УДК 621.391

ФОРМАЛИЗАЦИЯ ПОВЕДЕНИЯ ПРОТОКОЛОВ ИНФОРМАЦИОННОГО ОБМЕНА, ПРЕДСТАВЛЕННЫХ МОДЕЛЯМИ НА ОСНОВЕ АППАРАТА Е-СЕТЕЙ

Е.В. Дуравкин,
Е.Б. Коровченко
Харьковский национальный университет радиоэлектроники

Запропоновано метод верифікації протоколів інформаційного обміну на основі використання Е-мереж і формальних граматик. Метод дозволяє зменшити розмірність моделі, тим самим зменшуючи ефект «комбінаторного вибуху».

Proposed a method of verification protocols for information exchange on the base of E-nets and formal grammars. The method allows reducing the dimension of the model of system, thereby reducing the effect of "combinatorial explosion".

Предложен метод верификации протоколов информационного обмена на основе использования Е-сетей и формальных грамматик. Метод позволяет уменьшить размерность модели, тем самым уменьшая эффект «комбинаторного взрыва».

Введение

Телекоммуникационные системы представляют собой разновидность параллельных систем, в которых устройства взаимодействуют посредством специальной среды передачи данных. Стремительное возрастание сложности телекоммуникационных систем повышает требования к протоколам, а также устройствам и программному обеспечению, необходимым для их корректного и эффективного функционирования. Проблематика доказательства корректности функционирования сложных распределенных систем, в частности протоколов информационного обмена в телекоммуникационных системах, неоднократно затрагивалась во многих работах [1-3]. Для решения этой задачи используются различные методы верификации [1-4].

При верификации телекоммуникационных систем наибольшее распространение получил метод «проверка на модели» (Model Checking) [3,4]. Однако существенным недостатком метода Model Checking является эффект «комбинаторного взрыва» пространства состояний при верификации сложных систем. Данный недостаток существенно сужает область применения метода для протоколов информационного обмена.

Следовательно, необходимо разработать метод, позволяющий решить задачу верификации протоколов информационного обмена, избегая «комбинаторного взрыва». В качестве аппарата моделирования протоколов информационного обмена для разрабатываемого метода был выбран аппарат Е-сетей. Выбор данного аппарата обусловлен его широким применением в данной области, большими вычислительными возможностями и способностью описывать параллельные взаимодействующие асинхронные процессы, что является характерным признаком протоколов информационного обмена [5].

Для формального описания поведения протоколов информационного обмена предлагается использовать теорию формальных грамматик, что позволяет логически структурировать и определить причинно-следственные связи между событиями модели системы и аналитически описать последовательность смены состояний анализируемой системы [6-8]. Таким образом, цель статьи, посвященная разработке метода построения формальных грамматик для описания поведения протоколов информационного обмена, представленных моделью на базе Е-сети, является актуальной.

I. Метод построения формальных грамматик для моделей протоколов информационного обмена на основе Е-сетей

В общем виде формальная порождающая грамматика G – это совокупность четырех объектов [6,7]:

$$G = (V_t, V_n, Pr, S), \tag{1}$$

где V_t – непустое конечное множество терминальных (обязательных) символов;

V_n – непустое конечное множество нетерминальных (необязательных) символов;

объединение множеств V_n и V_t, $L = V_n \cup V_t$ называется словарем L;

Pr – непустое конечное подмножество правил (продукций) вывода;

S – начальный символ грамматики.

Продукцией называется цепочка правил вида [7, 8]: $\delta(\alpha, \beta)$, где α – начальное значение вывода или входной символ, β – выходной символ, при этом $\alpha \to \beta$.

Основную классификацию вывода грамматик предложил Н. Хомский [6-8]. По иерархии Хомского, грамматики делятся на 4 типа, каждый последующий тип является более ограниченным подмножеством предыдущего, но, благодаря этому, имеет более строгие описательные свойства и легче поддается анализу:

- тип 0: неограниченные грамматики — возможны любые правила.
- тип 1: контекстно-зависимые грамматики — левая часть может содержать один нетерминальный символ, окруженный «контекстом» (последовательности символов, в том же виде присутствующие в правой части); сам нетерминальный символ заменяется непустой последовательностью символов в правой части.
- тип 2: контекстно-свободные грамматики — левая часть состоит из одного нетерминального символа без «контекста».
- тип 3: регулярные грамматики — левая часть может содержать как терминальный, так нетерминальный символ, окруженный «контекстом». Грамматики такого типа соответствуют конечным автоматам.

На практике чаще всего для формализации поведения моделей телекоммуникационных систем используются грамматиками типа 2 или 3 [7, 8].

Языком $L(G)$, порождаемым грамматикой G, называется множество цепочек:

$$L(G) = (\varphi \mid \varphi \in V_t \cup V_n, S \to \varphi). \tag{2}$$

Е-сеть задается как двудольный ориентированный граф, описываемый множеством [8]:

$$E = ((P, S, R), T, I(T), O(T), M_0), \quad (3)$$

где P – конечное непустое множество состояний $\{p_i\}$; S – периферийные состояния, описывающие связи с окружающей средой, $\{S\} \in P$; R – множество решающих состояний, содержит атрибуты срабатывания переходов, $\{R\} \in P$; T – непустое конечное множество переходов; $I(T), O(T)$ – функции связи переходов по входам и по выходам соответственно, аналогично функциям прямой и обратной инцидентности; M_0 – начальная маркировка сети.

Для построения формальной грамматики Е-сети необходимо:

1. Указать алфавит, из которого формируются слова языка. В соответствии с особенностями Е-сетей выделяется множество терминальных символов и множество нетерминальных символов.

2. Задать правила вывода формальной грамматики языка – перечислить правила, по которым из символов алфавита строятся их последовательности. На правила вывода налагается ряд ограничений, согласно которым определяется классификация грамматики.

Формальная грамматика, описывающая модель, построенную на базе аппарата Е-сети, задается следующим образом:

$$G = (V_t, N, Pr, S). \quad (4)$$

Множеством терминальных символов V_t является множество состояний протокола информационного обмена моделируемых вершинами-переходами и вершинами-позициями модели $\{A, B, C...Z; a, b...z\} \in V_t$, где $a, b...z$ - символьное обозначение вершин-переходов, $A, B, C...Z$ – символьное обозначение вершин-позиций ($\{T, J, F, MX, MY\}$ – не являются вершинами-позициями); N – непустое конечное множество нетерминальных символов, $V_t \cap N = 0$.

Множество нетерминальных символов N состоит из следующих элементов: $\{\cup, Tr, r(x), ",", "\} \in V_n$, где символ $\cup$ обозначает объединение нескольких полных цепочек языка, которые имеют общее начальное состояние, $r(x)$ содержит множество значений атрибутов, влияющих на правила срабатывания переходов, Tr – символ, указывающий на тип перехода (T, J, F, MX, MY), $","$ – символ разделения.

Определение ограничений, накладываемых на правила вывода продукций Е–сети является ключевым звеном построения формальной грамматики. Согласно классификации по Хомскому [5-7] грамматики Е-сетей относятся к классу укорачиваемых контекстно-свободных (УКС) либо регулярных грамматик (УКС без учета пустой цепочки). Цепочка, которая не содержит ни одного символа, называется пустой цепочкой (ε) [6,7]. Грамматика G для Е-сети считается УКС, если каждое правило из Pr имеет вид $A \to \beta$, где $A \in V_n, \beta \in (V_t \cup V_n)^*$. Грамматика G регулярна, если каждое правило из Pr имеет вид $A \to \beta$, где $A \in V_n, \beta \in (V_t \cup V_n)^+$. Обозначим через $(V_t \cup V_n)^*$ множество, содержащее все цепочки конечной длины в алфавите, включая ε цепочку, а через $(V_t \cup V_n)^+$ – множество, которое не содержит пустую цепочку [6, 7].

Особенностью аппарата Е-сетей является наличие пяти типов переходов [8]: T, J, F – безусловные (неуправляемые) переходы и MX, MY – управляемые переходы, логика которых зависит от предиката $r(x)$. Правила порождения продукций $\delta(\alpha, \beta) \in P$, для каждого типа переходов Е-сетей можно представить следующим способом.

Т-переход моделирует выполнение события, наступающего при выполнении одного условия. Для срабатывания *T*-перехода необходимо отсутствие метки в выходной позиции $p(B) = 0$ и наличие метки во входной позиции $p(A) = 1$. Пример *T*-перехода приведен на рис. 1.

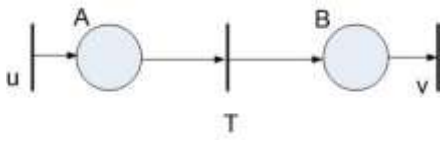

*Рис.*1. Структура *T*-перехода

Правило вывода для *T*-перехода имеет следующий вид:

$$L(T) = \{uT \mid T \to tv\} = \{utv\},$$
$$\{T\} \in N, \{u, v, t\} \in V_t. \quad (5)$$

F-переход используется для разветвления потоков условий или разветвление потока передаваемых данных. Для срабатывания *F*-перехода необходимо отсутствие меток в обоих выходных позициях, $p(B) = 0$ и $p(C) = 0$, и наличие метки во входной позиции $p(A) = 1$. Пример *F*-перехода приведен на рис.2.

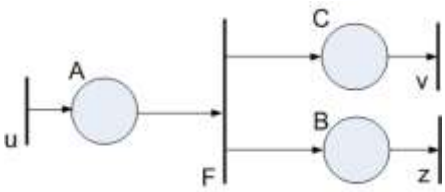

Рис. 2. Структура *F*-перехода

Разветвление потоков можно выразить следующим формализмом:

$$L(F) = \{uF \mid F \to fzv\} = \{ufzv\},$$
$$\{F\} \in N, \{u, z, v, f\} \in V_t. \quad (6)$$

J-переход используется для моделирования событий, требующих выполнения двух условий одновременно. Считается, что *J*-переход активен только в том случае, если позиции *A* и *B* содержат метку, $p(A) = 1$ и $p(B) = 1$, а позиция *C* не содержит, $p(C) = 0$ (рис. 3).

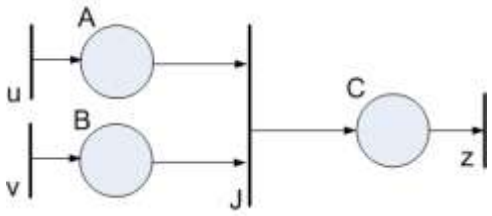

Рис. 3. Структура *J*-перехода

Правила вывода для *J*-перехода имеют следующий вид:

$$L(J)=\{vvJ \mid J \to jz\} \Rightarrow L(J)=\{uvjz\},$$

$$\{J\} \in N, \{u,v,z,j\} \in V_t. \qquad (7)$$

MX-переход задает направление потока меток в зависимости от значения предиката $r(S)$ (рис.4).

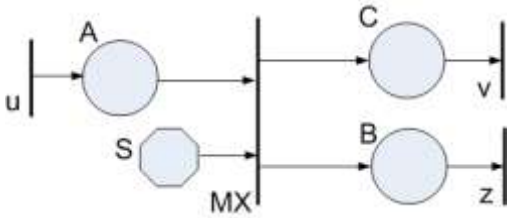

Рис. 4. Структура MX-перехода

Процедуру перехода можно представить следующим образом.

$$L(MX)=\{uSX \mid (u \to X, S \to x \mid X \to z) \Rightarrow r(S)=0 \wedge (p(C)=0) \to uxv\} \qquad (8)$$

или

$$L'(MX)=\{uSX \mid (u \to X, S \to x \mid X \to a) \Rightarrow r(S)=1 \wedge p(B)=0) \to uxz\}, \qquad (9)$$

где $\{S,X\} \in N, \{C,B,u,v,z,x\} \in V_t$.

В том случае, когда значение предиката $r(S)=0$, метка направляется в позицию *C* (8). Если значение предиката $r(S)=1$, метка направляется в позицию *B* (9).

MY-переход используется для моделирования приоритетной обработки различных потоков меток (рис. 5). При этом разрешающая процедура может быть определена различным образом: как операция сравнения фиксированных приоритетов меток или как функция от атрибутов.

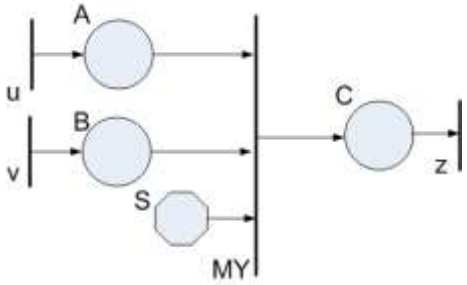

Рис. 5. Структура *MY* перехода

Формирование цепочки можно представить следующим образом:

$$L(MY)=\left\{uvSY\left|\begin{matrix}u\to y, v\to y, S\to y, Y\to z \Leftrightarrow r(S)=0\vee(p(B)=1, p(A)=1, p(C)=0)\to uyz\\ u\to y, v\to y, S\to y, Y\to z \Leftrightarrow r(S)=1\vee(p(B)=1, p(A)=1, p(C)=0)\to vyz\end{matrix}\right.\right\}\Rightarrow$$

$$\Rightarrow\{L(MY)=\{uyz\wedge vyz\}\}, \quad (10)$$

где $\{S,Y\}\in N, \{A,B,C,u,y,v,z\}\in V_t$.

Порядок срабатывания MY-перехода зависит не только от значения предиката $r(S)$, но и от размещения меток. Если и позиция A и позиция B содержат метки ($p(B)=1$ и $p(A)=1$), метка переходит в позицию C из позиции A при условии, что $r(S)=0$, или из позиции B, если $r(S)=1$ (10).

На основе правил вывода продукций для стандартных типов переходов Е-сетей формируется метод разбора всей модели протокола информационного обмена. При моделировании параллельных процессов наибольшее распространение приобрел метод «рекурсивного спуска» [7,8] или разбор методом «сверху вниз».

Данный метод применим, если выполняются следующие условия:

- каждая вершина дерева помечена символом из множества $(V_t\cup N\cup\varepsilon)$, при этом корень дерева помечен символом S (начальный символ);
- начальный символ S может принадлежать как к множеству V_t, так и к множеству N;
- если вершина дерева помечена символом A, где $A\in N$, а ее непосредственные потомки – символами $A_1, A_2 \ldots A_n$, где каждое $A_i\in(V_t\cup N)$, то $A\to A_1, A_2 \ldots A_i$ - правило вывода грамматики;
- символы $A_1, A_2 \ldots A_n$, принадлежащие множеству $r(x)$, не являются ветвями разбора.

Порядок формирования языка определяется набором заключительных состояний модели системы. Каждая цепочка, пройденная до достижения того или иного заключительного состояния, формирует свой язык. Для решения задач исследования телекоммуникационных систем наиболее часто используются L- и P-языки [9].

Преимуществом языка L-типа является построение единственной цепочки, приводящей к заключительной позиции. Таким образом, если на пути достижения заключительного состояния возможны тупики или конфликты, они будут выявлены кратчайшим путем. Для разрешения конфликтов и тупиков, возникающих в системе, используются контрпримеры на формальных грамматиках, которые заключаются в нахождении альтернативных путей поведения системы. Однако построение контрпримера на основе цепочки языка L-типа невозможно. При поиске контрпримера и альтернативных путей желательно использовать язык P-типа. Обобщенный вид формального языка L-типа Е-сетей можно представить следующим образом:

$$L(G)=(\beta\,|\,\beta\in V_t, \delta(S,\beta)\in(N\cup V_t)^+), \quad (11)$$

где β – последовательность запускаемых переходов; $\delta(s,\beta)$ – функция срабатывания переходов без учета пустой цепочки.

Язык P-типа:

$$L(G) = (\alpha \mid \alpha \in V_t, \delta(S, \beta) \in (N \cup V_t)^*), \quad (12)$$

где α – последовательность запускаемых переходов; $\delta(s, \alpha)$ - множество всех строк из алфавита A, включая пустую цепочку.

Можно заметить, что последовательность переходов языка L-типа является включением в язык P-типа. В качестве практической реализации предложенного метода построена формальная грамматика, описывающая процедуру определения ненагруженных каналов связи протокола балансировки сетевой нагрузки в распределенной системе, использующей агентную модель управления.

II. Построение формальной грамматики процедуры определения ненагруженных каналов связи протокола балансировки сетевой нагрузки

Задачей управляющего агента сегмента сети является реализация протокола, отвечающего за балансировку загруженности каналов связи при информационном обмене между компонентами распределенной системы [10].

Модель определения номеров ненагруженных каналов связи с допустимой пропускной способностью представлена на рис. 6.

Переходы модели имеют следующий смысл: T1 – формирование запросов управляющего агента к инцидентным каналам связи; T2 – посылка запроса k-му каналу связи; J1 – ответ k-го канала (ответ возможен только при знании необходимых атрибутов канала); T3, T4 – проверка изменения значения пропускной способности k-го канала связи; MX1, MX2, MX6, MX5, – проверка работоспособности канала связи при уменьшающейся пропускной способности; T7 – посылка на повторную проверку; T7 – посылка сообщения о пригодности канала связи; F1 – проверка соответствия качеству предоставляемых услуг; F2, F3 – занесение значения уменьшения пропускной способности в память; MX4 – проверка соответствия тактовому интервалу; MY1 – проверка метрики системы; T8 – запрос на новую проверку пропускной способности; T9 – выдача номера работающего канала; T – переход к следующей процедуре.

Позиции соответствуют следующим состояниям: s_o - начальное состояние (определение каналов связи, инцидентных управляющему агенту); $s_o req$ - генерирование запроса на обнаружение инцидентных каналов связи; $s_k ans$ - генерирование ответа k-м каналом связи; s_k – предоставление информации о k-м канале связи; $Check\_C^k$ – начало процедуры определения каналов связи с допустимой пропускной способностью; $|C_k^i - C_{k-1}^i| \_ in\_memory$ – фиксирование значения изменения пропускной способности на i-том и $i-1$ интервале времени; $Channel\_enable$ – занесение канала связи в список каналов с допустимой пропускной способностью; $Channel\_drop$ – занесение канала связи в список игнорируемых; *WORK CHANNEL* – определение рабочих каналов связи (вывод их номеров); *Dpor T* – сброс счетчика тактового интервала.

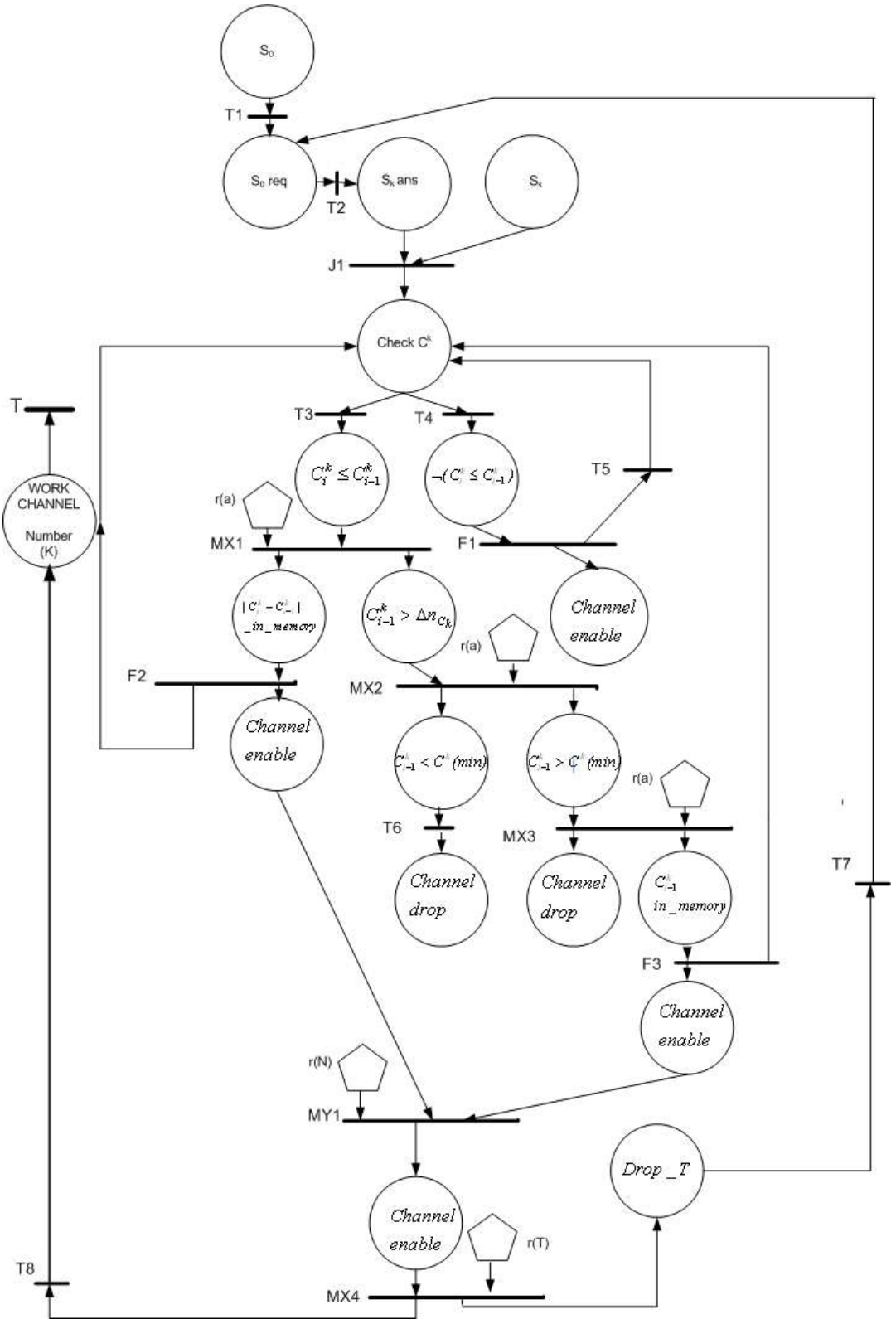

Рис.6. Граф Е-сети процедуры определения каналов связи с допустимой пропускной способностью

Для определения языка модели необходимо четко указать множество заключительных состояний, достижимых в ходе функционирования процедуры определения пропускной способности. Заключительным этапом процедуры является состояние

WORK CHANNEL, которое сохраняет номер канала связи, соответствующего требованиям. В соответствии с разработанным методом и выбранным типом языка процесс опроса смежных каналов связи и выбор наименее загруженного представляется следующим образом:

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Check\_C^k) \vee ((C_k^i - C_{k-1}^i)\_in\_memory)(Channel\_enable \mid r(N \mid N < N\mathbf{(max)}) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Check\_C^k) \vee ((C_k^i - C_{k-1}^i)\_in\_memory)(Channel\_enable \mid r(N \mid N < N\mathbf{(max)}) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_i^k < C_{i-1}^k) \rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k})C_{i-1}^k < C^k\mathbf{(min)}(Channel\_drop)$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(\neg(C_i^k < C_{i-1}^k))Channel\_enable, (Check\_C^k)^n(\neg(C_i^k < C_{i-1}^k)Channel\_enable \Rightarrow$$

$$\Rightarrow S_0S_0reg(S_kansS_k)(Check\_C^k(\neg(C_i^k < C_{i-1}^k))Channel\_enable)^n$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_i^k < C_{i-1}^k)(((C_k^i - C_{k-1}^i)\_in\_memory \mid r(a \mid a=0)))Channel\_enable,$$

$$Check\_C^k(C_i^k < C_{i-1}^k)(((C_k^i - C_{k-1}^i)\_in\_memory \mid r(a \mid a=0)))Channel\_enable... \Rightarrow$$

$$\Rightarrow S_0S_0reg(S_kansS_k)(Check\_C^k(C_i^k < C_{i-1}^k)(((C_k^i - C_{k-1}^i)\_in\_memory \mid r(a \mid a=0)))Channel\_enable)^l$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_i^k < C_{i-1}^k) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k})C_{i-1}^k > C^k\mathbf{(min)} \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N\mathbf{(max)})(Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \vee$$

$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k})C_{i-1}^k > C^k\mathbf{(min)} \vee ((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N\mathbf{(max)})(Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \vee$$

$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k})C_{i-1}^k > C^k\mathbf{(min)} \vee ((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N\mathbf{(max)})(Channel\_enable \mid r(T \mid T > T_z))(Drop\_T)S_0req,$$

где T_z – время жизни метки, C_{i-1}^k, C_i^k – пропускная способность на $i-1$ и i-м интервале времени k-го канала, $C^k\mathbf{(min)}$ – минимальная пропуская способность для k-го канала, Δn_{c^k} – допустимый интервал уменьшения пропускной способности канала связи, $N\mathbf{(max)}$ – метрика сегмента сети.

Язык *P*- типа позволяет определить множество состояний, задействованных для достижения *WORK CHANNEL*, и множество терминальных состояний, которые явля-

ются потенциально достижимыми в ходе функционирования. Как видно из предложенного описания, достижение состояния *WORK CHANNEL* возможно лишь в нескольких ситуациях и представлено цепочками:

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Check\_C^k) \vee ((C_k^i - C_{k-1}^i)\_in\_memory)(Channel\_enable \mid r(N \mid N < N(\mathbf{max})) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \vee$$

$$\vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k})C_{i-1}^k > C^k(\mathbf{min}) \vee ((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N(\mathbf{max}))(Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL$$

$$\cup$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_i^k < C_{i-1}^k) \Leftrightarrow r(a \mid a=0) \rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k})C_{i-1}^k > C^k(\mathbf{min}) \rightarrow$$

$$(Channel\_enable \mid r(N \mid N < N(\mathbf{max}))(Channel\_enable \mid r(T \mid T < T_z))WORK\_CHANNEL,$$

которые соответствуют требованиям, предъявленным к логике работы протокола. Цепочки

$$S_0S_0reg(S_kansS_k)Check\_C^k(\neg(C_i^k < C_{i-1}^k))Channel\_enable,(Check\_C^k)^n(\neg(C_i^k < C_{i-1}^k) \rightarrow$$

$$Channel\_enable... \Rightarrow S_0S_0reg(S_kansS_k)(Check\_C^k(\neg(C_i^k < C_{i-1}^k))Channel\_enable)^n \wedge$$

$$\wedge S_0S_0reg(S_kansS_k)Check\_C^k(C_i^k < C_{i-1}^k)((C_k^i - C_{k-1}^i)\_in\_memory \mid r(a \mid a=0)) \rightarrow$$

$$\rightarrow Channel\_enable, Check\_C^k(C_i^k < C_{i-1}^k)(((C_k^i - C_{k-1}^i)\_in\_memory \mid r(a \mid a=0))) \rightarrow$$

$$\rightarrow Channel\_enable... \Rightarrow S_0S_0reg(S_kansS_k)(Check\_C^k(C_i^k < C_{i-1}^k) \rightarrow$$

$$\rightarrow (((C_k^i - C_{k-1}^i)\_in\_memory \mid r(a \mid a=0))Channel\_enable)^l; \qquad (13)$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_i^k < C_{i-1}^k)r(a \mid a=0) \rightarrow$$

$$\rightarrow ((C_i^k - C_{i-1}^k) > \Delta n_{c^k})C_{i-1}^k < C^k(\mathbf{min})(Channel\_drop); \qquad (14)$$

$$S_0S_0reg(S_kansS_k)Check\_C^k(C_k^i < C_{k-1}^i) \Leftrightarrow r(a \mid a=0) \rightarrow$$

$$\rightarrow ((C_k^i - C_{k-1}^i) > \Delta n_{c^k})((C_k^i - C_{k-1}^i)\_in\_memory) \vee (Check\_C^k)(C_k^i - C_{k-1}^i) < \Delta n_{c^k}) \rightarrow$$

$$\rightarrow C_{i-1}^k > C^k(\mathbf{min}) \vee ((C_k^i - C_{k-1}^i)\_in\_memory) \rightarrow$$

$$\rightarrow (Channel\_enable \mid r(N \mid N < N(\mathbf{max}))(Channel\_enable \mid r(T \mid T > T_z))(Drop\_T)S_0req \qquad (15)$$

определяют потенциально достижимые состояния и возможности зацикливания (степень во главе состояния $(Check\_C^k...)^l$ и $(Check\_C^k...)^n$ указывает на наличие цикла) в модели системы, что может привести к снижению эффективности функционирования протокола и затрате большего времени на выявление рабочих каналов связи.

При использовании языка P-типа возможно определить цепочки, приводящие к нежелательному состоянию ($Channel\_drop$) (14), что, в свою очередь, позволяет устранить причины достижения подобных состояний и определить альтернативные пути.

Выводы

В статье предложен метод формализации поведения протоколов информационного обмена, представленных моделями на базе аппарата Е-сетей. Анализ показал, что для формализации поведения протоколов информационного обмена, представленных моделями на базе Е-сетей, наиболее часто используются грамматиками типа 2 или 3. Выделено два типа языков, которые достаточно полно позволяют описать поведение модели системы и цепочку изменения ее состояний. Выбранные типы языков имеют разную вычислительную мощность, что позволяет в зависимости от задач исследования использовать тот или иной тип языка.

Разработаны процедуры описания поведения переходов Е-сети средствами формальных грамматик (5)-(10). Это в дальнейшем позволит автоматизировать процедуру построения поведенческих цепочек, на базе которых выполняется анализ и верификация протоколов информационного обмена.

В качестве практической реализации предложенного метода построена формальная грамматика для процедуры определения каналов связи с допустимой пропускной способностью протокола балансировки нагрузки в сегменте сети. Определены потенциально достижимые состояния и возможные зацикливания при работе протокола (13)-(15), которые приводят к увеличению временных затрат на выявление рабочих каналов связи и, как следствие, снижению общей эффективности протокола.

Список литературы:

1 *Зайцев Д.А.* Верификация протоколов Ethernet // Научные труды Одесской национальной академии связи им. А.С. Попова. — №1. — 2004. — С. 41–47.

2 *Царьков Д.* Верификация распределённых программ методом проверки на модели: Дис. канд. физ.-мат. наук: 05.13.11 / Московский государственный университет им. М.В. Ломоносова. — 2002. — 185 с.

3 *Hoffman L.* Talking Model-Checking Technology // Communications of the ACM. – 2008. – V. 51. 77. – P. 110–112.

4 *Кларк Э.М., Грамберг О., Пелед Д.* Верификация моделей программ: Model Checking. – М.: МЦНМО, 2002. – 416 с.

5 *Лосев Ю.И. Шматков С.И. Дуравкин Е. В.* Применение методов анализа Е-сетей к моделям СОД // Радиотехника: Всеукр. межвед. науч.-техн. сб. – 2002. – Вып.132. – С. 149–151.

6 *Захаров Н. Г., Рогов В. Н.* Синтез цифровых автоматов: Учебное пособие. – Ульяновск: УлГТУ, 2003. – 136 с.

7 *Волкова И.А., Руденко Т.В.* Формальные грамматики и языки. Элементы теории трансляции. - ВМиК МГУ, 1998. – 62 с.

8 *Пентус А. Е., Пентус М. Р.* Теория формальных языков: Учебное пособие. — М.: Изд-во ЦПИ при механико-математическом ф-те МГУ, 2004. — 80 с.

9 *Питерсон Дж.* Теория сетей Петри и моделирование систем. – М. Мир, 1984. – 264с.

10 *Буханько А.Н., Безрук В.М., Дуравкин Е.В.* Алгоритмы управления каналами связи интеллектуального агента участка сети // Вісник національного університету «Львівська політехніка». – 2009. – № 645. – С. 68–72.