HARISMA
MARITIME SURVEILLANCE

ARES: A God of War to Bring “Peace" to Heterogeneous Robots

Nicolas Nicolaou, PhD

nicolas@algolysis.com

contact@algolysis.com

Funded by the European Union
NextGenerationEU

Cyprus tomorrow
RECOVERY AND RESILIENCE PLAN

Republic of Cyprus

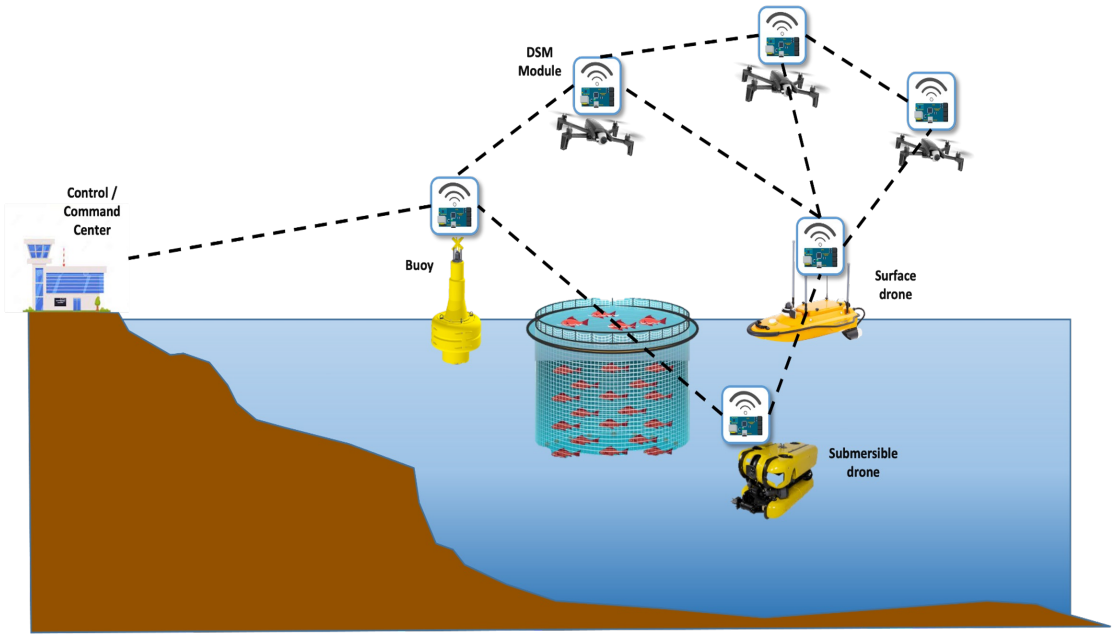
RESEARCH & INNOVATION FOUNDATION

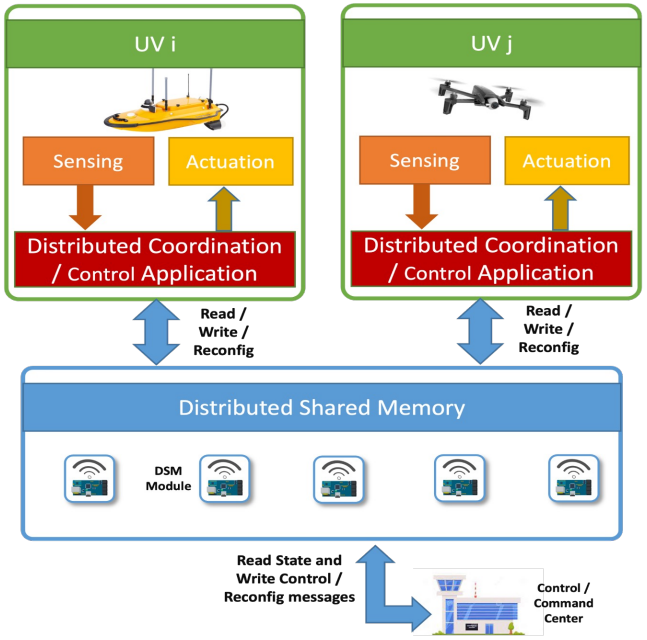
algolysis
algorithmic solutions

CYENS
CENTRE OF EXCELLENCE

CYPRUS MARINE & MARITIME INSTITUTE

CHARISMA Project – Architecture





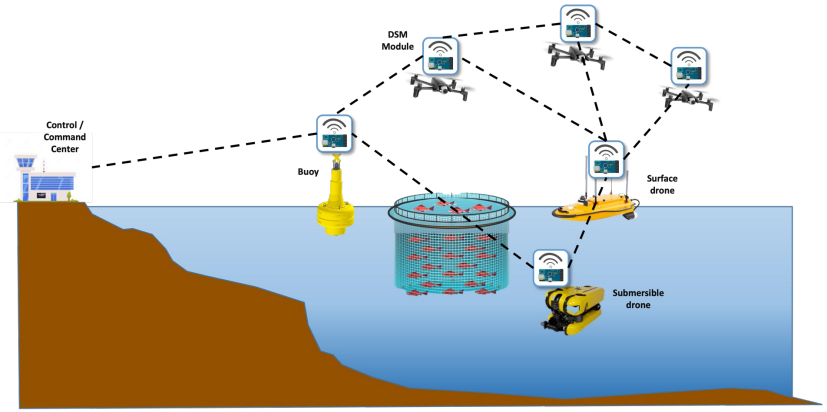
CHARISMA Project – Objectives

O1
- Build a HW module to extend swarm device capabilities

O2
- Develop an efficient DSM on top of the HW modules

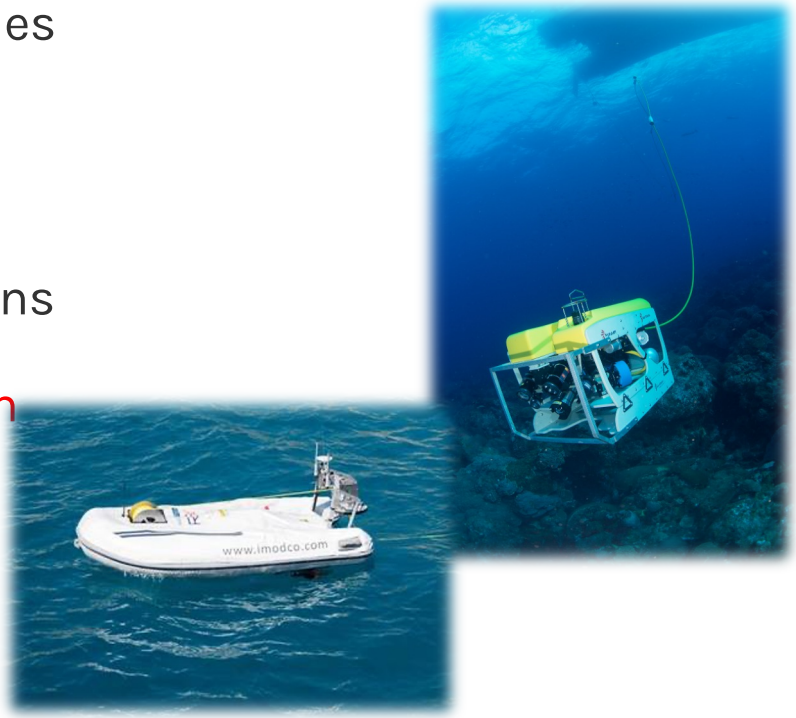
O3
- Develop swarm algorithms that will exchange info using the DSM

04
- Develop a digital twin that will allow an operator to have an overview and control the devices.

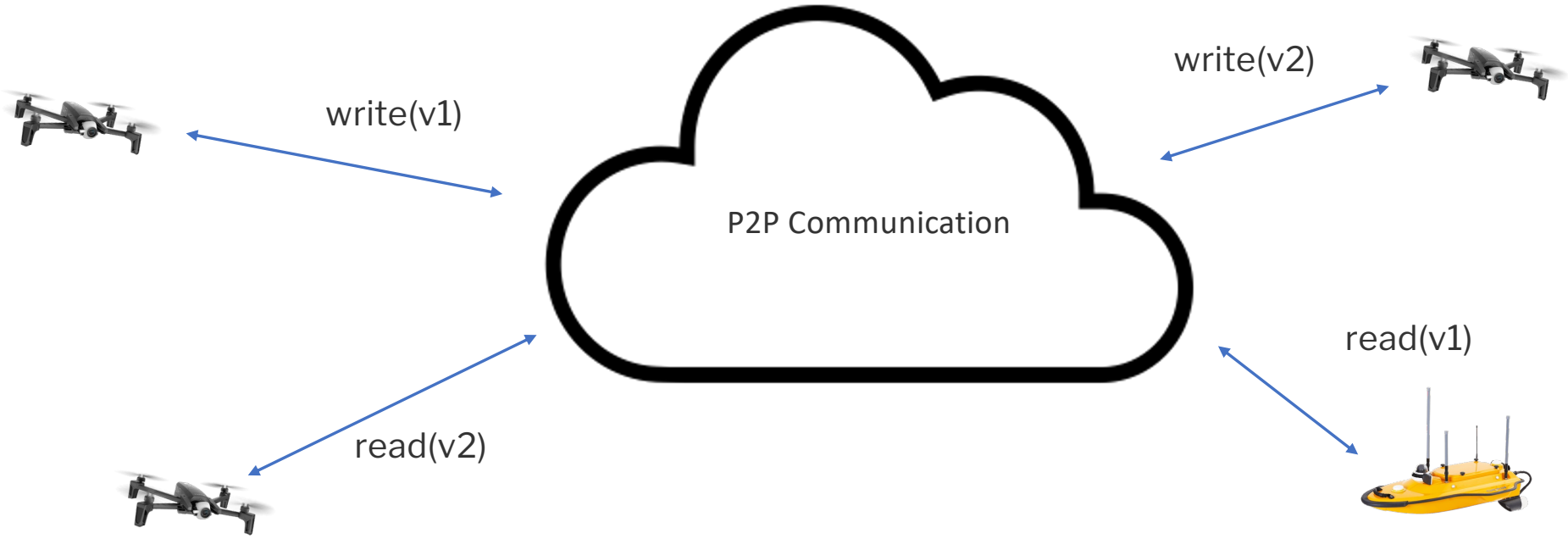


CHARISMA Project – Challenges

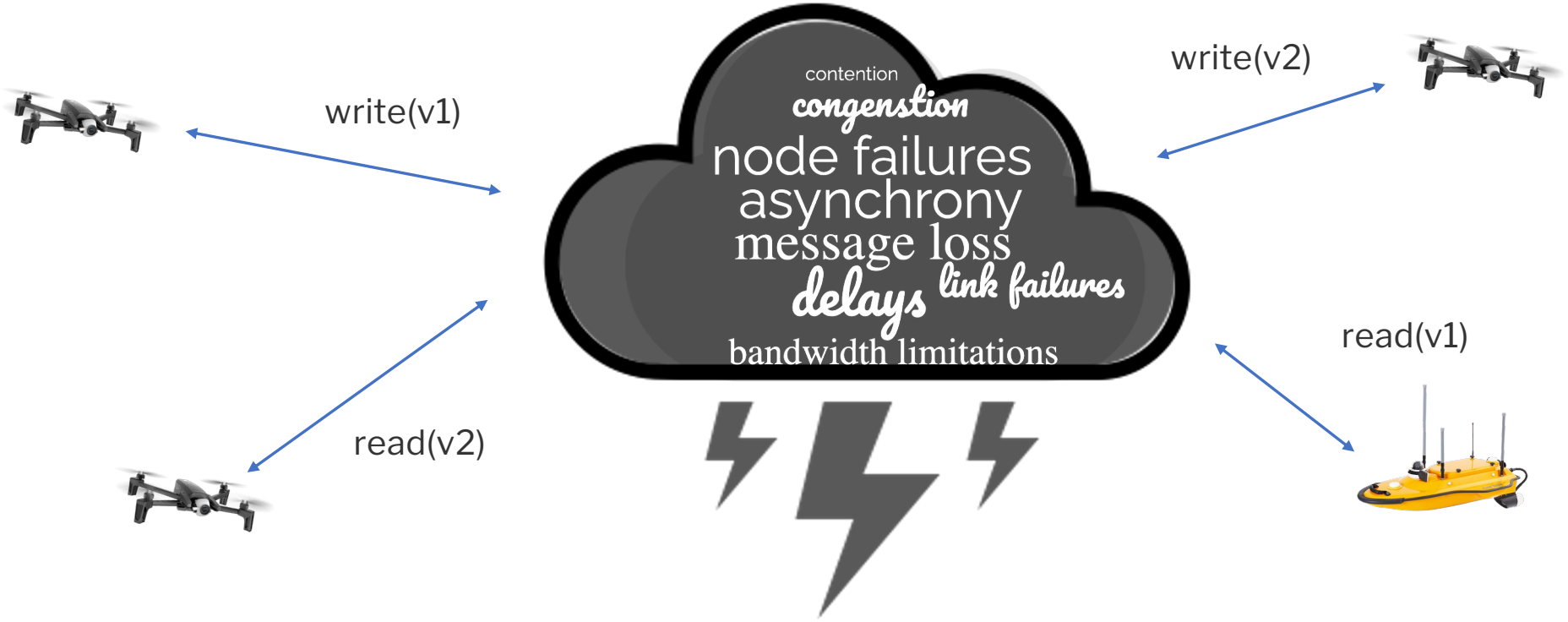
- Enable wireless communication over hundreds of meters
- Maintain a stable connection between the modules
- Enable seemles information exchange between devices
- Ensure the longevity of the service despite node failures
- Dynamically adapt the system to handle node joins and departures
- Build swarm algorithms that exchange minimum sensed information
- Achieve excellent collaboration between the various AUV
- Transmit minimum info to digital twin
- Allow operator in the loop



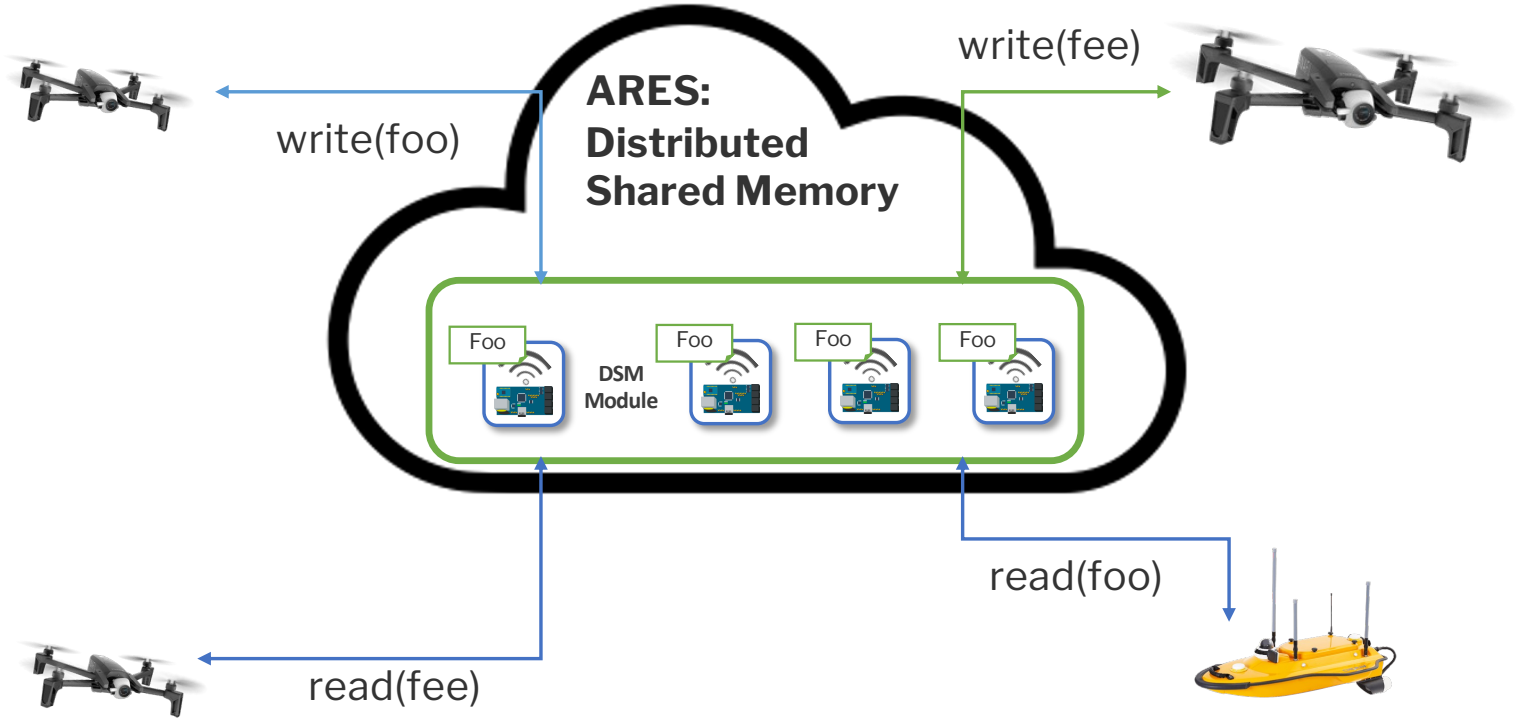
Communication Challenge



Communication Challenge



CHARISMA Communication at a glance



ARES: Adaptive, Reconfigurable, Erasure coded, atomic Storage

ARES: Implementation Characteristics

Modularity

- Read/Write operations are not aware of the underlying shared memory implementation
  - They are using the same access primitives

Erasure Coding

- Storage Efficiency
- The first EC dynamic atomic storage

Adaptivity

- Different shared memory algortihm may be used in every configuration
  - Satisfying application demands

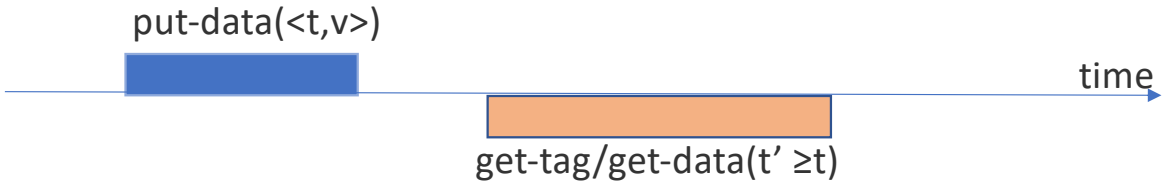
[NICOLAOU, N., CADAMBE, V., KONWAR, K., PRAKASH, N., LYNCH, N., AND MEDARD, M. ARES: Adaptive, Reconfigurable, Erasure Coded, Atomic Storage*. In Proc. of ICDCS, pp. 2195–2205 (2019*)]

Modularity: Data Access Primitives (DAP)

- Three logical operations:
  1. Get-tag(): discover the timestamp of the latest value written
  2. Get-data(): discover the value associated with the latest timestamp
  3. Put-data(): propagate the latest value and its associate timestamp
- Express read/write operations with respect to this 3 DAP

DAP Consistency Properties

- DAPs may be used to yield Consistent Implementations if they satisfy the following properties:
  - C1: If a put-data(<t,v>) completes before get-tag/get-data() it returns a tag ≥ t



  - C2: if get-data() returns <t,v> then put-data(<t,v>) completed before or is concurrent to get-data()

ARES read/wites using DAPs

Reader Protocol

- (t,v)=**get-data**()
- **put-data**(t,v)
- return v

Writer Protocol(val) (at w_i)

- t_{max}=**get-tag**()
- (t,v)= (<t_{max}+1,w_i>, val)
- **put-data**(t,v)
- return ACK

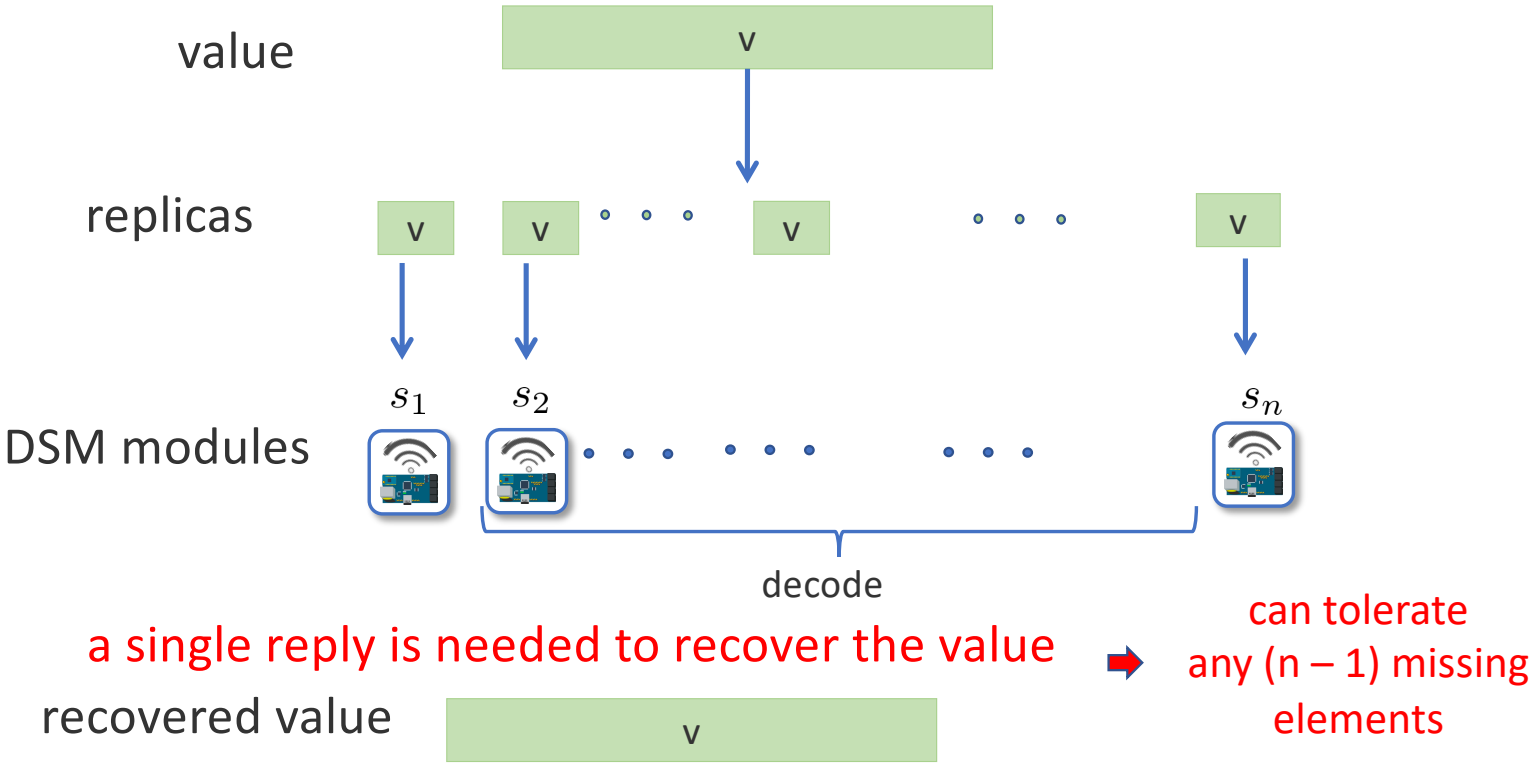
How are these DAPs implemented?

ARES DAPs: ABD/EC

- ABD: Replication Algorithm

- EC: Erasure Coded Algorithm

ARES ABD: Replication



ARES DAP: Algorithm ABD

get-tag() at p_i
- Request t from all modules
- Discover t_{max}=max(t) from the replies of a majority
- Return t_{max}

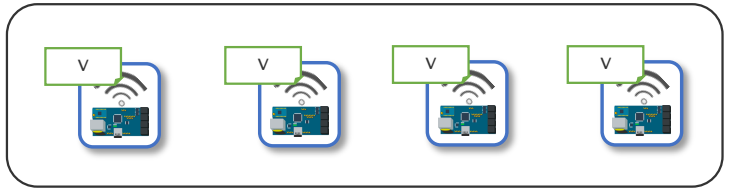
get-data() at p_i
- Request (t,v) from all modules
- Discover (t_{max},v) from the replies of a majority
- Return (t_{max},v)

put-data(<t,v>) at p_i
- Send (t,v) to all
- Wait until receiving ACK from majority

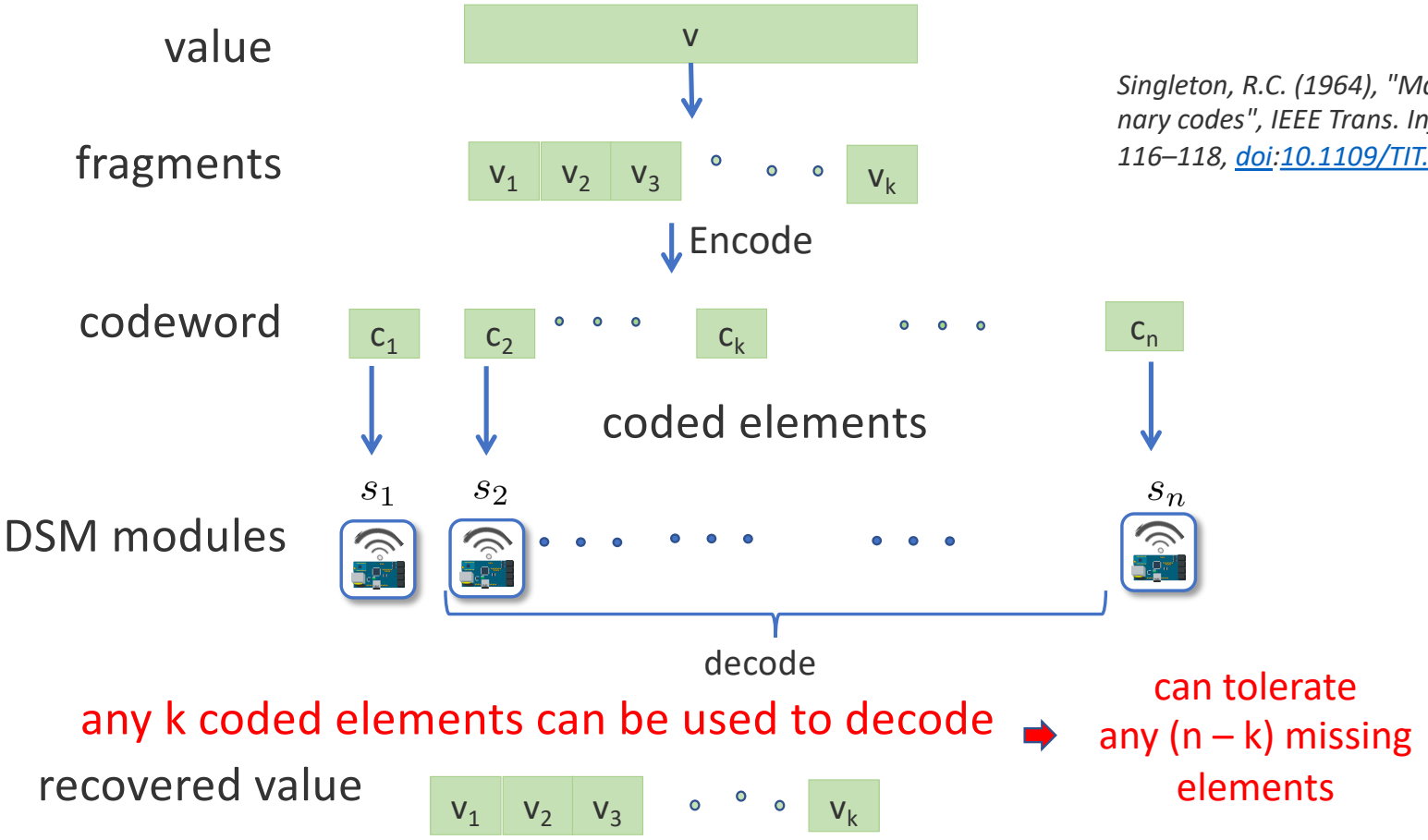
Receive(Get-Tag) at s_j
- Send t to the requester

Receive(Get-Data) at s_j
- Reply with local (t,v)

Receive(Put-Data, <t',v'>) at s_j
- If t' > t
  - <t,v> = <t', v'>
- Reply with ACK



ARES EC: Erasure Coding ([n, k] MDS Codes)

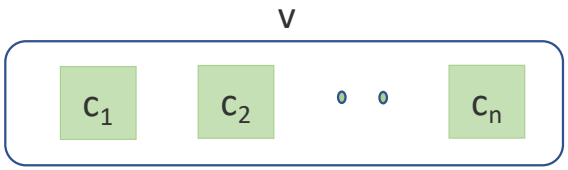


*Singleton, R.C. (1964), "Maximum distance q-nary codes", IEEE Trans. Inf. Theory, **10** (2): 116–118, doi:10.1109/TIT.1964.1053661*

any k coded elements can be used to decode ➡ can tolerate any (n – k) missing elements

ARES DAP: Algorithm EC

- Write Operation write(v)
  - Apply erasure coding on v
  - Send code c_i to module s_i



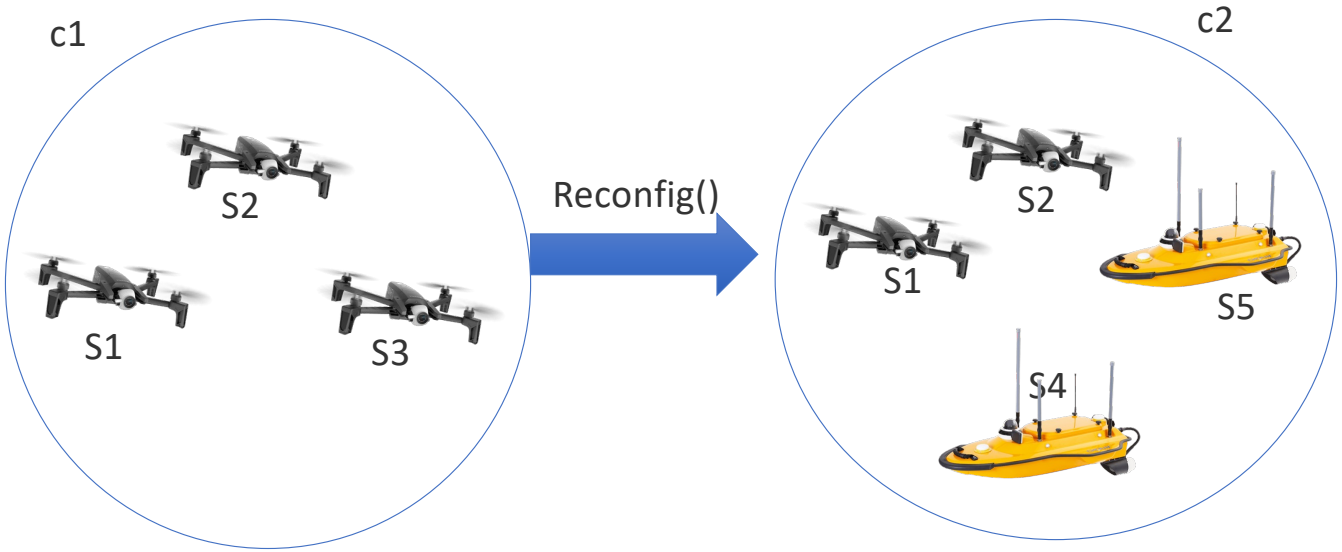
- Read Operation read()
  - Collect k codes from the servers
  - Decode and return the value of v

- Ensures Strong Consistency

[**NICOLAOU, N.**, CADAMBE, V., KONWAR, K., PRAKASH, N., LYNCH, N., AND MEDARD, M. ARES: Adaptive, Reconfigurable, Erasure Coded, Atomic Storage. *In Proc. of ICDCS, pp. 2195–2205 (2019*)]

Logevity: How ARES handles node failures/joins/departures?

- Configuration: set of nodes that host the data (dsm modules)
- Reconfiguration: Change the host configuration due to:
  - Device failures, Device Joins/Departures



LYNCH, N., AND SHVARTSMAN, A. RAMBO: A reconfigurable atomic memory service for dynamic networks. *In Proceedings of 16th International Symposium on Distributed Computing (DISC)* (2002), pp. 173–190.

Reconfiguration Service

- A recon operation performs 2 major steps:
  1) Configuration *Sequence Traversal*
  2) Configuration *Installation*
     - Transfers the object state from the old to the new configuration

```
6:  operation reconfig(c)
      if c ≠ ⊥ then
8:      cseq ←read-config(cseq)
        cseq ← add-config(cseq, c)
10:     update-config(cseq)
        cseq ← finalize-config(cseq)
12: end operation
```

attempt get to the latest configuration } (1)

introduce the new configuration
migrate the data to the new config
let servers know it is good to be finalized } (2)

Read/Write Operations using DAPs/Reconfigs

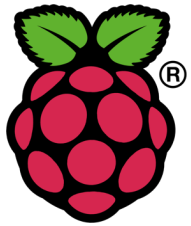
Read Protocol

- Traverse Config Sequence cseq
- Find μ = max(<c, F>) in cseq
- Set v = len(cseq)
- Discover for μ ≤ i ≤ v
  (t,v)=max(cseq[i].get-data())
- Do
  - cseq[v].put-data(t,v)
  - Traverse Sequence cseq
- while(|cseq| > v)

Write Protocol(val) (at w_i)

- Traverse Config Sequence cseq
- Find μ = max(<c, F>) in cseq
- Set v = len(cseq)
- Discover for μ ≤ i ≤ v
  t_{max}=max(cseq[i].get-tag())
- (t,v)= (<t_{max}+1,w_i>, val)
- Do
  - cseq[v].put-data(t,v)
  - Traverse Sequence cseq
- while(|cseq| > v)

Experimental Evaluation - Goals



- Examine the performance of the protocol in:
  - CLOUDLAB
  - Small Computing Devices: RPis W Zero 2

CloudLab

- Compare with similar approaches


JAEGER

- Trace Bottlenecks
  - Distributed Tracing: OpenTelemetry + Jaeger

Algorithm Comparison - CloudLab

- **MWMR ABD:** Baseline Non-Reconfigurable DSM Algorithm
- **ABD/EC ARES:** Two different versions of ARES
- **Cassandra**
  - Cassandra parameters: RF=*n*, CL=majority (*n*/2 +1)
  - Read/Write requests: Cassandra-driver Python library
- **Redis**
  - Redis WAIT command: a majority (*n*/2 + 1) of waiting write acks
  - Read/Write requests: Redis-driver Python library

Experimental Testbed

- **Experimental Deployment**
  - Fed4FIRE+: a federation of testbeds
  - **39 nodes** from 4 different testbeds in the EU and the US.
    - Grid5000 (France), InstaGeni (NYU, UCLA, Utdallas), Virtual Wall (Belgium), CloudLab(Utah)
  - Each server is deployed on a different machine (picking 1 from EU and 2 from US until we meet the server bound) .
  - Clients are all deployed in the remaining machines in a round robin fashion.

- **Performance Metric**
  - Average Operation latency of all clients (**Communication + Computation Overhead**).

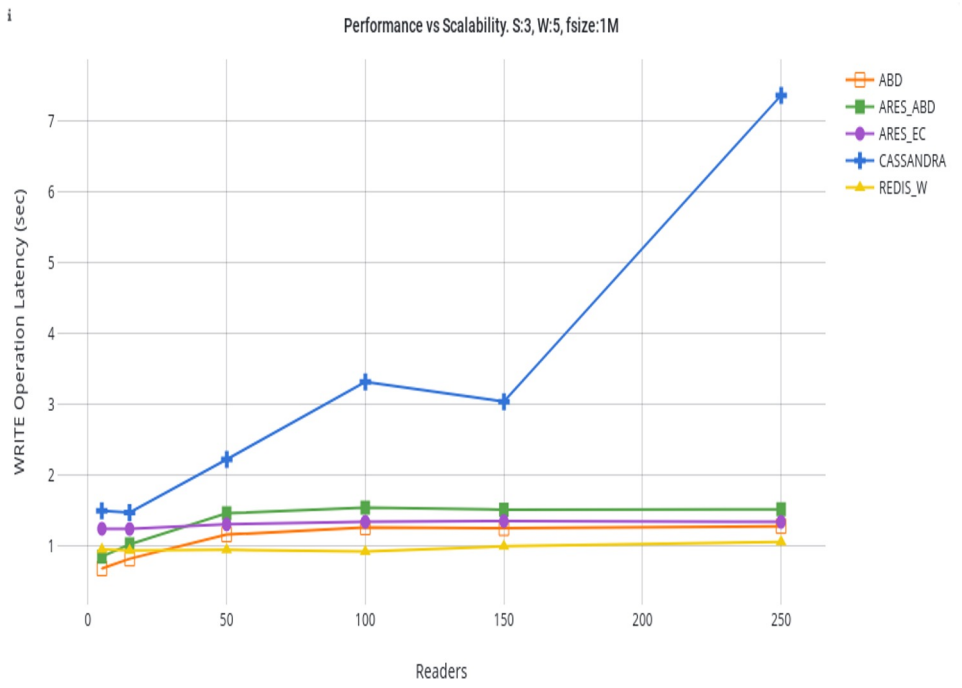
Experimental Scenarios

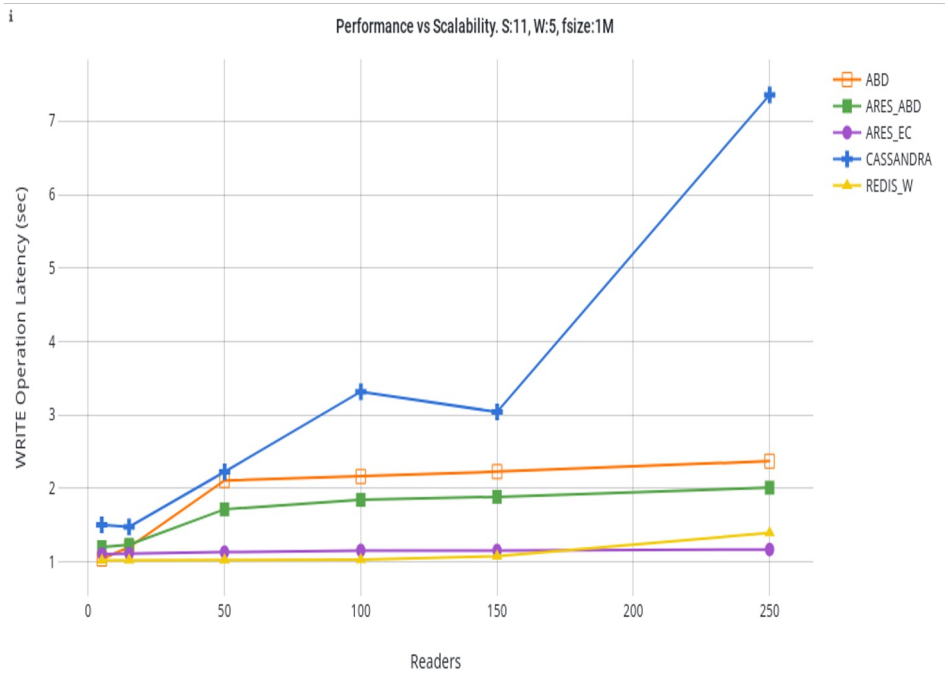
- **Scalability**
  - Variable number of participants

- **Stress Test**
  - Various Loads
  - Erasure Code Fragmentation Factor k

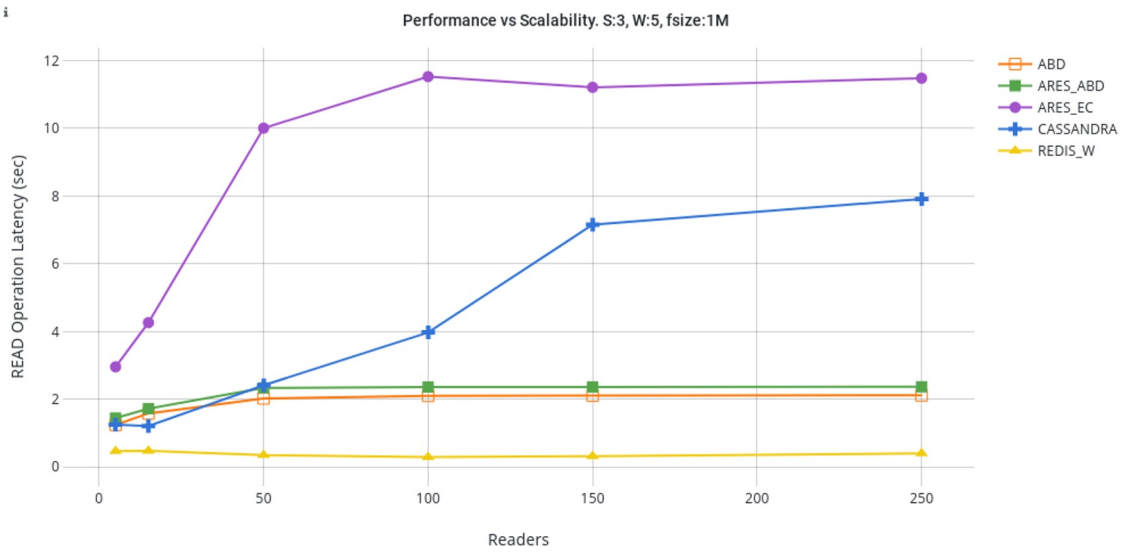
- **Fault Tolerance**
  - Liveness of the service under failures

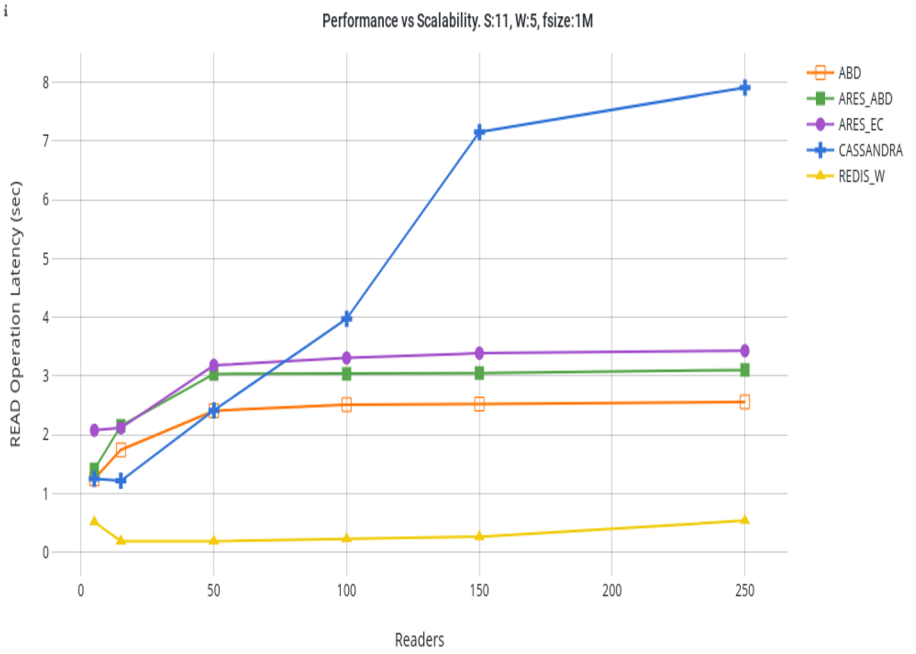
Results: Scalability Tests



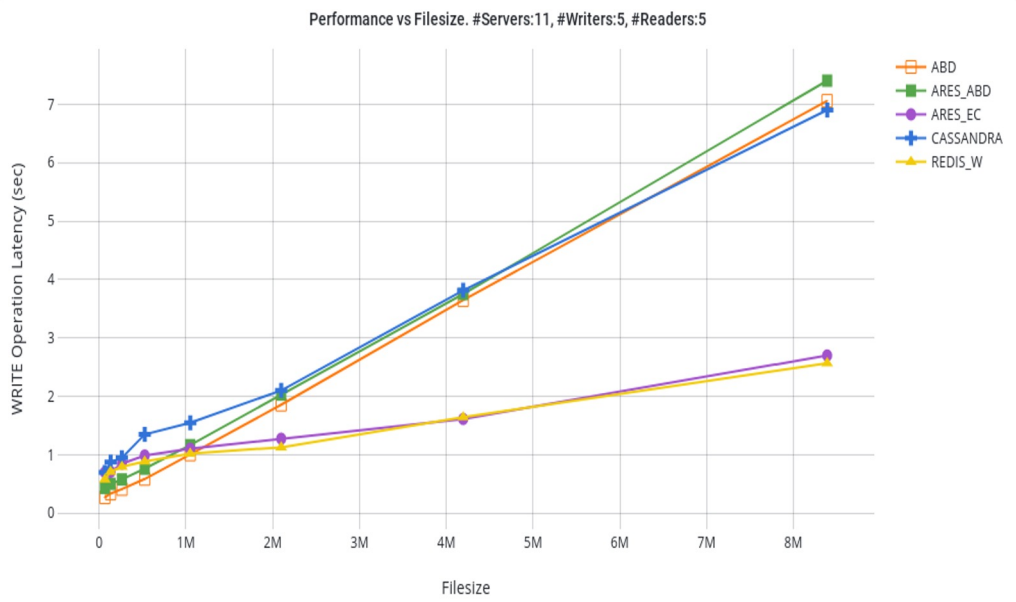


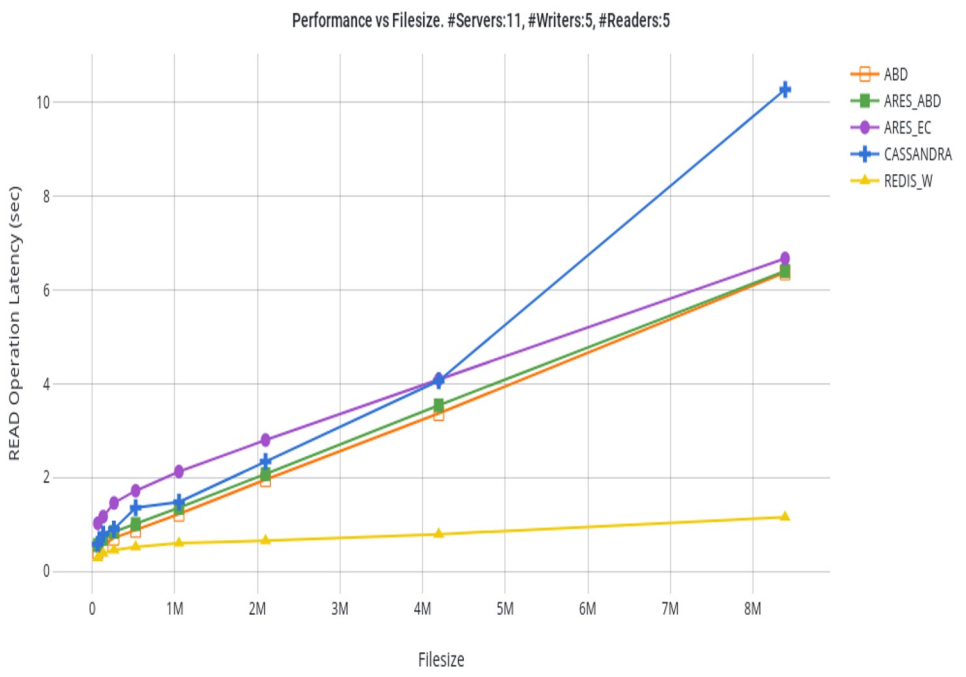
Results: Scalability Tests



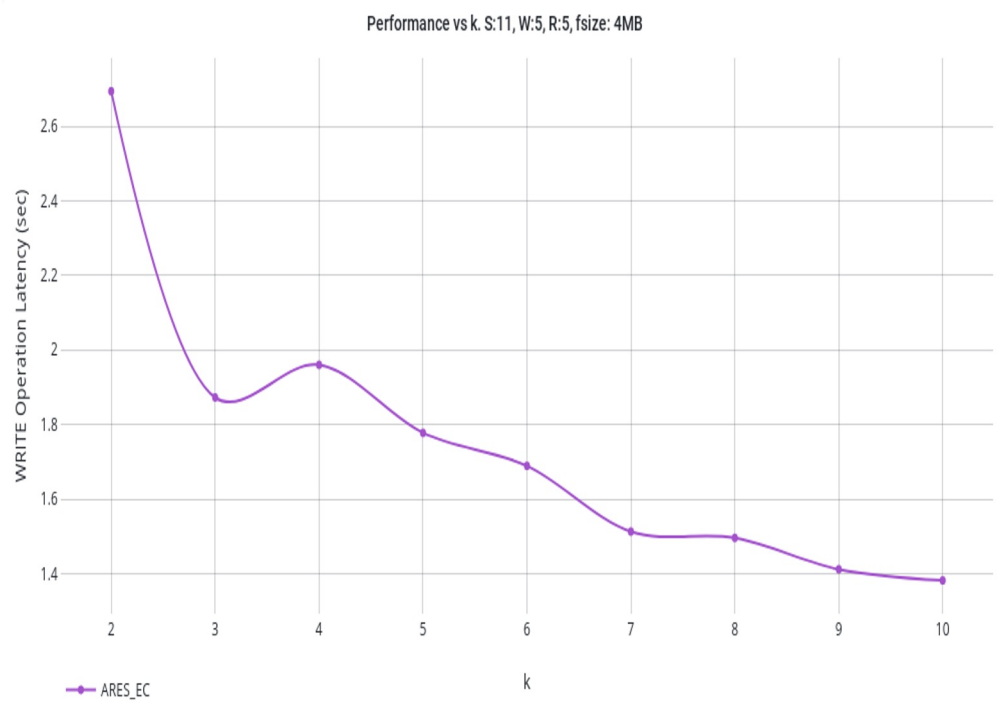


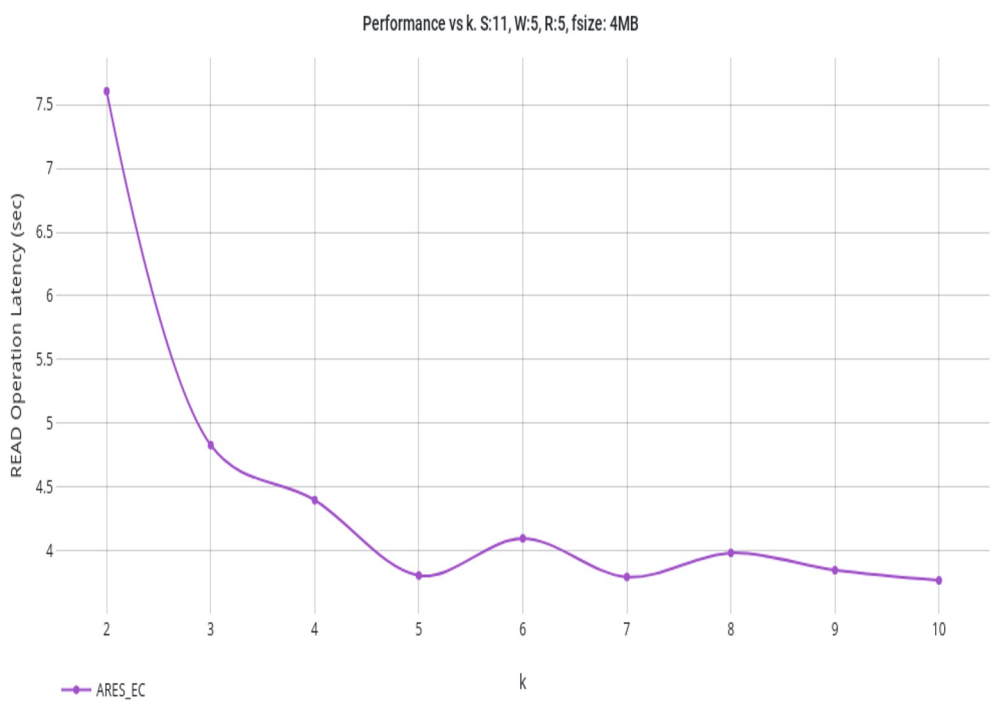
Results: Various Loads



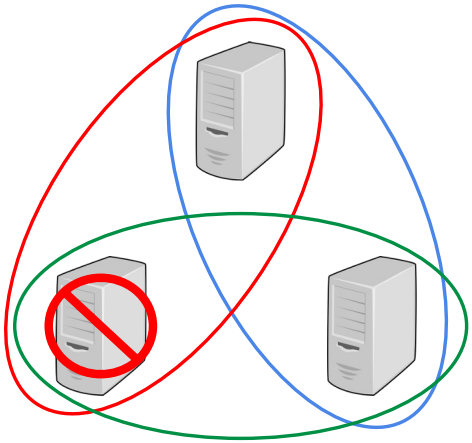


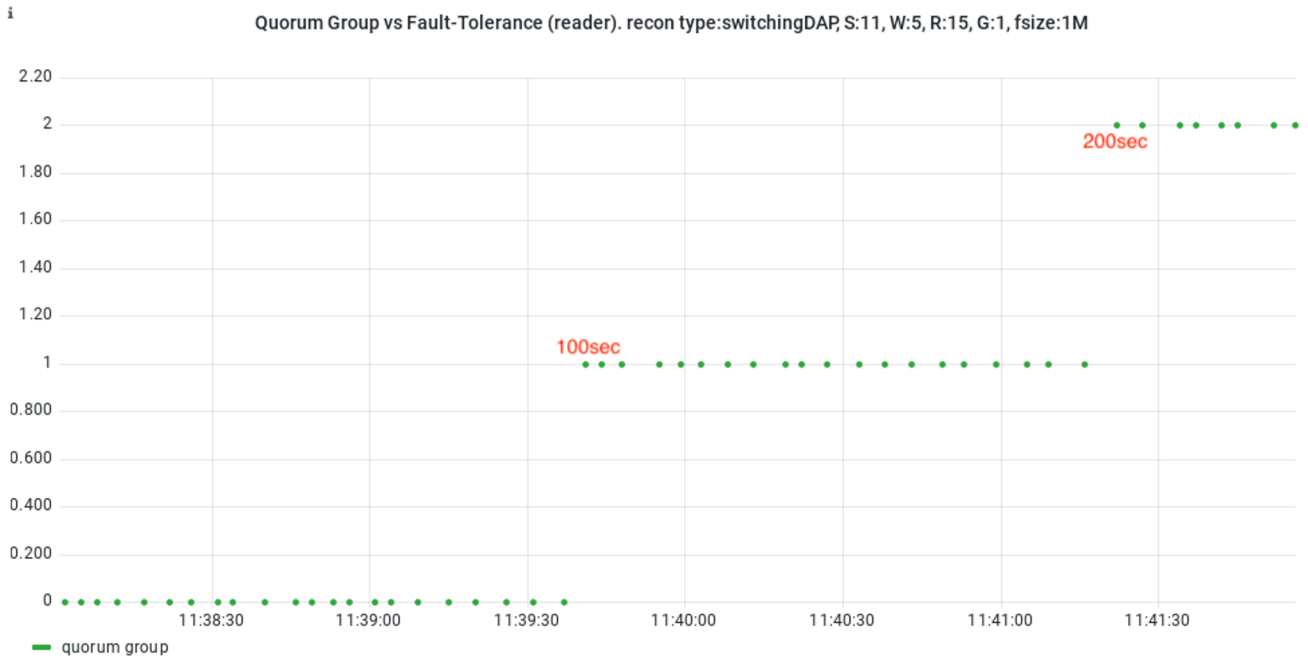
Results: Variable k





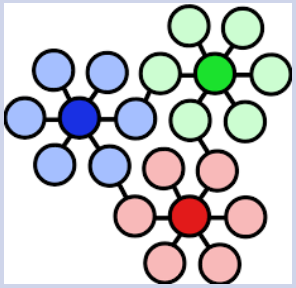
Results: Liveness

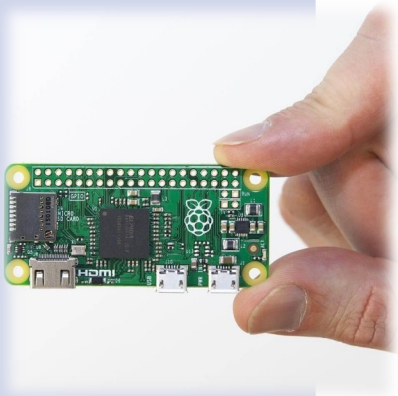




Distributed Tracing: Testbeds

Emulab	RPIs Zero 2
Processors: two 2.4 GHz 64-bit 8-Core Intel Xeon E5-2630 "Haswell" processors **RAM:** 64 GB **Network Interfaces:** 1 GB (Gigabit) and 10 GB (10 Gigabit) network interface cards (NICs) **Nodes**: 3 **Ping time:** Mean: 0.87 ms Max: 2.11 ms Min: 0.72 ms Std: 0.87 ms	**Processor:** quad-core 64-bit ARM Cortex-A53 processor clocked at 1GHz and 512MB of SDRAM **RAM:** 512 MB **Network Interfaces**: Wi-Fi 2.4GHz and 5GHz **Nodes**: 3 **Ping time:** Mean: 12.631304548995104 ms Max: 93.03711401298642 ms Min: 5.770321004092693 ms Std: 12.631304548995104 ms

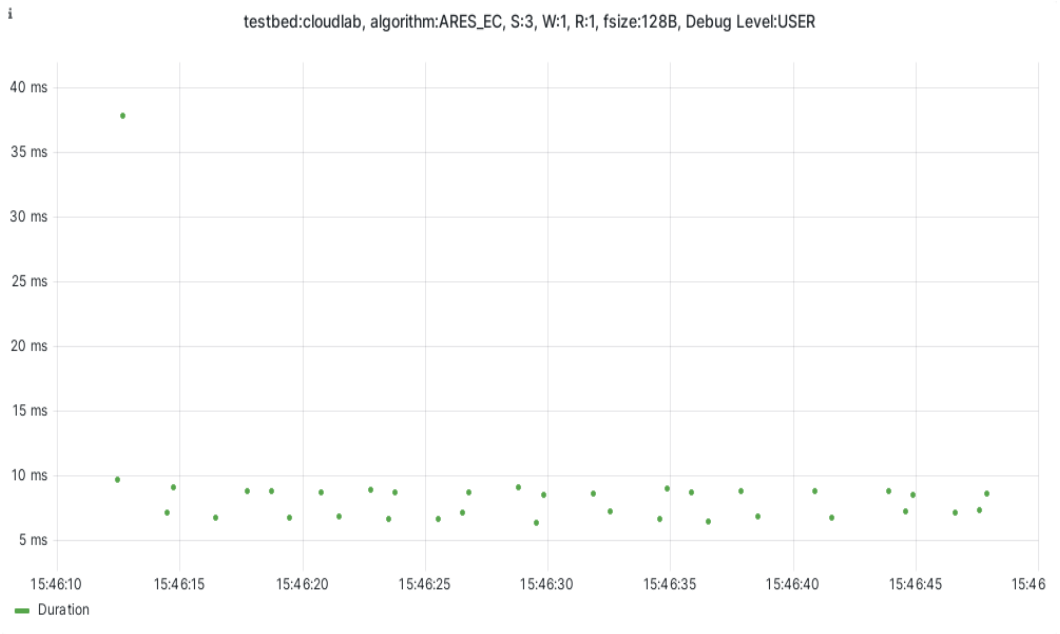




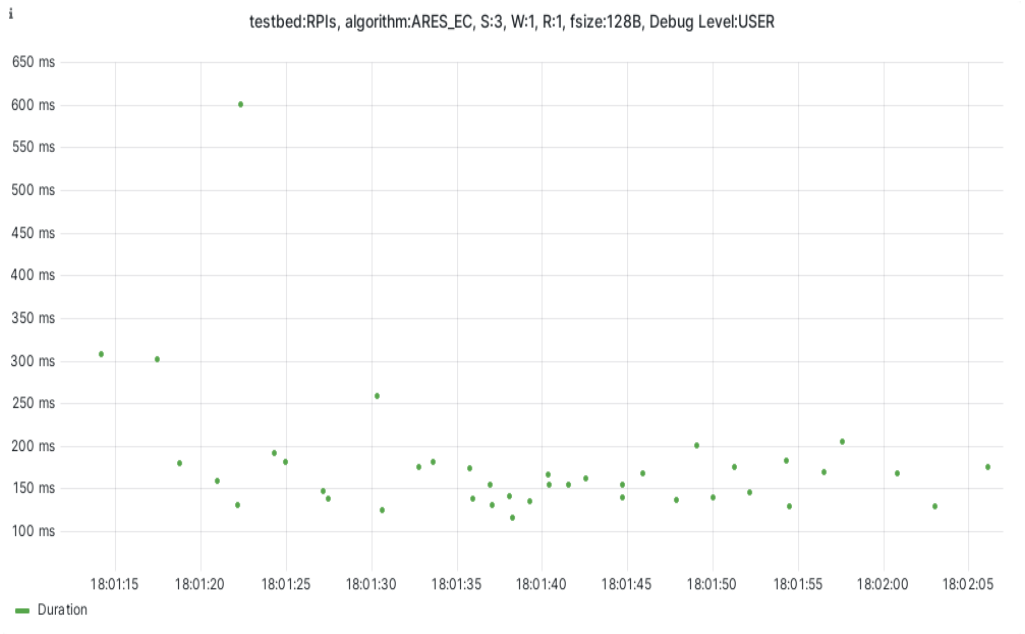
Experimental Scenarios

#Scenarios	Algorithm	#Servers	#Writers	#Readers	#writes	#reads	[1..wInt]	[1..rInt]	object size
1-2	ARES_EC ARES_ABD	3	1	1	20	20	3	3	128 B 4 MB

Scenario 1 Results (ARES_EC, 128B)

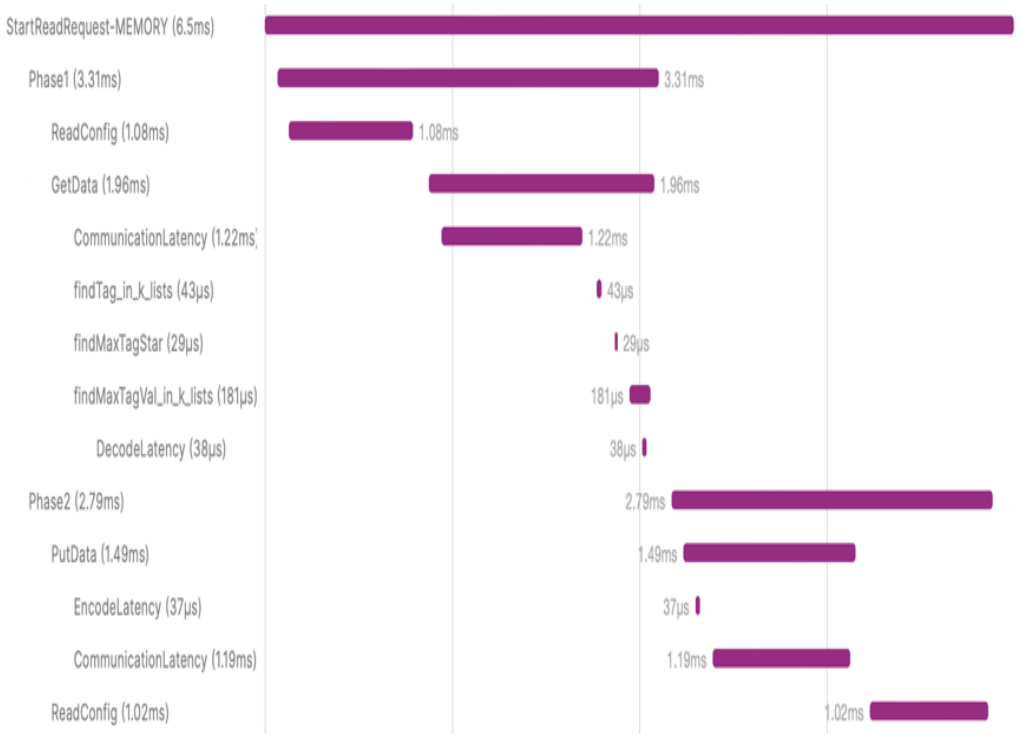


time-series, testbed:cloudlab, algorithm:ARES_EC, S:3, W:1, R:1, fsize:128B

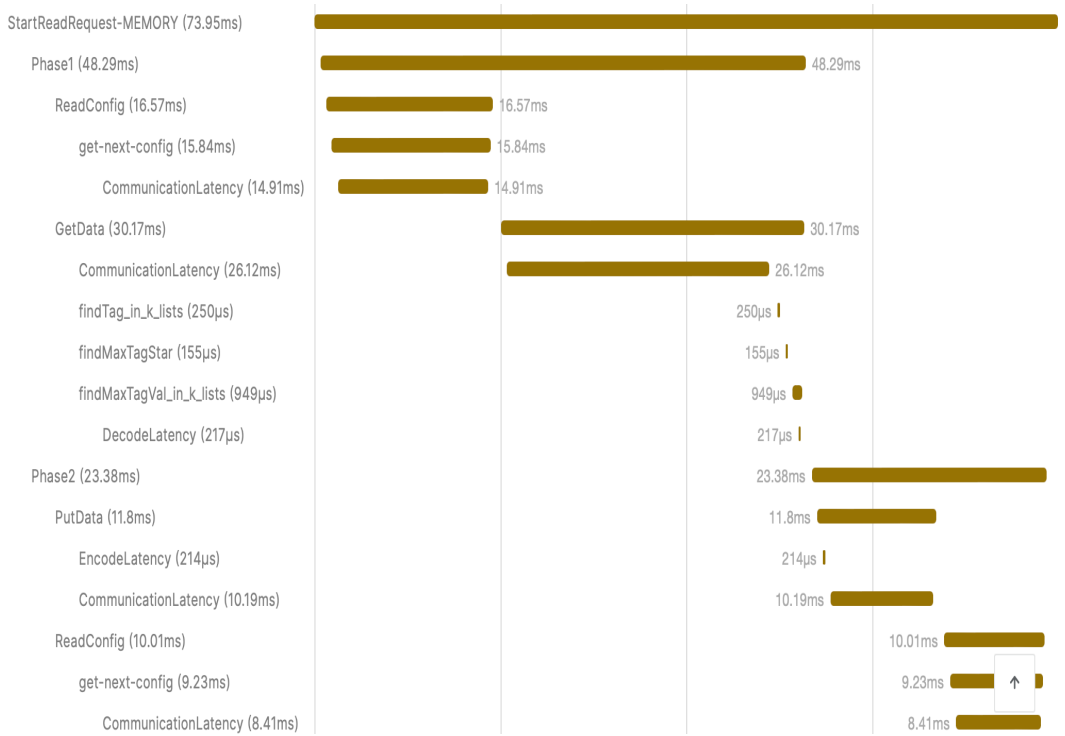


time-series, testbed:RPIs, algorithm:ARES_EC, S:3, W:1, R:1, fsize:128B

Scenario 1 Results (ARES_EC, 128B)

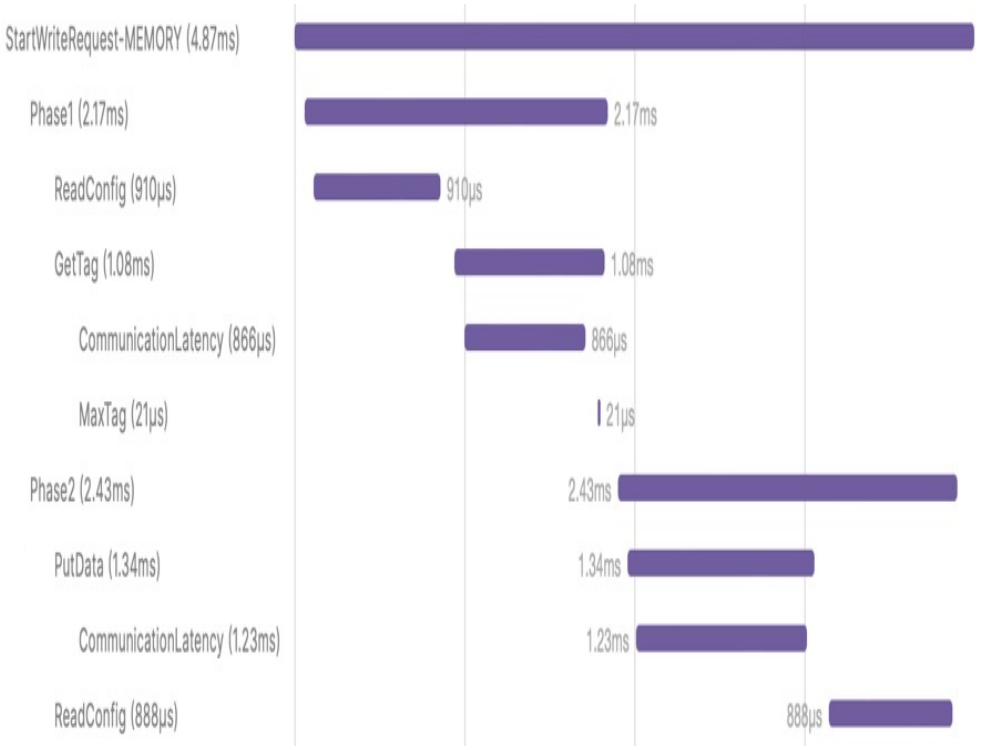


READ, testbed:cloudlab, algorithm: ARES _EC, S:3, W:1, R:1, fsize:128B

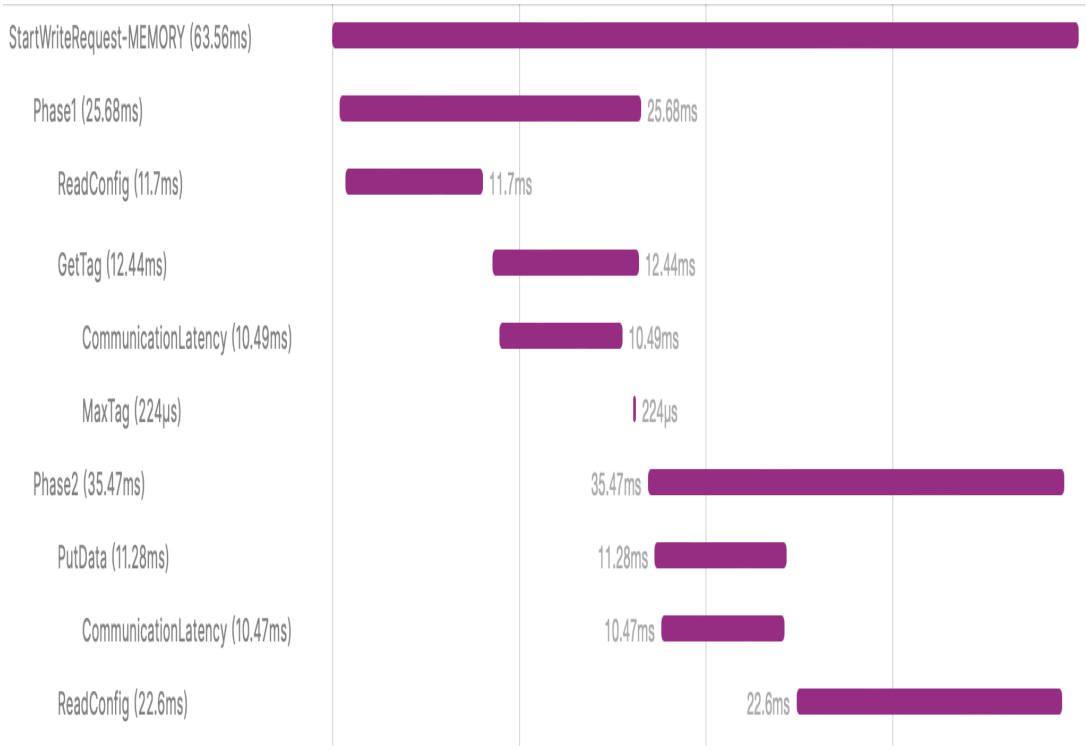


READ, testbed: RPIs, algorithm: ARES _EC, S:3, W:1, R:1, fsize:128B

Scenario 2 Results (ARES_ABD, 128B)

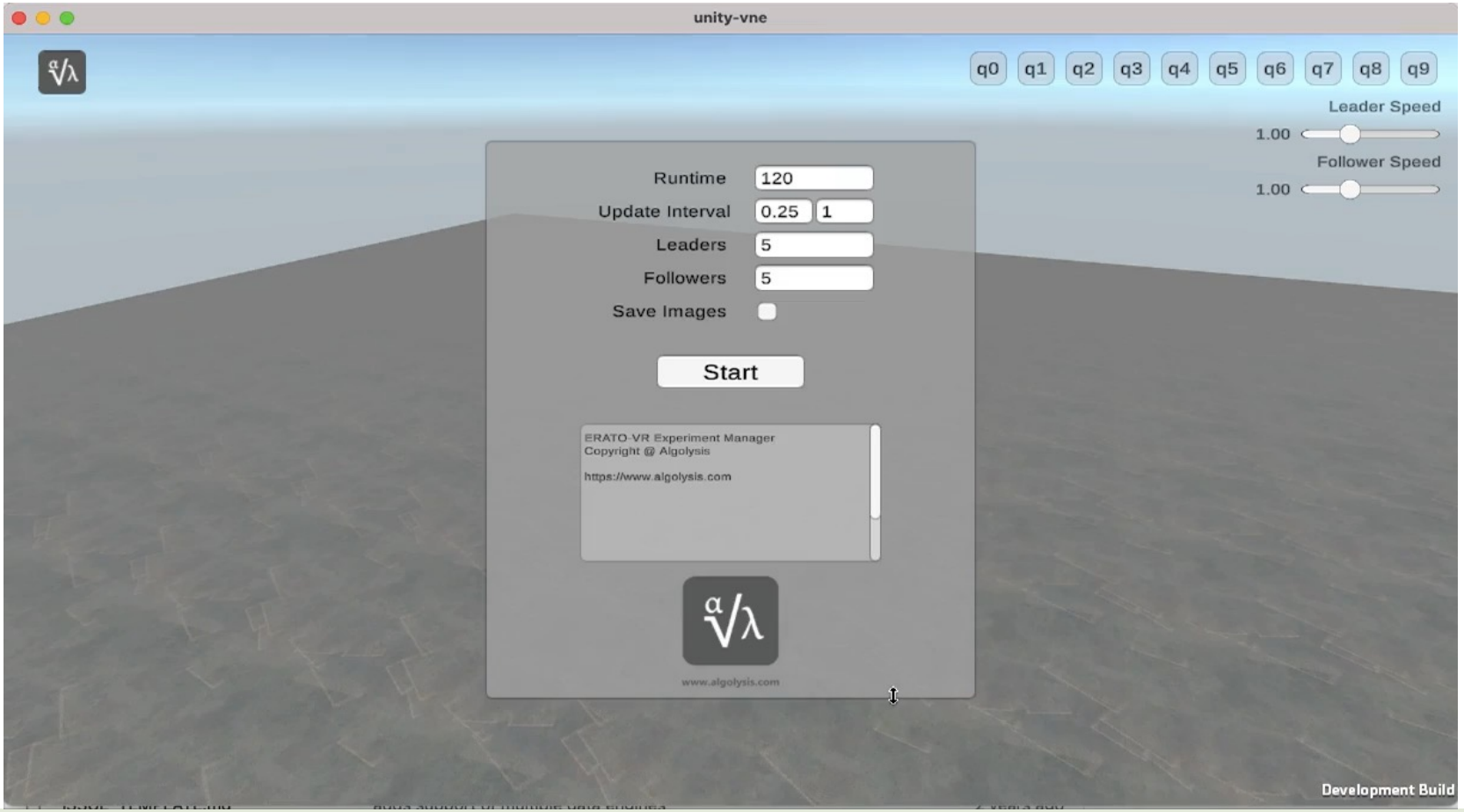


READ, testbed:cloudlab, algorithm: ARES _ABD, S:3, W:1, R:1, fsize:4MB, Debug Level:DSMM



READ, testbed: RPIs, algorithm: ARES _ABD, S:3, W:1, R:1, fsize:128B, Debug Level:DSMM

Evaluation of DSM in Virtual Environment



Thank you!

contact@algolysis.com

Funded by the European Union
NextGenerationEU

Cyprus tomorrow
RECOVERY AND RESILIENCE PLAN

Republic of Cyprus

RESEARCH & INNOVATION FOUNDATION

algolysis
algorithmic solutions

CYENS
CENTRE OF EXCELLENCE

CYPRUS MARINE & MARITIME INSTITUTE