Сервис обмена сообщениями с возможностью приоритезации, отправки широковещательных сообщений, а также запроса ответа либо гарантии доставки.

Краткое описание

Сервис обмена сообщениями представляет собой многопоточный, асинхронный сервер, реализованный с применением библиотеки boost::asio и стека протоколов TCP::IP. В процессе работы сервиса N различных процессов могут зарегистрироваться в сервисе и отправлять друг другу сообщения нескольких типов, различающихся способом доставки:

- Сообщения без гарантий доставки (не будут доставлены неактивным клиентам)
- Сообщения с гарантией доставки (можно отправлять сообщения для неактивного клиента, они будут доставлены, когда он подключится к сервису)
- Сообщения с ответом (когда клиент, отправивший сообщение, блокируется до получения ответа)
- Широковещательное сообщение (по сути сообщение без гарантий доставки всем зарегистрированным в сервисе клиентам)

Кроме того, для сообщений может быть установлен приоритет отправки: CASUAL (обычный), IMPORTANT (важный), CRITICAL (критический).

Протокол обмена

Сервис работает через TCP/IP и реализует следующий протокол прикладного уровня:

- Формат сообщений представлен в таблице:

	Назначение поля	**Длина (байт)**	**Примечание**
Заголовок	Маркер начала сообщения	1	Фиксированное значение ‘&’
	Длина сообщения (с учетом заголовка)	8	Int64_t
	ID (Идентификатор сообщения)	8	Int64_t
	Приоритет	1	CASUAL=0, IMPORTANT=1, CRITICAL=2
	КОМАНДА	10	C-string
	ЛОГИН (Адресат)	10	C-string
	Уровень загрузки сервера	1	
	Пользовательское сообщение		Минимум 1 байт

- Перечень используемых команд:
  LOGIN- регистрация сокета логином, заданным в поле ЛОГИН заголовка
  CONFIRM - подтверждение регистрации логина, а также подтверждение приема сообщения с заданным ID
  ABORT – отказ регистрации с заданным логином (логин уже зарегистрирован)
  CLIENTS - запросить перечень активных клиентов
  S_ANSWER – отправка сообщения с ожиданием ответа

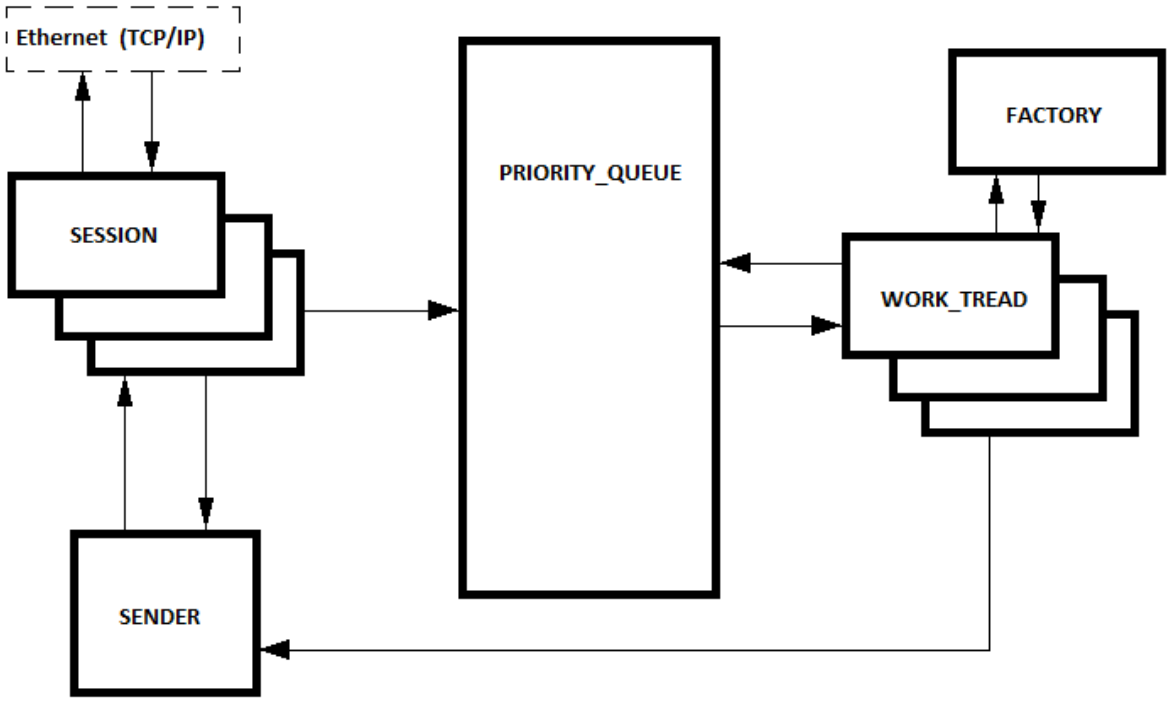
ANSWER - ответ на сообщение с заданным ID
S_STRONG - отправка сообщения с гарантией доставки
S_WEAK - отправка сообщения без подтверждения получения
BROADCAST – послать сообщение всем зарегистрированным в сервисе (без подтверждения получения)
DISCONNECT – штатное отключение клиента
LOAD - запрос текущей загрузки сервера
LOAD_ANS – ответ на запрос загрузки сервера

Архитектура сервиса

Архитектура сервиса реализована на следующих принципах:

- Сетевой обмен реализован с применением библиотеки boost::asio и использует для функционирования стек протоколов TCP/IP.
- Все используемые операции из boost::asio асинхронные и запускаются на общем объекте типа io_context, работающем сразу в нескольких потоках.
- Для взаимодействия с клиентами используется пул объектов Session, каждый из которых владеет сокетом и непосредственно инициирует операции чтения и записи.
- Все поступающие сообщения распаковываются объектами Session и попадают в единую очередь PriorityQueue, где они размещаются с учетом установленного приоритета и порядка поступления.
- Несколько рабочих потоков WorkThred одновременно работают с очередью PriorityQueue. В их задачу входит обработка сообщений, выявление адресата и выбор способа отправки.
- Для непосредственной отправки сообщения адресату объекты WorkThread пользуются функционалом объекта Sender
- В задачи объекта Sender входит поиск нужного объекта Session по логину адресата и передача сообщения на отправку.

Диаграмма взаимодействия описанных объектов схематически представлена на рисунке:



Алгоритм обработки сообщений

Сообщения без гарантии доставки, широковещательные сообщения и ответы на сообщения обрабатываются одинаково:

1) При поступлении такого сообщения оно помещается в очередь PriorityQueue. Одновременно уведомляются потоки WorkThred о наличии работы.
2) Объект WorkThred, вызвавшийся обработать сообщение, определяет тип и адресата сообщения (или нескольких в случае широковещательного сообщения) и передает сообщение объекту Sender.
3) Объект Sender определяет какой из сокетов (объектов Session) соответствует адресату и отдает сообщение на отправку.
4) Если в процессе отправки произошла ошибка или выясняется, что адресат в настоящий момент отключен, сообщение теряется.

Сообщения с гарантией доставки и сообщения, требующие обязательного ответа, обрабатываются следующим образом:

1) При поступлении такого сообщения оно также помещается в очередь PriorityQueue. Затем уведомляются потоки WorkThred о наличии работы.
2) Объект WorkThred вызвавшийся обработать сообщение, определяет тип сообщения и создает для сообщения объект TimerKeeper, содержащий в себе таймер, настроенный на срабатывание через заданный промежуток времени. Длительность промежутка устанавливается при запуске сервиса в файле конфигурации.
3) Далее объект WorkThred определяет адресата сообщения и передает сообщение объекту Sender.
4) Объект Sender определяет какой из сокетов (объектов Session) соответствует адресату и отдает сообщение на отправку.
5) Предполагается, что клиент, получив такое сообщение, отправит ответное сообщение с командой CONFIRM или ANSWER и идентификатором ID подтверждаемого сообщения.
6) Любой объект SESSION получив сообщение CONFIRM или ANSWER помещает идентификатор подтвержденного сообщения в специальный лист ConfirmedList. Обработка сообщения с командой CONFIRM на этом заканчивается, а сообщения с командой ANSWER далее обрабатывается как обычное сообщение без подтверждения.
7) При срабатывании таймера объект TimerKeeper проверяет лист ConfirmedList и, если в нем не оказывается идентификатора сообщения, прикрепленного к TimerKeeper, это сообщение возвращается обратно в очередь PriorityQueue.
8) Таким образом, если в процессе отправки произошла ошибка или клиент по каким-то причинам не успел подтвердить получение сообщения, оно будет отправлено повторно.
9) Если же в процессе отправки сообщения Sender выясняет, что клиент в настоящий момент отключен, сообщение помещается в резервное хранилище и будет возвращено в PriorityQueue при подключении адресата.

!!! Важно понимать, что клиент должен дать ответ за отведенное время. Если же размер сообщения не позволяет клиенту уложиться в срок, необходимо пользоваться сообщениями без гарантии доставки. Поскольку в сервисе используется протокол TCP, такое сообщение скорее всего будет доставлено, при условии наличия активного получателя.

Настройки сервиса

Для настройки сервиса используется файл “server.config” , расположенный в рабочей папке сервиса. Указанный файл должен иметь следующий формат:

```
Log_files_path=D:\\_temp\\
Buffer_size=1024
Port=9000
Waiting=time 50
```

где:

“Log_files_path”- путь для сохранения лог-файлов;

“Buffer_size” – размер буфера для чтения сообщения (должен быть больше 50 байт). Данный параметр может влиять на быстродействие и может подбираться исходя из средней длины сообщений в сервисе;

“Port” – порт для первоначального подключения клиента к сервису;

“Waiting_time” – интервал времени, за который клиент должен дать ответ или подтвердить сообщение (при необходимости).

Возможная схема масштабирования

Сервис разрабатывался с учетом теоретической возможности масштабирования, когда несколько серверов будут работать параллельно. Для реализации масштабирования выбрана горизонтальная схема “persistent connection”, когда каждый клиент держит открытое соединение с каждым сервером. Балансировка загрузки серверов должна осуществляться клиентом по следующей схеме:

- каждый сервер соединен с каждым клиентом
- сервер определяет степень своей загрузки по числу сообщений, находящихся в данный момент в очереди. Чем выше это число, тем сильнее загружен сервер.
- клиент получает информацию о загрузке сервера в каждом сообщении, а также может запросить ее отдельно (такой запрос является приоритетным и обрабатывается без помещения в очередь).
- на основе информации о загрузке серверов клиент может выбрать с каким сервером ему работать в настоящий момент.