RFC XXXX: Local Social Networking Protocol (LSNP)

Category: Experimental
Status: Draft
Author: Ann Franchesca Laguna
Date: June 20, 2025

Table of Contents

---

1. Abstract

The Lightweight Social Networking Protocol (LSNP) is a decentralized communication framework that facilitates social networking functionality within local area networks (LANs) without relying on centralized servers or internet connectivity. LSNP leverages the simplicity and efficiency of plaintext, key-value structured messages, and utilizes UDP broadcast and unicast communication methods. This protocol supports a range of peer-to-peer interactions, including dynamic peer discovery, real-time messaging, ad hoc file sharing, and seamless group engagement. Additionally, LSNP enables the deployment of interactive applications such as games and collaborative tools within local networks, offering a low-overhead and scalable approach to localized social networking environments.

---

2. Introduction

The Lightweight Social Networking Protocol (LSNP) is engineered to operate effectively within disconnected or minimally resourced environments where conventional network infrastructure is constrained or absent. Target deployment scenarios include educational institutions, post-disaster zones, rural communities, and temporary installations such as exhibitions or public events. LSNP facilitates peer-to-peer communication and collaboration within local area networks (LANs), offering essential services such as identity dissemination, messaging, content

distribution, and lightweight interaction. Its low overhead and decentralized architecture enable rapid deployment and efficient operation without reliance on centralized servers or persistent internet connectivity.

LSNP adopts a simple, human-readable protocol based on plaintext key-value messaging, utilizing UDP broadcast and unicast for intra-network communication. This design supports dynamic peer discovery, real-time information exchange, and user-driven interactivity within LAN environments. The protocol's modularity and minimal resource footprint make it suitable for constrained hardware and software platforms, promoting accessibility and extensibility. LSNP presents a robust solution for local communication infrastructures through its emphasis on simplicity and resilience, enhancing digital participation and networked collaboration in resource-limited contexts.

---

3. Terminology

- **Peer**: A device participating in LSNP
- **Profile**: User's identity message
- **Post**: Public message to all peers
- **DM**: Private message to one peer
- **Token**: Expiring text credential
- **Chunk**: Part of a multi-packet file
- **TTL**: Time-to-live in seconds
- **GameID**: Identifier for a game session
- **GroupID**: Identifier for a group

---

4. Protocol Overview

- **Transport**: UDP (broadcast & unicast)
- **Port**: 50999
- **Broadcast Address**: Broadcast Address of the Network
- **Encoding**: Plain UTF-8 text, key-value format
- **Separator**: KEY: VALUE
- **Terminator**: Blank line (\n\n) [Each message should end with a (\n\n)]

---

5. Message Types

5.1 PROFILE

Announces a user's identity to the network, including user ID, display name, status, and optional profile picture. This message is periodically broadcast to let peers discover and update contact information. Profile pictures are transmitted in base64 format with MIME type metadata. Any host who does not support profile pictures should be able to accept message that have AVATAR_* keys but will just disregard the AVATAR_* keys.

Format:

TYPE: PROFILE
USER_ID: dave@192.168.1.10
DISPLAY_NAME: Dave
STATUS: Exploring LSNP!
AVATAR_TYPE: image/png
AVATAR_ENCODING: base64
AVATAR_DATA: iVBORw0KGgoAAAANSUhEUgAAAAUA…

Field Descriptions:

- TYPE: Always PROFILE.
- USER_ID: Unique sender address [username@ipaddress]
- DISPLAY_NAME: User's public name.
- STATUS: Short message or mood.
- AVATAR_TYPE: MIME type of the image. (optional)
- AVATAR_ENCODING: Currently always base64.(optional)
- AVATAR_DATA: Image data encoded as base64 (under ~20 KB).(optional)

Non-verbose Printing:

Show only the display_name and status. Optionally, show profile picture.

5.2 POST

Sends a public broadcast message to all followers. Non-followers should not receive any posts. Posts are typically used for social updates or announcements. The TTL defines how long the post remains valid, and the token ensures the sender is authorized to broadcast.

Format:

TYPE: POST
USER_ID: dave@192.168.1.10
CONTENT: Hello from LSNP!
TTL: 3600
MESSAGE_ID: f83d2b1c
TOKEN: dave@192.168.1.10|1728941991|broadcast

Field Descriptions:

- TIMESTAMP: Unix Timestamp
- TTL: Expiration window in seconds. [Default is 3600 but should be changeable upon runtime]
- TOKEN: Scoped for broadcasting. [user_id|TIMESTAMP+TTL|broadcast]
- MESSAGE_ID: Identifier of original message. [Randomly generated 64-bit binary in hex format]

In LSNP, timestamps are represented as UNIX timestamps — the number of seconds that have passed since January 1, 1970 (UTC).

Non-verbose Printing:

Show only the display_name (user_id if display name is not recorded) and content. Optionally, show profile picture.

5.3 DM

Delivers a private message directly to a single recipient. It includes sender and recipient identifiers, content, and a token authorizing the communication. DMs are not visible to other peers and should be sent over unicast.

Format:

TYPE: DM
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
CONTENT: Hi Bob!
TIMESTAMP: 1728938500
MESSAGE_ID: f83d2b1d
TOKEN: alice@192.168.1.11|1728942100|chat

Field Descriptions:

- FROM, TO: IDs of sender and receiver.
- TOKEN: Scoped for chat. [user_id|TIMESTAMP+TTL|chat]

Non-verbose Printing:

Show only the display_name (user_id if display name is not recorded) and content. Optionally, show profile picture.

5.4 PING

Used for basic presence signaling, typically sent periodically over broadcast. It informs peers that the sender is active and listening. It may trigger a PROFILE response from recipients. Note that this should be automatically sent by the peer without user intervention.

Format:

TYPE: PING
USER_ID: alice@192.168.1.11

Field Descriptions:

- USER_ID: who is alive.

Non-verbose Printing:

Do not display anything

5.5 ACK

Acknowledges receipt of a message or content, typically identified by a MESSAGE_ID. Useful for confirming delivery of DMs, file chunks, or game actions. It helps manage flow control and retransmission in the UDP environment. Note that this should be automatically sent by the peer without user intervention.

Format:

TYPE: ACK
MESSAGE_ID: f83d2b1c
STATUS: RECEIVED

Field Descriptions:

- MESSAGE_ID: Identifier of original message.
- STATUS: e.g., RECEIVED.

Non-verbose Printing:

Do not display anything

5.6 FOLLOW

Indicates a user's interest in subscribing to another user's updates. It functions similarly to social network "follow" actions and is authorized by a scoped token. Clients can use this to filter or prioritize content.

Format:

TYPE: FOLLOW
MESSAGE_ID: f83d2b1c
FROM: alice@192.168.1.11
TO: dave@192.168.1.10
TIMESTAMP: 1728939000
TOKEN: alice@192.168.1.11|1728942600|follow

Field Descriptions:

- ACTION: Implicit by TYPE (FOLLOW vs UNFOLLOW).
- TOKEN: Scoped as follow. [user_id|TIMESTAMP+TTL|follow]

Non-verbose Printing:

“User alice has followed you”

5.7 UNFOLLOW

Signals the intent to unsubscribe from another user's updates. Like FOLLOW, it requires a token and records the event with a timestamp. It allows users to manage their social graph on a decentralized basis.

Format:

TYPE: UNFOLLOW
MESSAGE_ID: f83d2b1c
FROM: alice@192.168.1.11
TO: dave@192.168.1.10
TIMESTAMP: 1728939050
TOKEN: alice@192.168.1.11|1728942650|follow

Field Descriptions:

- ACTION: Implicit by TYPE (FOLLOW vs UNFOLLOW).
- TOKEN: Scoped as follow. [user_id|TIMESTAMP+TTL|follow]

Non-verbose Printing:

“User alice has unfollowed you”

5.8 FILE_OFFER

Announces the intention to send a file to another user, including metadata like filename, size, type, and a description. The recipient can accept or ignore the offer. If the file is ignored, all packets received pertaining to the file will be ignored. It sets up a context for subsequent FILE_CHUNK messages.

Format:

TYPE: FILE_OFFER
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
FILENAME: photo.jpg
FILESIZE: 204800
FILETYPE: image/jpeg
FILEID: a1b2c3d4
DESCRIPTION: Vacation snapshot
TIMESTAMP: 1728939100
TOKEN: alice@192.168.1.11|1728942700|file

Field Descriptions:

- FILENAME, FILESIZE, FILETYPE: Metadata.
- FILEID: Unique file identifier.
- DESCRIPTION: Optional label.
- TOKEN: Scoped for file. [user_id|TIMESTAMP+TTL|file]

Non-verbose Printing:

“User alice is sending you a file do you accept?”

5.9 FILE_CHUNK

Transmits a piece of a file in base64-encoded form, referencing the file via its ID and indicating its index in the sequence. Used in combination with FILE_OFFER to transfer files over unreliable UDP. The receiving peer must reassemble chunks based on their index.

Format:

TYPE: FILE_CHUNK
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
FILEID: a1b2c3d4
CHUNK_INDEX: 0
TOTAL_CHUNKS: 5
CHUNK_SIZE: 256
TOKEN: alice@192.168.1.11|1728942700|file
DATA: VGVzdCBjaHVuayAxLi4u

Field Descriptions:

- CHUNK_INDEX, TOTAL_CHUNKS: For reassembly.
- CHUNK_SIZE: Number of bytes for chunks [The maximum chunk size is set by the sender]
- DATA: Base64-encoded payload.

Non-verbose Printing:

Do not print anything until all the chunks are completed. “File transfer of *filename* is complete”

5.10 FILE_RECEIVED

Notifies the sender that the file was received successfully. It includes a status code (e.g., COMPLETE) and references the original file ID. This optional message can help the sender know when to stop transmitting or retry missing chunks.

Format:

TYPE: FILE_RECEIVED
FROM: bob@192.168.1.12
TO: alice@192.168.1.11
FILEID: a1b2c3d4
STATUS: COMPLETE
TIMESTAMP: 1728939500

Field Descriptions:

- STATUS: e.g., COMPLETE.

Non-verbose Printing:

Do not print anything.

5.11 REVOKE

Invalidates a previously issued token, immediately preventing further use. Useful for session management or emergency denial of access. This message is advisory, and clients must track revoked tokens locally.

Format:

TYPE: REVOKE
TOKEN: alice@192.168.1.11|1728942100|chat

Field Descriptions:

- TOKEN: Identifies which credential is revoked. [user_id|TIMESTAMP+TTL|chat]

Non-verbose Printing:

Do not print anything.

5.12 TICTACTOE_INVITE

Initiates a Tic Tac Toe game between two users, specifying who moves first and assigning a symbol (X or O). The game is identified by a GAMEID. The invitee may accept by replying with a move.

Format:

TYPE: TICTACTOE_INVITE
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
MESSAGE_ID: f83d2b2b
SYMBOL: X
TIMESTAMP: 1728940000
TOKEN: alice@192.168.1.11|1728943600|game

Field Highlights:

- GAMEID: Unique game session. [gX where X is a number between 0-255]
- SYMBOL: Symbol used by the sender

Non-verbose Printing:

alice is inviting you to play tic-tac-toe.

5.13 TICTACTOE_MOVE

Sends a game move including position, symbol, and turn number, associated with a specific game session. Both players use this message to exchange moves. It enables synchronized gameplay without a central server.

Format:

TYPE: TICTACTOE_MOVE
FROM: bob@192.168.1.12

TO: alice@192.168.1.11
GAMEID: g123
MESSAGE_ID: f83d2b2d
POSITION: 4
SYMBOL: O
TURN: 2
TOKEN: bob@192.168.1.12|1728943600|game

Field Highlights:

- GAMEID: Unique game session.
- POSITION: For moves.Integer 0–8 (top-left to bottom-right)

  0 | 1 | 2

  3 | 4 | 5

  6 | 7 | 8

- RESULT, WINNING_LINE: End-of-game state.
- POSITION: Integer 0–8 (top-left to bottom-right)
- SYMBOL: The player's symbol (must match their assigned role)

Non-verbose Printing:

Print the board.

5.14 TICTACTOE_RESULT

Announces the outcome of a Tic Tac Toe game, such as WIN, LOSE, or DRAW. The message includes the winning line (if applicable) and the game ID. It serves as the game's final state and can trigger a client-side summary.

Format:

TYPE: TICTACTOE_RESULT
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
MESSAGE_ID: f83d2b2e
RESULT: WIN
SYMBOL: X
WINNING_LINE: 0,1,2
TIMESTAMP: 1728940123

Field Highlights:

- GAMEID: Unique game session.
- POSITION, TURN: For moves.
- RESULT: One of WIN, LOSS, DRAW, or FORFEIT
- WINNING_LINE: Optional; indices of winning cells (e.g., 0,1,2)

Non-verbose Printing:

Print only the board and whose turn it is.

5.15 LIKE

Expresses appreciation or approval for a specific post, referencing it by timestamp. The message may also signal an UNLIKE to retract the action. Likes are ephemeral and advisory, with client-side counting optional.

Format:

TYPE: LIKE
FROM: bob@192.168.1.12
TO: alice@192.168.1.11
POST_TIMESTAMP: 1728938391
ACTION: LIKE
TIMESTAMP: 1728940500
TOKEN: bob@192.168.1.12|1728944100|broadcast

Field Descriptions:

- FROM: User ID of the liker
- TO: User ID of the original post's author
- POST_TIMESTAMP: The TIMESTAMP value of the POST being liked
- ACTION: One of LIKE or UNLIKE
- TIMESTAMP: When the like/unlike was sent
- TOKEN: Authorization token with broadcast scope [user_id|TIMESTAMP+TTL|broadcast]

Non-verbose Printing:

alice likes your post [post y message]

5.16 GROUP_CREATE

Creates a new user group and sets its initial membership. A group has a unique ID and a human-readable name. Members can exchange messages or participate in group games or file transfers.

Format:

TYPE: GROUP_CREATE
FROM: alice@192.168.1.11
GROUP_ID: tripbuds2025
GROUP_NAME: Trip Buddies
MEMBERS: alice@192.168.1.11,bob@192.168.1.12,charlie@device888
TIMESTAMP: 1728941000
TOKEN: alice@192.168.1.11|1728944600|group

Field Highlights:

- GROUP_ID: Unique identifier (e.g., alphanumeric string)
- GROUP_NAME: Human-readable name
- MEMBERS: Comma-separated list of user IDs
- FROM: Creator's user ID
- TOKEN: Must have group scope
- TIMESTAMP: Creation time

Non-verbose Printing:

"You've been added to Trip Buddies"

5.17 GROUP_UPDATE

Modifies the membership of an existing group by adding or removing users. Only the group creator or authorized users may perform updates. Group state must be managed locally on each peer. Only send to group members.

Format:

TYPE: GROUP_UPDATE
FROM: alice@192.168.1.11
GROUP_ID: tripbuds2025
ADD: dan@device777
REMOVE: charlie@device888
TIMESTAMP: 1728941300
TOKEN: alice@192.168.1.11|1728944900|group

Field Highlights:

- ADD: Comma-separated list of user IDs to add
- REMOVE: Comma-separated list to remove
  Can include either or both

Non-verbose Printing:

The group “Trip Buddies” member list was updated.

5.18 GROUP_MESSAGE

Sends a text message to all members of a specific group. Group messages rely on known membership and use a token with the appropriate scope. They enable efficient communication within defined social clusters.

Format:

TYPE: GROUP_MESSAGE
FROM: bob@192.168.1.12
GROUP_ID: tripbuds2025
CONTENT: Just uploaded the photos!
TIMESTAMP: 1728941400
TOKEN: bob@192.168.1.12|1728945000|group

Field Highlights:

- CONTENT: Text for group communications.

Non-verbose Printing:

- bob@192.168.1.12 sent “Just uploaded the photos!”

---

6. User Discovery

In LSNP, peer discovery is facilitated through periodic broadcasting of network presence information. Each participating node announces its availability and identity by transmitting two distinct message types—**PING** and **PROFILE**—at regular 300-second intervals using broadcast communication. The PING message acts as a lightweight heartbeat signal, informing others on the network that a peer is active and reachable. Meanwhile, the PROFILE message conveys additional metadata such as the peer's name, capabilities, or status, allowing for richer interaction and recognition among nodes. The default message is PING unless a PROFILE message is sent within the interval. This decentralized mechanism ensures that peers can dynamically detect and update their view of other users within the local network, maintaining an up-to-date and resilient social topology without relying on centralized coordination.

---

7. Expiration

Expiration refers to the concept that a token remains valid only for a limited duration after it has been issued. Its main purpose is to prevent the reuse or abuse of tokens once they have outlived their intended timeframe. To validate expiration, the system compares the timestamp embedded in the token against the current time. If the token's timestamp indicates a time in the past, it is considered expired and must be rejected.

---

8. Revocation

Revocation refers to the mechanism that allows a token to be explicitly invalidated before its natural expiration time. This capability is particularly useful in scenarios where a user logs out, switches devices, or when suspicious or unauthorized activity is detected.

To implement revocation in LSNP, each peer maintains a lightweight, in-memory revocation list containing token hashes or unique token identifiers. When a peer receives a message accompanied by a token, it consults this list to determine if the token has been previously revoked. If the token appears on the list, the message is rejected, effectively blocking unauthorized or stale sessions from being processed. This approach provides a simple yet effective way to enhance security and control over active tokens in a decentralized environment.

---

9. Scoping

Scoping defines the set of actions a token is authorized to perform, serving as a mechanism to enforce permission boundaries within a system. Its primary purpose is to contain potential damage in the event a token is leaked or misused. By assigning specific scopes—such as broadcast, post, or dm—to each token, the system ensures that tokens are only used for their intended functions.

In LSNP, enforcing token scopes involves validating the intended operation against the token's declared scope. For example, if a token is granted the broadcast scope, it should not be permitted to send direct messages (dm). If such a mismatch occurs—where the operation requested exceeds or deviates from the token's scope—peers must reject the request to maintain protocol integrity and prevent unauthorized actions. This approach upholds strict access controls in decentralized settings.

In the **Local Social Networking Protocol (LSNP)**, a **SCOPE** defines the area or permission boundary within which a given action (such as a message or token use) is valid. It is especially relevant for **token validation** and **admin commands**.

Here are the defined scope values and their meanings:

Scope	Used In	Purpose
chat	DM, REVOKE	Allows sending direct messages to another peer.
file	FILE_OFFER, FILE_CHUNK	Grants permission to initiate and transmit file transfers.
broadcast	POST, LIKE	Used for public posts and post interactions like likes/unlikes.
follow	FOLLOW, UNFOLLOW	Permits managing social relationships (subscribe/unsubscribe).
game	TICTACTOE_*	Restricts interaction to specific game sessions (e.g., moves, results).
group	GROUP_*,	Grants permission to create, update, message, or manage groups.

10. Game

In LSNP's decentralized gaming model, all messages are stateless, requiring each peer to maintain its own local copy of the game board without relying on shared session data. To ensure consistency and prevent desynchronization, clients are programmed to silently ignore or reject invalid or conflicting moves. Additionally, peers can implement protective behaviors such as automatically declining repeated game invitations from the same source or disconnecting inactive games after a specified timeout, thereby supporting a smooth and self-regulated user experience within the protocol's lightweight framework.

Acknowledgement & Retry:

After sending a `TICTACTOE_INVITE` or `TICTACTOE_MOVE`, the sender should expect an `ACK` message containing a unique `MESSAGE_ID`. If no ACK is received within a short timeout (e.g., 2 seconds), the sender should retry the message up to 3 times.

Duplicate Detection:

Recipients must ensure idempotency by checking for duplicate moves based on `GAMEID` and `TURN`. If a duplicate is received, it should be acknowledged again but ignored for game logic.

---

11. Groups

In LSNP, group functionality is fully decentralized, meaning each peer is independently responsible for maintaining an up-to-date record of group membership. To perform any group-related action—such as sending messages to a group or modifying its state—a valid token with the group scope is required, ensuring that only authorized operations are executed. Group identifiers (IDs) play a critical role in distinguishing groups and are both case-sensitive and expected to be unique within the local network. This approach preserves LSNP's lightweight and autonomous structure while supporting dynamic, multi-user collaboration.

---

12. Token Format and Validation

Example:

alice@192.168.1.11|1728943600|game

Validation:

For a token to be considered valid in LSNP, it must meet three critical criteria. First, the token must not be expired; its embedded timestamp must still fall within the acceptable time window, ensuring it hasn't outlived its intended lifespan. Second, the scope of the token must precisely align with the action being attempted—for example, a token scoped for `group` cannot be used for a `broadcast` operation. Third, the token must not appear on the revocation list maintained by the receiving peer; any token flagged as revoked is to be ignored and rejected outright. These three checks—expiration, scope match, and revocation status—must all be satisfied for a token to be accepted and processed by the protocol.

---

13. Implementation Considerations

In LSNP, binary file chunks and avatar images are encoded using Base64 to ensure they can be safely transmitted within plaintext message structures. This encoding facilitates the inclusion of binary data in the protocol's key-value format without introducing parsing issues. All messages exchanged between peers must be UTF-8 encoded, ensuring consistent interpretation of textual content across different systems and languages. Furthermore, because LSNP operates in a stateless and decentralized manner, each peer is individually responsible for maintaining its own local record of game state and group membership. This local tracking enables smooth interaction without relying on centralized storage or coordination.

---

14. Security Considerations

To ensure message authenticity in LSNP, peers should verify the source IP address by comparing the `FROM` field declared in each message with the actual IP address from which the UDP packet was received. This verification step helps detect spoofed or manipulated source information that could compromise the integrity of peer interactions. If a mismatch is detected—indicating a potential spoofing attempt or network anomaly—the message should either be logged for auditing purposes or silently discarded to prevent unauthorized influence on local state. This safeguard reinforces LSNP's trust model by making peer identity checks more robust in the absence of encryption.

In LSNP, all network traffic is openly visible to all participating peers, reflecting the protocol's lightweight and transparent design. By default, no encryption is applied to messages, which simplifies implementation but also introduces the potential for sensitive data to be observed by unintended recipients. Given the lack of cryptographic safeguards, tokens transmitted across the network may be susceptible to spoofing or reuse if intercepted.

---

15. Verbose Mode

All LSNP client implementations **must support a verbose mode** to aid in debugging and development. When enabled, verbose mode should:

- Print all **incoming and outgoing messages** in human-readable format.
- Include **timestamp**, **sender IP**, and **message type** in each log entry.
  Log all **token validations**, **acknowledgements**, **packet retransmissions**, and **dropped packets** (if simulated).
  Indicate retry attempts for **lost Tic Tac Toe moves** and **file chunks**.
- Optionally color-code or prefix logs for clarity (e.g., `SEND >`, `RECV <`, `DROP !`).

Verbose mode can be toggled via a command-line flag (e.g., `--verbose`) or configuration setting.

---

16. References

- RFC 4648 – Base64

---

17. AI Disclaimer

The protocol is primarily designed by the author. AI tools, primarily ChatGPT and CoPilot, are leveraged for the writing of the RFC. AI was leveraged to help formulate message structures and RFC formatting. Additionally, it supported the writing of the grading rubric, collaboration policy, and testing guidelines. All AI-generated content was thoroughly reviewed, validated, and adapted to meet the functional and educational goals of the project. No AI-generated code or documentation was submitted without understanding or modification.

---

18. Appendix A: Sample Tic Tac Toe Game Log

Game Log

1. Invite from Alice

TYPE: TICTACTOE_INVITE
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
SYMBOL: X
TIMESTAMP: 1728940000
TOKEN: alice@192.168.1.11|1728943600|game

2. Alice Move (position 0)

TYPE: TICTACTOE_MOVE
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
POSITION: 0
SYMBOL: X
TURN: 1
TOKEN: alice@192.168.1.11|1728943600|game

```
 X |   |

-----------

  |   |

-----------

  |   |
```

3. Bob Move (position 4)

TYPE: TICTACTOE_MOVE
FROM: bob@192.168.1.12
TO: alice@192.168.1.11
GAMEID: g123
POSITION: 4
SYMBOL: O
TURN: 2
TOKEN: bob@192.168.1.12|1728943600|game

```
 X |   |

-----------

   | O |

-----------

   |   |
```

4. Alice Move (position 1)

TYPE: TICTACTOE_MOVE
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
POSITION: 1
SYMBOL: X
TURN: 3
TOKEN: alice@192.168.1.11|1728943600|game

```
 X | X |

-----------

   | O |

-----------

   |   |
```

5. Bob Move (position 5)

TYPE: TICTACTOE_MOVE
FROM: bob@192.168.1.12
TO: alice@192.168.1.11
GAMEID: g123
POSITION: 5
SYMBOL: O
TURN: 4
TOKEN: bob@192.168.1.12|1728943600|game

```
 X | X |

-----------

  | O | O

-----------

  |   |
```

6. Alice Move (position 2)

TYPE: TICTACTOE_MOVE
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
POSITION: 2
SYMBOL: X
TURN: 5
TOKEN: alice@192.168.1.11|1728943600|game

```
 X | X | X

-----------

  | O | O

-----------

  |   |
```

7. Result

TYPE: TICTACTOE_RESULT
FROM: alice@192.168.1.11
TO: bob@192.168.1.12
GAMEID: g123
RESULT: WIN
SYMBOL: X
WINNING_LINE: 0,1,2
TIMESTAMP: 1728940123

 X | X | X

  | O | O

  | |
