CS-740 Blockchain Design Architecture

A REPORT ON THE PROJECT ENTITLED

A Lightweight Model Based Evolutionary Consensus Protocol in Blockchain as a service for IOT

Group Members:

K S ANKITA	242IS014
PRINCE KUMAR	242IS020
AASTHA GABA	242IS035

II SEMESTER M-TECH CSE-IS

DEPT. OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL

Contents

Abstract

The rapid growth of the Internet of Things (IoT) demands secure, scalable, and efficient data exchange platforms. Blockchain as a Service (BaaS) has emerged as a promising solution due to its decentralization, immutability, and auditability. However, traditional consensus protocols used in blockchain are resource-intensive and unsuitable for IoT devices with limited computational capacity. To address this gap, this project implements a lightweight model-based evolutionary consensus protocol known as Proof of Evolutionary Model (PoEM). PoEM leverages supervised machine learning to dynamically select block producers based on node characteristics, reducing energy consumption and enhancing efficiency. The system evolves its decision-making model over time and can adapt to node dynamics such as joining and exiting. Our implementation demonstrates how PoEM can enable low-power IoT devices to effectively participate in a secure and decentralized blockchain system. The simulation using Mininet and Python validates PoEM's performance in achieving consensus with low latency, high scalability, and synchronized outputs across all nodes.

Introduction

The convergence of Blockchain and the Internet of Things (IoT) has opened new possibilities for building secure, decentralized, and trustworthy systems. IoT is rapidly expanding across domains such as smart cities, healthcare, logistics, and manufacturing, creating a vast network of interconnected, resource-constrained devices. As IoT systems become increasingly autonomous and data-intensive, ensuring data integrity, transparency, and security becomes critical.

Blockchain technology, known for its decentralization, immutability, and auditability, offers a natural solution to address trust and security challenges in IoT. Blockchain as a Service (BaaS) further simplifies the adoption of blockchain by providing a cloud-based infrastructure for developing and deploying blockchain applications. BaaS enables developers to focus on application logic while abstracting away the complexity of underlying blockchain frameworks.

However, integrating blockchain into IoT environments introduces significant challenges. Most existing consensus protocols—such as Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT)—are either computationally intensive or communication -heavy. These protocols are designed for high-performance servers or powerful nodes and are not feasible for typical IoT devices like sensors, embedded boards, and edge devices with limited CPU, memory, and power. This mismatch creates a critical bottleneck for deploying blockchain-based systems in real-world IoT applications.

To overcome these challenges, there is a growing need for lightweight, efficient, and scalable consensus mechanisms tailored to IoT. The motivation behind this project stems from the desire to develop a consensus protocol that minimizes resource consumption while maintaining security, fairness, and adaptability. This led to the exploration and implementation of the Proof of Evolutionary Model (PoEM)—a novel consensus mechanism that leverages machine learning to intelligently select block producers. Unlike rule-based methods, PoEM evolves over time by learning from node behavior and transaction history, making it ideal for dynamic IoT environments.

1 Challenges in Integrating Blockchain and IoT

1.1 Resource Constraints

IoT devices typically have limited computational power, memory, and energy, which make them unsuitable for traditional consensus mechanisms like Proof of Work (PoW) and Proof of Stake (PoS). These protocols require significant computational resources and communication overhead, which are not feasible for many IoT devices, particularly sensors or edge devices.

1.2 Latency and Scalability

Traditional consensus protocols may also struggle with latency and scalability when applied to large IoT networks, as IoT devices generate massive amounts of data that need to be processed and validated quickly. Furthermore, as IoT networks grow, the need for scalable solutions that can handle increasing transaction volumes becomes critical.

1.3 Energy Efficiency

IoT devices are often battery-powered, and the energy consumption associated with traditional consensus mechanisms can lead to shorter device lifespans and higher operational costs.

These limitations highlight the urgent need for a new class of consensus mechanisms that are lightweight, adaptable, and secure—capable of operating under the constraints of IoT ecosystems while preserving the decentralized nature of blockchain.

2 The Proof of Evolutionary Model (PoEM)

PoEM appears to be a promising solution that seeks to address these challenges by incorporating machine learning (ML) into the consensus process. The idea of using a model that adapts over time by learning from node behavior and transaction history provides several benefits:

2.1 Adaptability

PoEM's ability to evolve based on past behaviors allows it to adjust dynamically to changing IoT environments. This is particularly useful in environments where node behavior is unpredictable, and traditional consensus protocols may struggle.

2.2 Efficient Block Production

By intelligently selecting block producers based on learning outcomes, PoEM minimizes resource consumption, ensuring that IoT devices are not burdened with resource-heavy tasks.

2.3 Reduced Energy Consumption

As PoEM is designed to be lightweight and tailored for IoT environments, it can help conserve energy, which is crucial for battery-powered IoT devices.

2.4 Security and Fairness

Although PoEM reduces computational overhead, it still maintains security and fairness by learning and adapting to node behaviors, potentially creating a system that is both scalable and resilient.

3 Blockchain as a Service (BaaS) for IoT

The concept of BaaS can simplify the process of adopting blockchain in IoT environments by providing developers with an easy-to-use platform that abstracts much of the complexity associated with blockchain deployment. With BaaS, developers can focus more on the application layer, leveraging the security and trust features of blockchain without worrying about managing the infrastructure. For IoT systems, this can lead to faster development cycles and easier integration into existing infrastructure.

4 Background on Blockchain and IoT

The Internet of Things (IoT) refers to a vast network of interconnected physical devices—ranging from sensors and wearables to smart appliances—that collect and exchange data autonomously. With the advent of 5G, edge computing, and autonomous systems, IoT is rapidly evolving to enable real-time data processing and decision-making at the edge of the network. However, this decentralization comes at the cost of increased complexity in managing data trust, security, and privacy.

Blockchain technology, with its distributed ledger structure, provides an ideal solution to address these concerns. It ensures tamper-proof recording of data, decentralized validation through consensus mechanisms, and transparent traceability of events. Blockchain can mitigate many IoT-specific challenges, including unauthorized access, single points of failure, and data manipulation.

Blockchain as a Service (BaaS) further enhances the integration of blockchain with IoT by offering managed platforms that abstract the underlying complexity. BaaS platforms—such as those by IBM, Microsoft Azure, and AWS—allow IoT developers to quickly deploy and manage blockchain infrastructure without requiring deep domain expertise in cryptographic protocols or consensus design.

Despite its potential, the practical adoption of blockchain in IoT has been hindered by the mismatch between the resource demands of blockchain and the limited capabilities of IoT devices. This has brought consensus mechanisms to the center of attention in research and development efforts for scalable blockchain-based IoT systems.

5 Proposed Approach: Proof of Evolutionary Model (PoEM)

The proposed approach, named **Proof of Evolutionary Model (PoEM)**, is a lightweight, machine learning-based consensus protocol designed specifically for BaaS-enabled IoT environments. It replaces rule-based consensus mechanisms with an adaptive model that ranks nodes based on features to determine the most suitable block producer.

Core Idea

- PoEM uses supervised machine learning to perform node ranking and block producer selection.
- Each node runs a shared pre-trained model to evaluate all participating nodes in a given consensus round.

Key Components

- **Node Features:** Includes numerical (e.g., CPU usage, success rate) and class features (e.g., node type).
- **Feature Processing:** Class features are one-hot encoded; numerical features are normalized.
- **Training Dataset:** Derived from system logs and includes labeled data for model learning (1 for selected producer, 0 otherwise).

Consensus Workflow

1. **Model Deployment:** A trained PoEM model is distributed to all nodes in the system.
2. **Consensus Execution:**
   - Each node inputs the feature set into the PoEM model.
   - The model outputs a sorted node selection list (SNSL).
   - The top-ranked node becomes the block producer.
   - The block is validated and accepted by all other nodes.
3. **Model Evolution:** New data from system activity is used to incrementally update the model to improve adaptability.

Cold Start and Node Dynamics

- **Cold Start:** Existing protocols like PoW or PoS are used initially to generate training data.
- **Node Joining:** New nodes are randomly given a chance to become block producers until recognized by the model.
- **Node Exit:** Exit tables and default rules ensure consensus continues smoothly if a node drops unexpectedly.

Security and Privacy

- PoEM operates as a "black box," hiding block producer selection logic and enhancing resistance to targeted attacks.
- Game theory is used to demonstrate that PoEM reduces attacker incentive in 51% attack scenarios.

Efficiency and Scalability

- PoEM achieves **O(n)** communication complexity and **O(1)** computation cost for each node.
- It significantly outperforms traditional protocols (e.g., PoW, PBFT) in large-scale, dynamic IoT settings.

6 Consensus Mechanism Design

The design of the PoEM consensus mechanism aims to balance efficiency, security, and adaptability within the constraints of IoT environments. Unlike traditional consensus methods that statically assign mining responsibilities or rely on resource-heavy computations, PoEM introduces a dynamic and intelligent selection of miners using evolutionary principles. This approach allows the system to adapt to the changing behavior and capabilities of IoT nodes, improving fairness, reducing energy consumption, and increasing resilience to attacks.

PoEM relies on historical data and real-time metrics to make informed decisions regarding which nodes are most suitable to produce the next block. This reduces unnecessary communication overhead and ensures that only capable and trustworthy nodes are involved in the consensus process.

7 Evolutionary Miner Selection using Genetic Algorithms

At the heart of PoEM is the miner selection process based on Genetic Algorithms (GAs). Genetic Algorithms are a class of evolutionary computation techniques inspired by natural selection and genetics. In the context of PoEM, GAs are used to evolve a population of candidate miners by selecting, recombining, and mutating potential candidates over successive generations.

Each node in the network is represented as a candidate solution (chromosome) characterized by various attributes such as historical reliability, energy level, response time, and past transaction accuracy. The GA operates through the following steps:

- **Initialization:** A population of candidate miners is initialized based on active nodes.
- **Evaluation:** Each candidate is evaluated using a fitness function that incorporates miner selection criteria.
- **Selection:** The fittest candidates are selected for reproduction.
- **Crossover and Mutation:** New candidates (offspring) are generated by recombining and mutating the selected candidates.
- **Replacement:** The least fit candidates are replaced by the offspring to form the next generation.

This evolutionary cycle continues until a termination condition is met, such as reaching a maximum number of generations or achieving a satisfactory fitness threshold. The final selected candidate(s) are designated as block producers for the current round.

8 Fitness Function and Criteria for Miner Selection

The fitness function is a crucial component of PoEM as it determines how suitable each node is for participating in the consensus process. The function is multi-objective and considers the following factors:

- **Energy Efficiency:** Nodes with higher available energy are preferred to ensure sustainability.

- **Latency:** Nodes with lower communication and processing latency are favored.
- **Reputation Score:** Derived from historical behavior, including successful block production and absence of malicious activities.
- **Transaction Accuracy:** Nodes that have a high rate of valid transaction verification are rated higher.
- **Connectivity:** Nodes with stable and reliable network connections are prioritized.

The overall fitness score is calculated as a weighted sum or composite of these attributes. This dynamic and data-driven evaluation ensures that miner selection remains fair, secure, and context-aware.

9 Implementation of PoEM

The core of the Proof of Evolutionary Model (PoEM) consensus mechanism is governed by an objective function that evaluates the fitness of nodes based on their historical behavior and dynamically learned models. The fitness function consists of three key components:

1. **Experience Loss**: Measures how well the model predicts the performance of a node based on labeled training data.
2. **Regularization Term** (γ): Prevents overfitting by penalizing excessively large model weights.
3. **Constrained Punishment Term** ($\alpha\eta_i$): Penalizes nodes that frequently fail to produce valid blocks when selected as block producers.

The overall objective function used in our implementation is given below:

$$\min_{\omega} \sum_{i=1}^{m} \log\left(1 + \exp\left(-Label_i \cdot \vartheta_\xi(\omega, nfi)\right)\right) + \frac{\gamma}{2}\|\omega\|_2^2 - \alpha\eta_i, \quad i \in \{1, 2, \ldots, m\}$$

Where:

- ϑ_ξ is the model calculation function used to evaluate node fitness; in our case, it is realized using logistic regression.
- ω represents the model parameters (weights).
- γ is a regularization parameter used to avoid weight dominance.
- η_i is the number of block production failures for node i.
- α is the punishment factor that determines the severity of penalizing failed nodes.
- nfi denotes the feature input for node i.
- $Label_i$ is the ground truth label used during supervised learning to indicate success or failure in block production.

This function is implemented programmatically by simulating each node as a machine learning agent (PoEMNode class), which uses logistic regression to estimate its own suitability for block production. Nodes collect runtime system features (e.g., CPU usage, network latency), and the model computes a fitness score. Nodes with frequent block production failures receive a reduced score through the $\alpha\eta_i$ punishment term, ensuring fairness and reliability in miner selection.

9.1 Fitness Prediction

The fitness score for each node is predicted using a sigmoid function applied to the weighted sum of input features. This is equivalent to applying a logistic regression model. To reflect the punishment for past failures in block production, a penalty term based on the number of failures (η_i) is subtracted from the sigmoid output. This follows the form of the objective function described earlier.

$$z = \omega \cdot feature\_vec + b \tag{1}$$

$$sigmoid(z) = \frac{1}{1 + \exp(-z)} \tag{2}$$

$$PoEM\_score = sigmoid(z) - \alpha \cdot \eta_i \tag{3}$$

This mechanism ensures that nodes with frequent failures are gradually deprioritized in the consensus process.

The corresponding implementation in Python is shown below:

```python
def predict_fitness(self, feature_vec, alpha=0.5):
    weight = self.model.coef_[0]
    bias = self.model.intercept_[0]

    z = np.dot(weight, feature_vec) + bias
    sigmoid = 1 / (1 + np.exp(-z))

    eta = self.fitness_history.count(0)  # Count of failures
    poem_score = sigmoid - alpha * eta

    return poem_score
```

Listing 1: Fitness Prediction in PoEMNode

```python
def evolve_model(self):
    # Evolve using feedback (simplified)
    if len(set(self.fitness_history)) > 1:  # Ensure both classes
        self.model.fit(np.array(self.feature_history), np.array(self
    .fitness_history))
    self.feature_history.clear()
    self.fitness_history.clear()
```

Listing 2: Model Evolution in PoEMNode

```python
def one_round(nodes):
    print("\n--- PoEM Consensus Round Start ---\n")

    all_features = {}

    # Step 1: Each node collects features
    for name, node in nodes.items():
        all_features[name] = node.collect_features()

    # Step 2 & 3: Each node predicts fitness and votes
    all_votes = []
    for voter_name, voter_node in nodes.items():
        scores = {n: voter_node.predict_fitness(all_features[n]) for n in nodes}
        voted = vote_strategy(scores)
        all_votes.append(voted)
        print(f"{voter_name} votes for {voted}")

    # Step 4: Aggregate votes
    vote_counts = Counter(all_votes)
    winner = vote_counts.most_common(1)[0][0]
    print(f"\nWinner of consensus round: {winner}\n")

    # Step 5: Propose block (simulated as broadcast)
    print(f"{winner} proposes a block")

    # Step 6: Validate block
    for name in nodes:
        if name != winner:
            print(f"{name} validates block from {winner}     [VALID]")

    # Step 7: Model evolution
    for name, node in nodes.items():
        node.feature_history.append(all_features[name])
        node.fitness_history.append(1 if name == winner else 0)
        node.evolve_model()

    print("\n--- PoEM Consensus Round End ---\n")
```

Listing 3: One Round of PoEM Consensus

10 Consensus Process Lifecycle

The PoEM consensus process follows a lifecycle designed to minimize overhead while maintaining trust and transparency in block validation. The lifecycle consists of the following stages:

1. **Monitoring and Data Collection:** The system continuously monitors node activity, resource availability, and network conditions.

2. **Candidate Evaluation:** Based on the collected data, a set of eligible nodes is evaluated using the genetic algorithm and fitness function.
3. **Miner Selection:** One or more optimal nodes are selected as block producers.
4. **Block Proposal:** The selected miner(s) propose a new block containing validated transactions.
5. **Block Verification:** Neighboring or designated validator nodes verify the proposed block for correctness and legitimacy.
6. **Block Finalization and Broadcast:** Once validated, the block is finalized and added to the blockchain, and the new state is broadcasted across the network.
7. **Learning and Adaptation:** Feedback from this round is used to update the learning models for future miner selection.

This lifecycle ensures that PoEM remains lightweight and adaptive, making it ideal for dynamic, resource-constrained IoT environments.

11 Output

In our project, we are using Mininet as the network emulator and Ryu as the SDN controller. The Ryu controller is used to handle custom logic for packet forwarding and traffic flow. Below are the steps and outputs captured from our setup:

11.1 Starting the Ryu Controller

We start the Ryu controller using the following command:

```
ryu-manager myapp.py
```

11.2 Creating Topology using Mininet

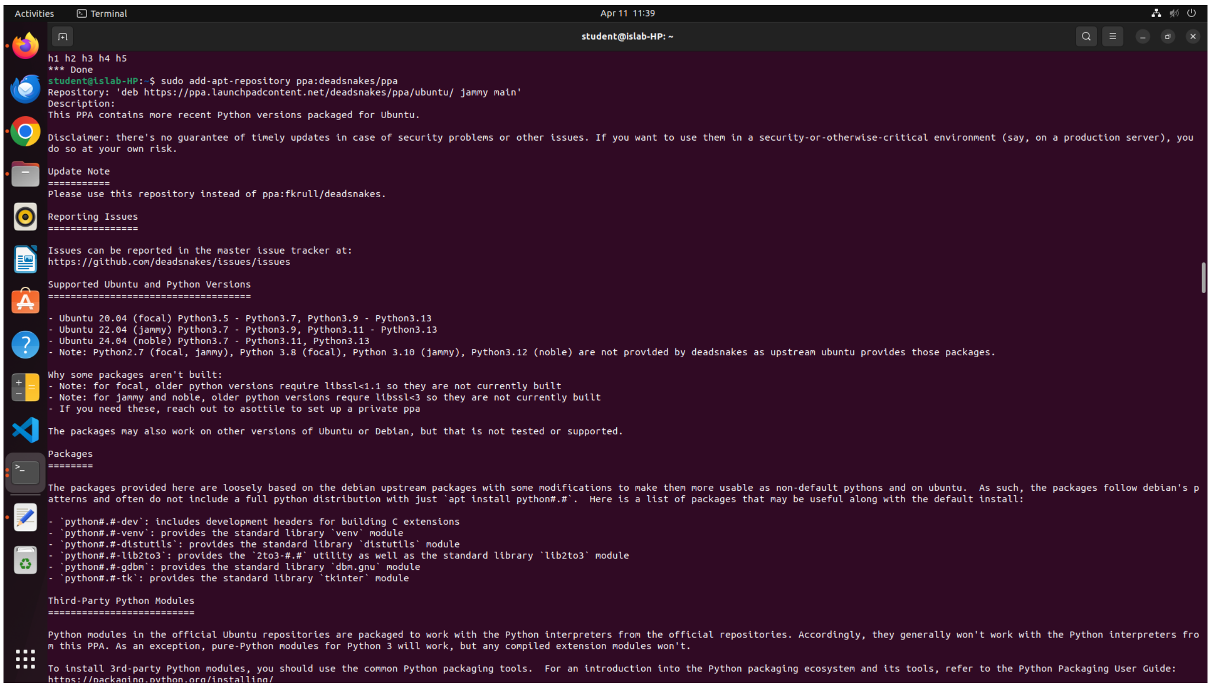
Mininet is launched with a custom topology or default topology. For example:

```
sudo mn --controller=remote --topo single,3
```

11.3 Ping Test Between Hosts

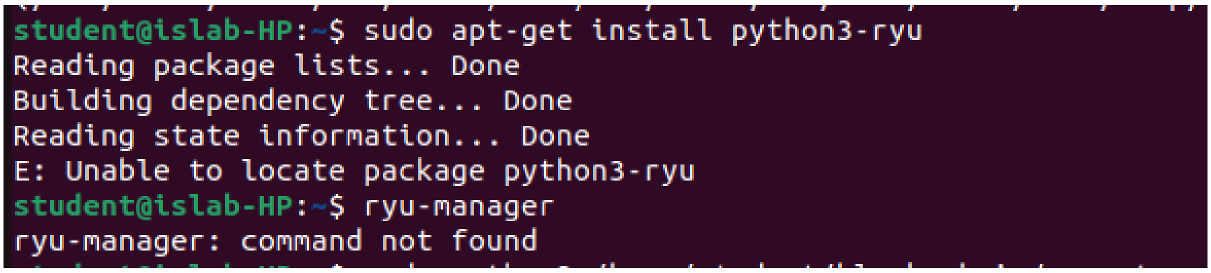
A simple ping test verifies connectivity between hosts:

```
mininet> h1 ping h2
```



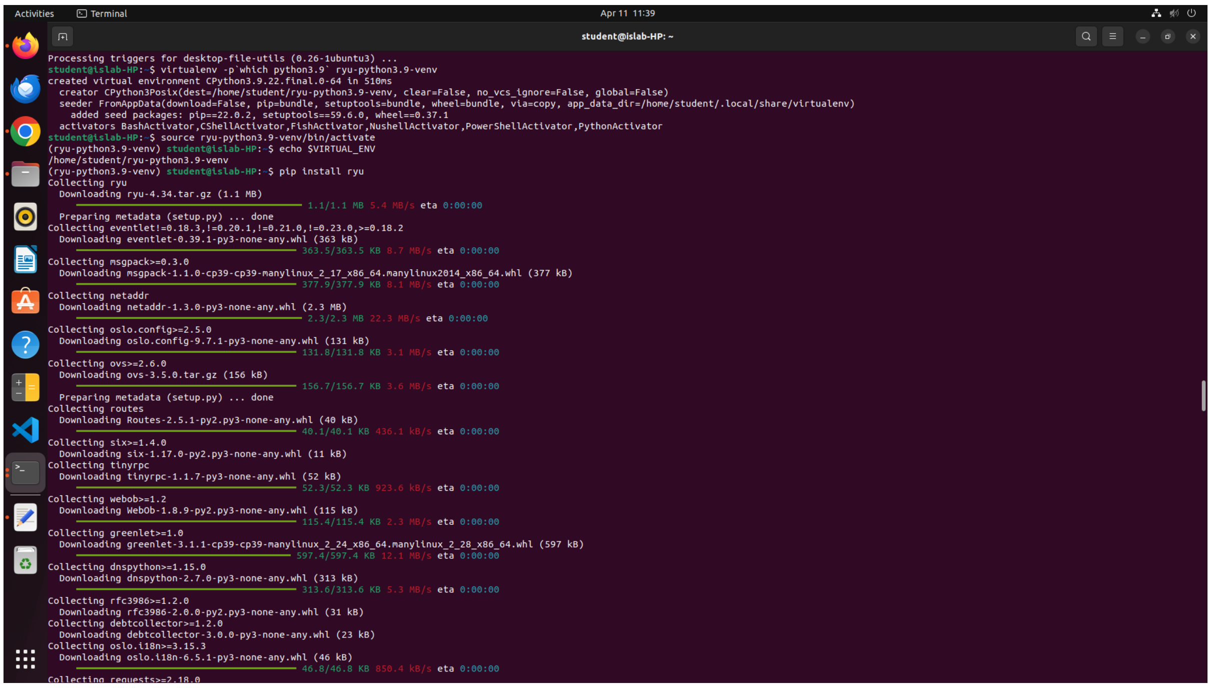
```
h1 h2 h3 h4 h5
*** Done
student@islab-HP:~$ sudo add-apt-repository ppa:deadsnakes/ppa
Repository: 'deb https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu/ jammy main'
Description:
This PPA contains more recent Python versions packaged for Ubuntu.

Disclaimer: there's no guarantee of timely updates in case of security problems or other issues. If you want to use them in a security-or-otherwise-critical environment (say, on a production server), you
do so at your own risk.

Update Note
===========
Please use this repository instead of ppa:fkrull/deadsnakes.

Reporting Issues
================

Issues can be reported in the master issue tracker at:
https://github.com/deadsnakes/issues/issues

Supported Ubuntu and Python Versions
====================================

- Ubuntu 20.04 (focal) Python3.5 - Python3.7, Python3.9 - Python3.13
- Ubuntu 22.04 (jammy) Python3.7 - Python3.9, Python3.11 - Python3.13
- Ubuntu 24.04 (noble) Python3.7 - Python3.11, Python3.13
- Note: Python2.7 (focal, jammy), Python 3.8 (focal), Python 3.10 (jammy), Python3.12 (noble) are not provided by deadsnakes as upstream ubuntu provides those packages.

Why some packages aren't built:
- Note: for focal, older python versions require libssl<1.1 so they are not currently built
- Note: for jammy and noble, older python versions requre libssl<3 so they are not currently built
- If you need these, reach out to asottile to set up a private ppa

The packages may also work on other versions of Ubuntu or Debian, but that is not tested or supported.

Packages
========

The packages provided here are loosely based on the debian upstream packages with some modifications to make them more usable as non-default pythons and on ubuntu.  As such, the packages follow debian's p
atterns and often do not include a full python distribution with just `apt install python#.#`.  Here is a list of packages that may be useful along with the default install:

- `python#.#-dev`: includes development headers for building C extensions
- `python#.#-venv`: provides the standard library `venv` module
- `python#.#-distutils`: provides the standard library `distutils` module
- `python#.#-lib2to3`: provides the `2to3-#.#` utility as well as the standard library `lib2to3` module
- `python#.#-gdbm`: provides the standard library `dbm.gnu` module
- `python#.#-tk`: provides the standard library `tkinter` module

Third-Party Python Modules
==========================

Python modules in the official Ubuntu repositories are packaged to work with the Python interpreters from the official repositories. Accordingly, they generally won't work with the Python interpreters fro
m this PPA. As an exception, pure-Python modules for Python 3 will work, but any compiled extension modules won't.

To install 3rd-party Python modules, you should use the common Python packaging tools.  For an introduction into the Python packaging ecosystem and its tools, refer to the Python Packaging User Guide:
https://packaging.python.org/installing/
```

Figure 1: Step-1 Ryu Controller Installation



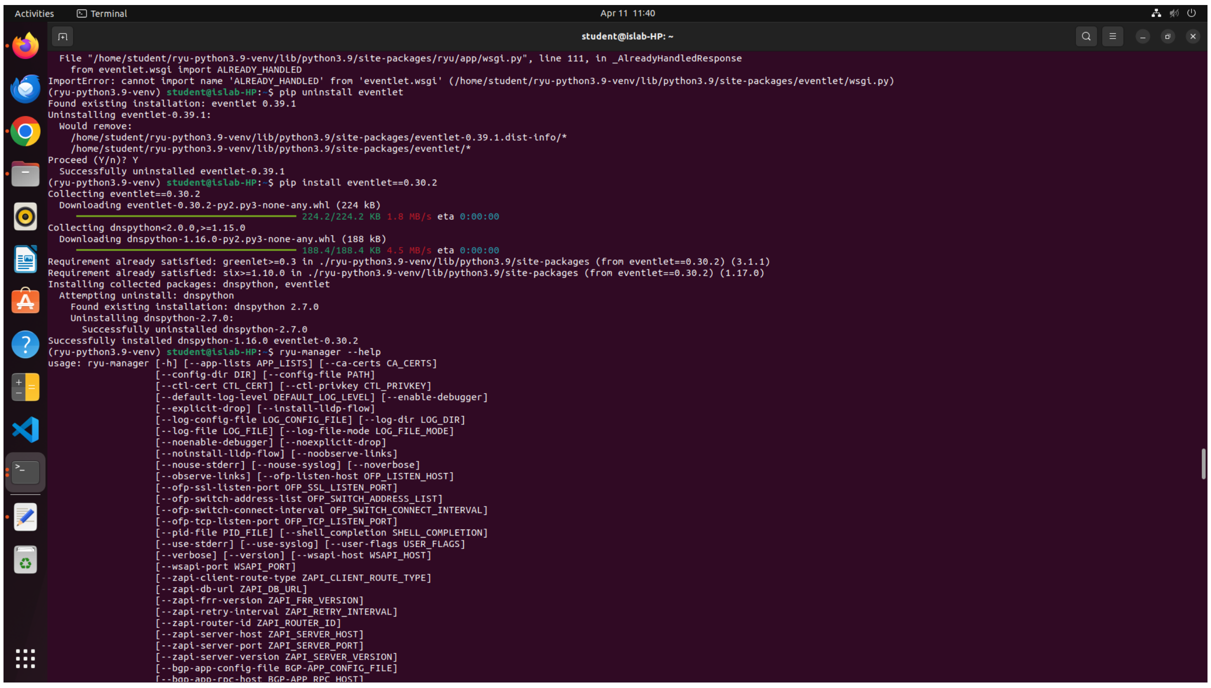
```
student@islab-HP:~$ sudo apt-get install python3-ryu
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package python3-ryu
student@islab-HP:~$ ryu-manager
ryu-manager: command not found
```

Figure 2: Step-2



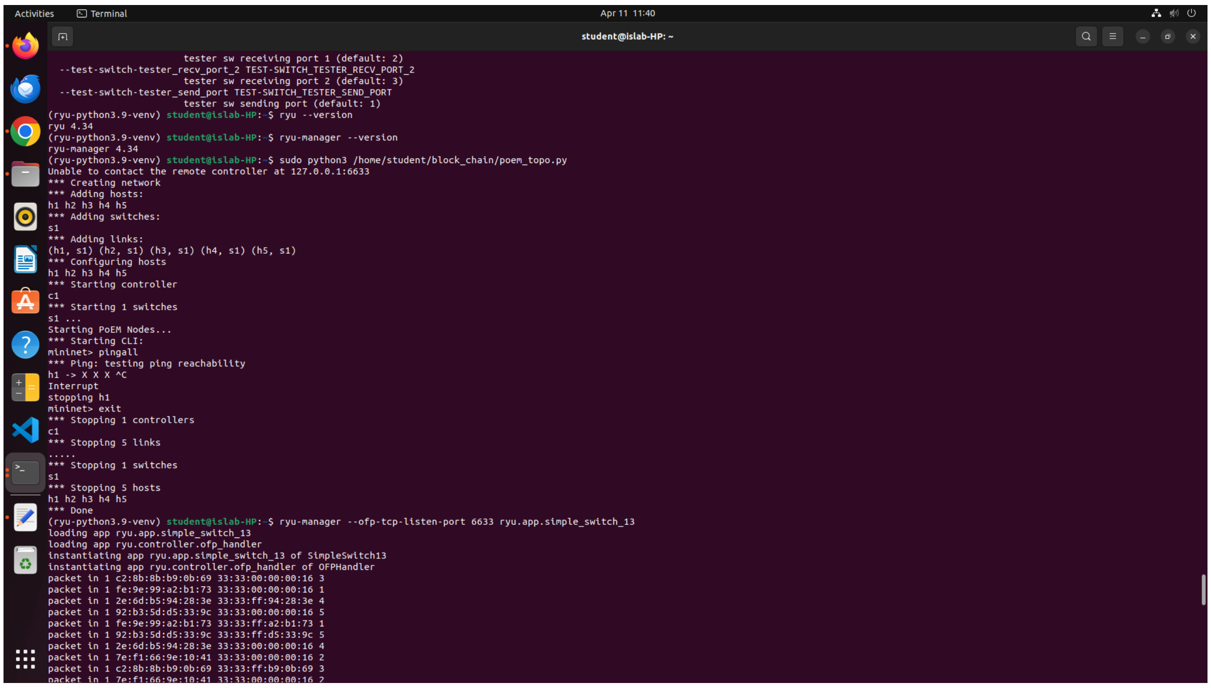
```
Processing triggers for desktop-file-utils (0.26-1ubuntu3) ...
student@islab-HP:~$ virtualenv -p`which python3.9` ryu-python3.9-venv
created virtual environment CPython3.9.22.final.0-64 in 510ms
  creator CPython3Posix(dest=/home/student/ryu-python3.9-venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/student/.local/share/virtualenv)
    added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
student@islab-HP:~$ source ryu-python3.9-venv/bin/activate
(ryu-python3.9-venv) student@islab-HP:~$ echo $VIRTUAL_ENV
/home/student/ryu-python3.9-venv
(ryu-python3.9-venv) student@islab-HP:~$ pip install ryu
Collecting ryu
  Downloading ryu-4.34.tar.gz (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 5.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting eventlet!=0.18.3,!=0.20.1,!=0.21.0,!=0.23.0,>=0.18.2
  Downloading eventlet-0.39.1-py3-none-any.whl (363 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.5/363.5 KB 8.7 MB/s eta 0:00:00
Collecting msgpack>=0.3.0
  Downloading msgpack-1.1.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (377 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 377.9/377.9 KB 8.1 MB/s eta 0:00:00
Collecting netaddr
  Downloading netaddr-1.3.0-py3-none-any.whl (2.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 22.3 MB/s eta 0:00:00
Collecting oslo.config>=2.5.0
  Downloading oslo.config-9.7.1-py3-none-any.whl (131 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 131.8/131.8 KB 3.1 MB/s eta 0:00:00
Collecting ovs>=2.6.0
  Downloading ovs-3.5.0.tar.gz (156 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 156.7/156.7 KB 3.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting routes
  Downloading Routes-2.5.1-py2.py3-none-any.whl (40 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 40.1/40.1 KB 436.1 kB/s eta 0:00:00
Collecting six>=1.4.0
  Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Collecting tinyrpc
  Downloading tinyrpc-1.1.7-py3-none-any.whl (52 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.3/52.3 KB 923.6 kB/s eta 0:00:00
Collecting webob>=1.2
  Downloading WebOb-1.8.9-py2.py3-none-any.whl (115 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.4/115.4 KB 2.3 MB/s eta 0:00:00
Collecting greenlet>=1.0
  Downloading greenlet-3.1.1-cp39-cp39-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (597 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 597.4/597.4 KB 12.1 MB/s eta 0:00:00
Collecting dnspython>=1.15.0
  Downloading dnspython-2.7.0-py3-none-any.whl (313 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 313.6/313.6 KB 5.3 MB/s eta 0:00:00
Collecting rfc3986>=1.2.0
  Downloading rfc3986-2.0.0-py2.py3-none-any.whl (31 kB)
Collecting debtcollector>=1.2.0
  Downloading debtcollector-3.0.0-py3-none-any.whl (23 kB)
Collecting oslo.i18n>=3.15.3
  Downloading oslo.i18n-6.5.1-py3-none-any.whl (46 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.8/46.8 KB 850.4 kB/s eta 0:00:00
Collecting requests>=2.18.0
```

Figure 3: Step-3 Creating Virtual environment



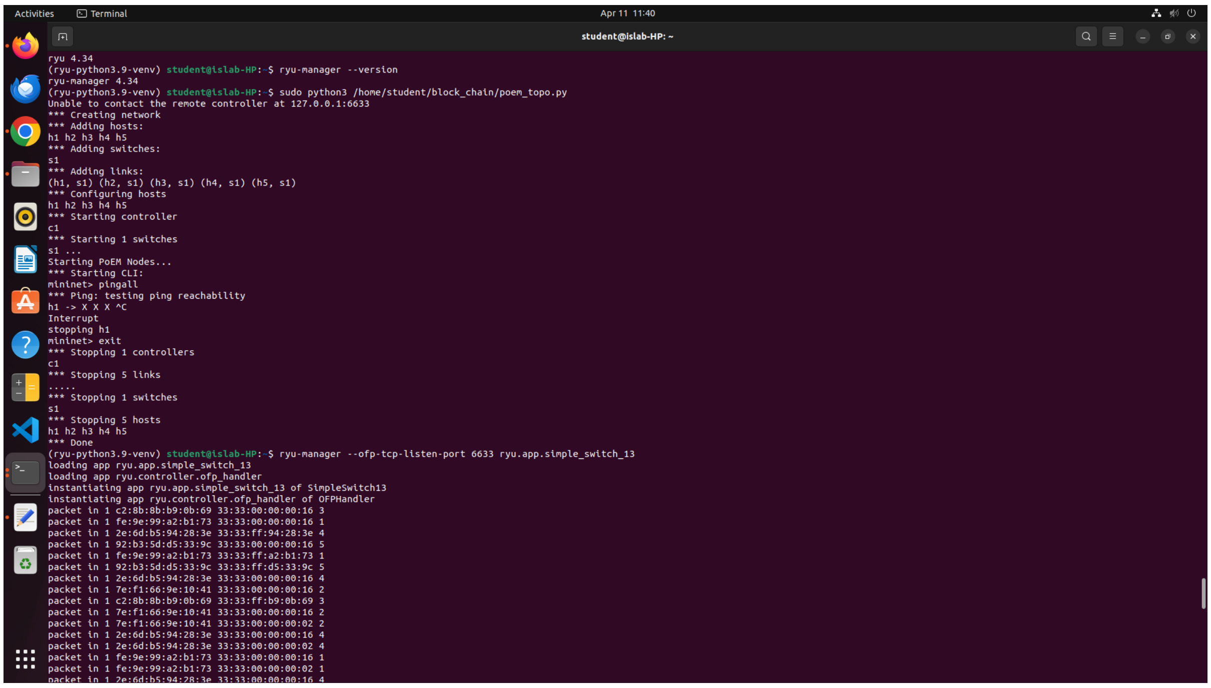
```
  File "/home/student/ryu-python3.9-venv/lib/python3.9/site-packages/ryu/app/wsgi.py", line 111, in _AlreadyHandledResponse
    from eventlet.wsgi import ALREADY_HANDLED
ImportError: cannot import name 'ALREADY_HANDLED' from 'eventlet.wsgi' (/home/student/ryu-python3.9-venv/lib/python3.9/site-packages/eventlet/wsgi.py)
(ryu-python3.9-venv) student@islab-HP:~$ pip uninstall eventlet
Found existing installation: eventlet 0.39.1
Uninstalling eventlet-0.39.1:
  Would remove:
    /home/student/ryu-python3.9-venv/lib/python3.9/site-packages/eventlet-0.39.1.dist-info/*
    /home/student/ryu-python3.9-venv/lib/python3.9/site-packages/eventlet/*
Proceed (Y/n)? Y
  Successfully uninstalled eventlet-0.39.1
(ryu-python3.9-venv) student@islab-HP:~$ pip install eventlet==0.30.2
Collecting eventlet==0.30.2
  Downloading eventlet-0.30.2-py2.py3-none-any.whl (224 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 224.2/224.2 KB 1.8 MB/s eta 0:00:00
Collecting dnspython<2.0.0,>=1.15.0
  Downloading dnspython-1.16.0-py2.py3-none-any.whl (188 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.4/188.4 KB 4.5 MB/s eta 0:00:00
Requirement already satisfied: greenlet>=0.3 in ./ryu-python3.9-venv/lib/python3.9/site-packages (from eventlet==0.30.2) (3.1.1)
Requirement already satisfied: six>=1.10.0 in ./ryu-python3.9-venv/lib/python3.9/site-packages (from eventlet==0.30.2) (1.17.0)
Installing collected packages: dnspython, eventlet
  Attempting uninstall: dnspython
    Found existing installation: dnspython 2.7.0
    Uninstalling dnspython-2.7.0:
      Successfully uninstalled dnspython-2.7.0
Successfully installed dnspython-1.16.0 eventlet-0.30.2
(ryu-python3.9-venv) student@islab-HP:~$ ryu-manager --help
usage: ryu-manager [-h] [--app-lists APP_LISTS] [--ca-certs CA_CERTS]
                   [--config-dir DIR] [--config-file PATH]
                   [--ctl-cert CTL_CERT] [--ctl-privkey CTL_PRIVKEY]
                   [--default-log-level DEFAULT_LOG_LEVEL] [--enable-debugger]
                   [--explicit-drop] [--install-lldp-flow]
                   [--log-config-file LOG_CONFIG_FILE] [--log-dir LOG_DIR]
                   [--log-file LOG_FILE] [--log-file-mode LOG_FILE_MODE]
                   [--noenable-debugger] [--noexplicit-drop]
                   [--noinstall-lldp-flow] [--noobserve-links]
                   [--nouse-stderr] [--nouse-syslog] [--noverbose]
                   [--observe-links] [--ofp-listen-host OFP_LISTEN_HOST]
                   [--ofp-ssl-listen-port OFP_SSL_LISTEN_PORT]
                   [--ofp-switch-address-list OFP_SWITCH_ADDRESS_LIST]
                   [--ofp-switch-connect-interval OFP_SWITCH_CONNECT_INTERVAL]
                   [--ofp-tcp-listen-port OFP_TCP_LISTEN_PORT]
                   [--pid-file PID_FILE] [--shell_completion SHELL_COMPLETION]
                   [--use-stderr] [--use-syslog] [--user-flags USER_FLAGS]
                   [--verbose] [--version] [--wsapi-host WSAPI_HOST]
                   [--wsapi-port WSAPI_PORT]
                   [--zapi-client-route-type ZAPI_CLIENT_ROUTE_TYPE]
                   [--zapi-db-url ZAPI_DB_URL]
                   [--zapi-frr-version ZAPI_FRR_VERSION]
                   [--zapi-retry-interval ZAPI_RETRY_INTERVAL]
                   [--zapi-router-id ZAPI_ROUTER_ID]
                   [--zapi-server-host ZAPI_SERVER_HOST]
                   [--zapi-server-port ZAPI_SERVER_PORT]
                   [--zapi-server-version ZAPI_SERVER_VERSION]
                   [--bgp-app-config-file BGP-APP_CONFIG_FILE]
                   [--bgp-app-rpc-host BGP-APP_RPC_HOST]
```

Figure 4: Step-4



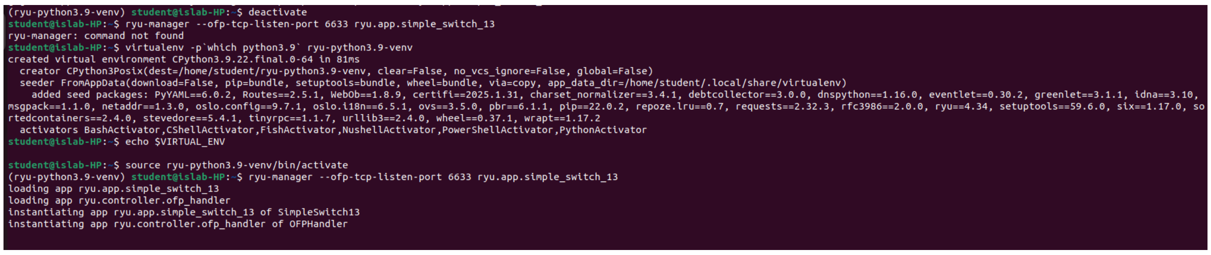
```
                        tester sw receiving port 1 (default: 2)
  --test-switch-tester_recv_port_2 TEST-SWITCH_TESTER_RECV_PORT_2
                        tester sw receiving port 2 (default: 3)
  --test-switch-tester_send_port TEST-SWITCH_TESTER_SEND_PORT
                        tester sw sending port (default: 1)
(ryu-python3.9-venv) student@islab-HP:~$ ryu --version
ryu 4.34
(ryu-python3.9-venv) student@islab-HP:~$ ryu-manager --version
ryu-manager 4.34
(ryu-python3.9-venv) student@islab-HP:~$ sudo python3 /home/student/block_chain/poem_topo.py
Unable to contact the remote controller at 127.0.0.1:6633
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c1
*** Starting 1 switches
s1 ...
Starting PoEM Nodes...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X ^C
Interrupt
stopping h1
mininet> exit
*** Stopping 1 controllers
c1
*** Stopping 5 links
.....
*** Stopping 1 switches
s1
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
(ryu-python3.9-venv) student@islab-HP:~$ ryu-manager --ofp-tcp-listen-port 6633 ryu.app.simple_switch_13
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 1 c2:8b:8b:b9:0b:69 33:33:00:00:00:16 3
packet in 1 fe:9e:99:a2:b1:73 33:33:00:00:00:16 1
packet in 1 2e:6d:b5:94:28:3e 33:33:ff:94:28:3e 4
packet in 1 92:b3:5d:d5:33:9c 33:33:00:00:00:16 5
packet in 1 fe:9e:99:a2:b1:73 33:33:ff:a2:b1:73 1
packet in 1 92:b3:5d:d5:33:9c 33:33:ff:d5:33:9c 5
packet in 1 2e:6d:b5:94:28:3e 33:33:00:00:00:16 4
packet in 1 7e:f1:66:9e:10:41 33:33:00:00:00:16 2
packet in 1 c2:8b:8b:b9:0b:69 33:33:ff:b9:0b:69 3
packet in 1 7e:f1:66:9e:10:41 33:33:00:00:00:16 2
```

Figure 5: Ping Test



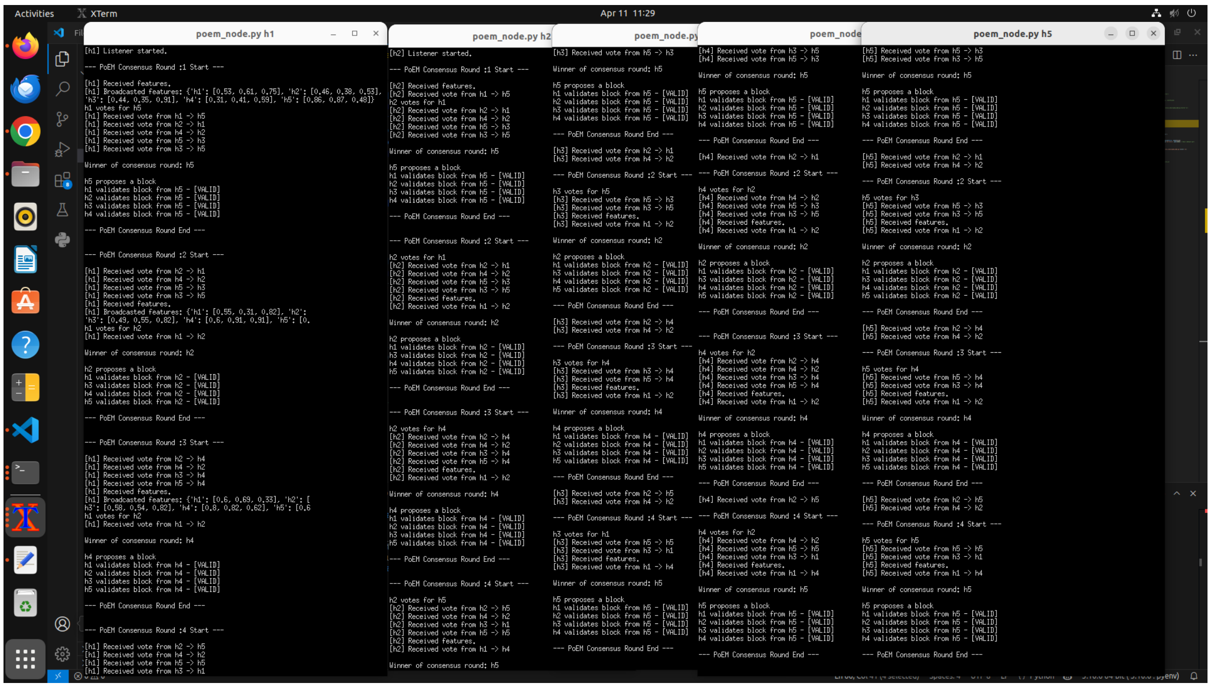
```
ryu 4.34
(ryu-python3.9-venv) student@islab-HP:~$ ryu-manager --version
ryu-manager 4.34
(ryu-python3.9-venv) student@islab-HP:~$ sudo python3 /home/student/block_chain/poem_topo.py
Unable to contact the remote controller at 127.0.0.1:6633
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1)
*** Configuring hosts
h1 h2 h3 h4 h5
*** Starting controller
c1
*** Starting 1 switches
s1 ...
Starting PoEM Nodes...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X X ^C
Interrupt
stopping h1
mininet> exit
*** Stopping 1 controllers
c1
*** Stopping 5 links
.....
*** Stopping 1 switches
s1
*** Stopping 5 hosts
h1 h2 h3 h4 h5
*** Done
(ryu-python3.9-venv) student@islab-HP:~$ ryu-manager --ofp-tcp-listen-port 6633 ryu.app.simple_switch_13
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
packet in 1 c2:8b:8b:b9:0b:69 33:33:00:00:00:16 3
packet in 1 fe:9e:99:a2:b1:73 33:33:00:00:00:16 1
packet in 1 2e:6d:b5:94:28:3e 33:33:ff:94:28:3e 4
packet in 1 92:b3:5d:d5:33:9c 33:33:00:00:00:16 5
packet in 1 fe:9e:99:a2:b1:73 33:33:ff:a2:b1:73 1
packet in 1 92:b3:5d:d5:33:9c 33:33:ff:d5:33:9c 5
packet in 1 2e:6d:b5:94:28:3e 33:33:00:00:00:16 4
packet in 1 7e:f1:66:9e:10:41 33:33:00:00:00:16 2
packet in 1 c2:8b:8b:b9:0b:69 33:33:ff:b9:0b:69 3
packet in 1 7e:f1:66:9e:10:41 33:33:00:00:00:16 2
packet in 1 7e:f1:66:9e:10:41 33:33:00:00:00:02 2
packet in 1 2e:6d:b5:94:28:3e 33:33:00:00:00:16 4
packet in 1 2e:6d:b5:94:28:3e 33:33:00:00:00:02 4
packet in 1 fe:9e:99:a2:b1:73 33:33:00:00:00:16 1
packet in 1 fe:9e:99:a2:b1:73 33:33:00:00:00:02 1
packet in 1 2e:6d:b5:94:28:3e 33:33:00:00:00:16 4
```

Figure 6: Loading Ryu Controller



```
(ryu-python3.9-venv) student@islab-HP:~$ deactivate
student@islab-HP:~$ ryu-manager --ofp-tcp-listen-port 6633 ryu.app.simple_switch_13
ryu-manager: command not found
student@islab-HP:~$ virtualenv -p`which python3.9` ryu-python3.9-venv
created virtual environment CPython3.9.22.final.0-64 in 81ms
  creator CPython3Posix(dest=/home/student/ryu-python3.9-venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/student/.local/share/virtualenv)
    added seed packages: PyYAML==6.0.2, Routes==2.5.1, WebOb==1.8.9, certifi==2025.1.31, charset_normalizer==3.4.1, debtcollector==3.0.0, dnspython==1.16.0, eventlet==0.30.2, greenlet==3.1.1, idna==3.10,
msgpack==1.1.0, netaddr==1.3.0, oslo.config==9.7.1, oslo.i18n==6.5.1, ovs==3.5.0, pbr==6.1.1, pip==22.0.2, repoze.lru==0.7, requests==2.32.3, rfc3986==2.0.0, ryu==4.34, setuptools==59.6.0, six==1.17.0, so
rtedcontainers==2.4.0, stevedore==5.4.1, tinyrpc==1.1.7, urllib3==2.4.0, wheel==0.37.1, wrapt==1.17.2
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
student@islab-HP:~$ echo $VIRTUAL_ENV

student@islab-HP:~$ source ryu-python3.9-venv/bin/activate
(ryu-python3.9-venv) student@islab-HP:~$ ryu-manager --ofp-tcp-listen-port 6633 ryu.app.simple_switch_13
loading app ryu.app.simple_switch_13
loading app ryu.controller.ofp_handler
instantiating app ryu.app.simple_switch_13 of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figure 7: Starting Ryu Manager



```
[h1] Listener started.

--- PoEM Consensus Round :1 Start ---

[h1] Received features.
[h1] Broadcasted features: {'h1': [0.53, 0.61, 0.75], 'h2': [0.46, 0.38, 0.53],
'h3': [0.44, 0.35, 0.91], 'h4': [0.31, 0.41, 0.59], 'h5': [0.86, 0.87, 0.48]}
h1 votes for h5
[h1] Received vote from h1 -> h5
[h1] Received vote from h2 -> h1
[h1] Received vote from h4 -> h2
[h1] Received vote from h5 -> h3
[h1] Received vote from h3 -> h5

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :2 Start ---

[h1] Received vote from h2 -> h1
[h1] Received vote from h4 -> h2
[h1] Received vote from h5 -> h3
[h1] Received vote from h3 -> h5
[h1] Received features.
[h1] Broadcasted features: {'h1': [0.55, 0.31, 0.82], 'h2':
'h3': [0.49, 0.55, 0.82], 'h4': [0.6, 0.91, 0.91], 'h5': [0.
h1 votes for h2
[h1] Received vote from h1 -> h2

Winner of consensus round: h2

h2 proposes a block
h1 validates block from h2 - [VALID]
h3 validates block from h2 - [VALID]
h4 validates block from h2 - [VALID]
h5 validates block from h2 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :3 Start ---

[h1] Received vote from h2 -> h4
[h1] Received vote from h4 -> h2
[h1] Received vote from h3 -> h4
[h1] Received vote from h5 -> h4
[h1] Received features.
[h1] Broadcasted features: {'h1': [0.6, 0.69, 0.33], 'h2': [
h3': [0.58, 0.54, 0.82], 'h4': [0.8, 0.82, 0.62], 'h5': [0.6
h1 votes for h2
[h1] Received vote from h1 -> h2

Winner of consensus round: h4

h4 proposes a block
h1 validates block from h4 - [VALID]
h2 validates block from h4 - [VALID]
h3 validates block from h4 - [VALID]
h5 validates block from h4 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :4 Start ---

[h1] Received vote from h2 -> h5
[h1] Received vote from h4 -> h2
[h1] Received vote from h5 -> h5
[h1] Received vote from h3 -> h1
```

```
[h2] Listener started.

--- PoEM Consensus Round :1 Start ---

[h2] Received features.
[h2] Received vote from h1 -> h5
h2 votes for h1
[h2] Received vote from h2 -> h1
[h2] Received vote from h4 -> h2
[h2] Received vote from h5 -> h3
[h2] Received vote from h3 -> h5

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :2 Start ---

h2 votes for h1
[h2] Received vote from h2 -> h1
[h2] Received vote from h4 -> h2
[h2] Received vote from h5 -> h3
[h2] Received vote from h3 -> h5
[h2] Received features.
[h2] Received vote from h1 -> h2

Winner of consensus round: h2

h2 proposes a block
h1 validates block from h2 - [VALID]
h3 validates block from h2 - [VALID]
h4 validates block from h2 - [VALID]
h5 validates block from h2 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :3 Start ---

h2 votes for h4
[h2] Received vote from h2 -> h4
[h2] Received vote from h4 -> h2
[h2] Received vote from h3 -> h4
[h2] Received vote from h5 -> h4
[h2] Received features.
[h2] Received vote from h1 -> h2

Winner of consensus round: h4

h4 proposes a block
h1 validates block from h4 - [VALID]
h2 validates block from h4 - [VALID]
h3 validates block from h4 - [VALID]
h5 validates block from h4 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :4 Start ---

h2 votes for h5
[h2] Received vote from h2 -> h5
[h2] Received vote from h4 -> h2
[h2] Received vote from h3 -> h1
[h2] Received vote from h5 -> h5
[h2] Received features.
[h2] Received vote from h1 -> h4

Winner of consensus round: h5
```

```
[h3] Received vote from h5 -> h3

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---

[h3] Received vote from h2 -> h1
[h3] Received vote from h4 -> h2

--- PoEM Consensus Round :2 Start ---

h3 votes for h5
[h3] Received vote from h5 -> h3
[h3] Received vote from h3 -> h5
[h3] Received features.
[h3] Received vote from h1 -> h2

Winner of consensus round: h2

h2 proposes a block
h1 validates block from h2 - [VALID]
h3 validates block from h2 - [VALID]
h4 validates block from h2 - [VALID]
h5 validates block from h2 - [VALID]

--- PoEM Consensus Round End ---

[h3] Received vote from h2 -> h4
[h3] Received vote from h4 -> h2

--- PoEM Consensus Round :3 Start ---

h3 votes for h4
[h3] Received vote from h3 -> h4
[h3] Received vote from h5 -> h4
[h3] Received features.
[h3] Received vote from h1 -> h2

Winner of consensus round: h4

h4 proposes a block
h1 validates block from h4 - [VALID]
h2 validates block from h4 - [VALID]
h3 validates block from h4 - [VALID]
h5 validates block from h4 - [VALID]

--- PoEM Consensus Round End ---

[h3] Received vote from h2 -> h5
[h3] Received vote from h4 -> h2

--- PoEM Consensus Round :4 Start ---

h3 votes for h1
[h3] Received vote from h5 -> h5
[h3] Received vote from h3 -> h1
[h3] Received features.
[h3] Received vote from h1 -> h4

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---
```

```
[h4] Received vote from h3 -> h5
[h4] Received vote from h5 -> h3

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---

[h4] Received vote from h2 -> h1

--- PoEM Consensus Round :2 Start ---

h4 votes for h2
[h4] Received vote from h4 -> h2
[h4] Received vote from h5 -> h3
[h4] Received vote from h3 -> h5
[h4] Received features.
[h4] Received vote from h1 -> h2

Winner of consensus round: h2

h2 proposes a block
h1 validates block from h2 - [VALID]
h3 validates block from h2 - [VALID]
h4 validates block from h2 - [VALID]
h5 validates block from h2 - [VALID]

--- PoEM Consensus Round End ---


--- PoEM Consensus Round :3 Start ---

h4 votes for h2
[h4] Received vote from h2 -> h4
[h4] Received vote from h4 -> h2
[h4] Received vote from h3 -> h4
[h4] Received vote from h5 -> h4
[h4] Received features.
[h4] Received vote from h1 -> h2

Winner of consensus round: h4

h4 proposes a block
h1 validates block from h4 - [VALID]
h2 validates block from h4 - [VALID]
h3 validates block from h4 - [VALID]
h5 validates block from h4 - [VALID]

--- PoEM Consensus Round End ---

[h4] Received vote from h2 -> h5

--- PoEM Consensus Round :4 Start ---

h4 votes for h2
[h4] Received vote from h4 -> h2
[h4] Received vote from h5 -> h5
[h4] Received vote from h3 -> h1
[h4] Received features.
[h4] Received vote from h1 -> h4

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---
```

```
[h5] Received vote from h5 -> h3
[h5] Received vote from h3 -> h5

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---

[h5] Received vote from h2 -> h1
[h5] Received vote from h4 -> h2

--- PoEM Consensus Round :2 Start ---

h5 votes for h3
[h5] Received vote from h5 -> h3
[h5] Received vote from h3 -> h5
[h5] Received features.
[h5] Received vote from h1 -> h2

Winner of consensus round: h2

h2 proposes a block
h1 validates block from h2 - [VALID]
h3 validates block from h2 - [VALID]
h4 validates block from h2 - [VALID]
h5 validates block from h2 - [VALID]

--- PoEM Consensus Round End ---

[h5] Received vote from h2 -> h4
[h5] Received vote from h4 -> h2

--- PoEM Consensus Round :3 Start ---

h5 votes for h4
[h5] Received vote from h5 -> h4
[h5] Received vote from h3 -> h4
[h5] Received features.
[h5] Received vote from h1 -> h2

Winner of consensus round: h4

h4 proposes a block
h1 validates block from h4 - [VALID]
h2 validates block from h4 - [VALID]
h3 validates block from h4 - [VALID]
h5 validates block from h4 - [VALID]

--- PoEM Consensus Round End ---

[h5] Received vote from h2 -> h5
[h5] Received vote from h4 -> h2

--- PoEM Consensus Round :4 Start ---

h5 votes for h5
[h5] Received vote from h5 -> h5
[h5] Received vote from h3 -> h1
[h5] Received features.
[h5] Received vote from h1 -> h4

Winner of consensus round: h5

h5 proposes a block
h1 validates block from h5 - [VALID]
h2 validates block from h5 - [VALID]
h3 validates block from h5 - [VALID]
h4 validates block from h5 - [VALID]

--- PoEM Consensus Round End ---
```

Figure 8: Output on Every Xterm Node

12 Conclusion

In this report, we explored the growing demand for efficient and adaptive consensus mechanisms in Blockchain-as-a-Service (BaaS) environments, especially those integrated with IoT devices. Traditional consensus algorithms like Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT), although foundational, suffer from critical drawbacks such as high energy consumption, scalability limitations, and unsuitability in highly dynamic, resource-constrained environments like IoT.

The proposed **PoEM (Proof of Evolutionary Model)** consensus protocol addresses these issues by using a machine learning-based Constrained Learning to Rank (CLTR) model to dynamically and intelligently select block producers. Unlike traditional methods, PoEM introduces a lightweight, energy-efficient mechanism that supports both permissioned and permissionless systems, allows dynamic node entry/exit, and improves overall consensus efficiency. It strikes a balance between decentralization and performance, making it highly practical for real-world BaaS-based IoT systems.

References

1. Zhao, X., Liu, Y., Yang, H., & Zhang, J. (2023). *Lightweight Model-Based Evolutionary Consensus Protocol in Blockchain as a Service for IoT*. IEEE Transactions on Industrial Informatics.