

# AMS: AN ADAPTIVE MIDDLEWARE SYSTEM FOR WIRELESS AD HOC NETWORKS<sup>1</sup>

Cho-Yu Jason Chiang, Ritu Chadha, Gary Levin, Shihwei Li, Yuu-Heng Cheng and Alex Poylisher  
Telcordia Technologies  
Piscataway, NJ 08854

## ABSTRACT

*The characteristics of ad hoc networks greatly increase the complexity of programming distributed applications with respect to both sending and receiving messages. This is because the network programming API based upon TCP and UDP was designed for the relatively stable, stationary Internet. The semantics of the API does not address the emerging issues from the dynamic mobile wireless ad hoc networks. One promising approach for alleviating this problem is to develop applications on top of a middleware system to shield applications from dealing with the dynamics of ad hoc networks. However, conventional middleware systems were not designed to cope with the frequently changing communications characteristics of military ad hoc networks.*

*In order to address the above problem, we have designed and implemented a prototype of AMS<sup>1</sup>, an Adaptive Middleware System with the following features. First, this middleware system is specifically designed to deal with dynamic and unreliable networks. Second, it shields communicating entities from coping with frequently changing networks via an API with semantics that provide a suitable abstraction of the underlying dynamic networks. Third, it allows individual applications to define their own communications requirements while the middleware system can set the lowest common requirements for all the communications per the military situations. Lastly, this middleware system has an interface that allows an external control system to adjust the values of its parameters to optimize the overall system performance.*

## INTRODUCTION

Due to rapid advances in wireless networking technologies, today's networks depend far less on fixed infrastructure and have shifted towards providing networking services on demand. The emergence of wireless ad hoc networks in the military context is a demonstration of this. Such networks do not require any pre-installed infrastruc-

ture, and may be stationary or mobile. Moreover, they pose various stringent requirements in areas such as security, reliability, robustness, bandwidth consumption, power efficiency, etc. They are often highly dynamic due to voluntary node participation, unrestricted node movement, and unpredictable loss of connectivity.

In the past, wireless ad hoc networking research focused on enabling fundamental network functions spanning the physical layer, the MAC layer, and the IP layer [25]. There was not enough attention being paid to developing applications for use in the wireless ad hoc environments. Possible reasons are the following. First, the assumption was that the current transport protocols, primarily TCP and UDP, would continue to be used, and therefore the network programming API would remain the same. This would indicate that developing applications for wireless ad hoc networks would be no different from developing applications for the Internet. Second, network programming for the Internet is generally regarded as a mature and well-understood field. Software engineering discipline and tools for facilitating the design, development, and testing of Internet applications are readily available. Third, research dealing with the underlying network stack layers for wireless ad hoc networks is still at the prototyping stage, and therefore the field of distributed application development for such networks is still in its infancy.

Having gained some experience with the implementation of a distributed network management application [24] for use in wireless ad hoc networks, we observed that the characteristics of such networks significantly increase the programming complexity in terms of both sending and receiving messages. To ensure that communications always perform as well as the underlying ad hoc network allows without overloading the network, software developers need to take into account a set of network conditions such as end-to-end route connectivity, available bandwidth on the wireless links, and transport protocol behavior over volatile links. This is because the network stack was designed for the infrastructure-based, bandwidth-abundant, and typically reliable Internet. As a result, the stack does not provide adequate support for applications running over wireless ad hoc networks where there is no infrastructure, bandwidth is unstable, and networking services are generally unreliable.

---

<sup>1</sup> The research reported in this document/presentation was performed in connection with contract number DAAD19-01-C-0062 with the U.S. Army Research Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as presenting the official policies or position, either expressed or implied, of the U.S. Army Research Laboratory, or the U.S. Government unless so designated by other authorized documents. Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Our view is that tools for developing network-based applications for use in wireless ad hoc networks require immediate attention, for the following reasons. First, the current network programming API based on the well-known TCP and UDP Internet transport protocols is not a suitable abstraction for use in wireless ad hoc networks. The TCP protocol is known to have performance issues in environments with poor link connectivity due to its built-in congestion control mechanism [23]. Also, irresponsible use of UDP without considering the network conditions and traffic patterns may aggravate congestion in an already problematic network. Second, due to volatile radio links that could be affected by interference as well as node movement, system designers and application developers have to carefully design and implement their systems in order to obtain the desired degree of reliability over unreliable networks. In addition, software testing tools and guidelines are also needed because of the difficulty of generating and executing test cases for dynamic, versatile ad hoc networks. Third, there are already military applications under development that are to be deployed over wireless ad hoc networks such as those being built under the FCS [26] and WIN-T [27] programs of US Army. If application development issues are not properly addressed, the risk of building distributed applications over wireless ad hoc networks is much higher.

One approach for alleviating these problems is to develop applications on top of a middleware system which sits between the applications and the network stack. The applications can thereby be completely shielded from dealing with the dynamic networks. This approach shifts the programming complexity in dealing with the dynamic networks from the applications to the middleware that they are built upon.

Conventional middleware systems such as CORBA [28] lack the capability to meet the stringent communications requirements imposed by the military ad hoc networks. For such networks, (i) communications requirements may change dynamically based on criteria such as planned missions, geographical locations and battlefield status, thus resulting in a need to adapt the communications behavior on demand; (ii) concerns about QoS, security, survivability, bandwidth efficiency, etc., require a middleware system to be more versatile than the state-of-the-art systems; and (iii) conventional middleware treats packet losses, network failures, and intermittent node connectivity as rare exceptions, whereas these situations are commonplace in ad hoc networks and should be handled differently.

In order to address the above problem, we have designed and implemented a prototype of AMS, an Adaptive Middleware System with the following features. First, this middleware system is specifically designed to deal with dynamic and unreliable networks. Second, it shields com-

municating entities from dealing with the frequently changing network conditions via an API with semantics that provide a suitable abstraction of the underlying dynamic networks. Third, it allows individual applications to define their own communications requirements while the middleware system can dynamically adjust the minimum requirements for the outstanding communication sessions per the military situational requirements. Lastly, this middleware system has an interface that allows an external control system to adjust the values of its parameters to optimize the overall system performance.

The rest of this paper is organized as follows. In the next section we discuss in more detail the motivation for designing an adaptive middleware system and the challenges posed by such a design. Following this, we present the design of our adaptive middleware system and elaborate on the rationale behind the design. Then we describe our experience with building a distributed application for managing ad hoc networks using this middleware system, and articulate the benefits of using it from the perspective of the applications. We also discuss the planned enhancements to this middleware system, the related research work and the issues that remain to be resolved before concluding this paper.

## SYSTEM DESIGN REQUIREMENTS

In the previous section we have touched upon some issues inherent in wireless ad hoc networks that motivate the design of an adaptive middleware system. By building applications on top of an adaptive middleware system, the applications can benefit from not having to directly deal with the underlying dynamic network, and from automatic performance optimization. We derived a set of system design requirements by answering the questions below:

- *What are the features an adaptive middleware system should possess?* Since the goal is to design a middleware system that adapts the outstanding communications to the dynamic network condition changes for the applications, the system should possess the following features: (i) extensibility, flexibility, and dynamic configuration capability; (ii) ability to observe the communication requirements and respond to network condition changes through appropriate adaptation; and (iii) ability to perform online optimization to enhance the global network effectiveness and efficiency.
- *What are the semantics that should be provided by the API of an adaptive middleware system to the applications?* The system should provide a suitable abstraction of the underlying network, be it static or dynamic. Applications should not be directly bound to any specific protocol implementation in the network stack. Instead, they should be able to specify their own communications requirements when invoking a method on the pro-

gramming interface as a binding contract. They should be notified of communication failures only when the binding contract cannot be satisfied; this notification should provide sufficient information for the applications to be able to deduce the cause of the communication problems and determine how to react to the problems. With the current network programming API, applications sometimes may have to make blind decisions due to lack of finer-grained feedback about the current status of the network from the returned values or the exceptions thrown from API invocations.

- *How should an adaptive middleware system adapt itself to environmental condition changes?* Our view is that autonomous adaptation in itself is a very complicated process. Rather than providing the capability of adaptation within the middleware system, the system should provide a behavior adaptation interface that allows any external control entity to dynamically adapt its runtime behavior. Then the question to answer becomes: *what are the semantics that should be provided by the behavior adaptation API of an adaptive middleware system to an external control entity?*

### ADAPTIVE MIDDLEWARE SYSTEM

Based on the requirements identified in the previous section, we have designed and implemented an adaptive middleware system to address these requirements. The software prototype has been used to support the implementation of a distributed application whose purpose is to manage mobile wireless ad hoc networks.

The high-level architecture of this adaptive middleware system is illustrated in Figure 1. This middleware system consists of two layers: distribution layer and transport layer. The rationale behind having two layers inside this system is that we want to keep the components interfacing with the applications and those interfacing with the network transport completely separate. This design approach offers the following advantages:

- Since there is an interface between the distribution layer and the transport layer, it allows the middleware system to perform dynamic module chaining to service applications that have different distribution requirements with the most suitable network transport technologies. For example, if a new transport protocol can provide some sort of reliable transport services similar to those provided by TCP and is more suitable for ad hoc network communications, the middleware system could use this new transport protocol instead of TCP under certain situations. The module chaining is done at the communication session setup time. However, it can be changed during the session without the applications using the middleware system being aware of the change,

as long as their communications requirements can still be satisfied.

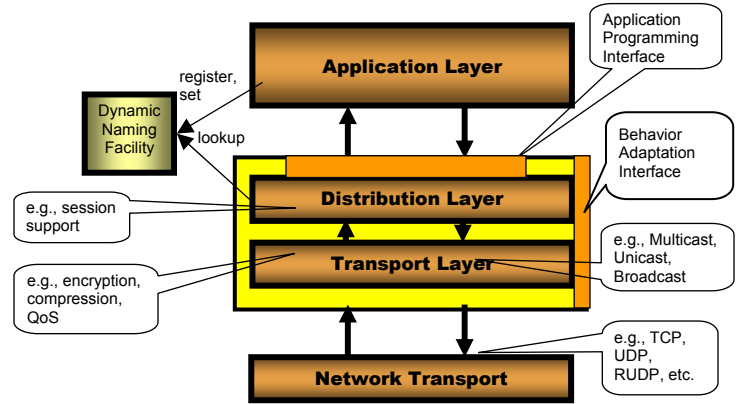


Figure 1. The high-level architecture of AMS

- The separation of distribution and transport functionalities also realizes the concept of *delayed binding* [29]. Past research has reported that the capacity of wireless ad hoc networks is primarily determined by traffic patterns. If the average number of hops for a packet to travel is much lower than the network diameter, a wireless ad hoc network would have higher overall throughput [30]. In addition, for role-based distributed applications such as those built with the client-server model and those organizing their distributed instances in a certain fashion, an application instance can use abstract role identifiers to specify the intended message recipients. This is especially useful when the roles of the distributed application instances could experience changes in highly dynamic network environments. This approach simplifies the application logic because the application delegates the identifier resolution responsibility to the middleware system. As a result, the distribution layer is responsible for resolving abstract role identifiers of the communication endpoints to their DNS representations, and the transport layer in turn maps the DNS representations to the network identifiers, which could be bundle identifiers in the context of Delay/Disruption Tolerant Networks [21][22], a multicast group ID if there exists underlying multicast support, unicast and broadcast IP addresses, or a mix of all the above.

We also have attempted to address the system design requirements on programming interface semantics. The distribution layer exposes a network programming API to the application layer. Applications request *distribution connections* for their communication needs from a *distribution connection manager* on the distribution layer. The concept of distribution connection is akin to that of Internet sockets, except that (i) a distribution connection is not tightly

bound to a fixed transport technology; (ii) it enables the mapping of abstract role identifiers to DNS representations based on the given resolution system; and (iii) it supports both unicast and multicast communications with the same set of APIs, which significantly facilitates the development of distributed applications because multicast messaging is usually preferred over multiple unicast messaging in bandwidth-deficient network environments. By building distributed applications using the API of AMS, developers do not need to be concerned about making the transport protocol selection decision at the development time.

Consequently, one of the most salient features of this design is that an application may select an appropriate distribution connection manager from a pool of available managers in the middleware system, and request a suitable distribution connection for its communication needs. The API also allows the application to specify its specific communications requirements. The communications requirements span a wide range of possibilities including session timeout requirements, communication authentication requirements, communication security requirements, QoS requirements, data compression requirements, just to name a few. When an application specifies the set of its communication requirements and requests a new distribution connection, the distribution connection manager will verify whether it can create a session that satisfies the requirements. The decision may also involve a handshaking process with the destination(s) and/or the dynamic maintenance of an end-to-end communication path. Once the distribution connection is created and made available to the application, the set of requirements is regarded as a binding contract between the middleware system and the application. The design philosophy is that as long as the middleware system can meet the given requirements, it does its best to honor the contract and maintain the session. Only if the terms listed in the binding contract fail to be observed will the middleware system notify the application. In principle, the notification message contains sufficient information for the application to decide whether it should try to set up a new contract by changing some of the requirements, or it should give up for a while and maybe try later with either the same or modified requirements.

The design of the transport layer is structurally similar to that of the distribution layer. A distribution connection needs to pass its requirements to a transport connection manager to request a transport connection. The transport connection manager may opt to create a new transport connection, or return an existing connection that this distribution connection will share with its peers. Also, the transport connection manager may opt to adjust the outstanding transport connections as long as the modified

transport connections can still satisfy the requirements of the distribution connections.

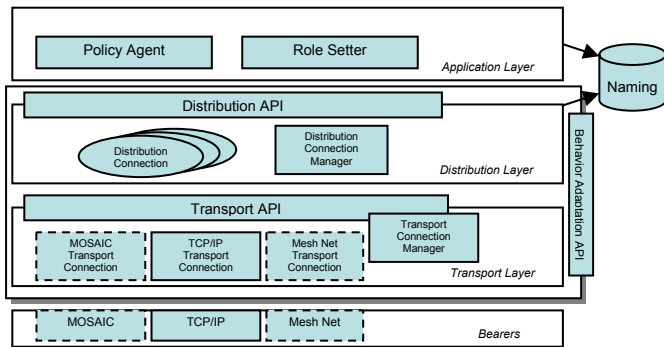
The other interface supported by the middleware system is designed to allow an external control system to adjust the behavior of the middleware system on-the-fly by passing behavior adjustment directives to the middleware system via this interface. Although the adaptation logic could be built directly into the middleware system, in this design they are separated for the sake of modularity, so that the middleware system can be integrated with an available adaptive control system. Based on this architecture design decision, we define an interface that allows an external control system to dynamically set the minimum communications requirements according to the overall system optimization goals and/or global/regional requirements. We also deem that the directives should be independent of the middleware implementation. Therefore, the global system communication directives are set at a high level and it is the responsibility of the middleware implementation to interpret the requirements and take appropriate actions.

The advantages of adopting this decoupling design paradigm are two-folds. First, there is a clear boundary between the local, autonomous optimization within the middleware system and non-local optimization and coordination from external directives. For example, the middleware system could decide by itself to switch from using one network transport to using another network transport because of its observation of the network stack performance. It could also encrypt the outgoing messages in all outstanding communication sessions when it receives appropriate directives from an external control system. To illustrate this concept, in the context of military communications security, one of the directives might be *InfoCon Level* [31]. This directive may influence the decision of what encryption standards should be used for message deliveries so that communications will be conformant with the global/regional security requirements of the system. Second, both middleware and control technologies are under active research today. By not requiring a tight coupling between the directives and the middleware implementation, we can maintain the stability of this control API, and obviate the need to make frequent modifications and extensions to the signatures of methods in the API when existing modules are updated, new modules are added to the middleware system, or more sophisticated adaptation logic is inserted into the external control system.

## A USE CASE STUDY

We have built a software prototype of AMS based on the design described in the previous section. This software prototype has been used as the middleware system for a distributed policy-based network management system [24]. In this section, we briefly describe this application and

then discuss the benefits of using this middleware system from the perspective of application development.



**Figure 2** A distributed role-based application and the middleware prototype—a use case study

In Figure 2, we give a high-level illustration of the application that achieves its goal of enabling distributed management for wireless ad hoc networks by organizing its distributed policy agents in a hierarchical fashion with different roles assigned to them. A *policy agent* performs management functions by enforcing a set of policies. The hierarchy is built and maintained autonomously, and the role of a policy agent determines what other policy agents in the hierarchy it needs to communicate with. The management actions are mostly done locally, but some of them require data collection and action coordination among a group of policy agents.

The possible roles for policy agents include GPA, DPA, and LPA. The *GPA* (Global Policy Agent) is the node at the root of the hierarchy; *DPAs* (Domain Policy Agents) are intermediary nodes in the hierarchy; and *LPAs* (Local Policy Agents) are leaf nodes in the hierarchy. Due to the unpredictable nature of the wireless ad hoc networks, the role of a node and the node assuming a specific role are always subject to change. Dynamic, autonomous role adjustments are accomplished by the role setter shown in Figure 2. The *role setter* is in charge of updating the Naming subsystem, which is responsible for resolving the mapping of roles to the domain names.

According to its role, a policy agent communicates with other policy agents. Although each policy agent performs its management functions locally, there are two major activities requiring inter-node communications. The first is policy distribution that allows any policy change to be distributed from the GPA downward in the hierarchy, and the second is event-based reporting which allows management information to be reported upward in the hierarchy.

By building this application on top of AMS, we reap the following benefits that would not be available without using AMS. Below we summarize our experience.

- *Simplify the complexity of the application implementation.* Most components consisting of the application do not need to be concerned about the physical node they should send their messages to. They can specify the intended recipients at a higher semantics level such as *parent, children, one-hop neighbors, root*, etc. The application delegates the responsibility of maintaining the communications session to the middleware, which handles situations such as IP address changes (mobile IP) and destination node switches (dynamic role change) transparent to the application, if permissible, during the course of communications. For example, in the actual implementation of the policy agent, some component needs to communicate with all its children under certain situations. The component only needs to inform the middleware system about the intended recipients, namely, children, and does not need to figure out or keep track of what nodes they are.
- *Adapt communications behavior based on global states, not just local states.* There is no need for each thread of communications to watch for the situational communications requirements. The middleware will be notified of any required communications behavioral changes through the directives received via the Behavior Adaptation Interface shown in Figure 2. For example, as a military unit marches into certain geographical region that is presumed to be the ready-for-battle region, all communications from the applications will be secured automatically without requiring the awareness from the applications.
- *Allow an application to respond to network conditions quickly.* The application has the option to set the timeout value in the communications requirements when requesting a distribution connection. It would not be tied to the timeout value as specified by a transport protocol. For example, the role setter may request a reliable round-trip message exchange for the heartbeat beacon verification purpose that has to complete in three seconds, regardless of the transport protocol the middleware system opts to use. If the message exchange has not completed by the specified time, the role setter would be notified of this communications failure and its possible cause.
- *Enable the building of a testing framework that greatly facilitates the testing process.* Testing distributed applications is not easy, and it is even harder if the testing is supposed to be conducted over real wireless ad hoc networks. By building the applications on top of middleware, it becomes possible to set up a virtual testing framework simulating the distributed network environment by controlling the behavior of middleware instances in the system. For example, we have developed a software tool that allows us to run multiple instances of the aforementioned distributed application on the

virtual machines hosted by one machine. Each application instance is running on a dedicated virtual machine. The tool allows developers to modify the radio link connectivity between the virtual nodes either manually or through scripting control. Thus, developers can come up with a set of testing scenarios simulating the dynamic network environments, at the convenience of seeing the global state change and examine what have happened on the virtual nodes by reading their logs on one single machine.

## DISCUSSION

In this section, we will describe the planned enhancements to this system, the other related research work, and the issues that remain to be resolved in our future work.

The current prototype of AMS is implemented in Java. It has an extensible software architecture design. The architecture can support a variety of plug-in elements. Both distribution layer and transport layer are designed following the factory method pattern [32] to facilitate the insertions and deletions of plug-ins. On the distribution layer, specifically it allows insertions of new classes of distribution connection managers and distribution connections. The current implementation has point-to-point and point-to-multipoint distribution connection managers, their distribution connections, and it supports the setting of communications requirements in connection timeout and data compression. On the transport layer, it allows the insertion of new classes of transport connections and communications module plug-ins. The current implementation has one transport connection manager, transport connections that are coupled with TCP and UDP transport protocol, and a data compression plug-in. As a result, part of the planned future work includes the additions of new types of connections, the support for other communications requirements such as encryption, authentication, mobile session support, etc., and the additions of communications module plug-ins needed to support the communications requirements.

On a different note, the adaptive middleware research receives considerable attention in the recent years. Research work in this area covers various aspects of building an adaptive middleware system. An area of interest was on the automatic adaptation through context-awareness [1][5][6]. The emphasis was on enabling active service deployment and reconfiguration of service composition to support mobile computing, which generally assumes the model of mobile clients and stationary servers. There was a survey paper on the adaptive resource management [3]. Some of the context-aware adaptive middleware research also incorporates different flavors of policy control into their framework to enable flexibility and extensibility [1]. However, none of these middleware systems was designed to support distributed applications for use in wireless ad

hoc networks in which nodes could be disconnected from each other frequently and also indefinitely.

The other research direction is on the middleware software framework and design patterns [2][7][8]. Reflection is a popular technique that was reported in some research [1][4][6][9]. Component-based and object-based software implementation technologies were examined to derive new programming framework for building adaptive middleware.

Another area of interest was focused on proposing suitable system architecture to support their specific system adaptation needs. Some research was related to pervasive computing [10][11][12], and some was related to supporting the sensor network operations [14][15][16]. The research falling under this category typically builds their work based on the specific domain assumptions for the intended applications. The proposed architecture pattern was not designed to provide a framework for building a flexible, extensible middleware system for wireless ad hoc network environments in general.

Also there was research on the layering architecture and the configurability of the middleware [16][17][18][19]. Some of them might be considered as preliminary work and others presenting their designs without giving supportive arguments stating the design philosophy of and reasons for their function separations.

Distributed applications based on the CORBA [28] model was prevailing in the research reports. However, that may not be a good system paradigm for distributed applications in wireless ad hoc network environments due to the network disconnection issues [13][20]. Our view is that we need to enhance the existing model in terms of API semantics and the component architecture, which are the main focuses in our future work.

## SUMMARY

In this paper, we present an adaptive middleware system that was designed to support building distributed applications for use in wireless ad hoc networks. This middleware system is novel in the sense that it possesses all of the following features: (i) capability of dealing with dynamic and unreliable network environments, (ii) an API with semantics that provide applications on top of it a suitable abstraction of the underlying dynamic networks; (iii) customizable communication requirements per the application needs; and (iv) capability to be controlled to conform to the military situational requirements and/or optimize the overall system performance by adapting itself to environmental condition changes. We also describe our experience of building a distributed network management application on top of this middleware system. The design philosophy of this middleware system is discussed and the benefits of using this middleware system are articulated.

## REFERENCES

- [1] A. Chan and S.-N. Chuang, "MobiPADS A Reflective Middleware for Context-Aware Mobile Computing," *IEEE Transactions of Software Engineering*, Vol. 29, No. 12, pp. 1072-1085, December 2003.
- [2] N. Badr, A. Taleb-Bendiab, M. Randles, and D. Reilly, "A Deliberative Model for Self-Adaptation Middleware Using Architectural Dependency," *Proceedings of the 15<sup>th</sup> International Workshop on Database and Expert Systems Applications*, Zaragoza, Spain, August 30th - September 4th 2004.
- [3] H. A. Duran-Limon, G. S. Blair and G. Coulson, "Adaptive Resource Management in Middleware: A Survey," *IEEE Distributed Systems Online* 1541-4922, Vol.5, No. 7, July 2004.
- [4] Q. Han, S. Gutierrez-Nolasco and N. Venkatasubramanian, "Reflective Middleware for Integrating Network Monitoring with Adaptive Object Messaging," *IEEE Network*, pp. 56-65, January/February 2004.
- [5] P. Bellavista, A. Corradi, R. Montanari, and C. Stefanelli, "Context-Aware Middleware for Resource management in the Wireless Internet," *IEEE Transactions of Software Engineering*, Vol. 29, No. 12, pp. 1086-1099, December 2003.
- [6] L. Capra, W. Emmerich, and C. Mascolo, "CARISMA: Context-Aware Reflective Middleware System for Mobile Applications," *IEEE Transactions of Software Engineering*, Vol. 29, No. 10, pp. 929-944, October 2003.
- [7] V. Subramonian and C. Gill, "A Generative Framework for Adaptive Middleware," *Proceedings of the 37<sup>th</sup> Hawaii International Conference on System Sciences*, 2004.
- [8] J. Mitchell and A. J. Sanchez-Ruiz, "An Architectural Pattern for Adaptive Middleware Infrastructure," *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration*, Las Vegas, October 2003.
- [9] G. Huang, T. Liu, H. Mei, Z. Zheng, Z. Liu, and G. Fang, "Towards Autonomic Computing Middleware via Reflection," *Proceedings of the 28<sup>th</sup> Annual International Computer Software and Applications Conference*, 2004.
- [10] J. Al-Muhtadi, S. Chetan, A. Ranganathan, and R. Campbell, "Super Spaces: A Middleware for Large-Scale Pervasive Computing Environments," *Proceedings of the 2<sup>nd</sup> International Conference on Pervasive Computing and Communications Workshop*, 2004.
- [11] E. Niemela and T. Vaskivuo, "Agile Computing of Pervasive Computing Environments," *Proceedings of the 2<sup>nd</sup> International Conference on Pervasive Computing and Communications Workshop*, 2004.
- [12] M. Mamei and f. Zambonelli, "Programming Pervasive and Mobile Computing Applications with the TOTA Middleware," *Proceedings of the 2<sup>nd</sup> International Conference on Pervasive Computing and Communications Workshop*, 2004.
- [13] M. O. Junginger and Y. Lee, "A Self-Organizing Publish/Subscribe Middleware for Dynamic Peer-to-Peer Networks," *IEEE Network*, pp. 38-43, January/February 2004.
- [14] Y. Yu, Bhaskar Krishnamachari, and V. K. Prasanna, "Issues in Designing Middleware for Wireless Sensor Networks," *IEEE Network*, pp. 15-21, January/February 2004.
- [15] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo, "Middleware to Support Sensor Network Applications," *IEEE Network*, pp. 6-14, January/February 2004.
- [16] P. Ferreira, L. Veiga, and C. Ribeiro, "OBIWAN: Design and Implementation of a Middleware Platform," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 11, pp. 1086-1099, November 2003.
- [17] S. Bottcher and C. Dannewitz, "A Reconfigurable Message Oriented Middleware Architecture," *Proceedings of the International Parallel and Distributed Processing Symposium*, 2003.
- [18] A. D. McKinnon, K. E. Dorow, T. R. Damania, O. Haugan, W. E. Lawrence, D. E. Bakken, and J. C. Shovic, "A Configurable Middleware Framework with Multiple Quality of Service Properties for Small Embedded Systems," *Proceedings of the 2<sup>nd</sup> International Symposium on Network Computing and Applications*, 2003.
- [19] K. Dorow, "Flexible Fault Tolerance in Configurable Middleware for Embedded Systems," in *Proceedings of the 27<sup>th</sup> International Computer Software and Applications Conference*, 2003.
- [20] I. Burcea, H.-A. Jacobsen, E. d. Lara, V. Muthusamy, and M. Petrovic, "Disconnected Operation in Publish/Subscribe Middleware," *Proceedings of the 2004 IEEE International Conference on Mobile Data Management*, 2004.
- [21] K. Fall, "A Delay Tolerant Network Architecture for Challenged Internets," *Proceedings of SIGCOMM'03*, August 25-29, 2003, Karlsruhe, Germany.
- [22] "Disruption Tolerant Networking", BAA from DARPA 2004, <http://www.darpa.mil/ato/solicit/DTN/>.
- [23] H. Balakrishnan, S. Seshan, E. Amir and Randy H. Katz, "Improving TCP/IP Performance over Wireless Networks," In *Proceedings of the 1st ACM International Conference on Mobile Computing and Networking (Mobicom)*, 1995.
- [24] R. Chadha, Y-H Cheng, C. Chiang, S. Li, and G. Levin "Policy-Based Mobile Ad Hoc Network Management", *Proceedings of the IEEE 5th International Workshop on Policies for Distributed Systems and Networks*, Yorktown Heights, New York, June 7-9 2004.
- [25] C. S. Ram Murthy, B. S. Manoj, "Ad Hoc Wireless Networks: Architectures and Protocols," Prentice Hall, May 2004.
- [26] US Army, "Future Combat Systems", <http://www.army.mil/fcs/>
- [27] US Army, "Warfighter Information Network-Tactical", <http://peoc3t.monmouth.army.mil/WIN-T/WIN-T.html>
- [28] CORBA, [www.corba.org](http://www.corba.org)
- [29] J. Vetter and K. Schwan, "Techniques for delayed binding of monitoring mechanisms to application-specific instrumentation points," *Proceedings of the International Conference on Parallel Processing*, 1998.
- [30] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks", *MobiCom*, 2001.
- [31] Information Operations Condition, <https://infosec.navy.mil/Documents/>
- [32] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns, Elements of Reusable Object-Oriented Software," Addison Wesley, 1995.