Tugas 1

Alamat Repository: https://github.com/Alif-N/ds25/tree/main/synchronization/time_sync

Mochammad Alif Nasrullah/5025231314

Pendahuluan: Tugas 1 ini adalah melakukan Eksplorasi dan Analisis Implementasi Time Synchronization dari repository yang sudah disediakan: https://github.com/rm77/ds25/tree/main/synchronization/time_sync

Telah di sediakan 4 node (node A, node B, node C, dan node D). Node C bertugas sebagai logger untuk mencatat semua kejadian antara node node lain saat berkomunikasi

Node A (192.168.122.193:5000): Node yang dapat berkomunikasi dengan node lain (B, D).
Node B (192.168.122.147:5001):Node yang dapat berkomunikasi dengan node lain (A, D).
Node C (192.168.122.164:9999): Node yang memiliki tugas menjadi logger (mencatat seluruh aktivitas node lain).
Node D (192.168.122.94:5002):Node yang dapat berkomunikasi dengan node lain (A, B).

```
[D] Listening on ('0.0.0.0', 5002)
[D] Peers: A@192.168.122.193:5000, B@192.168.122.147:5001
[D] Collector: ('192.168.122.164', 9999)
[D] Clock offset=-0.600s, proc_delay=10ms

Type messages:
  @NAME your message      -> send to one peer
  /broadcast your message -> send to all peers
  /peers  /help  /exit
```

Beberapa informasi yang ditampilkan dan perintah yang dapat dilakukan oleh node untuk melakukan komunikasi dengan node lain

- @(nama node lain:A/B/D) pesan_yang_ingin_dikirim
- /broadcast pesan_yang_ingin_dikirim
- /peers untuk melihat peers (node) mana saja yang saling terhubung
- /help untuk melihat bantuan lebih detail
- /exit untuk keluar dari program

```
Type messages:
  @NAME your message      -> send to one peer
  /broadcast your message -> send to all peers
  /peers  /help  /exit

[A] SENT chat -> B: id=CHAT-A-10022406 text='Hello from A'
[A] SENT chat -> D: id=CHAT-A-10022457 text='Hello from A'
[A] RECV ack from D id=ACK-D-10022467 L_in=4->5 V_in={'D': 2, 'A': 2, 'B': 0}->{'A': 3, 'B': 0, 'D': 2}
/broadcast halo
[A] SENT chat -> B: id=CHAT-A-10046077 text='halo'
[A] SENT chat -> D: id=CHAT-A-10046077 text='halo'
[A] RECV ack from D id=ACK-D-10046089 L_in=9->10 V_in={'D': 4, 'A': 5, 'B': 0}->{'A': 6, 'B': 0, 'D': 4}
@B halo
[A] SENT chat -> B: id=CHAT-A-10058144 text='halo'
[A] RECV chat from B id=CHAT-B-10068782 L_in=1->12 V_in={'B': 1, 'A': 0, 'D': 0}->{'A': 8, 'B': 1, 'D': 4
[A] SENT ack  -> B: id=ACK-A-10068794 for=CHAT-B-10068782
```

Gambar diatas merupkan gambar ketika node telah mengirimkan atau menerima suatu

pesan, bisa kita lihat ketika node mengirimkan pesan maka akan muncul “SENT chat ->” dengan id chat lalu diikuti dengan pesan yang dikirimkan
Lalu ketika Node A menerima acknowledgment (ACK) dari node D. Nilai **Lamport Clock** (L_in) dari node D berubah dari 4 menjadi 5. Nilai **Vector Clock** (V_in) menunjukkan bahwa node D memiliki timestamp 2, node A memiliki 2, dan node B memiliki 0 sebelum penerimaan, sementara setelah penerimaan, node A memiliki 3, dan node D tetap 2

```
root@ds-pc-3:/home/ds25/synchronization/time_sync/node_c# bash run.bash
[COLLECTOR] Listening on ('0.0.0.0', 9999)
```

Gambar diatas merupakan hasil dari menjalankan bash run.bash pada folder node_c dimana node ini bertugas sebagai logger untuk mencatat riwayat komunikasi yang terjadi antar node. Terlihat pada gambar bahwa logger mendengar pada semua alamat pada port 9999

```
root@ds-pc-3:/home/ds25/synchronization/time_sync/node_c# bash run.bash
[COLLECTOR] Listening on ('0.0.0.0', 9998)
[COLLECTOR] CHAT {'type': 'chat', 'kind': 'chat', 'id': 'CHAT-A-10022406', 'sender': 'A', 'receiver': 'B
mono_send': 10022.406228765, '_collector_arrival': 1764698364.138704}
[COLLECTOR] CHAT {'type': 'chat', 'kind': 'chat', 'id': 'CHAT-A-10022457', 'sender': 'A', 'receiver': 'D
'mono_send': 10022.457186356, '_collector_arrival': 1764698364.1892006}
[COLLECTOR] CHAT {'type': 'chat', 'kind': 'ack', 'id': 'ACK-D-10022467', 'sender': 'D', 'receiver': 'A',
152, 'mono_send': 10022.467967889, '_collector_arrival': 1764698364.1998897}
[COLLECTOR] CHAT {'type': 'chat', 'kind': 'chat', 'id': 'CHAT-A-10046077', 'sender': 'A', 'receiver': 'B
nd': 10046.077769869, '_collector_arrival': 1764698387.8097045}
[COLLECTOR] CHAT {'type': 'chat', 'kind': 'chat', 'id': 'CHAT-A-10046077', 'sender': 'A', 'receiver': 'D
nd': 10046.077840798, '_collector_arrival': 1764698387.8098695}
```

Pesan Log yang dicatat pada logger bisa kita lihat seperti gambar diatas dan memuat informasi seperti berikut:

- Jenis pesan: chat
- kind: Bisa berupa "chat" atau "ack" (acknowledgment).
- id: ID pesan unik yang dikirimkan.
- sender: Pengirim pesan.
- receiver: Penerima pesan.
- mono_send: Waktu pengiriman pesan dalam format monotonic time.
- _collector_arrival: Waktu kedatangan pesan di pengumpul (collector)

Protokol Transport

Program ini menggunakan protokol UDP (User Datagram Protocol). UDP memungkinkan pengiriman pesan tanpa jaminan pengurutan atau keandalan yang tepat, karena UDP tidak melakukan kontrol aliran, pemeriksaan kesalahan, atau pengendalian ulang jika data hilang. Cocok untuk aplikasi chat interaktif ini, dimana kecepatan dan kesederhanaan lebih penting daripada keandalan penuh

Protokol Message Exchange

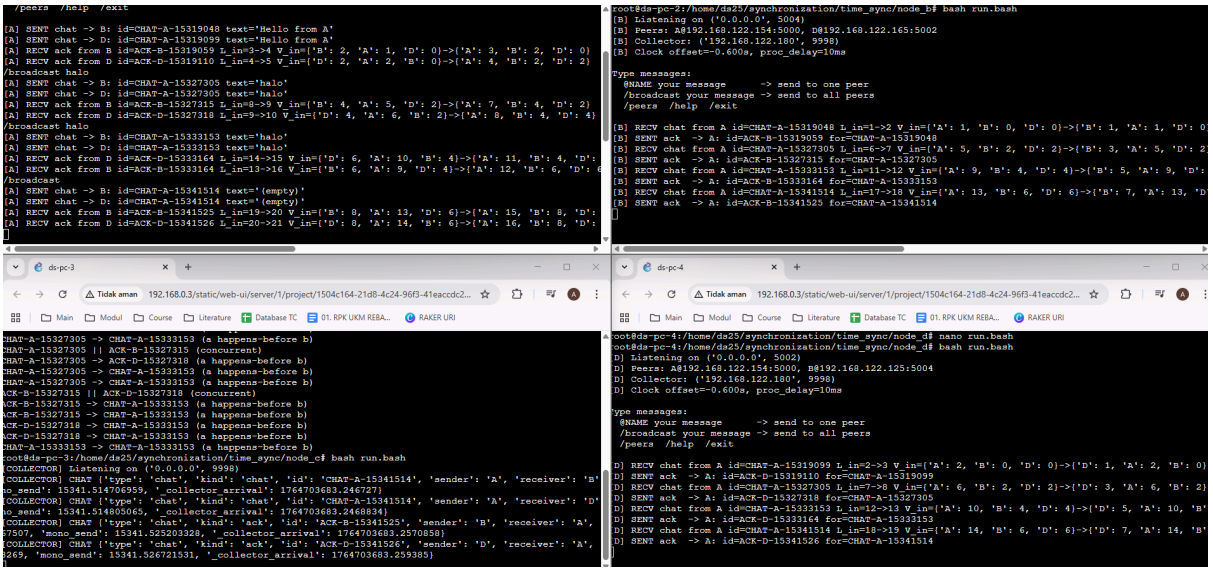
Program menggunakan JSON sebagai format pesan untuk pertukaran data antar peer. Setiap pesan berisi informasi seperti tipe pesan, ID, pengirim, penerima, teks pesan, dan nilai waktu (Lamport dan Vector Clocks). Pesan ini dikirim melalui socket UDP dengan fungsi ‘_send_to’ dan ‘_make_payload’. Pesan pada program ini juga di kategorikan dengan 2 jenis: “chat” untuk pesan obrolan dan “ack” untuk acknowledge atau pesan konfirmasi

Protokol Time Synchronization

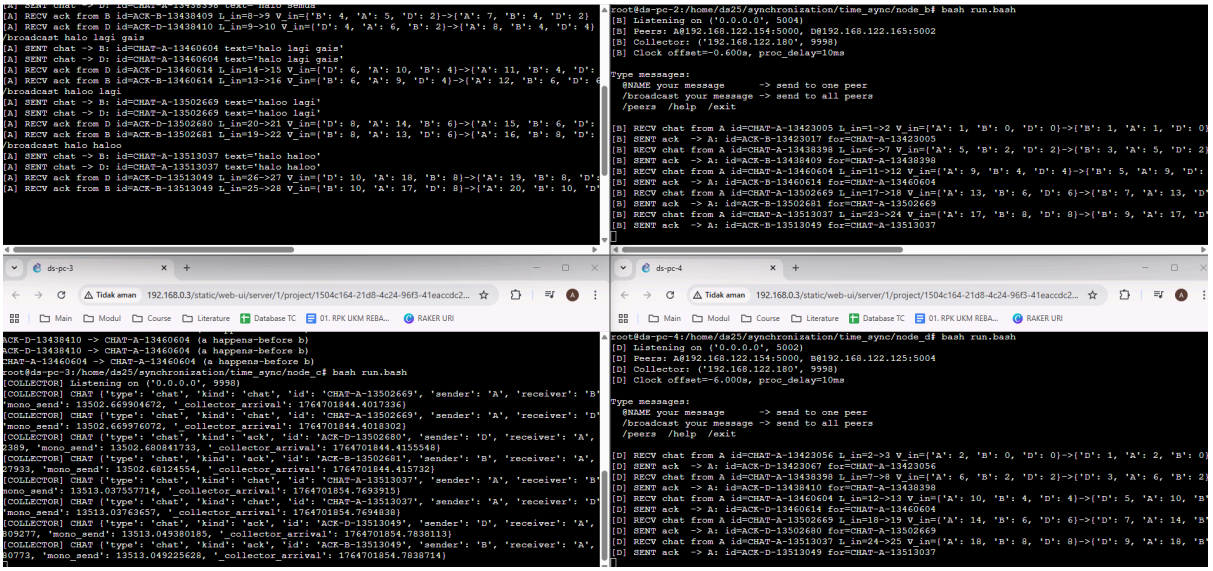
- **Lamport Clocks**: Digunakan untuk menjaga urutan logis dari pesan yang dikirim dan diterima. Setiap pengiriman pesan meningkatkan nilai Lamport clock (L), dan penerimaan pesan akan menyesuaikan Lamport clock untuk memastikan urutan yang benar.
- **Vector Clocks**: Digunakan untuk merekam waktu secara lebih terperinci, dengan menyimpan clock untuk setiap peer dalam sistem. Setiap pengiriman dan penerimaan pesan memperbarui nilai-nilai ini untuk memastikan sinkronisasi waktu yang lebih akurat antar peer

Simulasi jika terjadi time-drift

Clock_offset sama



Clock_offset berbeda



saya merubah setting clock_offset pada node D menjadi 6000 untuk mensimulasikan time-drift

Dengan mengatur clock_offset yang berbeda kita dapat mensimulasikan kondisi di mana node memiliki ketidaksesuaian waktu yang dapat terjadi dalam sistem nyata (misalnya, jika node berada di zona waktu berbeda atau memiliki masalah sinkronisasi jam). Dalam

simulasi ini, clock_offset mempengaruhi: Urutan pesan, Pengolahan pesan, Penggunaan Lamport Clocks dan Vector Clocks untuk mempertahankan konsistensi logis dalam urutan pesan

Bisa dilihat perbedaan lamport clock dan vector clock dari dua gambar tersebut, ketika clock_offset dari tiap node sama maka urutan pesan urut secara konsisten dan ketika clock_offset tidak sama pada node node tersebut maka terjadi inkonsistensi urutan yang kita bisa lihat di nilai dari lamport clock dan vector clock

Kesimpulan

Dalam tugas ini kita telah mencoba simulasi dari pengiriman pesan antar node dengan menggunakan protokol UDP juga dengan protokol time synchronization untuk mengetahui konsistensi dan urutan waktu pesan yang masuk. dalam simulasi ini juga kita mencoba case time-drift dan dari case tersebut kita dapat menganalisis betapa penting nya sinkronisasi waktu pada sistem yang terdistribusi untuk menjaga urutan (konsistensi) pesan tersampaikan dengan baik