CSDS 600 Distributed Algorithm

Analysis of Singhal - Kshemkalyani’s Vector Clock

Presented by Shanmuga Ganesh Thiruppathy

Spring 2024

Algorithmic Background

- Random Flooding [Distributed (Message Propagation) Algorithm]
  - *Decentralized* Communication & *Broadcasting* Messages
  - Ensuring Coverage & Scalability
  - Trade-off between Efficiency and Overhead

- Singhal-KShemKalyani’s [Vector Clock]
  - Form of reasoning about *causality* and maintaining *consistency* of events
  - Event Ordering (partial ordering of events)
  - Detecting Concurrent Events (2 events neither precedes nor succeeds the other)

Objective

To analyze Singhal-KShemKalyani's vector clock algorithm,

❏ By change the number of processes and assess its performance

❏ We'll consider the *maximum* and *average* messages constructed for the **send event** and the updates made in the **receive event**. This will help us understand its behavior in various scalability conditions

Choosing Metrics for Performance Evaluation

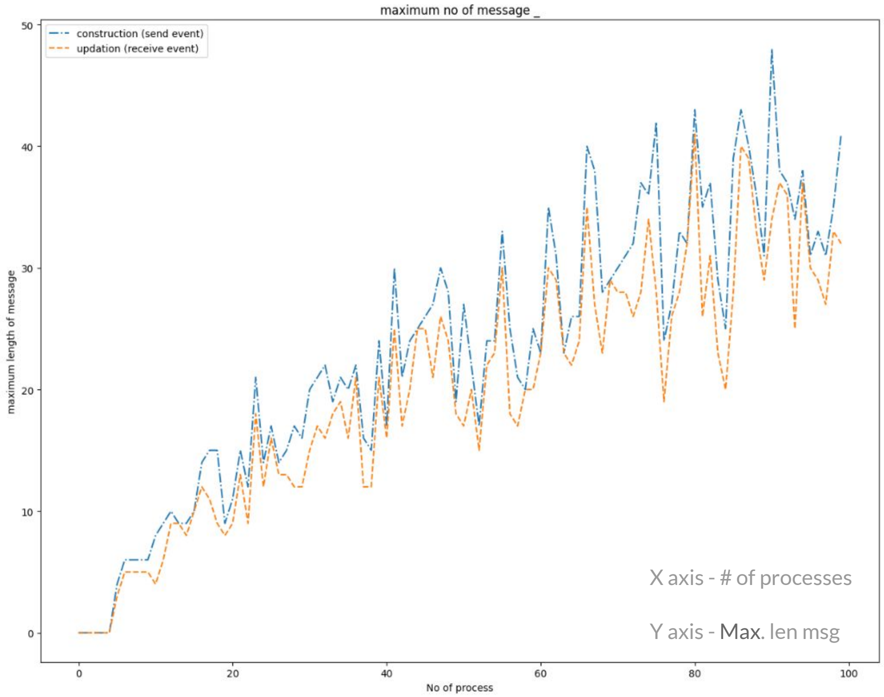
- Message *Construction*
  - When a process **sends** a message to other process
  - For example, $P_3 \rightarrow P_1$, message = $\{ (pid_i , ts_i) \}$, $\forall$ i $\in$ *Processes* & $LS_3 < LU_1$

- Message *Updation*
  - When a process **receives** a message from other process
  - For example, $P_1 \leftarrow P_3$, message contains $\{ (pid_i , ts_i) \}$, Update iff $ts_i > TS_i$ & i $\in$ elements *pid* in message

ts_i → timestamp for process i in *message*

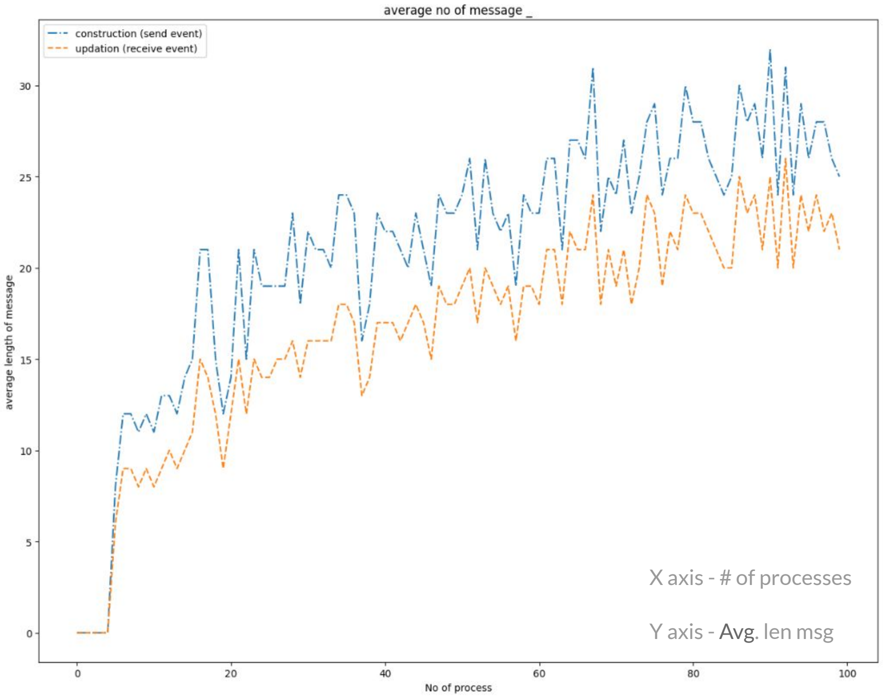
TS_i → Timestamp of the process i in *Matrix*

Graphical Analysis



X axis - # of processes

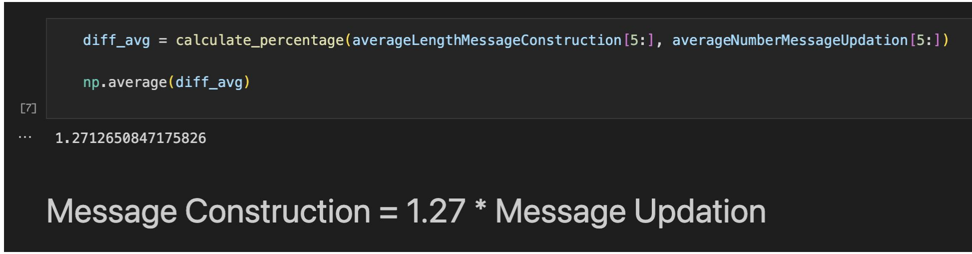
Y axis - Max. len msg



X axis - # of processes

Y axis - Avg. len msg

Understanding the Result



```
diff_avg = calculate_percentage(averageLengthMessageConstruction[5:], averageNumberMessageUpdation[5:])

np.average(diff_avg)
```

[7]

```
... 1.2712650847175826
```

Message Construction = 1.27 * Message Updation

- ❑ Message *Construction*
  - ❑ it includes its vector clock (current local timestamp always) in the message
  - ❑ The sender's vector clock includes timestamps for **all processes**
    - ❑ Which might have updated previously by other send events, which this sender *didn't knew*.

- ❑ Message *Updation*
  - ❑ Update operation involves taking only the greater (>) value of each entry received

→ As a result, only some of the Messages received might be updated, due to a **higher** timestamp value

Future Work

❏ I will explore *non-clique* topologies like the **DFS spanning tree**

❏ I will simulate the *realistic algorithmic condition* by utilizing **multi-threading** for the concurrent nature for the distributed algorithm

Thank you

Q & A session