Network Working Group
Internet-Draft
Intended status: Informational
Expires: 23 September 2026

T. Ribeiro Silva
March 2026

Crystal Network Protocol (CNP) Version 1.0
draft-ribeiro-silva-cnp-00

Abstract

CNP is a decentralized general purpose network protocol, meant to allow the seamless production of decentralized yet internet dependent applications.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 September 2026.

Table of Contents

1. Introduction 2
2. Terminology 2
3. Format 2

3.1. Key 2
3.2. Cipher Text and Nonce 2
3.3. Ciphers and Hashes 3
3.4. Method 3
4. Node 3
5. No Response Protocol 4
6. Use of TCP 4
7. Packet Lifetime 4
8. Security Considerations 4
9. Normative References 4
Author’s Address 4

1. Introduction

At this point in time, web applications (critically, ones regarding communication) are highly centralized and often rely on a single entity. As a means to make decentralized applications easier to implement, this protocol defines a standard for such.

2. Terminology

Over the text, the words "crystal", "node", "packet", and "blob" will be used. In the said order, to describe an inter-connected network, a member of the crystal, a blob with the CNP format, a contiguous sequence of bytes. Also, this document follows the [RFC2119] definitions.

3. Format

A packet is a blob with a fixed-length of one kibibyte, the starting byte signaling the method, 64 bytes being a key, 12 bytes for nonce, and the remaining 947 bytes being the cipher text. The cipher text being preferably AES-GCM-SIV derived, the nonce stored at its field.

3.1. Key

This field is used to identify a packet, which has to be derived from a hash, which is preferably blake2b. There is no verification regarding the hash algorithm used, specifically because since the key is for identification purposes, if the implementation uses argon2id or does not use a hashing algorithm, what would happen is a higher likelihood of collisions, thus poor behavior.

3.2. Cipher Text and Nonce

Cipher text is a preferably encrypted blob, and the nonce field SHOULD be used to store a nonce.

3.3. Ciphers and Hashes

The use of an encryption method is highly recommended to maintain security, and it is recommended to use AES-GCM-SIV since it has built-in authentication and nonce mis-use protection.

Even so, the protocol itself is not dependent on the implementation that derived the packet, as the server is agnostic about the contents of the cipher text and nonce. As for the key, it is used as a blob.

3.4. Method

If 0xFF the method means "insert", if 0x00 "get". Insert being used to clone a packet from node-to-node across the crystal, and "get" to fetch a packet.

4. Node

A node SHALL keep IPv6s for running the protocol, MAY have a method to keep these persistent, the data-structure SHOULD have a limit of items, the node MAY store IPv6s it contacts in any shape or form (preferably yes, for the crystal to grow), the node MAY drop IPv6s it identifies as permanently shut down, and the node SHOULD be listening to port 3017 by default and accepting connections.

When a node registers a packet by itself, it sends the packet to all the nodes it has (IPv6s) with the method "insert". Those which MAY retransmit their packets to any other node.

Upon receipt of an "insert", the packet SHOULD be stored in the node in some manner, but MAY refuse if the node assumes that doing so would be problematic for it.

When a node encounters itself on a position of needing a packet, it sends a "get" to all nodes it holds. This request MAY be retransmitted to their own nodes.

Upon receipt of a "get", the node SHALL seek if it has a packet with the key of the received packet. If it does, the node SHALL "insert" it on the sender.

If a node receives a packet, and it already has one with the same key, the new one is ignored.

All implementations that by any shape or form retransmit a packet SHOULD limit retransmission rate to prevent flood behavior.

The retransmission of a "get" SHOULD be sequential.

5. No Response Protocol

Given that "get" and "insert" neither require a response to work. If a "get", after sending this request to all of its nodes, and the crystal does not have the requested key, instead of responding "not found", the node simply does not receive a packet with this key. Same for insert, it simply can be inserted, no guarantee.

6. Use of TCP

Used to guarantee the delivery of packets. For sending a packet, a node SHALL make a connection with another, send the packet, and either one MAY close the connection right after.

7. Packet Lifetime

All packets SHALL be deleted after an hour, but MAY be deleted before.

8. Security Considerations

The protocol by itself is insecure, it does not guarantee any sort of security. Even so, it is recommended (and explicitly said so) that encryption is used on the data blob (mentioned as cipher text), and for the key to be derived from a cryptographic hash. These do increase the safety significantly.

Another problem could be DoS related, but these can be mitigated with proper firewalls, and out-of-spec solutions.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

Author’s Address

Theo Ribeiro Silva
Email: theo.ribeiro.5000@protonmail.com, theo.ribeiro.5000@gmail.com