# Летняя школа по параллельному программированию 2016

Проект «Разработка системы параллельного программирования x-project»

Библиотека исполнительной системы

(runtime support system)

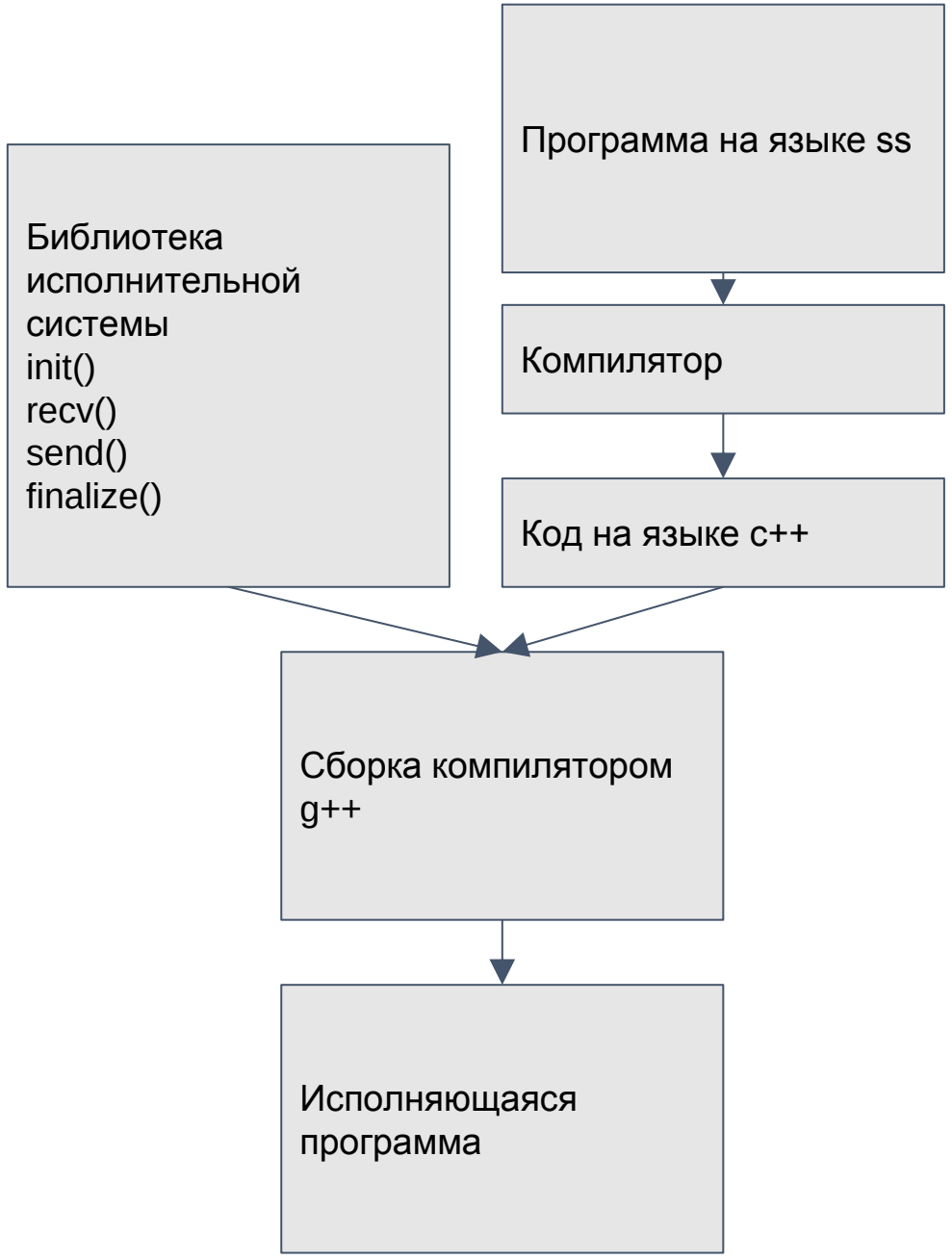
Исполнители: Дроздов Э.К, Свильпова А.Д.; ФПМИ, 1 курс.

Руководитель проекта: Городничев М.А., Киреев С.Е.

Дата доклада: 15.07.2016

# Цель работы

Разработка библиотеки функций для обеспечения поддержки выполнения программ в системе программирования x-project



# Постановка задачи: требования

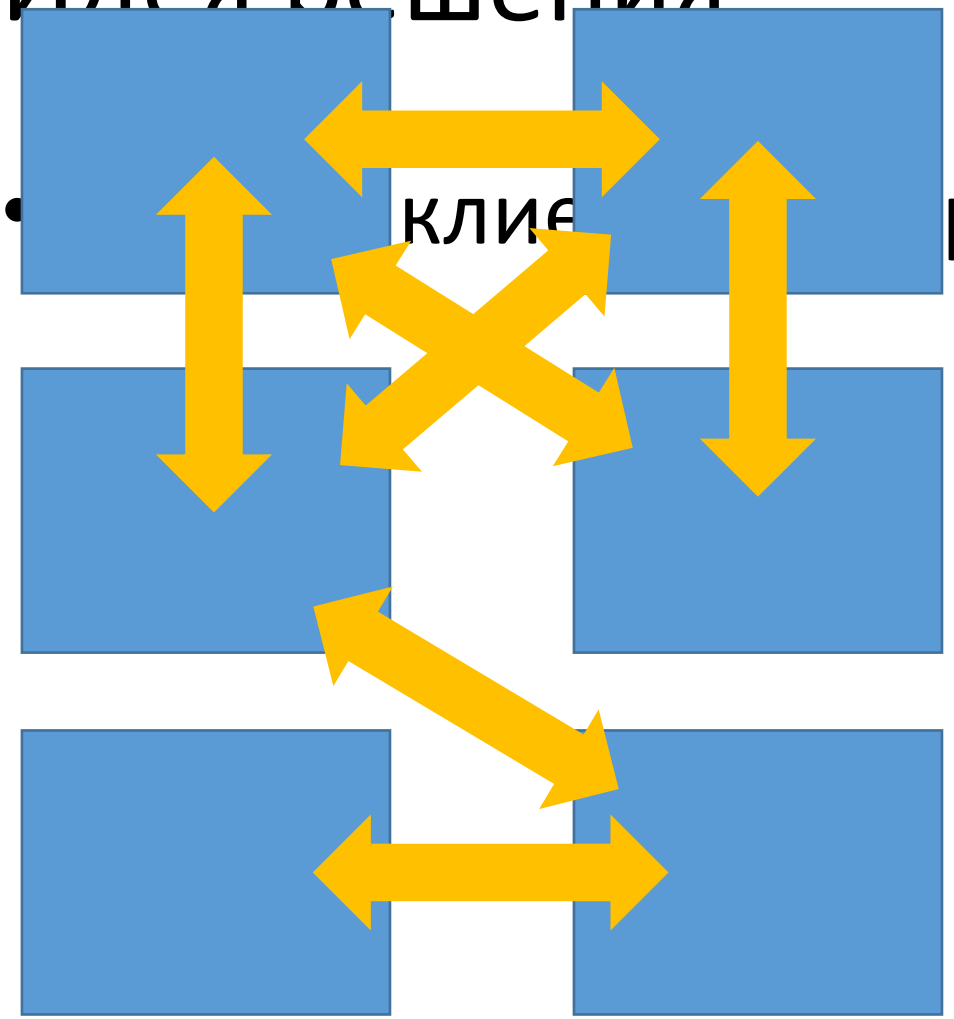
- Обеспечить отправку и получение сообщений процессами на основе локальной адресации процессов.

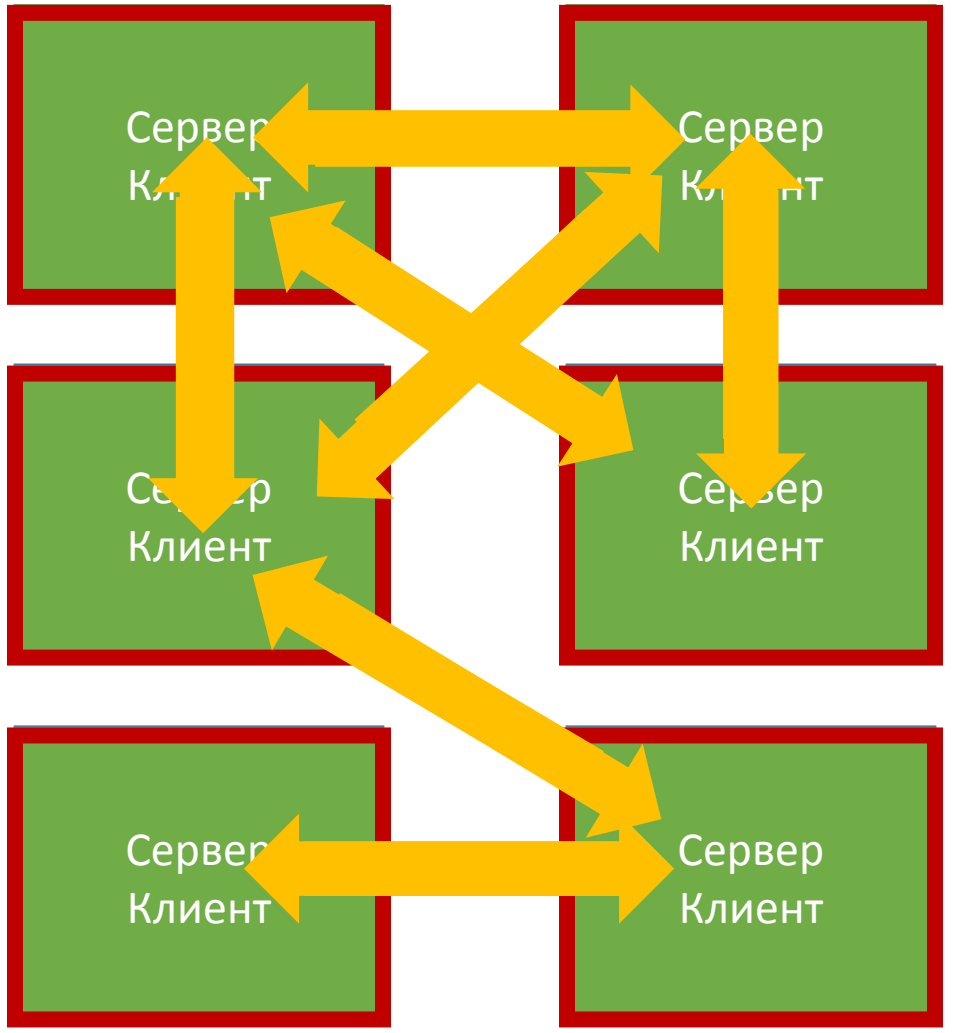
Нефункциональные требования:

- Простота интерфейса библиотеки
- Приём сообщений в многопоточном режиме
- Устранение простоев при отправке и приёме сообщений: за счет асинхронных коммуникаций

# Идея решения

- клие р





# Идея решения: набор функций библиотеки

- Установка соединения между процессами
  - Init()
- Обмен сообщениями
  - Send( кому, что)
  - Recv( от кого, что)
- Завершение работы
  - End()
- Ограничения

# init(): установка соединения.

- Создание клиента и сервера на каждом процессе
- Установка соединений между процессами-соседями

# send(), recv(): обмен сообщениями.

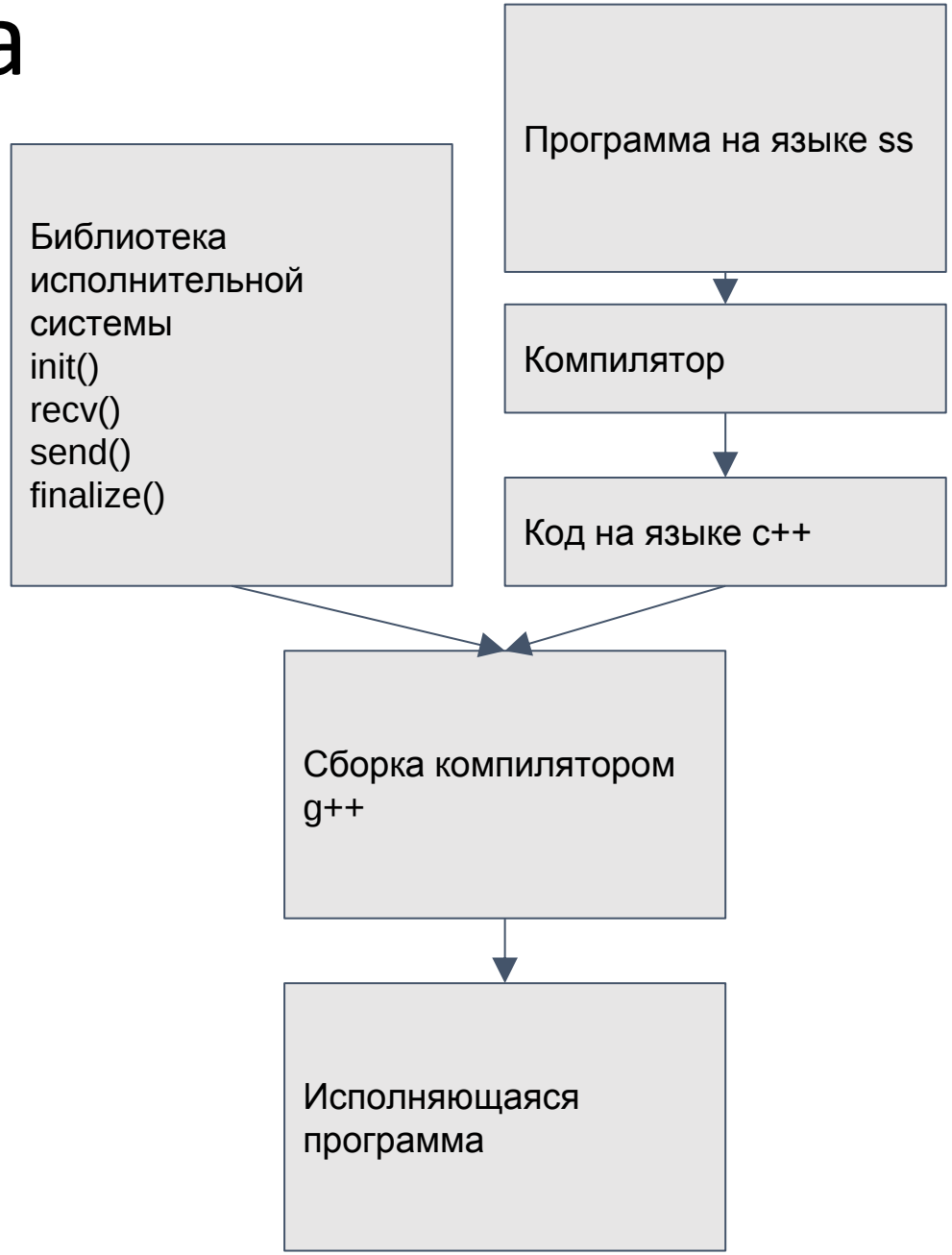
- Посылка/ приём сообщения
- Решение проблемы локальных адресов и простоев при посылке сообщения
  - Буфер и очереди сообщений
  - Таблица адресов

# finalize() : завершение работы.

- Закрытие сокетов
- Освобождение памяти

# Реализация: структура кода и средства реализации

Язык C
/lib
/src
/include
Makefile



# Реализация: ограничения

- Размер сообщения фиксирован
  - Размер буфера
- Количество принятых сообщений
  - Переполнение буфера
- Количество подключений к серверу
  - Количество потоков
  - Переполнение таблицы адресов

# Тестирование

- Было проведено модульное тестирование функций, предоставляемых библиотекой ИС.

# Заключение: результаты

Разработана библиотека для поддержки исполнения программ в системе x-project

- Простые функции для обмена сообщениями между процессами
- Программирование осуществляется в терминах логических локальных адресов, а процессы отображаются на ресурсы выч. системы автоматически

# Заключение: перспективы разработки

- Оптимизация
  - Уменьшение накладных расходов
  - Ускорение посредством объединений процессов в группы
- Расширение функциональности
  - Размер сообщений,
  - Обеспечить возможность отображать несколько логических процессов задачи на один вычислительный узел, обеспечить динамическое порождение, уничтожение, перераспределение логических процессов

# Спасибо за внимание!

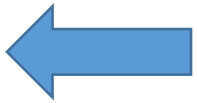
*Дополнительные слайды*

- Клиент
- Сервер
- Посылка сообщения
- Приём сообщения
- Буфер
- Таблица адресов
- Race condition

# Клиент

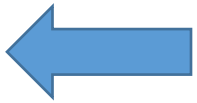
- Для каждого «соседа»
  - Создание сокета
  - Подключение к серверу
  - Регистрация сокета в таблице адресов

- Отсутствие сервера
  - Ожидание



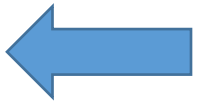
# Сервер

- Создание сокета
  - Через какой порт открывать сокет?
- Прослушивание сокета
- Создание потока с обработчиком для каждого подключения
  - Ожидание сообщения
  - Приём сообщения и отправка его в буфер



# Посылка сообщения

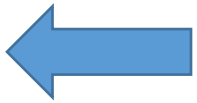
- Локальность адресов
  - Выяснить по таблице адресов, какой сокет подключен к нужному серверу
  - Прикрепление идентификатора к сообщению
- Переполнение буфера
  - Ожидание



# Приём сообщения

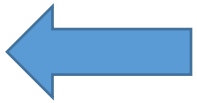
- Локальность адресов
  - Выяснение отправителя по таблице адресов
  - Обращение к соответствующей очереди сообщений

- Отсутствие сообщения
  - Ожидание сообщения



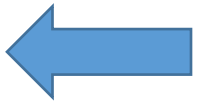
# Буфер

- Структура буфера
  - Упрощённый локальный адрес процесса-соседа
  - Очередь сообщений
- Создание очереди сообщений для каждого «соседа»
- Смысл
  - Разрешение вопроса о местонахождении сообщений от конкретного отправителя
  - Нет необходимости ожидать при отправке



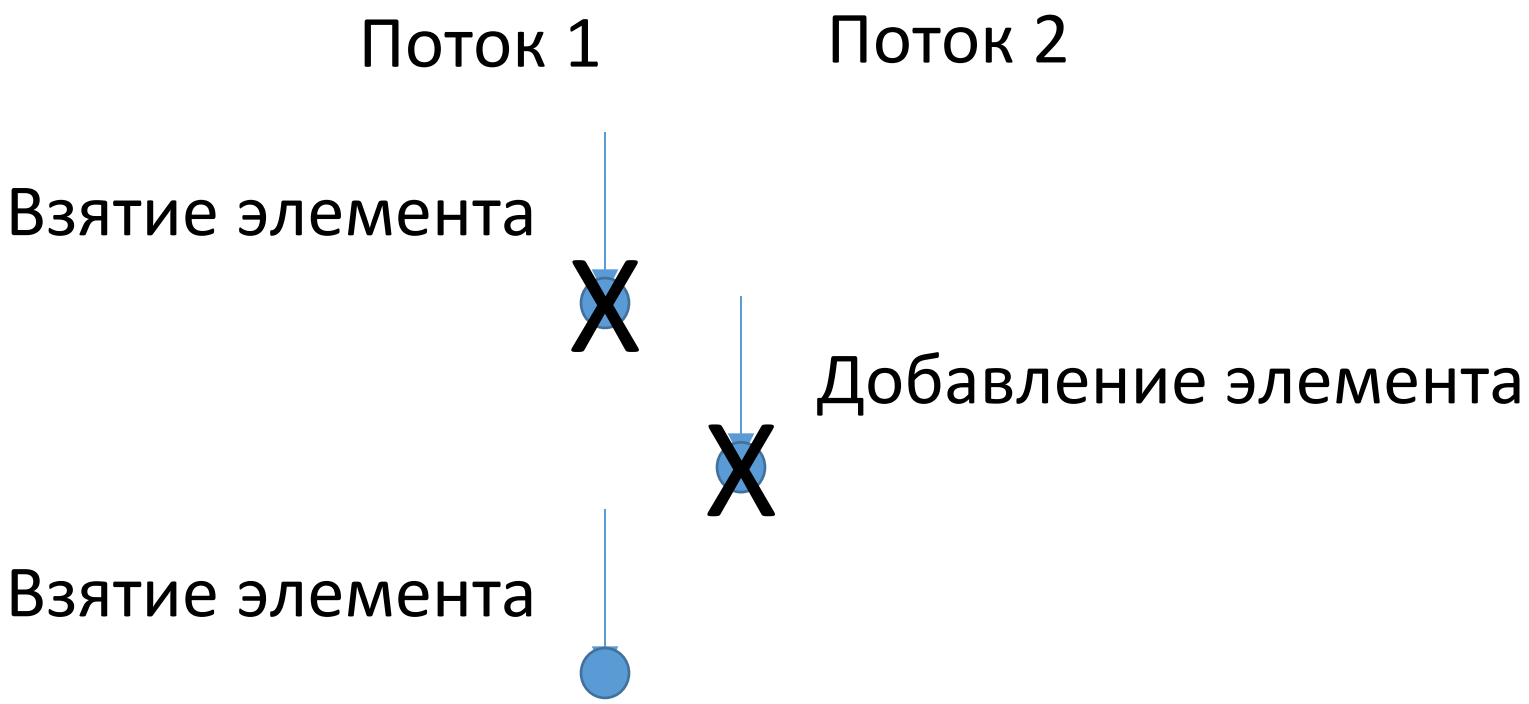
# Таблица адресов

- Структура таблицы
  - Внешний адрес сервера
  - Внутренний адрес сервера
  - Сокет, подключённый к серверу
  - Какой локальный адрес на сервере для меня
- Смысл
  - Решение вопросов с локальностью адресов при приёме и отправке сообщений



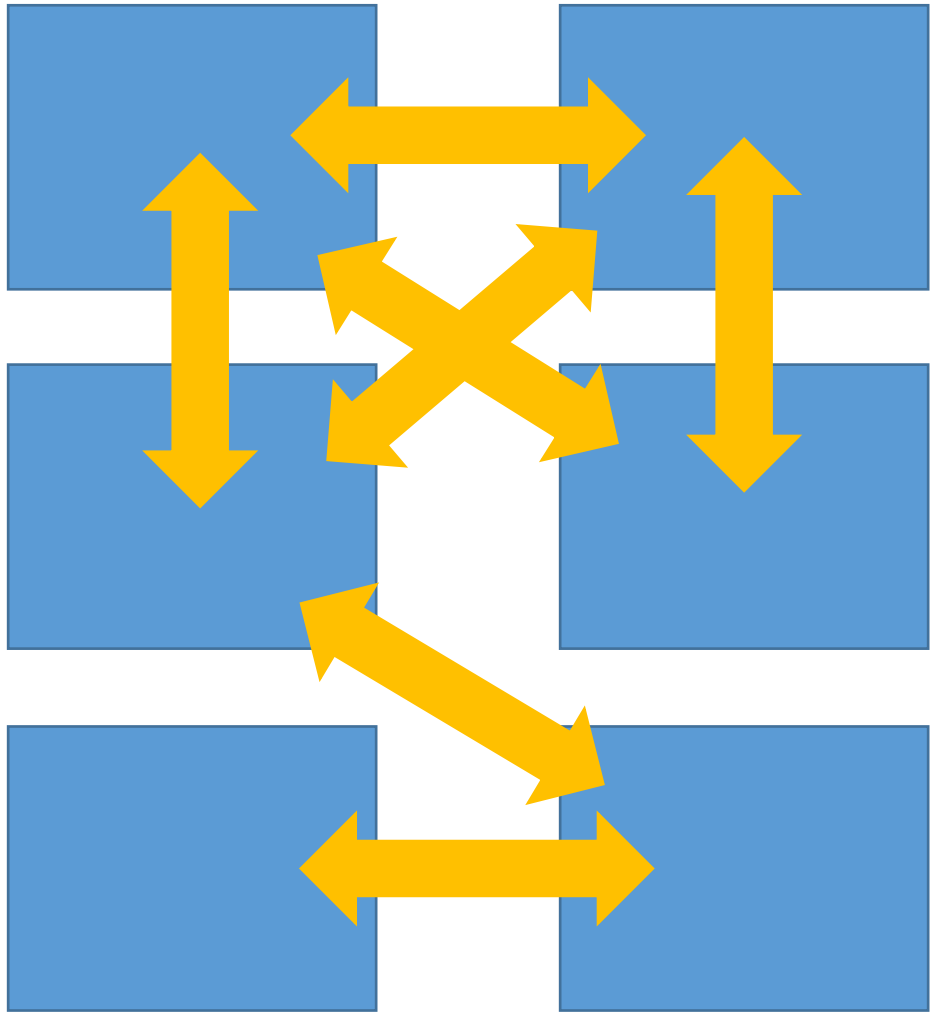
# Очередь сообщений и race condition

- Одновременное обращение к очереди разными потоками (server и recv)
  - Функции Mutex_lock и Mutex_unlock



# Идея решения: init()

- Модель клиент-сервер



# Идея решения

- Модель клиент-сервер