APPENDIX

JACK CAI

Contents

Code Snippet 1

```python
import matplotlib.pyplot as plt
import numpy as np
import random

class User:
    _window_size = None
    _steps = None

    def __init__(self, n):
        self._window_size = n
        self._steps = [n]

    def Increase(self, alpha):
        self._window_size += alpha

    def Decrease(self, beta):
        self._window_size = max(1, int(self._window_size * beta))

    def Log(self):
        self._steps.append(self._window_size)

    def GetWindowSize(self):
        return self._window_size

    def GetSteps(self):
        return self._steps
```

```

class Host:
    _alpha = 0
    _beta = 0
    _max_window_size = 0
    _clients = []

    def __init__(self, seed, alpha=1, beta=0.5, max_window_size=200):
        random.seed(seed)
        self._alpha = alpha
        self._beta = beta
        self._max_window_size = max_window_size

    def addClients(self, n):
        for i in range(n):
            self._clients.append(User(random.randint(5, 15)))

    def Step(self):
        sum_windows = 0
        for client in self._clients:
            sum_windows += client.GetWindowSize()

            if (sum_windows > self._max_window_size):
                # Congestion event
                for client in self._clients:
                    client.Decrease(self._beta)
                    client.Log()
                return

        for client in self._clients:
                    client.Increase(self._alpha)
                    client.Log()

    def Plot(self, x, y):
        xpoints = np.array(self._clients[x].GetSteps())
        ypoints = np.array(self._clients[y].GetSteps())

        plt.plot(xpoints, ypoints)
        plt.show()

if __name__ == "__main__":
    my_host = Host(42069)
    my_host.addClients(2)

    for i in range(1000):
        my_host.Step()
    my_host.Plot(0, 1)
```

Code Snippet 2

```python
if __name__ == "__main__":
    my_host = Host(42069)
    my_host.addClients(200)
```

Code Snippet 3

```python
if __name__ == "__main__":
    my_host = Host(42069, 1, 0.5, 5000)
```

Code Snippet 4

```python
if __name__ == "__main__":
    my_host = Host(42069, 10, 0.5, 5000)
```

Code Snippet 5

```python
if __name__ == "__main__":
    my_host = Host(42069, 1, 0.1, 5000)
```

Code Snippet 6

```python
if __name__ == "__main__":
    my_host = Host(42069, 1, 0.9, 5000)
```

Code Snippet 7

```python
class Host:
    _alpha = 0
    _beta = 0
    _max_window_size = 0
    _clients = []
    _step_count = 1
```

```python
def Step(self):
        sum_windows = 0
        for client in self._clients:
            sum_windows += client.GetWindowSize()

            if (sum_windows > self._max_window_size):
                # Congestion event
                for client in self._clients:
                    client.Decrease(self._beta)
                    client.Log()
                self._alpha = max(1, int(self._alpha * self._beta))
                return

        for client in self._clients:
                    client.Increase(self._alpha)
                    client.Log()
        self._alpha += self._step_count
        self._step_count += 1
```

Code Snippet 8

```
class Host:
    _alpha = 0
    _beta = 0
    _max_window_size = 0
    _clients = []

```

```
def Step(self):
        sum_windows = 0
        for client in self._clients:
            sum_windows += client.GetWindowSize()

            if (sum_windows > self._max_window_size):
                # Congestion event
                for client in self._clients:
                    client.Decrease(self._beta)
                    client.Log()
                self._alpha = max(1, int(self._alpha * self._beta))
                return

        for client in self._clients:
                    client.Increase(self._alpha)
                    client.Log()
        self._alpha *= 2

```

Code Snippet 9

```
class User:
    _alpha = 0
    _beta = 0
    _window_size = None
    _steps = None

    def __init__(self, n, alpha, beta):
        self._alpha = alpha
        while (0 == beta):
            beta = random.random()
        self._beta = beta
        self._window_size = n
        self._steps = [n]

    def Increase(self):
        self._window_size += self._alpha

```

```python
    def Decrease(self):
        self._window_size = max(1, int(self._window_size * self._beta))
```

```python
class Host:
    _max_window_size = 0
    _clients = []

    def __init__(self, seed, max_window_size=20):
        random.seed(seed)
        self._max_window_size = max_window_size

    def addClients(self, n):
        for i in range(n):
            self._clients.append(User(random.randint(1, 10),
            ↪   random.randint(1, 10), random.random()))

    def Step(self):
        sum_windows = 0
        for client in self._clients:
            sum_windows += client.GetWindowSize()

            if (sum_windows > self._max_window_size):
                # Congestion event
                for client in self._clients:
                    client.Decrease()
                    client.Log()
                return

        for client in self._clients:
                    client.Increase()
                    client.Log()
```

Cheriton School of Computer Science, University of Waterloo
Email address: jack.cai@uwaterloo.ca