

# Khaled Elmeleegy

---

## CONTACT INFORMATION

Rice University  
Department of Computer Science  
6100 Main St. MS132  
Houston, TX 77005 USA

*Voice (Cell):* (713) 933-5994  
*Fax:* (713) 348-5930  
*E-mail:* kdiaa@cs.rice.edu  
*WWW:* www.cs.rice.edu/~kdiaa

## RESEARCH INTERESTS

**General Area:** Experimental software systems including networked systems, operating and distributed systems. I am also interested in protocol design for networked systems.  
**Topics:** Reliability and scalability of local area networks, system support for high performance network servers.

## EDUCATION

**Rice University**, Houston, Texas USA

Ph.D. in Computer Science – (expected May 2008)

- Dissertation Topic: “Enhancing the reliability and the scalability of Ethernet networks”

M.S. in Computer Science, May 2004

- Dissertation Topic: “Lazy asynchronous I/O for event driven servers”

Advisor: First worked under the supervision of Prof. Willy Zwaenepoel, then after he left Rice I worked under the supervision of Prof. Alan L. Cox.

**Alexandria University**, Alexandria, Egypt

B.S. with distinction and honors in Computer Science and Automatic Control, August, 1998

## PROFESSIONAL EXPERIENCE

**Rice University**, Houston, Texas USA

*Research Assistant*

**August, 2001 - present**

Includes both research and coursework for the Ph.D. and the Masters degrees.

**IBM Thomas J. Watson Research Center**, Yorktown Heights, New York USA

*Summer Intern*

**May - August, 2003**

Working with the K42 operating system group.

**Alexandria University**, Alexandria, Egypt

*Teaching and Research Assistant*

**January, 2000 - August 2001**

Includes teaching various undergraduate courses plus doing research work on concurrency control in database management systems.

**MobiNiL** (a cell phone operator), Cairo, Egypt

*Software Engineer*

**April - December, 1999**

Working in the IT department.

## TEACHING EXPERIENCE

**Rice University**, Houston, Texas USA

I was the teaching assistant for both graduate and undergraduate courses. Specifically, I was the teaching assistant for the following courses:

- Operating Systems and Concurrent Programming.
- Distributed Systems.
- Intermediate Programming.
- Automata, Formal Languages, and Computability.

**Alexandria University**, Alexandria, Egypt

I was the teaching assistant for the following undergraduate courses:

- Operating Systems.
- Database Management Systems.
- Introductory programming for non-CS majors.

REFEREED  
PUBLICATIONS

1. Khaled Elmeleegy, and Alan L. Cox “EtherProxy: Scaling The Ethernet By Suppressing Broadcast Traffic”. In submission.
2. Khaled Elmeleegy, Alan L. Cox, and T. S. Eugene Ng. “Understanding And Mitigating The Effects Of Count To Infinity In Ethernet Networks”. To appear in IEEE/ACM Transactions on Networking.
3. Khaled Elmeleegy, Alan L. Cox, and T. S. Eugene Ng. “EtherFuse: An Ethernet Watchdog”. In the proceedings of ACM SIGCOMM 2007, Kyoto, Japan, August 2007.
4. Khaled Elmeleegy, Alan L. Cox, and T. S. Eugene Ng. “On Count-to-Infinity Induced Forwarding Loops in Ethernet Networks”. In the proceedings of INFOCOM 2006, Barcelona, Spain, April 2006.
5. Anupam Chanda, Khaled Elmeleegy, Alan L. Cox, and Willy Zwaenepoel. “Causeway: Support For Controlling And Analyzing The Execution Of Web-Accessible Applications”. In the proceedings of Middleware 2005, Grenoble, France, November 2005.
6. Anupam Chanda, Khaled Elmeleegy, Alan L. Cox, and Willy Zwaenepoel. “Causeway: Operating System Support For Controlling And Analyzing The Execution Of Distributed Programs”. In the proceedings of the Tenth Workshop on Hot Topics in Operating Systems (HotOS-X), Santa Fe, NM, June 2005.
7. Khaled Elmeleegy, Anupam Chanda, Alan L. Cox, and Willy Zwaenepoel. “A Portable Kernel Abstraction For Low-Overhead Ephemeral Mapping Management”. In the proceedings of the USENIX 2005 Annual Technical Conference, Anaheim, CA.
8. Khaled Elmeleegy, Anupam Chanda, Alan L. Cox, and Willy Zwaenepoel. “Lazy Asynchronous I/O For Event Driven Servers”. In the proceedings of the USENIX 2004 Annual Technical Conference, Boston, MA.

OTHER  
PUBLICATIONS

1. Khaled Elmeleegy, Alan L. Cox, and T. S. Eugene Ng. “Supplemental Note on Count-to-Infinity Induced Forwarding Loops in Ethernet Networks”. Technical Report TR06-878, Department of Computer Science, Rice University, 2006.
2. Anupam Chanda, Khaled Elmeleegy, Romer Gil, Alan L. Cox, and Willy Zwaenepoel, “An Efficient Threading Model to Boost Server Performance”. Technical Report TR04-440, Department of Computer Science, Rice University, 2004.

TALKS

1. “Enhancing Ethernet’s Reliability”, Invited talk at Purdue University, November 2007.
2. “EtherFuse: An Ethernet Watchdog”, Conference talk at SIGCOMM, August 2007.
3. “On Count-to-Infinity Induced Forwarding Loops in Ethernet Networks”, Conference talk at INFOCOM, April 2006.
4. “A Portable Kernel Abstraction For Low-Overhead Ephemeral Mapping Management”, Conference talk at the USENIX Annual Technical Conference, April 2005.
5. “Lazy Asynchronous I/O For Event Driven Server”, Conference talk at the USENIX Annual Technical Conference, June 2004.

HONORS AND  
ACCOMPLISHMENTS

- Student travel grants for the conferences OSDI 2006, NSDI 2007, and SOSP 2007.
- Fellowship from Rice University to join its Ph.D. program at the Computer Science department.
- The Onsi Sawiris Scholarship award (2000/2001), which is an Egyptian scholarship for outstanding Egyptian students in the field of engineering to continue their studies the US.
- Teaching assistance scholarship from the Faculty of Engineering, University of Alexandria (January 2000 - July 2001).
- Receiving a governmental scholarship for outstanding Egyptian undergraduate students, 1993-1998.

PATENTS

Khaled Elmeleegy, Alan L. Cox, and T. S. Eugene Ng. "System and Method For Preventing Count-to-Infinity Problems in Ethernet Networks", Filed April 2007.

SOFTWARE  
ARTIFACTS

(Names refer to projects listed at pages 5-6.)

- **EtherProxy:** A software prototype of an Ethernet device that suppresses broadcast traffic in Ethernet.
- **EtherFuse:** A software prototype of an Ethernet device that detects anomalies in traffic patterns, then deduces Ethernet failures and consequently takes corrective actions.
- **Software Ethernet Bridge:** A software prototype of an Ethernet bridge based on the latest IEEE standard spanning tree protocol.
- **Bridge Simulator:** A simulator of a network of Ethernet bridges running the spanning tree protocol based on the latest IEEE standard. The simulator also includes a modified version of the latest spanning tree protocol that I designed to fix a flaw in the existing protocol.
- **Causeway:** An operating system support facilitating development of meta-applications, like priority scheduling and performance debugging, to control and analyze distributed programs.
- **The sf\_buf interface:** A portable kernel interface that facilitates the efficient usage of kernel virtual-to-physical ephemeral memory mappings. It was implemented in the FreeBSD kernel.
- **liblaio:** A user level library that implements Lazy Asynchronous I/O.
- **ServLib:** A library and kernel support for high performance multi-threaded servers.

REFERENCES

Available upon request.

I have worked on the following projects:

**Enhancing Ethernet's Reliability and Scalability:** Ethernet is a dominant networking technology. It has been used for tens of years by millions of people. Surprisingly, it still has significant reliability and scalability problems. My dissertation research addresses those problems.

Ethernet's reliability problems are its sometimes-slow convergence time, the occasional formation of forwarding loops in its active topology, and packet misforwarding in the event of some failures. A main reason for the Ethernet's slow convergence is the count-to-infinity problem. This problem results from stale information about the network topology persisting in the network in the event of some failures. This confuses the Ethernet's spanning tree protocol and can substantially slow down the healing of the network. I have thoroughly studied this problem, its causes, and its effects. I have also uncovered races in the Ethernet specification, which when coupled with the count-to-infinity problem can lead to a temporary forwarding loop in the network. Forwarding loops could lead to network congestion and instability. I take two different approaches to address these reliability problems. First, I take a fully backward compatible approach to mitigate the problems by introducing a new device that is to be plugged into the network, called EtherFuse. The second approach is at the protocol level where I augment the RSTP protocol, the current Ethernet spanning tree protocol, to fix its design flaw causing the count-to-infinity problem.

On the other hand, Ethernet's main scalability problem is that many protocols that use it rely on broadcast traffic. To address this problem, I also take a fully backward compatible approach to suppress broadcast traffic by introducing a new device that is to be plugged into the network, called EtherProxy.

- **EtherFuse:** A new device that can be inserted into an existing Ethernet to speed the reconfiguration of the spanning tree and detects and breaks forwarding loops. Moreover, in event of failures, it avoids packet misforwarding in an Ethernet network. The EtherFuse is inserted at redundant links in the network. It is backward compatible with any of Ethernet's standard spanning tree protocols and requires no change to the existing hardware, software, or protocols. I've implemented a prototype of the EtherFuse using the Click modular router and used it on emulab to experimentally demonstrate the effectiveness of the EtherFuse.
- **RSTP with Epochs:** A backward compatible fix to the current Ethernet spanning tree protocol (RSTP) that eliminates the count to infinity problem from the RSTP protocol and thus substantially reducing its convergence time to a roundtrip time across the network. It relies on adding sequence numbers to protocol messages which allows for associating messages to epochs. This allows for detecting and eliminating messages carrying stale information. I've built a simulator for RSTP and RSTP with Epochs to demonstrate the effectiveness of my solution.
- **EtherProxy:** A new device that can be inserted into an existing Ethernet to suppress broadcast traffic allowing an Ethernet network to scale to a much larger size. Previously, people relied on breaking an Ethernet network into smaller segments, e.g. using VLANs. Then, those segments are connected together using layer 3 routers. This is because a router does not forward broadcast traffic or multicast traffic. Thus, each segment is a separate broadcast domain. This approach requires plenty of error-prone manual configuration. First, if VLANs are used, they must be created and configured. Second, subnets need to be created. Third, address assignment needs to be managed. Forth, routers connecting network segments need to be configured. Moreover, this approach is costlier than just using Ethernet alone. This is because layer 3 routers are significantly more expensive than layer 2 Ethernet switches. Conversely, an EtherProxy requires no configuration and is backward compatible with existing software, hardware, and protocols. It relies on caching for broadcast suppression. Broadcast is mainly used in the Ethernet to discover resources or services. This discovery process can happen over and over again for the the same resource or service from different hosts. An EtherProxy caches this information and whenever a

host wants to discover a resource or a service, the EtherProxy intercepts the broadcast discovery request. Then, using its cached information, it either directly replies to the host from its cache or replaces the broadcast address of the request with the appropriate unicast address of the requested service it has in its cache. I've implemented a prototype of the EtherProxy using the Click modular router and used it on emulab to experimentally demonstrate the effectiveness of the EtherProxy.

**Studying the SIP Express Router (SER)**, a frequently used server in the industry to enable VoIP services, and identifying its performance bottlenecks. Then making changes to the server architecture to eliminate those bottlenecks to improve the server throughput and scalability.

**Causeway:** An operating system support facilitating development of meta-applications, like priority scheduling and performance debugging, to control and analyze distributed programs. Meta-applications use Causeway to inject and access metadata on application execution paths to implement their specific goals. Causeway has two components: (1) Interfaces to inject and access metadata, and (2) Automatic propagation of metadata. Using Causeway we could rapidly implement a distributed priority enforcement system where the priority of a task is injected and propagated as metadata, and accessed to implement global priority scheduling across different components of a distributed system. This required writing only about 150 lines of code on top of Causeway. With this system we obtained global priority enforcement on an implementation of the TPC-W benchmark. A prototype of Causeway was built in the FreeBSD kernel and user level libraries.

**The sf\_buf interface:** A portable kernel interface that facilitates the efficient usage of kernel virtual-to-physical ephemeral mappings. The interface was used in many of the kernel subsystems, e.g pipes, memory disks and the networking subsystems. The evaluation of such subsystems has shown significant performance improvement when the interface was used. This led to its adoption by the FreeBSD community and now it ships with the stock FreeBSD kernel.

**The Lazy Asynchronous I/O (LAIO) library:** A new API that can be used as a wrapper around any system call to make it non-blocking. The goal is to make event-driven programming simpler - by avoiding kernel non-blocking I/O or helper threads. Then to demonstrate the effectiveness of LAIO, I evaluated its performance by modifying two event driven servers (thttpd and Flash) to use LAIO then benchmarking those two servers.

**Benchmarking the K42 operating system** using the TPCW benchmark and the MySQL database running on a multi-processor machine at IBM T.J. Watson (K42 group). Then, studying the lock implementation provided by K42 and the way it impacts the applications' performance especially on multi-processor machines. This was to identify locks-related bottlenecks and to improve the lock implementation to eliminate those bottlenecks.

**ServLib:** A threading library that exports the pthreads interfaces, but achieves high performance by aggressive management of stacks and kernel threads, and execution of server operations in an event-driven manner. Servlib was evaluated by using it with Apache and MySQL and it achieved better performance than both Linux (1-1) pthreads and FreeBSD's (N-1) pthreads.

**A decentralized quota management system for PAST, the peer-to-peer archival storage facility:** The approach used is based on having the quota information for every user of the system replicated on multiple nodes in the system. Then, to update this information, the system relies on Byzantine fault tolerance protocols among the nodes storing the replicas to be able to tolerate faulty or malicious nodes.