Middleware for Supporting Content Sharing in Dynamic Networks

Mohan Kumar[1], Sharma Chakravarthy[1], Sanjay Madria[2], Mark Linderman[3] and Waseem Naqvi[4]
[1]University of Texas-Arlington, TX, [2]Missouri University of Science and Technology, MO; [3]Air Force Research Lab, Rome, NY; and [4]Raytheon Corporation, MA

Abstract This paper proposes a middleware architecture to enable seamless content sharing in highly dynamic networks, such as those involving a number of aerial unmanned autonomous vehicles (UAVs), which are characterized by constantly changing connectivity and network topology. The proposed middleware maintains up-to-date network status as well as metadata information about data objects in the network nodes, masking the content sharing applications from the network dynamics and faults. Mechanisms for node discovery, message passing and network management are developed. A strategy is also proposed for efficient replication to ensure availability of data in highly mobile environments. The middleware includes a message acknowledgement strategy that can be used on top of UDP for reliable communication in volatile wireless environments. Furthermore, the middleware supports database queries in changing networks. In the future, proposed middleware will be utilized for fault-tolerant database query processing.

1. INTRODUCTION

Aerial unmanned autonomous vehicles (UAVs) are mobile, and they can communicate with each other wirelessly if they are within communication range of each other. The bandwidth for wireless communication between the nodes is a function of the physical distance, the connectivity of each node, and their velocities. As the connectivity between any pair of UAVs or nodes is dynamic, it is essentially a mobile ad hoc network with highly mobile nodes. UAVs employed to perform certain tasks in surveillance, reconnaissance etc., are required to share content (e.g., pictures) gathered by each other, query for data and communicate fused information and decisions. In order to ensure seamless content sharing and accessibility, it is imperative to have middleware support. In this paper, we propose the architecture of an efficient middleware for a mobile ad hoc network of UAVs to enable seamless data sharing, accessibility and query processing.

Approved for public release; distribution unlimited (Case Number: 88ABW-2011-2947)

The middleware includes modules for: network configuration; message passing; metadata maintenance and replication management.

Various services that can be provided by the middleware include, publish/subscribe, fault-tolerance, efficient query processing and others.

2. RELATED WORK

Several middleware architectures have been developed in the recent past to support mobile ad hoc networks (MANETs) [1], sensor networks [5] and pervasive systems [3,4]. Boulkenafed and Issarny [1] develop a comprehensive middleware for data sharing in MANETs. The focus of the work however is minimizing energy consumption. Kalasapur et al.[3] developed an elegant middleware for service provisioning in pervasive systems with mobile nodes. Kumar and Tamhane [4] have developed a resource management mechanism for pervasive systems with underlying ad hoc networks. None of these works consider dynamic networks such as that of UAVs, where node mobility is a regular feature rather than a rarity. Christman and Johnson [2] discuss a customized self configuring architecture designed for UAVs. However, they do not deal with on content sharing and query processing. In contrast, the proposed middleware is an attempt to address this important issue in UAV based networks.

3. ARCHITECTURE

Figure 1 illustrates components of the proposed middleware. The middleware on each node is also responsible for publishing the capabilities and important contents (or metadata of the contents) as well as subscribing (for information from other nodes) to meet the needs of that node. This middleware on each node supports services as well as interacts with the middleware in other nodes in solving a task collaboratively. As device-to-device connectivity is intermittent, one of the major challenges is to ensure availability and accessibility of data in a continuous fashion. In particular, the middleware collects information proactively, processes information, handles subscriptions, responds to continuous queries, and coordinates collaborative computations. The challenge is to overcome communication dynamics to create a system where information content is available and accessible at all the nodes in the system. It is important to provide a uniform view to all nodes and facilitate query processing effectively and efficiently in constantly changing networks. Furthermore, seamless data access and query processing should be facilitated in the presence of dynamicities associated with the data objects A middleware service framework [3] developed for Seamless Service Composition (SeSCo) in dynamic pervasive computing environment is adopted here for provisioning middleware services.

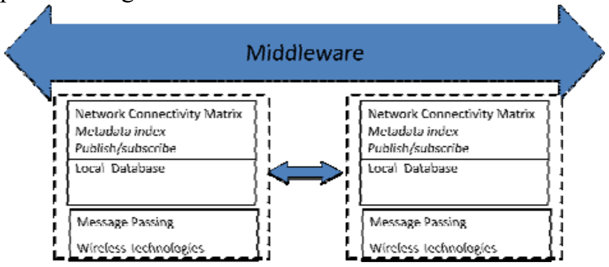
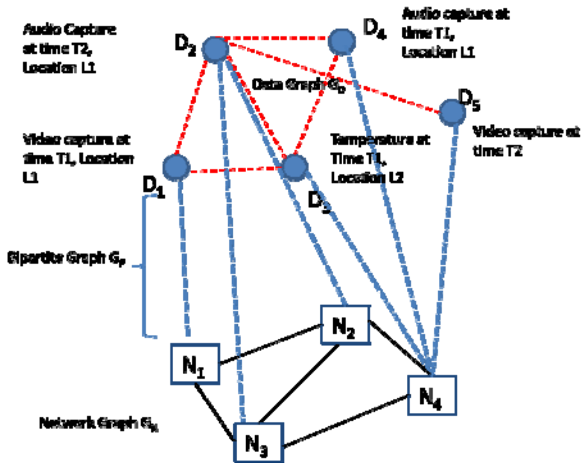


Figure 1. Middleware Architecture

SeSCo is not only flexible in defining and generating services/tasks, but it can also be modified to handle the type of dynamic ad-hoc network we expect to encounter in this proposal. The service provisioning mechanism captures the essence of devices and services around users, and facilitates the creation, composition and deployment of services dynamically. At the core of our current approach is the use of graph theoretic and sub-graph matching techniques to ensure network status awareness and data access. In the proposed middleware, a graph structure will be created to capture the essence of data objects/services, corresponding computing nodes and the relationship among the data objects as well as the nodes. The associated middleware tools facilitate the response to queries in dynamic heterogeneous environment comprising mobile (UAVs) nodes. The proposed service provisioning framework is flexible in representing metadata and services, and adaptive to changing environments. The proposed strategy is described below. All the data objects present within a region, and their attributes are captured in the graph G_D. Each data object (or MIO) is described as a vertex D_i, for $i \leq k$ in the graph, G_D. The link between two vertices D_i and D_j within the graph G_D, represents the relationship between the two data objects. In general the data graph described by $G_D = (D,R,\delta,\rho)$, represents the set of data objects, where D represents the data set (e.g., $D_1,D_2,\ \ D_k$). The edge set R represents the relationship between the data objects. The vertex attribute set δ represents the attributes of the data- how, where, and when the data object was created, its TTL and the physical node on which it is stored. The edge attribute set denoted by ρ, represents the relationship between two data objects - e.g., group, temporal and spatial relationships etc. In the Figure 2, data objects D_1, D_2, D_3, D_4, D_5 and their inter-relationships are shown in the graph G_D at the top. Figure 2 depicts a scenario where data objects correspond to different media objects of an *event* acquired by sensors distributed in space and time. For example, D_2 is the audio data of an event at time T_2, D_5 is the video data of the same event at the same time T_2, while D_4 is the video capture of the same event at time T_1. D_2 and D_4 may correspond to, for example, the video data of the same event at time T_2 but from two different locations in space. Thus all the data objects are related to each other as illustrated by G_D. To avoid ambiguity, we refer to the nodes of the data graph as *vertices* and the nodes of the network graph (below) as *nodes*. In reality, the data graph is likely to be much larger, we consider a small graph for simplicity. The characteristics of the network of physical nodes (or UAVs) and their connectivity are maintained in another graph G_N. In general the network graph $G_N = (N,C,\eta,\xi)$, represents the network of physical nodes (or UAVs), where N represents the node set (e.g., $N_1,N_2,...N_n$). The set C represents the connectivity among nodes. The vertex attribute set represents the attributes of the node – data objects, residual battery energy, and other features. The edge attribute set denoted by ξ represents the channel characteristics of the wireless link between two nodes – e.g., type of link (Wi-Fi, satellite, Bluetooth etc.) bandwidth, latency etc. In Figure 3, nodes N_1,N_2,N_3,N_4 and their interconnectivities are shown in the graph G_N at the bottom. The edge attributes show the type of connection and other features. For example, nodes N_3 and N_4 may have a 1 Mbps Wi-Fi link. Other connection attributes are not shown in the figure for the sake of clarity. The Figure shows a network of 4 nodes. In reality there are likely to be more nodes, and nodes leave or join the network constantly. Maintaining the connectivity graph in such dynamic situations and propagating the updates will require adoption of ad hoc networking strategies.

A logical bipartite graph is created between the data objects and the physical nodes. In other words, the data-to-node bipartite graph, G_P connects a vertex D_i of G_D to a node N_j of G_N. In Figure 2 the graph in the middle, GP comprising the edges from data objects to the physical nodes is G_P. The edge set E_P of G_P represents location of each data object. In effect G_P also shows the data availability and accessibility options and incorporates replication information. In the proposed service provisioning mechanism all the three graphs will be available at the command centers. Data updates and invalidations will be propagated from the nodes (UAVs) to the command center at predetermined intervals of time. We have developed algorithms to establish paths, modify and repair the graphs efficiently when nodes (UAVs) move around in the environment, and to assimilate new nodes and their data services seamlessly. Other features of the middleware include:



Audio Capture at time T2, Location L1
D_2
D_4
Audio capture at time T1, Location L1
Data Graph G_D
D_5
Video capture at time T2
Video capture at time T1, Location L1
Temperature at Time T1, Location L2
D_1
D_3
Bipartite Graph G_P
N_2
N_1
N_4
Network Graph G_N
N_3

Figure 2. Data management scheme

i) Creation and maintenance of data graphs, node graphs, and the data-to-node bipartite graphs;
ii) Efficient and seamless methods for modifying graphs;
iii) Proactive computations for such tasks as *join*, *fuse*, and others; and
iv) Integration of publish/subscribe data delivery options.

The second task involves graph modifications when the following events take place - data gets updated, invalidated or moves, nodes leave or join the network, and the command center changes preferences and priorities. We have developed algorithms for finding multiple paths in a graph that is constantly changing. An edge is the link between two nodes; for example, in Figure 1, $N_1 \rightarrow N_2$ is an edge between nodes N_1 and N_2. A path is a series of edges in the network. For example, in Figure 1, $N_1 \rightarrow N_2 \rightarrow N_4$ is a path for transmitting messages from node N_1 and N_4. Secure and speedy access to data and connectivity between two devices that carry data objects to be fused is a challenge in mobile and uncertain environments. Therefore, it is important to have multiple paths between pairs of nodes. In the above example, $N_1 \rightarrow N_2 \rightarrow N_4$ and $N_1 \rightarrow N_3 \rightarrow N_4$ are two parallel paths between N_1 and N_4.

Message format for different types of messages is shown below. A message carries data and control bits from one node to another. Source ID and destination ID are 5-bit unique addresses of the corresponding nodes (UAVs). Multiple flags are used to indicate the type of data in the payload.

SP	SP	D	N	M	Source ID	Destination ID	Timestamp	Payload

Figure 1. Message Format

Initially, each node is by itself and not connected to other nodes in the network. At time T=T1 (say) when another node comes within communication range, a neighbor discovery message DISCOVER is sent by one or both the nodes. The DISCOVER message comprises, the source identifier (ID), the destination ID (typically a Broadcast address), a timestamp, meta data information, and flag **D** =1, indicating that the payload contains Meta data information. Each node receiving the DISCOVER message, responds with a message. The NEIGHBOR message contains, the source identifier (ID), the destination ID is (typically a Broadcast address), a timestamp, flag bit **N** =1, indicating that the payload contains information about the node's connectivity as well as metadata information, and flag bit **I** =1, indicating that the payload contains a node's intended direction of travel.

Flag bit	**Payload**	**Description**
D-Discover	Meta data	Discover new nodes
N-Neighbor	Meta data Adjacency Matrix	Connectivity info.
I-Direction	Node direction	Travel direction
M-Data	Result of query execution	01: MOVE 11:COPY 10: Intermediate
A-Alive	None	House keeping
Q-Quit	None	Known mobility

Table 1: Interpretation of flags

A series of DISCOVER/NEIGHBOR messages among nodes within communication range results in an ad hoc network comprising connected nodes (UAVs). Similar DISCOVER/NEIGHBOR messages are exchanged periodically either to gain new or retain existing connections. At the end of this process, each node in the ad hoc network has information about connectivity as well as metadata of each of its neighboring nodes. Each node maintains its connectivity with other nodes in the connectivity graph G_N, and exchanges connectivity information. It is likely that there are one or more such ad hoc networks and isolated nodes in the system at any given time.

Data exchange between nodes: Consider a snap shot of the system as shown in Figure 3. Let N_1, N_2, N_3 be three nodes in the physical network. Data items (relations) *R1* and *R2, R3* are represented as data nodes and reside on nodes N_1, N_2 and N_3 respectively.

Each physical node in the network exchanges information about what Relational data it contains as a part of meta data exchange which is included in the DISCOVER message. Hence node N_1 knows that nodes

N_2 and N_3 contain data items R_2 and R_3 respectively. Similarly the other two nodes get to know that node N_1 contains data item D_1.

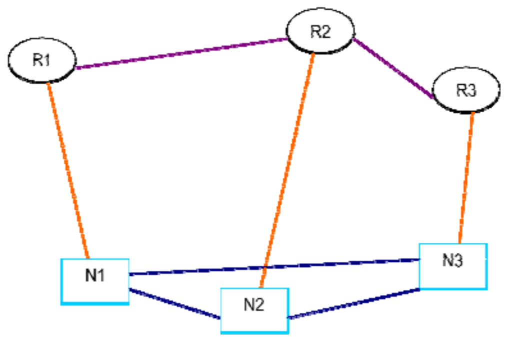


Figure 3. Network Snapshot

When a query arrives at node N_1, a query plan is generated based on the current connectivity of the network. The generated plan contains operations that need to be executed at various nodes with the query execution beginning at N_1. Operations local to the node N_1 are carried out first and when the first MOVE command is executed, the results of the partial query execution needs to be transferred to the next physical node in the plan.

The flag **M** is set to indicate that the payload contains: Result of the local operations at node N_1; The query plan; and a counter value to indicate that the next node in sequence needs to execute the query starting at the instruction pointed to by the counter value.

Topology Maintenance: To indicate its presence in the network, each node sends periodic ALIVE message to those not connected to it. This message is essentially a multicast message. The flag **A** is set to indicate that the message is an ALIVE message. A node's absence is determined due to the missing ALIVE message and changes are made in the connectivity graph of each of the neighbors. Also, a node can voluntarily leave a network by sending a QUIT message (also multicast) and hence changes are made in the connectivity graph of each of the neighbors. The flag Q is set to indicate that the node is voluntarily leaving the network.

To address fault-tolerance, the middleware exploits the graph representation and adoption of existing graph algorithms. First, the bipartite graph G_P allows finding of alternative sources of data objects. Second, we have developed algorithms for pre-computation of paths, alternative paths, minimum cost spanning trees with and without specific edges and nodes. Pre-computed values can be used in responding to queries in the presence of time constraints. These algorithms are not shown in the paper due to limited space. The following steps describe the actions when a query is made at a command center node:

1. a task (e.g., fuse audio and video data at time T1) is generated to address the query;
2. the data graph G_D is processed to find the corresponding data objects (*D*2 and *D*5 in the example);
3. the data-to-node bipartite graph is scanned to find the appropriate physical nodes that hold the data objects (node N2 in the example); and
4. the network graph is explored to find the network paths if needed and the corresponding nodes are contacted for the required data objects.

In the above scenario, suppose node *N*4 moves away (or goes to sleep), the graph structure shows the required data objects can be retrieved from other nodes (or nodes) , as indicated in Figure 2.

4. REPLICATION

In order to ensure accessibility and fault-tolerance, each data object is replicated on one other node. In the following we discuss the criteria for choosing an appropriate node for replication. Eventually, there exists only one replica of a given data item. However, more than one candidate nodes will be considered as a possible replica. The cost function described in the latter part of this section is used to choose the best node for replicating data. Suppose, $\boldsymbol{N_s}$ represents the source node, where the original copy of data item $\boldsymbol{D_i}$ was acquired and $\boldsymbol{N_c}$ represents the candidate node that will contain a replica of data object $\boldsymbol{D_i}$.

When $\boldsymbol{N_s}$ decides to replicate its contents on another node $\boldsymbol{N_c}$, a node from the set of the nodes that are immediate neighbors of the source node is selected as candidate node for replication. Immediate neighbors are those nodes which are directly connected to the source node. The source node tries as much to replicate all its tuples on the chosen candidate node.

For each of the above selected candidate nodes, a cost function $\boldsymbol{C_{s,c}}$ is computed. The node with the lowest cost is selected as a candidate for replication.

The cost function to determine the candidate node for replication is dependent on the following factors:

<u>Bandwidth:</u> Defines the closeness of $\boldsymbol{N_c}$ from $\boldsymbol{N_s}$ in terms of bandwidth. Greater bandwidth is desirable.

<u>Link stability</u>: This is a measure of stability of the link between nodes $\boldsymbol{N_s}$ and $\boldsymbol{N_c}$. Greater stability of the link between the two nodes implies better longevity.

The link stability $LS_{s,c}$ measured at N_s, is equal to rate of change of received signal strength for messages sent by N_c. Let r_1 and r_2 be the received signal strength at time t_1 and t_2 respectively. ($t_2 > t_1$). Current Link stability between N_s and N_c, is given by:

$LS_{s,c} = d_r/d_t$, where $d_r = r_2 - r_1$ and $d_t = t_2 - t_1$.

If $r_2 > r_1$ then $\boldsymbol{LS_{s,c}}$ is positive, this indicates increasing link stability between $\boldsymbol{N_s}$ and $\boldsymbol{N_c}$. If $\boldsymbol{r_2 < r_1}$ then $\boldsymbol{LS_{s,c}}$ is negative, this indicates decreasing link stability between $\boldsymbol{N_s}$ and $\boldsymbol{N_c}$.

Degree of the node ***Nc***: Greater the degree of a node, better is the accessibility of replicated data.

Cost function: When the source node $\boldsymbol{N_s}$ decides to replicate its data item $\boldsymbol{D_i}$ on a candidate node $\boldsymbol{N_c}$, it takes into account the recent history (from the last replication decision) to make the replication decision. History includes information about the above mentioned three factors. A table is maintained to store the average values for each node the source node is connected to. Each row in the table can be considered as a vector of values. Each element in this vector is normalized and has a value between 0 and 1. For each candidate node, a table is maintained to store the RSS values recorded in time. At any point in time, the link stability can be determined using 2 consecutive RSS values from the table ($t_1<t_2<t_i$) as shown in Table2. Suppose the current value of bandwidth between nodes N_1 and N_2 is 500 Kpbs and the maximum value of bandwidth is 1Mbps. The normalized value of bandwidth is 500Kbps/1Mbps, which is equal to 0.5. Similarly, normalization is applied to degree by dividing the actual degree of connectivity of the node with the maximum.

For link stability, normalization is applied as follows:
If the node ***Ns*** decides to replicate its data at say time t_i, it takes into account the current value of RSS r_i (at t_i)and the previous value r_i-*1*(at t_i-*1*)**.** The current link stability of the link between $\boldsymbol{N_s}$ and $\boldsymbol{N_c}$ is equal to the normalized value of rate of change of RSS. Let '***T***' be the time period between the last replication decision and the current replication decision. The maximum value of rate of change of RSS during '***T***' is computed and used in the normalization.

$$LS'_{s,c} = \frac{dr/dt}{Max(dr/dt)}$$

The above value lies in the range [-1, 1].
Normalized link stability is computed as $(LS'_{s,c}+\alpha)/\beta$ where α=1 and β=2.The normalized value has a range [0,1].

After each parameter is normalized, a weight is associated with it. The weight is an indication of the relative importance of a parameter with respect to other parameters in the computation of the cost function. Let w_b, w_l and w_d be the weights assigned for average bandwidth, link stability and average degree respectively.
Hence the cost function $\boldsymbol{C_{s,c}}$ to identify the most suitable node for replication is given by,

Cs,c =1/(w_b*average bandwidth + w_l link stability + w_d * average degree).
Also, $w_b + w_l + w_d = 1$.
A candidate node with a lower value of $\boldsymbol{C_{s,c}}$ is desirable for replication.
The network in this scenario is dynamic and the connectivity between nodes (physical nodes) keeps changing very often. The above chosen parameters in the cost function serve as a guiding factor to select a replica.

Bandwidth: During the process of replication, depending on how much data the source node has acquired, the amount of data that needs to be transferred (during a replication period) to the candidate node can be significant. Also, taking into consideration the dynamic nature of the network, delays in transmission can lead to some data not being transferred to the candidate node. If the candidate node is queried (instead of the source node) the result of the query may not be accurate due to lack of data necessary for carrying out the execution of the query. Bandwidth plays an important role in transmission delay, since greater the bandwidth more data can be transmitted on the link in lesser time. If the bandwidth between a source node and a candidate is high, data can be replicated quickly.

Degree: The degree of a node is the number of links incident at the node. If the source node of a data item goes out of range (isolated) for a long period of time, any query involving data on source node cannot be successfully executed. If a node '***Nc***' is connected to several other nodes, data on '***Nc***' will be available to other connected (direct or indirect) nodes for a longer period of time. It is essential that the replica of a data item (tuples) be accessible when the actual source of data goes out of range. If the candidate node has a low degree then the chances of it going out of range is greater. Hence degree of a node plays an important role in the availability of data.

Link Stability: Link stability is a measure of a source node's connectivity with the candidate node. Greater the link stability between 2 nodes, the more stable is the connectivity between the two nodes. A source node '***Ns***' always replicates its data items (tuples) on a single candidate node '***Nc***'. It is important to have stable connectivity between '***Ns***' and '***Nc***'. An unstable link leads to incomplete or inaccurate replication. If the link stability between 2 nodes is high, the source node gets more time to replicate its data. When a query hits a candidate node, it will have the most recent data (since the source node can continuously transmit the recently acquired tuples) required for successful query execution.

In Figure 4 is shown a sample network comprising 14 nodes. Node N1 is reference node and the problem is to identify a suitable node for replicating N1's data. In the sample network, available bandwidth for any node is B (1 Mbps), which is divided equally among the number of links at that node. The RSS values at node N1 when communicating with N2, N6 and N14 are shown in Table 2. For this experiment we use w_b =0.4, w_l =0.4 and w_d= 0.2. These values can be tuned based on the situation and importance of a given parameter. Table 3 shows the costs associated with replicating the data on the three nodes.

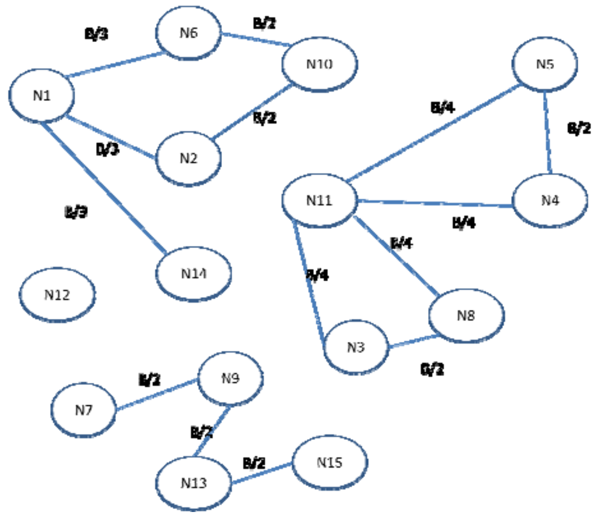


Figure 4. Sample Network

	T1	T2	T3	T4	T5	T6
N1 and N2	5	7	4	8	6	5
N1 and N6	5	7	6	4	7	8
N1 and N14	5	7	6	4	7	6

Table 2. RSS Values for Connectivity with N1

Node	Bandwidth	Degree	Link Stability	Cost
N2	0.33	0.14	0.37	3.33
N6	0.33	0.14	0.66	2.35
N14	0.33	0.07	0.33	3.7

Table 3. Cost of replication on nodes N2, N6 and N14

5. MESSAGE ACKNOWLEDGEMENT

The nodes in the scenario communicate with each other over wireless links. TCP does not scale well such wireless networks as nodes frequently disconnect and rejoin. However, for critical applications, reliable transfer of packets is imperative. Wireless links use open air as the transmission medium and are subject to many uncontrollable quality-affecting factors such as weather conditions, urban obstacles, multipath interferences, large moving objects, and mobility of wireless end devices. As a result, wireless links exhibit much higher Bit Error Rate (BERs) than wired links. Limitations of radio coverage and node mobility require frequent handoffs, thus resulting in temporal disconnections and reconnections between the communicating end hosts during a communication session. During a disconnection both the retransmission and ACK's will be lost, repeated failed retransmissions results in stalling the TCP transmission for a much longer period (Slow start). In view of the above, we have developed a simple mechanism for acknowledging User Datagram Protocol (UDP) packets at the application layer.

6. CONCLUSIONS

In this paper we presented a middleware architecture for highly mobile environments. In particular the architecture is suitable for a network of UAVs sharing content. The proposed middleware supports network and data management, database query processing, replications and fault tolerance. Several elegant mechanisms and algorithms have been developed. During the next phase of the development, we will demonstrate the effectiveness of the proposed middleware for fault-tolerant database query processing in highly mobile environments.

ACKNOWLEDGEMENTS

This project is funded by Air Force Research Lab (AFRL) award number FA8750-09-2-0199. Approved for public release; distribution unlimited (Case Number: 88ABW-2011-2947).

REFRENCES

1. M. Boulkenafed and V. Issarny, Middleware service for Mobile Ad Hoc Data sharing, enhancing and data availability, Middleware 2003, Lecture Notes in Computer Science, 2003, Volume 2672/2003, 999, pgs. 6-25.
2. H. C. Christmann and E. N. Johnson, "Design and Implementation of a Self-configuring Ad-hoc Network for Unmanned Aerial Systems" AIAA 2007, California.
3. S. Kalasapur, M. Kumar, and B. Shirazi, Dynamic Service Composition in Pervasive Computing Systems, IEEE Transactions on Parallel and Distributed Systems, Volume 18, Issue 7, July 2007 Page(s):907 - 918.
4. S. Tamhane and M. Kumar, Middleware for Decentralised Fault Tolerant Service Execution using Replication in Pervasive Systems, Sixth International Workshop on Middleware Support for Pervasive Computing (PerWare2010), PerCom 2010, Mannheim, Germany, March 29-Apr. 02, 2010.
5. Y. Yu, B. Krishnamachari, Issues in designing middleware for wireless sensor networks, IEEE Network 18 (February) (2004) 15–21.