# 新月プロトコル/0.5 ドラフト#2

新月プロジェクト

2005 年 8 月 11 日

## 1 はじめに

この文書は P2P 匿名掲示板-新月-のノード間の通信プロトコルについて定義する。新月の匿名掲示板ネットワークは多数のノードの相互接続によって構成される。ノード間で相互に通信が行えるようプロトコルが規定されている必要がある。新月ネットワークのノード間で交換されるメッセージは新月プロトコルに従わなければならない。この文書で定義する新月プトロコルのバージョンは 0.5 である。

## 2 目的

この文書の目的はノードが相互にメッセージを交換できるように、プロトコルを規定することである。そして、新月プロトコルの実装アプリケーションの開発ができる新月プロトコルの文書を提供することである。

## 3 新月プロトコルの歴史

新月プロトコル/0.5 が作成される以前も、新月プロトコルの文書は公開されていた。しかし、当時の文書は、すでに存在していた実装の shinGETsu の補足的な資料でしかなかった。新月がインターネット上でに広がるにつれて、新月プロトコルを元に実装された、様々な新月の実装が登場し始めた。各実装は一見、相互にネゴシエーションが行えていたと思えた。しかし、実装者ごとに新月プロトコルの解釈が異なっていたため、新月プロトコルを満たしきれていない実装が現れた。また、署名のシステムは文書化されていたものの不透明な点が数多くあった。そこで、これまで作成されてきた資料を元に、包括的

な新月プロトコルの文書を作成する必要があった。

# 4　新月ネットワーク

## 4.1　新月ネットワークの構成

新月のネットワークはノード間の接続によって構成される。図 1 は新月のネットワーク構成の概念図である。図中の点は、新月のネットワークのノードである。各々のノードはネットワーク上で一意な名前を持つ。この一意な名前をノード名という。

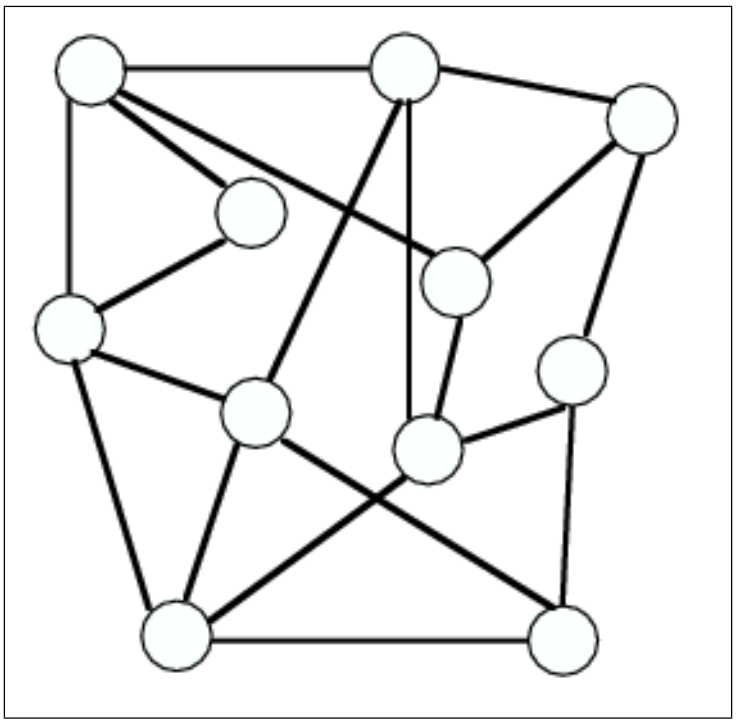

図 1　新月のネットワーク構成

ネットワーク上の各々のノードは、他のノードと通信を行うために他のノード名を保持する。あるノードが保持している他のノード名のことを隣接ノードと呼ぶ。隣接ノード同士は互いに相手のノード名を保持していることが望ましいが、保証される必要はない。

ノード間の通信は HTTP/1.0 または HTTP/1.1[1] の仕様に従って行う。ノード間のリクエスト・レスポンスのメッセージは GET メソッドの規約に従わなければならない。ノード間のレスポンスのメッセージの文字エンコーディングは UTF-8 でなければならない。

新月のノード間の通信コマンドは最終的に URL[2] として構築される。したがって、他のノードに送信するコマンドは URL の仕様に従わなければならない。

## 4.2 ノード

新月プロトコルを実装しているプログラムをノードという。新月ネットワークに接続するノードは新月プロトコルを満たすよう正確に実装しなければならない。プログラムの新月プロトコルの実装にあたっては、HTTP、URL の仕様も満たさなければならない。ノードはネットワーク上で一意の名前を持つ。これをノード名 (4.1) といった。

ノード名の形式は次のように定義する。

ホスト名:ポート番号/パス名

ここでホスト名は DNS 名または IPv4 の IP アドレスとする。ポート番号は 10 進表現の整数である。

新月プロトコルは、ノード名の構成要素であるポート番号を指定しない。各々のノードはノード名に使用するポート番号を自由に設定することができる。ノード名に使用するポート番号が 1024 未満の場合は、ウェルノウンポート番号 [3] に従うべきである。

## 4.3 メッセージ

ノード間でリクエスト・レスポンス時に交換する情報をメッセージという。メッセージの内容は多岐にわたる。例えば、/ping コマンドに対するレスポンスである通信相手のノードの IP アドレスがある。そして、/get、/head コマンドで交換するファイルもある。

通信相手ノードにリクエストメッセージを送信し、レスポンスメッセージを受信するノードは、圧縮されていない通常のメッセージに加えて、gzip 形式で圧縮しているメッセージも解釈できることが望ましい。ただし、gzip 形式で圧縮したメッセージをノードに送信する場合、通信相手ノードはリクエストのメッセージヘッダを解析し、ノードが gzip 形式で圧縮されたメッセージを解釈できるかどうか判断しなければならない。もし、判断後ノードが gzip 形式で圧縮されたメッセージが解釈可能であれば、gzip 形式で圧縮したメッセージを送信する。

圧縮されているメッセージを解釈できるノードは、できるだけ通信相手ノードにそれが可能な旨を通知すべきである。メッセージの圧縮を行うことにより、メッセージを圧縮することができ、圧縮通信の利点を享受できる。

# 5　ノード間のプロトコルコマンド

ノード間の通信はプロトコルコマンドの送信することによって行う。プロトコルコマンドはノード名から生成した URL のパス部の末尾に付加する。プロトコルコマンドを付加した URL の例を示す。

http://example.com:8000/server.cgi/ping

次に、ノード間で通信を確立するためのプロトコルコマンドを示す。ここで\n は改行コード (0x0a) を表わす。

/ping

ノードは「PONG\n 相手ノードの IP アドレス」を返す。

/node

ノードはノード自身が接続しているノードを 1 つ選択し、そのノード名を返す。

/join/ノード名

ノードは、相手ノードが指定するノード名が有効であることを /ping によって確かめ、自ノードのノード保持リストに加え、「WELCOME」または「WELCOME\n 別のノードのノード名」を返す。 相手ノードが指定するノード名が有効でないときはそれ以外のものを返す。 相手ノードのノード名は、接続しようとしている相手ノード自身を示すものでなければならない。 相手ノードのノード名のホスト名は、省略することができる。 ノード名のパスは/を + に置き換えたものとする。 相手ノードにはノード保持リストに加えてもらい、レスポンスで指定したノードに接続することを期待する。

/bye/ノード名

ノードは相手ノードが指定するノード名が有効であることを /ping によって確かめ、自ノードのノード保持リストから削除し、 「BYEBYE」を返す。

/have/ファイル名

ファイル名のファイルを持っていれば「YES」、そうでなければ「NO」を返す。

/get/ファイル名/時刻引数

ファイル名で指定したファイルの、時刻引数を満たすレコードを返す。 時刻引数は次のうちどれかひとつである。

「時刻」　指定した時刻のレコード
「-時刻」　指定した時刻以前のレコードすべて
「時刻-」　指定した時刻以降のレコードすべて
「時刻-時刻」　指定した時刻の間のレコードすべて
「時刻/識別子」　指定した時刻と識別子のレコード

/head/ファイル名/時刻引数

/get と同様のレスポンスである。しかし、各レコードの時刻と識別子のみを返す。

/have/ファイル名

ファイル名のファイルを持っていれば「YES」、そうでなければ「NO」を返す。

/update/ファイル名/時刻/識別子/ノード名

ノードにファイルが更新されたことを知らせる。 もしすでにファイルの更新に関する処理をしているのなら何もしない。 自分の持っているファイルならば、それを更新したのち、 ノード名を自ノードのノード名に書き換えて接続しているノードにも通知する。 そうでなければそのまま、送信されてきたプロトコルコマンドを接続しているノードに転送する。 ノード名のパスは/を + に置き換えたものとする。 /join と同様にホスト名を省略できる。

/

ノード固有のメッセージを表示する。 通信には使わないので、何を出力してもよい。

### 5.1 ファイルの伝播

ノードは、ファイルの内容に変更が生じたときに、変更された事実を/update コマンドを使用して他のノードへ通知することができる。ファイルの変更があったことを通知するかどうかの判断の処理は、実装によって異なる。

## 6 ファイル

### 6.1 ファイルの定義

新月プロトコルにおけるファイルとは、/get 命令または/head 命令に対するレスポンスのメッセージボディのことを指す。ファイルはレコードの集合である。ファイル内のレコードの順番に制約はない。

### 6.2 ファイル名

ファイル名は、半角英数字とアンダースコア (_) のみで構成する。ファイル名は prefix_basename という形式である。 prefix、basename に使える文字は、半角英数字のみである。prefix はファイルの種類を表す。basename は、文字エンコーディングが UTF-8 で表現されるファイル名を、16 進数表現に変換した文字列である。16 進数表現に用いることができる文字は半角英字の A-F、半角数字のみである。

### 6.3 レコード

レコードとはファイルを構成する行のことである。レコードの形式は、次のように定義する。

タイムスタンプ<>識別子<>本文

識別子は本文の MD5 値である。レコードの要素は、<>で区切る。タイムスタンプは、新月プロトコルで定義する基準時刻 (7) から経過した整数値の秒数を表す。

### 6.4 本文の書式

本文を構成する要素は「<>」で区切る。 本文は複数の名前付きフィールドで構成される。名前付きフィールドの形式は「名前:値」である。名前付きフィールドの名前に使える文字は、半角英数字とアンダースコア (_) のみである。名前付きフィールドの名前は重複してはならない。 名前付きフィールドの stamp, id は予約されており、それぞれタイムスタンプ、識別子を表わす。本文での名前付きフィールドの出現順番は問わない。

名前付きフィールドのフィールド値には「<」「>」が含まれてはならない。 例外として「<文字列>」(タグと呼ぶ) を含むことができる (文字列は長さが 1 文字以上であって、「<」「>」が含まれてはならない)。 どのようなタグが使えるのかはアプリケーションが定める。

## 7 時刻

新月プロトコルで用いる基点時刻は「1970 年 1 月 1 日午前 0 時 (グリニッジ標準時)」である。ある時刻は、基点時刻を基準として表される整数値の秒で表す。

## 8 ブラケットリンク

アプリケーションは、ファイルの内容をなんらかのデータ形式に変換し、結果を出力する場合において、本文に次に述べるようなブラケットリンクがあるときは、文字列が指定するリンクを作成しなければならない。

アプリケーション type が存在するとする。標準的なブラケットリンクの形式は [[/type/文字列]] である。文字列の形式はアプリケーション (type) が定義する。つまり、文字列の解釈はアプリケーションに委ねられる。/type は省略することができる。/type を省略したブラケットリンクの形式である [[文字列]] は、本文が記録されているファイル形式の type が省略されていると考える。

現在、アプリケーションとして定義されている thread、note のブラケットリンクの作成例を次に示す。

[[すれっど]]

本文が記録されているファイル形式が thread なら、thread の「すれっど」へリンクする。

[[/list/りすと]]

list の「りすと」へリンクする。

[[すれっど/7889e7db]]

本文が記録されているファイル形式が thread なら、thread「すれっど」の識別子「7889e7db」を持つレコードへリンクする。

[[/thread/すれっど/7889e7db]]

thread「すれっど」の識別子「7889e7db」を持つレコードへリンクする。

# 9 署名

署名の暗号システムは、公開鍵暗号方式に従う。暗号化のアルゴリズムは RSA である。メッセージダイジェストのアルゴリズムは MD5 である。

## 9.1 文字列と多倍長整数間の変換アルゴリズム

文字列と多倍長整数間は、文字列の最下位 6 ビットから Base64 アルゴリズムの変換テーブルに準拠して変換する。文字列の先頭文字が整数の最下位 6 ビットに対応し、文字列の最後尾が最上位ビットに対応する。このとき、想定される文字列の字数が足りない時は、数値の末尾のビットに「0」を補充する。つまり、数値から文字変換時に文字列の末尾に A を付加する。

## 9.2 鍵生成アルゴリズム

素数テストにはミラーテストを使用する。最初の 10 個の素数に対して、テストを通過した擬素数を素数とみなす。公開乗数 $e$ は 65537 とする。トリップ生成文字列から、生成する素数を $p$、$q$ とする。トリップ生成文字列\$key とはクライアントから取得する署名文字列である。

### ステップ 1 ハッシュの計算

トリップ生成文字列を hashs とする。ここでの + 演算子は文字列の連結演算を意味する。MD5 の計算アルゴリズムを md5() と定義する。

$$hashs = md5(\$key) + md5(\$key +' pad1') + md5(\$key +' pad2') + md5(\$key +' pad3') \quad (1)$$

**ステップ** 2 **ハッシュ文字列から** p、q **への変換**

hashs の 0 から 27 バイト、28 から 63 バイトをそれぞれ p、q に代入する。このとき、リトルエンディアンで処理する。つまり、hashs の最上位バイトが p(q) の最下位バイトに対応するように処理する。

p については、

$$p.num[0] - p.num[6] \div hashs[0-27] \text{ (28 バイト)} \quad (2)$$

である。q については、

$$q.num[0] - q.num[8] \div hashs[28-63] \text{ (36 バイト)} \quad (3)$$

である。

**ステップ** 3 p、q **の前処理**

ステップ 2 で求めた p、q に対して、次のふたつの演算を行う。 p が 216 ビットで表される小さな因数になるのを防ぐために (A) の演算をする。 q が 280 ビットで表される小さな因数になるのを防ぐために (B) の演算をする。これにより、公開鍵 n が 496 ビットで表される数値を下回ることを抑制する。

(A). p.num[6] の下から 24 ビット目 (216 ビット目、第 215 ビット) を 1 にする。
(B). q.num[8] の下から 24 ビット目 (280 ビット目、第 279 ビット) を 1 にする。

**ステップ** 4 p、q **を** RSA **に適する素数への変換**

q を q 以上で最小の擬素数とする。 p を p 以上で最小の擬素数とする。次の関係が成立する p、q を求める。(p - 1)(q - 1) と e が互いに素で、かつ

$$t = 0\text{x}7743 \tag{4}$$
$$de \equiv 1 \ (mod \ (p-1)(q-1)) \tag{5}$$
$$n = pq \tag{6}$$

上記 3 式の関係が成立するとする。上記 3 式が成立する $t$、$d$、$n$ に対し

$$t^{ed} \equiv t \ (mod \ n) \tag{7}$$

上記の関係を満たす $p$、$q$ が得られるまで、 $p = p + 2$、$q = q + 2$ として (5) 式から、(7) 式が成立するまで繰り返す。

**ステップ 5 鍵の命名**

ステップ 4 で生成した $n$ を公開鍵、$d$ を秘密鍵と称する。文字列に変換する際は上記変換則 (9.1) を用いて 86 文字とする。

## 9.3 署名対象文字列

署名対象文字列を次のように定義する。

署名対象文字列 = field1:value<>field2:value2<>...

このフィールド書式は新月プロトコルの本文の書式に従う。フィールドの出現順序は target フィールドで指定された順番に従う。

## 9.4 署名アルゴリズム

RSA 暗号化 $m^d \equiv c \ (mod \ n)$ に使用する $m$ を求める。署名対象ハッシュ文字列をとする。署名対象ハッシュ文字列の長さは 64 バイトである。署名対象ハッシュ文字列 Mes は次のようにして求めることができる。

Mes[0-63] = md5(署名対象文字列) = md5(field1:value1<>field2:value2<>...)

署名対象ハッシュ文字列 Mes の長さが 64 バイトに満たない場合、Mes を数値へ変換する時に、Mes の空きバイトには 0 を仮定する。 64 バイトを超える場合、超えた部分は無視する。文字列から数値への対応は次式で定義する。

$$m.num[0-15] \div Mes[0-63] \text{ (64 バイト)} \tag{8}$$

ただし、リトルエンディアンで処理する。つまり、Mes の最上位バイトは m の最下位バイトである。

### 9.5 レコードの削除

ファイル内の削除信号を察知したノードは、削除信号が示すレコードを削除することができる。削除信号とは、削除対象のレコードが書き込まれた時刻を指す remove_stamp 名前付きフィールドと、削除対象のレコードの識別子を指す remove_id 名前付きフィールドを本文の構成要素として持つレコードである。もちろん、削除信号は通常のレコードとなんらかわるものではない。削除信号は、ノード間のメッセージ交換を通じて伝播してする。削除信号を含むファイルを受信したノードは、削除信号が示すレコードを削除することができる。

## 10 プラグインとアプリケーション

ユーザに機能を提供するプログラムをプラグインと呼ぶ。プラグインのうち、ファイルの書式を定義するプラグインを アプリケーションと呼ぶ。現在、アプリケーションには list、thread、note の 3 種類がある。

# 11　プラグイン

## 11.1　プラグインとアプリケーション

プラグインとは、なんらかの機能を手供するプログラムである。特にレコード本文の要素を定義するプラグインをアプリケーションという。この文書執筆時、存在するアプリケーションは list、thread、note である。

## 11.2　フィールド

アプリケーションは、レコード本文の要素として複数の名前付きフィールドを持つことができる。名前付きフィールドは、アプリケーションごとに定義を行ってもよい。名前付きフィールドの形式を次に示す。

field_name:field_value

field_value は名前付きフィールドの名前をである。filed_value は名前付きフィールドの値である。なお、フィールドの区切り文字は「<>」である。

## 11.3　アプリケーション間で共通する名前付きフィールド

pp 名前付きフィールドには、アプリケーション間で共通のフィールドがある。次に、アプリケーション間で共通する名前付きフィールドを示す。

### 署名に関連する名前付きフィールド

署名に関連する名前付きフィールドとその説明を示す。

pubkey
: 署名の公開鍵を記す。

sign
: 署名を記す。

target
: どのフィールドを署名するか「,」で区切って並べる。

**削除に関連する名前付きフィールド**

削除に関連する名前付きフィールドのフィールド名とその説明を記す。

remove_stamp
: 同じファイル中で削除する stamp を指定する。p

remove_id
: 同じファイル中で削除する id を指定する。

## 11.4 アプリケーションの名前付きフィールド

3 種類のアプリケーションである、list、thread、note で定義している名前付きフィールドを示す。

list

type
: リンク先のアプリケーション (ファイルの種類)

link
: リンクの名前

name
: 投稿者の名前

body
: 本文

ただし、タグは改行を表わす「<br >」のみが使える。「<BR >」や「<br / >」は不可。

thread

attach
: 添付ファイルの拡張子

suffix
: 付ファイルを base64 エンコードしたもの

mail
: 投稿者のメールアドレス

name

投稿者の名前

body

本文

ただし、タグは改行を表わす「<br >」のみが使える。「<BR >」や「<br / >」は不可

note

base_stamp

編集したもとの記事の stamp

base_id

編集したもとの記事の id

body

本文

ただし、タグは改行を表わす「<br >」のみが使える。「<BR >」や「<br / >」は不可

# 参考文献


[1] IEFT:Hypertext Transfer Protocol – HTTP/1.1,
http://www.ietf.org/rfc/rfc2616.txt

[2] IEFT:Uniform Resource Locators (URL),
http://www.ietf.org/rfc/rfc1738.txt

[3] IANA:WELL KNOWN PORT NUMBERS,
http://www.iana.org/assignments/port-numbers