References

Java Network Programming, Harold, O’Reilly, pp 67-104


BitTorrent Specification
http://wiki.theory.org/BitTorrentSpecification

Reading

Java Network Programming, Harold, O’Reilly, pp 67-104

BitTorrent Specification
http://wiki.theory.org/BitTorrentSpecification
Parsing

'li2e3:cate'

How to parse the above bencoded string?

Common Parsing Cases

• Fixed length tokens
• Token with size indicated
• Special characters demarking tokens
Fixed-length tokens

Example: Gnutella Message Header

<table>
<thead>
<tr>
<th>Descriptor ID</th>
<th>Payload Descriptor</th>
<th>TTL</th>
<th>Hops</th>
<th>Payload Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>19</td>
<td>22</td>
</tr>
</tbody>
</table>

• Easy to parse
• Variable length data causes problems
Special Characters Demarking Tokens

li2ei345ee

<h1>A Header</h1>

java=properties
file=example

HTTP/1.1 200 OK
Date: Tue, 05 Sep 2000 19:31:14 GMT
Server: Apache/1.3.9 (Unix) PHP/3.0.12
Last-Modified: Mon, 04 Sep 2000 21:03:56 GMT

Special characters indicate start and/or end of a token

In Bencoding lists, integers & dictionaries use this
Issue

What happens if the token contains the special character?

Example: C-based strings

“One line\nSecond Line\n”

\ indicates the next character is special

How to include the \ character in a string

“One line\nStill one line”
Token with size indicated

3:cow4:spam

Note that in BEncoding the size is indicated using a special character.

Why doesn't BEncoding use special character to demark start & end of a string?
Some low level Parsing
Java String methods

"cat;man;ran".split(";"):

Returns an array of String [ "cat", "man", "ran"];

See
http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html#sum
for valid arguments of split().

**StringTokenizer**

```java
parts = new java.util.StringTokenizer("cat,man;ran;,fan", ",; ");
while (parts.hasMoreElements())
{
    System.out.println(parts.nextToken());
}
```

**Output**

cat
man
ran
fan
Some Useful Smalltalk Collection Methods

'cat;man;ran' tokensBasedOn: $;

Result

OrderedCollection ('cat' 'man' 'ran')

'cat. man... ran.'

    piecesCutWhere:
        [:each :next | each = $. and: [next = Character space]]
    do: [:each | Transcript show: each printString; cr]

Result

'cat.'
' man...'
' ran.'
Some Useful Smalltalk Collection Methods

'cat\man\ran'
    runsFailing: [:each | each = $\]
    do: [:each | Transcript show: each; cr]

Result

cat
man
ran

'cat\man\ran'
    runsSatisfying: [:each | each ~= $\]
    do: [:each | Transcript show: each; cr]

Result

cat
man
ran
Java Streams

Java Streams do not have many methods that aid in parsing

read()

Avoid PrintStream – println() is platform dependent

“PrintStream is evil and network programmers should avoid it like the plague”

readLine()

Text claims that readLine() is buggy

Avoid using this method to read data from a socket

Data Input/Output Streams

Are used for binary data

Don’t use unless protocol is binary

If protocol is binary these streams are only good between Java clients and servers
Smalltalk Streams – Some Useful Methods

peek
Answer what would be returned with a self next, without changing position. If the receiver is at the end, answer nil.

peekFor: anObject
Answer false and do not move the position, if the next object is not anObject, or if the receiver is at the end. Answer true and increment the position if the next object is anObject.

skipToAll: aCollection
Skip forward to the next occurrence (if any) of aCollection. If found, leave the stream positioned before the occurrence, and answer the receiver; if not found, answer nil, and leave the stream positioned at the end.

throughAll: aCollection
Answer a subcollection from the current position through the occurrence (if any, inclusive) of aCollection, and leave the stream positioned after the occurrence. If no occurrence is found, answer the entire remaining stream contents, and leave the stream positioned at the end.

upTo: anObject
Answer a subcollection from position to the occurrence (if any, exclusive) of anObject. The stream is left positioned after anObject. If anObject is not found answer everything.

upToAll: aCollection
Answer a subcollection from the current position up to the occurrence (if any, not inclusive) of aCollection, and leave the stream positioned before the occurrence. If no occurrence is found, answer the entire remaining stream contents, and leave the stream positioned at the end.

skipUpTo: anObject
Skip forward to the occurrence (if any, not inclusive) of anObject. If not there, answer nil. Leaves positioned before anObject.

next: anInteger
Read the next anInteger elements
upToAll: and Java

upToAll: is a useful method

    sdsu.io.ChunkReader

http://www.eli.sdsu.edu/java-SDSU/docs/sdsu/io/ChunkReader.html

Reads up to a given string in a stream or string

    read = new sdsu.io.ChunkReader("catEOMmatEOM", "EOM")
    while (read.hasMoreElements() )
    {
        System.out.println( read.readChunk());
    }

Result

cat
mat
Regular Expressions

\((\+|-)?\d+(\.\d*)?((e|E)(\+|-)?\d+)?\)

Java & Smalltalk support Regular expressions

Sun Regular Expression Tutorial
http://java.sun.com/docs/books/tutorial/extra/regex/index.html

See
• java.util.regex package
• java.util.Scanner (JDK 1.5)

VisualWorks
• package Regex11
• Documentation is RxParser class methods
Grammars

\[ <\text{Term}> ::= <\text{Integer}> | <\text{List}> | <\text{Dictionary}> | <\text{String}> \]
\[ <\text{Integer}> ::= i <\text{digit}>^* e \]
\[ <\text{List}> ::= l <\text{Term}>^* e \]
\[ <\text{digit}> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \]
\[ <\text{Dictionary}> ::= d (<\text{String}><\text{Term}>)^* e \]
\[ <\text{String}> ::= n:<\text{character}>^n \]
\[ <\text{character}> ::= a | b | ... \]

Compiler Compilers

- Generate Parsers from a grammar
- JavaCC [https://javacc.dev.java.net/](https://javacc.dev.java.net/)
Some Security

BitTorrent Overview

Step 1
You contact a BitTorrent web site
You get information about a file

Step 2
You contact the BitTorrent Tracker for the file
Tracker provides information about peers with parts of the file

Step 3
You contact peers for parts of the file
How to prevent a Peer from changing the file?

Some Possibilities

• Encrypt the file
• Provide Hash of the file
Encryption

Two basic types of encryption:

• Shared key encryption
  One key both encrypts and decrypts

• Public/Private key encryption
  One key encrypts, another decrypts
**Public/Private Key Encryption**

A public key is something that is well known, i.e. published.

The server will then use its own private key to decrypt the information.

**Basics**

Let
- PubKey = public key
- PriKey = private key
- M = message
- F() = encrypt function

Properties of F

F(PubKey, M) is encrypted

F(PriKey, M) is encrypted

F(PubKey, M) is different than F(PriKey, M)

M == F(PubKey, F(PriKey, M))

M == F(PriKey, F(PubKey, M))

Given F(key, M) it is hard to find M without the other key
How to use Public/Private Key Encryption

Keeping a Secret

Alice has a secret, M, to communicate to Bob in public

Alice computes secret = F(Bob’sPubKey, M)

Alice sends the result to Bob

Bob computes F(Bob’sPriKey, secret ) to get M

It will be hard for anyone else to compute M from secret
Document Authorship Authentication/Digital Signature

Bob has a document, M, that he

• Wants to make public
• Provide proof to everyone that the document came from him

Bob computes Doc = F(Bob’sPrivateKey, M)

Bob publishes Doc & his public Key

Since Bob’s public key, F(Bob’sPubKey, Doc), generates the message, Alice knows the message came from Bob
Common Public/Private Key Encryption Algorithms

RSA (Rivest, Shamir, Adleman)
DSA (Digital Signature Algorithm)

Java & VisualWorks implement these algorithms
RSA

**Public Key**

Key contains n & e where

\[ n = p \times q, \quad p \quad \text{and} \quad q \text{ are primes} \]
\[ e \text{ relatively prime to } (p-1)*(q-1) \]

p & q must be kept secret

**Private Key**

Key contains n & d

\[ d = e^{-1} \mod ((p-1)*(q-1)) \]

**Encrypting**

Let m be a message such that \( m < n \)

Let c be the encrypted message

\[ c = m^e \mod n \]

If m \( \geq n \) then break into block smaller than n and encrypt each block

**Decrypting**

\[ m = c^d \mod n \]
Example

Example is from page 467-8 of Schneier

Alice’s Keys

Let

\[ p = 47. \]
\[ q = 71. \]

Then \[ n = p \times q = 3337 \]
\[ e = 79. \]

Then \[ d = 79^{-1} \mod 3220 = 1019 \]

So Alice’s public key is

\[ n = 3337 \]
\[ e = 79 \]

Alice’s private key is

\[ d = 1019 \]
Sending a Message to Alice

Let $m = 41$

To send the message to Alice we compute

$$c = m^e \mod n = 41^{79} \mod 3337 = 875$$

We send 875 to Alice

Alice computes

$$c^d \mod n = 857^{1019} \mod 3337 = 41$$
Why Won’t Public key Algorithms work for BitTorrent?
One-Way Hash Functions

Let M be a message (sequence of bytes)

A one-way hash function f() such that:

- f maps arrays of bytes to arrays of bytes
- f(M) is always the same length
- Given an M it is easy to compute f(M)
- Given f(M) it is hard to compute M
- Given M it is hard to find N such that f(M) = f(N)

Common One-way Hash Functions

MD5 - Message Digest 5
SHA - Secure Hash Algorithm
Validating Contents of a Message

Alice has a message, M, for Bob

Alice via secure channels sends f(M) to Bob

Alice give M to Trent

Trent delivers M1 to Bob

Bob computes f(M1) and compares it to value from Alice

If f(M1) = f(M) Trent did not modify the message
Using MD5 & SHA in Java

```java
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SampleCode {
    public static void main(String args[]) throws NoSuchAlgorithmException {
        MessageDigest sha = MessageDigest.getInstance("SHA");
        sha.update("Hi mom".getBytes());
        byte[] shaHash = sha.digest();
        System.out.println(new String(shaHash));

        MessageDigest md5 = MessageDigest.getInstance("MD5");
        md5.update("Hi mom".getBytes());
        byte[] md5Hash = md5.digest();
        System.out.println(new String(md5Hash));
    }
}
```
Using MD5 & SHA in Smalltalk

Load the MD5 & SHA parcels & in workspace do

MD5 hash: 'Hi mom'
#[114 83 12 28 50 54 225 209 32 37 154 83 76 243 148 235]

SHA hash: ‘Hi mom’
#[98 21 61 218 186 198 119 88 241 144 60 211 87 250 5 236 219
187 235 16]

Convenience method

(SHA hash: 'Hi mom') asHexString
'62153DDABAC67758F1903CD357FA05ECDBBEBB10'