Полное руководство по сетевому программированию для разработчиков игр. Часть 1 (скучная).

Автор: **x84**

Предисловие

"Вот так... Копишь миллионы, копишь...
А потом БАЦ! ... тортом тебе в морду!"
Очень-очень известный и всеми любимый герой IT

Когда я впервые столкнулся с необходимостью написать приложение, которое могло бы взаимодействовать с таким же приложением, запущенным на другом компьютере, я был неприятно удивлен дефицитом полезной русскоязычной документации по этому делу. Конечно, я знал английский, и для меня не составило особого труда разобраться в тонкостях сетевого программирования, но что делать человеку который не знает ничего кроме русского ("русский язык велик и могук!" :))? Ответа нет... И даже если он знает английский язык, то поначалу это ему не очень-то поможет. Лично я не видел еще ни одного систематизированного и каталогизированного источника информации о программировании сетевых приложений, который бы в той или иной степени охватывал весь этот огромный хаос.

Итак, уже сделано все, что касается однопользовательских режимов игры, однако было бы неплохо добавить возможность игры по сети. И ты, конечно, даже и не представляешь, с чего начать... С Интернетом ты ранее сталкивался только в двух случаях: форум на gamedev.ru и навязчивые pop-ups от порносайтов... Хорошо, я тебе помогу, вернее, тебе поможет мой CGNP. Для того чтобы не было недоразумений, я сразу оговорюсь, что написанное

повезло - man pages не могут быть "не под рукой" ;)). Еще я хотел бы сделать предупреждение: представленный ниже материал не претендует на полноту освещения затронутых в нем тем, а также на абсолютную точность.

И наконец, перед тем, как мы окунемся в омут с головой, я дам еще один совет: дружище, выучи все-таки английский! Он тебе очень пригодится. Ведь когда ты захочешь стать гуру сетевого программирования, тебе придется прочесть очень много RFC-документов, а ошибки перевода и неправильного толкования технических спецификаций являются "бомбами замедленного действия"!

Модель OSI

Чтобы понять все принципы взаимодействия компьютеров на расстоянии, надо знать так называемую модель OSI (ISO OSI == International Organization for Standardization Open System Interconnection - Взаимодействие Открытых Систем по Стандарту Международной Организации по Стандартизации)... Теперь можем сделать перерыв, чтобы ты, уважаемый читатель, смог еще пять раз перечитать предыдущее предложение и понять его смысл, после чего мы разберемся, что такое OSI, и с чем ее едят...

Итак, модель OSI определяет несколько "уровней" взаимодействия компьютеров на расстоянии (я намеренно избегаю словосочетания "по сети", и ты скоро поймешь почему). Вот эти уровни:

---

7. Прикладной

Это уровень, максимально приближенный к пользовательскому интерфейсу. Пользователи конечного программно продукта не волнует, как передаются данные, зачем и через какое место... Он сказали "ХОЧУ!" - а мы, программисты, должны им это обеспечить. В качестве примера можно взять на рассмотрение любую сетевую игру: для игрока она работает на этом уровне. Пользователь куда то ткнул, в интерфейсной части программы зафиксирована его команда. Что надо передать? Что то приняли, что произошло в мире игры?

6. Представительский

Здесь программист имеет дело с данными, полученными от низших уровней. В основном, это конвертирование и представление данных в удобоваримом для пользователя виде.

5. Сеансовый

передавать данные, как цельный поток. Здесь на сцену вступают протоколы HTTP, FTP, Telnet, SMTP и т.д.

4. Транспортный

Осуществляет контроль над передачей данных (сетевых пакетов). То есть, проверяет их целостность при передаче, распределяет нагрузку и т.д. Этот уровень реализует такие протоколы, как TCP, UDP и т.д. Для нас представляет наибольший интерес.

3. Сетевой

Логически контролирует адресацию в сети, маршрутизацию и т.д. Должен быть интересен разработчикам новых протоколов и стандартов. На этом уровне реализованы протоколы IP, IPX, IGMP, ICMP, ARP. В основном, управляется драйверами и операционными системами. Сюда влезать, конечно, стоит, но только когда ты знаешь, что делаешь, и полностью в себе уверен.

2. Канальный

Этот уровень определяет, как биты собираются в пакеты, какую служебную информацию надо передавать и как на неё реагировать, но поле данных каждого пакета максимально сырое и представляет из себя просто биты в навал, нет вообще ни какой последовательности пакетов в длинных информационных потоках. В поле данных просто вложены пакеты сетевых протоколов, всё, что чего не хватает должно быть сделано в них, или ещё выше.

1. Аппаратный (Физический)

Контролирует передачи представление битов и служебных сигналов физическими процессами и отправку физических сигналов между аппаратными устройствами, входящими в сеть. То есть управляет передачей электронов по проводам. Нас он не интересует, потому что все, что находится на этом уровне, контролируется аппаратными средствами (реализация этого уровня - это задача производителей хабов, мультиплексоров, повторителей и другого оборудования). Мы не физики-радиолюбители, а геймдевелоперы.

Итак, подведем небольшой итог к тому, что было представлено... Мы видим, что, чем выше уровень - тем выше степень абстракции от передачи данных, к работе с самими данными. Это и есть смысл всей модели OSI: поднимаясь все выше и выше по ступенькам ее лестницы, мы все меньше и меньше заботимся о том, как данные передаются, мы все больше и больше становимся заинтересованными в самих данных, нежели в средствах для их передачи. Каждый следующий уровень скрывает в себе предыдущий, облегчая жизнь пользователю этого уровня, будь он программист, радиоинженер или твоя подруга, которая не знает, как настроить MS Outlook Express...

Начнем потихоньку... :)

Сокеты и бла-бла-бла...

У каждой уважающей себя современной операционной системы есть средства для взаимодействия с другими компьютерами. Самым распространенным среди программистов средством для упомянутых целей являются сокеты. Сокеты - это API (Application Programming Interface - Интерфейс Программирования Приложений) для работы с уровнями OSI. Сокеты настолько гибки, что позволяют работать почти с любым из уровней модели OSI. Хочешь - формируй IP-пакеты руками и займись хакингом, отправляя "неправильные" пакеты, которые будут вводить сервера в ступор, хочешь - займись более благоразумным делом и создай новый удобный голосовой чат, хочешь - игрульку по сети гоняй, не хочешь - твое право, но этот случай мы в данном руководстве не рассматриваем... :)

Когда мы создаем сокет (socket - гнездо), мы получаем возможность доступа к нужному нам уровню OSI. Ну а дальше мы можем использовать соответствующие вызовы для взаимодействия с ним. Для того чтобы понять сокеты, можно провести аналогию с телефонным аппаратом и телефонной трубкой. Сокеты устроены таким образом, что они могут взаимодействовать с ОС на любом уровне OSI, скрывая ту часть реализации, которой мы не интересуемся (тебя же не волнует, как работает телефон, когда ты набираешь 03). Телефоны и сокеты бывают разные: бывают старые телефоны с дисковым набором и бывают низкоуровневые сокеты для работы с Ethernet-фреймами, бывают супер-модные цифровые телефоны и бывают сокеты для работы с верхними уровнями стека протоколов... и т.д. Причем вызовы для всех типов сокетов одни и те же, что, имхо, очень удобно. Когда мы создаем сокет, мы также заставляем систему организовать два канала: входящий (это как громкоговоритель у телефона) и исходящий (микрофон). Осуществляя чтение и запись в эти каналы, мы приказываем системе взять на себя дальнейшую судьбу данных, т.е. передать и проследить, чтоб данные дошли вовремя, в нужной последовательности, не искаженные и т.п. Система должна давать (и дает) максимум гарантий (для каждого уровня OSI - гарантии свои), что данные будут переданы правильно. Наша задача - поместить их в очередь, а на другом конце - прочитать из входящей очереди и обработать должным образом. Все остальное - нам ни к чему. Еще один плюс - сокеты переносимы. То есть изначально концепция сокетов была разработана в Berkeley, поэтому классическая реализация сокетов называется Berkeley sockets или BSD sockets (BSD == Berkeley Software Distribution). В дальнейшем, почти все ОС тем или иным образом унаследовали эту реализацию. В каждой ОС степень поддержки сокетов разная, но точно могу сказать: в современных операционных системах MS и *nix - сокеты поддерживаются настолько, насколько нам, геймдевелоперам, они могут понадобиться. Больше нам и не нужно, потому что мы не кодим под экзотические ОС, потому что, в свою очередь, геймеры (они наша целевая аудитория) на таковых не сидят. Однако по мере изучения

мы будем придерживаться классической реализации BSD sockets, и стараться по минимуму использовать си-зависимый код.

Короче, сокеты надо представлять себе так: сокет - это окно в мониторе, которое выходит во внешний мир. Если у кого-то еще открыто такое же окно, то можно контактировать. Ну что? Если вы уже достаточно раскочегарились, уважаемые читатели, то пора к делу.

Страницы: **1** 2 3 Следующая »

#OSI, #сокеты

18 мая 2003 (Обновление: 20 янв 2011)

Комментарии [4]

Контакт
Сообщества
Участники
Каталог сайтов
Категории
Архив новостей