# Распределённые алгоритмы

mk.cs.msu.ru → Лекционные курсы → Распределённые алгоритмы

Блок 8

Симметричный протокол раздвижного окна

Лектор:
**Подымов Владислав Васильевич**
E-mail:
**valdus@yandex.ru**

ВМК МГУ, 2023/2024, весенний семестр

# Симметричный протокол раздвижного окна

Англ. Balanced Sliding Window Protocol

Будем для краткости называть этот протокол BSWP

BSWP — это коммуникационный протокол (протокол передачи данных), т.е. протокол, предназначенный для передачи информации от одного узла р.с. другому

BSWP относится ко второму уровню модели OSI (канальному) и задаёт передачу данных между двумя узлами по физическому соединению

Передача производится одновременно в две стороны

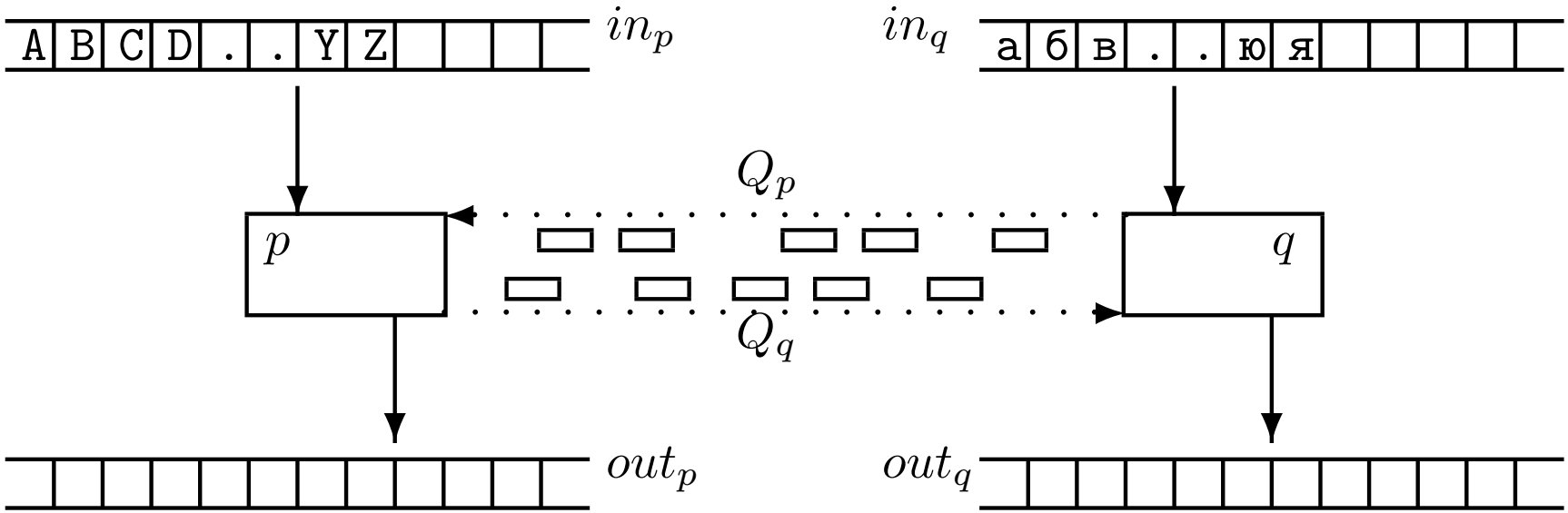
В протоколе используется асинхронный обмен сообщениями

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

Точнее, «в реальности» сообщения в канале также могут искажаться, но считается, что адресат способен обнаруживать ошибки этого вида

# 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: основная идея

Поток данных разбивается на блоки данных одинакового размера

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

Для синхронизации доставки сообщений в протоколах передачи данных 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): $\mathfrak{c}_p$, $\mathfrak{c}_q$

# BSWP: основная идея

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

- отправленный блок данных $w = in_A[i]$ и
- подтверждение получения узлом $A$ пакетов от $B$ с номерами $0, 1, \ldots, (i - \mathfrak{c}_A)$

Таким двойным назначением пакетов определяются текущие границы «окна» номеров блоков, пересылаемых узлами: если $A$ получил пакет (**pack**, $w$, $i$), то он

- использует номер $i$ для обновления наибольшего такого полученного номера $i_{\max}$ и наименьшего такого **не**полученного номера $j_{\min}$
- уверяется в том, что все его пакеты с номерами $0, 1, \ldots, (i_{\max} - \mathfrak{c}_B)$ успешно доставлены
- исходя из трактовки $i$ и «симметричности» и необходимости гарантированно передать данные, отправляет $B$ только блоки с номерами из интервала $(i_{\max} - \mathfrak{c}_B, j_{\min} + \mathfrak{c}_A)$ — окна передачи узла $A$

# BSWP: устройство узла

*Симметричность* протокола означает, что узлы $p$ и $q$ идентичны: единственное их различие определяется константами $\mathfrak{c}_p$ и $\mathfrak{c}_q$

Поэтому достаточно описать только устройство узла $p$

# BSWP: устройство узла

## Переменные узла:

- $\ell_p$ типа $\mathbb{N}_0$ с начальным значением 0
  - Содержательно, это левая створка (граница) текущего окна, наименьший возможный номер в отправляемых пакетах
- $r_p$ типа $\mathbb{N}_0$ с начальным значением 0
  - Содержательно, это наименьший номер ещё не полученного пакета ($j_{\mathsf{min}}$ из пояснений)
  - То есть $(r_p + \mathfrak{c}_p - 1)$ — это правая створка окна, наибольший номер пересылаемых пакетов
- $in_p$ типа $ARR[\mathcal{T}]$
  - Это поток данных для отправки в $q$, представленный здесь в виде неограниченного массива
  - Точный выбор типа $\mathcal{T}$ неважен, и все значения предполагаются заданными в начале выполнения и неизменными
- $out_p$ типа $ARR[\mathcal{T}^\perp]$ с начальным значением $(\perp, \perp, \perp, \dots)$
  - Этим массивом обозначаются данные, принятые от $q$

# BSWP: устройство узла

В узле $p$ содержится три действия:

- $\mathbf{S}_p$: отправка пакета узлом $p$
- $\mathbf{R}_p$: приём и обработка пакета узлом $p$
- $\mathbf{L}_p$: потеря пакета, отправленного узлу $p$

Каждое из этих действий будет представлено двумя частями:

1. Псевдокод $\mathcal{C}$ (предполагается, что весь этот код выполняется на одном переходе)
2. Предусловие $\mathcal{P}$ — утверждение, истинность которого обозначает возможность действия выполниться

Переход $\gamma \to \gamma'$ входит в действие $\Leftrightarrow$

- $\gamma'$ получается из $\gamma$ согласно смыслу кода и
- утверждение $\mathcal{P}$ истинно в $\gamma$

Предусловие будем записывать в фигурных скобках перед кодом, к которому оно относится

# BSWP: устройство узла

---

**Действие $\mathbf{S}_p$:**

$\{\ell_p < r_p + \mathfrak{c}_p\}$

1. Выбрать $i \in \mathbb{N}_0$: $\ell_p \le i < r_p + \mathfrak{c}_p$
2. $send_q(\mathbf{pack}, in_p[i], i)$

---

Отправляется блок с произвольным номером из окна $[\ell_p, r_p + \mathfrak{c}_p)$

Предусловием обеспечивается возможность выбора такого номера — непустота окна

Створки окна постепенно увеличиваются при выполнении других действий

# BSWP: устройство узла

$Q_X$ — так обозначим (*надёжную*) очередь сообщений в коммуникационной подсистеме, адресованных узлу $X$

---

**Действие $\mathbf{R}_p$:**

{очередь $Q_p$ непуста}

1. $receive_q(\underline{\textbf{pack}}, w, i)$
2. Если $out_p[i] = \perp$:
   - 2.1 $out_p[i] := w$;
   - 2.2 $\ell_p := \max(\ell_p, i - \mathfrak{c}_q + 1)$;
   - 2.3 $r_p := \min(j \mid out_p[j] = \perp)$;

---

Узел принимает пакет из очереди, и если не принимал такой пакет раньше, то обрабатывает его:

- Записывает в $out_p$
- Обновляет створки окна

# BSWP: устройство узла

---

**Действие $\mathsf{L}_p$:**

{очередь $Q_p$ непуста}

1. $receive_q(\underline{\textbf{pack}}, w, i)$

---

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

# BSWP: спектр возможных проблем выполнения

1. Если створки окон обоих процессов «захлопнутся» ($\ell_p \geq r_p + \mathfrak{c}_p$), то произойдёт блокировка (deadlock)
2. Если створки перестанут изменяться, то узлы будут вечно передавать блоки одного ограниченного диапазона — это активный тупик (livelock)
3. Если ширина окна будет неограниченно увеличиваться, то передача станет слишком неэффективной
4. Если створки окна сдвинутся так, что перестанут покрывать непереданный блок данных, то он больше никогда не будет отправлен
   Например, если узел посчитает, что блок данных передан и его можно больше не покрывать окном, хотя это не так
5. Если узел примет блок данных, не оповестив должным образом партнёра об этом, то створки окна могут не обновиться как должны

Попробуем научиться формулировать и обосновывать такие непростые свойства протоколов