ΙΔΡΥΜΑ ΕΡΕΥΝΑΣ ΚΑΙ ΚΑΙΝΟΤΟΜΙΑΣ
Ευρωπαϊκή Ένωση
Ευρωπαϊκό Ταμείο Περιφερειακής Ανάπτυξης
Διαρθρωτικά Ταμεία της Ευρωπαϊκής Ένωσης στην Κύπρο
Κυπριακή Δημοκρατία

EXPLORING THE USE OF STRONGLY CONSISTENT DISTRIBUTED SHARED MEMORY IN 3D NETWROKED VIRTUAL ENVIRONMENTS

algolysis
algorithmic solutions

EratoVR

CONCEPT/0618/0064

April, 2021

VIRTUAL ENVIRONMENTS

- ✓***Virtual*** *and* ***Augmented*** *Reality:*
  - ✓*One of the key driving technologies of the 4th Industrial Revolution*
  - ✓*Radically disrupt almost every business sector*
  - ✓*Transform the way we live and interact with our environment*
- ✓Virtual Environments (VEs) are considered among the most elaborate computer-based simulations possible to date

Virtual Environments

- However, algorithms making possible the NVEs of today are:
  - reaching their limits,
  - proving unreliable,
  - suffer asynchronies; and
  - deployed over an inherently fault-prone network infrastructure.

- New **scalable**, **robust**, and **responsive** strategies that can support the needs of the NVEs of tomorrow are necessary.
  - Surgeries –> precise timing, sync guarantees, fault tolerance
  - Multiuser Games -> small delays are utmost importance
  - Virtual Classes -> scalability and concurrency

VIRTUAL ENVIRONMENT ARCHITECTURES

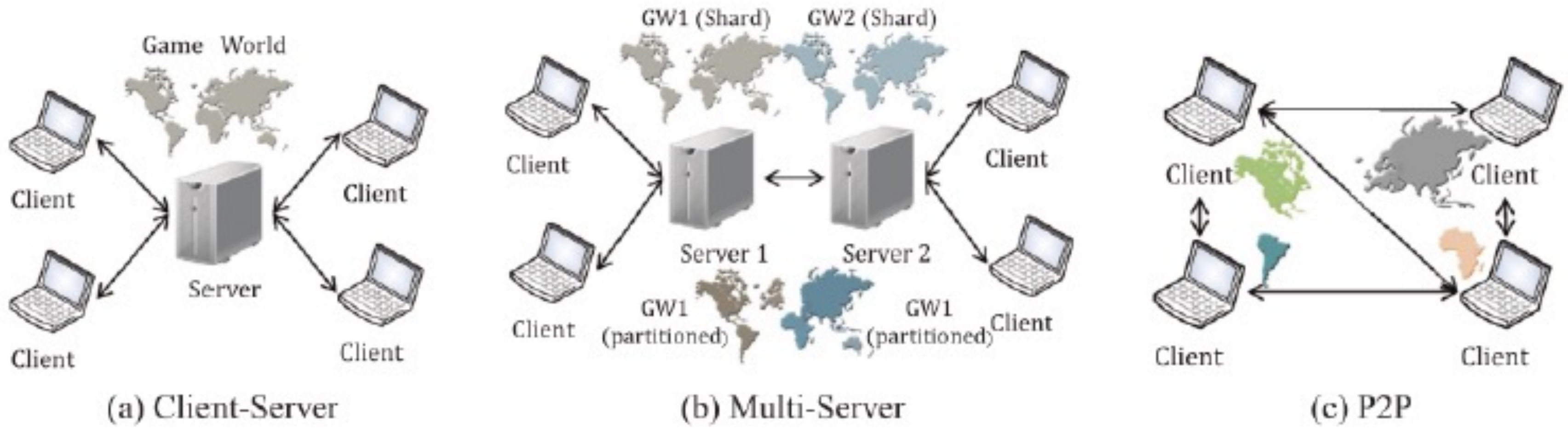


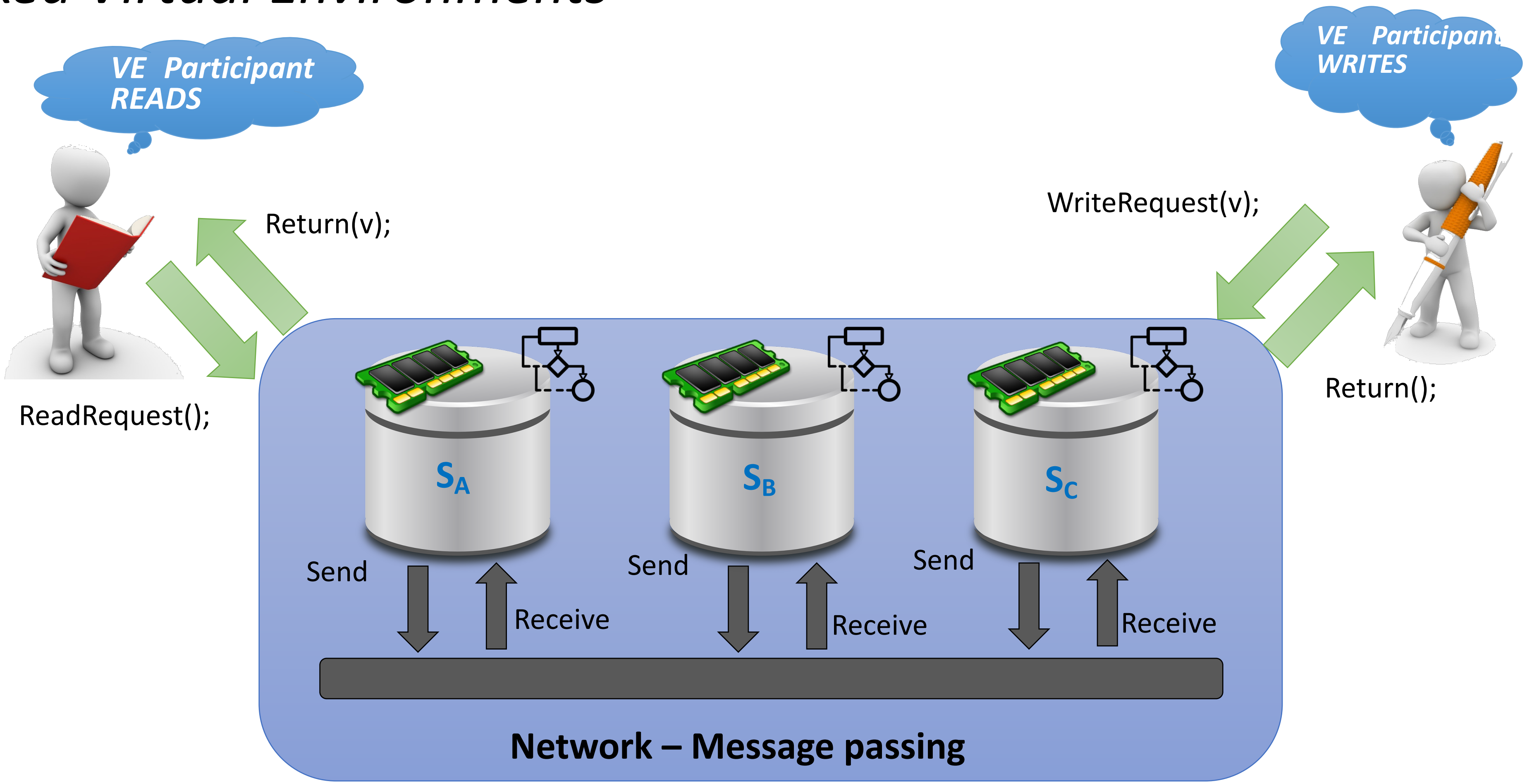
Figure 1 - Different architectures of Virtual Environments (image from Yahyavi and Kemme [21])

Architecture	Pros	Cons
Client-Server	+ Simplicity + Easy management + Consistency control	−− Scalability −− Fault tolerance −− Cost
Multi-Server	+ Scalability + Fault tolerance	− Isolation of players − Complexity −− Cost
Peer-to-Peer	++ Scalability ++ Cost + Fault tolerance	− Harder to develop − Consistency control − Cheating

Figure 2 - Comparison of different architectures (from Yahyavi and Kemme [21]).

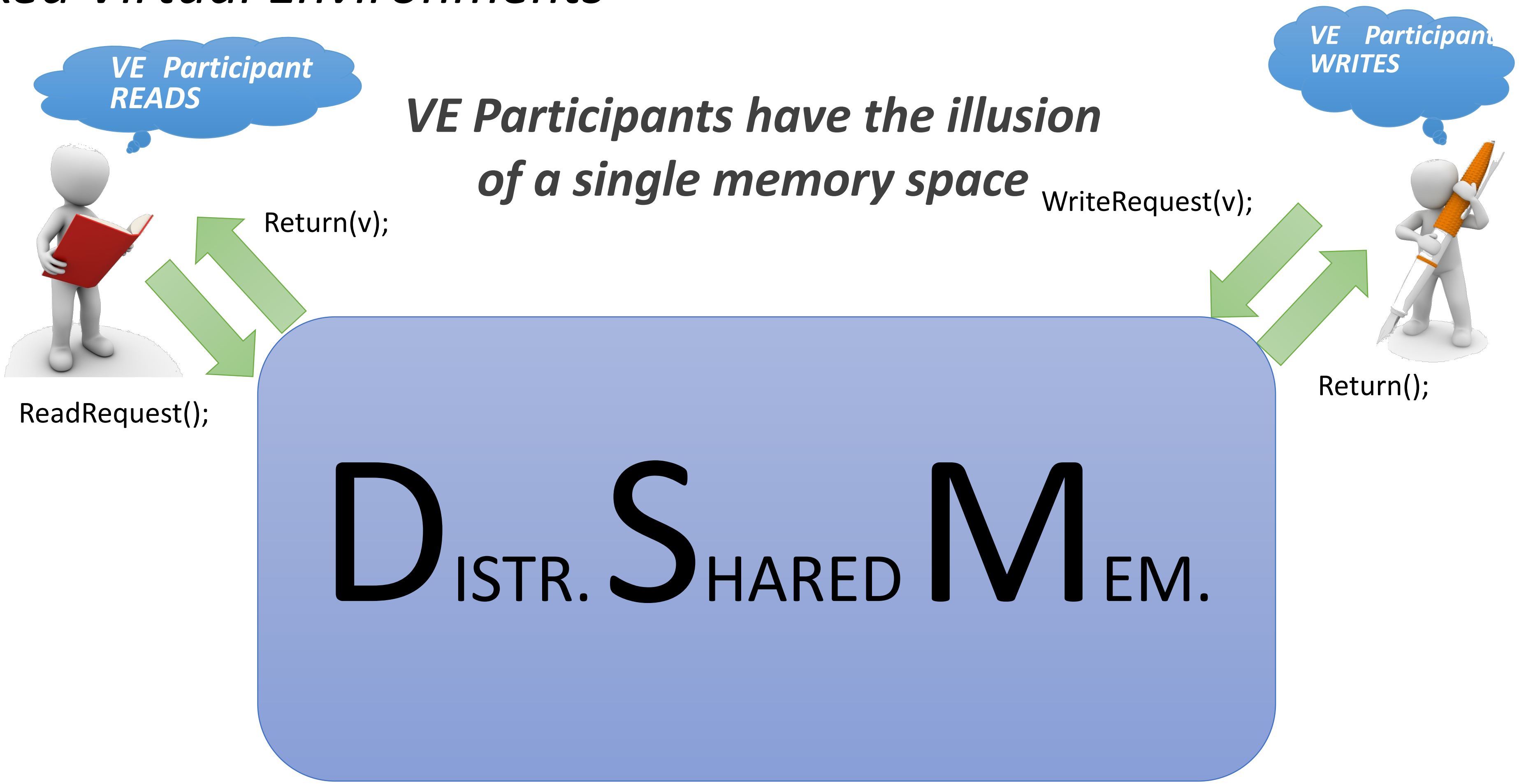
PROJECT GOAL

Goal: *Use Strongly Consistent Distributed Shared Memory in 3D Networked Virtual Environments*



PROJECT GOAL

Goal: *Use Strongly Consistent Distributed Shared Memory in 3D Networked Virtual Environments*



Project Goal

Goal: *Use Strongly Consistent Distributed Shared Memory in 3D Networked Virtual Environments*

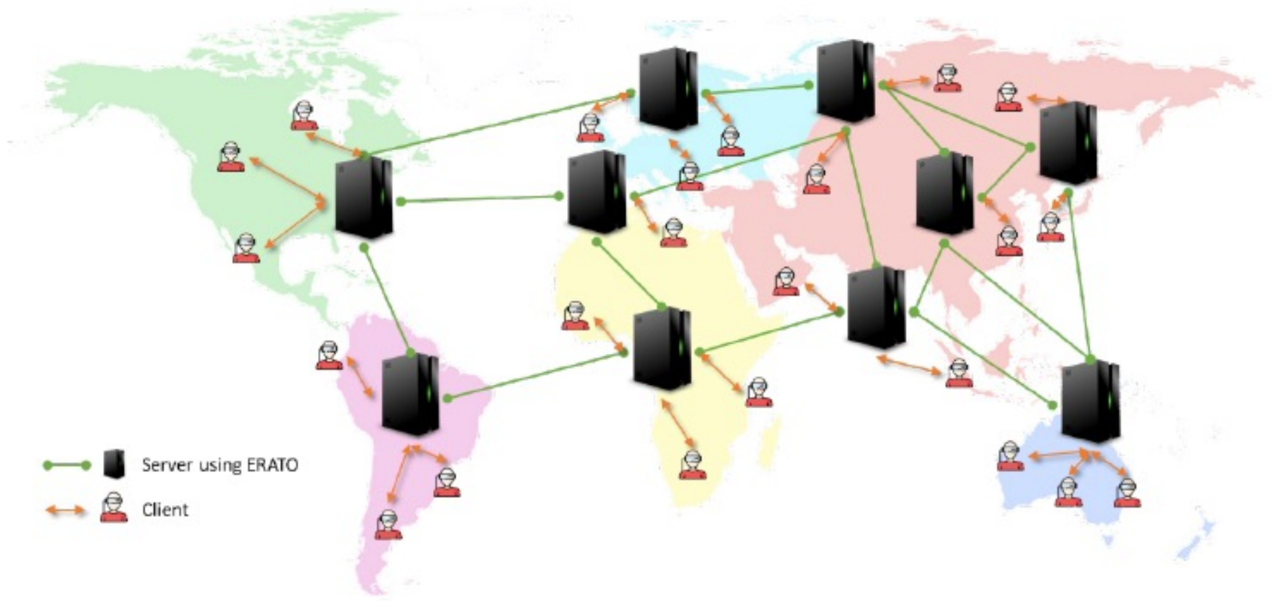


Figure 3 - A Distributed VE powered by ERATO. Interconnected clusters of servers handle state consistency and synchronization using transparently the ERATO DSS. Clients connect to a seemingly unified VE and smoothly transition across servers and interact with other clients and users. (The number of network connections and clients are for illustration only)

PROJECT WORK PLAN

	Work Package (WP)	Tasks
Phase 1	WP3: PoC Implementation	*Task 3.1 – Implement the distributed atomic shared memory ERATO*
		Task 3.2 – Implement a suitable 3D interactive NVE for a distributed lab validation
		Task 3.3 – Implement interfaces for utilizing the DSM (T3.1) in the NVE (T3.2)
Phase 2	WP4: Experimental Validation of PoC	*Task 4.1 – Deploy the PoC software implementation in a lab environment*
		Task 4.2 – Scalability tests
		Task 4.3 – Concurrency tests

Table 1: Summary of the two project Phases along with the respective WPs and Tasks

INTRODUCTION

What if… all the data located at one replica node?

EASY TO PROGRAM!!! But…

- *Single point of failure*
- *Not fault tolerant*
- *Not efficient – performance bottleneck*
- *Not very available*

One Replica server…

- ~~*Not fault tolerant,*~~
- ~~*Not efficient,*~~
- ~~*Not very available*~~

Approach: To mask failures we **replicate** the objects.

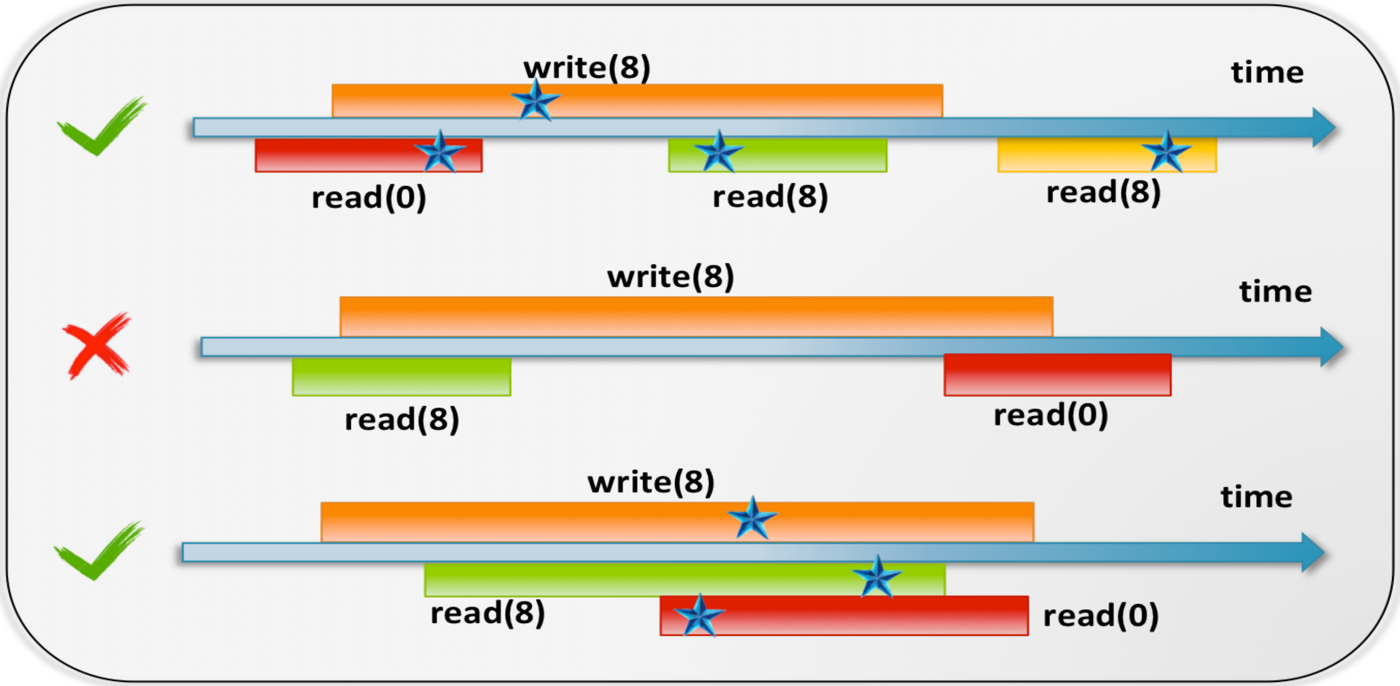
Challenge: Providing ***consistency – atomicity*** *[L79]* when read and write operations concurrently access different replicas

Challenge: Making read and write operations ***efficient***

- ❑In particular, in terms of ***communication exchanges***

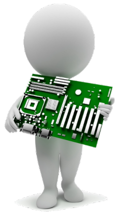
ATOMICITY [L79]

"Shrink" the interval of each operation to a serialization point so that the behavior of the object is consistent with its sequential type



***Notice:** Operations are not instant!*

DSM System Model



Components – Collection of processes

- Set $\mathcal{W}$ Writers and Set $\mathcal{R}$ Readers
- Set $\mathcal{S}$ replica servers (maintaining copy of the object) organized in Quorums

Operations

- ***write(v):*** updates the object value to *v*
- ***read():*** retrieves the object value
- ***Well-Formedness*** (only a single operation at a time)

Communication

- Message-Passing
- Asynchronous
- Point-to-point Reliable Channels (messages are not lost or altered)

Failures

- *Crashes* – Failure prone processes
- *All but one quorum may be faulty*

Efficiency Metrics

Message complexity — The *worst case* number of messages exchanged (Failure Free scenario).

Operation latency ← ***Computation time*** + ***Communication delay***

Computation time: computation “***steps***” in each operation.

Communication delay: accounts communication “***exchanges***”.

A collection of sends and receives for a specific message type within the protocol

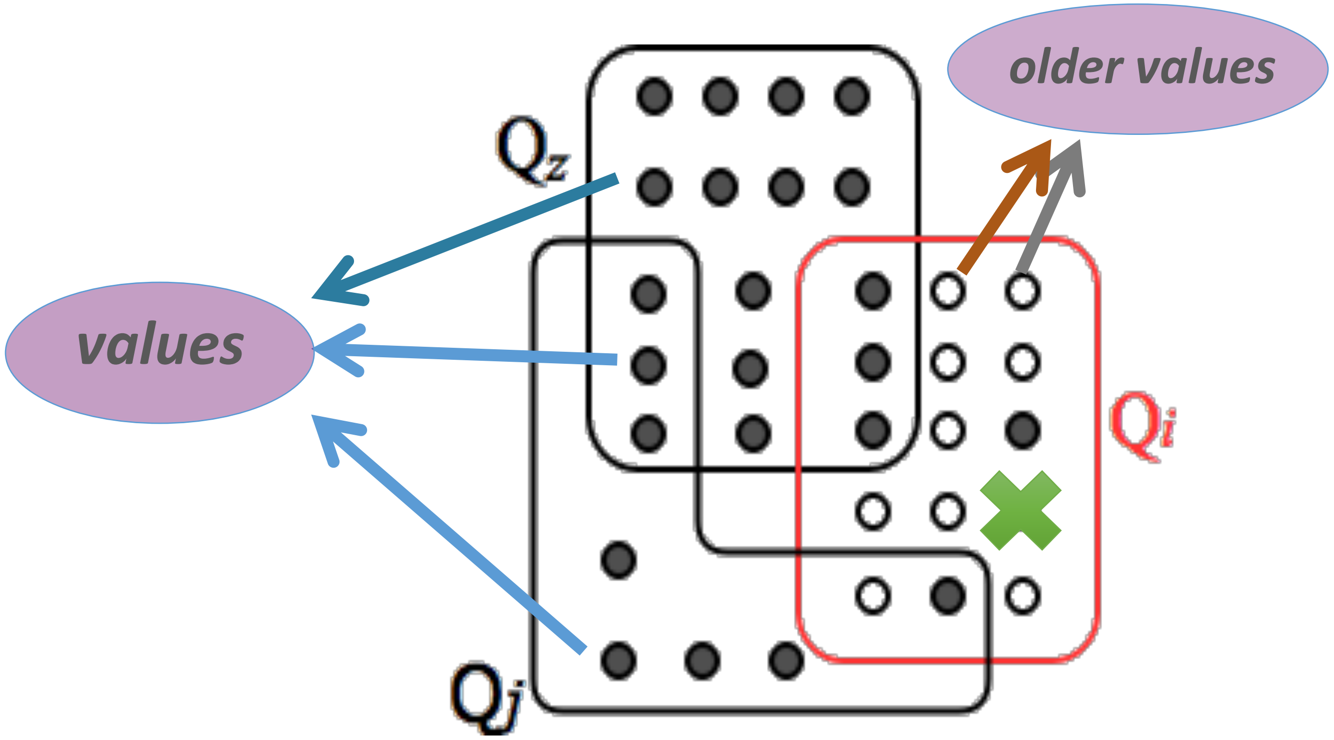
Related Work

Model	Algorithm	Read Exch	Write Exch	Read Comm	Write Comm
SWMR	ABD [ABD96]	4	2	$4\|S\|$	$2\|S\|$
SWMR	OH-SAM [HNS17]	3	2	$\|S\|^2 + 2\|S\|$	$2\|S\|$
SWMR	SLIQ [GNS08]	2 or 4	2	$4\|S\|$	$2\|S\|$
SWMR	ERATO	2 or 3	2	$\|S\|^2 + 3\|S\|$	$2\|S\|$
MWMR	ABD-MW [LS97]	4	4	$4\|S\|$	$4\|S\|$
MWMR	OH-MAM [HNS17]	3	4	$\|S\|^2 + 2\|S\|$	$4\|S\|$
MWMR	CWFR [GNRS11]	2 or 4	4	$4\|S\|$	$4\|S\|$
MWMR	ERATO-MW	2 or 3	4	$\|S\|^2 + 3\|S\|$	$4\|S\|$

For this project, we choose algorithm **ERATO-MW** for the underlying DSM Service

QUORUMS

Quorum System: *Given a set of servers, a quorum system is a collection of subsets of servers with **non-empty** pairwise intersections.*



Q_z, Q_j and ***Q_i*** are quorums.

Quorum System is the set $\{Q_i, Q_j, Q_z\}$

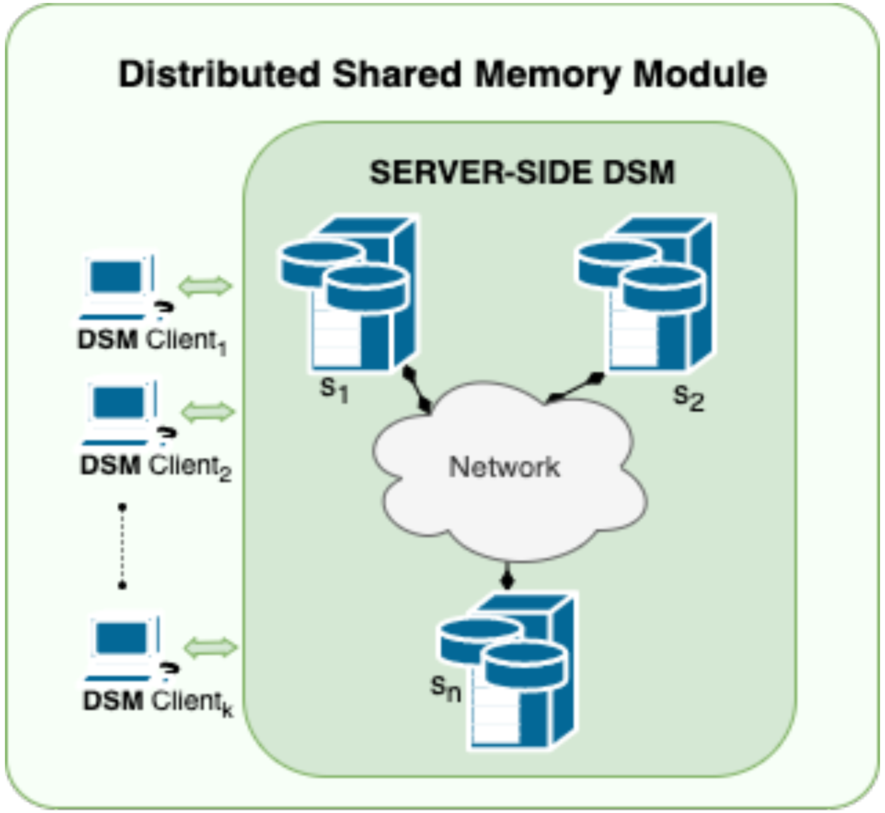
Faulty Quorum: Contains a faulty process, i.e., Q_i

ERATO-MW Fault-Tolerance: All but one quorums may crash

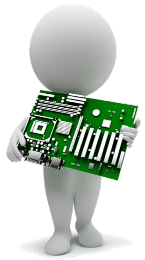
DSM Implementation

Main Tasks

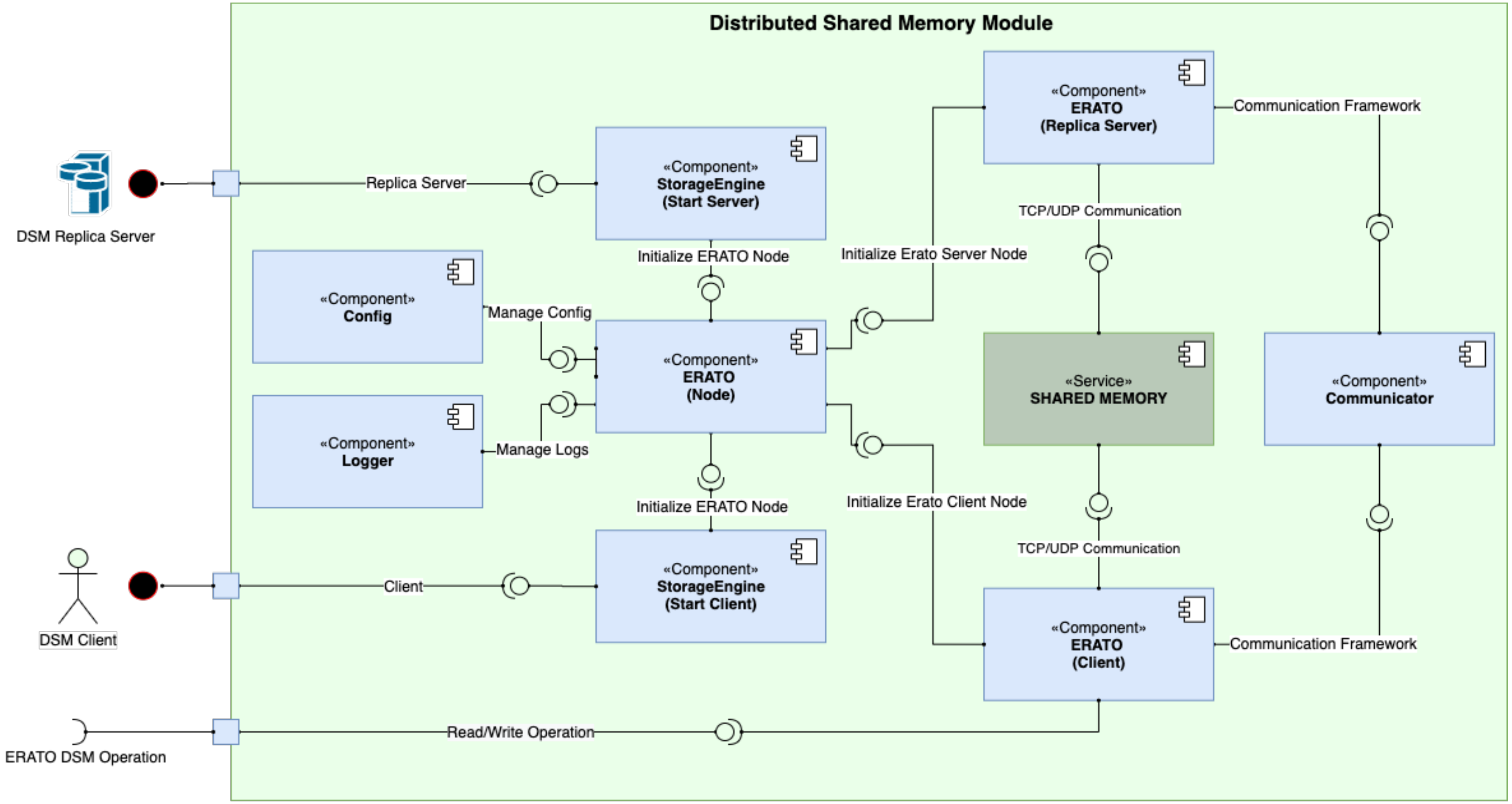
- Build the communication framework that supports communication:
  - Client-to-server
  - Server-to-server
- Implement the read/write protocols for both clients and servers based on the designed principles of ERATO
- Implement a strategy of dividing the replica servers into Quorums
- Evaluate the correctness of the implementation through exhausting test-runs



DSM Architecture

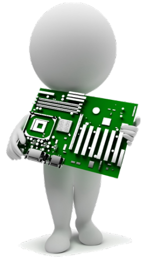


DSM Implementation



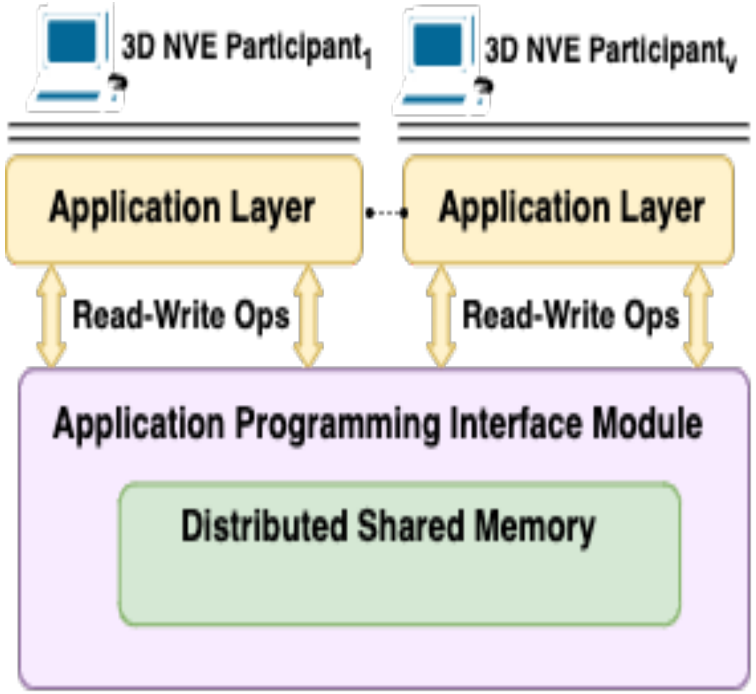
Software Components of the DSM

3D NVE IMPLEMENTATION



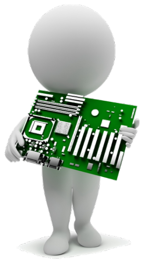
Main Objective

- Build a 3D Networked Virtual Environment (NVE) with features and complexity parameterization that are necessary for conducting the lab validation in the next phase.
- For the implementation of the NVE we used the **Unity3D** game engine to provide the asynchronous real-time processing nature of a game engine.



NVE Architecture

3D NVE IMPLEMENTATION

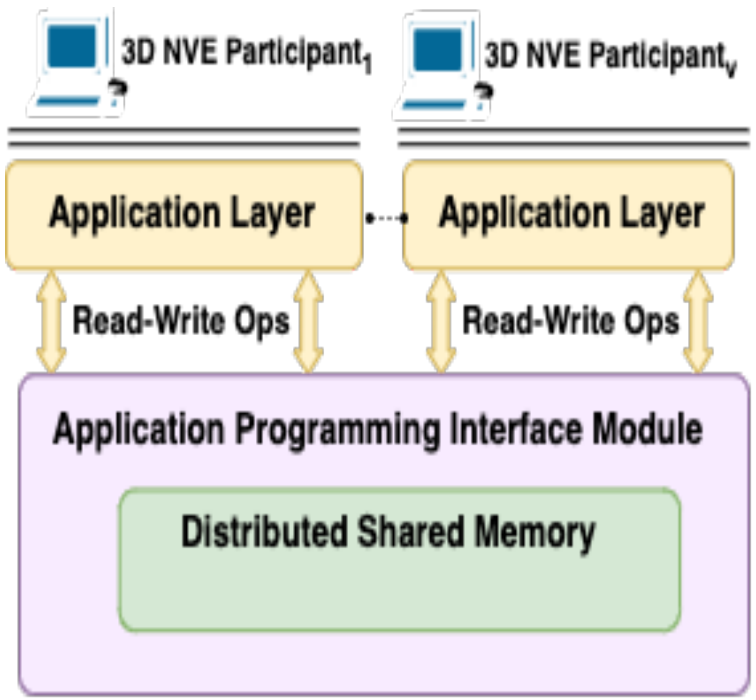


NVE Concept

- **Leader-Follower** in drone flocks. A set of drones acting as leaders and each is followed by a set of drones.

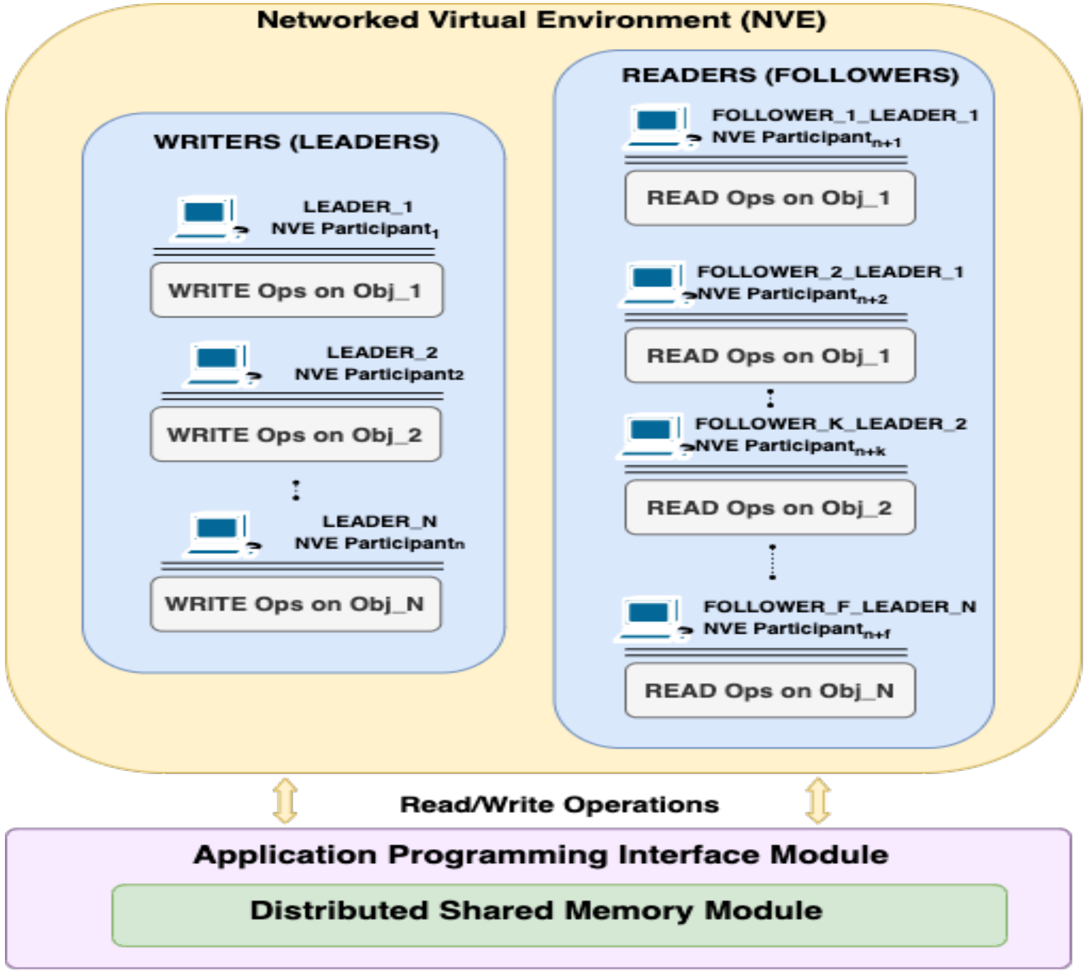
Why?

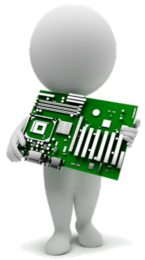
- Introduces ***write operations*** when leaders update their position in 3D space.
- Introduces ***read operations*** when followers retrieve their leader's position
- Allows us to examine ***scalability*** of the service
- Allows us to examine ***fault-tolerance*** of the service



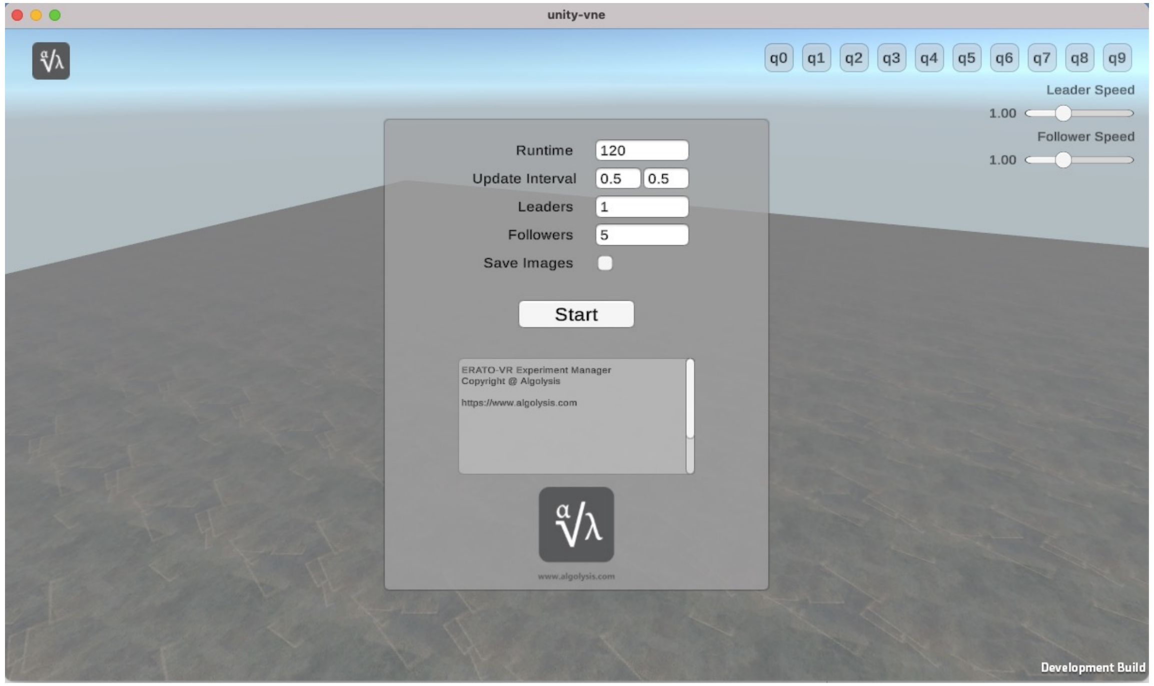
NVE Architecture

3D NVE Implementation

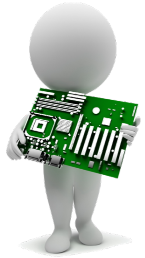




3D NVE Interface



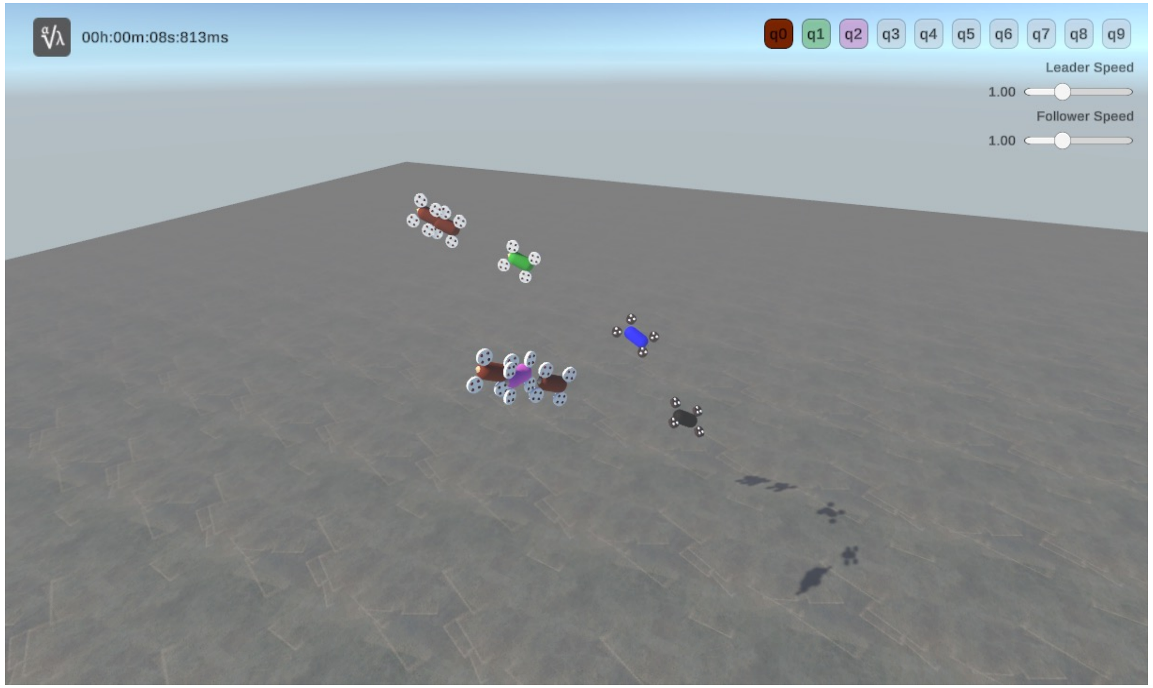
Start-Up Interface



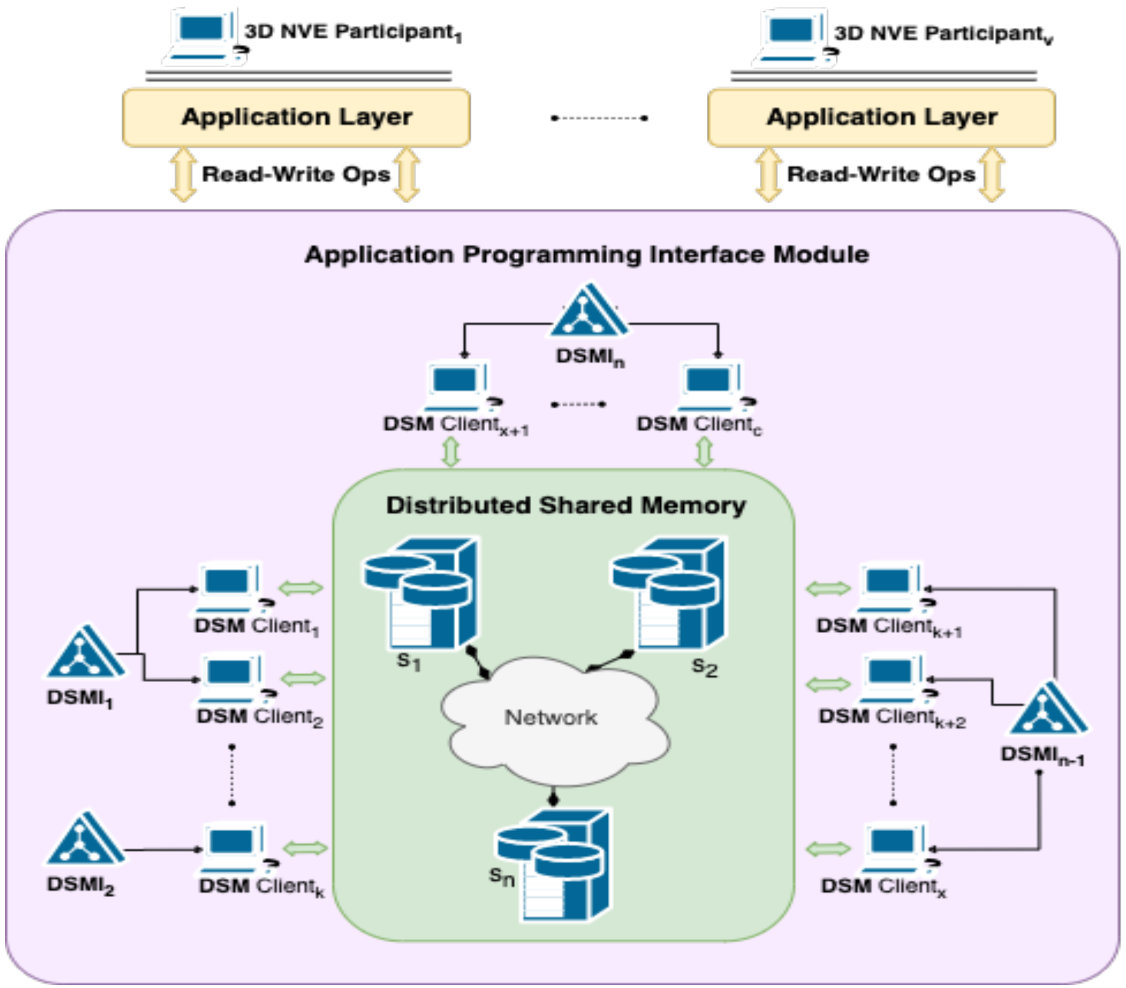
- ✓ It allows the user to select various runtime parameters

Runtime Interface

- ✓ Uses drone-coloring for observing the system behaviour

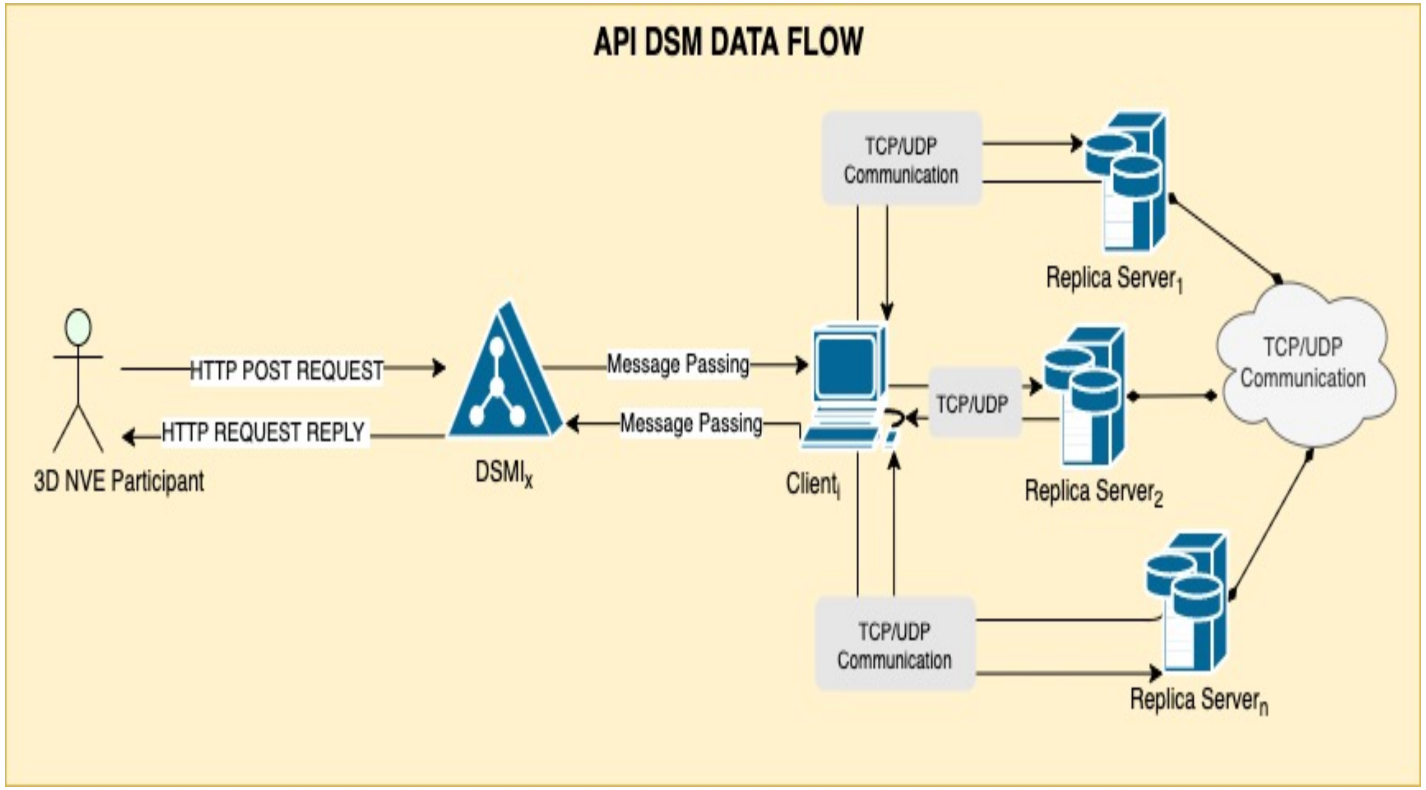


INTERFACES IMPLEMENTATION

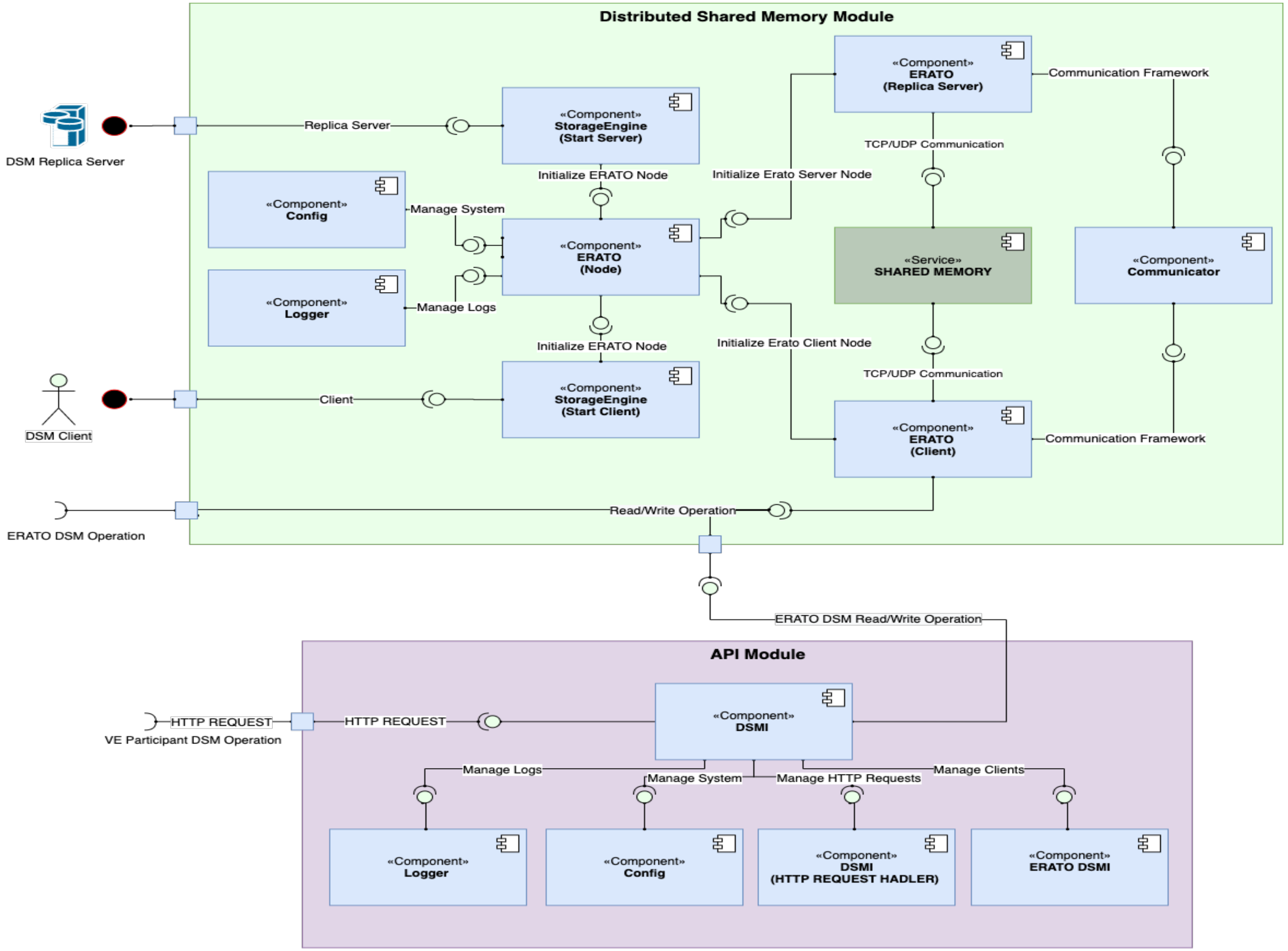


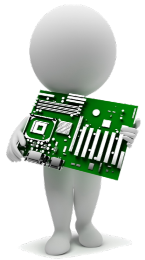
The API is the glue that ties together the DSM and the VE

API DATAFLOW

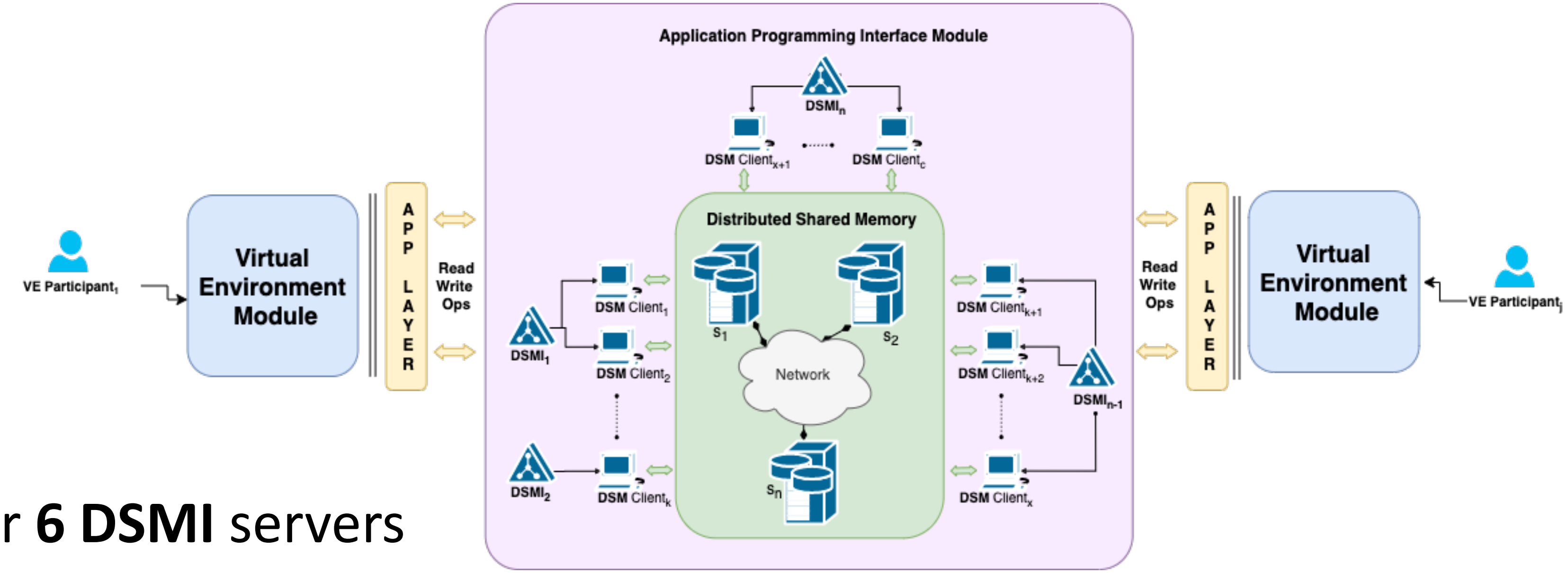


RESTfull API ARCHITECTURE





EXPERIMENTATION SETUP



- **API Module Configuration: 3** or **6 DSMI** servers
  - **one-to-one** relation of NVE Participants and DSMI's
- **DSM Service: 3** or **5 Replica** servers
- **Deployment:** over a network (i.e., either LAN or Ethernet)
- **Communication:** ***point-to-point*** bidirectional links implemented with ***DropTail*** queue.
- **Topologies:** on ***Raspberry Pi's*** and on ***Amazon Web Services***.

DSM System Model

Scalability

- $\mathcal{F} \in [1, 2, 5, 10, 15]$, $\mathcal{W} \in [1, 2, 5]$, $\mathcal{S} \in [3, 5]$
- $\mathcal{R}$ = ***|W| x |F|***

Contention

- **Fix Scheme** – operations invoked at the interval *[0.5, 1.0]*
- **Stochastic Scheme** – operations invoked at random interval [0.25…*1.0*].

Fault-Tolerance

- Introducing processor fail-crashes in the system in order to verify the guarantees and the responsiveness of the service

Network & Processing Capabilities

- Manipulating the deployment location **[rpis, aws]** and,
- Manipulating the dsmis participation **[3, 6]**.

EMPIRICAL RESULTS - SCALABILITY

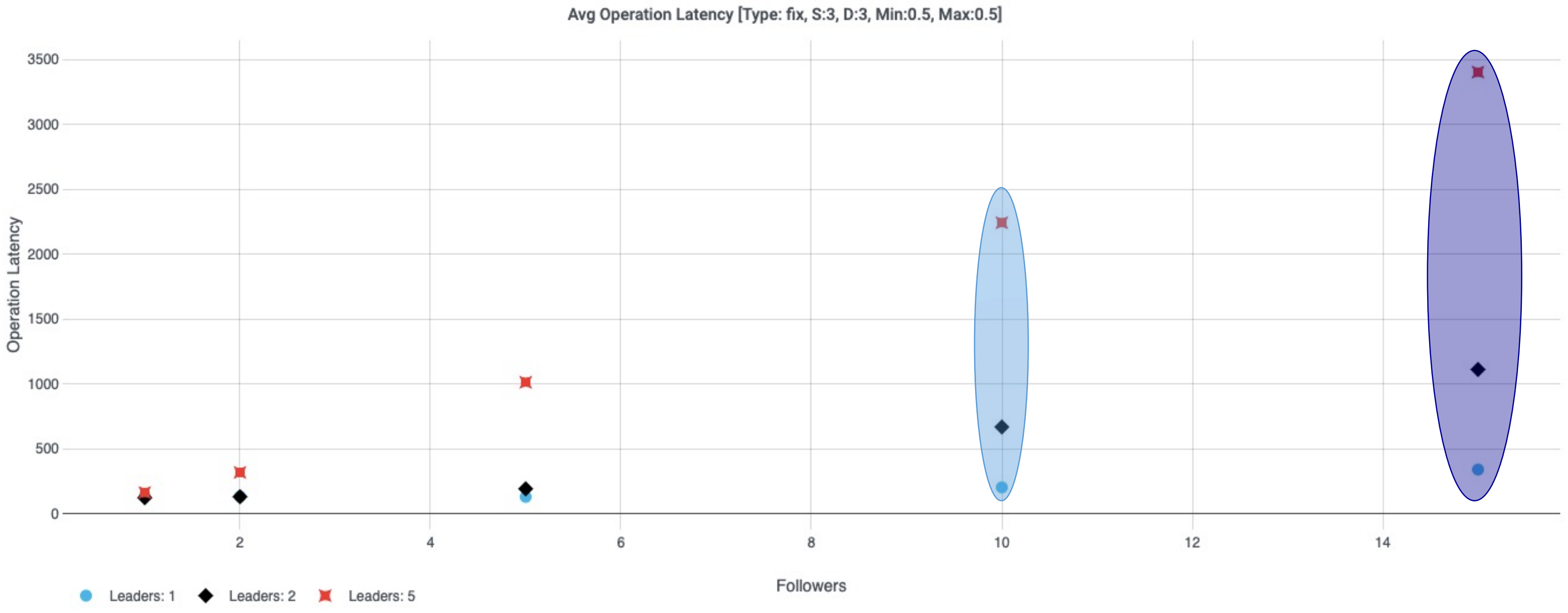


Figure 2 Average operation latency as the number of leaders and their followers increases

Scalability: the increasing number of readers and servers has a negative impact on the PoC software.

- Latency improves when we reduce the participants.

Empirical Results - Scalability

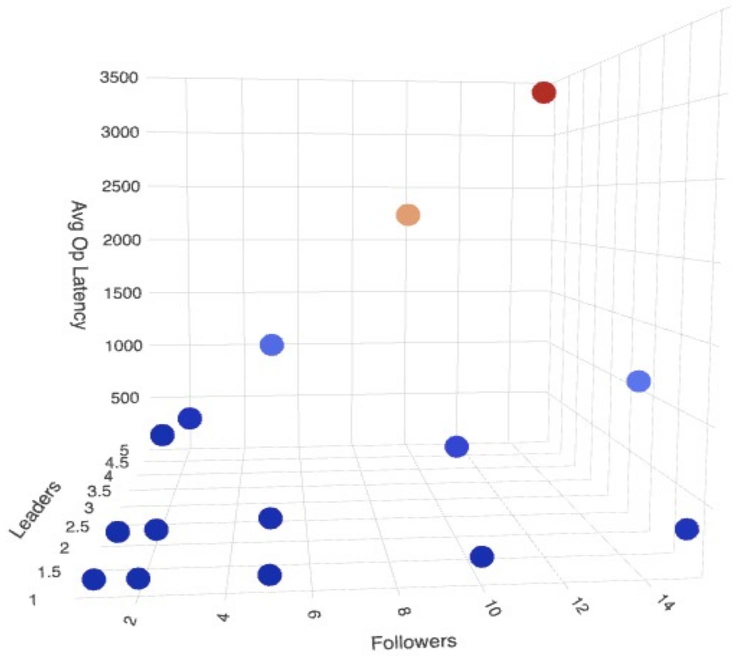


Figure 5 - Average Operation Latency as the leader and follower participation increases Type:fix, S:3, Min:0.5, Max:0.5

Figure 6 - Average Read/Write Operation Latency – Type:fix, S:3, L:1, F:2

Figure 8 - Average Read/Write Operation Latency – Type:fix, S:3, L:5, F:15

Scalability: the increasing number of readers and servers has a negative impact on the PoC software.

- Latency improves when we reduce the participants.

EMPIRICAL RESULTS - CONCURRENCY

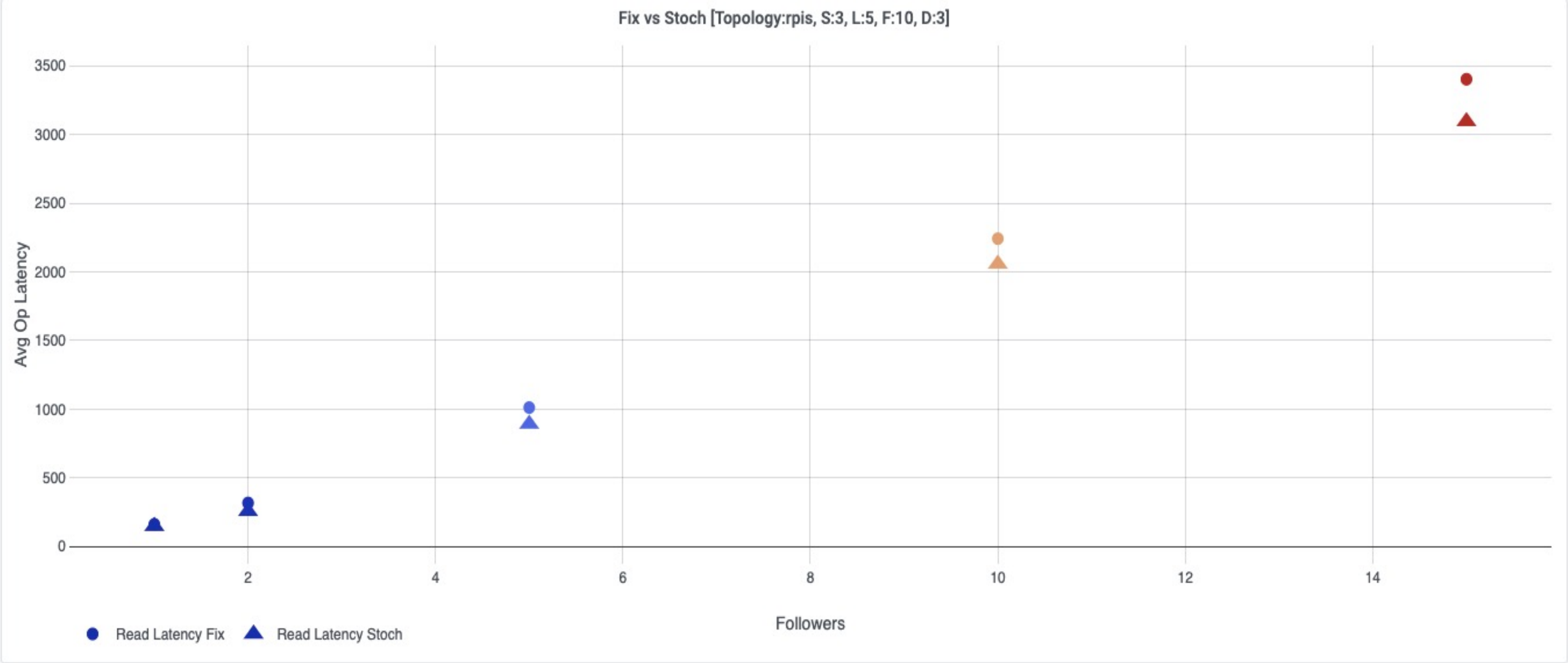


Figure 11 - Fix vs Stochastic Scheme – Topology:rpis, S:3, L:5, F:10, D:3

Contention: In the stochastic scheme operations complete faster.

- ***Why?*** Invocation time intervals are distributed uniformly.
- Fixed scheme causes congestion in the system.

EMPIRICAL RESULTS - TOPOLOGY

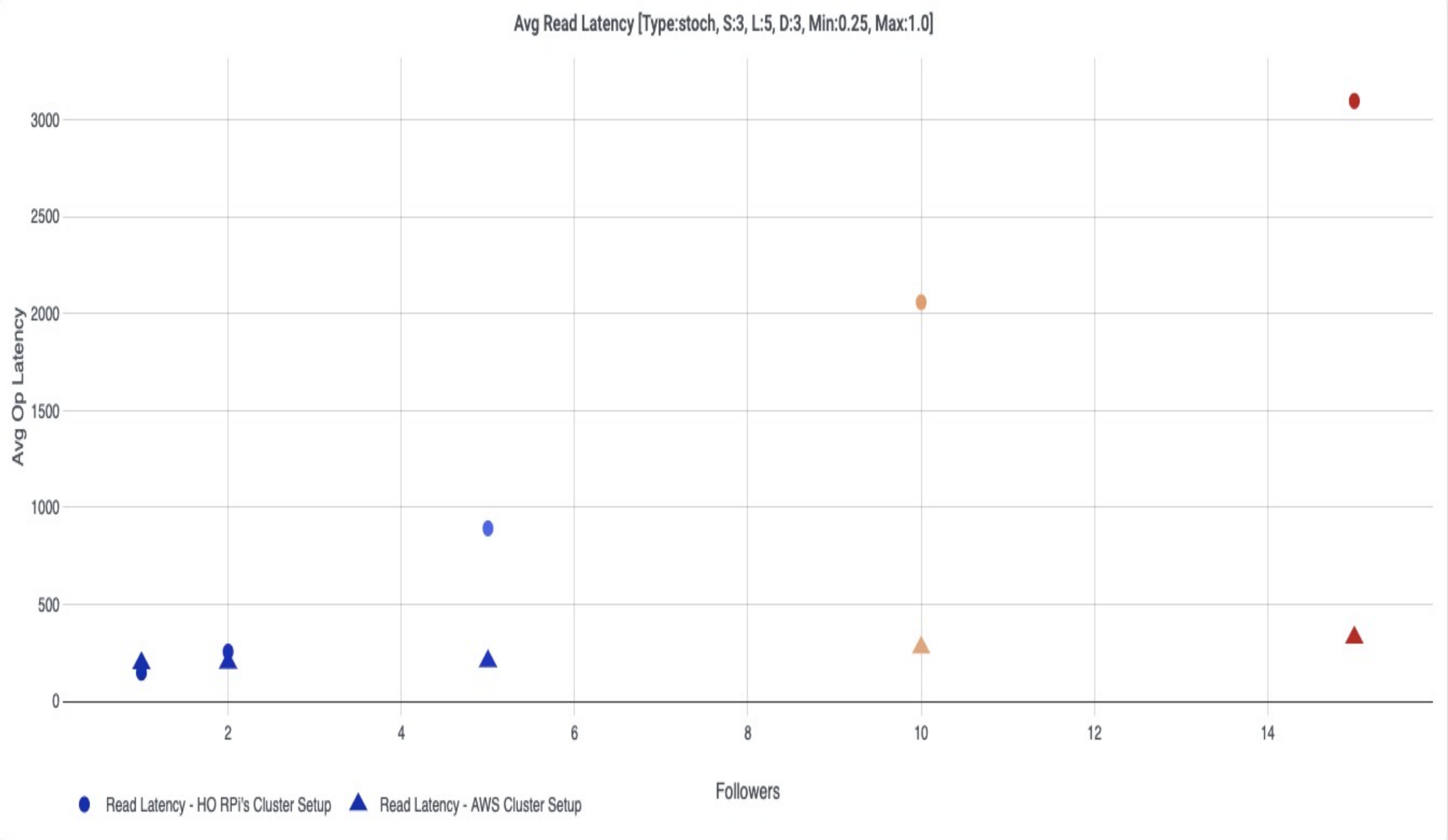


Figure 18 - The Deployment Effect on the Average Operation Latency – Type:Stoch, S:3, L5: Min:0.25, Max:1.0, DSMI's:3

Topology: impacts performance.

- The PoC software performs better when deployed on AWS.
- Expected as the nodes reserved on AWS have much higher capabilities than the Raspberry Pi nodes hosted at the HO.

Empirical Results - Topology

Figure 19 - The Deployment Effect on Read/Write Operations latency RPIS vs AWS – Type:Stoch, Leaders 5, Followers 2

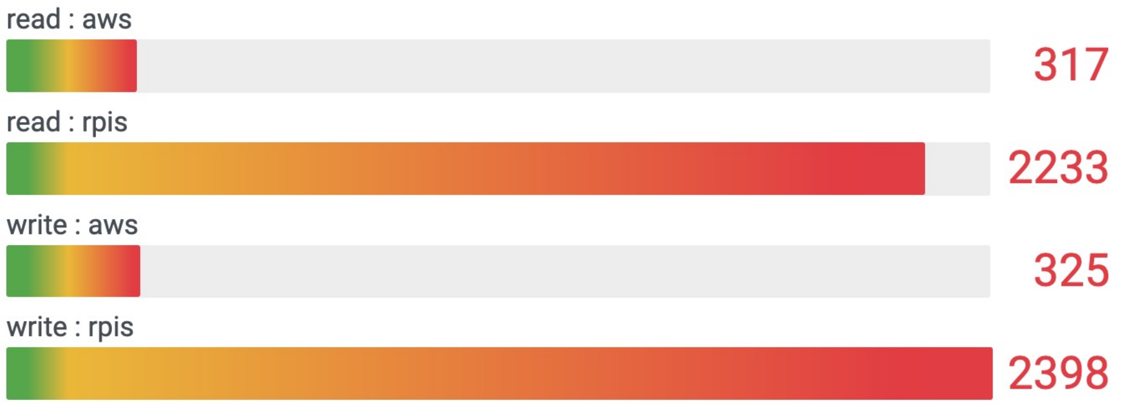


Figure 20 - The Deployment Effect on Read/Write Operations latency RPIS vs AWS – Type:Stoch, Leaders 5, Followers 10

Topology: impacts performance.

- The PoC software performs better when deployed on AWS.
- Expected as the nodes reserved on AWS have much higher capabilities than the Raspberry Pi nodes hosted at the HO.

EMPIRICAL RESULTS – DSMI's

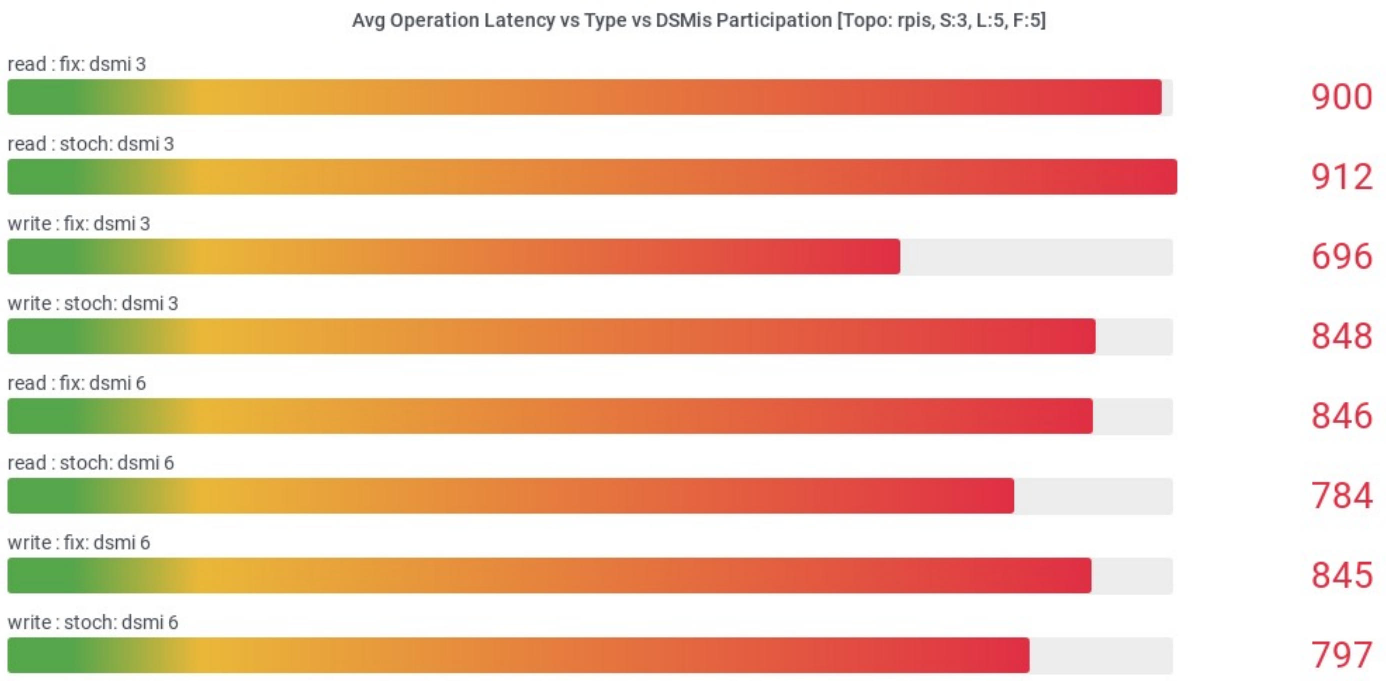


Figure 23 – How the Average Operation Latency of operations is affected as the number of DSMI's increases and the invocation scheme changes

DSMI's selection: impacts performance.

- Increasing the DSMI's in the service from 3 to 6 decreases the operation latency.
- While increasing the DSMI's appears to benefit the average operation latency of the system, the overall operation latency is still prohibited.

Empirical Results – Fault Tolerance

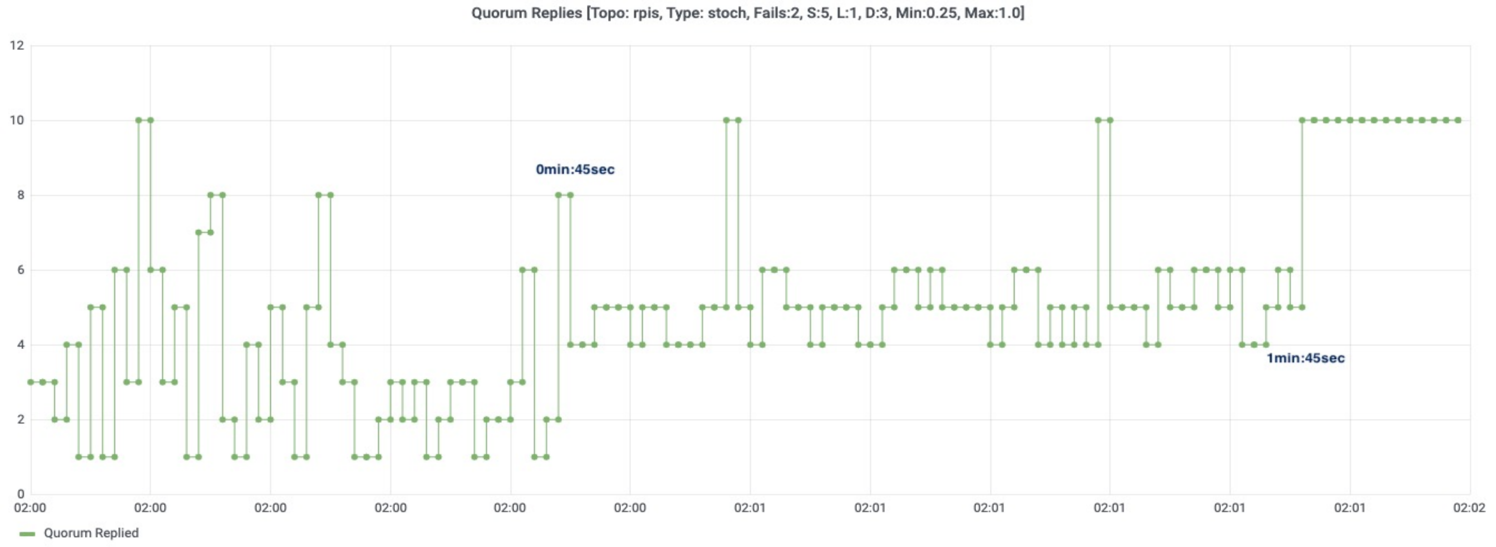


Figure 13 – DSM service consists 10 Quorums, 2 replica server failures in total, Replica Server B crashes at 0:45 then, Replica Server A crashes at 1:45

QuorumID	Replica Servers	QuorumID	Replica Servers
1	A, B, C	6	A, D, E
2	A, B, D	7	B, C, D
3	A, B, E	8	B, C, E
4	A, C, D	9	B, D, E
5	A, C, E	10	C, D, E

Empirical Results – Fault Tolerance

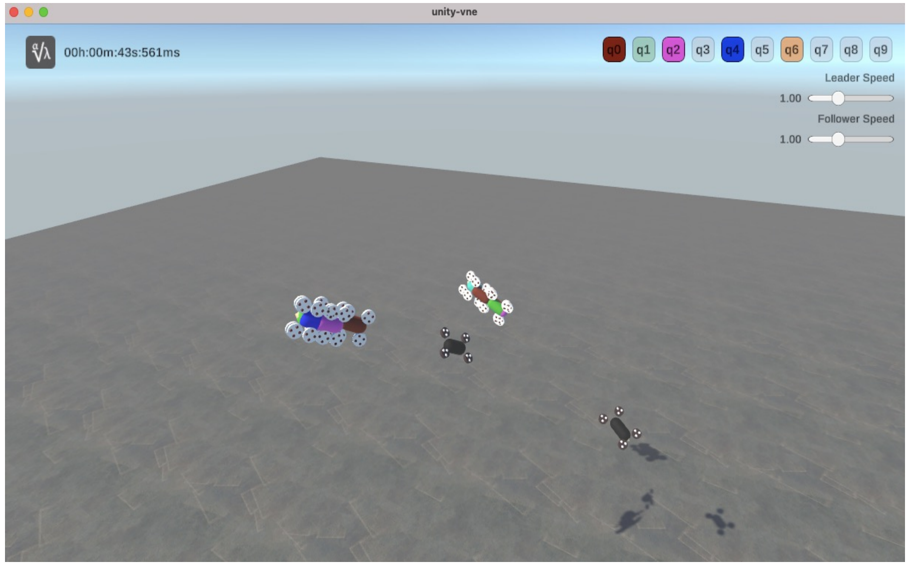


Figure 15 – A visual snapshot from the NVE interface at time 0:43 of the execution where all quorums non-faulty

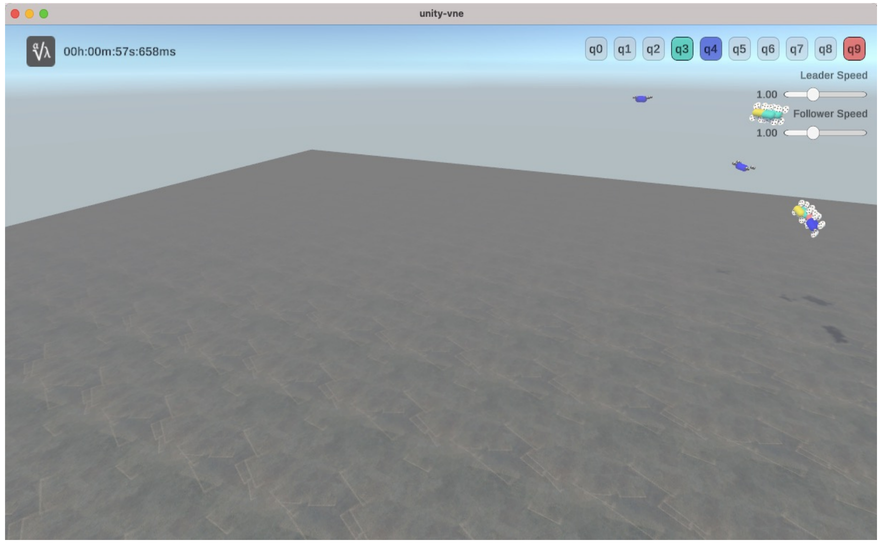


Figure 16 – A visual snapshot from the interface moments after ServerB crashes, Faulty Quorums IDS: 1 (q0), 2 (q1), 3 (q2), 7 (q6), 8 (q7), 9 (q8)

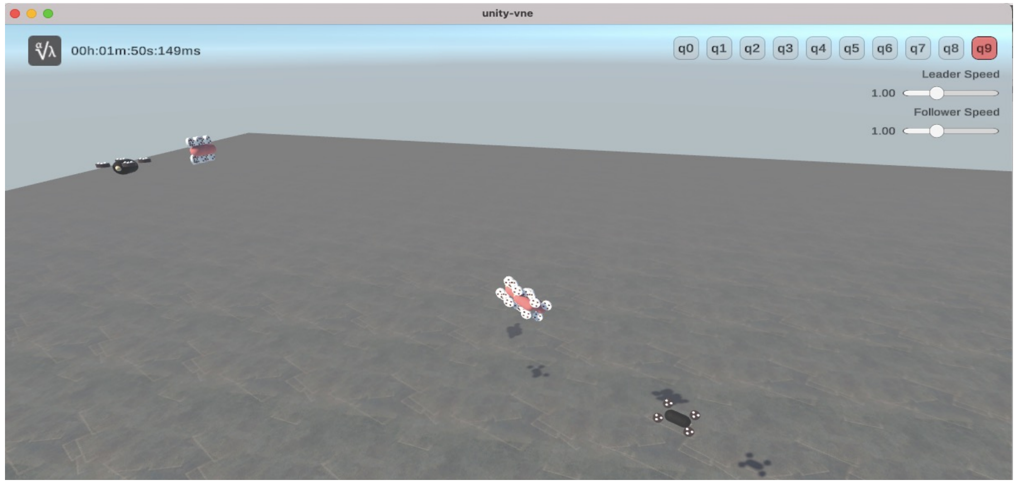


Figure 17 – A visual snapshot from the interface moments after ServerA crashes, All quorums are faulty except from quorum with id 10 (q9)

EMPIRICAL RESULTS - OVERVIEW

Scalability: the increasing number of clients and servers has a negative impact on **the PoC software**.

Contention: Stochastic scheme -> ops complete faster.

- ***Why?*** Invocation time intervals are **distributed uniformly**.
- Fixed scheme causes **congestion** in the system.

Topology & Processing Capabilities:

- substantially impacts the performance of the PoC!

CONCLUSIONS

- NVE's are time sensitive applications requiring small delays when obtaining data from remote locations, e.g., operations **less than 100ms**

- Results suggest that read/write operation latencies demand more than **200ms** in scenarios with small congestion and few participants, when the replicas are deployed on cheap, commodity hardware like Raspberry Pi's.

- Unfortunately, the average delay increases significantly, **up to 3000ms**, as participation increases and higher contention is assumed.

- ✓ Things appeared more promising when the service was deployed on more powerful virtual machines on AWS
  - ✓ Stable latency of **300ms even during worst-case scenarios**

What's Next?

- Results suggest that the technology offers promising capabilities, but it is not yet mature to be deployed widely

In order to make it ready, we plan to exploit our results:

  - Device & use new more robust algorithmic solutions (both for the DSM & NVE)
  - Utilize different transfer protocol techniques (i.e., UDP)
  - Attempt to decrease algorithms messages size

Goal: Reduce latency and yield better results for time-sensitive NVE applications.

Together with close collaborations in Cyprus and abroad, we plan to

  - ✓ Device follow-up projects and,
  - ✓ Seek funding from National, European and International agencies in order to improve the technology.

THANK YOU!

algolysis

algorithmic solutions

www.algolysis.com

research@algolysis.com