RoboSats Federation Reputation with Nostr Integration

Name and Contact Information:

- Name - Rijul Singla
- Email - rijulsingla57@gmail.com
- Telegram username - rijul57
- Discord username - rijul57
- Country - India
- University - IIT-BHU (2nd year)
- Github - rijul57 (github.com)
- Linkedin - https://www.linkedin.com/in/rijul-singla-319265252/

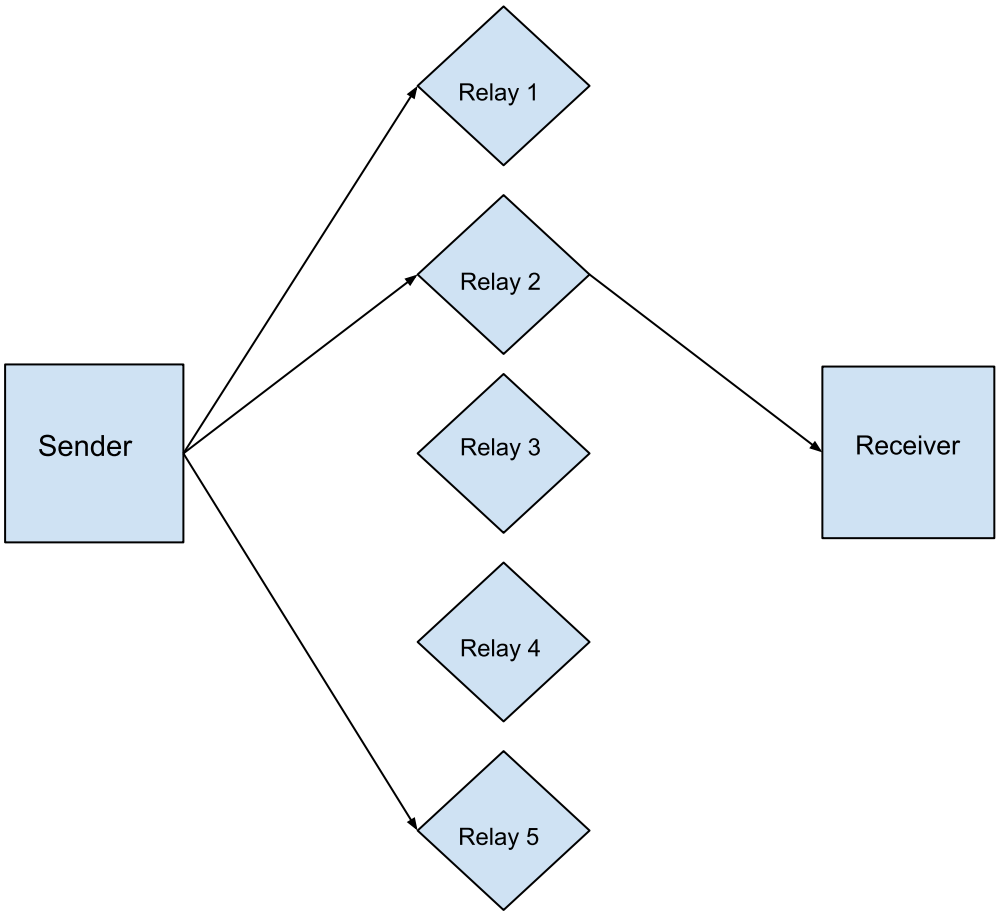
Synopsis

The project aims to implement a secure and decentralised rating system for the coordinators through the nostr network. The current rating system is static and resides only with the coordinator. The aim is to implement a decentralised rating system using the nostr network preventing none among coordinators, users, or federation from having control.

Theoretical Details

Nostr is a network protocol designed to provide a **censorship-free** social media platform. It works by people sending notes to multiple relays. Multiple relays are important since they decentralize the network away from a single fracture point. The receiving user can then request

the note from the relay.



However, these decentralized capabilities of nostr can also be utilised in creating a robust, transparent rating system. **Ratings** can be sent as a **nostr note** to prevent tampering of ratings and also ensure uniqueness of ratings. A nostr node is a **json object**:

```
{
    "id": "<32-bytes lowercase hex-encoded sha256 of the serialized event data>",
    "pubkey": "<32-bytes lowercase hex-encoded public key of the event creator>",
    "created_at": "<unix timestamp in seconds>",
    "kind": "<integer between 0 and 65535>",
    "tags": [
        ["e", "5c83da77af1dec6d7289834998ad7aafbd9e2191396d75ec3cc27f5a77226f36", "wss://nostr.example.com"],
        ["p", "f7234bd4c1394dda46d09f35bd384dd30cc552ad5541990f98844fb06676e9ca"]
        //.....
    ],
    "content": "<arbitrary string>",
    "sig": "<64-bytes lowercase hex of the signature of the sha256 hash of the serialized event data, which is the same as the 'id' field>"
}
```

- **Kinds** specify how clients should interpret the meaning of each event and the other fields of each event. The meaning of tags depends on the kind being used.

- **Tags** store additional info such as pubkey and eventId.

Implementation Details

The first step is to implement the note functionality. The proposal involves **2 such notes**.
The first note is created when the user clicks the “Submit Rating” button. The structure of the note looks like

- pubkey : Robot’s pubkey
- content : Formatted rating
- tags:
    - p : Federation pubkey
    - p : Coordinator pubkey
    - subject : Ordr unique identifier
    - rating : User’s rating

In a nostr network, users subscribe to sender’s npub key. When the sender posts a note, relays send data to the subscribers. **Our requirement is fundamentally different from this**. We have users sending their notes to the federation making it impossible for the federation to subscribe to every user to receive data. **The federation therefore fetches data based not on the pubkey but on the value of the kind.** (The value of kind and which relays to use shall be discussed with the mentors). The fetching of data will be periodic.

```
const intervalId = setInterval(requestData, k); // Sends request every k seconds
```

This time period will depend on various factors:

1. Volume of traffic
2. Speed of federation backend
3. Quality of relays used (paid/free)

The exact parameter has to be arrived at depending on the exact implementation and testing.

The serialisation of the note will be as follows:

1. 0,
2. Robot pubkey,
3. The 2 p tags as an array of arrays of non-null strings

And the Id will be sha256 of this serialization.

The federation does a similar serialization for the donation invoice to store in the backend.
The rationale behind this is to prevent duplication. Such a serialization ensures that every rating note for a particular order has the same noteId which matches its invoiceId. Now the task of matching reduces to a simple one-to-one mapping with the stored invoiceIds in the backend.

I will use the following implementation to send notes via nostr.

```
// Connect to relay by creating a socket
var relay = "wss://relay.roygbiv.guide";
var socket = new WebSocket( relay );

// Send note after successful connection to relay
socket.addEventListener('open', async function( e ) {
    var event = {
        "pubkey"     : pubKey,
        "created_at" : Math.floor( Date.now() / 1000 ),
        "kind"       : "",
        "tags"       : [
            ["p", federationPubKey]
            ["p", coordinatorPubKey]
            ["subject", orderId]
            ["rating", rating]
        ],
    }

    // Sign the note
    var signedEvent = await getSignedEvent(event, privKey);
    try {
        socket.send(JSON.stringify([ "EVENT", signedEvent ]));
        console.log('Event message sent successfully.');
    } catch (error) {
        console.error('Error sending event message:', error);
    }
});

async function getSignedEvent(event, privateKey) {
    var eventData = JSON.stringify([
        0,
        event['pubkey'],
        event['tags'],
    ]);
    event.id  = bytesToHex( await sha256( ( new TextEncoder().encode( eventData ) ) ) );
    event.sig = await schnorr.sign( event.id, privateKey );
    return event;
}
```

The federation after fetching the note by its kind then performs the following checks:

1. **Verify the signature of the note (using the “elliptic” library)**
2. **Search noteId against the invoiceIds stored in backend.**

If the note passes these checks, the federation now creates a 2nd “confirmatory” note to some relay. The structure of the 2nd note is:

```
{
    "id": "675bab...8c1221",
    "pubkey": "bf2376...26bce",
    "created_at": 1673347337,
    "kind": "",
    "tags": [
        ["e", "6fb4d86...ade63da"],
        ["p", "6fb4d86...ade63da"]
    ],
    "content": "",
    "sig": "6fb4d8...a329186"
}
```

- e : Robot note Id

Now the final part is implementing the UI on the webapp

1. **Make a “Submit Rating” button.** Upon clicking the button, the webapp client makes a connection to the relay and sends the note.
2. **Show the average rating for the coordinator** on the coordinator-info page. This is implemented by performing the following count operations
    - Count by pubkey of coordinator
    - Count the (1 - 5) ratings for that coordinator

A weighted mean is taken over the ratings to display the average rating value.

Project Timeline

Community Bonding and Onboarding(May 9, 2024 - May 23, 2024)	
`May 9, 2024 - May 23, 2024	• Get familiar with the codebase and the mentors. • Discuss my approaches, logic, and milestones with the mentor.
Project Period(May 23, 2024 - Aug 15, 2024)	
May 23, 2024 - May 31, 2024	• Create the “Submit Rating” button • Send test notes • Document the code
Jun 1, 2024 - Jun 15, 2024	• Create verify and check functions for the federation. • Send the confirmatory notes

	• Build necessary tests. • Keep on documenting things.
Jun 16, 2024 - Jul 1, 2024	• Add coordinator rating info on webpage UI. • Setting up the required software and applications for testing.
Jul 2, 2024 - Jul 7, 2024	• Discuss the stretch goal with the mentor and finalize what to do.
Jul 8, 2024 - Jul 31, 2024	• Work on the stretch goal. • Extensive testing, catching bugs, fixing them and further improving the code coverage.
Aug 1, 2024 - Aug 15, 2024	• The final 2 weeks have been left free for completing any remaining work (if any). This provides sufficient cushion for making sure that the timeline is followed.

Future Deliverables

I intend to stay with the community after the Summer of Bitcoin. Integrating Robosats can have many more applications, including encrypted chat between buyer and seller and many more. I would love to extend nostr integration over the platform and potentially working on other new features. It would give me immense pleasure

to be a part of the RoboSats community and improve the lives of those who use Robosats.

Benefits to Community

This project can have huge benefits for the community using RoboSats. A rating system would allow users to easily navigate and choose among the different coordinators. It would also incentivize the coordinators to better service their customers and resolve disputes while punishing the non-compliant ones.

Biographical Information

I'm **Rijul Singla**, a sophomore currently pursuing Computer Science and Engineering at the Indian Institute of Technology (BHU) Varanasi and am expected to graduate in the year 2026.

I got interested in the world of programming and software in my freshman year. Since then, I've been very enthusiastic about learning various web technologies and languages. I'm proficient in coding in languages like JavaScript, Python and C++. I have also worked with various frameworks like django and react. Moreover, I have a good grasp of Git and Github. Being a specialist at codeforces, I also have a good understanding of data structures and algorithms. In my leisure time, I like to learn about historical kingdoms, play football , and practice on my piano.

References

- https://guide.summerofbitcoin.org/
- nips/01.md at master · nostr-protocol/nips (github.com)

- Coordinator ratings over Nostr · Issue #1097 · RoboSats/robosats (github.com)