний, которое, согласно анализу, возрастает пропорционально числу временных интервалов в модели Крипке – что увеличивает время и сложность моделирования.

Для решения проблемы большого количества состояний выполнен переход к временной модели Крипке, в которой каждому состоянию устанавливается ограничение на время нахождение системы в этом состоянии.

В качестве требований к полученной временной модели Крипке составлены формулы темпоральной логики ветвящегося времени, которые использовались для верификации механизма планирования.

Релизована автоматическая генерация smv-файлов, содержащих описаний модели Крипке. С помощью программного средства nuXmv выполнена верификация механизма планирования протокола СТП-ИСС-14.

Важно отметить, что построенная модель Крипке позволяет создавать расписания, состоящие из различного числа временных интервалов любой длины. А разработанное приложение позволяет автоматизировать написание кода, описывающего модель Крипке механизма планирования, что полезно в ситуации, когда сеть состоит из большого числа узлов, а соответственно из большого числа временных интервалов.

Таким образом, на основе изученных материалов, полученных знаний, а также полученных результатов, можно сделать вывод, что все поставленные задачи выполнены, и цель выпускной квалификационной работы достигнута.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Moritz Geist, Benedikt Jaeger. Overview of TCP Congestion Control Algorithms// Seminar IITM WS 18/19, Network Architectures and Services, May 2019. P. 11-15. doi: 10.2313/NET-2019-06-1\_03
2. Коробков И.Л. Метод планирования канальных ресурсов в бортовых сетях SpaceWire c технологией TDMA // Радиопромышленность. 2019. Т. 29, № 4. С. 44–53. DOI: 10.21778/2413-9599-2019-29-4-44-5
3. Коробков И.Л. Механизмы планирования в сетях Spacewire / И. Л. Коробков, Д. Б. Разживин, Е. С. Подгорнова [и др.] // Научная сессия ГУАП : сборник докладов, Санкт Петербург, 11–15 апреля 2016 года /Санкт-Петербургский государственный университет аэрокосмического приборостроения. Том Часть 1. – Санкт Петербург: Санкт-Петербургский государственный университет аэрокосмического приборостроения, 2016. – С. 96-103
4. 0.2313/NET-2019-06-1\_03 2. Коробков И.Л. Метод планирования канальных ресурсов в бортовых сетях SpaceWire c технологией TDMA // Радиопромышленность. 2019. Т. 29, № 4. С. 44–53. DOI: 10.21778/2413-9599-2019-29-4-44-5
5. Шейнин Ю. Е., Оленев В. Л., Лавровская И. Я., Дымов Д. В., Кочура С. Г. Разработка, анализ и проектирование транспортного протокола СТП-ИСС для бортовых космических сетей SpaceWire. Научный журнал «Исследования наукограда» №1-2. Сиб. гос. аэрокосмич. ун-т. Красноярск, 2016. C. 21-30
6. Чумакова, Н. Ю. Применение сетей Петри для анализа механизмов планирования коммуникационных протоколов / Н. Ю. Чумакова, А. А. Мамонтова // Научно-технический вестник Поволжья. – 2024. – № 3. – С. 195-200. – EDN WFEZVL.
7. Питерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 1984. 264 с.
8. Оленев, В. Л. Методология формализованного проектирования коммуникационных протоколов на основе сетей Петри / В. Л. Оленев // Информация и космос. – 2022. – № 4. – С. 37-45. – EDN VEHYLM.
9. Теория автоматов / Э. А. Якубайтис, В. О. Васюкевич, А. Ю. Гобземис, Н. Е. Зазнова, А. А. Курмит, А. А. Лоренц, А. Ф. Петренко, В. П. Чапенко // Теория вероятностей. Математическая статистика. Теоретическая кибернетика: ВИНИТИ, 1976. 109—188 c
10. Lin, Chuang and Marinescu, Dana Cristian, "Application of Modified Predicate Transition Nets to Modeling and Simulation of Communication Protocols" (1986). Department of Computer Science Technical Reports. Paper 518.
11. Henri Habrias, Marc Frappier. Chapter 12. SDL // Software Specification Methods. — John Wiley & Sons, 2006. — 418 p.
12. Карпов Ю.Г. Model Checking. Верифицикация параллельных и распределительных программных систем. – СПб.: БХВ-Петербург, 2010. – 560с.
13. R. Alur, D. L. Dill: “A theory of timed automata,’‘ TCS, ı26(2), pp. 183‐235, 1994

# **ПРИЛОЖЕНИЕ А**

Таблица А. 1 - Описание состояний временной модели Крипке

|  |  |  |  |
| --- | --- | --- | --- |
| *Состояние* | *Предикаты* | *Описание* | *Функция времени* |
| 1 | 2 | 3 | 4 |
| S0 | rec\_first\_tc | Прием метки времени | ТМ(S0)=0 |
| S1 | new\_ep | Начало новой эпохи | ТМ(S0)=0 |
| S2 | wait\_trans,  I\_1 | Передача данных или ожидание во время допуска в начале эпохи | ТМ(S10)=1 |
| S3 | wait\_trans,  II\_1 | Передача данных или ожидание в окне ожидания метки времени в начале эпохи | ТМ(S10)=1 |
| S4 | rec\_tc | Прием метки времени во окне ожидания метки времени (начало эпохи) | ТМ(S0)=0 |
| S5 | comp1 | Корректировка значения длительности временного интервала после прихода метки времени окна ожидания метки времени в начале эпохи | ТМ(S0)=0 |
| S6 | wait\_trans,  III | Передача данных или ожидание вне окна ожидания метки времени | ТМ(S10)=1 |
| S7 | rec\_tc | Прием метки времени вне окна ожидания метки времени (прием неактуальной метки времени) | ТМ(S0)=0 |
| S8 | 1st\_or\_2nd | Прием первой или второй неактуальной метки времени (вне окна ожидания метки времени) | ТМ(S0)=0 |
| S9 | 3rd, TTS\_end | Прием третьей неактуальной метки времени. Остановка таймера временного интервала | ТМ(S0)=0 |
| S10 | stop | Прекращение узлом работы в соответствии с расписанием после прихода третьей неактуальной метки времени | ТМ(S10)=1 |
| 1 | 2 | 3 | 4 |
| S11 | rec\_tc | Прием очередной метки времени после остановки таймера временного интервала | ТМ(S0)=0 |
| S12 | comp\_3 | Корректировка значения длительности временного интервала после приема неактуальных меток времени | ТМ(S0)=0 |
| S13 | wait\_trans,  II\_2 | Передача данных или ожидание в окне ожидания метки времени в конце эпохи | ТМ(S10)=1 |
| S14 | rec\_tc | Прием метки времени в окне ожидания метки времени (конец эпохи) | ТМ(S0)=0 |
| S15 | comp2 | Корректировка значения длительности временного интервала после приема метки времени во время окна ожидания метки времени в конце эпохи | ТМ(S0)=0 |
| S16 | wait\_trans, I\_2 | Передача или ожидание во время допуска в конце эпохи | ТМ(S10)=1 |
| S17 | TTS\_end\_last | Истечение таймера временного интервала для последнего временного интервала эпохи | ТМ(S0)=0 |
| S18 | rec­\_tc | Получение метки времени во время допуска в начале или в конце эпохи | ТМ(S0)=0 |

Таблица А. 2 - Описание атомарных предикатов временной модели Крипке

|  |  |
| --- | --- |
| *Атомарный предикат* | *Описание* |
| 1 | 2 |
| rec\_first\_tc | Получение первой метки времени |
| new\_ep | Начало новой эпохи |
| wait\_trans | Передача данных или ожидание |
| I\_1 | Допуск в начале эпохи |
| rec\_tc | Получение метки времени |
| 1 | 2 |
| II\_1 | Окно ожидание метки времени в начале эпохи (первые K временных интервалов) |
| comp1 | Корректировка значения длительности временного интервала после приема метки времени в первые K временных интервалов эпохи |
| III | Интервалы вне окна ожидания метки времени |
| 1st\_or\_2rd | Получение первой или второй неактуальных меток времени |
| 3rd | Получение третьей неактуальной метки времени |
| TTS\_end | Остановка таймера временного интервала |
| stop | Прекращение узлом передачи данных в соответствие с расписанием |
| comp\_3 | Корректировка значения длительности временного интервала после приема метки после остановки таймера временного интервала |
| II\_2 | Окно ожидания метки времени в конце эпохи (последние K временных интервалов) |
| comp2 | Корректировка значения длительности временного интервала после приема метки времени в первые K временных интервалов эпохи |
| I\_2 | Допуск в конце эпохи |
| TTS\_end\_last | Истечение таймера временного интервала для последнего временного интервала |

# **ПРИЛОЖЕНИЕ Б**