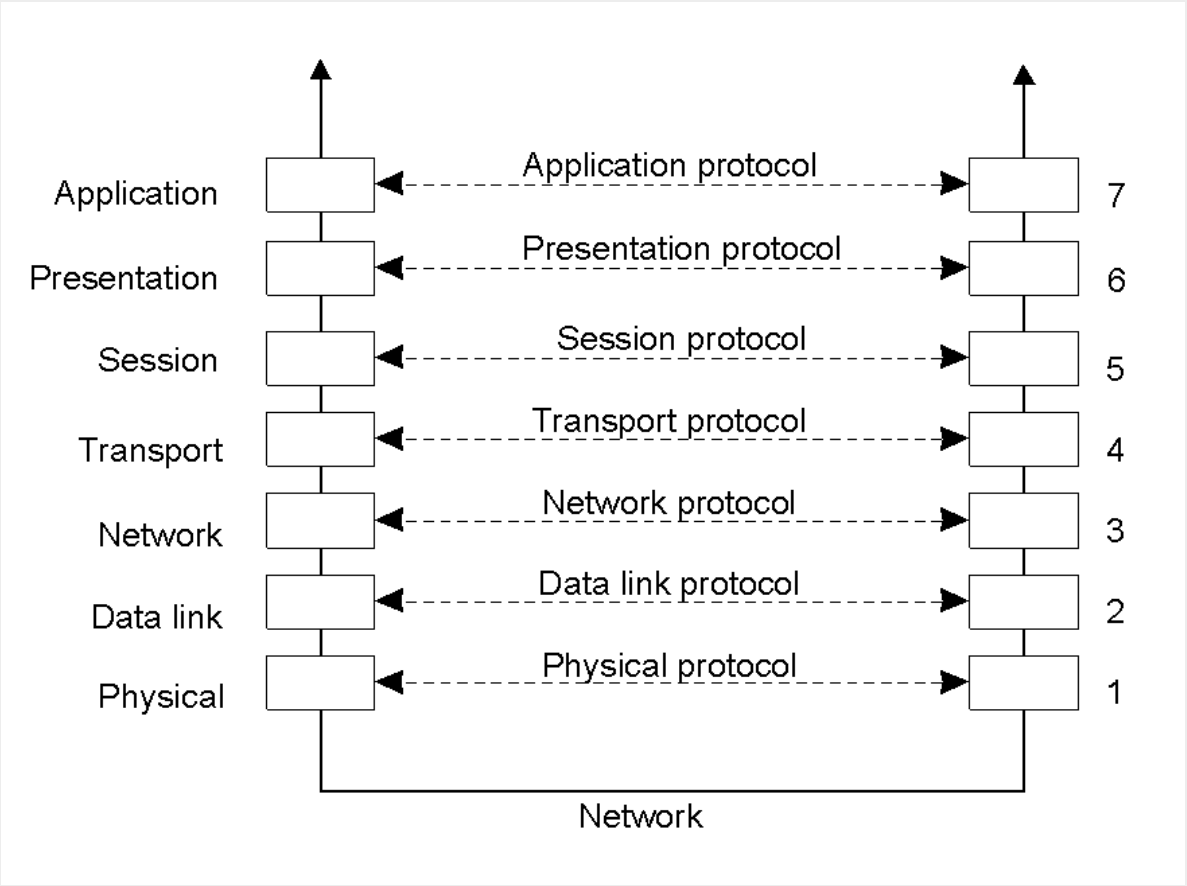
Распределенные системы

Протоколы

Уровни протоколов

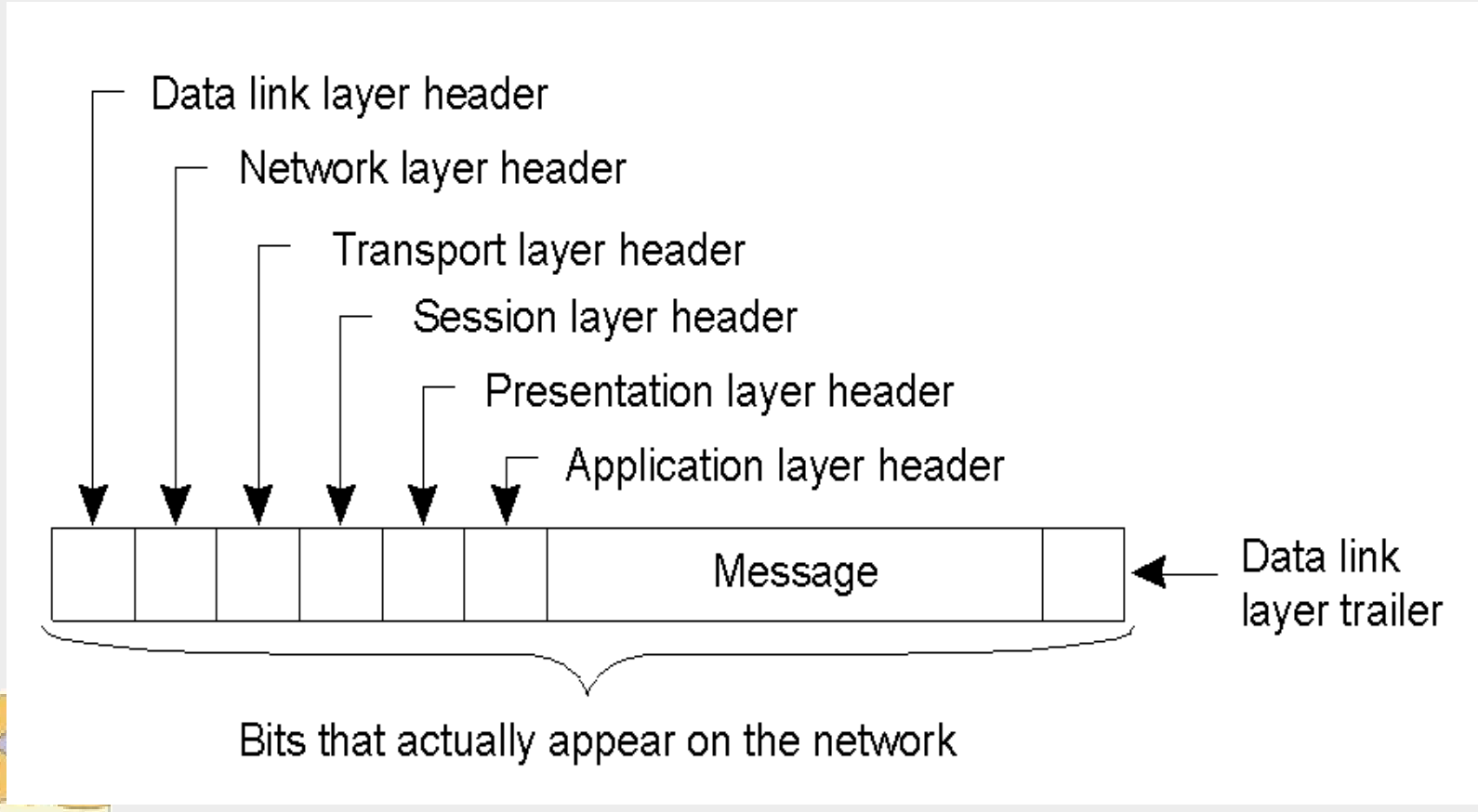
- Модель ISO OSI



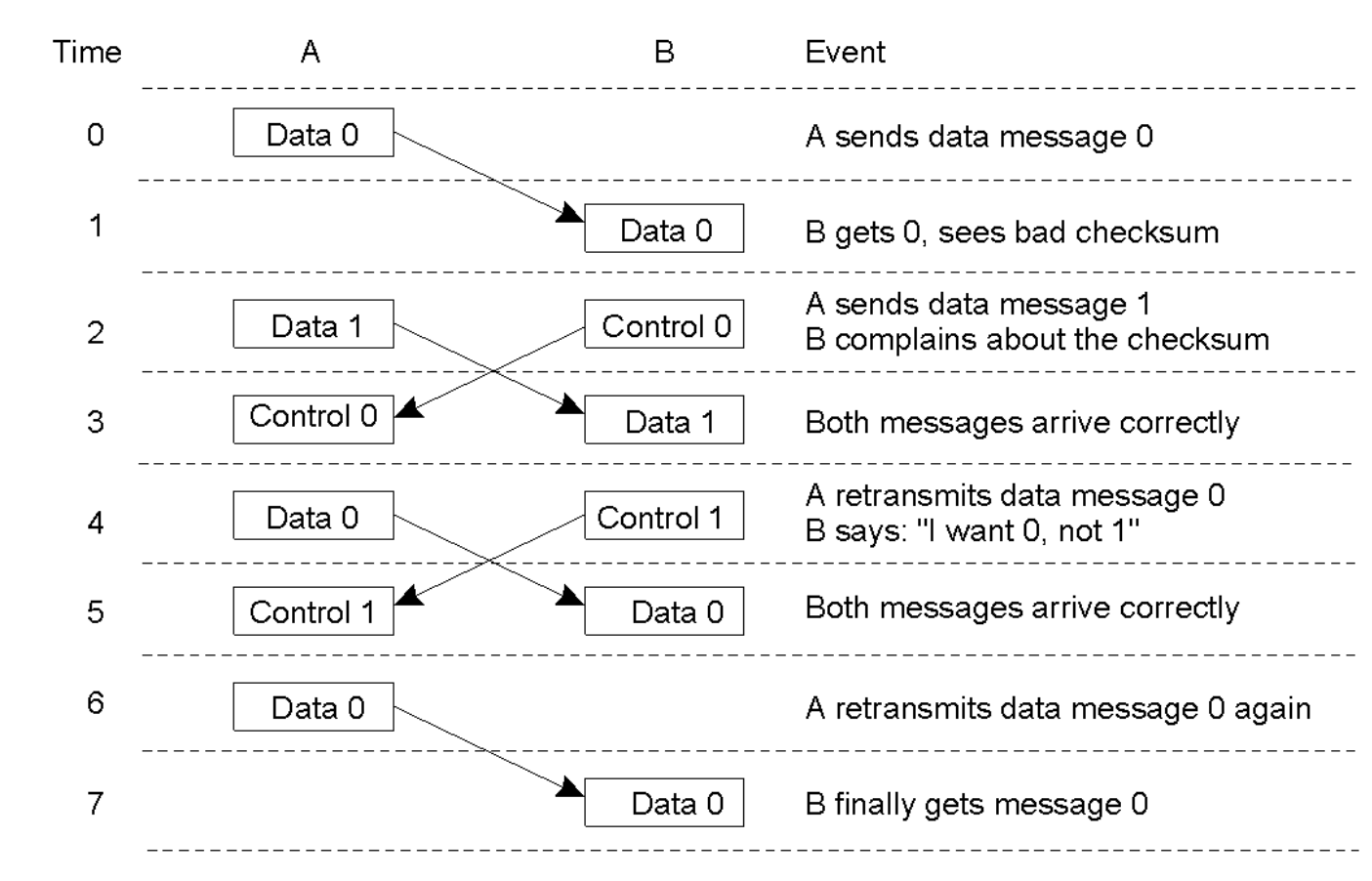
Модели взаимодействия

- Удаленный вызов процедур
- Удаленный вызов методов
- Использование очереди сообщений
- Потоки данных

Уровни протоколов

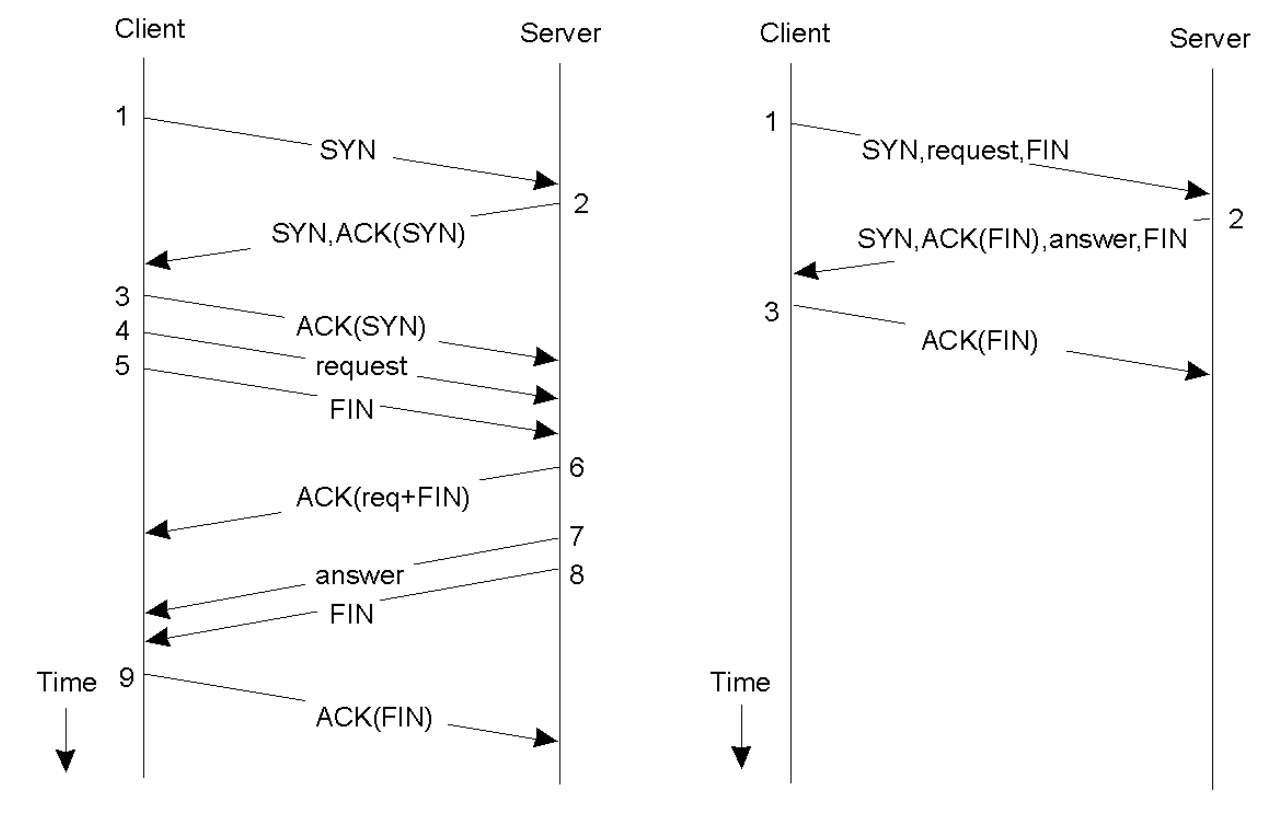


Прикладной уровень



- Взаимодействие между получателем и передатчиком

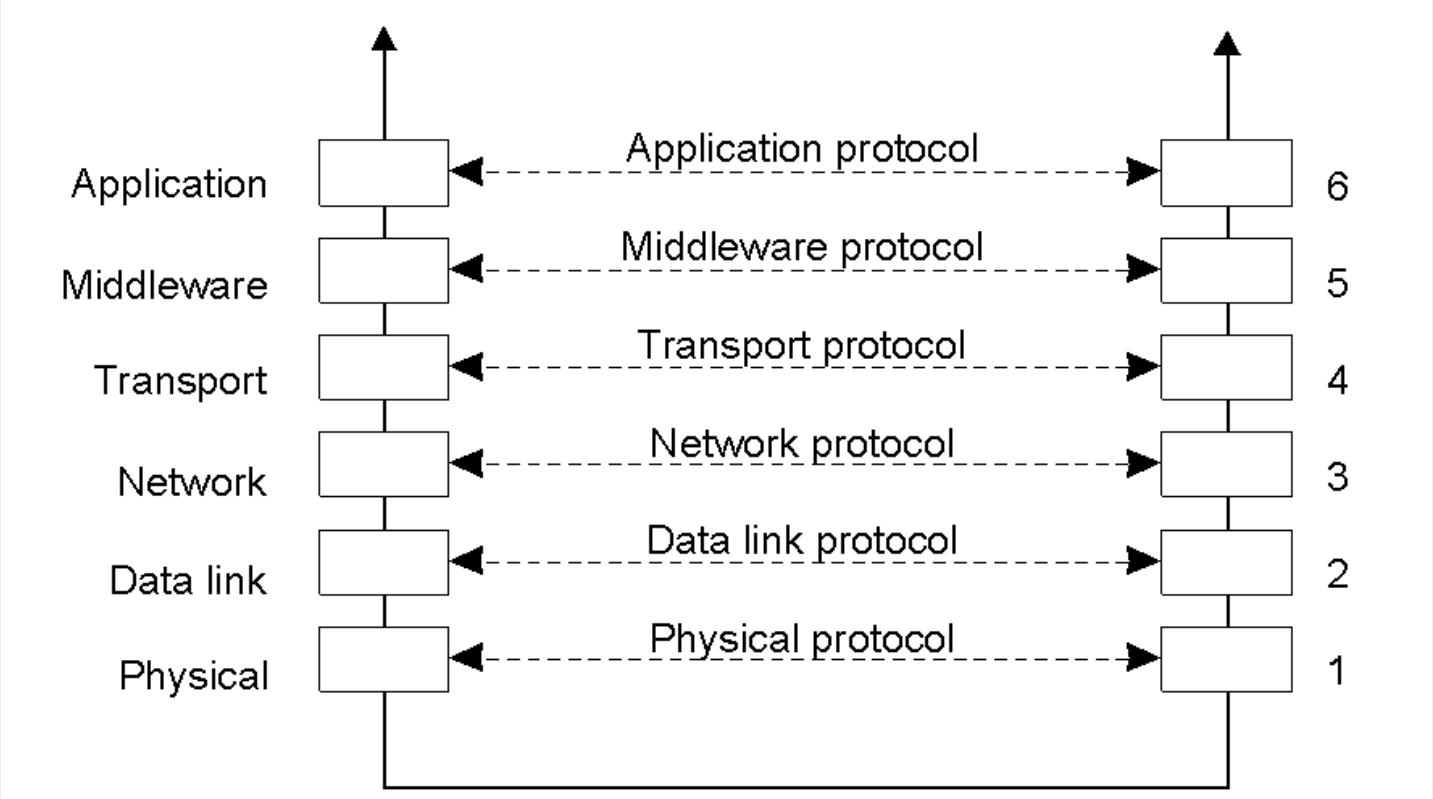
Клиент-серверное взаимодействие по TCP



a) Обычный порядок выполнения операций TCP

b) Транзакционный T/TCP.

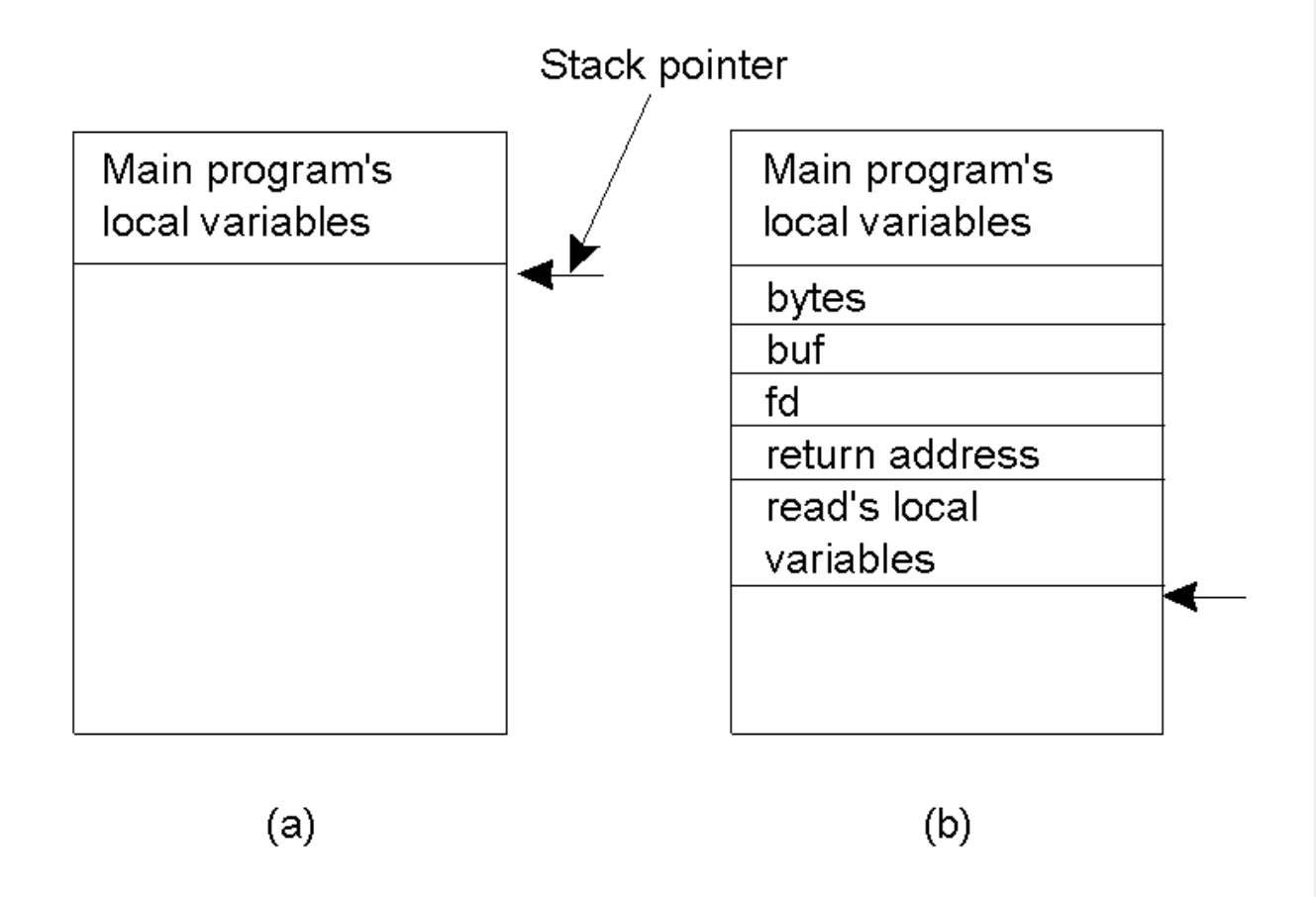
Протоколы промежуточного уровня



- Примеры: аутентификация, блокировки, транзакции

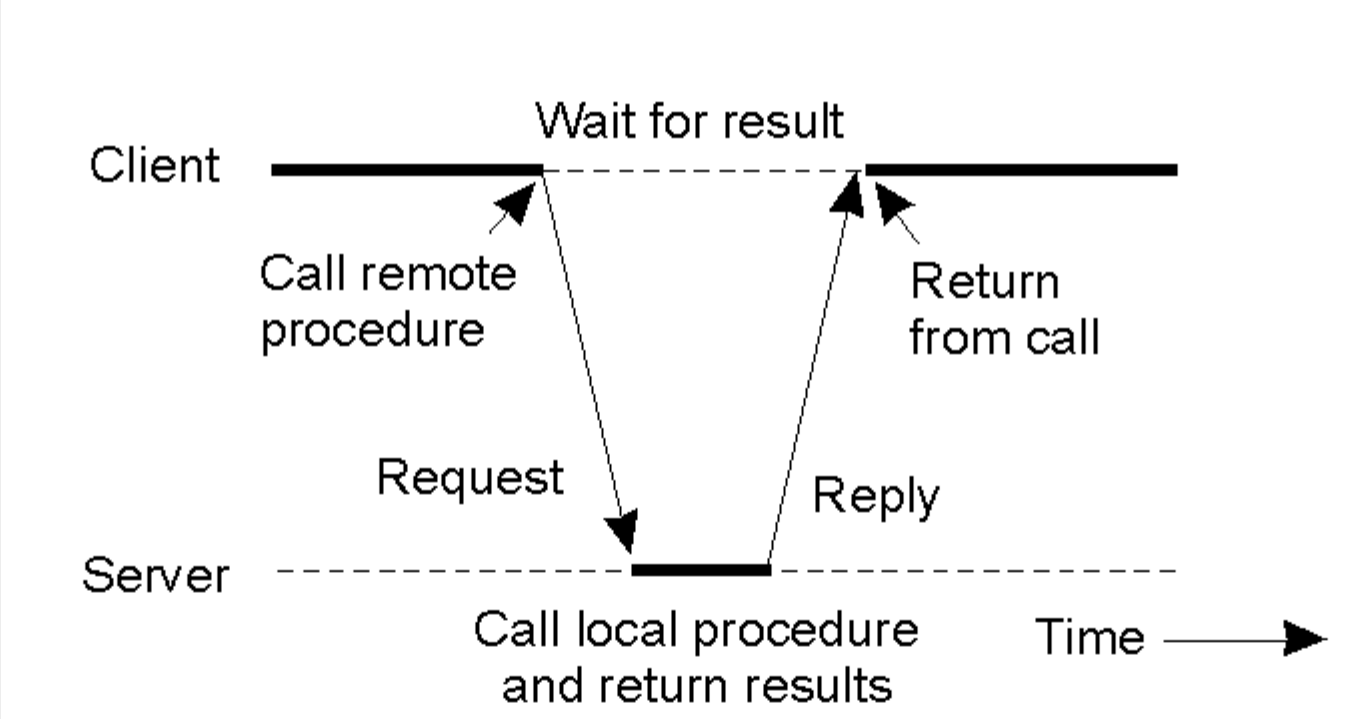
Удаленный вызов процедур

Передача параметров при вызове процедуры



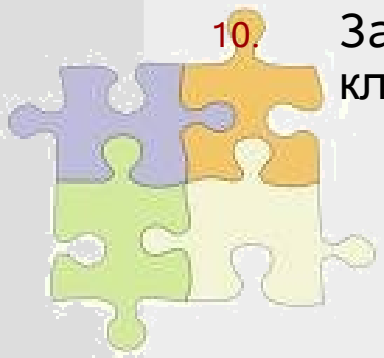
Stubs (Заглушки)

- Принцип взаимодействия между клиентом и сервером

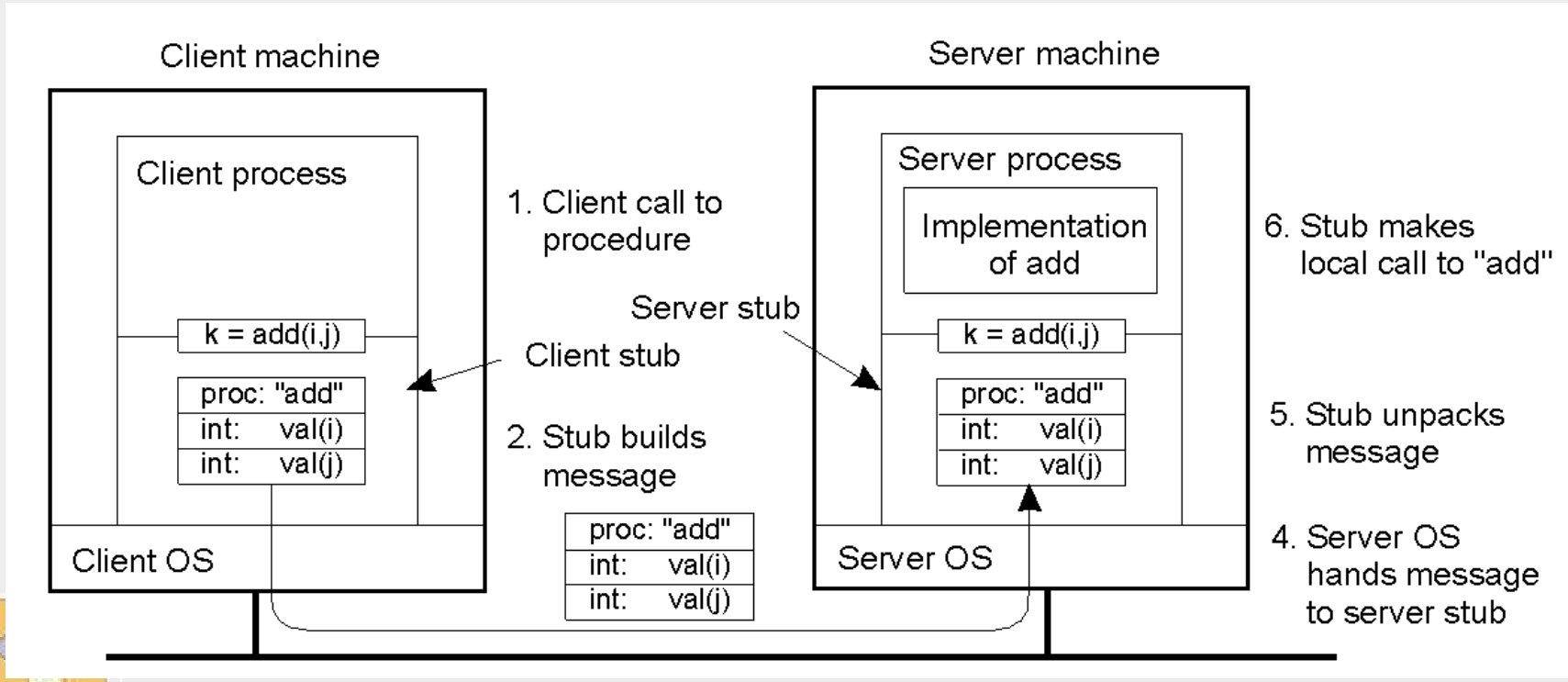


Шаги протокола RPC

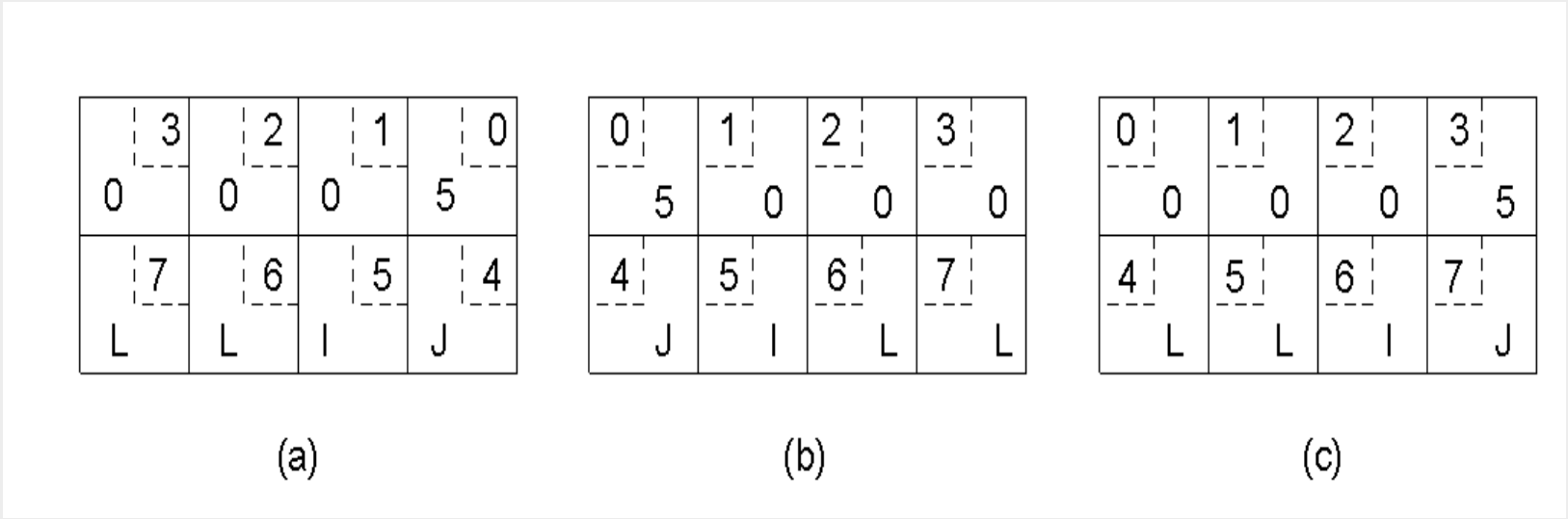
1. Клиент вызывает заглушку клиента как обычную процедуру
2. Заглушка клиента формирует сообщение и вызывает локальную ОС
3. Клиентская ОС передает сообщение для удаленной ОС
4. Удаленная ОС передает сообщение для заглушки сервера
5. Заглушка сервера принимает параметры и вызывает сервер
6. Сервер выполняет работу и возвращает результат заглушке сервера
7. Заглушка сервера формирует сообщение и вызывает ОС
8. Серверная ОС посылает сообщение клиентской ОС
9. Клиентская ОС передает сообщение клиентской заглушке
10. Заглушка клиента распаковывает результат и возвращает его клиенту



Передача параметров



Передача параметров и преобразование



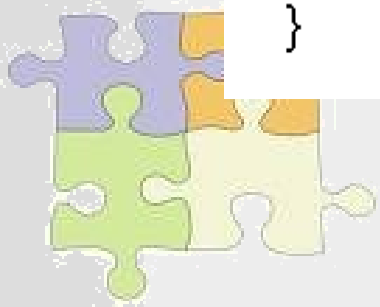
Вызов с параметрами 5, “JILL” между Intel Pentium и SPARC

Передача ссылок

Генерация заглушек и спецификация параметров

```
foobar( char x; float y; int z[5] )
{
   ....
}
```

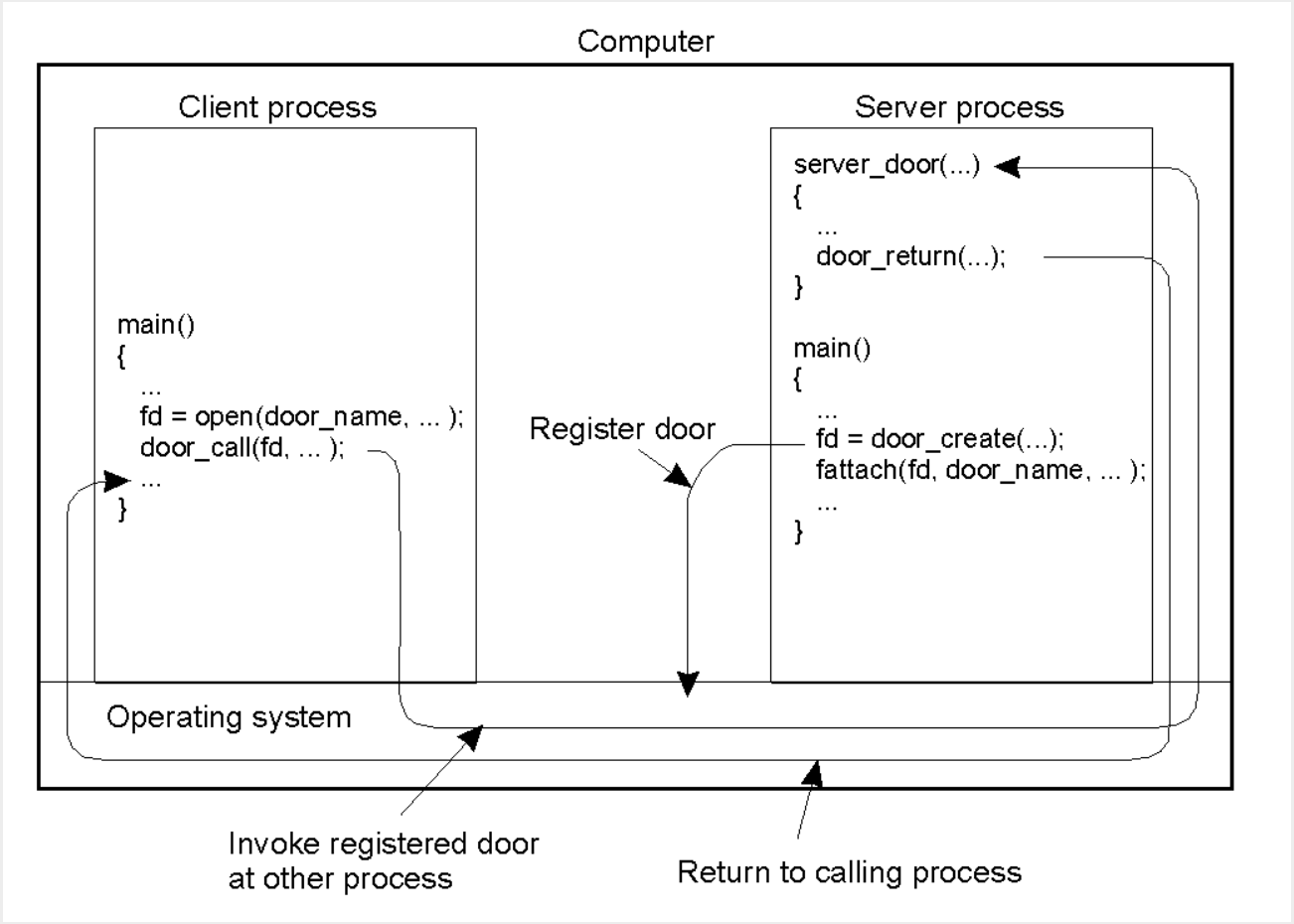
foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	



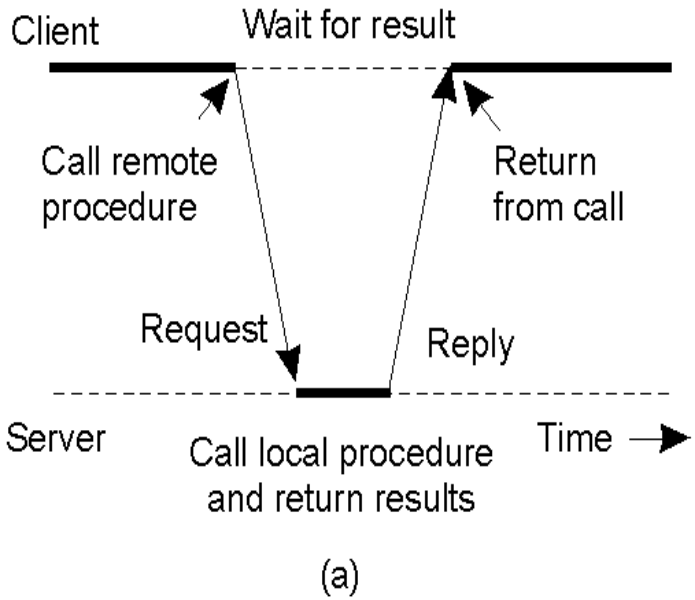
a) Процедура
b) Сообщение для передачи вызова

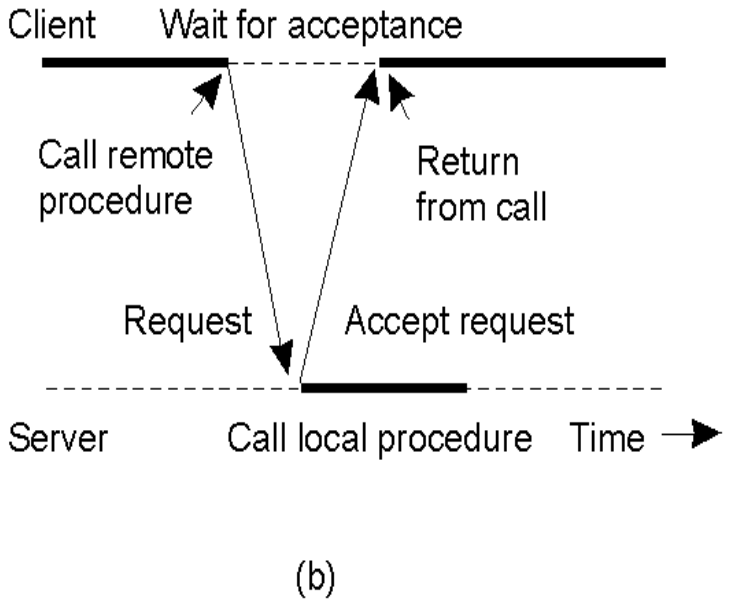
Входы

- Облегченный механизм RPC (IPC)



Асинхронный RPC



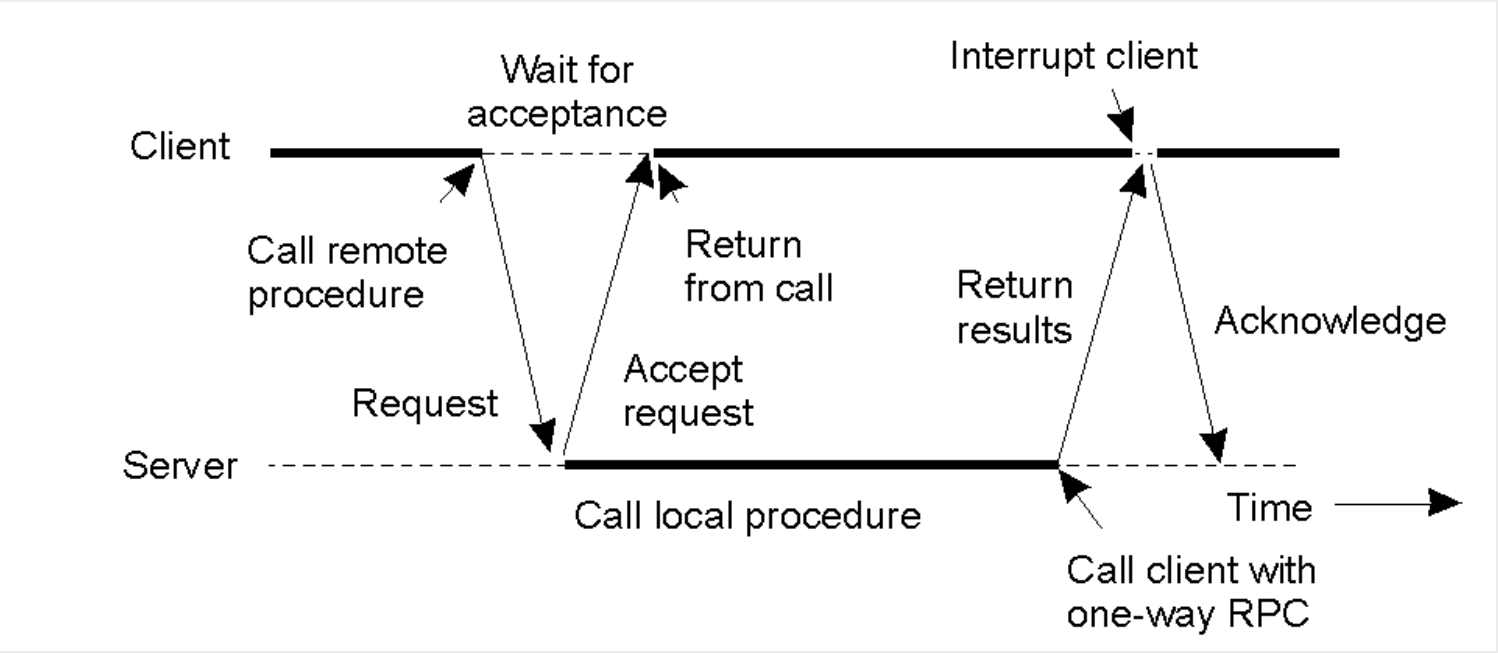


a) Синхронный вызов

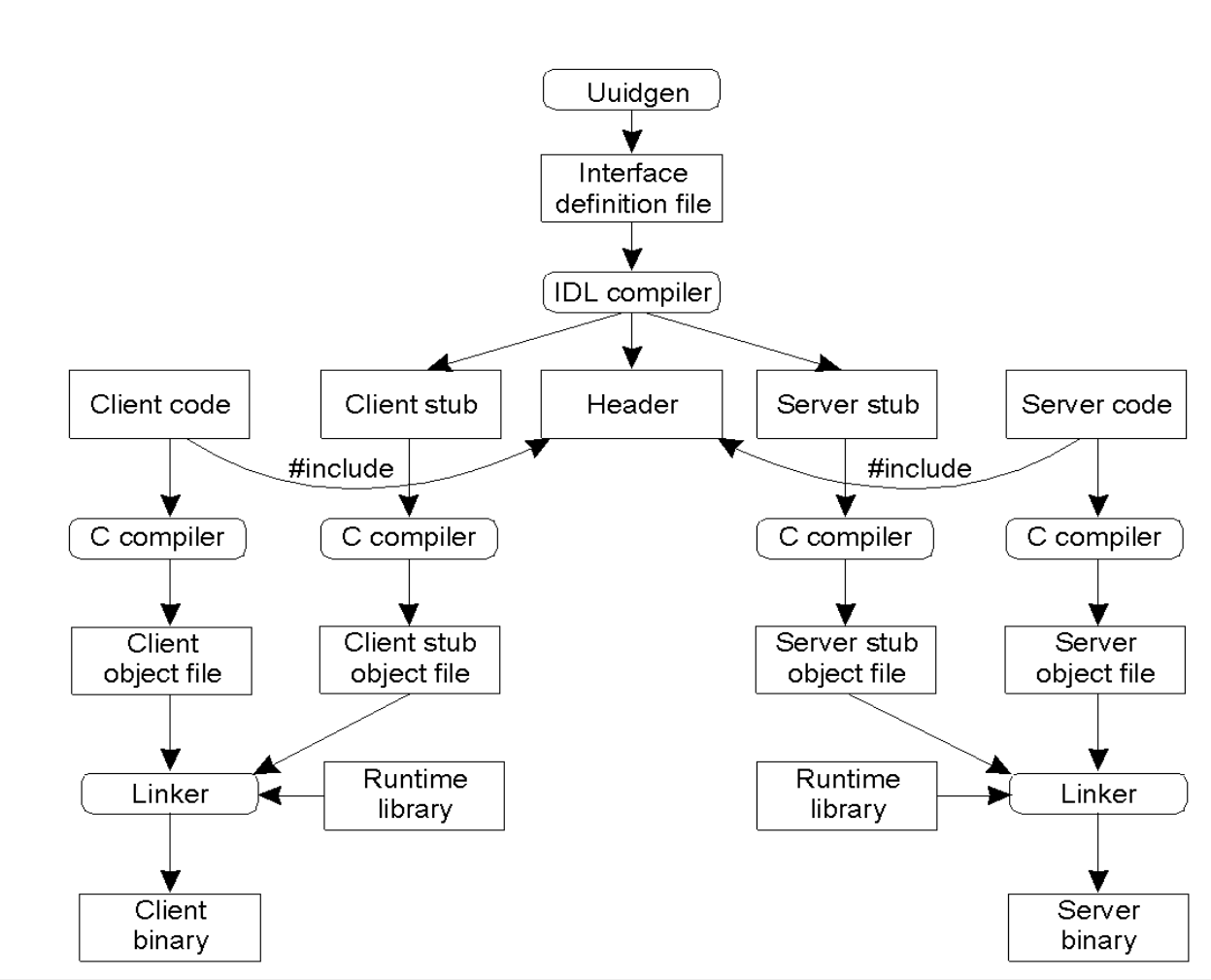
b) Асинхронный вызов

Асинхронный RPC

- Использование Callback функций



Написание клиента и сервера



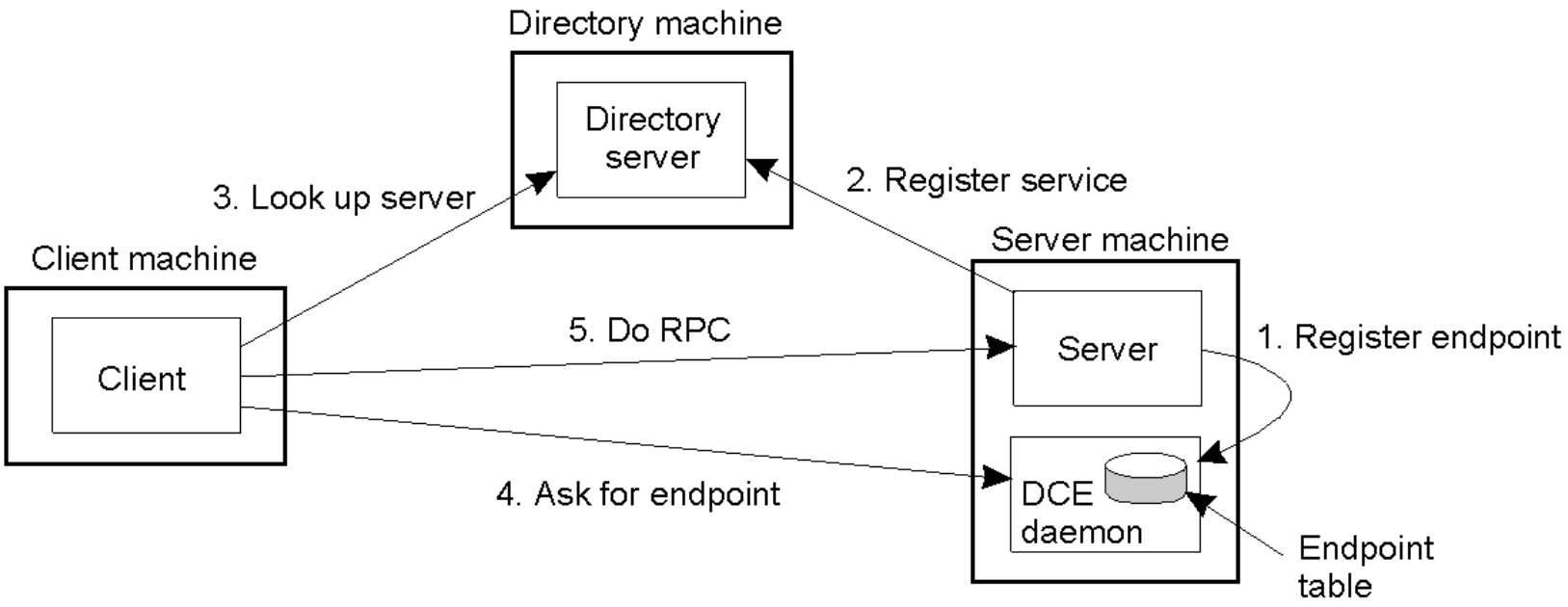
Пример IDL (DCE, CORBA)

```
module storage {
   interface textfile {
      void readln(
        inout short pos,
        out string line
      );
      void writeln(
        inout short pos,
        in string line
      );
      int get_pos();
   };
};
```

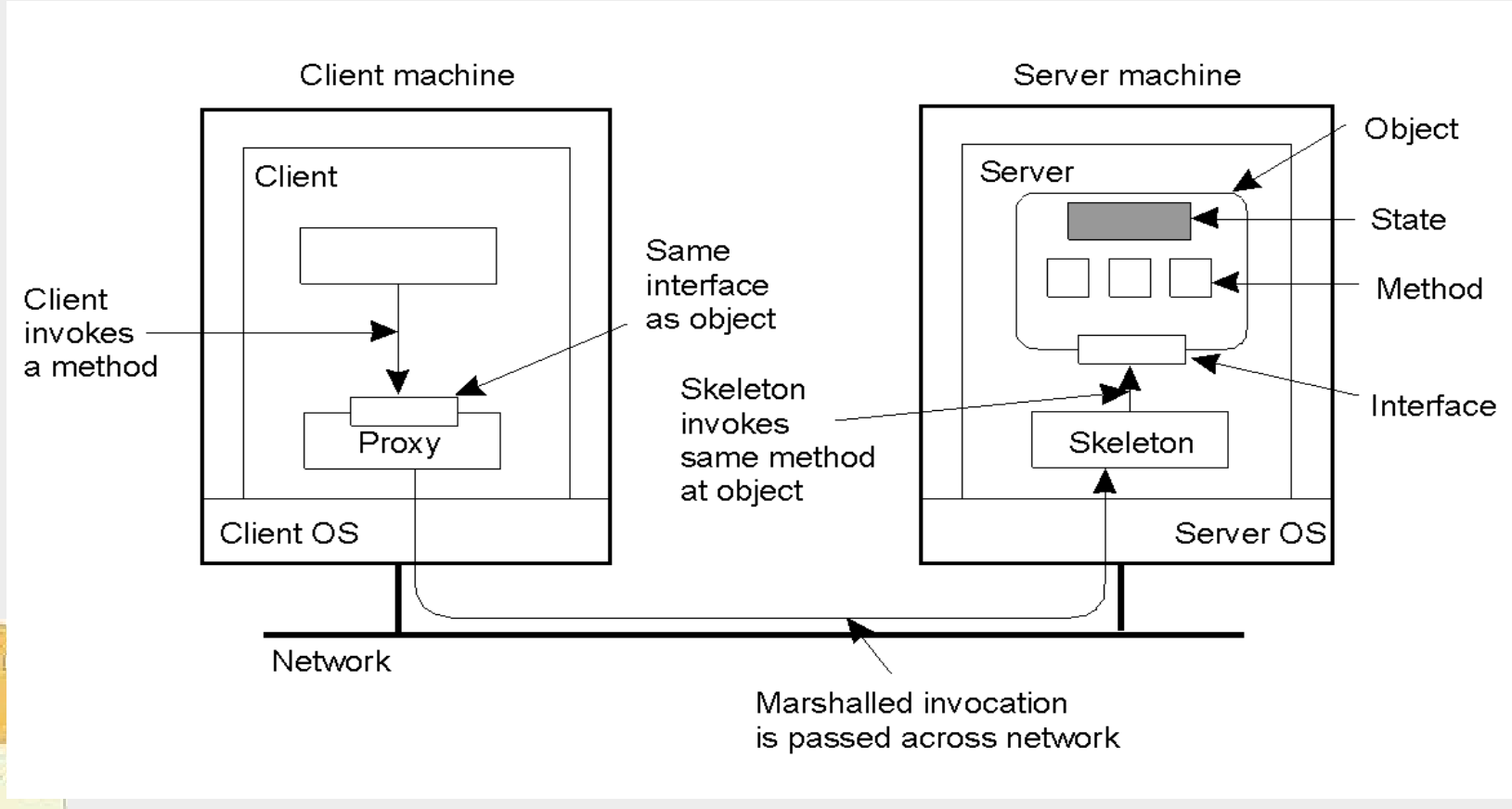
```
library storage {
   interface textfile {
      void readln(
        [in, out] short *pos,
        [out, string] char **line
      );
      void writeln(
        [in, out] short *pos,
        [in, string] char **line
      );
      int get_pos();
   };
};
```

Связывание клиента и сервера для DCE RPC

- Distributed Computing Environment, OSF

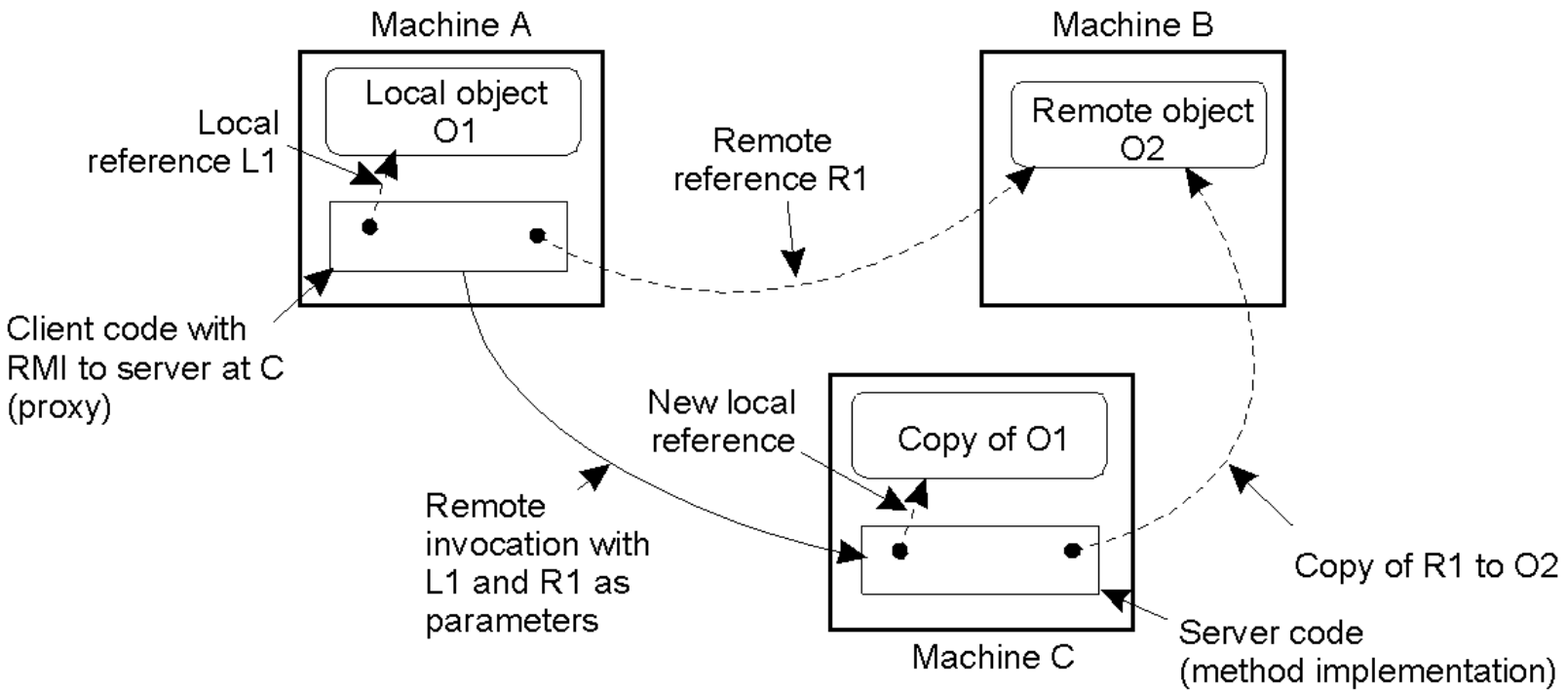


Распределенные объекты (DCE, RMI, J2EE, CORBA)



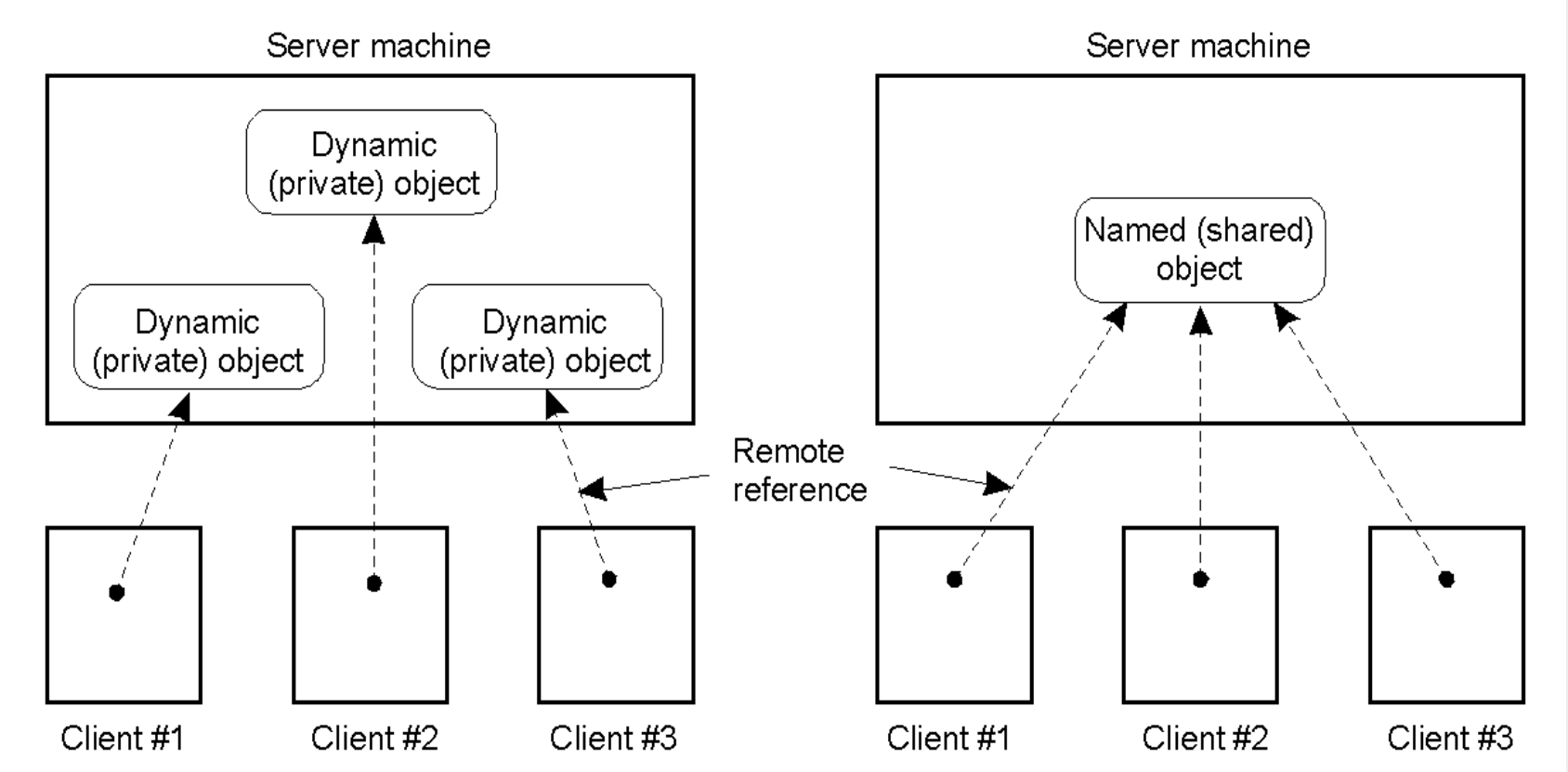
Сохраняемые и несохраняемые объекты

Передача параметров



- A вызывает метод объекта C с параметрами L1, R1
- Кодирование ссылки на объект в заглушке (адрес по сети)

Распределенная объектная модель DCE



- Динамическое обращение к объектам
- Модель именованных объектов

Пример IDL

```
union U switch (int) {
  case 1 : long x;
  case 2 :
  case 3 : string s;
  default: char c;
};
```

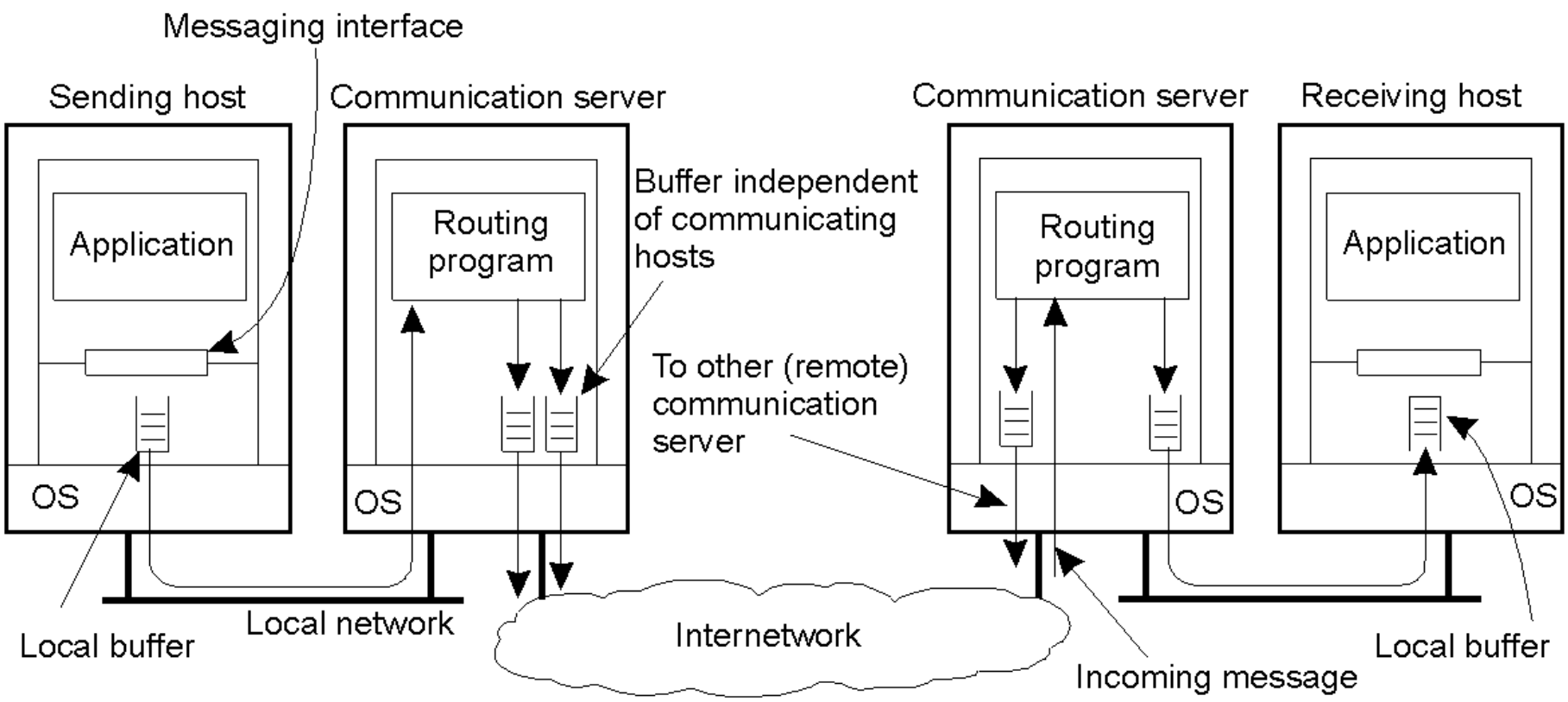
```
union U switch (int a) {
  case 1 : long x;
  case 2 :
  case 3 : string s;
  default: char c;
};
```

```
interface foo {
  void bar(
    [in] int len,
    [in, length_is(len)] float *addr
  );
  void xyz(
    [in, length_is(5)] short *addr
  );
};
```

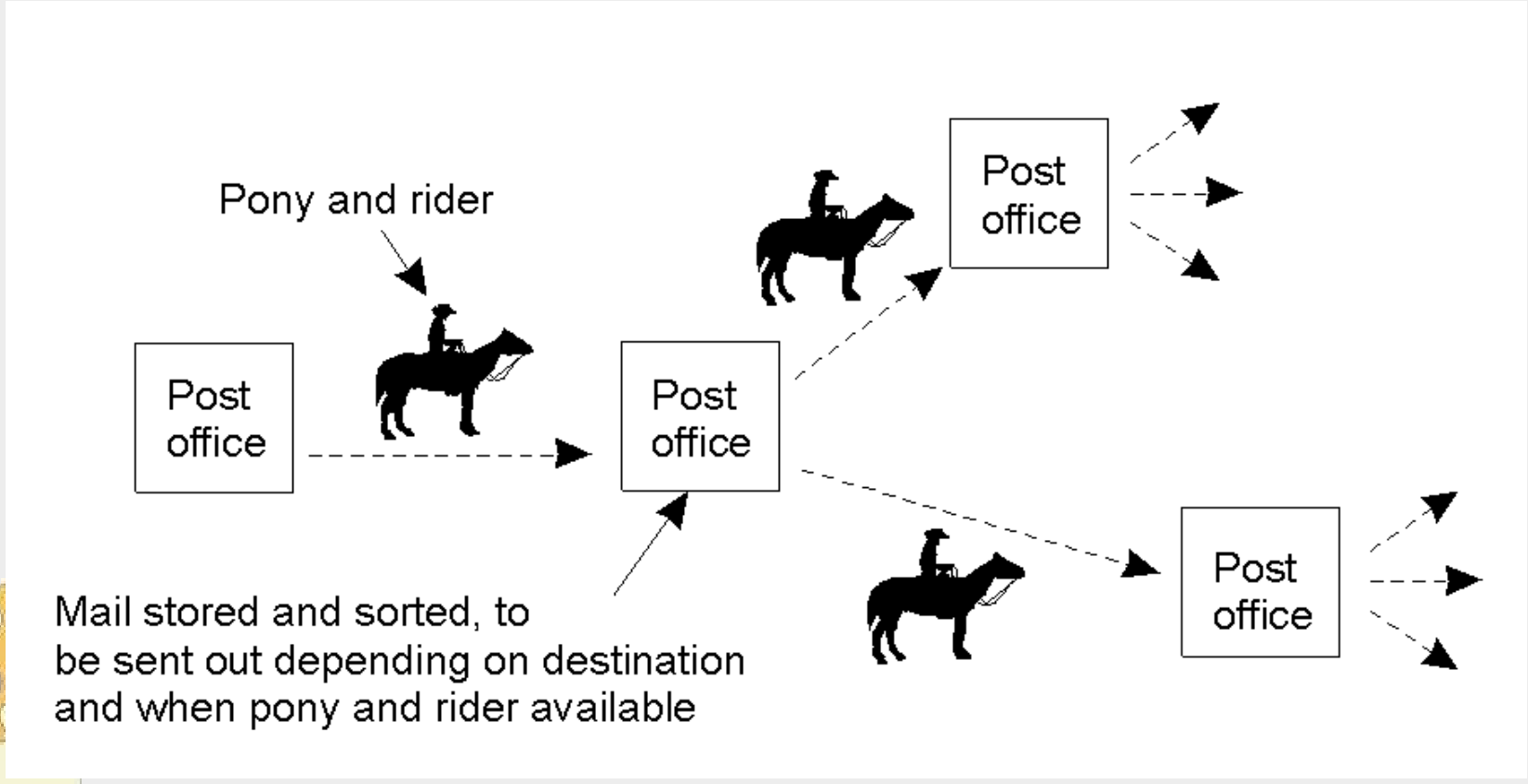
Известные реализации

- DCE
- Java RMI
- CORBA
- J2EE
- COM/DCOM/ActiveX

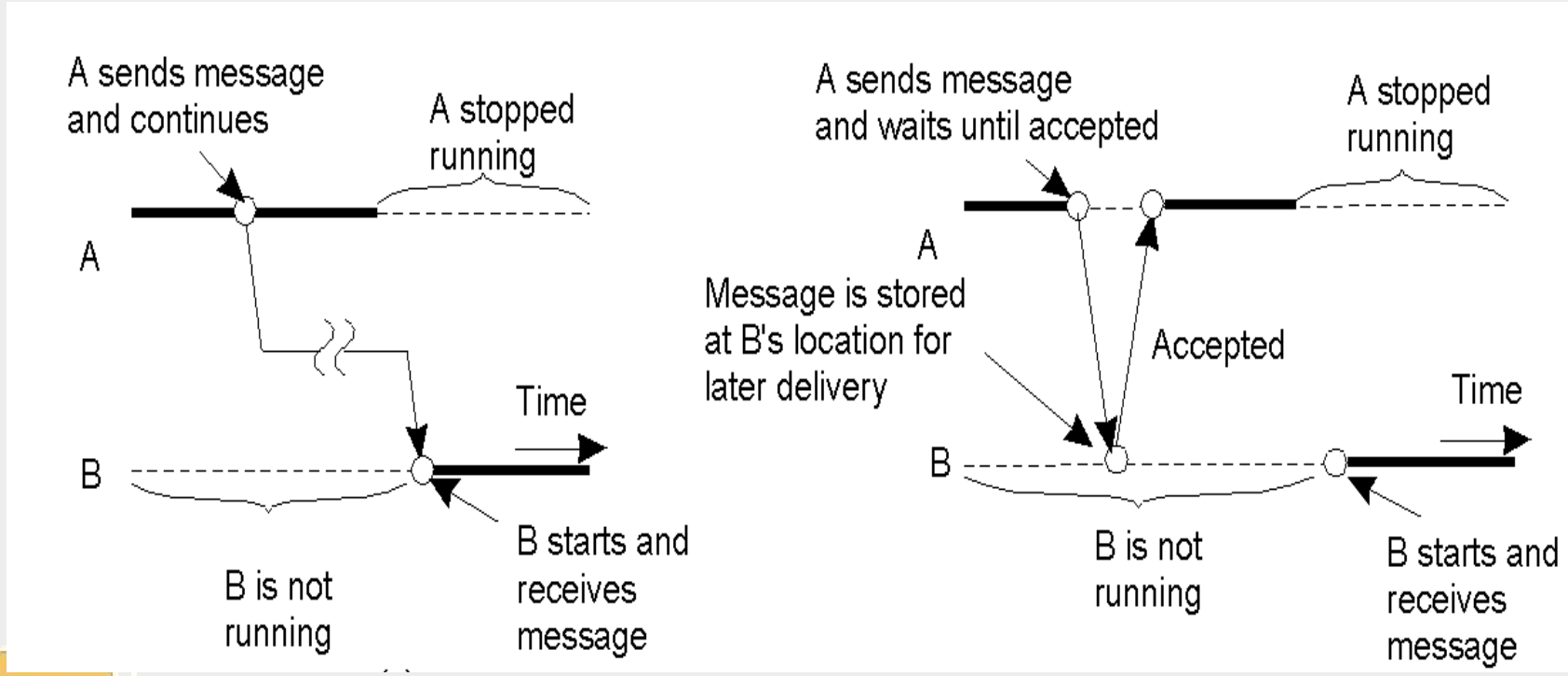
Сохранность и синхронность во взаимодействиях



Пример. Передача почты



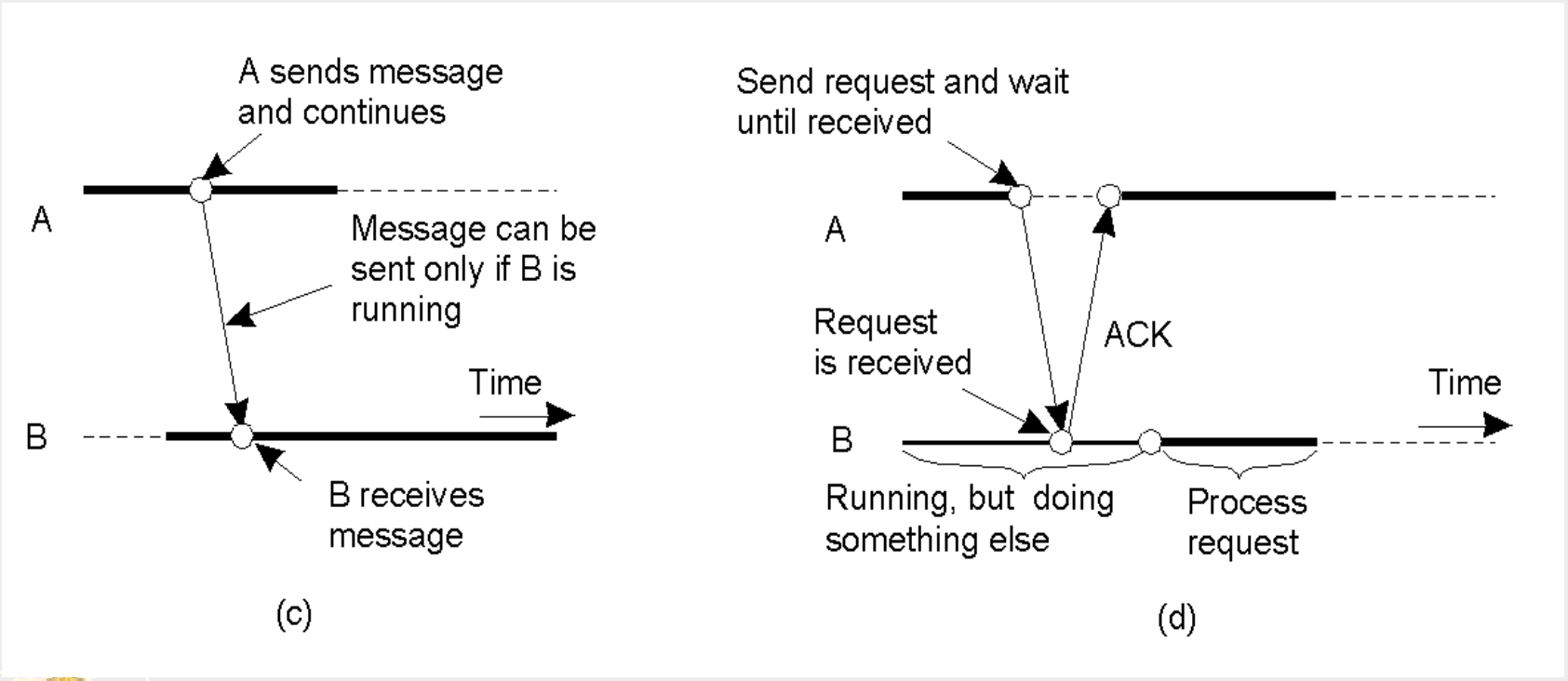
Сохранность и синхронность во взаимодействиях



a) Сохранное асинхронное взаимодействие

b) Сохранное синхронное взаимодействие

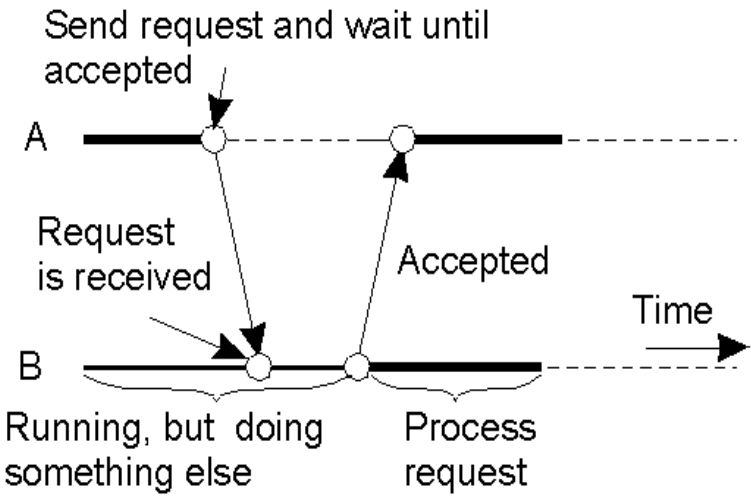
Сохранность и синхронность во взаимодействиях

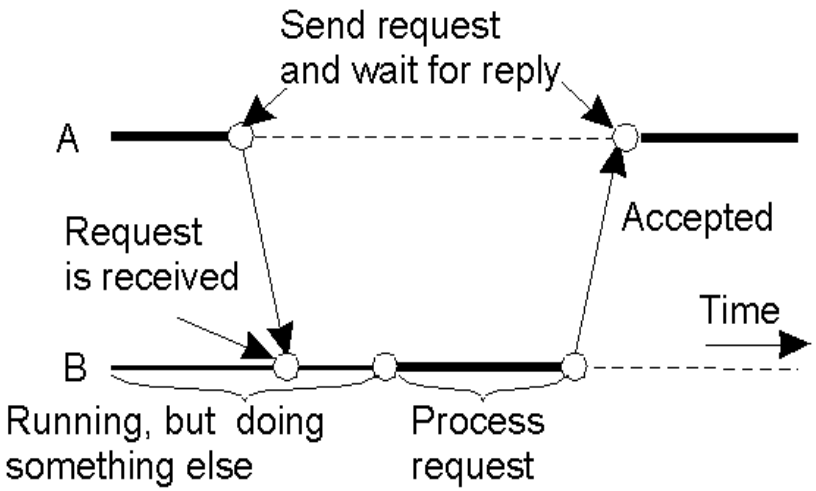


c) Нерезидентное асинхронное взаимодействие

d) Нерезидентное асинхронное взаимодействие с синхронизацией по доставке

Сохранность и синхронность во взаимодействиях





e) Синхронное нерезидентное взаимодействие с синхронизацией по получению

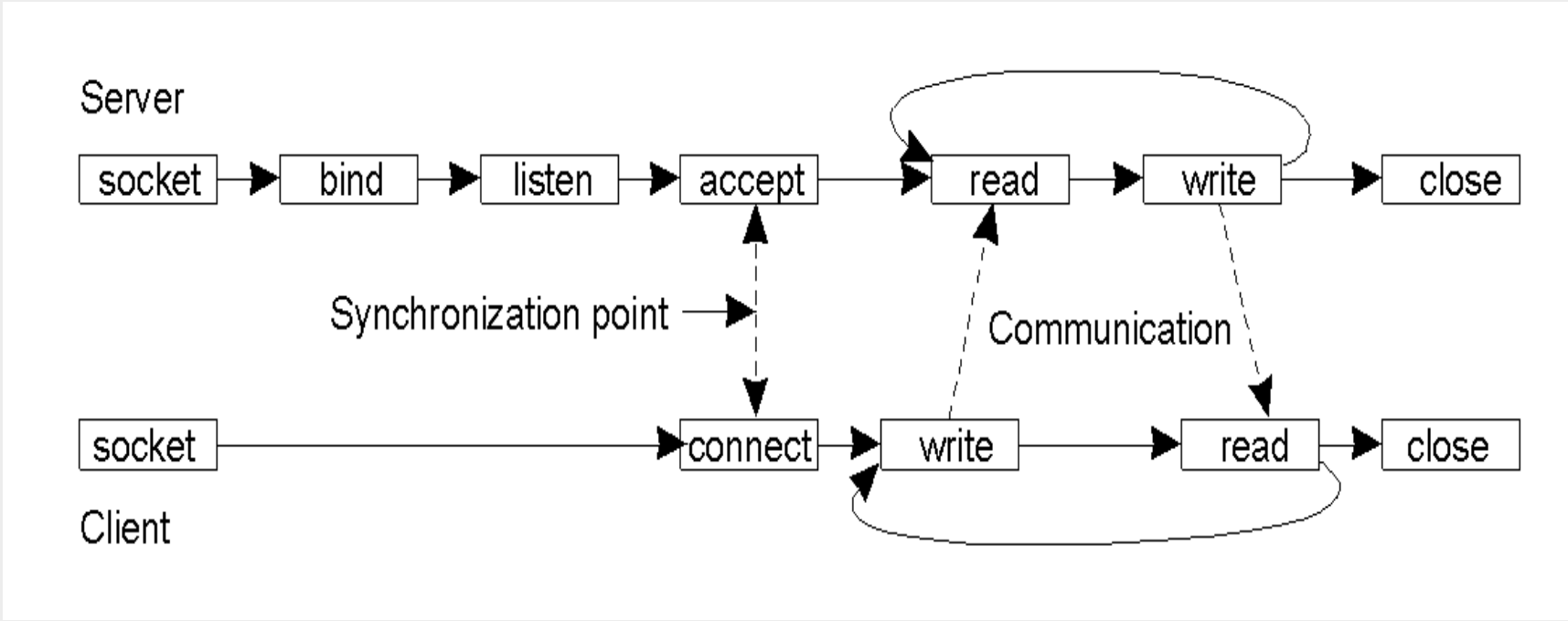
f) Синхронное нерезидентное взаимодействие с синхронизацией по выполнению

Нерезидентная связь на основе сообщений. Berkeley Sockets

- http://**www.myhost.com:8080/index** …..

Примитив	**Назначение**
Socket	Создание сокета
Bind	Привязка локального адреса
Listen	Готовность к принятию соединения
Accept	Ждать приема соединения
Connect	Установить соединение
Send	Послать данные
Receive	Принять данные
Close	Закрыть соединение

Диаграмма взаимодействия



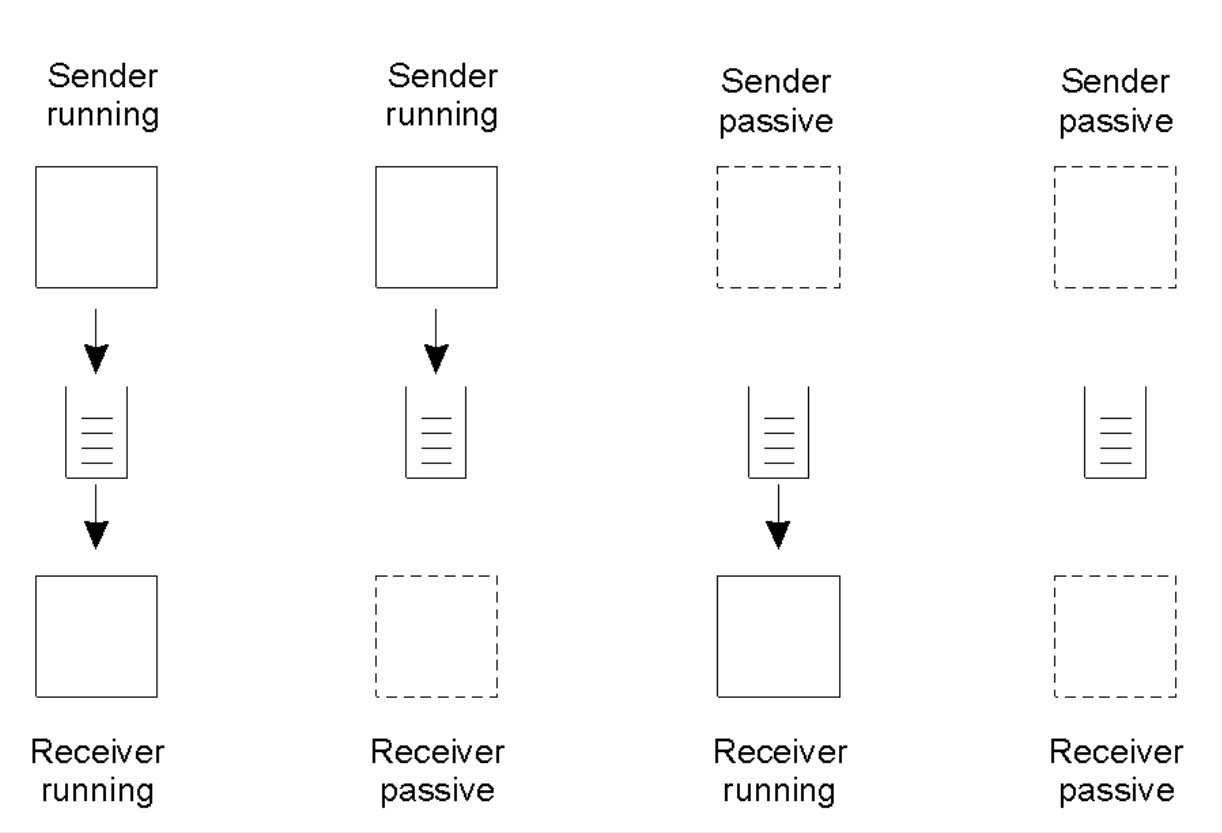
- Взаимодействие основанное на установлении соединения

Интерфейс MPI(Message-Passing Interface)

Примитив	Назначение
MPI_bsend	Append outgoing message to a local send buffer
MPI_send	Send a message and wait until copied to local or remote buffer
MPI_ssend	Send a message and wait until receipt starts
MPI_sendrecv	Send a message and wait for reply
MPI_isend	Pass reference to outgoing message, and continue
MPI_issend	Pass reference to outgoing message, and wait until receipt starts
MPI_recv	Receive a message; block if there are none
MPI_irecv	Check if there is an incoming message, but do not block

Взаимодействия на основе очереди сообщений

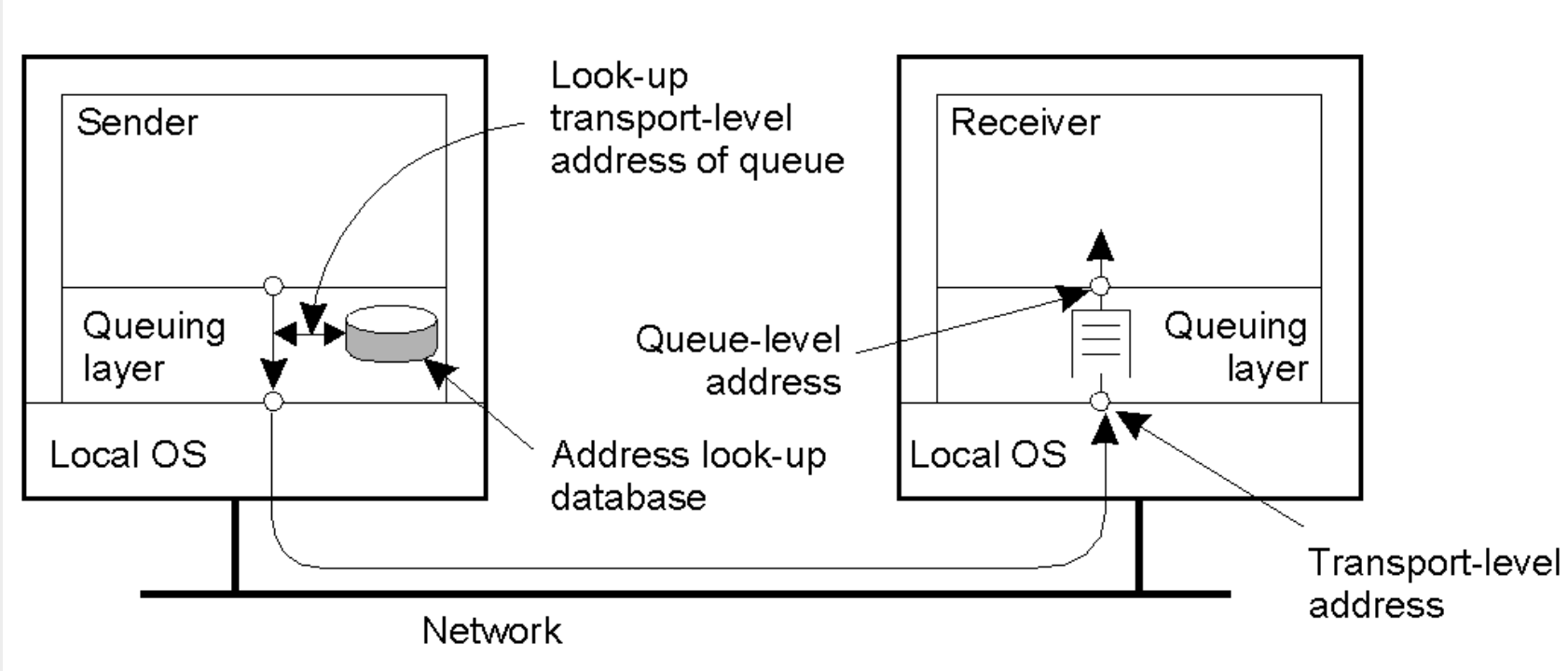
- Слабосвязанное взаимодействие



Взаимодействия на основе очереди сообщений

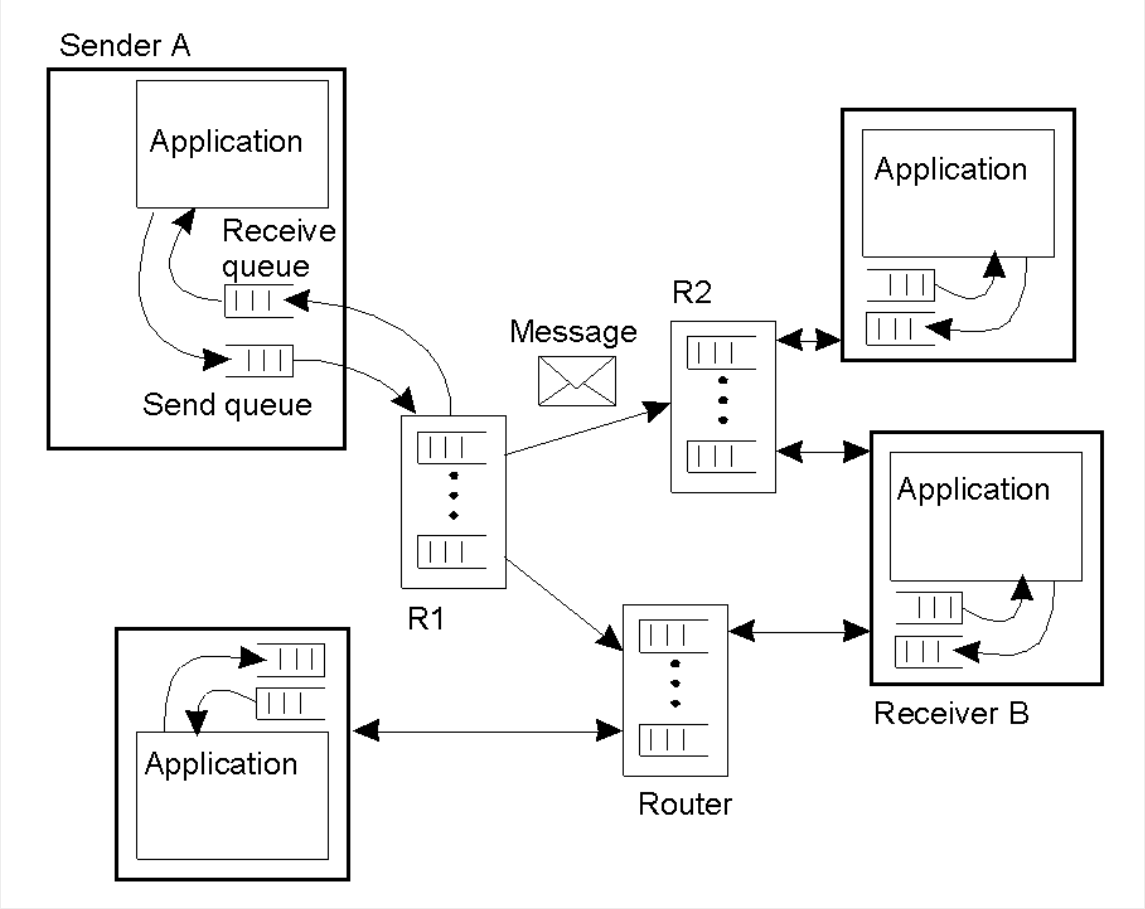
Примитив	Назначение
Put	Добавить в очередь
Get	Извлечь из очереди
Poll	Извлечь из очереди без ожидания
Notify	Установить обработчик сообщения и поступлении события

Общая архитектура механизма очередей сообщений

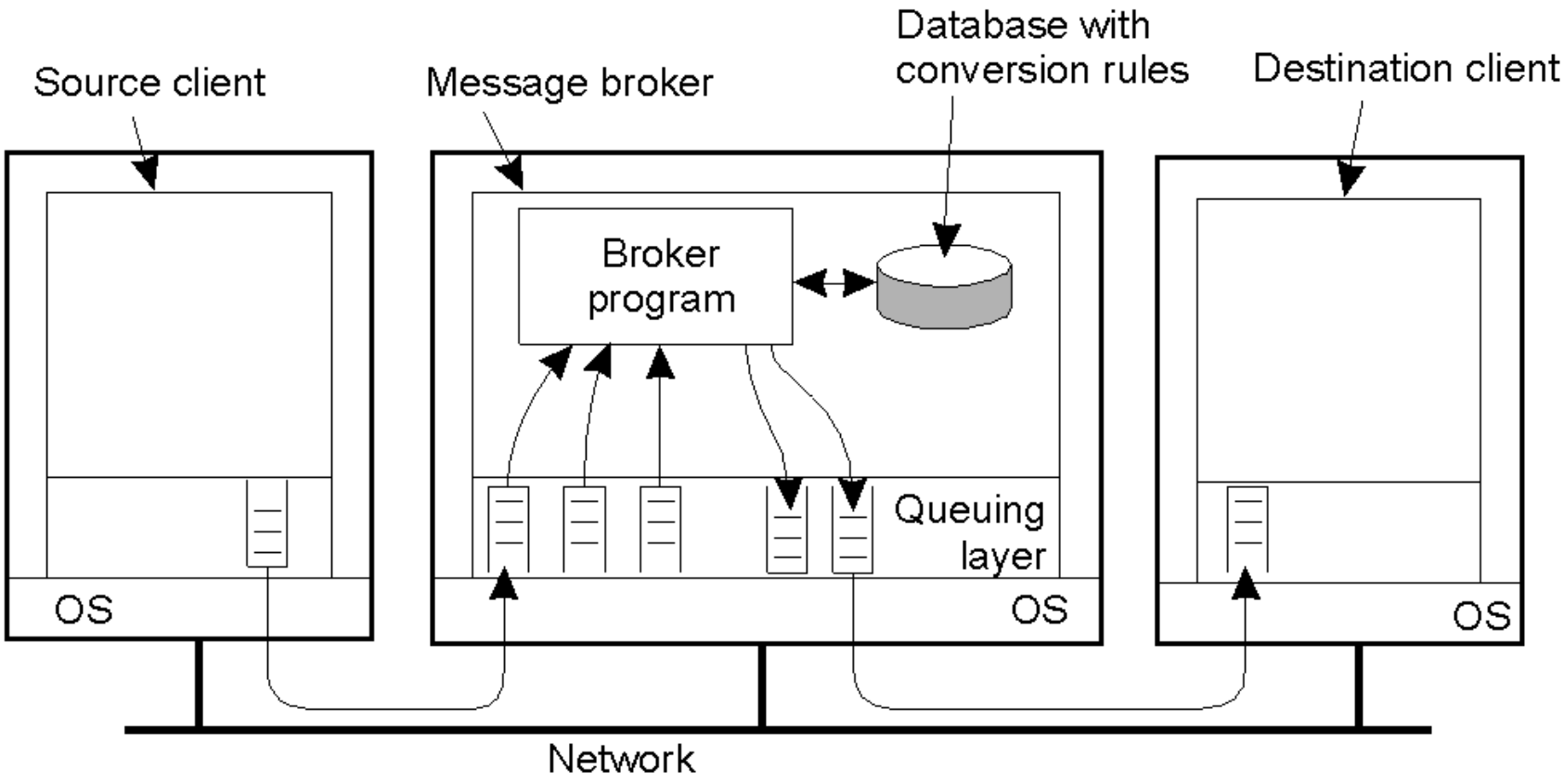


Общая архитектура механизма очередей сообщений

- Использование роутеров при передаче сообщений.

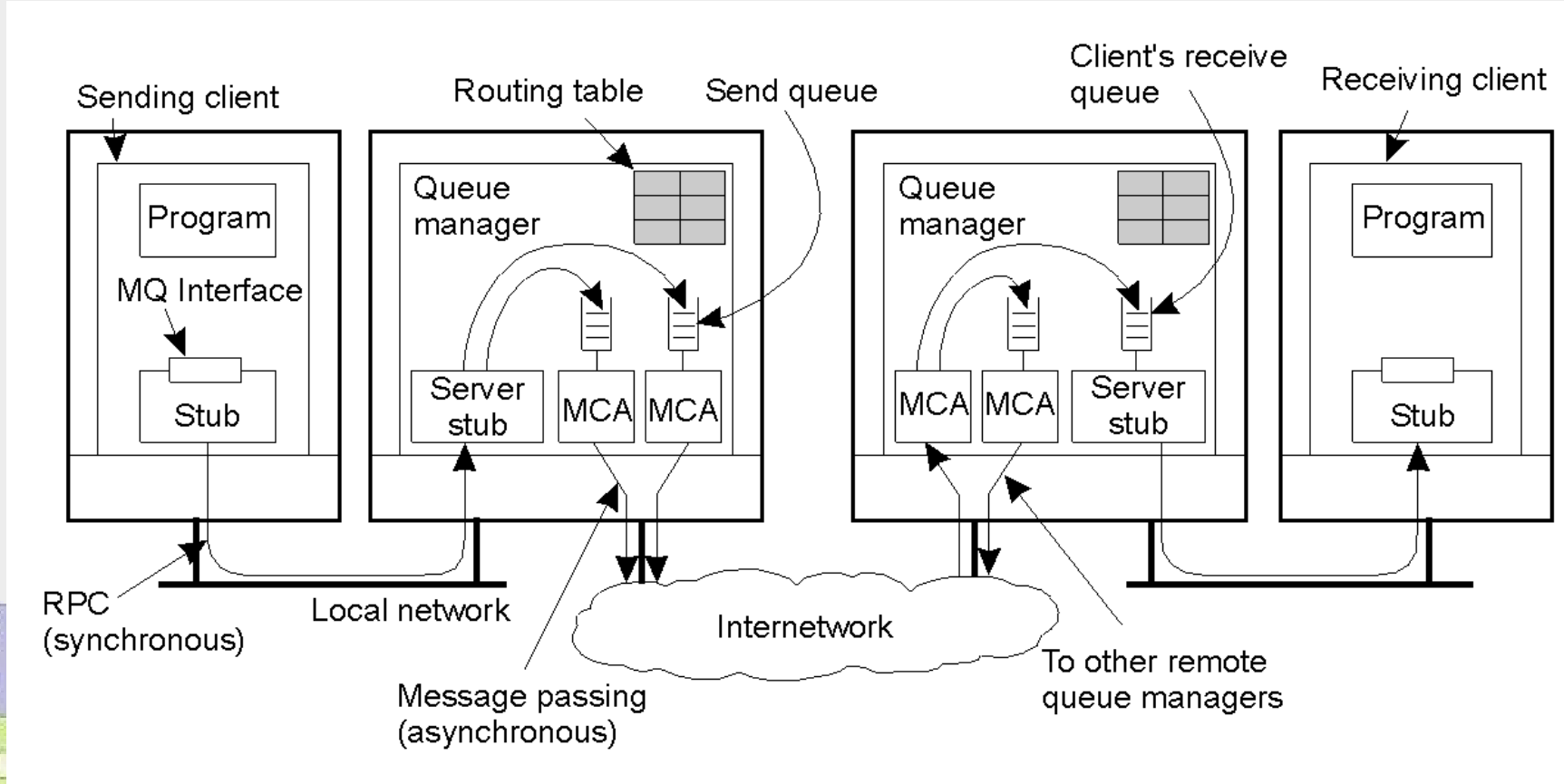


Брокеры сообщений



Пример: IBM MQSeries

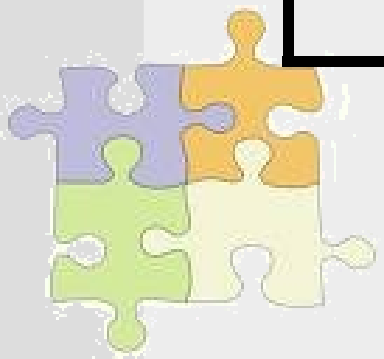
- Общая организация



Каналы(Channels)

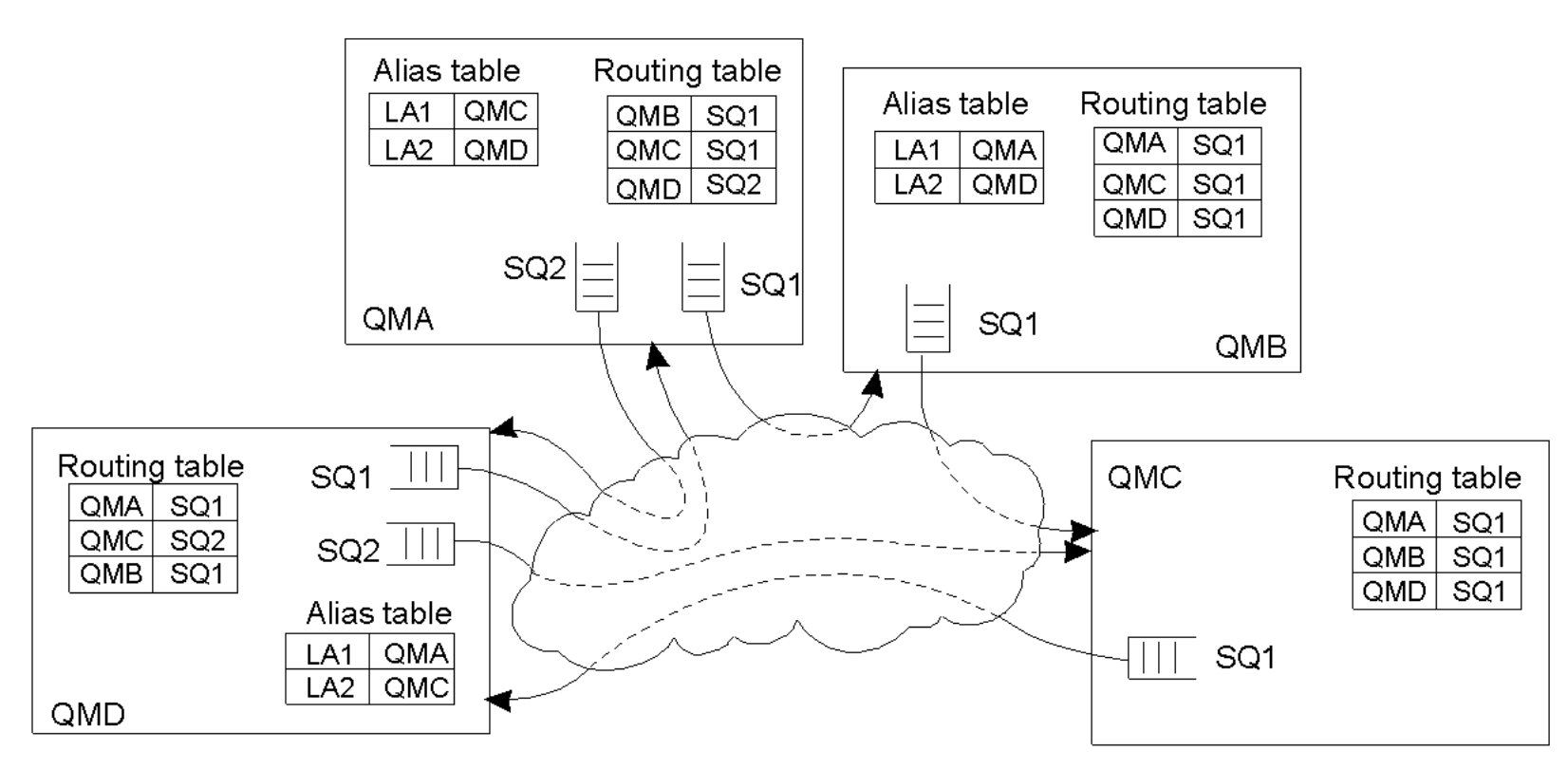
- Некоторые атрибуты описания канала

Атрибут	**Описание**
Transport type	Транспортный протокол
FIFO delivery	Порядок доставки сообщений
Message length	Максимальная длина сообщений
Setup retry count	Максимальное количество попыток запуска удаленного MCA
Delivery retries	Максимальное время попытки поместить сообщение в очередь



Передача сообщения

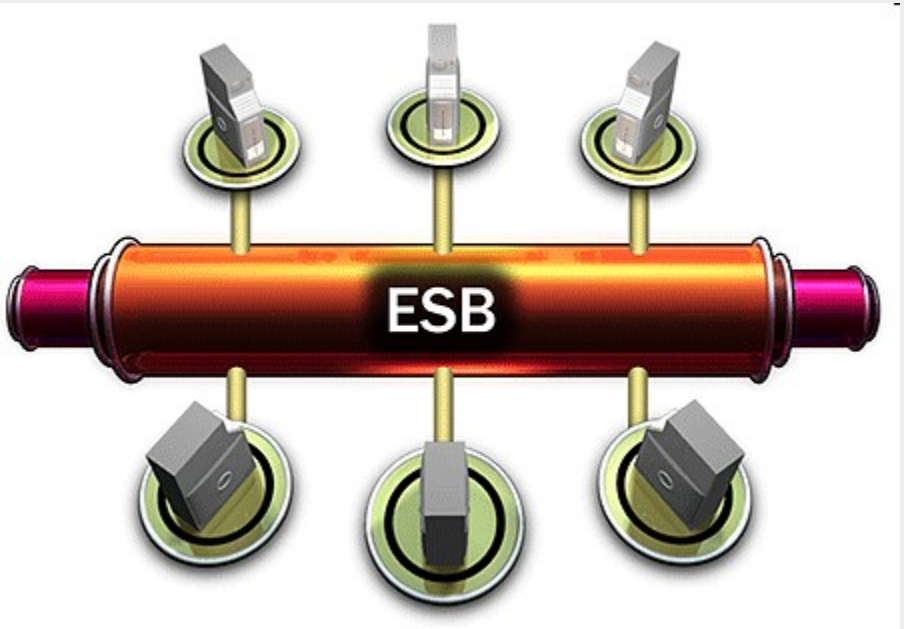
- Использование таблиц маршрутизации и псевдонимов



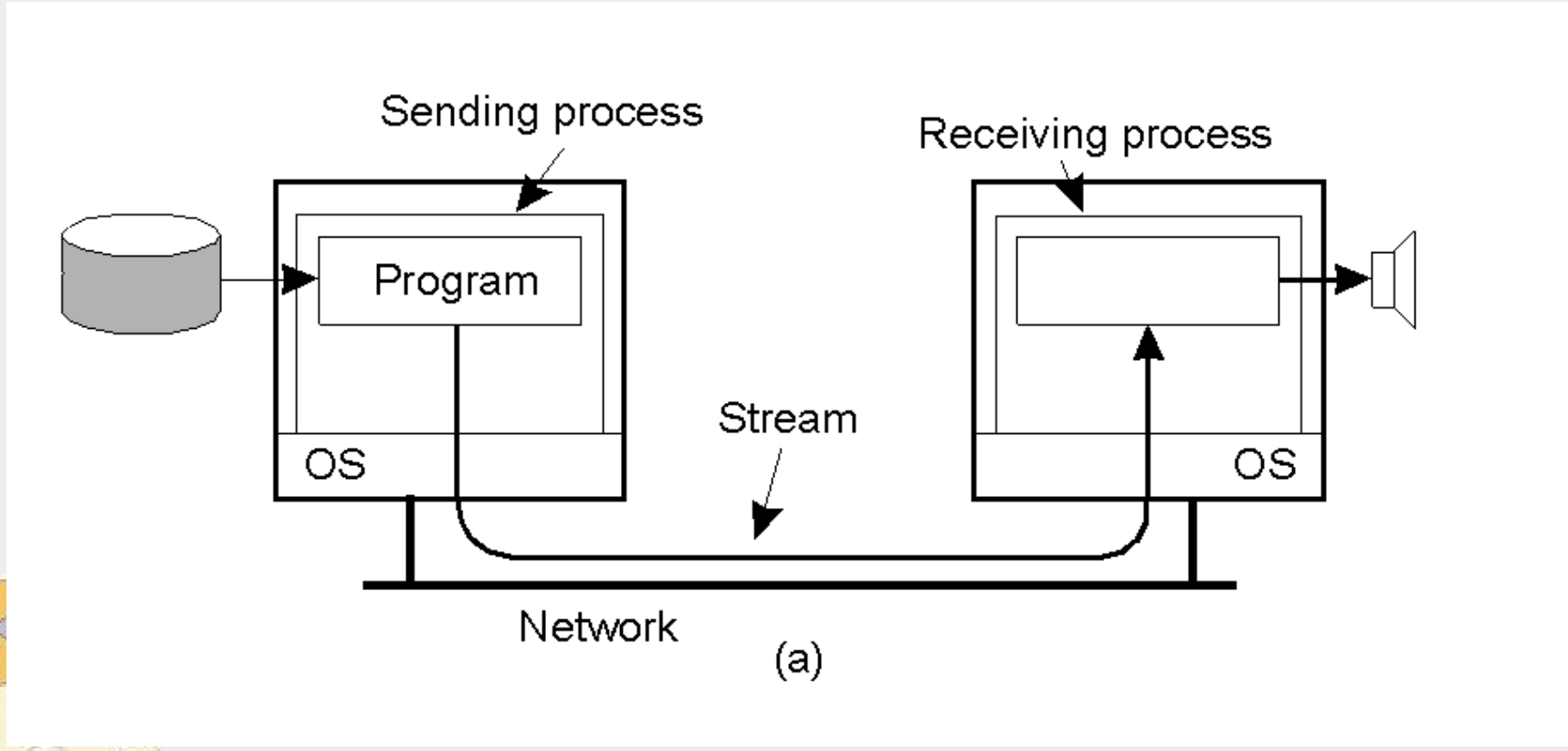
Известные реализации

- Windows Message Queues
- JMS (Java)

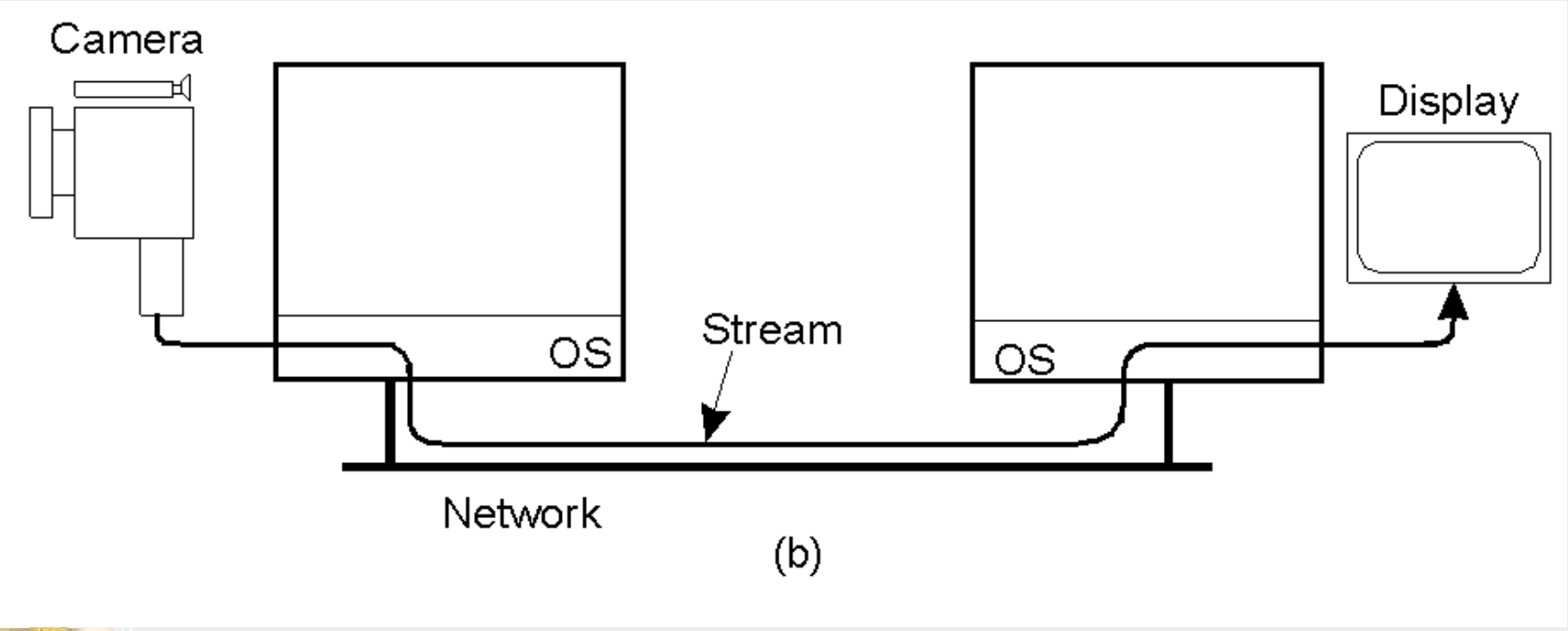
- Продукты семейства ESB (Enterprise Service Bus)



Потоки данных

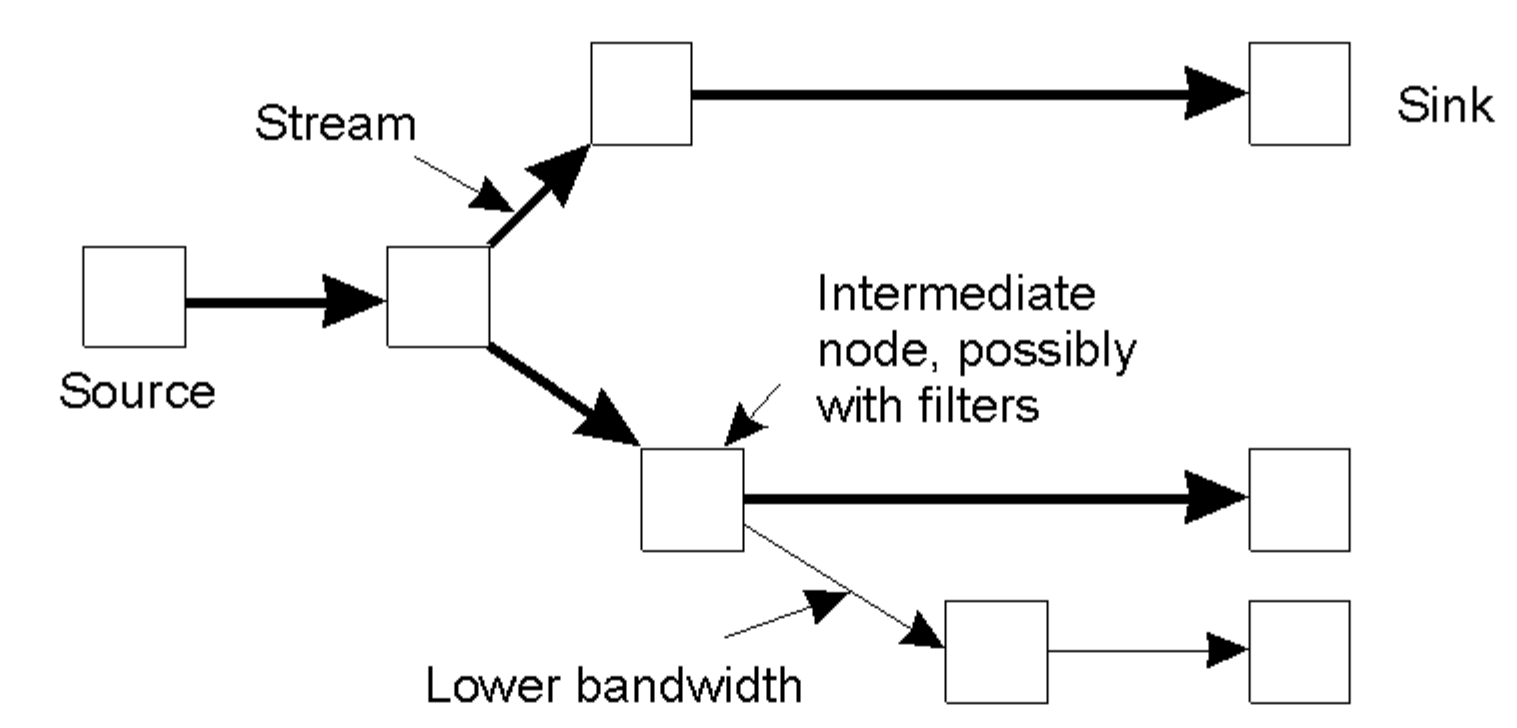


Потоки данных



Потоки данных

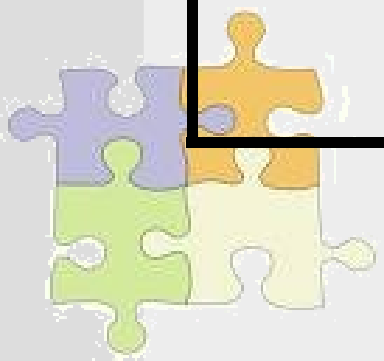
- Пример фильтрации потока данных



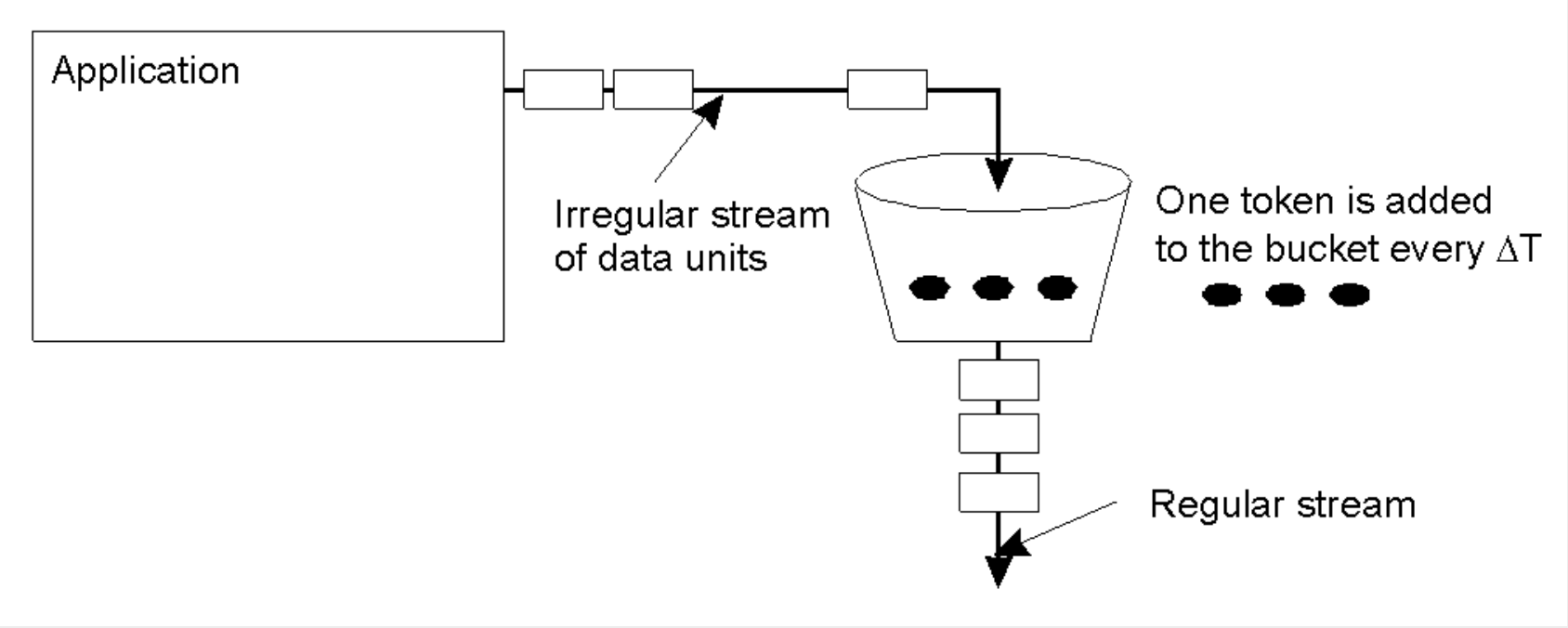
Спецификация QoS

- QoS – Quality Of Service

Характеристики входного потока	Требуемый сервис
▪Максимальный размер элемента данных ▪Скорость передачи корзины элементарных пакетов ▪Размер корзины ▪Максимальная скорость передачи	▪Чувствительность к потерям(bytes) ▪Чувствительность к интервалам (μsec) ▪Чувствительность к групповым потерям (data units) ▪Минимальная фиксируемая задержка (μsec) ▪Максимальное отклонение задержки (μsec) ▪Показатель гарантии(number)



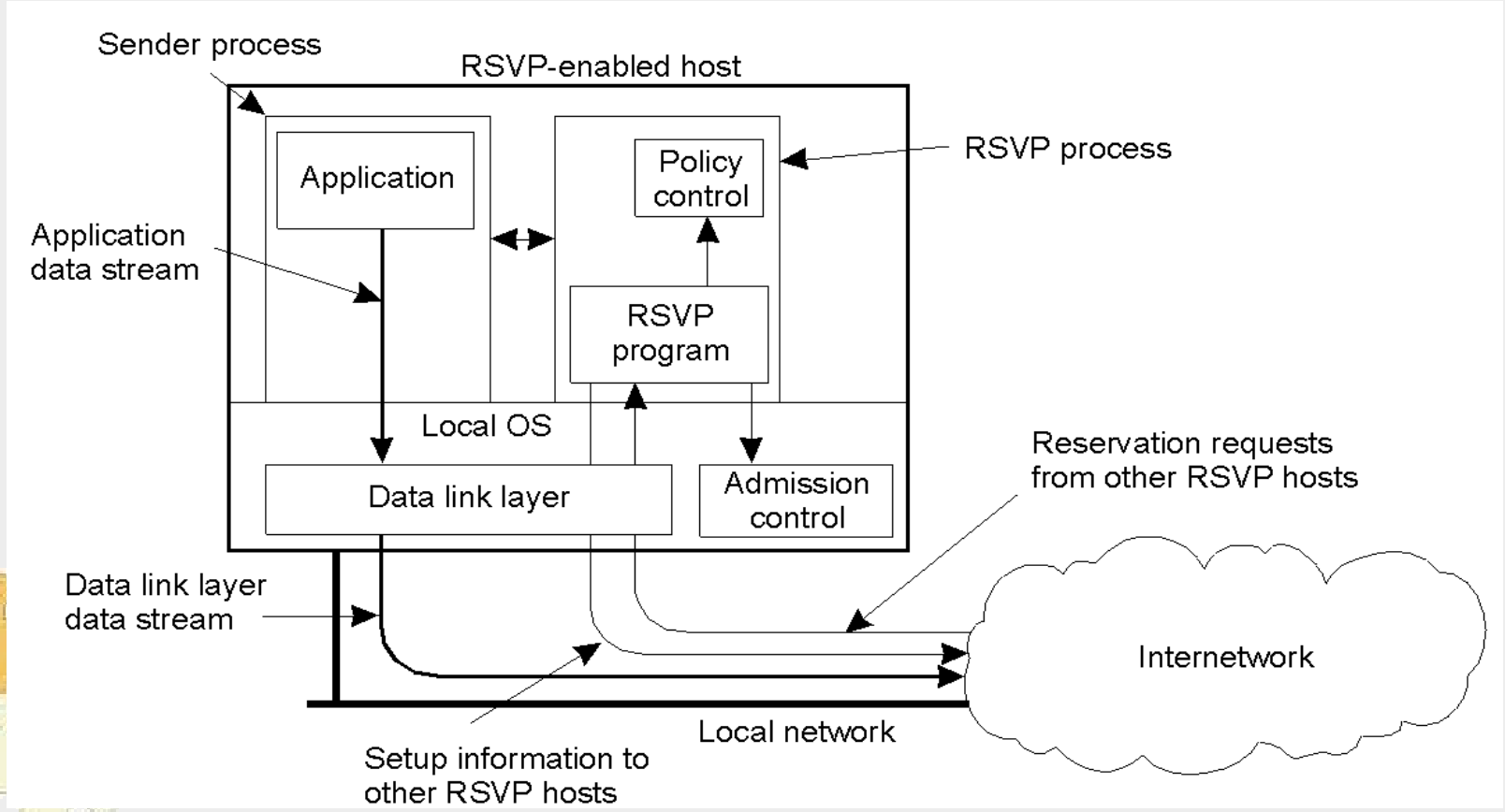
Спецификация QoS



- Принцип работы корзины пакетов

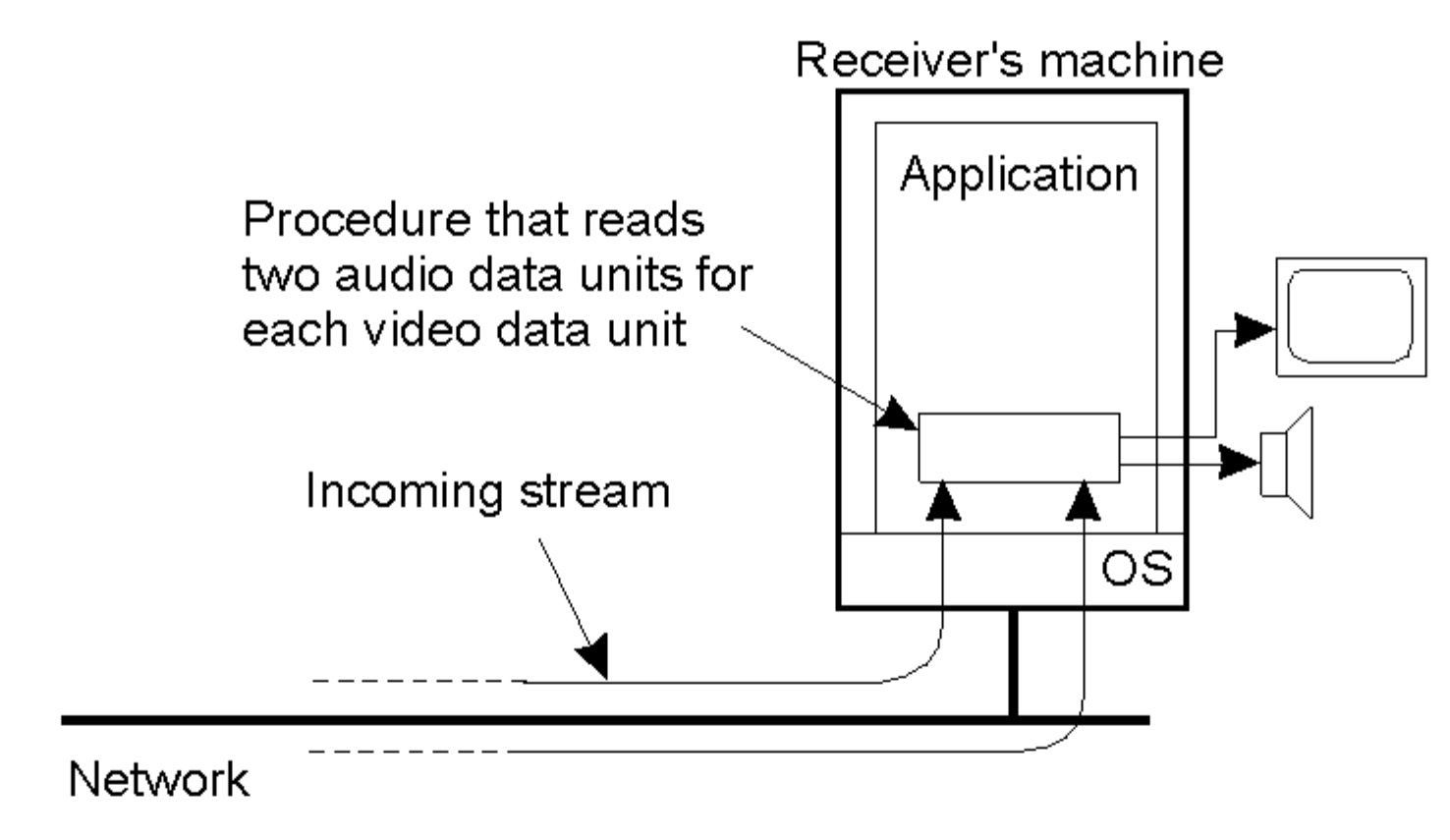
Создание потока

- **RSVP** – Resource reSerVation Protocol

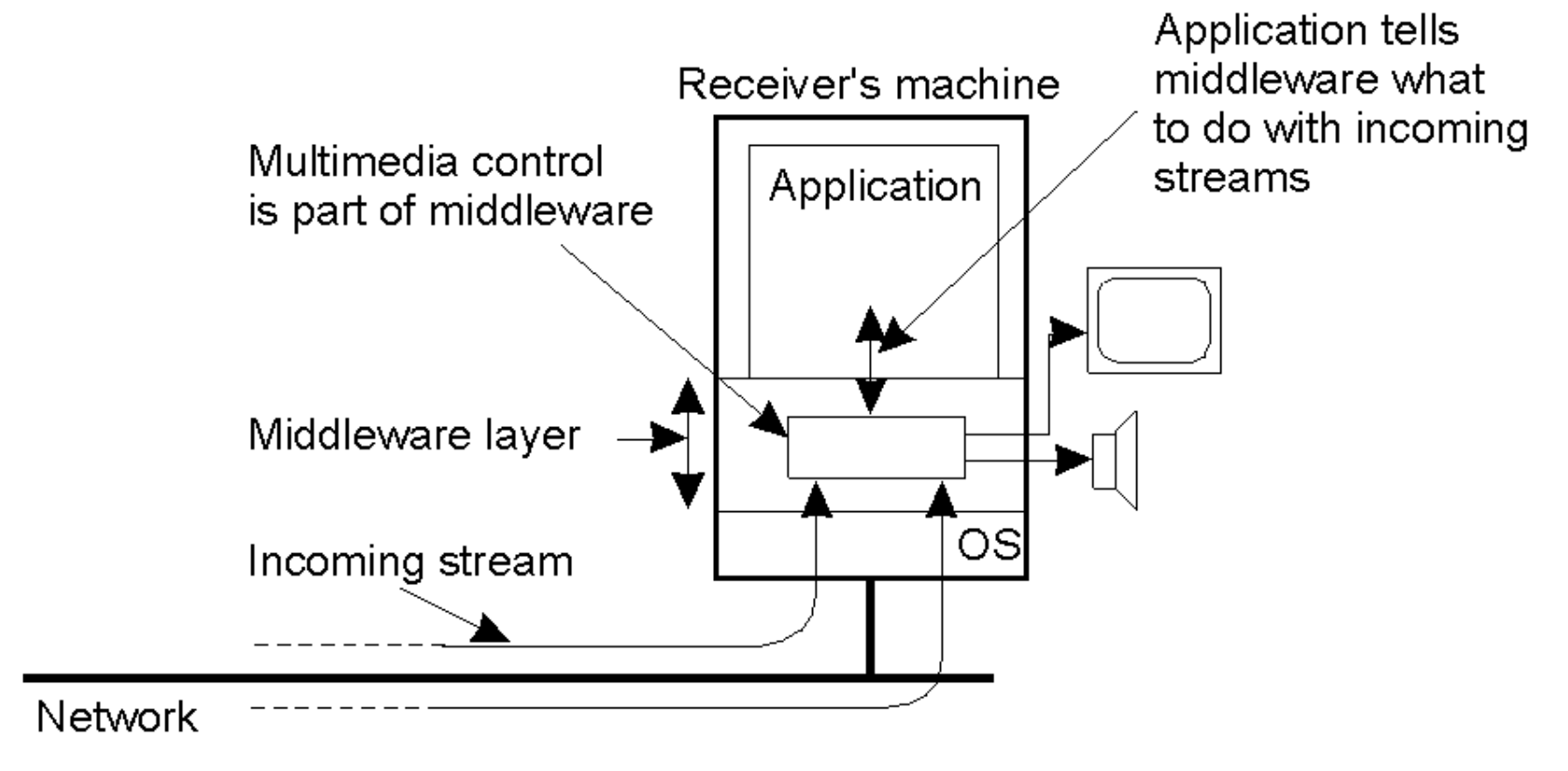


Механизм синхронизации

- Синхронизация данных на уровне приложения
- 2 чтения аудио данных + 1 чтение видео данных



Механизм синхронизации



- Использование промежуточного уровня