Kernel-Middleware Interaction to Support Adaptation in Pervasive Computing Environments

F. A. Samimi
P. K. McKinley
Department of Computer Science
Michigan State University
East Lansing, Michigan
{farshad, mckinley}@cse.msu.edu

S. M. Sadjadi *
School of Computer Science
Florida International University
Miami, Florida
sadjadi@cs.fiu.edu

P. Ge
Department of Computer Science
Michigan State University
East Lansing, Michigan
gepeng@cse.msu.edu

ABSTRACT

In pervasive computing environments, conditions are highly variable and resources are limited. In order to meet the needs of applications, systems must adapt dynamically to changing situations. Since adaptation at one system layer may be insufficient, cross-layer, or *vertical* approaches to adaptation may be needed. Moreover, adaptation in distributed systems often requires *horizontal* cooperation among hosts. This cooperation is not restricted to the source and destination(s) of a data stream, but might also include intermediate hosts in an overlay network or mobile ad hoc network. We refer to this combined capability as *universal* adaptation. We contend that the model defining interaction between adaptive middleware and the operating system is critical to realizing universal adaptation. We explore this hypothesis by evaluating the *Kernel-Middleware eXchange (KMX)*, a specific model for cross-layer, cross-system adaptation. We present the KMX architecture and discuss its potential role in supporting universal adaptation in pervasive computing environments. We then describe a prototype implementation of KMX and show results of an experimental case study in which KMX is used to improve the quality of video streaming to mobile nodes in a hybrid wired-wireless network.

Keywords: Adaptive middleware, pervasive computing, cross-layer adaptation, universal adaptation, multimedia communication, quality of service, video streaming, wireless network.

1. INTRODUCTION

Pervasive computing removes the traditional boundaries for how, when, and where humans and computers interact [29]. Software in such systems must adapt to changes in the surrounding physical environment as well as in the virtual environment. In particular, conditions at the "wireless edge" of the Internet are often highly dynamic, requiring software in mobile devices to balance several conflicting and possibly cross-cutting concerns, including quality-of-service on wireless connections, changing security policies, and energy consumption [15].

One approach to this problem is adaptive middleware [5, 11, 13, 28, 31, 24]. Since the traditional role of middleware is to hide resource distribution and platform heterogeneity from the business logic of applications, it is a logical place to put adaptive behavior related to concerns such as QoS, security, and energy management. Specifically, middleware can exploit application-level knowledge needed for adaptation, but can adapt the system transparently with respect to to the application code itself.

However, middleware alone cannot handle all types of adaptation. The operating system manages essential system resources that are typically not controllable from upper layers. Only the operating system can directly control adaptive behavior of hardware components, such as placing the network interface card in power save mode or adjusting the frequency of the CPU. Moreover, since the operating system is usually more responsive than application-level programs, implementing certain adaptive functionality there can produce more agile system behavior [19].

To address these issues, several *cross-layer* adaptation frameworks have been proposed recently [16, 8, 9, 3, 18]. In these systems, adaptive entities at different layers cooperate to realize system-wide adaptive behavior. For example, middleware can make better decisions by working with the operating system, which has a system-wide view of resource usage. Middleware might also ask the operating system to carry out certain tasks, such as custom CPU scheduling, to improve performance. Conversely, the operating system can incorporate application-specific semantics into tasks such as packet filtering and implementation of security policies.

This *vertical* cooperation among system layers is particularly important in pervasive computing contexts, where adaptation is needed frequently, but resources may be severely limited. However, adaptation in distributed environments also requires cooperation among multiple platforms. A common example of this *horizontal* cooperation is when a data stream is adapted at a source node via a particular action (such as enhanced error control or encryption) that requires a corresponding action at the receiving node(s). In addition, if the data stream traverses *multiple* hosts, as in an overlay network or mobile ad hoc network, then some or all of those nodes might also participate in the adaptation process, possibly using local cross-layer mechanisms. We refer to this combination as *universal* adaptation.

We contend that the model for kernel-middleware cooperation is a critical element in realizing universal adaptation. To explore this

*This research was performed while this author was a graduate student at Michigan State University.

2nd Workshop on Middleware for Pervasive and Ad-Hoc Computing Toronto, Canada

issue, we propose and evaluate the *Kernel-Middleware eXchange (KMX)*, a model for interaction between middleware and the operating system across a collection of cooperating nodes, such as a mobile ad hoc network or an overlay network. KMX is intended to be a general framework into which specific software adaptation technologies can be inserted. We have constructed a prototype that incorporates tools and techniques from the RAPIDware adaptive middleware project [1]. We present an experimental case study that illustrates the use of KMX to enhance video multicast streaming to mobile nodes in a hybrid wired-wireless network.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we introduce the KMX model and its role in universal adaptation. Section 4 describes the constituent components of our KMX prototype and how they interoperate. In Section 5, we present and discuss the case study in video streaming. Finally, in Section 6, we summarize the paper and discuss possible future research directions.

2. RELATED WORK

The KMX model and the case study are related to several research areas, but most closely to cross-layer adaptation and overlay networking. We briefly review projects that represent current research in the these areas.

Several recent projects address cross-layer cooperation for system-wide adaptation. Odyssey [16] is an early application-aware adaptation framework that extends the operating system and uses kernel-level interception and communication to monitor resource expectations and availability. The GRACE project [3] uses cross-layer approach to support optimized adaptation in multimedia applications within the constraints of energy, time, and bandwidth. The main idea is to coordinate local adaptations at the hardware, operating system, and application layers to achieve optimal global adaptation. DEOS [19] also addresses adaptation to enhance QoS in distributed multimedia applications. DEOS uses ELinux [21], which facilitates cross-layer adaptation by extending user/kernel communication and several subsystems of the Linux kernel. More recently, the developers of DEOS have investigated automatic generation of adaptive software [20]. The Milly Watt project [8] introduces models to make mobile computing more power efficient. It addresses the partnership between applications, operating system, and hardware to attain energy efficiency, for example, power-aware memory management [12].

Generally, the above projects focus on coordination between multiple layers on a single platform, as well as the exchange of events between the end nodes of a data stream. The KMX model also addresses these concerns, but in addition focuses on distributed coordination of adaptive actions among a collection of hosts cooperating to achieve a specific communication goal. In this sense, KMX is related to coordination mechanisms for overlay networks [4, 30] and mobile ad hoc networks [7, 6]. For example, Conductor [30] is an overlay network framework for distributed adaptation of data streams. The main idea is to adapt a data flow at intermediary nodes, on its way from the source to the destination, transparently to the applications running on the end nodes. On the other hand, Roofnet [7] is an experimental multi-hop wireless ad hoc network in which system-level mechanisms (specifically, MAC-level retransmissions) are used to enhance upper layer components (specifically, the routing protocol executed on each node).

Conductor and Roofnet use systems-oriented approaches to distributed coordination. While KMX has similar goals, it uses adaptive middleware as the key enabling technology for distributed adaptation. Moreover, rather than addressing a particular type of adaptation, KMX is intended to serve as a more general framework for building and extending distributed adaptive frameworks over commodity systems.

3. KMX MODEL

KMX has its origins in the RAPIDware project [1], which focuses on the design of high assurance adaptive middleware. RAPIDware uses a combination of computational reflection [14] and aspect-oriented programming [10] to support software recomposition at run time. Together, these technologies enable dynamic configuration by insertion of software *sensors* to detect environmental changes and insertion of *actuators* to realize corresponding responses. One example is MetaSockets [26], which extend regular Java sockets to provide adaptable communication services. MetaSockets enable middleware or an application to monitor the communication status and to insert, remove, and configure adaptation filters at run time, for example, using forward error correction (FEC) to make the data communication resilient to packet loss. Further, we have developed a generator framework, called TRAP [25], that enables a developer to augment a software component with a middleware infrastructure that supports adaptation transparently from applications. For example, TRAP can be used to extend an application to use MetaSockets, transparently to the original application code.

The KMX model extends these concepts (sensors, actuators, and adaptation transparency) to include not only middleware, but also the operating system kernel. Figure 1 shows the main components of the KMX architecture including *coordination and decision-making unit*, *sensors*, *actuators*, and *transient proxies*. The coordination and decision-making unit is a middleware component that manages adaptation globally and triggers required adaptation at multiple layers according to the specified policies. For example, if the network bandwidth becomes overloaded, it can decide which application must decrease its network bandwidth usage and accordingly triggers the required adaptation.

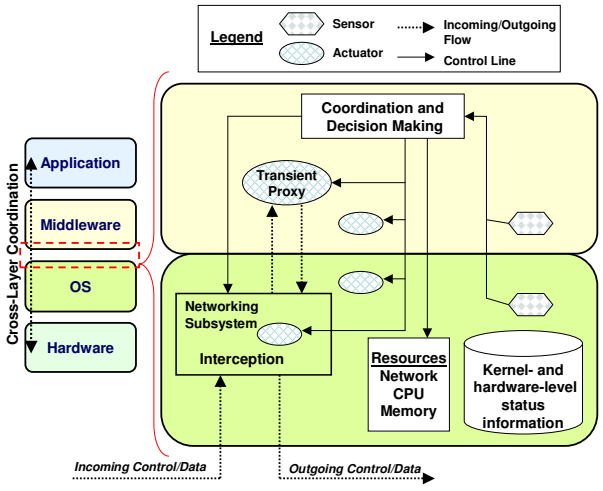

Figure 1: KMX architecture.

Sensors reside in the middleware and operating system layers and monitor the status of various subsystems to gather information for the coordination and decision-making unit. A specific sensor might monitor the number of packets dropped at the kernel due to buffer overflow and inform the coordination and decision-making

unit. On the other hand, actuators are middleware components or operating system modules that apply adaptation. For example, an actuator in the middleware might insert a component that handles frequent disconnections that occur over the wireless links, when a mobile user switches from a wired network to a wireless network. An actuator in the operating system kernel could be used to adapt the CPU frequency so as to save energy while providing the expected processing power.

Data flow interception is carried out in the networking subsystem. When a node on an ad hoc network is routing packets for other nodes, it can redirect a specific data stream to the user level for adaptation. A *transient proxy* is a specific type of user-level actuator that adapts data streams transparently to the application. For example, transient proxies can encrypt and decrypt data streams that carry sensitive information, implement adaptive error control, distill data streams to reduce bandwidth consumption, and so forth.

Figure 2 illustrates how the KMX model can be used to support universal adaptation by combining *vertical* and *horizontal* cooperation. Consider the adaptation of data streams that flow over a network. The operating system acts as a substrate that interacts with middleware and redirects a data flow into the middleware whenever adaptation is required. The adaptation is realized by transient proxies implemented as adaptive middleware components. For example, in a mobile ad hoc network, a wireless communication channel between two nodes might suddenly become noisy, with unacceptably high packet loss. In this case, first, these two nodes cooperate to detect the high packet loss rate. Next, to improve the situation, one of these nodes begins to intercept the data stream and apply stronger error correction methods, before forwarding packets over the noisy link. Accordingly, this node needs to inform the node at the other end of the link to participate in error control handling. Moreover, these actions should be transparent to the other nodes in the ad hoc network. In Section 5, we present an experimental case study that deals with such a situation.

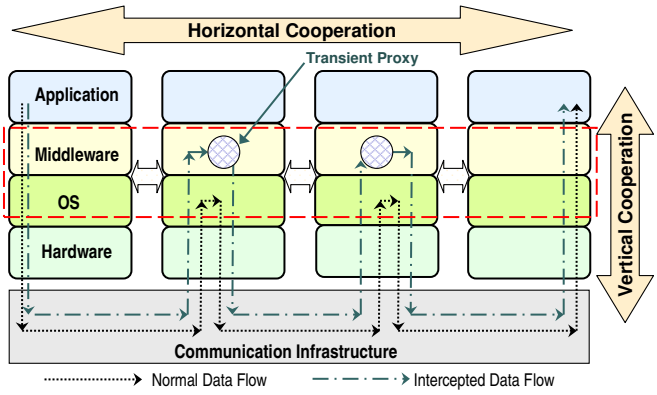

Figure 2: KMX universal adaptation framework.

4. PROTOTYPE IMPLEMENTATION

We have built a rudimentary KMX prototype for the case study presented in this paper. Although KMX is a generic model, we have deployed it on Linux, where most operating system services reside in the kernel. The prototype shows how the KMX model can be implemented using adaptive middleware technologies from the RAPIDware project and off-the-shelf facilities available in Linux.

We use MetaSockets [26] to build transient proxies for mobile environments. The composition and behavior of a MetaSocket can be adapted at run time in response to changing conditions. For example, a loss-detector filter can monitor the packet loss and another filter can perform forward error correction (FEC) encoding upon detection of intolerable packet loss. Figure 3 shows the MetaSendMSocket architecture. MetaSendMSocket is a meta-level component over the regular multicast socket that enables adaptation of a stream which is sent through the socket. A similar meta-level component enables configuration of filters in the receiver multicast socket corresponding to the filters at the sender socket. Details of the MetaSocket mechanism can be found in [26]. Further, we use TRAP [25] to weave the MetaSockets into the middleware components. TRAP/J, a Java implementation of TRAP, is a generator that takes as input a Java application and a list of classes that are to be made adaptable. TRAP/J uses the AspectJ compiler to generate an *adapt-ready* version of the application. Specifically, an aspect is generated for each of the specified classes, providing a hook that can be used to introduce new behavior at run time. In our case study, we use TRAP/J to enable run-time reconfiguration of transient proxies.

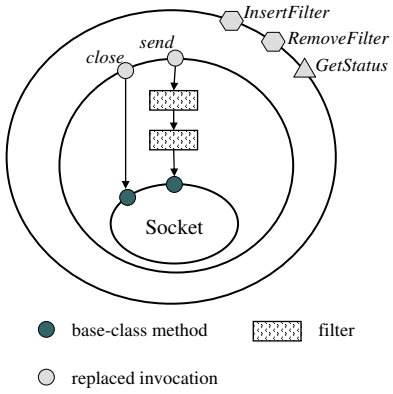

Figure 3: MetaSendMSocket architecture.

Finally, the implemented prototype uses off-the-shelf tools to configure the Linux kernel for routing and intercepting multicast packets. The SMCRoute [27] tool enables configuration of multicast routing in the Linux kernel. Specifically, a software component can use SMCRoute to customize multicast routes in the kernel routing subsystem. In addition, the Linux kernel supports packet manipulation by a framework called Netfilter. The iptables facility [23] in Linux can be used to configure Netfilter to intercept data streams that traverse the kernel routing subsystem. Specifically, a software component can use iptables to define rules for intercepting packets that pass through the kernel and meet a specific set of criteria (such as protocol type, sender address, and receiver port number). In our case study, when high packet loss is detected in a communication channel, packets belonging to a specified data stream arriving at a node can be intercepted and passed up to a transient proxy, which adapts the stream by FEC encoding the packets before they are sent to the next node.

5. CASE STUDY

We conducted a case study that uses the KMX prototype to enhance QoS in wireless multimedia streaming. Figure 4 shows the experimental testbed, a hybrid network that combines wired and wireless network segments. In our experiments, a video stream is multicasted over a local area network and is routed to a group of laptop computers operating in ad hoc mode. Wireless communi-

cation is prone to packet loss. In this experiment, FEC filters are inserted into the transient proxies as needed to improve the multimedia QoS for the wireless nodes. Specifically, we use block erasure codes [22], which compensate packet loss by sending a number of parity packets. These codes are especially beneficial in multicast streaming when the link layer retransmissions do not exist and packets are lost at multiple nodes independently. In the scenario presented here, we consider just one path in the multicast tree to show the effectiveness of the KMX model.Whenever an ad hoc node experiences intolerable packet loss, the video stream is intercepted and passed to the transient proxy at the Router node. The middleware proxies on the Router and Receiver nodes apply FEC encoding/decoding on the video stream transparently to the video application. In general, using transient proxies, adaptations such as FEC can be applied only to those parts of the network that need them.

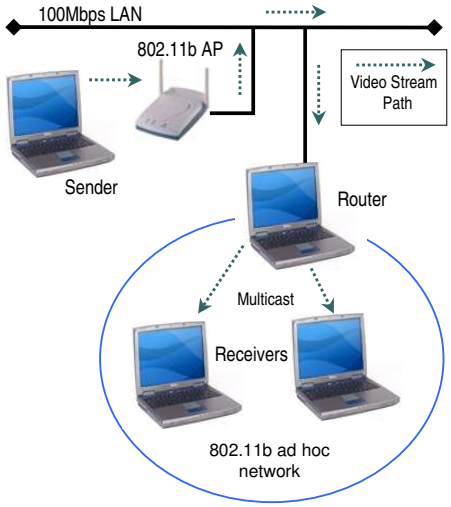

Figure 4: The experimental testbed.

5.1 Experiment Setup

The testbed consists of two Mobile Pentium 4 laptops with 1GB RAM (Router and Receiver nodes) and one Mobile Pentium III laptop with 256MB RAM (Sender node). The transmission power of the network adapter card on the Router (Cisco 340) has been reduced to 1mW to make it possible to conveniently perform the tests over short distances. The data rate over the ad hoc network is 2Mbps. The Sender and Receiver run Windows XP Professional, and the Router runs Linux Red Hat 9.

For video streaming and playback we used a video application developed at our laboratory, enabling us to instrument it for various measurements such as number of packets dropped due to delay. However, we emphasize that the KMX framework is general and off-the-shelf commercial applications could be used. The video clip is the first 55 seconds of a standard reference video [2]; its specifications are given in Table 1. Multicast packets are 1KB at the application-level.

The routing subsystem at the Router node is originally configured to to route the multicast packets, sent by the video application on the Sender, towards the Receivers in the wireless segment of the network. When packet loss in the wireless segment of the network increases, KMX can use iptables to configure the Linux

Table 1: Video specifications.

Video	Highway drive
Number of Frames	1375 (original highway drive contains 2001)
Frame Rate	25 fps
Playing Time	55 seconds
Encoding	Mpeg-4 avi Using DivX Pro 5.0.5 Codec 200 kbps encoding rate Max key frame interval: 12 frames
Video Size	352*288 (CIF)
File Size	1386 KB (on hard disk)
Streaming	1 KB packet-size (including a 48-byte application-layer header) 1905 packets totally

kernel on the Router to intercept the multicast packets sent by the Sender and redirect them to a transient proxy. Using TRAP/J, we have incorporated MetaSockets into the transient proxy so that FEC can be added dynamically. In these experiments, the transient proxy on the Router breaks each packet into two packets and performs FEC (4 , 2) encoding. Specifically, each packet is split into two packets, then four FEC encoded packets are created by the FEC encoder filter. Only two out of four encoded packets are needed at the Receiver for the FEC decoder filter to generate the original packet.

5.2 Empirical Results

We demonstrate the effect of video stream adaptation when an ad hoc node is experiencing high packet loss. Figure 5 shows the percentage of packets that arrive at a Receiver in the two situations. In the first situation, the Router forwards multicast packets without interception. In the second situation, the transient proxy augments the intercepted stream with FEC information. Figure 5 plots the average of 6 runs for two situations. FEC introduced by the transient proxy improves the overall packet reception rate from 72.9% to 91.4%.

Since the percentage of received packets does not directly reflect the effect on the video quality, we also use VQM [17] to evaluate the quality of the received video. VQM facilitates off-line perception-based estimation of video quality, in addition to the traditional peak signal to noise ratio measurement. We used video-conferencing model in VQM to measure the quality of the received video. This model is optimized for video-conferencing applications with bit rates from 10 kbps to 1.5 Mbps (the encoding rate of the video in this case study is 200 kbps).

Figure 6 shows the quality of the received video measured by VQM in the two situations described above. We configure VQM to break 52 second (1300 frames) of the video into six 12-second segments, such that each segment overlaps the last 4 seconds of the previous segment. Each segment is scored between 0 to 5, where 5 indicates imperceptible impairment in the received video. Table 2 shows the overall packet loss and the corresponding quality estimation in the two situations for the entire video (55 seconds). Adaptation by the transient proxy improves the VQM score from 2.13 to 3.94 (a 84.97% boost), which shows a significant improvement in the quality of the received video.

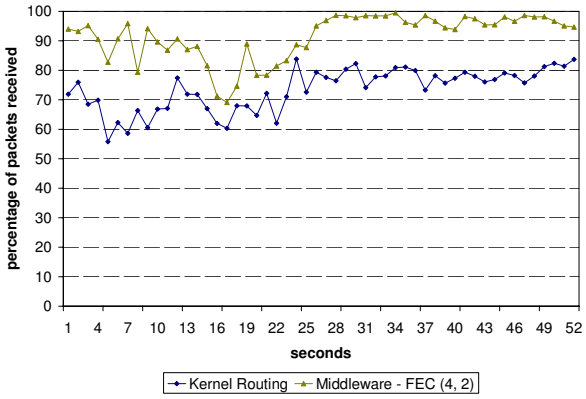

Figure 5: Percentage of packets received at the receiver.

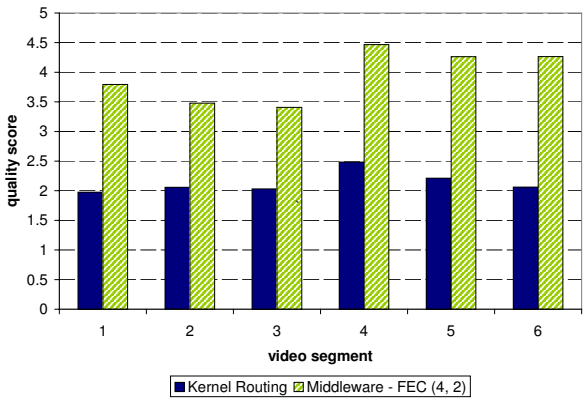

Figure 6: Video quality measurement.

5.3 Discussion

As can be seen from the above results, intercepting and applying FEC filters at the intermediary nodes can significantly improve quality of the received video. Another possible strategy for the above scenario is to perform end-to-end FEC encoding, with FEC encoding at the wired sender and FEC decoding at the receivers. However, such an approach consumes more resources and generates more traffic without providing a QoS better than the presented approach. This approach is not efficient because virtually all the packet loss occurs in the wireless segment of the network. Further, KMX interception can be applied selectively to streams experiencing poor conditions, and the error control can be implemented in a manner best suited to the data type.

Another issue in live multimedia streaming is delay, especially in interactive applications such as video conferencing. In such real-time applications, if packets reach the destination later than expected, the video player at the application layer will drop them. The delay introduced by the FEC encoding and decoding depends on the processing power and load of the intermediary nodes. If FEC processing cannot be performed fast enough, it may cause some packets to miss their playback deadline at the receiver. This situation can degrade, rather than improve, the quality of the received video. Therefore, in general, it may also be necessary to adapt the stream in other ways at the sender, for example, by changing the video encoding bit rate, in addition to FEC adaptation at the intermediary nodes.

	Kernel Routing	Middleware FEC (4, 2)
Total Packet Loss (middleware layer)	27.01%	8.6%
Root Cause Analysis [Blurring] / [Jerky Motion] / [Block Distortion]	[2.5%] / [45%] / [73.5%]	[0%] / [32%] / [59.83%]
VQM Score (video conferencing model)	2.13	3.94

Table 2: Overall packet loss and video quality.

6. CONCLUSIONS AND FUTURE WORK

The KMX project addresses the interaction between the middleware and operating system layers to support universal adaptation. The KMX approach is particularly well suited to mobile ad hoc networks and overlay networks, where a set of nodes cooperate with each other to accomplish services. The KMX prototype incorporates the adaptive middleware methods from the RAPIDware project and applies them on a commodity operating system. The presented experiment demonstrates a scenario where such a cooperation improves the quality of video multicasting over a hybrid wired-wireless network.

We intend to build more complex systems based on the commodity platforms to further investigate KMX capabilities. For example, we can deploy different types of transient proxies and configure them dynamically to provide QoS services. The KMX model can also be used to implement application specific (and adaptive) routing in overlay networks, universal energy management in MANETs, and dynamic security policies in both environments. Finding suitable nodes for the transient proxies and relocating them is also an interesting issue. These topics will be addressed in our future studies.

Further Information. Related papers and technical reports of the Software Engineering and Network Systems Laboratory can be found at the SENS website: `http://www.cse.msu.edu/sens`. Additional details on the RAPIDware project, including software downloads, can be found at: `http://www.cse.msu.edu/rapidware`.

Acknowledgements. This work was supported in part by the U.S. Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, and in part by National Science Foundation grants CCR-9912407, EIA-0000433, EIA-0130724, and ITR-0313142.

7. REFERENCES

[1] The RAPIDware Project. Michigan State University, Department of Computer Science and Engineering, `http://www.cse.msu.edu/rapidware`.

[2] Video Traces for Network Performance Evaluation. Arizona State University, `http://trace.eas.asu.edu/`.

[3] S. V. Adve, A. F. Harris, C. J. Hughes, D. L. Jones, R. H. Kravets, K. Nahrstedt, D. G. Sachs, R. Sasanka, J. Srinivasan, and W. Yuan. The Illinois GRACE Project: Global Resource Adaptation through CoopEration. In *Proceedings of the Workshop on Self-Healing, Adaptive, and self-MANaged Systems (SHAMAN) (held in conjunction with the 16th Annual ACM International Conference on Supercomputing)*, June 2002.

[4] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, pages 131–145, Banff, Canada, October 2001.

[5] G. Blair, G. Coulson, and N. Davies. Adaptive middleware for mobile multimedia applications. In *Proceedings of the 8th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pages 259–273, 1997.

[6] M. Conti, G. Maselli, G. Turi, and S. Giordano. Cross-layering in mobile ad hoc network design. *IEEE Computer*, 37(2):48–51, February 2004.

[7] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 134–146, September 2003.

[8] C. Ellis. The case for higher level power management. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, page 162. IEEE Computer Society, March 1999.

[9] J. Flinn and M. Satyanarayanan. Energy-aware adaptation for mobile applications. In *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*, pages 48–63, December 1999.

[10] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Videira Lopes, J. M. Loingtier, and J. Irwin. Aspect-oriented programming. In *Proceedings of the European Conference on Object-Oriented Programming (ECOOP)*, pages 220–242. Springer-Verlag LNCS 1241, June 1997.

[11] F. Kon, M. Román, P. Liu, J. Mao, T. Yamane, L. C. Magalhães, and R. H. Campbell. Monitoring , security, and dynamic configuration with the dynamicTAO reflective ORB. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2000)*, pages 121–143, New York, April 2000. Springer-Verlag.

[12] A. R. Lebeck, X. Fan, H. Zeng, and C. S. Ellis. Power aware page allocation. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*, pages 105–116, November 2000.

[13] T. Ledoux. OpenCorba: A reflective open broker. *Lecture Notes in Computer Science*, 1616:197–214, July 1999.

[14] P. Maes. Concepts and experiments in computational reflection. In *Object Oriented Programming Systems Languages and Applications (OOPSLA)*, pages 147–155. ACM Press, 1987.

[15] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng. Composing adaptive software. *IEEE Computer*, 37(7):56–64, 2004.

[16] B. D. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. R. Walker. Agile application-aware adaptation for mobility. In *Proceedings of the Sixteen ACM Symposium on Operating Systems Principles*, pages 276–287, October 1997.

[17] M. Pinson and S. Wolf. *Video Quality Measurement User's Manual*. U.S. Department of Commerce, February 2002.

[18] C. Poellabauer, H. Abbasi, and K. Schwan. Cooperative run-time management of adaptive applications and distributed resources. In *Proceedings of the 10th ACM Multimedia Conference*, pages 402–411, France, December 2002.

[19] C. Poellabauer and K. Schwan. Kernel support for the event-based cooperation of distributed resource managers. In *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2002)*, San Jose, California, September 2002.

[20] C. Poellabauer, K. Schwan, S. Agarwala, A. Gavrilovska, G. Eisenhauer, S. Pande, C. Pu, and M. Wolf. Service morphing: Integrated system- and application-level service adaptation in autonomic systems. In *Proceedings of the 5th Annual International Workshop on Active Middleware Services (AMS 2003)*, pages 93–102, Seattle, Washington, June 2003.

[21] C. Poellabauer, K. Schwan, R. West, I. Ganev, N. Bright, and G. Losik. Flexible user/kernel communication for real-time applications in ELinux. In *Proceedings of the Workshop on Real Time Operating Systems and Applications and Second Real Time Linux Workshop (in conjunction with RTSS 2000)*, November 2000.

[22] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, 27(2):24–36, April 1997.

[23] R. Russell. Linux 2.4 Network Address Translation (NAT) HOWTO. `http://www.iptables.org/`, January 2002.

[24] S. M. Sadjadi and P. K. McKinley. ACT: An adaptive CORBA template to support unanticipated adaptation. In *Proceedings of the 24th IEEE International Conference on Distributed Computing Systems (ICDCS'04)*, Tokyo, Japan, March 2004.

[25] S. M. Sadjadi, P. K. McKinley, B. H. Cheng, and R. K. Stirewalt. TRAP/J: Transparent generation of adaptable Java programs. In *Proceedings of the International Symposium on Distributed Objects and Applications (DOA'04)*, Agia Napa, Cyprus, October 2004. To appear.

[26] S. M. Sadjadi, P. K. McKinley, and E. P. Kasten. Architecture and operation of an adaptable communication substrate. *The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03)*, May 2003.

[27] C. Schill. SMCRoute. `http://www.cschill.de/smcroute/`.

[28] D. C. Schmidt, D. L. Levine, and S. Mungee. The design of the TAO real-time object request broker. *Computer Communications*, 21(4):294–324, April 1998.

[29] M. Weiser. Ubiquitous computing. *IEEE Computer*, 26(10):71–72, October 1993.

[30] M. Yarvis, P. L. Reiher, and G. J. Popek. Conductor: A framework for distributed adaptation. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems*, pages 44–49, Rio Rico, Arizona, March 1999.

[31] J. A. Zinky, D. E. Bakken, and R. E. Schantz. Architectural support for quality of service for CORBA objects. *Theory and Practice of Object Systems*, 3(1), 1997.