Develop Average Calculator HTTP Microservice

```
from flask import Flask, jsonify, request
import requests
from collections import deque
import time

app = Flask(__name__)

# Configuration
WINDOW_SIZE = 10
TIMEOUT = 0.5  # 500 milliseconds

# Storage for numbers
numbers_window = deque(maxlen=WINDOW_SIZE)
unique_numbers = set()

# Helper function to fetch numbers from third-party server
def fetch_number(number_type):
    url = f"https://third-party-server.com/api/{number_type}"
    try:
        response = requests.get(url, timeout=TIMEOUT)
        if response.status_code == 200:
            return response.json().get('number')
    except (requests.RequestException, ValueError):
        return None

# Helper function to calculate average
def calculate_average(numbers):
    if numbers:
        return sum(numbers) / len(numbers)
    return 0

@app.route('/numbers/<string:numberid>', methods=['GET'])
def get_numbers(numberid):
    if numberid not in {'p', 'f', 'e', 'r'}:
        return jsonify({"error": "Invalid number ID"}), 400

    # Fetch number from third-party server
    new_number = fetch_number(numberid)
    if new_number is None:
        return jsonify({"error": "Failed to fetch number"}), 500

    # Prepare response data
    window_prev_state = list(numbers_window)

    # Update numbers window and unique numbers set
    if new_number not in unique_numbers:
        if len(numbers_window) == WINDOW_SIZE:
            oldest_number = numbers_window.popleft()
            unique_numbers.remove(oldest_number)
        numbers_window.append(new_number)
        unique_numbers.add(new_number)

    window_curr_state = list(numbers_window)
    avg = calculate_average(window_curr_state)

    response = {
        "windowPrevState": window_prev_state,
        "windowCurrState": window_curr_state,
        "numbers": [new_number],
        "avg": avg
    }

    return jsonify(response), 200

if __name__ == '__main__':
    app.run(debug=True)
```