# Распределенные алгоритмы

ЛЕКТОР:
Владимир Анатольевич Захаров

# ПРОГРАММА КУРСА

## ЧАСТЬ 1. Математические модели распределенных систем

1. Распределенные системы и их характерные особенности. Архитектура распределенных систем. Стандарт ISO Open System Interaction. Алгоритмические проблемы организации вычислений распределенных систем. Особенности распределенных алгоритмов.

# ПРОГРАММА КУРСА

## ЧАСТЬ 1. Математические модели распределенных систем

1. Распределенные системы и их характерные особенности. Архитектура распределенных систем. Стандарт ISO Open System Interaction. Алгоритмические проблемы организации вычислений распределенных систем. Особенности распределенных алгоритмов.
2. Математическая модель. Системы переходов. Системы с синхронным и асинхронным обменом сообщениями. Свойство справедливости выполнений системы. Зависимые и независимые события. Причинно-следственный порядок событий. Эквивалентность выполнений. Вычисления. Логические часы.

# ПРОГРАММА КУРСА

## ЧАСТЬ 2. Коммуникационные протоколы

3. Коммуникационные протоколы. Ошибки, возникающие при передаче сообщений. Симметричный протокол раздвижного окна: устройство протокола и обоснование его корректности. Протокол альтернирующего бита.

# ПРОГРАММА КУРСА

## ЧАСТЬ 2. Коммуникационные протоколы

3. Коммуникационные протоколы. Ошибки, возникающие при передаче сообщений. Симметричный протокол раздвижного окна: устройство протокола и обоснование его корректности. Протокол альтернирующего бита.
4. Коммуникационные протоколы, использующие таймеры: описание устройства и обоснование корректности.

# ПРОГРАММА КУРСА

## ЧАСТЬ 2. Коммуникационные протоколы

3. Коммуникационные протоколы. Ошибки, возникающие при передаче сообщений. Симметричный протокол раздвижного окна: устройство протокола и обоснование его корректности. Протокол альтернирующего бита.
4. Коммуникационные протоколы, использующие таймеры: описание устройства и обоснование корректности.
5. Задача маршрутизации. Алгоритмы построения кратчайших путей в графе. Алгоритмы маршрутизации.

# ПРОГРАММА КУРСА

## ЧАСТЬ 3. Распределенные алгоритмы

6. Волновые алгоритмы: определение, основные свойства, область применения. Примеры волновых алгоритмов.

# ПРОГРАММА КУРСА

## ЧАСТЬ 3. Распределенные алгоритмы

6. Волновые алгоритмы: определение, основные свойства, область применения. Примеры волновых алгоритмов.
7. Задача избрания лидера. Избрание лидера на кольцах. Избрание лидера в произвольных сетях.

# ПРОГРАММА КУРСА

## ЧАСТЬ 3. Распределенные алгоритмы

6. Волновые алгоритмы: определение, основные свойства, область применения. Примеры волновых алгоритмов.
7. Задача избрания лидера. Избрание лидера на кольцах. Избрание лидера в произвольных сетях.
8. Задача обнаружения завершения вычисления. Алгоритм обнаружения завершения вычисления. Применение алгоритмов обнаружения завершения вычислений для обнаружения блокировки.

# ПРОГРАММА КУРСА

## ЧАСТЬ 3. Распределенные алгоритмы

6. Волновые алгоритмы: определение, основные свойства, область применения. Примеры волновых алгоритмов.
7. Задача избрания лидера. Избрание лидера на кольцах. Избрание лидера в произвольных сетях.
8. Задача обнаружения завершения вычисления. Алгоритм обнаружения завершения вычисления. Применение алгоритмов обнаружения завершения вычислений для обнаружения блокировки.
9. Задача сохранения моментального состояния.

# ПРОГРАММА КУРСА

## ЧАСТЬ 4. Обеспечение отказоустойчивости

10. Задача обеспечения отказоустойчивости распределенных систем. Невозможность построения робастных асинхронных систем. Синхронные робастные алгоритмы принятия решения.

# ПРОГРАММА КУРСА

## ЧАСТЬ 4. Обеспечение отказоустойчивости

10. Задача обеспечения отказоустойчивости распределенных систем. Невозможность построения робастных асинхронных систем. Синхронные робастные алгоритмы принятия решения.
11. Стабилизирующиеся алгоритмы. Общие принципы построения стабилизирующихся алгоритмов.

# Лекция 1.

Распределенные системы и их характерные особенности.
Архитектура распределенных систем.
Алгоритмические проблемы организации вычислений распределенных систем.
Особенности распределенных алгоритмов.

# Распределенные системы

**Распределенной системой** называют всякую вычислительную систему, в которой несколько **автономных** компьютеров или программ вступают во **взаимодействие** друг с другом.

Автономность вычислительных агентов предполагает их собственное независимое управление.

Взаимодействие вычислительных агентов предполагает обмен информации между ними.

# Распределенные системы

**Распределенной системой** называют всякую вычислительную систему, в которой несколько **автономных** компьютеров или программ вступают во **взаимодействие** друг с другом.

Автономность вычислительных агентов предполагает их собственное независимое управление.

Взаимодействие вычислительных агентов предполагает обмен информации между ними.

Взаимодействующие компьютеры, процессы или программы называют **узлами** распределенной системы.

# Распределенные системы

**Распределенной системой** называют всякую вычислительную систему, в которой несколько **автономных** компьютеров или программ вступают во **взаимодействие** друг с другом.

Автономность вычислительных агентов предполагает их собственное независимое управление.

Взаимодействие вычислительных агентов предполагает обмен информации между ними.

Взаимодействующие компьютеры, процессы или программы называют **узлами** распределенной системы.

Механизм организации связи между узлами называют **коммуникационной подсистемой** .

# Распределенные системы

**Распределенной системой** называют всякую вычислительную систему, в которой несколько **автономных** компьютеров или программ вступают во **взаимодействие** друг с другом.

Автономность вычислительных агентов предполагает их собственное независимое управление.

Взаимодействие вычислительных агентов предполагает обмен информации между ними.

Взаимодействующие компьютеры, процессы или программы называют **узлами** распределенной системы.

Механизм организации связи между узлами называют **коммуникационной подсистемой** .

**Распределенные алгоритмы** — это алгоритмы, определяющие функционирование распределенных систем.

# Распределенные системы

## Примеры распределенных систем:

- глобальные коммуникационные вычислительные сети,

# Распределенные системы

## Примеры распределенных систем:

- глобальные коммуникационные вычислительные сети,
- локальные вычислительные сети,

# Распределенные системы

## Примеры распределенных систем:

- глобальные коммуникационные вычислительные сети,
- локальные вычислительные сети,
- многопроцессорные компьютеры,

# Распределенные системы

## Примеры распределенных систем:

- глобальные коммуникационные вычислительные сети,
- локальные вычислительные сети,
- многопроцессорные компьютеры,
- системы взаимодействующих процессов.

# Распределенные системы

## Зачем нужны распределенные системы?

# Распределенные системы

## Зачем нужны распределенные системы?

1. Обмен информацией.

# Распределенные системы

## Зачем нужны распределенные системы?

1. Обмен информацией.
2. Совместное использование ресурсов.

# Распределенные системы

## Зачем нужны распределенные системы?

1. Обмен информацией.
2. Совместное использование ресурсов.
3. Повышение надежности за счет дублирования.

# Распределенные системы

## Зачем нужны распределенные системы?

1. Обмен информацией.
2. Совместное использование ресурсов.
3. Повышение надежности за счет дублирования.
4. Повышение производительности за счет параллельного выполнения.

# Распределенные системы

## Зачем нужны распределенные системы?

1. Обмен информацией.
2. Совместное использование ресурсов.
3. Повышение надежности за счет дублирования.
4. Повышение производительности за счет параллельного выполнения.
5. Упрощение проектирования за счет специализации.

# Вычислительные сети

**Вычислительная сеть** — это совокупность компьютеров, соединенных между собой при помощи механизмов коммуникации, позволяющих компьютерам сети обмениваться информацией. Этот обмен осуществляется путем отправления и приема **сообщений** .

# Вычислительные сети

**Вычислительная сеть** — это совокупность компьютеров, соединенных между собой при помощи механизмов коммуникации, позволяющих компьютерам сети обмениваться информацией. Этот обмен осуществляется путем отправления и приема **сообщений** .

Вычислительные сети могут быть **глобальными** или **локальными** .

# Вычислительные сети

## Глобальные сети

Узлы глобальной сети обычно далеко отстоят друг от друга. Каждый узел такой сети представляет собой отдельную вычислительную установку. Основное назначение глобальных сетей — обеспечить обмен информацией между пользователями в разных узлах сети.

# Вычислительные сети

## Глобальные сети

Узлы глобальной сети обычно далеко отстоят друг от друга. Каждый узел такой сети представляет собой отдельную вычислительную установку. Основное назначение глобальных сетей — обеспечить обмен информацией между пользователями в разных узлах сети.

## Локальные сети

Узлы локальной сети обычно не слишком удалены друг от друга. Каждый узел такой сети — это небольшая специализированная станция. Локальные сетей предназначены, прежде всего, для обмена информацией и обеспечения совместного использования ресурсов.

# Вычислительные сети

## Основные отличия между глобальными и локальными сетями

1. Параметры надежности.

# Вычислительные сети

## Основные отличия между глобальными и локальными сетями

1. Параметры надежности.
2. Время связи.

# Вычислительные сети

## Основные отличия между глобальными и локальными сетями

1. Параметры надежности.
2. Время связи.
3. Однородность аппаратуры и программного обеспечения.

# Вычислительные сети

## Основные отличия между глобальными и локальными сетями

1. Параметры надежности.
2. Время связи.
3. Однородность аппаратуры и программного обеспечения.
4. Взаимное доверие.

# Глобальные сети

Глобальные сети всегда устроены на основе принципа **двухточечного соединения** .

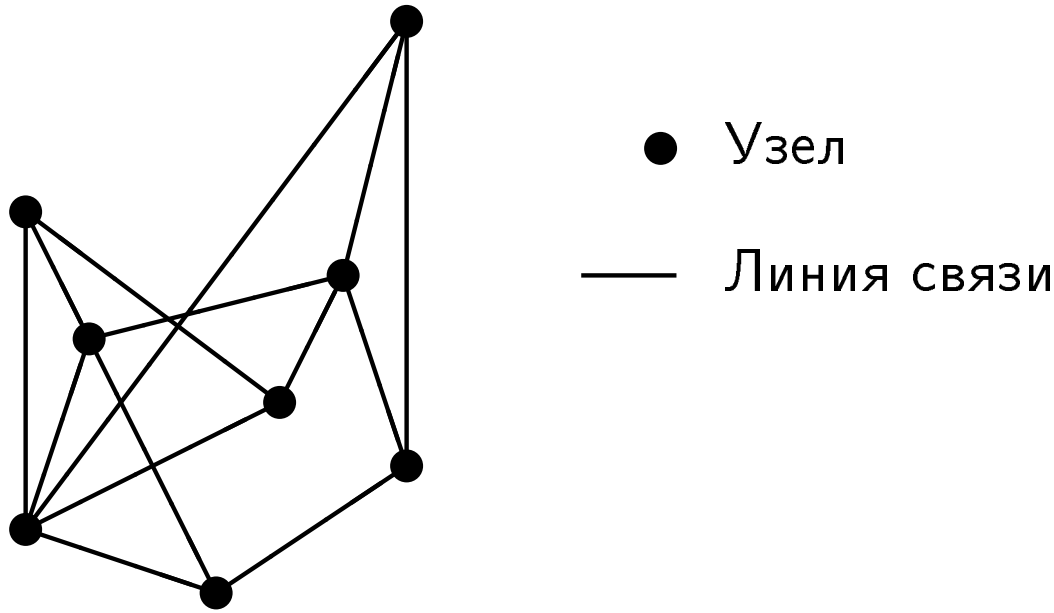



Рис.: Пример сети с двухточечным соединением.

# Глобальные сети

## Алгоритмические проблемы для глобальных сетей

1. Обеспечение надежного обмена данными по двухточечной линии связи.

# Глобальные сети

## Алгоритмические проблемы для глобальных сетей

1. Обеспечение надежного обмена данными по двухточечной линии связи.
2. Выбор каналов связи — задача маршрутизации пакетов.

# Глобальные сети

## Алгоритмические проблемы для глобальных сетей

1. Обеспечение надежного обмена данными по двухточечной линии связи.
2. Выбор каналов связи — задача маршрутизации пакетов.
3. Устранение перегрузки в сети.

# Глобальные сети

## Алгоритмические проблемы для глобальных сетей

1. Обеспечение надежного обмена данными по двухточечной линии связи.
2. Выбор каналов связи — задача маршрутизации пакетов.
3. Устранение перегрузки в сети.
4. Предотвращение блокировки (тупиков) — задача неблокируемой коммутации.

# Глобальные сети

## Алгоритмические проблемы для глобальных сетей

1. Обеспечение надежного обмена данными по двухточечной линии связи.
2. Выбор каналов связи — задача маршрутизации пакетов.
3. Устранение перегрузки в сети.
4. Предотвращение блокировки (тупиков) — задача неблокируемой коммутации.
5. Обеспечение безопасности.

# Глобальные сети

## Алгоритмические проблемы для глобальных сетей

1. Обеспечение надежного обмена данными по двухточечной линии связи.
2. Выбор каналов связи — задача маршрутизации пакетов.
3. Устранение перегрузки в сети.
4. Предотвращение блокировки (тупиков) — задача неблокируемой коммутации.
5. Обеспечение безопасности.

Какие еще алгоритмические проблемы возникают для глобальных сетей?

# Локальные сети

Связь между узлами локальной сети обеспечивает единый механизм — **шина** , — к которому подключены все узлы.

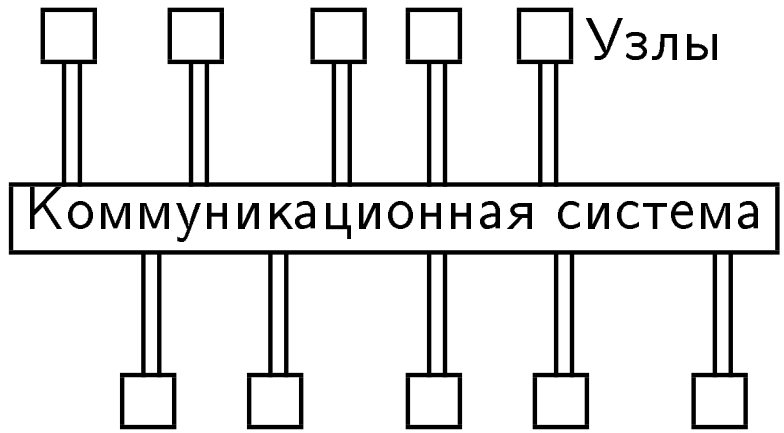



Рис.: Локальная сеть с шинной организацией.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.
3. Обнаружение завершения.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.
3. Обнаружение завершения.
4. Обеспечение корректного использования ресурсов.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.
3. Обнаружение завершения.
4. Обеспечение корректного использования ресурсов.
5. Обнаружение и устранение тупиков.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.
3. Обнаружение завершения.
4. Обеспечение корректного использования ресурсов.
5. Обнаружение и устранение тупиков.
6. Распределенное обслуживание файлов.

# Локальные сети

## Алгоритмические проблемы для локальных сетей

1. Широковещательное распространение и синхронизация.
2. Избрание лидера.
3. Обнаружение завершения.
4. Обеспечение корректного использования ресурсов.
5. Обнаружение и устранение тупиков.
6. Распределенное обслуживание файлов.

Какие еще алгоритмические проблемы возникают для локальных сетей?

# Многопроцессорные компьютеры

**Многопроцессорный компьютер** — это вычислительная машина, которая содержит несколько процессоров, размещенных в корпусе одного устройства.
Если многопроцессорный компьютер предназначен для ускорения вычислений, то его называют **параллельным компьютером** .
А если он предназначен для повышения надежности вычислений, то его называют **вычислительной системой с дублированием** .

# Многопроцессорные компьютеры

**Многопроцессорный компьютер** — это вычислительная машина, которая содержит несколько процессоров, размещенных в корпусе одного устройства.
Если многопроцессорный компьютер предназначен для ускорения вычислений, то его называют **параллельным компьютером** .
А если он предназначен для повышения надежности вычислений, то его называют **вычислительной системой с дублированием** .

В параллельном компьютере всякое вычисление распадается на подвычисления, каждое из которых выполняется в одном из его узлов.

# Многопроцессорные компьютеры

**Многопроцессорный компьютер** — это вычислительная машина, которая содержит несколько процессоров, размещенных в корпусе одного устройства.
Если многопроцессорный компьютер предназначен для ускорения вычислений, то его называют **параллельным компьютером** .
А если он предназначен для повышения надежности вычислений, то его называют **вычислительной системой с дублированием** .

В параллельном компьютере всякое вычисление распадается на подвычисления, каждое из которых выполняется в одном из его узлов.

В системе с дублированием в каждом узле выполняется все вычисление целиком; после этого полученные результаты сравниваются между собой для того, чтобы обнаружить и исправить ошибки.

# Многопроцессорные компьютеры

Коммуникация между процессорами осуществляется либо через шину, либо посредством двухточечной связи.

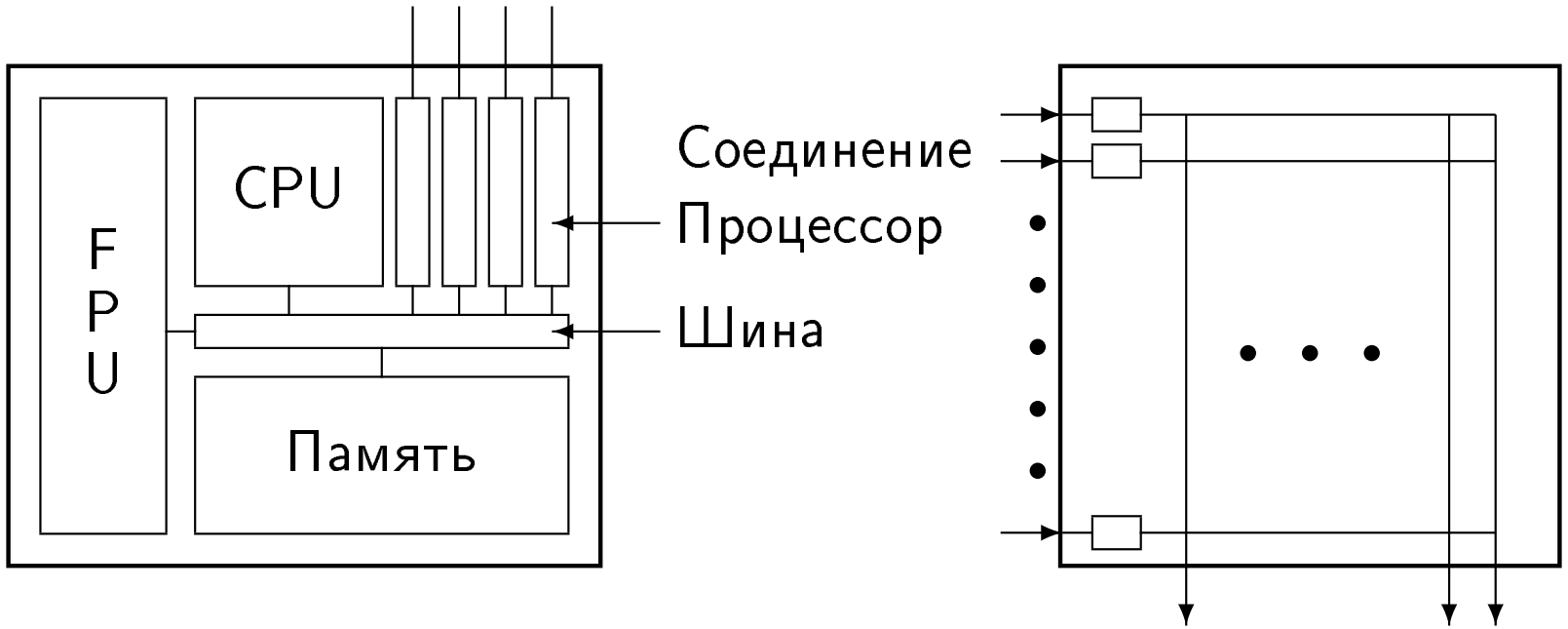



Рис.: Схема Transputer и маршрутизатор.

# Многопроцессорные компьютеры

## Алгоритмические проблемы для многопроцессорных компьютеров

1. Реализация системы передачи сообщений — маршрутизация, устранение перегрузки и блокировок.

# Многопроцессорные компьютеры

## Алгоритмические проблемы для многопроцессорных компьютеров

1. Реализация системы передачи сообщений — маршрутизация, устранение перегрузки и блокировок.
2. Реализация виртуальной разделяемой памяти.

# Многопроцессорные компьютеры

## Алгоритмические проблемы для многопроцессорных компьютеров

1. Реализация системы передачи сообщений — маршрутизация, устранение перегрузки и блокировок.
2. Реализация виртуальной разделяемой памяти.
3. Уравновешивание нагрузки.

# Многопроцессорные компьютеры

## Алгоритмические проблемы для многопроцессорных компьютеров

1. Реализация системы передачи сообщений — маршрутизация, устранение перегрузки и блокировок.
2. Реализация виртуальной разделяемой памяти.
3. Уравновешивание нагрузки.
4. Устойчивость к необнаруживаемым сбоям.

# Многопроцессорные компьютеры

## Алгоритмические проблемы для многопроцессорных компьютеров

1. Реализация системы передачи сообщений — маршрутизация, устранение перегрузки и блокировок.
2. Реализация виртуальной разделяемой памяти.
3. Уравновешивание нагрузки.
4. Устойчивость к необнаруживаемым сбоям.

Какие еще алгоритмические проблемы возникают для многопроцессорных компьютеров?

# Взаимодействующие процессы

Для упрощения проектирования сложного программного обеспечения можно представить программу в виде совокупности процессов, каждый из которых выполняет строго очерченную простую задачу.

# Взаимодействующие процессы

Для упрощения проектирования сложного программного обеспечения можно представить программу в виде совокупности процессов, каждый из которых выполняет строго очерченную простую задачу.

Если в основу проекта положено представление программы в виде семейства взаимодействующих процессов, то полученная в результате программная система будет *логически* распределенной, хотя вполне возможно, что процессы при этом будут выполняться на одном и том же вычислительном устройстве, и в этом случае такая система не будет *физически* распределенной.

# Взаимодействующие процессы

## Алгоритмические проблемы для систем взаимодействующих процессов

1. Атомарность операций записи и считывания.

# Взаимодействующие процессы

## Алгоритмические проблемы для систем взаимодействующих процессов

1. Атомарность операций записи и считывания.
2. Синхронизация процессов.

# Взаимодействующие процессы

## Алгоритмические проблемы для систем взаимодействующих процессов

1. Атомарность операций записи и считывания.
2. Синхронизация процессов.
3. Сборка «мусора».

# Взаимодействующие процессы

## Алгоритмические проблемы для систем взаимодействующих процессов

1. Атомарность операций записи и считывания.
2. Синхронизация процессов.
3. Сборка «мусора».
4. Обмен сообщениями.

# Взаимодействующие процессы

## Алгоритмические проблемы для систем взаимодействующих процессов

1. Атомарность операций записи и считывания.
2. Синхронизация процессов.
3. Сборка «мусора».
4. Обмен сообщениями.

Какие еще алгоритмические проблемы возникают для систем взаимодействующих процессов?

# Архитектура распределенных систем

Коммуникационная подсистема строится из отдельных модулей, каждый из которых служит для выполнения очень специальной функции при поддержке других модулей. Модули образуют строгую **иерархию** : каждый модуль может обращаться только к тем модулям, которые непосредственно предшествуют ему в этом иерархическом порядке. Эти модули называются **уровнями** .

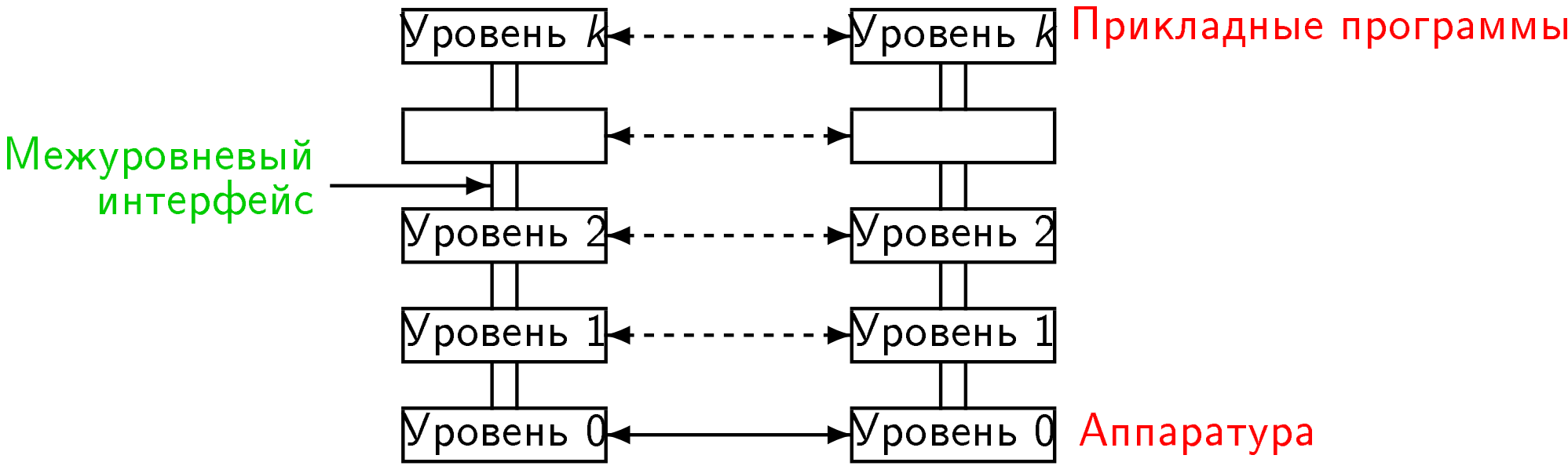



Рис.: Многоуровневая архитектура сети.

# Архитектура распределенных систем

**Архитектурой** сети называется совокупность всех модулей (уровней), вместе с определениями соответствующих интерфейсов и протоколов.

# Архитектура распределенных систем

**Архитектурой** сети называется совокупность всех модулей (уровней), вместе с определениями соответствующих интерфейсов и протоколов.

Необходимо обеспечить совместимость программного обеспечения узлов сети.

# Архитектура распределенных систем

**Архитектурой** сети называется совокупность всех модулей (уровней), вместе с определениями соответствующих интерфейсов и протоколов.

Необходимо обеспечить совместимость программного обеспечения узлов сети.

Поэтому были разработаны стандартные сетевые архитектуры.

# Архитектура распределенных систем

**Архитектурой** сети называется совокупность всех модулей (уровней), вместе с определениями соответствующих интерфейсов и протоколов.

Необходимо обеспечить совместимость программного обеспечения узлов сети.

Поэтому были разработаны стандартные сетевые архитектуры.

Два стандарта получили «официальный» статус:

- Эталонная модель **Взаимодействия Открытых Систем (Open-Systems Interconnection)**, утвержденная Международной Организацией Стандартов, ISO.
- Модель OSI для локальных сетей: стандарт IEEE, утвержденный Институтом инженеров по электротехнике и электронике, IEEE.
- Семейство протоколов TCP/IP.

# Архитектура распределенных систем

## Эталонная модель OSI

Самый низкий уровень иерархии (уровень 0) — аппаратный.
В определении уровня 0 приводятся описания типов используемых проводов, напряжения в цепи, и т.д.
В 0/1-интерфейсе описаны процедуры передачи непереработанной информации на уровне 1 по проводам связи.

# Архитектура распределенных систем

## Эталонная модель OSI

Самый низкий уровень иерархии (уровень 0) — аппаратный.
В определении уровня 0 приводятся описания типов используемых проводов, напряжения в цепи, и т.д.
В 0/1-интерфейсе описаны процедуры передачи непереработанной информации на уровне 1 по проводам связи.

Эталонная модель OSI имеет семь уровней:

1. **физический уровень**,
2. **уровень передачи данных (канальный уровень)**,
3. **сетевой уровень**,
4. **транспортный уровень**,
5. **сеансовый уровень**,
6. **представительский уровень**,
7. **прикладной уровень**.

# Эталонная модель OSI

1. **Физический уровень**

# Эталонная модель OSI

## 1. Физический уровень

предназначен для передачи последовательностей битов по каналам связи. В 1/2-интерфейсе указываются процедуры, посредством которых следующий уровень обращается к служебным функциям физического уровня. Служебные функции физического уровня ненадежны: при передаче потока данных в нем может возникнуть беспорядок.

# Эталонная модель OSI

## 1. Физический уровень

предназначен для передачи последовательностей битов по каналам связи. В 1/2-интерфейсе указываются процедуры, посредством которых следующий уровень обращается к служебным функциям физического уровня. Служебные функции физического уровня ненадежны: при передаче потока данных в нем может возникнуть беспорядок.

## 2. Канальный уровень

# Эталонная модель OSI

## 1. Физический уровень

предназначен для передачи последовательностей битов по каналам связи. В 1/2-интерфейсе указываются процедуры, посредством которых следующий уровень обращается к служебным функциям физического уровня. Служебные функции физического уровня ненадежны: при передаче потока данных в нем может возникнуть беспорядок.

## 2. Канальный уровень

предназначен для исправления ненадежности физического уровня и предоставления вышестоящим уровням надежных каналов передачи данных между узлами, непосредственно соединенными друг с другом.

# Эталонная модель OSI

3. Сетевой уровень

# Эталонная модель OSI

## 3. Сетевой уровень

предназначен для того, чтобы обеспечить связь между всеми узлами сети, а не только теми, между которыми есть физический канал связи. На этом уровне проводится выбор маршрутов в сети для соединения удаленных друг от друга узлов, а также управление нагрузкой по потоку сообщений в каждом узле и канале. Из-за сбоев в работе узлов некоторые сообщения могут быть потеряны, дублированы или перемешаны.

# Эталонная модель OSI

## 3. Сетевой уровень

предназначен для того, чтобы обеспечить связь между всеми узлами сети, а не только теми, между которыми есть физический канал связи. На этом уровне проводится выбор маршрутов в сети для соединения удаленных друг от друга узлов, а также управление нагрузкой по потоку сообщений в каждом узле и канале. Из-за сбоев в работе узлов некоторые сообщения могут быть потеряны, дублированы или перемешаны.

## 4. Транспортный уровень

# Эталонная модель OSI

## 3. Сетевой уровень

предназначен для того, чтобы обеспечить связь между всеми узлами сети, а не только теми, между которыми есть физический канал связи. На этом уровне проводится выбор маршрутов в сети для соединения удаленных друг от друга узлов, а также управление нагрузкой по потоку сообщений в каждом узле и канале. Из-за сбоев в работе узлов некоторые сообщения могут быть потеряны, дублированы или перемешаны.

## 4. Транспортный уровень

предназначен для того, чтобы исправить и тем самым скрыть ненадежность сетевого уровня, т.е. обеспечить надежную сквозную передачу данных из одного узла в другой.

# Эталонная модель OSI

## 5. Сеансовый уровень

# Эталонная модель OSI

## 5. Сеансовый уровень

предназначен для организации и поддержания сеансов связи между узлами сети. На сеансовом уровне разработаны средства **восстановления сообщений**, если какой-либо узел выходит из строя, а также механизмы **взаимного исключения**, если критические операции запрещается выполнять одновременно.

# Эталонная модель OSI

## 5. Сеансовый уровень

предназначен для организации и поддержания сеансов связи между узлами сети. На сеансовом уровне разработаны средства **восстановления сообщений**, если какой-либо узел выходит из строя, а также механизмы **взаимного исключения**, если критические операции запрещается выполнять одновременно.

## 6. Представительский уровень

# Эталонная модель OSI

## 5. Сеансовый уровень

предназначен для организации и поддержания сеансов связи между узлами сети. На сеансовом уровне разработаны средства **восстановления сообщений**, если какой-либо узел выходит из строя, а также механизмы **взаимного исключения**, если критические операции запрещается выполнять одновременно.

## 6. Представительский уровень

предназначен для преобразования данных в тех случаях, когда формы представления информации в одном узле отличаются от форм представления информации в другом узле, а также для **сжатия** и **восстановления** данных, для **шифрования** и **дешифрования** сообщений.

# Эталонная модель OSI

## 5. Сеансовый уровень

предназначен для организации и поддержания сеансов связи между узлами сети. На сеансовом уровне разработаны средства **восстановления сообщений**, если какой-либо узел выходит из строя, а также механизмы **взаимного исключения**, если критические операции запрещается выполнять одновременно.

## 6. Представительский уровень

предназначен для преобразования данных в тех случаях, когда формы представления информации в одном узле отличаются от форм представления информации в другом узле, а также для **сжатия** и **восстановления** данных, для **шифрования** и **дешифрования** сообщений.

## 7. Прикладной уровень

# Эталонная модель OSI

## 5. Сеансовый уровень

предназначен для организации и поддержания сеансов связи между узлами сети. На сеансовом уровне разработаны средства **восстановления сообщений**, если какой-либо узел выходит из строя, а также механизмы **взаимного исключения**, если критические операции запрещается выполнять одновременно.

## 6. Представительский уровень

предназначен для преобразования данных в тех случаях, когда формы представления информации в одном узле отличаются от форм представления информации в другом узле, а также для **сжатия** и **восстановления** данных, для **шифрования** и **дешифрования** сообщений.

## 7. Прикладной уровень

предназначен для обеспечения потребности пользователей в передаче файлов, в использовании электронной почты, виртуальных терминалов и пр.

# Модели OSI для локальных сетей

В локальных сетях некоторые из рассмотренных нами уровней становятся излишними.

# Модели OSI для локальных сетей

В локальных сетях некоторые из рассмотренных нами уровней становятся излишними.
Сетевой уровень становится практически ненужным.

# Модели OSI для локальных сетей

В локальных сетях некоторые из рассмотренных нами уровней становятся излишними.
Сетевой уровень становится практически ненужным.
Транспортный и представительский уровни упрощаются, поскольку шина по сравнению с сетью с двухточечным соединением обладает ограниченной неопределенностью.

# Модели OSI для локальных сетей

В локальных сетях некоторые из рассмотренных нами уровней становятся излишними.
Сетевой уровень становится практически ненужным.
Транспортный и представительский уровни упрощаются, поскольку шина по сравнению с сетью с двухточечным соединением обладает ограниченной неопределенностью.

Модели OSI имеют три основных уровня:

1. **физический уровень**,
2. **подуровень управления доступом к среде**,
3. **подуровень управления логической связью**.

# Модели OSI для локальных сетей

## 2.1. Подуровень управления доступом к среде

предназначен для разрешения тех конфликтов, которые возникают между узлами, использующими общую коммуникационную среду.

# Модели OSI для локальных сетей

## 2.1. Подуровень управления доступом к среде

предназначен для разрешения тех конфликтов, которые возникают между узлами, использующими общую коммуникационную среду.

## 2.2. Подуровень управления логической связью

имеет почти такое же назначение, какое имеет уровень передачи данных (канальный уровень) в модели OSI, — он предназначен для управления обменом данными между узлами. Этот уровень нужен для того, чтобы следить за ошибками и управлять потоком информации.

# Особенности языков программирования

## Параллелизм.

Чтобы выразить параллелизм достаточно определить несколько процессов , каждый из которых является последовательной процедурой. В языке могут быть возможности для статического и для динамического определения процессов.

# Особенности языков программирования

## Параллелизм.

Чтобы выразить параллелизм достаточно определить несколько процессов , каждый из которых является последовательной процедурой. В языке могут быть возможности для статического и для динамического определения процессов.

## Коммуникация.

Коммуникация осуществляется посредством обмена сообщениями (обеспечивает как коммуникацию, так и синхронизацию) и/или при помощи совместно используемой памяти (обеспечивает только коммуникацию).

# Особенности языков программирования

## Параллелизм.

Чтобы выразить параллелизм достаточно определить несколько процессов , каждый из которых является последовательной процедурой. В языке могут быть возможности для статического и для динамического определения процессов.

## Коммуникация.

Коммуникация осуществляется посредством обмена сообщениями (обеспечивает как коммуникацию, так и синхронизацию) и/или при помощи совместно используемой памяти (обеспечивает только коммуникацию).

## Недетерминизм.

Недетерминизм выражается при помощи охраняемых команд . Охраняемая команда — это список операторов, каждому из которых предшествует булево выражение (предохранитель). Процесс может продолжить свое выполнение, выбрав любой оператор, предохранитель которого принимает значение true.

# Особенности распределенных алгоритмов

## 1. Отсутствие сведений о глобальном состоянии.

Узлам распределенной системы открыт доступ только к их собственным состояниям, а не к состоянию всей системы. Поэтому нельзя управлять вычислением, основываясь на информации о глобальном состоянии системы.

# Особенности распределенных алгоритмов

## 1. Отсутствие сведений о глобальном состоянии.

Узлам распределенной системы открыт доступ только к их собственным состояниям, а не к состоянию всей системы. Поэтому нельзя управлять вычислением, основываясь на информации о глобальном состоянии системы.

Узлы системы не могут оценить состояние коммуникационной подсистемы, поскольку содержимое каналов связи не поддается прямому наблюдению со стороны узлов.

# Особенности распределенных алгоритмов

## 1. Отсутствие сведений о глобальном состоянии.

Узлам распределенной системы открыт доступ только к их собственным состояниям, а не к состоянию всей системы. Поэтому нельзя управлять вычислением, основываясь на информации о глобальном состоянии системы.

Узлы системы не могут оценить состояние коммуникационной подсистемы, поскольку содержимое каналов связи не поддается прямому наблюдению со стороны узлов.

## 2. Отсутствие глобальной шкалы времени.

Отношение предшествования по времени, введенное на множестве событий, которые возникают при выполнении распределенного алгоритма, **не является тотальным**.

# Особенности распределенных алгоритмов

## 3. Недетерминизм.

Выполнение распределенной системы, как правило, оказывается недетерминированным, ввиду того, что быстродействие различных компонентов системы оказывается разным и может изменяться со временем.

# Особенности распределенных алгоритмов

## 3. Недетерминизм.

Выполнение распределенной системы, как правило, оказывается недетерминированным, ввиду того, что быстродействие различных компонентов системы оказывается разным и может изменяться со временем.

Все это вместе взятое —

- отсутствие сведений о глобальном состоянии,
- отсутствие глобальной шкалы времени,
- недетерминизм.

— приводит к тому, что задача проектирования распределенных алгоритмов становится весьма замысловатой.

# Пример связи с передачей одного сообщения

## Задача надежного обмена информацией по ненадежному каналу связи.

Два процесса $a$ и $b$ связаны друг с другом в некоторой сети, которая позволяет передавать данные от одного процесса к другому. Сообщение может быть доставлено сколь угодно поздно после своего отправления, а может и вообще потеряться в сети.

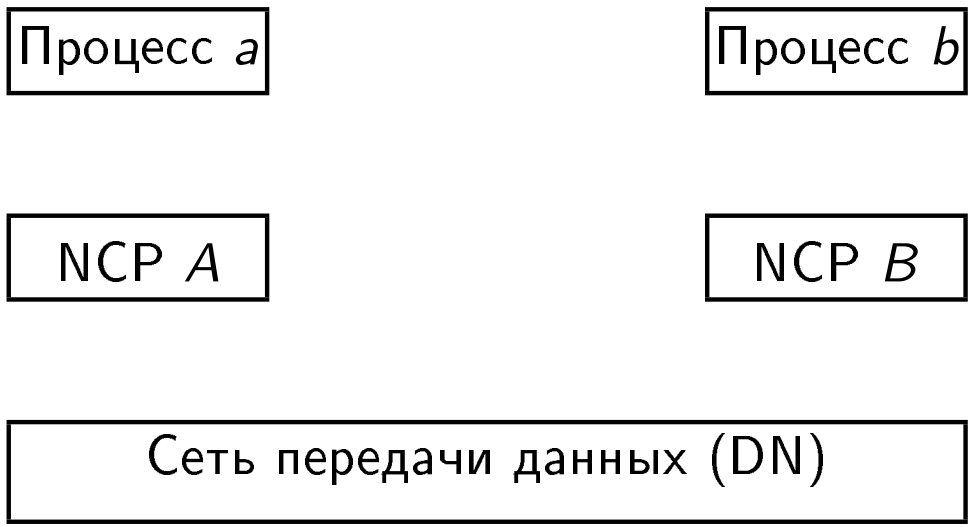
# Пример связи с передачей одного сообщения

## Задача надежного обмена информацией по ненадежному каналу связи.

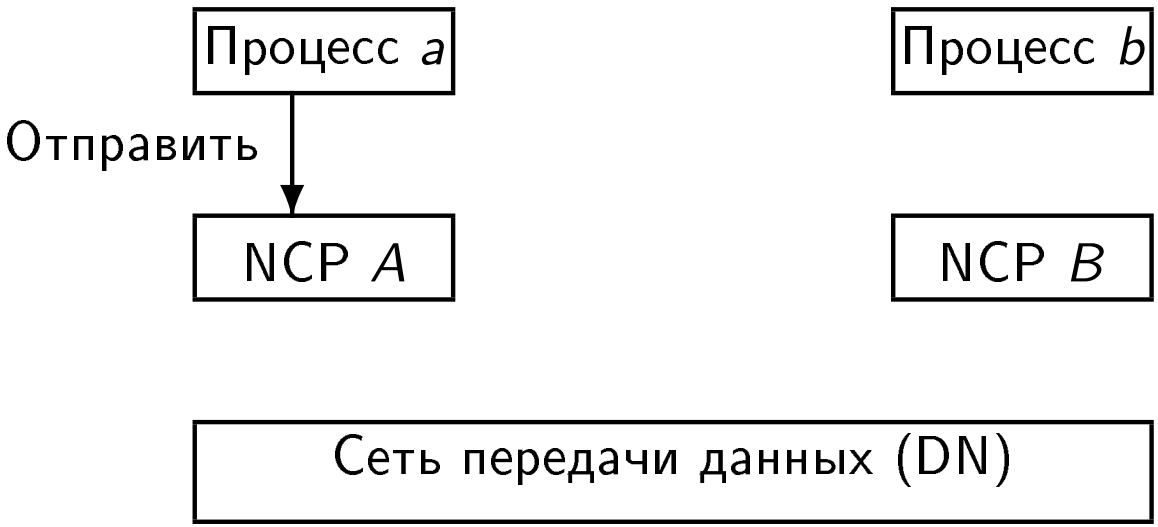
Два процесса $a$ и $b$ связаны друг с другом в некоторой сети, которая позволяет передавать данные от одного процесса к другому. Сообщение может быть доставлено сколь угодно поздно после своего отправления, а может и вообще потеряться в сети.

Для передачи и приема сообщений используются процедуры управления сетью (NCP), при помощи которых процессы $a$ и $b$ получают доступ к сети. Процесс $a$ начинает устанавливать соединение, передав одну единицу информации $m$ процедуре $A$ . Взаимодействие между процедурами управления сетью через сеть передачи данных DN должно быть организовано так, чтобы информация $m$ была доставлена процессу $b$ (при посредничестве NCP $B$ ), и после этого процесс $a$ должен быть оповещен об этой доставке (при посредничестве NCP $A$ ).

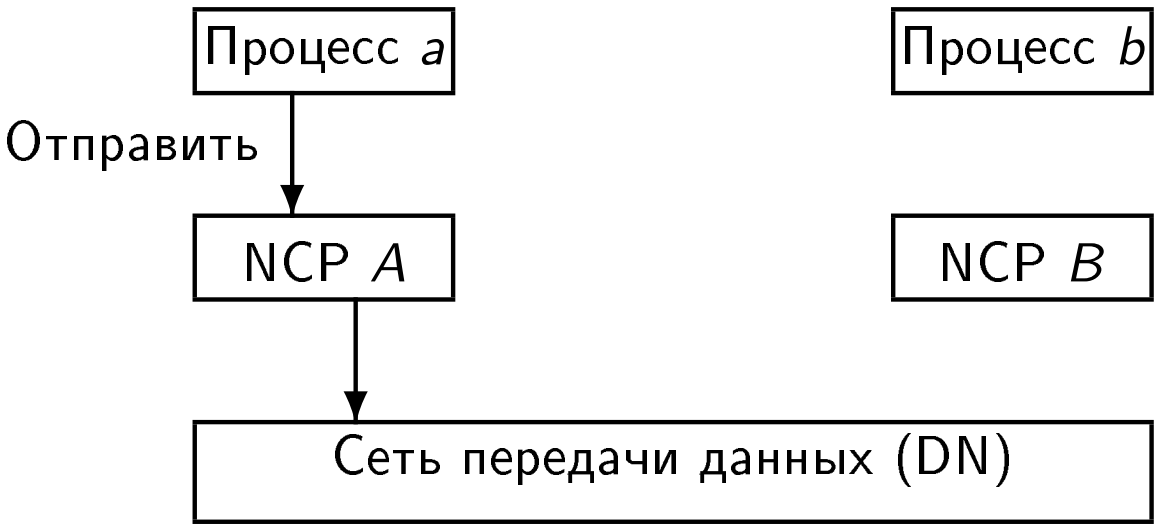
# Задача надежного обмена информацией



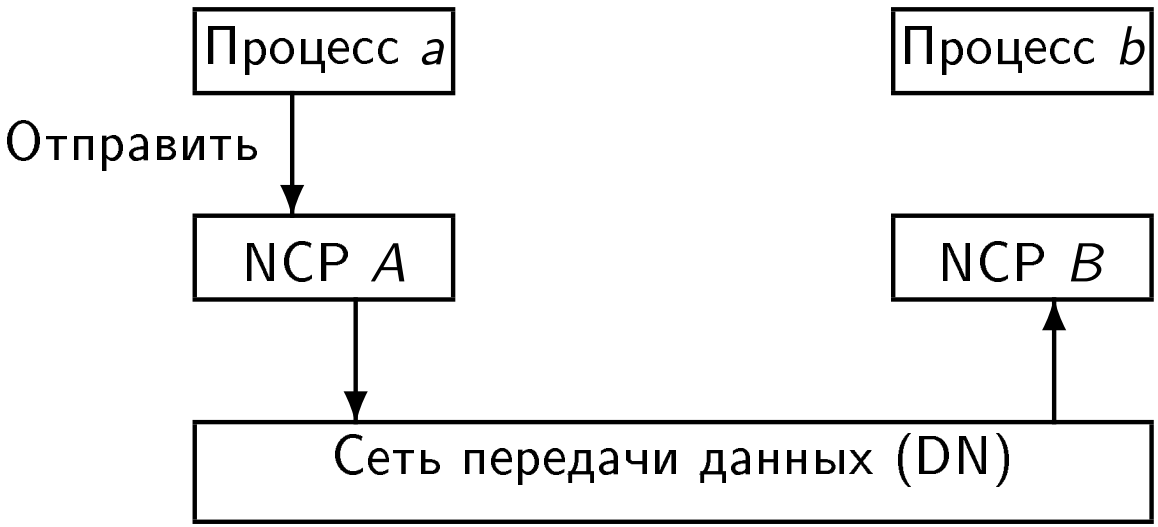
# Задача надежного обмена информацией



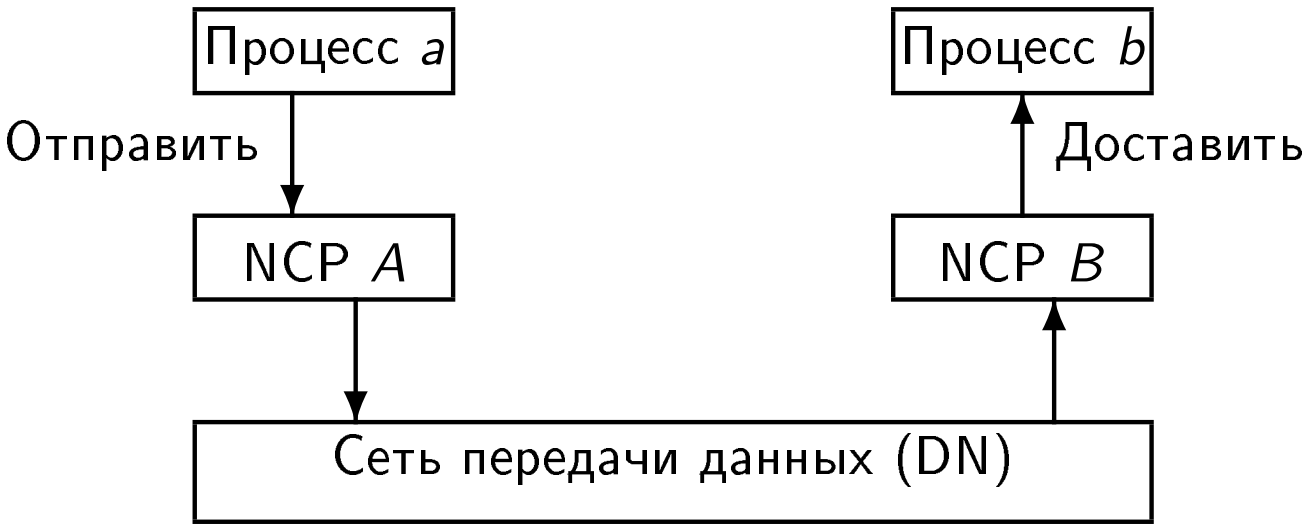
# Задача надежного обмена информацией



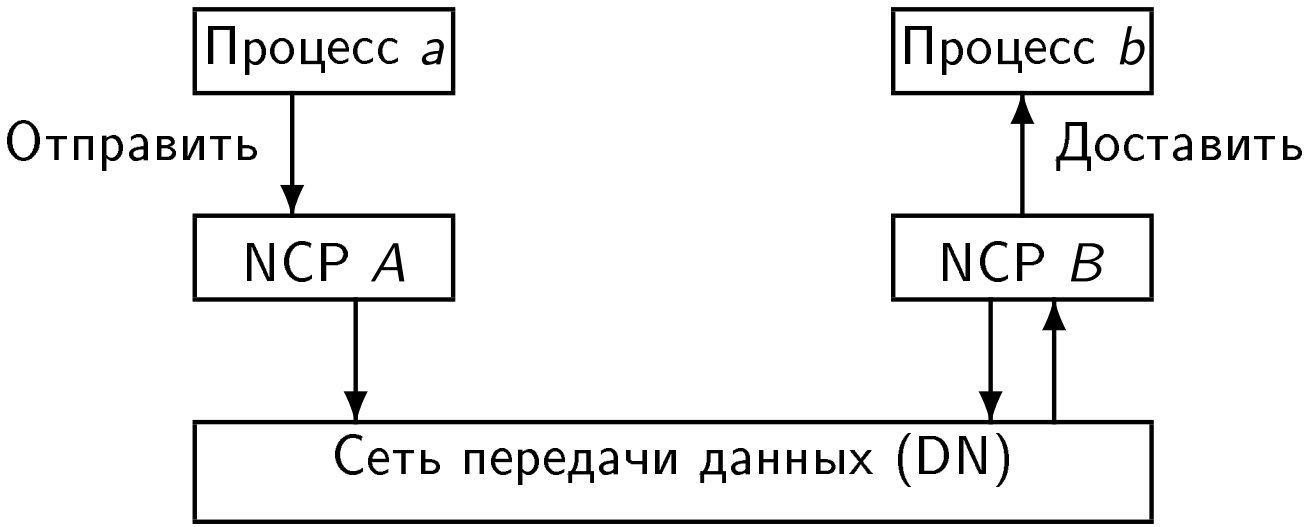
# Задача надежного обмена информацией



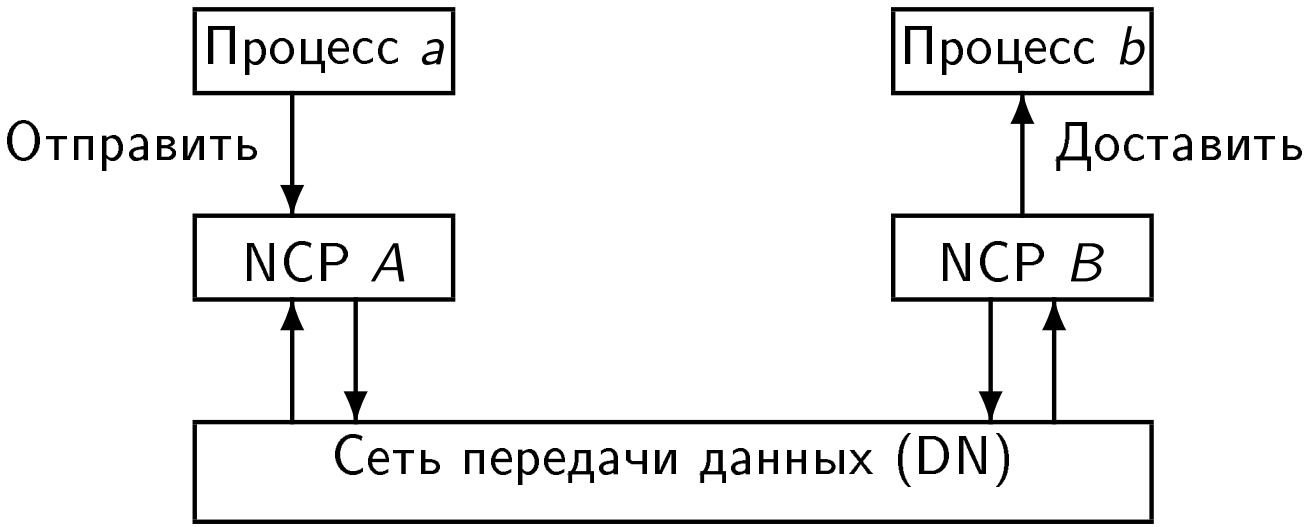
# Задача надежного обмена информацией



# Задача надежного обмена информацией



# Задача надежного обмена информацией



# Задача надежного обмена информацией


Рис.: Упрощенная архитектура сети.

# Задача надежного обмена информацией

Ненадежность сети вынуждает NCP $A$ и $B$ вступить в диалог.

# Задача надежного обмена информацией

Ненадежность сети вынуждает NCP $A$ и $B$ вступить в диалог.

## Правила диалога.

- NCP сохраняют информацию о состоянии диалога, но после завершения обмена сообщениями вся информация о состоянии диалога стирается.

# Задача надежного обмена информацией

Ненадежность сети вынуждает NCP $A$ и $B$ вступить в диалог.

## Правила диалога.

- NCP сохраняют информацию о состоянии диалога, но после завершения обмена сообщениями вся информация о состоянии диалога стирается.
- Инициализация информации о состоянии диалога называется открытием диалога.

# Задача надежного обмена информацией

Ненадежность сети вынуждает NCP $A$ и $B$ вступить в диалог.

## Правила диалога.

- NCP сохраняют информацию о состоянии диалога, но после завершения обмена сообщениями вся информация о состоянии диалога стирается.
- Инициализация информации о состоянии диалога называется открытием диалога.
- Сброс состояния называется закрытием диалога. После закрытия диалога NCP перейдет в то состояние, в котором она пребывала перед открытием этого диалога (закрытое состояние ).

# Задача надежного обмена информацией

Ненадежность сети вынуждает NCP $A$ и $B$ вступить в диалог.

## Правила диалога.

- NCP сохраняют информацию о состоянии диалога, но после завершения обмена сообщениями вся информация о состоянии диалога стирается.
- Инициализация информации о состоянии диалога называется открытием диалога.
- Сброс состояния называется закрытием диалога. После закрытия диалога NCP перейдет в то состояние, в котором она пребывала перед открытием этого диалога (закрытое состояние ).
- Информация $m$ была потеряна , если процесс $a$ был оповещен о приеме этой информации процессом $b$ , но сама информация не была доставлена процессу $b$ .

# Задача надежного обмена информацией

Ненадежность сети вынуждает NCP $A$ и $B$ вступить в диалог.

## Правила диалога.

- NCP сохраняют информацию о состоянии диалога, но после завершения обмена сообщениями вся информация о состоянии диалога стирается.
- Инициализация информации о состоянии диалога называется открытием диалога.
- Сброс состояния называется закрытием диалога. После закрытия диалога NCP перейдет в то состояние, в котором она пребывала перед открытием этого диалога (закрытое состояние ).
- Информация $m$ была потеряна , если процесс $a$ был оповещен о приеме этой информации процессом $b$ , но сама информация не была доставлена процессу $b$ .
- Информация $m$ была дублирована , если она была доставлена дважды.

# Задача надежного обмена информацией

## Надежная связь невозможна.

Допустим NCP $B$ выходит из строя и перезапускается в закрытом состоянии после того, как NCP $A$ уже отправила сообщение $m$ . Тогда ни NCP $A$ , ни NCP $B$ не могут сказать, была ли информация $m$ уже доставлена в момент поломки NCP $B$ .

# Задача надежного обмена информацией

## Надежная связь невозможна.

Допустим NCP $B$ выходит из строя и перезапускается в закрытом состоянии после того, как NCP $A$ уже отправила сообщение $m$ . Тогда ни NCP $A$ , ни NCP $B$ не могут сказать, была ли информация $m$ уже доставлена в момент поломки NCP $B$ .

Процедура $A$ не может этого сделать, потому что она не может судить о том, какие события произошли в NCP $B$ (из-за отсутствия сведений о глобальном состоянии)

# Задача надежного обмена информацией

## Надежная связь невозможна.

Допустим NCP $B$ выходит из строя и перезапускается в закрытом состоянии после того, как NCP $A$ уже отправила сообщение $m$ . Тогда ни NCP $A$ , ни NCP $B$ не могут сказать, была ли информация $m$ уже доставлена в момент поломки NCP $B$ .

Процедура $A$ не может этого сделать, потому что она не может судить о том, какие события произошли в NCP $B$ (из-за отсутствия сведений о глобальном состоянии)

Процедура $B$ не может этого сделать, потому что она вышла из строя и была перезапущена в закрытом состоянии.

# Задача надежного обмена информацией

## Надежная связь невозможна.

Допустим NCP $B$ выходит из строя и перезапускается в закрытом состоянии после того, как NCP $A$ уже отправила сообщение $m$ . Тогда ни NCP $A$ , ни NCP $B$ не могут сказать, была ли информация $m$ уже доставлена в момент поломки NCP $B$ .

Процедура $A$ не может этого сделать, потому что она не может судить о том, какие события произошли в NCP $B$ (из-за отсутствия сведений о глобальном состоянии)

Процедура $B$ не может этого сделать, потому что она вышла из строя и была перезапущена в закрытом состоянии.

Если NCP $A$ вновь отправит сообщение NCP $B$ , и NCP $B$ доставит это сообщение, то произойдет дублирование информации.

# Задача надежного обмена информацией

## Надежная связь невозможна.

Допустим NCP $B$ выходит из строя и перезапускается в закрытом состоянии после того, как NCP $A$ уже отправила сообщение $m$ . Тогда ни NCP $A$ , ни NCP $B$ не могут сказать, была ли информация $m$ уже доставлена в момент поломки NCP $B$ .

Процедура $A$ не может этого сделать, потому что она не может судить о том, какие события произошли в NCP $B$ (из-за отсутствия сведений о глобальном состоянии)

Процедура $B$ не может этого сделать, потому что она вышла из строя и была перезапущена в закрытом состоянии.

Если NCP $A$ вновь отправит сообщение NCP $B$ , и NCP $B$ доставит это сообщение, то произойдет дублирование информации.

Если подтверждение $a$ будет отправлено, тогда как доставка информации $m$ не состоялась, то случится потеря информации.

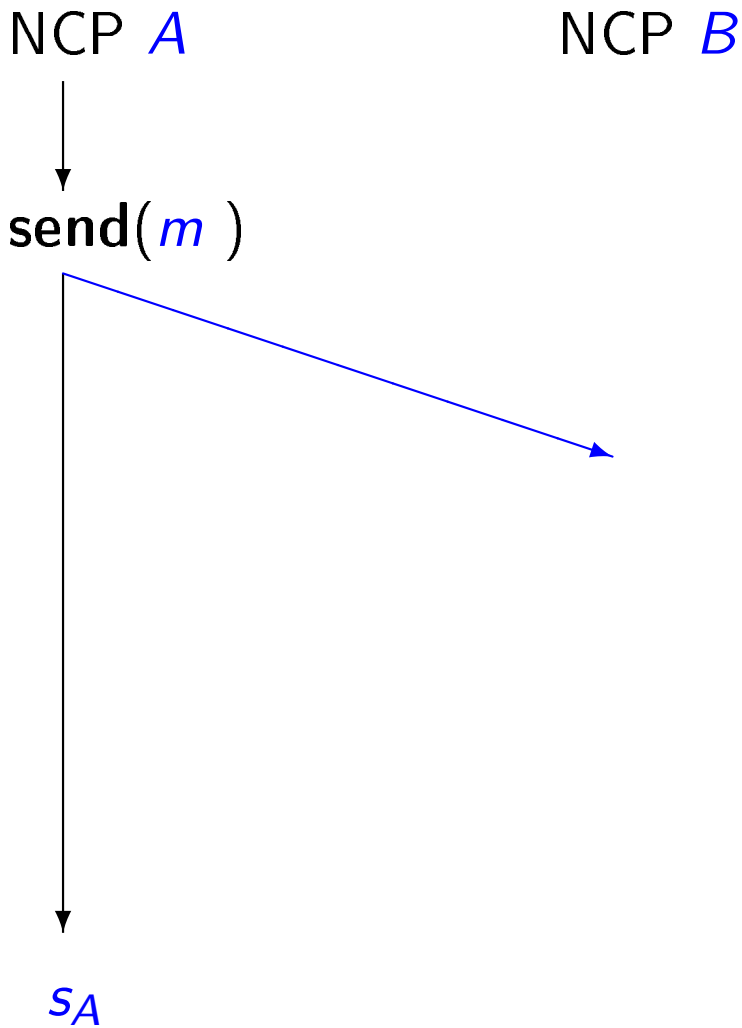
# Задача надежного обмена информацией

NCP *A*

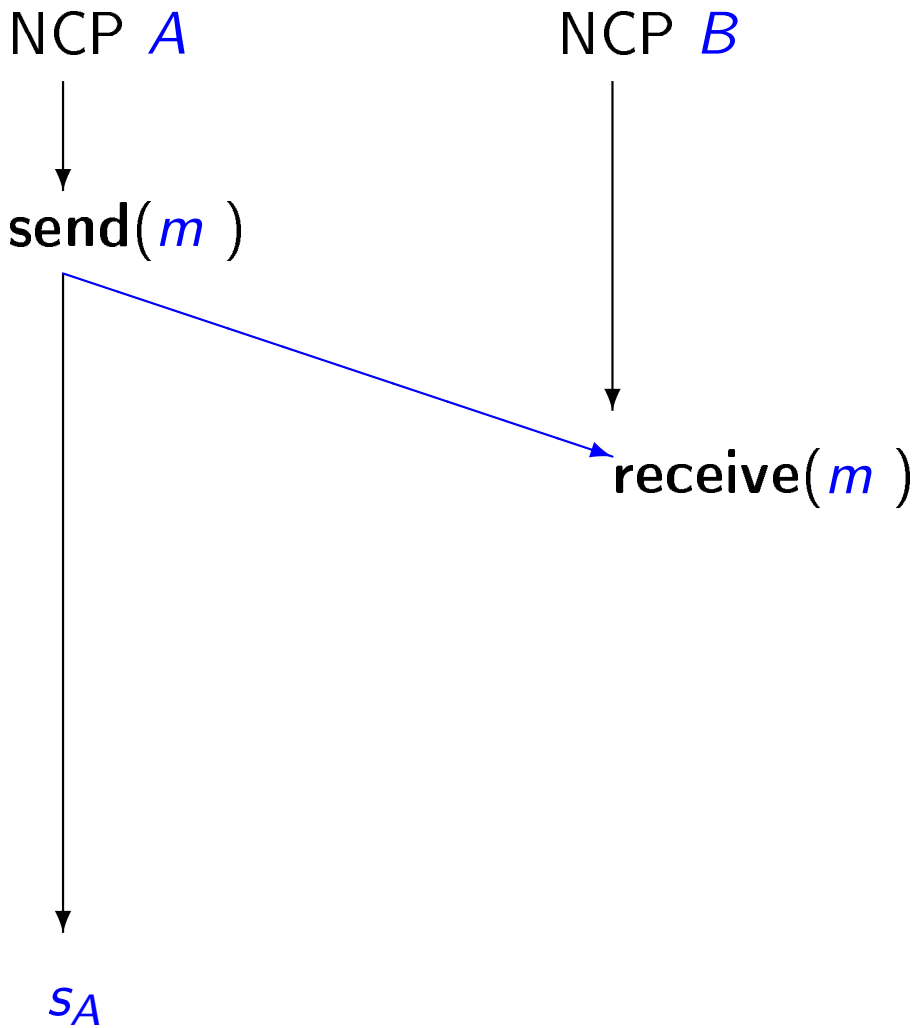
NCP *B*

# Задача надежного обмена информацией

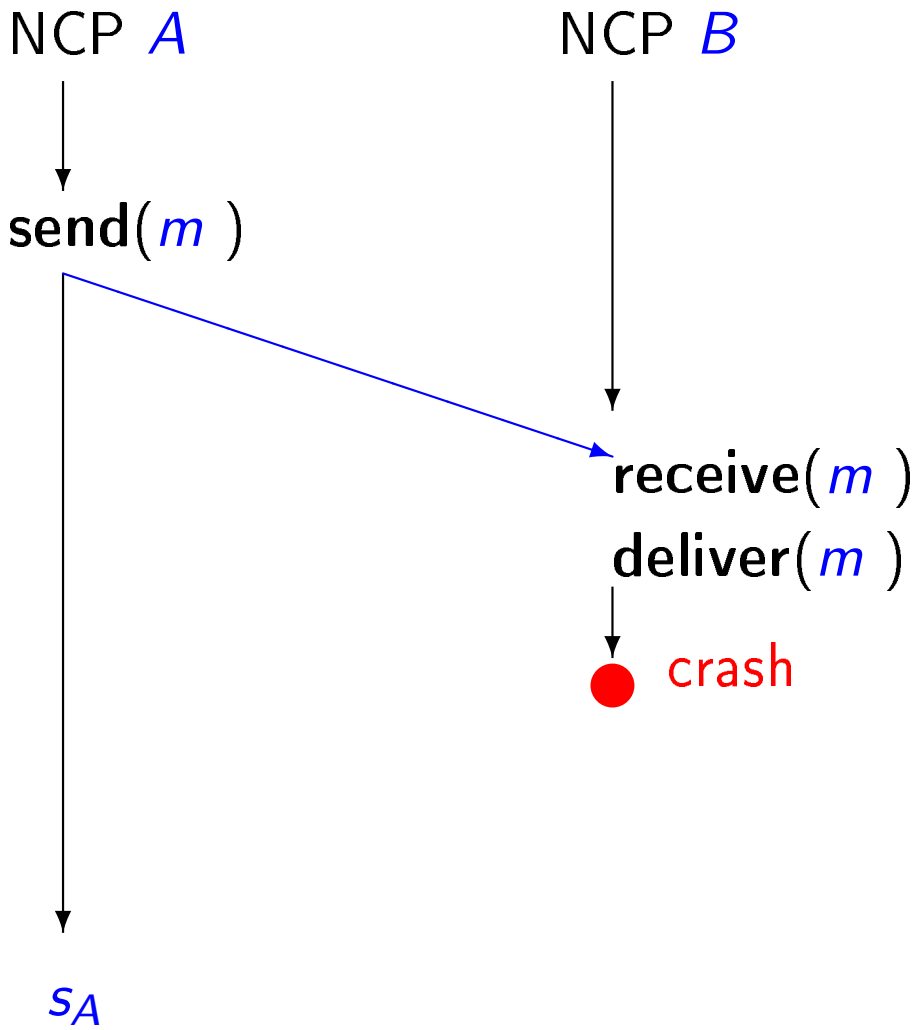
# Задача надежного обмена информацией



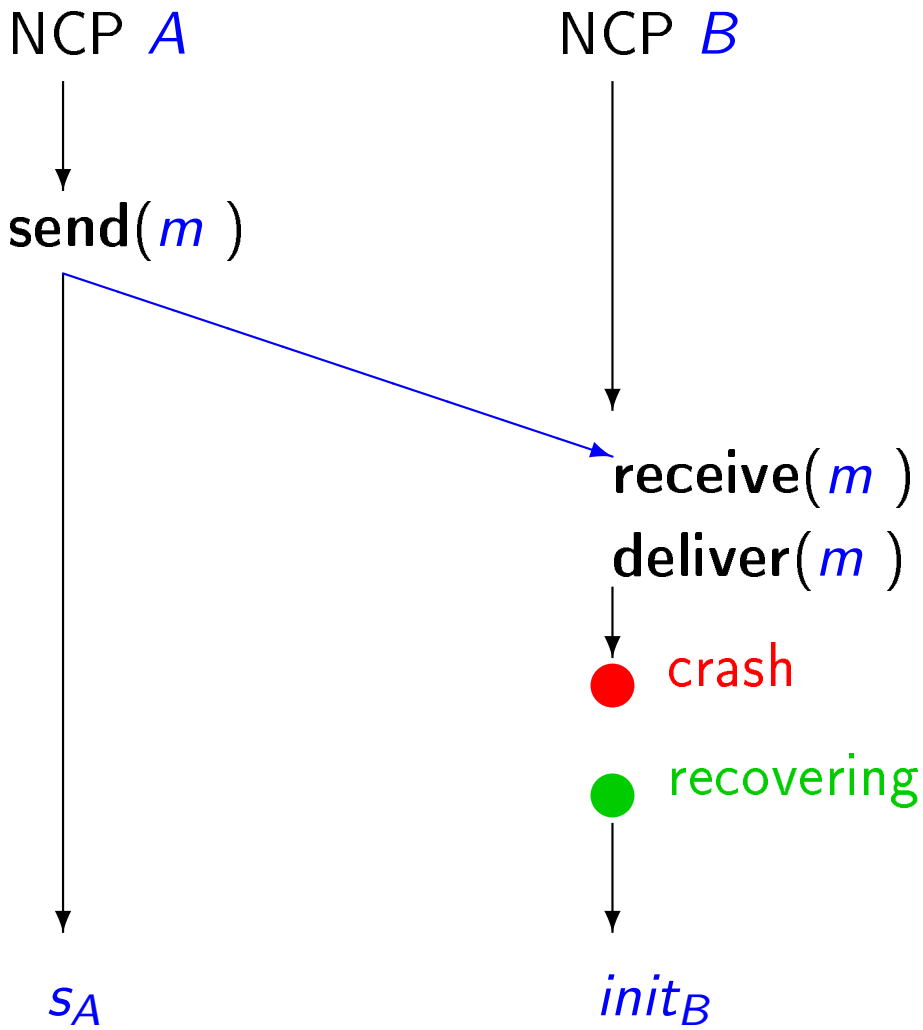
# Задача надежного обмена информацией



# Задача надежного обмена информацией



# Задача надежного обмена информацией



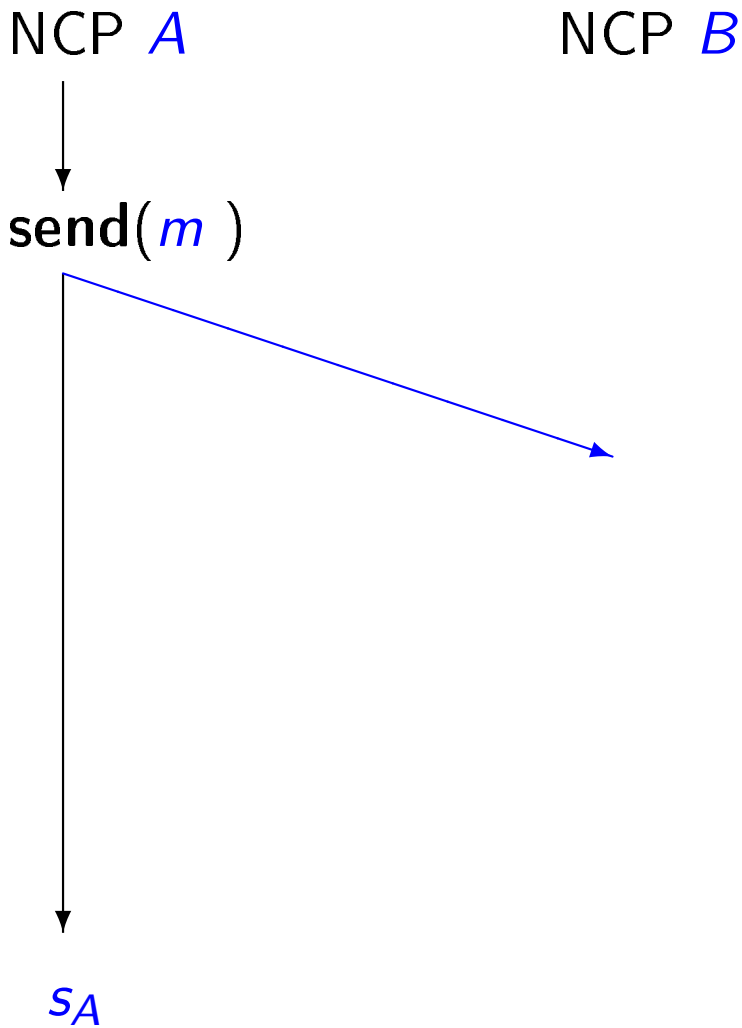
# Задача надежного обмена информацией

NCP *A*

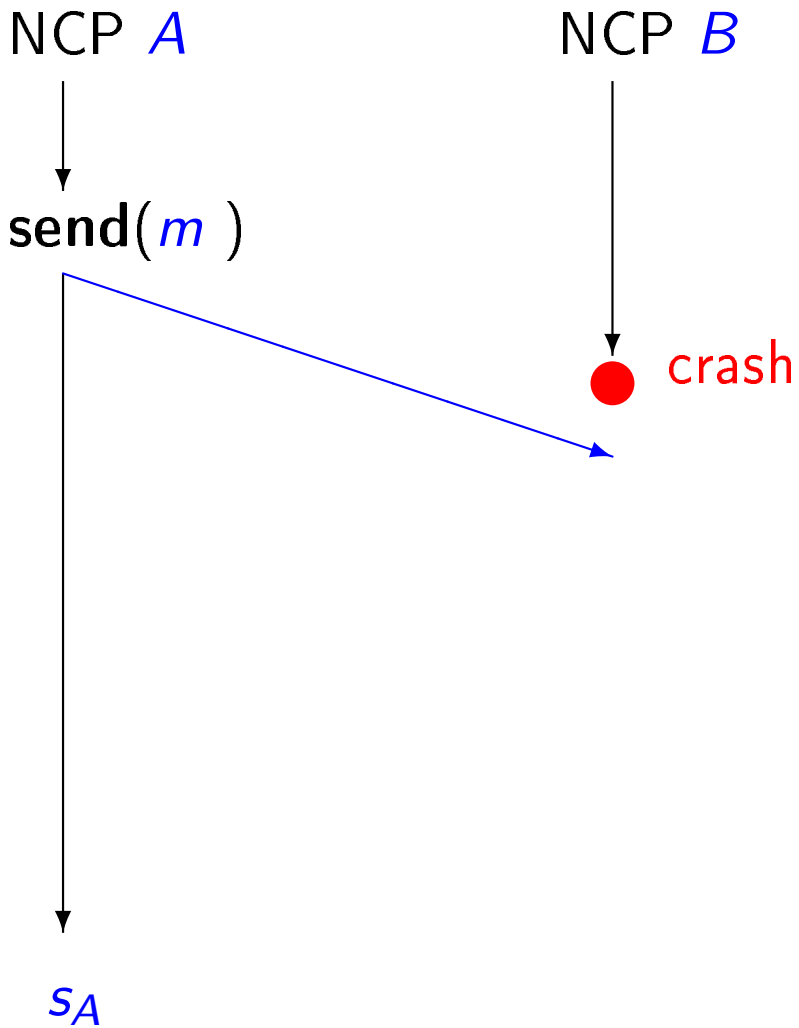
NCP *B*

# Задача надежного обмена информацией

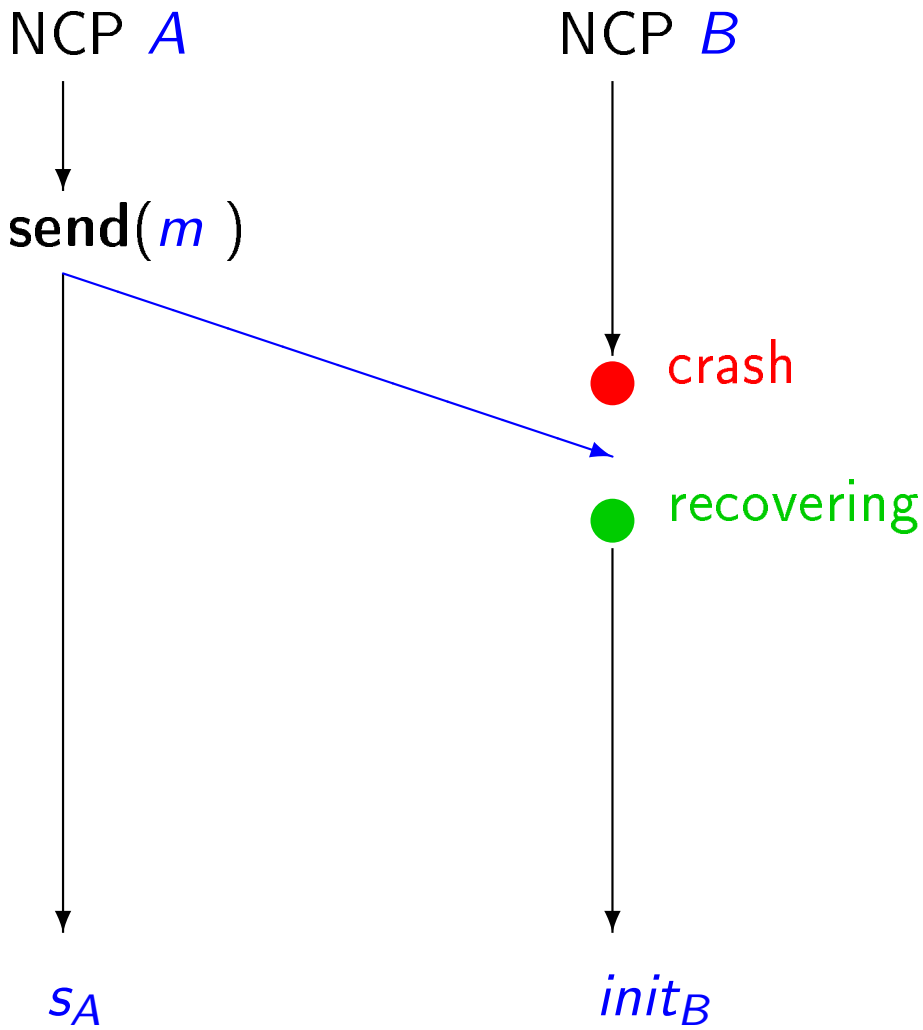
# Задача надежного обмена информацией



# Задача надежного обмена информацией



# Задача надежного обмена информацией



# Задача надежного обмена информацией

## Диалог с одним сообщением.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *close*

Потеря сообщения происходит всякий раз, когда сеть отказывается доставить сообщение по назначению, однако, здесь нет никакой возможности для дублирования сообщения.

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

1. *NCP A* : $send\ \langle \mathbf{data}, m \rangle$

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *потеряно в сети*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *потеряно в сети*
4. *NCP A* : *timeout, send* $\langle \textbf{\textit{data}}, m \rangle$

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *потеряно в сети*
4. *NCP A* : *timeout*, *send* $\langle \textbf{\textit{data}}, m \rangle$
5. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Однако и этот протокол не является надежным: при задержке подтверждения возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *DN* $\langle \textbf{\textit{ack}} \rangle$ *потеряно в сети*
4. *NCP A* : *timeout*, *send* $\langle \textbf{\textit{data}}, m \rangle$
5. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
6. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : $send\ \langle \boldsymbol{data}, m_1 \rangle$

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout*, *send* $\langle \textbf{\textit{data}}, m_1 \rangle$

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout*, *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A :* send $\langle \textbf{data}, m_1 \rangle$
2. *NCP B :* receive $\langle \textbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \textbf{ack} \rangle$, close
3. *NCP A :* timeout, send $\langle \textbf{data}, m_1 \rangle$
4. *NCP B :* receive $\langle \textbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \textbf{ack} \rangle$, close
5. *NCP A :* receive $\langle \textbf{ack} \rangle$ (шаг 1), notify, close

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : send $\langle \mathbf{data}, m_1 \rangle$
2. *NCP B* : receive $\langle \mathbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \mathbf{ack} \rangle$, close
3. *NCP A* : timeout, send $\langle \mathbf{data}, m_1 \rangle$
4. *NCP B* : receive $\langle \mathbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \mathbf{ack} \rangle$, close
5. *NCP A* : receive $\langle \mathbf{ack} \rangle$ (шаг 1), notify, close
6. *NCP A* : send $\langle \mathbf{data}, m_2 \rangle$

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : send $\langle \mathbf{data}, m_1 \rangle$
2. *NCP B* : receive $\langle \mathbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \mathbf{ack} \rangle$, close
3. *NCP A* : timeout, send $\langle \mathbf{data}, m_1 \rangle$
4. *NCP B* : receive $\langle \mathbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \mathbf{ack} \rangle$, close
5. *NCP A* : receive $\langle \mathbf{ack} \rangle$ (шаг 1), notify, close
6. *NCP A* : send $\langle \mathbf{data}, m_2 \rangle$
7. *DN* $\langle \mathbf{data}, m_2 \rangle$ потеряно в сети

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{data}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{data}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{ack} \rangle$, *close*
3. *NCP A* : *timeout, send* $\langle \textbf{data}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{data}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{ack} \rangle$, *close*
5. *NCP A* : *receive* $\langle \textbf{ack} \rangle$ *(шаг 1), notify, close*
6. *NCP A* : *send* $\langle \textbf{data}, m_2 \rangle$
7. *DN* $\langle \textbf{data}, m_2 \rangle$ *потеряно в сети*
8. *NCP A* : *receive* $\langle \textbf{ack} \rangle$ *(шаг 2), notify, close*

# Задача надежного обмена информацией

## Диалог с двумя сообщениями.

Кроме того, при передаче нескольких сообщений возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
3. *NCP A* : *timeout*, *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
4. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$, *close*
5. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$ *(шаг 1)*, *notify*, *close*
6. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$
7. *DN* $\langle \textbf{\textit{data}}, m_2 \rangle$ *потеряно в сети*
8. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$ *(шаг 2)*, *notify*, *close*

Замедление передачи сообщения обнаружить невозможно, ввиду отсутствия глобальной шкалы времени.

# Задача надежного обмена информацией

Диалог с тремя сообщениями.

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *NCP B* : *receive* $\langle \textbf{\textit{close}} \rangle$, *close*

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m \rangle$, *deliver* $m$, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *NCP B* : *receive* $\langle \textbf{\textit{close}} \rangle$, *close*

В этом случае прежние сценарии, приводящие к ошибкам, уже невозможны.

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : $send\ \langle \mathbf{data}, m_1 \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *потеряно в сети*

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *потеряно в сети*
5. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_1 \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify*, *send* $\langle \textbf{\textit{close}} \rangle$, *close*
4. *DN* $\langle \textbf{\textit{close}} \rangle$ *потеряно в сети*
5. *NCP A* : *send* $\langle \textbf{\textit{data}}, m_2 \rangle$
6. *DN* $\langle \textbf{\textit{data}}, m_2 \rangle$ *потеряно в сети*

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

1. *NCP A* : send $\langle \mathbf{data}, m_1 \rangle$
2. *NCP B* : receive $\langle \mathbf{data}, m_1 \rangle$, deliver $m_1$, send $\langle \mathbf{ack} \rangle$
3. *NCP A* : receive $\langle \mathbf{ack} \rangle$, notify, send $\langle \mathbf{close} \rangle$, close
4. *DN* $\langle \mathbf{close} \rangle$ потеряно в сети
5. *NCP A* : send $\langle \mathbf{data}, m_2 \rangle$
6. *DN* $\langle \mathbf{data}, m_2 \rangle$ потеряно в сети
7. *NCP B* : timeout, retransmit $\langle \mathbf{ack} \rangle$ (для шага 2)

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

| | | |
|---|---|---|
| 1. | *NCP A* : | *send* $\langle \textbf{\textit{data}}, m_1 \rangle$ |
| 2. | *NCP B* : | *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$ |
| 3. | *NCP A* : | *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close* |
| 4. | *DN* | $\langle \textbf{\textit{close}} \rangle$ *потеряно в сети* |
| 5. | *NCP A* : | *send* $\langle \textbf{\textit{data}}, m_2 \rangle$ |
| 6. | *DN* | $\langle \textbf{\textit{data}}, m_2 \rangle$ *потеряно в сети* |
| 7. | *NCP B* : | *timeout, retransmit* $\langle \textbf{\textit{ack}} \rangle$ *(для шага 2)* |
| 8. | *NCP A* : | *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close* |

# Задача надежного обмена информацией

## Диалог с тремя сообщениями.

Однако и этот протокол не является надежным: возможна потеря сообщения.

| | | |
|---|---|---|
| 1. | *NCP A* : | *send* $\langle \textbf{\textit{data}}, m_1 \rangle$ |
| 2. | *NCP B* : | *receive* $\langle \textbf{\textit{data}}, m_1 \rangle$, *deliver* $m_1$, *send* $\langle \textbf{\textit{ack}} \rangle$ |
| 3. | *NCP A* : | *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close* |
| 4. | *DN* | $\langle \textbf{\textit{close}} \rangle$ *потеряно в сети* |
| 5. | *NCP A* : | *send* $\langle \textbf{\textit{data}}, m_2 \rangle$ |
| 6. | *DN* | $\langle \textbf{\textit{data}}, m_2 \rangle$ *потеряно в сети* |
| 7. | *NCP B* : | *timeout, retransmit* $\langle \textbf{\textit{ack}} \rangle$ *(для шага 2)* |
| 8. | *NCP A* : | *receive* $\langle \textbf{\textit{ack}} \rangle$, *notify, send* $\langle \textbf{\textit{close}} \rangle$, *close* |
| 9. | *NCP B* : | *receive* $\langle \textbf{\textit{close}} \rangle$, *close* |

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m, x \rangle$
2. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m, x \rangle$, *send* $\langle \textbf{\textit{ack}}, x, y \rangle$
3. *NCP A* : *receive* $\langle \textbf{\textit{ack}}, x, y \rangle$, *notify, send* $\langle \textbf{\textit{close}}, x, y \rangle$, *close*
4. *NCP B* : *receive* $\langle \textbf{\textit{close}}, x, y \rangle$, *deliver m, close*

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A* : $send\ \langle \mathbf{data}, m, x \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A* : $send\ \langle \mathbf{data}, m, x \rangle$
2. *NCP A* : $timeout,\ retransmit\ \langle \mathbf{data}, m, x \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m, x \rangle$
2. *NCP A* : *timeout, retransmit* $\langle \textbf{\textit{data}}, m, x \rangle$
3. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m, x \rangle$ *(на шаге 2), send* $\langle \textbf{\textit{ack}}, x, y_1 \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A* : *send* $\langle \textbf{\textit{data}}, m, x \rangle$
2. *NCP A* : *timeout, retransmit* $\langle \textbf{\textit{data}}, m, x \rangle$
3. *NCP B* : *receive* $\langle \textbf{\textit{data}}, m, x \rangle$ *(на шаге 2), send* $\langle \textbf{\textit{ack}}, x, y_1 \rangle$
4. *NCP A* : *receive* $\langle \textbf{\textit{ack}}, x, y_1 \rangle$*, notify, send* $\langle \textbf{\textit{close}}, x, y_1 \rangle$*, close*

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A :* send $\langle \textbf{data}, m, x \rangle$
2. *NCP A :* timeout, retransmit $\langle \textbf{data}, m, x \rangle$
3. *NCP B :* receive $\langle \textbf{data}, m, x \rangle$ (на шаге 2), send $\langle \textbf{ack}, x, y_1 \rangle$
4. *NCP A :* receive $\langle \textbf{ack}, x, y_1 \rangle$, notify, send $\langle \textbf{close}, x, y_1 \rangle$, close
5. *NCP B :* receive $\langle \textbf{close}, x, y_1 \rangle$, deliver $m$, close

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A :* send $\langle \mathbf{data}, m, x \rangle$
2. *NCP A :* timeout, retransmit $\langle \mathbf{data}, m, x \rangle$
3. *NCP B :* receive $\langle \mathbf{data}, m, x \rangle$ (на шаге 2), send $\langle \mathbf{ack}, x, y_1 \rangle$
4. *NCP A :* receive $\langle \mathbf{ack}, x, y_1 \rangle$, notify, send $\langle \mathbf{close}, x, y_1 \rangle$, close
5. *NCP B :* receive $\langle \mathbf{close}, x, y_1 \rangle$, deliver $m$, close
6. *NCP B :* receive $\langle \mathbf{data}, m, x \rangle$ (на шаге 1), send $\langle \mathbf{ack}, x, y_2 \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A :* send $\langle \textbf{data}, m, x \rangle$
2. *NCP A :* timeout, retransmit $\langle \textbf{data}, m, x \rangle$
3. *NCP B :* receive $\langle \textbf{data}, m, x \rangle$ (на шаге 2), send $\langle \textbf{ack}, x, y_1 \rangle$
4. *NCP A :* receive $\langle \textbf{ack}, x, y_1 \rangle$, notify, send $\langle \textbf{close}, x, y_1 \rangle$, close
5. *NCP B :* receive $\langle \textbf{close}, x, y_1 \rangle$, deliver $m$, close
6. *NCP B :* receive $\langle \textbf{data}, m, x \rangle$ (на шаге 1), send $\langle \textbf{ack}, x, y_2 \rangle$
7. *NCP A :* receive $\langle \textbf{ack}, x, y_2 \rangle$, reply $\langle \textbf{nocon}, x, y_2 \rangle$

# Задача надежного обмена информацией

## Диалог с тремя сообщениями (исправленный вариант).

Однако и этот протокол не является надежным: возможно дублирование сообщения.

1. *NCP A :* send $\langle \textbf{data}, m, x \rangle$
2. *NCP A :* timeout, retransmit $\langle \textbf{data}, m, x \rangle$
3. *NCP B :* receive $\langle \textbf{data}, m, x \rangle$ (на шаге 2), send $\langle \textbf{ack}, x, y_1 \rangle$
4. *NCP A :* receive $\langle \textbf{ack}, x, y_1 \rangle$, notify, send $\langle \textbf{close}, x, y_1 \rangle$, close
5. *NCP B :* receive $\langle \textbf{close}, x, y_1 \rangle$, deliver $m$, close
6. *NCP B :* receive $\langle \textbf{data}, m, x \rangle$ (на шаге 1), send $\langle \textbf{ack}, x, y_2 \rangle$
7. *NCP A :* receive $\langle \textbf{ack}, x, y_2 \rangle$, reply $\langle \textbf{nocon}, x, y_2 \rangle$
8. *NCP B :* receive $\langle \textbf{nocon}, x, y_2 \rangle$ в ответ на $\langle \textbf{ack}, x, y_2 \rangle$, deliver $m$, close

# Задача надежного обмена информацией

## Диалог с четырьмя сообщениями.

1. NCP $A$ : send $\langle \mathbf{data}, m, x \rangle$
2. NCP $B$ : receive $\langle \mathbf{data}, m, x \rangle$, send $\langle \mathbf{open}, m, x, y \rangle$
3. NCP $A$ : receive $\langle \mathbf{open}, m, x, y \rangle$, send $\langle \mathbf{agree}, m, x, y \rangle$
4. NCP $B$ : receive $\langle \mathbf{agree}, m, x, y \rangle$, deliver $m$, send $\langle \mathbf{ack}, x, y \rangle$, close
5. NCP $A$ : receive $\langle \mathbf{ack}, x, y \rangle$, notify, close

# Задачи для самостоятельного решения

## Задача 1.

Покажите, что в протоколе с 4 сообщениями возможно дублирование или потеря сообщений ввиду того, что NCP $A$ вынуждена считаться с возможностью выхода из строя NCP $B$. В этом случае дублирование или потеря сообщений возникает даже в том случае, если NCP $B$ в действительности из строя не выходит.

# Задачи для самостоятельного решения

## Задача 2.

Постройте протокол с двумя сообщениями, который никогда не допускает потери сообщений (хотя может дублировать сообщения).

# Задачи для самостоятельного решения

## Задача 2.

Постройте протокол с двумя сообщениями, который никогда не допускает потери сообщений (хотя может дублировать сообщения).

Подсказка: воспользуйтесь идентификационными номерами сообщений, которые были введены в исправленной версии диалога с тремя сообщениями.

# Задачи для самостоятельного решения

## Задача 2.

Постройте протокол с двумя сообщениями, который никогда не допускает потери сообщений (хотя может дублировать сообщения).

Подсказка: воспользуйтесь идентификационными номерами сообщений, которые были введены в исправленной версии диалога с тремя сообщениями.

ДОКАЖИТЕ, что построенный протокол действительно не теряет ни одного сообщения.

# Задачи для самостоятельного решения

## Задача 3.

Постройте протокол с пятью сообщениями, который предотвращает потерю информации и допускает дублирование только тогда, когда одна из NCP действительно выходит из строя.

КОНЕЦ ЛЕКЦИИ 1.