HÖGSKOLAN SKÖVDE

Institutionen för kommunikation och information
Examensarbete i datavetenskap 45hp
Avancerad nivå
Vårterminen 2010

Evaluation of the effect of stabilization time in eventually consistent systems

Mac Svan

Evaluation of the effect of stabilization time in eventually consistent systems

Submitted by Mac Svan to the University of Skövde as a dissertation towards the degree of M.Sc. by examination and dissertation in the School of Humanities and Informatics.

Date

I hereby certify that all material in this dissertation which is not my own work has been identified and that no work is included for which a degree has already been conferred on me.

Signature: 

Evaluation of the effect of stabilization time in eventually consistent systems

Mac Svan

Abstract

The effect of using the eventual consistency model is evaluated, compared to the effect of immediate consistency, by increasing the stabilization time in both models and using the immediate consistent system as a baseline for evaluations. The immediate consistent system performs better if the information and the decisions are replicated adequately fast throughout the system. When the stabilization time increases the effectiveness of eventual consistency emerges, which is most obvious when time constraints make it difficult to propagate information and decisions.

By using a simulation to extract data for evaluations, it is verified in this research that as the stabilization time between different parts of a system increases, the eventual consistency will always outperform the immediate consistent system. This statement is valid in all situations where consistency models are useful.

Of secondary importance in the research, by adding more decision layers to the eventual consistency model, the performance output is increased significantly, as swift and less well calculated decisions can be thwarted and corrected using the second decision layer.

Key words: Eventual consistency, immediate consistency, replica update completion time, autonomous decision making, layered decision making.

Contents

1 Introduction

Ensuring consistency when replicating data is essential and when using the eventual consistency model (Birrell, et al., 1982; Andler, et al., 2000; Syberfeldt, 2007) all nodes in the network strive towards consistency. During updates, the network is temporarily in an inconsistent state, and under these periods nodes can read tentative information (i.e., not yet agreed upon) from other parts, which increases the availability (Gustavsson & Andler, 2002).

In order for the eventual consistent system to reach consistency, it must have the ability to solve conflicts and integrate propagated updates into its system (Syberfeldt, 2007, p.34). If no updates occur in the network for a while, and the conflict resolution manager solves the potential problems, the network eventually reaches a consistent state where all nodes share the same replicated data.

The research on the eventual consistency is sparse and this thesis aims to evaluate if this model can be beneficial as compared to other consistency models. Immediate consistency, which avoids conflicts by ensuring isolation during update (Sheth, et al., 1991), is used as a baseline for the evaluations. Data for evaluations is extracted in a simulator, running a ground based air defense scenario. Nodes, in this case missile defense systems, form decisions and communicate with each other. A threat evaluation and weapon allocation process decide which missile defense system should fire on which aircraft (Roux & van Vuuren, 2007). As the belligerents on the battlefield moves rapidly, the effect of a long stabilization time in the network affects the missile defense system performance.

As an evaluation of the effectiveness of eventual consistency has not been performed in any research, it serves as the main motivation for conducting this research and writing this thesis. The hypothesis used in this work predicts that when the time it takes for an update to propagate throughout the system increases, the eventual consistent system will outperform the immediate consistent system. It is also assumed that the positive effects of eventual consistency emerge quicker in environments where the systems response time is limited, such as in air defense scenarios.

The rest of this thesis is structured as follows. Firstly, chapter 2 describes the background leading up to the actual research question. This includes distributed processing, consistency models in distributed systems, and also ground based air defense with threat evaluation and weapon allocation. Secondly, chapter 3 describes the purpose of this work, stating the research question as well as forming a hypothesis around this. Motivations for conducting this research as well as the objectives are also lined out. In chapter 4, the method for proving the hypothesis is decided, and the choice falls on simulating the consistency models. Chapter 5 details the construction of the simulation and scenarios as well as running actual simulations accompanied with preparing and presenting the results. Finally, in chapter 6 the thesis is summarized including contributions and future work.

2 Background

This research aims to evaluate the effect of acting on tentative information in eventually consistent systems compared to acting on stable information. This evaluation is done in the scope of distributed systems, where each node in the system has the ability to act without knowledge of the network topology or specific information sent from other nodes.

The background to the research question and the purpose are outlined in this part of the work, starting in section 2.1 by explaining the details of a distributed processing followed by the central parts of consistency models, explained in 2.1.4, which are being used in distributed systems. In section 2.2 this is further linked together with the ground based air defense scenario, and essentially how to analyze threats and allocate weapons which is described in section 2.4.

2.1 Distributed processing

A system using *distributed processing* is a collection of autonomous physical nodes all connected over a network where each of these nodes have a local memory and the means to create new information based on input (Kent, 1987, p.17; Coulouris, et al., 2005, p.1-7). In order to understand the terminology of distributed processing, the single *computer system* is one physical node which has the ability to execute *threads* serving as the operating systems abstraction of an activity. One or more threads can form a *process* (Coulouris, et al., 2005, p.228). Spanning over multiple physical nodes, the collection of the processes is thus referred to as distributed processing whereas all the physical nodes together is a *distributed system*.

The following section explains multisensory data fusion and what we gain from using decentralized data fusion, in section 2.1.1, which is used in distributed systems. This is followed by section 2.1.2 and what advantages shared resources presents and what this distributed system needs to function properly; data replication in section 2.1.3, and control of transactions between the nodes and securing that the entire network is consistent in section 2.1.4.

2.1.1 Decentralized data fusion

An increase in accuracy and robustness can be achieved by using multisensory data fusion (Hall & Llinas, 1997). Readings from different sensors combined can decrease or eliminate errors made from faulty readings (e.g. one out of ten sensors showing significant readings). Should one sensor fail, other sensors can be used in its place. By combining different sensor readings, new information can be extracted in the data fusion process. For example, by using only one acoustic sensor it would not be possible to detect a sounds origin, but by using three connected acoustic sensors it is possible to triangulate the position. This can be achieved by the use of data fusion (Brännström, 2004).

By using decentralized data fusion architecture, the aim is to remove the fusion process as well as decision making from a centralized node or part of the network. Instead, each single node has its own sensors and decision process. According to Durrant-Whyte (2000), there are two other important issues in decentralized data fusion networks. First, there is no central communication process, so nodes cannot deliver a message to the entire network and be sure that everyone has received it. This is because of the second part, that the single nodes do not have any knowledge of the

network topology; instead they are only aware of the nodes that are directly connected to them. If there is a need to send messages to the entire network, then they have to rely on that the message will propagate through the network and that all nodes are reached eventually. There is however no way for the sending node to know if its message was successfully sent to all parts of the network.

Although this premise of using decentralized data fusion appears to be negative as the topology is unknown, there are benefits from this architecture that stands out clearly in this research. Primarily, the availability of the network increases (Haerder & Reuter, 1983; Grime & Durrant-Whyte, 1994). This means that nodes can fail while the network as a whole still remains functional. Additionally, nodes can dynamically be added to the network without the need to restructure it from the beginning. By decentralizing the decision process, called *decentralized control*, all nodes can independently form decisions and act upon them.

The entire system can be divided into a pure decentralized solution or as a hierarchical solution. There are some advantages and disadvantages with these approaches, explained by Grime & Durrant-Whyte (1994), as explained next:

- *Pure decentralized solution*. All nodes have the ability to distribute messages and decisions throughout the network, in all directions. The topology of the network is thus completely decentralized, meaning that it is possible to add and remove nodes. The downside is that messages do not travel in a predetermined way, but randomly throughout the network, possible leading to conflicts between the nodes. They can also amplify their own results by looping them back to themselves, something called *rumor propagation* (Zanette, 2002). In this way, they confirm a weak decision by getting the same information from another node, when in fact it is their own message bounced around the network.
- *Hierarchical decentralized solution*. Each node in the system can, like the pure decentralized solution, send messages and form decisions. However, in a hierarchical approach there are no loops as messages are passed along the network in a structured way. This increases the complexity of the network, as the logical hierarchy on top of the network takes time to configure, while it at the same time reduces the possibility of conflicts. If a node is added or lost, the structure of the network must be recalculated; at least some small parts but in some cases the entire network.

The reason for using a decentralized solution can be to increase availability, load balancing and performance (Haerder & Reuter, 1983; Gong & Aldeen, 1997). In this research, we are only interested in the benefits of increased availability. In defense applications the advantages over a centralized architecture becomes more apparent (Durrant-Whyte, 2000), which is important in this work and explained in more detail in section 2.2.

2.1.2 Distributed shared physical resources

In systems using decentralized data fusion each node has the ability to form decisions and act upon them. It is thus possible for one of these nodes to use its resources in aiding other parts in the network. This is called *distributed shared resources*. The resources here are physical resources, thus not processor time or the use of distributed shared computer memory. As an example, in a missile defense system node A, node B and node C are initially identical systems, each having the ability to fire missiles. If an

incoming threat approaches node B, which currently has lost the ability to fire its missiles, both node A and node C could use theirs to support node B.

According to Kee, et al. (2006), this distributed shared resource architecture can be divided into two major parts; *the resource selection* and *binding processes*. Firstly, resource selection is important for the network as it discovers and identifies the resources that are available to put into use. Secondly, when the availability of resources is known to the network, the binding process allocates resources to a specific task. In the aforementioned example, it is thus important for node A and node C to know that node B has lost its ability to fire missiles, so they can step in and take the shot.

There are three important advantages standing out in sharing resources, compared to using them locally without any knowledge of the other parts in the network (Kee, et al., 2005). Firstly, if a local resource is exhausted or for some other reason lacking the capability to handle a certain situation, then other resources in the network can aid in the task. Secondly, it is possible to gather power by using multiple resources upon on single task. Finally, if parts of the network fail, then it may still be possible for the remaining resources to cope with the new situation solving both their own tasks as well as the failed resources.

2.1.3 Replicating data

Since the nodes in the distributed system are decentralized but not isolated there is a need to replicate data among them (Pacitti, et al., 1999; Domenici, et al., 2004). This is done by sending information over the network. While it is not possible to instantaneously send this information, taking a short amount of time is preferably as the replication of information is crucial in decision making. By replicating important data, sensors readings, and decisions made by single nodes, the information can be used by the entire network. This improves the reliability in the network, meaning that it reduces the risk of losing information by storing it on separate locations. This process should run in the background, always replicating the data over the network (Syberfeldt, 2007). The main advantage of replicating data is that each single node thus obtains more information than it can perceive locally using only its own sensors to collect information.

The difference between backing up information on a remote storage and replicating data between nodes needs to be clarified; the first stores information over an extensive period of time, while replicating data occurs in a consecutive order in an uninterrupted manner. This also means the faulty information, such as an incorrect sensor reading, propagates through the network. Once this value is changed the replication process overwrites the previous state, thus losing all historical information in the network.

In this report the term *node* is used to denote a structure in the system. Replication of data does not necessarily only include nodes, but also works at other granularities of the network (e.g., variables and set of nodes). Instead of denoting this each time, hereafter this thesis uses the term node when discussing replicated data, consistency, locking, and other architectural aspects.

2.1.4 Concurrency control

As explained previously, a distributed system has the ability to share both resources and data. When information is replicated it is essential to be able to control the concurrency; if information passed along nodes indeed should be transferred to all

parts in the network, then the information have to stay intact and not interfere with other information being sent. Since we have replicated data over the network, we must have some type of concurrency control to solve this issue. In this sense, *concurrency* means that multiple parts (e.g., tasks, nodes) replicate its information concurrently to other parts of the network (Bernstein, et al., 1987, p.III). If, for instance, the sensor readings in two nodes update at the same time and propagate through the network, then there could be a conflict. In order to avoid this problem, *concurrency control* is used to make sure that the correct information is passed along the network without conflicts, and also that the nodes can collect the new information without complication (Thomasian, 1998).

When, for example, nodes share activities by communicating with each other, we can talk about a *transaction* which is typically a sequence of interactions between databases (Kung & Robinson, 1981). In order for a transaction to succeed it must complete as a whole entity and cannot be left in an intermediate state of inconsistency. Haerder & Reuter (1983) describes paying with a credit card as a single transaction; even though there are multiple messages between the card and the bank (e.g., is it the right pin code and is there money on the account), the unit as a whole is one transaction. If something unexpectedly occurs while paying with the credit card, violating the rules of the bank, the entire transaction is aborted and the updated variables reverted back to the state prior to the start of the transaction. Thus, in this example, it is impossible to buy something with a credit card while at the same time still having the same money remaining on the account.

Following the rules of ACID, a term coined by Haerder & Reuter (1983) and described in detail below, the transaction process in the network is done as reliable as possible. *ACID* is an acronym for atomicity, consistency, isolation, and durability, and each of these parts must be fulfilled in order to ensure correctly passing of information.

- *Atomicity*. The effects does only come into play if the entire transaction succeed; if there are missing parts, inconsistencies or other inaccuracies violating the consistency, then the transaction fails and is thus being rolled back to a previous stable state. In other words, the transaction can either succeed or fail following an all or nothing-rule.
- *Consistency*. The transaction takes an affected part in the network from one consistent state to another. This means that the system should not violate the rules of the network by leaving it in an intermediate state of inconsistency. If the transaction cannot ensure that the system is placed in a correct state, it should be rolled back to its initial state.
- *Isolation*. The goal of the concurrency control is to ensure isolation. During the transaction the data can be locked in an isolated state meaning that other parts cannot send or view the affected parts. Two simultaneously executing transactions should thus not be able to interfere with each other. This eliminates send conflicts, but also reducing the risk of another part in the network reading partly updated information. There are different degrees of isolation, and this work use the strong strict two-phase isolation, explained below.
- *Durability*. If the transaction is complete, it should be durable and fault-tolerant. Even if the system fails, the effects of the transaction should remain intact.

There are different approaches to achieve concurrency control, with two primary methods; optimistic and pessimistic control, as described in Figure 2.1. By using *optimistic concurrency control* the operation is carried out and the affected parts are afterwards validated before the transaction can take effect. The *pessimistic concurrency control* ensures that the transaction is valid before commencing. This is done by locking down all the affected variables as a part of the validation ensuring that the coming operation is possible to conduct and that it will not end by leaving the transaction in an invalid state. Once this validation is completed, the operation commences and the transaction takes effect. Finally, the affected parts are unlocked.

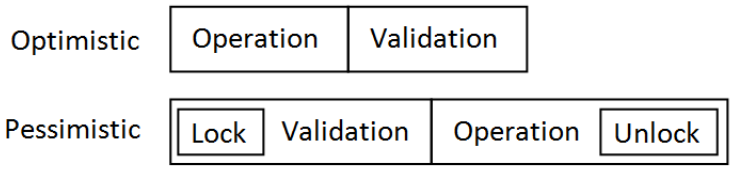


Figure 2.1. Locking process of optimistic and pessimistic concurrency control.

The pessimistic approach is both easier to construct and use, but as problems in availability became apparent the optimistic approach was developed by Kung and Robinson (1981), and it should perform better in most situations, especially where conflicts rarely occur.

As seen in Figure 2.2, conflicts are handled differently depending on the consistency model in use. If all parts affected by the transaction are locked in the initial validation phase, as done in the pessimistic approach, no conflicts can occur. When using the optimistic approach, with no locking, conflicts are still possible. Once a conflict occurs, the system can roll back to a previous valid state or solve the conflict.

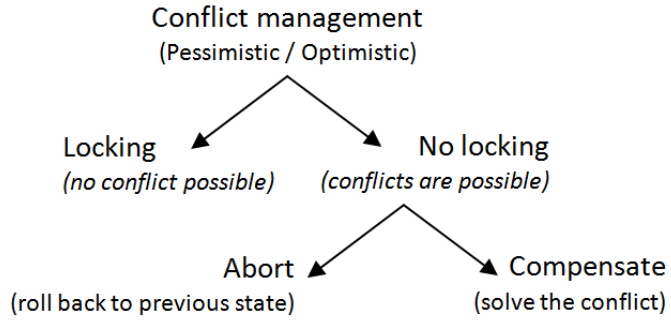


Figure 2.2. Conflict management using locking or no locking.

Choosing a method for concurrency control depends on the task, and it could be possible to use both depending on specific situations (Kung & Robinson, 1981; Atkins & Coady, 1992). The optimistic and pessimistic concurrency control, and how they handle conflicts, is detailed below.

- *Optimistic concurrency control*. The optimistic approach is that even if conflicts occur we have the ability to handle or tolerate them. This means in practice that the operation in the transaction is done without any locking or validation process and that other parts in the network remain the ability to both read and write to the affected part simultaneously. Optimistic concurrency control thus uses a low isolation according to the ACID rules. Once the transfer is complete the affected parts are validated, as shown in Figure 2.1. If a conflict occurs in this process it is handled here, usually by rolling back to a previous state or compensate for the conflict in some way, as seen in Figure

2.2. Using optimistic concurrency control is preferable when conflicts seldom occur (Atkins & Coady, 1992, p. 191), which is the case when there is a low update frequency or transactions complete in a short amount of time. If something violates the consistency rules at the end of the transaction and it is not possible to compensate for this, all updates are discarded (Haerder & Reuter, 1983).

- *Pessimistic concurrency control*. The pessimistic approach is based on avoiding conflicts. Thus, the parts affected by the transaction are locked down in the beginning, and initially there is a validation ensuring that it is possible to conduct the operation. This ensures that the ACID is obeyed, and as seen in Figure 2.1 the lock is released in the end of the transaction. With the pessimistic approach other parts are completely locked out from reading or writing during the transaction. This is, in contrast to the optimistic approach, useful when conflicts do occur and the overhead of using the pessimistic approach does not exceed the performance loss of conflict resolution (Atkins & Coady, 1992, p. 191).

Finally, choosing a consistency model depends on the task. As pointed out, it is preferably to use an optimistic approach if conflicts rarely occur (Kung & Robinson, 1981; Atkins & Coady, 1992, p. 191). Optimistic approaches are also more cost-effective, while pessimistic are more robust, thus you trade off concurrency (i.e., reliability) for availability by using optimistic concurrency control, and vice versa (Herlihy, 1990).

2.2 Consistency in distributed systems

Following the ACID rules, explained in 2.1.4, once the concurrency control in the distributed network is ensured there is a need for replica consistency. There are multiple models for achieving consistency. They all share the same goal to agree upon the effects of an update once it has been propagated out in the network among the nodes.

This section initially covers the consistency maintenance problem, in section 2.2.1, followed by the concept of replica update completion time, in section 2.2.2. The consistency maintenance problem foreshadows two primary consistency models being used in this thesis, detailed in section 2.2.3 and 2.2.4. Section 2.2.5 concludes with the topic internal consistency and temporal validity.

2.2.1 Consistency maintenance problem

In this research a distributed system is being used, described in section 2.1, and this system has multiple connected nodes which in between data are being replicated, described in section 2.1.3. When replicating data there is a need to control the concurrency, described in section 2.1.4, and once this is ensured, consistency among the replicas needs to be addressed; which is the *consistency maintenance problem*, addressed by Sun and Chen (2002) as *consistency maintenance*.

When a node decides to update itself and new data is propagated throughout the network, problems with the consistency between replicas must be handled. Reaching consistency between nodes depends on the architecture and topology of the distributed system as well as the chosen consistency model. A basic example illustrated in Figure 2.3, four nodes initially share the same data. The connections between the nodes are not illustrated here, but all nodes are fully connected to each

other and they have two variables (A and B) which, once updated at a single node, need to be replicated to all nodes. In the second step in this example, node 3 changes the value of its variable B from 2 to 6. This new value is thus propagated to the other nodes and after some time all nodes has been updated, as seen in the last step in the figure.

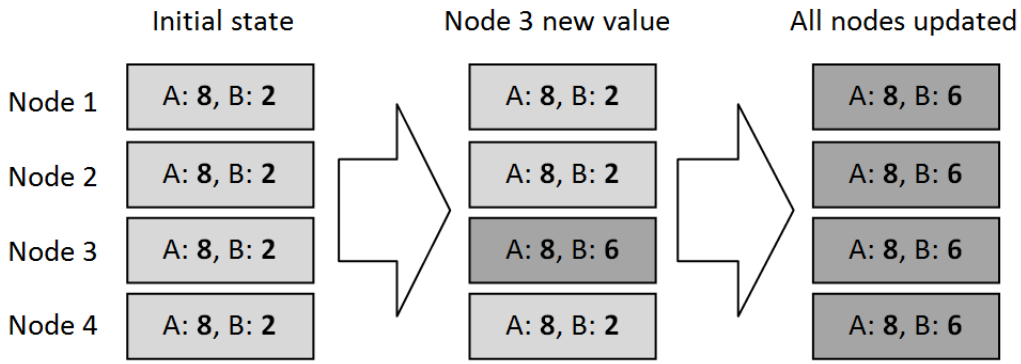


Figure 2.3. Illustrating the concept of consistency among nodes.

The goal in Figure 2.3 is to propagate the update, and consistency is met once all nodes in the network agree upon the effects of that update. However, during the update multiple problems need to be addressed. As in this example with only four nodes, and if updates in the network seldom occurs, updates can be propagated with relative ease once they occur. On the other hand, if there are several nodes, sparsely connected and updates occur frequently, it could be more problematic to maintain the consistency in the system (Sun & Chen, 2002).

Illustrated in Figure 2.4, the update is seen as a process moving along a timeline. The present time in the figure illustrates in what range the current information is useful; i.e., when the information still is up to date and relevant. As seen in the figure, the present is still relevant once the effects have been agreed upon by all nodes (i.e., all nodes can make use of the update).

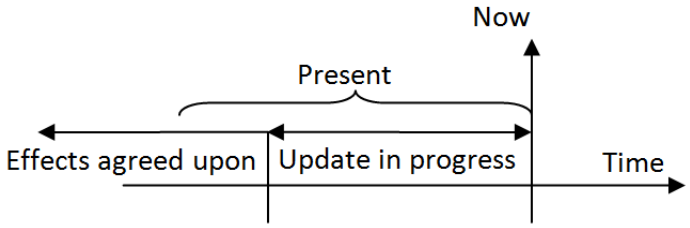


Figure 2.4. Conceptual illustration of relevant present information.

If the update takes too long (e.g., due to slow network speed, or if the present only is relevant in a short time frame), once the effects have been agreed upon this newly updated information is obsolete before it actually comes into effect. This is illustrated in Figure 2.5. Depending on location of the nodes the present can differ in different parts of the system. When the difference in the present does have an impact depends on the specific system.

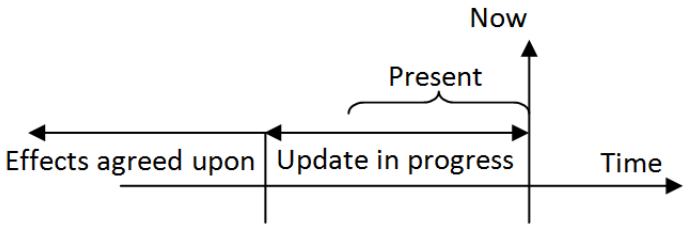


Figure 2.5. Conceptual illustration where the update takes too long.

There are multiple consistency models available which approaches the consistency maintenance problem, and in this work two models are of primary focus; eventual and immediate consistency. Eventual consistency allows for conflicts since there is a way to handle them, and the immediate consistency avoids conflicts. Both these models are detailed in section 2.2.3 and 2.2.4, after replica update completion time, a common term used in both models, has been described in the following section.

2.2.2 Replica update completion time in a consistent system

The unifying term *replica update completion time*, using the acronym RUCT henceforward, is specifically the time it takes from the point where one update starts until the effects of that update is replicated and agreed upon in the network. It is used to measure the time it takes to complete an update in a consistency model.

The RUCT depends on a wide variety of factors and it beneficial to decrease this time as much as possible. Different aspects which affects the time it takes to reach replica consistency are explained below.

- *Size of network*. The number of nodes in the network is of uttermost importance for the RUCT. By adding nodes to the network the time it takes to complete a transaction will always increase.
- *Topology of the network*. If the network is sparsely connected it takes a longer time to ensure isolation in the network, thus increasing the RUCT. If the topology changes (i.e., connections between nodes, or nodes being dynamically added or removed) it will increase the time further.
- *Committing speed*. The time it takes for data to be sent between two nodes is a factor. It is not possible to make this go faster than the speed of light, but it could go much slower than that optimal speed. The time it takes for the isolation, validation and other processes to come into effect decreases the transfer speed and increase the overall time.
- *Amount of conflicts*. The more conflicts that occurs, the more operations will have to be rolled back or solved. This also depends on the concurrency approach used, but when conflicts occur it does add to the overall time.
- *Locking of variables*. If variables need to be locked in order to ensure isolation, this adds to the overall time, a process which is highly dependent upon the size and topology of the network.

At some point the RUCT could be seen as negligible if the only factors delaying the time are those of the speed of light. As mentioned above, the real RUCT is however usually limited by more than the speed of light. What could be seen as a negligible or a problematic time delay all depend on the demands of the specific scenario. If the system need to respond within a certain time frame then the RUCT need to land within that frame. As an example, in a scenario where the effects of an update is

relevant for 10 seconds and the RUCT rarely takes longer than 1 millisecond, the actual delay caused by the update completion time between replicas is negligible. However, should the RUCT usually take between 5 and 15 seconds, then the delay is indeed problematic.

2.2.3 Immediate consistency

Using the consistency model *immediate consistency*, the goal is to avoid conflicts. Once the update has been properly propagated, if the RUCT is less than the present time frame the updated data is useful, as described in section 2.2.2 and illustrated in Figure 2.4.

When using optimistic concurrency control, described in section 2.1.4, with immediate consistency the validation occurs during the update. In the immediately consistent system tentative information is not useful, so during the update process in Figure 2.4 the partially updated data cannot be used for decision making. During the update the data is validated. If anything would leave the system in a conflicted state (prohibited in immediately consistent systems) the update is aborted (Datta & Son, 2002).

Pessimistic concurrency control, described in section 2.1.4, applies locking on parts affected by the update. During the validation process, locking ensures isolation and guarantees that no conflicts can occur. Parts affected by the locking are isolated from other parts of the network. At the end of the update the locks are released and all parts affected by the update have agreed upon the effects of it.

There are both advantages and disadvantages following this approach of consistency. Conflicts never occur and information extracted from a node is always the same no matter where the extraction is made. If the network can reach consistency in a short or negligible amount of time, then this consistency model is useful. However, in most situations it is difficult to achieve these results as some parts in the update process always takes time (Gustavsson & Andler, 2002). This is also the disadvantage; the tentative data during updates cannot be used, and since parts are isolated during update the availability in the network decreases.

As a consequence of using immediate consistency the system is always in a stable state, where no tentative data can be used and no conflicts can occur. As the network base its decision upon stable data agreed upon in all parts of the network, it ensures that resources in the network used in such an effective way as possible (Haerder & Reuter, 1983), given the prerequisite that the RUCT is less than the present time frame.

2.2.4 Eventual consistency

Using the consistency model *eventual consistency*, the goal is to increase availability which is done by allowing conflicts. Eventual consistency (Birrell, et al., 1982; Andler, et al., 2000; Syberfeldt, 2007) ensures that the nodes in the network all strive towards consistency and that it will be reached if no update occurs for a while in the network (i.e., the actual time it takes depends on the topology of the network, as well as other aspects). During the update process, data can be read tentatively which increase availability.

The consequence of using eventual consistency is that there are two types of values which can be extracted from variables; tentative and stable. The tentative values exist during the update process, whereas they become stable once the effect has been

agreed upon by all nodes. This means that the tentative values are more recent (or equally recent) as the stable values. If there are available stable values, they can preferably be used. If the RUCT takes longer to complete than the usefulness of the data being replicated, tentative data can be used.

According to Syberfeldt (2007, p.34), it is possible to prove that the eventual consistent network strives towards and can reach consistency if it can handle conflicts. If more than one update is being processed simultaneously affecting the same parts, there is a conflict that needs to be solved before the data can become stable. As a possible solution, a time stamp can be included in all updates. The oldest of the conflicted update is thus discarded, and the conflict has been solved. Since the network of nodes consists of multiple autonomous units, this requires that all internal clocks are synchronized; especially if updates are frequently occurring. If it is not possible so solve a conflict, the entire update can be rolled back to its previous state. Once the effect of an update has been agreed upon in the entire network, the data is considered stable. As new updates already can be underway, there is no way that a single node can be sure that it has the most recent data; just that the current data is stable at the moment.

Like using immediate consistency, there are both advantages and disadvantages of using eventual consistency. An eventually consistent system can act directly, regardless of which state it is in, using the tentative information available to assess the situation. This ensures that system can act, regardless of the RUCT and time span when the data is useful, thus making it more available than the system using immediate consistency. On the downside, if RUCT is small and the window where the data is useful is large, it is better to act on stable data. If the system acts on tentative information, which is possible using eventual consistency, it could negatively affect the decision making.

There are three ways of running applications based on eventual consistency; acting on the stable, the tentative data or use them both. The first method which uses stable data differs from immediate consistency in that the parts in the process of being updated remains available.

The second method is acting on tentative data. As soon as the data starts to propagate through the network nodes actively use the new information. This leads to more hasty decisions; something that could be seen as beneficial or problematic all depending on the specific scenario.

Finally, the third method primarily uses the stable data, but if the RUCT during an update increase beyond the point of usefulness, the system can act on the current tentative data propagated so far. Thus the system can act on both tentative and stable information (Syberfeldt, 2007, p. 72-73).

2.2.5 Internal consistency and temporal validity

It is important to differentiate between *internal consistency* and *temporal validity*. The former handles rules internally in the nodes making sure no invalid writing occur and the latter controls consistency between nodes (Ramamritham, 1993a). Temporal validity is also called *external consistency* (Lin & Peng, 1993), a terminology that perhaps easier relates the difference between the two but as Ramamritham uses the term temporal validity in his state of the art article from 1993, that term is used in this thesis work.

Internal consistency controls values written to a specific part of a node ensuring that rules set up are being followed. If a node, as an example, keeps track on colors of cars, the data is internally consistent should the value be, for instance, “green” or “red”. The internal consistency rule prevents the value “house” to be written internally, as it is not a color.

As the research conducted in this thesis work uses nodes acting in real-time there is a need to maintain consistency between them and the actual physical environment (Ramamritham, 1993b). Here the temporal validity is being used to ensure that the information passed is consistent and preferably up to date and still valid. Temporal validity differentiates between absolute consistency and relative consistency.

Explained from the point of view in this work, *absolute consistency* is the connection between the environment the network is operating in and one of the nodes. A timestamp can keep track if the current information in the node is valid at the moment. The *relative consistency* is maximum time between two different readings used to control the same process. Two nodes may be temporally valid, but if the duration between the readings of these values is above the threshold for the relative consistency, the reading is not valid.

By using eventual consistency acting on tentative data the relative consistency between nodes can never be assured, as where immediate consistency and eventual consistency which acts on stable data can globally ensure relative consistency. The absolute consistency is irrelevant to the use of consistency models, as it only serves as the connection between the nodes and the environment. In this research is always assured that local nodes are fully aware of its surroundings. It is important to notice that the concept of temporal validity can be complicated, as explained by Ramamritham, but as this work focus on the evaluation of eventual consistency it is safe to assume that both the internal consistency as well as the rules of temporal validity never can be violated.

2.3 Ground based air defense scenario

In order to ground this work by such realistic means as possible the effects of eventual consistency is studied and evaluated in a ground based air defense scenario based on a distributed system. This section initially describes general information about air defense scenarios, and section 2.3.1 describes the use of them in distributed systems. Section 2.3.2 covers limited resources and finally section 2.3.3 which details the decentralized weapon control.

Since it is not feasible to conduct this research using real aircrafts and surface-to-air missiles it is simulated using a computer program. This has been shown to be a feasible way in related works (Nguyen, 2002; Johansson & Falkman, 2009b). However, in this research the focus is not on perfect and flawless military realism, only that it is a believable scenario. A real military scenario involves an overall more sophisticated approach to the entire situation and it can be really advanced to construct in a realistic way; especially the psychologically effect that is involved in combat (Bowley & Lovaszy, 1999). Here the focus instead lies on the effects of eventual consistency and by keeping this study simple and effective (i.e., possible to compare and contrast) it is possible to extract useful data while at the same time eliminating all parts that could make the construction of a scenario difficult.

In a ground based air defense scenario there are different types of units. The terms used in this work are as follows:

- *Aircraft*. Could be a hostile, neutral or friendly aircraft. Aircrafts in this research do not engage in aerial combat, thus only attacking ground units. Could also be referred to as *fighter*.
- *Defended asset*. A defended asset is a ground based unit or structure of importance, usually a military installation. It is not important to specify what type of asset it is; only that is has substantial value and should be defended from incoming aircrafts. The consistency model is evaluated by the number of remaining defended assets at the end of each test scenario. Could be referred to as only *asset* in the right context.
- *Missile defense systems*. A system equipped with missiles, personal and communication with other missile defense systems. While this system could be divided into several smaller parts such as radar stations, it is on purpose kept simple in this study. It has thus everything needed for thwarting an enemy attack in the simulations; the ability to gather information, forming a decision and launching surface-to-air missiles to intercept aircraft. Could be referred to as *battery*, *missile defense* or, when in the right context, just *defense*.

2.3.1 Using air defense scenarios in a distributed system

To the author's knowledge, prior work in the area of ground based air defense scenarios does not take into consideration that there is a delay in synchronizing the missile defense systems for counter attacks. The reason for this could be that it is usually not constructed as a distributed decision process. If it indeed is a distributed process, the reason excluding the delay time could be that the synchronization process does not have such a huge impact on the situation. However, taking the speed of the aircrafts into consideration it is more likely that the delay then has been excluded because of the complexity it adds to an already advanced scenario.

Primarily there are two reasons for choosing a ground based air defense scenario to evaluate the sought after effects of stabilization time in different consistency models:

1. Time constraints makes delays critical, and the time it takes to reach a decision among the missile defense systems could be a real problem, especially in situations with multiple aircrafts and defense systems (Johansson & Falkman, 2009a; Naeem, et al., 2009). Acting on tentative information clearly stands out as a possible strategy in situations where time constraints make it difficult to stabilize the network in a realistic and reasonable timeframe.
2. Should the network of missile defense system not act properly or within a certain time frame, the results is measureable in a concrete way as defended assets indeed will be lost during attacks. In a specific scenario consistency models will perform in different ways and it is thus possible to compare them by measuring a gain-loss ratio of the battlefield resources. This ratio is also variable as the replica update completion time differs between scenarios.

2.3.2 Limited resources

Surface-to-air missiles are both expensive and in a limited supply in reality. While the defended asset have a higher value and are more important than the missiles, it should be noted that there is a limited amount of defending resources (Johansson & Falkman,

2009b). Therefore it is crucial that the amount of resources is taken into consideration when constructing the scenarios and evaluating the results.

2.3.3 Decentralized weapon control

The missile defense system should always be able to fire upon incoming aircraft. If the network loose all communication, which is a possibility in a real scenario, it is imperative that single defense systems still have the ability to act (Roux & van Vuuren, 2007). Since the network is decentralized all nodes have that ability. When the network acts on tentative information the defense systems will always make use of its decentralized weapon control. The system waiting for a consistent state in the entire network will instead fire once it has been decided upon which nodes in the network should fire upon which incoming aircrafts.

2.4 Threat evaluation and weapon allocations

Decentralized ground based air defense systems make use of the process of threat evaluation and weapon allocation to determine which missile defense system should fire. While threat evaluation and weapon allocation can be used in different scenarios, the focus in this thesis lies in that area (i.e., missile defense system), as explained in section 2.2. Thus, the process involves assessing the nature of the incoming aircrafts and if needed assigning proper countermeasures to deal with the situation (Roux & van Vuuren, 2007; Johansson & Falkman, 2008; Naeem, et al., 2009). A common acronym for threat evaluation and weapon allocation is TEWA, which henceforward is used.

Section 2.4.1 covers the treat evaluation in air defense and section 2.4.2 cover the weapon allocation, the two parts of TEWA. Section 2.4.3 details this process in a decentralized manner and section 2.4.4 cover the concept of decentralized layered decision making.

2.4.1 Threat evaluation

According to Roux & van Vuuren (2007), the term threat evaluation can be interpreted as different types of concepts and needs to be defined for this thesis. While threat evaluation can be included as a part of the Intelligence Preparation of the Battlefield process, the threat evaluation we are interested in here is the more direct form of threat an incoming aircraft poses to a defended asset.

When assessing the threat of an incoming aircraft, many factors come into play and have to be weighed against each other. By splitting the threat evaluation in two parts, *intent* and *capability* (Nguyen, 2002; Paradis, et al., 2005; Roux & van Vuuren, 2007) it is easier to assess the situation. As an example, a fourth or fifth generation advanced multirole fighter poses a great threat towards enemy forces due to its high capability to inflict damage and its intent to do so. At the same time, while retaining the capability to inflict damage, the aircraft poses no threat to its own forces due to its non-hostile intent. Before any weapons could be fired upon incoming aircrafts, it is thus important to analyze both the aircrafts intent and capability. At some degree of threat, decided by experts in the field in a real scenario, the TEWA process moves on to the second step of allocating weapon resources and neutralizing the incoming target.

2.4.2 Weapon allocation

The second part of TEWA is the weapon allocation, where defense resources are being put into use if the threat evaluation indicates that it is necessary to intercept the incoming aircrafts. Once the decision has been made, the system should use an appropriate amount of resources to handle the situation as best as possible. This area has more substantial research than the threat evaluation part of TEWA (Ahuja, et al., 2007; Roux & van Vuuren, 2007), meaning specifically that there is a lot of research into single weapon target locking. However, public research in distributing resources among different missile defense systems is sparse (Roux & van Vuuren, 2007).

There are multiple approaches as how to handle threats and allocate weapon resources. In this research no missile defense system has a dedicated defended asset and all incoming threats are treated equal which means that any missile defense system can fire upon any incoming threat as long as it benefits the global situation (i.e., as many defended assets as possible are being defended). This is done by communicating between missile defense systems deciding which missile should be allocated to which aircraft, a method used to maximize inflicted damage and minimize resources used (Paradis, et al., 2005). If the communication takes too long the missile defense systems also retain the ability to act solo on the tentative information. As mentioned in section 2.2, it is not always simple coordinating counterattacks and it can be a time consuming process (Naeem, et al., 2009). Combining delayed decisions with the ability to act on tentative information is utilized to evaluate the effects of using eventual consistency as compared to immediate consistency.

2.4.3 Decentralized threat evaluation and weapon allocation

In a decentralized TEWA process, each missile defense system can calculate the threat posed to its defended asset and act to defend it without acquiring any other knowledge from defense systems in the area. There are a lot of research in the area of decentralized systems (e.g., Durrant-Whyte, 2000; Coulouris, et al., 2005), yet the specific decentralized TEWA process has not been mentioned in any research as of yet. Roux & van Vuuren (2007) writes a state of the art article about the research in TEWA without mentioning a decentralized solution.

Still it is important for this research to use this decentralized approach, but it is kept simple in concept. Whereas the regular TEWA process is formed in a central node and then instantaneously distributed among the missile defense systems, this research instead leaves the TEWA to each individual node. This brings forth the benefits mentioned in section 2.1, and should be put into contrast of the old approach to determine in which situations each of them stands out as superior.

2.4.4 Decentralized layered decision making

Using multiple layers of decision making can improve the overall defense capabilities. In this way, the primary decision can be discarded if a secondary decision is found to be superior. In order to be able to use the secondary decision properly the following two factors must apply:

1. The final action is delayed for a period while waiting for other information that has been formed and propagated throughout the network. The delay is neither too long, as it would delay any action, or too short as new information would not be received on time to form a secondary decision.

2. The primary plan of action does not require an instant response. The total waiting time must not work against the overall benefit of using multiple layers of decisions.

Pignaton de Freitas, et al., (2009) describes this process as a *negotiated decision-making process*. In four steps (depicted in Figure 2.6) the network of nodes form decisions and share them in order to end up with each node taking such an appropriate action as possible. First, each node forms a primary decision, declaring itself as a candidate to solve the situation or not. If the node considers itself a candidate it will, in the second step, propagate this information throughout the network. After some predetermined time, should no node declare itself as the candidate, the node best fit to solve the task is the node closest to the task even if it did not declare itself to the others (i.e., or some other algorithm can be used for selecting the best candidate). This ensures that some node always tries to solve the task, providing robustness to the protocol. Finally, if multiple nodes declare themselves as candidates, the node with the highest capabilities (e.g., most number of remaining missiles) assumes the task. While the term negotiated decision-making process sound like an active negotiation between nodes, no actual back-and-forth discussion occurs. Only information about decisions is being propagated through the network.

Using a layered decision approach is only useful in eventual consistency as the immediate consistent system always act on an update all parts in the network agree upon; thus, that is currently best option, and while a second decision could provide more useful information it is more likely that the delayed decision just uses up valuable time in immediate consistency. Delaying the decision does in fact have the same effect as a longer RUCT, so while it is theoretical possible that it provides better results, this is usually not the case.

In this work, relating to the threat evaluation and weapon allocation process, as aircrafts move in on the intended objective in form of a defended asset the missile defense system, working decentralized, form a decision on the local node. This decision is the primary cause of action considering the information known currently to the node and is called primary decision in Figure 2.6.

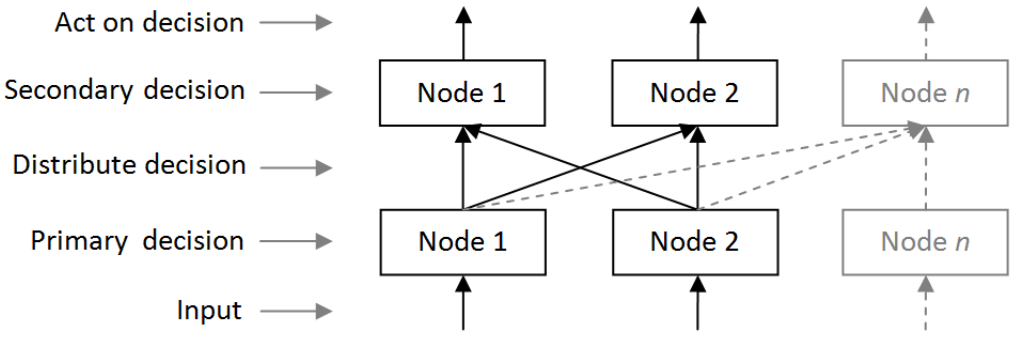


Figure 2.6. Illustration of layered decision making showing *n* number of nodes.

After each node has formed a primary decision they have the ability to act (i.e., a decision in this case can be to fire a missile or not to fire a missile). However, by holding the action and distributing the decision to the other nodes (and possibly receiving new information), they are given a second chance to come up with a better response to the incoming threat. By using this information they can form a secondary decision, which should be superior to the primary decision. After this decision is formed the nodes can take the appropriate action.

It should be noted that nodes using layered decision making bases their secondary decision on new information, and not only on decisions made from other nodes. While a primary decision propagated out in the network does have a huge impact on other nodes' decision making (e.g., a missile defense system declares that it will fire within seconds), all information received is used in the secondary decision. Once the secondary decision is finalized, the node acts on that decision.

Consider the following Table 2.1, where *IC* is immediate consistency, *EC* eventual consistency and *ECL* eventual consistency with layered decision making.

	IC	EC	ECL	IC	EC	ICL
Missile detected, battery #1	1000	1000	1000	1000	1000	1000
Missile detected, battery #2	1005	1005	1005	1005	1005	1005
Layered decision time	N/A	N/A	20	N/A	N/A	20
RUCT	10	10	10	1000	1000	1000
Battery #1 launches	1010	1000	1020	2000	1000	1020
Battery #2 launches	No	1005	No	No	1005	1025

Table 2.1. Six different situations using three different approaches.

In the table above, the immediate consistent system performs better than the eventual consistent system given a short RUCT. When the RUCT increases to t=1000 the immediate consistency, while still only launching one missile, takes an immense time to form the decision. In that situation, even when firing two missiles, using eventual consistent system could be the preferably choice as it actually acts within a short time frame instead of waiting for 1000 seconds. Finally, the eventual consistent system using layered decision making performs almost as good as the immediate consistent system when using a low RUCT, but it does not waste resources unless the RUCT is high, at which point it acts instead of waiting for replica consistency among the nodes. Times used in this example were selected to illustrate the effect and not reflect a specific actual event.

The layered decision making can act as a failsafe, whereas it is better to act than not act, but at the same time it is better to use more information when forming a decision instead of acting directly, which is the reason for the delayed secondary decision.

3 Problem definition

This chapter states the purpose, motivation and primary objectives for this thesis work.

3.1 Purpose

The purpose of this dissertation is to evaluate the effect of replica update completion time in an eventually consistent distributed system. The hypothesis in this work states that immediately consistent system with negligible replica update completion time will perform better than the system acting on tentative information. It is further thought, and should be demonstrated, that when the time it takes for the update to take effect at all replicas increases in the immediate consistent system, the benefits of the eventual consistency emerges, as seen in Figure 3.1.

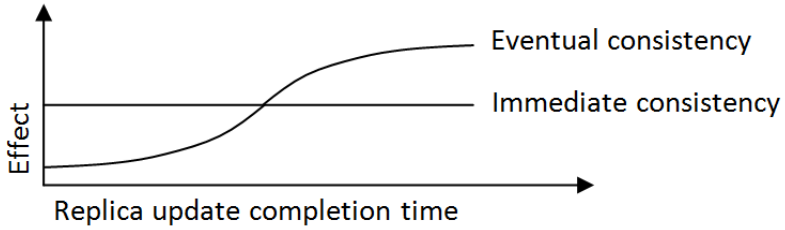


Figure 3.1. The hypothesis showing the effect of eventual consistency

3.2 Hypothesis

The hypothesis, from section 3.1, states that the benefits of acting on tentative information in an eventually consistent system emerge as the replica update completion time in the baseline system increases. The motivation for conducting this research is primarily to find out if the positive effects can be used and applied in a real world scenario, and it is thus important to show that the effects exist at all before starting to construct scenarios.

As this research evaluates the effects in an air defense scenario, the scenario also serves as the example showing that the hypothesis is valid. Naeem, et al. (2009) shows that their algorithm finds an optimal solution in what they call “a matter of seconds”, and while it is difficult speculate on the effects in such a short timeframe, Johansson & Falkman (2009a) shows that it could very well take close to an hour to construct a perfect countermeasure attack on incoming enemy aircrafts. While they also point out that it is totally unrealistic to wait that long for the perfect solution, this also serves as an example showing that in extreme situations acting on tentative information would stand out as the superior approach. Waiting an hour to act in an air defense scenario is the same as to not act at all. Adding more to the point, after an hour the aircrafts will be in completely different places so once the perfect solution has been calculated, it will still be unusable in the new situation. This motivates the hypothesis and when the RUCT is too long it is beneficial to use an eventually consistent system as compared to an immediate consistent system, as illustrated in Figure 3.1. This research investigates whether this point exists, and if it could be used in practice.

The graphical representation of Figure 3.1 is a hypothetical illustration showing that there exists a breaking point somewhere where the effects of one consistency model stand out as superior compared to the other. While this curve seems to be logarithmic

in structure, it could in fact be linear without affecting the hypothesis. This hypothetical illustration was constructed before any actual research was being conducted, and has since not been modified.

3.3 Motivation

An evaluation in the area of eventually consistent system acting on tentative information is, to the author's knowledge, non-existent at the moment. Therefore this serves as the primary motivation for conducting this research. Further, the effects should be applicable and grounded in a real world situation, as these consistency models are practical implementations and not abstract solutions to problems. This leads to the interesting aspect whether or not this research can be useful in real applications. Because of this the test scenario chosen is in the area of air defense since it is a realistic and highly relevant research area as well as a situation where the effects would stand out. The open research in the field of air defense is limited, partly due to the fact that governmental and private sector works has been classified (Roux & van Vuuren, 2007; Johanssson & Falkman, 2008), and the effects of eventual consistency have, as mentioned, not been studied at all making this research interesting and unique.

3.3.1 Motivation for choosing an air defense scenario

It is possible to evaluate the effects of eventually consistent systems in other settings than that of an air defense scenario. It is important to mention this; else it would appear that the research and use of eventual consistency is locked to the military domain. To name a few other examples, the effects are important in banking as concurrent transactions done from different places to the same account indeed can cause trouble. This also applies to stock trading if people try to buy the same shares at the same time as well as in traffic control. Another example, where the stabilization time is much longer, is the printed phone book which holds tentative information about people's numbers and its consistency cannot be guaranteed. While it is possible to choose any of these scenarios to evaluate the effects, the benefits of using an air defense scenario motivate the choice (as described in the background, in particular section 2.3.1).

3.4 Objectives

The objective is to find a way to evaluate the sought-after effects and be able to extract information to evaluate the eventually consistent systems. Five primary objectives must be addressed in order to reach the goal and fulfill the purpose and motivation, stated in section 3.1 and 3.3, and to validate or falsify the hypothesis stated in section 3.2.

3.4.1 Selecting evaluation method

The first objective is to clearly select a usable approach for evaluation and to motivate why it has been selected. In order to evaluate the sought after effects from section 3.1, we could choose to *analyze*, *simulate* or *experiment* to achieve results. Another approach would be to make a *literature survey*, but since there is no existing literature on the subject, at least not public open research (Roux & van Vuuren, 2007), it is not possible to evaluate the effects this way. Thus a literature survey can be ruled out in this early stage.

It is further possible to select multiple methods for the evaluation, for instance both analyze and simulate in order to achieve results.

The three presented evaluation methods above should be analyzed according to feasibility and output. It is important that the evaluation method can produce output useful for verifying the hypothesis, and the selection of evaluation method should result in the best possible approach for this research.

3.4.2 Tools used for the implementation

There are different approaches to implement the scenario, and the most important part is that it should be possible to evaluate the results (objective from section 3.4.1) with the selected implementation model. It has to be determined which tools and programs as well as computational power that are available for use in this research, and specific choices has to be motivated.

All possible approaches should be listed, where each details the benefits and downsides of being chosen as the primary evaluation method in this research.

3.4.3 Important parts and outlining the implementation

One of the motivations for conducting this research is that it is grounded in the real world, and that the results also can be applied and used in practical situations. In order to generate results serving as basis for the evaluation this scenario must be rapidly changing where time constraints makes it difficult to stabilize the system, as explained in section 2.3.1. Already stated in the motivation, an air defense scenario is being used to evaluate these effects. Important in this objective is also to ensure that this scenario fulfills the requirements of realism and accuracy while it at the same time not loses too much tractability.

This objective consists of determining which parts of the air defense scenario as well as the threat evaluation and weapon allocation should be included to maintain high realism, and still be able to extract valuable information for the evaluation. This can be done by relating to the choices made earlier in section 3.4.1 and 3.4.2. By breaking out important parts from literature on the subject the implementation process can be outlined. The distributed system as well as the consistency models should be included in this step, and the level of accuracy in these should be addressed. Outlining these important parts and address in which order they should be implemented is the final step in this objective.

3.4.4 Constructing scenarios

A complete implementation of the simulator is nothing without well weighted scenarios that can be used to extract information. These scenarios serve as a link between the implementation and the evaluation phase.

Validating the implementation by selectively testing different parts ensures that it produce the sought after effects, and this way it can be determined which factors having the most significant impact. This is done by using the parts presented in section 3.4.3, and test each of them ensuring that they work as expected in relation to the literature they derive from.

Once the validation is complete, scenarios adapted for extracting information specifically for the evaluation of the eventual consistency should be constructed. The scenarios should be as tractable and accurate as possible. While these scenarios should not be micromanaged into something that produce sought after effects (i.e.,

forcing expected results), it is important that these scenarios in fact produce data which can be used in relation to the hypothesis, no matter is if is to falsify the expected outcome or to verify it.

3.4.5 Extract information, evaluate and present results

From the constructed scenarios information needs to be extracted in a way enabling it to be used for the evaluation. It needs to be decided which output is to be considered important, and how to accurately handle and format it for further use. This is done by counting the remaining battlefield resources after an elapsed scenario, and then collecting these results into groups and finally removing outliers (i.e., values considered to be abnormal to the situation). By using different consistency models in otherwise identical scenarios it is possible to extract the information needed to extensively test the hypothesis.

The results should be presented in a way which enables the drawing of conclusions as well as making it useful for future work. The evaluations should clearly state in which situations one approach stands out as superior.

4 Method

Solving the objectives presented in section 3.4 requires a method, and this part of the work explains the process for fulfilling the purpose of this work, presented in section 3.1. As it is essential to select the evaluation method priory to any outlining of the implementation phase, the choice of evaluation is selected and motivated in this chapter. This also includes the selection of the tools being used, as it connects the evaluation method with the implementation. Excluded research methods are also briefly presented and detailed as why they are not used in this thesis work.

Section 4.1 details the evaluation method selected and section 4.2 compares and selects methods and tools used for implementation. In section 4.3 the actual construction of the simulation and the simulator is detailed, and section 4.4 outlines the process of constructing scenarios used for both validation and evaluations. Finally section 4.5 outlines the process used for running scenarios, extracting data and presenting them.

4.1 Selecting evaluation method

In section 3.4.1 it is stated that the evaluation method should be the first part in the research to be selected. Three methods are outlined: simulations, analysis and experimentations. For the evaluation of eventual consistency all methods seem feasible, but as stated in section 3.3 the evaluation should be conducted using an air defense scenario thus narrowing the usefulness of some methods.

4.1.1 Simulating the effects

The primary objective in this thesis is to evaluate result of the presented consistency models in realistic scenarios; not construct a solid proof that hold for every presented scenario. We do not cover extreme scenarios. During simulation, multiple runs in the scenario cover a large number of situations resulting in both an average and a median output used for conducting evaluations. The advantages of this would be that it is closer to the real world scenario as compared to the analysis, and if well constructed it comes close to the realism of experiment as well. It is also possible to simulate with a high accuracy, and simulations can be done using basic computer equipment.

The disadvantage of simulation would be that the evaluation only covers the chosen situations and could not be seen as a complete proof. While some extreme situations in the scenario occur during simulation, it could not be known that the final results cover the entire variety of situations that can be presented in the scenario.

4.1.2 Analyzing the effects

In an analysis the results need to cover all possible situations in the model. A concrete proof must be presented and once this is done it will be rigorously tested to ensure that it is solid. This has the advantage, should the proof hold, that it cover all situations in the scenario. An analysis can be completed using only mathematical statements and proofs.

The disadvantage would be that in order to make the model tractable the accuracy of the scenario must be decreased. If the scenario should be as realistic as possible, as stated in section 3.3, there is a need for a high accuracy. This will leave the analysis with a difficult proof to solve. Further, adding to the disadvantages, it is hard to analyze worst case scenarios as well as other extreme and unpredictable situations.

4.1.3 Conducting experiments to evaluate the effect

Experiments reflect the real world, and do remove the reality gap that occurs when using simulations. It is more difficult to point at unrealistic situations as compared to simulations and analyses since the experiment show great realism should the scenario be well constructed. Experiments also increase the details of the scenario, something that is difficult to simulate or analyze at the same level.

As this research is conducted using an air defense scenario, with the motivation for this choice stated in section 3.3.1, it is not feasible at all to conduct experiments. Even simplifying the experiment radically (e.g., changing fighter planes to small remote controlled planes) it would still cost a huge amount to conduct this research. If the experiment should be too simple it would also lose its advantages, rendering it less useful than simulations and analysis.

4.1.4 Method evaluation

Ruling out experiments can be done in an early stage, as it would cost too much to conduct this research in that way regardless if it was with real aircrafts and missiles or models. Experiments are thus not explained in further detail. This leaves analyzing the effects or simulating them as two reasonable approaches, or using both of them. The selected way to evaluate the effects is to simulate them, and the reason for choosing this method is explained in this section.

The benefit of using a specific evaluation method does not stand out as clear as the disadvantages of using the other one. In this research using simulations stand out as superior, as the disadvantages of analysis are presented below.

- *Complex interaction*. Even if the number of components used in the scenario is kept at a minimum they do interact with each other quickly producing complex situations. As an example, one active part generates new data that is later used in another interaction with other parts in the network. In simulations this works out itself naturally while the weakness in an analysis is that these interactions must be found, analyzed and included to complete the analytic proof.
- *Uncertainty*. In the scenario, as well as in the real world, there is a great deal of uncertainty involved. As an example, a missile has a certain kill probability, but this can differ a great deal depending on slight variations. The skill of the enemy pilot as well as small changes in humidity can cause the missile to either hit or entirely miss its target. This is difficult to analyze. A simulation on the other hand can be implemented using random values to address this problem.
- *No results are based on analysis*. This is perhaps the most important parts, as new research is easier to conduct if it has a foundation to build from. As no other research in the area of threat evaluation and weapon allocation uses analysis, but instead simulations, this serves as a great motivation for choosing simulations.

The greatest downside of analyzing is that accuracy (i.e., the realism) must be sacrificed in order to make it a tractable scenario (Cheung & Kramer, 1994), as seen in Figure 4.1. The figure is based on the work of Horvitz (1990, p.138-141), where he addresses the problem of increasing complexity (i.e., more sub-problems) when adding more variables to the presented problem. By adding more functionality and

variables to a scenario, the complexity increases exponentially creating a combinatorial state explosion problem, explained also by Cheung and Kramer (1994).

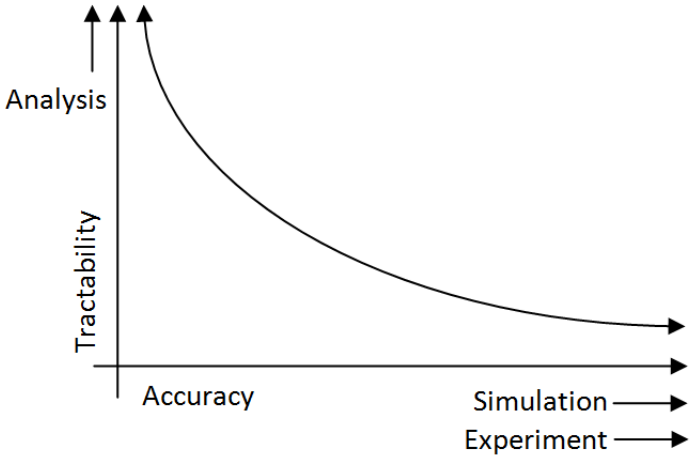


Figure 4.1. An increase in tractability decreases the accuracy (Horvitz, 1990).

While the complexity also increases in simulations when adding more variables to the problem, the effect is not as apparent. The simulation does not need to cover all situations, but if the analysis should present a solid proof sub-problems must be addressed. Depending on the research, the amount of sub-problems can vary and in this research they would factor too many unknown variables even when keeping the realism to a simple level (e.g., aircraft attributes and defense deployment). This leaves simulation as the most tractable research method, even if basic analysis would be possible.

As stated in section 3.4.1, it is also possible to conduct this research using multiple evaluation methods, and it is possible to include a basic analysis as a complement to the simulation. However, as stated in the previous section, an analysis will be simple and not reflect the real world, so the benefits of including analysis are vague.

As one of the primary motivations for conducting this research is that it should be applicable to a practical situation, the use of including analysis adds little benefit unless the scenario could be made general. While a general proof indeed could be useful it falls outside of the scope of this thesis. Thus the only method for evaluation is simulations.

4.2 Tools used for the implementation

By using a computer to run simulations, results can be extracted and further analyzed. This section outlines the tools available for constructing the implementation, and motivates specific choices in both software and hardware. Included in this step are also specifications of computers used in the simulation.

4.2.1 Selecting simulation tool

There are primarily two approaches for implementing the simulation. The first is to implement an own simulator using a programming language known to the author while the second being using a known simulation tool and with the help of it develop scenarios. Both approaches do have advantages and disadvantages, as described next.

- *Implement the simulator from scratch*. The primary advantage by coding a simulator from scratch is that one is in full control of the development. Should something malfunction or act strange, the cause of the problem could be

traced. The disadvantages are that it is less credible to develop the simulation tools from scratch as they are not tested and proven for scientific work. It is further difficult for other researches to duplicate the results if they are not provided with the source code of the implementation.

- *Using a simulation tool*. The advantage of using a recognized simulation tool is that the results are more credible and it is also easier to extract the results and present them, should the simulation tool have built in statistics. In an abstract way scenarios could be presented in this work and it is possible for others to duplicate the results using the same program. The disadvantages of using a simulation tool are that lacking functionality can be difficult to add and there is a need for an evaluation process selecting the right simulation tool. Also, problems with the simulations can be more difficult to detect.

Being in full control of the simulation process is the most important part in this thesis work. The choice was made because it is important to be able to control the functionality and follow the information flow in the scenarios. Creating a simulator from scratch provides the tractability and accuracy needed for the chosen scenarios.

In order to counteract the disadvantages of implementing simulator from scratch, the entire code, simulator and scenarios is available on request from the author. Information about how receive the code and the executable simulator including scenarios is presented in the appendix.

4.2.2 Software and hardware used in creating the simulator

As stated in section 4.2.1, the simulator should be implemented using a programming language known to the author. In this case C# (ISO/IEC 23270:2006), using Microsoft Visual C#®, was used for the construction the simulator and scenarios and in order to graphical represent the scenarios Microsoft XNA® Game Studios provided a quick and functional solution. While other languages was possible, the choice of C# and XNA came with the fact that the programming environment was free, easy to prototype and work with as well as adequately fast at executing the constructed scenarios. XNA provides an easy way to create graphics, which is used for visual validation.

The computers used for simulating the scenarios are standard desktop computers with the specification of a regular PC the year 2010. There is no exact specification as simulations are being executed on different machines. As simulations provide useful information within minutes there is no real need to use powerful computers. It is further important to notice that running simulations on a computer is not equal to actually implementing and testing this in a real world scenario; thus the exact specifications of the computer is irrelevant to the research. The following configuration is an example and the average hardware used in computers in this research.

- *Processor*: 2-4 cores @ ~2 GHz
- *Memory*: 2-4 GB RAM
- *Hard drive*: Standard 7200RMP mechanical disk
- *Operating system*: Windows 7

4.3 Constructing the simulation

When constructing a simulation, some architectural decisions need to be considered before any actual implementation can be conducted. Outlined in this section are these decisions (ranging from section 4.3.1 to 4.3.10) and why they are important in this research. Keeping the simulation as simple as possible while still retaining the full capability to verify the hypothesis is one of the primary goals with these decisions. In section 4.3.11 the actual implementation process of the simulator is discussed.

4.3.1 Using a time triggered or event triggered simulation

There are two interesting approaches as how to construct the simulation in this thesis; either using a time triggered or an event triggered implementation. Kopetz (1991) describes these two in detail in his book on the matter; the key points are described below.

1. *Time triggered simulation.* The time triggered simulation progress the simulation by polling entities in the system with activity at a constant rate. The flow of the simulation is kept running as all entities in the system receives an event each update; whether or not the entity reacts to this event are situation dependent.
2. *Event triggered simulation.* Acting on *significant events*, the event triggered simulation reacts to actions or messages from other parts in the system. The flow of the simulation is kept running as new events triggers other parts of the system to react. Events less than significant are neglected.

Deciding whether to use a time triggered or an event triggered approach is first and foremost an architectural design decision. Scheler and Schröder-Preikschat (2006) remarks that this decision could be postponed, as it has little to do with the actual functionality of the simulation, but at the same time states that such a critical decision usually is decided in the beginning of the work as failing to do so could be very time expensive.

Time triggered simulations are used in this thesis. The decision is based on the fact that units on the battlefield move in accordance with time and their velocities; movement is not based on events. Because of the enforced regularity in time triggered simulations they are also more predictable than event triggered (Kopetz, 1991), making the validation process easier. The actual time used in simulations is variable, and can be paused or forwarded faster than real time.

4.3.2 Random number distribution in the simulation

Ensuring a working random distribution is essential before any actual scenarios can be simulated. As this is indeed a simulation the random number generator must be deterministic. *Deterministic randomness* means that it follows a predetermined algorithm making it possible to reproduce the exact same random sequence, which is important in simulation as it helps during debugging but more importantly enables reproducibility of previously executed scenarios (Perros, 2003, p. 27). This type of randomness is created using a *Pseudo Random Number Generator* (PRNG), also called *Deterministic random bit generator* (DRBG), and it give the illusion of true randomness which should be statistically indistinguishable from true random values (Barker, et al., 2007). This statement of the output being indistinguishable from true randomness is usually solid unless conducting extensive random-tests on the number

sequence (Marsaglia & Tsang, 2002); these tests often even fails the most advanced random generators.

Due to the fact that the simulation has to use deterministic randomness, the output can never be true randomness; it will always follow some sort of predetermined path and at some point it will usually repeat itself (Perros, 2003, p. 29), thus being both predictable and periodic. This does not necessarily constitute a problem, thus the final object is to ensure that the randomness is sufficiently unpredictable (i.e. when using a new seed) and that the produced random scenarios cover a wide selection of possible battlefield settings.

4.3.3 Observability

As briefly mentioned in section 4.2.2, the software chosen to implement this simulation provides for a way to render graphics. In order to increase the observability, the decision to implement graphics decrease the overall development time as faulty behavior of entities (e.g., aircrafts and missiles) moving in space and time can easier be detected. This can be compared to validating the simulation only by examine numbers. Sergent (2008) concludes that it is critical in all simulations to verify and validate the model, but that no perfect approach exists and it is always scenario dependent on which model to choose.

Graphical animation is used in this research, explained by Sergent (2008) as a validation technique where "the model's operational behavior is displayed graphically as the model moves through time". This decision is based upon military tactics, where a bird's eyes view of the battlefield often is used providing the commanders with great observability. This simulation uses a two-dimensional map projected straight from the top. Belligerents are represented by abstract figures which are not to scale (due to the fact that aircrafts would not be visible compared to the size of the battlefield). As time progresses, moveable units position are updated on the map. Their heading, spawning areas, resources (e.g., missiles) and identity are displayed. Further adding to observability, the current variable values of all belligerents are being displayed along with the current scenario configuration. Units hit by a missile changes color as they are destroyed.

The graphics does not add to the scientific results in the final evaluation besides aiding in validating them. Furthermore, the graphics can be turned off during actual long-run simulations in order to increase performance.

4.3.4 Replica update completion time delay

The RUCT is specifically the time it takes from the point where one update starts until the effects of that update is replicated and agreed upon in the network, and is detailed in section 2.2.2. It is important to be able to measure the RUCT, in order to evaluate different scenarios. Further, the RUCT must range from negligible time to a problematically long time (i.e., which is scenario dependent) in order to be able to verify the hypothesis. In a real scenario, multiple delays would add up to the overall RUCT; network delays, conflicts, geographical locations, to name a few. In simulation these delays must also apply in order to reach the point where the RUCT in the scenario become problematic.

Without taking delays into account, it is highly unlikely that the RUCT will be anything but negligible if the simulation runs on one computer. In order to construct a useful simulation, delays needs to be emulated. There are two approaches; either

emulate the delay at all places where they would apply in the real world, or force a single delay onto the entire update. The decision to use a single delay has been chosen as the primary goal is to verify the hypothesis and not to emulate delays at different places in the simulation. That would require domain knowledge in many different areas, and would take time from the real research goal. More importantly, in simulations the delay would become difficult to adjust; a single value is more tractable and accomplishes the same goal.

Replica update completion time delay is a static value with the purpose to emulate a time delay during an update. This serves as the main input variable in the simulation, and by using this delay any RUCT can be emulated and evaluated. Figure 4.2 illustrates the process; the RUCT comprises of the actual update and the delay time.

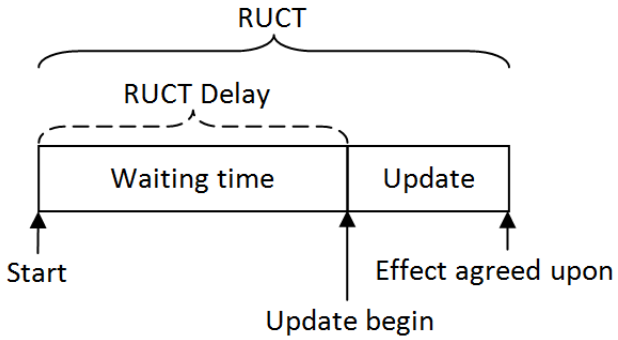


Figure 4.2. Illustration of RUCT delay and when the update actually occurs.

As the time in the simulator is variable and can be paused, mentioned in section 4.3.1, the actual update process, illustrated in Figure 4.2, runs with the time halted which eliminates the need to control another variable during simulations. The means that the applied delay equals the exact time it takes to complete the update.

4.3.5 Placing belligerents on the battlefield

In order to extract useful information from simulations, it must be possible to place belligerents in positions that make sense to scenarios. Two primary approaches exist; either randomly place belligerents within a certain area or place them manually at each location.

- *Random placement area*. Using this approach, predetermined areas randomly spawn new belligerents at the start of a trial run in the simulation. This has the benefit that a huge amount of similar but slightly different scenarios can be produced. The downside is strange behavior which could occur, such as all aircrafts spawning really close to each other at the corner of the placement area.
- *Predetermined exact placement*. Each belligerent is placed at a specific location. This has the benefit that scenarios play out exactly the way they were designed to do, whereas the downside is that it takes much more time to construct different trial runs, and there is a need for domain knowledge to cover all situations needed for evaluations.

This research uses random placement areas for belligerents. The decision is based on the fact that the negative aspects of predetermined placement would become problematic to overcome. Further, the negative aspects of using random placement could be converted into something useful, as it can produce some of the most extreme situations showing the full distribution of the output from scenarios. Should these

extreme situations be problematic during evaluations, then they can be considered as outliers and removed during the data preparation.

The actual random placement area can in practice look like any two-dimensional form. In this work, the placement areas are represented by circles as they are easy to use and validate (i.e., is inside if the distance between the units position and the circles middle point is less than the radius). It is possible to use multiple placement areas, so these areas can constitute to more complex forms if needed during simulation. There are two simple ways of spawning units in a circle, described next.

- *Spawn inside a square, check if inside*. Producing a single large square covering all spawning locations, the position is randomly selected within the square and then tested if inside a circle.
- *Random angle, random length*. From the middle of the circle, a random angle and a random length (no longer than the radius) produce a spawning location. If the random length is squared rooted, the distribution is uniformed.

The spawning square is the selected method as it produces a useful feature; spawning locations can overlap and it is not necessary to select which circle to spawn within. The result is uniformed over the entire area, but takes more computer resources as failed spawning points have to be recalculated.

4.3.6 Concurrency control

When a network is replicating data between nodes there is a need for concurrency control, as stated in section 2.1.4. Even in simulation, using concurrency control can be a difficult task and while it is preferable to use the optimistic concurrency control, this simulation uses the pessimistic approach as it will not affect the actual research goal and it is easier to implement and validate (e.g., no conflicts can occur).

Strong strict two-phase locking ensures that information is passed between two nodes without any interaction from other parts of the network (Bernstein et al. 1987). This is also called *rigorous two-phase locking* depending on the author, but what is essential is that other nodes are unable to either read or write data to the isolated parts during the transaction. Used in this simulation, the node which initiates an update has the ability to lock all parts affected by the update instantly in one single time step. While unrealistic in a network spanning over multiple physical locations; this research does not focus on evaluating different concurrency controls and this method ensures that information is passed correctly according to the ACID-rules.

4.3.7 Consistency model

Identical scenarios in simulation using different consistency models enable a way to evaluate the effects of using the eventual consistency model by comparing it to the immediate consistency, which serves as a baseline for this evaluation. The immediate and eventual consistent systems are described in section 2.2.3 and section 2.2.4.

In this simulation, the immediate consistent system avoids conflicts when propagating updates by locking down all affected parts and just focusing on completing the update, as stated in 4.3.6 using strong locking. As it completes, a decision has been formed which is carried out by the affected parts. What occurs during the update is irrelevant to this research; instead what matters is that it takes a measurable amount of time and that no conflicts did occur during the update. The RUCT delay is being used

to simulate this time, and conflicts are avoided by only by using the strong strict two-phase locking.

Eventual consistency can be used, as stated in section 2.2.4, in multiple ways. Used in this research to verify the hypothesis the eventual consistency acts on tentative information, either directly on its primary decision or waits a predetermined time in order to form a secondary decision. In simulation, once a final decision has been made the node acts instantly on this decision while concurrently propagating the decision to the other parts of the network.

Regardless of the consistency model in use, the RUCT takes the same amount of time no matter the physical location of the missile defense system located on the battlefield. This eliminates conflicts even in the eventual consistency model, as all parts in the network are affected by the update at the same time. The RUCT delay enforce the time it takes to complete the update; should two update be underway at the same time (which is possible in eventual consistency) the second update, which also always is the most recent, will overwrite the results from previous updates.

4.3.8 Layered decision making

While the primary research goal is to verify the hypothesis stated in 3.2, one of the main motivation behind this thesis is to evaluate if eventual consistency indeed can be useful, as detailed in section 3.3. As explained in section 2.4.4, using multiple layers can improve decision making in eventual consistency. While it is possible to use any number of decision layers, this research limits the decisions to one primary and one secondary, based on the steps used by Pignaton de Freitas, et al., (2009). When reaching a secondary decision the node will act on that decision (i.e., and not continuously form new temporary decisions). In simulation, a decision is formed on a single node and propagated throughout the network. During a predetermined time, the node waits for information from other nodes. If it receives information from another node within this time span, it has the ability to form a second decision and act on this should it be preferable to the primary decision.

In reality, the information sent can be everything from major decisions such as firing missiles, to minor indications of vague movements in the battlefield horizon, all used to form an optimal decision. In this research only primary decisions are being propagated out in the network, thus simplifying the TEWA process as it is not the research goal.

In this simulation, the eventual consistency can either use no layered decisions, or two layered decisions, as described above. The time between the primary and secondary decision is a configurable variable.

4.3.9 Belligerents and resources

On the battlefield, four types of primary resources can be found; the aircraft and the missile defense system which both has the use of missiles, and the defended asset around which the belligerents battle about. These resources are described in section 2.3. When used in simulations, additional functionality above what it necessary for the evaluation is omitted as it increases the complexity. Below the belligerents functionality and variables are described in detail.

- *Aircraft*. The aircraft has a velocity (i.e., speed and direction), a number of missiles as well as a designated defended asset. If different types of aircrafts

are being used at the same time, this is identified as well. Finally it has a current position which is continuously modified by the velocity.

- *Missile defense systems*. The missile defense system has a location and both an initial as well as a remaining number of missiles. The defense system is stationary.
- *Defended asset*. The stationary defended asset has a location and information whether or not it has been hit by a missile.
- *Air to ground missile (ATGM)* and *Surface to air missile (SAM)*. Fired from either aircraft or missile defense systems, the missile has a position, velocity and target. The missile also has a range, which described the maximum distance the missile can travel without running out of fuel. Actually using fuel in simulations is omitted; instead missiles are only fired if they are close enough to reach their target in time. The number of missiles attached to an aircraft is variable and can be randomized between a maximum and minimum value. Missiles are set to always hit if they reach its target, as this is not an evaluation of missile effectiveness.

Aircrafts also communicate information between each other, in a simple way, thus decreasing the possibility that all aircrafts attacks the same defended asset. This increases the observability in the simulation, as aircraft do not cluster in tight groups.

4.3.10 Threat evaluation and weapon allocation

The threat evaluation used in this simulation simple and straightforward. All aircraft targets of hostile intent are treated the same, and no distinction between trajectories and ordinance is being used in calculations. This means that the first target in sight has the highest neutralization priority.

The weapon allocation process used in this simulation is more complex than the threat evaluation. When using immediate consistency, the future position of the target is estimated based on the trajectory and the time it takes to complete an update. The missile defense system closest to the target at that point fires its missile. The eventual consistency model acts on tentative information and launch its missile at the same time it finalize its decision. Both consistency models ensure that, no matter the circumstances, multiple missiles launched from the same missile defense system never intercepts the same target.

4.3.11 Implementing the simulator

Following an incremental process the simulator is constructed step by step where each part are being tested before advancing with the development. Below follows an outline of the indented implementation process and while it is important to follow a structured implementation process programming is never straightforward and features are added once needed. However, it is always important to validate newly implemented parts before moving on. The process described below includes all parts previously detailed in this section, and does only include the actual implementation process and not precise programming decisions.

1. *Time triggered event system.* The first step is to ensure that the simulated time flows like intended; it could be paused, simulated in real time or as fast as the computer manages. The simulated time used to trigger event should be separated from the real time.

2. *Random generator and seed value.* Ensuring that the random generation, while it is still deterministic, is sufficiently random to not cause problems with the simulation and that a random sequence can be repeated using the same seed value.
3. *Battlefield belligerents.* Only represented without any implemented functionality, the battlefield belligerents can be place on the battlefield at predetermined positions. They do no move.
4. *Graphical representation*. In order to validate results and detect anomalies, a graphical representation, as well as text output, is implemented at an early stage. This includes, but is not limited to, the ability to render attack radius of aircrafts as well as missile directions.
5. *Random spawning locations*. It is preferable to randomly place belligerents within a predetermined area in order to be able to create multiple scenarios fast. As this eliminates the need to by hand placing belligerents, this is something implemented in a quite early stage.
6. *Basic functionality*. Starting with the aircrafts, a function moving them towards a defended asset is implemented. Once they are in range of their mission objective, a missile is fired towards it. This missile has a velocity just like the aircraft. As it hits the defended asset, both the missile and the asset are destroyed. On their way towards the objectives, aircrafts can come in the range of the missile defense systems. They also have the ability to fire missiles. All decisions are decentralized formed at the local entity.
7. *Realistic attributes*. The aircraft, defense systems and missiles, as well as the scenario settings acting behind the scene in the simulation, get realistic values implemented in this step. This step does not include any sort of rigorously research as what is to be considered realistic since this is done during the construction of scenarios. Realistic can thus be seen, in this context, as something that is adequately accurate and rooted in the real world which assists the construction of the simulator. This includes limited resources.
8. *Consistency models and concurrency control*. The two primary models, eventual and immediate constituency, are implemented. A locking process used for immediate consistency is implemented to strictly obey the rules of ACID. The RUCT delay is included in this step.
9. *Selection of defended asset*. Aircrafts should to some extent communicate on which defended asset they approach in order to avoid multiple aircrafts attacking the very same. This includes attacking the closest asset possible while at the same time avoiding those already eliminated; these aspects have to be weight against each other.
10. *Weapon allocation.* The step includes calculating the future position and forming decisions based on this information, also ensuring that no missile defense system fires multiple missiles at the same target.
11. *Decentralized layered decision making*. In two steps decisions propagate out through the network, with the second decision having the ability to overturn the primary decision if it were to be superior.
12. *Multiple trials and scenarios.* Once a trial run has been completed, the battlefield is reset, the RUCT delay increase and another trial commence. At a

predetermined point the RUCT delay does not increase any further, but instead resets to its starting value and a new scenario, based on random spawning locations, is created. The simulation runs until all scenarios have been completed; a value also predetermined.

13. *Scenario file*. A scenario setting file enables customized scenarios. The scenario file is loaded upon the start of the simulation.
14. *Save results to file.* The final step is to save the output to a file on the hard drive. The information contained on the file is the actual scenario settings, as well as the remaining amount of defended assets and missile at different RUCT.

4.4 Constructing scenarios

As mentioned in the objectives, a well constructed simulation does not produce relevant output unless it uses properly constructed scenarios. This section outlines the important parts of constructing these scenarios, starting by selecting aircrafts and missiles in section 4.4.1, which will later be used in the research. After this, knowledge about parameters and settings is acquired in section 4.4.2, followed by defense deployments in section 4.4.3 and finally the construction and outlining of a base scenario in section 4.4.4.

4.4.1 Research on aircrafts and missile defense system attributes

In section 4.3.11 (step 7) semi-realistic attributes without rigorously research was implemented in order to assist the construction of the simulator. This section covers more in detail different battlefield units and their attributes. The two goals in finding and using these units are listed next.

1. *Using modern-day military units*. The aircrafts and missiles should be in active service at the moment, but should at the same time not be to newly introduced since accurate information then will be increasingly difficult to find (due to classification). They should also be commonly known and not be prototypes or created in low quantities. Fulfilling these requirements, at least three aircraft with three ATGMs (compatible with the aircraft) and finally three SAMs should be included in this research.
2. *Attributes to the selected units*. As decided in section 4.3.9, only a few selected attributes should be used in simulations. These should be identified for the selected units, and since some information is classified specific choices needs to be motivated.

4.4.2 Knowledge about the simulation

Acquiring knowledge about relationships between settings in the simulation is essential in order to construct well formed scenarios. Aircraft settings as well as missiles derive from the process outlined in the previous section, and together they are used in three major steps. While this seem like an incremental process, it is impossible to just test one part and then move on. Thus, an iterative process of these three steps is used to acquire all knowledge needed.

1. *Battlefield resource weighing.* Three primary resources as well as two secondary are available on the battlefield, with the primary being aircrafts, missile defense systems and defended assets while the secondary being surface-to-air missiles and air-to-ground missiles. Starting out with 10 of each

primary resource, and 1 of each secondary resource, the actual output from the battlefield can be tested. If most of the assets are being destroyed it is possible to increase the amount of missile defense systems or simply assign more surface-to-air missiles to each unit. On the other hand, reducing the number of aircrafts can result in the same effect. The output should also stay relevant no matter the consistency model in use, as scenarios proving the hypothesis are identical with the exception of the consistency model. As the evaluation focus on the amount of destroyed defended assets, the number of assets is set to a level appropriate when considering output, and other resources are modified with this value as a basis.

2. *Simulator speed and significant factors.* There is no need to run long scenarios if they do not produce more useful output than short scenarios. In this step multiple scenarios are constructed in order to determine which value indeed impacts the results. As an example, if a simulation run consist of 100 scenarios but the result actually equals a simulation run with 10 scenarios using the same configuration (with the exception of the seed value), then 10 scenarios would be preferable. Of further importance, since the simulation is time triggered, and because the time itself is emulated, the actual time steps have an impact on the results as well as simulation speed. Constructing these scenarios ensures a high quality output without wasting computer resources.
3. *Primary parameter settings.* In section 4.4.1 the real world specifications of aircrafts and missiles were identified. In this step, all these values are put to the test at their extremes and beyond. The motivation for testing variables over a large span is because it is not possible to perfectly simulate pilot behavior, which is a vital factor in combat. Thus the final scenario will always have a margin of error and this parameter test ensures that even with this error included the scenario produce relevant output. This also increases understanding of which parameter settings affect the other parts of the battlefield. As a simple example, it is easy to figure out that more ordinances will cause more damage to defended assets, and knowledge about these relationships is essential. Extreme situations should however still be in the range of what is considered possible and realistic (e.g., an aircraft cannot travel at light speed or even close to it).

When acquiring knowledge the code is at the same time more rigorously validated, and the code is optimized and corrected when needed. When these steps are completed, the simulator is capable of running all sorts of scenarios contained within regular mission parameters and knowledge about the construction of these scenarios has been acquired.

4.4.3 Aircraft attack mission and defense deployment

In a military setting, where aircrafts attacks defended assets, the task the pilot faces is to maximize the damage done to the opponent while at the same time evading dangerous situations. If it is possible to avoid missile defense systems altogether that would be highly preferable. The defending side, on the other hand, deploys their defense in order to intercept as many aircrafts as possible. In a real battle this is a strategic decision with two sides trying to get the upper hand, and scenarios constructed for simulation need to reflect this without letting any side getting a noteworthy advantage.

In simulation it is preferable to use a small portion of the battlefield since simulations runs fast with less battlefield resources and, more importantly, scenarios are easier to construct. Figure 4.3 shows a portion of the battlefield. The aircrafts, traveling from west to east on route to attack the defended assets, need to pass over the missile defense systems. This would seem like a less well calculated decision by the pilots, as a slightly longer attack route circling the batteries would avoid danger altogether. However, as it is just a portion of the battlefield it can be assumed that batteries would be placed strategically around the defended assets.

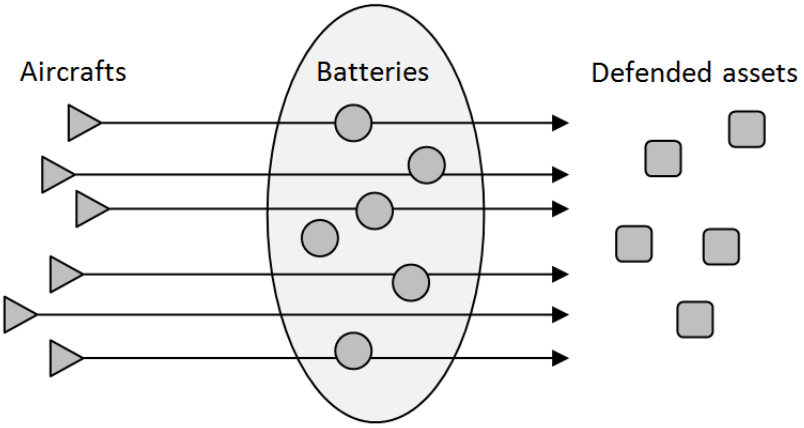


Figure 4.3. A portion of the battlefield used in the scenarios.

A more comprehensive scenario would include different angles of attack, and also better defense deployment, as seen in Figure 4.4. While this visually looks better and eliminates questions about short-sighted and straight forward attack plans, as seen in Figure 4.3, it does not further add useful information to the simulation and decrease performance by using more resources.

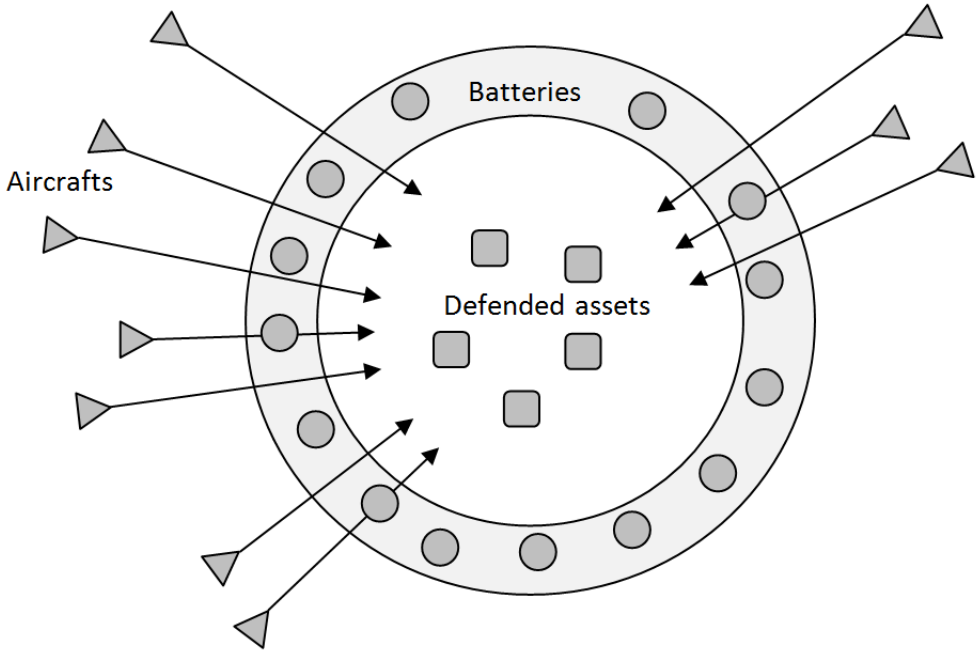


Figure 4.4. Aircraft attacking from different angles with proper defense deployment.

In order to assure that results do not diverge between using a portion or a more extensive layout during simulation both scenarios above is implemented and tested. One single and advanced layout of a battlefield is constructed and evaluated compared

to a simple portion of the battlefield. This ensures that simulations can be simplified without producing less relevant output.

4.4.4 Base scenario

A base scenario can be constructed once the simulator has been validated and it has been assured that a portion of the battlefield can be used in simulations. This scenario should weigh all important parts against each other producing the most relevant output possible considering the limitations of the simulator (i.e., as it is not possible to construct a perfect simulator it will always have limitations, but optimizing the output is essential). From this base scenario minor changes can be applied to study how the output differs and how it affects results, and it should be used as the primary scenario for the final evaluation.

One of the most essential parts in the base scenario is to set the upper limit of RUCT delay to a value where after no actual changes to the output occur. This depends on the speeds of both missiles and aircrafts, as well as their ranges, and is optimized by testing. If the RUCT delay is set to high it will impact the time it takes to complete a simulation, without affecting the actual evaluation (i.e., it will produce static output). Similar to the RUCT delay, the time span between a primary and secondary decision in layered decision making also needs to be modified by trial and error. Both the RUCT delay and decision time can be modified once all other variables in the base scenario have been fixed.

4.5 Extract information, evaluate and present results

When using simulations to evaluate the effects, there are different approaches as how to extract, evaluate and present the results, detailed in this section. This section starts with an overall approach as how to conduct research.

4.5.1 Running the simulations

Using the base scenario from section 4.4.4 combined with a preset initial seed value, the simulator runs multiple scenarios derived from the scenario setting file. Each consistency model is simulated separately. The RUCT is being delayed on purpose for each consistency model, as explained in section 4.3.4. By ranging from zero delay to the upper limit, the RUCT delay increases for each new trial run in a scenario. Once the upper limit has been reached, the output from the scenario is saved and the seed value increases linearly at which point a new scenario is randomly constructed using the spawn areas. When all scenarios in the simulations have been completed, the output is saved to a file on the hard drive.

The simulation runs autonomously once started. The only value being modified during a scenario is the RUCT delay time, which is linearly increased. In order to extract an average value which can be used for proving the hypothesis, multiple scenarios using different seeds follow each other until new scenarios no longer have an impact on the result (i.e., an impact that is noteworthy).

Three more types of scenarios are executed. First, using eventual consistency with layered decision making, five simulations using different delayed decision times are completed using the same base scenario with the exception of fewer simulation runs. Secondly, using the base scenario but running with both fewer and more aircrafts, data is extracted to verify that the hypothesis stays valid in all situations. Third and

finally, a missile defense system using short range missiles is used to evaluate short time frames.

4.5.2 Prepare data for evaluations

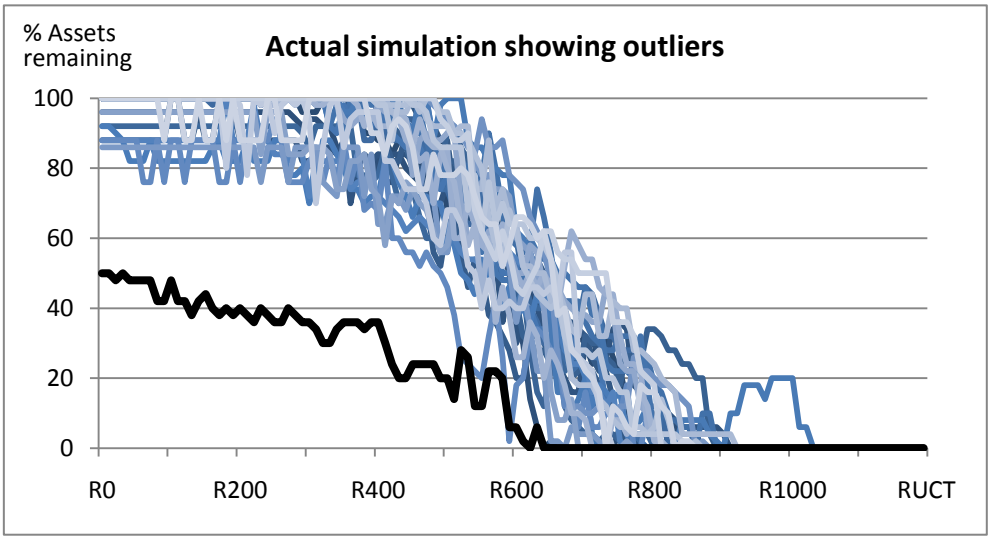
Once all simulations have been completed, data has to be prepared so it can be used for evaluations. The files saved on the hard drive consist of raw data from the trial runs, and they are formatted in three matrixes. The data sets consist of remaining assets (in two formats) as well as remaining SAMs. In this section the goal is to remove outliers, and then prepare the data so the immediate consistent system can be used as a baseline compared with the eventual consistent system.

While the scenario configuration file employs predetermined spawn areas, there is still a chance that a randomly set spawning location within this area is illogical from a tactical point of view. Based on the definition by Hawkins (1980) this would be identified as an *outlier* ("an observation which deviates so much from other observations as to arouse suspicion […]", p. 1). As an example, consider the data set in Table 4.1 representing the output on 7 different levels using 5 different seeds. This data set is a cut out of a large data set deriving from actual simulations.

Seed	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7
1000	100	100	100	100	100	100	100
1001	100	98	100	90	98	100	98
1002	100	100	100	100	100	100	100
1003	88	88	100	100	88	88	88
1004	50	48	38	36	36	32	30

Table 4.1. Data example set for illustrating outliers.

While not identical, seed 1000 to 1003 shows similar patterns, while seed 1004 clearly deviates from the other values. This becomes even more obvious when illustrated in Graph 4.1, together with multiple other scenarios, as the highlighted line drops far below the rest of the output. For some reason, seed 1004 has produced a defense deployment highly unlikely as multiple aircrafts pass through the defenses. This is defined as an outlier in the data set.



Graph 4.1. The outlier, Seed 1004, drop below the other output in the data set.

The outlier in the example above is easy to detect, both with algorithms and visual inspection, but identifying an outlier in a large data set can be difficult. Since there is a lot of uncertainty in this research (e.g., pilot skills and defense deployment) it is also

difficult to clearly determine what is to be considered an outlier, so the use of complex methods is avoided.

Using a tractable method, the data sets are arranged in quartiles. Output between the upper and lower bounds is considered statistically significant; everything outside is considered an outlier and is thus removed from the data set used for evaluations.

Located between the lower (*Q1*) and upper (*Q3*) quartile is 50% of the data set. The distance between Q1 and Q3 is the *interquartile range* (*IQR*), and is used in order to find the outermost boundaries. The *lower boundary* is located at $Q1 - (1.5 * IQR)$ whereas the *upper boundary* is located at $Q3 + (1.5 * IQR)$. All data outside these boundaries is considered as outliers. The percentage of removed data depends on the distribution, but is usually a few percent in this research.

Once the outliers have been removed, it is time to prepare the data so the output from the immediate consistent systems can serve as a baseline for evaluations. The data set is normalized to a straight leveled line, thus also modifying the data set of the eventual consistency. The output is the difference between the consistency models. Considering the following equation:

$$\Delta assets = ic_i - ec_i,$$

where $\Delta assets$ is the difference between the two consistency models while ic_i and ec_i is the data set for the immediate and eventual consistency respectively. As an example, if the immediate consistency has 5 defended assets remaining, and the eventual consistency has 7 defended assets remaining then difference would be calculated as:

$$\Delta assets = \ 5 - 7 = -2$$

This signifies, in this example, that the immediate consistent system outperforms the eventual consistent system. For each time step in the data set this difference is calculated. Once completed for the entire data set, the output from the eventual consistent system is formed around the newly created baseline.

While there are many interesting aspect to exanimate and analyze, the primary goal is to evaluate the number of remaining defended assets, which has been described in detail above. Of secondary importance are the remaining resources in form of surface to air missiles. These data sets are also prepared by removing outliers, but no baseline is calculated as it is not used in the primary evaluations and not related to the verification of the hypothesis.

4.5.3 Presenting the results

Once the data has been prepared graphs can be constructed. There are multiple graphs showing different situations and settings; short explanations are presented when needed. The following graphs are of interest and are presented in the results:

- *Comparing eventual and immediate consistency*. Using the immediate consistency as a baseline, the consistency models can be compared. This is the primary research goal. There are two graphs; one with outliers removed and one with them included.
- *Remaining defended assets*. Using the same data as above, the immediate consistent system is not used as a baseline but both consistency models show the remaining assets.

- *Distribution of consistency models results*. Using box plots and graphical areas, the distribution is illustrated.
- *More and less aircrafts*. Using otherwise same settings, more and less aircrafts verifies that the hypothesis stays valid in multiple situations.
- *Short range missile*. By using short range missiles, the breaking point moves as compared to long range missiles. Immediate consistency is used as a baseline.
- *Layered decision making*. Multiple simulations using different delay times for the decision, compared to the immediate consistency as a baseline.
- *Layered decision defended assets*. The remaining assets, not using the immediate consistency are compared. The most successful layered decision delay time is compared to the eventual consistency without the use of a delay, as well as the immediate consistency model.
- *Remaining missiles*. The actual remaining missiles, not using the immediate consistency as a baseline.

Once the results have been presented, there is a discussion about them and conclusions can be drawn.

5 Results

The primary work conducted in this research coupled with analysis is presented in this chapter. Following the method previously outlined in order to solve the objectives and verify the hypothesis this chapter goes into detail about the construction of the simulation and scenarios as well as an analysis of the output from the simulations.

Covered in section 5.1 is a brief presentation of the results as well as an overview of the finished simulator. Section 5.2 describes the construction of the simulator relating back to the architectural decisions outlined in the method. Following this, in order to produce relevant output specific scenarios have to be constructed and attributes have to be identified, and is detailed in section 5.3. Once the simulator can run the constructed scenarios it is possible to extract data and evaluate results in great detail, which is done in section 5.4. Concluding the results are section 5.5 discussing results as well as section 5.6 outlining related works.

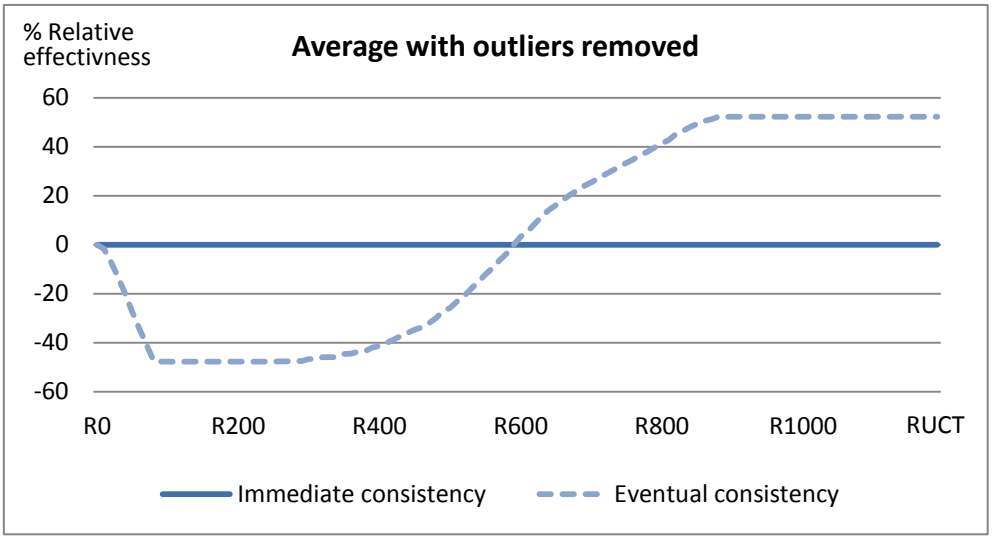
5.1 Overview of results and simulator

Briefly presented in section 5.1.1 is the final result in this research, without going into any specific details. Section 5.1.2, covers the finished simulator more in detail.

5.1.1 Overview of the results

The hypothesis is valid in all situations in the presented and tested scenarios. When the RUCT increase the eventually consistent system will always outperform the immediately consistent system, as predicted in section 3.2. At which point one consistency model performs better than the other depends on the exact scenario, and is outside the scope of this thesis, but there is always a breaking point from where the eventual consistency continuously outperforms the immediate consistency. The overview presented here derives from the presentation of the results in section 5.4.

Illustrated in Graph 5.1, eventual consistency acting on tentative decisions is compared to immediate consistency formatted as a baseline. Clearly visible around RUCT 600, there is a breaking point from where the eventual consistency continuously outperforms the immediate consistent system. The vertical axis represents the difference between the consistency models, measured in percentage points. Outliers have been removed from the output represented in the graph.



Graph 5.1. Effect of using eventual consistency compared to an immediate consistency baseline.

The output represented in Graph 5.1 has been collected from a scenario constructed with specific design goals which is the reason for its close resemblance to the hypothesis in Figure 3.1. This hypothesis however stays valid in all situations, with the rare exception when all defended assets always are destroyed (i.e., meaning that no consistency model is useful).

The horizontal axis in Graph 5.1 represents the replica update completion time and the vertical axis the relative effectiveness in percentage when comparing both consistency models. These scales will henceforth be used when presenting results, as well as the vertical axis representing the actual number of remaining assets when it applies.

When compared to the hypothesis in Figure 3.1, it is noticeable in Graph 5.1 that the eventual consistent system actually performs equal to the immediate consistent system when the RUCT is 0, before drastically dropping in performance. The reason for this is that the system reaches consistency before other parts need to form new decision, and was something not foreseen during the conceptual phase when constructing the hypothesis.

5.1.2 An overview of the simulator

This section overviews the functionality of the simulator by explaining how different parts work, without going into detail about the underlying implemented code. As the overview ensures overall understanding about the simulator, the architectural decisions during actual implementation are then further detailed in the forthcoming section 5.2.

Starting by explaining the controls in the simulator, the user can adjust the speed the scenarios progress in as well as pausing to analyze specific situations. Adjusting the speed does not affect the output produced. It is further possible to adjust the level of graphics used to maximize performance.

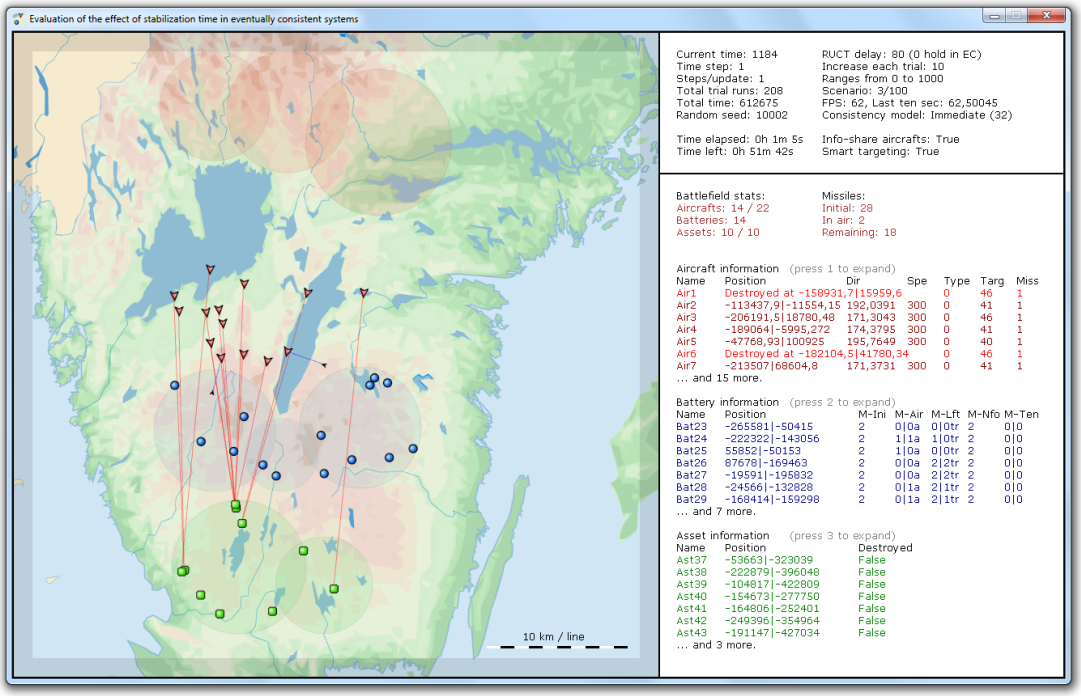


Figure 5.1. A screenshot from the entire simulator.

Figure 5.1 shows what the finished product looks like during a basic air-defense scenario. The simulator runs pre-constructed scenarios. These are created in

accordance with the demands presented in section 4.3. Once a scenario has been loaded, the simulator runs the scenario from the beginning to the end and finalizes it by saving the results to a file. The user can conduct an early stoppage of the simulation; both by directly closing down the program and by saving the results produced up to the point of the stoppage. In all other ways the simulator runs automatically once it has stated.

It is important to point out that the actual output has nothing to do with the graphics produced on screen, but showing graphics during simulations give the advantage of both visual validation as well as ease of scenario construction. Due to the fact that the functionality (i.e., the code used to produce output for evaluations) executes on the CPU and the graphics render separately using the graphic cards pipeline, the actual performance loss when using onscreen graphics is negligible. Visible in the background is a basic map of Sweden, set to scale with the actual battlefield scenario, further assisting the visual validation concerning correct velocities and distances.

Figure 5.2. Four types of battlefield resources.

In Figure 5.1 multiple objects move and interact with each other. These are the main battlefield resources, described in section 2.3, and in Figure 5.2 a close up clearly visualize the differences which are distinguishable both by color and form; (a) aircraft, (b) missile defense system, (c) defended asset, and (d) missiles, both ATGM and SAM.

Figure 5.3. Two spawn areas filled with missile defense systems.

When constructing scenarios the battlefield resources must be located in predetermined areas and this is done by using multiple spawning locations. Figure 5.3 illustrates two spawning locations for missile defense systems close to the lake. Inside a spawning location, which is represented by the form of a circle, battlefield resources spawn randomly according to the selected spawn type with the position determined by the current seed value. In reality spawning areas for aircraft could be airfields or aircraft carriers, and the placement of missile defense systems is a strategically military decision.

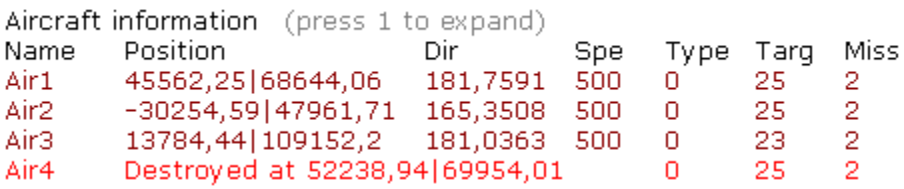


Figure 5.4. Close-up on the aircraft information in the simulator.

Statistics rendered in the simulator change rapidly as the scenario progress. There are three types of statistics displayed on the right lower side during visualization mode in the simulator. Figure 5.4 shows statistics for a couple of aircrafts, and their actual position, direction, speed, type, target and remaining missiles are being displayed. Starting with the *name*, a string and number combines into a unique identifier for the aircraft. The *position* is represented by a coordinate system in meters, where the actual origin in the Euclidean space is located in the middle of the battlefield. The *direction (dir)* is shown in degrees, with 0 degrees pointing straight to the north. The actual origin as well as direction is not important to the output as long as all battlefield resources follow the same measurements. The *speed (spe)* of the aircraft is represented in meters per second, and combined with the direction this is the velocity of the aircraft. The aircrafts *type* is used if there are different aircrafts (i.e., using different settings) on the battlefield at the same time, thus making them easier to distinguish from each other. The *target (targ)* is the unique number of the defended asset the aircraft has locked on to, and finally, the *missiles (miss)* show the remaining number of missiles the aircraft is carrying. This information changes once an aircraft has been destroyed, shown in Figure 5.4, or if leaves the battlefield.

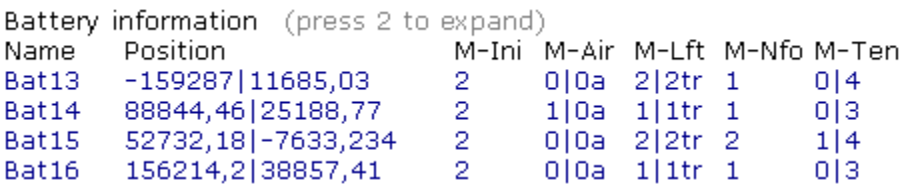


Figure 5.5. Close-up on defense system information in the simulator.

In Figure 5.5, like the statistics of the aircraft, *name* is the unique identifier and *position* is the location on the battlefield. Unlike the aircraft, missile defense systems are stationary. The *initial missiles (M-Ini)* represents the number of SAM the defense system starts the scenario with. *Missiles in the air (M-Air)* represents the batteries own missiles currently intercepting targets, whereas the *missiles left (M-Lft)* represent the remaining missiles. The second value in these two columns shows the current tentative decisions formed by the batteries; the tentative decision is carried out should no new information arrive in time to revert the decision. *Missile information (M-Nfo)* shows how many other missiles launches the battery is aware of at this specific moment. The final column, *tentative missile information (M-Ten)*, shows the amount of information the battery has received from others which are used to form its decision (i.e., if the consistency model in use allows tentative action).

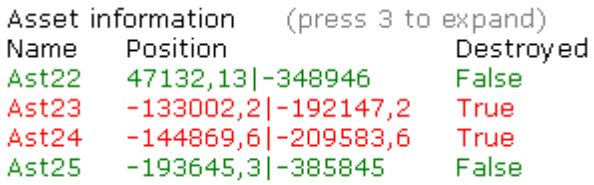


Figure 5.6. Close-up on the defended asset information in the simulator.

The statistics representing the defended assets are *name*, *position* and if they are *destroyed* (or not), as shown in Figure 5.6. Their name can be used to track which aircraft is attacking which defended asset, as this information is displayed by the aircraft. In the figure both a destroyed asset as well as an intact is being shown.

Each of the three information outputs shown in Figure 5.4, Figure 5.5 and Figure 5.6 can be expanded thus showing the entire statistics of a single battlefield entity. This is useful during scenarios involving a large number of belligerents.

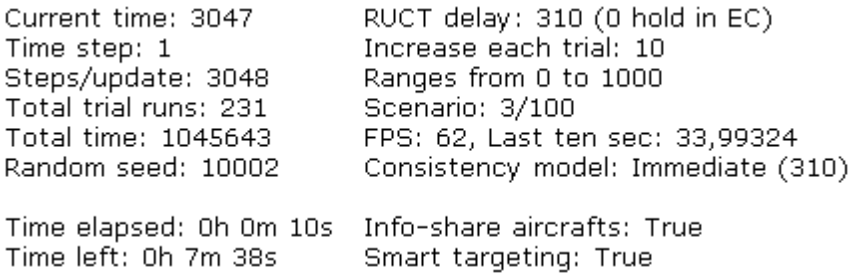


Figure 5.7. Information output and settings in the simulator.

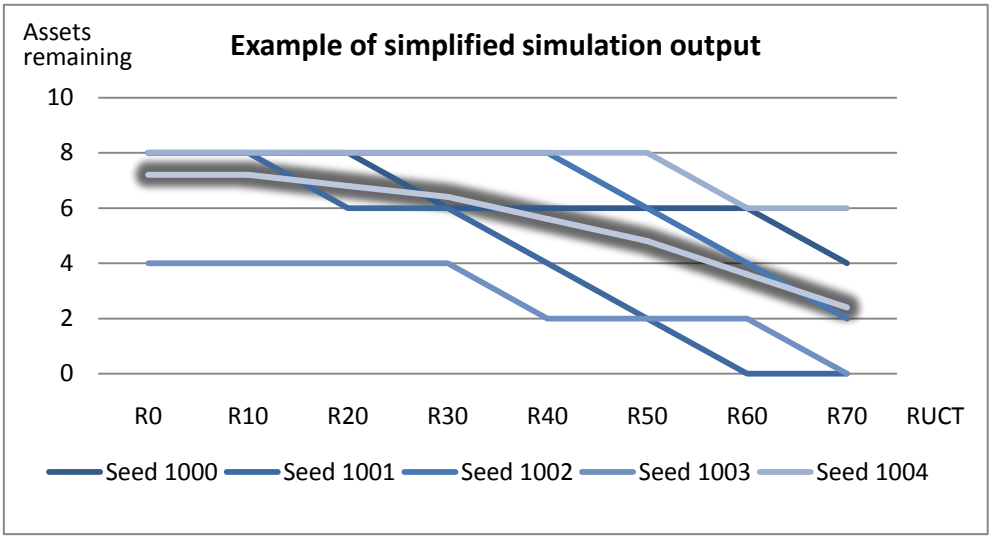
At the right top of the screen in the simulator, shown in Figure 5.7, the current scenario settings are being displayed. The *RUCT delay* is the forced communication delay between the missile defense systems, and the *hold in EC* represents the time a tentative decision is delayed before carried out when using eventual consistency. In the figure the RUCT delay is increased with 10 each trial, until it reaches the limit of 1000 at which point is resets to 0. At the current moment, 3 out of 100 scenarios have been completed. Each scenario uses a unique seed which results in new random spawn positions. These are the most important statistics, and this figure is explained in further detail in the appendix.

Once a scenario is complete, output is saved to a file on the hard drive. The file starts with the settings in the scenario, which is then followed by actual output produced during simulation. There are three matrixes; one presenting remaining defended assets in percentage, one the real number of remaining assets and finally the number of remaining SAMs. The seed and RUCT are placed in the rows and columns respectively. An example of output, although kept short on purpose, is presented in Table 5.1 using immediate consistency. Here the RUCT increases with 10 seconds each time step and there are 5 trial runs with different seeds; a scenario constructed for actual evaluations uses finer granularities and more trial runs. The numbers in the matrix represent the amount of remaining defended assets, and the initial number of assets is set to 8 in this example.

Seed	RUCT 0	RUCT 10	RUCT 20	RUCT 30	RUCT 40	RUCT 50	RUCT 60	RUCT 70
1000	8	8	8	6	6	6	6	4
1001	8	8	6	6	4	2	0	0
1002	8	8	8	8	8	6	4	2
1003	4	4	4	4	2	2	2	0
1004	8	8	8	8	8	8	6	6
Average	7,2	7,2	6,8	6,4	5,6	4,8	3,6	2,4

Table 5.1. A small portion of an output file, showing different seeds and RUCT.

Using the data from Table 5.1 it is possible to construct a graph, presented in Graph 5.2, representing the average number of defended assets remaining in the scenario. Even when using a small amount of collected data, which is done in this example, a clear trend can be seen with a decreasing amount of defended assets. In this figure all seed values are being plotted, with the average output drawn highlighted on top. In actual evaluations the data is prepared (e.g., by removing outliers).



Graph 5.2. The output from Table 5.1 with the average highlighted on top.

In the end of output the file the average values, arrived from combining all seeds, gives a quick statistical overview of the results which can be used for initial evaluations. This concludes the overview of the simulator with all important parts explained.

5.2 Constructing the simulation

In order to verify the hypothesis stated in section 3.2, a simulation must be constructed. The decision to use simulations were already discussed and motivated in the method section 4.1.4 and accompanied with the decision to use C# and XNA for the construction, presented in section 4.2.2, this part focus on the development of the simulation. This section includes all the material presented in the background chapter, but as this indeed is a simulation some parts are simplified on purpose, while other parts benefit from the fact that the scenarios run locally on one single machine thus eliminating problems occurring in network solutions.

The section cover the parts presented in section 4.3. Architectural decisions in the simulation are discussed in short detail ranging from section 5.2.1 to 5.2.10.

Functionality is clarified by examples when needed. In section 5.2.11 the actual implementation of the simulator is described briefly.

5.2.1 Using a time triggered simulation

The decision to use a time triggered simulation was discussed in section 4.3.1. Each and every part in the simulation (e.g., missile defense systems or aircrafts) is polled with activity at a constant rate. This activity is the current time in the simulation, and it is up to the parts receiving this information to act when it is appropriate.

The time is emulated and because of this actual performance on a machine running the simulation does not affect the output, and it is possible to speed up, slow down or pause this time using a multiplier. The time increase with a predetermined number each updates, which is the poll rate chosen during scenario configuration. The amounts of updates are a variable which can be set during actual simulations. If increased, more polls are sent and time progress faster.

As an example, is the update rate is 1000 and the polling interval is 0.1, then 10 seconds worth of simulations progress during that real time second. If the update rate is set to 0 then no polling occurs and the simulation pauses. Performance on the machine running the simulation is instrumental in deciding how many updates can be completed each real time second. This also stands in direct relationship with the amount of belligerents on the battlefield.

Extending the example above, the simulator also has the ability to predict the remaining time until all trials have been completed. If the predicted time is 60 real time seconds, then it means that the emulated time left in the scenario is around 6000 seconds (i.e., 1000 in update rate, 0.1 in polling rate and 60 real time seconds). During this time, 60000 polls containing the current emulated time will be sent to all parts in the simulation. If the user increases the update rate to 3000, it means that the polls are sent faster and the scenario will complete in around 20 real time seconds. However, 60000 polls will still be sent and the output is not affected at all.

5.2.2 Random number distribution in the simulation

As stated in section 4.3.2, the random number distribution in simulators should be deterministic thus enabling reproducibility of scenarios. This simulator uses a pseudo random number generator pre built into the programming language of C# (MSDN Library, 2010) which produces numbers that are both deterministic and periodic. It is important to ensure that the randomness is sufficiently indistinguishable from true randomness. Using the visualization feature, Figure 5.8 is showing four random distributions represented by multiple spawned defended assets within a predefined area. Each of the images uses different seed values, and upon closer inspection it is clear that the spawning locations differ while they at the same time do not cluster or show clear patterns (i.e., patterns can always be found, even with true randomness, but nothing clearly stands out at visual inspection).

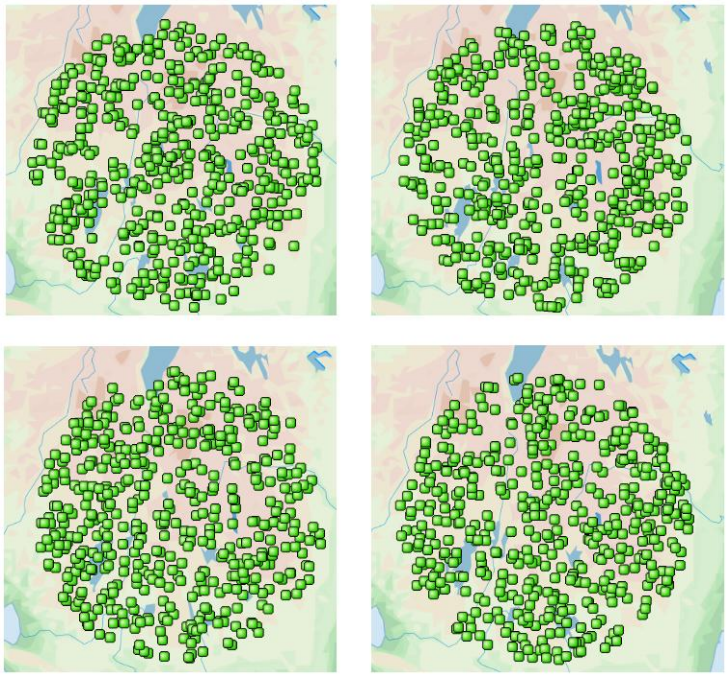


Figure 5.8. Visual validation showing four different spawning locations.

The MSDN Library (2010) states that the random-class in C# "represents a pseudo-random number generator, [which is] a device that produces a sequence of numbers that meet certain statistical requirements for randomness". C# is well known programming language approved by ISO (ISO/IEC 23270:2006), so combining the verified information from the random-class in the MSDN Library (2010) with the visual inspection in Figure 5.8 validates that the selected pseudo-random number generator is sufficient for the use in this simulation when randomly spawning units on the battlefield.

5.2.3 Observability

The simulator has already been described in detail in section 5.1.1, but there the focus was on the visual interface on the simulator. As explained in the method on observability in section 4.3.3, the reason for using graphics is to validate the simulation. The battlefield is a bird's eyes view over a map, drawn in 2D, of the southern parts of Sweden. While not at all essential for the output, the map makes validation of distances and velocities easier which is one essential part on the battlefield to validate. The user has the ability to turn on and off multiple visual validation tools, shown in Figure 5.9.

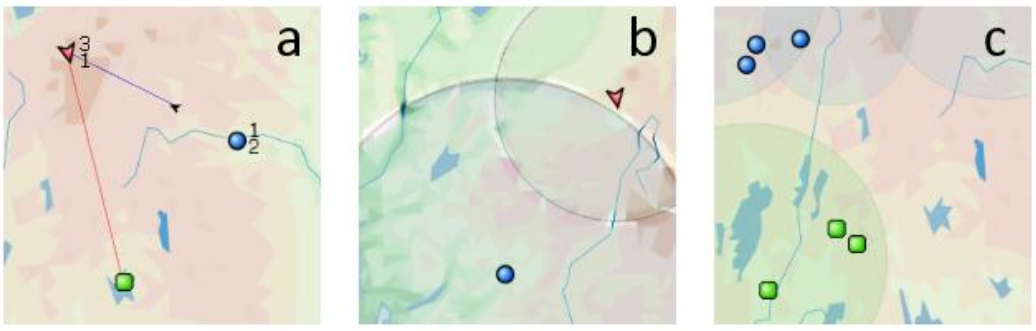


Figure 5.9. Visual validation tools.

Shown in (a), the amount of missiles remaining for each belligerent is written next to it on top, and the number below is the unique identifier for the belligerent. Further drawn, as lines, are the headings of the aircraft as well as the missile. This is useful especially when multiple missiles are launched at the same target.

In (b) the missile ranges are drawn as circles. As seen, the battery has a longer reach than the aircraft, and if the aircraft continues in the same heading it will soon enter the missile range. As the simulation can be paused and stepped forward one step at the time, it is possible to clearly see at which point in time the battery launches its missile. As an example of uttermost importance for the validation of this research, when using eventual consistency acting on tentative information, if the aircraft enters inside the range of a battery when the time is t=1000 and enters into the range of another battery at t=1005, then the first defense have 5 seconds to propagate its decision to the other battery. If failing to do so within this time frame (which depends on the RUCT), both batteries will fire its missiles. Using this visual inspection, the consistency models are validated very precise.

Finally in (c), the spawning locations of the batteries and assets are drawn, using colored circles. This is useful when inspecting scenarios and was helpful when validating the random number generator, as described in section 5.2.2. However, the most important part is that it can be used to ensure that no unit on the battlefield can spawn within another unit's missile range, as it would be an illogical situation.

5.2.4 Replica update completion time delay

As decided in section 4.3.4, the replica update completion time is emulated using a delay which is forced upon the updates. It was further decided that this delay would be one single value, and not partly distributed at places where delays might occur.

While fairly advanced in concept, the use of the delay is simple in simulation. When one part in the simulation would like to tell another part something, this information is formed as a message with a timestamp. The timestamp is the sum of the current emulated time and the RUCT delay. Once this time is reached during the simulation, the message is instantly sent to the receiver (i.e., as the time is emulated, it is possible to conduct this sending process in no time at all).

Illustrated in Figure 5.10, the RUCT delay serves as the main input variable during trial runs. Usually starting at 0, the RUCT delay is increased each trial by a preset value. This means that communication between parts in the simulation takes longer and longer time to complete, and it affects the effectiveness of the consistency models.

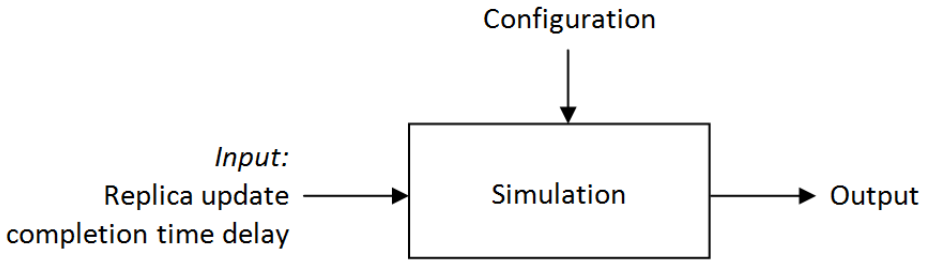


Figure 5.10. Input in form of RUCT delay to the simulation.

As an example using the current time t=1000 with a delay of 20 seconds, battery #1 would like to send information to all 10 other batteries. 10 messages are formed with information coupled with a timestamp of t=1020. At t=1015 battery #5 would also

like to send information. This means that the messages from battery #1 still is being delivered when battery #5 sends its messages. If the delay on the other hand was less than 15 seconds, then battery #5 could account for the information sent from battery #1 before sending information.

5.2.5 Placing belligerents on the battlefield

All belligerents on the battlefield randomly spawn within predetermined areas, as decided in section 4.3.5. The area uses the shape of a circle, but multiple circles can be used to form more advanced shapes. Once all areas have been mapped, a large squared area covers all locations (i.e., done separately depending on belligerent) and a random position is created within this square. The position is then tested against the spawning locations, and if it is inside at least one circle the unit is spawn at that location. If it is outside, the process repeats.

This method has the benefit of being completely uniformed in its positioning, and overlapping spawn areas does not affect this, as seen in the exaggerated example in Figure 5.11.

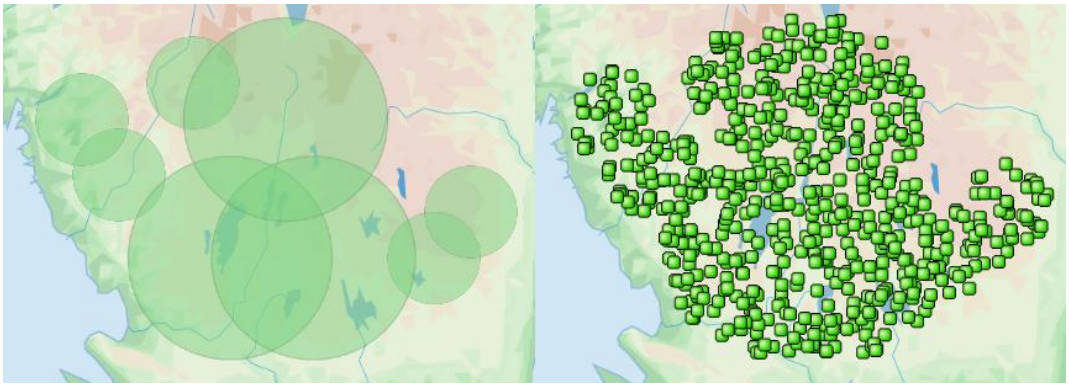


Figure 5.11. Random spawning locations overlapping, with and without assets.

If using widely separated and small spawning locations, there is a risk that the placement process will have to be recalculated multiple times as its misses the circles. As this is done only at the beginning of each trial, and not at each time step, the performance loss is negligible even in extreme situations.

5.2.6 Concurrency control

As stated in section 4.3.6, this research does not focus on evaluating concurrency control so strong strict two-phase locking process ensure that all parts in the network are isolated during updates. Concurrency control over real networks is difficult, but in simulation the locking completes in no time at all (i.e., no emulated time, and negligible real time) and the ACID rules are automatically obeyed no matter the circumstances.

All nodes with the ability to send information has a variable flag telling if they are locked or not. When using immediate consistency, as one node would like to send information it tells a locking function of its intent, and this function flags all other nodes as locked. When the update has been completed, the lock is released the same way. Using eventual consistency, the entire network is never locked so flags are never used. Sending information between two nodes is done one at the time, so conflicts can never occur.

5.2.7 Consistency model

Two consistency models are used in simulations; the immediate consistency and the eventual consistency acting on tentative information.

When using immediate consistency in this simulation, a node can instantly lock all other nodes to commit its update in the network. It is not possible for locked nodes to conduct any communication with other parts for the duration of the RUCT. When an aircraft is detected, the missile defense systems have to agree upon who should fire. A central abstract function makes this decision. Using a central process might seem contradictable to section 2.1.1, but it is used as a simplification of the communication between nodes. As pointed out in section 4.3.7, what really happens during the update is irrelevant to the research; the only important thing is that they form a decision which they all can agree upon. Finally, more than one decision cannot be formed at the same time, as all nodes are locked. However, more than one aircraft can be included in the decision (e.g., 10 missile defense systems can calculate and decide how to bring down 10 aircrafts in one single decision).

The use of eventual consistency in the simulation is more advanced. No single system can locally decide what nodes should do; thus nodes form their own decisions. Once a decision has been made, this is propagated out in the network of nodes. As explained in section 5.2.4 and 5.2.6, the message (i.e., decision) is delayed for the duration of the RUCT, but once it is sent no conflicts can occur. The decision can be a tentative decision which is delayed, or a concrete action taken by the battery. All nodes keep track on what is happening on the battlefield, and they use this information to form their own decisions. If nothing happens during a period extending over the RUCT, all nodes share the same information (i.e., the network eventually get consistent).

5.2.8 Layered decision making

The nodes in the eventual consistent system can form tentative decisions, called primary decisions, as explained in section 2.4.4 and outlined for the use in simulations in section 4.3.8. If no other information is received for a preset amount of time, the node acts on its primary decision. If information from other parts is received during this time, the node can instead form a secondary decision and act on that instead.

Two layers of decisions are being used. In the simulation, when a battery detects an aircraft it forms the decision whether to fire or not. This is the primary decision, which is tentative and the node hold the decision for a predetermined amount of time. At the same time it sends information to the other nodes that it intends to launch a missile, and at which time it will do so. If the node receives information during the decision hold that another node will fire before its intended launch, the primary decision is reformed into a secondary decision not to fire. The secondary decision is never propagated to other nodes. In these simulations, as a simplification, all internal clocks in the nodes are accurately synchronized. This eliminates the problem of very close calls and how nodes should act if the primary decision to fire is within the margin of error in time synchronization.

The layered decision delay time needs to be specified, and is variable during scenario configuration. It is highly dependent on the situation, and it would require domain knowledge when setting this time in a real world scenario. In simulations, short test scenarios can quickly determine what the optimal delay time is. As a rule of thumb, if the delay is set low it will never affect the eventual consistency in a negative way, but

increasing the delay time could bring more positive effects and, if set to high, have a negative impact on the consistency model.

5.2.9 Belligerents and resources

Battlefield belligerents and missiles are simplified as much as possible in the simulation, as explained in section 4.3.9. Only the most important attributes which constitutes to the research are used.

All speeds and directions are implemented as floating point numbers, the number of resources as integers, and finally if they are destroyed or not by the use of Boolean values. Aircraft carries a number of ATGMs, which are attached to the aircraft until the point where they are released, as they afterwards acts as an autonomous unit on the battlefield. This is the same with SAMs launched from missile defense systems.

As stated in section 4.3.9, aircrafts share information between each other. In contrast to the more rigorously designed communication between missile defenses, the aircrafts communication is fundamental in design and purpose. As they spawn on the battlefield, they need to head toward a mission objective which is a defended asset. The closest one is the most reasonable to attack, but this usually means that all aircrafts attack the same defended asset, which decreases observability. Starting with one aircraft, the closest asset is selected by calculating the distance to all. Once the closest has been selected, its decision range is increased by 5%, which is a variable tested in simulation to ensure a good distribution, but has no impact on the outcome of the scenario. After this point, the next aircraft selects another defended asset to head toward. When accounting for those already selected, its calculated range is 5% longer which decrease the likelihood of choosing the same (i.e., it is still possible, if it is considerable closer than other assets on the battlefield). Once an asset has been destroyed, new ones are selected the same way.

5.2.10 Threat evaluation and weapon allocation

This simulation uses a simpler TEWA than that presented, for instance, by Johansson & Falkman (2008) and Naeem et al. (2009) since the entire battlefield setting with TEWA is just used to evaluate consistency models. As stated in section 4.3.10, all targets are treated with the same threat level.

The weapon allocation is more advanced. Using immediate consistency, the current trajectory of the aircraft is taken into the calculation and it is decided which aircraft will be closest to specific missile defenses when the decision is finalized. As the kill probabilities are omitted in this research, the decision on which battery should fire rests on the amount of remaining missiles and the distance to the target at that point.

The weapon allocation is more straight forward using eventual consistency. As the missile defense systems in this research act as their own radar station, and the radar range is set to the missile range during simulation, the battery will always form the decision to fire unless it has no remaining missiles or already received information that another battery will launch a missile. When using layered decisions, the amount of missiles or range to the target is not taken into consideration; the one missile defense first allocating the target will fire its missile.

5.2.11 Implementing the simulator

As decided in section 4.2.2, the simulator is created using C# XNA. Only essential parts are detailed in this section due to extent of the code, and most of the architectural decisions have been covered in section 5.2.1 to 5.2.10.

The class *Battlefield* is the central part of the simulator connecting all belligerents and resources, and these classes are illustrated in the UML Diagram 5.1. The simulator is driven forward by the main loop, in the class *Main*, connected to the Battlefield.

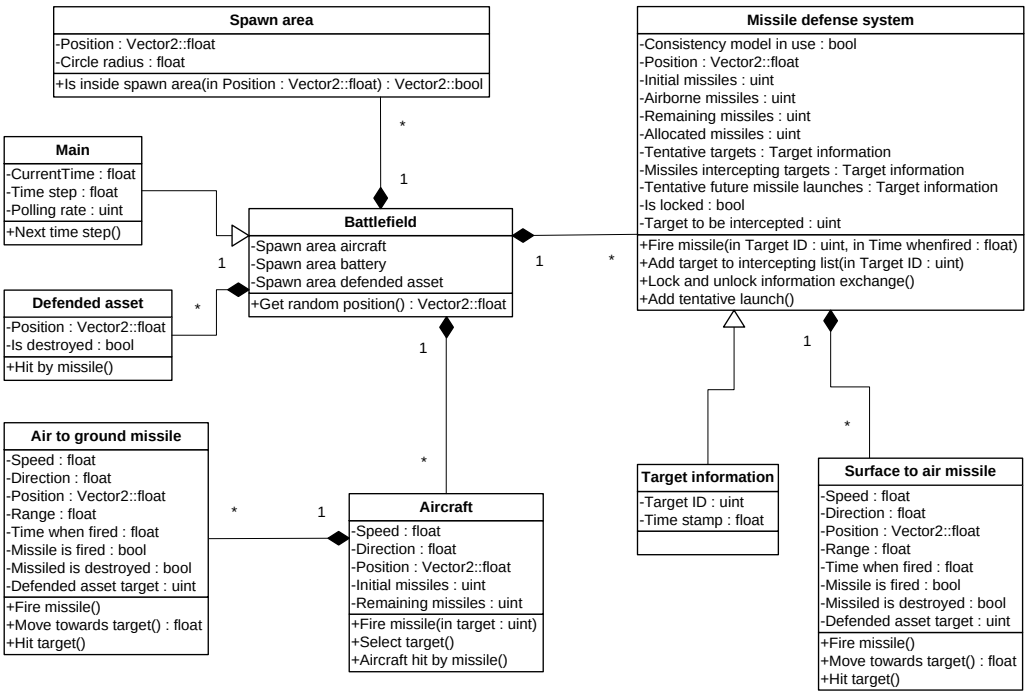


Diagram 5.1. UML diagram of the simulator.

The main loop keeps track of the current time, how many polls should be processed each real time second as well as the time interval between each poll. The main loop only interacts with the class *Battlefield*, which receives the polls and the delta time since last poll. Since the time interval do not change during simulation, the delta time is always identical and is not affected by actual computer performance.

Connected to the missile defense systems are the three primary units on the battlefield, as well the random spawn areas. Connected to the aircraft and missile defense systems are the ATGMs and SAMs respectively. When the time progress, the battlefield tells all units to update, and these afterwards inform the battlefield of their status and position. The battlefield then calculates if an aircraft is within the range of a battery or if a missile currently intercepts a target. It keeps track on the resources on the battlefield, and once all aircraft has been destroyed or left the battlefield, the scenario restarts using either a different seed or the same seed with an increased RUCT delay. At the same time the results from the trial run is saved to a class (not shown in the UML diagram). Once all trials have been completed, the results are saved to an output file on the hard drive.

One complete run in the simulator, step by step, is outlined in the appendix as it is lengthy and of less importance to the actual research.

5.3 Constructing scenarios

During the development of the simulator simple scenarios was constructed and tested, in order to validate the implementation. This section describes actual parameter settings as well as the standard scenario used in the primary evaluations. In section 5.3.1 aircraft and missiles are selected and the choices are motivated, and section 5.3.2 then validates the simulator using the selected resources. Section 5.3.3 covers the defense deployment and section 5.3.4 outlines a base scenario.

5.3.1 Research on aircrafts and missile defense system attributes

Finding accurate and reliable information about modern-day aircrafts and missiles is difficult. Even more difficult is to know their actual performance on the battlefield since it is one thing to theoretically be able to fly at a certain speed and another thing to actually be doing it. The military setting is used to test a hypothesis and even when modifying these values greatly, the hypothesis stays valid. With this in mind, the actual attributes of aircrafts and missile defense systems presented here are approximations at best, and uses less reliable sources than all other parts of this research. The first step is to find modern-day aircrafts, ATGMs and SAMs.

- *Aircrafts*. Most modern-day military powers uses 4^{th} or 4.5^{th} generation aircrafts, while some 3^{rd} generation aircraft still remains in service (e.g., Saab 37 Viggen, a 3^{rd} generation aircraft, was retired in 2005 from the Swedish Air Force). Multiple 5^{th} generation aircrafts are being developed but only one is in active service, which is the F-22 Raptor (U.S. Air Force, 2009). Since most aircrafts perform very similar, the choice is not critical and three 4.5^{th} generation aircraft has been selected for this research; the Boeing F/A-18E/F Super Hornet, the Eurofighter Typhoon and the Saab 39 Gripen. The F-22 Raptor is, at least on the paper, the most capable fighter aircraft in the world and is included in this research as well.
- *ATGM*. The air-to-ground missiles should be compatible with the selected aircrafts. Since the defended assets used in this research only are identified as objects of substantial value (section 2.3), the ATGMs selected can vastly differ in design. This is something positive, showing that the hypothesis is valid in multiple situations. The AGM-65 Maverick, the AGM-154 Joint Standoff Weapon, the AGM-88 HARM and the Taurus KEPD 350 all have different design goals and fulfill the requirements since they can be used on the selected aircrafts.
- *SAM*. This is perhaps the most difficult battlefield resource to select since there are numerous SAMs on the market still in active service. There performance and design goals greatly differs. Four missiles, which are famous and highly distinguishable in performance, have been selected; the MIM-104 Patriot PAC 2, the V-75 SA-2 Guideline, the S-300PMU/SA-10 and the FIM-92 Stinger.

Once the aircrafts and missiles have been selected, the attributes selected for the use in this research needs to be identified. The information about this comes from official web pages and respected military web pages. However, since these aircrafts and missiles are being sold by private companies, even official figures can be incorrect or exaggerated. As mentioned, this is of less importance when proving the hypothesis.

The four aircrafts used in this research are listed in Table 5.2, showing the only important attribute in this research; the speed. The listed speed is official information, but multiple aspects with this number remain questionable. First of all, the speed is listed in Mach, which is the speed of sound. At sea level, this is 340.29 meters per second, but at altitude this decrease, and so does the wind resistance applied on the aircraft. Saab AB (2010) does state that their Saab 39 Gripen can travel at supersonic speed at all altitudes, which means that it travels in at least 1 Mach at sea level. The listed speed is at high altitude. Traveling at maximum speed also require the use of afterburners, which increase the fuel consumption to a level unsustainable for actual long-term flight (Flack, 2005, p. 19).

Aircraft	Listed speed	Source
Boeing F/A-18E/F Super Hornet	Mach 1.8+	Boeing Defense, Space and Security, 2010
Eurofighter Typhoon	Max Mach 2.0	Eurofighter Typhoon, 2010
Saab 39 Gripen	Mach 2.0 and supersonic at all altitudes	Saab AB, 2010
F-22 Raptor	Mach 2 with supercruise capability	U.S. Air Force, 2009

Table 5.2. Specification of the aircrafts used in this thesis.

The engines used in the listed aircrafts all have capabilities of sustaining a speed over 1 Mach for an extended period of time, called *supercruise*. This is done without the use of afterburners, which in turn keeps the fuel consumption at a reasonable level. A 5^{th} generation fighter aircraft can by definition supercruise (U.S. Air Force, 2009), and Saab 39 Gripen was able to supercruise at Mach 1.2 at 28,000 feet (Saab AB, 2009), which equals 360 m/s or slightly higher than the speed of sound. However, according to Cumpsty (2003, p. 181) fighter aircrafts carrying significant military payloads usually travels subsonic, at Mach 0.8-0.9. This is more likely the speed fighter aircrafts travel during missions; only in situations where they engage in air-to-air combat do they increase their speed close to the maximum specification.

The ordinance (i.e., ATGM) carried by the aircrafts is listed in Table 5.3, and includes very different types of missiles. The AGM-154 Joint Standoff Weapon is a precision strike weapon designed to engage different types of ground targets at range. It can carry different kinds of explosives. The AGM-65 Maverick is a tactical missile used for close air support. It can attack armor, air defenses and fuel storage facilities or other objects of medium size on the battlefield. The AGM-88 High-Speed Anti-radiation Missile (HARM) is used for countering and destroying enemy radar-equipped air defenses. The last missile is the Taurus KEPD 350, which has a long reach and can breach bunkers and attack other ground targets.

Aircraft	Speed	Range	Source
AGM-154 Joint Standoff Weapon	Subsonic	Low-altitude, 22 km; high-altitude, 116 km.	The US Navy, 2009a
AGM-65 Maverick	Supersonic	31 km	The US Navy, 2009b
AGM-88 HARM	Sonic at 340+ m/s	91+ km	The US Navy, 2009c
Taurus KEPD 350	Mach 0.6 - 0.95	500+ km	Taurus Systems GmbH, 2010

Table 5.3. Specification of the ATGMs used in this thesis.

Unlike the aircrafts, it is more likely that these missiles in fact travel at their design speed. As two of the missiles have diffuse speeds (subsonic and supersonic), their speed is set to 320 km/s and 360 m/s, to represent the slight difference. However, once a missile has been fired no counter measures can be taken so the actual flight speed is just used to keep the scenario realistic. Of greater importance is the range, as a long range perhaps (i.e., scenario dependent) let the aircraft fire its missile before the defense systems on the ground has been able to intervene. Listed in Table 5.3, the AGM-154 Joint Standoff Weapon has a range between 22 and 116 km, depending on altitude. It is possible to use both these values and others in between during simulation.

The most important resource on the battlefield in this research is the SAM. As an example, if the SAM has a short range there is less time to communicate between batteries geographically separated. Because of the importance, four missiles have been selected with different specifications, listed in Table 5.4.

Aircraft	Speed	Range	Source
MIM-104 Patriot PAC 2	Mach 5	160 km	Global Security, 2008a
V-75 SA-2 Dvina/Guideline	Mach 4.5	50 km	Global Security, 2008b
Almaz S-300PMU/SA-10	2000 m/s	150 km	Kopp, 2010
FIM-92 Stinger	750 m/s	1-8 km	Global Security, 2005

Table 5.4. Specification of the SAM used in this thesis.

The MIM-104 Patriot PAC 2 is a missile produced in great numbers and widely distributed around the world. It is used as an aerial interceptor and has a great range. The V-75 SA-2 Dvina/Guideline has also been produced in great numbers, and is still active in multiple countries although it is an older design. Its range is noteworthy shorter than the MIM-104 Patriot, and since their goals are identical it is interesting to use both in this research. The Almaz S-300PMU/SA-10, said to be one of the most capable missiles currently in service (Kopp, 2010), can intercept fighter aircrafts but also cruise missiles traveling at Mach >10 and stealth fighters depending on the exact missile model. The range of the S-300 is also depends on the model, and could range from 40 km to 400 km, but used in this thesis is 150 km since it is verifiable. Finally, the well known FIM-92 Stinger is used in this research to show what impact the range has upon the hypothesis, as this missile weapon is handheld, small in size and with a range at least ten times shorter than that of the other missile defense systems. While

FIM-92 Stinger is not constructed to bring down 4.5th generation aircrafts, although possible, it still applies a noteworthy impact on the hypothesis and can thus be used for evaluations by comparing it with other missile defense systems.

5.3.2 Knowledge about the simulation

When acquiring knowledge about the simulation, which is something essential in order to construct scenarios, multiple small and specialized scenarios are constructed. As explained in section 4.4.2, an iterative process is used, as it is not possible to test one setting and move on without going back later. When a relationship or variable can be definitely defined, this is done in detail. However, most relationships are highly scenario dependent. All settings have been tested to the extreme ends and the important parts found in this iterative process are outlined next.

- *Spawning location of aircrafts and missile defense systems.* Distances between belligerents greatly affect the outcome. The spread can consist of three primary situations and stand in direct relationship with each other. First, the aircrafts can cluster together close or travel in a column. This increase their chance of survival rate should they not pass over cluster of missile defenses or another column. Secondly, the aircrafts can line up attacking at a wide front. This increases their survival rate unless the defended assets also line up to meet the aircrafts. Finally, if aircrafts spawn randomly within a large area the best general defense is to spawn missile defenses within a random large area as well. While not performing at maximum potential, this deployment does not leave any bottlenecks in the scenario which is something to prefer.
- *Aircrafts relationship to defended assets deployment.* The deployment of the defended assets does not affect the outcome at all if the aircrafts communicate their attack routes and plans between each others, which is a variable setting and likely used in real combat. If all aircrafts act solo, spreading the defended assets increase their kill probability.
- *Number of aircrafts and missile defense systems.* Of great importance to the outcome, if there are numerous missile defense systems and few aircrafts, less defended assets will be destroyed. This stays even truer when using eventual consistency acting on tentative information, as the missile defense systems then tend to over neutralize targets without wasting too much shared physical resources (i.e., they already outnumber the aircrafts). This also works in reverse.
- *Number of SAMs.* Information communicated between missile defense systems can take time, but a local missile defense system always keeps track of its own missiles. Thus, one strategically placed missile defense system with 10 missiles in store can easily shoot out 10 approaching aircrafts, as decisions are local and instant. Using few missile bases with multiple SAMs however makes the evaluation of the consistency models impractical, and by only using one missile system an evaluation is impossible. This means that one or perhaps a few missiles each gives the most valuable output.
- *Number of ATGMs.* If missile defense systems are within attack radius when aircraft approach their goals, the number of ATGMs affects the output. As a strategic decision it would however be quite useless to let an aircraft destroy an asset and afterwards bring down the aircraft. With this in mind, missile defense systems try to bring down aircrafts before they enter their missile

range; if this is the case, the number of intact defended assets stands in direct relationship with the amount of ordinance carried by the aircraft. This direct relationship is partly avoided by letting the assets stay close to the missile defenses, but at the same time not to close. Thus, it should be possible for one aircraft carrying two missiles to first bring down one defended asset but then be destroyed on the way to the next.

- *SAM range.* The range has two impacts on the scenario. First of all, a shorter range leaves less time to communicate decisions. Secondly, a shorter range easily leaves gaps in the defenses, which increase the need to use more missile defense systems to meet the incoming threat. The actual relationship between the battlefield resources is scenario dependent, and works in reverse with longer distances.
- *SAM speed.* As destroyed aircrafts directly reads as "destroyed" for other missile defenses, a higher speed of missiles decrease the risk of missiles being used in over neutralization the targets. The higher the speed of the missiles, the less useful is the scenario for evaluating the effects.
- *Aircraft speed.* The faster the aircraft moves, the less time does the missile defense systems have to communicate decisions, and vice versa. Doubling the speed would result in half the time to communicate the decision, as this relationship is close to linear.
- *Number of simulation runs.* The number of simulations using the same configuration file is one of the settings less scenario dependent. After 10 simulation runs, no matter the other settings, the output usually lies within 10% of what is to be considered the absolute average. This is sufficient for initial validations of a configuration. At least 100 simulation runs is preferably when extracting output for actual evaluations.
- *RUCT delay setting.* This value is used to force a RUCT on communication. As explained in section 2.2.2, at some point increasing the RUCT further would equal not actually being able to communicate information between the defense systems. This point is scenario dependent but easy to find; if aircrafts travel slow and SAMs have a long range, then this value will be high, and vice versa.
- *Simulation steps.* As the time is emulated, one time step can consist of any number (e.g., 10 seconds or 1 millisecond). The simulation becomes more precise with a short time step. At some point, decreasing the time step does not affect the output at all, but it does further decrease the simulation performance. The actual time step is scenario dependent, with speeds of aircraft and ranges of SAMs as the primary factors. Using the selected resources, somewhere around 1 second is preferably as a time step, but quick output can be achieved using everything up to 10 seconds (i.e., as an aircraft travels 3000 meters in 10 seconds, this is a rather extreme distance as a single time step).
- *More belligerents, less spread in results.* If the number of aircrafts, missile defense systems and assets increases, spread between output decreases. If the scenario employs 5 aircrafts and 5 missile defense systems, then there is a chance that they will spawn on opposite parts of the battlefield creating an unlikely scenario (i.e., an outlier). However, if there are 50 of each belligerent,

the chance that this would happen decreases drastically. The amount of outliers in both scenarios is actually the same, but the spread is less evident.

- *Number of battlefield resources.* The number of missile defense systems has an almost exponential effect on the performance of the simulator, since they all communicate with each other. If the minimum is set to 10 defense systems, 20 defense systems lowers the performance of the simulator to 30%, and 50 defense systems outputs 1% in speed performance as compared to 10 systems. The amount of missile defense systems does not affect the output as much as other parameter settings do, so using more than 20 missile defenses is not useful. Using more defended assets does affect the spread of the data set while not affecting the median value; during testing at least 30 defended assets was found to produce results which were easy to present graphically, while it did not affecting the actual outcome or result.

Testing constructed scenarios quick, to get a good overview, is important. Knowledge acquired in this section validates that even short and fast scenarios outputs semi-valid data. If the target scenario configuration takes one hour to complete and run 100 simulations with 1 second time steps, then an identical scenario configuration with 10 simulations and 10 second time steps would take roughly one half of a minute to complete (i.e., 100 times faster). These results have been validated to stay within reasonable ranges, even if they are not suitable for actual evaluations.

It has been shown that extreme values do not affect the output in a negative way; just that other resources have to weigh up for these extremes. As an example, if there are 10 aircrafts and 10 missile defense systems, and their interaction on the battlefield produce relevant output, increasing the aircraft speed drastically will suddenly give the missile defense systems a hard time solving their task. In this situation, more missile defense systems or just longer missile ranges (i.e., switching missile types) can effectively counter the extreme aircraft speed. Thus, it does not matter which SAMs the missile defense systems uses as long as their performance stand in a realistic relationship with the aircrafts. This works the same way with all parameter settings.

Finally, locating the upper limits of the RUCT delay decrease the time a simulation has to run before the seed is changed and the simulation is being reset. The eventually consistent system acting on tentative information reaches this point rather quickly as compared to the immediately consistent system. Once this point has been reached, increasing the RUCT should not affect the output further (i.e., not even an infinite RUCT would impact the results). As stated, this point is highly scenario dependent, and running short test scenarios of the intended final configuration quickly identifies the upper boundaries. The final configuration RUCT delay is then set to a value slightly above this.

5.3.3 Aircraft attack mission and defense deployment

As stated in section 4.4.3, the goal with this section is to create a single advanced defense deployment scenario with aircraft attacking from different angles. The result from this advanced scenario is then compared to the results of a small portion of the battlefield ensuring that these simplifications of the battlefield do not affect the result (i.e., the hypothesis). Settings used in this validation derive from the previous two sections (5.3.1 and 5.3.2).

The first scenario constructed is the small portion of the battlefield. It is pictured in Figure 5.12 and do not look very realistic, as the aircrafts mindlessly moves over the missile defense systems (i.e., if there would have been any battlefield reconnaissance prior to the attack, the aircrafts would choose a different approach). On the battlefield there are 10 of each primary unit, carrying one missile each. The scenario is easy to construct (e.g., weighing of resources). The scenario also fulfills the primary goal with using aircrafts and missile defense systems; decisions are being communicated between nodes (i.e., defense systems), and if it takes time to send these decisions it will impact the result.

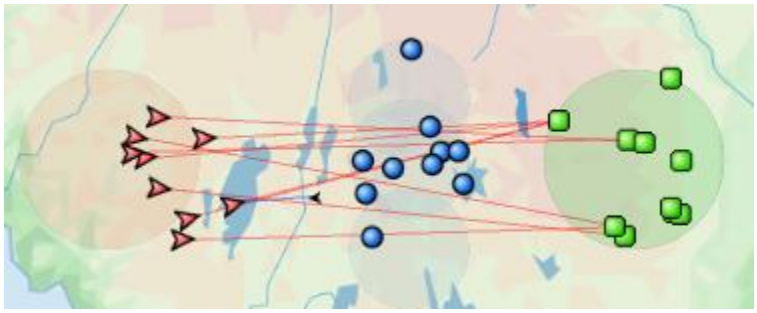
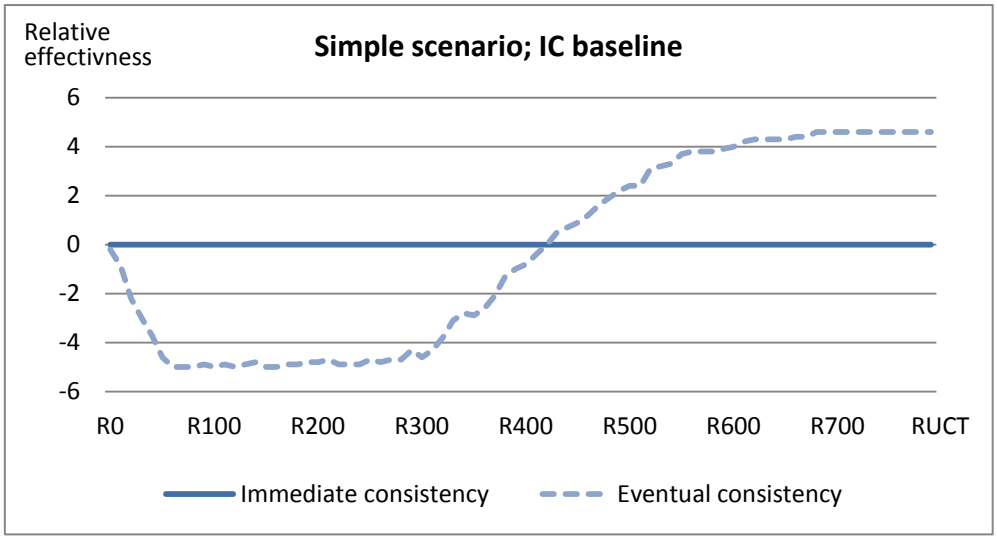


Figure 5.12. A cut out of the battlefield, showing attack angles, units and spawn areas.

This specific scenario runs 10 different simulations using the same configuration file, and seen in Graph 5.3 the output of eventual and immediate consistency is plotted on a graph where the vertical axis represents the difference in remaining resources and the horizontal axis is the RUCT in time steps of 100 seconds.



Graph 5.3. Simulation results from a cut out of the battlefield.

Constructing a scenario which would make sense on a battlefield is more complex. The most important part, at least when looking from a bird's eye view, would be to surround defended assets with missile defense systems. This means that no clear and obvious way around the missiles exists and the pilots will have to navigate through in order to reach their mission objectives. An example of this is illustrated in Figure 5.13, where pilots travel close together in order to press through defenses, and the deployed missile defenses are covering a much larger area compared to the portion in Figure 5.12. It should be noted that this scenario is not to be considered as a realistic military deployment on the battlefield, but rather something that greatly differs from the simple scenario.

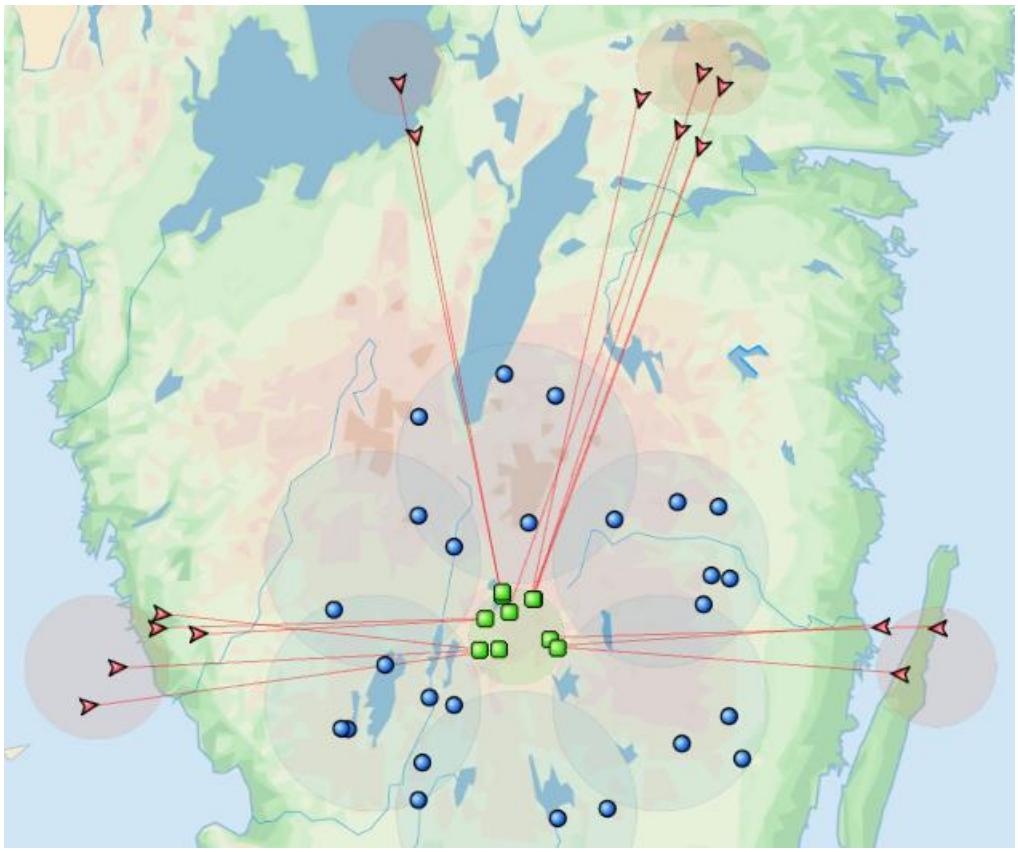
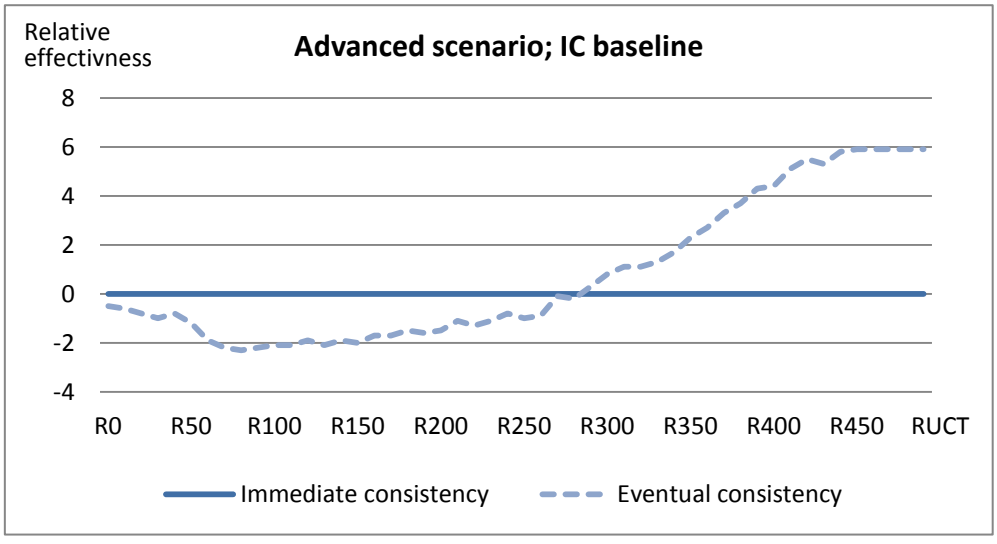


Figure 5.13. A complex scenario covering units with multiple attack angles.

Similar to the graph presented for the simple defense deployment, Graph 5.4 shows the output of remaining resources given the two different consistency models. While not identical graphs, they clearly show the same tendencies as the immediate consistent defense system outperforms the eventual consistent up to a certain point where its performance decreases.



Graph 5.4. Simulation results from a more complex layout of the battlefield.

Following these simulations it has been shown that no matter the defense deployment, the same tendency appears. This is sufficient evidence to use a portion of the battlefield during actual evaluations. It should be noted that the goal was never to create scenarios producing identical output, so the difference actually detectable (e.g., the portion of the battlefield seems to perform better using immediate consistency)

depend on the time spent in constructing these scenarios. Instead the slightly different results are positive as they show that the breaking point exists even without micromanaging scenarios to perfection.

5.3.4 Base scenario

As stated in section 4.4.4, the base scenario is used for all major simulations producing the output which is then used in evaluations. This section covers specific choices in the scenario, and motivates why they are selected. Pointed out on several occasions, the attributes of battlefield belligerents should derive from real military resources but the specific scenarios should just be used to verify the hypothesis and the entire base scenario is formed around this aspect. This means that resources will be weighed against producing the most valuable output possible.

In section 5.3.1 multiple aircrafts and missiles were selected for the use in simulations. One of each has been selected for the base scenario, and is listed next.

- *Aircraft*. The selected aircraft is the Boeing F/A-18E/F Super Hornet. The choice is of the least importance, as the aircrafts only differs slightly. The speed remain an unknown factor, but following Flack (2005, p. 19) and Cumpsty (2003, p. 181) the aircraft most likely travels subsonic during missions. The speed is set to 300m/s (equals Mach 0.9 at sea level, and around Mach 1 at altitude).
- *ATGM*. The missile carried by the aircraft is the AGM-154 Joint Standoff Weapon. It travels between 22 and 116km depending on altitude (The US Navy, 2009a). The distance is set to 100km and the speed is set to 340 m/s.
- *SAM*. Having the most significant impact on the simulation is the SAM. The selected missile is the MIM-104 Patriot PAC 2, and the speed is set to its maximum potential of Mach 5 (1700 m/s) and the range is set to 160km.

As shown in section 5.3.3, a portion of the battlefield can be used. The spawning locations need to be separated by such distance that belligerent do not spawn within the own missile range. Aircrafts attack from the north, traveling over the missile defense systems to reach their objectives in the south. As for the aircrafts, they will carry 1 to 8 ATGM each and that the missile defense system each employs two SAMs. The reason for using more missiles than one is because the spread of the output becomes more evenly distributed. Tests did show that it however not affects the hypothesis. The number of defended assets was first set to 10 and tested, but as each step in lost assets represented 10%, this number was increased up to 50. It was decided against using more than 50 assets because of the performance loss in the simulator.

With these configurations decided, the number of aircrafts and missile defense systems is set to an appropriate level. The definite numbers are found by running test configurations. To get the most output of this setting, 25 aircrafts and 16 missile defense systems deploys on the battlefield. This levels the output from the eventual consistency at 50%; however, using 35 aircrafts or just 15 aircraft would also be sufficient to verify the hypothesis, so the actual number is of less importance. Shown in Figure 5.14 is the initial setting of the base scenario, with the correct number of belligerents as well as attack directions, remaining number of missiles and spawn areas outlined.

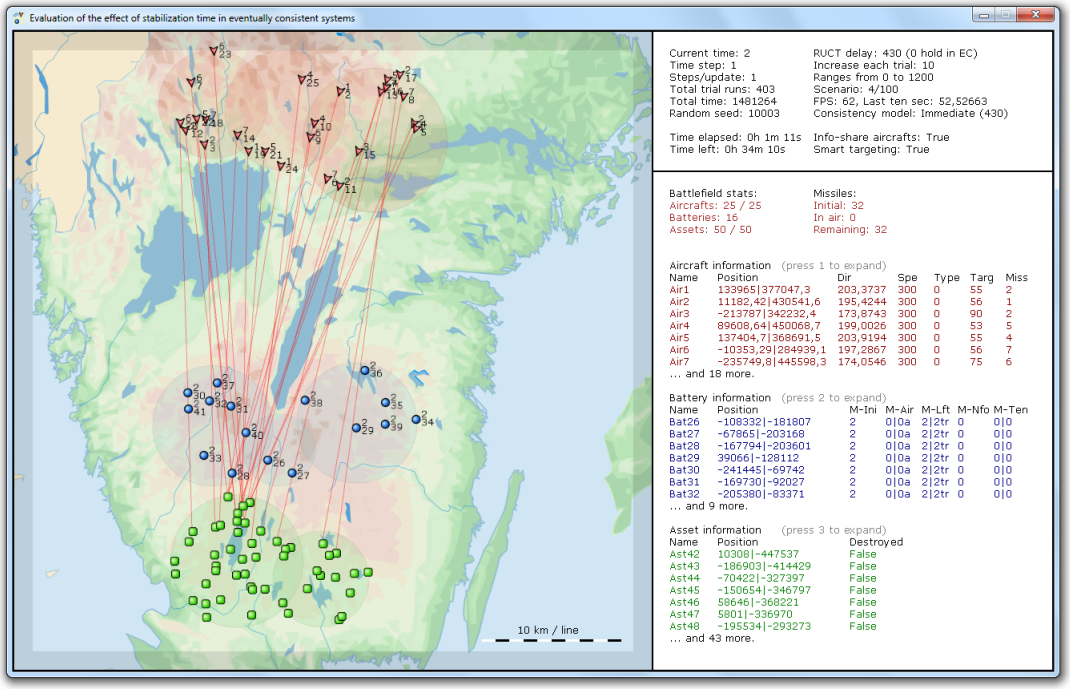


Figure 5.14. The base scenario, showing all belligerent on the field in their initial positions.

Finally, number of scenarios, time steps and RUCT delay are selected. 1000 scenarios are used in the final evaluation. The time step is set to 1 second, which means that an aircraft move 300 meters each update. In comparison, due to the size of the battlefield it takes an aircraft close to an hour to move across in real time. The RUCT delay limits are tested, and after 1200 emulated seconds the RUCT is beyond the point where a further increase makes any difference. Each increase in RUCT is set to 10. This concludes the base scenario, and the entire configuration is covered in the appendix.

5.4 Extract information, evaluate and present results

This section covers the running of the simulations in section 5.4.1, followed by the data preparation in section 5.4.2. Once the data is prepared, graphs showing the output are constructed and presented in section 5.4.3.

5.4.1 Running the simulations

As the base scenario was completed in section 5.3.4, the goal with this section is to run the scenario making sure to extract the information needed for evaluations. All scenarios are being executed using graphics, as two short test runs determined that the actual performance loss was less than 3%. Running scenarios for hours with graphics validates that they are on the right track, with no unforeseen behaviors occurring. The following scenarios are executed.

- *Immediate consistency*. Running 1000 scenarios, with 120 trials each.
- *Eventual consistency*. Running 1000 scenarios, with 12 trials each. After 10 trials, increasing the RUCT did not further affect the scenario. Thus, when comparing the two consistency models the output from the eventual consistency is filled out with the value from the final trial run in each of its scenarios.

- *Eventual consistency with layered decision making*. Running four different configurations with 100 scenarios and 12 trials each. Time steps of 5 seconds. Decision delays set to 50, 150, 500 and 700.
- *Immediate and eventual consistency, more and less aircrafts*. Using the base scenario with 30 aircrafts. Running 100 scenarios, with respectively 120 and 12 trials each. Repeated using 20 aircrafts. Time steps of 5 seconds.
- *Immediate and eventual consistency, using short range missiles*. Using a short range scenario with the same aircrafts and the FIM-92 Stinger used as the SAM. Using 5 aircrafts with 1 missile each, 3 FIM-92 Stingers launch locations with 2 missiles at each location and 5 defended assets. RUCT delay step 1, maximum RUCT at 100. Running 500 scenarios.

The simulations took approximately 5 hours to complete, using the full capacity of the computer. It should be noted that during the validation of the simulator, scenario taking less than 10 seconds to complete also provided useful information closely resembling the hypothesis. The final simulations however distribute the data better, making the removal of outliers less situation dependent also ensuring that as many situations as possible (e.g., random spawning locations) have been covered.

5.4.2 Prepare data for evaluations

The output produced from the simulations need to be prepared before it can be evaluated. The data is formatted in large matrixes, and it is not possible to simple present hundreds of thousands of values and draw conclusions. Following the method in section 4.5.2, the data preparation consists of the following steps:

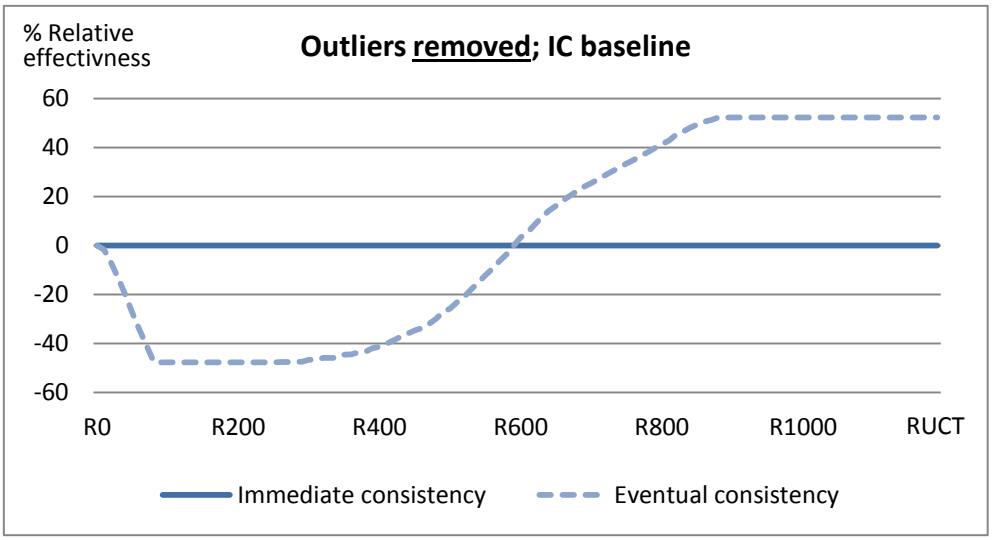
1. *Calculate true average*. Including outliers, the average values of the simulations are calculated before further preparing the data.
2. *Calculate the quartiles*. In order to detect outliers in the data set, the quartiles must be calculated. The first value calculated is the median of all scenarios; not to be confused with the average value. If each scenario included 120 trials, then 120 median values is calculated. After that the quartiles is calculated. This is done using Microsoft Excel, resulting in the first and third quartiles as well as the upper and lower boundaries.
3. *Locating outliers.* Everything outside the upper and lower bounds is considered an outlier in the data set. A new matrix is created, identical in size compared to the output. Each position in the matrix conducts a test against the original output and the newly calculated quartiles. If the value is within the upper and lower boundaries, the original output is copied to the new matrix position; else the value is set to NULL. In this way all outliers are located.
4. *Calculate the average.* Out of the remaining values in the data set, once the outliers have been removed, the average value is calculated.
5. *Preparing the baseline.* The immediate consistent system is used as a baseline. This means that it is a straight line, whereas the eventual consistency forms around this line. The difference is calculated, where the actual immediate consistent system now equals 0 in all steps, and the eventual consistent system either is above or below this value given a certain RUCT.

This process is repeated for all scenarios, including the amount of remaining missiles, but no baseline is calculated for the missiles. Once all the data has been prepared, the graphs and box plots is formed to represent the output.

5.4.3 Presenting the results

Presented in this section is the main body of the research, put together in graphs. They are ordered in the priority of this research. Unless specifically stated, outliers have been removed from the graphs. When specific times in graphs are discussed, it is done in relation with the currently used scenario.

Starting with the output from the base scenario, Graph 5.5 shows simulations using both eventual consistency as well as immediate consistency. The immediate consistency is used as a baseline, resulting in the effect of using the eventual consistency becomes more apparent during visual evaluations. The vertical axis represents the difference between the consistency models, and the horizontal axis is the RUCT. The data in the graph has not been tampered with and the lines are not trend lines; the reason for the smoothness is due to the size of the simulation. Each consistency model has conducted 120’000 trial runs, given a well weighted average result.

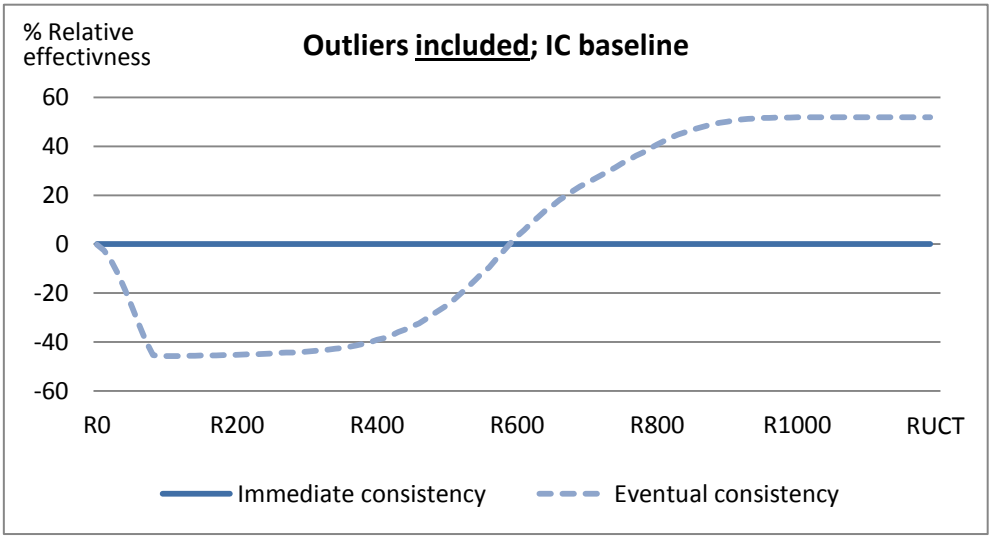


Graph 5.5. Effect of using eventual consistency compared to an immediate consistency baseline.

In this scenario, the breaking point occurs around RUCT 600, after which the eventual consistent system increasingly outperforms the immediate consistent system. Looking back at the hypothesis in Figure 3.1 (which was drawn prior to the construction of the simulation and has not been changed since) this graph is very similar in shape and outcome. The results show that a breaking point exists.

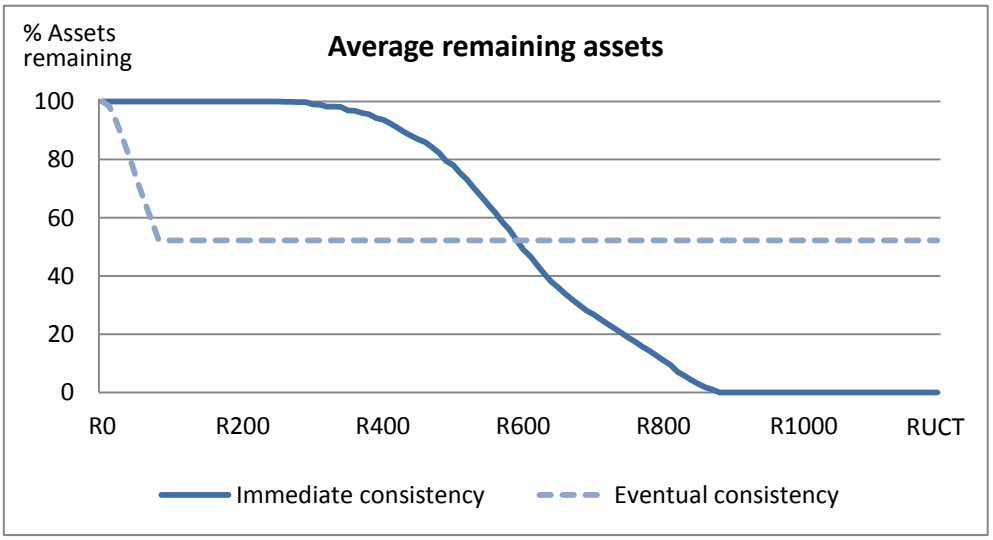
One clear distinction is however visible; the consistency models are equally effective given RUCT 0. After RUCT 0 the performance of the eventual consistency drops quickly for the duration between RUCT 0 and 100, before leveling and then slowly starting to rise again. The reason for this is that the eventual consistent system reaches consistency between every decision to fire missile (i.e., due to the short RUCT), but as the RUCT increase this becomes increasingly difficult and around RUCT 100 the system never reaches consistency, at which point all batteries act on tentative information.

In Graph 5.6, the actual output without removing the outliers is presented. Comparing it with Graph 5.5, they are both very similar in shape with the slightly noticeable difference that the eventual consistent system seems to perform better; around 3-5%. The difference is actually that without outliers removed, the immediate consistent system performs slightly worse (e.g., an extreme outlier, which should be removed, makes a rather large impact on the average in this graph). However, even with outliers included the hypothesis is valid.



Graph 5.6. Effect of using eventual consistency compared to an immediate, including outliers.

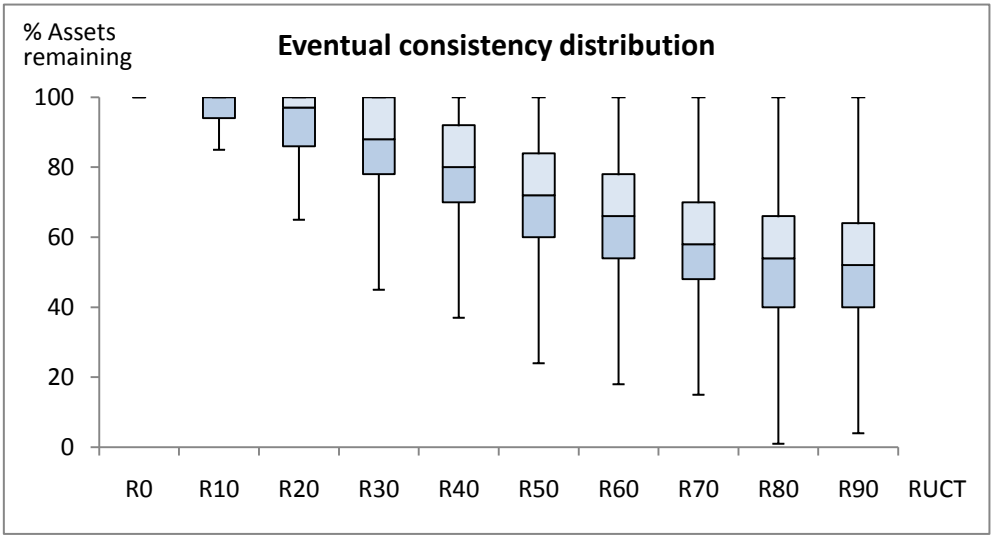
While the hypothesis used the immediate consistent system as a baseline, it is of interest to inspect the actual number of remaining resources. Graph 5.7 shows the amount of remaining resources in percentages. As the base scenario includes 50 defended assets, each asset represent 2%. The immediate consistent system destroys all aircrafts when the RUCT is low, leaving 100% of the assets intact, but as the RUCT increase the performance of the immediate consistent system decrease. When the RUCT is too long the immediate consistent system will never intercept even one single aircraft, resulting in 0% effectiveness, regardless of scenario configurations.



Graph 5.7. The actual number of remaining defended assets, in percentage.

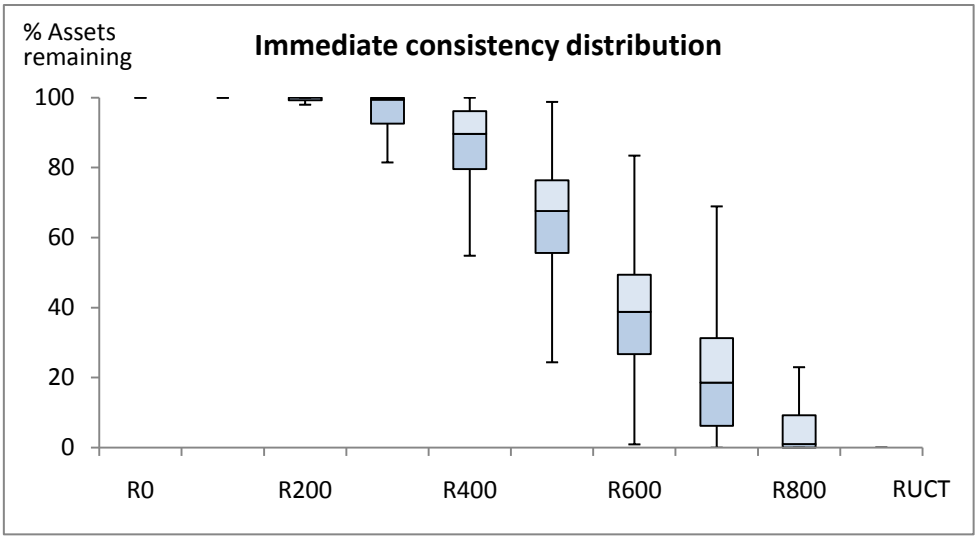
As of now, all graphs presented have used the average output of multiple trials. If only the average value validated the hypothesis, it would not count for much. It is thus interesting to look at the distribution of the data sets.

Starting with the output in the eventual consistency, Graph 5.8 shows box plots of each different RUCT used in the base scenario. At RUCT 0, the box plot is not visible. This is because of the way the data was prepared in section 5.4.2; the upper and lower boundaries of the quartiles are both 100, as more than 50% of the output destroys 100% of the aircrafts. After RUCT 0, the eventual consistency drops in effectiveness and the changes in distribution range up to RUCT 90 at which point increasing the RUCT further does not impact the scenario. Although the first and third quartile (i.e., 50% of the output) and the median are following the curve, the distribution is rather wide. This does not affect the hypothesis, as the immediate consistent system always drops to 0% effectiveness at some point.



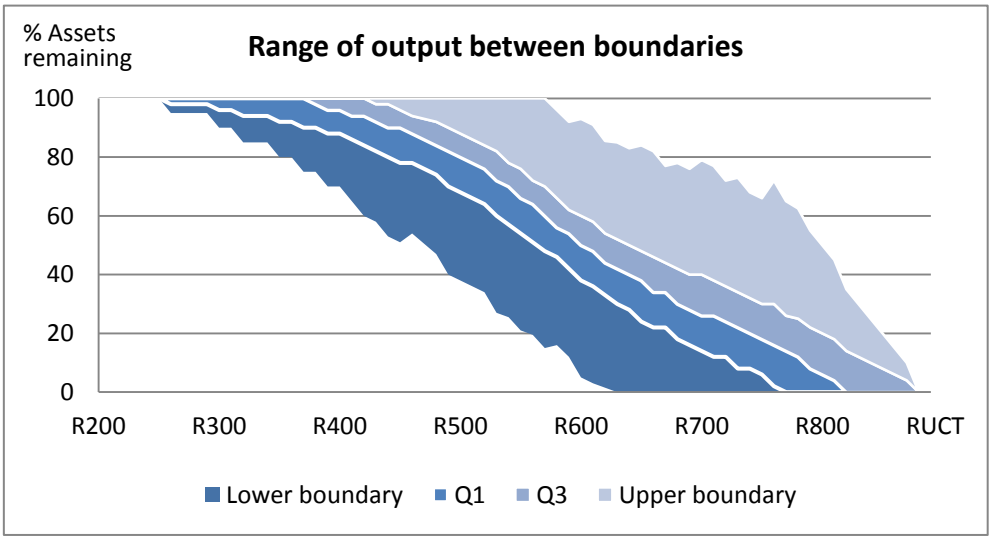
Graph 5.8. Box plot distribution of eventual consistency in the base scenario.

In Graph 5.9 the distribution of immediate consistency is also drawn using box plots. However, since the simulation runs at higher granularities than visually recommended to use box plots, the boxes are the average of 10 trial runs put together. At RUCT 0 and RUCT 100 the box plots are not drawn, due to the same reasons described above for Graph 5.8. In this scenario, at RUCT 900 the effectiveness of immediate consistency drops to 0%. The distribution is less wide than the eventual consistency. Using immediate consistency, missile defenses never waste resources which in turn make them more reliable.



Graph 5.9. Box plot distribution of immediate consistency in the base scenario.

Graph 5.10 represents the same data used for the box plots in Graph 5.9, but instead of clustering them in boxes the four boundaries of the quartiles of immediate consistency are drawn using a graphical area. The median is the line between Q1 and Q3. It should be noted that the lower and upper values of RUCT have been removed from this graph, as they do not produce any visible information about the distribution.

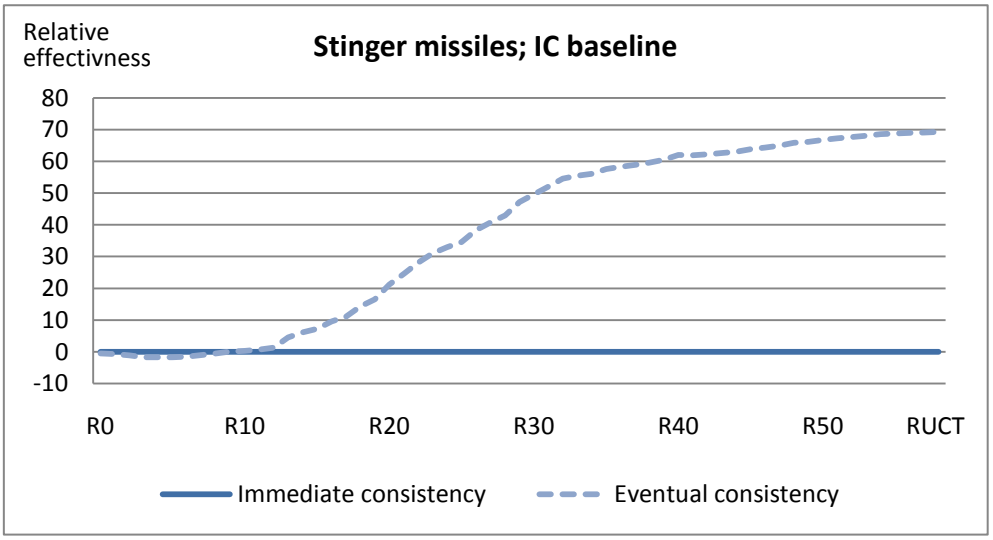


Graph 5.10. Area representing the distribution in immediate consistency.

What can be said about the distribution in general is that increasing the number of scenarios, using different seeds, will smooth the distribution curves. In the three distribution graphs above, the base scenario consisting of 1000 scenario is sufficient for showing the range of the output.

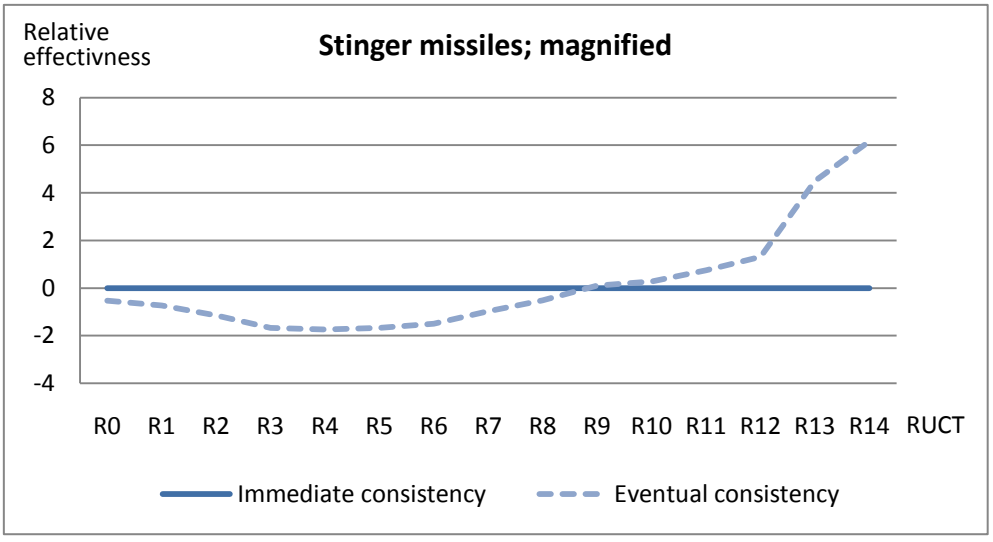
As the results of the base scenario and the distribution of the data have been presented, it is clear that the hypothesis is valid using this specific configuration. The breaking point can however be pushed both sideways, and up and down, by using different scenario configurations. Three specific situations are presented next; first using short range missiles, and then increasing and decreasing the number of aircrafts in the base scenario.

The FIM-92 Stinger missile presented in section 5.3.1 is a short range missile. Using the same aircrafts, but with a slightly modified scenario Graph 5.11 is showing the effect of using eventual consistency compared to the immediate consistency as a baseline.



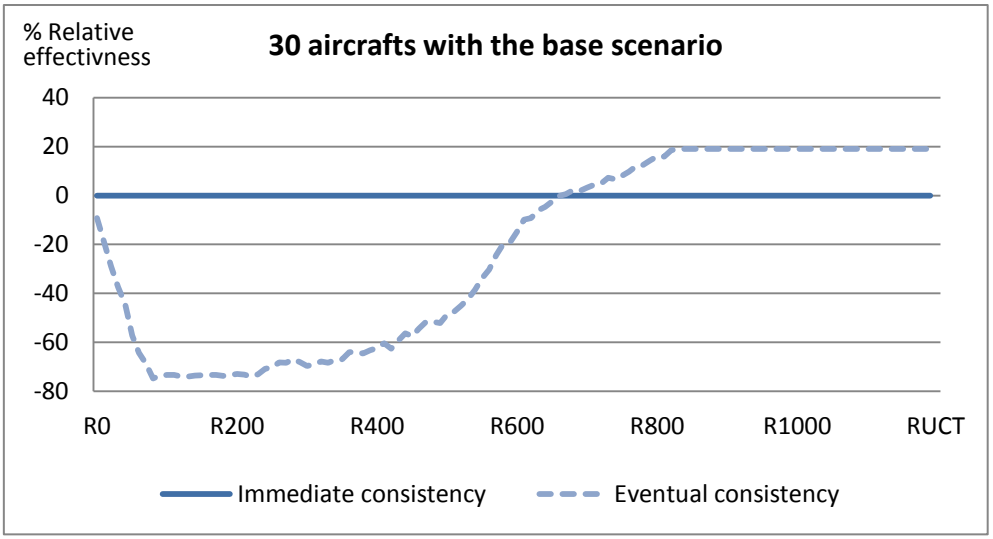
Graph 5.11. Eventual consistency effective with short range stinger missile.

In Graph 5.11 the actual breaking point is almost not visible, so Graph 5.12 shows the same results magnified on the first portion of the graph. Clearly visible, at RUCT 9 the effects of using eventual consistency outperform the immediate consistency. The reason that the effects emerge quickly is a result of the short range of the Stinger and the high speed of the aircraft.



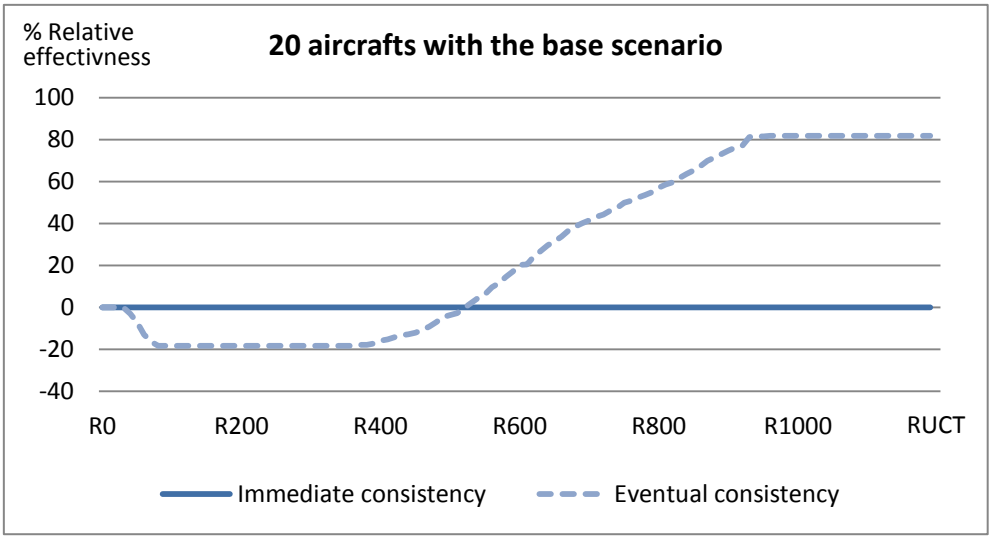
Graph 5.12. Magnification of Graph 5.11.

While the number of belligerent on the battlefield in the base scenario has been selected to produce a well weighted output, increasing or decreasing the amount of resources does not affect the hypothesis, as illustrated in Graph 5.13. In this scenario, as aircrafts carries multiple missiles, increasing the number of aircrafts rather quickly change the output as the remaining defended assets stand in direct proportion to the collected amount of ATGMs.



Graph 5.13. Using five more aircrafts than the regular base scenario.

Concurrently, decreasing the amount of aircrafts has an opposite effect, as illustrated in Graph 5.14. When comparing the two graphs it however is clear that they exhibit the exact same behavior, both showing a breaking point and after that point the eventual consistent system outperforms the immediate consistent system. While it does look like using fewer aircrafts benefits the eventual consistency model, it should be noted that more SAMs, in proportion to the amount of aircrafts, are being fired in the second scenario (which is not illustrated). Thus, it is not possible to make a general assumption about what consistency model benefits from which battlefield layout; just that that breaking point does emerge in both scenarios.



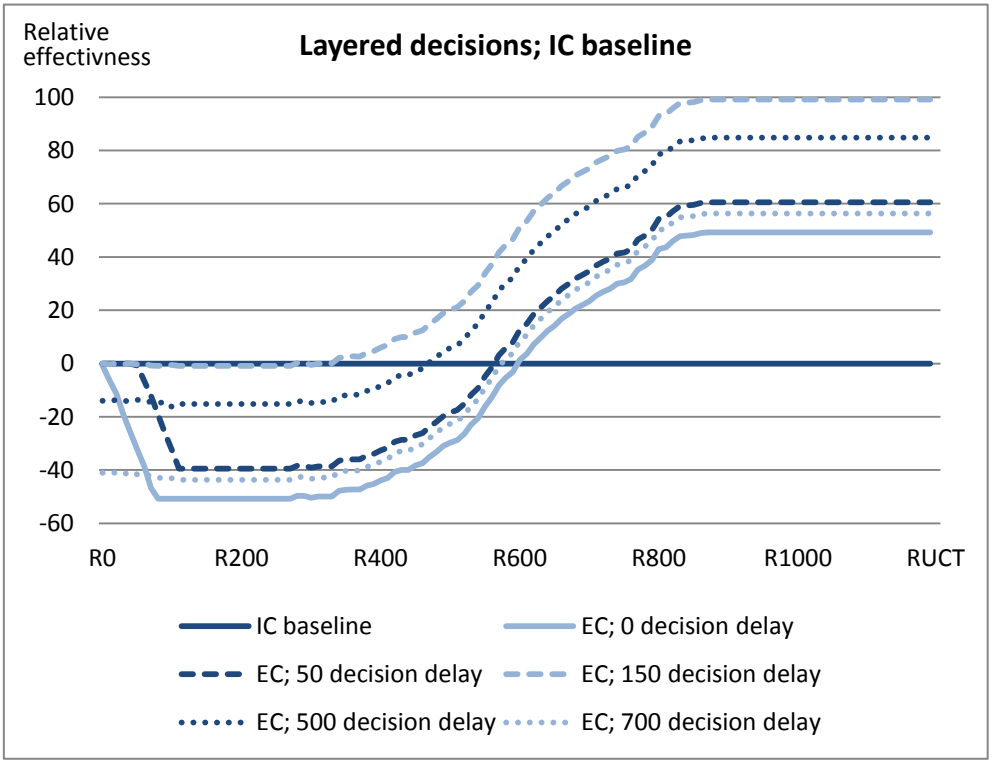
Graph 5.14. Using five less aircrafts than the regular base scenario.

It is possible to modify any other belligerent on the battlefield, to get another result. For instance, increasing the amount of missile defense systems will bring down more aircrafts increasing the benefits of using eventual consistency, and using longer ranges on missiles will push the breaking point up to higher RUCTs.

The graphs presented above show that the hypothesis is valid using different configurations. In Graph 5.13, should even more aircrafts be added (in this specific situation, around 40), no matter the consistency model, all defended assets will always be destroyed. This marks the exception where there is no breaking point, as both

models fail to defend the assets. In Graph 5.14, using even less aircrafts, there is a possibility that the eventual consistency always manages to bring down all aircrafts, thus being equally effective or more effective than the immediate consistency. However, running a large number of scenarios, using some random placement the immediate consistency will perform slightly better using low RUCTs, as the eventual consistency tend to over neutralize targets. This means that there is a breaking point in all situations. The exception would be if there are an infinite number of missiles, at which point using consistency models are unnecessary.

Presented next is the layered decision making, which has been described in detail in section 2.4.4. In section 5.4.1 it was decided that four different decision delay times should be used with the base scenario and these are presented in Graph 5.15 along with immediate consistency as a baseline and eventual consistency with no decision layers.

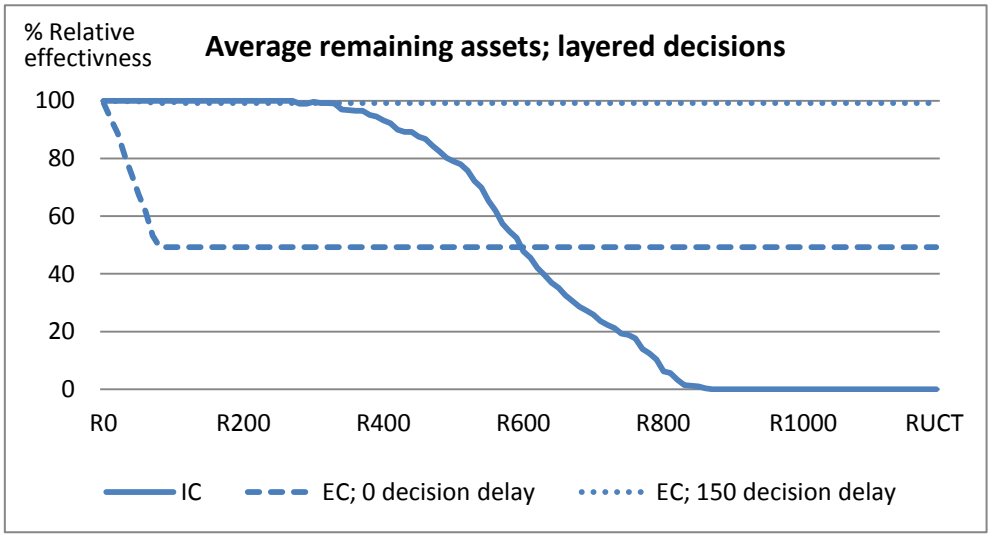


Graph 5.15. Different decision times in layered decision making.

As clearly seen, one configuration using a delayed decision performs exceptionally well compared to the other configurations. This is the 150 second delay; however, an actual time cannot be discussed in general terms as it is highly scenario dependent. Also noticeable is that the 50 seconds delay performs better than the standard eventual consistency without a secondary decision layer. This indicates that every delay value between 0 and 150 have a positive impact on the decision making process. When increased up to 500 in delay time, the layered decision still performs better than the eventual consistency with the exception if the RUCT is low. Further increasing the delay, at 700 the layered decision is almost equal compared to the eventual consistency without layers, and if further increasing the time delay it is not preferable to use layered decisions at all.

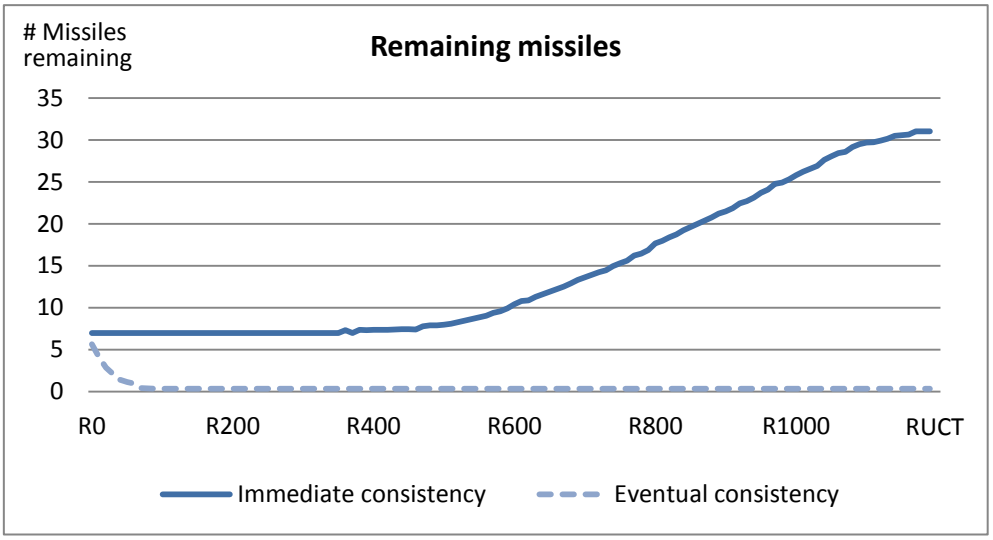
Illustrating how effective the optimal layered decision making really is, Graph 5.16 shows the remaining defended assets in percentages over an extended period of time. While the eventual consistency without any layered decision making outperforms the

immediate consistency as expected around RUCT 600, the 150 seconds layered decision performs at 99.2% without decreasing in effect. The immediate consistent system actually performs at 100% for a period of time when the RUCT is low, but then quickly drops in effectiveness.



Graph 5.16. Effectiveness of layered decision making.

Finally, the remaining number of missiles using the different consistency models is illustrated in Graph 5.17. In the base scenario there are a total of 32 SAMs, which is the amount of remaining missiles presented. As the immediate consistent system never waste resources; when its own performance decline it uses less and less missiles up to the point where no single missile is fired. The eventual consistency quickly uses up all missiles, thus over neutralizing multiple targets.



Graph 5.17. Remaining number of SAMs in the base scenario.

This concludes the presentation of the results. The results are discussed next.

5.5 Discussion

In this research I implemented a simulator which tested the effectiveness of using eventual consistency compared to immediate consistency. The hypotheses formed in the beginning of the research were shown to be correct.

Eventual consistency uses equally or more resources compared to the immediate consistent system. This is because of the eventual consistencies eagerness to act contrary to not acting. While wasting resources is unnecessary, it was shown on multiple occasions that wasting resources was preferable to not act at all.

The effects of the results were expected, with the exception of the very effective layered decision making, illustrated in Graph 5.15 and Graph 5.16. Improving and further testing application areas of the layered decision making would be interesting.

In this research multiple parts were simplified. Increasing the realism in the simulation would tell more about the location of the breaking point, but not change the fact that the hypothesis is valid in all situations (i.e., where consistency models still are useful). The most unknown variable in the research is military tactics and everything connected to the actual battle taken place in the simulation. It could be said that it is incorrectly implemented but still valid enough to verify the hypothesis. As an example, two aircrafts move over two batteries which are defending two assets. The batteries communicate with each other, and the best course of action is to fire at one aircraft each. If they are unable to form a decision, they can act on tentative information, which leaves a 50% chance of them intercepting the same aircraft. Now, if this decision takes one day to complete it is hard to argue against that it is better to fire on the same aircraft, than to not launch any missiles at all (i.e., waiting for a day). Thus, no matter the incorrectness in the battlefield deployment and tactics, the main research question and hypothesis is valid.

A relationship between battlefield belligerents is obvious when looking at different scenarios. The faster the aircrafts travel and the shorter the missiles reach, the more useful is eventual consistency. This can be formulized into one sentence:

> *"Short time frames increase the usefulness of eventual consistency."*

Using this sentence, a more specific definition of the usefulness of eventual consistency can be formed. It is difficult to speculate in which areas eventual consistency would be useful, but this denominator can apply to multiple situations:

> *"When two or more independent parts are communicating and sharing information essential for decisions, it is always preferable to reach a common consent about a course of action. If it is critical to act as failing to do so would worsen the situation, the circumstances are such that it is preferable to act on tentative information than not act at all."*

In all situations where this denominator applies, it could be useful to act on tentative decisions. An example is crisis management, where people in command usually act on tentative decisions which is better than to not act at all or linger with decisions (Lind Nilsson & Gustafsson, 2006, p. 50-51). Computer system formed to support human decision making could in those situations also make suggestions based on tentative decisions.

My conjecture is that eventual consistency can be useful given the right implementation and under appropriate circumstances. It seems to be important to fully know the benefits and even more so the negative aspects of eventual consistency before implementing and using it. In reality it might not be effective to use this model in a military setting if the breaking point occurs first after a long time span, but it was very advantageous during the process of finding and showing that there is a breaking point after which the eventual consistent system outperforms the immediate consistent system.

The most interesting part not fully examined is the eventual consistent system with a layered decision making, as it benefits from the immediate consistent system but does not share its weaknesses. In Graph 5.16 overwhelmingly positive results were produced. By using this model, the eventual consistency using layered decision making also continuously produced perfect results. Implementing a more rigorous layered decision making is the most interesting future aspect arrived from this work.

5.6 Related work

The research on eventual consistency is sparse. Birrell et al. (1982) implemented Grapevine, a multicomputer system which ensured that replicas in different parts of the systems eventually shared the same information. Gustavsson and Andler (2002) argue for the benefits of using an eventually consistent protocol in replicated databases which sets a high demand on availability. Syberfeldt (2007) has implemented eventual consistency in his replication protocol PRiDe, and it allows for temporal divergences of replicas as long as they eventually converge.

In order to complete this research, a ground based air defense scenario using threat evaluation and weapon allocation has been used. This has been extensively researched on by Roux and van Vuuren (2007), Johansson and Falkman (2008, 2009a, 2009b), Paradis et al. (2005) and Naeem et al. (2009), and is well researched topic in the military, although usually classified as pointed out by Roux and van Vuuren (2007).

The use of layered decision making in eventual consistency derives from the ideas of Pignaton de Freitas et al. (2009), but is implemented in a slightly other way in this research as no missile base can have the ability to fire without declaring its intent to do so. This eliminates the risk of multiple bases avoiding firing upon a target, but it also increases the risk of over neutralization. When weighing this against each other, it is better to fire two missiles than not fire any, explaining the changes (i.e., improvements considering this scenario) to the layered decision making.

To the author’s knowledge, no research on the actual effect of using eventual consistency has been conducted.

6 Conclusion

This final chapter summarizes the work in section 6.1, outlines the two major contributions in section 6.2 and described future work in section 6.3.

6.1 Summary

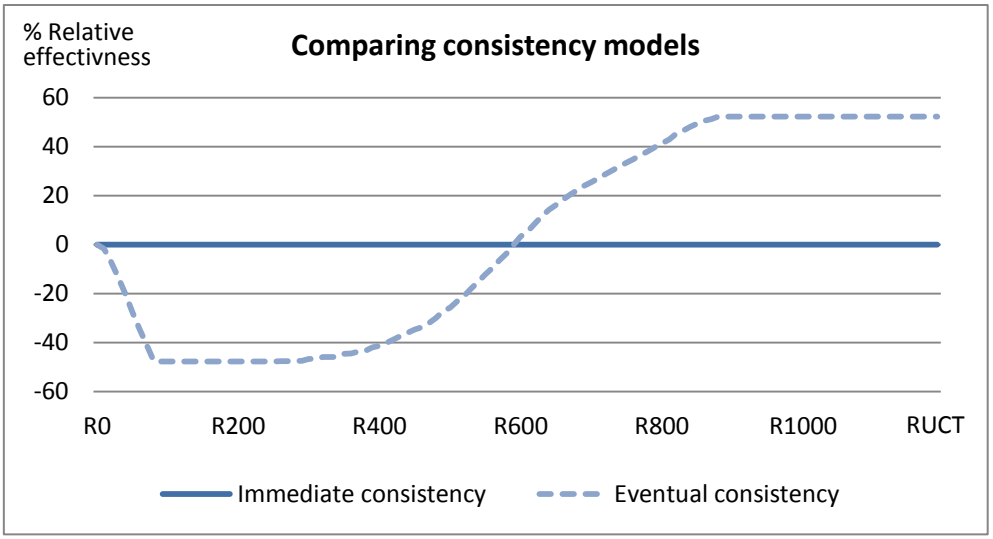
Stated in the purpose in section 3.1, the goal with this research was to evaluate the effects of replica update completion time in an eventually consistent system. The hypothesis in section 3.2 states that the positive effects of using eventual consistency should emerge when the RUCT is increased. Compared with the use of an immediate consistent system as a baseline, the actual effect was evaluated.

Multiple nodes connected to each other capable of replicating data and forming decisions were required to verify the hypothesis. Since the positive effects of eventual consistency was thought to become apparent during situations pressured by tight time constraints, the use of a ground based air defense was selected as the intended platform for data extraction. Aircrafts traveling over missile defense systems on their way to a defended asset pressure the missile defense system to form decisions regarding proper countermeasures. Each defense system acts as an autonomous unit, which is capable of forming decisions and replicate them to other parts of the networked defense. The actual evaluation of the consistency models was measured by the amount of saved defended assets, depending on the defense effectiveness in forming decisions during the short window the aircrafts passes over.

In the method it was decided that the most tractable solution was to use simulations when constructing the ground based air defense scenario. The choice fell on implementing a simulator from the bottom and up, capable of running specifically constructed scenarios. The simulator was simplified as much as possible, while still retaining the realistic features such as velocities of aircrafts, which is an attribute deriving from the real world.

The constructed scenarios are also simplifications of real world military strategy. The only important aspect is that aircrafts travel over missile defenses, and if the missile defenses fail to act, or act inappropriately, the defended assets are lost to the aircrafts missiles. The simulator executes multiple scenarios where data is extracted and then used for evaluations.

The results presented in section 5.4.3 verify the hypothesis. When using eventual consistency, as RUCT increase it will outperform a system using immediate consistency under the same circumstances. Graph 6.1 is from the presentation of the research, and shows a breaking point where the eventual consistency continuously outperforms the immediate consistency.



Graph 6.1. Effect of using eventual consistency compared to an immediate consistency baseline.

In all situations where consistency models are useful, the eventual consistent system will outperform the immediate consistent system when the RUCT increase above a certain point. The exact RUCT where one consistency model is preferably over another is scenario dependent; the fact that there will be a breaking point is true regardless of scenario and context. The eventual consistency model is especially useful in stressful situations where a time constraint makes it difficult to form a decision all parts can agree upon.

Of secondary importance to the research, an improvement to the eventual consistency using multiple decision layers was implemented. Missile defenses form more than one decision, and this increases their performance noticeably.

6.2 Contributions

The primary contribution from this work is the verification that there exists a breaking point where the eventually consistent system outperforms the corresponding immediate consistent system.

Layered decision making has been shown to be very useful in most situations given the right time to delay decisions. Using eventual consistency with multiple layers of decision has been shown to closely resemble the effects of using immediate consistency when the RUCT is low, while at the same time avoiding the bottleneck of locking which is conducted by the immediate consistency.

6.3 Future work

This evaluation of eventual consistency leaves much for the future, as it has been shown that the effects actually are beneficial compared to immediate consistency during specific circumstances.

- *Other application areas for eventual consistency*. The use of a ground based air defense scenario was selected because it was easy to verify the hypothesis in that setting. However, the use of eventual consistency might be useful in other areas. Crisis management and traffic control both need to form decisions during a short amount of time, making eventual consistency useful.
- *Extensive research on layered decision making*. An interesting aspect in this research is the layered decision making in combination with the eventual

consistency. Conducting more research in this area, finding more precise delay times and using more than two layers of decision would be of interest.

- *Using real statistics*. Real statistics from air defense scenarios is required in order to make assumptions if the shown breaking point resides within the frame where it would have an impact in a real world scenario. This means that without these statistics no conclusions can be drawn as to where the effectiveness of eventual consistency outperforms the immediate consistent system.
- *Improving the simulator*. While sufficiently advanced for verifying the hypothesis, the simulator is simplified in multiple ways. Improvements to the simulator, especially in the field of TEWA and message propagation, are required in order to make more precise statements as where the breaking point is, and which factors that have the greatest impact on the result. While the simulator does not have to perfectly simulate the real world, it needs to incorporate the factors which affect the outcome, such as missile kill probabilities, and then combine these with real statistics.

References

Ahuja, R.K. et al., 2007. Exact and heuristic methods for the weapon target assignment problem. *Operations research*, 55(6), pp.1136-1146.

Andler, S.F., Eriksson, J. & Lundin, M., 2000. Distribution of control using active storage. *Control Engineering Practice*, 8(6), pp.681–687.

Atkins, M.S. & Coady, M.Y., 1992. Adaptable concurrency control for atomic data types. *ACM Transactions on Computer Systems (TOCS)*, 10(3), pp.190-225.

Barker, E. et al., 2007. NIST Special Publication 800-57. *NIST Special Publication*, 800(57), pp.1-142.

Bernstein, P., Hadzilacos, V. & Goodman, N., 1987. *Concurrency Control and Recovery in Database Systems*, Addison-Wesley Publishing Company.

Birrell, A.D. et al., 1982. Grapevine: An exercise in distributed computing. *Communications of the ACM*, 25(4), pp.260-274.

Boeing Defense, Space and Security, 2010. Boeing F/A-18E/F Super Hornet Block II. Available at: http://www.boeing.com/defense-space/military/fa18ef/docs/EF_overview.pdf [Accessed August 31, 2010].

Bowley, D. & Lovaszy, S., 1999. Use of Combat Simulations and Wargames in Analytical Studies. In *Proceedings of SimTecT 99*.

Brännström, M. et al., 2004. Distributed data fusion in a ground sensor network. In *Proc 7th International Conference on Information Fusion*.

Cheung, S.C. & Kramer, J., 1994. Tractable dataflow analysis for distributed systems. *IEEE Transactions on Software Engineering*, 20(8), pp.579–593.

Coulouris, G., Dollimore, J. & Kindberg, T., 2005. *Distributed systems* 4th ed., Pearson Education Limited.

Cumpsty, N., 2003. *Jet Propulsion: A Simple Guide to the Aerodynamic and Thermodynamic Design and Performance of Jet Engines* 2nd ed., Cambridge: Cambridge University Press.

Datta, A.H. & Son, S.H., 2002. A study of concurrency control in real-time, active database systems. *IEEE Transactions on Knowledge and Data Engineering*, pp.465–484.

Domenici, A. et al., 2004. Replica consistency in a Data Grid. *Nuclear Inst. and Methods in Physics Research, A*, 534(1-2), pp.24–28.

Durrant-Whyte, H., 2000. A Beginner's Guide to Decentralised Data Fusion. *Technical Document of Australian Centre for Field Robotics, University of Sydney, Australia*, pp.1-27.

Eurofighter Typhoon, 2010. Eurofighter: Technical Data. Available at: http://www.eurofighter.com/eurofighter-typhoon/technicaldata.html [Accessed August 31, 2010].

Flack, R.D., 2005. *Fundamentals of Jet Propulsion with Applications*, New York: Cambridge University Press.

Global Security, 2005. FIM-92A Stinger Weapons System: RMP & Basic. Available at: http://www.globalsecurity.org/military/systems/ground/stinger-specs.htm [Accessed October 17, 2010].

Global Security, 2008a. Patriot TMD. Available at: http://www.globalsecurity.org/space/systems/patriot-specs.htm [Accessed October 17, 2010].

Global Security, 2008b. V-75 SA-2 Guideline. Available at: http://www.globalsecurity.org/military/world/russia/v-75-specs.htm [Accessed October 17, 2010].

Gong, Z. & Aldeen, M., 1997. Stabilization of Decentralized Control Systems. *Journal of Mathematical Systems, Estimation, and Control*, 7(1), pp.1-16.

Grime, S. & Durrant-Whyte, H.F., 1994. Data fusion in decentralized sensor networks. *Control engineering practice*, 2(5), pp.849–863.

Gustavsson, S. & Andler, S.F., 2002. Self-stabilization and eventual consistency in replicated real-time databases. In *Proceedings of the first workshop on Self-healing systems*. ACM Press. pp. 105–107.

Haerder, T. & Reuter, A., 1983. Principles of Transaction-Oriented Database Recovery. *Computing Surveys*, 15(4), pp.287-317.

Hall, D.L. & Llinas, J., 1997. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1), pp.6–23.

Hawkins, D.M., 1980. *Identification of Outliers*, London: Chapman and Hall.

Herlihy, M., 1990. Apologizing versus asking permission: Optimistic concurrency control for abstract data types. *ACM Transactions on Database Systems (TODS)*, 15(1), pp.96–124.

Horvitz, E.J., 1990. *Computation and action under bounded resources*. PhD Dissertation. Standford University.

International Standards Office, 2006. *ISO/IEC 23270:2006 - Information technology - Programming languages - C#*, Geneva: ISO.

Johansson, F. & Falkman, G., 2008. A Bayesian network approach to threat evaluation with application to an air defense scenario. In *Proceedings of the 11th International Conference on Information Fusion*. pp. 1352-1358.

Johansson, F. & Falkman, G., 2009a. An empirical investigation of the static weapon-target allocation problem. In *Proceedings of the 3rd Skövde Workshop on Information Fusion Topics (SWIFT2009)*.

Johansson, F. & Falkman, G., 2009b. A testbed based on survivability for comparing threat evaluation algorithms. In *Proceedings of the SPIE, Symposium on Defense, Security and Sensing*.

Kee, Y.S. et al., 2005. Efficient resource description and high quality selection for virtual grids. In *Proceedings of the Fifth IEEE Symposium on Cluster Computing and the Grid (CCGrid)*.

Kee, Y.S. et al., 2006. Robust Resource Allocation for Large-scale Distributed Shared Resource Environments. In *High Performance Distributed Computing, 2006 15th IEEE International Symposium on*. pp. 341–342.

Kent, C.A., 1987. *Cache Coherence in Distributed Systems*, DEC Western Research Lab.

Kopetz, H., 1991. Event-triggered versus time-triggered real-time systems. *Operating Systems of the 90s and Beyond*, pp.86–101.

Kopp, C., 2010. Almaz S-300 - China's "Offensive" Air Defense. Available at: http://www.strategycenter.net/research/pubID.93/pub_detail.asp [Accessed October 17, 2010].

Kung, H.T. & Robinson, J.T., 1981. On optimistic methods for concurrency control. *ACM Transactions on Database Systems (TODS)*, 6(2), pp.213–226.

Lin, K. & Peng, C., 1996. Enhancing external consistency in real-time transactions. *ACM SIGMOD Record*, 25(1), pp.26-28.

Lind Nilsson, I. & Gustafsson, L., 2006. *Ledarskapets inre och yttre resa*, Lund: Studentlitteratur.

Marsaglia, G. & Tsang, W.W., 2002. Some difficult-to-pass tests of randomness. *Journal of Statistical Software*, 7(3), pp.1–9.

MSDN Library, 2010. Random Class. Available at: http://msdn.microsoft.com/en-us/library/system.random.aspx [Accessed September 8, 2010].

Naeem, H. et al., 2009. A Novel Two-Staged Decision Support based Threat Evaluation and Weapon Assignment Algorithm. *International Journal of Computer Science and Information Security*, 2(1), pp.1-7.

Nguyen, X.T., 2002. Threat Assessment in Tactical Airborne Environments. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*. pp. 1300-1307.

Pacitti, E., Minet, P. & Simon, E., 1999. Fast Algorithms for Maintaing Replica Consistency in Lazy Master Replicated Databases. *Institut national de recherche en informatique et en automatique*.

Paradis, S. et al., 2005. Threat Evaluation and Weapons Allocation in Network-Centric Warfare. In *Information Fusion, 2005 8th International Conference on*. pp. 1078-1085.

Perros, H.G., 2003. *Computer simulation techniques: the definitive introduction*, Computer Science Department, North Carolina State University, Raleigh, North Carolina, USA.

Pignaton de Freitas, E. et al., 2009. Multi-Agent Support in a Middleware for Mission-Driven Heterogeneous Sensor Networks. *The Computer Journal*.

Ramamritham, K., 1993a. Real-time databases. *Distributed and Parallel Databases*, 1(2), pp.199-226.

Ramamritham, K., 1993b. Time for Real-Time Temporal Databases? In Proceeding of the International Workshop on an Infrastructure for Temporal Databases.

Roux, J.N. & van Vuuren, J.H., 2007. Threat evaluation and weapon assignment decision support: A review of the state of the art. *Journal of the Operations Research Society of South Africa*, 23(2), pp.151–187.

Saab AB, 2009. Gripen Supercruises. Available at: http://www.gripen.com/en/MediaRelations/News/2009/090121_gripen_supercruises.htm [Accessed August 31, 2010].

Saab AB, 2010. Gripen Technical Summary. Available at: http://www.gripen.com/en/GripenFighter/TechnicalSummary.htm [Accessed August 31, 2010].

Sargent, R.G., 2008. Verification and validation of simulation models. In *Proceedings of the 40th conference on Winter simulation*. pp. 157-169.

Scheler, F. & Schröder-Preikschat, W., 2009. Time-Triggered vs. Event-Triggered: A matter of configuration. In *Proceedings of the GI/ITG Workshop on Non-Functional Properties of Embedded Systems*. pp. 107–112.

Sheth, A., Leu, Y. & Elmagarmid, A., 1991. Maintaining Consistency of Interdependent Data in Multidatabse Systems. *Technical Report CSD-TR-91-016, Computer Science Department, Purdue University*.

Sun, C. & Chen, D., 2002. Consistency Maintenance in Real-Time Collaborative Graphics Editing Systems. *ACM Transactions on Computer-Human Interaction*, 9(1).

Syberfeldt, S., 2007. *Optimistic Replication with Forward Conflict Resolution in Distributed Real-Time Databases*. Linköping University.

Taurus Systems GmbH, 2010. KEPD350. Available at: http://www.taurus-systems.de/html/kepd350.html [Accessed August 31, 2010].

The US Navy, 2009a. AGM-154 Joint Standoff Weapon (JSOW). Available at: http://www.navy.mil/navydata/fact_display.asp?cid=2100&tid=300&ct=2 [Accessed August 31, 2010].

The US Navy, 2009b. AGM-65 Maverick Guided Missile. Available at: http://www.navy.mil/navydata/fact_display.asp?cid=2200&tid=500&ct=2 [Accessed August 31, 2010].

The US Navy, 2009c. AGM-88 HARM Missile. Available at: http://www.navy.mil/navydata/fact_display.asp?cid=2200&tid=300&ct=2 [Accessed August 31, 2010].

Thomasian, A., 1998. Concurrency control: methods, performance, and analysis. *ACM Computing Surveys (CSUR)*, 30(1), pp.70–119.

U.S. Air Force, 2009. F-22 Raptor. Available at: http://www.af.mil/information/factsheets/factsheet.asp?id=199 [Accessed August 31, 2010].

Zanette, D.H., 2002. Dynamics of rumor propagation on small-world networks. *Physical review E*, 65(4), p.41908.

Appendix

I. Code to the simulator

The simulator is programmed in C# using Microsoft XNA®, as stated on several occasions. In order to compile and execute the simulator only the code would simply not be sufficient, as project settings, naming of files and lack of images would hinder the process. In order for others to validate the results presented in the thesis, as stated in 4.2.1, the code, the pre-compiled simulator as well as the scenarios is available upon request from the author.

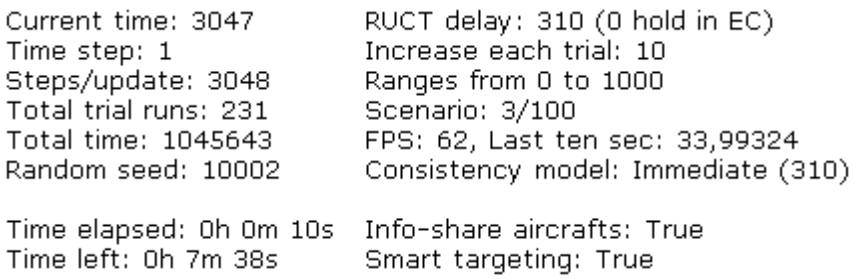
Contact information:
Mac Svan
E-mail: mac.svan@gmail.com

II. Images and illustrations used in the thesis

All images and illustrations used in the thesis have been created by the author for the use in this thesis, and if they are based on the works of others this is mentioned in the text. The only exception is the map used in the simulator; while it is partially modified, the original is derived from a free map released to the public domain (without any restrictions) by its author on Wikipedia.

III. Statistics from the simulator described in detail

As mentioned in section 5.1.2, the statistics from the simulator is described in detail with the use of the illustration from the simulator below.



- *Current time.* The virtual time in seconds since the start of the specific configuration in the current trial run. Entities on the battlefield (i.e. aircrafts, missiles) move according to their velocity when the time increases.
- *Time step.* Each update-loop advances the current time with this amount. Decreasing this number increase accuracy but decrease performance, and vice versa. At some point decreasing this number further does not affect accuracy but still decrease performance; this exact point depends on the current scenario.
- *Steps/update.* Represents the number of time step-updates during each loop. Increasing this number, a variable adjustable during simulations, increase the speed the simulation progress in. It does not affect the output. Adjusting the steps/update is used to maximize performance on the computer running the simulation.
- *Total trial runs.* Number of total trials tested since the start of the simulation.
- *Total time.* Total virtual time processed during this simulation.

- *Random seed.* The current random seed value. It is possible to set the starting seed. After that, the seed increase with 1 between each trial run.
- *RUCT delay.* The current RUCT delay. The parenthesis includes the delay used in layered decision making in eventual consistency.
- *Increase each trial.* How much the RUCT delay increases each configuration step in the trial.
- *Ranges from X to Y.* The range the RUCT delay is used; X is usually set to 0 whereas Y is set to a value where it no longer affects the output.
- *Scenario.* Current scenario in the simulation. Each scenario use a unique seed (in the current simulation), which is used to randomly place battlefield resources.
- *FPS.* The current frames per second, as well as an average during the last ten seconds. This value is useful when maximizing the steps/update in accordance to the computers performance. Low FPS results in a scenario difficult to visually validate.
- *Consistency model.* Displaying which consistency model that is currently in use.
- *Time elapsed.* Real time elapsed, in hours, minutes and seconds, since the start of the simulation. Helps keeping track on how long a specific scenario takes to complete.
- *Time left.* Calculated time left until the scenario completes. This is based on the current progress in the scenario combined with the time elapsed. Very accurate, and is useful during scenario construction (e.g., a scenario can quickly and exponentially increase the run time with slight changes to the primary configuration, without adding any actual benefits to the output).
- *Info-share aircrafts.* Can be set to true or false. When *true*, aircrafts share basic information between each other. This makes them attack different defended assets to some extent.
- *Smart targeting.* Can be set to true or false. When *true*, aircraft attack the closest defended asset while also taking into account the info-share between aircrafts. When set to *false*, aircraft randomly attack a assets on the battlefield.

IV. One complete time step in the simulator

As stated in 5.2.11, one complete run in the simulator, step by step, is outlined in this section.

The positions of the belligerents are randomly selected on the battlefield. Each time step the code is sequentially executed following the basic rules of the battlefield. This sequence is detailed below.

1. The virtual clock increase with the amount of one time step.
2. The aircraft and missiles move according to their velocities resulting in a new position.
3. If there is an airborne SAM, the battlefield checks the position of the SAM and the aircraft.
    a. If the missile is on top of the aircraft, both the aircraft and the missile are destroyed.

    b. If there are no more aircrafts the battlefield resets all values and randomly spawns new locations for the belligerents.

4. If there is an airborne ATGM, the battlefield checks the position of the ATGM and the defended asset.
    a. If the missile is on top of the defended asset, both the defended asset and the missile are destroyed.
    b. If there are no more defended assets the battlefield resets all values and randomly spawns new locations for the belligerents.
5. The battlefield informs the missile defense system of the position of the aircraft.
    a. If the aircraft is within the missile defense system's firing range, it checks to ensure it has remaining missiles and then launch a SAM. The missile will act on its own during the next time step.
6. The battlefield informs the aircraft of the position of the defended asset.
    a. If the aircraft is within firing range, it checks to ensure that it has a remaining ATGM and then fire its missile. The missile will act on its own during the next time step.
7. If the aircraft is moving outside the borders of the battlefield, it is marked as "left battlefield". It is not further updated.
8. If all aircrafts have left the battlefield or has been destroyed, the scenario resets. In all other situations the scenario is still active and this process is sequentially executed again from step 1.

V. The base scenario

The base scenario is used during all simulation runs, and is also partly modified to meet specific demands. When changes are made to the base scenario, it is explicitly stated in the text. These are the primary settings used in the simulator.

10000	Starting random seed
25	Number of aircrafts
300,0	Aircraft speed m/s
100000,0	ATGM range in meters
340,0	ATGM speed m/s
1	Minimum number of missiles
8	Maximum number of missiles
16	Number of batteries
160000,0	SAM range in meters
1700,0	SAM speed m/s
2	SAMs/battery
50	Number of assets
TRUE	Smart aircraft targeting
TRUE	Information share between aircrafts
0,0	Layered decision making, time to hold decision
0,0	Starting RUCT
1200,0	End RUCT
10,0	RUCT increase
5,0	Time step in seconds
1000	Number of scenarios
3	Number of spawn areas: Aircrafts
70000,0	Position X in meters
350000,0	Position Y in meters
120000,0	Circle radius
-80000,0	Position X in meters
400000,0	Position Y in meters
100000,0	Circle radius
-200000,0	Position X in meters
410000,0	Position Y in meters
90000,0	Circle radius
3	Number of spawn areas: Batteries
40000,0	Position X in meters
-120000,0	Position Y in meters
100000,0	Circle radius
-90000,0	Position X in meters
-165000,0	Position Y in meters
60000,0	Circle radius
-200000,0	Position X in meters
-125000,0	Position Y in meters
100000,0	Circle radius
2	Number of spawn areas: Assets
-160000,0	Position X in meters
-350000,0	Position Y in meters
110000,0	Circle radius
-20000,0	Position X in meters
-380000,0	Position Y in meters
80000,0	Circle radius