Распределенные алгоритмы

ЛЕКТОР: В.А. Захаров

Лекция 3.

Коммуникационные протоколы.
Ошибки, возникающие при передаче сообщений.
Симметричный протокол раздвижного окна.
Устройство протокола раздвижного окна.
Принципы обоснования корректности протоколов
Обоснование корректности протокола раздвижного окна.
Модификации протокола.

Коммуникационные протоколы

Основное назначение коммуникационного протокола — передача данных , т.е. получение информации от одного узла сети и доставка ее по назначению другому узлу сети.

При передаче данных возможны ошибки (потеря, дублирование, искажение).

Эти ошибки нужно обнаруживать и исправлять.

Для этого в протоколе ведется учет состояния информации.

Для использования состояния информации применяется управление соединением — инициализация и аннулирование состояния информации.

Инициализация называется установлением соединения, а аннулирование — завершением соединения.

Коммуникационные протоколы

Мы рассмотрим один из таких протоколов — **симметричный протокол раздвижного окна** (Balanced Sliding Window Protocol).

Он предназначен для обмена данными между двумя узлами сети, которые имеют прямое физическое соединение (например, по оптоволоконному кабелю).

Это — вполне асинхронный протокол, относящийся к уровню управления передачей данных — второму уровню эталонной модели OSI.

Мы не будем рассматривать управление соединением для этого протокола. Предполагается, что физическое соединение обычно функционирует непрерывно в течение очень долгого времени, а не устанавливается и завершается периодически.

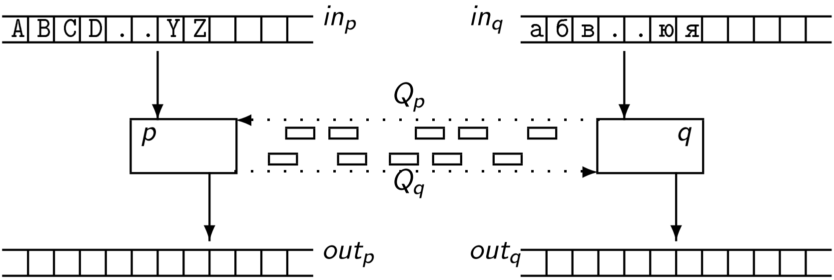
Ошибки, возникающие при передаче сообщений

При физическом соединении сообщения не могут обгонять друг друга, и они также не могут дублироваться. Поэтому рассматриваются только ошибки потери сообщения.

Содержание сообщения, передаваемого по физическому каналу связи, может быть повреждено. Тем не менее можно предполагать, что процесс-получатель способен обнаруживать искажения сообщений, например, при помощи счетчиков четности или кодирования с исправлением ошибок (коды Хэмминга, Рида-Маллера и др.).

Симметричный протокол раздвижного окна

Постановка задачи.



Процессам p и q требуется передать потоки данных in_p и in_q друг другу и записать полученные данные в массивы out_p и out_q.
В канале связи возможны помехи, приводящие к потере сообщений.

Общая идея алгоритма

Общая идея алгоритма

- Входные данные одного процесса служат для подтверждения получения сообщений от другого процесса.

Общая идея алгоритма

- Входные данные одного процесса служат для подтверждения получения сообщений от другого процесса.
- Сообщения — *пакеты* — это наборы вида $\langle \mathbf{pack}, w, i \rangle$, где w — *информационное слово* , а i — *порядковый номер* пакета.

Общая идея алгоритма

- Входные данные одного процесса служат для подтверждения получения сообщений от другого процесса.
- Сообщения — *пакеты* — это наборы вида $\langle \mathbf{pack}, w, i \rangle$, где w — *информационное слово* , а i — *порядковый номер* пакета.
- Пакет $\langle \mathbf{pack}, w, i \rangle$, отправленный процессом p , передает слово $w = in_p[i]$ процессу q и подтверждает успешное получение ряда пакетов, отправленных процессом q .

Общая идея алгоритма

- Входные данные одного процесса служат для подтверждения получения сообщений от другого процесса.
- Сообщения — *пакеты* — это наборы вида $\langle \mathbf{pack}, w, i \rangle$, где w — *информационное слово* , а i — *порядковый номер* пакета.
- Пакет $\langle \mathbf{pack}, w, i \rangle$, отправленный процессом p , передает слово $w = in_p[i]$ процессу q и подтверждает успешное получение ряда пакетов, отправленных процессом q .
- Процесс p может «опережать» процесс q на некоторое заданное число пакетов ℓ_p , если мы постановим, что отправление пакета $\langle \mathbf{pack}, w, i \rangle$ процессом p подтверждает получение слов с номерами $0, 1, \ldots, (i - \ell_p)$ от процесса q .

Общая идея алгоритма

- Входные данные одного процесса служат для подтверждения получения сообщений от другого процесса.
- Сообщения — *пакеты* — это наборы вида $\langle \textbf{pack}, w, i \rangle$, где w — *информационное слово* , а i — *порядковый номер* пакета.
- Пакет $\langle \textbf{pack}, w, i \rangle$, отправленный процессом p , передает слово $w = in_p[i]$ процессу q и подтверждает успешное получение ряда пакетов, отправленных процессом q .
- Процесс p может «опережать» процесс q на некоторое заданное число пакетов ℓ_p , если мы постановим, что отправление пакета $\langle \textbf{pack}, w, i \rangle$ процессом p подтверждает получение слов с номерами $0, 1, \ldots, (i - \ell_p)$ от процесса q .
- Константы опережения ℓ_p и ℓ_q известны процессам p и q .

Общая идея алгоритма

Таким образом, в протоколе соблюдаются два принципа:

Общая идея алгоритма

Таким образом, в протоколе соблюдаются два принципа:

1. Процесс p может отправить слово $in_p[i]$ (в пакете $\langle \mathbf{pack}, in_p[i], i \rangle$) только после того, как будут занесены в память все слова, начиная с $out_p[0]$ и оканчивая $out_p[i - \ell_p]$, т.е. когда будет выполняться неравенство $i < s_p + \ell_p$, где $s_p = \min\{j : out_p[j] = udef\}$.

Общая идея алгоритма

Таким образом, в протоколе соблюдаются два принципа:

1. Процесс p может отправить слово $in_p[i]$ (в пакете $\langle \mathbf{pack}, in_p[i], i \rangle$) только после того, как будут занесены в память все слова, начиная с $out_p[0]$ и оканчивая $out_p[i - \ell_p]$, т.е. когда будет выполняться неравенство $i < s_p + \ell_p$, где $s_p = \min\{j : out_p[j] = udef\}$.
2. Как только p получает пакет $\langle \mathbf{pack}, w, i \rangle$, отпадает необходимость в повторной передаче слов, начиная с $in_p[0]$ и оканчивая $in_p[i - \ell_q]$.

Симметричный протокол раздвижного окна

$$
\begin{array}{llll}
\textbf{var}\ s_p,\ a_p & : \textbf{integer} & & \textbf{init}\ 0,\ 0\ ; \\
\quad in_p & : \text{array of word} & & (*\ \text{Data to be sent}\ *)\ ; \\
\quad out_p & : \text{array of word} & & \textbf{init}\ udef,\ udef,\ \ldots\ ;
\end{array}
$$

$\mathbf{S}_p$: $\{\ a_p \leq i < s_p + \ell_p\ \}$ **begin** send $\langle \mathbf{pack}, in_p[i], i \rangle$ to q **end**

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i \rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i \rangle$;

  if $out_p[i] = udef$ **then**

    begin $out_p[i] := w$; $a_p := \max(a_p, i - \ell_q + 1)$;

      $s_p := \min\{j |\ out_p[j] = udef\}$

    end

  (* **else** игнорировать повторное получение пакета *)

end

$\mathbf{L}_p$: $\{\ \langle \mathbf{pack}, w, i \rangle \in Q_p\ \}$

begin $Q_p := Q_p \setminus \{\langle \mathbf{pack}, w, i \rangle\}$ **end**

Симметричный протокол раздвижного окна

1. Действие $\mathbf{S}_p$ осуществляет отправление i-го входного слова процесса p,
2. Действие $\mathbf{R}_p$ осуществляет прием слова процессом p,
3. Действие $\mathbf{L}_p$ моделирует потерю пакета, адресатом которого является процесс p.

Симметричный протокол раздвижного окна

var s_p, a_p : integer init 0, 0 ;
in_p : array of word (* Data to be sent *) ;
out_p : array of word init $udef$, $udef$, ... ;

$\mathbf{S}_p$: $\{\ a_p \leq i < s_p + \ell_p\ \}$ **begin** send $\langle \mathbf{pack}, in_p[i], i\rangle$ to q **end**

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$
begin receive $\langle \mathbf{pack}, w, i\rangle$;
if $out_p[i] = udef$ **then**
begin $out_p[i] := w$; $a_p := \max(a_p, i - \ell_q + 1)$;
$s_p := \min\{j |\ out_p[j] = udef\}$
end
(* **else** игнорировать повторное получение пакета *)
end

$\mathbf{L}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$
begin $Q_p := Q_p \setminus \{\langle \mathbf{pack}, w, i\rangle\}$ **end**

Симметричный протокол раздвижного окна

$\mathbf{S}_p$: $\{\ a_p \le i < s_p + \ell_p\ \}$ **begin** send $\langle \mathbf{pack}, in_p[i], i \rangle$ to q **end**

Данные для отправления выбираются из раздвижного окна

$$a_p \le i < s_p + \ell_p$$

Симметричный протокол раздвижного окна

S$_p$: $\{\ a_p \le i < s_p + \ell_p\ \}$ **begin** send $\langle$**pack**$, in_p[i], i\rangle$ to q **end**

Данные для отправления выбираются из раздвижного окна

$$a_p \le i < s_p + \ell_p$$

Предполагается, что

Левая створка : $a_p = \min\{i : out_p[i + \ell_q] = udef\}$ —
наименьший номер того элемента в массиве in_p ,
получение которого еще не подтвердил процесс q .
Значит, $in_p[a_p]$ еще нужно отправлять.

Симметричный протокол раздвижного окна

S$_p$: $\{\ a_p \le i < s_p + \ell_p\ \}$ **begin** send $\langle$**pack**$, in_p[i], i\rangle$ to q **end**

Данные для отправления выбираются из раздвижного окна

$$a_p \le i < s_p + \ell_p$$

Предполагается, что

Левая створка : $a_p = \min\{i : out_p[i + \ell_q] = udef\}$ — наименьший номер того элемента в массиве in_p , получение которого еще не подтвердил процесс q . Значит, $in_p[a_p]$ еще нужно отправлять.

Правая створка : $s_p + \ell_p - 1$, где $s_p = \min\{j : out_p[j] = udef\}$ — наименьший номер того элемента в массиве out_p , в который еще не записаны полученные данные. Значит, $in_p[s_p + \ell_p]$ еще рано отправлять в качестве подтверждения о получении данных.

Симметричный протокол раздвижного окна

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i\rangle$;

  if $out_p[i] = udef$ **then**

   begin $out_p[i] := w$;

    $a_p := \max\left(a_p, i - \ell_q + 1\right)$;

    $s_p := \min\left\{j \middle| out_p[j] = udef\right\}$

   end

  (* **else** игнорировать повторное получение пакета *)

end

Симметричный протокол раздвижного окна

$\mathbf{R}_p$: $\{ \ \langle \mathbf{pack}, w, i \rangle \in Q_p \ \}$
    begin receive $\langle \mathbf{pack}, w, i \rangle$;
        if $out_p[i] = udef$ **then**
            begin $out_p[i] := w$;
                $a_p := \max(a_p, i - \ell_q + 1)$;
                $s_p := \min\{j \mid out_p[j] = udef\}$
            end
        (* **else** игнорировать повторное получение пакета *)
    end

Получив сообщение, процесс вначале проверяет, не было ли идентичное сообщение получено ранее (в этом случае процесс имеет дело с повторным получением сообщения).

Симметричный протокол раздвижного окна

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$
begin receive $\langle \mathbf{pack}, w, i\rangle$;
  if $out_p[i] = udef$ **then**
   begin $out_p[i] := w$;
     $a_p := \max(a_p, i - \ell_q + 1)$;
     $s_p := \min\{j \mid out_p[j] = udef\}$
   end
  (* **else** игнорировать повторное получение пакета *)
end

Получив сообщение, процесс вначале проверяет, не было ли идентичное сообщение получено ранее (в этом случае процесс имеет дело с повторным получением сообщения).
Если это не так, то слово, содержащееся в сообщении, записывается в выходной массив, и при этом значения переменных a_p и s_p изменяются.

Симметричный протокол раздвижного окна

$\mathbf{L}_p$: $\{ \langle \mathbf{pack}, w, i \rangle \in Q_p \}$
begin $Q_p := Q_p \setminus \{\langle \mathbf{pack}, w, i \rangle\}$ **end**

Симметричный протокол раздвижного окна

$\mathbf{L}_p$: $\{ \langle \mathbf{pack}, w, i \rangle \in Q_p \}$
$\quad$ **begin** $Q_p := Q_p \setminus \{\langle \mathbf{pack}, w, i \rangle\}$ **end**

Моделирование потери сообщения проводится путем удаления произвольного сообщения из множества сообщений Q_p, пребывающих на этапе пересылки от процесса q к процессу p.

Симметричный протокол раздвижного окна

Что плохого может случиться?

Симметричный протокол раздвижного окна

Что плохого может случиться?

1. Створки окон обоих процессов могут «захлопнуться», и процессы будут обречены (безуспешно) ожидать сообщений друг от друга (блокировка , deadlock);

Симметричный протокол раздвижного окна

Что плохого может случиться?

1. Створки окон обоих процессов могут «захлопнуться», и процессы будут обречены (безуспешно) ожидать сообщений друг от друга (блокировка , deadlock);
2. Створки окон могут «застыть», и процессы будут обречены передавать одни и те же сообщения (активный тупик , livelock);

Симметричный протокол раздвижного окна

Что плохого может случиться?

1. Створки окон обоих процессов могут «захлопнуться», и процессы будут обречены (безуспешно) ожидать сообщений друг от друга (блокировка , deadlock);
2. Створки окон могут «застыть», и процессы будут обречены передавать одни и те же сообщения (активный тупик , livelock);
3. Данные могут быть потеряны при передаче, и процесс не заметит этого;

Симметричный протокол раздвижного окна

Что плохого может случиться?

1. Створки окон обоих процессов могут «захлопнуться», и процессы будут обречены (безуспешно) ожидать сообщений друг от друга (блокировка , deadlock);
2. Створки окон могут «застыть», и процессы будут обречены передавать одни и те же сообщения (активный тупик , livelock);
3. Данные могут быть потеряны при передаче, и процесс не заметит этого;
4. Процесс может «забыть» передать данные;

Симметричный протокол раздвижного окна

Что плохого может случиться?

1. Створки окон обоих процессов могут «захлопнуться», и процессы будут обречены (безуспешно) ожидать сообщений друг от друга (блокировка , deadlock);
2. Створки окон могут «застыть», и процессы будут обречены передавать одни и те же сообщения (активный тупик , livelock);
3. Данные могут быть потеряны при передаче, и процесс не заметит этого;
4. Процесс может «забыть» передать данные;
5. Створки окна могут раздвигаться, отдалаяясь друг от друга неограниченно широко.

Симметричный протокол раздвижного окна

Требования корректности.

Нужно доказать, что протокол работает правильно, т.е. каждое слово из входного массива in_p процесса p будет рано или поздно записано в выходной массив out_q процесса q, и наоборот.

Симметричный протокол раздвижного окна

Требования корректности.

Нужно доказать, что протокол работает правильно, т.е. каждое слово из входного массива in_p процесса p будет рано или поздно записано в выходной массив out_q процесса q, и наоборот.

Более строго это выражается двумя требованиями.

1. *Безопасная доставка сообщений.* В каждой достижимой конфигурации протокола выполняются соотношения

$$out_p[0..s_p - 1] = in_q[0..s_p - 1] \quad \text{и} \quad out_q[0..s_q - 1] = in_p[0..s_q - 1].$$

Симметричный протокол раздвижного окна

Требования корректности.

Нужно доказать, что протокол работает правильно, т.е. каждое слово из входного массива in_p процесса p будет рано или поздно записано в выходной массив out_q процесса q , и наоборот.

Более строго это выражается двумя требованиями.

1. *Безопасная доставка сообщений.* В каждой достижимой конфигурации протокола выполняются соотношения

$$out_p[0..s_p-1] = in_q[0..s_p-1] \quad \text{и} \quad out_q[0..s_q-1] = in_p[0..s_q-1].$$

2. *Неизбежная доставка сообщений.* Для каждого целого числа $k \geq 0$, в ходе выполнения протокола будет достигнута конфигурация, в которой $s_p \geq k$ и $s_q \geq k$.

А как доказывается корректность протоколов?

Как обосновывать свойства систем переходов

Классификация свойств

Многие свойства распределенных алгоритмов, нуждающиеся в проверке, относятся к одному из двух типов:
условие безопасности и *условие живости* .

Как обосновывать свойства систем переходов

Классификация свойств

Многие свойства распределенных алгоритмов, нуждающиеся в проверке, относятся к одному из двух типов:
условие безопасности и *условие живости* .

Условие безопасности требует, чтобы каждая достижимая конфигурация в любом выполнении системы обладала определенным свойством.

Как обосновывать свойства систем переходов

Классификация свойств

Многие свойства распределенных алгоритмов, нуждающиеся в проверке, относятся к одному из двух типов:
условие безопасности и *условие живости* .

Условие безопасности требует, чтобы каждая достижимая конфигурация в любом выполнении системы обладала определенным свойством.

Условие живости требует, чтобы хотя бы одна достижимая конфигурация в любом выполнении системы обладала определенным свойством.

Свойства безопасности

Свойством безопасности алгоритма является всякое свойство, которое можно сформулировать в виде следующего предложения:

«Для любого выполнения алгоритма утверждение P истинно в каждой конфигурации выполнения».

Более кратко это свойство можно сформулировать так:

«Утверждение P всегда истинно».

Свойства безопасности

Свойством безопасности алгоритма является всякое свойство, которое можно сформулировать в виде следующего предложения:

«Для любого выполнения алгоритма утверждение P истинно в каждой конфигурации выполнения».

Более кратко это свойство можно сформулировать так:

«Утверждение P всегда истинно».

Основной прием, при помощи которого удается показать, что утверждение P всегда истинно, заключается в доказательстве того, что P является *инвариантом* .

Свойства безопасности

Пусть задана система переходов $S = (\mathcal{C}, \to, \mathcal{I})$ и рассматриваются некотрые свойства конфигураций P и Q .

Свойства безопасности

Пусть задана система переходов $S = (\mathcal{C}, \rightarrow, \mathcal{I})$ и рассматриваются некотрые свойства конфигураций P и Q .

Запись $P(\gamma)$, где γ — конфигурация, будет обозначать логическое выражение (формулу), принимающее истинное значение в том случае, если утверждение P справедливо для конфигурации γ , и ложное значение в противном случае.

Свойства безопасности

Пусть задана система переходов $S = (\mathcal{C}, \to, \mathcal{I})$ и рассматриваются некотрые свойства конфигураций P и Q .

Запись $P(\gamma)$, где γ — конфигурация, будет обозначать логическое выражение (формулу), принимающее истинное значение в том случае, если утверждение P справедливо для конфигурации γ , и ложное значение в противном случае.

Запись $\{P\} \to \{Q\}$ будет использоваться для обозначения того, что для каждого перехода $\gamma \to \delta$ в системе S истинность $P(\gamma)$ влечет истинность $Q(\delta)$.

Свойства безопасности

Пусть задана система переходов $S = (\mathcal{C}, \to, \mathcal{I})$ и рассматриваются некотрые свойства конфигураций P и Q .

Запись $P(\gamma)$, где γ — конфигурация, будет обозначать логическое выражение (формулу), принимающее истинное значение в том случае, если утверждение P справедливо для конфигурации γ , и ложное значение в противном случае.

Запись $\{P\} \to \{Q\}$ будет использоваться для обозначения того, что для каждого перехода $\gamma \to \delta$ в системе S истинность $P(\gamma)$ влечет истинность $Q(\delta)$.

Определение 3.1.

Утверждение P называется **инвариантом** системы S , если

1. $P(\gamma)$ истинно для каждой начальной конфигурации $\gamma \in \mathcal{I}$, и
2. $\{P\} \to \{P\}$.

Свойства безопасности

Теорема 3.1.

Если утверждение P является инвариантом системы переходов S, то для любого выполнения E системы S утверждение P будет истинно в каждой конфигурации этого выполнения.

Свойства безопасности

Теорема 3.1.

Если утверждение P является инвариантом системы переходов S, то для любого выполнения E системы S утверждение P будет истинно в каждой конфигурации этого выполнения.

Доказательство: Рассмотреть произвольное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ системы S и воспользоваться индукцией.

$\square$

Свойства безопасности

Теорема 3.1.

Если утверждение P является инвариантом системы переходов S, то для любого выполнения E системы S утверждение P будет истинно в каждой конфигурации этого выполнения.

Доказательство: Рассмотреть произвольное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ системы S и воспользоваться индукцией. $\square$

Теорема 3.2.

Если Q является инвариантом системы переходов S, и для каждой конфигурации $\gamma \in \mathcal{C}$ выполняется $Q(\gamma) \implies P(\gamma)$, то для любого выполнения системы S утверждение P будет истинно в каждой конфигурации выполнения.

Свойства безопасности

Задачи.

1. Верно ли, что утверждение, которое является истинным в каждой конфигурации любого выполнения, обязательно является инвариантом?

Свойства безопасности

Задачи.

1. Верно ли, что утверждение, которое является истинным в каждой конфигурации любого выполнения, обязательно является инвариантом?
2. Приведите пример такой системы переходов S и такого утверждения P, что P всегда истинно в системе S, но при этом не является инвариантом S.

Свойства безопасности

Задачи.

1. Верно ли, что утверждение, которое является истинным в каждой конфигурации любого выполнения, обязательно является инвариантом?
2. Приведите пример такой системы переходов S и такого утверждения P, что P всегда истинно в системе S, но при этом не является инвариантом S.
3. Предположим, что P_1 и P_2 — это инварианты системы S. Докажите, что $(P_1 \vee P_2)$ и $(P_1 \wedge P_2)$ также являются инвариантами.

Свойства живости

Свойством живости алгоритма является всякое свойство, которое можно сформулировать в виде следующего предложения:

«Для любого выполнения алгоритма утверждение P истинно в некоторой конфигурации выполнения».

Более кратко это свойство можно сформулировать так:

«Утверждение P когда-то обязательно становится истинным».

Свойства живости

Свойством живости алгоритма является всякое свойство, которое можно сформулировать в виде следующего предложения:

«Для любого выполнения алгоритма утверждение P истинно в некоторой конфигурации выполнения».

Более кратко это свойство можно сформулировать так:

«Утверждение P когда-то обязательно становится истинным».

Основной прием, при помощи которого удается показать, что утверждение P когда-то обязательно становится истинным, заключается в использовании *функций нормировки* и *отсутствия блокировки* .

Свойства живости

Рассмотрим систему переходов S и предикат P. Будем считать, что предикат **term** по определению принимает истинное значение в каждой заключительной конфигурации, и ложное значение в конфигурациях, которые не являются заключительными.

Свойства живости

Рассмотрим систему переходов S и предикат P. Будем считать, что предикат **term** по определению принимает истинное значение в каждой заключительной конфигурации, и ложное значение в конфигурациях, которые не являются заключительными. Обычно бывает нежелательно достичь заключительной конфигурации прежде, чем выполнится «целевой» предикат P; в этом случае говорят, что произошла *блокировка* . Если же цель достигнута, то выполнение можно завершить; в этом случае говорят, что произошло *правильное завершение* .

Свойства живости

Рассмотрим систему переходов S и предикат P. Будем считать, что предикат **term** по определению принимает истинное значение в каждой заключительной конфигурации, и ложное значение в конфигурациях, которые не являются заключительными. Обычно бывает нежелательно достичь заключительной конфигурации прежде, чем выполнится «целевой» предикат P; в этом случае говорят, что произошла *блокировка*. Если же цель достигнута, то выполнение можно завершить; в этом случае говорят, что произошло *правильное завершение*.

Определение 3.2.

Пусть задан признак правильного завершения выполнений P. Система S **правильно завершает выполнения** (или, иными словами, **свободна от блокировки**), если предикат $(\mathbf{term} \implies P)$ всегда является истинным для системы S.

Свойства живости

В основу функций нормировки положено математическое понятие *фундированного* множества.

Свойства живости

В основу функций нормировки положено математическое понятие *фундированного* множества.

Определение 3.3.

Частично упорядоченное множество $(W, <)$ называется **фундированным**, если из его элементов нельзя составить бесконечно убывающую последовательность

$$w_1 > w_2 > w_3 \cdots .$$

Свойства живости

В основу функций нормировки положено математическое понятие *фундированного* множества.

Определение 3.3.

Частично упорядоченное множество $(W, <)$ называется **фундированным**, если из его элементов нельзя составить бесконечно убывающую последовательность

$$w_1 > w_2 > w_3 \cdots .$$

Примеры фундированных множеств:

- множество натуральных чисел с обычным отношением порядка,
- множество всех наборов длины n, состоящих из натуральных чисел, с отношением лексикографического порядка.

Свойства живости

Отсутствие бесконечно убывающих последовательностей, состоящих из элементов фундированного множества, можно использовать для доказательства того, что некоторое утверждение P наверняка станет истинным. Для этого нужно показать, что существует функция f, отображающая $\mathcal{C}$ в фундированное множество W так, что каждый переход приводит к тому, что либо значение f уменьшается, либо P становится истинным.

Свойства живости

Отсутствие бесконечно убывающих последовательностей, состоящих из элементов фундированного множества, можно использовать для доказательства того, что некоторое утверждение P наверняка станет истинным. Для этого нужно показать, что существует функция f, отображающая $\mathcal{C}$ в фундированное множество W так, что каждый переход приводит к тому, что либо значение f уменьшается, либо P становится истинным.

Определение 3.4.

Пусть заданы система переходов S и утверждение P. Функция f, отображающая множество $\mathcal{C}$ в фундированное множество W, называется **функцией нормировки** (по отношению к P), если для каждого перехода $\gamma \to \delta$, либо выполняется $f(\gamma) > f(\delta)$, либо $P(\delta)$ принимает истинное значение.

Свойства живости

Теорема 3.3.

Пусть заданы система переходов S и утверждение P. Если S правильно завершает выполнения, и существует функция нормировки f (по отношению к P), то для каждого выполнения S утверждение P становится истинным в некоторой конфигурации.

Теорема 3.3.

Пусть заданы система переходов S и утверждение P. Если S правильно завершает выполнения, и существует функция нормировки f (по отношению к P), то для каждого выполнения S утверждение P становится истинным в некоторой конфигурации.

Доказательство:

Допустим, что есть <u>бесконечное</u> выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ системы S, в котором P всегда ложно.

Свойства живости

Теорема 3.3.

Пусть заданы система переходов S и утверждение P. Если S правильно завершает выполнения, и существует функция нормировки f (по отношению к P), то для каждого выполнения S утверждение P становится истинным в некоторой конфигурации.

Доказательство:

Допустим, что есть бесконечное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ системы S, в котором P всегда ложно.

По определению функции нормировки f, последовательность $s = (f(\gamma_0), f(\gamma_1), \ldots)$ является убывающей.

Свойства живости

Теорема 3.3.

Пусть заданы система переходов S и утверждение P. Если S правильно завершает выполнения, и существует функция нормировки f (по отношению к P), то для каждого выполнения S утверждение P становится истинным в некоторой конфигурации.

Доказательство:

Допустим, что есть бесконечное выполнение $E = (\gamma_0, \gamma_1, \gamma_2, \ldots)$ системы S, в котором P всегда ложно.

По определению функции нормировки f, последовательность $s = (f(\gamma_0), f(\gamma_1), \ldots)$ является убывающей.

Так как W — фундированное множество, последовательность s должна быть конечной. Противоречие. $\square$

Симметричный протокол раздвижного окна

var $s_p,\ a_p$: **integer** **init** 0, 0 ;
in_p : array of word (* Data to be sent *) ;
out_p : array of word **init** $udef$, $udef$, ... ;

$\mathbf{S}_p$: $\{\ a_p \le i < s_p + \ell_p\ \}$ **begin** send $\langle \mathbf{pack}, in_p[i], i \rangle$ to q **end**

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i \rangle \in Q_p\ \}$
begin receive $\langle \mathbf{pack}, w, i \rangle$;
if $out_p[i] = udef$ **then**
begin $out_p[i] := w$; $a_p := \max(a_p, i - \ell_q + 1)$;
$s_p := \min\{j \mid out_p[j] = udef\}$
end
(* **else** игнорировать повторное получение пакета *)
end

$\mathbf{L}_p$: $\{\ \langle \mathbf{pack}, w, i \rangle \in Q_p\ \}$
begin $Q_p := Q_p \setminus \{\langle \mathbf{pack}, w, i \rangle\}$ **end**

Симметричный протокол раздвижного окна

Рассмотрим утверждение

$$
\begin{array}{llr}
P \equiv & \forall i < s_p : \ out_p[i] \neq udef & (0p) \\
\wedge & \forall i < s_q : \ out_q[i] \neq udef & (0q) \\
\wedge & \forall i : \langle \mathbf{pack}, w, i \rangle \in Q_p \Rightarrow w = in_q[i] \wedge (i < s_q + \ell_q) & (1p) \\
\wedge & \forall i : \langle \mathbf{pack}, w, i \rangle \in Q_q \Rightarrow w = in_p[i] \wedge (i < s_p + \ell_p) & (1q) \\
\wedge & \forall i : out_p[i] \neq udef \Rightarrow out_p[i] = in_q[i] \wedge (a_p > i - \ell_q) & (2p) \\
\wedge & \forall i : out_q[i] \neq udef \Rightarrow out_q[i] = in_p[i] \wedge (a_q > i - \ell_p) & (2q) \\
\wedge & a_p \leq s_q & (3p) \\
\wedge & a_q \leq s_p & (3q)
\end{array}
$$

Симметричный протокол раздвижного окна

Рассмотрим утверждение

$$
\begin{array}{llr}
P \equiv & \forall i < s_p : \ \ out_p[i] \neq udef & (0p) \\
\wedge & \forall i < s_q : \ \ out_q[i] \neq udef & (0q) \\
\wedge & \forall i : \langle \mathbf{pack}, w, i \rangle \in Q_p \Rightarrow w = in_q[i] \wedge (i < s_q + \ell_q) & (1p) \\
\wedge & \forall i : \langle \mathbf{pack}, w, i \rangle \in Q_q \Rightarrow w = in_p[i] \wedge (i < s_p + \ell_p) & (1q) \\
\wedge & \forall i : out_p[i] \neq udef \Rightarrow out_p[i] = in_q[i] \wedge (a_p > i - \ell_q) & (2p) \\
\wedge & \forall i : out_q[i] \neq udef \Rightarrow out_q[i] = in_p[i] \wedge (a_q > i - \ell_p) & (2q) \\
\wedge & a_p \leq s_q & (3p) \\
\wedge & a_q \leq s_p & (3q)
\end{array}
$$

Теорема 3.4.

Утверждение P является инвариантом алгоритма раздвижного окна.

Доказательство Теоремы 3.4.

Воспользуемся определением инварианта и покажем, что

1. $P(\gamma)$ истинно для каждой начальной конфигурации $\gamma \in \mathcal{I}$, и
2. $\{P\} \rightarrow \{P\}$.

1. **Базис.**
   В начальной конфигурации очереди Q_p и Q_q пусты, для всякого i, значения $out_p[i]$ и $out_q[i]$ равны $udef$, значения переменных a_p, a_q, s_q и s_p равны 0.

   Отсюда следует истинность утверждения P для каждой начальной конфигурации γ.

Доказательство Теоремы 3.4.

2. **Индуктивный переход.**
   Далее рассмотрим поочередно все переходы протокола и покажем, что при каждом из них утверждение P сохраняет истинность.
   При этом будем иметь в виду, что значения массивов in_p и in_q никогда не изменяются.

Доказательство Теоремы 3.4.

2. **Индуктивный переход.**
   Далее рассмотрим поочередно все переходы протокола и покажем, что при каждом из них утверждение P сохраняет истинность.
   При этом будем иметь в виду, что значения массивов in_p и in_q никогда не изменяются.

   Обоснование $\{P\} \rightarrow \{P\}$ проведем на примере действия $\mathbf{R}_p$ приема сообщения процессом p .

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i\rangle$;

  if $out_p[i] = udef$ **then**

   begin $out_p[i] := w$;

    $a_p := \max\left(a_p, i - \ell_q + 1\right)$;

    $s_p := \min\{j |\, out_p[j] = udef\}$

   end

end

Доказательство Теоремы 3.4.

R_p: $\{ \langle \mathbf{pack}, w, i \rangle \in Q_p \}$

begin receive $\langle \mathbf{pack}, w, i \rangle$;

 if $out_p[i] = udef$ **then**

 begin $out_p[i] := w$;

 $a_p := \max(a_p, i - \ell_q + 1)$;

 $s_p := \min\{j | \, out_p[j] = udef\}$

 end

end

Условие (0p): $\forall i < s_p : \;\; out_p[i] \neq udef$

остается верным после выполнения оператора

$s_p := \min\{j | \, out_p[j] = udef\}$.

Доказательство Теоремы 3.4.

R$_p$: { $\langle \mathbf{pack}, w, i\rangle \in Q_p$ }
begin receive $\langle \mathbf{pack}, w, i\rangle$;
if $out_p[i] = udef$ **then**
begin $out_p[i] := w$;
$a_p := \max(a_p, i - \ell_q + 1)$;
$s_p := \min\{j \mid out_p[j] = udef\}$
end
end

Условие (0q): $\forall i < s_q: \ out_q[i] \neq udef$
остается верным, поскольку действие $\mathbf{R}_p$ не изменяет значения переменной s_q и не записывает в массив $out_q[i]$ неопределенного значения $udef$.

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i\rangle$;

if $out_p[i] = udef$ **then**

begin $out_p[i] := w$;

$a_p := \max\left(a_p, i - \ell_q + 1\right)$;

$s_p := \min\left\{j |\, out_p[j] = udef\right\}$

end

end

Условие (1p):

$\forall i : \langle \mathbf{pack}, w, i\rangle \in Q_p \implies w = in_q[i] \wedge (i < s_q + \ell_q)$

сохраняет истинность, поскольку действие $\mathbf{R}_p$ не добавляет новых пакетов в очередь Q_p и не уменьшает значения переменной s_q .

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{ \langle \mathbf{pack}, w, i \rangle \in Q_p \}$

```
begin receive ⟨pack, w, i⟩ ;
        if out_p[i] = udef then
           begin out_p[i] := w ;
                    a_p := max(a_p, i − ℓ_q + 1) ;
                    s_p := min{j | out_p[j] = udef}
           end
end
```

Условие (1q):

$\forall i : \langle \mathbf{pack}, w, i \rangle \in Q_q \implies w = in_p[i] \wedge (i < s_p + \ell_p)$

сохраняет истинность, поскольку действие $\mathbf{R}_p$ не добавляет новых пакетов в очередь Q_q и не уменьшает значения переменной s_p.

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i\rangle$;

if $out_p[i] = udef$ **then**

begin $out_p[i] := w$;

$a_p := \max\left(a_p, i - \ell_q + 1\right)$;

$s_p := \min\{j \mid out_p[j] = udef\}$

end

end

Условие (2p):

$\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$

остается истинным в связи с тем, что

1) $\mathbf{R}_p$ при получении пакета $\langle \mathbf{pack}, w, i\rangle$ из очереди Q_p записывает w в элемент $out_p[i]$ выходного массива, а из условия (1p) следует, что $w = in_q[i]$;

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{ \langle \mathbf{pack}, w, i \rangle \in Q_p \}$

begin receive $\langle \mathbf{pack}, w, i \rangle$;

if $out_p[i] = udef$ **then**

begin $out_p[i] := w$;

$a_p := \max(a_p, i - \ell_q + 1)$;

$s_p := \min\{j | \, out_p[j] = udef\}$

end

end

Условие (2p):

$\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$

остается истинным в связи с тем, что

1) $\mathbf{R}_p$ при получении пакета $\langle \mathbf{pack}, w, i \rangle$ из очереди Q_p записывает w в элемент $out_p[i]$ выходного массива, а из условия (1p) следует, что $w = in_q[i]$;

2) присваивание $a_p := \max(a_p,\ i - \ell_q + 1)$ гарантирует, что неравенство $a_p > i - \ell_q$ будет верно и после выполнения $\mathbf{R}_p$.

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{ \langle \mathbf{pack}, w, i \rangle \in Q_p \}$

begin receive $\langle \mathbf{pack}, w, i \rangle$;

if $out_p[i] = udef$ **then**

begin $out_p[i] := w$;

$a_p := \max(a_p, i - \ell_q + 1)$;

$s_p := \min\{j | \, out_p[j] = udef\}$

end

end

Условие (2q):

$$\forall i : out_q[i] \neq udef \implies out_q[i] = in_p[i] \wedge (a_q > i - \ell_p)$$

остается истинным в связи с тем, что $\mathbf{R}_p$ не изменяет значений переменной a_q и массива out_q .

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i\rangle$;

if $out_p[i] = udef$ **then**

begin $out_p[i] := w$;

$a_p := \max\left(a_p, i - \ell_q + 1\right)$;

$s_p := \min\left\{j \mid out_p[j] = udef\right\}$

end

end

Условие (3р): $a_p \le s_q$

остается истинным в связи с тем, что

1) $\mathbf{R}_p$ не изменяет значения переменной s_q ;

Доказательство Теоремы 3.4.

$\mathbf{R}_p$: $\{\ \langle \mathbf{pack}, w, i\rangle \in Q_p\ \}$

begin receive $\langle \mathbf{pack}, w, i\rangle$;

if $out_p[i] = udef$ **then**

begin $out_p[i] := w$;

$a_p := \max\left(a_p, i - \ell_q + 1\right)$;

$s_p := \min\left\{j \middle|\, out_p[j] = udef\right\}$

end

end

Условие (3р): $a_p \leq s_q$

остается истинным в связи с тем, что

1) $\mathbf{R}_p$ не изменяет значения переменной s_q ;

2) когда при получении пакета $\langle \mathbf{pack}, w, i\rangle$ выполняется присваивание $a_p := \max(a_p, i - \ell_q + 1)$, из условия (1р) следует неравенство $i < s_q + \ell_q$, и поэтому неравенство $a_p \leq s_q$ остается верным.

Доказательство Теоремы 3.4.

R$_p$: { $\langle \mathbf{pack}, w, i\rangle \in Q_p$ }
  begin receive $\langle \mathbf{pack}, w, i\rangle$;
    if $out_p[i] = udef$ **then**
      begin $out_p[i] := w$;
        $a_p := \max\left(a_p, i - \ell_q + 1\right)$;
        $s_p := \min\{j \mid out_p[j] = udef\}$
      end
  end

Условие (3q): $a_q \leq s_p$
остается истинным в связи с тем, что значение переменной a_q не изменяется, а значением переменной s_p может лишь увеличиться после выполнения действия $\mathbf{R}_p$.

Доказательство Теоремы 3.4.

$\mathbf{S}_p$: $\{\ a_p \leq i < s_p + \ell_p\ \}$ **begin** send $\langle \mathbf{pack}, in_p[i], i \rangle$ to q **end**

$\mathbf{L}_p$: $\{\ \langle \mathbf{pack}, w, i \rangle \in Q_p\ \}$ **begin** $Q_p := Q_p \setminus \{\langle \mathbf{pack}, w, i \rangle\}$ **end**

Задача: доказать, что эти действия процессов также сохраняют инвариант P .

$\square$

Симметричный протокол раздвижного окна

Теорема 3.5.

Симметричный протокол раздвижного окна удовлетворяет требованию безопасной доставки сообщений, т.е. в каждой достижимой конфигурации протокола выполняются соотношения

$out_p[0..s_p - 1] = in_q[0..s_p - 1]$ и $out_q[0..s_q - 1] = in_p[0..s_q - 1]$.

Доказательство:

Следует из Теоремы 3.2 (о свойстве инвариантов) и Теоремы 3.4.

Симметричный протокол раздвижного окна

Теорема 3.5.

Симметричный протокол раздвижного окна удовлетворяет требованию безопасной доставки сообщений, т.е. в каждой достижимой конфигурации протокола выполняются соотношения

$out_p[0..s_p - 1] = in_q[0..s_p - 1]$ и $out_q[0..s_q - 1] = in_p[0..s_q - 1]$.

Доказательство:

Следует из Теоремы 3.2 (о свойстве инвариантов) и Теоремы 3.4.

Из условий

(0p): $\forall i < s_p : \ out_p[i] \neq udef$

(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$

следует выполнимость равенства $out_p[0..s_p - 1] = in_q[0..s_p - 1]$,

Симметричный протокол раздвижного окна

Теорема 3.5.

Симметричный протокол раздвижного окна удовлетворяет требованию безопасной доставки сообщений, т.е. в каждой достижимой конфигурации протокола выполняются соотношения

$out_p[0..s_p - 1] = in_q[0..s_p - 1]$ и $out_q[0..s_q - 1] = in_p[0..s_q - 1]$.

Доказательство:

Следует из Теоремы 3.2 (о свойстве инвариантов) и Теоремы 3.4.

Из условий

(0p): $\forall i < s_p : \; out_p[i] \neq udef$

(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$

следует выполнимость равенства $out_p[0..s_p - 1] = in_q[0..s_p - 1]$,

а из условий (0q) и (2q) следует выполнимость равенства $out_q[0..s_q - 1] = in_p[0..s_q - 1]$.

Симметричный протокол раздвижного окна

Чтобы убедиться в том, что доставка сообщений неизбежна, необходимо

Симметричный протокол раздвижного окна

Чтобы убедиться в том, что доставка сообщений неизбежна, необходимо

- ввести допущения справедливости,
- а также ввести ограничения на значения ℓ_p и ℓ_q .

Симметричный протокол раздвижного окна

Чтобы убедиться в том, что доставка сообщений неизбежна, необходимо

- ввести допущения справедливости,
- а также ввести ограничения на значения ℓ_p и ℓ_q.

Без этих ограничений протокол не будет обладать свойством живости.
(Почему?)

Симметричный протокол раздвижного окна

Ограничения

- В качестве ℓ_p и ℓ_q можно взять любые неотрицательные константы, удовлетворяющие неравенству $\ell_p + \ell_q > 0$.

Симметричный протокол раздвижного окна

Ограничения

- В качестве ℓ_p и ℓ_q можно взять любые неотрицательные константы, удовлетворяющие неравенству $\ell_p + \ell_q > 0$.
- Выдвигаются два требования справедливости:

  F1. Если бесконечно часто возникает возможность отправки пакета, то этот пакет будет отправляться бесконечно часто.

Симметричный протокол раздвижного окна

Ограничения

- В качестве ℓ_p и ℓ_q можно взять любые неотрицательные константы, удовлетворяющие неравенству $\ell_p + \ell_q > 0$.
- Выдвигаются два требования справедливости:

F1. Если бесконечно часто возникает возможность отправки пакета, то этот пакет будет отправляться бесконечно часто.

F2. Если один и тот же пакет отправляется бесконечно часто, то и принимается он также бесконечно часто.

Симметричный протокол раздвижного окна

Створки окон процессов p и q «разъезжаются» не слишком далеко друг от друга.

Симметричный протокол раздвижного окна

Створки окон процессов p и q «разъезжаются» не слишком далеко друг от друга.

Лемма 3.1.
В любой достижимой конфигурации выполняются неравенства

$$s_p - \ell_q \leq a_p \leq s_q \leq a_q + \ell_p \leq s_p + \ell_p.$$

Симметричный протокол раздвижного окна

Доказательство:

Из условий
(0p): $\forall i < s_p : \ out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p$$

Из условий
(0p): $\forall i < s_p : \; out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p$$

Из условий
(0p): $\forall i < s_p : \ out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.
Свойство (3p): $a_p \leq s_q$
обеспечивает выполнимость неравенства $a_p \leq s_q$.

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p \quad \leq s_q$$

Из условий
(0p): $\forall i < s_p : \; out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.
Свойство (3p): $a_p \leq s_q$
обеспечивает выполнимость неравенства $a_p \leq s_q$.

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p \leq s_q$$

Из условий
(0p): $\forall i < s_p : \ out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.
Свойство (3p): $a_p \leq s_q$
обеспечивает выполнимость неравенства $a_p \leq s_q$.
Из (0q) и (2q) следует $s_q \leq a_q + \ell_p$.

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p \ \leq s_q \ \leq a_q + \ell_p$$

Из условий
(0p): $\forall i < s_p : \ out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.
Свойство (3p): $a_p \leq s_q$
обеспечивает выполнимость неравенства $a_p \leq s_q$.
Из (0q) и (2q) следует $s_q \leq a_q + \ell_p$.

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p \;\; \leq s_q \;\; \leq a_q + \ell_p$$

Из условий
(0p): $\forall i < s_p : \; out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.
Свойство (3p): $a_p \leq s_q$
обеспечивает выполнимость неравенства $a_p \leq s_q$.
Из (0q) и (2q) следует $s_q \leq a_q + \ell_p$.
И, наконец, из (3q) следует неравенство $a_q + \ell_p \leq s_p + \ell_p$.

□

Симметричный протокол раздвижного окна

Доказательство:

$$s_p - \ell_q \leq a_p \ \leq s_q \ \leq a_q + \ell_p \ \leq s_p + \ell_p.$$

Из условий
(0p): $\forall i < s_p : \ out_p[i] \neq udef$
(2p): $\forall i : out_p[i] \neq udef \implies out_p[i] = in_q[i] \wedge (a_p > i - \ell_q)$
инварианта P следует неравенство $s_p - \ell_q \leq a_p$.
Свойство (3p): $a_p \leq s_q$
обеспечивает выполнимость неравенства $a_p \leq s_q$.
Из (0q) и (2q) следует $s_q \leq a_q + \ell_p$.
И, наконец, из (3q) следует неравенство $a_q + \ell_p \leq s_p + \ell_p$.

$\square$

Симметричный протокол раздвижного окна

Створки окон процессов p и q не могут «сомкнуться» одновременно, т.е. протокол никогда не будет заблокирован.

Симметричный протокол раздвижного окна

Створки окон процессов p и q не могут «сомкнуться» одновременно, т.е. протокол никогда не будет заблокирован.

Лемма 3.2.
В любой достижимой конфигурации допустимо хотя бы одно из двух действий: отправление пакета $\langle \mathbf{pack}, in_p[s_q], s_q \rangle$ процессом p или отправление пакета $\langle \mathbf{pack}, in_q[s_p], s_p \rangle$ процессом q .

Симметричный протокол раздвижного окна

Доказательство:

Согласно ограничению $\ell_p + \ell_q > 0$ по крайней мере одно из неравенств Леммы 3.1

$$s_p - \ell_q \leq a_p \leq s_q \leq a_q + \ell_p \leq s_p + \ell_p$$

является строгим, т.е.,

$s_q < s_p + \ell_p \quad \vee \quad s_p < s_q + \ell_q$.

Симметричный протокол раздвижного окна

Доказательство:

Согласно ограничению $\ell_p + \ell_q > 0$ по крайней мере одно из неравенств Леммы 3.1

$$s_p - \ell_q \leq a_p \leq s_q \leq a_q + \ell_p \leq s_p + \ell_p$$

является строгим, т.е.,

$s_q < s_p + \ell_p \quad \vee \quad s_p < s_q + \ell_q$.

Из инварианта P также следуют неравенства

(3p): $a_p \leq s_q$ и (3q): $a_q \leq s_p$,

и поэтому справедливо соотношение

$(a_p \leq s_q < s_p + \ell_p) \quad \vee \quad (a_q \leq s_p < s_q + \ell_q)$,

Симметричный протокол раздвижного окна

Доказательство:

Согласно ограничению $\ell_p + \ell_q > 0$ по крайней мере одно из неравенств Леммы 3.1

$$s_p - \ell_q \leq a_p \leq s_q \leq a_q + \ell_p \leq s_p + \ell_p$$

является строгим, т.е.,

$s_q < s_p + \ell_p \quad \vee \quad s_p < s_q + \ell_q$.

Из инварианта P также следуют неравенства

(3p): $a_p \leq s_q$ и (3q): $a_q \leq s_p$,

и поэтому справедливо соотношение

$(a_p \leq s_q < s_p + \ell_p) \quad \vee \quad (a_q \leq s_p < s_q + \ell_q)$,

Это соотношение означает, что либо действие $\mathbf{S}_p$ допустимо для $i = s_q$, либо действие $\mathbf{S}_q$ допустимо для $i = s_p$.

□

Симметричный протокол раздвижного окна

Теорема 3.6.

Симметричный протокол раздвижного окна удовлетворяет требованию неизбежной доставки сообщений, т.е. для каждого целого числа $k \geq 0$, в ходе любого выполнения протокола будет достигнуты конфигурация, в которой $s_p \geq k$ и $s_q \geq k$.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1.
$s_p - \ell_q \leq s_q \leq s_p + \ell_p$,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.
Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q.
Тогда, согласно Лемме 3.2.,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда, согласно допущению F1,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда, согласно допущению F1, один из этих пакетов (например, $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$) отправляется бесконечно часто, и,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда, согласно допущению F1, один из этих пакетов (например, $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$) отправляется бесконечно часто, и, согласно допущению F2,

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда, согласно допущению F1, один из этих пакетов (например, $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$) отправляется бесконечно часто, и, согласно допущению F2, он должен приниматься также бесконечно часто.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \textbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \textbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда, согласно допущению F1, один из этих пакетов (например, $\langle \textbf{pack}, in_q[\sigma_p], \sigma_p \rangle$) отправляется бесконечно часто, и, согласно допущению F2, он должен приниматься также бесконечно часто.

Получение процессом p пакета $\langle \textbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ приводит к тому, что значение s_p (равное σ_p) увеличивается.

Доказательство:

Предположим, что есть вычисление C, в котором значения *хотя бы одной* из переменных s_p и s_q увеличиваются лишь конечное число раз. Тогда, согласно неравенству Леммы 3.1. $s_p - \ell_q \leq s_q \leq s_p + \ell_p$, значение другой переменной также не может увеличиваться бесконечно часто.

Пусть σ_p и σ_q — наибольшие значения переменных s_p и s_q. Тогда, согласно Лемме 3.2., либо отправление пакета $\langle \mathbf{pack}, in_p[\sigma_q], \sigma_q \rangle$ процессом p, либо отправление пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ процессом q допустимо бесконечно долго после того, как переменные s_p, s_q, a_p и a_q примут свои окончательные значения.

Тогда, согласно допущению F1, один из этих пакетов (например, $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$) отправляется бесконечно часто, и, согласно допущению F2, он должен приниматься также бесконечно часто.

Получение процессом p пакета $\langle \mathbf{pack}, in_q[\sigma_p], \sigma_p \rangle$ приводит к тому, что значение s_p (равное σ_p) увеличивается.

Это противоречит выбору значения σ_p.

Симметричный протокол раздвижного окна

Задачи.

1. Покажите, что симметричный протокол раздвижного окна не удовлетворяет требованию неизбежной доставки сообщения, если из двух допущений справедливости F1 и F2 выполняется только допущение F2.

Симметричный протокол раздвижного окна

Задачи.

1. Покажите, что симметричный протокол раздвижного окна не удовлетворяет требованию неизбежной доставки сообщения, если из двух допущений справедливости F1 и F2 выполняется только допущение F2. А что произойдет, если будет выполняться только допущение F1?

Симметричный протокол раздвижного окна

Задачи.

1. Покажите, что симметричный протокол раздвижного окна не удовлетворяет требованию неизбежной доставки сообщения, если из двух допущений справедливости F1 и F2 выполняется только допущение F2. А что произойдет, если будет выполняться только допущение F1?
2. В каком месте обоснования корректности протокола используется свойство очередности передачи сообщений по каналам связи?

Симметричный протокол раздвижного окна

Задачи.

1. Покажите, что симметричный протокол раздвижного окна не удовлетворяет требованию неизбежной доставки сообщения, если из двух допущений справедливости F1 и F2 выполняется только допущение F2. А что произойдет, если будет выполняться только допущение F1?
2. В каком месте обоснования корректности протокола используется свойство очередности передачи сообщений по каналам связи?
3. Докажите, что если в симметричном протоколе раздвижного окна $\ell_p + \ell_q = 1$ и начальные значения переменных a_p и a_q равны $-\ell_q$ и $-\ell_p$, то равенства $a_p + \ell_q = s_p$ и $a_q + \ell_p = s_q$ всегда выполняются.

Реализации протокола раздвижного окна

Алгоритм в том виде, в котором он был представлен, непригоден для реализации, так как в каждом процессе хранится бесконечный объем информации (массивы *in* и *out*) и используются неограниченные порядковые номера.

Реализации протокола раздвижного окна

Алгоритм в том виде, в котором он был представлен, непригоден для реализации, так как в каждом процессе хранится бесконечный объем информации (массивы *in* и *out*) и используются неограниченные порядковые номера.

Пусть $L = \ell_p + \ell_q$.

Реализации протокола раздвижного окна

Алгоритм в том виде, в котором он был представлен, непригоден для реализации, так как в каждом процессе хранится бесконечный объем информации (массивы *in* и *out*) и используются неограниченные порядковые номера.

Пусть $L = \ell_p + \ell_q$.

Лемма 3.3.

Отправление пакета $\langle \mathbf{pack}, w, i \rangle$ процессом p допустимо только тогда, когда $i < a_p + L$.

Доказательство:

Задача. Решить самостоятельно.

Реализации протокола раздвижного окна

Алгоритм в том виде, в котором он был представлен, непригоден для реализации, так как в каждом процессе хранится бесконечный объем информации (массивы *in* и *out*) и используются неограниченные порядковые номера.

Пусть $L = \ell_p + \ell_q$.

Лемма 3.3.
Отправление пакета $\langle \mathbf{pack}, w, i \rangle$ процессом p допустимо только тогда, когда $i < a_p + L$.

Доказательство:

Задача. Решить самостоятельно.

Лемма 3.4.
Если $out_p[i] \neq udef$, то выполняется неравенство $i < s_p + L$.

Доказательство:

Задача. Решить самостоятельно.

Реализации протокола раздвижного окна

Из этих лемм следует, что

Реализации протокола раздвижного окна

Из этих лемм следует, что

- Процесс p должен хранить в памяти только слова из отрезка $in_p[a_p \dots s_p + \ell_p - 1]$, поскольку он может отправлять только эти слова. Указанный отрезок назовем окном передачи процесса p.

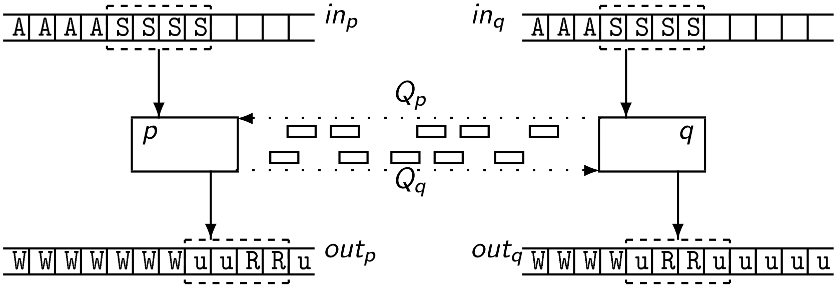
Реализации протокола раздвижного окна

Из этих лемм следует, что

- Процесс p должен хранить в памяти только слова из отрезка $in_p[a_p \dots s_p + \ell_p - 1]$, поскольку он может отправлять только эти слова. Указанный отрезок назовем окном передачи процесса p.
- Процесс p может ожидать приема только тех слов, которые он разметит ячейках из отрезка $out_p[s_p \dots s_p + L - 1]$, поскольку процесс q может отправлять только эти слова. Указанный отрезок назовем окном приема процесса p.

Реализации протокола раздвижного окна

Протокол с окнами приема и передачи.



A = Сброшенный вход

S = Окно передачи

W = Записанные слова

R = Принятые слова

u = Неопределенные значения

 = Окна передачи/приема

Реализации протокола раздвижного окна

В том случае, если каналы поддерживают очередность передачи сообщений, порядковые номера слов также можно ограничить.

Теорема 3.7.

Утверждение P', определяемое следующее формулой

$$
\begin{array}{rll}
P' \equiv & P & \\
\wedge & \langle \mathbf{pack}, w, i\rangle \text{ следует за } \langle \mathbf{pack}, w', i'\rangle \text{ в } Q_p \Rightarrow i > i' - L & (4p) \\
\wedge & \langle \mathbf{pack}, w, i\rangle \text{ следует за } \langle \mathbf{pack}, w', i'\rangle \text{ в } Q_q \Rightarrow i > i' - L & (4q) \\
\wedge & \langle \mathbf{pack}, w, i\rangle \in Q_p \implies i \geq a_p - \ell_p & (5p) \\
\wedge & \langle \mathbf{pack}, w, i\rangle \in Q_q \implies i \geq a_q - \ell_q & (5q)
\end{array}
$$

является инвариантом протокола раздвижного окна при условии, что в каналах связи поддерживается очередность передавемых сообщений.

Доказательство:

Задача. Решить самостоятельно.

Реализации протокола раздвижного окна

Отсюда следует

Теорема 3.8.

Из утверждения P' вытекают следующие соотношения

$$\langle \mathbf{pack}, w, i \rangle \in Q_p \implies s_p - L \leq i < s_p + L$$

и

$$\langle \mathbf{pack}, w, i \rangle \in Q_q \implies s_q - L \leq i < s_q + L.$$

Реализации протокола раздвижного окна

Отсюда следует

Теорема 3.8.

Из утверждения P' вытекают следующие соотношения

$$\langle \mathbf{pack}, w, i\rangle \in Q_p \implies s_p - L \leq i < s_p + L$$

и

$$\langle \mathbf{pack}, w, i\rangle \in Q_q \implies s_q - L \leq i < s_q + L.$$

Доказательство:

Рассмотрим пакет $\langle \mathbf{pack}, w, i\rangle \in Q_p$.

Реализации протокола раздвижного окна

Отсюда следует

Теорема 3.8.

Из утверждения P' вытекают следующие соотношения

$$\langle \mathbf{pack}, w, i\rangle \in Q_p \implies s_p - L \leq i < s_p + L$$

и

$$\langle \mathbf{pack}, w, i\rangle \in Q_q \implies s_q - L \leq i < s_q + L.$$

Доказательство:

Рассмотрим пакет $\langle \mathbf{pack}, w, i\rangle \in Q_p$.
Согласно свойству (1p) имеем неравенство $i < s_q + \ell_q$, из которого по Лемме 3.1 следует неравенство $i < s_p + L$.

Реализации протокола раздвижного окна

Отсюда следует

Теорема 3.8.

Из утверждения P' вытекают следующие соотношения

$$\langle \mathbf{pack}, w, i \rangle \in Q_p \implies s_p - L \leq i < s_p + L$$

и

$$\langle \mathbf{pack}, w, i \rangle \in Q_q \implies s_q - L \leq i < s_q + L.$$

Доказательство:

Рассмотрим пакет $\langle \mathbf{pack}, w, i \rangle \in Q_p$.
Согласно свойству (1p) имеем неравенство $i < s_q + \ell_q$, из которого по Лемме 3.1 следует неравенство $i < s_p + L$.
Согласно свойству (5p) имеем неравенство $i \geq a_p - \ell_p$, из которого по Лемме 3.1 следует неравенство $i \geq s_p - L$.

Реализации протокола раздвижного окна

Отсюда следует

Теорема 3.8.

Из утверждения P' вытекают следующие соотношения

$$\langle \mathbf{pack}, w, i \rangle \in Q_p \implies s_p - L \leq i < s_p + L$$

и

$$\langle \mathbf{pack}, w, i \rangle \in Q_q \implies s_q - L \leq i < s_q + L.$$

Доказательство:

Рассмотрим пакет $\langle \mathbf{pack}, w, i \rangle \in Q_p$.
Согласно свойству (1p) имеем неравенство $i < s_q + \ell_q$, из которого по Лемме 3.1 следует неравенство $i < s_p + L$.
Согласно свойству (5p) имеем неравенство $i \geq a_p - \ell_p$, из которого по Лемме 3.1 следует неравенство $i \geq s_p - L$.
Аналогичным образом доказывется соотношение о пакетах из очереди Q_q .

Реализации протокола раздвижного окна

Как видно из Теоремы 3.8,

Реализации протокола раздвижного окна

Как видно из Теоремы 3.8, достаточно, чтобы нумерация пакетов проводилась по модулю k, где $k \geq 2L$.

$$s_p - L \leq i < s_p + L$$

Действительно, располагая значением переменной s_p, а также значением переменной i по модулю k, процесс p может вычислить номер i.

Реализации протокола раздвижного окна

Особо интересный вариант протокола раздвижного окна возникает в том случае, когда $L = 1$, т.е. когда $\ell_p = 1$ и $\ell_q = 0$ (или наоборот). В качестве начальных значением переменных a_p и a_q в этом случае выбирается не 0, а числа $-\ell_p$ и $-\ell_q$.

Такой вариант алгоритма раздвижного окна называется протоколом чередующихся (альтернирующих) битов; он пренднaзначен для односторонней передачи данных.

Реализации протокола раздвижного окна

Особо интересный вариант протокола раздвижного окна возникает в том случае, когда $L = 1$, т.е. когда $\ell_p = 1$ и $\ell_q = 0$ (или наоборот). В качестве начальных значением переменных a_p и a_q в этом случае выбирается не 0, а числа $-\ell_p$ и $-\ell_q$.

Такой вариант алгоритма раздвижного окна называется протоколом чередующихся (альтернирующих) битов; он пренднaзначен для односторонней передачи данных.

Задача.

Покажите, что в этом случае в протоколе достаточно использовать только одну из двух переменных a_p или s_p (и только одну из переменных a_q или s_q).

Реализации протокола раздвижного окна

Задача.
Насколько существенным для свойств корректности и неизбежности доставки сообщений в алгоритме раздвижного окна являются следующие требования:

1. Допущение справедливости F1,
2. Допущение справедливости F2,
3. Очередность следования пакетов в канале связи,
4. Невозможность дублированяи пакетов в канале связи,
5. Надежность работы процессов алгоритма?

КОНЕЦ ЛЕКЦИИ 3.