Augur: A Delay Aware Forwarding Protocol for Delay-Tolerant Networks

Ahmad El Shoghri*, Branislav Kusy#, Raja Jurdak*, Neil Bergmann*

* School of Information Technology and Electrical Engineering, The University of Queensland, Australia
# Digital Productivity Flagship, Commonwealth Scientific and Industrial Research Organization, Australia

Abstract—Delay Tolerant Networks (DTN) are characterized by the absence of continuous connectivity resulting in high delivery delays that may exceed the acceptable limit for practical applications. In this paper, we address this issue by introducing Augur. Augur is a new routing protocol for DTNs targeted to minimize delays of message delivery. The routing scheme benefits from the spatiotemporal history data of the nodes to route messages only through gateways having less expected delay to deliver a message to its destination. We demonstrate through a comparative evaluation that Augur outperforms the state of the art DTN protocols in terms of delivery probability, overhead ratio and latency. We found that at low traffic rates Augur reduces the overhead ratio by up to 94%, and by up to 88% at high traffic. We also observed that the improvement in latency was reduced by up to half over the existing protocols in both traffic rates while still improving the delivery probability of messages.

Keywords—Mobile ad-hoc network, delay tolerant network, spatiotemporal data, opportunistic network, opportunistic network environment ONE, network simulation

I. INTRODUCTION

Recently, many wireless networks have emerged in various applications that lack pervasive network infrastructure, including extreme environments like deep space, deep oceans, battlefields and developing regions. The purpose of these networks is to gather data through wireless nodes dispersed in the field and transmit them to specific destinations for future use. However, in these environments there are no guarantees of continuous connectivity between the nodes, and this causes low delivery probability and long delivery delays. Some examples could be interplanetary networks, wild life tracking such as tracking cows and flying foxes, or providing internet connectivity to rural areas through vehicular ad hoc networks (such as buses, taxis, trains and tams). In these examples, data delivery delays are excessive. Although there are protocols to route in these environments, the nonexistence of an explicit delay aware protocol creates a gap in this area. Studies have shown that the participating nodes in the mentioned scenarios exhibit some periodic patterns in their daily tasks and their everyday movement. For instance, people commute with high regularity in terms of starting positions, destinations, starting and arrival times and routes they follow. Studies have also shown the existence of periodic mobility patterns for a range of animals and birds for certain activities like daily foraging and yearly migration. Another example is the scenario of bus movement which follows a specific time table to different stops and stations. Therefore, if the periodicity characteristic is used to estimate in advance the future presence of nodes at certain locations or the times at which they are reachable, significant improvements can be made to the performance of the intermittent wireless networks.

This paper aims to show how spatiotemporal data can be extracted from delay tolerant network (DTN) nodes. Then, the availability of this information will be exploited to produce a new algorithm, Augur and opportunistically route messages to their correspondent destinations in the lowest possible delay times. The remainder of the paper will analyze two aspects of Augur: its performance dynamics and how it compares to other protocols. First, the response of the new protocol is explored in relation to different simulation parameters, namely the message creation rate and the node buffer size. Second, the performance of Augur is compared to other protocols in terms of delivery probability, overhead ratio and latency.

II. BACKGROUND

Traditional data networks are modeled using connected graphs whereby the existence of at least one end to end path between any source-destination pair is always guaranteed [1]. Indeed, a message sent from a mobile device to a specific destination goes through a well designed network infrastructure consisting of communication towers, bridges and routers. In order to have Internet connectivity a device should be directly connected to one element of the infrastructure forming the network. If there is no connectivity when a message needs to be sent data will typically be discarded or dropped by the node carrying the information. For example, the widely used TCP is a connection oriented protocol which means that source and destination sides have to find definite stable routes between them before their application processes can send data [1]. In TCP/IP networks, data are routed through fixed routers over network links to provide paths to various destinations guaranteeing fast and reliable transmission. Characteristics like continuous connectivity, very low packet loss, stable general network topology and low propagation delay are implicitly assumed [2]. However, in DTN not all of these assumptions exist at the time when packet delivery is needed and sometimes none of them exist [1][2]. DTN especially addresses significant link delays and intermittently connected links in challenging
network environments that cannot be served by conventional TCP/IP protocol models [9][10][17].

Wireless computing devices often use a Mobile ad hoc network (MANET) which is an infrastructure-less self-configuring network that is effective in various networking applications where the infrastructure is absent, destroyed or impractical. MANET technology is used in data monitoring applications because of its ability to facilitate the collection of sensor data originating from distributed sensors in fields such as air pollution analysis, wild life animal tracking and vehicle traffic monitoring [2][3][21]. However, when the sensors are placed on highly mobile sensors the continuous change of the network topology is challenging. The objects’ motion will cause frequent link changes and disruption in the network; nodes might be interchanging data when they are within others’ range and might be disconnected when no neighbors exist in their radio transmission range [4][5][6]. This type of network is known as an Intermittently Connected Network (ICN) [1][2].

DTN is becoming an effective approach used to deal with the technical issues of ICN environments when the latter lacks continuous network connectivity. DTN is a class of network characterized by its intermittent connectivity [16]. DTNs lack an instantaneous end-to-end path between source and destination resulting in long variable propagation delays. Under conditions of extreme delays, limited bandwidth, node mobility and recurring communication obstructions, nodes should cooperate to guarantee the delivery of their messages and to maximize their lifetime. Thus, their protocol should adopt store-carry-forward communication approaches through which data are incrementally moved and stored throughout the network so that it will eventually reach its destination [7][8].

Routing in DTN seeks to maximize the delivery probability of messages and minimize their delivery delays [2]. Delays in DTN are unavoidable since it is an opportunistic network, however very long delays in some application scenarios limit the benefits that could be gained from this technology. To make this technology more effective requires more research and new routing protocols that actively minimize delays.

A parallel line of research has focused on DTN using personal communication devices. These devices hold valuable mobility information that can be exploited to improve DTN performance. Nearly 3.3 billion people worldwide use cell phones. Mobile phones are integrated with GPS, Wi-Fi, Bluetooth and large storage which create many contact opportunities. They are inexpensive and versatile tools [14]. Recent research used devices like laptops and phones to capture essential features of mobility for behavior modeling purposes. Spatiotemporal mobility statistics provide realistic measures to understand a mobile user’s behavioral preferences and transfer opportunities relevant to DTNs [15]. The time dependent behavior of a node and its periodic reappearance around similar times at certain locations for specific durations can estimate future presence of the node and aid future routing decisions [15]. The users holding the devices have a high probability of meeting again at similar times for similar durations. Mobility models can heavily affect the performance of the DTN protocols in terms of delivery delay. Therefore incorporating spatiotemporal characteristics in DTN routing protocols are expected to improve DTN performance [14][15].

III. ROUTING IN DTN

Routing in DTN is a challenging task due to the lack of a constant network topology over time. Therefore, routing protocols in DTN use store-carry-forward approaches. Mobile nodes have to take a series of independent relaying/forwarding decisions as they move. Some protocols prefer to take simple forwarding decisions such as relaying to every contact within range. Other protocols go for more complex decisions such as ones based on mobility patterns, energy availability, number of copies allowed (single copy/multiple copies) and other temporal conditions [18]. The following sections review the most common DTN protocols based on their forwarding decisions.

A. Store-Carry-Forward

Store-carry-forward has become a key concept used in DTN technologies [2][9][11]. Store-carry-forward is an asynchronous message passing paradigm that a node follows after receiving a message. The “Store” phase is adding the message to the node’s buffer which allows the data to wait for a suitable time or peer to forward the message. “Carry” is the stage that allows the message to propagate to other regions of the network physically through the movement of the node carrying the data instead of relying on its transmission through the limited available network media. Finally, “Forward” is the stage when the node decides to send the message to another node due to the availability of other better candidates or to the message’s final destination [9].

B. Flooding Based Routing Approaches

1) Epidemic
In the Epidemic routing protocol, a node carrying a message forwards a copy of the message to all other nodes encountered on its path. Similarly, the receiving nodes will follow the same behavior [16][18]. If the encountered node already has a copy, the message is ignored. In this algorithm, the message will spread through all available nodes in the network until hopefully being delivered to the destination by one of them [16][19]. Clearly, this protocol maximizes the message delivery probability at the expense of very high consumption of the available node and network resources such as the network bandwidth and the node’s memory [16].

2) Spray and Wait
The Spray and Wait protocol exists in two main versions namely, vanilla and binary. The two mentioned versions differ in the way they flood the L copies of the message to the nodes during their “spray phase”, where L is initialized by the source node. Vanilla is the simplest and it transmits the L copies to the first L-1 distinct encountered nodes. A node following the binary version of Spray and Wait starts by transferring half of the L copies to the first node it encounters and the other half to the second encountered node. The receiving nodes then follow the same behavior and so on. The second phase is identical for
both versions, where after a receiving node is left with one message it enters the “wait phase”. In this phase, the message can be delivered only by direct delivery when the carrying node meets the destination. This algorithm reduces the use of the network bandwidth through the parameter \( L \) that controls the number of copies of the message in the network. The disadvantages for this protocol are high delays and system failure if the nodes that received the copy of the message never cross paths with the destination [2][19].

C. Estimation Based Routing Approaches

Probabilistic History based on Encounters and Transitions ProPHET is a well known routing protocol in DTN that aims to use statistics of previous encounters made by a node with other neighbors. These encounters will be used to build and update a probability decision metric that estimates the probability of delivering the message to the destination. The delivery predictability estimate increases at each node encounter and decreases exponentially through time. The ProPHET protocol also takes into account the case where two nodes rarely meet but they frequently encounter a node in common through a transitivity parameter \( \beta \) [16][18]. While ProPHET uses information about the delivery probability, it does not take into account the expected time delay for that delivery.

IV. AUGUR ROUTING PROTOCOL

In this section our delay aware protocol based on spatiotemporal data is presented. First, we introduce how a node gathers spatiotemporal information to build its routing decision metric. Second, we present how the nodes make use of this data to forward a message greedily to the optimal local candidate having the least expected delay until hopefully reaching its destination.

In DTN, a node is characterized by a specific interface transmission radio range. A connection or a contact is realized when two nodes are within each other’s radio transmission range. The time interval in which the two nodes stay in contact is called contact durations. Therefore by combining these two parameters the connections and their contact duration a node can derive all meeting and intermeeting times (time between consecutive meetings) with other nodes during a day. Basically, Augur makes use of the time series of meeting and intermeeting occasions to understand the underlying movement topology, to predict future contacts with the sinks and to route messages only through nodes having less expected delay until reaching their destinations.

A. Time Series of Connections and their Durations

The time series is an ordered sequence of observed data on a variable of interest during a specified time interval. Usually, a time series is represented by \( Y = \{ y_t ; t \in T \} \) where \( T \) is the index set and \( t \) is an instance of the index set.

In Augur, each node in the network produces two independent time series of connections and contact durations at the end of each day. Fig. 1 shows a portion of the time series of connections and durations in terms of time of the day. These time series relate the observations seen during a day with a specific time and location at which they occurred. Augur saves up to 30 observation days in its history. At this stage, a node is able to derive a new time series which summarizes its delays to meet a sink from any point in time within the day. The delay value in \( y \), where \( t \) is a point in time within the current day, is computed as follows. From the historical data, Augur averages all the delays during the previous days from time \( t \) till the reach of the earliest sink. This average is then used as the current day estimate at time \( t \) to reach the next sink. Similarly, a node will follow the same procedure in creating the time series vector holding the average contact durations. Based on this information, from time \( t \), each node can predict its delay until the next opportunity to meet a sink and also its contact duration time with that sink. Consequently, this knowledge will greatly assist nodes in performing effective routing decisions in DTNs.

B. Augur Forwarding Algorithm

This section presents the details of the Augur forwarding algorithm whose pseudocode is shown at the end of this section. When two or more nodes are within contact range, each of the nodes broadcasts a predicted delay value that reflects the node’s earliest opportunity to deliver the message to one of the sinks. This broadcasted delay has two components: the time until the node reaches a sink; and the time till the message reaches its turn to be delivered depending on its position in the queue\(^1\). By adding these two components the node’s expected delay time for message delivery can be calculated. The queue delay is then compared with the duration of contact with the sink to check the availability of enough time to deliver the message. This check insures that a node having less delay does not get overloaded with relayed messages that it is unable to deliver. If the queue delay exceeds the contact duration the node will look to its next connection with a sink and broadcasts the correspondent delay

\(^1\) In Augur, messages are prioritized based on their creation time, where newer messages receive higher priority.
value till delivery. In general, the nodes in the network will tend to relay as many of their messages to neighbors broadcasting lowest delay values. Fig. 2 shows the interaction between two nodes in contact.

The buffer strategy adopted in Augur in the case of full buffers is dropping the oldest message in the node’s buffer to guarantee the fastest possible delivery for newer messages. In this way, the message will follow the least encountered delay route at each contact and hence improve the protocol performance in terms of all metrics, especially the latency.

Augur algorithm

1. IF (thisNode.connectionSize > 0)
   2. messages = thisNode.getMsgCollection.orderDescendingCreationTime
   3. IF (otherNode.isSink)
   4. Update connection time series
   5. Update duration time series
   6. Update delay decision metric
   7. IF (thisNode.msgCollection.size > 0)
   8. Deliver messages
   9. ELSE
   10. Broadcast delay
   11. IF (thisNode.delay < otherNode.delay)
   12. Accept relay message m from otherNode
   13. IF (m.size > thisNode.bufferSize || thisNode.hasMessage(m))
   14. Decline message
   15. While (thisNode.freeBufferSize < m.size)
   16. oldestMessage = messages.getLastItem().creationTime
   17. IF (oldestMessage.creationTime < m.creationTime)
   18. msgCollection.remove(oldestMessage)
   19. thisNode.receive(m)
   20. ELSE
   21. for msg : messages
   22. Relay msg

In order to analyze the benefits and performance of Augur algorithm mentioned in section IV and compare it with other protocols ONE 1.4 was used since it includes different routing protocols such as Epidemic, Spray and Wait, ProPHET and Direct Delivery and visualizes the simulations interactively in real-time and provides the various reports after their completion. Fig. 3 shows a screenshot of ONE simulator in action.

In order to simulate the behavior of Augur in different occasions we extended the ONE simulator and added 2 new routing classes AugurRouter.class and RelayedMessagesInfo.class. The core of Augur algorithm is implemented in the AugurRouter class which models the gateway of each node and it is responsible for building and updating the node’s forwarding decision metric. We also extended the original ActiveRouter and MessageRouter classes to AugurActiveRouter.class and AugurMessageRouter.class to realize the correct behavior of Augur.

V. SIMULATION SETUP

A. Simulation Tools

To evaluate Augur the Opportunistic Network Environment (ONE) was used. ONE is a special simulator designed to address routing in Delay Tolerant Network environments. ONE is a powerful tool used to implement realistic DTN scenarios with different routing protocols, mobility traces and event generations.

B. Motivating Target Application

There are various scenarios that exhibit regular patterns with episodic connectivity in DTN. An interesting scenario to examine is in the area of vehicular ad hoc networks using the Helsinki City map that already exists in the libraries of ONE. This application considers a city bus network in which the communication nodes consist of buses moving along predefined bus trajectories in Helsinki city. The buses travel circularly or back and forth on the routes of this map. This route based model consists of predefined locations as bus stops or bus stations in which each node waits for some time before it continues the journey to the next stop using the shortest path algorithm. The sinks or hotspots are fixed and considered to have stationary movement located at specific bus stations of Helsinki map. In this scenario, all the bus nodes move with a constant speed on the map routes to maintain a repetitive movement every day.
For this scenario, Table I summarizes the setup and the different simulation configurations used for our experiments:

<table>
<thead>
<tr>
<th>Environment Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total simulation time</td>
<td>432000 seconds = 5 days</td>
</tr>
<tr>
<td>World size</td>
<td>Helsinki City Map</td>
</tr>
<tr>
<td>Number of Bus Nodes</td>
<td>25</td>
</tr>
<tr>
<td>Number of fixed sinks</td>
<td>5</td>
</tr>
<tr>
<td>Bus Movement Model</td>
<td>BusMovement (MapRouteMouvement)</td>
</tr>
<tr>
<td>Interface Transmission Speed</td>
<td>250kBps</td>
</tr>
<tr>
<td>Interface Transmission Range</td>
<td>100 meters</td>
</tr>
<tr>
<td>Node Buffer Size</td>
<td>10M; Infinite</td>
</tr>
<tr>
<td>Node Movement Speed</td>
<td>7 m/s</td>
</tr>
<tr>
<td>Message Creation Starting Time</td>
<td>172800 seconds = after 2 days</td>
</tr>
<tr>
<td>Message Creation Rate</td>
<td>15; 30; 60; 120; 240 seconds</td>
</tr>
<tr>
<td>Message Time To Live</td>
<td>120 minutes * (the TTL value of 120 minutes is large enough for this scenario)</td>
</tr>
<tr>
<td>Message Size</td>
<td>250KB</td>
</tr>
</tbody>
</table>

VI. PERFORMANCE EVALUATION AND DISCUSSION

A. Performance Metrics

In all the experiments considered in this paper, the DTN routing protocols are analyzed and evaluated using the following terminologies and performance metrics:

- Created messages (CM): is the total number of messages created by all the nodes during the simulation.
- Relayed messages (RM): is the number of messages that were successfully transmitted between nodes.
- Delivered messages (DM): is the total number of unique messages successfully received by the sink nodes.

Delivery probability (DP): reflects the fraction of created messages that has been received correctly at the receiver side within the simulation time period.

\[ DP = \frac{DM}{CM} \]

Overhead ratio (OR): estimates the number of transmitted messages which are not used for delivery, compared to the number of transmitted messages which are directly used for delivery.

\[ OR = \frac{RM - DM}{DM} \]

Average latency (AL): gives the average of the times taken by the delivered message from their creation to their first delivery at the destination sink.

\[ AL = \frac{\sum_{m=1}^{DM} (delivery\ time\ of\ Mn - creation\ time\ of\ Mn)}{DM} \]

2 The TTL value of 120 minutes is large enough for this scenario.

B. Simulation Results

1) Exploring the Performance of Augur:

To capture the performance of Augur under different possible available resources, we start our study by exploring how Augur responds to the variation of the nodes’ buffer size. To show this the message creation rate is fixed to 1 message every 20 seconds at each node and the message time to live TTL value is set to 120 minutes. The experiment is run for 9 different buffer sizes consecutively [4MB; 6MB; 8MB; 10MB; 12MB; 14MB; 16MB; 18MB; infinite] and 4 main metrics are tracked: delivery probability, overhead ratio, average delay and channel utilization.

Fig. 4 shows that increasing the nodes’ buffer size results in a higher number of successfully delivered messages since the nodes are carrying higher number of messages that can be delivered during their contact time with the sinks. This can be clearly shown through the increase in the channel utilization metric plot for buffer sizes from 4MB to 14MB. Further increases in the buffer size seem to give negligible improvement in the delivery ratio due to saturation in channel utilization and in opportunities to deliver messages as can be seen in the channel utilization metric that reached around 99% for a buffer size larger than 14MB.

Fig. 4. Performance of Augur for varying buffer size

Also notice that the overhead ratio decreases with the increase of the nodes’ buffers size. The explanation for this phenomenon is because the same number of messages should have been created by the end of each simulation (32400, same message creation interval); the larger the buffer size the more likely that the node getting in contact range of other nodes also has the same candidate message because it was not dropped before due to a full buffer or TTL expiry. The increase in the delivery probability in addition to a lower number of relays will lead to a lower overhead ratio. For the last metric which is the average latency, Fig. 4 shows that it increases with the increase of the buffer size however this higher delay is primarily due to more delivered messages. The larger buffer size allowed the delivery of larger number of messages, by that means the messages waited for longer time in the buffer until being delivered and consequently altered the overall average latency.

2) Comparing Augur with other Protocols

In this section, we conduct a comparative evaluation among the proposed Augur algorithm, SnW in its two versions
binary and vanilla where L=5 for both, ProPHET, Epidemic, as well as two simple protocols namely Direct Delivery and First Contact for comparison purposes. In Direct delivery no relays are permitted, whereas in First Contact the node that creates the message relays 1 copy of it to the first neighbor it meets and then no further relays of this message are allowed. The metrics of interest are delivery probability, overhead ratio, average delay and channel utilization. Here, we highlight that the desirable properties of the DTN routing protocol are to maximize delivery probability, minimize average latency and minimize the overall number of relays.

An important metric for the remaining simulations is the buffer size which may have an impact on the performance of the protocols. Fig. 5 illustrates the effect of increasing the buffer size on the different metrics for Augur and other protocols. The figure shows that the performance of Augur is relatively insensitive to the increase in buffer size, whereas all other protocols are much more sensitive. When the buffer size is increased the other protocols get much worse latency values.

![Fig. 5. Relative gain/loss between 10MB and infinite buffer.](image)

It should be noticed that our proposed protocol has no assumptions about buffer size or preference towards any other parameter. For this reason, since our protocol is delay aware and this paper focuses on minimizing the delay metric in DTN, we choose a buffer size of 10MB to not disadvantage other protocols.

**Delivery probability**

Fig. 6 shows the delivery probability of the seven different routing schemes under various traffic loads varying from 1 message every 15 seconds to 1 message every 240 seconds. The increase in the message creation interval resulted also in an increase in the delivery ratio of all the protocols we considered. However, the delivery probability of Augur is always higher than the delivery probabilities of the other 6 schemes through all the experiments. The delivery probability of Augur becomes higher as the message creation rate slows starting with 0.4 at 15 seconds till reaching around 0.99 for a message interval of 120 seconds and above. In the case when the message creation interval was short namely every 15s and 30s, the delivery ratio of Augur was higher than all the other protocols with a value of 0.45 and 0.76 respectively followed by Direct Delivery with 0.44 and 0.72 respectively. Note that Direct Delivery performed better than the rest of the protocols except Augur in terms of delivery probability at high traffic loads. This means that when nodes are facing high traffic in the network it is better to stop taking any relaying decision similar to those done by the other protocols. These protocols appear to overload specific nodes with high number of messages. The nodes become unable to deliver them later on due to the limited TTL, buffer size, bandwidth and contact time with the sinks. This leaves other nodes in the network with a small number of messages to deliver during their contact time with the sinks and the remaining contact time is wasted. This can be clearly seen in Fig. 7 by the lower usage of the available channel by SnW, ProPHET, Epidemic and First Contact protocols at 15s and 30s, however on the contrary, Augur followed smarter strategies of load balancing among the nodes leading to a more efficient use of channel capacity that reached around 94% and 77% at 15 and 30 respectively. It is worth noting that the flooding based routing protocols used almost the full channel capacity at all times but most of the delivered messages were redundant and previously delivered by other nodes. These messages are not counted in this metric because the links were not efficiently used.

![Fig. 6. Delivery probability for considered protocols for varying traffic loads](image)

**Overhead ratio**

Another important metric is the overhead ratio which gives an idea about how efficient the protocol is in terms of correct relay decisions and energy consumption.
Fig. 8 illustrates the influence of the message interval on the overhead ratio. Obviously, the overhead ratio of Direct Delivery remains 0 since no relays are made by this protocol. Augur performs significantly better than SnW, ProPHET and Epidemic in terms of overhead ratio with the ability to keep its value less than 0.5 for all simulations. In this scenario, Augur reduces the overhead ratio by 77% to 94% for low traffic rates and by 73% to 88% for high traffic rates. The low value of overhead ratio of Augur at high traffic loads means that only accurate relays were made or in other words most of the relayed messages to other nodes were successfully delivered. This can be understood by looking at the low number of relays made by Augur at 15s shown in Fig. 9 compared to those made by the other protocols. This decision of limiting the number of relays at high traffic loads allowed Augur to excel in saving bandwidth and maintain higher delivery accuracy.

The delivered messages are sorted in ascending order with respect to their delivery latency, and then the ordered messages are divided into 10 equal size bins and the average latency for these bins is calculated. The first bin includes the fastest 10% of messages; the second one includes the fastest 20% of the messages, and so on. Fig. 10 shows the average latency of the protocols for different traffic loads, 15, 30, 60 and 120 respectively. These plots can illustrate two metrics at a time. The x axis approximates the delivery percentage of each protocol and the y axis gives the average delay for a given percentage of delivered messages. Note that the right most point of each trace indicates the overall delivery rate and the overall average latency of the simulation for the corresponding protocol.

From the figures, we can state that at low traffic rates Augur, binary Spray and Wait and vanilla Spray and Wait perform similarly in terms of average latency with an average difference around 3.5 seconds for Augur across all bins. In other words, Augur is able to deliver each message 3.5 seconds quicker than the closest other protocol candidate. The time saved is approximated by a reduction of 3% to 54%. As the speed of message creation increases the performance of the other protocols deteriorates and that the delay difference between Augur and the closest competitor starts to be clearly seen. The average delay improvement of Augur compared to the closest performing protocol increases to reach 25 and 53 seconds respectively for the rates of 60 and 30 seconds scenarios. At the highest tested traffic rate of 15 seconds, Augur outperforms all other protocols in reducing the delivery delays. Augur saves around 60 seconds and decreases the average latency by 32% to 46% from binary Spray and Wait which was the best among the rest of the protocols. The simulations showed that Augur performs best again in the latency metric followed by Spray and Wait in its two versions then ProPHET, Epidemic, Direct Delivery and First Contact which came last. Higher delivery probability was reached by Direct Delivery for fast message creation rates at the expense of delivery latency. This is due to keeping 1 copy of the message in the network so no redundant deliveries are taking place.

Latency

Next, the performance of the all protocols is analyzed in terms of latency.
the bandwidth of other new deliveries. Augur follows the same strategy and improves its latency by load balancing among nodes, relaying and delivering newer messages first, dropping the oldest message in case of a full buffer and following lower delay routes for quick deliveries.

The analysis clearly shows that Augur gives the best results for delivery probability, overhead ratio and latency for all traffic loads. The superiority of Augur stems mainly from the use of spatiotemporal data of meeting and intermeeting occasions with sink nodes. This information gives the nodes the ability to approximate delays until likely future connections with sinks in addition to their durations. In the adopted periodic movement scenario, routing based on this metric showed an exact match between the estimated values and the actual meetings. This led to a lower value of overhead ratio, and since the forwarding algorithm leads the messages to follow the least possible delay routes, we noticed that the messages are delivered faster, saving more bandwidth and giving the opportunity for other messages to be delivered. Consequently, Augur increases the overall delivery ratio.

VII. CONCLUSION AND FUTURE WORK

Delay in DTN is unavoidable because it is an opportunistic network, however very long delays in some time critical scenarios limit the benefits that could be gained from this technology. Since real life movement often exhibits some periodicity factor such as ones present in taxis, buses and human mobility, it has been shown in this work through Augur that with the use of spatial and temporal information of nodes, the required delay until the delivery of a particular message by a given node can be estimated and this improves the quality of service of routing protocols in DTN. Simulation results clearly showed that the Augur routing protocol outperforms Spray and Wait, ProPHET, Epidemic, Direct Delivery and First Contact routing significantly in terms of the performance metrics especially the overhead ratio and the average latency. The overhead ratio was reduced by 77% to 94% and by 73% to 88% at low and high traffic rates respectively, while the average latency was improved by 3% to 54% at low traffic rates and by 32% to 46% at high traffic rates.

Note that in this work, the sinks are considered stationary and that all nodes are moving periodically with a constant speed each day. This effectively means that the spatial aspects have limited effects on the results. An interesting direction in this research is to remove these constraints and further explore the performance of Augur in real data traces and more complex movement. This would require a weighted moving average delay where larger credibility weights are given to the more recent events. Augur is expected to provide a framework for using spatiotemporal history of nodes encounters to improve the protocol delivery rates and to reduce its latencies.

REFERENCES


