**Лекция за 07.03.2012, часть 1**

***Стек протоколов.***

Раньше практически каждая программа, работающая по сети, должна была реализовать свой собственный протокол передачи данных, начиная чуть ли не с физического уровня. Разумеется, это создавало множество проблем, и необходимость повторной разработки того или иного кода приводила как к ошибкам, так и к потере эффективности и времени. В связи с этим, в конце 70-х – начале 80-х годов прошлого века возникла концепция ***стек протоколов***.

При любой передаче данных, эти данные всегда можно разделить на две части – ***полезные данные*** (полезная нагрузка) и ***заголовок данных*** (служебная информация, необходимая для правильного декодирования полезных данных). Идея стека протоколов состоит в разделении протоколов по зонам ответственности. Какие-то протоколы отвечают за пользовательские приложения, какие-то – за доставку данных, какие-то – за физический уровень (непосредственное представление данных в виде электрических сигналов). При этом весь пакет данных более высокого уровня считается полезной нагрузкой для нижележащего протокола (рис. 1).

*сторона отправителя*

*сторона получателя*

Header

Payload

Header

Payload

Header

Payload

Header

Payload

Header

Payload

Header

Payload

III ур.

I ур.

II ур.

*передача данных*

*Header –* заголовок данных

*Payload –* полезная нагрузка

*(рис. 1)*

На стороне получателя происходит принятие пакета данных из канала связи. Пользуясь данными, расположенными в заголовке, данные декодируются, и извлекается полезная нагрузка. По той же схеме полезная нагрузка отправляется на дешифровку следующим уровнем обработки этого стека. Процесс повторяется, пока не дойдет до самого верхнего уровня (уровня приложения), где данные уже непосредственно предоставляются пользователю. Эта идея и получила название стек протоколов.

Не стоит считать, что эта концепция была единственной из рассматриваемых в свое время, но, тем не менее, сейчас она является общим местом всей сетевой индустрии и никаких других решений уже не применяется.

В чем же состоят основные ***достоинства*** этого подхода? Тут можно привести примерно те же соображения, что и в случае объектно-ориентированного подхода.

Модель обладает развитой инкапсуляцией и абстракцией данных. На каждом уровне стека протоколов мы абстрагируемся как от представления данных этажом выше, так и от представления данных этажом ниже. Эти данные хранятся в пакетах, которые имеют в той или иной степени фиксированную структуру и обрабатываются единообразно на каждом уровне стека. То есть, фактически, для передачи данных нам не важно, какой протокол физического уровня мы используем (например, будь то Ethernet или GPRS).

Так же у этого подхода имеются и ***недостатки***. Стоит отметить, что изначально эти недостатки оценивались очень высоко и временами воспринимались чуть ли не фатально.

Недостатки проистекают из необходимости передавать большое количество служебной информации, возможно даже избыточной. Избыточной она может быть из-за того, что информация, использующаяся на разных уровнях стека протоколов, вполне может дублироваться. Но в связи с тем, что обработка данных осуществляется последовательно, мы вынуждены все равно вставлять эту дублирующуюся информацию в соответствующие заголовки.

Помимо всего прочего, к нижнему уровню заголовки, накладываясь друг на друга, вырастают до довольно заметных размеров. Например, минимальная длина заголовков IP и TCP составляет 20 октетов. Вместе 40 октетов. Минимальная длина заголовка Ethernet составляет 14 октетов. Вместе 54 октета. Максимальная длина фрейма Ethernet 1500 октетов. Таким образом, общий объем передаваемой служебной информации составляет примерно 3% от общего количества передаваемых данных. При этом если мы используем более короткие пакеты данных, то избыточность возрастает еще больше. Мы можем произвольно варьировать размер передаваемой полезной нагрузки, но размер заголовков мы варьировать не можем.

Чтобы бороться с передачей дублированных данных разрабатывались различные стеки протоколов и, в результате, эта проблема была решена одной из самых первых. И примерно в 80-е годы стал вопрос о ***стандартизации*** стека протоколов.

В США 80-е годы ознаменовались периодом глобальной стандартизации инженерных механизмов. Кроме отраслевых стандартов, что существовали у них довольно давно, в это время там начали стандартизировать все подряд. Тут проявились разные подходы к стандартизации, которые есть в Европе (и в России) и есть в США.

В чем же разница? У нас есть государственный комитет по стандартам Госстандарт, который на основании некоторой внутренней, причем достаточно закрытой и непрозрачной процедуры, принимает решение о введении нового стандарта. После этого этим стандартом обязаны пользоваться.

В США же стандарты всегда вырастали из отраслевых стандартов путем создания группы заинтересованных лиц (special interesting group). То есть, когда в какой-либо отрасли назревает идея что-либо стандартизировать, стороны, заинтересованные в этом, собираются и обсуждают этот стандарт. Однако и у этой схемы есть свои недостатки. К примеру, срок принятия стандартов при ней может быть очень длительным и некоторые стандарты обсуждаются десятками лет, а заканчиваются только когда в них перестает быть нужда. В целом же можно сказать, что в каких-то ситуациях лучше работает советско-российская схема стандартизации (проистекающая из немецкой схемы), а в каких-то – англо-британская.

Примерно в том же ключе происходила стандартизация стека протоколов. Этим занимались две разные организации – американский институт стандартов ANSI (American National Standards Institute) и организация Open System Initiative.

ANSI не является государственной организацией, и сама по себе утверждает стандарты, но не принимает их. Она занимается протежированием групп заинтересованных лиц, которые могут обмениваться информацией через эту организацию. В свою очередь, организация Open System Initiative была создана для выработки единых стандартов программного обеспечения. При этом не стоит думать, что эта организация имеет отношению к открытому ПО, и стандарты, которые принимает Open System Initiative, далеко не всегда связаны с открытым кодом и подобными вещами.

Постепенно работа по стандартизации в рамках ANSI перетекла под эгиду ISO (International Organization for Standardization), в результате чего в середине 90-х годов появилась модель, которая получила название модель ISO ANSI.

Безусловно, группы, работающие в рамках ANSI и ISO, и группы, работающие в рамках Open System Initiative, были осведомлены о работе друг друга. При этом утверждается, что промежуточными результатами они не делились. Тем не менее, они получили настолько похожие модели, что с тех пор эта модель стека называется ***референсная модель стека ISO/OSI***.

Поначалу эту модель постигла неутешительная судьба. Практически сразу она была воспринята в штыки очень многими специалистами, занимающимися созданием и проектированием программного обеспечения. В первую очередь отмечалась сложность этой модели и ее кажущаяся избыточность. Например, в модели TCP/IP четыре уровня, тогда как в модели ISO/OSI уровней семь. С другой стороны, на тот момент не существовало реализации ни одного стека протоколов, который бы полностью соответствовал модели ISO/OSI, а стек TCP/IP на тот момент уже был известен и стабилен.

Стандарт проникал в массы с трудом. Хотя он и строился на довольно серьезных предположениях, это был некий теоретический стандарт, навязанный сверху. Если бы развитие велось снизу вверх, от использования прикладных решений, отношение к этому стандарту было бы иным. Но в те времена было так много реализаций различных сетевых стеков, что найти что-то общее между ними всеми было довольно сложно, поэтому стандарт строился на основе теоретических соображений.

Первым стеком, полностью соответствующим спецификации ISO/OSI, был стек ISDN (высокоскоростное соединение поверх телефонной линии). После его появления плотина была прорвана, так как стало понятно, что модель ISO/OSI может быть референтным образом реализована и, как показал опыт, реализована весьма неплохо.

Немного ***терминологии:***

***Протокол*** – совокупность форматов представления данных и правил их обработки.

***Представление данных*** – бывает ***пакетным*** и ***непрерывным***.

При пакетном представлении данные разделяются на фрагменты постоянной или переменной длины, которые и называются пакетами (или фреймами в англоязычной литературе, если речь идет о протоколах физического уровня). Протоколами с непрерывным представлением данных являются протоколы, в которых по каким-либо причинам затруднительно выделить фиксированные фреймы. Как правило, это относится к протоколам физического уровня, либо, наоборот, к пользовательским протоколам.

Рассмотрим поподробнее ***модель ISO/OSI***.

|  |  |
| --- | --- |
| 7 ур. | Application/user |
| 6 ур. | Presentation |
| 5 ур. | Seance |
| 4 ур. | Transport |
| 3 ур. | Network |
| 2 ур. | Logical Link Control |
| 1 ур. | Physical |

***Уровень приложения (Application/user)***. Уровень, на котором работают непосредственно пользовательские программы. Например, протоколы HTTP и SMTP, протоколы ICQ или GoogleTalk и т.д.

***Уровень представления (Presentation)***. Именно к этому уровню в свое время было выражено больше всего претензий со стороны программистов, которые реализовывали настоящие стеки протоколов. Отчасти это, возможно, оправдано, потому что во всех реализованных стеках с этим уровнем связано наибольшее число трудностей.

Задача приложения на этом уровне – взять данные из протокола пользовательского уровня (уровня приложения), сериализовать, т.е. порезать на куски постоянной или переменной длины, а так же представить все данные в виде машинно-независимого представления. Именно машинно-независимое представление и реализуется на уровне представления с большим трудом.

Типичным примером является проблема представления big-endian и little-endian. Чуть ли не самая частая ошибка при написании протоколов передачи данных связана с разными порядками следования байт в слове на разных процессорах. Другим примером можно привести слишком большее количество кодировок.

В современной литературе при описании уровня представления машинно-независимое представление иногда опускают вовсе, однако не стоит забывать, что по стандарту он есть. И более того, реализации уровня представления, где машинно-независимое представление осуществлено, существуют. Например, протокол RPC (Remote Procedure Call).

***Сеансовый уровень (Seance)***. Отвечает за поддержание сеанса связи. Сеансом называется некий процесс взаимодействий приложений по сети, в котором можно выразить начало и конец сеанса. Например, работа с браузером, устанавливающим соединение с веб-серверами, которые имеют ограничения по времени сохранения сессии, т.е. если время ожидания истекло, соединение нужно установить заново. Другой пример – TCP сеанс.

Здесь тоже есть свои трудности, потому что существует большое количество протоколов, в которых сеансы не поддерживаются принципиально. Например, многие протоколы онлайн передачи аудио и видео данных.

Еще одна задача, которая ставится на сеансовом уровне – аутентификация пользователей. Однако в большинстве современных реализаций аутентификацию берет на себя пользовательский уровень, а на сеансовом уровне, если и происходит какая-то аутентификация, то, как правило, аутентификация удаленной системы, например, по IP-адресу.

***Транспортный уровень (Transport)***. Отвечает за доставку данных и поддержание целостности данных.

***Сетевой уровень (Network)***. Отвечает за пространство имен и маршрутизацию. Пространство имен должно быть в любой сети, причем, желательно обеспечить в нем уникальность для возможности выделения получателя. Примером пространства имен можно привести пространство IP-адресов.

Два нижних уровня относятся к физическим уровням и связанны с непосредственным представлением данных в физических сетях.

***Физический уровень (Physical)***. Связан с непосредственным представлением данных в виде последовательности сигналов, которые передаются по каналу связи. Альтернативное название – Media Access Control (MAC).

***Уровень LLC*** ***(Logical Link Control)***. По задумке модели ISO/OSI этот уровень должен был играть по отношению к физическому уровню ту же роль, что транспортный по отношению к сетевому. Т.е. уровень LLC должен был отвечать за поддержание целостности данных на уровне физического канала связи.

Однако в большинстве стеков, как и уровень представления, он остается нереализованным.

Здесь интересно отметить протокол NetBEUI (NetBIOS Extended User Interface) для небольших одноранговых сетей, который был распространен в 90-е годы в системах Microsoft. Его особенностью было то, что он был организован на уровне LLC, а остальных верхних уровней, кроме пользовательского, там не было. Т.е. данные пользовательского уровня непосредственно вкладывались в специализированный LLC фрейм.

***Расширение*** модели.

Как уже было отмечено, модель, представленная в стандарте, периодически встречает неудовольствие. Частично это происходит из-за наличия уровней, которые многими признаются мертвыми (уровень LLC и уровень представления), но, с другой стороны, периодически кому-то, наоборот, не хватает уровней.

Например, в современных книгах Microsoft представлена восьмиуровневая модель, где над уровнем приложения размещен уровень User Integration, являющийся уровнем абстракции рабочего стола, т.е. неким единым пользовательский интерфейсом. А в литературе по менеджменту программного обеспечения над седьмым уровнем можно встретить надстройку порой до двенадцатого уровня. В этом случае добавляются уровни, связанные с управлением программным обеспечением, с документным оборотом, с управлением предприятием и т.д.

Физический уровень же сейчас делится на две части – ***MII (Media Independent Interface)*** и, непосредственно, физический уровень. Носители данных в настоящее время могут быть разными – ими могут являться как коаксиальный кабель, так и оптоволоконный кабель, и даже беспроводная среда. Идея разделения состоит в том, что форматы фреймов данных, всегда постоянные для данных типов сетей, относятся к уровню Media Independent Interface, а непосредственное представление данных в виде электрических или световых импульсов относится к строго физическому уровню.

В заключении, стоит отметить, что все, что лежит между физическим уровнем и уровнем представлений, всегда является строго пакетными протоколами. Строго говоря, на физическом уровне тоже нет никаких физических фреймов. Сигналы, которые мы передаем в канале связи - непрерывный поток 0 и 1, который присутствует, даже если устройство не передает данные, в таком случае просто передается некий сигнал заполнитель, оповещающий о наличии устройства. Поэтому, особенно, если мы выделяем MII и физический уровень, можно говорить, что на физическом уровне представление данных является непрерывным. Точно так же на пользовательском уровне представление данных может быть не пакетным, например, в случае аудио или видео потока или SMTP протокола.