УДК: 004.052, 510.643

Название: Верификация мультиагентной модели протокола скользящего окна

Автор:

Гаранина Н.О. (Институт систем информатики им. А.П. Ершова СО РАН)

Аннотация: Многие варианты коммуникационного протокола скользящего окна, который предназначен обеспечивать надежную передачу данных по ненадежным каналам, были специфицированы и верифицированы с использованием различных техник проверки, таких как доказательство теорем, проверка моделей и их комбинаций. В данной работе предлагается рассмотреть спецификацию протокола скользящего окна как мультиагентной модели. Темпоральные и эпистемические свойства протокола сформулированы с помощью логики знаний и времени.

Ключевые слова: мультиагентные системы, коммуникационные протоколы, логика знаний и времени

1. Введение. Протокол скользящего окна [16] — это коммуникационный протокол, обеспечивающий передачу данных от отправителя к получателю по связывающему их каналу. В силу того, что отправитель, получатель и канал коммуникации работают асинхронно и имеют разные мощности ресурсов, такие как скорость передачи данных, объем используемой памяти, скорость обработки информации и т.п., установить надежность протокола является непростой задачей. Для верификации разнообразных вариантов данного протокола применялись различные формальные и неформальные методы: доказательство вручную [11], различные техники автоматической проверки моделей [18], включая проверку с использованием сетей Петри, использование систем автоматического доказательства теорем [4], комбинированные техники верификации [13].

Отдельный интерес представляет подход к верификации данного протокола, трактующий его как протокол, основанный на знаниях. В работе [8] рассмотрено семейство основанных на знаниях протоколов передачи данных. Вручную доказана корректность этого семейства протоколов, и показано, что стандартные протоколы передачи данных, такие как синхронный AUY-протокол [2], AB-протокол [3], протокол Стеннинга [14], являются реализациями соответствующих протоколов, основанных на знаниях. Протоколы, основанные на знаниях, дают более ясное понимание того, что происходит в протоколах, и позволяют обосновать поведение отправителя и получателя, которые рассматриваются как агенты системы. Используя данный подход, авторы работы [15] вручную провели эпистемический анализ варианта модели TCP-протокола, где показали, что их модель протокола корректна, если канал может переупорядочивать и удалять сообщения. Кроме того, доказано, что для каждого агента глубина его знаний о содержании сообщения может достигать порядкового номера этого сообщения, но при этом содержание не является

общеизвестным.

Однако, насколько известно, исследования по проверке (model checking) мультиагентной модели протокола скользящего окна до сих пор не проводились. В работе [12] рассматривается инструмент символьной проверки мультиагентных моделей MCMAS, и в качестве поясняющего примера взята общая задача передачи данных, специальным случаем которой является протокол скользящего окна. Преимущество метода проверки моделей в данном случае, помимо полной автоматизации, заключается в простоте изменения в модели параметров протокола, таких как свойства надежности канала (дублирование, потеря и переупорядочивание сообщений), информация, наблюдаемая агентами отправителем и получателем и т.п. Проверка моделей также позволяет проверить рациональность действий агентов, например, то, что отправитель высылает сообщение (новую порцию сообщений), только если получатель получил предыдущие, или если какое-либо сообщение было потеряно каналом передачи.

В данной работе предлагается специальная мультиагентная модель для протокола скользящего окна — так называемая *интерпретированная система* [7], которая позволяет строго сформулировать работу протокола и выразить его эпистемические и темпоральные свойства, используя подходящую логику. Интерпретированная система — это, в сущности, помеченная система переходов, снабженная средствами определения *поведения и знаний агентов*. Таким образом, отличие предлагаемой мультиагентной модели от многочисленных моделей протокола скользящего окна, использовавшихся при проверке моделей (model checking), заключается в том, что формально заданы действия агентов, и информация, на основе которой они действуют, что позволяет проверить не только свойства живости и безопасности, но и свойства рациональности агентов. Построенную таким образом модель протокола несложно перевести на язык какого-либо инструмента проверки мультиагентных моделей и проверить интересующие свойства. В силу конечности самой модели возникает необходимость прибегнуть к технике независимости от данных [17] и индукции, пользуясь подходом из [13]. Темпоральные и эпистемические свойства протокола сформулированы с помощью логики знаний и времени. Далее в разд. 2 описаны особенности протокола скользящего окна и связанные с ним задачи. В разд. 3 дается понятие интерпретированной мультиагентной системы и логики спецификации. Разд. 4 содержит интерпретированную мультиагентную систему для протокола скользящего окна и формулировку основных свойств. Разд. 5 — заключение.

Работа выполнена при финансовой поддержке Интеграционного гранта Сибирского Отделения Российской Академии Наук № 15/10 “Математические и методологические аспекты интеллектуальных информационных систем”.

2. Протокол скользящего окна. Протокол скользящего окна [14, 16] — это комму-

никационный протокол, который должен гарантировать надежную передачу данных через ненадежную среду коммуникации. Можно рассматривать вариант протокола, в котором получатель и отправитель являются одним процессом, и среда передает сообщения в двух направлениях, доставляя сообщения от одного отправителя-получателя к другому, как в [16], но здесь удобнее считать отправителя и получателя разными компонентами системы, связанными одним двунаправленным каналом, хотя часто рассматривают два канала: канал для передаваемых данных и канал для передачи подтверждения получения.

Отправитель получает данные из входного потока. Эти данные должны быть переправлены через получателя в выходной поток. Данные отправляются получателю через канал, который является ненадежным, т.е. он может изменять пересылаемые данные. Отправитель помечает каждую порцию данных номером, чтобы получатель мог определить правильность получаемых данных. Если с данными все в порядке, получатель подтверждает получение, высылая номер полученных данных либо ожидаемых данных. Также существуют варианты протокола, в которых отправитель высылает номер испорченных данных [16]. Отправитель может заново отослать неподтвержденные данные, которые сохраняются в так называемом *окне передачи*. Получатель хранит полученные данные в *окне приема*, чтобы иметь возможность передать их в правильном порядке в выходной поток.

Таким образом, протокол имеет следующие особенности (список неполон), изменение которых порождает его различные варианты:

- отправитель, получатель и канал действуют синхронно или асинхронно;
- канал может быть двунаправленным, либо могут использоваться два канала в различных направлениях: от отправителя к получателю и обратно;
- размеры окон передачи и приема, как и максимальный номер, используемый в нумерации данных, могут варьироваться;
- ненадежный канал может изменять данные следующим образом: терять, дублировать, переупорядочивать или портить сообщения, а также изменять их по-разному для отправителя и получателя;
- время жизни отправленного сообщения в канале может быть ограничено или неограничено.

Более общим понятием является стандартная задача передачи данных, сформулированная в работе [8] следующим образом:

Рассматриваются два процесса: *отправитель* и *получатель*. У отправителя S имеется входная лента с бесконечной последовательностью элементов данных $X = \langle x_0, x_1, \ldots \rangle$. S читает эти элементы данных и пытается отправить их получателю R, который должен записать их на выходную ленту. Требуется, чтобы выполнялись следующие условия: (1) свойство безопасности — в любой момент времени последовательность данных, записанная R на выходную ленту, являлась префиксом последовательности X; (2) свойство живости — если среда передачи данных удовлетворяет условиям справедливости, то каждый элемент x_i последовательности X будет записан отправителем R.

Протоколы, решающие задачу передачи данных, имеют высокоуровневое представление в виде *протоколов, основанных на знаниях* [8]. Основной особенностью таких протоколов является то, что действия агентов отправителя и получателя по отправке сообщений регулируются их знаниями. А именно, агент-отправитель посылает сообщение, только если он знает, что предыдущее его сообщение было доставлено, или агент-получатель отправляет сообщение об ошибке, только если он "не знает" полученные данные (получил не те данные, которые ожидал). Согласно определению понятия знания в мультиагентных системах [7] в протоколах, основанных на знаниях, выбор следующего действия агента зависит от самой системы, в отличие от стандартных протоколов, где реализация условий передачи сообщений может быть различной для разных версий. Поэтому проверка знаний агентов в стандартных протоколах полезна, чтобы убедиться, что агенты делают то, что необходимо, тогда и только тогда, когда это необходимо. Также можно исследовать зависимость знаний агентов от информации, которая доступна для их наблюдения в системе.

Оставшаяся часть работы посвящена моделированию протокола скользящего окна как мультиагентной интерпретированной системы и формулированию свойств, подлежащих проверке на получившейся модели.

3. Интерпретированная система и логика спецификации. Дадим краткое определение логики спецификации CTL-K$_n$, которая является комбинацией пропозициональной логики знаний PLK [7] и логики деревьев вычислений CTL [5]. Семантика CTL-K$_n$ определена в терминах отношения выполнимости $\models$ в интерпретированных системах, которые являются специальным видом помеченных систем переходов.

Впервые понятие интерпретированных систем было введено в работе [6], однако мы будем пользоваться модифицированным более детальным определением из [9]. Интерпретированные системы могут быть определены следующим образом. Пусть $\{true, false\}$ — булевские константы, $Prop$ — конечный алфавит пропозициональных переменных. Обо-

значим множество агентов как $A = \{1, ..., n\}$, множество локальных состояний каждого агента $i \in A$ как L_i, множество его возможных действий как Act_i, множество локальных состояний и действий среды как L_e и Act_e соответственно. Пусть множество глобальных состояний системы — это $G = L_1 \times ... \times L_n \times L_e$, где каждый элемент $(l_1, ..., l_n, l_e) \in G$ представляет некоторое состояние всей системы. Считаем, что множество протоколов $P_i : L_i \to 2^{Act_i}$ для $i \in A$ представляет поведение каждого агента, и протокол $P_e : L_e \to 2^{Act_e}$ — поведение среды. Пусть $Act = Act_1 \times ... \times Act_n \times Act_e$ — множество совместных действий. Функция переходов $t : G \times Act \to G$ моделирует вычисления в системе. Интуитивно, имея начальное состояние ι, множество протоколов и функцию переходов, можно построить (возможно, бесконечную) структуру, которая представляет все возможные вычисления системы. Для таких структур существуют различные формализмы, однако, поскольку нас интересуют темпорально-эпистемические свойства, то мы будем пользоваться следующим формальным определением интерпретированной системы.

Определение 1. *(интерпретированной системы)*
Для данного множества агентов $A = \{1, ..., n\}$ и множества пропозициональных переменных $Prop$ *интерпретированная система* — это пара $M = (\mathcal{K}, \mathcal{V})$, причем $\mathcal{K} = (G, T, \sim_1, ..., \sim_n, \iota)$, где

- G — множество глобальных состояний системы;
- $\iota \in G$ — начальное состояние;
- $T \subseteq G \times G$ — всюду определенное бинарное отношение на G такое, что $(w, w') \in T$ тогда и только тогда, когда $t(w, act) = w'$ для некоторого $act \in Act$;
- $\sim_i \subseteq G \times G$ $(i \in A)$ — эпистемическое отношение неразличимости для каждого агента $i \in A$ такого, что $w \sim_i w'$ тогда и только тогда, когда $l_i(w') = l_i(w)$, где функция $l_i : G \to L_i$ возвращает локальное состояние агента i из глобального состояния w;
- $\mathcal{V} : G \longrightarrow 2^{Prop \cup \{true, false\}}$ — функция означивания для пропозициональных переменных $Prop$ такая, что $true \in \mathcal{V}(w)$ для всех $w \in G$. $\mathcal{V}$ сопоставляет каждому состоянию множество пропозициональных переменных, истинных в этом состоянии.

Синтаксис логики спецификации CTL-K$_n$ определяется следующим образом.

Определение 2. *(синтаксиса CTL-K$_n$)*
Синтаксис логики CTL-K$_n$ состоит из формул, построенных из булевских констант, пропозициональных переменных, связок $\neg$, $\wedge$, $\vee$, и следующих модальностей. Пусть $i \in \{1, ..., n\}$, φ и ψ будут формулами. Тогда формулами с модальностями являются

- модальности знаний: $K_i\varphi$ и $S_i\varphi$ (читается как ‘агент i знает’ и ‘агент i предполагает’);
- модальности времени: $\mathbf{AX}\varphi$, $\mathbf{EX}\varphi$, $\mathbf{AG}\varphi$, $\mathbf{EG}\varphi$, $\mathbf{AF}\varphi$, $\mathbf{EF}\varphi$, $\mathbf{A}\varphi\mathbf{U}\psi$, и $\mathbf{E}\varphi\mathbf{U}\psi$ (читается как **A** — ‘для всякого будущего’, **E** — ‘для некоторого будущего’, **X** — ‘на следующем шаге’, **G** — ‘всегда’, **F** — ‘когда-нибудь’, **U** — ‘пока’).

Семантика CTL-K$_n$ следует семантике этих логик.

Определение 3. *(семантики CTL-K_n)*
Отношение выполнимости $\models_M$ между интерпретированными системами, состояниями и формулами задается индуктивно согласно структуре формулы. Для булевских констант, пропозициональных переменных, связок и модальностей времени отношение выполнимости стандартно. Для модальностей знаний выполнимость определяется следующим образом. Пусть $w \in G$, $i \in \{1, ..., n\}$, φ — формула, тогда

- $w \models_M (K_i\varphi)$ если и только если для каждого w': $w \sim_i w'$ влечет $w' \models_M \varphi$;
- $w \models_M (S_i\varphi)$ если и только если для некоторого w': $w \sim_i w'$ и $w' \models_M \varphi$.

Семантика формулы φ в интерпретированной системе M — это множество всех состояний M, в котором выполняется данная формула: $M(\varphi) = \{w \mid w \models_M \varphi\}$.

Семантика конструкций CTL стандартна и интуитивно понятна из пояснений к описанию их синтаксиса. Неформально семантика конструкции $K_i\varphi$ определяется следующим образом: агент знает какой-либо факт φ, если этот факт верен во всех состояниях системы, в которых агент имеет одну и ту же информацию, т.е. его локальные состояния одинаковы.

4. Интерпретированная система для протокола скользящего окна. Вообще говоря, в задаче передачи данных значения транспортируемых элементов данных могут принадлежать бесконечной области определения. Поэтому, как и в работе [13], мы будем использовать то, что протокол скользящего окна обладает свойством независимости от данных [17], и, следовательно, далее будем считать, что данные в нашей модели совпадают с присвоенными им номерами. Нужно заметить, что в таком случае невозможна “мутация” данных в канале передачи, то есть что элемент данных прибывает под чужим номером.

Кроме того, наша модель является параметрической относительно размеров окон приема и передачи, а также времени жизни сообщений в канале. Можно показать с помощью индукции, что проверямые свойства предлагаемой модели сохраняются с соответствующим изменением этих параметров, однако это выходит за рамки данной работы.

Вариант протокола скользящего окна, исследуемый в данной работе, имеет следующие характеристики:

- отправитель, получатель и канал действуют асинхронно;
- канал является двунаправленным;
- передаваемые данные нумеруются числами от 0 до 3;
- размер окон передачи и приема равен 2;
- время жизни отправленного сообщения в канале ограничено;
- канал может изменять данные следующим образом: терять, дублировать, переупорядочивать сообщения отправителя и терять и дублировать сообщения получателя.

Чтобы задать интерпретированную систему переходов, нам нужно определить (1) множество агентов и среду; (2) локальные состояния агентов и среды; (3) действия агентов и среды; (4) протоколы агентов и среды; (5) функцию переходов для агентов и среды; (6) начальные состояния и (7) означивание пропозициональных переменных. Для краткости изложения мы будем пользоваться следующими соглашениями. Пусть i — номер элемента данных из диапазона порядковых номеров данных $i \in [0..3]$, а операции $\oplus, \ominus$ — сложение и вычитание по модулю 4.

(1) Множество агентов и среда

Множество агентов A равно $\{S, R\}$, где S — отправитель и R — получатель. Средой E будет являться канал передачи данных.

(2) Локальные состояния

Определяются значениями соответствующих локальных переменных. Дополнительно выделим *наблюдаемые переменные агента*, которые влияют на его действия, но недоступны для изменения данным агентом.

Получатель R. $L_R = \prod_{i\in[0..3]} r_i \times win_r \times ack$, где

• $r_i \in \{0, 1\}$ — массив записываемых данных ($r_i = 1$ — элемент данных с номером i записан, $r_i = 0$ — элемент данных с номером i не записан), $i \in [0..3]$,

• $win_r \in [0..3]$ — указатель на верхнюю границу окна приема,

• $ack \in [0..3] \cup \{all, err\}$ — подтверждение для элемента данных с номером i или для всех данных из окна приема, либо ошибка.

Отправитель S. $L_S = \prod_{i\in[0..3]}(s_i \times timer_i) \times win_s$, где

• $s_i \in \{-1, 0, 1\}$ — массив передаваемых данных ($s_i = -1$ — элемент данных с номером i не передан, $s_i = 0$ — элемент данных с номером i передан, $s_i = 1$ — получено подтверждение для элемента данных с номером i), $i \in [0..3]$,

• $timer_i \in [0..9]$ — время ожидания подтверждения элемента данных с номером i, $i \in [0..3]$,

• $win_s \in [0..3]$ — указатель на верхнюю границу окна передачи.

Среда (канал передачи) E. $L_E = ack_e \times data_e \times event$, где

• $ack_e \in [0..3] \cup \{all, err\}$ — подтверждение от получателя в канале,

• $data_e \in [0..3] \cup \{err\}$ — номер элемента данных от отправителя в канале,

• $event \in \{arrival, none\}$ — события в системе.

Наблюдаемые переменные. $Obs_R = \{data_e, event\}$, $Obs_S = \{ack_e, event\}$.

Таким образом, множество глобальных состояний определяется как $G = L_R \times L_S \times L_E$ и определяется значениями 18-ти локальных переменных.

(3) Действия

Получатель R. $Act_R = \{write_i, clearR_i, moveWin_r, ackn\}$, где

• $write_i$ — записать элемент данных с номером i, $i \in [0..3]$,

• $clearR_i$ — удалить элемент данных с номером i, $i \in [0..3]$,

• $moveWin_r$ — передвинуть окно,

• $ackn$ — подтвердить получение элемента данных.

Отправитель S. $Act_S = \{send_i, ack_i, clearS_i, moveWin_s, startTime_i, Time_i, stopTime_i\}$, где

• $send_i$ — послать элемент данных с номером i, $i \in [0..3]$,

• ack_i — обработать подтверждение для элемента данных с номером i, $i \in [0..3]$,

• $clearS_i$ — удалить элемент данных с номером i, $i \in [0..3]$,

• $moveWin_s$ — передвинуть окно,

• $startTime_i, Time_i, stopTime_i$ — запустить, обновить и остановить таймер ожидания подтверждения для элемента данных с номером i, $i \in [0..3]$.

Среда (канал передачи) E. $Act_E = \{dlvAck, dlvData, doEvent\}$, где

• $dlvAck$ — доставить подтверждение элемента данных от получателя,

• $dlvData$ — доставить элемент данных от отправителя,

• $doEvent$ — породить событие системы.

(4) Протоколы

В нашей мультиагентной интерпретированной системе протокола скользящего окна протоколы действий агентов и среды определяются тривиально следующим образом: в своем локальном состоянии агент выполняет те действия, которые могут быть выполнены. Действия, которые агент может выполнить, задаются с помощью следующей функции переходов.

Определим функцию переходов локальным образом: для действий каждого агента и среды по отдельности. Определяются пред- и постусловия: $(pre_1, ..., pre_{18}) \times acts \longrightarrow (post_1, ..., post_{18})$, где для каждого $j \in [1..18]$ (1) pre_j — предусловие для соответствующей переменной; если предусловие отсутствует, то это означает, что переменная может принимать любое значение; (2) $acts$ — действия, которые соответствующий агент может выполнить в этом состоянии (3) $post_j$ — постусловие для соответствующей переменной;

если постусловие отсутствует, то это означает, что переменная имеет то же значение, что и в предусловии. Отметим, что агенты и среда в предусловии явно имеют значения только своих локальных и наблюдаемых переменных и свои действия, а в постусловии могут меняться только значения их локальных переменных. Для краткости примем следующие сокращения для обозначения множеств состояний в предусловиях для всех $i \in [0..3]$:

$arriv_i = (event = arrival \wedge r_i = 0 \wedge data_e = i \wedge (win_r = i \vee win_r = i \oplus 1))$ —

прибытие элемента данных с номером i в диапазоне окна;

$start_i = (s_i = -1 \wedge (win_s = i \vee win_s = i \oplus 1))$ —

элемент данных с номером i может быть отправлен;

$arrivAck_i = (event = arrival \wedge s_i = 0 \wedge ack_e = i)$ —

прибыло подтверждение для элемента данных с номером i.

(5) Функции переходов

Получатель R.

• $(arriv_i) \times write_i \times ackn \longrightarrow (r_i = 1, ack = data_e)$ — отметить, что прибыл элемент данных с номером i, и отправить подтверждение;

• $(r_i = 1 \wedge r_{i\oplus 1} = 0 \wedge win_r = i \oplus 1) \times clearR_i \times moveWin_r \longrightarrow (r_i = 0, win_r = win_r \oplus 1)$ — отметить, что элемент данных с номером i отправлен в выходной поток, и сдвинуть окно приема;

• $(r_i = 1 \wedge r_{i\oplus 1} = 1 \wedge win_r = i \oplus 1) \times clearR_i \times moveWin_r \times ackn \longrightarrow (r_i = 0, r_{i\oplus 1} = 0, win_r = win_r \oplus 2, ack = all)$ — отметить, что элементы данных с номерами i и $i \oplus 1$ отправлены в выходной поток, и сдвинуть окно приема.

Отправитель S.

• $(start_i) \times send_i \times startTime_i \longrightarrow (s_i = 0, timer_i = 1)$ — если элемент данных не отправлен из окна передачи, то отметить, что элемент данных с номером i отправлен получателю, и запустить его таймер;

• $(arrivAck_i) \times ack_i \longrightarrow (s_i = 1)$ — отметить, что пришло подтверждение для элемента данных с номером i;

• $(timer_i = 9) \times clearS_i \times stopTime_i \longrightarrow (s_i = -1, timer_i = 0)$ — когда время ожидания подтверждение для элемента данных с номером i вышло, то отметить его как неотправленное, и обнулить его таймер;

• $(s_i = 1 \wedge s_{i\ominus 1} = 0 \wedge win_s = i) \times stopTime_i \longrightarrow (timer_i = 0)$ — если получено подтверждение для элемента данных с номером i из старшего элемента окна передачи, то остановить его таймер;

• $(s_i = 1 \wedge s_{i\oplus 1} = 0 \wedge win_s = i \oplus 1) \times moveWin_s \times stopTime_i \longrightarrow (s_i = -1, win_s = win_s \oplus 1, timer_i = 0)$ — если получено подтверждение для элемента данных с номером i, то сдвинуть окно передачи на 1;

• $((s_i = 1 \wedge s_{i\oplus 1} = 1 \vee ack_e = all) \wedge win_s = i \oplus 1) \times moveWin_s \times stopTime_i \times stopTime_{i\oplus 1} \longrightarrow (s_i = -1, s_{i\oplus 1} = -1, win_s = win_s \oplus 2, timer_i = 0, timer_{i\oplus 1} = 0)$ — если получены подтверждения для элементов данных с номерами i и $i \oplus 1$, то сдвинуть окно передачи на 2;

• $(timer_i > 0 \wedge timer_i < 9) \times Time_i \longrightarrow (timer_i = timer_i + 1)$ — обновить время ожидания подтверждения для элемента данных с номером i, если таймер для него запущен.

Среда (канал передачи) E.

• $(true) \times dlvAck \times dlvData \times doEvent \longrightarrow (ack_e = \{ack_e, ack, err\}$[1]$, data_e = \{0..3, err\}, event = \{arrival, none\})$ — дублировать, доставить и испортить подтверждение; доставить, дублировать, переставить и испортить данные; породить событие доставки либо пустое действие. Константа $true$ в предусловии действий среды означает, что она выполняет эти действия в любом состоянии нашей модели.

Пусть множество локальных функций переходов — это $T_l = \{t_1, ..., t_n\}$, где $t_j = Pre_j \times acts_j \longrightarrow Post_j$. Тогда тотальная функция переходов t определяется множеством глобальных функций $t_{\theta(kn)} = \bigcap_{j\in\theta(kn)} Pre_j \times \prod_{j\in\theta(kn)} acts_j \longrightarrow \bigcap_{j\in\theta(kn)} Post_j$, где $\theta(kn)$ — сочетание из n по k. Заметим, что некоторые глобальные предусловия $\bigcap_{j\in\theta(kn)} Pre_j$ пусты, и в этом случае никаких действий не выполняется. По определению локальных функций t_j, если глобальное действие $\prod_{j\in\theta(kn)} acts_j$ выполняется, то глобальное постусловие $\bigcap_{j\in\theta(kn)} Post_j$ всегда непусто.

(6) Начальные состояния

• $\iota = (r_0 = 0, r_1 = 0, r_2 = 0, r_3 = 0, win_r = 1, ack = err, s_0 = -1, timer_0 = 0, s_1 = -1, timer_1 = 0, s_2 = -1, timer_2 = 0, s_3 = -1, timer_3 = 0, win_s = 1, ack_e = err, data_e = err, event = none)$ — в начальном состоянии нет ни принятых, ни отправленных данных, отправитель готов отправить элементы данных с номерами 0 и 1, получатель готов их принять, данные в канале и подтверждения пока ошибочны, и никаких событий не происходит.

(7) Означивание пропозициональных переменных

Определим означивание следующих пропозициональных переменных следующим образом. Каждой переменной сопоставляется множество состояний, в которых выполняются условия на значения соответствующих локальных переменных, в то время как значения остальных локальных переменных произвольны.

• $DifWindow = \neg(win_r = win_s \wedge win_r = win_s \oplus 1)$ — окна приема и передачи не пересекаются, то есть номера ожидаемых элементов данных не совпадают с номерами посылаемых элементов данных.

• $RecEmpty = (r_{win_r} = 0 \wedge r_{win_r\oplus 1} = 0)$ — окно приема пусто.

[1]Запись $var = \{val_1, ..., val_n\}$ означает недетерминированный выбор значения переменной var из множества $\{val_1, ..., val_n\}$.

• $GoodData_i = (event = arrival \wedge s_i = 0 \wedge data_e = i)$ — доставлен высылавшееся элемент данных с номером i.

• $GoodAck_i = (event = arrival \wedge s_i = 0 \wedge (ack_e = i \vee ack_e = all))$ — получено подтверждение, что доставлено высылавшийся элемент данных с номером i.

Используя вышеизложенное, можно построить интерпретированную систему протокола скользящего окна M_{SW}. Имея данную строгую формализацию протокола можно проверять его свойства с помощью подходящих инструментов проверки мультиагентных моделей. Выразим интересующие нас свойства с помощью логики CTL-K$_n$.

Свойство безопасности заключается в том, чтобы получатель передавал данные в выходной поток в том порядке, в котором он их получил. В случае нашей модели для этого необходимо и достаточно, чтобы в его окно приема не попадали неправильные данные. Поскольку по определению локальных функций перехода получателя ошибочные данные вида err он просто никогда не записывает, то остается лишь проверить, что он не записывает элементы данных с номерами, не попадающими в его окно приема. В таком случае формула безопасности выглядит следующим образом: $Safety = \mathbf{AG}DifWindow \rightarrow \mathbf{AX}RecEmpty$. Заметим, что канал не может переупорядочивать подтверждения, в отличие от данных.

Свойство живости, т.е., что при условии справедливой работы канала получатель рано или поздно получит то, что отправлял отправитель, можно выразить формулой $Live_R = \bigwedge_{i\in[0..3]} \mathbf{AG}(s_i = 0 \rightarrow \mathbf{AXAF}r_i = 1)$. Условие справедливости в данном случае заключается в том, что канал не может портить сообщения отправителя бесконечно долго: $Fair_S = \bigwedge_{i\in[0..3]} \mathbf{AGAF}GoodData_i$. Отметим, что в случае протокола скользящего окна необходимо также условие живости для отправителя, т.е. то, что он рано или поздно получит подтверждение, что отправленные им данные получены, иначе он никогда не сдвинет окно и не начнет отправлять очередную порцию данных. В нашей модели это формула $Live_S = \bigwedge_{i\in[0..3]} \mathbf{AG}(r_i = 1 \rightarrow \mathbf{AXAF}s_i = -1)$, и она должна выполняться при аналогичном условии справедливости для подтверждений получателя $Fair_R = \bigwedge_{i\in[0..3]} \mathbf{AGAF}GoodAck_i$.

Для предложенной модели свойство рациональности действий отправителя можно записать формулой $Rational_S = K_S(K_R(r_i = 1) \vee win_s = win_r \ominus 1) \rightarrow \mathbf{AX}(timer_i = 0)$, которая выражает, что, если отправитель знает, что получатель получил данные либо уже сдвинул окно приема, то он на следующем шаге останавливает таймер этих данных и не пытается послать их снова. Более естественно это свойство могло быть определено формулой логики знаний и времени с модальностями прошлого CTL$_{\mathrm{P}}$K [9]: $Rational_S = K_S\mathbf{AH}K_R(r_i = 1) \rightarrow \mathbf{AX}(timer_i = 0)$, т.е., что, если получатель знает, что отправитель когда-то в прошлом получил данные, то он останавливает таймер, однако эта логика здесь

не рассматривается.

5. Заключение. В данной работе была предложена мультиагентная интерпретированная система протокола скользящего окна, и сформулированы основные свойства этого протокола в логике знаний и времени CTL-K_n, а именно, свойства безопасности, живости и рациональности отправителя. Выполнимость этих свойств в ближайшем будущем проверить с помощью инструментов проверки мультиагентных моделей VerICS[10], MCMAS[12] и Examiner[1]. Также планируется построить мультиагентные модели и проверить эпистемические свойства других коммуникационных протоколов.

Список литературы

1. Гаранина Н.О. Проверка моделей распределенных систем с помощью аффинного представления данных. // Моделирование и анализ информационных систем. 2010. Т. 17, №4. С. 52-59.
2. Aho A. V., Ullman J. D., Yannakakis M. Modeling communication protocols by automata. // 20th Symp. on Foundations of Computer Science: Proc. San Juan, Puerto Rico: IEEE Computer Society, 1979. P. 267–273.
3. Bartlett K. A., Scantlebury R. A., and Wilkinson P. T. A note on reliable full-duplex transmission over half-duplex links. // Communications of the ACM. 1969. V. 12. P. 260–261.
4. Chkliaev D., Hooman J., de Vink E. P. Verification and Improvement of the Sliding Window Protocol. // Lecture Notes in Computer Science. 2003. V. 2619. P. 113-127.
5. Clarke E.M., Grumberg O., Peled D. Model Checking. MIT Press, 1999. 324 p.
6. Fagin R., Halpern J.Y. Modelling knowledge and action in distributed systems // Distributed Computing. 1989. V. 3, I. 4. P. 159-177.
7. Fagin R., Halpern J.Y., Moses Y., Vardi M.Y. Reasoning about Knowledge. MIT Press, 1995. 519 p.
8. Halpern J. Y., Zuck L. D. A little knowledge goes a long way: knowledge-based derivations and correctness proofs for a family of protocols // Journal of the ACM. 1992. V. 39(3). P. 449–478.
9. Kacprzak M., Lomuscio A., Penczek W. Verification of Multiagent Systems via Unbounded Model Checking // The Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS '04): Proc. IEEE Computer Society Washington, DC, USA, 2004. V. 2. P. 638-645.
10. Kacprzak M., Nabialek W., Niewiadomski A., Penczek W., Pólrola A., Szreter M., Wozna

B., Zbrzezny A. VerICS 2007 — a Model Checker for Knowledge and Real-Time // Fundamenta Informaticae. 2008. V. 85, Num. 1-4. P. 313-328.

11. Knuth D.E. Verification of link-level protocols // BIT. 1981. V. 21. P. 31–36.
12. Lomuscio A., Raimondi F. MCMAS: A Model Checker for Multi-agent Systems // Lecture Notes in Computer Science. 2006. V. 3920. P. 450-454.
13. Stahl K., Baukus K., Lakhnech Y., Steffen M. Divide, abstract, and model-check // Lecture Notes in Computer Science. 1999. V. 1680. P. 57–76.
14. Stenning N.V. A data transfer protocol // Computer Networks. 1976. V.1. P. 99–110.
15. Stulp F. and Verbrugge R. A knowledge-based algorithm for the Internet protocol (TCP) // Bulletin of Economic Research. 2002. V. 54(1). P. 69–94.
16. Tanenbaum A.S. Computer Networks (Third Edition). Prentice-Hall International, 1996. 933 p.
17. Wolper P. Expressing interesting properties of programs in propositional temporal logic // The 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages: Proc. ACM New York, NY, USA, 1986. P. 184-193.
18. Zhao Y., Yang Z., Xie J., Liu Q. Formal Model and Analysis of Sliding Window Protocol Based on NuSMV // Journal of Computers. 2009. V. 4, N. 6. P. 519-526.

UDK: 004.052, 510.643

Title: Model Checking Multi-agent Sliding Window Protocol

Author: Garanina N.O. (A.P. Ershov Institute of Informatics Systems SB RAS)

Abstract: Many variants of the communication sliding window protocol were specified and verified using various techniques like theorem proving, model checking and their combinations. In this paper we consider a specification of the Sliding window protocol as a multi-agent affine model. Temporal and epistemic properties of the protocol are expressed in Logic of Knowledge and Time.

Keywords: multiagent systems, communication protocols, logic of knowledge and time