Системное программное обеспечение локальных компьютерных сетей

Форматы данных и проектирование пользовательских протоколов

Автор: Денис Пынькин
Читает: Алексей Демидчук

2014 – 2015

e-mail: denis.pynkin@bsuir.by

http://goo.gl/32cTB

СЧАСТЬЕ ДЛЯ ВСЕХ, ДАРОМ, И ПУСТЬ НИКТО НЕ УЙДЕТ ОБИЖЕННЫЙ!

Представление данных

Ясные протоколы лежат в основе хорошей практики.

Схожесть задач

- Проектирование форматов файлов для сохранения данных приложений в постоянном хранилище данных.
- Проектирование протоколов прикладного уровня для передачи (в том числе и через сеть) данных и команд между взаимодействующими программами.

Важное при проектировании прикладных протоколов

- способность к взаимодействию;

Важное при проектировании прикладных протоколов

- способность к взаимодействию;
- прозрачность;

Важное при проектировании прикладных протоколов

- способность к взаимодействию;
- прозрачность;
- расширяемость;

Важное при проектировании прикладных протоколов

- способность к взаимодействию;
- прозрачность;
- расширяемость;
- экономичность транзакций.

Типы данных

С машинной точки зрения удобно хранить данные сложной структуры в бинарном виде – все поля имеют характерный для конкретной машины формат и указатели в виде адресов.

Типы данных

С машинной точки зрения удобно хранить данные сложной структуры в бинарном виде – все поля имеют характерный для конкретной машины формат и указатели в виде адресов.
Однако такие формы представления слабо подходят для длительного хранения и передачи.

Сериализация данных

Часто ЯП и/или платформа предоставляет средства для сериализации данных и восстановления исходной структуры из байтового потока.

Минусы сериализации данных при использовании в протоколах

- проблемы взаимодействия между машинами с разной архитектурой;

Минусы сериализации данных при использовании в протоколах

- проблемы взаимодействия между машинами с разной архитектурой;
- непрозрачность для других средств;

Минусы сериализации данных при использовании в протоколах

- проблемы взаимодействия между машинами с разной архитектурой;
- непрозрачность для других средств;
- для сетевых протоколов часто целесообразно представлять структуру данных не в виде одного большого двоичного объекта, а в виде последовательности транзакций или сообщений, которые могут быть отклонены принимающей машиной.

Когда выгодно использовать бинарное представление?

- Необходимость передачи больших блоков данных и разработчик формата действительно позаботился о достижении максимальной плотности полезной информации в потоке.
  Пример: мультимедиа форматы.

Когда выгодно использовать бинарное представление?

- Необходимость передачи больших блоков данных и разработчик формата действительно позаботился о достижении максимальной плотности полезной информации в потоке.
  Пример: мультимедиа форматы.
- Существует жесткое ограничение времени и/или инструкций, необходимых для интерпретации данных.
  Пример: протоколы сетевого уровня.

Текстовое представление данных

Дуг Макилрой (изобретатель pipes):

- Будьте готовы к тому, что вывод каждой программы станет вводом другой, еще неизвестной программы.

Текстовое представление данных

Дуг Макилрой (изобретатель pipes):

- Будьте готовы к тому, что вывод каждой программы станет вводом другой, еще неизвестной программы.
- Не загромождайте вывод посторонней информацией.

Текстовое представление данных

Дуг Макилрой (изобретатель pipes):

- Будьте готовы к тому, что вывод каждой программы станет вводом другой, еще неизвестной программы.
- Не загромождайте вывод посторонней информацией.
- Избегайте строгих табличных или двоичных форматов ввода.

Текстовое представление данных

Дуг Макилрой (изобретатель pipes):

- Будьте готовы к тому, что вывод каждой программы станет вводом другой, еще неизвестной программы.
- Не загромождайте вывод посторонней информацией.
- Избегайте строгих табличных или двоичных форматов ввода.
- Не настаивайте на интерактивном вводе.

Текстовые потоки – универсальный формат передачи данных

Они просты для чтения, записи и редактирования человеком без использования специальных инструментов.

Текстовые потоки – универсальный формат передачи данных

Они просты для чтения, записи и редактирования человеком без использования специальных инструментов.
Текстовые форматы – прозрачны.

Текстовые потоки – универсальный формат передачи данных

Если необходима производительность, то можно внедрить сжатие текстового потока на уровне выше или ниже.
Часто такая конструкция является более производительной, чем бинарное представление.

Текстовые потоки – универсальный формат передачи данных

Упрощается интерпретация и анализ взаимодействия приложений, а также написание тестовых программ.

Текстовые потоки – универсальный формат передачи данных

Серверные процессы часто запускаются с помощью суперсерверов подобных inetd/xinetd, так что сервер получает команды на стандартный ввод и отправляет ответ на стандартный вывод.

Текстовые потоки – универсальный формат передачи данных

Можно взаимодействовать с сервером или клиентом с помощью программ `telnet` или `netcat`.

Пример: SMTP

telnet smtp.mail.ru 25
Trying 94.100.177.1...
Connected to smtp.mail.ru.
Escape character is '^]'.
S: 220 smtp3.mail.ru ESMTP ready
C: HELO my.server.org
S: 250 smtp3.mail.ru
C: MAIL FROM: test_testerson@mail.ru
S: 250 2.0.0 OK
C: RCPT TO: denis.pynkin@bsuir.by
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "."on a line by itself
C: from: test_testerson@mail.ru
C: to: denis.pynkin@bsuir.by
C: subject: test
C: Test message
C: .
S: 250 OK id=1PmNuu-0006QX-00

C: QUIT

Пример: POP3

telnet pop.mail.ru 110
S: +OK
C: USER test_testerson@mail.ru
S: +OK
C: PASS test01
S: +OK Welcome!
C: STAT
S: +OK 2 95232
C: LIST
S: +OK 2 messages (95232 octets)
S: 1 40269
S: 2 54853
S: .
C: RETR 1
S: +OK 40269 octets
Here is the message
C: DELE 1
S: +OK message 1 deleted
C: QUIT
S: +OK POP3 server at mail.ru signing off

Пример: FTP

Управляющее соединение FTP:

telnet ftp.mgts.by 21
S: 220 Welcome to ByFly FTP service.
C: USER anonymous
S: 331 Please specify the password.
C: PASS test_testerson@mail.ru
S: 230 Login successful.
C: PASV
S: 227 Entering Passive Mode (86, 57, 151, 3, 47, 76)
% ip=86.57.151.3 port=47*256+76=12108
C: LIST
%%%%% Здесь устанавливается информационное соединение №1 %%%%%
S: 150 Here comes the directory listing.
S: 226 Directory send OK.
C: PASV
S: 227 Entering Passive Mode (86, 57, 151, 3, 191, 44)
C: RETR README
%%%%% Здесь устанавливается информационное соединение №2 %%%%%
S: 150 Opening BINARY mode data connection for README (197 bytes).
S: 226 File send OK.
C: QUIT
S: 221 Goodbye.

Пример: FTP

Информационное соединение №1:

telnet ftp.mgts.by 12108

```
-rw-r--r-- 1 0 0 0 Apr 19 2010 2ban_me.html
-rw-r--r-- 1 0 0 197 Jan 27 2010 README
lrwxrwxrwx 1 0 0 18 Jan 27 2010 backports.org -> pub/backports.org/
drwxr-xr-x 2 0 0 23 May 20 2010 blog
drwxr-xr-x 10 0 0 4096 Feb 07 08:48 byfly
```

Пример: FTP

Информационное соединение №2:

telnet ftp.mgts.by 48940

```
Welcome to ByFly public archive (ftp.byfly.by)
located at MGTS, Minsk, Belarus
sponsored by Beltelecom (www.beltelecom.by)
2 x Intel(R) Xeon(R) CPU X3210 @ 2.13GHz
8 Gb RAM, 6 TB SATA Storage
```

RFC 3117

Проектирование прикладных протоколов

Disclaimer: основано на ~~реальных событиях~~ опыте проектирования протокола Blocks eXtensible eXchange Protocol (BXXP).

Возможные пути при проектировании протокола прикладного уровня

1. Попытайтесь найти существующий протокол, который более или менее подходит для ваших нужд.

Возможные пути при проектировании протокола прикладного уровня

1. Попытайтесь найти существующий протокол, который более или менее подходит для ваших нужд.
2. Определите модель для обмена данными поверх инфраструктуры WWW, если более или менее подходит для ваших нужд.

Возможные пути при проектировании протокола прикладного уровня

1. Попытайтесь найти существующий протокол, который более или менее подходит для ваших нужд.
2. Определите модель для обмена данными поверх инфраструктуры WWW, если более или менее подходит для ваших нужд.
3. Определите модель для обмена данными поверх инфраструктуры электронной почты, если она более или менее подходит для ваших нужд.

Возможные пути при проектировании протокола прикладного уровня

1. Попытайтесь найти существующий протокол, который более или менее подходит для ваших нужд.
2. Определите модель для обмена данными поверх инфраструктуры WWW, если более или менее подходит для ваших нужд.
3. Определите модель для обмена данными поверх инфраструктуры электронной почты, если она более или менее подходит для ваших нужд.
4. Разработать свой собственный протокол с 0.

Определение свойств проектируемого протокола

1. Является ли протокол ориентированным на соединения?

Определение свойств проектируемого протокола

1. Является ли протокол ориентированным на соединения?
2. Нужно ли использовать запросы и ответы для обмена сообщениями?

Определение свойств проектируемого протокола

1. Является ли протокол ориентированным на соединения?
2. Нужно ли использовать запросы и ответы для обмена сообщениями?
3. Нужна ли возможность обмена асинхронными сообщениями?

Ориентация на соединения

- Прикладной протокол ориентирован на соединения (т.е. работает поверх TCP или SCTP)
- Без установления соединения – к ним относятся прикладные протоколы, для которых нет нужды в задержках на установление/разрыв соединений, а также поддержке надежной передачи данных.
  При необходимости надежность реализуется средствами прикладного протокола.

Обмен сообщениями

Под сообщением подразумевается простая структура данных, которыми обмениваются слабо-связные системы.
В качестве противовеса можно привести сильно-связные системы, например RPC.
Проблема в том, что слабо- и сильно-связные системы – граничные части целого спектра.

Асинхронный обмен

Синхронная peer-to-peer модель

Модель в стиле «запрос-ответ», где каждая из сторон может являться клиентом и/или сервером.
Большинство протоколов являются синхронными: почта, WWW и т. п.

Часть протоколов не может быть построена по синхронным принципам — например: сетевые файловые системы, системы именования, мультикаст-сообщения и т.п.

Механизмы протокола

Как и какие задачи выполняет протокол?

- Структурирование – заголовки, концевики.

Механизмы протокола

Как и какие задачи выполняет протокол?

- Структурирование – заголовки, концевики.
- Кодирование – представление.

Механизмы протокола

Как и какие задачи выполняет протокол?

- Структурирование – заголовки, концевики.
- Кодирование – представление.
- Отчетность – каким образом описываются ошибки.

Механизмы протокола

Как и какие задачи выполняет протокол?

- Структурирование – заголовки, концевики.
- Кодирование – представление.
- Отчетность – каким образом описываются ошибки.
- Асинхронность – как производится обмен независимыми сообщениями.

Механизмы протокола

Как и какие задачи выполняет протокол?

- Структурирование – заголовки, концевики.
- Кодирование – представление.
- Отчетность – каким образом описываются ошибки.
- Асинхронность – как производится обмен независимыми сообщениями.
- Аутентификация – каким образом стороны идентифицируют и проверяют друг друга.

Механизмы протокола

Как и какие задачи выполняет протокол?

- Структурирование – заголовки, концевики.
- Кодирование – представление.
- Отчетность – каким образом описываются ошибки.
- Асинхронность – как производится обмен независимыми сообщениями.
- Аутентификация – каким образом стороны идентифицируют и проверяют друг друга.
- Конфиденциальность – защита от перехвата или измения данных.

Структурирование (Framing)

Octet-stuffing

Например SMTP – команды заканчиваются переводом каретки (CR-LF)

Структурирование (Framing)

Octet-stuffing

Например SMTP – команды заканчиваются переводом каретки (CR-LF)

Octet-counting

Сообщение разделяется на заголовок и тело; в заголовке указывается размер данных в теле сообщения.

Структурирование (Framing)

Octet-stuffing

Например SMTP – команды заканчиваются переводом каретки (CR-LF)

Octet-counting

Сообщение разделяется на заголовок и тело; в заголовке указывается размер данных в теле сообщения.

Connection-blasting

Например информационное соединение FTP.

Кодирование (Encoding)

Зависит от задачи!
Под кодированием подразумевается формат представления команд и данных.
Например для SMTP он определяется в RFC 822.
В Интернете MIME является «де-факто» стандартом представления данных.

Отчетность (Reporting)

«Теория ответных кодов»

Состоит из 3-х цифр:

1. Успех-неуспех, либо постоянное или временное состояние;
2. ответственная часть системы;
3. идентификация конкретной ситуации.

Код может дополняться текстовым сообщением для человека.

Отчетность (Reporting)

«Теория ответных кодов» имеет 2 нерешенных недостатка:

- коды используются как для ответа на операцию, так и для индикации изменения состояния прикладного протокола;
- код не указывает, кому адресуется ошибка – пользователю, администратору или программисту.

Асинхронность (Asynchrony)

Протокол является асинхронным, если он поддерживает независимые обмены по одному соединению.

Наиболее широко распространенный подход – pipelining, позволяет делать множество запросов к серверу, но требует, чтобы запросы были обслужены последовательно.

Аутентификация (Authentication)

Механизмы специфичны для каждого протокола. Например, FTP использует один механизм, HTTP – другой, а SMTP (классический) не использует его вовсе.

Рекомендуется к использованию SASL (RFC 222 – Simple Authentication and Security Layer) – фреймворк для аутентификации сторон, использующий в том числе OTP (one-time-passwords).

Конфиденциальность (Privacy)

Как правило, подразумевается использование криптографических средств для шифрования данных.

Наиболее часто используемый механизм – TLS/SSL – использует шифрование соединения, а не отдельных объектов.

Конфиденциальность (Privacy)

Можно использовать различные подходы для организации:

- прослушивать на одном порту незащищенные соединения, а на другом – защищенные (SSL);
- прослушивать и защищенные, и незащищенные соединения на одном и том же порту, что требует поддержки со стороны протокола прикладного уровня (TLS).

Примеры протоколов

Механизм	ESMTP	FTP	HTTP 1.1
Структурирование	stuffing	blasting	counting
Представление	RFC 822	бинарное	MIME
Отчетность	3 цифры	3 цифры	3 цифры
Асинхронность	pipelining	нет	pipelining и chunking
Аутентификация	SASL	user/pass	user/pass
Конфиденциальность	SASL или TLS	нет	TLS (SSL)

Свойства протокола

При проектировании протокола необходимо учитывать некоторые свойства:

1. Масштабируемость
2. Эффективность
3. Простоту
4. Расширяемость
5. Устойчивость

Масштабируемость (Scalability)

Хорошо спроектированный протокол должен быть масштабируемым.

Не все протоколы поддерживают асинхронность.
Частая практика – использовать несколько соединений для утилизации канала, однако при этом необходимо учитывать:

- свойства транспортного протокола;
- сервер трактует каждое новое соединение по протоколу прикладного уровня, как независимое (проблема аутентификации и доступа к ресурсам сессии).

Эффективность (Efficiency)

Хорошо спроектированный протокол должен быть эффективным.

Например, использование octet-staffing'а упрощает реализацию протокола, в то время, как octet-counting потребляет немного меньше ресурсов.

Простота (Simplicity)

Хорошо спроектированный протокол должен быть простым.

Хорошее правило для определения насколько протокол прост:

- в хорошо спроектированном протоколе усилия для изменения пропорциональны сложности изменений;
- в плохо спроектированном протоколе необходимо приложить много усилий для любых изменений.

Расширяемость (Extensibility)

Хорошо спроектированный протокол должен быть расширяемым.

Невозможно предсказать заранее, с какими проблемами столкнется протокол.
Таким образом желательно предусмотреть способы расширения функциональности.

Устойчивость (Robustness)

Хорошо спроектированный протокол должен быть устойчивым.

Внезапно! Но для протоколов принцип устойчивости Постела «будь либерален к тому, что принимаешь и консервативен к тому, что отсылаешь» может привести к обратному эффекту.

Проблема в количестве и качестве различных реализаций одного и того же протокола.

Спасибо за внимание!
Вопросы?