Перед организацией сетевого взаимодействия необходимо распределить функции между общающимися процессами: кто из них будет выполнять функции клиента, а кто сервера. Роли могут изменяться динамически (в один момент времени процесс является клиентом (шлет запрос), в другой – сервером (обслуживает запрос)). При этом сервер может обслуживать одного клиента (рис. 1), либо многих (рис. 2).



Рис.1 Один клиент

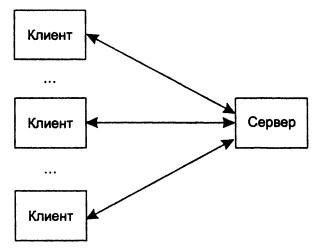


Рис.2 Несколько клиентов

После распределения функций, необходимо определить протокол обмена (формат передачи сообщений, размер, наличие повторной передачи, подтверждений и т.д.). В данной работе первым рассматривается протокол транспортного уровня TCP, который является потоковым надежным протоколом с установлением соединения, сетевое взаимодействие производится на основе сокетов. Сокет – это два значения, идентифицирующие конечную точку. В случае TCP/IP конечная точка идентифицируется IP адресом и портом. IP адрес – 32-битное число, для удобства записываемое по байтно: «192.168.10.5», где точка – разделитель байтов. Порт – идентификатор процесса, по которому ОС определяет, какому сокету пришли данные.

Протокол TCP «гарантирует доставку данных» (производит повторную передачу при необходимости, устраняет дублирование при получении двух копий пакета), производит упорядочивание данных, управление потоком. Так же, данный протокол автоматически рассчитывает таймаут, по истечении которого будет происходить повторная передача (данный таймаут будет зависеть от размеров сети, её загруженности).

Стоит отметить, что протокол является потоковым, а значит, в нем нет понятия о сообщении (здесь уместна аналогия с последовательным портом, когда мы не можем заранее определить границу сообщений). Поэтому необходимо в приложении обеспечить механизм выявления сообщений из потока байт. Решить данную проблему позволяют два способа: передача сообщений по разделителям и передача с указанием длины сообщения. Был использован 2 способ.

В общем виде алгоритм работы сервера, следующий:

1. Создание сокета. При этом локальные адрес и порт сокету назначаются из числа свободных;
2. Привязка сокета к удаленному адресу с помощью вызова bind
3. Перевод сокета в состояние прослушивания соединений с помощью вызова listen. При этом данные передавать через сокет нельзя, единственная его задача – получение запросов на соединение;
4. Прием соединений клиентов с помощью вызова accept. Данный вызов блокирует выполнение потока, пока не придет соединение от клиента, в результате которого создастся новый сокет, связанный с адресом клиента;
5. Далее происходит прием/передача данных.

Алгоритм клиента:

1. Клиент создает сокет;
2. Сокет привязывается к удаленному хосту;
3. Производится обмен.