Параллельная верификация модели протокола обмена маршрутной информацией

И.А. Коротков, В.А. Крищенко
МГТУ им. Н. Э. Баумана
Россия, 105005, Москва, 2-я Бауманская ул., 5
E-mail: twee@tweedledee.org, mstu@sevik.ru

В протоколе обмена маршрутной информацией RIP существует проблема образование ложных маршрутов и маршрутных петель. Ставится задача нахождения интервалов значений таймеров протокола, позволяющих предотвратить образование маршрутных петель для заданной топологии сети. Предлагается способ решения поставленной задачи, включающий формальное описания стандарта протокола RIP, построение по данному описанию и заданной топологии сети конечной модели и её последующую формальную верификацию. Для верификации модели вычислительной сети, использующей протокол RIP, применяется разработанный параллельный верификатор моделей, описанных на языке Promela.

PARALLEL VERIFICATION OF ROUTING INFORMATION PROTOCOL MODEL / I.A.Korotkov, V.A. Krishchenko (Bauman Moscow State Technical University, 2-nd Baumanskaya, 5, Moscow, 105005, Russia).

Count-to-infinity is a well-known problem of Routing Information Protocol (RIP). RIP uses different timers to control its routing table so it could be possible to prevent routing loops by choosing appropriate timer values. RIP formal specification is given to create protocol model. Model generator is used to create Promela models from network topology description. Parallel model checker is used to find the conditions when count-to-infinity problem could be avoided.

,

Введение

Группа протоколов обмена маршрутной информацией под общим названием RIP [1, 2, 3] достаточно широко используется в IP-сетях. Все версии протокола RIP используют для расчета метрики маршрутов распределенный вариант алгоритма Беллмана-Форда [4], который может приводить к возникновению ложных маршрутов и циклов маршрутизации.

В ряде широко применяемых реализаций протокола RIP включен так же дополнительный таймер удержания [5], не входящий в текущий стандарт протокола [2]. Существуют рекомендуемые системным администраторам значения таймеров протокола, призванные уменьшить вероятность возникновения ложных циклов маршрутизации в сети. Представляет интерес вопрос, как для заданной топологии сети найти автоматически такие значения таймеров, которые исключат возможность образования ложных циклом маршрутизации.

В работах [6, 7] было дано формальное доказательство корректности протокола RIP, в [6] была описана его модель на языке Promela [8]. Однако эти работы рассматривали случай отсутствующих или разовых ошибок в сети, а предлагаемая модель не включала в себя таймеры протокола. В силу этого результаты указанных работ невозможно напрямую использовать для поставленной задачи определения диапазонов значений таймеров, позволяющих избежать возникновения циклов маршрутизации.

Статья организована следующим образом.

1) Первая часть посвящена формализации протокола RIP, которая необходима для создания его модели

2) Во второй части на примере рассматривается процесс образования ложных маршрутных петель.

3) Третья часть содержит описание метода поиска значений таймеров протокола RIP, которые позволили бы избежать образования циклов маршрутизации.

4) Четвертая часть содержит пример результатов работы метода.

1 Формальное описание протокола RIP

На основании стандартов [2, 3] и описания работы таймера удержания [5] можно дать более формальное описание подмножества протокола RIP с таймером удержания. Следует отметить, что протоколы RIP2 [2] и RIPng [3] различаются главным образом в форматах передаваемых сообщений и используемом сетевом протоколе, поэтому дальнейшее описание применимо к ним обоим.

В протоколе RIP маршрутизатор не хранит никакой информации о сетевой топологии, кроме таблицы маршрутизации и информации и непосредствнно подключеннйх к ему сетях. Пусть $L = \{0, 1, \ldots, \mu\}$ — множество всех возможных значений метрик маршрутов. Протокол RIP использует метрику, равную числу маршрутизаторов до сети назначения. Значение метрики $\mu = 16$ выбрано в качестве «бесконечности» и означает недоступность сети.

Обозначим множество всех маршрутизаторов использующей RIP системы как R, а множество всех сетей системы — как N. Обозначим множество сетей, подключённых к маршрутизатору r, как $E_r \subset N$, а множество сетей, известных маршрутизатору r в момент времени t_k, как $A_r^k \subseteq N$

Таблица маршрутизации маршрутизатора $r \in R$ в момент времени t_k является следующим отображением:

$$\alpha_r^k : A_r^k \to R \times L \times N.$$

Запись в таблице маршрутизации о маршруте является упорядоченной тройкой $\alpha_r^k(n) = \langle h, l, s \rangle$.

1) Сеть назначения данного маршрута: $n \in A_r^k$.

2) Следующий маршрутизатор в маршруте: $h \in R$.

3) Рассчитанная метрика маршрута: $l \in L$.

4) Если маршрутизатор не подключён к сети n непосредственно ($n \notin E_r$), то $s \in E_r$ — сеть, откуда была получена информация о данном маршруте. Если маршрутизатор подключён к сети непосредственно ($n \in E_r$), то метрика маршрута до этой сети не меняется: $\forall k$: $\alpha_r^k(n) = \langle r, 1, n \rangle$.

В дальнейшем условимся для краткости опускать индекс k, когда он очевиден или неважен. В начальный момент времени маршрутизатор располагает информацией только о подключённых к нему сетях ($A_r^0 = E_r$), и начальная таблица маршрутизации содержит информацию только о них:

$$\forall n \in E_r : \alpha_r^0(n) = \langle r, 1, n \rangle.$$

В стандарте [2] заданы три операции с таблицами маршрутизации:

1) добавление в таблицу маршрута до сети n: $A_r^k = A_r^{k-1} \cup \{n\}$, $\alpha_r^k(n) = \langle h, l, s \rangle$, $l < \mu$;

2) пометка сети n как временно недостижимой: $\alpha_r^k(n) = \langle h, \mu, s \rangle$;

3) удаление информации о маршруте до сети n из таблицы: $A_r^k = A_r^{k-1} \setminus \{n\}$.

Пусть $C_{r,d} = E_r \cap E_d$ — множество таким сетей, к которым непосредственно подключены оба маршрутизатора r и d. Пусть $H_r^v = \langle d \in R | v \in C_{r,d} \rangle$ — множество всех маршрутизаторов, доступных от маршутизатора r через сеть v при помощи только непосредственной маршрутизации. Тогда маршрутизатор $r \in R$ регулярно рассылает сообщение с маршрутной информацией

$$M_r^v \subseteq A_r \times \{r\} \times L$$

через сеть $v \in C_{r,d}$ каждому своему соседу $d \in H_r^v$. Для рассылки сообщений используется широковещательная рассылка.

При использовании правила «расщеплённый горизонт с недопустимым обратным маршрутом» маршрутизатор проверяет, не совпадает ли сеть v, куда посылается сообщение M_r^v, с сетью, откуда получена информация о рассылаемом маршруте. Если они совпадает, то в качестве метрики маршрута в сообщении указывается «бесконечность» в качестве борьбы с маршрутными циклами. Поэтому сообщение с маршрутной информацией M_r^v формируется по следующим правилам:

$$\forall n \in A_r : (\alpha_r(n) = \langle h, l, s \rangle \wedge s \neq v) \Rightarrow \langle n, r, l \rangle \in M_r^v,$$
$$\forall n \in A_r : (\alpha_r(n) = \langle h, l, s \rangle \wedge s = v) \Rightarrow \langle n, r, \mu \rangle \in M_r^v.$$

При получении сообщений с маршрутной информацией маршрутизатор выполняет некоторые действия со своей таблицей маршрутизации. Выделим следующие события, связанных с получением информации о маршруте:

1) событие σ_{new} происходит, когда получена информация о новом маршруте;
2) событие σ_{same} — когда получена информация о том же маршруте от того же маршрутизатора;
3) событие $\sigma_{shorter}$ — когда получена информация о более коротком пути до уже известной сети;
4) событие σ_{inf} — когда получено сообщение, что уже известный путь стал недоступен.

Пусть $M_{h,r}^s$ — сообщение с маршрутами, полученное маршрутизатором r от маршрутизатора h через сеть s. В соответствии со спецификацией протокола

$$(\langle n, h, l \rangle \in M_{h,r}^s \wedge n \notin A_r \wedge l < \mu - 1) \Rightarrow \sigma_{new},$$

$$(\langle n, h, l \rangle \in M_{h,r}^s \wedge \alpha_r(n) = \langle h, l_{old}, s_{old} \rangle \wedge l < \mu - 1) \Rightarrow \sigma_{same},$$
$$(\langle n, h, l \rangle \in M_{h,r}^s \wedge \alpha_r(n) = \langle h_{old}, l_{old}, s_{old} \rangle \wedge l < l_{old}) \Rightarrow \sigma_{shorter},$$
$$(\langle n, h, l \rangle \in M_{h,r}^s \wedge \alpha_r(n) = \langle h, l_{old}, s_{old} \rangle \wedge l \geqslant \mu - 1) \Rightarrow \sigma_{inf}.$$

Помимо таймера $T_{send} = 30$ с., управляющего периодичностью рассылки сообщений, каждый маршрутизатор должен использовать ряд таймеров, связанных с каждый маршрутом. Каждый элемент маршрутной таблицы с метрикой $1 < l < \mu$ связан с запущенным таймером истечении времени маршрута по-умолчанию имеющим значение $T_{route} = 180$ с.

Маршрутизатор сбрасывает этот таймер, когда одно случается одно из следующих событий: σ_{new}, σ_{same}, или $\sigma_{shorter}$. Каждый маршрут о недостижимой сети с метрикой $l = \mu$ связан с запущенным таймером уборки мусора, имеющим по-молчанию значение $T_{flush} = 120$ с. При использовании таймера удержания каждая запись о недостижимой сети связана также с таймером удержания, равным по-умолчанию $T_{hold} = 180$ с. При использовании таймера удержания должно выполняться условие $T_{hold} \leqslant T_{flush}$, поэтому при использовании таймера удержания рекомендуется использовать значение $T_{flush} = 240$ с.

Пусть β_r^k — отображение между маршрутами, известными маршрутизатору r и текущими значениями их таймеров в момент времени t_k :

$$\beta_r^k : A_r^k \setminus E_r \to [0, T_{route}] \times [0, T_{flush}] \times [0, T_{hold}].$$

Свяжем следующие события с моментами срабатывания таймеров маршрута:

1) событие $\sigma_{expired}$ происходит в момент срабатывания таймера истечения времени машрута;

2) событие σ_{flush} — в момент срабатывания таймера уборки мусора;

3) событие σ_{hold} — в момент срабатывания таймера удержания.

Рассмотрим возможные состояния маршрута до сети n от маршрутизатора r. Жизненный цикл маршрута можно описать в качестве конечного автомата с выходом (рис. 1), который описывается кортежем $F_n = \langle \Sigma, \Gamma, S, s_0, \delta, \omega \rangle$, где:

1) $\Sigma = \{\sigma_{new}, \sigma_{same}, \sigma_{shorter}, \sigma_{inf}, \sigma_{expired}, \sigma_{hold}, \sigma_{flush}\}$ — входной алфавит автомата;

2) $\Gamma = \{\gamma_{valid}, \gamma_{hold}, \gamma_{remove}, \gamma_{none}\}$ — выходной алфавит автомата;

3) $S = \{s_{valid}, s_{hold}, s_{invalid}, s_{unknown}\}$ — множество состояний маршрута;

4) начальное состояние $s_0 = s_{valid}$, если $n \in E_r$ (маршрутизатор подключён к сети непосредственно);

5) начальное состояние $s_0 = s_{unknown}$, если $n \notin E_r$.

6) $\delta : S \times \Sigma \to S$ — функция переходов, как показано на рис. 1.

7) $\omega : S \times \Sigma \to \Gamma$ — функция выхода, как показано на рис. 1.

Символы выходного алфавита автомата имеют следующий смысл:

1) γ_{valid} — в таблицу маршрутизации добавлена информация о достижимой сети;

2) γ_{hold} — сеть отмечена в маршрутной таблице как недоступная;

3) γ_{remove} — маршрут удалён из таблицы в ходе уборки мусора;

4) γ_{none} — маршрутная таблица не меняется.

Тогда выходной алфавит можно рассматривать как множество предикатов, причём верны следующие утверждения:

$$\gamma_{valid} \Rightarrow (\alpha_r^k(n) = \langle h, l+1, s \rangle \wedge \beta_r^k(n) = \langle T_{expire}, 0, 0 \rangle),$$

$$\gamma_{hold} \Rightarrow (\alpha_r^k(n) = \langle h, \mu, s \rangle \wedge \beta_r^k(n) = \langle 0, T_{flush}, T_{hold} \rangle),$$

$$\gamma_{remove} \Rightarrow (A_r^k = A_r^{k-1} \setminus \{n\}).$$

Кроме того, при событии γ_{valid} сбрасывается таймер устаревания маршрута, а при событии γ_{hold} запускаются таймеры удержания и таймер сборки мусора.

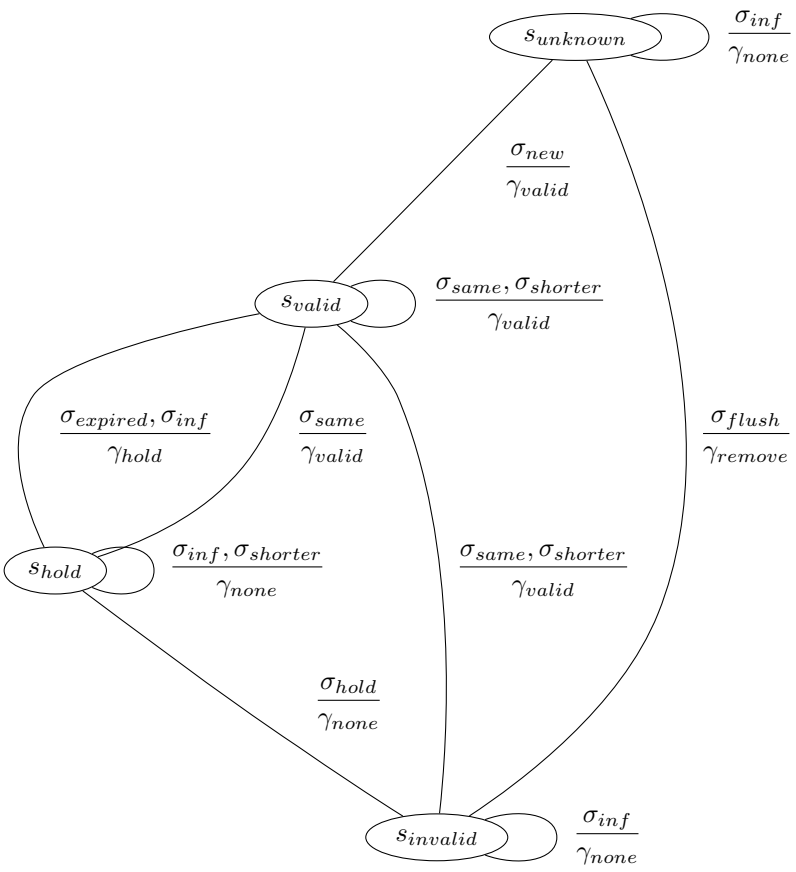

Рис. 1 — Конечный автомат состояний маршрута в протоколе RIP

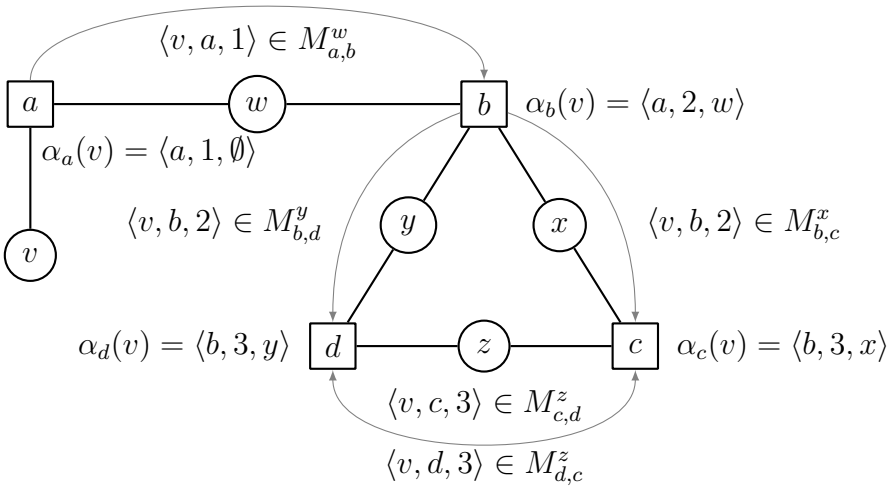

Рис. 2 — Система, подверженная возникновению маршрутных петель

2 Проблема образования маршрутных петель

На рис. 2 показана простейшая вычислительная сеть, которая подвержена образованию циклов при использовании стандартного протокола RIP2 даже при использовании правила расщеплённого горизонта.

В работах [4, 7] было доказано, что при отсутствии ошибок в сети каждый маршрутизатор RIP будет создаст полную верную таблицу маршрутизации в течение конечного отрезка времени. Когда это случится, в системе будут постоянно рассылаться сообщения с маршрутами до сети v, показанные на рис. 2 (маршруты до прочих сетей не показаны для упрощения примера).

Допустим, что затем происходят следующие события.

1) Сеть w перестает нормально функционировать, после чего сработает таймер истечения маршрута (событие $\sigma_{expired}$) до сети v у маршутизатора b , тогда $\alpha_b(v) = \langle a, \mu, w\rangle$.

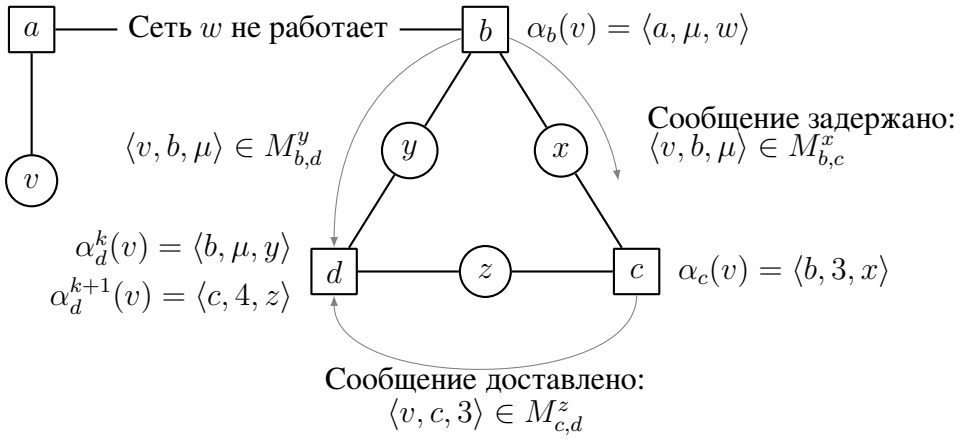

Рис. 3 — Первая стадия образования маршрутной петли

2) Маршрутизатор b отправляет сообщение $\langle v, b, \mu\rangle \in M^x_{b,c}, \langle v, b, \mu\rangle \in M^y_{b,d}$.

3) Маршрутизатор c отправляет сообщение $\langle v, c, 3\rangle \in M^z_{c,d}$.

4) Маршрутизатор d принимает сообщение $\langle v, b, \mu\rangle \in M^y_{b,d}$, в результате $\alpha_d(v) = \langle b, \mu, y\rangle$ (рис. 3).

5) Маршрутизатор d принимает сообщение $\langle v, c, 3\rangle \in M^z_{c,d}$, в результате $\alpha_d(v) = \langle c, 4, z\rangle$. Маршрутизатор c принимает сообщение $\langle v, b, \mu\rangle \in M^x_{b,c}$.

6) Маршрутизатор d отправляет сообщение $\langle v, d, 4\rangle \in M^y_{d,b}$.

7) Маршрутизатор b принимает сообщение $\langle v, d, 4\rangle \in M^y_{d,b}$, в результате $\alpha_b(v) = \langle d, 5, y\rangle$ (рис. 4). Маршрутизатор b начинает сообщать об этом маршруте маршрутизатору c. Таким образом, сформирована ложная маршрутная петля (рис. 4).

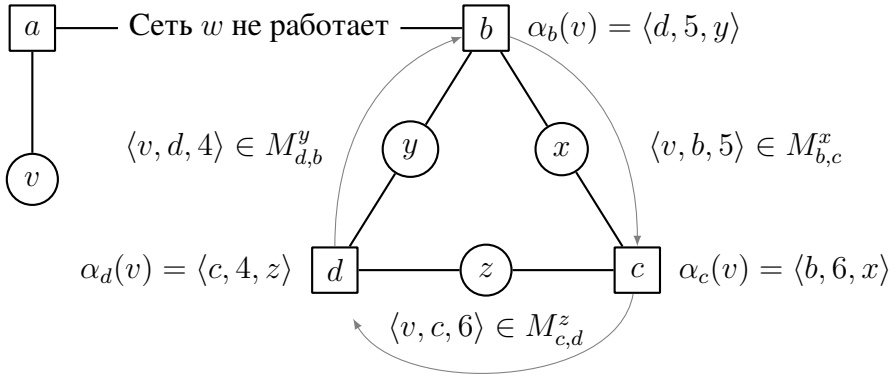

Рис. 4 — Вторая стадия образования маршрутной петли

Кроме правила расщепленного горизонта, в протоколе RIP существует два способа борьбы с маршрутными петлями — инициированные обновления [2], когда маршрутизатор рассылает сообщения об изменённом маршруте в течение нескольких секунд, и таймер удержания [5],

который задаёт момент времени, в течение которого маршрутизатор игнорирует сообщения о доступных маршрутах сети, недавно ставшей недоступной.

Предположим теперь, что в рассмотренной системе используется протокол RIP с включенными инициированными обновлениями. Если инициированное сообщение от маршрутизатора b к маршрутизатору c потеряно, то ситуация будет развиваться аналогичным образом и возникнет такая же маршрутная петля. Кроме того, даже без потери сообщения может иметь место случай, когда волна инициированных обновлений пересечется с регулярными сообщениям: например, маршрутизатор c успеет послать регулярное сообщение маршрутизатору d в момент между получением маршрутизатором d инициированного сообщения от b и до получения c такого же сообщения.

Как видно из изложенного выше, использование инициированных сообщений не приведет к исчезновению возможностей для возникновения циклов маршрутизации, а только к уменьшению вероятности их образования. Использование же таймеров удержания, возможно, могло бы привести к предотвращению циклов при некоторых значениях таймеров.

3 Поиск значений таймеров, устраняющих циклы

Сеть из маршрутизаторов, использующих протокол RIP, можно рассматривать как систему с конечным числом состояний, состоящую из независимых процессов. Поскольку в протоколе RIP нет какой-либо синхронизации между маршрутизаторами, то передачу сообщений между ними можно моделировать при помощи асинхронных каналов. Такая система является конечной, поскольку любая использующая протокол RIP сеть не может иметь «диаметр», превышающий 15 маршрутизаторов. Таким образом, верифицируемая система является распределенной системой с ограниченным числом процессов.

Метод верификации конечных моделей путём полного перебора состояний [9] может быть применен для решения поставленной задачи при зафиксированных значения таймеров T_{route}, T_{hold} и T_{flush}. Для рассмотренного примера очевидным нарушения корректности системы будет $\alpha_b(v) = \langle c, l, x\rangle \vee \alpha_b(v) = \langle d, l, y\rangle$

Интервалы всех таймеров протокола RIP выбираются с периодом, равном значению таймера рассылки сообщение $T_{send} = 30$ с, причём значения интервалов таймеров не превышают нескольких минут. Таким образом, множество возможных значений любого таймера протокола содержит лишь несколько элементов, и при верификации можно воспользоваться перебором всех возможных значений таймеров T_{route}, T_{hold} и T_{flush}.

Поскольку в протоколе RIP используется групповая рассылка и простой транспортный протокол UDP, то в случае проблем в сети рассылающая сторона не предпринимает никаких специальных действий. При поиске условий для полного устранения возникновения маршрутных петель в качестве модели ошибок можно рассмотреть следующую:

– сеть может быть исправна (сообщение корректно передаётся) или неисправна (сообщение в этом случае теряется);

– сеть меняет своё состояние случайным образом перед передачей по ней сообщения протокола RIP.

При необходимости можно ограничить максимальное число смены состояний для каждой сети.

На рис. 5 представлена функциональная схема разработанного и реализованного метода. В качестве языка описания верифицируемой модели выбран язык Promela [8] как достаточно распространённый язык описания конечных моделей, состоящих из конкурирующих процессов.

Язык Promela далек от нотации приведенного в первой части формального описания протокола, а ручной переход от подобного описания к коду на языке Promela может привезти к ошибкам. Поэтому было принято решение использовать разработанный анализатор формул в

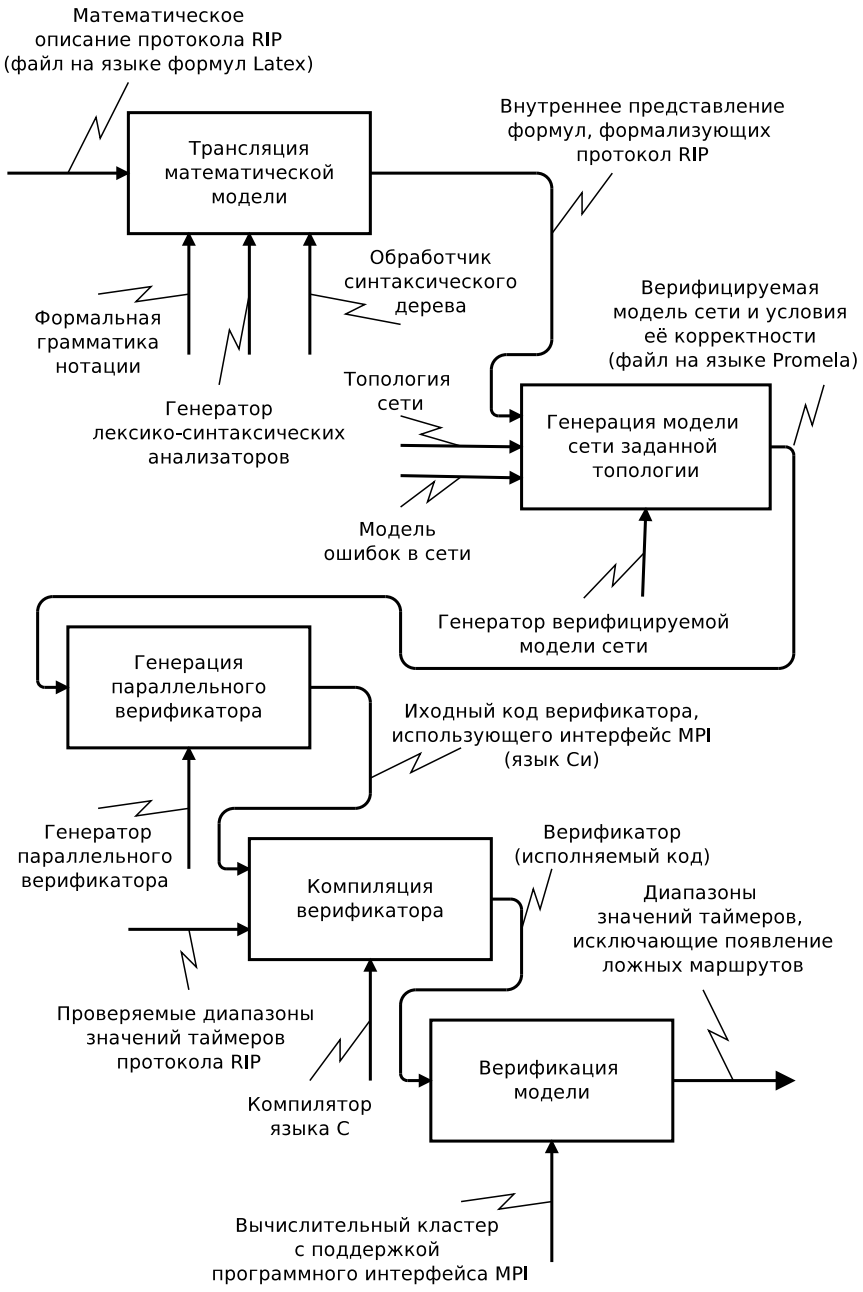

Рис. 5 — Функциональная схема метода поиска множества значений таймеров

нотации Latex [10] для автоматизированной генерации части кода на языке Promela. Таким образом, первым шагом метода является разбор формального описания протокола RIP в нотации формул Latex, а вторым — генерация кода на языке Promela.

В качестве исходного формального описания вступает упрощенный вариант формализации протокола из первой части статьи. В частности, в нём хранятся и передаются маршруты только до единственной сети, поскольку при полном переборе состояний системы гораздо эффективнее рассматривать маршруты до каждой сети в отдельных экспериментах в силу их независимости.

Для описания топологии сети в разработанной системе используется язык, основанный на языке разметки YAML. На рис. 6 показано описание сети, соответствующей показанной на рис. 2. В примере используется модель ошибок, использующая неограниченных изменения состояния всех сетей, кроме сети v (она не меняет своего состояния) и сети w (она может поменять его единожды, то есть в какой-то момент стать неисправной навсегда).

Предварительные вычислительные эксперименты показали, что распространённое про-

```
defaults:
    errors: -1
networks:
    v:
        routers: [a]
        errors: 0
    w:
        routers: [a, b]
        errors: 1
    x:
        routers: [b, c]
    y:
        routers: [b, d]
    z:
        routers: [c, d]
condition:
    network: v
    router: b
    via: a
```

Рис. 6 — Пример описания исследуемой сети

граммное обеспечение для верификации конечных моделей Spin [8] трудно использовать для проверки положительных случаев, требующих полного перебора состояний, из-за высоких требований к памяти. В силу этого для окончательных экспериментов использовалась собственная система параллельной верификации, которая позволяет использовать мощности MPI-кластера [11]. Данный верификатор так же использует в качестве входных данных модель на языке Promela.

При верификации с целью получения наиболее высокой производительности используется генерация исходного кода верификатора на языке Си. Данный код затем компилируется и выполняется на кластере, поддерживающим программный интерфейс MPI для обмена сообщениями между узлами кластера [11]. Последние два этапа выполняются для всех элементов множества проверяемых значений таймеров. В случае успешного прогона верификатора кортеж проверяемых значений таймеров добавляется к множеству найденных.

Параллельная верификация конечных моделей позволяет существенно расширить область применения метода, а использование достаточно распространенного языка описания конечных моделей Promela позволяет при необходимости провести эксперименты с различными верификаторами.

В настоящий момент разработанная система проверяет только сети, в которых значения T_{route}, T_{hold} и T_{flush} одинаковы для всех маршрутизаторов в системе.

4 Результаты экспериментов

Реализованный программный комплекс были использован при проведении экспериментов для рассмотренной сети из четырёх маршрутизаторов (рис. 2), которая известная тем, что допускает возникновение циклов маршрутизации в случае использования протокола RIP без таймера удержания.

В ходе вычислительных экспериментов проверялись все комбинации из следующих возможные значения таймеров с учетом ограничения $T_{hold} \leqslant T_{flush}$:

1) T_{route}: 60 с., 90 с., 120 с., 150 с.;

2) T_{flush}: 120 с., 150 с., 180 с., 210 с., 240 с.;

3) T_{hold}: 120 с., 150 с., 180 с., 210 с., 240 с..

Для исследуемой сети были рассмотрены две модели ошибок: в первом случае сети x, y, z меняли своё состояние сколь угодно много раз (рис. 6), во втором — не более двух раз. В таблице 1 приведены только те комбинации значений таймеров, которые успешно прошли проверку и которые имеют минимальное значение T_{hold} для данных T_{flush} и T_{route}. Показано также число найденных состояний в системах для обоих вариантов модели ошибок («неограниченные ошибки» и «ограниченные ошибки», соответственно).

Из результатов видно, что использование модели неограниченных ошибок приводит к меньшему числу состояний. С точки зрения ограничений на значения таймеров результаты эксперимента совпали для обоих случаев.

Таблица 1 — Результаты проверки значений таймеров

T_{route}, с.	T_{flush}, с.	T_{route}, с.	Количество состояний	
			Ограниченные ошибки	Неограниченные ошибки
60	120	120	0.86^8	0.54^8
60	150	120	1.10^8	0.73^8
60	180	120	1.34^8	0.94^8
60	210	120	1.53^8	1.18^8
60	240	120	1.70^8	1.44^8
90	180	180	1.67^8	1.38^8
90	210	180	1.76^8	1.69^8
90	240	180	1.91^8	1.81^8
120	240	240	2.15^8	1.92^8

Как видно из таблицы, для рассматриваемой сети в случае совпадающих значений таймеров у всех маршрутизаторов для предотвращения образования циклов должно выполняться условие $T_{route} \leqslant \frac{T_{hold}}{2}$. Таким образом, в результате применения созданного метода могут быть найдены условия, приводящие к исключению возможностей для образования ложных циклов маршрутизации в заданной сети.

Заключение

Предложен и реализован метод поиска значений таймеров протокола RIP, предотвращающих образование ложных циклов маршрутизации, для чего была описана формальная модель протокола обмена маршрутной информацией RIP и разработан транслятор формального описания модели протокола в программу на языке описания конечных моделей Promela.

Исследована возможность использования таймера удержания для минимальной топологии, подверженной образованию маршрутных петель при использовании правила расщепленного горизонта. Для проведения экспериментов выбрана собственная система параллельной верификации моделей.

Работа выполнена при частичной поддержке Российского фонда фундаментальных исследований, грант № 09-07-00468.

Список литературы

1. *Hedrick, C.* Routing Information Protocol / C. Hedrick. — http://www.apps.ietf.org/rfc/rfc1058, 1988.

2. *Malkin, G.* RFC 2453: RIP version 2 / G. Malkin. — http://www.apps.ietf.org/rfc/rfc2453, 1998.

3. *Malkin, G.* RFC 2080: RIPng for IPv6 / G. Malkin, R. Minnear. — http://www.apps.ietf.org/rfc/rfc2080, 1997.

4. *Bellman, R.* On a Routing Problem / R. Bellman // *Quarterly of Applied Mathematics*. — 1958. — Vol. 16, no. 1. — Pp. 87–90.

5. *Bruno, A.* CCDA Exam Certification Guide, 2nd Edition / A. Bruno, J. Kim. — Cisco Press, 2003. — P. 696.

6. *Bhargavan, K.* Routing Information Protocol in HOL/SPIN / K. Bhargavan, C. A. Gunter, D. Obradovich // Proceedings of the 13th International Conference on Theorem Proving in Higher Order Logics (TPHOLs '00). — 2000. — Pp. 53–72.

7. *Pei, D.* A Formal Specification for RIP Protocol / D. Pei, D. Massey, L. Zhang; UCLA CSD Technical Report TR-040046. — 2004. — P. 16.

8. *Holzmann, G.* The SPIN Model Checker: Primer and Reference Manual / G. Holzmann. — Addison-Wesley, 2003. — P. 608.

9. *Кларк, Э. М.* Верификация моделей программ. Model Checking / Э. М. Кларк, О. Грамберг, Д. Пелед. — МЦНМО, 2002. — С. 416.

10. *Крищенко, В. А.* Проблема генерации программного кода по математической нотации / В. А. Крищенко // Новые информационные технологии в автоматизированных системах: материалы тринадцатого научно-практического семинара / Моск. гос. ин-т электроники и математики. — М., 2010. — С. 176–183.

11. *Коротков, И. А.* Параллельное построение пространства состояний конечной системы / И. А. Коротков // Новые информационные технологии в автоматизированных системах: материалы тринадцатого научно-практического семинара / Моск. гос. ин-т электроники и математики. — М., 2010. — С. 169–176.