Directional Networking in GPS Denied Environments—Time Synchronization

Derya Cansever and Gilbert Green
Army CERDEC
Aberdeen Proving Ground MA, USA

Jun Sun, Carl Fossa, Laura Herrera, and Devin Kelly
MIT Lincoln Laboratory
Lexington MA, USA

Abstract—**Mobile networks using directional antennas have many desirable properties such as Low Probability of Detection (LPD) and concentration of electromagnetic energy, resulting in improved range and data rates. To function properly, directional networks heavily depend on GPS use. This paper describes a decentralized algorithm, called "Fast RTSR", for directional networking to re-construct time and position data when GPS is not available. We show that the Fast RTSR algorithm allows the entire network to achieve time synchronization with convergence time of O(n), where n is the number of nodes in the network. In our simulation experiments, the Fast RTSR algorithm can improve the convergence time by as much as 400 times over the previous baseline RTSR algorithm [1]. Our preliminary results also show that a constant timestamp error and node mobility will not impact the accuracy of the Fast RTSR algorithm.**

Keywords—GPS; directional networking; wireless; algorithms; tactical networks

I. INTRODUCTION

Mobile networks using directional antennas have many desirable properties such as Low Probability of Detection (LPD) and concentration of electromagnetic energy, resulting in larger reception ranges and higher data rates. Also, depending on the antenna beam width, directional networking has the potential to significantly enhance spatial re-use of the frequency spectrum. Given the scarcity of frequency spectrum, this could be a very appealing property. These advantages come with the challenge of keeping track of other nodes' locations in the network and, for Time Division Multiplexing (TDMA) networks, for common precise timing among participating nodes. Requirements for precise time and location information become even more acute during the initialization phase, where nodes need to discover and connect with their peers to form a network. Note that these challenges, especially those related to location determination, would be less severe in networks driven by omni-directional antennas. Many Directional Mobile ad hoc Networks (D-MANETS) tend to rely on Global Positioning System (GPS) or its variants to keep track of time and position of the nodes in the network. The receive power for GPS signals is approximately

-160 dBW). This could make the received GPS signals vulnerable to malicious or even unintentional interference. This paper is concerned with methods for network nodes to re-construct each other's local clock times and converge upon a common time with the high degree of precision needed in D-MANET technologies. Toward the goal of identifying the best methodology to reconstruct time information in the absence of reliable GPS signals, we considered the following commonly used techniques:

- RF-Based Solutions: Using RF-based measurements to synchronize time and measure node range.

- Satellite Doppler: Using Doppler measurements from multiple satellites along with satellite catalog data to determine time and position.

- LTE: Use existing LTE base-stations for time and position.

- Differential GPS: A receiver with a known location broadcasts a correction signal to GPS receivers.

- GPS Relay (UAS): A set of unmanned aerial systems (UAS) that transmit (or retransmit) the GPS signal at a higher SNR than the original signal.

- Startracker: Using the precise location of one or more stars.

- Opportunistic Signals: Opportunistically take advantage of existing RF signals (i.e., FM radio, DTV, LTE, etc.) transmitted from known locations.

- Localization with Anchors: Determine the location (either absolute or relative) of at least three nodes, and then use these known locations to localize the rest of the network.

However, based on the analysis in [1], none of the aforementioned solutions completely solves the time synchronization problem. [1] developed a new approach which we refer to as the Baseline RF Time Synchronization and Ranging (RTSR) algorithm. The basic concept behind

baseline RTSR is to use a low data rate signal to transmit time messages between adjacent nodes. Nodes transmit packets with time information that allows neighbors to compute RTT. This in turn allows nodes to calculate the time difference between themselves and their neighbors and adjust their clocks towards the average clock value of its neighbors.

We propose a Fast RTSR algorithm with a much faster convergence speed than the baseline RTSR algorithm. We show that the convergence time of the Fast RTSR algorithm is O(n), where n is the number of nodes in the network. In our simulation experiments, the time required for a network to reach a certain time synchronization threshold when using Fast RTSR is only 0.25% of the time required when using baseline RTSR (an improvement by a factor of 400)Fast RTSR is optimized in the following three areas: 1) The fraction of nodes that are transmitting at any time; 2) The set of antenna sectors on which a node will transmit or receive; 3) The algorithm used by a node to update its clock based on its neighbors' time information.

When using Fast RTSR, each node needs to decide whether it will transmit or receive for a given period of time, since it cannot do both with the current antenna technology. If too many nodes decide to transmit at the same time, most transmissions will fail since only a small number of nodes are listening. Here, we derive the optimal probability of a node being in transmit mode.

During the first phase of the Fast RTSR algorithm, nodes will obtain information about their neighbors. Once a node knows the existence of some of its neighbors, it can choose to transmit or receive only on the antenna sectors where a neighbor currently exists. Compared with baseline RTSR, this approach reduces wasted transmissions on antenna sectors that do not point toward other nodes.

Lastly, using Fast RTSR nodes will propagate to each other a *randomly* picked time (out of the set of times received from neighbors) instead of attempting to converge to the average clock value of the entire network as with baseline RTSR.

When applying the RTSR algorithm to a real communication system, there are a number of additional issues which must be considered, such as timestamp error and node mobility. Both Fast RTSR and baseline RTSR require accurate estimation of RTT, which in turn requires an accurate timestamp of a packet that reflects the moment that packet transmission begins over the channel. However, due to the random processing delay at the operating system and the transmission chain, this timestamp may not be accurate. A C++ discrete event simulation was written to examine the impact of the random processing delay on the performance of the RTSR algorithm. We show that a constant random processing delay will *not* impact the performance of RTSR. A non-deterministic random processing delay, however, can impair the RTSR algorithm such that the time may not converge to the desired accuracy. We also demonstrate that node mobility, assuming

node speed is less than one thousand miles per hour, will not impact the performance of RTSR algorithm.

The rest of this paper is organized as follows: Section II gives a detailed description of our directional antenna model. In Section III, an overview of the baseline RTSR algorithm is presented first, followed by a description of the Fast RTSR algorithm and its performance results. Section IV discusses the impact of timestamp error and the impact of mobility on the RTSR algorithm. Section V concludes the paper.

II. PRELIMINARIES

In this paper, we consider a tactical network where each node has a directional antenna. We adopt the antenna model in [2]. The model assumes an antenna with S sectors. Each sector has a conical radiation pattern, covering an angle of 2π/S. The sectors are fixed with non-overlapping beam directions, hence collectively covering the entire circular region surrounds the antenna (shown in Figure 1). We also assume there is no omni-directional mode of transmission. Before transmitting or receiving data, the antenna must pick a sector to transmit or receive the data.

To successfully decode a packet, the receiving node must be located within a radius r of the transmitting node. We say node A lies in sector k of node B if node A is within an angle $[2\pi(k-1)/S, 2\pi k/S)$ of node B and the distance between A and B is less than r. As in any wireless communication system, interference will occur when the signals of two or more transmitters are colliding at the sector on which the receiving node is listening. More specifically, let nodes A and B lie in sector k of node C. Let node C lie in sector i of node A and in sector j of node B. We say that packet collision occurs at node C if the following events occur: node C is receiving on sector k; node A is transmitting on sector i; node B is transmitting on sector j. To communicate with neighbors, a node needs to switch between transmitting and receiving because it cannot transmit and receive at the same time. Moreover, a node needs to switch from one sector to another sector. The switching time from transmitting to receiving and from sector i to sector j are both assumed to be negligible. This assumption is reasonable for a fast switching directional antenna.
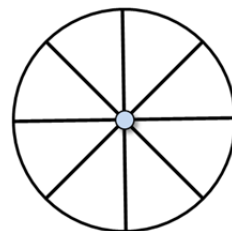

Figure 1: Sectors in a directional antenna.

Although nodes in the network are unsynchronized initially, we want to impose a transmission structure for each node to opportunistically exchange whatever information they have

about both their own as well as their neighbors' concept of time. The transmission structure here consists of three levels: epoch, frame and slot. An *epoch* is defined the period over which a node gathers received time information from its neighbors for the purpose of deciding how to modify its local clock time. It may consist of 10's or even 100's of frames depending on the expected density of the network. During each *frame*, a node decides whether to transmit or receive. If it chooses to receive, it will choose a sector and receive in that sector for the entire frame duration. If it chooses to transmit, the frame is further divided into a number of *slots*. The node will transmit in a sector for the duration of an entire *slot* before switching to another sector for transmission. The structure of epoch, frame and slot can be different for different node. That is, node A's frame length can be different from node B's frame length. Note also that the aforementioned timing structure is used for time synchronization purpose only. The system may or may not use it for normal communication (i.e., TDMA transmission) after the network is synchronized.

III. RTSR ALGORITHM

The basic idea behind RTSR is to use a low data rate signal to transmit time messages between adjacent nodes. Each node, in addition to its own time, transmits a subset of the time measurements it has received from other nodes. These additional measurements are used to enable "passive" Round Trip Time (RTT) measurement which allows a node to calculate its clock difference from the neighboring nodes. After repeatedly adjusting their clocks based on clock measurements of their neighbors, nodes can achieve network wide time synchronization. The primary advantage of RTSR is that there is no single node that has any more impact on the synchronization algorithm than any other node. This significantly limits the ability of a smart attacker to compromise the time synchronization algorithm. In the following, we first briefly describe a Baseline version of the RTSR algorithm which was presented in [1]. We then describe a Fast RTSR algorithm which exhibits significant performance improvement in convergence time.

A. Baseline RTSR Algorithm

In Baseline RTSR, the number of time slots in a frame is S (the number of sectors in the directional antenna). At the beginning of a frame, a node decides whether to transmit or receive for the entire frame according to the probability p_T. If a node choose to transmit, it sweeps through all of its antenna sectors (i.e., one sector per time slot). On the other hand, if it chooses to receive during that frame it will listen on one of its antenna sectors for the entire frame. Most transmissions will not be heard due to the mismatch between the transmitter's transmission direction and the receiver's receiving direction. However, once a while, transmission and reception directions will align, which will result in a successful transmission of the time information from the transmitter to the receiver. At the end of epoch n, node i will use the equation below to update

its clock to a new value $t_i(n)$ based on the time information received from its neighbors during the previous epoch. [1]:

$$t_i(n) = t_i(n-1) - \alpha \left(\frac{\sum_{k=1}^{K}[t_i(n-1) - t_k(n-1)]}{K} \right)$$

K is the total number of nodes from which node i has received clock information in epoch n. Node i computes the clock deviation between its own clock and each neighbor's clock, averages the the clock deviations, and updates its clock by adding or subtracting a fraction of that time (governed by the parameter α) For the baseline algorithm to perform well, we want as many exchanges of time information as possible in the network. The following theorem will guide us on choosing the optimal value of p_T to maximize the number of successful transmissions in the network.

Theorem 1: Given n nodes uniformly placed in an area of size A, with antenna sector area A_S, the optimal transmission probability p_T^* is given by:

$$p_T^* = \arg \max_{p_T} p_T(1 - p_T)[e^{-(n-2)\lambda} + \lambda e^{-(n-2)\lambda}(1 - p_T/s)]$$

where $\lambda = A_S/A$.

Proof: Let N_s be the number of successful transmissions occur in the network at a random instant. We then define I_l as follows:

$$I_l = \begin{cases} 1 & \text{if link } l \text{ has a successful transmission} \\ 0 & \text{otherwise} \end{cases}$$

Let L be a random variable denoting the total number of directional links in the network. We can write N_s as follows:

$$N_s = \sum_{l=1}^{L} I_l$$

Our objective is to choose p_T to maximize $E[N_s]$. $E[N_s]$ can be computed as follows:

$$E[N_s] = E[I_l] \cdot E[L]$$

Since $E[L]$ does not depend on p_T, we need to simply maximize $E[I_l]$. We know that

$$E[I_l] = \Pr(\text{link } l \text{ has a successful transmission})$$

Let (s, d) denote the two end points of link l, where s is the transmitter. Without loss of generality, we assume node d is receiving on sector k of its antenna. We then have

$$\Pr(\text{link } l \text{ has a successful transmission}) = \Pr(E_1) \cdot \Pr(E_2) \cdot \Pr(E_3)$$

where the events E_1, E_2, and E_3 are defined as follows:

- $E_1 \triangleq$ node s is transmitting
- $E_2 \triangleq$ node d is receiving
- $E_3 \triangleq$ no other node is interfering with node d's reception

Event E_3 can be further divided into the following events:

- $E_4 \triangleq$ node s is the only node located in the receiving sector of node d
- $E_5 \triangleq$ one or more additional nodes are located in the sector k of node d, but none of them is transmitting towards node d

Given n nodes are uniformly placed in an area of size A and the area of a single antenna sector is A_S, the probabilities are computed as follows:

- $\Pr(E_1) = p_T$
- $\Pr(E_2) = 1 - p_T$
- $\Pr(E_4) = (1 - A_S/A)^{n-2} \simeq e^{-(n-2)\lambda}$ where $\lambda = A_S/A$ is small
- $\Pr(E_5) \simeq \lambda e^{-(n-2)\lambda} \cdot (1 - p_T/s)$

To get $\Pr(E_5)$, we use the first order approximation. Instead of computing the probability that one or more additional nodes are located in sector k of node d, we simply calculate the probability that one node is located in sector k of node d. Let $b = e^{-(n-2)\lambda}$. The optimization problem becomes the following:

$$\max_{p_T} p_T(1 - p_T)[b + \lambda b(1 - p_T/s)]$$

Q.E.D.

If the probability of E_3 occurring is fairly large (i.e., the interference in the network is small), we simply need to maximize $p_T(1 - p_T)$, resulting the optimal $p_T = 0.5$.

Using the baseline RTSR algorithm, a transmitter will transmit on all of its antenna sectors during a frame, even those that do not point toward any neighbors. This can generate a lot of wasted transmissions. The value of α and the epoch length (i.e. the update interval) also impact the algorithms convergence speed.

B. Fast RTSR Algorithm

The baseline RTSR algorithm can take a long time to converge for certain choices of parameter values and certain topologies. In this section, we will describe an improved RTSR algorithm called Fast RTSR which achieves network time synchronization much more quickly. The fast RTSR algorithm consists of two stages: first, a Neighbor Discovery stage during which each node needs to find as many neighbors as possible; second, a Time Update stage during which nodes will only transmit or receive on antenna sectors that point to at least one neighbor. Our goal in the Neighbor Discovery stage is to maximize the number of neighboring node pairs that are aware of each other's existence during a given interval. We first describe a neighbor discovery algorithm which we call NDA1.

In NDA1, when a node is trying to find its neighbors, it has to transmit and receive on all of its antenna sectors since it has not figured out where its neighbors are located. As in Baseline RTSR, time is divided into frames comprised of S slots. At the beginning of a frame, a node needs to decide whether it will transmit or receive according to the probability p_T. The optimal p_T is chosen such that the network will have as many information exchanges as possible since the goal is to find as many neighbors as possible. Therefore, the optimal p_T obtained from Theorem 1 still applies here. Also like Baseline RTSR, nodes will transmit on one antenna sector for the duration of an entire slot, or listen on one sector for the duration of an entire frame. The information packet sent by a transmitting node can consist simply of its unique node ID. Once node B receives a packet is from node A on antenna sector k B knows that the A is a neighbor on sector k. If node A does not know that node B is a neighbor on sector k, A may not transmit on that sector in which in the subsequent Time Update stage. Hence, for knowledge of the neighboring node to be useful, both the transmitting node and the receiving node need to be aware of each other's existence.

NDA1 described in the previous paragraph will yield a set of node pairs that are aware of each other's existence. A second neighbor discovery algorithm, called NDA2, performs even better for a network with small geographic coverage. In NDA2, a transmitting node A still sends out a packet on antenna sector k for an entire slot. However, once node A finishes sending on sector k, it does not immediately move to a new sector for transmission. Instead, in the next time slot, A will switch to receiving on the same sector k. This gives neighboring node B a chance to immediately transmit back to A if B is located in sector k of node A. NDA2 takes advantage of correct transmitter/receiver alignment by allowing the receiver to transmit as well, which improves the probability that both transmitter and the receiver learn of each other's existence.

Figure 3 validates the performance improvement of NDA2 over NDA1. For comparison, we consider a network formed by placing 100 nodes uniformly over a square of 3300 meters by 3300 meters. The transmission range of each node is assumed to be 525 meters. The resulting network is shown Figure 2 which contains 350 distinct neighbor nodes pairs. After 160 time slots, 5.5 node pairs are discovered using NDA1 compared with 48 discovered using NDA2. This trend continues if more time slots are used for neighbor discovery. NDA2 is able to consistently discover 50 to 60 more neighbor pairs than NDA1.

NDA2 performs well in networks with small propagation delays. The efficiency of NDA2 will decrease if the duration of a time slot has to be enlarged to account for the propagation delay (i.e., the transmitter needs to wait for the propagation delay of its own packet and that of the replying packet). The neighbor discovery algorithms described here not only can be used to achieve time synchronization but also can be used for

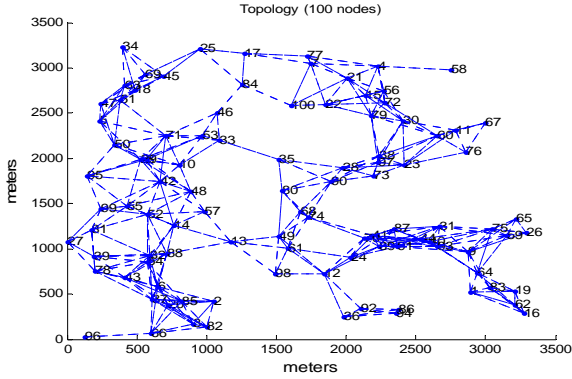periodic discovery of new neighbors after the network achieved time synchronization.

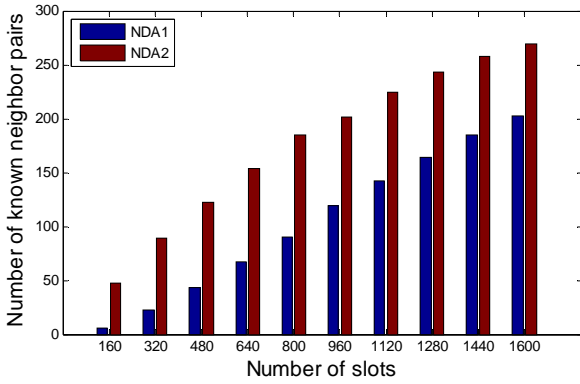
Figure 2: Topology of a 100 nodes network.

Figure 3: The number of neighbor pairs that knew each other's existence at the end of the specified number of time slots.

In the subsequent Time Update stage which follows Neighbor Discovery, each node will form its own frame structure. If a node knows the existence of four neighbors, it will set its frame duration to be four slots. At the beginning of each frame, a node will decide whether to transmit or receive based on p_T^*. Let S_N denote the set of sectors where at least one neighbor exists. Once a node decides to transmit, it will transmit on sectors that are in S_N, one time slot for each sector. If it decides to receive, it will randomly choose a sector from S_N on which to receive. This has the advantage of transmitting or receiving only on sectors where neighbors exist. The time update algorithm is defined as follows:

- At the start of the Time Update stage, node i generates a random integer weight W_i from $\{1, ..., J\}$ where J is large such that the probability W_i equals W_j is very small. For example, we can let J equal n^2 or n^3.
- Each node i will also have a pair of integers (U_i, V_i) to keep track of the origin of node i's clock. Specifically, U_i denotes the original node from whom node i derived its clock value; V_i denotes the number of nodes that have the same clock value as node i.

(U_i, V_i) are used to determine when the algorithm will stop.
- During each time slot, a transmitting node j will send three pieces of information to its neighbors: its current clock reading, (U_j, V_j), and W_j.
- Once anode i receives a time measurement from its transmitting node j, node i will check whether its clock is already synchronized with node j.
  - If the two clocks are synchronized, node i will set $V_i = \max\{V_i, V_j\}$. Since node i and node j are synchronized, both node i and node j's clock are based on the same node U_i. If $V_i \geq V_j$, this implies that node i has the latest information on the number of nodes which are synchronized with node U_i. If $V_i < V_j$, node j has the latest information, hence node i needs to update its value from node j.
  - If the two clocks are not synchronized, node i will compare W_i and W_j. If $W_i \geq W_j$, node i will do nothing. If $W_i < W_j$, , node i will update its clock to synchronize with node j's clock. Node i will also let $U_i = U_j$ and $V_i = V_j + 1$. V_i is increased to reflect the fact that there is one more node in the network (i.e., node i) that is synchronized with node U_i.
- During the time slot when V_i is equal to n for the first time, node i knows that the time synchronization is achieved for the entire network. However, it needs to let other nodes know that $V_i = n$. Hence, node i will continually transmit its time information to its neighbors for three additional frames after the first time at which it computes that V_i equals n.

At the end of Fast RTSR, every node in the network will be synchronized with a single clock value. Moreover, every node will also have received a signal that informs it that the entire network is synchronized (i.e., every node has the same clock value). Note that knowing when the entire network synchronizes is important since this information tells a node whether the RTSR algorithm should be stopped.

Theorem 2: Given a network with n nodes where each node has a finite number of neighbors, let $T(n)$ be the expected completion time of the Fast RTSR Algorithm. We have $T(n) = O(n)$.

We will briefly go over the major steps of the proof (see [3] for detailed proof). To prove the above theorem, we need to consider a random variable T_s, which is the amount of time that it takes for a node to have a successful transmission to one of its neighbors. For a node with finite number of neighbors, T_s depends only the number of neighbors; hence, it is not a function of n and is bounded. Consequently, E[T_s] is bounded also. Without loss of generality, let node k be the node with the largest weight (i.e., $W_k = \max\{W_1, \cdots, W_n\}$). As information exchange with node k takes place, the number of

nodes that are synchronized with node k will increase by one every $E[T_s]$ amount of time. Hence, We have $T(n) = O(n)$.

The significant convergence speed improvement of Fast RTSR over Baseline RTSR is shown in Figure 5. The topology of the network is shown in Figure 4. We consider a network of 20 nodes, each uniformly distributed in a 1500 meters by 1500 meters square. The transmission range of a node is 525 meters. The convergence time is defined as the time at which the maximum clock difference between any node in the network is less than 1 μs. The baseline RTSR algorithm uses an epoch length (update interval) of 800 time slots and an α value of 0.4. Transmission collision is also modeled in our simulation. Figure 5 shows the convergence process of *one run* of Fast RTSR and one run of Baseline RTSR. Fast RTSR converges in 240 time slots, while Baseline RTSR converges in 120,784 time slots. Running both algorithms twenty times, on average, the convergence time of the Fast RTSR algorithm is 336 time slots, and the convergence time of the baseline RTSR algorithm is 133,480 time slots. This is an improvement in convergence speed by nearly a factor of 400.
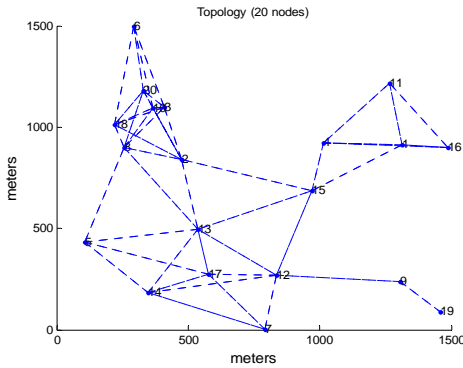

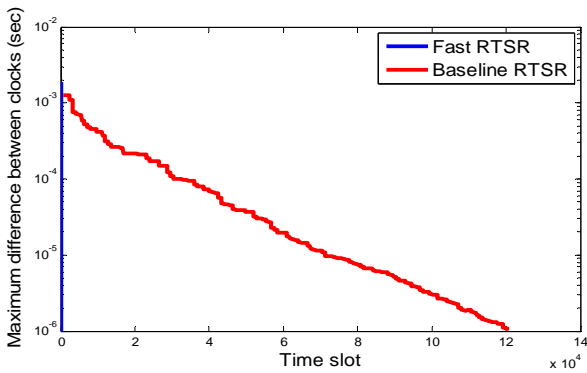**Figure 4: Topology of a 20 node network.**

Figure 5: Convergence process of the baseline RTSR algorithm and the Fast RTSR algorithm.

Factors that impact RTSR algorithm: the number of neighbors used in the algorithm

IV. PRACTICAL ISSUES OF RTSR

Our simulations indicate that the RTSR algorithm would be able to synchronize time across a network with infrequent signaling requirements. From analysis in the previous section, we see that the Fast RTSR algorithm allows a large network to achieve time synchronization quickly. However, throughout our analysis, we made an implicit assumption that the RF propagation delay from node i to node j can be measured accurately. We also assume that our network is static. Both of these assumptions, however, are unlikely to hold in reality. In the following section, we will discuss the impact of the RF propagation delay estimation and mobility on the performance of the RTSR algorithm.

A. Time Stamp Error

RF propagation delay measurement is based on the time difference between the moment that a timing packet is received at the receiver and the moment that the timing packet is leaving the transmitter. Ideally, the timing packet should include the time at which the packet is leaving the transmitter so that the receiver can use this information when it calculates the propagation delay. This requires knowing exactly when a timing packet will be sent out before this timing packet is even created. Alternatively, we can generate a packet which contains the current clock reading at the time of the packet generation. This packet will be transmitted after Δ_p seconds. Δ_p represents all of the Random Processing Delay (RPD) before the timing packet gets transmitted. It includes random latency in the operating system, scheduling delay, queueing delay, and the processing time in the transmission chain. Below we present our initial finding on the impact of Δ_p on the performance of RTSR algorithm. A detailed C++ Discrete Event Simulation (DES) is created to study the performance of RTSR algorithm. Currently, the DES simulates Baseline RTSR. Δ_p is first modeled as a constant and then as a uniformly distributed random variable. The impact of a constant Δ_p on the convergence of Baseline RTSR is shown in Figure 6. Figure 6 shows the convergence processes of constant random process delay of 10 ms, 1 ms, and 0.1 ms. The time synchronization is unaffected by the constant random processing delay.
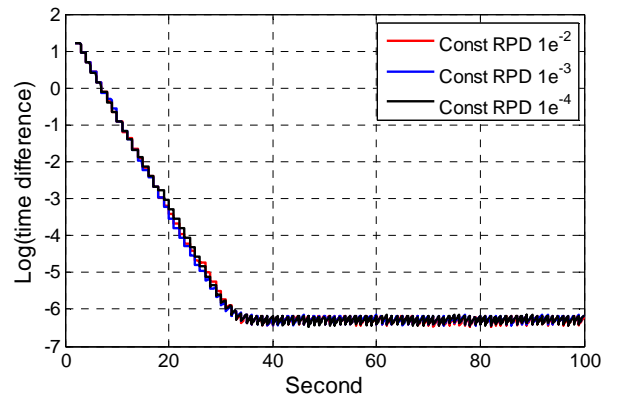

Figure 6: Maximum time difference among nodes with a constant random process delay (Baseline RTSR).

When the random process delay is uniformly distributed within a time interval of greater than [0 1e-5], Figure 7 shows that the network time *cannot* converge to the desired accuracy of 1 μs.
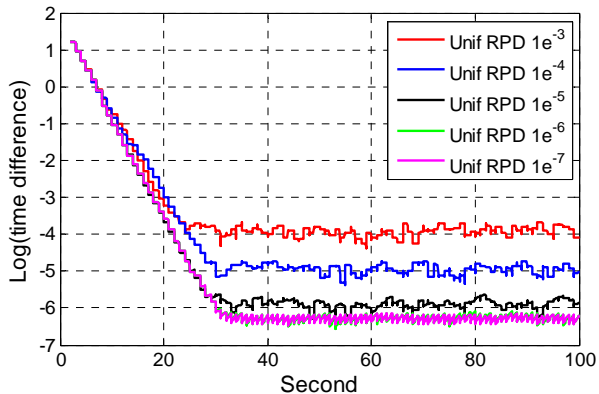

Figure 7: Maximum time difference among nodes with a uniformly distributed random process delay (Baseline RTSR).

B. RTSR Algorithm in A Mobile Network

In a mobile network, the impact of mobility on the time synchronization can be minimized by a properly designed RTSR algorithm. First, consider the following example of how round trip time is measured in the RTSR algorithm. Let two nodes, A B, be placed six kilometers apart. Suppose node A first sends a timing packet to node B. At the same time, nodes A and B are moving toward each other at 25 meters per second (56 miles per hour). After 20 μs, the timing packet arrives at node B. Instead of immediately sending a reply packet which contains node B's clock information, node B waits for five seconds before sending out its own timing packet to node A. Node B's timing packet takes 19.17 μs to reach node A since the distance between node A and node B is now 5750 meters. Upon receiving node B's timing packet, node A can deduce the round trip time between node A and node B to be 39.17 μs, and the propagation delay is 19.6 μs. The estimated propagation delay has an error of 19.17 – 19.6 = 0.4 μs. To reduce the estimation error, it is therefore critical for node B to reply back node A within a short amount of time after receiving node A's initial timing packet. For different node speed, Table 1 shows the propagation time estimation error if node B replies back A after five seconds.
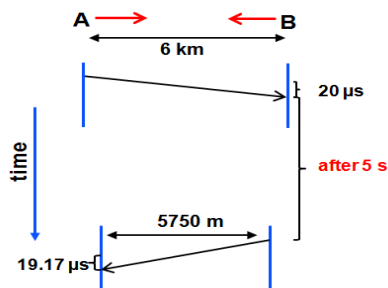

Figure 8: Round trip time estimation for mobile nodes.

Speed (m/s)	Speed (mi/hr)	Propagation Delay Variation (μs)
25	56	0.4
50	112	0.85
340	760.5	5.67

Table 1: Variation on the propagation delay estimation for mobile nodes.

V.   PATH FORWARD

Based on our analysis and simulations, it appears that an optimization enabled synchronization approach using RF signals can successfully enable tactical directional networking in a GPS denied environment. We are currently investigating the integration of the RSTR distributed algorithm into a directional networking environment for Army's use.

VI.   BIBLIOGRAPHY

[1] S. Pudlewski, "RF Based Time Synchronization and Ranging for Communications in a GPS-Contested Environment," in *IEEE MILCOM*, Tampa, 2015.

[2] R. R. Choudhury and N. H. Vaidya, "Ad Hoc Routing Using Directional Antennas," in *Technical Report, University of Illinois at Urbana Champaign*, 2002.

[3] J. Sun, C. Fossa, L. Herrera and D. Kelly, "Fast RF Time Synchronization and Ranging Algorithm," in *MIT Lincoln Laboratory Technical Report*, 2016.