Tugas 2

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

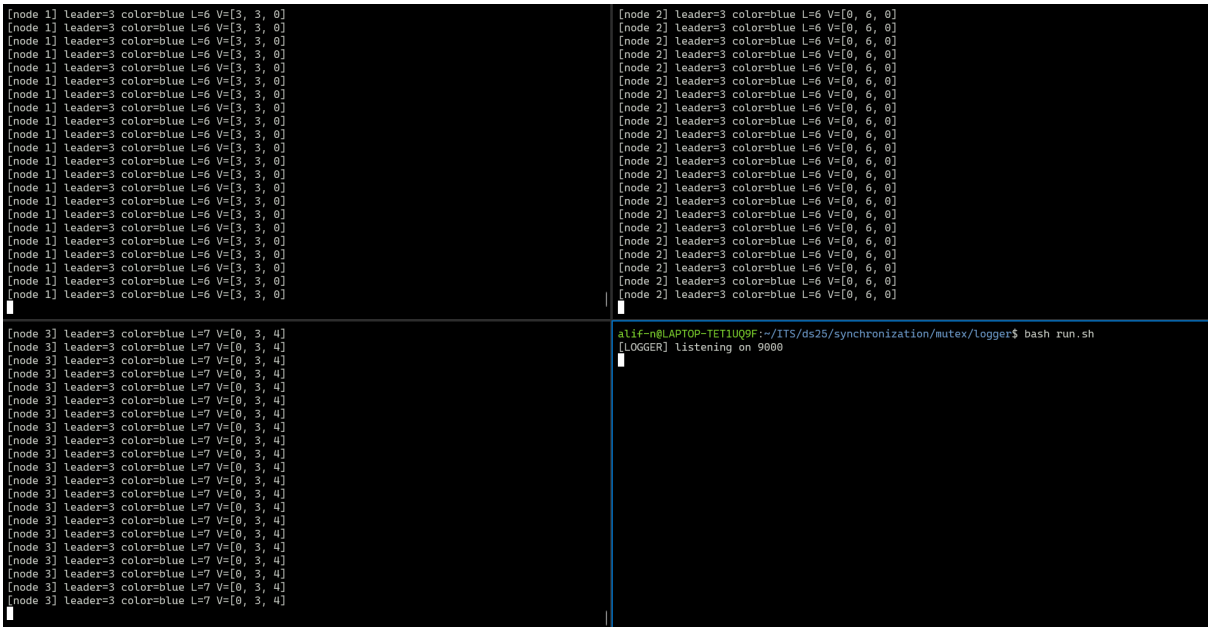
Mochammad Alif Nasrullah/5025231314

Pendahuluan: Tugas 2 ini adalah melakukan Eksplorasi dan Analisis Implementasi Mutual Exclusion dari repository yang sudah disediakan: https://github.com/rm77/ds25/tree/main/synchronization/mutex

Telah di sediakan 4 node (node 1, node 2, node 3, dan logger). logger bertugas sebagai mencatat log semua kejadian antara node node lain saat berkomunikasi dan 1 node lagi untuk melakukan testing

sebelumnya mohon maaf karena saya menjalankan eksplorasi ini di local dikarenakan UI terminal di GNS3 tidak responsif secara ukuran, banyak tulisan dan command yang tertumpuk tumpuk sehingga menyusahkan untuk di capture dan susah untuk melakukana analisis berikutnya

karena dijalankan pada local maka untuk IP semua node adalah localhost dengan port yang berbeda beda yakni: 8001,8002,8003, dan 9000



Dapat dilihat pada gambar tersebut tiap node yang telah dijalankan, dalam node1 hingga node 3 akan selalu menampilkan beberapa informasi yakni “[node 1] leader=3 color=<nil> L=0 V=[0, 0, 0]” nama node, leader dari seluruh node, value dari variabel color, nilai L (Logical Timestamp) dan nilai V (vector)

Untuk pemilihan leader pada program ini hanya dengan membandingkan nilai dari node nya (Node1=1, Node2=2, Node3=3) jadi jika semua node dijalankan maka yang menjadi leader adalah node3 dengan nilai tertingga yaitu 3 dan jika node3 mati maka akan mencari nilai terbesar berikutnya

Protokol Transport

Program ini menggunakan **TCP** dengan berbagai port yang sudah disebutkan di awal, TCP digunakan untuk komunikasi yang andal antar node. TCP lebih menjamin pengiriman data yang urut dan minim kehilangan data, cocok untuk perintah yang membutuhkan konsistensi dan keandalan.

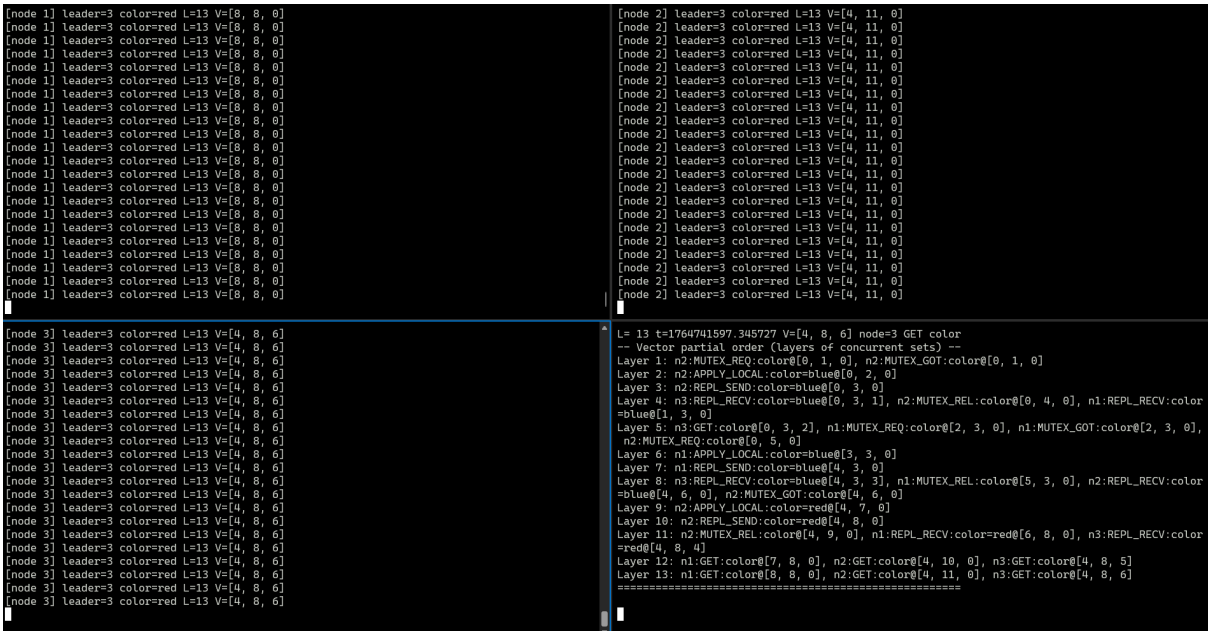
Program ini juga menggunakan **UDP** untuk komunikasi yang lebih cepat, namun tidak menjamin pengiriman atau urutan data. Cocok untuk pengiriman pesan atau notifikasi yang tidak memerlukan konfirmasi penerimaan

Protokol Message Exchange

- Dengan perintah CMD: node mengirimkan perintah seperti PUT / GET untuk menyimpan atau mengambil data. Pesan ini dikirim melalui TCP atau UDP tergantung dengan kebutuhan
- MUTEX: menggunakan mutex untuk menghindari case race condition, memastikan hanya satu node yang dapat memperbarui data pada satu waktu
- Replikasi dan Sinkronisasi: setelah data diperbarui pada satu node, perubahan tersebut direplikasi ke node lain untuk menjaga konsistensi data antar node

Hasil Test 1 dengan Mutex 1

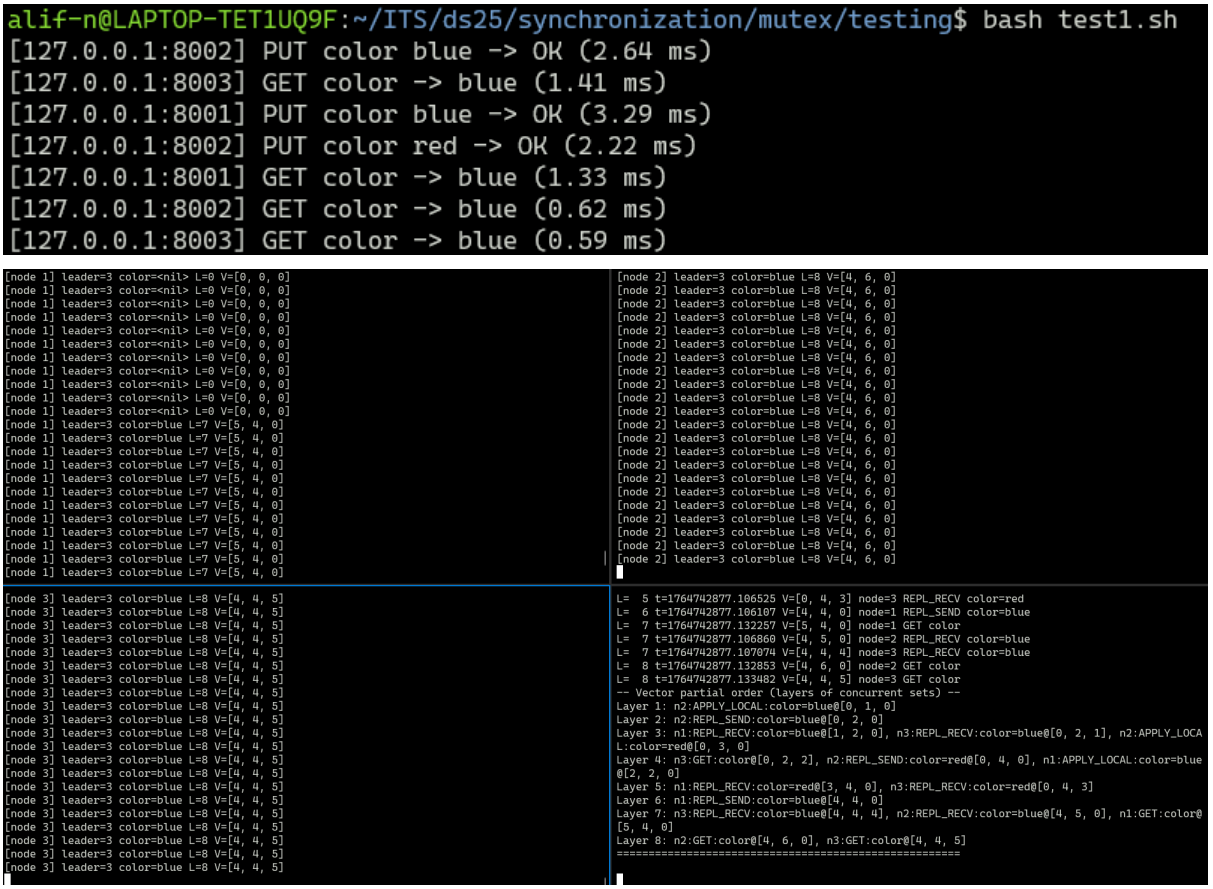
```
alif-n@LAPTOP-TET1UQ9F:~/ITS/ds25/synchronization/mutex/testing$ bash test1.sh
[127.0.0.1:8002] PUT color blue -> OK (6.93 ms)
[127.0.0.1:8003] GET color -> blue (1.85 ms)
[127.0.0.1:8001] PUT color blue -> OK (3.29 ms)
[127.0.0.1:8002] PUT color red -> OK (5.37 ms)
[127.0.0.1:8001] GET color -> red (1.50 ms)
[127.0.0.1:8002] GET color -> red (0.52 ms)
[127.0.0.1:8003] GET color -> red (0.66 ms)
```



terdapat beberapa perintah pada test 1 yakni “PUT color blue” pada node1 dan berhasil mengubah pada nilai variabel color pada node1 menjadi blue lalu melakukan replikasi nilai color sehingga semua node memiliki nilai color blue lalu dibuktikan dengan perintah “GET color ” pada node2 yang mengembalikan nilai blue. selanjutnya kita mencoba dengan menjalankan perintah berbarengan yakni ` race "PUT color blue" "PUT color red" `

dikarenakan kita menggunakan mutex=1 maka hanya 1 node yang boleh mengubah data pada satu waktu, dan replikasi ke node lain akan terjadi setelah pembaruan selesai di node pertama, didalam node bisa dilihat bahwa semua node sempat berubah nilai menjadi blue lalu berubah nilai nya menjadi red sesuai dengan perintah ganda di awal yaitu ubah nilai menjadi biru dan ubah nilai menjadi merah dan program berhasil menjalankan dengan berurutan dan konsisten di tiap node nya

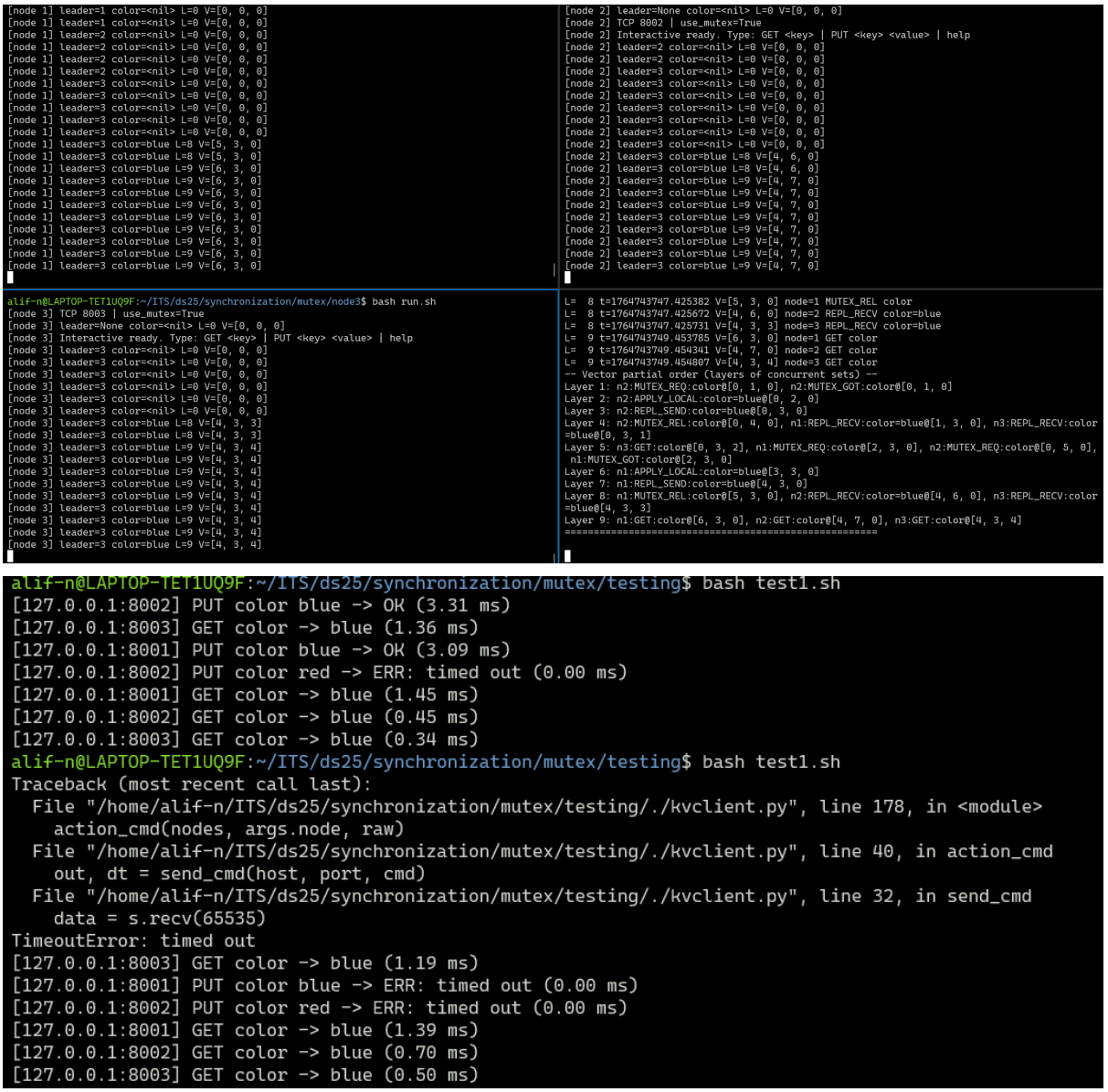
Hasil Test 1 dengan Mutex 0



Dengan perintah yang sama dengan percobaan pertama tapi kali ini kita set untuk nilai mutex dari program adalah 0, secara teori jika kita set nilai mutex 0 maka kita akan mematikan penguncian saat pengisian data, memungkinkan pembaruan data oleh beberapa node secara bersamaan dan mengakibatkan inkonsistensi data.

Setelah saya melihat apa yang terjadi pada tiap node, pada perintah yang berbarengan yakni “PUT color blue” dan “PUT color red”, program mengembalikan nilai OK pada kedua perintah tersebut tetapi ketika saya melihat di tiap tiap node tidak pernah ada menyimpan nilai blue dan langsung bernilai red sehingga hasil analisis saya untuk perintah pertama “PUT color blue” tidak tersimpan karena terjadi **race condition** dengan perintah selanjutnya dan dianggap selesai oleh sistem dengan mengembalikan nilai OK padahal tidak ada node yang menyimpan data tersebut

Hasil Test 1 dengan Mutex 0



Kali ini saya mengubah nilai mutex=5 dan saya ingin tahu bagaimana hasil dari program nya, sesuai dengan gambar diatas saya menjalankan kembali dengan test/perintah yang sama seperti sebelumnya, bisa dilihat dari gambar bahwa terjadi error saat perintah bersamaan yakni “PUT color blue” dan “PUT color red” sistem hanya bisa menjalankan perintah yang di tulis pertama yaitu put color blue dan tiap node berhasil menyimpan nilai color tersebut lalu error untuk perintah selanjutnya. saya melakukan test beberapa kali dan tiap test tetap terjadi error yang sama yakni time out. Menurut saya hal ini mungkin karena mutex yang semakin tinggi berarti penguncian lebih ketat/komplek padahal sistem kita tidak membutuhkan hal tersebut sehingga menurunkan kinerja

Kesimpulan

- **--use-mutex 1**: Mengaktifkan penguncian untuk memastikan hanya satu node yang mengubah data pada satu waktu, mencegah race condition namun dapat mengurangi kinerja.

- **--use-mutex 0**: Menonaktifkan penguncian, memungkinkan pembaruan data oleh beberapa node secara bersamaan, meningkatkan kinerja tetapi berisiko menyebabkan inkonsistensi data.

- **--use-mutex 5**: Mengatur penguncian yang lebih ketat atau lebih kompleks, berpotensi menurunkan kinerja lebih jauh dengan pengelolaan penguncian yang lebih rumit, tetapi memberikan kontrol lebih besar dalam hal konsistensi data dan konflik

Kita harus mengetahui kebutuhan sistem kita sebelum menentukan nilai dari mutex sehingga kita bisa lebih bijak dan berhati hati sesuai dengan kebutuhan sistem