48. Прикладные протоколы TCP\IP на примере протоколов почтового обмена SMTP и POP3, протокола HTTP.

Олифер. Компьютерные сети.

http://zakon.znate.ru/docs/index-14115.html?page=153

Прикладной уровень

Прикладной уровень стека TCP/IP соответствует трем верхним уровням модели OSI: прикладному, представления и сеансовому. Он объединяет сервисы, предоставляемые системой пользовательским приложениям. За долгие годы применения в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и служб прикладного уровня. К ним относятся такие распространенные протоколы, как протокол передачи файлов (File Transfer Protocol, FTP), протокол эмуляции терминала telnet, простой протокол передачи почты (Simple Mail Transfer Protocol, SMTP), протокол передачи гипертекста (Hypertext Transfer Protocol, HTTP) и многие другие. Протоколы прикладного уровня развертываются на хостах (конечный узел).
На прикладном уровне работает большинство сетевых приложений.

В массе своей эти протоколы работают поверх TCP или UDP и привязаны к определённому порту, например:

HTTP на TCP-порт 80 или 8080,
FTP на TCP-порт 20 (для передачи данных) и 21 (для управляющих команд),
SSH на TCP-порт 22,
запросы DNS на порт UDP (реже TCP) 53,
обновление маршрутов по протоколу RIP на UDP-порт 520.

Эти порты определены Агентством по выделению имен и уникальных параметров протоколов (IANA).
К этому уровню относятся: DHCP, Echo, Finger, Gopher, HTTP, HTTPS, IMAP, IMAPS, IRC, NNTP, NTP, POP3, POPS, QOTD, RTSP, SNMP, SSH, Telnet, XDMCP.

SMTP (англ. Simple Mail Transfer Protocol — простой протокол передачи почты) — это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.
ESMTP (англ. Extended SMTP) — масштабируемое расширение протокола SMTP. В настоящее время под «протоколом SMTP», как правило, подразумевают ESMTP и его расширения.

SMTP реализуется несимметричными взаимодействующими частями: SMTP- клиентом и SMTP-сервером. Важно отметить, что этот протокол ориентирован на передачу данных по направлению от клиента к серверу, следовательно, SMTP-клиент работает на стороне отправителя, а SMTP-сервер — на стороне получателя. SMTP-сервер должен постоянно быть в режиме подключения, ожидая запросов со стороны SMTP-клиента.
Логика работы протокола SMTP действительно является достаточно простой (как это и следует из его названия). После того как, применяя графический интерфейс своего почтового клиента, пользователь щелкает на значке, инициирующем отправку сообщения, SMTP-клиент посылает запрос на установление TCP-соединения на порт 25 (это назначенный порт SMTP-сервера). Если сервер готов, то он посылает свои идентифицирующие данные, в частности свое DNS-имя. Затем клиент передает серверу адреса (имена) отправителя и получателя. Если имя получателя соответствует ожидаемому, то после получения адресов сервер дает согласие на установление TCP-соединения, и в рамках этого надежного логического канала происходит передача сообщения. Используя одно TCP-соединение, клиент может передать несколько сообщений, предваряя каждое из них указанием адресов отправителя и получателя. После завершения передачи TCP- и SMTP- соединения разрываются. Если в начале сеанса связи SMTP-сервер оказался не готов, то он посылает соответствующее сообщение клиенту, в ответ тот снова

посылает запрос, пытаясь заново установить соединение. Если сервер не может доставить сообщение, то он передает отчет об ошибке отправителю сообщения и разрывает соединение. После того как передача сообщения благополучно заканчивается, переданное сообщение сохраняется в буфере на сервере.

Для приёма почты почтовый клиент должен использовать протоколы POP3 или IMAP. Работа с SMTP происходит непосредственно на сервере получателя. Поддерживает функции: установление соединения, аутентификация, передача данных.

Чтобы доставить сообщение до адресата, необходимо переслать его почтовому серверу домена, в котором находится адресат. Для этого обычно используется запись типа MX (англ. Mail eXchange — обмен почтой) системы DNS. Если MX запись отсутствует, то для тех же целей может быть использована запись типа A.

В настоящее время протокол SMTP является стандартным для электронной почты и его используют все клиенты и серверы.
Протокол был разработан для передачи только текста в кодировке ASCII, кроме того, первые спецификации требовали обнуления старшего бита каждого передаваемого байта. Это не даёт возможности отсылать текст на национальных языках (например, кириллице), а также отправлять двоичные файлы (такие как изображения, видеофайлы, программы или архивы). Для снятия этого ограничения был разработан стандарт MIME, который описывает способ преобразования двоичных файлов в текстовые. В настоящее время большинство серверов поддерживают 8BITMIME, позволяющий отправлять двоичные файлы так же просто, как текст.

Безопасность SMTP и спам
Изначально SMTP не поддерживал единой схемы авторизации. В результате этого спам стал практически неразрешимой проблемой, так как было невозможно определить, кто на самом деле является отправителем сообщения — фактически можно отправить письмо от имени любого человека. В настоящее время производятся попытки решить эту проблему при помощи спецификаций SPF, Sender ID, DKIM. Единой спецификации на настоящий момент не существует.

POP3 (англ. Post Office Protocol Version 3 — протокол почтового отделения, версия 3) используется почтовым клиентом для получения сообщений электронной почты с сервера. Обычно используется в паре с протоколом SMTP. Предыдущие версии протокола (POP, POP2) устарели. Стандарт протокола POP3 определён в RFC 1939. Расширения и методы авторизации определены в RFC 2195, RFC 2449, RFC 1734, RFC 2222, RFC 3206, RFC 2595.
Существуют реализации POP3-серверов, поддерживающие TLS и SSL.

В протоколе POP3 предусмотрено 3 состояния сеанса:

- Авторизация (клиент проходит процедуру аутентификации).
- Транзакция (клиент получает информацию о состоянии почтового ящика, принимает и удаляет почту).
- Обновление (Сервер удаляет выбранные письма и закрывает соединение).

HTTP (сокр. от англ. HyperText Transfer Protocol — «протокол передачи гипертекста») — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом. HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов. В 2006 году в Северной Америке доля HTTP-трафика превысила долю P2P-сетей и составила 46 %, из которых почти половина — это передача потокового видео и звука.

HTTP используется также в качестве «транспорта» для других протоколов прикладного уровня, таких как SOAP, XML-RPC, WebDAV.
Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (англ. Uniform Resource Identifier) в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы, но ими могут быть логические объекты или что-то абстрактное. Особенностью протокола HTTP является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

HTTP — протокол прикладного уровня, аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами. Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Достоинства

- Простота (протокол настолько прост в реализации, что позволяет с лёгкостью создавать клиентские приложения).
- Расширяемость (Возможности протокола легко расширяются благодаря внедрению своих собственных заголовков, сохраняя совместимость с другими клиентами и серверами. Они будут игнорировать неизвестные им заголовки, но при этом можно получить необходимую функциональность при решении специфической задачи).
- Распространённость (При выборе протокола HTTP для решения конкретных задач немаловажным фактором является его распространённость. Как следствие, это обилие различной документации по протоколу на многих языках мира, включение удобных в использовании средств разработки в популярные IDE, поддержка протокола в качестве клиента многими программами и обширный выбор среди хостинговых компаний с серверами HTTP).

Недостатки и проблемы

Большой размер сообщений (Использование текстового формата в протоколе порождает соответствующий недостаток: большой размер сообщений по сравнению с передачей двоичных данных. Из-за этого возрастает нагрузка на оборудование при формировании, обработке и передаче сообщений. Д ля решения данной проблемы в протокол встроены средства для обеспечения кэширования на стороне клиента, а также средства компрессии передаваемого контента.
Нормативными документами по протоколу предусмотрено наличие прокси-серверов, которые позволяют получить клиенту документ с наиболее близкого к нему сервера. Также в протокол было внедрено diff-кодирование, чтобы клиенту передавался не весь документ, а только его изменённая часть.

Отсутствие «навигации» (Хотя протокол разрабатывался как средство работы с ресурсами сервера, у него отсутствуют в явном виде средства навигации среди этих ресурсов. Например, клиент не может явным образом запросить список доступных файлов, как в протоколе FTP. Предполагалось, что конечный пользователь уже знает URI необходимого ему документа, закачав который, он будет производить навигацию благодаря гиперссылкам. Это вполне нормально и удобно для человека, но затруднительно, когда стоят задачи автоматической обработки и анализа всех

ресурсов сервера без участия человека. Решение этой проблемы лежит полностью на плечах разработчиков приложений, использующих данный протокол.
Например, со стороны клиента используются веб-пауки — специальные программы, которые составляют список ресурсов сервера проходя по всем найденным гиперссылкам. Со стороны сервера данная проблема решается с помощью карты сайта (англ. site map) — специальной веб-страницы, где перечислены все доступные для посещения ресурсы. Она предназначена не только для людей, играя аналогичную содержанию в книге роль, но и полезна для тех же роботов-пауков позволяя уменьшить глубину — минимальное необходимое количество переходов с главной страницы. Для тех же целей служат файлы формата Sitemap, которые предназначены уже непосредственно для роботов. Полностью эта проблема решена в расширяющем HTTP протоколе WebDAV с помощью добавленного метода PROPFIND.
Данный метод позволяет не только получить дерево каталогов, но и список параметров каждого ресурса.

Нет поддержки распределённости (Протокол HTTP разрабатывался для решения типичных бытовых задач, где само по себе время обработки запроса должно занимать незначительное время или вообще не приниматься в расчёт. Но в промышленном использовании с применением распределённых вычислений при высоких нагрузках на сервер протокол HTTP оказывается беспомощен. В 1998 году W3C предложил альтернативный протокол HTTP-NG (англ. HTTP Next Generation) для полной замены устаревшего с акцентированием внимания именно на этой области. Идею его необходимости поддержали крупные специалисты по распределённым вычислениям, но данный протокол до сих пор находится на стадии разработки.