

Распределенные алгоритмы и системы

mk.cs.msu.ru → Лекционные курсы → Распределенные алгоритмы и системы

Блок 9

Симметричный протокол раздвижного окна

Лектор:

Подымов Владислав Васильевич

E-mail:

valdus@yandex.ru

Коммуникационные протоколы

Основное назначение **коммуникационного протокола** (или, по-другому, — **протокола передачи данных**) — получение информации от одного узла сети и доставка её другому узлу сети

При передаче данных возможны **коммуникационные неисправности**, которые следует обнаруживать и исправлять

Для отслеживания и обработки ошибок используется **управление соединением**:

- ▶ **Установление соединения** с инициализацией необходимой служебной информации
- ▶ Поддержка служебной информации для корректной передачи данных
- ▶ **Завершение соединения** с удалением информации о соединении

Симметричный протокол раздвижного окна

Англ. Balanced Sliding Window Protocol

Будем для краткости называть этот протокол **BSWP**

BSWP предназначен для передачи данных между двумя узлами по физическому соединению и относится ко второму уровню модели OSI (канальному)

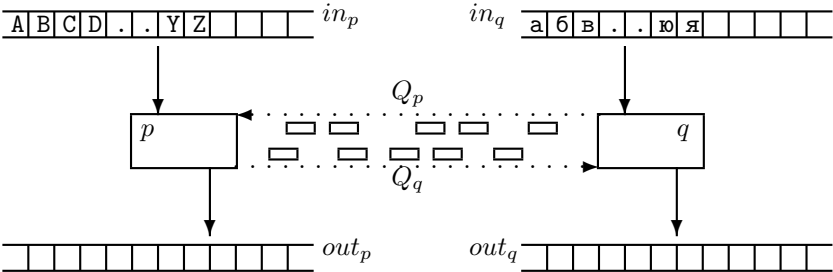
В протоколе используется асинхронный обмен сообщениями

В описании и анализе протокола не рассматривается управление соединением: полагаем, что соединение непрерывно установлено на время всего выполнения протокола

Будем считать, что канал представляет собой **очередь**, допускающую **потерю сообщений**, а в остальном надёжную

Чуть более точно, в канале допускаются искажения сообщений, однако считается, что адресат способен обнаруживать ошибки этого вида (например, с использованием счётчиков чётности или специальных кодов, исправляющих ошибки)

BSWP: постановка задачи



Узлам p и q требуется передать друг другу потоки данных in_p , in_q , записав их в массивы out_q и out_p соответственно

BSWP: идея алгоритма

Поток данных разбивается на **блоки** данных одинакового размера

Каждый блок w пересылается в виде **пакета** (**pack**, $\langle w, i \rangle$) типа **pack**, содержащего этот блок и его **порядковый номер**, $i \in \mathbb{N}_0$

Для синхронизации доставки сообщений в коммуникационных протоколах нередко используются служебные подтверждающие сообщения

В BSWP такие сообщения в явном виде не пересылаются, и их роль играют пересылаемые пакеты

Обоим узлам p , q известны особые заранее заданные параметры протокола (**константы опережения**): ℓ_p , ℓ_q

BSWP: идея алгоритма

Пакет (**pack**, $\langle w, i \rangle$), отправленный узлом A узлу B , обозначает

- ▶ отправленный блок данных $w = in_A[i]$ и
- ▶ подтверждение получения узлом A пакетов от B с номерами $0, 1, \dots, (i - \ell_A)$

Таким двойным назначением пакетов определяются текущие границы «окна» номеров блоков, пересылаемых узлами:

- ▶ Если узел A получил пакет (**pack**, $\langle w, i \rangle$), то он уверяется в том, что все блоки с номерами $0, 1, \dots, (i - \ell_B)$ успешно доставлены, и далее отправляет блоки только с номерами $k: k > i - \ell_B$
- ▶ Если j — это наименьший номер блока, ещё не полученного от B , то A может отправлять только блоки с номерами $k: k < j + \ell_A$

BSWP: устройство узла

Симметричность протокола означает, что узлы p и q идентичны, единственное различие определяется константами ℓ_p и ℓ_q

Поэтому достаточно описать только устройство узла p

В узле p содержится три процедуры (семейства действий):

- ▶ S_p : отправка очередного пакета узлу q
- ▶ R_p : приём пакета от узла q
- ▶ L_p : действие, обозначающее потерю пакета, предназначенного для доставки в узел p

Переменные узла:

var $s_p : \mathbb{N}_0 = 0$;

var $a_p : \mathbb{N}_0 = 0$;

var $in_p : \text{array of word} =$ все данные для отправки;

var $out_p : \text{array of word} = (\perp, \perp, \dots)$;

word — тип пересылаемых блоков данных, дополненный особым значением \perp , обозначающим, что значение отсутствует

array of T — массив элементов типа T

BSWP: устройство узла

Процедура S_p :

1. Произвольно **справедливо** выбрать $i \in \mathbb{N}_0$: $a_p \leq i < s_p + \ell_p$
 2. $send(\mathbf{pack}, \langle in_p[i], i \rangle)$
-

Номера блоков для отправки выбираются из **окна**

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

Створки (границы) этого окна способны изменяться (двигаться вперёд) по ходу получения пакетов от q

BSWP: устройство узла

Q_p — так обозначим (*надёжную*) очередь сообщений в коммуникационной подсистеме, адресованных узлу p

Процедура R_p :

Предусловие: очередь Q_p непуста

1. $receive(\mathbf{pack}, \langle w, i \rangle)$
2. Если $out_p[i] = \perp$:
 - 2.1 $out_p[i] := w$;
 - 2.2 $a_p := \max(a_p, i - \ell_q + 1)$;
 - 2.3 $s_p := \min(j \mid out_p[j] = \perp)$;

Получив пакет из очереди, узел проверяет, не получал ли он уже блок с этим номером раньше

Если не получал, то этот блок записывается в out_p и створки окна обновляются согласно

- ▶ пакету как подтверждению (**левая створка** a_p) и
- ▶ новому неполученному блоку с наименьшим номером (**правая створка** $s_p + \ell$)

BSWP: устройство узла

Процедура L_p :

Предусловие: очередь Q_p не пуста

1. *receive*(**pack**, $\langle w, i \rangle$)
-

Потеря пакета в канале равносильна его чтению из канала без обновления состояния узла

BSWP: спектр возможных проблем выполнения

1. Створки окон обоих процессов могут «захлопнуться» ($a_p \geq s_p + \ell_p$), что приведёт к **блокировке** (deadlock)
2. Створки могут вечно оставаться неизменными, и процессы будут вечно передавать блоки одного ограниченного диапазона — **активный тупик** (livelock)
3. Створки могут неограниченно отдаляться друг от друга, приводя к неэффективности передачи
4. Узел может не отправить блок данных с некоторым номером из-за расширения створок или по другим причинам
5. Узел может принять данные, не уверив партнёра в том, что они приняты
6. Узел может быть уверен, что данные успешно доставлены, хотя это не так

И как же сформулировать и доказать правильность работы протокола?