```
BIP: 14
Layer: Peer Services
Title: Protocol Version and User Agent
Author: Amir Taaki <genjix@riseup.net>
        Patrick Strateman <bitcoin-bips@covertinferno.org>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0014
Status: Final
Type: Standards Track
Created: 2011-11-10
Post-History: 2011-11-02
```

In this document, bitcoin will be used to refer to the protocol while Satoshi will refer to the current client in order to prevent confusion.

Past Situation

Bitcoin as a protocol began life with the Satoshi client. Now that the community is diversifying, a number of alternative clients with their own codebases written in a variety of languages (Java, Python, Javascript, C++) are rapidly developing their own feature-sets.

Embedded in the protocol is a version number. Primarily this version number is in the "version" and "getblocks" messages, but is also in the "block" message to indicate the software version that created that block. Currently this version number is the same version number as that of the client. This document is a proposal to separate the protocol version from the client version, together with a proposed method to do so.

Rationale

With non-separated version numbers, every release of the Satoshi client will increase its internal version number. Primarily this holds every other client hostage to a game of catch-up with Satoshi version number schemes. This plays against the decentralised nature of bitcoin, by forcing every software release to remain in step with the release schedule of one group of bitcoin developers.

Version bumping can also introduce incompatibilities and fracture the network. In order that the health of the network is maintained, the development of the protocol as a shared common collaborative process requires being split off from the implementation of that protocol. Neutral third entities to guide the protocol with representatives from all groups, present the chance for bitcoin to grow in a positive manner with minimal risks.

By using a protocol version, we set all implementations on the network to a common standard. Everybody is able to agree within their confines what is protocol and what is implementation-dependent. A user agent string is offered as a 'vanity-plate' for clients to distinguish themselves in the network.

Separation of the network protocol from the implemention, and forming development of said protocol by means of a mutual consensus among participants, has the democratic disadvantage when agreement is hard to reach on contentious issues. To mitigate this issue, strong communication channels and fast release schedules are needed, and are outside the scope of this document (concerning a process-BIP type).

User agents provide extra tracking information that is useful for keeping tabs on network data such as client implementations used or common architectures/operating-systems. In the rare case they may even provide an emergency method of shunning faulty clients that threaten network health- although this is strongly unrecommended and extremely bad form. The user agent does not provide a method for clients to work around and behave differently to different implementations, as this will lead to protocol fracturing.

In short:

- Protocol version: way to distinguish between nodes and behave different accordingly.
- User agent: simple informational tool. Protocol should not be modified depending on user agent.

Browser User-Agents

RFC 1945 vaguely specifies a user agent to be a string of the product with optional comments.

```
 Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.6) Gecko/20100127 Gentoo Shiretoko/3.5.6
```

User agents are most often parsed by computers more than humans. The space delimited format, does not provide an easy, fast or efficient way for parsing. The data contains no structure indicating hierarchy in this placement.

The most immediate pieces of information there are the browser product, rendering engine and the build (Gentoo Shiretoko) together with version number. Various other pieces of information as included as comments such as desktop environment, platform, language and revision number of the build.

Proposal

The version field in "version" and "getblocks" packets will become the protocol version number. The version number in the "blocks" reflects the protocol version from when that block was created.

The currently unused sub_version_num field in "version" packets will become the new user-agent string.

Bitcoin user agents are a modified browser user agent with more structure to aid parsers and provide some coherence. In bitcoin, the software usually works like a stack starting from the core code-base up to the end graphical interface. Therefore the user agent strings codify this relationship.

Basic format:

```
/Name:Version/Name:Version/.../
```

Example:

```
 /Satoshi:5.64/bitcoin-qt:0.4/
/Satoshi:5.12/Spesmilo:0.8/
```

Here bitcoin-qt and Spesmilo may use protocol version 5.0, however the internal codebase they use are different versions of the same software. The version numbers are not defined to any strict format, although this guide recommends:

- Version numbers in the form of Major.Minor.Revision (2.6.41)
- Repository builds using a date in the format of YYYYMMDD (20110128)

For git repository builds, implementations are free to use the git commitish. However the issue lies in that it is not immediately obvious without the repository which version precedes another. For this reason, we lightly recommend dates in the format specified above, although this is by no means a requirement.

Optional -r1, -r2, ... can be appended to user agent version numbers. This is another light recommendation, but not a requirement. Implementations are free to specify version numbers in whatever format needed insofar as it does not include (,), : or / to interfere with the user agent syntax.

An optional comments field after the version number is also allowed. Comments should be delimited by brackets (...). The contents of comments is entirely implementation defined although this BIP recommends the use of semi-colons ; as a delimiter between pieces of information.

Example:

```
/BitcoinJ:0.2(iPad; U; CPU OS 3_2_1)/AndroidBuild:0.8/
```

Reserved symbols are therefore: / : ( )

They should not be misused beyond what is specified in this section.

- / separates the code-stack
- : specifies the implementation version of the particular stack
- (and) delimits a comment which optionally separates data using ;

Timeline

When this document was published, the bitcoin protocol and Satoshi client versions were currently at 0.5 and undergoing changes. In order to minimise disruption and allow the undergoing changes to be completed, the next protocol version at 0.6 became peeled from the client version (also at 0.6). As of that time (January 2012), protocol and implementation version numbers are distinct from each other.