AFRL-AFOSR-VA-TR-2019-0096

Network Coding for Strong Consistency Semantics in Distributed Shared Memory Networks

Nancy Lynch
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

12/14/2018
Final Report

DISTRIBUTION A: Distribution approved for public release.

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

DISTRIBUTION A: Distribution approved for public release

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Executive Services, Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
16-04-2019	Final Performance	30 Sep 2017 to 29 Sep 2018

4. TITLE AND SUBTITLE
Network Coding for Strong Consistency Semantics in Distributed Shared Memory Networks

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA9550-14-1-0403

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)
Nancy Lynch

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
77 MASSACHUSETTS AVE
CAMBRIDGE, MA 02139-4301 US

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AF Office of Scientific Research
875 N. Randolph St. Room 3112
Arlington, VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/AFOSR RTA2

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
AFRL-AFOSR-VA-TR-2019-0096

12. DISTRIBUTION/AVAILABILITY STATEMENT
A DISTRIBUTION UNLIMITED: PB Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The abstraction of shared memory is a powerful tool that can greatly simplify design and implementation of software systems for distributed
settings. The key property of a shared memory abstraction is the consistency guarantees that it provides under concurrent access. The most intuitive consistency model is atomicity since its semantics closely reassemble these of a sequential register. Emulating shared atomic memory proves to be challenging especially if this is to be done in a distributed message passing setting with unreliable components, as is often the case in networked systems. To this end standard techniques and solutions have been developed, however these have high communication and storage costs. In this project, we explore several algorithms for efficient implementations of shared memory services that use erasure-codes. We also derive provable strong consistency and liveness properties of these algorithms in the failure-prone setting. Novelty of this work is the fusion of standard replication techniques with network coding techniques. Shared memory services for distributed systems emulate a memory abstraction providing users with a consistent view of its content under concurrent access, with a best effort to ensure availability and survivability.

15. SUBJECT TERMS
Network Coding

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			NGUYEN, TRISTAN

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

DISTRIBUTION A: Distribution approved for public release

Unclassified	Unclassified	Unclassified	UU	**PAGES**	**19b. TELEPHONE NUMBER** *(Include area code)* 703-696-7796

DISTRIBUTION A: Distribution approved for public release

Final Report for AFOSR Contract FA9550-14-1-0403: Network Coding for Strong Consistency Semantics in Distributed Shared Memory Networks

1 Basic Information

Principal Investigator: Nancy Lynch
Primary Contact Email: lynch@csail.mit.edu
Primary Contact Phone Number: 617-253-7225
Organization Name: MIT
Program Manager: Tristan Nguyen
Reporting Period start date: 09/30/2014
Reporting Period end date: 09/29/2018

Summary: Our project involved developing techniques for implementing atomic (linearizable) data storage on distributed servers, using coding techniques.

2 Research report

Continuation of earlier work: Our first work in this area was [1] below, which was completed at around the starting time of this contract. This paper develops the CAS and CASGC algorithms for implementing atomic memory efficiently on data servers using erasure coding methods. Subsequently, we wrote a complete, improved version for journal publication [2]. This paper appeared in the journal Distributed Computing during this reporting period.

[1] Viveck Cadambe, Nancy Lynch, Muriel Medard, and Peter Musial. A Coded Shared Atomic Memory Algorithm for Message Passing Architectures. IEEE International Symposium on Network Computing and Applications (NCA14), 2014. Best Paper Award.

[2] Viveck R. Cadambe, Nancy Lynch, Muriel Medard, and Peter Musial, A Coded Shared Atomic Memory Algorithm for Message Passing Architectures. Distributed Computing 30(1): 49-73, February, 2017.

Abstract: This paper considers the communication and storage costs of emulating atomic (linearizable) multi-writer multi-reader shared memory in distributed message-passing systems. The paper contains three main contributions:

1. We present an atomic shared-memory emulation algorithm that we call Coded Atomic Storage (CAS). This algorithm uses erasure coding methods. In a storage system with N servers that is resilient to f server failures, we show that the communication cost of CAS is $\frac{N}{N-2f}$. The storage cost of CAS is unbounded.

DISTRIBUTION A: Distribution approved for public release

2. We present a modification of the CAS algorithm known as CAS with Garbage Collection ($CASGC$). The $CASGC$ algorithm is parameterized by an integer δ and has a bounded storage cost. We show that the $CASGC$ algorithm satisfies atomicity. In every execution of $CASGC$ where the number of server failures is no bigger than f, we show that every write operation invoked at a non-failing client terminates. We also show that in an execution of $CASGC$ with parameter δ where the number of server failures is no bigger than f, a read operation terminates provided that the number of write operations that are concurrent with the read is no bigger than δ. We explicitly characterize the storage cost of $CASGC$, and show that it has the same communication cost as CAS.

3. We describe an algorithm known as the Communication Cost Optimal Atomic Storage (CCOAS) algorithm that achieves a smaller communication cost than CAS and $CASGC$. In particular, CCOAS incurs read and write communication costs of $\frac{N}{N-f}$, measured in terms of number of object values. We also discuss drawbacks of CCOAS as compared with CAS and $CASGC$.

Viveck Cadambe and Zhiying Wang (earlier postdocs on this project, now faculty members at Penn State and U.C. Irvine, respectively) have studied techniques for efficiently storing multiple versions of a data object in a distributed storage system. They have written several papers on this topic, containing a variety of new storage techniques and matching lower bounds. During this reporting period, they produced a full arXiv version summarizing the work:

[3] Zhiying Wang and Viveck R. Cadambe. Multi-Version Coding - An Information Theoretic Perspective of Consistent Distributed Storage. Report available online at http://arxiv.org/abs/1506.00684.

Lower bounds for shared-memory emulation: In the following papers, we consider the fundamental limitations on the storage cost for implementing atomic memory (and some weaker forms of memory) in a distributed system. Our results are a series of information-theoretic lower bounds on these costs. They are proved using interesting lower bound and impossibility proof techniques that are adapted from distributed computing theory:

[4] Zhiying Wang and Viveck R. Cadambe and Nancy Lynch. Information-Theoretic Lower Bounds on the Storage Cost of Shared Memory Emulation. Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), Chicago, IL, pages 305-313, July 2016.

[5] Zhiying Wang and Viveck R. Cadambe and Nancy Lynch. Information-Theoretic Lower Bounds on the Storage Cost of Shared Memory Emulation. http://arxiv.org/abs/1605.06844 (2016). Full version of [4].

Abstract: The focus of this paper is to understand storage costs of emulating an atomic shared memory over an asynchronous, distributed message passing system. Previous literature has developed several shared memory emulation algorithms based on replication and erasure coding techniques, and analyzed the storage costs of the proposed algorithms. In this paper, we present the first known information-theoretic lower bounds on the storage costs incurred by shared memory emulation algorithms. Our storage cost lower bounds are universally applicable, that is, we make no assumption on the method of encoding the data.

DISTRIBUTION A: Distribution approved for public release

We consider an arbitrary algorithm A that implements an atomic multi-writer single-reader (MWSR) shared memory variable whose values come from a finite set V over a system of N servers connected by point-to-point asynchronous links. We require that in every fair execution of algorithm A where the number of server failures is smaller than a parameter f, every operation invoked at a non-failing client terminates. We define the storage cost of a server in algorithm A as the logarithm (to base 2) of the number of states it can take on; the total storage cost of algorithm A is the sum of the storage cost of all servers. Previously, it was known that the storage cost cannot be smaller than $\frac{N}{N-f} \log |V|$. We develop three new lower bounds on the storage cost of algorithm:

1. In our first lower bound, we show that if algorithm A does not use server gossip, then the total storage cost is lower bounded by $\frac{2N}{N-f+1} \log |V| + o(\log |V|)$.
2. In our second lower bound we show that the total storage cost is at least $\frac{2N}{N-f+2} \log |V| + o(\log |V|)$, even if the algorithm uses server gossip.
3. In our third lower bound, we consider algorithms where the write protocol sends information about the value in at most one round. For such algorithms, we show that the total storage cost is at least $\frac{\nu N}{N-f+\nu-1} \log |V| + o(\log |V|)$, where ν is the minimum of $f+1$ and the number of active write operations of an execution.

Our first and second lower bounds are approximately twice as strong as the previously known bound $\frac{N}{N-f} \log |V|$. Furthermore, our first two lower bounds apply even for regular, single-writer single-reader (SWSR) shared memory emulation algorithms. Our third lower bound is much larger than our first and second lower bounds, although it is applicable to a smaller class of algorithms where the write protocol has certain restrictions. In particular, our fourth bound is comparable to the storage cost achieved by most shared memory emulation algorithms in the literature, which naturally fall under the class of algorithms studied.

Improved algorithms for coded atomic storage: Recently, we have been working on improving upon our earlier CAS and $CASGC$ algorithms for Coded Atomic Storage. Papers [6] and [7] reduce the time costs considerably, while simplifying the algorithm. Papers [8] and [9] tackle the problem of integrating server-failure recovery into the algorithms.

[6] Kishori M Konwar, N. Prakash, Erez Kantor, Muriel Medard, Nancy Lynch, and Alexander A. Schwarzmann. Storage-Optimized Data-Atomic Algorithms for Handling Erasures and Errors in Distributed Storage Systems. 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 720-729, May 2016.

[7] Kishori M Konwar, N. Prakash, Erez Kantor, Muriel Medard, Nancy Lynch, and Alexander A. Schwarzmann. Storage-Optimized Data-Atomic Algorithms for Handling Erasures and Errors in Distributed Storage Systems. arXiv:1605.01748. Full version of [6].

Abstract: Erasure codes are increasingly being studied in the context of implementing atomic memory objects in large scale asynchronous distributed storage systems. When compared with

DISTRIBUTION A: Distribution approved for public release

the traditional replication based schemes, erasure codes have the potential of significantly lowering storage and communication costs while simultaneously guaranteeing the desired resiliency levels. In this work, we propose the Storage-Optimized Data-Atomic ($SODA$) algorithm for implementing atomic memory objects in the multi-writer multi-reader setting. $SODA$ uses Maximum Distance Separable (MDS) codes, and is specifically designed to optimize the total storage cost for a given fault-tolerance requirement. For tolerating f server crashes in an n-server system, $SODA$ uses an $[n, k]$ MDS code with $k = n - f$, and incurs a total storage cost of $\frac{n}{n-f}$. $SODA$ is designed under the assumption of reliable point-to-point communication channels. The communication cost of a write and a read operation are respectively given by $O(f^2)$ and $\frac{(d+1)n}{n-f}$, where d denotes the number of writes that are concurrent with the particular read. In comparison with the recent $CASGC$ algorithm, which also uses MDS codes, $SODA$ offers lower storage cost while pays more on the communication cost.

We also present a modification of $SODA$, called $SODA_{err}$, to handle the case where some of the servers can return erroneous coded elements during a read operation. Specifically, in order to tolerate f server failures and e error-prone coded elements, the $SODA_{err}$ algorithm uses an $[n, k]$ MDS code such that $k = n - 2e - f$. $SODA_{err}$ also guarantees liveness and atomicity, while maintaining an optimized total storage cost of $\frac{n}{n-f-2e}$.

[8] Kishori Konwar, Nancy Lynch, Muriel Medard and Prakash Narayana Moorthy. RADON: Repairable Atomic Data Object in Networks. Proceedings of the 20th International Conference on Principles of Distributed Systems (OPODIS 2016), Madrid, Spain, December 2016.

[9] Kishori Konwar, Nancy Lynch, Muriel Medard and Prakash Narayana Moorthy. RADON: Repairable Atomic Data Object in Networks. arXiv:1605.05717. Full version of [8].

Abstract: In this paper, we provide fault-tolerant algorithms, for implementing atomic memory service in asynchronous network of storage nodes, having the ability to perform background repair of crashed nodes, thereby increasing the durability of the storage service. A node that crashes is assumed to lose all its data, both from the volatile memory as well as the stable storage. A repair operation of a node in the crashed state is triggered externally, and is carried out by concerned node via message exchanges with other active nodes in the system. Upon completion of repair, the node reenters active state, and resumes participation in ongoing/future read, write and repair operations.

We argue that under arbitrary conditions where there is no restriction on the number of repair processes being performed in relation to reads and writes, it is not possible to simultaneously achieve liveness and atomicity. Therefore, we introduce two network "stability" conditions, N1 and N2, which are mostly likely to be respected by practical storage networks. Next, we design the RADONL algorithm which guarantees liveness and safety as long as condition N1 holds in the network. However, the algorithm may violate safety of an execution if N1 is not observed by the network. Next, under the assumption of the slightly stronger network condition N2, we design the algorithm RADONS, which guarantees atomicity of any execution, but liveness is guaranteed only if N2 holds. The guarantee of safety in RADONS comes at the cost of adding an additional phase in the read and write operations, compared to RADONL. Both RADONL and RADONS use replication of data for fault tolerance.

DISTRIBUTION A: Distribution approved for public release

Further, we provide a third algorithm, called RADONC, that is based on erasure codes. RADONC guarantees liveness and atomicity under the assumption of N1, and significantly improves upon the storage and communication costs of RADONL and RADONS, under scenarios when the number of write operations concurrent with a read or a repair operation is bounded.

Finally, in our most recent work, we have developed a novel two-layer erasure-coded fault-tolerant distributed storage system, LDS, which offers atomic access for read and write operations. The front-end layer provides low-latency access and temporary storage for client operations, and uses the back-end layer for persistent storage. This flexible architecture allows for separate implementations of atomicity protocols and and erasure-code protocols. This architecture is particularly suitable in modern edge computing environments, such as Internet-of-Things.

[10] Kishori M Konwar, N. Prakash, Muriel Medard and Nancy Lynch, A Layered Architecture for Erasure-Coded Consistent Distributed Storage. Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), Washington D.C., July, 2017.

[11] Kishori M Konwar, N. Prakash, Muriel Medard and Nancy Lynch, A Layered Architecture for Erasure-Coded Consistent Distributed Storage. arXiv:1703.01286. Full version of [10].

Abstract: Motivated by emerging applications to the edge computing paradigm, we introduce a two-layer erasure-coded fault-tolerant distributed storage system offering atomic access for read and write operations. In edge computing, clients interact with an edge-layer of servers that is geographically near; the edge-layer in turn interacts with a back-end layer of servers. The edge-layer provides low latency access and temporary storage for client operations, and uses the back-end layer for persistent storage. Our algorithm, termed Layered Data Storage (LDS) algorithm, offers several features suitable for edge-computing systems, works under asynchronous message-passing environments, supports multiple readers and writers, and can tolerate $f_1 < \frac{n_1}{2}$ and $f_2 < \frac{n_2}{3}$ crash failures in the two layers having n_1 and n_2 servers, respectively.

We use a class of erasure codes known as regenerating codes for storage of data in the back-end layer. The choice of regenerating codes, instead of popular choices like Reed-Solomon codes, not only optimizes the cost of back-end storage, but also helps in optimizing communication cost of read operations, when the value needs to be recreated all the way from the back-end. The two-layer architecture permits a modular implementation of atomicity and erasure-code protocols; the implementation of erasure-codes is mostly limited to interaction between the two layers.

We prove liveness and atomicity of LDS, and also compute performance costs associated with read and write operations. In a system with $n_1 = \Theta(n_2)$, $f_1 = \Theta(n_1)$, $f_2 = \Theta(n_2)$, the write and read costs are respectively given by $\Theta(n_1)$ and $\Theta(1) + n_1\mathcal{I}(\delta > 0)$. Here δ is a parameter closely related to the number of write operations that are concurrent with the read operation. The cost of persistent storage in the back-end layer is $\Theta(1)$. The impact of temporary storage is minimally felt in a multi-object system running N independent instances of LDS, where only a small fraction of the objects undergo concurrent accesses at any point during the execution. For the multi-object systems, we identify a condition on the rate of concurrent writes in the system such that the overall storage cost is dominated by that of persistent storage in the back-end layer, and is given by $\Theta(N)$.

DISTRIBUTION A: Distribution approved for public release

Coding-based data management in the presence of uncertainty and change: In volatile network environments, node connectivity and availability changes rapidly. This poses a challenge to efficient repair of failed nodes in a network distributed storage: traditional erasure-correcting and regenerating codes perform poorly, because they rely on access to a sufficiently large number of surviving nodes, and/or specific subsets of them. The following papers are devoted to potential solutions to problems arising in the above settings.

In [12], we consider a stochastic model for network storage, motivated by such scenarios. Under this model, we analyze an RLNC-based regenerating coding scheme, and show that with high probability it performs surprisingly well in repairing the coded data in such dynamic environments.

[12] Abdrashitov, Vitaly and Medard, Muriel, Durable network coded distributed storage, Proceedings of 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), p. 851856, 2015.

We further study this coding scheme in [13] and demonstrate its efficiency in preventing data loss for a wide range of system parameters. In particular, the scheme can perform well even with relatively low field size (order of 17). In [13] we study a generalization of the setting of regenerating codes, motivated by applications to storage systems consisting of clusters of storage nodes. This scenario can happen when the user data is spread across distinct geographically separated data-centers of a cloud-service provider. Alternatively, this can be the case of user-defined cloud-of-clouds, when the user data is spread across data centers corresponding to multiple cloud service providers in order to improve high availability in a local geographic zone, flexibility to avoid vendor lock-in, and data security.

[13] Abdrashitov, Vitaly and Medard, Muriel, Staying Alive – network coding for data persistence in volatile networks, Signals, Systems and Computers, 2016 Asilomar Conference on, 2016.

In [14] we model a data file as coded and stored across mn nodes on n clusters, with m nodes per cluster. Nodes represent entities that can fail. Node repair is accomplished by downloading data from both other clusters, and the surviving nodes in the same cluster. We identity the optimal trade-off between storage-overhead and inter-cluster (IC) repair-bandwidth, and present optimal code constructions for a class of parameters. Our results imply that it is possible to simultaneously achieve both optimal storage overhead and optimal minimum IC bandwidth, for sufficiently large values of nodes per cluster. The simultaneous optimality comes at the expense of intra-cluster bandwidth, and we obtain lower bounds on the necessary intra-cluster repair-bandwidth.

[14] N. Prakash and Vitaly Abdrashitov and Muriel Medard, A Generalization of Regenerating Codes for Clustered Storage Systems, Proceedings of the 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2016.

In [15] we consider a communication problem in which an update of the source message needs to be conveyed to one or more distant receivers that are interested in maintaining specific linear functions of the source message. The setting is one in which the updates are sparse in nature, and where neither the source nor the receiver(s) is aware of the exact difference vector, but only know the amount of sparsity that is present in the difference-vector. Under this setting, we are interested in devising linear encoding and decoding schemes that minimize the communication cost involved.

DISTRIBUTION A: Distribution approved for public release

We show that the optimal solution to this problem is closely related to the notion of maximally recoverable codes (MRCs), which were originally introduced in the context of coding for storage systems. In the context of storage, MRCs guarantee optimal erasure protection when the system is partially constrained to have local parity relations among the storage nodes. In our problem, we show that optimal solutions exist if and only if MRCs of certain kind (identified by the desired linear functions) exist. We consider point-to-point and broadcast versions of the problem, and identify connections to MRCs under both these settings.

[15] N. Prakash and Vitaly Abdrashitov and Muriel Medard, Communication Cost for Updating Linear Functions when Message Updates are Sparse: Connections to Maximally Recoverable Codes, (arXiv:1605.01105) https://arxiv.org/abs/1605.01105

ARES: Adaptive, Reconfigurable, Erasure coded, atomic Storage: Implementing *atomic*, shared, read/write storage systems is a fundamental problem in distributed computing. Implementations that utilize erasure-codes, have recently been introduced to reduce the storage, communication, and latency costs compared to the traditional replication approaches. Such algorithms require that the collection of storage hosts is known a priori and remains the same within the execution of the algorithm. However, to ensure survivability and scalability, a storage service should be able to dynamically mask hosts failures, joins, and removals without any service interruptions.

In [16] we examine *dynamic* atomic storage algorithms that use erasure-codes and allow the collection of storage hosts to change in the course of an execution. We begin by presenting a generic algorithmic structure for atomic storage algorithms, using three data access primitives (DAP): (i) GetTag, (ii) PutData, and (iii) GetData. A number of tag-based algorithms can be converted to the proposed form. We define the properties that DAPs must satisfy in order for the converted algorithms to preserve atomicity. By utilizing DAPs, we then present ARES, that implements a reconfigurable, atomic storage service in the message passing environment. The usage of DAPs allows ARES to be oblivious to the mechanics of the underlying atomic storage algorithm giving ARES two main advantages over previous dynamic solutions: (i) it can use any logical timestamp-based atomic algorithm designed for the static environment and able to be expressed with the DAPs, and (ii) it can be adaptive, namely it can deploy a different algorithm per configuration without affecting correctness. To demonstrate the use of ARES, we propose a new *two-round* erasure-code based algorithm for emulating multi-writer, multi-reader (MWMR) atomic objects in asynchronous, message-passing environments, with near-optimal communication and storage costs. The algorithm is expressed using the proposed primitives and can be directly used by ARES, giving rise to the first reconfigurable erasure-coded atomic storage.

[16] Viveck Cadambe, Nicolas Nicolaou, Kishori M. Konwar, N. Prakash, Nancy Lynch, Muriel Medard ARES: Adaptive, Reconfigurable, Erasure coded, atomic Storage, (arXiv:1805.03727) https://arxiv.org/abs/1805.03727

FLECKS: Fast Lean Erasure-coded Consistent Key-value Store In [17], we present FLECKS, a strongly consistent key-value (KV) store that simultaneously improves storage and

DISTRIBUTION A: Distribution approved for public release

communication costs, using erasure codes, while maintaining same tolerance levels as replication for practical regimes of interest. FLECKS supersedes the combined advantages of other erasure-code based strongly consistent distributed storage protocols, SODA [6] and CAS-GC [1]. In addition, FLECKS captures several other properties desirable in practical use: low network bandwidth usage, low number of rounds per operation, efficient garbage collection of successful operations, etc. We implement FLECKS and compare with standard replication based leaderless algorithm (called the ABD algorithm). Our implementation is faithful to our algorithmic steps, and hence does not rely on external or ad-hoc mechanisms so that the unimportant tweaks do not obscure the essential ingredients for performance gain in our approach. We show that relaying messages is a powerful technique that helps the erasure coded system operate at the same fault-tolerance level of replicated system, while ensuring liveness of GET operations.

Algorithm design perspectives. FLECKS is designed with practical considerations in mind therefore, FLECKS embodies many desirable attributes of an effective distributed storage system such as, ability to tolerate client and server failures; graceful handling of asynchrony, without assuming common clock or failure detector or using timeouts etc; low communication rounds during read (GET) and write (PUT) operations; highly available and near-optimal storage and communication costs. FLECKS can handle any number of concurrent PUTs and GETs on the stored data. Strong consistency and availability properties of FLECKS are analytically proven.

Implementation. We implemented a software stack that can deploy the following atomicity algorithms on OpenStack and Amazon AWS cloud platforms: (a) an optimized version of the ABD algorithm, which is a typical quorum based replication scheme for implementing atomicity ; and (d) FLECKS algorithm which uses erasure-codes algorithm. For each of these algorithms, multiple atomic objects (so far, tested up to 10,000 objects) can be implemented with a set of servers (tested up to 60 servers) and tens of readers and writers issuing read and writes requests. The overall implementation consists of 4 main components:

1. *Protocol Implementation* The core implementation of the above ABD and FLECKS algorithms implemented in golang. A C based-based socket library called ZMQ is used for point-to-point reliable communication among the processes.

2. *Orchestration Tools* The orchestration component is used to control the processes for the algorithms such as, deploy the algorithm, start or stop experiment, gather experimental data, deliberately crash, etc;

3. *Consistency Checker* Though the protocol is provably strongly consistent in theory, we wanted to ensure that our implementation also provides this guarantee. Validating strong consistency of an execution requires precise clock synchronization across all processes, so that one can track operations with respect to a global time. This is impossible to achieve in a distributed system where clock drift is inevitable. To circumvent this, we deploy all the docker containers in a single beefy machine so that every process observers the same clock running in the VM.

   Our checker gathers meta-data regarding an execution, and this data includes start and end times of all the operations, as well as other parameters like logical timestamps used by

DISTRIBUTION A: Distribution approved for public release

the protocol. The checker logic is based on the conditions appearing in Lemma 13.16 (in Distributed Algorithms by Nancy Lynch), which provide a set of sufficient conditions for guaranteeing strong consistency. The checker validates strong consistency property for ever KV pair individually for the execution under consideration.

4. *Workload Generator.* We run a separate VM that communicates with the client VMs, and acts as the workload generator. Standard off-the shelf workload generators like YCSB (by Yahoo) tests with a single writer/single reader, whereas FLECKS is designed to support multi-reader multi-writer. We view readers and writers as proxy clients (like coordinators in Cassandra, a popular KV store), and our workload generation is aimed at testing the proxy clients as well. The workload generator simply triggers a batch of operations to be run by each proxy client in the system. For example, the workload generator sends a message to writer 1 to fire 50000 sequential PUTs, with an inter-operation time of 1 ms. The writer for each PUT picks the KV pair for that operation either based on a uniform or zipfian distribution on the set of stored KV pairs. By controlling the number of reader and writer clients, and the inter-operation time, we create executions for a wide range of the PUT-GET ratios for test purposes.

Experimental Validation. We experimentally studied FLECKS under a wide range of scenarios and compare with a quorum based replicated system. Our implementation closely resembles the algorithm specification of FLECKS and hence does not rely on additional distributed computing services or primitives like leader-election or consensus protocol to achieve coordination and synchronization. Our experiments show that FLECKS achieves substantially lower storage and bandwidth costs and has a significantly lower latency of operations than the replication-based mechanisms.

[17] N. Prakash, Kishori M. Konwar, Muriel Médard and Nancy Lynch, FLECKS: Fast Lean Erasure-coded Consistent Key-value Store. (submitted to VLDB 2019).

The SNOW Theorem Revisited: In highly-scalable storage systems for Web services, data is sharded into separate *objects*, also called *shards*, across several servers. Transaction isolation, while reading the objects, is at the heart of consistent data access when concurrent updates are present. In practice, systems experience a much higher number of READ transactions, consisting only of read operations, compared to WRITE transactions; consequently, lowering latency of READ transactions boosts service performance. In "The SNOW Theorem" paper by Haonan Lu et. al., the authors proposed four desirable properties in transaction processing systems for achieving low-latency of READ transactions, with asynchronous and reliable communications, and referred to them collectively as the *SNOW properties*: The underlying properties, in the context of an execution, are (*i*) *strict serializability* (S) property where read and WRITE transactions seem to occur atomically; (*ii*) *non-blocking* (N) property implies that for every read operation on any object, during a READ transaction, the response at the corresponding server is non-blocking; (*iii*) *one version and one round* (O) property implies every read operation, during a read transaction, completes in *one-round* of client-server communication and the respective server responds with only one version of the object value; and (*iv*) *concurrent* WRITE *transactions* (W) property states that

DISTRIBUTION A: Distribution approved for public release

READ transactions can have concurrent WRITE transactions. Then they argued that it is impossible to implement all the four properties, in the same system, even with at least three clients. They referred to their result as the *SNOW* theorem, and they posed the two-client setting as an open question.

In [18] we revisit the results of the SNOW theorem, resolve the open question and provide some new interesting results listed below.

1. We resolve the two-client scenario: We prove that even with two clients, without client-to-client messaging, it is impossible to design an transaction processing system which satisfies the SNOW properties.

2. We provide a rigorous proof of the SNOW theorem for systems with at least three clients, i.e., we show that it is impossible to implement a transaction processing system, consisting of at least three clients, even with client-to-client messaging, that satisfies the SNOW properties.

3. We derive a useful property for executions of algorithms that implement objects of data types considered in our work that helps us show the *strict serializability* property (S property) of algorithms presented in the paper.

4. We present an algorithm with *multiple writers, single reader* (MWSR) which satisfies the SNOW properties, with client-to-client messaging.

5. We present an algorithm, for *multiple-writer, multiple-reader* (MWMR) setting in the absence of client-to-client messaging, which satisfies the "S", "N", "W" properties and a weaker version the O property "o", where we use "o" to refer to the one-version requirement of a read operation, but it can take multiple rounds of communication to complete a read operation. Collectively, we refer to the "S", "N", "o" and "W" properties as the "SNoW" property.

6. We present an algorithm in the MWMR setting which satisfies the "S", "N", "W" properties and a property "$\tilde{o}$", which refers to the one-round requirement of a read operation, but a server can response with multiple versions of a shard. We refer to these properties as the "SN$\tilde{o}$W" property.

[18] Kishori M Konwar, Wyatt Lloyd, Haonan Lu, Nancy Lynch, The SNOW Theorem Revisited. (to be submitted to PODS 2019) https://arxiv.org/abs/1811.10577.

3 Patents Filed and Granted

The follwing patents, emerged out of this project, were filed through MIT.

1. TLO Case #:18616J
   Disclosure Date: 05/04/2016

DISTRIBUTION A: Distribution approved for public release

Disclosure Title: Storage-Optimized Data Atomic Algorithmi for Handling Errors and Erasures in Distributed Storage System.
Inventors: Muriel Medard, Erez Kantor, Alexander A Schwarzmann, Kishori M Konwar, Prakash Narayana Moorthy and Nancy Lynch

2. TLO Case #:19129
Disclosure Date: 12/20/2016
Disclosure Title: Techniques for de-duplicating Network coded Distributed Storage.
Inventors: Muriel Medard, Prakash Narayana Moorthy and Vitaly Abdrashitov

3. TLO Case #:19369
Disclosure Date: 05/01/2017
Disclosure Title: A Layered Distributed Storage Algorithm for Edge Computing Systems.
Inventors: Muriel Medard, Kishori Mohan Konwar, Prakash Narayana Moorthy and Nancy Lynch

4 Archival Publications

[1] Viveck Cadambe, Nancy Lynch, Muriel Medard, and Peter Musial. A Coded Shared Atomic Memory Algorithm for Message Passing Architectures. IEEE International Symposium on Network Computing and Applications (NCA14), 2014. Best Paper Award.

[2] Viveck R. Cadambe, Nancy Lynch, Muriel Medard, and Peter Musial, A Coded Shared Atomic Memory Algorithm for Message Passing Architectures. Distributed Computing 30(1): 49-73, February, 2017.

[3] Zhiying Wang and Viveck R. Cadambe. Multi-Version Coding - An Information Theoretic Perspective of Consistent Distributed Storage. Report available online at http://arxiv.org/abs/1506.00684.

[4] Zhiying Wang and Viveck R. Cadambe and Nancy Lynch. Information-Theoretic Lower Bounds on the Storage Cost of Shared Memory Emulation. Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), Chicago, IL, pages 305-313, July 2016.

[5] Zhiying Wang and Viveck R. Cadambe and Nancy Lynch. Information-Theoretic Lower Bounds on the Storage Cost of Shared Memory Emulation. http://arxiv.org/abs/1605.06844 (2016). Full version of [4].

[6] Kishori M Konwar, N. Prakash, Erez Kantor, Muriel Medard, Nancy Lynch, and Alexander A. Schwarzmann. Storage-Optimized Data-Atomic Algorithms for Handling Erasures and Errors in Distributed Storage Systems. 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 720-729, May 2016.

[7] Kishori M Konwar, N. Prakash, Erez Kantor, Muriel Medard, Nancy Lynch, and Alexander A. Schwarzmann. Storage-Optimized Data-Atomic Algorithms for Handling Erasures and Errors in Distributed Storage Systems. arXiv:1605.01748. Full version of [6].

DISTRIBUTION A: Distribution approved for public release

[8] Kishori Konwar, Nancy Lynch, Muriel Medard and Prakash Narayana Moorthy. RADON: Repairable Atomic Data Object in Networks. Proceedings of the 20th International Conference on Principles of Distributed Systems (OPODIS 2016), Madrid, Spain, December 2016.

[9] Kishori Konwar, Nancy Lynch, Muriel Medard and Prakash Narayana Moorthy. RADON: Repairable Atomic Data Object in Networks. arXiv:1605.05717. Full version of [8].

[10] Kishori M Konwar, N. Prakash, Muriel Medard and Nancy Lynch, A Layered Architecture for Erasure-Coded Consistent Distributed Storage. Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), Washington D.C., July, 2017.

[11] Kishori M Konwar, N. Prakash, Muriel Medard and Nancy Lynch, A Layered Architecture for Erasure-Coded Consistent Distributed Storage. arXiv:1703.01286. Full version of [10].

[12] V. Abdrashitov and Médard, M., "Durable Network Coded Distributed Storage", Allerton 2015.

[13] V. Abdrashitov and Médard, M., "Staying Alive - network coding for data persistence in volatile networks", invited paper, Asilomar 2016.

[14] P. Narayana Moorthy, Abdrashitov, V., and Médard, M., "A Generalization of Regenerating Codes for Clustered Storage Systems", invited paper, Allerton 2016.

[15] P. Narayana Moorthy and Médard, M. "Communication Cost for Updating Functions when Message Updates are Sparse: Connections to Maximally Recoverable Codes", invited paper, Allerton 2015.

[16] Viveck Cadambe, Nicolas Nicolaou, Kishori M. Konwar, N. Prakash, Nancy Lynch, Muriel Medard ARES: Adaptive, Reconfigurable, Erasure coded, atomic Storage, (arXiv:1805.03727) https://arxiv.org/abs/1805.03727.

[17] N. Prakash, Kishori M. Konwar, Muriel Médard and Nancy Lynch, FLECKS: Fast Lean Erasure-coded Consistent Key-value Store. (submitted to VLDB 2019).

[18] Kishori M Konwar, Wyatt Lloyd, Haonan Lu, Nancy Lynch, The SNOW Theorem Revisited. (to be submitted to PODS 2019) https://arxiv.org/abs/1811.10577.

5 Other Information

Changes in research objectives: None
Change in AFOSR Program Manager: None
Extensions granted or milestones slipped, if any: NCE was granted until Sep 30, 2018 to complete the work

DISTRIBUTION A: Distribution approved for public release

AFOSR Deliverables Submission Survey

Response ID:10730 Data

1.

Report Type

Final Report

Primary Contact Email

Contact email if there is a problem with the report.

hmjidila@mit.edu

Primary Contact Phone Number

Contact phone number if there is a problem with the report

617-258-0680

Organization / Institution name

MIT

Grant/Contract Title

The full title of the funded effort.

Network Coding for Strong Consistency Semantics in Distributed Shared Memory Networks

Grant/Contract Number

AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550

Principal Investigator Name

The full name of the principal investigator on the grant or contract.

Nancy Lynch

Program Officer

The AFOSR Program Officer currently assigned to the award

Beth Snyder

Reporting Period Start Date

09/30/2014

Reporting Period End Date

09/29/2018

Abstract

The abstraction of shared memory is a powerful tool that can greatly simplify design and implementation of software systems for distributed
settings. The key property of a shared memory abstraction is the consistency guarantees that it provides under concurrent access. The most intuitive consistency model is atomicity since its semantics closely reassemble these of a sequential register. Emulating shared atomic memory proves to be challenging especially if this is to be done in a distributed message passing setting with unreliable components, as is often the case in networked systems. To this end standard techniques and solutions have been developed, however these have high communication and storage costs. In this project, we explore several algorithms for efficient implementations of shared memory services that use erasure-codes. We also derive provable

strong consistency and liveness properties of these algorithms in the failure-prone setting. Novelty of this work is the fusion of standard replication techniques with network coding techniques. Shared memory services for distributed systems emulate a memory abstraction providing users with a consistent view of its content under concurrent access, with a best effort to ensure availability and survivability.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

sf0298_12.3.18.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

sf0298_12.3.18.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

New discoveries, inventions, or patent disclosures:

Do you have any discoveries, inventions, or patent disclosures to report for this period?

No

Please describe and include any notable dates

Do you plan to pursue a claim for personal or organizational intellectual property?

Changes in research objectives (if any):

Change in AFOSR Program Officer, if any:

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Dec 03, 2018 10:56:09 Success: Email Sent to: hmjidila@mit.edu