ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

УДК 681.3

В. В. Никифоров, В. И. Шкиртиль

ОЦЕНКА ВРЕМЕНИ ДОСТАВКИ СООБЩЕНИЙ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ С CAN-ИНТЕРФЕЙСОМ

Предложен подход к оценке верхней границы времени доставки сообщений между узлами распределенных систем реального времени, построенных на базе CAN-интерфейса. Показано, что для проверки гарантий своевременности доставки сообщений между узлами таких систем могут использоваться методы, аналогичные методам проверки гарантий своевременности выполнения задач в однопроцессорных системах реального времени.

Ключевые слова: *системы реального времени, распределенные программные комплексы, CAN-интерфейс, выполнимость задач.*

Введение. В ряду ключевых требований к реализации целевых функций систем реального времени (СРВ) выделяются такие, как надежное, предсказуемое поведение, эффективное использование аппаратных средств, обеспечение гарантий своевременного выполнения возлагаемых на СРВ задач. Методы построения и реализации программных приложений для СРВ и методы анализа соответствующих вычислительных моделей, направленные на обеспечение этих требований, развиваются с 1970-х гг. по настоящее время [1—4]. Были разработаны рациональные методы построения программных комплексов с использованием независимых задач реального времени, а также комплексов взаимосвязанных задач, разделяющих общие аппаратные и информационные ресурсы.

В многозадачной системе длительность временного интервала выполнения задачей возлагаемых на нее функций (время отклика) зависит не только от объема исполняемых действий, но и от продолжительности ожидания требуемых ресурсов. Этим обусловлено взаимное влияние (интерференция) задач, приводящее к более или менее значительному увеличению времени их отклика. Для различных архитектур комплексов прикладных задач СРВ найдены методы обеспечения гарантий своевременности выполнения функций системы в условиях такой интерференции.

В частности, для произвольного статического назначения приоритетов невытесняемых задач τ_1, τ_2,…, τ_n, обслуживаемых одним процессором, определен способ вычисления *времени отклика* R_i для каждой из задач τ_i. Удовлетворительный результат сравнения величины R_i с допустимым для нее значением D_i гарантирует своевременность выполнения задачи τ_i.

В распределенных СРВ прикладные программы рассредоточены по узлам, соединяемым коммуникационными каналами. В последнее время для локальных распределенных СРВ широко используется сетевой интерфейс CAN (Controller Area Network) [5—6], обеспечивающий связь узлов сети путем доступа к общей шине. Очередность передачи сообщений, генерируемых в узлах сети, определяется статически присваиваемыми сообщениям значениями

приоритетов. Приоритеты сообщений характеризуют отношение порядка предоставления коммуникационного ресурса. Порожденное в одном из узлов сети с CAN-интерфейсом сообщение с низким приоритетом ожидает момента, когда коммуникационная система окажется свободной от передачи сообщений с более высоким приоритетом. Таким образом, при заданном быстродействии коммуникационной системы время доставки сообщения зависит не только от его длины, но и от загруженности сети более приоритетными сообщениями, т.е. имеет место интерференция передаваемых по сети сообщений, аналогичная интерференции задач многозадачного приложения, исполняемого одним процессором. Для оценки продолжительности выполнения распределенных функций СРВ актуальна разработка методов оценки времени доставки сообщений в условиях такой интерференции.

В настоящей статье показано, как известные методы оценки своевременности выполнения прикладных задач СРВ могут быть адаптированы к оценке своевременности доставки сообщений по сети с CAN-интерфейсом; приводятся метод оценки времени отклика невытесняемых задач с произвольным назначением статических приоритетов и необходимые сведения о CAN-интерфейсе; рассматривается подход к адаптации метода оценки времени отклика невытесняемых задач, который обеспечивает проверку гарантий своевременности доставки сообщений по сети с CAN-интерфейсом.

Оценка выполнимости независимых задач. Согласно принципу структурного соответствия асинхронный характер поступления внешних данных приводит к построению программного приложения СРВ в виде десятков асинхронно исполняемых программных компонентов. Действие каждого из асинхронных компонентов должно завершиться своевременно, в рамках фиксированного временного интервала.

Как правило, число таких компонентов значительно превышает число имеющихся в системе процессоров. Это означает, что асинхронные программные компоненты исполняются в квазипараллельном режиме, предполагающем попеременное переключение процессора между обслуживаемыми им программными компонентами. Возникает интерференция программных компонентов: продолжительность исполнения отдельного компонента зависит не только от необходимого ему объема процессорного времени, но и от того, как часто и как надолго процессор будет переключаться на исполнение других компонентов.

Для проверки гарантий своевременности завершения выполнения программных компонентов строятся вычислительные модели приложений в формате, допускающем использование алгоритмов оценки выполнимости приложений и составляющих их задач. Применение допустимых форматов предполагает наложение ряда ограничений на методы планирования и структуру взаимодействия программных компонентов.

Задачи и задания. Программное приложение СРВ строится в виде комплекса задач $\tau_1, \tau_2, \ldots, \tau_n$ — программных модулей, каждый из которых представляет собой внутренне замкнутую по передачам управления последовательную программу [5]. Составляющие комплекс задачи являются кооперативными: они ориентированы на совместное функционирование для достижения общих целей, стоящих перед системой.

Вычислительные модели, предназначенные для анализа выполнимости задач, представляются в виде, отвлеченном от содержательной составляющей обрабатываемых данных. Назначение таких моделей состоит в демонстрации ограничений, накладываемых на порядок следования системных событий, возникающих в ходе работы СРВ. При этом под *системным событием* понимается изменение условий распределения системных ресурсов (в первую очередь — ресурса процессорного времени). При рассмотрении вопросов функционирования СРВ различают два понятия:

— задача (task) — статический объект τ_i, элемент статической структуры программного приложения;

— задание (job) — динамический объект $\tau_i^{(j)}$, $j=\overline{1,\infty}$, процесс исполнения задачи τ_i.

Символами $\tau_i^{(1)}$, $\tau_i^{(2)}$, $\tau_i^{(3)}$, … обозначаются последовательно возникающие (порождаемые) задания на выполнение задачи τ_i.

В простейших вычислительных моделях каждая задача характеризуется следующими параметрами:

T_i — период активизации задачи τ_i,

C_i — максимальный объем процессорного времени для выполнения задани $\tau_i^{(j)}$,

D_i — допустимая продолжительность выполнения каждого из заданий $\tau_i^{(j)}$,

$\mathrm{prio}(\tau_i)$ — приоритет задачи τ_i.

Пусть $t_{\text{в}}\left(\tau_i^{(j)}\right)$ и $t_{\text{з}}\left(\tau_i^{(j)}\right)$ — соответственно моменты времени возникновения и завершения задания $\tau_i^{(j)}$. Задание $\tau_i^{(j)}$ существует (является *действующим*) в рамках интервала времени $\left[t_{\text{в}}\left(\tau_i^{(j)}\right),\ t_{\text{з}}\left(\tau_i^{(j)}\right)\right]$. Продолжительность интервала существования

$$r_i^{(j)} = t_{\text{з}}\left(\tau_i^{(j)}\right) - t_{\text{в}}\left(\tau_i^{(j)}\right)$$

называется *временем отклика* задания $\tau_i^{(j)}$.

Время отклика однотипных заданий может варьироваться: при $j \neq k$ значения $r_i^{(j)}$ и $r_i^{(k)}$ могут существенно различаться. Такое различие возникает по двум причинам. Во-первых, вследствие различий в объемах вычислений, выполняемых в рамках заданий $\tau_i^{(j)}$ и $\tau_i^{(k)}$. Если обозначить символом $c_i^{(j)}$ требуемый для выполнения задания $\tau_i^{(j)}$ объем процессорного времени (суммарную продолжительность непосредственного облуживания задания центральным процессором), то при $j \neq k$ значения $c_i^{(j)}$ и $c_i^{(k)}$ могут не совпадать. Другая возможная причина различий величин $r_i^{(j)}$ и $r_i^{(k)}$ состоит в том, что аппаратные и информационные ресурсы, требуемые для выполнения задачи τ_i (в частности, ресурс процессора), могут, в рамках интервалов существования $\tau_i^{(j)}$ и $\tau_i^{(k)}$, быть привлечены к выполнению других заданий. Введенная выше величина R_i (время отклика задачи) формально определяется как максимально возможное значение времени отклика экземпляров задачи τ_i:

$$R_i = \max\{j=1, 2, \ldots | r_i^{(j)}\}. \qquad (1)$$

Задачу τ_i называют асинхронной задачей, если ее задания $\tau_i^{(j)}$ следуют не строго периодически, при этом не чаще, чем с интервалом T_i:

$$t_{\text{в}}\left(\tau_i^{(j+1)}\right) - t_{\text{в}}\left(\tau_i^{(j)}\right) \geq T_i.$$

Спецификации функций СРВ содержат ограничения на время отклика заданий: для всех заданий типа τ_i допустимая продолжительность интервала существования ограничена величиной D_i, т.е.

$$t_{\text{з}}\left(\tau_i^{(j)}\right) - t_{\text{в}}\left(\tau_i^{(j)}\right) \leq D_i.$$

В условиях перекрытия интервалов существования заданий решение о том, какому из заданий следует предоставить процессор, принимается в соответствии с используемым порядком назначения приоритетов — дисциплиной планирования.

Дисциплина планирования определяется способом назначения целочисленных приоритетов $\mathrm{prio}\left(\tau_i^{(k)}\right)$ действующих заданий: если $\mathrm{prio}\left(\tau_x^{(k)}\right) < \mathrm{prio}\left(\tau_y^{(l)}\right)$, то задание $\tau_x^{(k)}$ считается более приоритетным, чем $\tau_y^{(l)}$. Ниже рассматриваются системы со статическими приоритетами задач: значения $\mathrm{prio}(\tau_i)$ определяются для каждой из задач при конструировании системы и являются основными параметрами планирования.

Одна из особенностей дисциплины планирования — возможность *вытеснения* текущего задания. Механизм вытеснения действует следующим образом: если в момент времени $t_{\text{в}}\left(\tau_x^{(k)}\right)$ текущим является задание типа $\tau_y^{(l)}$, приоритет которого ниже, чем $\mathrm{prio}\left(\tau_x^{(k)}\right)$, то процессор переключается на исполнение задания $\tau_x^{(k)}$ (оно становится текущим), а задание $\tau_y^{(l)}$ переводится в состояние „*вытеснено*“ (контекст задания $\tau_y^{(l)}$ консервируется, с тем чтобы через какое-то время в соответствии с принятой дисциплиной планирования его исполнение было продолжено).

В ряду составляющих приложение задач могут быть выделены *невытесняемые* задачи, отличающиеся тем, что соответствующие им задания не переводятся в состояние „вытеснено“, даже если на интервале их существования активизируются более приоритетные задачи. Пусть в системе независимых задач все задачи τ_1, τ_2, τ_3, … являются невытесняемыми. Известно, что в таком случае время отклика [2] каждой из задач определяется методом последовательных приближений:

$$R_i = C_i + \sum_{\mathrm{prio}(\tau_j)<\mathrm{prio}(\tau_i)} C_j \left\lceil \frac{R_i}{T_j} \right\rceil + \max\{\mathrm{prio}(\tau_j) > \mathrm{prio}(\tau_i) | C_j\}, \qquad (2)$$

где суммирование осуществляется по всем задачам τ_j, более приоритетным по отношению к задачам τ_i; символом $\lceil x \rceil$ обозначается ближайшее к x сверху целое число.

CAN-интерфейс. Многие микроконтроллеры имеют в своем составе интерфейсные модули, реализующие коммуникационный интерфейс CAN [5, 6]. Такие микроконтроллеры используются в автомобилях, системах автоматизации технологических процессов, медицинской аппаратуре и других системах реального времени.

При использовании локальной сети, оснащенной CAN-интерфейсом, элементарные сообщения, передаваемые в однопроводную коммуникационную линию, разделяются интервалами пассивного состояния линии (логический нуль). Структура стандартного сообщения в упрощенном виде представлена на рис. 1. Сообщение начинается стартовым фреймом, за которым следуют информационные поля и стоповый фрейм.

Рис. 1

Узлы сети могут начинать передачу сообщения только тогда, когда линия находится в пассивном состоянии. При появлении в линии стартового фрейма все узлы синхронизируются его фронтом. Может оказаться, что два и более узла одновременно генерировали старто-

вый фрейм, — возникает конфликт по запросу на использование коммуникационной линии. Этот конфликт должен быть устранен в ходе формирования поля арбитража.

Поле арбитража длиной 11 бит имеет двоякое назначение. Во-первых, код, размещаемый в поле арбитража, идентифицирует тип передаваемого сообщения. Во-вторых, значение этого кода определяет приоритет сообщения: нулевой код соответствует наиболее приоритетным сообщениям; чем больше значение двоичного числа, размещаемого в поле арбитража, тем ниже приоритет передаваемого сообщения. Типы передаваемых сообщений распределены между узлами сети, так что если два узла сети одновременно начали передавать сообщения, то это сообщения различных типов.

Пассивное состояние коммуникационной линии соответствует символу „1“ (рецессивный бит), активное состояние — символу „0“ (доминантный бит). В рамках поля арбитража несколько узлов осуществляют попытку передачи очередного сообщения. Каждый передающий узел контролирует состояние коммуникационной линии в ходе передачи каждого бита. Узел, передающий пассивный бит, прекращает попытку передачи сообщения, если он обнаруживает, что существует конкурирующий узел, который в этот момент передает активный бит. К моменту, когда завершается передача битов поля арбитража, только один узел продолжает передачу. Таким образом, в ходе дальнейшей передачи наиболее приоритетного сообщения предотвращается одновременная передача сообщений, генерируемых другими узлами.

Четырехбитовый код поля длины задает число байт (от 0 до 15), размещаемых в поле данных. За полями длины, данных и контрольной суммы (CRC) следует поле ACK — поле подтверждения приема. Это означает, что принимающий узел сети, обнаружив адресованное ему сообщение, помечает его как принятое в рамках формата передаваемого сообщения, а не последующих сообщений. Такое решение повышает оперативность обмена данными.

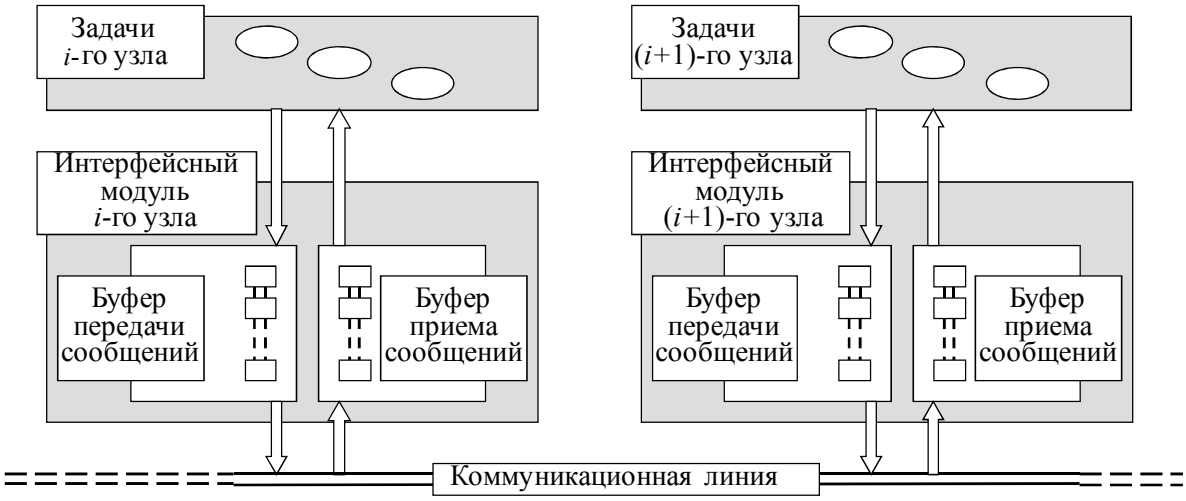

Рис. 2

Порядок доставки сообщений через сеть иллюстрируется схемой, представленной на рис. 2. Отдельный узел сети содержит ряд прикладных задач, которые генерируют сообщения, адресуемые либо соседним задачам, либо задачам, содержащимся в других узлах. В последнем случае сгенерированное сообщение направляется в интерфейсный модуль и размещается в его буфере. В буфере интерфейсного модуля сообщения упорядочиваются по приоритетам. В момент, когда линия переходит в пассивное состояние, интерфейсный модуль приступает к передаче самого приоритетного из сообщений, находящихся в его буфере. Эта попытка окажется удачной, если одновременно интерфейсный модуль какого-либо другого узла не пытается передать более приоритетное сообщение.

Проверка гарантий своевременности доставки сообщений. Определим *время доставки* сообщения как длительность временного интервала, в течение которого сообщение

конкретного типа, порожденное прикладной задачей, гарантированно доставляется в узел сети с задачей-адресатом. Как следует из вышерассмотренного порядка доставки сообщения через сеть с CAN-интерфейсом, при заданной частоте передачи битов по коммуникационной линии время доставки определяется не только длиной поля данных, но и продолжительностью ожидания момента освобождения коммуникационной линии от передачи более приоритетных сообщений. Иными словами, сформированные задачами сообщения конкурируют за возможность получения ресурса коммуникационной линии аналогично тому, как невытесняемые задания в каждом из узлов системы конкурируют за предоставление ресурса процессора. Наличие такой аналогии позволяет распространить рассмотренный выше метод оценки времени отклика невытесняемых задач на время доставки сообщений по сети с CAN-интерфейсом.

Пусть m_i — тип сообщений, передаваемых по сети; $m_i^{(j)}$ — очередное сообщение этого типа; $t_{\text{в}}\left(m_i^{(j)}\right)$ и $t_{\text{з}}\left(m_i^{(j)}\right)$ — моменты поступления $m_i^{(j)}$ в интерфейсные модули соответственно передающего и приемного узлов; продолжительность $r_i^{(j)}$ доставки экземпляра $m_i^{(j)}$ сообщения типа m_i определяется разностью $t_{\text{з}}\left(m_i^{(j)}\right) - t_{\text{в}}\left(m_i^{(j)}\right)$. Период T_i следования сообщений m_i равен минимально возможному интервалу между поступлениями однотипных сообщений в интерфейсный модуль передающего узла. Приоритет сообщений типа m_i определяется кодом типа сообщения, размещаемым в поле арбитража. Величиной, аналогичной величине C_i объема процессорного времени для выполнения заданий $\tau_i^{(j)}$, является длительность временного интервала, в течение которого коммуникационная линия передает сообщения типа m_i:

$$C_i = \frac{(44 + 8N_i)}{v},$$

где v_i — скорость передачи данных по коммуникационной линии (число бит в секунду), N_i — число байт в поле данных сообщений типа m_i.

Тогда время доставки R_i сообщений типа m_i формально определяется выражением (1), а его значение — решением уравнения (2). Иными словами, уравнение (2), используемое для оценки времени отклика в системе невытесняемых задач, может быть применено и для оценки времени доставки сообщений в сетях с CAN-интерфейсом.

Заключение. Использование предложенного метода вычисления максимального времени доставки сообщений в локальных сетях с CAN-интерфейсом позволяет строить программные приложения для таких сетей с проверкой гарантий своевременной реализации прикладных функций, исполнение которых разнесено по различным узлам системы. Таким образом может быть достигнуто надежное, предсказуемое поведение программных приложений реального времени, распределенных по узлам локальной сети, а также эффективное использование аппаратных средств, обеспечивающих не только своевременную обработку актуальной информации в узлах сети, но и своевременный обмен данными между компонентами, связанными коммуникационными линиями с CAN интерфейсом.

СПИСОК ЛИТЕРАТУРЫ

1. *Liu C. L.*, *Layland J. W.* Scheduling algorithms for multiprogramming in hard real-time environment // J. of the ACM. 1973. Vol. 20. P. 46—61.

2. *Liu J. W. S.* Real-Time Systems. NJ: Prentice Hall, 2000. 590 p.

3. *Данилов М. В.* Методы планирования задач в системах реального времени // Программные продукты и системы. 2001. № 4. С. 28—35.

4. *Никифоров В. В.* Выполнимость приложений реального времени на многоядерных процессорах // Тр. СПИИРАН; Под общ. ред. *Р. М. Юсупова*. 2009. Вып. 8. С. 255—284.

5. *Никифоров В. В.* Разработка программных средств для встроенных систем. СПб: СПбГЭТУ, 2000. 74 с.

6. Введение в CAN 2.0B интерфейс. М.: ООО Микро-Чип, 2001: [Электронный ресурс]: <www.microchip.ru>.

Сведения об авторах

Виктор Викентьевич Никифоров — д-р техн. наук, профессор; Санкт-Петербургский институт информатики и автоматизации РАН, лаборатория технологий и систем программирования; E-mail: nik@iias.spb.su

Вячеслав Иванович Шкиртиль — канд. техн. наук, доцент; Санкт-Петербургский институт информатики и автоматизации РАН, лаборатория технологий и систем программирования; E-mail: jvatlas@mail.rcom.ru

Рекомендована СПИИРАН

Поступила в редакцию
08.02.10 г.