Supporting Information on Demand with the DisServicePro Proactive Peer-to-peer Information Dissemination System

Silvia Rota, Giacomo Benincasa, Matteo Interlandi, Niranjan Suri, Brian Bonnlander, Jeffrey Bradshaw
Florida Institute for Human & Machine Cognition
Pensacola, FL USA
{srota, gbenincasa, minterlandi, nsuri, bbonnlander, jbradshaw}@ihmc.us

Mauro Tortonesi
University of Ferrara
Ferrara, Italy
mtortonesi@unife.it

Scott Watson, Kevin Boner
Space and Naval Warfare Systems Command
San Diego, CA USA{scott.c.watson, kevin.boner}@navy.mil

Abstract— **Tactical networks are highly dynamic environments characterized by constrained resources, limited bandwidth, and intermittent connectivity. The limits on communication cause significant delays in the delivery of information to edge users. This paper focuses on an approach to improve the timeliness of access to information via prediction and pre-staging. The approach also incorporates a learning mechanism to dynamically adapt the information prediction algorithm. This capability has been integrated into the DisService peer-to-peer information dissemination system, which opportunistically exploits any available connectivity to address the challenging environment. The extended system, called DisServicePro (for Proactive) predicts the information needs of edge users using their mission description, including the routes that users may take as part of the mission.**

DisServicePro extends the capabilities of DisService by efficiently and proactively disseminating information to the edge nodes by means of replication and forwarding policies. The proactive behavior is the result of the integration of policies and a distributed learning algorithm that takes into account the history of previously requested information, along with the characteristics of the target nodes and the mission. As new information becomes available, DisServicePro matches it against the mission profile and pushes relevant information to the edge nodes. Information that is selected to be pushed is sorted based on the predicted time to use as well as the confidence value of the prediction.

Keywords: tactical networks, peer-to-peer, proactive information dissemination, decision trees, dynamic information prioritization, pre-staging, information on-demand.

I. INTRODUCTION

Tactical military environments are characterized by dynamic topologies populated with highly mobile wireless devices that are operated with limited range, processing power, memory, and storage. This results in intermittent connectivity and frequent changes in bandwidth and latency. In spite of these challenges, users require robust, reliable and timely information dissemination on-demand. In these environments it is critical to fully exploit sporadic temporary connections when they are available. In this paper we present DisServicePro, a proactive information dissemination service for tactical networks. In previous work [1], we described DisService, a peer-to-peer information dissemination system that opportunistically discovers and exploits available connections and uses a Pub/Sub approach to disseminate information through broadcast and multicast. DisServicePro extends DisService capabilities by implementing a proactive approach that leverages the opportunistic protocols implemented in DisService with predictions and careful prioritization of information that has to be disseminated. These characteristics make DisServicePro particularly suitable for disseminating situational awareness and on-demand information. The main contributions of this work are summarized as follows:

- *An adaptable information model*: DisServicePro models the context and mission of each user and the content of information traveling in the network in order to anticipate the future needs of the users. The model is flexible and adapts to mission-specific requirements.
- *Innovative replication and forwarding policies*: DisServicePro proactively sends information to the nodes ahead of time by taking advantage of an efficient combination of replication and forwarding policies.
- *Proactive push and pull protocols*: As new information becomes available, DisServicePro matches it against the mission and user context profile and pushes relevant information to the selected nodes.
- *Information anticipation through predictions*: Information anticipation improves the data access time and decreases the transmission overhead when the system correctly chooses to not send irrelevant information. DisServicePro uses a distributed learning algorithm that learns and follows the evolution of the users' preferences over time. The learned preferences are utilized to infer which information is relevant in real time.

Dynamic context-aware information prioritization: DisServicePro dynamically prioritizes the information being exchanged both to a user as well as across users. The prioritization takes into account both the policies and the predictions.

The rest of the paper is organized as follows. Section II reviews related work. Two motivating scenarios are presented in section III. Then, section IV describes the DisServicePro architecture. Section V presents DisServicePro capabilities. Details about the performance evaluation are given in section VI. Finally, section VII presents conclusions and discusses future work.

II. RELATED WORK

Approaches that use machine learning for the purpose of information dissemination can be found in [4] and [5]. The ADAMANT framework proposed in [4] uses machine learning algorithms to improve adaptability of Pub/Sub systems in dynamic environments. In particular, three algorithms (decision tree, neural network, and linear logistic regression classifier) are trained off-line and their performance in predicting and configuring the most suitable transport protocol given the environmental constraints are compared. In contrast, DisServicePro aims to enhance the Pub/Sub approach by proactively sending information ahead of time and exploiting on-line learning. In [5], an approach to disseminate situation awareness information in MANETs is presented. The decision to forward or drop data is distributed among the peers, which prioritize the information based on its probability of being new to target nodes. The information novelty is learned locally by each peer using the proposed MALENA algorithm that uses a sliding window approach to train a machine learning algorithm on-line. This approach differs from DisServicePro in the type of information being disseminated. Disseminating on-demand information poses extra challenges, since the nodes' interests need to be modeled, learned, and made available to other nodes. Moreover, information has to be sent just to interested nodes in a timely manner.

Other approaches in the literature address the challenges of dynamic MANET environments by implementing ad-hoc policies and protocols. An interesting approach is proposed in [7]. The introduced RANDI algorithm takes into account resource constraints in terms of energy, bandwidth, and storage. The information is disseminated using a combination of on-demand query requests and replication policies that pre-stage the newest data on popular nodes. The algorithm also dynamically prioritizes the information based on its size and popularity. Another approach, called CSI [3], represents communication targets by behavioral profiles instead of identities. The profiles represent nodes' mobility in spatiotemporal patterns and are calculated off-line, assuming that they remain stable for several days. When sending information, nodes need to specify the target profile recipient. The information is forwarded based on a gradient ascent policy, assuming that nodes with similar profiles are spatially close to each other. When calculation of the target location is not possible, a heuristic dissemination algorithm is used. DisServicePro approach differs from all of them. In DisServicePro, the dynamic prioritization takes into account both the content of the information and the target node's interests learned on-line. Moreover, in DisServicePro, the information dissemination is proactive and takes advantage of a combination of both pull/push and replication/forwarding mechanisms.

Techniques for pre-staging information are also studied in different contexts. An interesting approach is described in [9], where the goal is to pre-fetch web pages that the user is likely to request in the future on the client node. Two kinds of predictors are utilized to decide which pages to pre-fetch among millions of links stored in a hash table on the local client. The first predictor is based on the PPM (Prediction by Partial Matching) algorithm and analyzes the history of previously visited pages, while the second one is composed of different content-based policies. Notice that here the context comprises two nodes (client and server) connected over the Internet, so there is no need to address the issues typically related to tactical networks and resource constrained nodes.

Approaches that study learning in dynamic environments and distributed learning have also been investigated. In [11], an algorithm to deal with concept drift using multiple classifiers is proposed. A new classifier is generated every time a concept drift is detected. Then, the classifiers are combined through dynamically weighted majority voting, where weights are determined based on classifiers' age and accuracy on current data. Prior classifiers are not discarded, allowing the system to learn cyclical environments. A different method is proposed in [6], where concept drift is handled by detecting changes in the probability distribution of the incoming data. The error rate of the learning algorithm is traced during time and, when it decreases under a threshold, a concept drift is detected and a new classifier is constructed. Given the constraints of tactical networks, the first approach is not suitable. Since the classifiers are never discarded, the approach is not scalable in terms of memory consumption. Moreover, all the classifiers need to be consulted in order to obtain a prediction, meaning that, in a distributed environment, the bandwidth consumption increases as a function of the number of classifiers to exchange. More interesting is the idea to modify the size of the training dataset window based on concept drift detection as proposed in [6]. However, the test section of the paper shows a drawback. The approach presents peaks of very low accuracy in correspondence to concept drifts, compared to a single classifier that has more stable performance. For DisServicePro, the performances stability during time is a valuable property that permits trusting the classifier predictions. In [10], Núñez presents an interesting approach that exploits machine learning to predict future faults in the context of network management. Like DisServicePro, the system needs to share information collected from different sources. The collected information is stored in a centralized repository that manages the regression trees. The information is sent in the form of rule-sets. The authors' assumption of fixed nodes and wired connections, and the usage of a centralized repository are not suitable for DisServicePro.

III. MOTIVATING SCENARIOS

Two scenarios are considered to help understand the requirements targeted by DisServicePro. In both of these

scenarios, the assumption is that the force being deployed carries portable computer devices (e.g., ruggedized Personal Digital Assistants (PDAs), laptop computers, or other man-wearable or man-portable computers), which will be generically referred to as nodes. These nodes have network connectivity that is limited and intermittent, caused by lack of radio coverage, resource contention, or the desire to operate in a clandestine manner and maintain radio silence. While on the mission, soldiers need to access a variety of information including maps, aerial reconnaissance, other sensor data (e.g., from unattended ground sensors), intelligence reports, and blue and red force tracking. Some of this data may be pre-loaded onto the nodes (e.g., maps of areas where the soldiers are expected to be, as determined by the mission). However, the nodes will need to receive new data after deployment for two reasons. The first possibility is that new data may become available that did not exist prior to deployment or was not accessible prior to deployment. The second possibility is that a change in the mission requirements or the mission execution (e.g., the soldiers having to deviate from the pre-planned path) requires new data to be sent to the nodes.

The first scenario involves a Non-combatant Evacuation Operation (NEO), where forces are sent into an area to secure an objective and evacuate identified personnel from an unstable or otherwise threatened area. A recent example of a NEO was Lebanon during 2006, when personnel were evacuated from Lebanon to Cyprus. One important characteristic of such scenarios is that the duration of the operation is typically short – on the order of hours instead of days. Shorter duration missions provide the possibility that prior to deployment of a force, the systems being carried by the soldiers can be preloaded with information relevant to the mission. Once the force is deployed in theatre, the system will need to automatically retrieve information that is relevant to the force and to each member of the force. For example, any information gathered from ground (or other) sensors that are along the path being traversed would be of interest. Any new information available about enemy or friendly activity or forces will need to be made available as well. Such information may originate from the operations center or from other teams or missions that are overlapping in space and or time. While the scope for the number of changes or deviations from the planned mission may be small, it is still important to push new relevant information to the force post deployment.

The second scenario involves a typical Ship to Objective Maneuver (STOM) that may last for several days. As in the NEO case, nodes can be pre-loaded with mission-relevant information while the force is on the ship. However, given the long duration of such missions, it is much more likely that there are changes to the mission and orders, and consequent adjustments to planned activities. In such cases, new map, reconnaissance, and other relevant information will need to be retrieved by the nodes. New information may also become available post deployment, either back at the operations center or from some other unit in theatre. If so, this information needs to be pushed to the nodes. Also, given the longer duration of these missions, the deployed force may have more autonomy in the planning and strategy for the execution of the mission, which will require a flexible information retrieval mechanism.

IV. DISSERVICEPRO ARCHITECTURE

DisServicePro extends DisService with proactive capabilities and it is integrated in the Agile Computing middleware [2]. The Agile Computing middleware is a set of components designed to support communications and computation in tactical edge networks. DisService is the component dedicated to information dissemination. DisService opportunistically exploits available storage and communications to increase information availability in the network and to achieve disruption tolerant behavior.

As shown in Figure 1, DisServicePro architecture is built upon DisService and utilizes its services. The Information Store is the structure on each node that maintains any information that is retrieved by the node or pushed to the node. After being used by the application, information may be retained by the Information Store for future use by other nodes. It may also temporarily store information that it is transferring from one node to another, acting as a courier, even if the local user may not need the information. DisServicePro exchanges data when its content is considered suitable in the operative contexts. In particular, two modules are defined: Local Node Context and the Peer Node Context. The Local Node Context contains information relevant to the local user and his or her mission. The Peer Node Context contains the same information as the Local Node Context, but for remote peer nodes. The context of peer nodes is useful to proactively push/pull information to/from other nodes preemptively. In particular, the Information Push component manages the proactive push of information to other nodes using the peer node contexts and the metadata associated with the information.

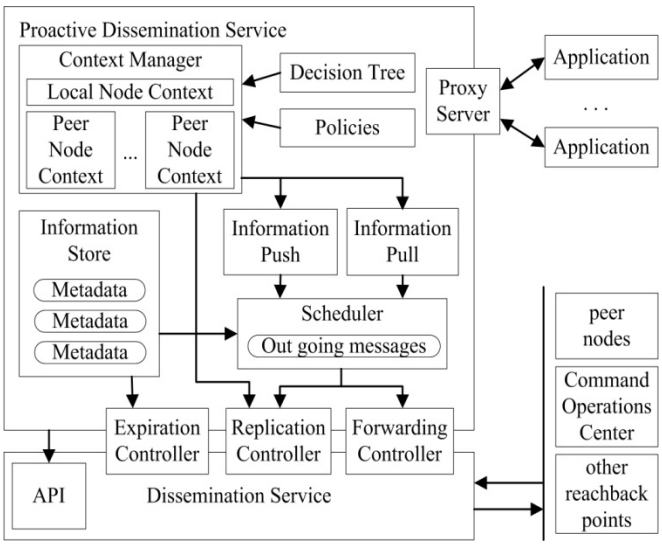

Figure 1: DisServicePro Architecure

On the other hand, the Information Pull component handles the proactive pull of information previously made available on the network. In addition, the Context Manager exchanges the local node context with peers and stores contexts received by other

nodes encountered during the mission. All the information exchanged by DisServicePro is dynamically prioritized by the Scheduler based on the user's context and preferences. The Scheduler handles a queue of outgoing data ordered by priority. The Expiration Controller manages the deletion of information stored in the local cache. Information is usually deleted when expired but other policies are possible. The Replication Controller uses the proactive pushing and pulling mechanisms to disseminate information; moreover, it permits the exchange of node context updates using special control messages. Finally, the Forwarding Controller defines policies to handle retransmission of data in cases in which the node acts as a courier for other nodes. Applications on the same node can communicate with the same instance of DisServicePro through the Proxy Server component. Applications written in C++, C#, and Java are supported.

V. DISSERVICEPRO CAPABILITIES

This section describes the important features of DisServicePro, which combine together to provide proactive information dissemination.

A. Adaptable Information Model based on the Specific Mission Context

In order to understand the users' interests and correctly anticipate their needs, DisServicePro collects information about both the information being exchanged in the network and the users' behavior. Modeling such information in an accurate, precise way is a key requirement for obtaining good performance: the better this model fits the characteristics of users and information, the more effective the predictions made by DisServicePro. On the other hand, operations in particular areas could potentially establish new types of information and user characteristics. For this reason, the DisServicePro information model is flexible and allows applications to customize and adapt it to their specific needs. The information model managed by DisServicePro is composed of two different parts: the *metadata*, a header attached to each discrete piece of information traveling in the network, and the *node context* which describes the user's tasks and mission. The metadata contains information about the content and the purpose of the information being sent. The default metadata has a few fixed attributes that model general characteristics shared by all information types. These attributes are: *Message ID*, *Receiver Timestamp*, *Source Timestamp*, *Source ID*, *Expiration Timestamp*, *Relevant Missions*, *Area of Interest*, *Location*, *Pedigree*, and *Importance*. Depending on the mission, these attributes can be extended with other relevant information. For instance, in all the tests presented in this paper, we have customized the metadata by adding these attributes: *Data Content*, *Data Format*, *Classification*, *Description*, and *Source Type*. Furthermore, the application can indicate which of the attributes should be utilized by the learning algorithm to infer the user's interests and which should not, providing a fine-tuning mechanism for the information model. On the other side, the *node context* models the characteristics of the users generally utilized in STOM and NEO operations. The attributes currently utilized in the node context are: *Node ID*, *Mission*, *Team*, *Role*, *Useful Distance*, *Paths*, *Current Location*, and *Preferences*. The *Preferences* attribute represents the node's interests that are automatically learned and *Paths* is a list of the node's possible projected paths. A typical path is a sequence of points on a map, called way-points, which can be GPS coordinates or can be calculated with azimuths and distances. Each point could be named or described with notes. Points may also have associated a date and time in order to better coordinate the movements of different units on the field. This information is particularly useful to predict the expected time to use of information, but is not always present.

B. Replication and Forwarding Mechanisms

As illustrated in Figure 1, DisService offers the possibility of customizing the behavior of its expiration, forwarding, and replication mechanisms by implementing appropriate controllers. DisServicePro extends DisService and implements its own version of these controllers. The Replication Controller has the objective of increasing the availability of the information in the network. Replication takes place any time the node encounters a new peer and any time new information is available at the node. Replication of data is performed using a combination of pull and push approaches. In particular, concepts of path and team are exploited, permitting the Replication Controller to find the right redundancy balance inside the team and among peers sharing the same path. The Forwarding Controller has the objective of improving the timely dissemination of the information in the network. Forwarding takes place any time new data is received. It employs policies that take into account both the push and pull algorithms, prioritization, as well as information about the network topology provided by DisService to decide whether to forward a data to a certain node.

C. Information Anticipation

The ultimate goal of DisServicePro is to minimize information access latency for users in tactical edge networks. The approach taken by DisServicePro is to anticipate information requests and pre-stage the information ahead of time, minimizing the impact of degraded or lost connectivity to the node. Anticipating relevant information is a challenging task, particularly in the case of distributed and heterogeneous systems. The relevant information is selected by ranking its metadata against a set of policies that were specifically designed to fit typical military mission requirements. Each policy ranks a specific aspect of the metadata and gives a measure of how much the information fits the policy requirements. The ranks generated by the single policies are then combined to obtain the information priority, as detailed in section V.G. The policies currently implemented in DisServicePro are the following:

Distance path-metadata: this policy ranks the metadata based on the distance between the geographical area of relevance of the information and the user's projected path.

Expected time-to-use: this policy tries to infer when the user will need the information in the future. The prediction is based on the timestamps that mark the way-points of the projected path. The policy computes the expected time to use of the information based on the timestamp of the waypoint closest to the area of interest of the informaion. Unfortunately, the timestamps are not

always available. In this case, DisServicePro is able to calculate the missing timestamps using distances between consecutive waypoints and ranks the information accordingly.

Expiration time: each piece of information has an associated expiration time that indicates for how long the information is expected to be useful. This policy assumes that newer data has higher utility than older data close to its expiration time, so newer data gets a higher rank. The policy calculates the percentage of remaining time with respect to the total expiration time duration. In this way, information with different expiration time durations can be compared.

Importance: applications that generate information can specify an explicit importance value for each data. A policy takes into account this value.

Although the sole use of these policies provides a rudimentary form of predictive capabilities, this may not be enough to produce acceptable performance in a dynamic environment. For this reason, DisServicePro exploits machine learning techniques that analyze the usage history of the local information in order to find patterns from which it extrapolates rules that match the user's interests. The machine learning problem is formulated as follow: each piece of information stored on the node has an associated *usage* value which indicates if the information has been utilized by the user or not. Given the usage history, the goal of machine learning is to minimize the error in predicting if new information will be utilized by the user or not. However, in tactical environments, maximizing the prediction accuracy is not the only performance metric and the learning algorithm must ensure other important characteristics. First, it must guarantee low resource consumption in terms of computation and storage in order to adapt to resource-constrained nodes such as small portable devices (PDAs). Another fundamental requirement is the capability to learn incrementally and to produce accurate predictions online and in real-time. Since there may be changes in the mission and orders, and consequent adjustments in planned activities and information requirements, the learning algorithm needs to follow the evolution of the user preferences during time and adapt quickly to these changes. Moreover, we assume that in most of the cases, training data will not be available prior to the mission. Finally, the learned interests model has to be compact and understandable in order to be easily shared with other peers improving the overall knowledge of the system. Given metadata as input, the goal of the machine learning algorithm is to classify the information in two categories: *useful* and *not useful*. The algorithm should also provide a score value indicating its confidence in the prediction. One of the best known machine learning algorithms to address this type of problem are decision trees. In particular, we found the C4.5 algorithm [8] to be well suited for our requirements. Two alternative algorithms have been implemented: *Cycle Algorithm* and *Window Algorithm*. Both of them are based on the C4.5 algorithm and have been designed to fit all the requirements listed above. The *Window Algorithm* is based on the *sliding window* principle. Training data is collected over time as entries are inserted in the training window one at a time. Once the data becomes too old, it is discarded, leaving space for newer data in the FIFO queue. This simple principle allows the algorithm to forget old data and learn from new data, following the evolution of the user's interests over time. Discarding old data also decreases the storage requirement. However, while building a decision tree, the size of the training dataset influences the prediction accuracy. Classical data mining approaches use very large datasets and do not suffer for small changes in the dataset size. But in DisServicePro, the training window is constrained in order to save resources. As a consequence, changes in the window size may have an impact on the performance. The ambitious goal of the *Cycle Algorithm* is to reduce the window size without losing prediction accuracy. We observed that not all the new data collected during a given time adds additional information to the decision tree. The idea behind the *Cycle Algorithm* is to test the current decision tree on the new data for each iteration and discard correctly classified data. On the other hand, the misclassified data is added to the training window providing a feedback that helps the decision tree to correct past errors.

Both the algorithms output their confidence value for each prediction. These values are used to rank the metadata, which is then combined with the ranks generated by the other policies.

D. Sharing the Learned Interests with Other Peers

In order to improve their predictive capabilities, the peers share both their local node context and the knowledge acquired by the learning algorithm. Sending the history of previously requested information that represents the user's interest would be too expensive, hence only the rule sets generated by the learning algorithm are exchanged. Since maintaining the continuously updated learned interests is crucial for the overall system performance, DisServicePro implements a custom knowledge exchange protocol that has been designed to be mainly event-driven. The protocol also exploits periodic updates to improve reliability. In order to save bandwidth, DisServicePro exchanges mostly small update messages that contain a portion of the node's interests. The user's interests are split into three sections, each of them identified by a version number. When one or more sections change locally at the user node, the relative updates are immediately propagated in the network. For example, the most common and frequent update is the user's GPS position. Therefore, small messages containing the current versions and the node GPS position are periodically broadcasted. This policy improves reliability because if an update message gets lost, the peers can request it again once they receive the periodic message. Moreover, some nodes, for instance ground sensors, do not follow a path, but they stay fixed on the ground. For these nodes, sending periodic messages is convenient to make the other peers aware of their existence, since they generally do not change their interests over time. The whole node context needs to be exchanged only when two nodes interact for the first time. The next time they come into contact, the peers first exchange the latest version numbers of each section that has previously received from the other node and then, if necessary, exchange any update sections.

The proactive pull, the proactive push, and the users' interest exchange protocols are completely independent of

each other. Thanks to this design, it is not necessary to update the interests while pulling or pushing the data, saving time and bandwidth in both cases.

E. Proactive Pushing of Information

In tactical wireless environments, nodes experience frequent disconnections and changes in network topology. In such situations, nodes do not have a clear and updated understanding of neither the information needs nor the information content of other nodes. For this reason, information anticipation happens in a distributed manner. Each peer takes advantage of DisServicePro predictive capabilities to infer its own information needs and pull the data ahead of time. At the same time, thanks to the distribution of the node contexts and the learned interests in the network, the nodes are able to anticipate the other peers' needs and proactively push useful information to them. Each peer actively seeks both the information locally generated and the data collected from other nodes to automatically infer when the information should be replicated or forwarded to other nodes based on their interests. The decision to proactively push or not to push information to a target node is based mainly on the target node's learned interests, but also the interests of nodes with similar characteristics (for instance, nodes with the same role or nodes coming from areas close to the target node's projected path) are taken into account. The proactive push takes place also when some nodes receive updated interest information from a peer. For instance, a possible scenario is when, due to a change in the mission, a node switches from its main projected path to an alternative path. In this case, the peer node has partially changed its node context meaning that now it is probably interested in different types of information.

The proactive push mechanism is completely event-driven and it has been designed to react immediately to any change in the environment. It also ensures the minimum delay in obtaining the needed information because each node replicates and forwards its data as soon as it becomes available without wasting time. These characteristics make the proactive push particularly fast and allow nodes to take advantage of resource-rich peers, even for very short periods of time.

F. Proactive Pulling of Information

Information anticipation happens also at the end-user nodes. Each node anticipates information the user may need based on the learned interests and proactively pulls the information ahead of time. The proactive pull is particularly appropriate to disseminate large or otherwise problematic information. This information is made available to other peers by disseminating its associated metadata, while the information itself remains on the source node. Each node that receives the metadata may decide whether to pull the data ahead of time or not. The decision is based on both the user node interests and also similar nodes' interests as in the push approach. If the user node decides to proactively pull the data, it sends a direct request to the source node, which answers with the desired information.

The proactive pull has an event-driven design, allowing it to immediately react not only to new metadata disseminated in the network, but also to any change in the user node's interests. When a drift in the user's interests is detected, DisServicePro searches the local cache and analyzes the metadata recently received to check whether the associated information has now become of interest to the user.

In DisServicePro, the push and pull mechanisms disseminate different types of information. As a consequence, they are independent of each other and they can apply customized policies for large and small data. For instance, in scenarios where the available bandwidth is highly constrained it may happen that there is not enough time to send all the interesting information. In such scenarios, the proactive pull may decide to focus on high priority information and to not send information with lower priority in order to save bandwidth, while the proactive push, in the same scenario, may decide to wait and try to send the low priority information after a while.

The pull component also handles specific search requests provided by the user. Users can express search queries over the metadata content, which are evaluated both locally and across the other nodes in the network.

G. Dynamic Information Prioritization and Scheduling

As described earlier, tactical networks are constrained in terms of connectivity, bandwidth, and latency. This implies that there may not be sufficient bandwidth to push or pull all the information that DisServicePro predicts as useful for the target nodes. DisServicePro addresses this challenge by dynamically prioritizing all the information being sent both to a user as well as across users. The prioritization takes into account the results given by all the policies and the decision tree predictions described in section V.C. These results are not binary values, but continuous values effectively ranking information based on the different aspects evaluated. These ranks are combined together through a weighted sum to obtain the priority value. The prioritization is dynamic, meaning that the information priority is determined every time the push or pull mechanism decides to send it. As a consequence, the same information might assume different priorities depending on which node is the target, and the current situation under which the data is sent. For instance, when information becomes available on a node, DisServicePro may decide to push it to three different targets if it predicts that the information is interesting for all the three nodes. As a consequence, it calculates three different priorities for the same information, one for each target node. If the information is more important or urgent for a certain node, this node has assigned the highest priority and it is served first. Another example is when a new peer connects to the network. In this case, DisServicePro may decide to push to that node more than one piece of information. Each piece is then prioritized and the one with the highest priority is sent first. As these examples illustrate, it can often happen that more than one piece of information needs to be sent at the same time. This data is handled by the DisServicePro scheduler, which orders the information to be sent by priority, and arranges it in an output queue. In the current implementation the policies utilized by DisServicePro are fixed. The prioritization process could be further improved by using policies dynamically generated at runtime. For instance, a policy may decide that soldiers being part of the same team have higher priority than soldiers of other teams, or that soldiers who are in contact with

and currently engaging the enemy have higher priority. The runtime policies may specify the relative weights of these conditions thereby ensuring fine-grained control.

VI. DisServicePro Performance Evaluation

This section presents some initial performance results of DisServicePro. Two sets of experiments were performed. The first set evaluated the performance of the pre-staging algorithm, without taking into account any learning. The second set evaluated the learning aspects of DisServicePro.

A. Pre-Staging Evaluation

The purpose of the pre-staging evaluation is to quantify the benefit, in terms of access time, of the proactive approach over a purely on-demand approach. The test simulated a simplified NEO mission with two nodes: a source node and a target node. The source node represents a COC (Command Operations Center); it stays in a fixed location and is pre-loaded with information items, each being approximately 10 KB in size. The target node represents a team in the mission situation: it moves along a path and periodically requests new information about the area surrounding its current location. The path is modeled as a sequence of 20 way-points; every 30 seconds, the target node reaches a subsequent way-point, updates DisServicePro with the coordinates of the current position, and sends out a request for all the information of interest within an 800m radius of the current position. Note that the timeline was accelerated only to speedup execution time. Upon receiving the request from the target node, the source node replies with all the information that matches the request that have not been already sent. The average delivery-time to retrieve an object that matches a request, the number of objects retrieved over the duration of the mission, and the percentage of hits and misses are measured. To expedite testing, the interval of time between the 20 way-points (30 seconds) and the items of information requested at each way point (approximately 12) were proportionally reduced by a factor of 10. The size (10 KB) was chosen for the information because this size is representative of the average data size typically transmitted in a tactical network environment. In order to simulate a realistic environment and not to unfairly stack the pre-staging test, a path with partially overlapping areas is used. In this way, even without pre-staging, it is possible for the target nodes to find information of interest for the current area that are locally cached due to a previous retrieval. The following bandwidth settings were used with both pre-staging and non-pre-staging approaches: 56 Kbps, 256 Kbps, and 512 Kbps. The 56 Kbps link is representative of the typical bandwidth for a portable SATCOM link, whereas 256 Kbps is the typical bandwidth for a UAV link and 512 Kbps is the typical bandwidth for high-capacity SATCOM link. The metadata was generated randomly. The list the possible values that the attributes can assume are as follows:

Source:	node_A, node_B.
Expiration_Time:	300000, 400000.
Relevant_Missions:	mission_X, mission_Y.
Left_Upper_Latitude:	*continuous values within the area*.
Left_Upper_Longitude:	*same as above*.
Right_Lower_Latitude:	*same as above*.
Right_Lower_Longitude:	*same as above*.
Location:	area_A, area_B, area_C, area_D, area_E.
Data_Content:	Map, Sensor_Data, Squad_Report, COC_Report, UAV_Image.
Data_Format:	JPG, MPG, XML, TXT.
Classification:	Unclassified, FOUO, ITAR, Secret, Top_Secret.
Description:	message_type1, message_type2.
Node_Type:	Video_Sensor, Acustic_Sensor, Sismic_Sensor, UAV, COC, Marine_Squad.
Importance:	1, 2, 3, 4, 5.

The receiver is interested only in the subset that matches the following ruleset:

NodeType ε {COC, Marine_Squad, UAV} AND
Data_Content ε {Squad_Report, COC_Report, UAV_Picture} AND
Classification ε {FOUO, ITAR, Unclassified, Secret}

OR

NodeType ε {Video_Sensor, Acustic_Sensor, Sismic_Sensor} AND
Data_Content ε {Sensor_Data} AND
Data_Format ε {MPG, JPG, XML}

OR

NodeType ε {COC, UAV} AND
Data_Content ε {Map, UAV_Picture, Sensor_Data}

To summarize, the test parameters were as follows:

Path:	composed of 20 way points
Request Interval:	30 seconds
Radius:	800m
Payload Size:	10KB
Number of Messages:	1000 (for the 512 Kbps and 256 Kbps cases) 100 (fro the 56 Kbps case)
Link Connectivity:	56 Kbps, 256 Kbps, 512 Kbps

The results shown below indicate a significant improvement in the information retrieval latency. In addition to the average latency, we measured the number of hits, the number of misses, and the number of false positives. False positives occur when DisServicePro incorrectly predicts and pushes information that is never requested by the end user node. In the first set of results, without pre-staging, there are no false positives. Also, there are some hits even without pre-staging because the end user node may request an item that it has already requested in the past. The results show that with pre-staging, the average latency is improved by a factor of 6.28, 4.12, and 6.75 times respectively, for the three different links of 56 Kbps, 256 Kbps, and 512 Kbps.

Table 1: Resuts Without Pre-Staging

Link Bandwidth	Average Latency (msec)	STD deviation	#hits	#Misses	#False Positives
56 Kbps	8726	16857	15	25	0
256 Kbps	11344	17914	57	123	0
512 Kbps	10131	17421	57	123	0

Table2: Results With Pre-Staging

Link Bandwidth	Average Latency (msec)	STD deviation	#Hits	#Misses	#False Positives
56 Kbps	1389	4666	36	6	26
256 Kbps	2749	8296	307	43	218
512 Kbps	1502	5360	332	30	218

Note that with pre-staging, there are also many more requests overall, given that no time is spent retrieving information when there is a hit.

B. Learning Evaluation

In the learning evaluation the two iterative algorithms have been compared against each other and against other algorithms. All the algorithms were tested on the *Adult* dataset obtained from the UCI Repository of Machine Learning databases [12]. This dataset is a collection of census information of people from different nations, such as sex, marital status, age, occupation etc. The task consists of classifying the set in people whose annual income is higher of 50000$ or lower.

Algorithm	Error Rate
Cycle Algorithm	13.98%
Window Algorithm	14.18%
C4.5 Original	15.54%
C4.5 Rules	14.94%
Voted ID3 (0.6)	15.64%
Voted ID3 (0.8)	16.47%
NBTree	14.10%
FSS Naïve Bayes	14.05%
IDMT (Decision Table)	14.46%
Naïve-Bayes	16.12%
Nearest-Neighbor (1)	21.42%
Nearest-Neighbor (3)	20.35%

The results obtained with *Cycle Algorithm* and *Window Algorithm* refer to the algorithms run with the settings as follow: for the *C45 Cycle Algorithm*:

initial window: 200 items
maximum window: 1200 items
increment: 200 items
maximum errors: 20% of increment
initial cycle window: 100% of actual window

For the *C45 Window Algorithm*:

initial window: 200 items
maximum window: 270 items
increment: 70 items

These results, while promising, are preliminary and have not yet been integrated into the overall DisServicePro application.

The results of error rate for the other algorithms are publicly available on the UCI Repository website.

VII. CONCLUSIONS AND FUTURE WORK

This paper presented DisServicePro, a proactive peer-to-peer dissemination service for tactical network environments that supports information on-demand. The architecture, motivational scenarios and initial experimental results are presented. The experimental results show the benefits of the proactive approach, in terms of average delivery time and the potential of the implemented learning algorithms. Future work will attempt to exploit the network characteristics to efficiently disseminate information in large MANETs. Topology state information and bandwidth prediction algorithms are going to be integrated in the system to address the current scalability issues. Moreover, DisServicePro approach is currently being compared to other recently proposed algorithms, including content-based and sequence-based pre-fetch prediction algorithms, to determine which maximizes the tradeoff between cache hit rate and bandwidth utilization.

ACKNOWLEDGEMENTS

This research has been sponsored by the Office of Naval Research under grant N000140910012.

REFERENCES

[1] N. Suri, G. Benincasa, S. Choy, S. Formaggi, M. Gilioli, M. Interlandi, J. Kovach, S. Rota, and R. Winkler, "Disservice: a peer-to-peer disruption tolerant dissemination service", MilCom, Boston, MA, 2009.

[2] N. Suri, M. Marcon, R. Quitadamo, M. Rebeschini, M. Arguedas, S. Stabellini, M. Tortonesi, C. Stefanelli, "An adaptive and efficient peer-to-peer service-oriented architecture for MANET environments with agile computing", in proceeding of the second IEEE Workshopon Autonominc Computing and Network Management, 2008.

[3] W. Hsu, D. Dutta, and A. Helmy, "Csi: a paradigm for behavior-oriented delivery services in mobile human networks" CoRR abs/0807.1153, 2008.

[4] J. Hoffert, D. Mack, and D. Schmidt, "Using machine learning to maintain pub/sub system qos in dynamic environments", 8th Workshop on Adaptive and Reflective Middleware, 2009.

[5] B. Xu, O. Wolfson, and C. Naiman, "Machine learning in disruption tolerant manets", TAAS 4, 2009.

[6] G. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection", Brazilian Symposium of Artificial Intelligence, 2004.

[7] O. Wolfson, B. Xu, and R. M. Tanner, "Mobile peer-to-peer data dissemination with resource constraints", MDM, 2007, pp. 16-23.

[8] R. Quinlan, "C45: programs for machine learning", Morgan Kaufmann Publishers, Inc., 1993.

[9] W. Zhang, D. B. Lewanda, C. D. Janneck, and B. D. Davison, "Personalized web prefetching in Mozilla", 2003, unpublished.

[10] M. Nunez, "Collective learning of knowledge for predicting events", Universidad de Malaga, Spain, unpublished.

[11] M. Karnick, M. D. Muhlbaier, and R. Polikar, "Incremental learning in non-stationary environments with concept drifts using a multiple classifier based approach", International Conference on Pattern Recognition, Glassboro, U.S., 2008.

[12] C. Blake, E. Keogh, and C. J. Merz, UCI Repository of Machine Learning databases, online reference: http://archive.ics.uci.edu/ml/