A MOBILE AGENT-BASED COMMUNICATIONS MIDDLEWARE FOR DATA STREAMING IN THE BATTLEFIELD

Marco Carvalho, Niranjan Suri, Marco Arguedas
Florida Institute for Human and Machine Cognition
40 S. Alcaniz St. Pensacola, FL 32502

(Paper Abstract ID: 1482)

ABSTRACT

In this paper we introduce the FlexFeed framework in the context of military combat operations. FlexFeed realizes the notion of Agile Computing for streaming data communications and implements a flexible, robust and efficient publish/subscribe infrastructure for dynamic ad hoc environments under resource and policy constraints. The framework uses mobile software agents for underlying configuration and policy enforcement. The paper illustrates the effectiveness of the framework with quantitative experiments over simulated scenarios.

INTRODUCTION

Dependable communication capabilities are amongst the most important technical requirements for mission success in military operations. Complex missions involving coalition forces, robotic support units, remote sensor beds and autonomous vehicles will require underlying communication infrastructures that are more flexible, efficient, and robust in order successfully operate in the face of enemy attacks.

Most communications between peers in the battlefield are either to exchange state and environmental information or to relay command and control messages. State information includes, for instance, relative position of troops and vehicles (enemy and friendly), sensor data from unmanned vehicles or sensor beds, situation data, etc. This type of data is often transmitted as streams of arbitrary durations, such as video-feeds from a camera sensor or continuous GPS position data from moving vehicles or troops.

Furthermore, the communications infrastructure must adjust to changes in overall mission goals and operations tempo. During monitoring and recognition missions, conserving power might be the primary objective function to extend the life of network resources. However, as engagement takes place, the communications infrastructure may need to quickly shift into a high performance mode to effectively support and optimize the kill chain as the primary objective.

In this paper we present FlexFeed, a mobile-agent based communications framework, applied to the battlefield scenario. Our proposal leverages from years of research in the fields of mobile ad hoc networks and intelligent software agents to build an efficient, self-configurable, and self-healing communications network for these types of environments.

After an introductory description of the environment and system requirements, we will discuss the related work in this area and the concepts proposed in FlexFeed. A brief description of the implementation details of framework will be then followed by case studies, presented and experimentally evaluated on a simulated network to illustrate FlexFeed capabilities.

COMMUNICATIONS IN THE BATTLEFIELD

As part of the Army's Objective Force to be deployed within the next decade, Future Combat Systems (FCS) is envisioned as a system of systems that will integrate several lightweight, highly mobile components including new generations of manned and unmanned military vehicles (Sharoni & Bacon, 1997; Pike, J., 2003). These light vehicles will partially replace heavy armored slower vehicles in order to bring unprecedented levels of dynamism and agility to the combat theater.

Furthermore, FCS operations will heavily rely on information superiority to quickly take control of the battlefield and appropriately react to enemy movements and changes of strategy. This capability depends on the notion of universal tasking, where resources and information are directly available at any time to the edge warriors and commanders in the field.

An enabling key-capability for this vision is an efficient and adaptive communications infrastructure to support and extend edge warrior capabilities and provide access to critical information at any time, while at the same time ensuring optimal resource utilization and security both at the infrastructure and information levels. Figure 1 shows a schematic view of some of the elements involved in these types of operations.

In general, a communications model capable of supporting FCS requirements is an ad-hoc publish/subscribe model. Soldiers and systems in the network will subscribe to sensor data and state information to plan and coordinate local tasks in response to high level instructions from the command and control center.

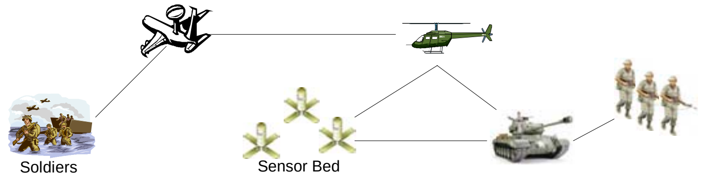


Figure 1 – Communications infrastructure in the battlefield

Based on FCS requirements, an appropriate data communications framework must be capable of satisfying the following requirements:

a) Ad hoc: In most cases, as illustrated in figure 1, networks between nodes will be ad hoc, formed by proximity during the operation itself. The communications infrastructure must not depend on pre-established infrastructural components or centralized management stations. This capability is important both from a scalability and robustness perspective, eliminating (or mitigating) single points of failure in the network.

b) Efficiency: Communications and computational resources in the battlefield are expected to be limited and often times, battery operated. Ad hoc sensor beds and small autonomous vehicles deployed during the operation will have a life-span strictly limited by their battery life. In most cases, it is imperative that the communications infrastructure operate efficiently across different types of applications and scenarios to extend the life of network resources.

c) Heterogeneity: Systems in the battlefield tend vary greatly in terms of computation and communications capabilities. Lightweight attack vehicles, small robotic units, and unmanned aerial vehicles will all have different degrees of computation and sensing capabilities and access to the wireless environment.

d) Application-aware Capabilities: A common limitation in most communications frameworks currently available is the lack of interaction between applications and the underlying data transmission protocols. This limitation is often accepted in lieu of the benefit of layer isolation and in making protocols interchangeable. For improved efficiency however, the communications infrastructure can and should benefit from data-aware protocols at all levels.

d) Robustness to External Attacks: The communications infrastructure must be able to resist to both physical and network attacks. Degradation with loss of communication resources must be graceful and most importantly, must be selective. Special types of operations and tasks that are critical to the overall operation must have precedence over less relevant tasks. This requirement goes beyond the conventional notion of quality of service in data networks. Ideally, the framework must be aware of the importance of data transmission not only in terms of data-type, source, and destination, but also in terms of high level goals and mission OPTEMPO in order to make prioritization decisions.

c) Robustness to Environmental Changes: The environmental conditions, topology, and size of the network will vary significantly. In the battlefield, nodes can arbitrarily join and leave the network. Nodes can be physically destroyed or made unavailable at any time. An appropriate communications infrastructure must be able to cope with these changes quickly and efficiently.

e) Reactive and Proactive OPTEMPO Adaptability: The framework must also be able to properly adapt to change in overall mission goals or situation in the battlefield. For instance, changes in operational tempo can be either pushed to or autonomously detected by framework nodes, which should automatically result in changes in the communications behavior. For instance, when precursors of engagement are identified, the framework, in accordance with global policies, must autonomously switch from a power efficient mode a low latency, high performance mode to support combat systems.

f) Proactive Resource Manipulation for Survivability and Improved Efficiency: This notion was initially proposed within the context of Agile Computing (Suri, 2002). It refers to the notion of granting the framework with the ability to proactively manipulate physical (or logical) resources in the framework in order to recover critical connectivity segments or to significantly improve performance.

g) High Level Policies for Monitoring and Control: From a human perspective, monitoring and control of such complex systems is a very difficult task. An appropriate framework for these types of systems must support interfaces to policy infrastructures that would allow humans to easily define and establish constraints and obligations to regulate the overall operation of the framework. From an optimization perspective, most policies would ultimately result in low level constraints taken into account by the framework when deciding about resource allocation.

In the last few years, a number of research proposals have been introduced to address some of these requirements. Common to most of them is the notion of a customizable publish/subscribe communications mechanism capable to efficiently support messaging and data streaming.

RELATED WORK

Conventional topic-based publish/subscribe systems such as such as Vitria (Skeen, 1998), TPS (Eugster et al., 2001) and JORAM (Maistre, 2003) leveraged form multicast protocols and the assumption of a clear hierarchy on data and events to build efficient multicast groups for topic-based data distribution. Multicast based protocols often provide an efficient solution to the problem but they assume that only nodes participating in the multicast group would share the data for distribution (at the level of the multicast tree). Furthermore, most multicast protocols assume data (or events) to be strictly hierarchical and processing capabilities for data transformation within the hierarchy must be available a priori at all nodes.

A number of gossip-based (or epidemic) protocols were also proposed in the same context (Lin and Marzullo, 1999; Ganesh et al., 2001; Eugster et al., 2003). In general, these are efficient and scalable protocols but assume no data hierarchy and often make no attempt for cost or constraint optimization based on data stream aggregation and filtering.

Multicast protocols specifically designed for peer-to-peer networks such Scribe (Rowstron et al., 2001) and HiCan (Ratnasamy et al., 2001) came to solve scalability issues in addressing and group coordination. They too, however, assumed that only nodes subscribed to the multicast group would participate in the multicast tree and that data processing capabilities are available at all nodes a priori.

Alternatives to the multicast option were also proposed at the level of unicast routing in the form of data-aware customized ad hoc routing protocols. An important example of these types of data-centric routing protocols is Directed Diffusion (Intanagonwiwat et al. 2000). The Directed Diffusion protocol proposes a highly scalable data-aware decentralized routing algorithm. The protocol supports the creation of data distribution trees including nodes that are not directly subscribing for the data. The protocol, however, also assumes that data transformation capabilities are available a priori at each node, which is not a realistic assumption for the types of environments envisioned in FCS.

More recently, Baehni et al. (2004) proposed a data-aware multicast protocol (daMulticast) for peer-to-peer networks. The approach leveraged from some of the data-centric techniques for data description and group membership, significantly improving reliability and at the same time reducing the memory complexity involved in maintaining group membership at each node.

In the most part, the approaches share the notion of using data-aware techniques for resource or performance optimization. The problem, however, is that data-aware frameworks are usually highly customized to a set of applications or data types, often requiring significant time and effort to support new scenarios or capabilities. In many cases, such changes are not even possible, as hardware might have been already deployed or might be under external administrative control, like in the case of combat or Military Operations Other than War (MOOTW) coalition operations.

THE FLEXFEED FRAMEWORK

In this paper we propose FlexFeed, a mobile-agent based communications framework designed to support highly customized data streams in mobile ad-hoc network environments under policy and resource constraints.

The concepts implemented in FlexFeed were first introduced by Carvalho et al. (2002). The fundamental ideas of the framework are based on the concepts of Agile Computing (Suri, 2002) where network and system resources are opportunistically exploited to transparently support application requests in a manner that is efficient, robust, and adaptable to changes in the environment.

The FlexFeed framework is essentially based on three core concepts: a) Opportunistic resource exploitation; b) Flexibility and run time self-configuration via on-demand code and process migration; and c) In-stream data processing. In the framework, these capabilities are combined and extended to address the requirements identified in the types of environments expected in FCS.

The framework uses data-aware mobile agents to better customize multicast trees and to provide in-stream data processing (i.e. to take advantage of the multi-hop nature of the communications path in these types of environments to distribute computation data processing loads). Specialized agents can be injected in the framework by authorized parties at run-time, allowing for great flexibility and support of highly specialized data streams. The overall behavior of the framework is regulated by high level policies defined, verified, and distributed by an integrated policy infrastructure designed for multi-agent systems. A proof-of-concept version of the FlexFeed framework was developed and tested both in simulated and physical environments. The framework was also demonstrated in actual live exercises conducted by the Army (ARL QL2, 2004) and the demonstrations for the Navy (ONR NAIMT, 2004). In the subsequent items, we will briefly discuss the implementation details of the framework, followed by experimental simulation results of illustrative case studies.

THE FRAMEWORK COMPONENTS

The FlexFeed framework is a distributed application-level middleware that is installed in all participating systems. The middleware provides an API that allows applications to specify services or requests for data streaming.

At the implementation level, the framework combines a mobile agent system with resource coordination and allocation mechanism and a policy infrastructure to determine and configure, at runtime, efficient data distribution trees between applications.

The mobile agent system gives the framework the ability to move code and computation between nodes to enable, on demand, new data-specific capabilities in nodes that will participate in the data distribution tree. Process migration is used to improve survivability and system performance. Although FlexFeed can be easily configured to work with different agent systems, our proof of concept implementation was developed on top of the NOMADS agent system (Suri et al., 2000; Groth and Suri, 2000).

NOMADS is a mobile agent system for Java-based agents. It provides two implementations: Oasis and Spring. Oasis incorporates a custom Java-compatible Virtual Machine (named Aroma) whereas Spring is a pure Java implementation. The Aroma VM is a clean-room VM designed to provide the enhanced capabilities of execution state capture and resource control.

The resource coordination component (referred to in this paper as the 'coordinator') is the intelligent part of the framework. It is responsible for realizing the notion of agile computing in the context of data streaming. The 'coordinator' can be implemented as a distributed process or as a centralized component operating in one of the nodes of the framework. All experiments and examples shown in this paper are based on one specific implementation of a centralized coordination algorithm (ULM) but decentralized alternatives are also available.

The policy infrastructure is independent of the framework. The goal of the policy framework is to provide a high level interface to the system in order to allow both human operators and applications to establish, query and modify high level requirements and constraints that will regulate how the framework should operate. Furthermore, the policy infrastructure is also responsible for validation, verification, disambiguation, and distribution of policies throughout the system. Policies can also be used to regulate and constrain the autonomous behavior of the framework, providing bounds for self-adjustments to operation tempo and to the proactive manipulation of resources. Currently, FlexFeed uses KAoS (Bradshaw et al., 1997; Bradshaw et al. 2002; Bradshaw et al., 2003) as its policy framework.

Access to these components is available at each node through a common API. In order to participate in the framework, applications at each node can obtain an instance of the *FlexFeedManager* Component (Figure 2). The FlexFeedManager provides the access API to the framework and allows applications to register, advertise capabilities, and request data streams from other resources.

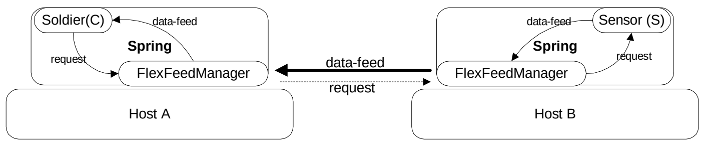


Figure 2 – The FlexFeed Framework.

Transparent to the applications, FlexFeedManagers at each node communicate in a peer-to-peer fashion to exchange state and plan resource utilization. When a client places a request for a data stream from a sensor as illustrated in figure 2, it specifies the source of the data and the requirements for the data stream (for example, resolution and frame rate in the case of a video stream).

That information, along with resource availability information from local nodes, is used to build the data distribution tree from source to client. If using a centralized coordinator, the planning is done at one single location using global state data. Decentralized coordination algorithms rely on peer to peer negotiation between FlexFeedManagers and use only local state for planning.

The client is allowed to specify any type of data, granted that it provides to the framework the necessary information for cost calculation and the code (in the form of mobile agents) necessary to manipulate (e.g. aggregation and filtering) the data for optimization. Because FlexFeed supports on-demand code deployment, trusted applications can provide new components to the framework at run time, enabling the support of previously unknown data types.

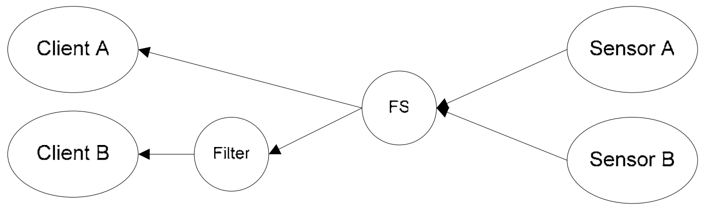


Figure 3 – A Complex Data Request

The client can also provide complex data processing requests such as the one illustrated in figure 3. In that example, the client is specifying (through a graph structure) two distinct data sources that should be merged with a specific (client-provided) processing element (FS) and then, discriminatively delivered to two sink nodes. Details about the data types and processing elements are embedded in the graph node and edge components, using a pre-defined data structure provided by the framework.

The FlexFeed framework will load the appropriate software components specified by the client (either from the client host of from a common codebase) and will identify the network resources necessary to support the request. The location of the logical fusion element (FS) can be at any intermediate node between the source and sink elements, based on resource availability, policies, and overall costs for computation and data transmission.

After mapping the request to the physical network, the framework will monitor environmental changes (such as significant variations in resource availability or link failure) to transparently recalculate and adjust the data tree until the request is terminated by the client.

CASE STUDIES AND EXPERIMENTAL RESULTS

The framework was tested in a simulated network where packet drops and bandwidth constraints could be carefully controlled. The goal of these tests was to demonstrate the effectiveness of on demand configuration of data-aware streaming in the improvement of data quality and reduction of jitter. These metrics are highly relevant to applications such as the remote control of unmanned vehicles.

The overhead of the framework was also measure in terms of induced latency in the stream. The computational overhead for running intermediate processing elements and filters was disregarded and the coordination mechanism used in the experiments was the centralized ULM (Carvalho, 2005) algorithm, based on an iterative version of Dijkstra's shortest path algorithm applied in localized parts of the graph.

The experiments were conducted on a 100baseT network with full connectivity. Bandwidth limitations between UAV and other nodes were simulated on a fixed wired network. Figure 4 provides a schematic view of the test networks.

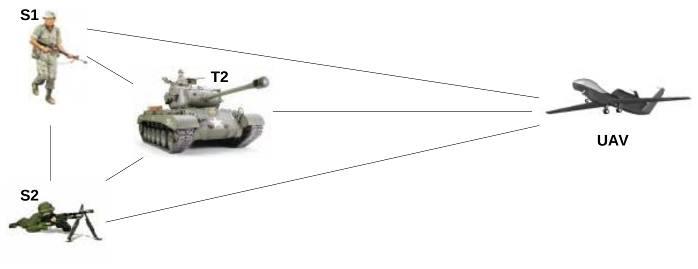


Figure 4 – Schematic illustration of the environment considered for experiments

In this configuration, nodes 'S1' and 'S2' represent two dismounted soldiers in direct communications range with each other and with a tank nearby 'T2'. All nodes are in communications range with an unmanned aerial vehicle 'UAV' on a fixed flying pattern over enemy territory.

The bandwidth available from the UAV to the remaining nodes is variable and can be severely constrained at different times. Our experimental procedure explores different operational scenarios on top of this configuration. The goal is to quantitatively illustrate how the FlexFeed framework improves data communications by reducing delays between image updates and the variance between packet arrival times (jitter). In our experimental setup, each node is represented by a separate laptop. The bandwidth limitations on the UAV are simulated by a gateway running NISTNet (Carson, 2002). Figure 5 shows the experimental setup designed to simulate the environment illustrated in figure 1.

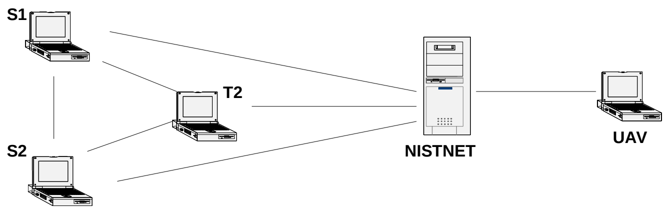


Figure 5 – Experimental setup

The centralized coordination node (not shown in figure 5) receives state information such as CPU and bandwidth availability from each of the 4 nodes involved in the test. The frequency of updates is proportional to the rate of change in these metrics. When a client makes a request for a data stream, it specifies the source node (UAV), frame rate, and resolution. The coordinator node will receive the request and will handle it appropriately, building a data distribution tree from the source, based on current global system state.

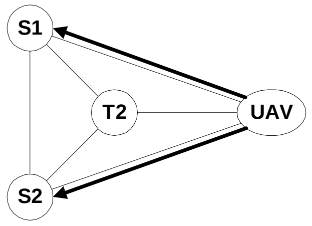
Optimizing Bandwidth Utilization

In the first scenario, soldier 'S1' temporarily assumes control of the UAV, taking it out of the flying pattern and closer to enemy positions. Unaware of the fact that the vehicle is now under remote control, soldier 'S2' also requests a video stream from the UAV's camera. In this example, both video streams were requested at a 320x240 resolution with 3 frames per second.

Under unconstrained conditions, the combined streams require the UAV to send approximately 50 KBps of data. In the initial condition the bandwidth limitation is 100 KBps (equivalent to unconstrained bandwidth in this example) so there are no packet drops and the average delay between images is 271 milliseconds, which represents a stream of approximately 2.69 frames per second. Note that the resulting frame-rate, even under unconstrained bandwidth conditions, only approximates the requested frame-rate. This is due to the delays involved in actual image capture (which is camera dependent), compression, and serialization.

The bandwidth available from the UAV is then progressively reduced to a maximum of 40 KBps, 30 KBps, and then 24 KBps. At each step, the average delay between images is measure at each client ‘S1’ and ‘S2’.

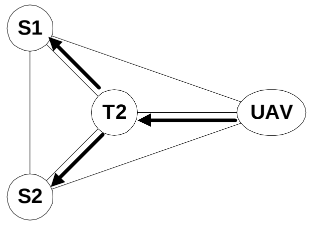
When the coordinator is inactive, that is, when the framework is not making any attempt to optimize data streams, the sensor (UAV) sends a unicast stream to each of the clients. Both streams will compete for the limited bandwidth and the delays at each client increase significantly with the reduction in bandwidth availability. These results are shown in figure 6, with their 95% confidence error margins.



	Client A	Client B
100 KBps	427 ± 6	430 ± 8
40 KBps	1230 ± 193	570 ± 23
30 KBps	2978 ± 690	4760 ± 5114
24 KBps	5354 ± 1920	8883 ± 5364

Figure 6 – Effects of bandwidth reduction without data stream coordination.

In this example, as the bandwidth availability decreases, the delays quickly increase to the point where critical tasks, such as the remote control of the UAV, are completely compromised. A minimum frame-rate of 2 fps is required[1], in this example, to safely navigate the UAV so it is clear that even small constraints in bandwidth availability can compromise this task. Furthermore, we can verify the well known bandwidth stealing behavior between clients, where bandwidth is not equally shared between streams. This behavior has been previously reported in IP networks (Tschudin and Ossipov, 2004) and could compromise critical tasks such as the control of the UAV.



	Client A	Client B
100 KBps	382 ± 4	401 ± 9
40 KBps	382 ± 4	383 ± 6
30 KBps	384 ± 9	381 ± 9
24 KBps	514 ± 20	514 ± 20

Figure 4 – Data distribution tree created by FlexFeed

When the FlexFeed coordinator is enabled, the framework identifies the stream requests and attempts to globally optimize data distribution. In this specific case, the coordinator (which, is a centralized process) determines that both streams are similar (in fact, equal) and the overall bandwidth utilization can be reduced with a multicast-like data distribution tree. The framework opportunistically identifies node ‘T2’ as a potential intermediate processing element and builds the distribution tree illustrated in figure 4.

Under the same bandwidth constraints, the framework ensures that the lowest capacity link (from the UAV) is not saturated and delays between images are kept within reasonable bounds.

Furthermore, the variance in delay (jitter) is significantly smaller, ensuring that critical processes maintain minimum levels of throughput and quality of service.

FlexFeed Overhead

The overhead of framework basically falls into two main categories: a) the number of additional control messages involved on sharing state between nodes (or between nodes and the centralized coordinator) and b) the time required to determine, locate, and configure the nodes that will participate in the data stream. Both factors are highly dependent on the type of coordination mechanism used in the framework (centralized, zone-based, or local), the complexity of the data, the scale of the network, its level of connectivity, and the frequency of state updates.

In our example, the network topology is static and variations in resource availability are small so our attention is focused primarily on the delays (latency) caused by the centralized coordination algorithm. Table 1 shows the average delays and their 95% confidence error margins observed in each test, both with and without the coordinator. The delays were measured as the average between the time of the second client ‘S2’ request and the time when the first image is delivered to that client.

	No Coordinator	Coordinator
100 KBps	978 ± 37	1845 ± 85
40 KBps	1175 ± 126	2187 ± 95
30 KBps	1184 ± 571	2330 ± 180
24 KBps	1942 ± 310	2418 ± 204

Table 1 – Average delays for starting the stream

When the coordinator is present (second column), there is an up-front cost in terms of latency that is due to the time spend in identifying and configuring network resources for data distribution. When the coordinator is not present, the response to the data request is relatively fast at first but the delays increase as the bandwidth is reduced. This is basically due to the fact that initial images are being lost on the saturated channel when the coordinator is not present.

FlexFeed versus Multicast

The data distribution tree presented in this example resembles a data multicast tree, often obtained with conventional data multicast algorithms. As previously noted, FlexFeed

[1] Although it is commonly accepted that a minimum of four frames per second is necessary to remotely operate robotic vehicles, for illustration purposes in this example, the minimum requirement for teleoperation is assumed to be two frames per second.

goes beyond conventional and data-aware multicast approaches by building a tree that include nodes that are not necessarily part of the multicast group (node T2 in this case). These nodes are opportunistically discovered and configured, at runtime by the framework, based on its current resource availability, role in the network, and system policies.

Furthermore, the configuration of node T2 can be highly data-dependent and arbitrarily complex. In these examples, the intermediate node was use merely as a splitting point for the data distribution tree. It could also have received customized code to perform in-stream data transformation or specialized filtering.

Consider the case where the request place by soldier ‘S2’ was for a lower resolution video stream from the same source. Multicast algorithms would often regard this as an independent request or would have assumed that one of the nodes in the multicast group would be able to reduce the resolution of the stream to include the new request in the data hierarchy. Data-centric protocol like Directed Diffusion would also depend on an intermediate node’s a priori capabilities to construct the low resolution data from the high resolution stream.

In FlexFeed, the request placed by soldier ‘S2’ can specify references to data-specific code that will be installed, on demand, on node ‘T2’ to act as a processing element. The code would be installed only in the necessary nodes (as determined by the coordination algorithms) and would be removed when no longer necessary.

Another extension of the same capability is the transparent enforcement of information release policies. In this case, upon S2’s data request, FlexFeed would query the policy framework for constraints or obligations involving the request. Consider, for example, that policies were previously defined to constrain unrestricted access from S2 to that specific data source. In that case the framework will, transparent to node ‘S2’, identify an intermediate node for policy enforcement and will deploy the customized data filters (specified as part of the policy) to ensure compliance with the specified requirements. This feature of Flex-Fleed has been extensively demonstrated by (Suri, Bradshaw et al, 2003; Suri, Carvalho et al, 2003) in multiple simulations and real life exercises.

CONCLUSIONS AND FUTURE WORK

In this paper we have introduced the FlexFeed framework in the context of military combat operations. The concept proposed in FlexFeed goes beyond current data-centric routing approaches and data-aware multicast. It realizes the notion of agile computing in the context of data communications and offers the basis for a truly customizable middleware for data communications in extreme environments.

The framework is currently implemented and has been tested in several small scale exercises including soldiers operating in conjunction with robotic units and remote systems. The framework currently relies on a centralized coordination algorithm for resource allocation. We are currently developing fully decentralized (and zone-based) algorithms to improve scalability, robustness, and performance.

REFERENCES

Baehni, S., Th, P. and Guerraoui, E., Data-Aware multicast. In *Proceedings of the 5th IEEE International Conference on Dependable Systems and Networks* (DSN '04), June 2004.

Bradshaw, J.M. et al.: “KAoS: Toward an Industrial-Strength Generic Agent Architecture.” Software Agents, AAAI Press/MIT Press, Cambridge, Mass. 1997, pp. 375-418.

Bradshaw, J. M., Suri, N., Breedy, M. R., Canas, A., Davis, R., Ford, K. M., Hoffman, R., Jeffers, R., Kulkarni, S., Lott, J., Reichherzer, T., & Uszok, A. (2002). Terraforming cyberspace. In D. C. Marinescu & C. Lee (Ed.), Process Coordination and Ubiquitous Computing. (pp. 165-185). Boca Raton, FL: CRC Press.

Bradshaw, J.M., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M., Acquisti, A., Benyo, B., Breedy, M., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., and Van Hoof, R. Representation and Reasoning for DAML-Based Policy and Domain Services in KAoS and NOMADS. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems* (AAMAS) 2003.

Carvalho, M. and Breedy, M. (2002) Supporting Flexible Data Feeds in Dynamic Sensor Grids Through Mobile Agents. In *Proceedings of the 6th International Conference in Mobile Agents* (MA 2002) Agents, Barcelona, Spain, October 2002.

Carvalho, M., Bertele, F., Suri, N. ULM – A Centralized Co-orindation Algorithm for FlexFeed. (to appear) In *Proceedings of the 9th World Multi-Conference on Systemics, Cybernetics and Informatics*. Florida, October 2005.

Carson, M. and Santay, D. NIST Net – A Linux-based Network Emulation Tool. 2002. Available online at: http://www-x.antd.nist.gov/nistnet/nistnet.pdf

Eugster, P.Th.,Guerraoui, R., Handurukande, S., Kermarrec, A.M., and Kouznetsov, P.. Lightweight Probabilistic Broadcast. In *Proceedings of the ACM Transactions on Computer Systems (TOCS)*, pages 341–374, november 2003.

Eugster, P.Th., Guerraoui, R., and Damm., C.H. On Objects and Events. In *Proceedings of the 16th ACM Conference on*

Object-Oriented Programming Systems, Languages and Applications(OOPSLA 2001), pages 131–146, October 2001.

Lin, M.J., and Marzullo, K.. Directional Gossip: Gossip in a Wide Area Network. In *Proceedings of the 3rd European DependableComputing Conference (EDCC)*, pages 364–379, September 1999.

Ganesh, A.J., Kermarrec, A.M., and Massouli´e, L.. Scamp: Peer-to-peer lighweight membership service for large-scale group communication. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC)*, 2001.

Groth, P. and Suri, N. Cpu resource control and accounting in the nomads mobile agent system. In *Proceedings of the ACM OOPSLA Workshop on Experiences with Autonomous Mobile Objects and Agent Based Systems*, Minneapolis, USA, Oct. 2000.

Maistre, F.. Joram. (2003) Available online at http://joram.objectweb.org; accessed on May 2005.

Skeen, D. Vitria's Publish-Subscribe Architecture: Publish-Subscribe Overview. http://www.vitria.com, 1998.

Pike, J. (2003). Future Combat Systems (FCS). Retrieved December 18, 2003 from http://www.globalsecurity.org/military/systems/ground/fcs.htm. Accessed on May 2005

Rowstron, A., Kermarrec, A.M., Castro, M., and P. Druschel. SCRIBE: The Design of a Large-Scale Event Notification Infrastructure. In *Proceedings of the 3rd International Workshop on Networked Group Communication (NGC)*, November 2001.

Ratnasamy, S., Handley, M., Karp, R., and Shenker, S., Application-Level Multicast Using Content-Addressable Networks. Lecture Notes in Computer Science, 2233:14–29, 2001.

Sharoni, A., & Bacon, L. (1997). The Future Combat System (FCS). A Technology Evolution Review and Feasibility Assessment. Armor, July-August, pp. 7-13.

Suri, N., Bradshaw, J.M., Carvalho, M., Cowin, T., Breedy, M., Groth, P., and Saavedra, R. Agile Computing: Bridging the Gap between Grid Computing and Ad-hoc Peer-to-Peer Resource Sharing. In P*roceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid* (CCGrid 2003).

Suri, N., Bradshaw, J.M., Breedy, M.R., Groth, P.T., Hill, G.A., and Jeffers, R. Strong Mobility and Fine-Grained Resource Control in NOMADS. In P*roceedings of the 2nd International Symposium on Agents Systems and Applications and the 4th International Symposium on Mobile Agents* (ASA/MA 2000). Springer-Verlag; 2000.

Suri, N., Carvalho, M., Bradshaw, J., Breedy, M., Cowin, T., Groth, P., Saavedra, R., and Uszok, A. Enforcement of communications policies in software agent systems through mobile code. In *Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 247. IEEE Computer Society, 2003.
Suri, N., Bradshaw, J., Carvalho, M., Breedy, M., Cowin, T., Saavedra, R., Kulkarni, S. Applying Agile Computing to Support Efficient and Policy-Controlled Sensor Information Feeds in the Army Future Combat Systems Environment. In P*roceedings of the U.S. Army 2003 Annual Collaborative Technology Symposiu.* 2003

Tschudin, Ch. and Ossipov, E., Estimating the Ad Hoc Horizon for TCP over IEEE 802.11 Networks, In *Proceedings of the Third Annual Mediterranean Ad Hoc Networking Workshop*, Bodrum, Turkey, June 2004.