prac4-ds

April 24, 2025

[1]:

```python
import random

# Simulated system clocks (in seconds), where 0 is the central server
clocks = {
    "Server": 10000,
    "Client_1": 10020,
    "Client_2": 9980,
    "Client_3": 10010,
    "Client_4": 10050
}

# Simulated round trip delays in seconds for each client (excluding server)
delays = {
    "Client_1": random.randint(1, 5),
    "Client_2": random.randint(1, 5),
    "Client_3": random.randint(1, 5),
    "Client_4": random.randint(1, 5)
}

def berkeley_algorithm(server_clock, clocks, delays):
    print("Initial Clocks:")
    for node, time in clocks.items():
        print(f"{node}: {time} sec")

    # Step 1: Server polls all clients
    time_differences = {}
    for client, client_time in clocks.items():
        if client != "Server":
            # Calculate estimated time difference considering delay
            delay = delays[client] / 2  # one-way delay
            estimated_time = client_time + delay
            time_diff = estimated_time - server_clock
            time_differences[client] = time_diff

    # Step 2: Server calculates average time difference
    avg_diff = sum(time_differences.values()) / len(time_differences)
    # Step 3: Server updates its own clock
    clocks["Server"] += avg_diff

    # Step 4: Server sends adjustment to each client
    for client, diff in time_differences.items():
        adjustment = avg_diff - diff
        clocks[client] += adjustment

    print("\nAdjusted Clocks:")
    for node, time in clocks.items():
        print(f"{node}: {round(time, 2)} sec")

# Run the algorithm
berkeley_algorithm(clocks["Server"], clocks, delays)
```

```
Initial Clocks:
Server: 10000 sec
Client_1: 10020 sec
Client_2: 9980 sec
Client_3: 10010 sec
Client_4: 10050 sec

Adjusted Clocks:
Server: 10016.88 sec
Client_1: 10014.38 sec
Client_2: 10015.88 sec
Client_3: 10015.38 sec
Client_4: 10014.38 sec
```

[ ]:

```

```