COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION

PROGRAM ANNOUNCEMENT/SOLICITATION NO./CLOSING DATE/if not in response to a program announcement/solicitation enter NSF 04-2

PD 04-2876 **12/04/03**

FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) (Indicate the most specific unit known, i.e. program, division, etc.)

CNS - DISTRIBUTED SYSTEMS

FOR NSF USE ONLY

NSF PROPOSAL NUMBER

0411098

DATE RECEIVED	NUMBER OF COPIES	DIVISION ASSIGNED	FUND CODE	DUNS# (Data Universal Numbering System)	FILE LOCATION
				111111111	

EMPLOYER IDENTIFICATION NUMBER (EIN) OR TAXPAYER IDENTIFICATION NUMBER (TIN)

591961248

SHOW PREVIOUS AWARD NO. IF THIS IS
☐ A RENEWAL
☐ AN ACCOMPLISHMENT-BASED RENEWAL

IS THIS PROPOSAL BEING SUBMITTED TO ANOTHER FEDERAL AGENCY? YES ☐ NO ☒ IF YES, LIST ACRONYM(S)

NAME OF ORGANIZATION TO WHICH AWARD SHOULD BE MADE

Florida State University

AWARDEE ORGANIZATION CODE (IF KNOWN)

0014894000

ADDRESS OF AWARDEE ORGANIZATION, INCLUDING 9 DIGIT ZIP CODE

Florida State University
Tallahassee, FL. 323064166

NAME OF PERFORMING ORGANIZATION, IF DIFFERENT FROM ABOVE

PERFORMING ORGANIZATION CODE (IF KNOWN)

ADDRESS OF PERFORMING ORGANIZATION, IF DIFFERENT, INCLUDING 9 DIGIT ZIP CODE

IS AWARDEE ORGANIZATION (Check All That Apply) (See GPG II.C For Definitions)
☐ SMALL BUSINESS ☐ MINORITY BUSINESS
☐ FOR-PROFIT ORGANIZATION ☐ WOMAN-OWNED BUSINESS

☐ IF THIS IS A PRELIMINARY PROPOSAL THEN CHECK HERE

TITLE OF PROPOSED PROJECT **Anemone - An Adaptive Network Memory Engine**

REQUESTED AMOUNT	PROPOSED DURATION (1-60 MONTHS)	REQUESTED STARTING DATE	SHOW RELATED PRELIMINARY PROPOSAL NO. IF APPLICABLE
$ **198,601**	**24** months	**06/01/04**	

CHECK APPROPRIATE BOX(ES) IF THIS PROPOSAL INCLUDES ANY OF THE ITEMS LISTED BELOW

☒ BEGINNING INVESTIGATOR (GPG I.A)
☐ DISCLOSURE OF LOBBYING ACTIVITIES (GPG II.C)
☐ PROPRIETARY & PRIVILEGED INFORMATION (GPG I.B, II.C.1.d)
☐ HISTORIC PLACES (GPG II.C.2.j)
☐ SMALL GRANT FOR EXPLOR. RESEARCH (SGER) (GPG II.D.1)
☐ VERTEBRATE ANIMALS (GPG II.D.5) IACUC App. Date _____
☐ HUMAN SUBJECTS (GPG II.D.6)
Exemption Subsection _____ or IRB App. Date _____
☐ INTERNATIONAL COOPERATIVE ACTIVITIES: COUNTRY/COUNTRIES INVOLVED (GPG II.C.2.j)
_____
☐ HIGH RESOLUTION GRAPHICS/OTHER GRAPHICS WHERE EXACT COLOR REPRESENTATION IS REQUIRED FOR PROPER INTERPRETATION (GPG I.E.1)

PI/PD DEPARTMENT
Computer Science

PI/PD FAX NUMBER
850-644-0058

PI/PD POSTAL ADDRESS

Tallahassee, FL 323064530
United States

NAMES (TYPED)	High Degree	Yr of Degree	Telephone Number	Electronic Mail Address
PI/PD NAME **Kartik Gopalan**	**PhD**	**2003**	**850-644-1685**	**kartik@cs.fsu.edu**
CO-PI/PD				
CO-PI/PD				
CO-PI/PD				
CO-PI/PD				

Electronic Signature

CERTIFICATION PAGE

Certification for Authorized Organizational Representative or Individual Applicant:

By signing and submitting this proposal, the individual applicant or the authorized official of the applicant institution is: (1) certifying that statements made herein are true and complete to the best of his/her knowledge; and (2) agreeing to accept the obligation to comply with NSF award terms and conditions if an award is made as a result of this application. Further, the applicant is hereby providing certifications regarding debarment and suspension, drug-free workplace, and lobbying activities (see below), as set forth in Grant Proposal Guide (GPG), NSF 04-2. Willful provision of false information in this application and its supporting documents or in reports required under an ensuing award is a criminal offense (U. S. Code, Title 18, Section 1001).

In addition, if the applicant institution employs more than fifty persons, the authorized official of the applicant institution is certifying that the institution has implemented a written and enforced conflict of interest policy that is consistent with the provisions of Grant Policy Manual Section 510; that to the best of his/her knowledge, all financial disclosures required by that conflict of interest policy have been made; and that all identified conflicts of interest will have been satisfactorily managed, reduced or eliminated prior to the institution's expenditure of any funds under the award, in accordance with the institution's conflict of interest policy. Conflicts which cannot be satisfactorily managed, reduced or eliminated must be disclosed to NSF.

Drug Free Work Place Certification

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Drug Free Work Place Certification contained in Appendix C of the Grant Proposal Guide.

Debarment and Suspension Certification

(If answer "yes", please provide explanation.)

Is the organization or its principals presently debarred, suspended, proposed for debarment, declared ineligible, or voluntarily excluded from covered transactions by any Federal department or agency? Yes ☐ No ☒

By electronically signing the NSF Proposal Cover Sheet, the Authorized Organizational Representative or Individual Applicant is providing the Debarment and Suspension Certification contained in Appendix D of the Grant Proposal Guide.

Certification Regarding Lobbying

This certification is required for an award of a Federal contract, grant, or cooperative agreement exceeding $100,000 and for an award of a Federal loan or a commitment providing for the United States to insure or guarantee a loan exceeding $150,000.

Certification for Contracts, Grants, Loans and Cooperative Agreements

The undersigned certifies, to the best of his or her knowledge and belief, that:

(1) No federal appropriated funds have been paid or will be paid, by or on behalf of the undersigned, to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with the awarding of any federal contract, the making of any Federal grant, the making of any Federal loan, the entering into of any cooperative agreement, and the extension, continuation, renewal, amendment, or modification of any Federal contract, grant, loan, or cooperative agreement.

(2) If any funds other than Federal appropriated funds have been paid or will be paid to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with this Federal contract, grant, loan, or cooperative agreement, the undersigned shall complete and submit Standard Form-LLL, ''Disclosure of Lobbying Activities,'' in accordance with its instructions.

(3) The undersigned shall require that the language of this certification be included in the award documents for all subawards at all tiers including subcontracts, subgrants, and contracts under grants, loans, and cooperative agreements and that all subrecipients shall certify and disclose accordingly.

This certification is a material representation of fact upon which reliance was placed when this transaction was made or entered into. Submission of this certification is a prerequisite for making or entering into this transaction imposed by section 1352, Title 31, U.S. Code. Any person who fails to file the required certification shall be subject to a civil penalty of not less than $10,000 and not more than $100,000 for each such failure.

AUTHORIZED ORGANIZATIONAL REPRESENTATIVE	SIGNATURE	DATE
NAME **Kirby W Kemper**	**Electronic Signature**	**Dec 8 2003 8:49AM**

TELEPHONE NUMBER	ELECTRONIC MAIL ADDRESS	FAX NUMBER
850-644-5260	**nsfaward@res.fsu.edu**	**850-644-1464**

*SUBMISSION OF SOCIAL SECURITY NUMBERS IS VOLUNTARY AND WILL NOT AFFECT THE ORGANIZATION'S ELIGIBILITY FOR AN AWARD. HOWEVER, THEY ARE AN INTEGRAL PART OF THE INFORMATION SYSTEM AND ASSIST IN PROCESSING THE PROPOSAL. SSN SOLICITED UNDER NSF ACT OF 1950, AS AMENDED.

PROPOSAL SUMMARY

There is a constant battle to break-even between continuing improvements in DRAM capacities and the demands for even more memory by modern memory-intensive applications. Such applications do not take long to hit the physical memory limit and start paging to disk, which in turn considerably slows down their performance. Earlier research efforts have attempted to completely eliminate large disk latencies by paging over a high-speed LAN to the unused memory of remote machines in the network. This approach can dramatically improve application performance by providing low-latency access to potentially "infinite" memory resources.

In spite of significant potential benefits of remote memory paging, there has been scant acceptance of techniques from prior research by the general user community in executing their real-world memory-intensive applications. The primary reason is that all earlier approaches advocate significant modifications to end-systems involved in the remote paging operations, either in terms of a specific programming interface for applications, or in terms of extensive changes to the operating systems and device drivers. *The primary goal of this proposal, called* **Anemone** *(Adaptive NEtwork MemOry engiNE), is to investigate the research challenges in building a transparent and adaptive network memory service.* Anemone avails remote memory resources for memory-intensive applications by pooling together the collective unused memory resources of nodes across a high-speed LAN.

Intellectual Merit: The goal that distinguishes this proposal from previous efforts is that Anemone requires no changes whatsoever to either the applications or the end-host operating systems and device drivers. Instead, the Anemone project aims to exploit two commodity features of modern operating systems - the standard Network File System (NFS) and RamDisks - in a novel manner. Anemone enables applications to transparently enjoy the benefits of access to remote memory without painstaking and extensive end-host modifications. This proposal aims to provide complete end-host transparency, low paging latencies over the network, fine-grained memory resource management, and fault-tolerance. The Anemone project would immediately benefit high-performance application community in reducing time and cost to develop and deploy memory-intensive applications. Some major application areas to benefit include weather prediction, scientific computing, database management, multimedia processing and high resolution graphics rendering.

Broader Impact: The broader impact of this proposal is along several dimensions. (i) The research will be primarily conducted with active involvement of graduate and undergraduate students, enabling them to hone their skills in designing and building real-world systems. Special efforts will be made to involve students from under-represented groups attending FSU from local community colleges and institutions such as Florida A&M University (a historically african-american institution). (ii) A direct result of Anemone project will be to augment FSU's computational infrastructure with a distributed network memory server testbed that could be actively used by high-performance computation researchers and scientists in FSU Computer Science department and School of Computational Science and Information Technology (CSIT). (iii) Software and systems developed as part of this effort will be open-sourced to general research community for additional contributions and enhancements. Results from this research will also be disseminated widely via publications in workshops, conferences and journals. (iv) The distributed tools resulting from Anemone project will be utilized in classroom projects to aid graduate and undergraduate students to experiment with systems aspects of supporting memory-intensive applications.

TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.

*Proposers may select any numbering mechanism for the proposal. The entire proposal however, must be paginated. Complete both columns only if the proposal is numbered consecutively.

PROJECT DESCRIPTION

1 Introduction

Rapid improvements in DRAM technology have resulted in an unprecedented increase in memory capacity that can be packed into standard off-the-shelf systems. Nevertheless, applications' hunger for more memory continues to always remain one step ahead of improvements in memory capacity. The issue is not whether one can provide enough DRAM to satisfy the modern memory-hungry applications but rather, provide more memory and they'll use it all up and ask for even more. This constant battle to break-even especially holds in the case of memory-intensive high-performance applications such as weather predictions, scientific computing, database management, multimedia processing and graphics applications such as high resolution volumetric rendering. It does not take very long for such applications to hit the physical memory limit and start paging to physical disk, which in turn considerably slows down the performance of these memory intensive applications.

While disk technology has continued to improve as well, the improvement in disk bandwidth and access latency has significantly lagged behind the improvements local area network (LAN) bandwidths and latencies. The 100Mbps LANs, which have been standard until recently, are rapidly being replaced by affordable Gigabit LANs with attractive cost-to-performance ratios. Consequently, instead of paging to a slow local disk, one can significantly reduce access latencies by by paging over a high-speed LAN to the unused memory of remote machines. This approach can dramatically improve the performance of memory-intensive applications by providing them access to potentially "unlimited" memory resources.

Indeed the very concept of paging over the network to reduce disk-access penalty is itself not new and has been explored in prior works [8, 13, 12, 14, 23, 30, 20, 24, 32]. While these efforts have resulted in important advancements over state-of-the art, their acceptance by user community in running real-world memory-intensive applications has been scant in spite of their enormous potential benefits. The primary reason is that all earlier approaches advocate significant changes to end-systems that execute the memory-intensive applications and those that provide their unused memory for remote paging. These changes range from application-specific programming interfaces that programmers need to follow, to changes in the end-host operating system and device drivers.

The primary goal of Anemone (Adaptive NEtwork MemOry engiNE) project is to investigate the research challenges in building a distributed network memory service for memory-intensive applications by pooling together the collective unused memory resources of nodes across a high-speed LAN. The novel goal that distinguishes the Anemone project from prior efforts is that no changes are required either to the memory-intensive applications or to the end-host systems. Instead, Anemone aims to exploit two standard features that arrive bundled with modern end-host systems - the standard RamDisks and Network File System (NFS) protocol stack - in a novel manner and thus eliminate painstaking and extensive end-system changes. The ultimate performance objective of Anemone is to enable users of memory-intensive applications to benefit from a transparent, fault-tolerant, and low-overhead mechanism of of paging to remote memory.

An obvious question that arises is that, instead of trying to pool together unused memory from end-hosts in a LAN, why one should not purchase a single very large memory vendor-specific enterprise server and convert it into a network memory server? The principal argument for building a system such as Anemone is the same as the reason why distributed clusters of commodity workstations are replacing closely coupled vendor-specific large-scale servers as pre-dominant high performance

computing platforms. Building a distributed network memory server out of commodity workstations already connected via a high-speed gigabit LAN is much more scalable, cost-effective, and capable of evolving with improvements in hardware technology, when compared to vendor-specific large-memory servers whose technology can quickly be obsoleted by improvements in commodity hardware within a few years of deployment. Thus Anemone permits memory server deployment to keep pace and even evolve with the technology trends.

The Anemone project attempts to tackle the following primary research challenges in building a large-scale distributed network memory server.

- The foremost goal of Anemone is to provide transparency for end-hosts in paging to remote memory. Specifically, the goal is to ensure that absolutely no changes are required either to the memory-intensive application or to the operating system and device drivers at the end-hosts. This not only enables legacy applications to benefit from availability of additional memory, but also enables the NMS to scale seamlessly when additional nodes are added to the LAN.

- The second important goal of Anemone is to minimize the latency overhead of paging over the network to a remote memory - the target performance goal being to contain the network transfer component of each remote paging request to within 100 microseconds.

- An important concern with remote paging is that remote systems that hold swapped out pages in their memory might crash leading to failures of memory-intensive applications. Thus, the third goal of Anemone is to investigate techniques to build fault-tolerance and reliability into the distributed memory server, such as by means of striping each swapped out page to multiple remote servers, or by making them persistent without impacting paging latencies.

- Finally, memory-intensive applications may need to share memory resources with other normal applications which could potentially result in load-imbalance among end-systems in the LAN. Thus maintaining load-balance is one of the primary objectives in a distributed memory server rather than merely being a secondary goal.

The rest of this proposal is organized as follows. In Section 2, we describe how two standard features of modern end-host systems, namely the NFS protocol stack and the RamDisk device, can be used to easily build a basic network memory server architecture and address the deficiencies of such a framework. In Section 3, we present the architectural design of Anemone - a distributed network memory server that addresses the inherent problems of the basic framework while maintaining transparency for end-host systems. Section 4 reviews related research. In Section 6, we provide the detailed research plan and timeline for Anemone project. Section 5 describes expected research and broader impact of this endeavor.

2 A Barebones Mechanism for Paging to Remote Memory

The primary goal of Anemone project is to provide a transparent mechanism for end-host systems to perform paging over remote memory. Two basic requirements to achieve this goal are (1) the end-host executing the memory intensive client must be capable of transferring the memory pages being paged out or paged in over the high-speed network instead of a local disk, and (2) a remote memory server must be capable of storing the paged out pages in its main memory instead of

disk. Both the goals can be attained if one makes extensive changes to kernel swap daemon on the end-host that communicates with a dedicated memory management process or kernel module on remote memory server. However, this approach takes away the transparency of remote paging since every end-host that wishes to benefit from remote memory paging would need a specialized kernel.

To maintain transparency in remote paging, Anemone exploits standard components that come bundled with end-host systems, namely the RamDisk and the Network File System (NFS). In fact, a very barebones remote paging mechanism can be readily constructed using RamDisk and NFS without significant effort while maintaining transparency. In this section, we outline the architecture of this barebones remote memory paging system, followed by an enhanced Anemone architecture in Section 3.

2.1 The RamDisk

A RamDisk is a portion of physical memory that one can set aside for use as a regular disk partition. In other words, user of an end-host can pretend to treat a section of memory as a disk partition and save data or files to it. RamDisk has several practical utilities. For instance, frequently used files can be placed in a RamDisk for low-latency access. A more conventional application of RamDisk is to enable a diskless client to maintain minimal file-system image in memory for normal operations. The principal advantage of RamDisk is that, being in-memory, it enables faster access to data files while providing the standard interface of a disk-partition or a file-system. Access to RamDisk in systems such as Linux is via the standard device interface (`/dev/ram0, /dev/ram1` etc.)

From the viewpoint of remote-paging, RamDisk is attractive due to its ability to be configured as an in-memory *swap device*. Traditional swap devices reside on disk with their main purpose being to support applications that need more virtual memory than available system RAM. RamDisk provides a simple mechanism to configure excess memory at a remote server as a *remote in-memory swap device*. The missing piece in the picture is a communication mechanism that enables the end-hosts running memory-intensive applications to access this remote in-memory swap device in a transparent fashion.

2.2 The Network File System (NFS)

The NFS [29] protocol provides transparent remote access to shared file-systems across a network. It enables clients to access remote files as if they were local files, in a manner that is network, OS, and transport protocol independent. NFS has established itself as a de-facto standard for remote file-system access due to its inherent simplicity - servers are stateless and clients communicate with servers using a simple protocol.

While NFS is primarily used as a mechanism to access data files located in remote file-systems, another not very uncommon the use of NFS is to support remote-swapping capability for diskless thin clients. Specifically, diskless clients can configure a portion of persistent storage (or a large file) on a remote server's file-system as as their swap device, mount it remotely using NFS and perform all their page-in/page-out operations over the network to the swap device on the remote server. While this approach enables diskless clients to execute large applications, there is considerable latency involved in page-in operations that require not just transfer of pages over the network but also the normal disk access latency at the remote server.

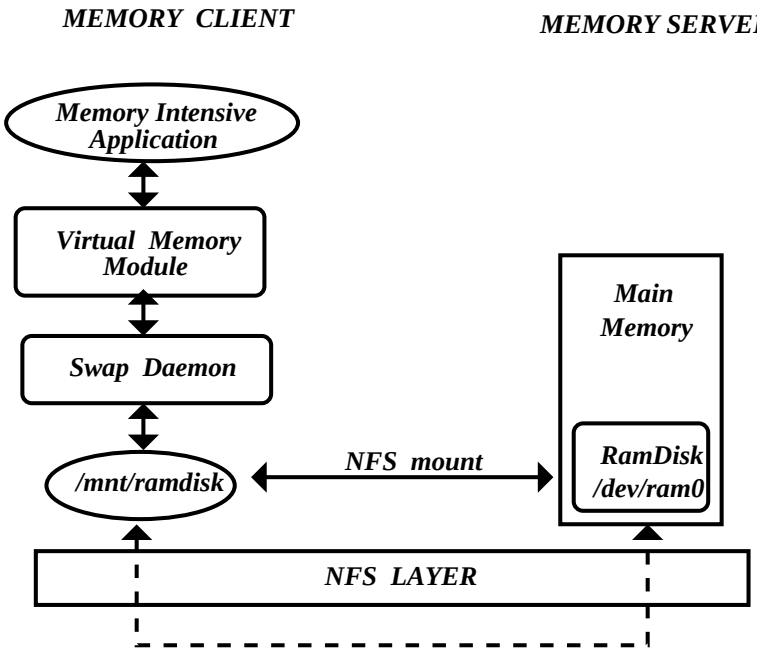

Figure 1: The Barebones Remote Memory Paging Architecture. Memory client mounts a remote RamDisk from Memory Server and configures it as a swap device. Applications on memory client transparently swap to remote memory instead of local disk.

2.3 The Barebones Architecture

The two standard facilities of RamDisk and NFS protocol can be used in conjunction to build a basic remote memory paging mechanism. The architecture of such a remote-paging mechanism is shown in Figure 1. A remote server, that has unused physical memory to spare for other memory-starved clients, creates an in-memory RamDisk of adequate size. The RamDisk is configured at the server as being capable of being mounted by any remote clients by means of NFS. Any client that wishes to benefit from paging over remote memory mounts the RamDisk at the remote server via NFS to its local file-system. Following this the mounted RamDisk can be configured as the primary swap device by the client using system-specific commands (such as `mkswap` and `swapon` in Linux). Any demand for additional memory from a memory-intensive application is automatically directed by the local swap daemon to the swap device located on remote RamDisk, which in turn is nothing but a convenient access technique to remote server's memory.

As far as the memory-intensive application is concerned, its need for additional memory is satisfied by the underlying virtual memory sub-system on the end-host (remote memory client). The virtual memory subsystem in turn interacts the swap daemon to page-out or page-in memory contents to/from the swap device. The fact that swap device is in fact a RamDisk residing in the memory of a remote server is transparent to the swap daemon itself. The transparency is achieved by use of NFS protocol to mount the remote RamDisk at a local mount point. Thus by combining together a number of readily available components, one can achieve a swapping capability to a remote memory server. The principal advantage of this approach is that no changes are required either to the memory-intensive applications or to the operating system and device drivers. The

only operations required at the client side involve mounting a remote RamDisk onto a local mount points. Similarly, on the server side, the only operations required is to configure a NFS mountable RamDisk for remote clients. While this barebones architecture is sufficient to get basic remote-paging setup and running, there are several critical limitations of this simple setup that necessitates the design of a more robust and flexible architecture that can be relied upon to support critical memory-intensive applications. In summary, what is required is a more flexible mechanism which builds upon the basic approach of using RamDisk and NFS.

3 Anemone - Design and Implementation

Although the architecture in Section 2 provides basic remote memory paging, the mechanism can be augmented in several ways to render it suitable for real-world production-scale applications. The desirable features of a full-scale remote memory paging system include maintaining transparency with growing memory requirements of memory-intensive applications, low latency paging operation, adaptability to variation in global memory availability and demands, and reliability of remote paging. The Anemone project explores the design space of distributed remote memory paging along these dimensions. The fundamental entity in Anemone is the *memory engine* service which functions as an intermediary between memory clients and servers. The memory engine transparently maps the client memory requirements to available global network memory and hides the complexity of memory resource management from both memory clients and servers. This section outlines the research challenges in Anemone architecture and provides the rationale behind specific design decisions.

3.1 The Memory Engine - An Agent for End-host Transparency

The first research question that Anemone tackles concerns the transparency to clients in accessing, and to servers in contributing, the distributed RamDisk resources in a local area network. Specifically, *is it possible to transparently pool together the distributed excess memory resources in a network such that neither the memory-intensive applications nor the operating systems/device drivers at both memory clients and servers entail any changes whatsoever in the remote memory paging process.*

While the barebones approach described in Section 2 provides a certain level of transparency, it turns out to be insufficient in handling even moderately complex situations. For instance, if a client requires more memory than that provided by a single remote RamDisk, it needs to configure another remote RamDisk (possibly from another server) to increase its remote memory swap space. The process of selecting additional remote RamDisks greatly increases the management complexity at the client end, since it requires detailed knowledge of all remote servers and the spare capacity available with each of them. Clearly, it is highly desirable to insulate the client from having to micro-manage such detailed server configuration issues. Furthermore, RamDisk servers themselves are ordinary end-hosts that contribute their unused memory to clients for remote paging. In fact the same end-host can switch its roles between being a memory client, while executing memory-intensive applications, and being a remote RamDisk server, when it has unused memory resources to contribute. Thus it is extremely important to operate both the clients and the RamDisk servers using unmodified applications and commodity operating systems/device drivers.

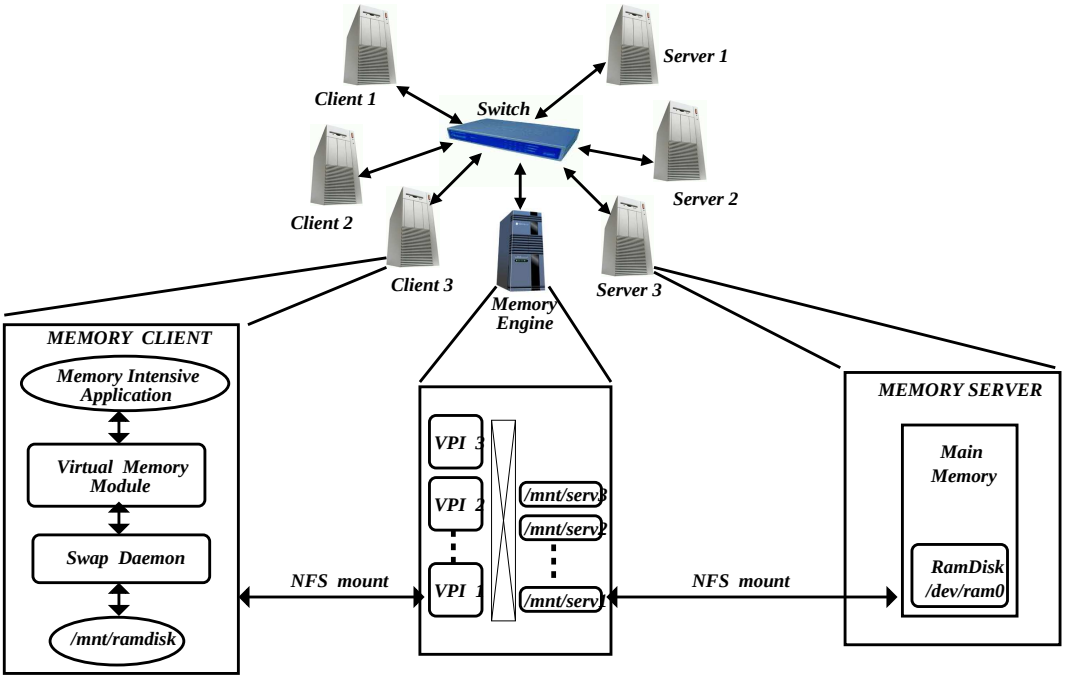

Figure 2: Architecture of Anemone. Memory clients mount a Virtual Paging Interface (VPI) from the Memory Engine, which in turn mounts RamDisk partitions from memory servers. The Memory Engine multiplexes client paging requests among back-end memory servers.

Anemone's addresses this challenge by introducing the notion of a *memory engine* service. Simply speaking, a memory engine acts as a bridge between memory clients on one side and servers hosting remote the RamDisk resources on the other side. Memory engine does not by itself provide any memory resources; rather it helps pool together, and present a unified access interface for, distributed RamDisk resources across the network. The primary function of a memory engine is to hide the complexity of managing global memory resources from the memory clients, and the complexity of managing client requirements from the memory servers.

Figure 2 shows the architecture of Anemone. The memory engine is a dedicated device connected to the same switched gigabit network as other memory clients and RamDisk servers. The memory engine has two network interface cards, both of which connect to the same switched LAN, the purpose being to isolate memory engine's communication with clients and servers on different interfaces. (Since hosts interconnect via switched LAN, the problem of collisions as in shared segment Ethernet, does not arise).

Client-side Interface: The client side operations in Anemone are very similar to the basic architecture in Section 2. The memory engine provides each client with a a single *virtual paging interface*(VPI). The VPI is simply a pseudo swap device on the memory engine that can be mounted by remote memory clients using the NFS protocol. Each memory client can subsequently configure its mounted VPI as the primary swap device (using `mkswap` and `swapon`) with a higher priority than any local disk based swap devices. Any subsequent paging operations, that result from additional memory demand on the client, will be directed to the remote VPI. The difference with barebones architecture lies in the fact that the VPI is not a real swap space on the memory engine. Rather, it

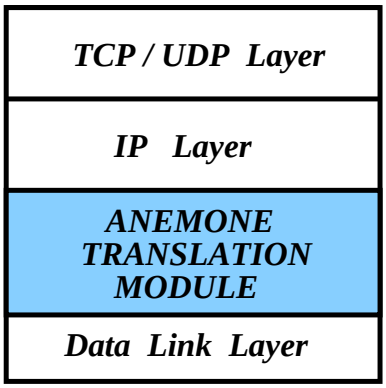

Figure 3: Translation module in the Memory Engine is located in the network protocol stack between the IP layer and the data link layer.

is a convenient front-end interface for clients to utilize the distributed back-end RamDisk resources on remote memory servers.

Server-side Interface: The memory engine maintains a simple NFS-based interface for interacting with back-end RamDisk server. Each host in the network, that wishes to contribute portion of its memory space for remote paging, configures a RamDisk which it then exports for NFS mount by the memory engine. The memory engine is configured to keep track of individual servers and mounts their respective RamDisks onto mount points in its local file system.

Mapping Paging Requests to RamDisk resources: The memory engine contains a *translation module* that maps client paging requests to the distributed RamDisk resources at the back end servers. In this manner, each client interacts with only one interface at the memory engine which in turn hides and efficiently manages the complexities of back-end server resource management. The translation module is an additional layer inside the network protocol stack of the memory engine's operating system. As shown in Figure 3 the translation module is inserted between the data-link layer and the IP layer of the network protocol stack. The reason for placing translation module in the protocol stack of memory engine's kernel is to reduce the latency of paging operating experienced by client applications. Specifically, a kernel level entity enables us to implement a novel latency reduction technique called *split page-in* which will be described in Section 3.2.

The translation module's responsibility is to intercept the paging requests sent by clients using the NFS protocol, and satisfy them by mapping each request to a back-end server. For instance, if a client makes a page-out request to NFS mounted VPI, the translation module intercepts and interprets the page-out request, identifies the back-end server which has idle RamDisk space to store client's page and saves it to the mounted RamDisk. Similarly, for a page-in request, the translation module identifies the back-end server where a client's page is saved. Subsequently, one option is for the translation module to retrieve and return the page from mounted RamDisk of server. A better option, from latency reduction point-of-view is to implement *split page-in* which is described in Section 3.2.

Most major operating systems provide convenient mechanisms to insert/remove custom modules (for instance `insmod` and `rmmod` facilities in Linux). Thus the translation module itself is structured as a pluggable module that interfaces with the runtime kernel of the memory engine at well defined

interfaces. The translation module, that resides completely in the memory engine, is the only kernel level entity required to support transparent remote memory paging. The services provided by the translation module removes the need to change the operating system, device drivers or applications at any memory client or RamDisk server.

Disparity between Page Size and Packet MTU: Another issue concerns the virtual memory page size. Most operating systems set the page size to around 4Kbytes or 8Kbytes. On the other hand, the default maximum packet size in Ethernets is 1500 bytes. Thus, on a commodity switched Ethernet, the NFS layer on clients would break up an 8Kbyte sized page being swapped to VPI into at least six fragments before transmission over the network. Thus, the translation module has the additional responsibility of re-assembling these packets before servicing the page-out request. This re-assembly could easily be taken care of by the IP layer on the memory engine, if translation module were built on top of the IP layer, instead of below it. However, split page-in mechanism described later requires access to MAC level headers which forces us to implement the translation module below the IP layer. Modern Gigabit Ethernet switches do support an additional option of *jumbo* frames which are 9Kbytes in size, though this is not the default configuration. Anemone project will also explore the feasibility and potential benefits of using jumbo frames, in addition to conventional Ethernet frames, for remote paging operations.

3.2 *Split Page-in* - A Latency Reduction Mechanism

The most attractive feature of paging to remote memory is that data transfer latencies over network are orders of magnitude smaller than transfer latencies to/from local hard disks. Specifically, remote memory paging is about 500 times faster than paging to local disks [32]. Nonetheless, network access latencies form an important component of the performance overhead in remote paging operation and any design must strive to minimize this latency and its potential impact on application performance.

Page-out vs. Page-in: An important distinction exists in terms of tolerable latencies between the operations of paging in and paging out of an application's memory pages. Specifically, page out operations do not typically lie in the critical path of application execution since the swap daemon on a client end-host can anticipate local physical memory shortages and schedule memory pages to be swapped out much in advance of additional memory space being required. In contrast, page in operations directly impact the critical execution path since an application that page-faults absolutely cannot continue without the required page being brought into local physical memory. Thus it is far more important to minimize the latency in page-in operation as compared to page-out operation.

Indeed our approach of providing transparency, by means of an intermediate memory engine, introduces an additional level of indirection in communication between memory clients and RamDisk servers. This in turn implies potential increase in paging latencies. As mentioned above, it is the page-in latencies that are more critical from application performance perspective and should be the prime target of any latency optimization effort.

We propose to employ a novel mechanism in Anemone called *split page-in*, that optimizes the page-in latency. *Split page-in* exploits the fact that NFS messages corresponding to the page-in request from client are small in size, whereas the return replies to the page-in message, that carry the actual contents of the requested page, are larger and constitute greater component of the page-in latency. The basic idea behind *split page-in* is that client first sends a page-in request to its VPI

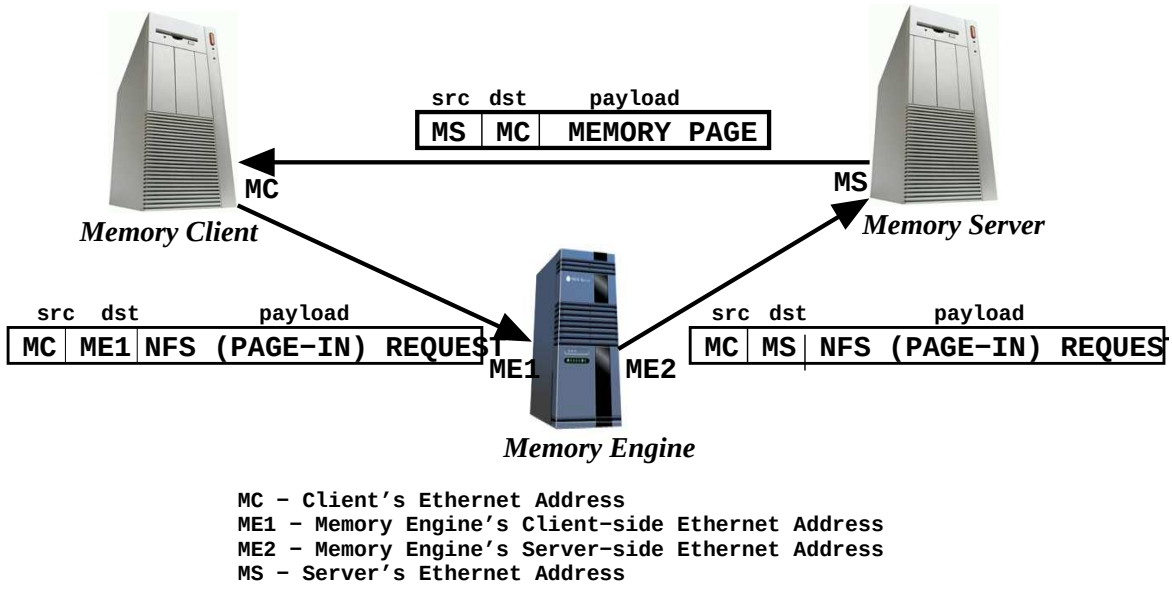

Figure 4: Translation module maps the NFS page-in request from a client into a request to one of the back-end server hosting the memory page. The request sent from Memory Engine to the back-end server contains the memory client's Ethernet address as the source (instead of memory engine's Ethernet address). This enables the server to reply directly to the client, avoiding an extra step through the Memory engine.

at the memory engine which in turn routes the request to a back-end RamDisk server. However, the back-end memory server returns the required memory page directly to the requesting client, instead of routing it again via the memory engine. This way, *split page-in* optimizes a potentially expensive two-step transfer of large memory pages into a direct single-step transfer from the RamDisk server to the memory client. The main challenge in implementing *split page-in* lies maintaining server transparency. Specifically, we need to ensure that the the back-end RamDisk server responds directly to client even though it receives the page-in request from the memory engine - all without any changes to operating system or device driver installed at the server.

This goal is achieved with a little help from from the memory engine. Figure 4 illustrates the translation process. Before the client's page-in request (an NFS request) is forwarded to a back-end server, the translation module fills the source MAC address field of the outgoing NFS packet with the hardware MAC address of the client requesting the page-in. Similarly, it also places the client's IP level source information in the source field of the outgoing packet's IP header. The purpose of this replacement is to make the back-end RamDisk server believe that the page-in request is coming directly from the client rather than through the memory engine. As a result, the RamDisk server responds directly to the client with the requested page and avoids the an additional hop through the memory engine. In addition to reducing the page-in latency by almost half, a pleasant side-effect of split-paging is that the memory engine is relieved of the load of having to forward paged-in memory contents from servers to clients.

3.3 Adaptation to Resource Availability

A guiding consideration in Anemone project is to enable the distributed remote memory paging system to adapt with varying resource availability. One of the features of Anemone is its ability to perform fine-grained resource management. There are three dimensions along which Anemone addresses resource management beyond merely enable client-server transparency.

Multiplexing RamDisk Resources: The basic architecture in Section 2 requires a dedicated remote RamDisk for each end-host client; a remote server that supports two different clients would need to set up a dedicated RamDisk for each client, which in turn increases the management complexity and reduces transparency on the server side. Ideally, we would like to keep the configuration complexity at the server to a minimum by multiplexing the resources of a single (large) RamDisk at each server among multiple clients.

Load-balancing Among Servers: Similarly, without some form of co-ordination among servers, it is likely that several clients might end up overloading a single server, while other servers in the network having excess memory resources might remain underutilized. RamDisk servers themselves are ordinary just ordinary workstations on a LAN. Results from earlier research [] indicates that significant performance impact can result from the number of CPU cycles required to serve remote pages. Thus a mechanism to balance the paging load among multiple servers would significantly improve the performance of distributed remote paging.

Again the memory engine of Anemone architecture enables such *resource multiplexing* and *load-balancing* by decoupling the client side interface from server side configuration issues. Each client merely interacts with its front-end VPI at the memory engine. Similarly, each back-end server exposes a single remote RamDisk interface to the memory engine. Now the memory engine, in turn, has complete flexibility in multiplexing client page-out requests among back-end RamDisk servers such that loads across multiple servers remains balanced.

Adapting to Resource Variations: Another key aspect of adaptability is Anemone's ability to react to increase/decrease in available global memory. For instance, as additional nodes with unused memory to spare are plugged into the LAN, their additional capacity must be seamlessly absorbed into the global RamDisk pool. Conversely, as memory servers leave the network the total system capacity must correspondingly scale down without visibly impacting active memory clients. The memory engine provides such adaptability by means of actively tracking the liveness and available resources of all RamDisk servers in the local network. Server tracking enables the total system capacity to scale or shrink with the available resources in distributed RamDisk servers.

The main challenge lies in tracking server capacities and loads while simultaneously maintaining server-side transparency. Liveness of servers by itself could be checked by means of simple heartbeat mechanisms (such as by using the standard `ping` messages). Tracking server loads in a transparent manner is a tougher task we do no wish to modify servers' operating system or device drivers in any fashion. One of the options Anemone will explore is to let the memory engine derive server load information by passively observing the latency of paging operation from each server. This option is the most attractive from viewpoint of maintaining server transparency, though the information thus derived is subject to occasional inaccuracies depending on how well the paging latencies reflect the back-end server load information. The second option is to execute a light-weight user-level process in the background at each server that provides periodic feedback on server load to the memory engine. This option is capable of providing accurate load information, though it introduces additional management complexity at the server side - something Anemone strives to

avoid.

All said, Anemone architecture by itself is not tied to any specific schemes for resource multiplexing, load-balancing or server load tracking. The PI's prior research experience in designing reliability mechanisms in a web server cluster [22] will be brought to bear upon these issues in the Anemone project as well.

3.4 Remote Paging Reliability

Another guiding consideration in design of Anemone is the overall reliability of remote memory paging mechanism. Technically, the consequences of any form of failure in remote swapping process is no different than failure of a local disk-based swap device. However, a degree of uncertainty does arise from the fact that critical swap contents are located remotely and that multiple components are involved in access to the remote swap device. Extensive work has been conducted by research community on the subject of reliability and fault-tolerance for distributed applications. The research goal of Anemone is study their applicability and impact on the performance of remote memory paging mechanism. Two specific reliability concerns that Anemone's architecture needs to address are (1) failure of one or more remote RamDisk servers that host a client's swapped-out page, and (2) the failure of the memory engine itself.

The most common mechanism to provide reliability against data loss from a server failure is by means of replication or striping. Presence of multiple remote RamDisk servers on the network affords the opportunity to perform much more reliable swapping by replicating or striping the data being paged out over multiple remote servers. The basic remote paging framework in Section 2 could potentially be made robust by having each client replicate/stripe the the paged out data to multiple remote servers. However, this would also imply extensive modification to client side OS or applications which compromises client side transparency. A dedicated memory engine in Anemone architecture opens up the possibility of removing this barrier to client side transparency. The memory engine could potentially replicate/stripe the data being paged out on behalf of the client machines, while the clients deal with only a single VPI at the memory engine. Thus one of the research goals of Anemone is to explore various replication and striping strategies and their impact on the performance of distributed remote memory paging.

The second natural concern is the reliability of memory engine itself. Memory engine is a dedicated access device on the network that that is critical for remote memory paging and could potentially become a single point of failure. One of the most extensively used mechanisms to ensure reliability of critical network network access devices is the *process-pair* concept, advocated originally in [5]. Though the basic principle of process-pair operations are the same across various critical devices, the specific implementation mechanisms vary depending upon the functionality of the device.

Anemone project proposes to apply the process-pair concept to provide reliability to the memory engine. The basic idea behind the process-pair approach is that the operations of a primary memory engine will be mirrored by a backup memory engine. The backup maintains exactly the same state as the primary but without actively interacting with remote clients or servers. Upon failure of the primary device, the backup takes over the operations transparently without visibly impacting memory client and server operations. The specific design issues in the application of process-pair approach to Anemone concerns (1) how the backup memory engine will parallely intercept the NFS packets destined to the primary memory engine and (2) how the backup will maintain

synchronization with the primary without impacting is normal operations, and (3) the mechanism of failure detection and recovery.

The PI's prior research experience in designing and implementing fault-tolerance mechanisms for network access devices [26] will be brought to bear upon these issues in the Anemone project as well.

4 Related Work

The concept of writing to main memory in order to get around the disk access latency problem is not new. Most major operating systems, such as Linux, Solaris, BSD, and Windows 2000/XP, provide the facility of software RamDisks which emulate the interface provided by block storage devices or file-systems. These are in-memory entities that can be used to store temporary data such as web caches, intermediate compilation results, and `/tmp` files, or even longer term information such as databases, web server contents, etc.

Similarly, the idea of harvesting the idle memory resources from a cluster of workstations in a LAN environment has been around for quite a while. Earliest efforts grew out of speeding up large memory database and transaction processing systems. The Massive Memory Machine project [16, 15] attempted to inter-connect a group of minicomputers with a global broadcast bus. Rather than harvesting remote memory for transparent paging operations, the principal objective of this effort was to provide high speed memory allocation, garbage collection, and transaction support for memory-resident database and file management systems. Several years later, related efforts [18, 34] did touch upon the performance benefits of paging to remote memory instead of local disks, though not as extensively as Anemone and other efforts. Similarly, the Dali [6] project developed a toolkit for building memory-resident databases in telecommunication networks and the work in [19] explored use of collective main memory in a network of workstations to improve the performance of transaction-based systems. Again, rather than transparent paging to remote memory, [6] and [19] focused on issues such as recovery, concurrency control and read/write performance for in-memory database and transaction processing systems.

The idea of paging to remote memory, instead of local disks, started getting serious attention in early 1990's. Comer and Griffioen [8] proposed a remote paging model in which client machines that needed additional memory paged to one of a set of dedicated remote memory servers. Felten and Zahorjan [13] advanced a step further by tracking by not requiring dedicated servers. Rather clients could utilize idle memory on any machine in the network via. a centralized registry. Clients selected one idle machine at random as their remote paging server. Both [8] and [13] employ a customized remote paging protocol to be followed by clients and servers. In contrast, the Anemone framework maintains transparency in paging at both client and server ends by using the standard NFS protocol. Unlike [8], Anemone does not impose any specific roles of servers or clients on individual machines. While the centralized registry in [13] enables clients to locate memory servers, it does not help achieve transparency - clients and servers still require extensive kernel modifications to communicate. Anemone's *memory engine* subsumes the role of centralized registry by not only enabling client-server communication, but also providing a virtual device interface which hides the details of back-end servers from clients.

Feeley et al. [12] proposed a Global Memory System (GMS) system that provides network-wide memory management support at the lowest level of the operating system for activities such as VM paging, memory mapped files and file caching. Unlike, GMS in which a distributed page-

replacement algorithm is closely integrated with the operating system, Anemone does not require any changes in the operating system and maintains client and server transparency by means of the *memory engine*. The Trapeze project [33, 2] developed an adaptive message pipelining technique, that reduces network latency for large messages in gigabit networks.

Another closely related effort is the Network RamDisk [14, 23] which harnesses unused memories in a workstation cluster. In addition to remote paging, Network RamDisk offers data reliability using data replication adaptive parity caching. Though Network RamDisk does not require any changes to the core operating system, it is implemented as a device driver that needs to be installed on each participating workstation. This implies installation and management complexity at each client and memory server in the network. In contrast, Anemone does not require any changes whatsoever at even the device driver level and maintains complete transparency by encapsulating core remote paging functionality in the *memory engine*. Thus barrier to deployment and use is greatly reduced since all management complexity is restricted to one memory engine rather than being spread all across the network.

Samson (Scalable Active Memory Server on A Network) [30] is one of the more recent works in which a dedicated memory server on the network actively attempts to predict pages that will be requested by the client and delivers these pages just-in-time in order to hide the paging latencies. Samson supports the execution of both legacy applications as well as "sophisticated" applications that wish to avail the active paging facilities via the *mmap()* interface. The memory server is implemented using Alpha workstations and a Myrinet backplane and requires extensively customized operating system and device drivers at both the client and the server ends. In contrast, Anemone attempts to make use of standard and readily available NFS and RamDisk features of most modern clients while localizing any OS changes to a single entity, namely the memory engine. Additionally, Anemone's memory engine can easily incorporate the "active" paging features of Samson. Specifically, the memory engine can actively anticipate and pre-fetch the pages required by clients.

Dodo [20, 1] is a user-level system that enables data-intensive applications to use remote memory in a cluster as an intermediate cache between local memory and disk. In contrast to global memory systems such as GMS [12] where operating system is modified, Dodo requires applications to make explicit use of remote memory by linking to a library in order to create, read, write, and delete remote memory regions. An explicit interface has the disadvantage that it requires the programmer to coordinate all data transfer to and from the remote memory cache. Though Dodo does provide a region-management library to simplify this process, it is mainly useful for applications with well-defined memory access patterns. In contrast, Anemone does not require any application level or operating system changes, thus permitting even legacy applications to benefit from remote memory access. A remote paging system incorporating QoS is presented in [24] that permits application-tailored remote memory access with different paging schemes for different areas of virtual memory. The scheme is specific to Nemesis [21] operating system and the price to be paid for application specific customization is modifications to the operating system at both client and server ends.

Other lines of work have attempted to harness collective memory resources for purposes other than remote paging. Simulation studies for a load sharing scheme that combines job migrations with the use of network RAM are presented in [32]. The NOW project [3] at Berkeley is based on the philosophy of using the vast aggregate resources of network clients to improve performance. One of the efforts [10] in NOW project implemented cooperative caching as part of the xFS file system [4]. The focus of the effort was to aggregate the file caches of many machines distributed on a LAN to form a more effective global file cache. Though network RAM was mentioned as one of

the possibilities in the NOW project, the PI could not find any published literature addressing this aspect. Other more recent efforts in cooperative caching that exploit remote memory in a workstation cluster have focused on alternative replacement/replication strategies [7, 9] and avoiding inclusiveness among cache hierarchy [31]. Cashmere [11] is a software DSM system that attempts to maximize the use of all memory available in a cluster, and thus get around the problem of disk access latency.

5 Benefits and Broader Impact

The technical impact of Anemone project is expected along the following dimensions.

1. Demonstrate the feasibility and competitive performance of a transparent remote memory paging mechanism, without the need for any modifications to applications or end-hosts. Anemone will essentially lower the barrier of deployment and use of remote memory paging mechanism in operational network environments.

2. Show that such a transparent mechanism can provide the same or even higher degree of reliability and resource management flexibility, when compared to traditional approaches for remote memory access.

3. Provide immediate and visible benefit to the high-performance application community in reducing the time and cost to develop, test and deploy memory-intensive applications. Some major application areas that will benefit include weather prediction, scientific computing, database management, multimedia processing and graphics applications such as high resolution volumetric rendering.

4. The results and experiences could lead to a re-examination of mechanisms used to harness other aggregate resources such as computation and storage, in a high-speed LAN environment. Specifically, the results could spur efforts to build more transparent access mechanisms along other resource dimensions.

The Anemone project also has indirect broader impacts along following dimensions:

1. The research will be primarily conducted with active involvement of graduate and undergraduate students, enabling them to hone their skills in designing and building real-world systems. Special efforts will be made to involve students from under-represented groups attending FSU from local community colleges and institutions such as Florida A&M University (a historically african-american institution).

2. A direct result of Anemone project will be to augment FSU's computational infrastructure with a distributed network memory server testbed. The testbed will be available for active use by high-performance computation researchers and scientists both within FSU (Computer Science department and School of Computational Science and Information Technology) as well as from other research institutions.

3. Software and systems developed as part of this effort will be open-sourced to general research community for additional contributions and enhancements. Results from this research will also be disseminated widely via publications in reputed conferences and journals.

4. The distributed tools resulting from Anemone project will be utilized in undergraduate and graduate classroom projects as tools for students to experiment with systems aspects of supporting memory-intensive applications.

6 Plan of Research

The PI has extensive experience in successfully designing, implementing, as well as coordinating the development of several operational systems that require substantial systems building expertise. Examples include *Gage* [22], Samson [30], Viking [27], Duplex [26], Wireless Rether [28], IRS [17] and Suez [25]. The ones that are most relevant to Anemone project are (i) Samson, an active memory server that provides remote paging services via extensive modifications to end-host clients and server operating systems, (ii) Gage, a distributed web server cluster that provides QoS guarantees, and (iii) Duplex, a framework of fault-tolerance for network access devices. The experience and insights gained from designing and building each of these systems will provide the basis for successfully completing the Anemone project.

The proposed project is scheduled to span two years, and involves one CS faculty (the PI) and two PhD students. The schedule is intentionally kept short for two years in order to complete an operational Anemone prototype in a short time frame.

Year 1: We will focus two activities in the first year. First, we will refine the design of Anemone presented in this proposal with specific implementation details of the Translation Module and Split Paging mechanisms in the Memory Engine. Since speed of implementing the Anemone system is as important as speed of the system itself, we will also start work on quickly building a prototype of Anemone which enables basic remote paging operations.

Year 2: In the first half of the second year, we will complete the implementation of basic remote paging functionality in Anemone prototype. Simultaneously, we will also perform the detailed design of resource adaptation and fault-tolerance components of the Memory Engine. The second half of the second year will be devoted to implementing and integrating the resource adaptation and fault-tolerance components. During this phase, we will also measure the performance of remote memory paging operations, identify potential performance bottlenecks and continue working on fine-tuning the prototype's performance.

7 Prior NSF Support

I do not have prior NSF support. This is my first proposal submission.

References

[1] A. Acharya and S. Setia. Availability and utility of idle memory in workstation clusters. In *Measurement and Modeling of Computer Systems*, pages 35–46, 1999.

[2] D. Anderson, J. Chase, S. Gadde, A. Gallatin, K. Yocum, and M. Feeley. Cheating the I/O bottleneck: Network storage with trapeze/myrinet. In *In Proc. of the Usenix Technical Conference. New Orleans, LA*, 1998.

[3] T. Anderson, D. Culler, and D. Patterson. A case for NOW (Networks of Workstations). *IEEE Micro*, 15(1):54–64, 1995.

[4] T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang. Serverless network file systems. In *In Proc. of the 15th Symp. on Operating System Principles*, pages 109–126, Copper Mountain, Colorado, Dec. 1995.

[5] J. Bartlett. A nonstop kernel. In *Proc. 8th ACM SIGOPS SOSP*, 1981.

[6] P. Bohannon, R. Rastogi, A. Silberschatz, and S. Sudarshan. The architecture of the dali main memory storage manager. *Bell Labs Technical Journal*, 2(1):36–47, 1997.

[7] F. Brasileiro, W. Cirne, E.B. Passos, and T.S. Stanchi. Using remote memory to stabilise data efficiently on an EXT2 linux file system. In *In Proc. of the 20th Brazilian Symposium on Computer Networks*, May 2002.

[8] D. Comer and J. Griffoen. A new design for distributed systems: the remote memory model. *Proceedings of the USENIX 1991 Summer Technical Conference*, pages 127–135, 1991.

[9] F.M. Cuenca-Acuna and T.D. Nguyen. Cooperative caching middleware for cluster-based servers. In *In Proc. of 10th IEEE Intl. Symp. on High Performance Distributed Computing (HPDC-10)*, Aug 2001.

[10] M. Dahlin, R. Wang, T.E. Anderson, and D.A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *Operating Systems Design and Implementation*, pages 267–280, 1994.

[11] S. Dwarkadas, N. Hardavellas, L. Kontothanassis, R. Nikhil, and R. Stets. Cashmere-VLM: Remote memory paging for software distributed shared memory. In *Proc. of Intl. Parallel Processing Symposium, San Juan, Puerto Rico*, pages 153–159, April 1999.

[12] M. Feeley, W. Morgan, F. Pighin, A. Karlin, H. Levy, and C. Thekkath. Implementing global memory management in a workstation cluster. *Operating Systems Review, Fifteenth ACM Symposium on Operating Systems Principles*, 29(5):201–212, 1995.

[13] E. Felten and J. Zahorjan. Issues in the implementation of a remote paging system. Technical Report TR 91-03-09, Computer Science Department, University of Washington, 1991.

[14] M. Flouris and E.P. Markatos. The network RamDisk: Using remote memory on heterogeneous NOWs. *Cluster Computing*, 2(4):281–293, 1999.

[15] H. Garcia-Molina, R. Abbott, C. Clifton, C. Staelin, and K. Salem. Data management with massive memory: a summary. *Parallel Database Systems. PRISMA Workshop*, pages 63–70, 1991.

[16] H. Garcia-Molina, R. Lipton, and J. Valdes. A massive memory machine. *IEEE Transactions on Computers*, C-33 (5):391–399, 1984.

[17] K. Gopalan and T. Chiueh. Multi-resource allocation and scheduling for periodic soft real-time applications. In *In Proc. of ACM/SPIE Multimedia Computing and Networking (MMCN2002), San Jose, CA*, Jan. 2002.

[18] L. Iftode, K. Li, and K. Petersen. Memory servers for multicomputers. *Digest of Papers. COMPCON Spring*, pages 538–547, 1993.

[19] S. Ioannidis, E.P. Markatos, and J. Sevaslidou. On using network memory to improve the performance of transaction-based systems. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '98)*, 1998.

[20] S. Koussih, A. Acharya, and S. Setia. Dodo: A user-level system for exploiting idle memory in workstation clusters. In *Proc. of the Eighth IEEE Intl. Symp. on High Performance Distributed Computing (HPDC-8)*, 1999.

[21] I.M. Leslie, D. McAuley, R. Black, T. Roscoe, P.T. Barham, D. Evers, R. Fairbairns, and E. Hyden. The design and implementation of an operating system to support distributed multimedia applications. *IEEE Journal of Selected Areas in Communications*, 14(7):1280–1297, 1996.

[22] C. Li, G. Peng, K. Gopalan, and T. Chiueh. Performance guarantee for cluster-based internet services. In *In Proc. of Intl. Conference on Distributed Computing Systems (ICDCS 2003), Providence, Rhode Island*, May 2003.

[23] E.P. Markatos and G. Dramitinos. Implementation of a reliable remote memory pager. In *USENIX Annual Technical Conference*, pages 177–190, 1996.

[24] I. McDonald. Remote paging in a single address space operating system supporting quality of service. Tech. Report, Dept. of Computing Science, University of Glasgow, Scotland, UK, 1999.

[25] P. Pradhan, K. Gopalan, and T. Chiueh. Design issues in system support for programmable routers. In *In Proc. of 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Schloss Elmau, Germany*, May 2001.

[26] S. Sharma, J. Chen, W. Li, K. Gopalan, and T. Chiueh. Duplex : A reusable fault-tolerance extension for network access devices.

[27] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. Viking: A multi-spanning-tree ethernet architecture for metropolitan area and cluster networks. In *In Proc. of IEEE/INFOCOM'04 Hong Kong, China*, March 2004.

[28] S. Sharma, K. Gopalan, N. Zhu, G. Peng, P. De, and T. Chiueh. Implementation experiences of bandwidth guarantee on a wireless lan. In *In Proc. of ACM/SPIE Multimedia Computing and Networking (MMCN2002), San Jose, CA*, Jan. 2002.

[29] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck. Network file system (NFS) version 4 protocol. Request for Comments - RFC 3530.

[30] E. Stark. SAMSON: A scalable active memory server on a network, Aug. 2003.

[31] T.M. Wong and J. Wilkes. My cache or yours? Making storage more exclusive. In *Proc. of the USENIX Annual Technical Conference*, pages 161–175, 2002.

[32] L. Xiao, X. Zhang, and S.A. Kubricht. Incorporating job migration and network RAM to share cluster memory resources. In *Proc. of the 9th IEEE Intl. Symposium on High Performance Distributed Computing (HPDC-9)*, pages 71–78, August 2000.

[33] K. Yocum, D. Anderson, J. Chase, S. Gadde, A. Gallatin, and A. Lebeck. Adaptive message pipelining for network memory and network storage. Technical Report CS-1998-10, Duke University Department of Computer Science, April 1998.

[34] Y. Zhou, L. Wang, D.W. Clark, and K. Li. Thread scheduling for out-of-core applications with memory server on multicomputers. In *Proceedings of the Sixth Workshop on Input/Output in Parallel and Distributed Systems, Atlanta, GA*, pages 57–67, 1999.

VITA
Kartik Gopalan

Address:	Computer Science Department, Florida State University Tallahassee, FL 32306-4530
Contact:	(850)-644-1685, fax: (850)-644-0058, e-mail: `kartik@cs.fsu.edu`

Education

State University of New York at Stony Brook	Computer Science	Ph.D. 2003
Indian Institute of Technology, Chennai, India	Computer Science	M.S. 1996
Delhi University	Computer Engineering	B.E. 1994

Academic Positions

Assistant Professor (Aug. 2003 – present)
Computer Science Department
Florida State University
Tallahasse, Florida 32306-4530

Other Professional Experience

Jan. 1998 – July. 2003	Research Assistant, Computer Science , Stony Brook University.
Dec. 2000 – Feb. 2001	Project Consultant, Siemens ICN.
Sept. 1999 – Feb. 2001	Lead Architect and Developer, Rether Network Inc.
Jun. 1999 – Aug. 1999	Summer Intern, Reuters Information Technology
Aug. 1997 – Jan. 1998	Teaching Assistant, Computer Science, Stony Brook University.
Jul. 1996 – Jun. 1997	Senior Software Engineer, Wipro Global Research and Development
Jul. 1994 – Jun. 1996	Research Assistant, Indian Institute of Technology, Chennai

List of Related Publications

1. C. Li, G. Peng, K. Gopalan and T. Chiueh, "Performance Guarantee for Cluster-Based Internet Services", In *23rd International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, Rhode Island, May 2003.
2. S. Sharma, J. Chen, W. Li, K. Gopalan and T. Chiueh, "Duplex : A Reusable Fault-Tolerance Extension for Network Access Devices". In *Intl. Conference on Dependable Systems and Networks (DSN) 2003*, San Francisco, CA, June 2003.

List of Other Significant Publications

1. K. Gopalan, T. Chiueh and Y.J. Lin, "Delay Budget Partitioning to Maximize Network Resource Usage Efficiency". To appear in *Proc. of INFOCOM'04*, Hong Kong, China, March 2004.

2. S. Sharma, K. Gopalan, S. Nanda and T. Chiueh, "Viking: A Multi-Spanning-Tree Ethernet Architecture for Metropolitan Area and Cluster Networks". To appear in *Proc. of INFOCOM'04*, Hong Kong, China, March 2004.
3. K. Gopalan and T. Chiueh, "Improving Route Lookup Performance Using Network Processor Cache", In *SC2002 High Performance Networking and Computing*, Baltimore, MD, November 2002.
4. S. Sharma, K. Gopalan, N. Zhu, G. Peng, P. De, T. Chiueh, "Implementation Experiences of Bandwidth Guarantee on a Wireless LAN", In *ACM/SPIE Multimedia Computing and Networking (MMCN2002)*, San Jose, CA, January 2002.
5. K. Gopalan and T. Chiueh, "Multi-Resource Allocation and Scheduling for Periodic Soft Real-Time Applications", In *ACM/SPIE Multimedia Computing and Networking (MMCN2002)*, San Jose, CA, January 2002.
6. P. Pradhan, K. Gopalan and T. Chiueh, "Design Issues in System Support for Programmable Routers", In *8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Schloss Elmau, Germany, May 2001.

Synergistic Activities

- Reviewer for IEEE Infocom 2004.
- Reviewer for ACM SAC 2004.
- Reviewer for University of Missouri Research Board (UMRB) 2004.
- Reviewer for IEEE Globecom 2003.
- Reviewer for Journal of Parallel and Distributed Computing 2003.
- Reviewer for IEEE Transactions on Networking 2002.
- Reviewer for New York Metro Area Networking Workshop 2002.

Collaborators & Other Affiliations

a. Collaborators and Co-authors in Past 48 Months Tzi-cker Chiueh (Stony Brook University), Jiawu Chen (Stony Brook University), Pradipta De (Stony Brook University), Chang Li (Stony Brook University), Wei Li (Stony Brook University), Yow-Jian Lin (Stony Brook University), Susanta Nanda (Stony Brook University), Anindya Neogi (IBM Research, India) Gang Peng (Stony Brook University), Prashant Pradhan (IBM Watson Research), Ashish Raniwala (Stony Brook University), Srikant Sharma (Stony Brook University), Ningning Zhu (Stony Brook University),

b. Graduate advisors Tzi-cker Chiueh (Ph.D., State University of New York at Stony Brook), C. Siva Rama Murthy (M.S., Indian Institute of Technology, Chennai, India).

c. Thesis advisor No students yet. New faculty.

SUMMARY PROPOSAL BUDGET

YEAR 1

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION
Florida State University

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR
Kartik Gopalan

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Kartik Gopalan - Assistant Professor**	0.00	0.00	1.00	$ 9,270	$
2.					
3.					
4.					
5.					
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0	
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	0.00	0.00	1.00	9,270	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0	
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0	
3. (2) GRADUATE STUDENTS				35,000	
4. (0) UNDERGRADUATE STUDENTS				0	
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0	
6. (0) OTHER				0	
TOTAL SALARIES AND WAGES (A + B)				44,270	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				1,842	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				46,112	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.)					
1 8-port Gigabit Ethernet Switch $ 400					
3 PCs 6,000					
TOTAL EQUIPMENT				6,400	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				1,500	
2. FOREIGN				2,500	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ 0					
2. TRAVEL 0					
3. SUBSISTENCE 0					
4. OTHER 0					
TOTAL NUMBER OF PARTICIPANTS (0) TOTAL PARTICIPANT COSTS				0	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				1,000	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0	
3. CONSULTANT SERVICES				0	
4. COMPUTER SERVICES				0	
5. SUBAWARDS				0	
6. OTHER				9,754	
TOTAL OTHER DIRECT COSTS				10,754	
H. TOTAL DIRECT COSTS (A THROUGH G)				67,266	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) **23512 (Rate: 46.0000, Base: 51112)**					
TOTAL INDIRECT COSTS (F&A)				23,512	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				90,778	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ 90,778	$

M. COST SHARING PROPOSED LEVEL $ **Not Shown** AGREED LEVEL IF DIFFERENT $

PI/PD NAME
Kartik Gopalan

ORG. REP. NAME*
Kirby kemper

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

 ***ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET**

SUMMARY PROPOSAL BUDGET

YEAR 2

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION: **Florida State University**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Kartik Gopalan**

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Kartik Gopalan - Assistant Professor**	0.00	0.00	2.00	$ 19,096	$
2.					
3.					
4.					
5.					
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0	
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	0.00	0.00	2.00	19,096	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0	
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0	
3. (2) GRADUATE STUDENTS				36,050	
4. (0) UNDERGRADUATE STUDENTS				0	
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0	
6. (0) OTHER				0	
TOTAL SALARIES AND WAGES (A + B)				55,146	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				3,616	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				58,762	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.)					
2 PCs $ 4,000					
TOTAL EQUIPMENT				4,000	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				1,500	
2. FOREIGN				2,500	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ 0					
2. TRAVEL 0					
3. SUBSISTENCE 0					
4. OTHER 0					
TOTAL NUMBER OF PARTICIPANTS (0) TOTAL PARTICIPANT COSTS				0	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				1,000	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0	
3. CONSULTANT SERVICES				0	
4. COMPUTER SERVICES				0	
5. SUBAWARDS				0	
6. OTHER				10,730	
TOTAL OTHER DIRECT COSTS				11,730	
H. TOTAL DIRECT COSTS (A THROUGH G)				78,492	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)					
29331 (Rate: 46.0000, Base: 63762)					
TOTAL INDIRECT COSTS (F&A)				29,331	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				107,823	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ 107,823	$

M. COST SHARING PROPOSED LEVEL $ **Not Shown** AGREED LEVEL IF DIFFERENT $

PI/PD NAME: **Kartik Gopalan**

ORG. REP. NAME*: **Kirby kemper**

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

 ***ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET**

SUMMARY PROPOSAL BUDGET

Cumulative

FOR NSF USE ONLY		
PROPOSAL NO.	DURATION (months) Proposed	Granted
AWARD NO.		

ORGANIZATION: **Florida State University**

PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR: **Kartik Gopalan**

A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)	NSF Funded Person-months CAL	ACAD	SUMR	Funds Requested By proposer	Funds granted by NSF (if different)
1. **Kartik Gopalan - Assistant Professor**	0.00	0.00	3.00	$ 28,366	$
2.					
3.					
4.					
5.					
6. ( ) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0	
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	0.00	0.00	3.00	28,366	
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)					
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0	
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0	
3. (4) GRADUATE STUDENTS				71,050	
4. (0) UNDERGRADUATE STUDENTS				0	
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0	
6. (0) OTHER				0	
TOTAL SALARIES AND WAGES (A + B)				99,416	
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				5,458	
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				104,874	
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING $5,000.) $ 10,400					
TOTAL EQUIPMENT				10,400	
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				3,000	
2. FOREIGN				5,000	
F. PARTICIPANT SUPPORT COSTS					
1. STIPENDS $ 0					
2. TRAVEL 0					
3. SUBSISTENCE 0					
4. OTHER 0					
TOTAL NUMBER OF PARTICIPANTS (0) TOTAL PARTICIPANT COSTS				0	
G. OTHER DIRECT COSTS					
1. MATERIALS AND SUPPLIES				2,000	
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0	
3. CONSULTANT SERVICES				0	
4. COMPUTER SERVICES				0	
5. SUBAWARDS				0	
6. OTHER				20,484	
TOTAL OTHER DIRECT COSTS				22,484	
H. TOTAL DIRECT COSTS (A THROUGH G)				145,758	
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)					
TOTAL INDIRECT COSTS (F&A)				52,843	
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				198,601	
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.C.6.j.)				0	
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				$ 198,601	$

M. COST SHARING PROPOSED LEVEL $ **Not Shown** AGREED LEVEL IF DIFFERENT $

PI/PD NAME: **Kartik Gopalan**

ORG. REP. NAME*: **Kirby kemper**

FOR NSF USE ONLY		
INDIRECT COST RATE VERIFICATION		
Date Checked	Date Of Rate Sheet	Initials - ORG

C ***ELECTRONIC SIGNATURES REQUIRED FOR REVISED BUDGET**

Budget Justification Page

Because the proposed research is experimental in nature, the project will require five PCs and one Gigabit Ethernet switch. The five PCs will be used by the PI and the research assistant as the base experimental platform to develop and test the Anemone architecture. Four of these PCs will be used as clients and servers and one will be used as the intermediate memory engine. The 8-port gigabit switch will provide connectivity between the machines in the experimental platform. Additional ports will provide room for expanding the cluster in the second year of the project.

Salary increases are estimated at 3% per year.

The amount requested under section G.6 includes tuition for a graduate student with an anticipated 10% annual increase.

The amount requested for travel will be used by the PI and research assistant for attending conferences to present papers/posters related to the Anemone research project.

The amount requested for materials and supplies will cover printing and presentation costs related to Anemone research such as toner cartridges, paper, transparencies etc.

Current and Pending Support

(See GPG Section II.D.8 for guidance on information to include on this form.)

The following information should be provided for each investigator and other senior personnel. Failure to provide this information may delay consideration of this proposal.

Investigator: **Kartik Gopalan**

Other agencies (including NSF) to which this proposal has been/will be submitted.

Support: ☐ Current ☒ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support
Project/Proposal Title: **Anemone: An Adaptive Network Memory Engine**

Source of Support: **NSF**
Total Award Amount: $ **201,348** Total Award Period Covered: **06/01/04 - 05/31/05**
Location of Project: **Florida State University**
Person-Months Per Year Committed to the Project. Cal: **0.00** Acad: **0.00** Sumr: **3.00**

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support
Project/Proposal Title:

Source of Support:
Total Award Amount: $ Total Award Period Covered:
Location of Project:
Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support
Project/Proposal Title:

Source of Support:
Total Award Amount: $ Total Award Period Covered:
Location of Project:
Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support
Project/Proposal Title:

Source of Support:
Total Award Amount: $ Total Award Period Covered:
Location of Project:
Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support
Project/Proposal Title:

Source of Support:
Total Award Amount: $ Total Award Period Covered:
Location of Project:
Person-Months Per Year Committed to the Project. Cal: Acad: Summ:

*If this project has previously been funded by another agency, please list and furnish information for immediately preceding funding period.

FACILITIES, EQUIPMENT & OTHER RESOURCES

FACILITIES: Identify the facilities to be used at each performance site listed and, as appropriate, indicate their capacities, pertinent capabilities, relative proximity, and extent of availability to the project. Use "Other" to describe the facilities at any other performance sites listed and at sites for field studies. USE additional pages as necessary.

Laboratory:

Clinical:

Animal:

Computer: **The Computer Science department has 8 UNIX servers, which include several ULTRA-SPARCs. Network Connections, file servers, printers, backups, and system assistance and maintenance for requested PCs will be provided by the Computer Science department.**

Office:

Other: **The FSU computer network reaches throughout the campus and currently connects more than 15,000 systems. Eighty-two campus buildings are on the network, including seventy-four connected by optical fiber to a high speed backbone.**

MAJOR EQUIPMENT: List the most important items available for this project and, as appropriate identifying the location and pertinent capabilities of each.

OTHER RESOURCES: Provide any information describing the other resources available for the project. Identify support services such as consultant, secretarial, machine shop, and electronics shop, and the extent to which they will be available for the project. Include an explanation of any consortium/contractual arrangements with other organizations.