18th ICCRTS

Command and Control Data Synchronization Service

DRAFT

Title: C2 Data Synchronization in Disconnected, Intermittent, and Low-bandwidth (DIL) Environments

Topic 3: Data, Information and Knowledge
Topic 4: Collaboration, Shared Awareness, and Decision Making

Robert Perkins (POC)
SSC PAC Code 53625
San Diego CA 92152
bob.perkins@navy.mil
619 553-1772

Fernando Dejesus
SSC PAC
Jayson Durham
SSC PAC

Robert Hastings
Moebious
John McDonnell
SSC PAC

Table of Contents

Table of Figures

Abstract

This paper discusses data synchronization methods to support C2 missions conducted in disconnected, intermittent, and latent, (DIL) maritime environments. A maritime specific C2 data synchronization service (C2SS) capability will provide data consistency to support synchronized operations of maritime forces. The purpose of this research is to discover what data synchronization capabilities are required in the maritime domain that are not currently addressed by commercial products and then develop a set of services, procedures, and/or processes that will provide an enhanced capability. A survey of capabilities that enable data synchronization is provided. A potential architecture is presented for implementing candidate technologies within a maritime tactical C2 framework. A use case is then presented as an operational thread for analyzing the collaboration required of plans and task information across operational and tactical users. Upon completion of the above, a measure of effectiveness, characterized as C2 entropy, will be developed to support evaluating the performance of data synchronization technologies under various DIL conditions. The objective of this work is to ultimately provide a C2 Synchronization Service to support maritime C2 operations in a net-centric capacity.

Background

Tactical situations increase the likelihood of a Disconnected, Intermittent, and Low-bandwidth (DIL) environment while simultaneously increasing the need for an updated and synchronized Common Operational Picture (COP). This paper provides a survey of technologies that support the synchronization of C2 Data in light of the distributed operations in a DIL environment. The focus of this effort is on C2-Data, primarily planning and situation awareness (SA) information.

Existing Command and Control (C2) systems that use event-based protocols to manage tracks may conserve bandwidth, but do not guarantee a common operating picture in disconnected, intermittent, and low-bandwidth (DIL) environments.

Self-synchronization will accelerate collaborative military decision-making and execution processes, and will enable future forces to improve tactical tempo and speed, as well as lethality and momentum heretofore not possible. The objective of this effort is to develop, integrate, and assess data synchronization to support tactical C2 data consistency in a DIL environment. The C2SS technology effort is designed to integrate with the Command Control Rapid Prototyping Continuum (C2RPC). C2SS technology will also be coordinated with the Office of Naval Research (ONR) 6.2 Data Synchronization efforts, and specifically the Data Movement Services (DMS) project.

Need

Tactical C2 requires the capability to support collaborative planning, distributed execution and mission-focused data delivery in a DIL environment. State-of-the-Industry data synchronization capabilities do not support Navy C2 tactical requirements. Maritime-related operational requirements and transport layer limitations of Navy platforms also present unique challenges in developing solutions. Synchronized data must support all echelons and allow the creation of the COP and the intelligence-running estimate. Additionally, it must support battle command, targeting, effects, and combat assessments.

Given the unique tactical requirements of maritime C2 it is virtually impossible to use an off-the-shelf solution. Factors to consider during selection include ease to install and administrators' ability to adapt to the situation and

environment; effectiveness; speed of replication; cost; and ability to comply with security and information assurance requirements. Increases in performance and flexibility generally increase the lines of code and cost. Other general functional requirements that may apply in order to perform in DIL environments include alignment with Afloat Core Services (ACS)[1], providing a mechanism and strategies for Dynamic/Smart data management, and forward caching.

Technical Approach

An examination of what functions are required to perform C2SS will be conducted. A solid foundation in understanding requirements will ensure relevance of the project. Disciplined use of the agile development process will provide speed and agility, allow for incremental development and testing, permit rationalized requirements, lower cost, and reduce risk. Emphasizing an understanding of the unique requirements of maritime C2 systems will ensure that commercial state-of-the-industry capabilities already available will be appropriately integrated and will not be duplicated.

The next step will be to survey potential technical solutions that may meet some functional requirements. Priorities may be changed/established by testing use cases with representative data. Table 1 summarizes functional requirements mapped to a sample of candidate solutions. Two potential technical solutions have been identified as viable by previous analysis. The Vector Clock algorithm is a candidate to be implemented as a subset in peer-to-peer and multi-master environments. The Secure Hash Algorithm (SHA-1/2/3) will be used as a part of distributed set reconciliation and incorporated to determine state changes and to facilitate synchronization on subset versus full replication. Additional gaps or shortfalls will then be identified in existing technologies and services. This information will be used as the underlying foundation for a functional architecture. SHA-1 will maintain eventually consistent distributed sets allowing optimistic updates and then reconciling the sets using a Merkle tree of SHA-1 checksums to detect set differences followed by the use of Version Vectors to exchange the correct updates based on causal ordering or indicate conflicting updates.

Unified Modeling Language (UML) artifacts will be used to specify, visualize, modify, construct, and document the artifacts of the C2SS as it is developed, including both static and dynamic modeling. The level of detail required will be evaluated as the development effort proceeds. Use of UML will reduce overall risk of the development effort. System level integration will include C2RPC, Maritime Tactical Command and Control (MTC2), Data Movement Services (DMS), Open Track Manager (OTM), and relevant maritime C2 architectures.

C2SS Functions

C2SS functions currently identified that are required to enable data synchronization in the maritime C2 DIL environments include Bandwidth Utilization, C2 Entropy/Data Integrity, Checkpointing, Filtering, Concurrent Distributed Updates, Conflict ID, Data Mapping, Eventual Consistency, Flow Control, Health and Status, Network Awareness, and Queue Management. The discovery process during the development effort may result in amendments in these functional requirements already identified.

[1] Core Services (CS) are a component of the overall system architecture and fundamentally act like a system application, or enhancement. CS v1.0/1.1 provides an initial, technically mature core services stack. ACS develops open-standards compliant products and supports the integration of those products into the CANES ACS stack.

Bandwidth Utilization

"Bandwidth usage is the amount of data transmitted and received by a particular computer or user. The more data exchange, the higher the potential to clog the network, and the more energy taken up by a particular user. As a result, some companies have bandwidth usage caps which are designed to prevent users from transmitting and receiving so much data that they slow the network down, impairing the experience for other users." [1]

"In the case of a computer, bandwidth usage refers to all inbound and outbound traffic. Inbound traffic is data which comes into a computer, as for example when someone downloads a file from a website. Outbound traffic is data which is transmitted by the computer, such as a file attached to an email. Typically, bandwidth usage is measured in bytes per second (bps), although other units of measurement may be utilized, and service providers express bandwidth limits in terms of gigabytes, with limits commonly being daily or monthly." [1]

Bandwidth is often limited by the underlying physical capabilities of the transmission media and hardware networking equipment. In a DIL environment, all available bandwidth may be utilized. However, it is somewhat counter intuitive to realize that almost all networks are not fully utilized. When bandwidth is not utilized it is essentially wasted. When bandwidth is underutilized, a synchronization service should apply techniques to pre-position data. When bandwidth is over utilized, a synchronization service should manage the data being transmitted so that the highest priority mission data is transmitted first.

Concurrent Distributed Updates

Concurrent Distributed Updates (CDU) enable two or more sites to change the same data while disconnected and to merge logical data upon reconnection.

Conflict Identification

Conflicts can occur in a replication environment when two or more databases make a change to the same piece of data at multiple sites and then the synchronization engine tries to apply those into a single database. Types of conflict considered for this problem include (but are not limited to) update conflicts, uniqueness conflicts, and delete conflicts.

Data Integrity

In computing, data integrity refers to maintaining and assuring the accuracy and consistency of data over its entire life-cycle, and is an important feature of a database or RDBMS system. Data warehousing and business intelligence in general demand the accuracy, validity, and correctness of data despite hardware failures, software bugs, or human error. Data that has integrity is identically maintained during any operation, such as transfer, storage, or retrieval. All characteristics of data, including business rules, rules for how pieces of data relate, dates, definitions, and lineage must be correct for its data integrity to be complete. When functions operate on the data, the functions must ensure integrity. Examples include transforming the data, storing history, and storing metadata. [2]

Data Mapping

Data mapping is the process by which two distinct data models are created and a link between these models is defined. Data models can include either metadata, an atomic unit of data with a precise meaning in regards to semantics, and telecommunications. The system uses the atomic unit system to measure the properties of electricity which contain the information. Data mapping is most readily used in software engineering to describe

the best way to access or represent some form of information. It works as an abstract model to determine relationships within a certain domain of interest. This is the fundamental first step in establishing data integration of a particular domain.

The main uses for data mapping include a wide variety of platforms. Data transformation is used to mediate the relationship between an initial data source and the destination in which that data is used. It is useful in identifying parts of data lineage analysis, the way in which data flows from one sector of information to another. Data mapping is also integral in discovering hidden information and sensitive data such as Social Security numbers when hidden within a different identification format. This is known as data masking. [3] In the context of synchronization, it is imperative that all of the data instances be mapped to the appropriate mission area.

Eventual Consistency

Consistency can be described as strong, weak, casual, or eventual. Eventual Consistency is a form of weak consistency. In Eventual Consistency; "The storage system guarantees that if no new updates are made to the object, eventually all accesses will return the last updated value. If no failures occur, the maximum size of the inconsistency window can be determined based on factors such as communication delays, the load on the system, and the number of replicas involved in the replication scheme. The most popular system that implements eventual consistency is DNS (Domain Name System). Updates to a name are distributed according to a configured pattern and in combination with time-controlled caches; eventually, all clients will see the update." [4]

Eventual Consistency is a required functionality because it allows nodes to continue to process information while the network is down. It would be unworkable to require Open Track Manager (OTM) to wait for all nodes on the network to be committed to the same track picture before it completes processing an update. An notional approach for Eventual Consistency is shown in figure 1.

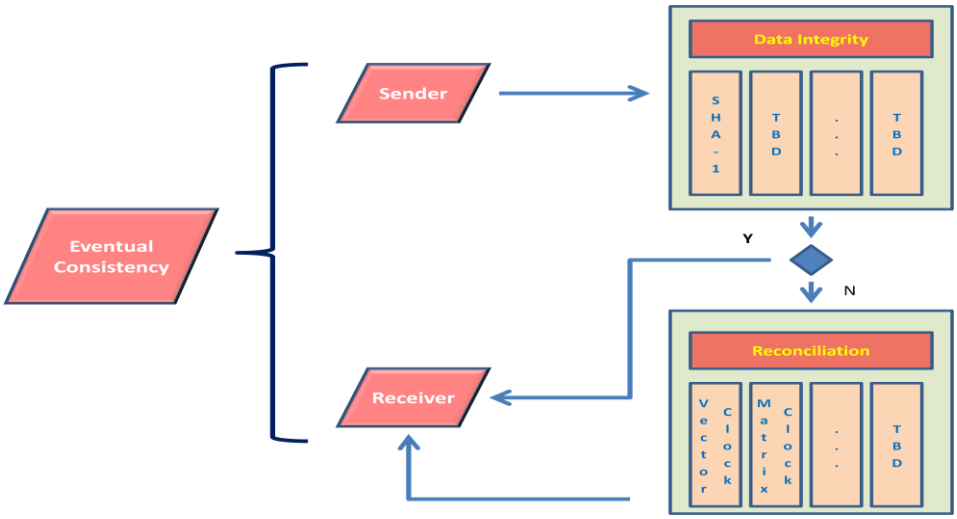

Figure 1. Eventual Consistency

Filtering

"A filter is used to restrict the data that is synchronized. Typically, the source provider applies the filter during change enumeration to restrict the changes that are sent. Sync Framework supports the following types of filters.

- Item filters restrict synchronization data to a subset of items, such as to only synchronize .txt files between two file folders, ignoring files of other types. Items do not change in a way that causes an existing item to move into or out of the filter. Item filters are simple to use, but the metadata used for synchronization grows in proportion to the number of items that are in the synchronization scope.
- Change unit filters restrict synchronization data to a subset of change units, such as to only synchronize the name and phone number fields of a contact, ignoring the remaining change units.

The filter can be defined by the synchronization application or by the source or destination provider, or it can be negotiated between the source and destination providers. The filter is used by the source provider when it detects changes. Change batches built during change detection include only the synchronization data that passes the filter. The filter can also be used by the destination provider when it applies changes. Changes that are applied to the destination replica include only the synchronization data that passes the filter". [5]

Flow Control

"In data communications, flow control is the process of managing the rate of data transmission between two nodes to prevent a fast sender from outrunning a slow receiver. It provides a mechanism for the receiver to control the transmission speed, so that the receiving node is not overwhelmed with data from the transmitting node. Flow control should be distinguished from congestion control, which is used for controlling the flow of data when congestion has actually occurred. Flow control mechanisms can be classified by whether or not the receiving node sends feedback to the sending node.

Flow control is important because it is possible for a sending computer to transmit information at a faster rate than the destination computer can receive and process it. This can happen if the receiving computer has a heavy traffic load in comparison to the sending computer or if the receiving computer has less processing power than the sending computer". [6]

Health and Status

Health and status refers to the database state, including the set of stored data. Entering, modifying, or deleting information changes the database state. The functional requirement is the capability to report this state.

Network Awareness

Awareness enables applications to sense changes to the network to which the database is connected. Three possible conditions are 1) no network connection, 2) a single network connection, and 3) multiple network connections. There is a requirement to transition smoothly between these three states.

These transitions create four possible scenarios; 1) A network connection becomes available when there was not one previously available, 2) A new network connection becomes available when one (or more) was already available, 3) The entire network connection disappears (drops out), leaving no connections, and 4) A network connection disappears, leaving one or more connections.

Queue Management

Queue management functionality is required to increase efficiency and speed of access to priority files. The queue database uses log files to accept, track, and maintain data. To enhance performance, all message transactions are written first to log files and memory, and then to the database file. The checkpoint file tracks the transaction log entries that have been committed to the database. An approach for Queue Management is shown in figure 2.

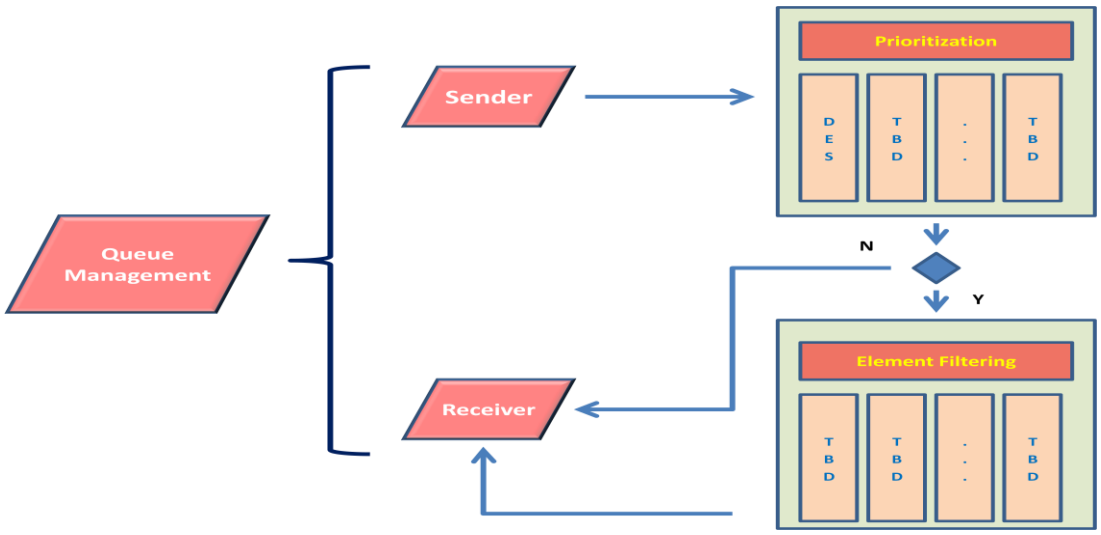

Figure 2. Queue Management

Analysis

Table 1 below depicts functional requirements required for the C2SS. The ideal C2SS Service will be composed of the functions in the first column and are ordered by priority. Candidate technology solutions are aligned in accordance with their Technology Readiness Level (TRL) level (*partial list shown*). Based on analysis the major gaps and consequently the functional areas of focus for this study are Eventual Consistency, Concurrent Distributed Operations, Entropy/Data Integrity, and Checkpointing.

Table 1. C2SS Functional Breakdown

			Implementation Options with respect to TRL				
	C2SS Function	**Priority**	**9**	**8**	**7**	**6**	**5**
1	**Queue Management**	1	DVDX VDX CST			DES	
2	**Priority Management**	1				DES	
3	**Network Awareness**	1	TCP			NAS NORM FFDS	
4	**Eventual Consistency**	1	DVDX VDX	DNS	Vector Clocks	Reconciliation (SHA-1)	
5	**Concurrent Distributed Updates**	1					DCVS
6	**Conflict ID**	1			Vector Clocks MobiLink	Sync Framework DDCR ANQF	Agreeing to Agree
7	**Data Integrity**	1			SHA-1		Checkpointing
8	**Bandwidth Utilization**	2	DVDX				
9	**Filtering**	2			Application		
10	**Compression**	2	LZW LZO		Delta	Kinematic	

					Filtering	
11	**Health and Status**	2	DVDX VDX CST			C2 Entropy
12	**Data Mapping**	3			DES	
13	**Flow Control***	3		Merge Replication		

Enabling Technologies

Enabling technologies for all data synchronization functional areas include Agreeing to Agree, COP Synchronization Tools (CST), Data Exchange Services (DES), Delta Compressing (DVDX), Distributed Concurrent Versions System (DCVS) Network Aware Service (NAS), NORM, Transmission Control Protocol (TCP), Federated Force Discovery Service (FFDS), Kinematic Filtering, Lempel-Ziv-Oberhumer (LZO), Lempel-Ziv-Welch (LZW), Merge Replication, Secure Hash Algorithm (SHA-1/2/3), Sync Framework, Variable Data Exchange (VDX), Vector Clocks (Matrix Clocks) etc. Other technologies are being added during the course of the research. The degree of coupling between these technologies needs to be examined. Detailed descriptions of technologies are contained in Appendix A.

Functional Architecture Model

The C2SS efforts will provide functionality consistent with the Clinger-Cohen Act. Specifically, this will include model-based multi-tiered EA/SOA/Cloud computing architecture frameworks and associated infrastructures. A number of references provide background and status of this transformation to EA/SOA/Cloud computing within respective DOD [1-7], DON [8-9], PEO/PMW [10-12], and SYSCOM [13-16] echelons. Note that the defense oriented EA/SOA/Cloud transformation efforts are actively working to align with other agencies EA/SOA/Cloud reference models and resources [17-18]. For maritime tactical C2 (MTC2), this effort is also benefiting from references that address these more MTC2 specific EA/SOA/Cloud challenges (Placeholder1) (somebody, 2012) [19-20].

The “C2SS Functional Architecture (C2SS FA)” in terms of the actual software (e.g. java code), is a collection of code that is called by other programs (e.g. MTC2 applications, such as OTM and P&T) for performing the respective data synchronization functions, as required and specified within the context of the particular application. For example, two different applications may require the same type of concurrency control functionality (i.e. consistency models) but different implementations of even the same type of consistency model (i.e. eventual consistency). In other words, for each separate application, the implementation of an eventual consistency capability may be quite different due to different application specific specifications and requirements. Thus, the C2SS interfaces support the definition of abstract methods and associated operator overloading. In the example of the two applications that both have eventual consistency needs, the difference in the respective application-specific requirements required the “overloading” of the “eventual-consistency method” with two different underlying implementations. The benefit of this type of object model is the ability to semantically describe the desired functionality in terms that the particular type of functionality is typically understood. This also allows the separation of concerns wherein data tier backend services are logically separated from the more user-interface and business logic related code.

In summary, a type of concurrency control functionality (i.e. eventual consistency) may be needed for multiple MTC2 applications but due to their context specific requirements, each may require their own application-specific solutions (i.e. OTM - "SHA-1 Pilot" versus Plans/Tasks - "TBD"). Thus, the emphasis on similar operational (i.e. behavioral) requirements and specifications that are delegated to the C2SS objects for implementation

As illustrated within the Table 1 C2SS Functional Breakdown, thirteen data synchronization related functions have been identified that when combined perform adequate data synchronization for the tactical maritime environment. A priority was assigned to each function. The survey (in progress) lists what capabilities might currently exist that could perform those functions along with their Technology Readiness Level (TRL). Of those thirteen functions, four have been identified for piloting a logically separated "C2SS Sub-Tier" of the "MTC2 Data Tier." These four were selected based on their priority and lack of high TRL solutions. Eventual Consistency has TRL 9 services though the proposed services for Conflict ID and Concurrent Distributed Updates are closely coupled with Eventual Consistency. Figure 3 (below) illustrates how C2SS elements and how those elements might interface with MTC2. Open Track Manager (OTM) applications may require a different interface than Plans and Task (P&T) applications.

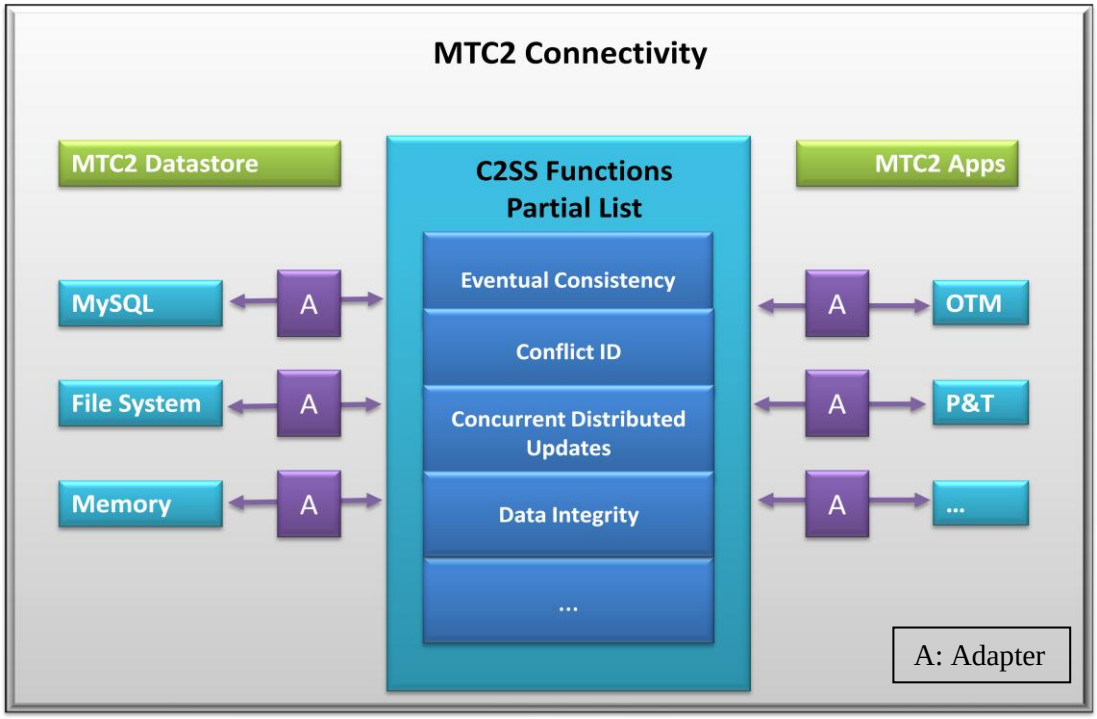

Figure 3. C2SS Diagram

Figure 4 (below) illustrates how the agile development methods will be incorporated to cyclically develop and test solutions. In this example the SHA-1/Merkle Tree techniques will be developed and tested. Other solutions will be developed and tested as appropriate.

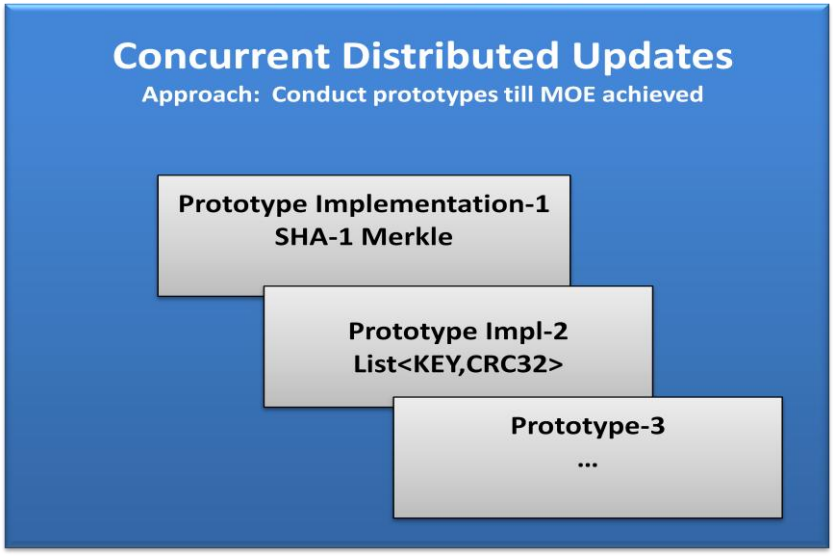

Figure 4. Prototyping Approach

Architecture Implementation Constructs

Figure 6 and Figure 7 illustrate the design for prototype implementation one, SHA-1 Implementation. The SHA-1 Implementation focuses on allowing Concurrent Distributed Updates while using implementations from Table 1 to provide Eventual Consistency, and Conflict Identification. This work is based on prior work for Master-Mirror Set Reconciliation based on a Merkle Tree of SHA-1 checksums. Figure 6 is an UML class diagram showing the classes involved in the implementation.

In the design updates are allowed at any time on the Client or Server, thus allowing for Concurrent Distributed Updates. Periodically the ClientSynchronizer and ServerSynchronizer exchange messages over a Network to determine which set elements need to be exchanged to synchronize the Client and Server. Since the Client and Server are allowed to be inconsistency until the synchronization is complete, but will be the same after the synchronization, the implementation achieves Eventual Consistency. In the case of potentially conflicting updates, Causal Ordering is determined by exchanging Version Vectors for elements that are missing or different between the Client and Server. Given a Causal Ordering indicates unrelated changes (or conflict), this implementation flags the conflict, but prefers the Server updates over the Clients (thus overwriting the Client's updates in case of conflict). How to handle conflicts for various C2 data types is future research.

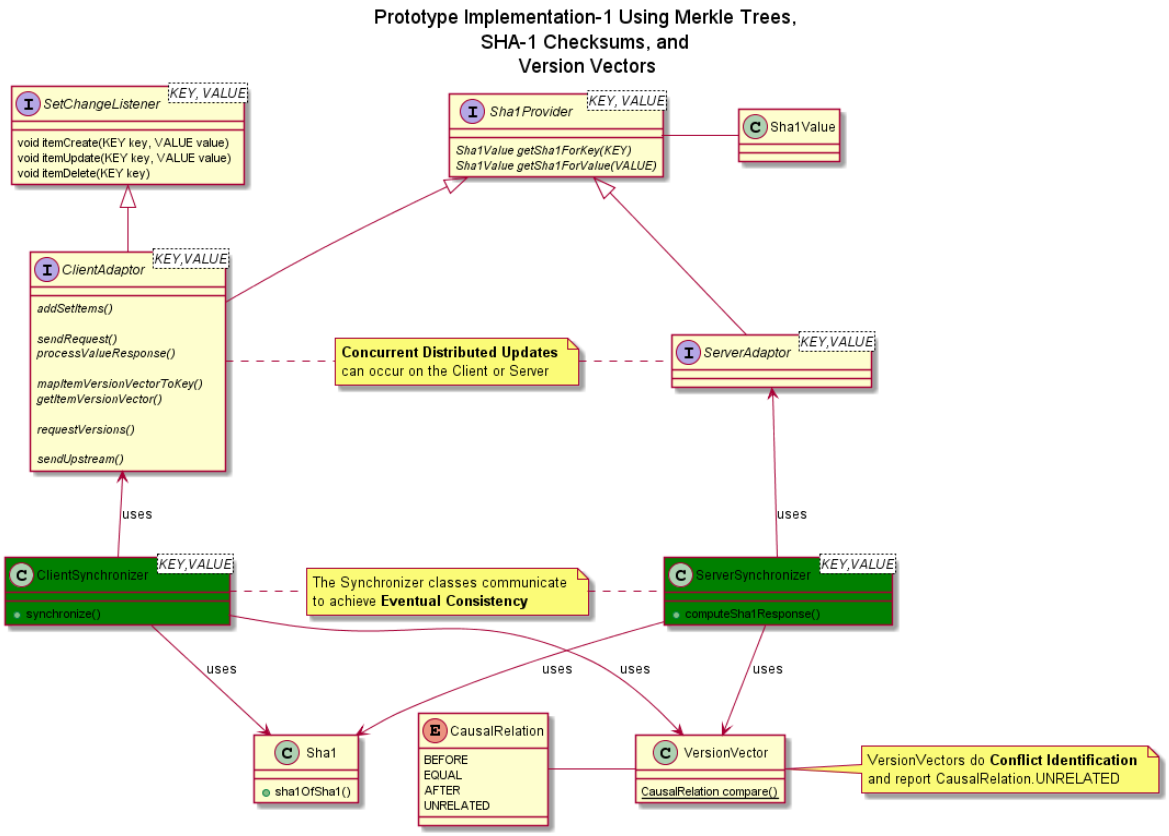

Figure 5. SHA-1 Implementation

The sequence of the exchanged messages is detailed in Figure 7 SHA-1 Sequence Diagram. There are essential three phases. The first phase, the loop, is used to exchange more and more of the Merkle tree depending on which branches of the tree indicate differences. The second phase is Version Vector exchange. Once the set of differences is discovered in the first phase, Version Vectors are exchanged. There are two forms of the exchange. The first form allows the server to compute the Version Vectors that are sent based on a fragment Merkle Tree of SHA-1 checksums, see SHA-1 Set Reconciliation for more details on this approach. The second form allows the client to request Vestion Vectors for set elements that are only in the Client set. The Client and Server must maintain Tombstones with Version Vectors for deleted set elements so that the algorithm can detect the difference between an added set element and a deleted set element. The third phase is exchanging the necessary set elements to resolve the differences between the Client and Server. Give that there are no updates to the Client or Server during a pass through this entire sequence diagram, the Client and Server will have the same state for their sets. If an updates happens during a pass, then differences may remain. However, the next pass will resolve these. Since the passes are run infinitely as soon as there is a pass without any updates during the pass the Client and Server will be synchronized. One of the key benefits of using a Merkle-Tree of SHA-1 checksums is that if the Client and Server are already synchronized their top-level SHA-1 values will match and the algorithm will stop after exchanging a single SHA-1 checksum from the Client to the Server. Otherwise, O(m * log (n)) data will be exchanged to synchronize.

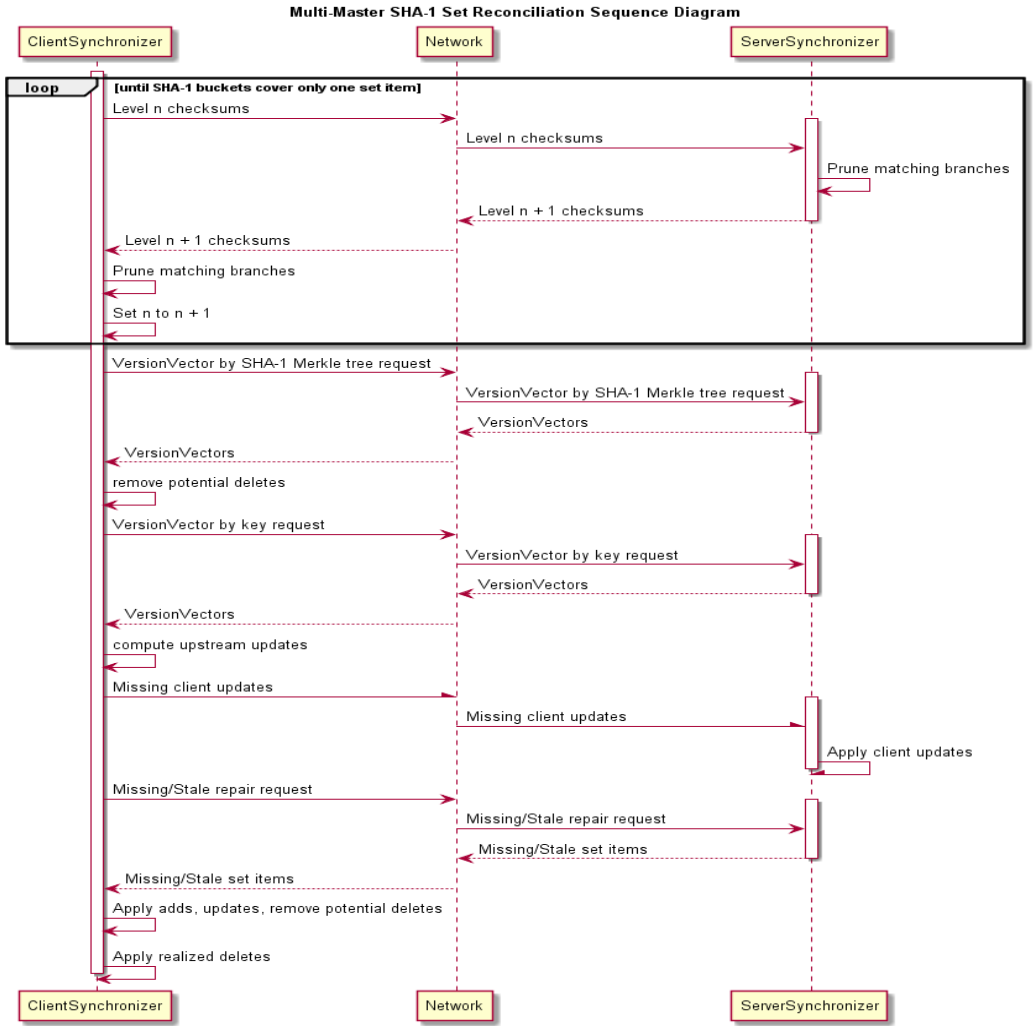

Figure 6. SHA-1 Sequence Diagram

Use Case across Operational and Tactical Nodes

For testing and experimentation, the C2SS project is using the Expanded Maritime Interdiction Operations (EMIO) scenario that was originally developed for C2RPC and more recently updated for the MTC2 DaaS LTE effort [NeedDaaSLTERef]. The scenario highlights MTC2 interaction between a Maritime Operations Center (MOC) ashore and a Carrier Task Force (CTF) afloat. The scenario is that reliable intelligence reports indicate that piracy activity is likely in the Area of Responsibility (AOR). Specifically, Yagi ships carrying hazardous cargo are being targeted for hijacking. As a result, the Commander has issued an order that all Yagi ships carrying hazardous cargo are Critical Contacts of Interest (CCI's). Figure 7 illustrates the topology of the MTC2 nodes ashore (i.e. MOC) and afloat (i.e. CTF platforms and coalition platform).

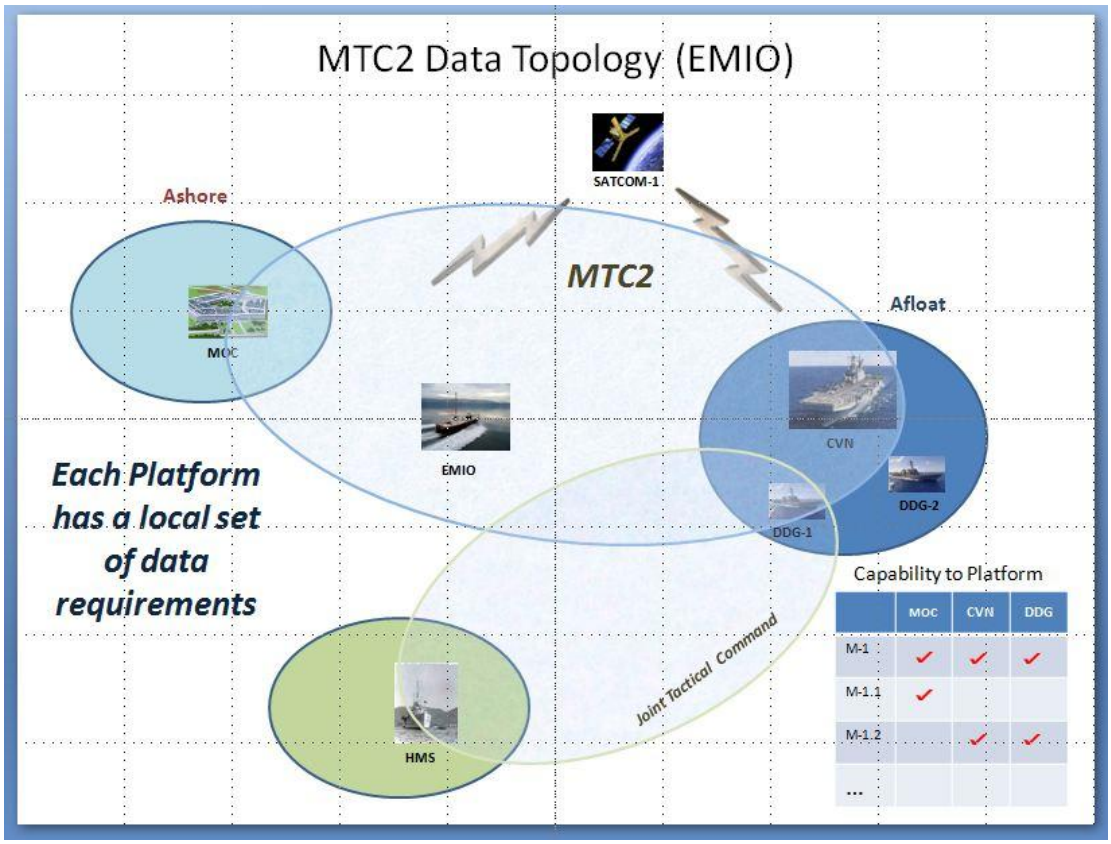

Figure 7. Network Topology of MTC2 Nodes

The purpose of the scenario is to highlight the following types of use-cases of MTC2 capabilities that are employed during this type of EMIO mission:

(1) Creation of a Commanders request for identifying and tracking Critical Contacts of Interest (CCIs);
(2) Identification of CCIs and addition to a "watch list";
(3) Visualization, updating, and distribution of CCI and EMIO mission information;
(4) Reapportionment of blue-force platforms for new EMIO mission requirements;
(5) Performance of EMIO situation awareness (SA) and track management (TM);
(6) Force readiness assessent and allocation of resources for EMIO platform-boarding operation;
(7) Information sharing and collaboration with coalition forces for execution of boarding of pirate ship.

For highlighting the value of C2SS capabilities, the following paragraphs focus on two examples where the MOC ashore is interacting with the carrier group command platform afloat, called CTF-FOO/CVN-BAR. The first example illustrates that a high priority CCI may not have been otherwise identified by the MOC due to DIL conditions between the MOC and CTF-FOO/CVN-BAR. In the second example, the C2SS capability enables better coordinated and more robust message communication between the CTF-FOO/CVN-BAR, MOC, and coalition-platform nodes, during DIL conditions.

Figure 8 highlights and visually illustrates where the MOC is able to quickly inspect the white shipping platforms with the area of responsibility (AOR) and identify CCIs. With the C2SS capability, a "synchronization status" widget is able to keep the MOC and CTF-FOO/CVN-BAR watch officers posted regarding the status of their respective connectivity. As illustrated in figure 7b [needs to be generated via Rob/Bret and mocked-up overlays], the "C2SS status widgets" alert the operators that they are operating under DIL conditions. In this particular case, the white shipping track management capabilites that are local (i.e. organic) to CTF-FOO/CVN-BAR have just

detected and verified a new track is entering the AOR. Unfortunately, due to an unanticipated SATCOM disconnect [satcom handoff malfunction ?], this new organic track does not appear on the MOC OTM display. Thus, without the C2SS widget, this track would not be included in the CCI selection process. Fortunately, while under disconnected conditions, the MOC OTM operator can submit a "C2SS request" regarding information that is provided once connection is reestablished. In this particular example, when the link is reestablished, the MOC officer is alerted of the new white shipping track. The manifest for the new track indicates that this is another Yagi vessel that is carrying hazardous material. Thus, this additional vessel is added to the respective EMIO CCI watch list. Without the C2SS service, the platform would not have been identified as a CCI and added to the EMIO CCI watch list.

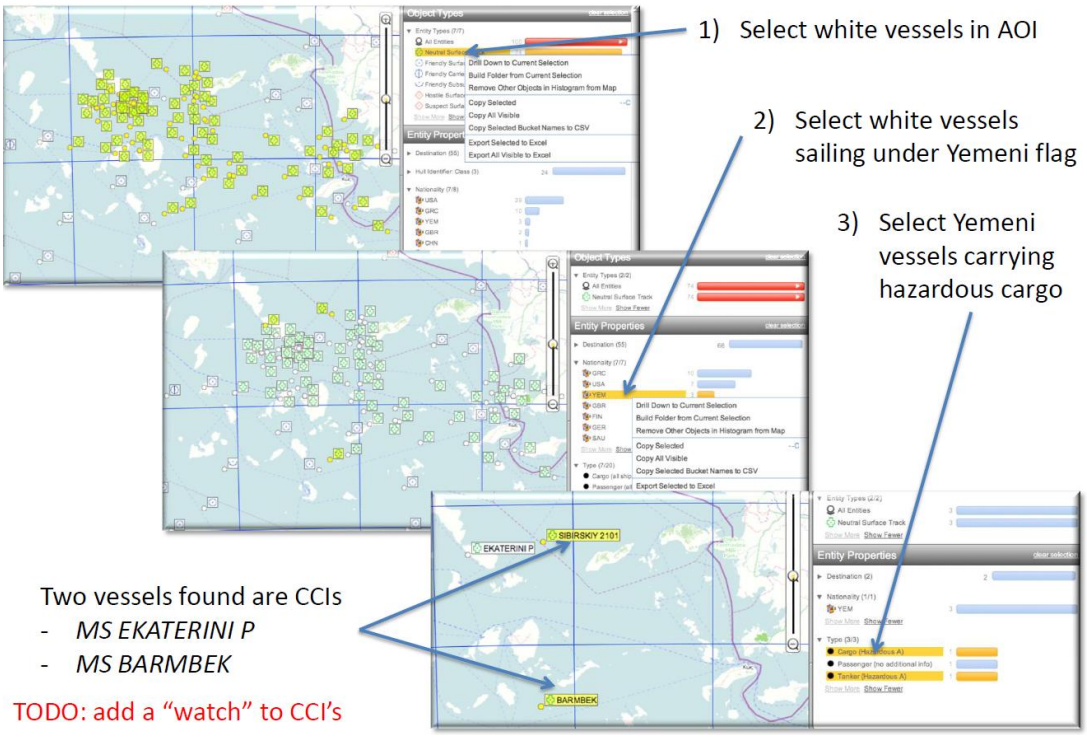

Figure 8. Interaction between MOC and CTF-FOO/CVN-BAR: Selection of CCIs

Figure 9 illustrates communication between the MOC, CTF-FOO/CVN-BAR, and a coalition platform that happens to be in the AOR and has a readiness level sufficent for boarding the suspect vessel. Unfortunately, within this C2SS version of the scenario, the communications between the MOC and CTF-FOO/CVN-BAR are still suffering from disconnections and intermittent connectivity. Fortunately, the "C2SS Status Widgets" for both the MOC and CTF-FOO/CVN-BAR are keeping the respective users updated of the status of the message transmissions. The widget also allows the operators to submit a "C2SS priority request" for high-priority reports and messages that relate to specific platform tracks, such as the coalition partner. In this case, due to the ability for each to independently identify the coalition platform as a viable coalition platform in the AOR that has the necessary level of readiness, they each are able to submit a high-priority request for any reports and messages relating to this specific coalition track. As the connectivity comes-and-goes, these high priority assignments allow robust, timely, and well-coordinated information sharing and interoperability with the coalition partner, in spite of the degraded communications due to DIL conditions.

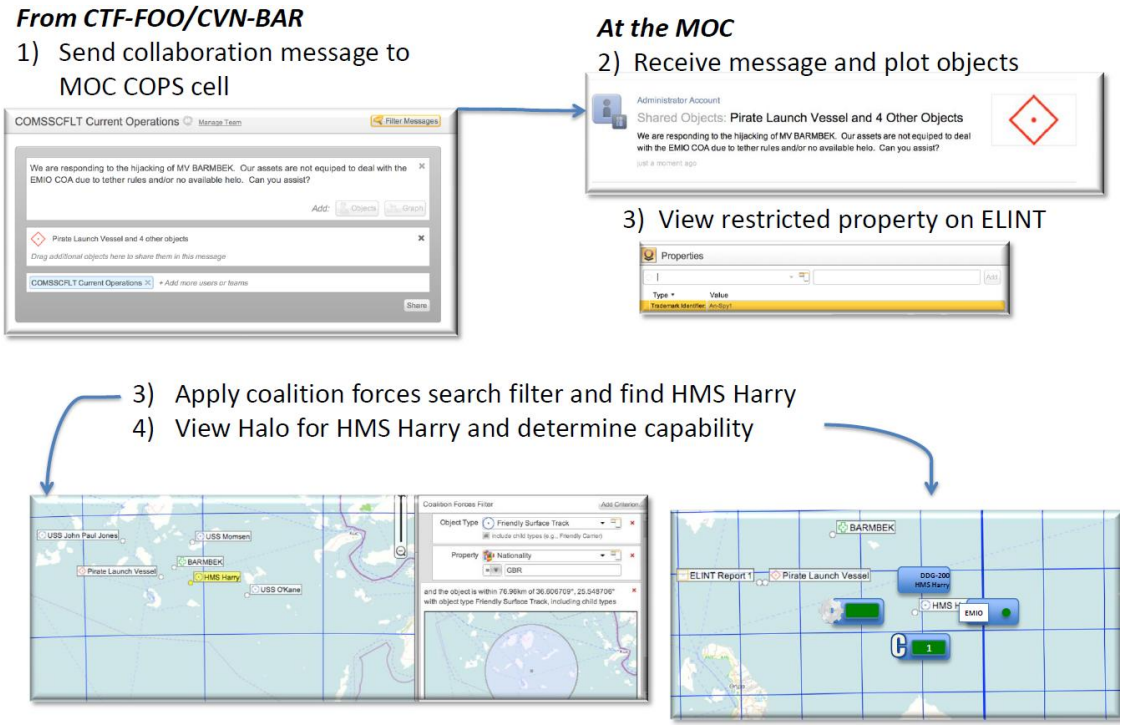

Figure 9. Example message communication between MOC and CTF-FOO/CVN-BAR

C2-Entropy Measure of Performance

A means to evaluate the effectiveness of the proposed C2SS is required. C2-Entropy and other methods previously proven effective for Command & Control Rapid Prototype Capability (C2RPC) will be used as Measures of Effectiveness (MOEs) for C2SS.

C2-Entropy is defined as a measure of new information over time. C2-Entropy will enable evaluation of associated metrics (or measures of effectiveness) for proposed solutions. For example, bandwidth utilized for synchronization is related to C2-Entropy because it is expected that if new information is not being shared, then the bandwidth utilized for synchronization would drop to zero. Conversely, if larger quantities of new information were being generated, then it would be expected that additional bandwidth would be utilized. However, the relationship between information and bandwidth is greatly influenced by how that information is encoded and shared. For example, compression techniques can reduce the bandwidth required. The use of kinematic predictions to filter what reports are transmitted can greatly reduce the amount of bandwidth required to transfer information about tracks as long as exact information is not required. Regardless of the exact mathematical relationship of information to bandwidth, it is clear that concurrently minimizing bandwidth and information loss is a goal for any synchronization service that operates in a DIL network. This MOE will assess bandwidth in relation to the number of set elements and the number of set modifications.

Consider the case where there are two empty sets; A and B. To synchronize the sets periodically all the elements from A are sent to B and then from B to A. In total, there are n elements so we would indicate that this approach is O(*n)* . Here *m* (then number of set modifications is not a factor because we are always sending all *n* elements. Conversely, suppose that changes are sent to set A event by event as they happen in near-real-time. And do the same for changes to B. Then this approach is O(*m)*. O(*m)* is the theoretical lower bounds because it exactly

matches the information that has changed. However, sending event by event changes does not handle the case where events may be lost during delivery. A DIL network environment means that network partitions must be taken into account. In addition to DIL, this approach considers hardware failures as well. If, set B is reset to the empty set after a hard drive failure, this approach can replicate the contents of set A to set B using O ($m * \log n$) bandwidth.

Furthermore, this approach can estimate a percentage of difference between two sets. This is called a measure of effectiveness, percent synchronized. When two sets have zero elements in common the percent synchronized is 0% and when two sets have all elements in common the percent synchronized is 100%.

This approach was validated during Trident Warrior 2008. COMPACFLT Data Fusion Center (DFC), required more insight to the measure of effectiveness of C2 track data synchronization that was used for master/mirror synchronization. Percent synchronized was used as a metric and the concept was well received by the operational users. It was further developed and subsequently deployed. The percent synchronized display shows the level of data synchronization for each unit. This enables the DFC to identify which unit is calling for support as a result of a mis-configuration because it is the unit whose percent synchronized is zero.

Another proposed MOE is one that was developed for C2RPC. This utilizes a graph with two area lines representing elements. The green area line represents the number of elements that are considered synchronized. The purple area line represents the total number of elements from both sets. When the sets are the same the green and purple area lines match. The differences between the green and purple indicate how many differences there are. See Figure 10.

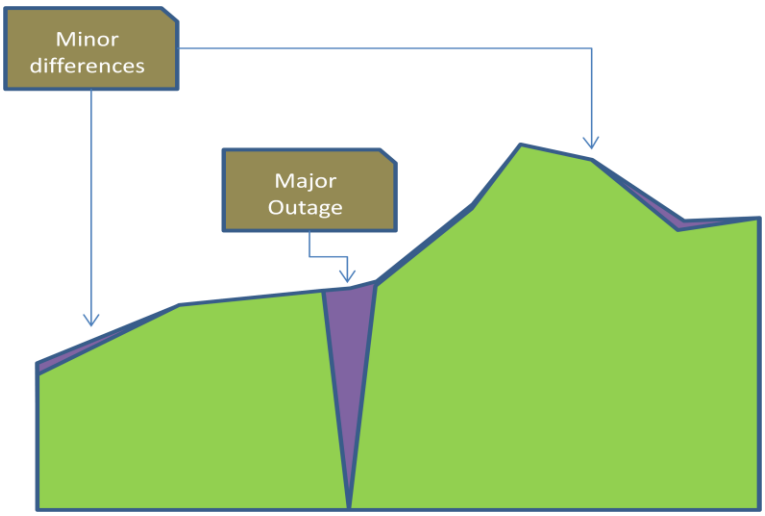

Figure 10. Geo-server Synchronization

Conclusion

Further study is required to confirm that the functional areas noted in this report sufficiently describe an ideal C2SS. A search for additional technologies will be conducted during the course of the project. Those leading to suitable commercial or open source implementations will be evaluated for inclusion

Technology gaps initially appear in two areas. There are technology gaps in performing specific functions identified in this paper to include technologies required to support Eventual Consistency, Concurrent Distributed Updates, and Conflict Identification. Technologies to support these functional areas specific to the maritime environment are not yet developed or are at low Technology Readiness Levels (TRL). The second integration issue includes those technologies required to integrate this group of data compression technologies with maritime C2 systems.

The initial technical area of focus will be vector clock algorithms. Furthermore, a Merkle tree of SHA-1/2 will be used in the integration to determine state changes and facilitate synchronization on a subset versus full data replication. Additional technologies will be examined to fill remaining gaps in functional requirements. Although these areas are pre-selected, the agile development model will allow adjustments as alternative solutions are discovered.

Bibliography

[1] "WiseGeek," [Online]. Available: http://www.wisegeek.com/what-is-bandwidth-usage.htm. [Accessed 15 January 2013].

[2] "Data Integrity," [Online]. Available: http://en.wikipedia.org/wiki/Data_integrity. [Accessed 15 January 2013].

[3] WiseGeek, "What is Data Mapping," [Online]. Available: WiseGEEK, http://www.wisegeek.com/what-is-data-mapping.htm. [Accessed 10 January 2013].

[4] W. Vogels, "All Things Distributed"," [Online]. Available: http://www.allthingsdistributed.com/2008/12/eventually_consistent.html. [Accessed 6 January 2013].

[5] "Filtering Synchronization Data," MSDN, [Online]. Available: http://msdn.microsoft.com/en-us/library/bb902843(v=SQL.105).aspx. [Accessed 2013].

[6] "Flow Control," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Flow_control_(data). [Accessed 2013].

References

1] Takai, "Cloud Computing Strategy," DOD CIO, July 2012 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA563989)
[2] Vietmeyer, "DoD Cloud Computing Strategy," 7th Ann. DoD Enterprise Architecture Conf., May 2012 (www.dodenterprisearchitecture.org/program/Documents/Vietmeyer%20DOD%20EA%20Conference%20--%20Cloud%20Computing%20Presentation.pdf)
[3] DISA, "State of C2 Evolution," May 2012 (http://www.disa.mil/News/Conferences-and-Events/DISA-Mission-Partner-Conference-2012/~/media/Files/DISA/News/Conference/2012/State_C2_Evolution.pdf)
[4] DISA, "C2 Way Ahead: Joint C2 Objective Architecture," 15 August 2011 (https://www.us.army.mil/suite/files/30184579)
[5] DISA, "Joint C2: Situational Awareness & Intel Global Command and Control System– Joint (GCCS-J)," May 2012 (http://www.disa.mil/News/Conferences-and-Events/DISA-Mission-Partner-Conference-2012/~/media/Files/DISA/News/Conference/2012/JointC2SituationalAwareness.pdf)
[6] DISA, "DoD ESI & The Joint Information Environment (JIE)," DISA Mission Partners Conf., May 2012 (http://www.disa.mil/News/Conferences-and-Events/DISA-Mission-Partner-Conference-2012/~/media/Files/DISA/News/Conference/2012/DoD_ESI_JIE.pdf)
[7] Pontius, "Coalition C2/Multinational Information Sharing: Current Capabilities and Challenges," ICCRTS, June 2011 (http://www.dodccrp.org/events/16th_iccrts_2011/plenary_presentations/Pontius.pdf)
[8] Cabrera, "System Level Architectures & the Mission-Level Systems Engineering Process," 7th Ann. DoD Enterprise Architecture Conf., May 2012 (www.dodenterprisearchitecture.org/program/Documents/CabreraRicardo_T2_@1530Wed%20(vs3).pdf)
[9] Cabrera, "Mission-Level Systems Engineering," 6th Ann. DoD Enterprise Architecture Conf., April 2011 (http://www.dodenterprisearchitecture.org/pastmeetings/Documents/RicardoCabrera.pdf)
[10] Garcia, P.,"Command and Control Rapid Prototyping Continuum (C2RPC): The Framework for Achieving a New C2 Strategy," 2011 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA546926)

[11] Garcia, "Maritime C2 Strategy: An Innovative Approach to System Transformation," ICCRTS, 2010 (http://www.dodccrp.org/events/15th_iccrts_2010/papers/147.pdf)
[12] Gizzi, "Command and control rapid prototyping continuum (C2RPC) transition: Bridging the Valley of Death," Proc. Eighth Ann. Acq. Res. Sym., 2011 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA543946)
[13] Morris et al., "Widget and Mobile Technologies a Forcing Function for Acquisition Change: Paradigm Shift Without Leaving Bodies Behind," Proc. Ninth Ann. Acq. Res. Sym., 2012 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA563537)
[14] Siordia and Zimmerman, "Engineering and Acquiring C4ISR Systems Based on SOA," ICCRTS, June 2010 (http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA525199&Location=U2&doc=GetTRDoc.pdf)
[15] Durham et al., "Enterprise Engineering Reference Model," AUVSI Unmanned Systems Interoperability Conference (USIC), 2011 (presentation not available online)
[16] Durham and Daswani, "Enterprise Lexicon Services (ELS): Pilot Tools and Web-Services for Acquisition and Mission Support," DoD Enterprise Arch. Conf., May 2010 (www.kzoinnovations.com/afei/ppt/durham.pdf)
[17] Hogan et al., "Nist cloud computing standards roadmap," NIST Publication, 2011 (http://www.cloudindustryforum.org/downloads/standards/NIST_CCSRWG_092_NIST_SP_500-291_Jul5.pdf)
[18] Newell and Levasseur, "Shared Information Access Services in SWIM Segment 2: An Architectural Assessment," Project Report ATC-383, Oct 2012 (https://www.ll.mit.edu/mission/aviation/publications/publication-files/atc-reports/Newell_2012_ATC-383_RP-3827_WW-25258.pdf)
[19] Gillette, "Cloud Computing and Virtual Desktop Infrastructures in Afloat Environments," NPS, June 2012 (http://calhoun.nps.edu/public/bitstream/handle/10945/7349/12Jun_Gillette.pdf?sequence=1)
[20] Reed et al., "Supporting Agile C2 with an Agile and Adaptive IT Ecosystem," ICCRTS, 2012 (http://www.dodccrp.org/events/17th_iccrts_2012/post_conference/papers/044.pdf; http://www.dodccrp.org/events/17th_iccrts_2012/post_conference/presentations/044.pdf)