# Transparent End-Host-Based Service Composition through Network Virtualization[*]

Stefan Götz Klaus Wehrle

Junior Research Group
Protocol Engineering & Distributed Systems
University of Tübingen
Auf der Morgenstelle 10c
72076 Tübingen, Germany

{stefan.goetz|klaus.wehrle}@uni-tuebingen.de

## ABSTRACT

Mobile devices have become a popular medium for delivering multimedia services to end users. A large variety of solutions have been proposed to flexibly compose such services and to provide quality-of-service guarantees for the resulting contents. However, low-level mobility artifacts resulting from network transitions (disconnected operation, reconfiguration, etc.) still prevent a seamless user experience of these technologies. This paper presents an architecture for supporting legacy applications with such solutions in mobile scenarios. Through network virtualization, it hides mobility artifacts and ensures connectivity at the network and transport level. Its adoption for multimedia applications poses unique challenges and advantages, which are discussed herein.

## Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Protocols*

## General Terms

Design, Experimentation

## Keywords

Legacy Support, Mobility, Multimedia, Service Composition

## 1. INTRODUCTION

Mobile devices form an increasingly attractive platform for multimedia applications. Corporate environments in particular obviate such mobile applications. Users ubiquitously access multimedia data through a variety of devices in their offices, meeting rooms, cars, at their customers' site, or at home. In these settings, video, audio, and textual data need to be continuously adapted. For example, users may expect a text-to-speech conversion of their e-mails while driving or tools for rich-media collaboration in real time over wireless links. Many of the challenges of providing multimedia services to users in such scenarios have been addressed by recent research in the areas of adaptability to changing networking environments, quality of service, and service composition [10, 6, 4].

However, to take advantage of these services, changes to the applications themselves and the underlying operating systems become necessary. Consequently, it is difficult and expensive to evaluate novel protocols, frameworks, and middleware, and to deploy them on end systems, resulting in low rates of adoption. Furthermore in today's systems, mobile users experience artifacts of mobility which impair service availability and quality. Thus, moving between such diverse environments as wired and cellular networks burdens users with administrative tasks. Multimedia and streaming media applications in particular exhibit sub-optimal quality or experience complete loss of services under varying network conditions or network transitions.

To address these issues, we propose a network virtualization layer with three main responsibilities in mobile scenarios: (1) to relieve the user of system re-configuration due to a changing networking environment (e.g., the transition from a wireless LAN to a GSM-based connection); (2) to hide changes in the networking environment from legacy applications; (3) to perform a mapping between legacy traffic and richer communication paradigms ranging from solutions for enhanced multimedia services and service composition to overlay-based routing or IPv4/IPv6 tunneling.

Our architecture achieves these tasks by allowing flexible protocol transformations on the end-system and leveraging overlay routing systems, service composition frameworks, and higher-level protocols in general at the network level. On the end system, network traffic needs to be intercepted, analyzed, transformed if necessary, and forwarded without requiring changes to applications or the operating system. At the network level, we use overlay routing services such as i3 [12] which provide mobility support and generic support for service composition. By combining both aspects, multi-

[*]This work was funded under grant SE207/WE by Landestiftung Baden-Württemberg

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*MSC'05,* November 11, 2005, Singapore.
Copyright 2005 ACM 1-59593-245-3/05/0011 ...$5.00.

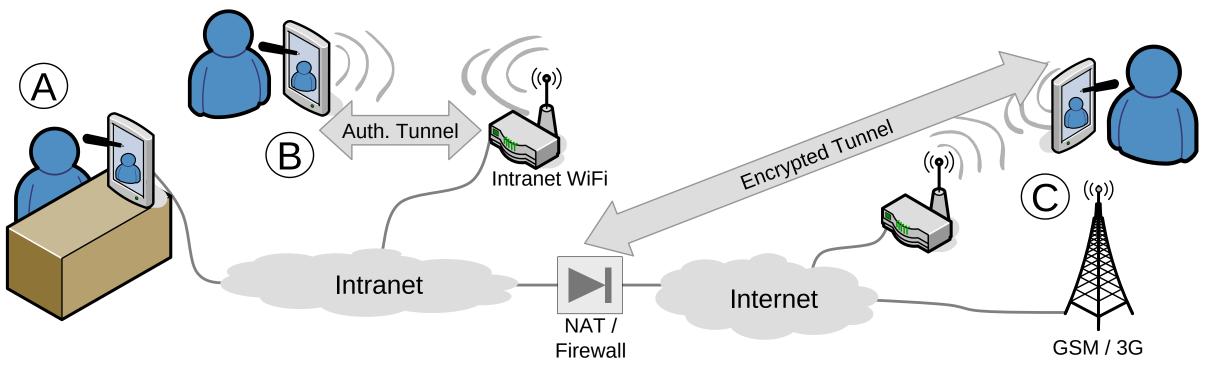

**Figure 1: Example scenario of mobile multimedia applications.**

media and other services can be delivered seamlessly to mobile legacy applications. This forms a flexible research platform for evaluating new multimedia protocols and frameworks with legacy applications in mobile environments.

While our architecture is generic and applies to a large variety of applications, this paper discusses opportunities and challenges of using it for mobile multimedia and service composition. It is organized as follows: section 2 introduces an application scenario which illustrates the necessity for seamless mobility support for multimedia applications; our architecture addressing this challenge is described in section 3; section 4 discusses architectural issues arising specifically in the context of mobile multimedia applications; section 5 analyzes related work and section 6 concludes.

## 2. MOBILE MULTIMEDIA SCENARIOS

Many mobile applications serve as prime scenarios for motivating multimedia service composition. Here, the user experience of multimedia services can particularly benefit from customization and adaptation of contents: data needs to be available in formats matching the capabilities of mobile devices. Furthermore, formats and their properties should adapt to the changing social, networking, and administrative environments in which the users move. In the remainder of this paper, we will use the following three scenarios to illustrate the challenges of mobile multimedia applications (cf. Figure 1).

In scenario A, a company employee uses a hand-held device in a docking station for a video conference with colleagues. When the device is undocked in scenario B, it should switch to the company wireless LAN, perform the necessary authentication through a VPN client, and adjust the quality of the video and audio streams to match the new connection properties. On their way home, the user may be engaged in a VoIP call with a friend and enter a cafe after leaving the company site (scenario C). Here, the device can choose between GPRS and a commercial Wi-Fi hotspot to maintain connectivity. Next to the VoIP or video conference application, the user then starts to download a file from a company server for which additional encryption is desired.

These scenarios call for modular solutions to implement multimedia services. The costs and effort of re-implementing components and frameworks for each service and system in a monolithic manner are prohibitive. Ideally, these service components can be orchestrated into service compounds, which are delivered to the end user. Our architecture facilitates and augments such solutions to make them available to legacy applications. It also allows to enrich legacy services to increase the functionality or enhance their quality.

The changes of network links and their properties due to user mobility manifest in two aspects. The first aspect is that connectivity needs to be maintained, not only by configuring network devices appropriately, but also by adapting to protocol requirements. From the scenario above, the transition from the wired to wireless link including the necessary VPN authentication illustrates this point. Therefore, the first goal of our architecture is to automate this tasks and require as little administrative interaction with the user as desired. The second aspect of mobility artifacts applies to applications. Most applications use TCP or UDP over IPv4 and can not handle mobility seamlessly. Thus, our second goal is to provide mobility support for legacy applications.

The third goal is to leverage new overlay networks and network architectures. In a multimedia context, this would allow novel protocols and frameworks (such as [6, 4]) to provide service composition and QoS guarantees to, e.g., a legacy video player. Furthermore, the reuse of legacy applications can significantly reduce the effort required to create realistic test and evaluation environments. Thus, our architecture can serve as a research platform to ease the development and evaluation of new protocols, platforms, and distributed systems.

## 3. ARCHITECTURE OVERVIEW

The overall goal of our end-system architecture is to introduce new protocols and additional functionality into the standard network stack without requiring changes to legacy applications or operating systems. We call the software implementing these features (and instances of it) a *proxy* as it is responsible for intercepting and relaying network traffic. In contrast to remotely deployed application-level proxies (e.g., remote caching HTTP proxies), our proxy resides on end systems and intercepts network traffic locally.

The network-related part of our architecture augments legacy protocols and applications with additional functionality. It exploits network-based services such as routing, transcoding, or encryption. In particular, the i3 overlay routing infrastructure provides support for host mobility [19] and service composition [13], and further protocols can be layered on top of it.

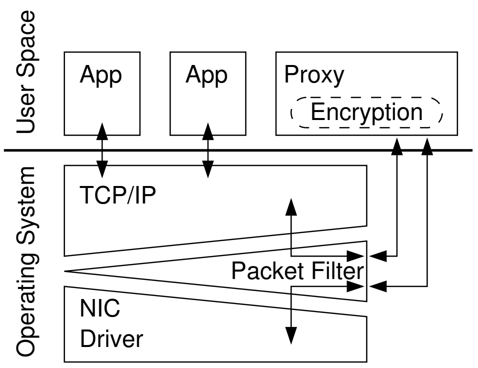

**Figure 2: Proxy architecture with packet filter and protocol transformers.**

## 3.1 End-System Architecture

As illustrated in Figure 2, the two main components of the proxy are a packet filter and a set of protocol transformers. While standard protocols like TCP/IP are implemented as part of the operating system kernel, the proxy is a regular user-level application. This fact substantially reduces the effort of developing and debugging protocol transformers and new protocols within the proxy. Depending on the underlying operating system, the packet filter cannot be integrated directly with the proxy but must instead be implemented as an in-kernel component. However, the packet filter and the protocol transformers connect through a generic interface which abstracts from these platform-dependent details.

### *3.1.1 Packet Interception*

The packet filter is responsible for intercepting packets going from applications to the network and vice versa. Here, interception means that packets leave their normal flow of processing in the operating system and are delivered to the proxy. Furthermore, it must be possible to inject arbitrary packets into this flow. In scenario B, the packets of the video-conferencing application need to be transmitted through the VPN tunnel. For outgoing packets, the packet filter intercepts them before they leave the system. It passes them to the proxy which encapsulates them with the VPN protocol and returns them to the packet filter. The filter then injects them into the network stack of the operating system from where they are sent to the wireless network. Incoming packets are handled symmetrically to decapsulate them and pass them to the conferencing application.

On many Unix systems, tun/tap devices [16] allow us to implement this form of packet interception as part of the proxy implementation and no in-kernel code is necessary. On Microsoft Windows systems, the proxy uses a Windows driver providing similar functionality to tun/tap devices. The driver also allows for conditional packet interception in order to deliver only relevant packets to the proxy application. The implementation as a driver requires no changes in the operating system.

### *3.1.2 Application Transparency*

The support for unmodified legacy applications is a central aspect of our end-system architecture. It is achieved by leaving the application programming interface and application binary interface between application and operating system intact. Instead, the proxy only interacts with applications by intercepting and relaying their network traffic. Thus, application transparency needs to be ensured at the protocol level.

| | |
|---|---|
| Application | Adaptation and transformation of data and protocols, integration of new protocols, service discovery & composition |
| Transport | Adaptation to link properties, link maintenance, QoS management |
| Network | Mobility support, (overlay) routing, encryption, authentication |
| Adapter | Link detection & selection |

**Table 1: Protocol transformations can be grouped by network layer.**

Due to the almost exclusive use of the IPv4 protocol in legacy applications and the benefits of i3 for mobility and service composition, the mechanisms for protocol transparency will be briefly illustrated on the example of this combination of protocols. While tunneling IP traffic over i3 itself is straightforward, service discovery is not because i3 communication endpoints are not identified by pairs of IP addresses and port numbers. Thus, the proxy associates i3 endpoints with unused *virtual* IP addresses (e.g., from a private address range such as 10.0.0.0/8). All traffic from an i3 endpoint is modified to appear to originate from a host with the associated virtual IP address. Conversely, packets with virtual destination IP addresses are encapsulated and tunneled to the associated i3 endpoint.

Virtual IP addresses are provided to applications by intercepting name resolution attempts such as DNS queries. These legacy mechanisms can thus be augmented with other schemes for name resolution and service discovery. They can range from hashing the DNS name locally into an i3 endpoint identifier to, e.g., complex QoS-aware negotiation protocols for locating services and composing communication paths.

### *3.1.3 Protocol Transformation*

Our end-system architecture is structured such that protocol transformations can be stacked on each other. After a packet is intercepted by the packet filter, it is fed to the transformation stack. The transformation modules interact through a generic interface for passing the modified packet on to the next module. Eventually, packets are either dropped or injected back into the regular network stack of the host operating system. This lends itself to the fact that different transformations apply to different protocol layers, as shown in Table 1. Service-composition and multimedia frameworks and protocols would typically be implemented at the application and transport layers.

As illustrated in Figure 3, protocol transformations can also be applied selectively. For example in scenario B, the internal company file server can be accessed directly over the wireless LAN. The video conferencing traffic going to the Internet is handled by the VPN module for connectivity, authentication, and encryption. The conferencing application in turn can be enhanced with higher-level transformations, e.g., QoS management or multicast and mobility support.

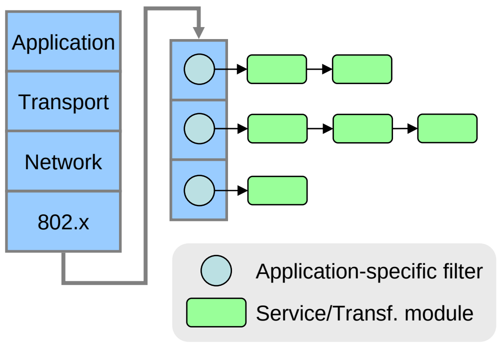

**Figure 3: Stack of transformation/adaption modules with their application-specific filters.**

#### 3.1.4 *Adaptability*

Based on external events or user intervention, modules for protocol transformation can be inserted to and removed from the transformation stack at run time. In concert with automatic configuration of network devices, a host becomes significantly more adaptable to a changing network environment. This applies equally to host mobility, such as vertical switch-overs (e.g., WLAN to GPRS), infrastructural requirements (e.g., VPNs, pay-per-use access), and fluctuations in link quality. Thus, the need for administrative action from the user in mobile environments can be reduced or eliminated.

Since automatic adaptation can exhibit side-effects unwanted by the user, we introduce a policy-based approach. A policy, as defined by the user, controls and restrict the actions the proxy may take. As outlined in scenario C in the cafe, the system may have a choice between an expensive but high-bandwidth Wi-Fi connection and the cheaper GPRS link with lower throughput. In such a situation, the proxy itself can detect a bandwidth-intensive application such as the download and thus provide the user with the faster connection. However, if the download is not of importance to the user, he can activate a low-cost policy forcing the proxy to choose the less expensive connection.

### 3.2 Network Architecture

The network-related part of our architecture utilizes the overlay routing services of the Internet Indirection Infrastructural i3 [12]. Its core idea is to communicate across one or more points of indirection which stands in contrast to end-to-end communication. This scheme decouples the act of sending from the act of receiving and can thus provide additional features like multicast, anycast, mobility support, or service composition.

Every point of indirection is identified by a unique ID in the form of a large integer or fixed-length bit string, respectively. Data packets carry an ID instead of a real IP address as the destination address. Thus with i3, data is addressed to an abstract notion of a service instead of a particular end host. In order to receive data via i3, hosts register so-called *triggers* with the i3 system. A trigger is an association of a destination ID with an IP address/port pair or another ID. i3 forwards all packets going to an ID to the trigger addresses registered with this ID. In a simple example, a receiver inserts a trigger associating an ID with the IP address and a port it listens. Accordingly, i3 delivers all data sent to the ID to the receiver.

Mobility support in i3 is based on the addressing scheme of using IDs instead of IP addresses. When a mobile host moves between networks and receives different IP addresses, it updates its i3 triggers accordingly. Consequently, the host remains accessible at the i3 level. i3 allows receivers to insert more than one trigger per ID, so the ID itself remains unique but is associated with multiple forwarding addresses. The packets which are sent to such an ID are forwarded to every associated trigger address, which effectively implements multicast communication. For service composition, i3 generalizes the concept of IDs to ID stacks. A packet with a destination ID stack must traverse all the triggers referenced in the stack, which can be regarded as source routing. Similarly, forwarding entries in triggers can also be ID stacks so a forwarded packet must go through all the IDs in the stack. Thus, both senders and receivers can control the route the packet takes including services and transformations the packet needs to traverse.

NAT gateways and firewalls do not limit the reachability of i3 clients, as long as outbound connections are permitted. In scenario C, IP connections from the Internet to the hand-held device can be blocked by the Wi-Fi firewall and NAT configuration. However, the device can still establish a connection to the Internet-based i3 service and the device's triggers are associated with this connection instead of its (unreachable private) IP address. Thus, i3 packets can reach the device despite NAT and a firewall.

In the proxy, i3 is implemented as a transformation module and is thus an optional component. However, its flexibility and functionality at the routing layer makes it an ideal addition to our architecture. Furthermore, higher-level protocols can exploit its features and its generic support for service composition.

## 4. DISCUSSION

This section analyzes the challenges of using our proxy architecture in a multimedia context.

### 4.1 Inferring Application Requirements

QoS and service-composition frameworks often rely on applications to explicitly indicate their requirements and capabilities. In many cases, feedback cycles between layers allow to determine the best compromise between user demands and application and network properties. For example, the video conferencing application can request a maximum acceptable latency and a minimum video frame rate from the service layer. This layer may then choose an appropriate encoding and decide whether additional services, e.g., subtitles for a video can meet these requirements.

By design, our proxy focuses on legacy applications and avoids direct interaction with applications. Thus, the application layer does not explicitly provide service specifications or requirements. Instead, this information must be inferred by the proxy itself. In many cases, it is sufficient to derive this data implicitly from application and system behavior. For example, the necessity to transcode between media formats can be deduced from the service being requested (a specific video), the requested format (e.g., AVI), and the actual format of the service (e.g., MPEG). The need for service composition can also arise from a changing operating environment. For example, the transition from a company network to a public network, as in scenario C, may trigger the activation of an encryption service.

The user's requirements for individual services can also be indicated explicitly to the proxy. First, the proxy may provide configuration dialogs for specific services or service classes. For example, the proxy may export a setting which controls whether text sub-titles for video streams are displayed or not, even if the legacy video player application is unaware of such a choice. While such external configuration may hamper usability to a certain degree, it may be acceptable for evaluation purposes or where there is no alternative to using a certain legacy application. Second, legacy name resolution can be exploited for service specification. Instead of passing regular URLs to legacy applications, URLs formatted to contain service composition paths and requirements can be used. While the application remains agnostic to this format, a service composition framework in the proxy can utilize the encoded information. However, such a possibly complex URL format is cumbersome to handle.

Multimedia applications depend on several properties of the whole system, such as available resources and network link characteristics. The proxy can centrally aggregate such properties and supply them to protocols implemented within the proxy. Where resource contention is an issue, resource allocation schemes can also be implemented centrally.

Quality-of-Service constraints can be inferred to a certain extent through observation of system behavior. Based on these observations, QoS parameters can be adjusted to provide higher quality to the user or to better utilize available resources. For example, the transition from the wired company network to the wireless LAN could result in low CPU utilization and high network utilization. This information indicates that the streaming video attempts to consume more network bandwidth than available. Switching to a computationally more complex compression scheme can result in a higher effective frame rate, i.e., better quality provided to the user.

### 4.2 Flow Identification

Since individual applications and their network connections have different requirements, the proxy must be able to differentiate between them. For example, the company wireless LAN may allow unrestricted access to internal services while the Internet is only accessible after authenticating with a VPN. The proxy can support such an environment with selective protocol transformation by running only remote traffic through the VPN. Similarly, the user may place different demands on different multimedia streams based on customized policies (e.g., giving the video conference application a higher priority than another background video stream). Consequently, the proxy must identify these streams and handle them individually in order to meet user demands.

The more accurate flow identification needs to be, the more knowledge about protocols and analysis of traffic is necessary. In simple cases, such as the VPN example, traffic flows can be distinguished based on transport-level information, i.e., IP address and ports. Closer analysis is required for multi-flow protocols like SCTP. It is to be noted that such a detailed packet inspection need not be implemented in the proxy in general but only in the respective transformation modules.

### 4.3 Performance

The structure of our proxy imposes a processing overhead for network traffic on end hosts. This overhead is due to intercepting, parsing, and processing packets in the proxy. At the current stage of implementation, no experimental results are available for a performance evaluation. Thus, a quantitative analysis follows.

Each intercepted packet is transferred from the operating system's network stack to the proxy for further processing. The proxy analyzes the packet to determine whether it is to be forwarded unmodified or passed to the transformation stack. Eventually, the proxy injects the packet back into the regular network stack. Thus, packet interception causes two context switches and two additional copies of each packet. Since data rates are low for mobile devices with wireless links, this overhead is assumed to be negligible.

Analyzing packets and forwarding them between transformation modules can be assumed to cause only a very modest performance impact. These operations are comparable in cost to those performed in the operating system's network stack. The encapsulation of a packet increases its size on the wire. For large packets, this can lead to additional packet fragmentation. Packet processing in transformation modules is potentially expensive but may not be regarded as overhead introduced by the proxy architecture itself.

## 5. RELATED WORK

Implementing and evaluating network protocols at user level has been an issue in operating system and network research for a long time [15, 8, 11, 9]. Where these solutions strive to replace kernel-level protocol stacks, they trade API compatibility for performance or security. In contrast, the support for legacy applications is a primary concern of our approach.

Other than evaluation approaches like Alpine [1], our end-system architecture does not attempt to replicate real execution environments for protocol implementations. Thus, it can be used on several platforms and remains more lightweight. Application-transparent architectures like CANS [3] or Conductor [18] share goals with our approach in hiding the mobility artifacts and supporting legacy applications. However, they are tied to their network architectures and intercept network traffic at the interface between application and operating system. While this results in fewer context switches and better performance, these solutions are heavily system dependent and require significantly more engineering effort.

Commercial applications like the ipUnplugged Roaming Client [5] achieve seamless connectivity with similar techniques for packet interception and redirection as ours. However, they solely focus on VPN and IPsec solutions and cannot serve as a generic research platform.

Delay-tolerant networking (DTN) [2] addresses the effects of mobility stemming from network fragmentation or disconnected operation. We assume our application scenarios to be typically faced with widely varying degrees of link qualities and properties rather than with longer periods of no connectivity. Thus, we view the work on DTN as being orthogonal to ours which could be very well integrated with the proxy.

Our architecture borrows substantially from the i3 proxy [7], including IP address virtualization [14] and DNS rewriting [17]. While the i3 proxy aims at redirecting legacy traffic via i3, our solution provides a framework for arbitrary network modifications, essentially a user-level network stack.

## 6. SUMMARY

While multimedia services and the composition of such services have been a long standing research topic, it remains difficult to evaluate and deploy new protocols, frameworks, and middleware systems in this area. We propose a research platform with an end-host-based architecture for network virtualization. It allows network traffic to be transformed at the user level while maintaining transparency towards legacy applications. This system significantly simplifies protocol deployment and evaluation, the adaptation to changing network environments, and extensions to legacy services. In mobile settings, it can hide mobility artifacts from users as well as legacy applications and support QoS and service composition.

## 7. REFERENCES

[1] D. Ely, S. Savage, and D. Wetherall. Alpine: A User-Level Infrastructure for Network Protocol Development. In *USITS*, pages 171–183, 2001.

[2] K. Fall. A Delay-Tolerant Network Architecture for Challenged Internets. Technical Report IRB-TR-03-003, Intel Research Laboratory at Berkeley, Jan. 2003.

[3] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS: Composable, Adaptive Network Services Infrastructure. In *USITS*, pages 135–146, 2001.

[4] X. Gu and K. Nahrstedt. Distributed Multimedia Service Composition with Statistical QoS Assurances. *IEEE Transactions on Multimedia*, May 2005.

[5] ipUnplugged AB, Homepage. http://www.ipunplugged.com/products.asp?mi=2.3.

[6] M. Kosuga, N. Kirimoto, T. Yamazaki, T. Nakanishi, M. Masuzaki, and K. Hasuike. A Multimedia Service Composition Scheme for Ubiquitous Networks. *Journal of Network and Computer Applications*, 25(4):279–293, 2002.

[7] K. Lakshminarayanan, I. Stoica, K. Wehrle, et al. Supporting Legacy Applications over i3. Technical Report UCB/CSD-04-134, UC Berkeley, May 2004.

[8] C. Maeda and B. N. Bershad. Protocol Service Decomposition for High-Performance Networking. In *Symposium on Operating Systems Principles*, pages 244–255, 1993.

[9] A. Mallet, J. Chung, and J. Smith. Operating Systems Support for Protocol Boosters. In *HIPPARCH Workshop*, June 1997.

[10] A. Misra, S. Das, A. McAuley, and S. K. Das. Autoconfiguration, Registration, and Mobility Management for Pervasive Computing. *IEEE Personal Communications*, 8(4):24–31, Aug. 2001.

[11] S. H. Rodrigues, T. E. Anderson, and D. E. Culler. High-Performance Local-Area Communication with Fast Sockets. In *Usenix Annual Technical Conference*, pages 257–274, 1997.

[12] I. Stoica, D. Adkins, S. Zhaung, et al. Internet Indirection Infrastructure. In *Proceedings of ACM SIGCOMM'02*, pages 73–86, Aug. 2002. Pittsburgh, PA.

[13] I. Stoica, K. Lakshminarayanan, and K. Wehrle. Support for Service Composition in i3. In *Proceedings of ACM Multimedia*, Oct. 2004. New York.

[14] G. Su and J. Nieh. Mobile Communication with Virtual Network Address Translation. Technical Report CUCS-003-02, Department of Computer Science, Columbia University, 2002.

[15] C. A. Thekkath, T. D. Nguyen, E. Moy, and E. D. Lazowska. Implementing Network Protocols at User Level. *IEEE/ACM Transactions on Networking*, 1(5):554–565, 1993.

[16] Universal TUN TAP Driver, Project Homepage. http://vtun.sourceforge.net/tun/.

[17] P. Yalagandula, A. Garg, M. Dahlin, L. Alvisi, and H. Vin. Transparent Mobility with Minimal Infrastructure. Technical Report TR-01-30, Department of Computer Sciences, University of Texas at Austin, 2001.

[18] M. Yarvis, P. L. Reiher, and G. J. Popek. Conductor: A Framework for Distributed Adaptation. In *Workshop on Hot Topics in Operating Systems*, pages 44–51, 1999.

[19] S. Zhuang, K. Lai, I. Stoica, et al. Host Mobility Using an Internet Indirection Infrastructure. In *Proceedings of ACM MobiSys*, 2003.