Report for “Peer-to-Peer Backup”

Anita Ghezel Ayagh Tehrani
Mohammadhosein Abdollahi

1. Overview of Assignment Objectives:

   The primary objective of this assignment is to simulate a peer-to-peer (P2P) system utilizing erasure coding for data storage. In a P2P system, multiple machines, referred to as peers, collaborate to store and share data without the need for a centralized server. Erasure coding is a technique used to protect data against node failures by encoding it into a set of redundant blocks, allowing for data reconstruction even if some blocks are lost.

2. Key Components of the Simulation:
   - Peers:
     - Peers represent individual machines participating in the P2P system. They are responsible for storing data, connecting, and disconnecting from the network, and sharing encoded blocks with other peers.
   - Data Encoding:
     - Data encoding involves transforming original data into encoded blocks using erasure coding algorithms. These encoded blocks are distributed among peers to provide redundancy and fault tolerance.
   - Data Storage:
     - Peers store both original data and encoded blocks received from other peers. They utilize local storage resources to maintain data availability and facilitate data reconstruction in case of peer failures.
   - Failure Scenarios:
     - The simulation accounts for various failure scenarios that may occur in the P2P system, including peer failures and data loss. When a peer fails,

it loses all the data it has stored, necessitating the retrieval of encoded blocks from other peers to restore its data.

By simulating these key components, we aim to assess the effectiveness of the P2P system in maintaining data availability and reliability over extended periods, considering factors such as peer connectivity, data redundancy, and erasure coding parameters.

3. Completion of Peer-to-Peer Back up:
   - First, we tried to complete the existing codes and make them run smoothly and correctly with both configures.

4. Implementation of Comparison Method:
   - We decided to use the average lifetime as the comparison point. We implemented this method in our simulation for a different range of nodes. This method increases the average lifetime (10 to 110 years) of selected nodes. For example, 5 out of 10 nodes (50%) have variant average lifetime.
     After implementing this method, we count 5 ratios:
     I. The number of restored blocks (that belong to the nodes that have a variant lifetime) to all restored blocks ratio.
     II. The number of restored blocks (that belong to the nodes that have a non-variant lifetime) to all restored blocks ratio.
     III. The number of backed-up blocks (that belong to the nodes that have a variant lifetime) to all backed-up blocks ratio.
     IV. The number of backed-up blocks (that belong to the nodes that have a non-variant lifetime) to all backed-up blocks ratio.
     V. All restored blocks to all backed-up blocks ratio.

     For implementation, we assigned each node an ID to count with more accuracy. We put the counter for the restored and backed-up blocks after calling the schedule_transfer function and define each calling belongs to restored or backed-up blocks.

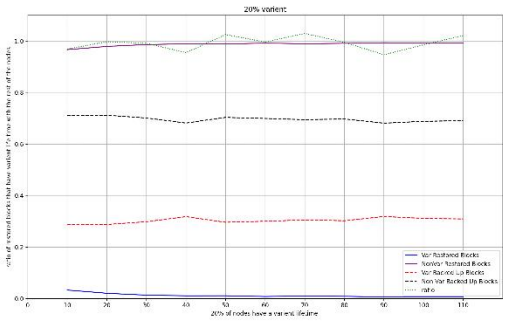
     Note 1: Our codes have this flexibility to change for different attributes of the nodes, but we decided to choose average lifetime because it was more interesting for us.

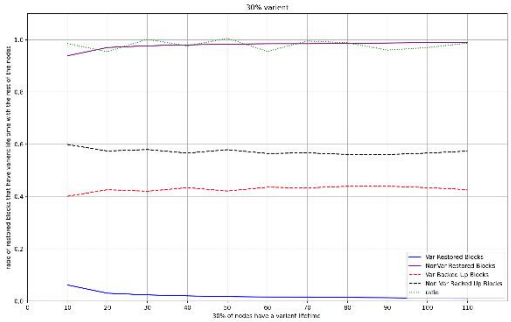
Note 2: Also, our codes have the flexibility to change the different percentages of variant nodes and it is discussed more in the plot section.

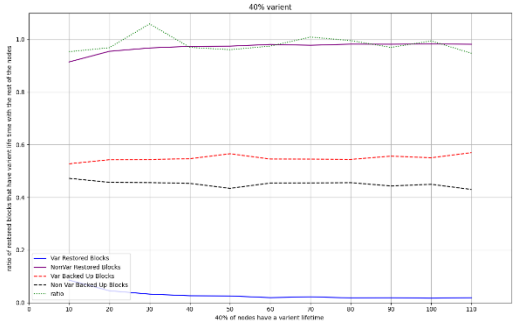
Note 3: we scheduled the simulation to run 10 times, so we have a more accurate result.

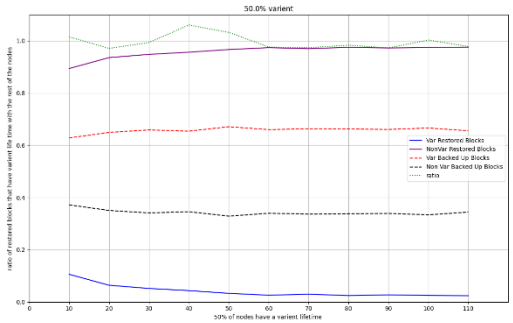
5. Plots of Peer-to-Peer Backup:

- Here are the plots of different percentages (20%, 30%, 40%, 50%) of variant nodes average lifetime:







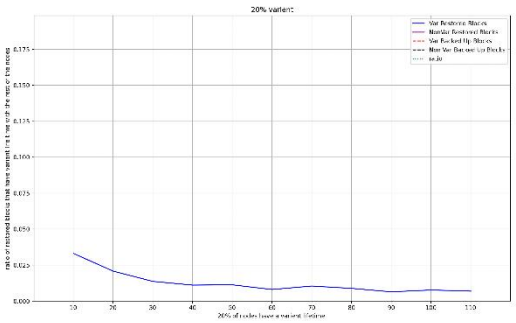


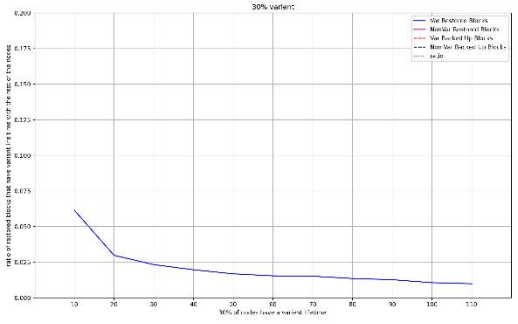
- As we see, restored blocks that had the variant lifetime decreased smoothly in different percentages with different starting points and closer to the 0 ratio when we have less variant lifetime in our nodes.
  On the other hand, restored blocks that had the non-variant lifetime completely wise versa.
  The backed-up blocks, almost have the same behavior during the simulation but the surprising point is when we have more than 40% of nodes that have a variant lifetime, the number of the blocks that backed up and the nodes that

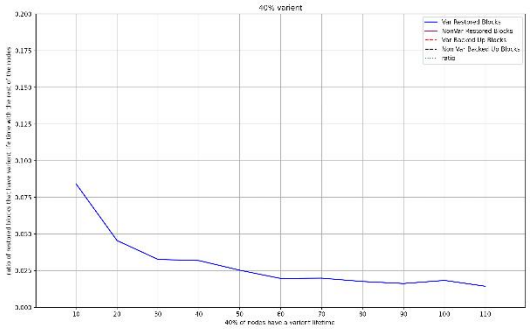
contain these blocks have a variant lifetime, is more than the other side. It is different from what we see in the under 40% plots.

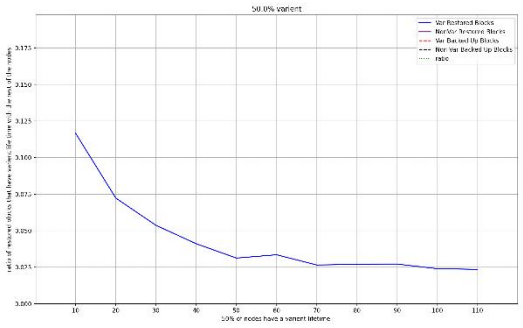
The ratio between restored and backed-up blocks is always around 1. Moreover, while the lifetime increases, the ratio is mostly under 1 because the nodes need less to be restored rather than backed up.

- Here are the plots with closer views of variant restored blocks:

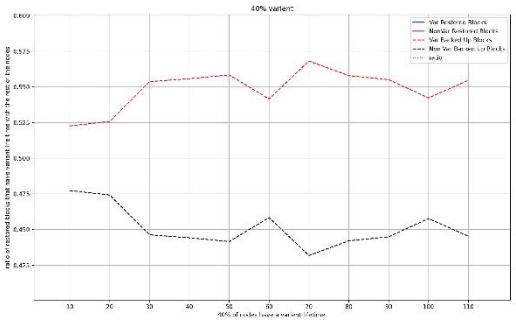


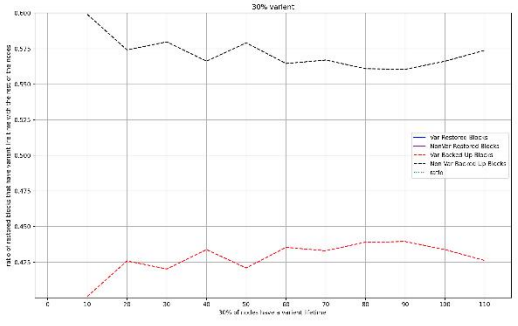




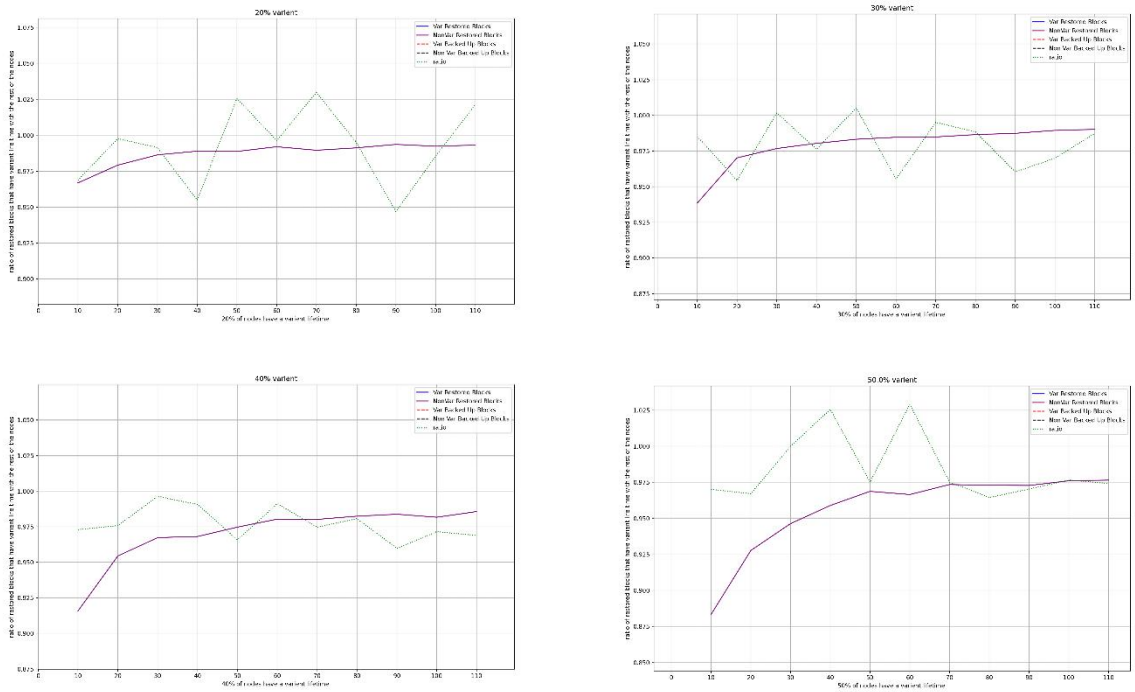


- Here are the plots with closer views of variant and non-variant backed-up blocks:





- Here are the plots with closer views of non-variant restored blocks and ratios:



6. Implementation of Peer-to-Peer Backup with Extension:
   - We decided to choose the selfish node. The selfish node has different definitions in different systems based on our needs. The definition of the selfish node that we implemented in our system is the selfish node doesn't upload anything. We add an attribute in the config file and the code-named selfish for the nodes and it is Boolean.
     First, we randomly select 4 nodes and change their selfish attribute to true. Then we count the number of nodes that are selfish and have a variant lifetime.

```
random_number = 4 #random.randint(3, 5)
countte = [0] * random_number
for l in countte:
    sim.nodes[l].selfish=True
counter_var_selfish = 0
for node in sim.nodes:
    if node.selfish and node.id % vr == 0:
        counter_var_selfish += 1
```

Based on the above codes, here are the results of counter_var_selfish in different variant percentages of nodes:

```
rage-p2p_extension.py
0 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
2024-02-05 16:27:08.740 Python[5007:81410] WARNING: Secure
ementing NSApplicationDelegate.applicationSupportsSecureRes
```

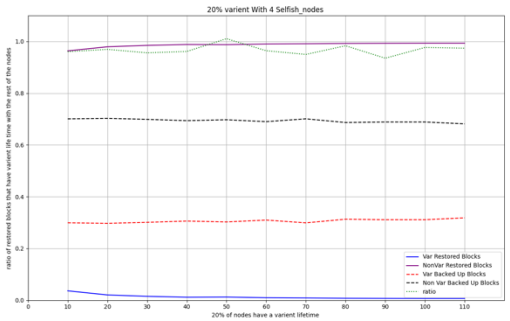
```
rage-p2p_extension.py
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
2024-02-05 16:23:41.880 Python[4889:79214] WARNING: Secure
ementing NSApplicationDelegate.applicationSupportsSecureRe
```

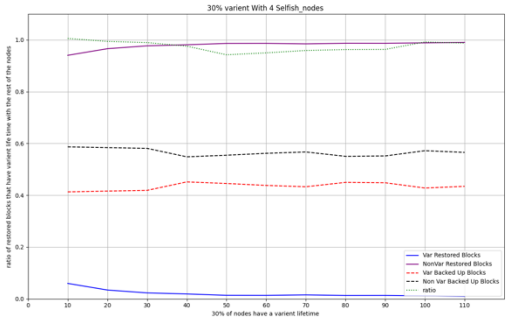
```
rage-p2p_extension.py
2 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
3 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
3 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
2024-02-05 16:17:56.997 Python[4713:75697] WARNING: Secure
ementing NSApplicationDelegate.applicationSupportsSecureRes
```

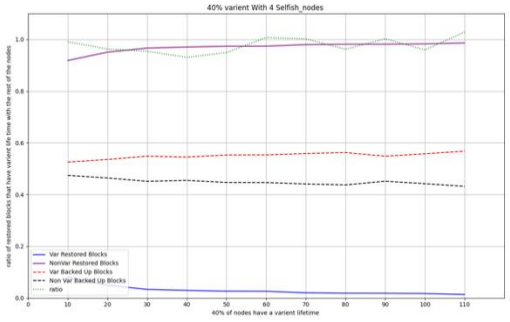
```
Hub/DC/Assignment2/storage-p2p_extension.py
1 selfish nodes were simulated And Have Varient Lifetime.
4 selfish nodes were simulated And Have Varient Lifetime.
3 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
0 selfish nodes were simulated And Have Varient Lifetime.
2 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
1 selfish nodes were simulated And Have Varient Lifetime.
3 selfish nodes were simulated And Have Varient Lifetime.
2024-02-05 16:04:23.680 Python[4381:68553] WARNING: Secure codi
t enabled for restorable state! Enable secure coding by impleme
ApplicationDelegate.applicationSupportsSecureRestorableState: a
ning YES.
```

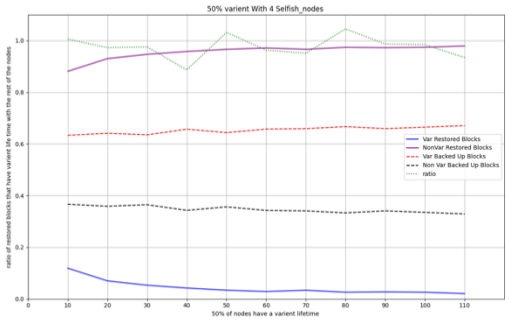
7. Plots of Peer-to-Peer Backup with Extension:

Here are the Plots of Peer-to-Peer Backup with Extension:

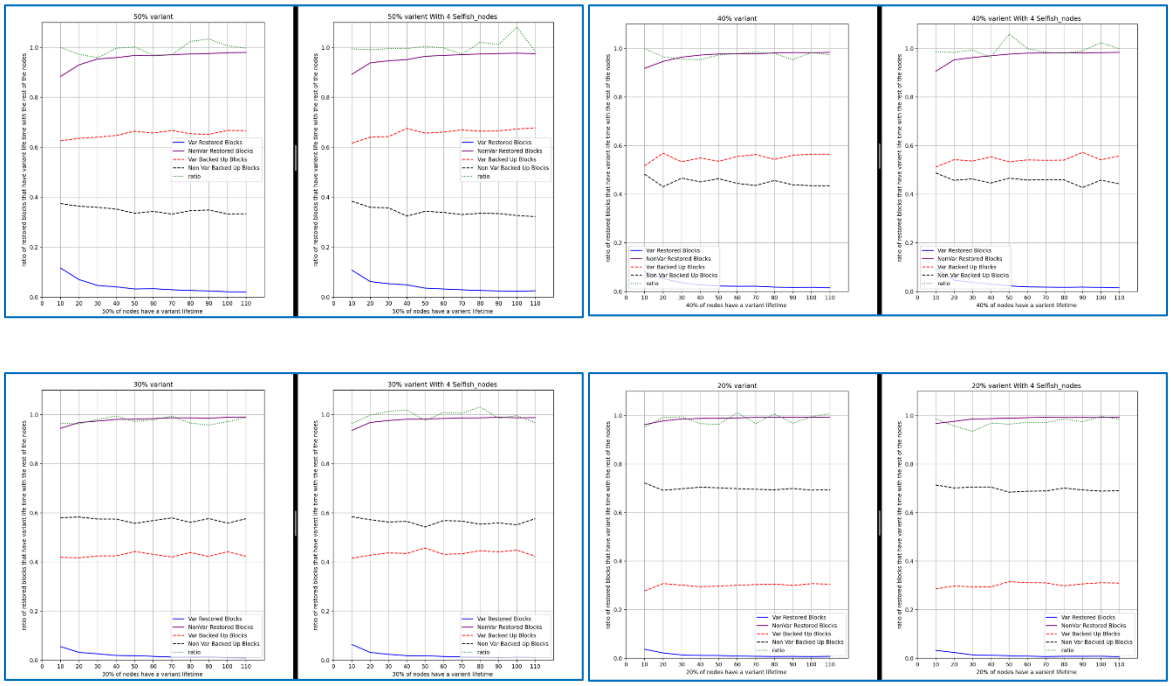






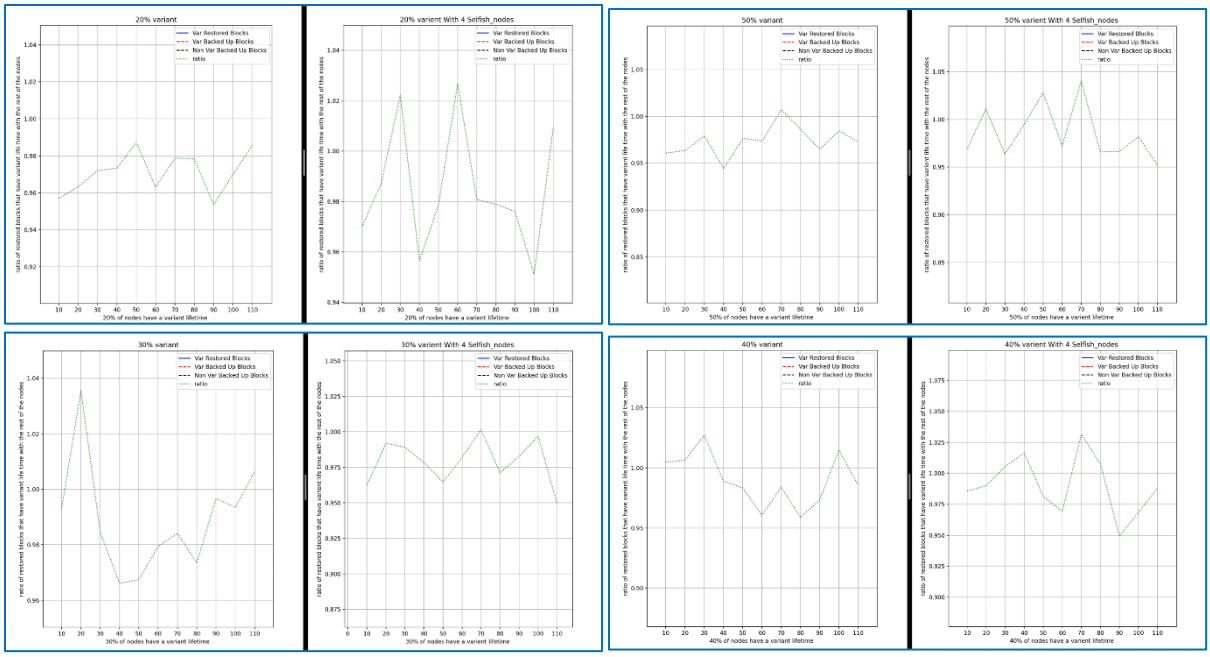


8. Comparing of Plots:



- In the above plots, we bring the same percentage of variants together to show the difference between them. As you can see, there is no bold difference between them, but the slopes of the plots of the restored blocks (variant and non-variant) with the selfish node are a bit higher.

- The notable observation arises when over 40% of nodes exhibit variant lifetimes; in this scenario, we observe an increase in both the number of blocks backed up and the presence of these blocks across nodes with variant lifetimes. This stands in contrast to the patterns observed in plots where the percentage of nodes with variant lifetimes is under 40%.



- In the first 2 plots (20% and 50%) they acted similarly. Without extension, the ratio between restored blocks and backed-up blocks is mostly under 1. In the second 2 plots (30% and 40%) they didn't act predictably.

9. Ambitious, Surprising, and interesting things:

- We thought a lot about finding comparison points to have noticeable plots.
- The interesting point here is that we considered some of the nodes as variants and some others as statics.
- We were somehow disappointed when our plots after the implantation of the extension were different from the plots without extension.
- We put too much effort into making this assignment more interesting, by implementing different types of algorithms.
- We implemented the extension and completed the code for the client-server configuration too.