Management and Service Discovery in Satellite and Avionic Networks

Todd Sproull and John W. Lockwood
Department of Computer Science and Engineering
Washington University in Saint Louis
St. Louis, MO 63130 USA
{todd,lockwood}@arl.wustl.edu

John Meier
Boeing Corporation
Saint Louis, MO, 63166, USA
Email:john.l.meier@boeing.com

Abstract— Command and control services manage network-attached assets deployed in distributed systems that can be separated by thousands of miles. Networks that rely on satellite communications to transit all data to a centralized control center are troubled by high latency due to long propagation delays to satellites and limited data transit over bandwidth constrained links. Low latency communications can be achieved by using a combination of distributed airborne and space-based systems. This research investigates how deployment of a Peer-to-Peer (P2P) overlay network in a region of conflict can reduce the latency for real time control and communication. This overlay network utilizes a hybrid of both satellite and aircraft links to provide services that best satisfy the immediate needs of ground units. Experiments have been performed with an emulation testbed using 147 compute nodes in the Emulab testbed to study the latency and throughput of the overlay network. The overlay network is developed using a Peer-to-Peer Application Programmers Interface (API) called JXTA. Nodes simulate resources requesting and offering several types of video and data services.

TABLE OF CONTENTS

1. INTRODUCTION

Communication links transfer data between satellites, unmanned airborne vehicles (UAVs), and devices on the ground. UAVs are used to analyze pollution, relay communications and host a variety of sensors. By providing data processing services within the nodes of the hierarchical networks, raw data can be locally and efficiently transformed into useful information. Overlay networks with *super nodes* placed strategically in the hierarchical network enables traffic to be effectively shaped and filtered.

Many centralized client/server architectures are used to process the compute intensive applications. They channel information between a centralized set of data processing and storage nodes. Often, networked platforms (UAVs, and earth orbiting satellites) are deployed thousands of miles away from a central processing center. Even with caching, client/server architectures are not well suited for network services in media intensive real time networks in dynamic mobile environments, due to changing topologies.

The latency and bandwidth constraints of long-distance communication networks are a challenge for real time media and data fusion applications. As more network devices are deployed, it becomes increasingly difficult to use a centralized processing center. Centralized architectures do not scale well to handle large volumes of information, provide robustness from failure, nor do they provide fast reaction times.

One service benefiting from distributed networks is the deployment of robots to provide medical assistance for injured soldiers. Research efforts such as the Trauma Pod [1] have investigated ways to deploy remote medical services. With *telesurgery*, surgeons perform operations on wounded soldiers using robots. Bandwidth intensive network applications such as streaming video allow surgeons to perform many life-saving operations from a remote location. The need for low latency communication is crucial to increase responsiveness to the remote surgeon during an operation.

Using distributed (rather than centralized) services to interconnect a diverse set of platforms increases scalability and real time performance. Our research investigates the tradeoffs in deploying a Peer to Peer (P2P) overlay network technology as compared to a centralized client/server approach. The architecture enables mobile devices to efficiently exchange large volumes of information using new P2P services rather than channeling all information through a central server.

This work was funded by a research grant from the Boeing Corporation.

IEEEAC paper# 1167, Version 4, Updated December 11, 2006

In heterogeneous networks, messages can flow between mobile devices on the ground, vehicles in the air, and satellites in earth orbit. Overlay networks with content-based routing services on P2P networks enhance real time decision making for multi-tiered communications. Overlay network services, such as content-based routing, to improve the Quality of Information (QoI) that flows between devices. Improved QoI enables transmission of useful information using minimal bandwidth. By using P2P services, mobile devices can communicate with less latency and bandwidth than they would using a centralized system.

Distributed services running on P2P networks enhance information rendezvous rates while reducing latency of information exchange. P2P establishes efficient overlays to enable content based routing. Overlays in P2P networks decrease routing time for advanced multi-tiered communications. The three tiers of communication (space, air, and ground) must seamlessly integrate low latency services on multiple platforms using overlays to provide scalable real time network services. Long-distance links between geosynchronous satellites and moving vehicles such as UAVs require a highly dynamic network environment.

UAVs like the *Shadow* fly at around 5000 feet while low-flyers such as the *Dragon Eye* and *Honeywell MAV* operate at 100's of feet. The bandwidth between these heterogeneous nodes scales logarithmically as the altitude increases. P2P technology is used by the highly dynamic ad hoc networks to improve reaction times in a service-oriented architecture (SOA).

Adhoc networks use overlays to improve neighbor node discovery, user authentication, and tunneling of sensitive data. Once set up, the overlay network facilitates discovery of additional nodes with minimum reaction time. The P2P API JXTA facilitates discovery of distributed services [2].

Our research utilizes distributed P2P networks with an overlay to reduce latency and maximize use of available network bandwidth. In this work, we measure the network metrics of the network latency and bandwidth as a function of the configuration of the network. We also measure an additional Measurement of Performance (MOP) to characterize the number of successful requests for P2P services. Specifically, we measure the MOPs for four services: transfer of streaming target tracking data (40 Kbps), still image transfer (100 Kbytes), streaming video (700 Kbps), and sensor query data (10 Kbyte).

We perform experiments using the emulation testbed laboratory, EMULAB [3]. Our experiments use up to 147 PCs to study link costs and we compare the overlay network against traditional client/server models of tasking resources.

The nodes distribute resource request messages using multicast communication and rendezvous nodes (also referred to as *super nodes*). There were significant challenges to interconnecting and managing a diverse set of mobile platforms, network nodes, and end systems with multicast.

A super node is a server, router, switch or other network device that has more memory, bandwidth, processing power, or better locality than other nodes in the P2P network. Super nodes reduce the need for multicast traffic and P2P chatter by serving as a rendezvous point for nodes deployed in the network.

Super nodes implemented with reconfigurable hardware, such as the Field Programmable Port Extender (FPX) [4], improve data processing services for applications in the network, enforce Quality of Service for Voice, transport Voice over IP (VoIP), and transcoded video.

The structure of this paper is now described. Section II discusses related work in the area of P2P networks. The overall system architecture is described in Section III. Section IV discusses the experiment and presents results from running on Emulab. Section V discusses future work with large testbeds and Section VI concludes our findings.

2. RELATED WORK

Today's adhoc networks support a diverse set of services that require different priorities and different allocations of bandwidth for traffic delivery. P2P topologies are scalable to meet the needs of hundreds, thousands, and even tens of thousands of users [5].

Overlay trees help P2P networks optimize the use of bandwidth by minimizing the overhead required to find peer servers [6]. In the related work of file sharing, it was found that the choice of which peer to use in the overlay had a large impact on performance. Picking the correct peer doubles the media file sharing capability in certain cases.

Simulations, such as *p-sim*, have shown how adaptive P2P topologies reduce latency in overlay links [7]. Past work focused on how the application benefited from a P2P deployment rather than measure the peer dynamics, performance of file sharing and searching, or work load of search queries.

Several network simulators provide some support for large scale P2P network experiments. P2PSim [8] is a discrete event simulator that models overlay networks such as Chord [9] and Tapestry [10]. These P2P implementations are created by P2PSim and do not model all of the features of these protocols [11]. PeerSim [12] is another example of a P2P network simulator. It provides support for super node topologies similar to our deployment strategy. Unfortunately, it does not model the network transport but, it does scale to large (1000's of nodes) networks. PeerSim utilizes it's own P2P protocols to simulate node behavior. This makes it difficult to compare to the well studied and academic P2P protocols such as [9]. As with other network simulators, both of these software

tools provide some of the functionality necessary to create realistic network experiments but, lack the flexibility and realism gained through an emulation testbed.

The use of redundant *super-peers* (also know as *super nodes*) improves the performance of P2P networks. Guidelines have been developed that suggest how to make best use of redundant nodes. Careful use of super-peer redundancy is needed to handle large aggregate processing loads at bottleneck nodes [13].

3. System Architecture

The network architecture of our system is shown in Figure 1. A high-level view of a modern ad hoc network is shown and the amount of delay associated with requesting services from a remote location is indicated by thickness of the lines.

Consider the requests of a soldier who needs to monitor a live video feed from a nearby hostile environment. Using traditional communications topologies, the request is sent to a high flying UAV which communicates with satellites and transmits the request to the remote command center potentially thousands of miles away. Once the request is granted, the video travels back to the command center and is finally retransmitted to the soldier as shown in Figure 1. This transmission path delays video to the user and increases bandwidth consumed in the network.

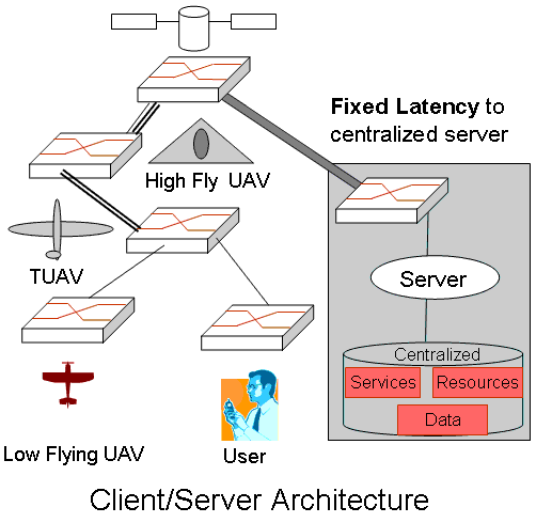

Figure 1. High level view of services in current ad hoc networks

Instead of using a centralized set of resources physically located farther away from the edge, the new architecture proposed in this paper (shown in Figure 2) uses a local set of services. We push the services and discovery mechanisms into the network. Unlike the traditional topology, the video request is routed from the UAV to the destination service. This approach eliminates large latency delays and bandwidth costs associated with providing these services to a remote location.

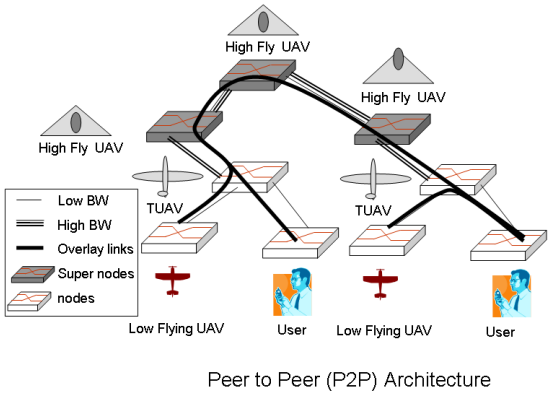

Figure 2. High level view of deploying services in ad hoc networks

Network Architecture

We created a model of a node to realistically emulate the process of discovering and requesting services in a delay and bandwidth-constrained network. The model consists of a Java application executing on a node in a testbed. Two different architectures are evaluated to facilitate the discovery and advertisement of services. The first architecture is a client/server and the second is a P2P architecture.

Client/Server Architecture—The client/server approach mimics communication in traditional ad hoc networks. Most communication passes through a central server or group of servers hosting services that route traffic to destinations. Clients register with the server before requesting services. During the registration process, clients transmit their physical location, unique node ID, and their IP address to the server. The server then routes the requests to the appropriate resources.

Peer to Peer— The P2P architecture uses an unstructured P2P topology built with the JXTA API. This API provides a mechanism for the discovery and advertisement of services and creates a secure overlay group of nodes. An initial implementation utilizes multicast to communicate between the nodes. An enhancement to this overlay involves the use of *rendezvous nodes* (super nodes). The use of multicast can lead to a large increase in network traffic. Super nodes, however, create structure in the overlay that reduces traffic. Regular nodes connect to super nodes to forward service advertisements and route service requests. Super nodes create a hierarchical topology by communicating amongst themselves and with the regular nodes directly connected to them. Nodes bootstrap by either broadcasting to discover the P2P group topology or by communicating with an initialization node. After bootstrapping, the node then attempts to join the group. Once authorized to join, the new node announces its physical location and it announces the services that it offers. Requests for services are then issued by the node through multicast or communication to a super node. If the service is found, the nodes establish an IP socket connection to transfer the data.

Client/Server with Caching— One optimization for the client/server architecture is the use of caching node advertisements in the network itself. This is similar, though with reduced functionality, to the use of super nodes. A caching node would provide some benefit to the client/server architecture when nodes are static. However, we envision nodes continuously moving from location in the grid to the next thus invalidating the cached values. The P2P approaches work well is this scenario by publishing the service advertisements to a super when the node relocates. If the client/server architecture sent update information to a caching node and queried it directly we would argue that it is in effect a P2P architecture and not a client/server with caching.

Node Architecture

Services Offered— We model a node that uses four types of services. The first type of service is for a high bandwidth, constant bit-rate, User Datagram Protocol (UDP) video stream that has a bandwidth of approximately 700 kbps. The second service is a low-bit rate service that uses UDP to send coordinate and sensor information with a bandwidth of approximately 40 kbps. The third service models an aerial camera which transmits 100kbyte images using the Transmission Control Protocol (TCP). The final service transmits 10 kbyte sensor queries using the TCP protocol. Nodes randomly select a service based on the distribution listed in Table 1.

Video	Track	Image	Sensor	Idle
10%	45%	10%	25%	10%

Table 1. Distribution of services for each node

Implementation of Nodes in the Overlay Network

In our experiments, nodes both request for and provide services. All nodes request and offer services for a fixed amount of time. A new service is requested once the previous service completes or times out. After completing a service, nodes either request another service or remain idle for one second before issuing another request. When the time for an experiment expires, the node completes all currently active services before exiting the overlay.

Nodes are assigned an initial physical location in a grid with specific coordinates on an (x,y) grid. Services are requested from and to specific locations. For example, a node at overlay position (32,53) might request a video stream from location (27,92).

Overlay Software—The software that establishes the overlay network was written in Java using approximately 2000 lines of code. The client/server portion of the code utilized Java Sockets for all communication. The P2P portion was implemented with JXTA 2.3.5 and unidirectional JXTA pipes to send and receive messages. In both implementations, each node created multiple Java threads to concurrently request and respond to services. Service locations were randomly distributed and the type of service requested is based on the distribution listed in the Table 1.

4. EXPERIMENTATION

Experiment Setup

In order to emulate a multi-tiered communication network, a large testbed was needed. Emulab was chosen since it is the one of largest academic testbed available. Emulab allows machines to be allocated, a network to be created, and experiments to be conducted in a way that is reproducible. The current testbed consists of 365 PCs, of which a subset of nodes can be allocated to perform experiments. Emulab provides a web interface to configure experiments and allows for administrative control of each node. After an experiment is run, a script is executed to collect statistics about the operation of the experiment and to report the number of successful service transactions, average latency per service and bandwidth utilization. Each experiment lasted approximately 15 minutes.

Two different types of topologies were deployed using Emulab. The first was a star topology with each node connecting to a central switch. The star topology was used to investigate how well the applications perform in an idealized environment. All of the experiments with this topology utilized the pc3000s Emulab nodes which are 3GHz, 64-bit Xeon processors equipped with 2 GBytes of RAM. This equipment minimizes the effects of the computing hardware relative to the network under test.

The second topology is hierarchical configuration with varying link delays and bandwidth constraints. The hierarchical network provides more a more realistic deployment scenario with models for different types of nodes requesting services at various rates. Due to the size these experiments a mix of Emulab hosts were deployed ranging from Pentium 3 850MHz PCs to the 3GHz Xeon nodes. In general, the fastest nodes available were deployed, giving a higher priority to assigning the server and super nodes with the most capable machines.

Effects of Latency and Bandwidth

Several experiments were conducted to measure how latency and bandwidth constraints affected the performance of the client/server architecture. The experiments measured performance in terms of the number of successfully completed service operations. In order for a node to complete a successful service, it must locate the service, request use of the service, and finally transfer the data associated with the service.

Bottleneck Link to the Server—The first experiment measured the total number of services completed as a function of increasing latency between the performance-critical connection to the server. This experiment used a star topology of 11 nodes (10 overlay nodes plus 1 server node) configured with a fixed, 100ms latency between nodes and variable latency to the server, as displayed in Figure 3. Figure 4 shows how

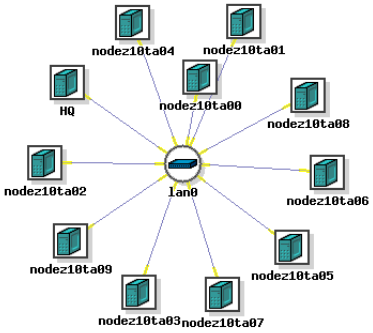

Figure 3. View of the 11 node star topology

the number of successful service requests decreased as the latency increased. The P2P architecture, which does not utilize the bottleneck link, performs better than the client/server architecture when the delay constrained server link becomes large. We found that once the delay to the server exceeded approximately 200ms, the P2P architecture delivers more services than the client/server architecture. In the client/server architecture, all communication is routed from a client to the server then to another client. The performance of the client-server architecture depends on the proximity of the clients to the server as well as the bandwidth of the links. This limits the scalability of the client/server architecture.

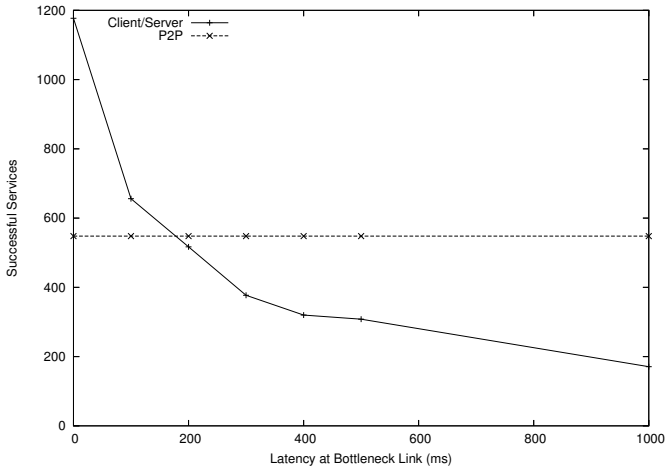

Figure 4. Number of successful services for the client/server architecture as bottleneck link increases. The 1 super node P2P architecture is also shown as a reference

In the next experiment, we deployed the same 11 node star topology as before. However, we set the propagation delay of the bottleneck link to a constant then varied the bandwidth. This experiment allowed us to parameterize link bandwidth for a variety of client/server architectures with a P2P approach using a fixed latency (50ms) on every link. From Figure 5, we observe that the P2P architecture completed more services in a fixed period of time then the client/server once the bandwidth to the server dropped below 50 Mbits/sec.

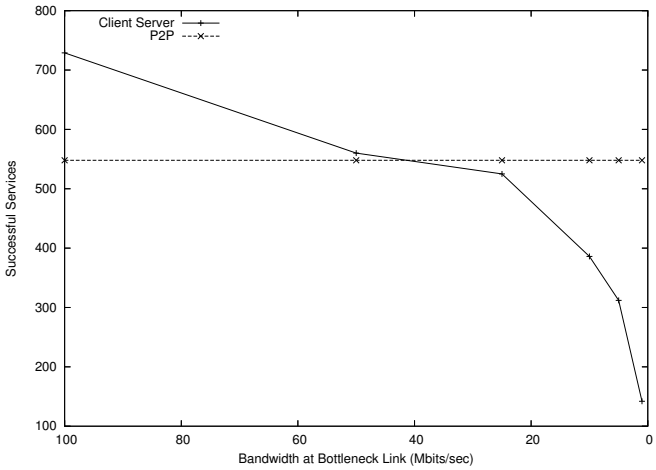

Figure 5. Number of successful services as the bandwidth is reduced for the client/server architecture on the bandwidth constrained link. The 1 super node architecture is provided as a reference

Overhead associated with P2P API

A P2P solution adds additional overhead when compared to a client/server architecture for discovery and communication of services. This section describes the amount of overhead that is inherent to the P2P architecture. The P2P overhead is calculated per service by the use of a separate port used for all P2P communication. The first experiment utilized three nodes to calculate the per service overhead from a sender to a receiver communicating through a Super Node. All communication between the sender and receiver was captured using *tcpdump*. Filters were applied to the output of tcpdump to analyze traffic by IP address and port number. The per service overhead includes the overhead introduced by JXTA using XML messages to establish a handshake between two nodes and push out the service. Table 2 lists the percentage of traffic that consists of service, and the overhead, with the rest consisting of background traffic on the LAN. In this table, the 700 Kbit/sec service creates more total traffic than the 40 Kbit/sec service which accounts for the difference in percentage of overhead traffic.

Service Type	Service %	Overhead %
700 Kbps UDP Video Stream	94.5 %	5.0 %
40 Kbps UDP Track Stream	51.7 %	48.3 %
100 Kbyte TCP Image Transfer	96.3 %	3.7 %
10 Kbyte TCP Sensor Reading	71.7 %	28.3 %

Table 2. Percentages of traffic associated with the service and the P2P overhead in terms of total bandwidth

Table 3 presents the amount of overhead traffic per successful service. Here the TCP and UDP services require roughly the same amount of overhead traffic for the P2P service requests and discovery, which is what we would expect. From the table, in order to request a 100 Kbyte TCP Image, an additional

41 Kbytes is necessary to discover the service in the overlay network and setup communication between the sender and receiver.

Service	Overhead (Kbytes)
700 Kbps UDP Video Stream	48
40 Kbps UDP Track Stream	48
100 Kbyte TCP Image Transfer	41
10 Kbyte TCP Sensor Reading	40

Table 3. Average overhead in bytes per successful service

Network Scalability

Larger experiments were performed to evaluate how well the P2P and client/server architectures scale. These experiments evaluated the many successful services completed within a fixed period of time as a function of network topologies which had differing sizes. The latency per service and bandwidth utilized per node were measured. In these experiments, a star topology was utilized as a reference that had a fixed latency and bandwidth between each node.

We conducted experiments using 11, 26, and 51-node configurations. The topologies required 17, 39, and 77 Emulab nodes, respectively. Traffic is routed through additional PCs to emulate the desired link characteristics for latency and bandwidth. Each network link incurred a latency of 50ms between the node and central switch. The link to the remote server was assigned a delay of 125ms. This latency modeled the penalty for accessing a distantly remote server in the client/server architecture. The bandwidth for each link was set to 100 Mbit/sec.

Several different types of architectures were explored in this scenario. The first architecture used a client/server approach. The next four architectures used a P2P overlay. The first P2P overlay used multicast, the next two used one and two super nodes, and finally we deployed a configuration with one super node in addition to multicast.

Figure 6 reports the number of successful services completed as the topologies increase in size. The number of successful services was computed as the sum of total successful services completed on each node. The approach with only the super node provides the best performance for the larger node experiments. Using multicast with one super node performs fairly well for the small to medium sized experiments. A multicast only approach is useful with the smaller sized nodes, but as the network increases in size, the performance starts to decline because of the large amounts of traffic created on the network. The client/server architecture performs worse than the P2P approaches.

An additional metric to evaluate the different architectures is measuring the amount of traffic created on the network for each experiment. This is measured in terms of total traffic

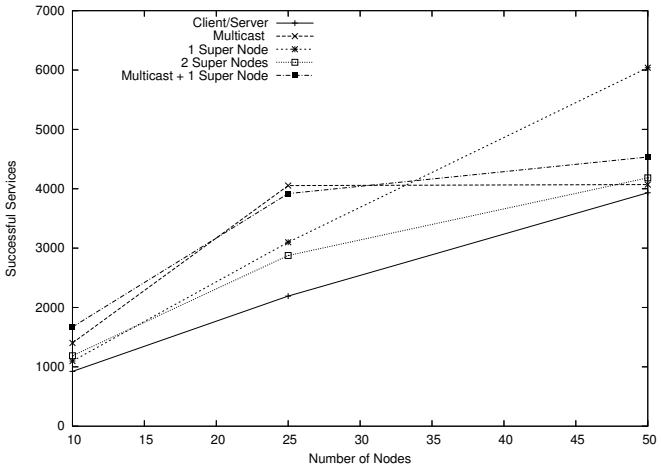

Figure 6. Successful number of services as the star topology increased in size. The multicast architecture outperforms the other approaches, with the combined super node and multicast configuration leading the remaining options

(Mbytes) and traffic per successful service (Mbytes/service). Figure 7 illustrates the total traffic generated by each experiment for a given topology. The results were obtained from reading switch counters before and after each experiment. From the figure, the multicast P2P approach creates the largest amount of traffic with increasing node sizes. The super node and multicast combination generates the second highest amount of traffic. This is no surprise due to the simple star topology and multicast sending service queries to each node, essentially broadcasting in this configuration.

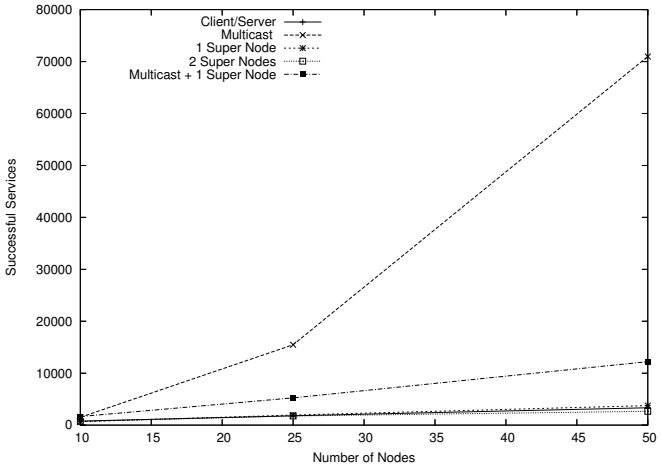

Figure 7. Total network traffic as the star topology increased in size with the multicast configuration demonstrating how poorly it scales in a star topology with larger number of nodes

The traffic per successful service is shown in Figure 8. Again, the multicast only approach is the most expensive in terms of bandwidth, requiring over 17 Mbytes per service completed. The super node P2P and client/server experiments require around 1 Mbyte per service.

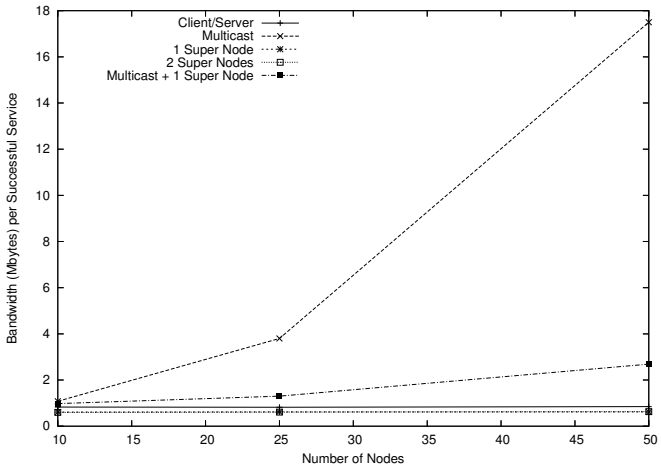

Figure 8. Network traffic per successful service as the star topology increased in size, with the multicast configuration providing the most expensive service per megabyte solution

The next statistics reported are the latencies associated with each service. Figures 9, 10, 11, and 12 measured in milliseconds the latency associated with each service. The latency for the UDP streams was measured from the time that the client requested the stream to the time when the client received the first byte of data. The TCP latency was measured as the time when the data transfer was complete.

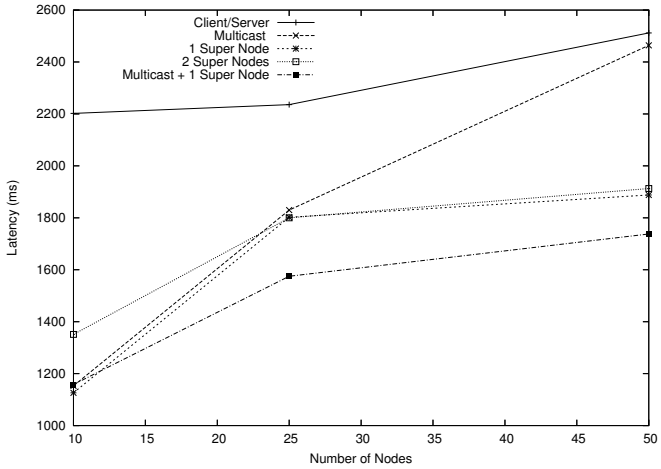

Figure 9. 700 Kbps UDP Stream Latency as the star topology increased in size. The client/server and multicast configurations do not scale well with larger sized topologies

The latency experiments can be divided into two groups, the UDP services and the TCP services. The multicast architecture was actually slower than the client/server in the 40 Kbps UDP stream. This is due to the amount of traffic generated from each node searching every node in the overlay for a particular service. Deploying super nodes eliminates that problem by caching service advertisements for nodes utilizing that super node. In the TCP services the latencies for the P2P approaches were around 2-3 times faster than the client/server model. With the UDP stream services offering around a 30%

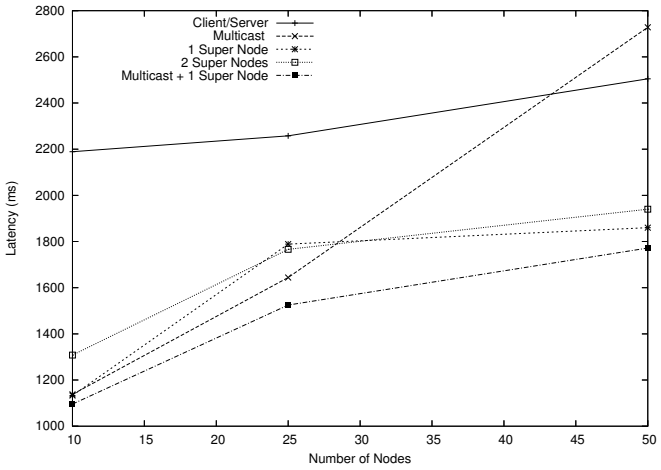

Figure 10. 40 Kbps UDP Stream Latency as the star topology increased in size, again the multicast configuration demonstrates a sharply raising latency for the larger topologies

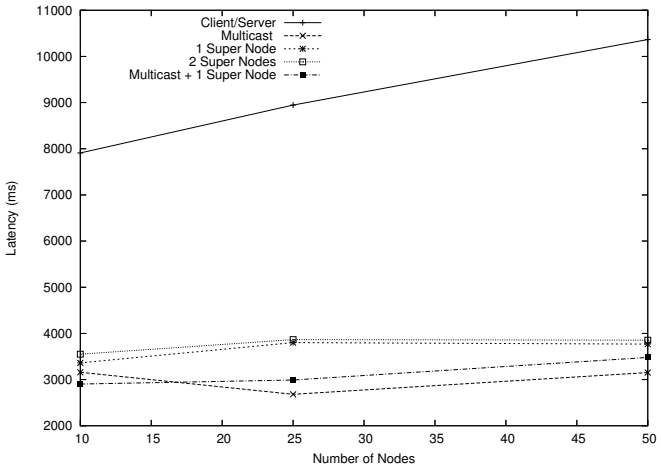

Figure 11. 100 Kbyte TCP Transfer Latency as the star topology increased in size, with the client/server configuration unable to scale as well as the P2P architectures

decrease in latency to discover the service. Excluding the multicast case, as networks grow larger than 90 nodes both types of services will benefit even more from the P2P architecture, especially applications transferring large amounts of data.

Hierarchical Network of Super Nodes

This section explores experiments deployed using a hierarchical network topology. In this tree topology, super nodes were placed at various locations near the root of the tree. This network includes a range of bandwidths and link delays with 1 Mbps links on the low flying nodes, 10 Mbps at the Tactical UAVs and 100 Mbps between high flying (X-45) nodes [14]. Delays between links are fixed at 40ms, 40ms, and 20ms for the Low Flying UAV, Tactical UAV and X-45 respectively. The delay link to the server in the client/server architecture

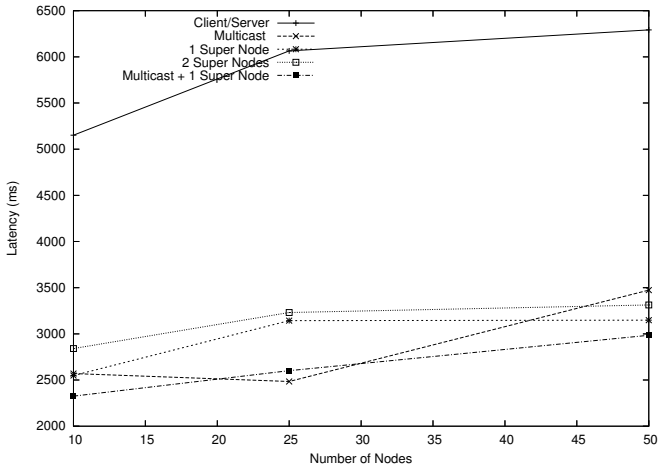

Figure 12. 10 Kbyte TCP Transfer Latency as the star topology increased in size. The client/server performs better in this smaller file transfer size, however not at the smaller latencies of the P2P architectures

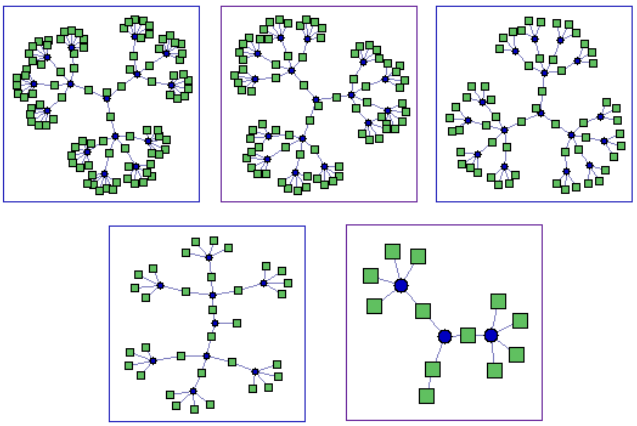

Figure 13. High level view of 92, 75, 54, 32, and 11 node hierarchical topologies

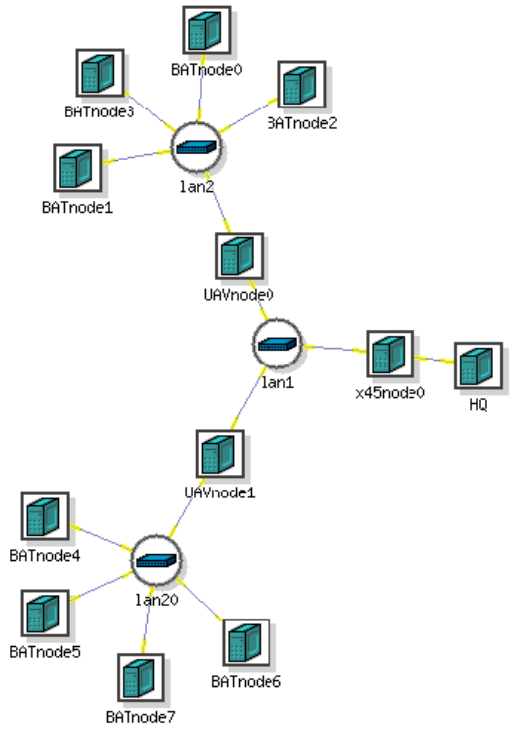

Figure 14. View of 11 node hierarchical topology

was set to 600ms. The 600ms delay is a result of the propagation time and queuing that takes place over a multihop satellite link or a wireless to ground infrastructure similar to [15]. The two common methods currently used to route video to a centralized remote set of processors are directly through a satellite (Figure 1) or relayed to a ground station completing the path through the internet. The latency experienced by the satellite is usually greater than 600 ms because the propagation delay to a geostationary satellite (250 ms), the relay and switching delay to a secondary satellite (250 ms) plus the jitter (100 ms) from the multiplexing and encoding comprise the 600 ms latency. The jitter is due to the multiplexer, modulator, coder, switch, decoder, demodulator and demultiplexer. The use of technology such as Turbo Code provide substantial improvement in error correction however increases jitter due to the large block size required during encoding and decoding. The hierarchical topologies consisted of 11, 31, 54, 75, and 92 nodes in the overlay. The total number of Emulab hosts required to support these experiments ranged from 19 PCs in the 11 node example up to 147 in the 92 node example. Again, this large increase is due to the additional nodes responsible for bandwidth and delay constraints placed between links.

Three different distribution types are simulated at various levels of the hierarchy. Table 2 lists the assumed distribution of services.

In order to better exploit the locality of the services, an assumption is made regarding the types of requests issued by the Low Flying UAV nodes. This assumption is that requests are only issued to nodes one hop away, or in the same subnet. This assumption is fairly reasonable given the fact that services are most valuable to the nodes closest to them. No restriction is placed on the Tactical and X-45 nodes, permitting requests for any node in the topology. These assumptions allow the low flying nodes near regions of interest access to those important services and a global service request scheme for the high flying nodes (Tactical and X-45).

In these experiments multicast was not deployed. With multicast enabled, nodes between routers are unable to communicate with each other in Emulab. Instead, three super nodes are deployed to investigate the benefits of increasing the number of super nodes.

Results—This section presents the results of the hierarchical topologies. Figures 15, 16, 17 report the successes, band-

	video	track	image	sensor	idle
Low Flying UAV	10%	45%	10%	25%	10%
Tactical UAV	25%	25%	25%	5%	20%
X-45	40%	5%	20%	5%	30%

Table 4. Distribution of different nodes in hierarchical topology

width, and bandwidth per service respectively. Figures 18, 19, 20, and 21 report the individual service latencies.

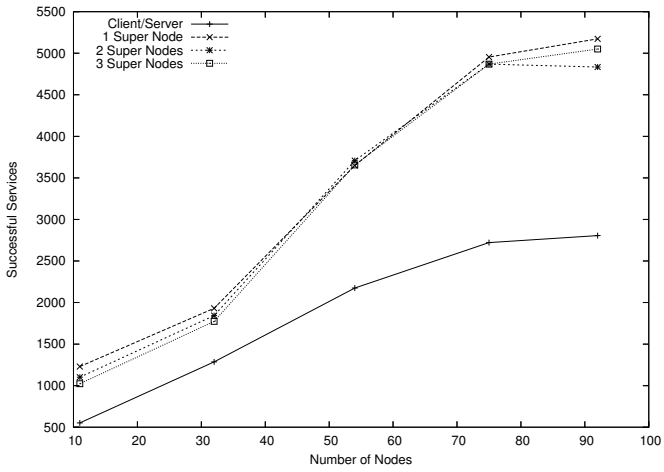

Figure 15. Successful number of services for localized communication hierarchical topologies

The number of successful services scales well in P2P architectures. The P2P nodes perform almost twice as many services as the client/server in the largest experiment. The client/server experiences an initial decrease in bandwidth per service due to the difference in topologies between the 11, 32, and 54 node experiments. The 54, 75, and 92 node experiments share a similar structure with an increase in total nodes at the edges.

The P2P architectures generate more total traffic as the topologies increase in size, however the number of services completed is also greater. It is important to note the rising costs of services per MByte in the client/server model in Figure 17 with larger experiments. The costs for the one super node example remains fairly stable even for larger experiments which is very encouraging for building larger systems. The bandwidths fluctuate for the two and three super node examples in the larger node topologies, however the costs are always considerably less than the client/server case.

The client/server model performs at 2-5 times the latency of the P2P architectures, depending on topology size and the service. In the 10KByte TCP transfer, Figure 21 the client/server approach increases by 17% moving 75 to 92 nodes, compared to the 1% increase experienced by the single super node P2P example.

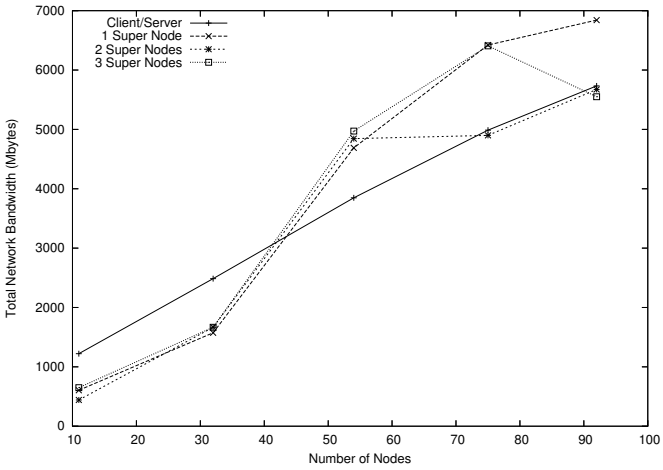

Figure 16. Total network traffic for hierarchical topologies

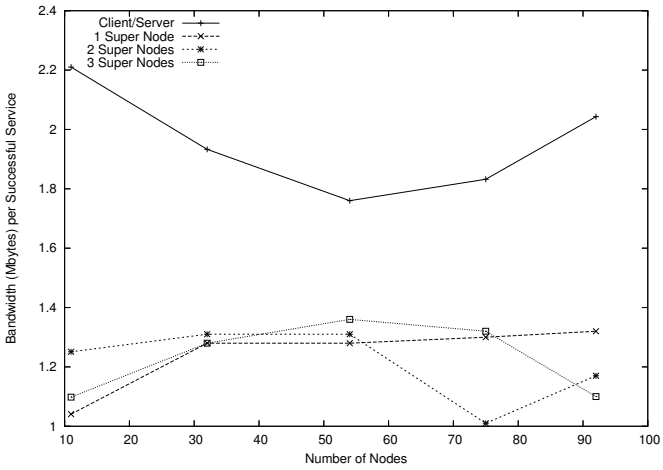

Figure 17. Network traffic per successful service for hierarchical topologies. The client/server approach requires more bandwidth for all sized topologies with a rising trend in the largest experiments

Use of Super Nodes

From the results, the addition of more than one super node does not necessarily improve performance on all topologies. Strategic location of the super node will impact the effectiveness of the P2P technology. For the larger experiments though, performance improvements were demonstrated when slower nodes were deployed in Emulab. For example, in the 92 node hierarchical, using a Pentium 3 850MHz PC proved inadequate acting as a super node for the other 91 nodes in the overlay. The bottleneck was simply the JAVA application consuming 99% of the CPU due to the overhead associated with handling service requests for a large number of nodes. When deploying additional super nodes to divide the load however, the number of successful experiments increased considerably.

The second benefit provided using additional super nodes is redundancy. When a super node fails in the P2P examples,

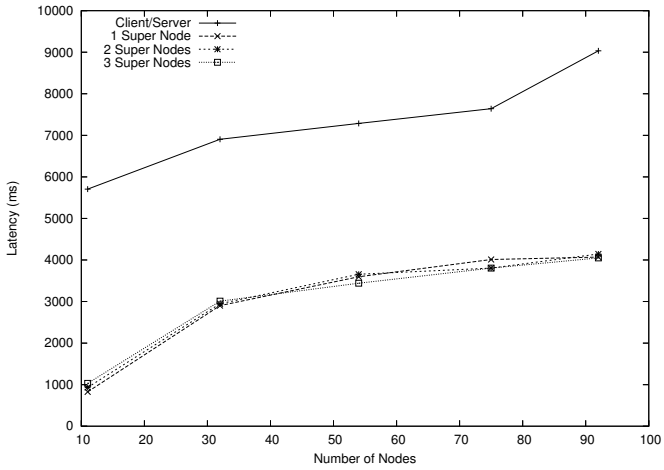

Figure 18. 700Kbps UDP Stream Latency for hierarchical topologies. The P2P architectures discover services at least twice as fast as the client/server

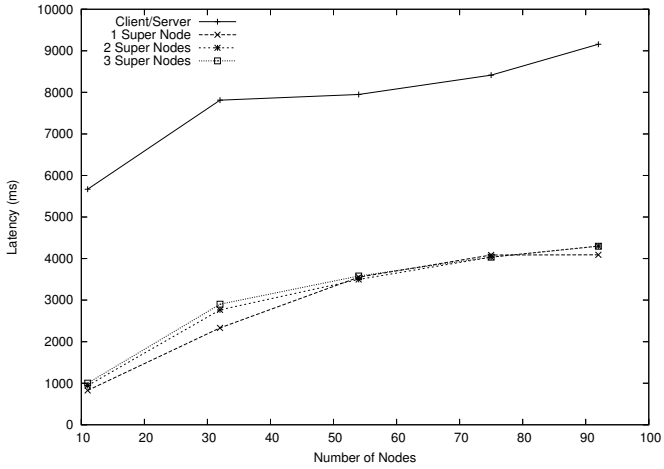

Figure 19. 40Kbps UDP Stream Latency for hierarchical topologies. The P2P architectures service latency outperforms the client/server in requesting the UDP data stream

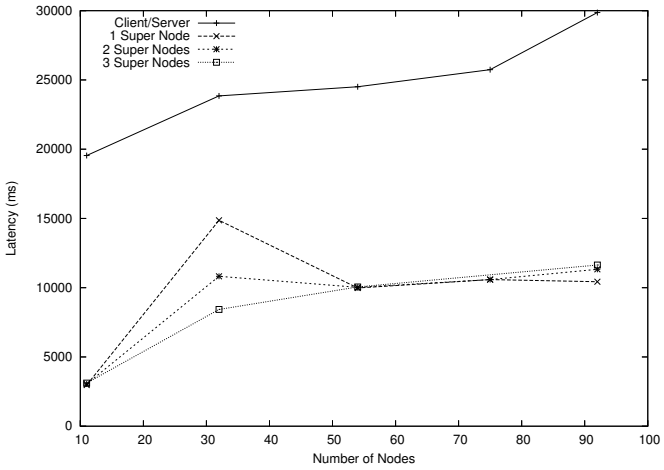

Figure 20. 100 Kbyte TCP Transfer Latency for hierarchical topologies. The client/server architecture continues to rise at an increasing rate for the largest experiments

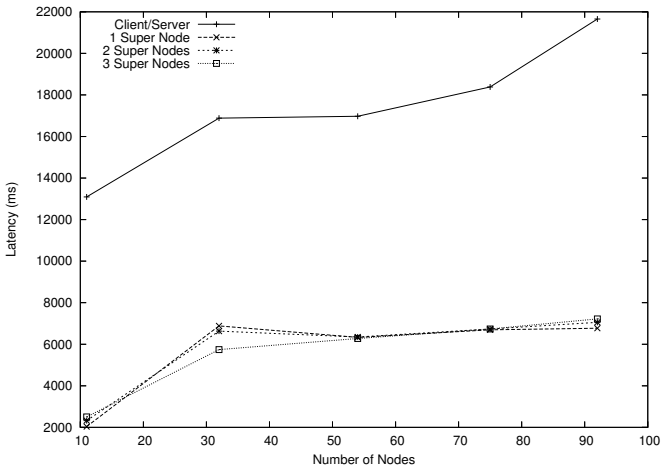

Figure 21. 10 Kbyte TCP Transfer Latency for hierarchical topologies with the P2P latencies scaling very well with larger topologies

the additional super nodes continue to operate to provide services. Each additional super node is capable of distributing resolving queries for every node in the network. In the client/server model, a failed server or bottleneck link will completely disrupt the use of services. Initial placement of the super nodes indicates two hierarchial levels from the edge is optimal.

Benefits of Emulation

Current P2P network simulators [8][12] lack the realism found in an actual implementation of all the P2P protocol's unique behavior. For example, with emulation, we are able to evaluate with greater confidence than a simulator, the latency required to complete a service. Also, with emulation more practical issues are exposed such as the amount of CPU processing necessary for a super node for support 100 nodes in a distributed environment.

One early result which came from the emulation and arguably would have came from a JXTA P2P network simulator (if it existed) was the importance of locality in nodes requesting services. When an arbitrary node requesting from any given neighbor the latency was almost as large as the client/server architecture. Making the assumption that most low flying nodes are generally interested in services from nearby nodes substantially increased the number of services completed and decreased the per service latency.

5. FUTURE WORK

In order to provide more realistic experiments with heterogeneous nodes, we plan to run experiments on the Open Network Laboratory (ONL). ONL contains Field Programmable Port Extenders (FPXs) [4] and Smart Port Cards (SPCs) [16] that perform router plug-in functions. Integrating the ONL [17] programmable hardware into larger scaled net-

work testbed simulations such as Emulab and Deter [18], is also planned to assess scalability and security of the network. ONL experiments will measure improvement to efficiency using redundant super nodes. Distributed network management schemes will also be evaluated to program network hardware, select the queuing strategies, and dynamically adjust policies based on real time monitoring of the network conditions in a live demonstration.

We will investigate QoS deployed throughout the network in both ONL and Emulab. In ONL hardware and software QoS will be deployed as an underlying network service to support higher priority traffic flows. Tradeoffs will investigate the effects of deploying the service in software or hardware and the compute resources required to provide various levels of service. The intersection of QoS policy management and security policy management will be explored using reconfigurable computing. With QoS deployed in key locations of the network, dynamic policy management will provide greater utilization of high priority resources.

Other future work will investigate adding more super nodes closer to the highest concentration of nodes. Also, we will deploy multicast when communication is most likely to occur within a small group. Tradeoffs exists between traversing long latency paths to discover services versus sending out a multicast request when the service is nearby. Ultimately, using the anticipated location of a service provider to decide wether to query a super node or issue a multicast request.

Additional considerations for loss within a wireless network will also be explored. Currently, packet losses are fairly infrequent due to the reliable links. However, Emulab does offer drop link probabilities for these links. It is expected that both the client/server and P2P architectures performance will degrade with these modified links. The client/server architecture will be impacted the most though, due to the additional hops to the communicate with the server.

Finally, we will investigate the placement and behavior super nodes to reduce the use of multicast, improve response time, reduce overhead, and increase the success rate for services in a mobile environment. We intend to use P2P technology to help create a deterministic, hardware and software infrastructure that is tolerant to faults and optimized for improved communications. Super nodes are used to establish more efficient coordination and communication between nodes. Super nodes implemented with reconfigurable hardware can also perform improved data processing services for applications in the network to enforce better QoS. We intend to validate our operational concepts using ONL programmable hardware in our scalable network testbed simulations. The software solution will be evaluated using wireless mobile ad hoc hardware in a real environment to verify the emulated results.

6. CONCLUSIONS

A continuum exists between client/server and peer-to-peer architectures. By deploying super nodes that communicate with P2P protocols, it is possible to reduce latency and increase the number of services supported in a multi-tiered network. Ad hoc networks need network wide services that discover neighboring nodes, advertise services, and authenticate nodes. We compare how applications using these services run using a peer-to-peer overlay network versus a traditional client/server architecture. Pure client/server architectures have poorer performance for very large distributed networks as compared to the P2P architecture. Even a caching client/server model would lack the ability to perform well in the presence of mobile nodes.

The instantiation of multiple services throughout the network reduce path delays between clients in an overlay network. Distributed services dramatically improve the performance of the geographically diverse set of resources (nodes). Lower latency access over limited bandwidth links to the requested service improves the success rate and reduces the overhead. A 2-5x reduction in latency, using with half the amount of bandwidth per service was shown with EMULAB.

Experiments performed using EMULAB demonstrated that P2P networks with super nodes can achieve provided excellent MOPs using real hardware and software. We assessed the improvement in effectiveness using link metrics mentioned above and service metrics. More dynamic, intelligent bandwidth management using distributed services can program network hardware, arbitrate multi-level security, select queuing strategies and dynamically adjust policies. Optimal use of limited bandwidth requires using real time monitoring of the network conditions.

P2P architectures enhance service distribution in ad hoc networks by reducing latency and improving bandwidth utilization. Even with a relatively small number of nodes, a P2P approach is able to provide more services than the client/server architecture as latency increases and bandwidth decreases. As the size of the network increased, the P2P approach provides almost twice as many services in the same amount of time as compared to the client/server. In addition to the increased number of services, the per service bandwidth and latency costs are also reduced. The baseline client/server performance provided a good metric to determine the value of inserting P2P technology in mobile ad hoc networks. Through distributed services and super nodes, networks of the future can be more effectively utilized and service more requests in less time.

REFERENCES

[1] "Trauma pod." `http://www.darpa.mil/DSO/-thrust/biosci/traumpod.htm`.

[2] L. Gong, "Jxta: A network programming environment," *IEEE Internet Computing*, vol. 5, no. 3, pp. 88–95,

2001.

[3] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, (Boston, MA), pp. 255–270, USENIX Association, Dec. 2002.

[4] J. W. Lockwood, "An open platform for development of network processing modules in reprogrammable hardware," in *IEC DesignCon'01*, (Santa Clara, CA), pp. WB–19, Jan. 2001.

[5] B. Doshi, L. Benmohamed, and A. DeSimone, "A hybrid end-to-end qos architecture for heterogeneous networks (like the global information grid)," in *Military Communications Conference*, Oct. 2005.

[6] T. Ng, Y. Chu, S. Rao, K. Sripanidkulchai, and H. Zhang, "Measurementbased optimization techniques for bandwidth-demanding peer-to-peer systems," 2003.

[7] S. Merugu, S. Srinivasan, and E. Zegura, "p-sim: A simulator for peer-to-peer networks," *mascots*, vol. 00, p. 213, 2003.

[8] "P2psim." `http://pdos.csail.mit.edu/-p2psim/`.

[9] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," 2001.

[10] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz, "Tapestry: A resilient global-scale overlay for service deployment," 2003.

[11] "Tools for peer-to-peer network simulation." `http://tools.ietf.-org/irtf/draft-irtf-p2prg-core-simulators-00.txt`.

[12] "Peersim." `http://peersim.sourceforge.-net/`.

[13] B. Yang and H. Garcia-Molina, "Designing a super-peer network," *icde*, vol. 00, p. 49, 2003.

[14] J. Montgomery, "The orbiting internet fiber in the sky," *BYTE*, vol. 22, no. 11, pp. 58–72, 1997.

[15] G. Xylomenos, G. Polyzos, P. Mahonen, and M. Saaranen, "Tcp performance issues over wireless links," *IEEE Communications Magazine*, pp. 52–58, Apr. 2001.

[16] J. D. DeHart, W. D. Richard, E. W. Spitznagel, and D. E. Taylor, "The smart port card: An embedded Unix processor architecture for network management and active networking," Tech. Rep. WUCS-01-18, Applied Research Laboratory, Department of Computer Science, Washington University in Saint Louis, August 2001.

[17] J. DeHart, F. Kuhns, J. Parwatikar, J. Turner, C. Wiseman, and K. Wong, "The open network laboratory," in *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, (New York, NY, USA), pp. 107–111, ACM Press, 2006.

[18] T. Benzel, R. Braden, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, "Experience with deter: A testbed for security research," in *Conference on testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom2006)*, (Barcelona,Spain), March 2006.

Todd Sproull *is a Doctoral Candidate pursuing his DSc in Computer Engineering at Washington University in Saint Louis. He is a Research Assistant and member of the Reconfigurable Network Group (RNG) at Washington University. His interests include peer-to-peer overlay networks, reconfigurable computing and network security. He has worked in the industry at Xilinx Research Labs, Network Physics, and IBM. He earned a BS degree in Electrical Engineering at Southern Illinois University in Edwardsville and an MS in Computer Engineering at Washington University. He is a member of IEEE, Tau Beta Pi and Eta Kappa Nu.*

John W. Lockwood *designs and implements networking systems in reconfigurable hardware. He leads the Reconfigurable Network Group (RNG) at Washington University. The RNG research group developed the Field programmable Port Extender (FPX) to enable rapid prototype of extensible network modules in Field Programmable Gate Array (FPGA) technology. He is an Associate professor in the Department of Computer Science and Engineering at Washington University in Saint Louis. He has published over 75 full-length papers in journals and major technical conferences that describe technologies for providing extensible network services in wireless LANs and in high-speed networks. Professor Lockwood has served as the principal investigator on grants from the National Science Foundation, Xilinx, Altera, Nortel Networks, Rockwell Collins, and Boeing. He has worked in industry for AT&T Bell Laboratories, IBM, Science Applications International Corporation (SAIC), and the National Center for Supercomputing Applications (NCSA). He served as a co-founder of Global Velocity, a networking startup company focused on high-speed data security. Dr. Lockwood earned his MS, BS, and PhD degrees from the Department of Electrical and Computer Engineering at the University of Illinois. He is a member of IEEE, ACM, Tau Beta Pi, and Eta Kappa Nu.*

John Meier *is a Boeing Technical Fellow in the Network Centric Thrust at Phantom Works, NCO Thrust. He has over 27 years of professional experience in avionic technology development specifically working in the area of intelligent networking, reconfigurable computing architectures and wireless network management. At Boeing, he is currently involved with several key technical activities including a major project on edge computing and intelligent distributed system management. Mr. Meier earned his BS from Southern Illinois University - Carbondale (SIU-C), MS University of Missouri - Rolla (UMR), and currently working on his PhD degrees from the Department Computer Science Engineering (CSE) at the Washington University in St. Louis. He is a member of IEEE and Tau Beta Pi.*