A Network Stack Compliant to the Needs of New Environments

Cheryl Pope, Henry Detmold, Dave Munro and Francis Vaughan
School of Computer Science
University of Adelaide, Australia 5005
ph: +61 8 8303-5833, fax: +61 8 8303-4366
Email: cheryl, henry, dave, francis@cs.adelaide.edu.au
keywords: Internet Protocols, Network Architectures, Mobile Computing, Compliant Systems

***Abstract*— Existing TCP/IP protocol stacks contain many assumptions about the underlying network that are embedded as control policies within the stack. This fact creates problems when the TCP/IP suite is used in non-traditional network environments, such as mobile networks. Attempts to adapt existing TCP/IP stacks to suit mobile networking are typically partial solutions limited to addressing one specific issue. In this paper, we argue that a holistic approach to supporting the needs of new networking environments is only feasible given a rigorous separation of policy from mechanism, a separation that current stack implementations lack. We introduce a modified stack model, the "Embedded Inverted Stack", which is an instantiation of the generic Compliant Systems Architecture [1]. This provides a general framework within which a stack implementation can comply with the particular needs of different networking environments whilst maintaining complete compatibility with existing TCP/IP applications. We focus particularly on how such an architecture can support mobile networking.**

I. Introduction

Joe is having lunch in a cafe, managing his email through a handheld device connected to a mobile network via an 11 Mb/s 802.11 wireless network. He finishes his coffee and goes outside for a post lunch stroll. His handheld and applications seamlessly handover to a 32 Kb/s CDMA based network. During his stroll, Joe moves into range of an alternative network offering higher bandwidth (lower load) but at a greater cost. An icon activates on the device, showing increased performance is available and the current cost. At this point, Joe is downloading streamed information about a piece of art he is passing, he decides he would prefer faster playback and turns a wheel input on his handheld so as to increase performance. The cost and playback performance increase until he is satisfied with the cost/performance trade-off, or the limit of bandwidth available is reached. Meanwhile, the final game of the World Cup has just ended and there is a rapid increase in communications on the mobile network. Bandwidth is now becoming scarce and possibly expensive. The built in policy of his Web browser application automatically reduces the streamed video playback to low quality black and white JPEG and voice playback or alternatively, if so instructed, pops up a dialogue asking Joe to decide what to do.

The above scenario is indicative of the adaptability we desire of mobile devices, however the reality of mobility is still far from this scenario. The networking research community has in recent years devoted a great deal of effort to enhancing existing TCP/IP stacks' support for mobile network devices [2]. These efforts have been hampered by the fact that TCP/IP stacks have been designed around assumptions which are generally accurate in fixed networks, but which are largely invalid in wireless networks. These assumptions include low error rates, relatively high bandwidth, symmetric connectivity between end points and stable configuration. In order to provide a TCP/IP stack appropriate to mobile devices, the parts of the stack based on these assumptions must be replaced or their effects overcome. The ability to do this with existing stacks is frustrated by the nature of typical implementations, in which these assumptions are encoded as policies embedded into the various stack layer protocols. This has led to systems that attempt to overcome one specific issue, most commonly the issue of congestion control [3] [4] [5]. Although these systems do provide ad hoc solutions to real problems, they do not address the underlying cause of those problems. Namely, that the services of the existing stack were not designed for a general network. Instead they were designed specifically for the Internet, as it existed at the time, and adapted as it changed. Since the physical properties of the underlying network stayed basically the same (with improvements in performance) the stacks continued to function. But the characteristics of the networks that make up the Internet could have been very different and the protocols have developed into their current state due to the historical accident that has led us to todays Internet. Mobile networks have radically changed the network characteristics and we now are exposed to the problems inherent in a design tied to specific (and different) network assumptions.

We believe that the historical definition of TCP/IP stack layers in terms of the services they provide is insufficiently flexible. Our contention is that in order to support the networking needs of new technologies, such as mobile devices, network stacks need an architecture that separates policies from the mechanisms of policy implementation and which supports, rather than inhibits, the ability for the policy decisions affecting a mechanism to be made at layers other than the layer at which that mechanism is implemented. One such software architecture is the Compliant Systems Architecture of Morrison and Warboys [1] [6].

The Compliant Systems Architecture (CSA) is a model for designing systems that can accommodate the needs of applications. Conventional architectures design for the predicted needs of the expected applications. In contrast, a compliant architecture is able to accommodate the needs of new applications which may not match those predicted. A CSA accomplishes this by separating policy and mechanism in the design and allowing policy decisions to be made by the components that are best able to make informed decisions. Changes to state which may require policy decisions is communicated through the use of upcalls [7] as well as traditional downcalls. In the CSA *all* components are designed in terms of mechanisms and policies. Practical constraints can then be considered to limit the policies and mechanisms of the components. By considering what policy decisions should be provided to higher layers and implementing a network architecture that makes it possible for higher layers to provide these policies, we are able to provide clean solutions to current problems in supporting mobile devices in the TCP/IP stack and to provide a flexible framework that can accommodate changes to the network characteristics and the requirements of future applications whose needs we do not yet know.

Creating a TCP/IP stack that is an instantiation of the CSA requires a change in the way we think about the layers. In this paper we show how the CSA can be applied to the design of a TCP/IP based network stack. We examine the tasks of the existing network layer model and present the mechanisms, state and policy upcalls and downcalls of an "Embedded Inverted Stack" (EIS) instantiation of the CSA. Finally, we present a design for the scenario described at the beginning of this paper to illustrate the advantage of implementing mobile application adaptivity within the EIS.

II. Application of the CSA to a Network Stack Architecture

Although the method of design is well specified, actual instantiation of a CSA is non-trivial. There are many degrees of freedom in deciding which policy decisions should be available to each of the component. The choice of these policy decisions and the components in which these policies reside must be well argued. The CSA specifies rules for policy placement to avoid meaningless policies such as placing policy in components that have no knowledge of the related mechanism. The existing layered architecture has served well and we retain the layered architecture for placing the network stack mechanisms. To create a network stack instantiation of the CSA we proceed as follows: for each protocol layer we determine what mechanism(s) the layer provides, what the default policy for the mechanism(s) will be, what policy(s) the layer should allow other layers to provide, and what communications are required to allow necessary state information to be exchanged between the layer implementing mechanism and the layer(s) implementing policy for that mechanism.

The means for communication between components is well defined within the CSA, so we begin with mapping this communication into the network stack. We return to the task of defining per layer policies and mechanisms in section III.

A. Communication between Policy and Mechanism

The network layer model is traditionally understood solely in terms of downcalls from higher layers to lower layers. For example, the application layer calling the transport layer when the application has data to send. It is well known, however, that the use of upcalls can significantly improve the performance of protocol stacks [7] [8]. In a Compliant Systems Architecture, policy and mechanism interact through the use of upcalls and downcalls as shown in Figure 1. The upcalls are a part of the CSA model they are not merely a means of optimisation. The upcalls and downcalls may propagate through one or more layers. This design has the effect of creating one or more inverted stack(s) embedded within

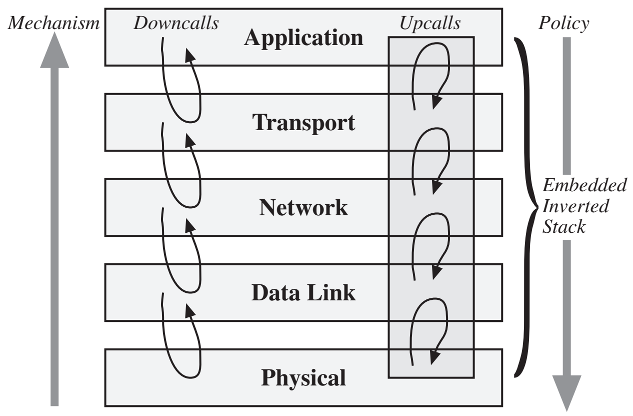

Fig. 1. Inverted Stack(s)

the main stack. At each layer of the network, a set of mechanisms is available to accomplish a task. For example, the physical layer has a mechanism for trading off power and bandwidth (through better signal). Most implementations try to make the "best" trade-off achieving adequate performance considering remaining battery life. However, the user or application may have specific preferences. Perhaps the user is just about to retire to their hotel and a power outlet or alternatively may prefer to preserve power for an important application they will be using later. Alternatively the user may not care and may be willing for the existing policy to make whatever trade-off it determines is best. The main issue is that, policy decisions should be made available to the highest layer possible.

The composition of upcalls and downcalls and the transparency of the layers (i.e. whether calls proceed through *all* higher layers or communicate directly with non-adjacent layers), is intentionally left unspecified. Our aim with this paper is to introduce the argument for separation of policy and mechanism and examine what these policies and mechanisms are in TCP/IP stacks. There is extensive published experience with the implementation of network stacks both in kernel and user space. A network stack modeled on the CSA represents a major departure from traditional models. The existing experience on implementation will need to be evaluated and reconsidered in terms of the new stack model.

III. Instantiation of the Embedded Inverted Stack Model for Mobile Network Devices

Having identified benefits arising from separation of policy from mechanism with sharing of state information, and described the mapping of the protocol layers on to the CSA, we now apply the CSA methodology as a conceptual framework, examining the individual network layers to:

- identify the mechanisms (down calls) they provide,
- identify points (up-call sites) at which policies should be sought from higher layers,
- identify default policies to operate in the absence of higher layer policy provision,
- and identify the state information necessary for supporting each policy framework.

One important aspect of the CSA is the specification of default policy. We preserve the existing TCP/IP stack behaviour as this default policy in all cases where TCP/IP currently implements policy. This allows our stack to retain compatibility with existing TCP/IP applications and devices. In the discussion that follows we only identify additional policies and present arguments for what layers should or should not be allowed to dictate these policies.

Although we believe that redesigning the entire TCP/IP stack as an instantiation of the CSA would lead to a much more flexible system, our focus for this example is on mobile devices. We therefore identify policy/mechanism separations that we expect to benefit mobile devices. Therefore this list should not be considered exhaustive of what could be done, but rather a particular instantiation to support mobile devices. In the following sections, we discuss each layer individually and summarise in Table I at the end of the discussion.

Physical Layer

The task of the physical layer is the transmission of the data in the form of a signal across the medium (air for wireless). The mechanism under the control of the physical layer is the power used for the transmission which affects the data rate that can be achieved and the cells that can be reached. This is not a simple dependency, i.e. higher power does not necessarily increase bandwidth. However, higher power use is often needed in poor conditions to achieve bandwidth equivalent to that available in good conditions. Power may be a limited resource, especially on handheld devices. Therefore it is likely to be of benefit to allow the policy determining the optimal data rate/power trade-off to be made at higher layers. For example a user may want some data to go through at a higher rate even if it severely reduces the remaining battery life, perhaps to reach a more distant cell to complete an important voice call in the case of congestion in closer cells. Alternatively a user may decide to prioritise his applications in response to low power or bandwidth. To perform policy decisions, higher layers must have access to the current power consumption rate and the current data rate. Upcalls are needed to notify higher layers of changes in the state of bandwidth or power. Downcalls for policy to increase or decrease bandwidth or power consumption should be available to the application layer to allow the user to make power/bandwidth trade-offs.

Data Link Layer

The data link layer synchronises access and provides reliable transmission of frames (multiple bits) across the physical link. Data compression, error correction, encryption, access control to shared medium, and handover control are typically performed by implementations of the data link layer. The data link layer handles data as frames. Since other network layers have no knowledge of frames, only policies that deal with general (not frame-specific) data-link mechanisms should be made available for decision at higher layers.

Data compression can be performed at either the data link and the application layer. However, the application can often perform compression more efficiently since it has greater knowledge about the characteristics of the data to be compressed. Link layer compression should be off for applications that perform their own compression and on otherwise. This does not vary during the lifetime of an application, therefore this policy is specified by a single downcall when the application begins. No upcalls are necessary to signal changes in the data link layer.

Error correction policy is an example of a policy that is correctly placed in the same layer as its mechanism. There is little benefit to turning error correction off. The possible power savings due to reduced computation and retransmission would be very small. Higher layers do not work with frame level information so they do not have the knowledge to meaningfully specify a frame level error correction policy. Medium access control (MAC) also relies on frame level information and MAC policy correctly resides at the data link layer.

Encryption is typically performed by the data link layer, but relies on information available only at the application layer. The keys, algorithm, security associations and whether authentication, encryption or both should be performed are policies that should be decided by the application layer. It is beneficial to support encryption on the link in cases where higher level encryption is not performed. Where different levels of encryption are possible, encryption type information should be available to higher layers. There are no event driven upcalls for encryption, but downcalls need to exist for turning encryption on or off and for determining which encryption algorithm is used.

Handover policy should not be limited to strongest signal. The network layer's packet routing and security policies should be able to define policy on whether handover occurs and which base station(s) or network(s) the device will communicate with [9]. The user may also want to determine handover policy, for example to decide his preferred cost/performance tradeoff.

Network Layer

The network layer's primary function is routing. In most instances applications are not concerned about how their packets are routed, only that they arrive. However, there may be cases where the application or user is concerned about the routing of their packets and may wish to explicitly determine the path of their packets or explicitly include or exclude certain hosts from the path. Support currently exists for specification of hosts that must be included in the path. There is some support in the existing TCP/IP stack specification for excluding hosts: by specifying a strict path other hosts are implicitly excluded.

The choice of routes may also affect cost. The application or user may wish to make trade-offs between the most efficient route (in terms of latency, throughput, etc) and the cost of traversing a particular route. The data link layer information regarding costs may need to be modified to represent overall path cost for outgoing links. This cost information should be available to the transport and application layers to support routing policy decisions.

IPv4 provides primarily routing. IPv6, the next version of IP also provides security services. In IPv6 encryption and authentication are performed by the network layer. In the discussions of the integration of security in the IPv6 network layer there was significant debate about whether this was the proper layer for implementation [10]. Although there are some clear advantages to network layer security, the application and user may also have security preferences which should be considered. We avoid entering into the IPv6 security layering debate, and hence suggest only that any encryption and authentication mechanisms as are provided in the network layer should be augmented with policy input from higher layers. This input includes the ability to specify whether or not the network layer should perform encryption and authentication and also up-calls to allow higher layers to make policy decisions in the event of security violations detected by the network mechanisms.

Transport Layer

The transport layer provides three services: reliable end-to-end data transfer, flow control between sender and

TABLE I

POLICIES & MECHANISMS IN THE EMBEDDED INVERTED STACK

Layer	*Mechanisms*	*State*	*Policies*
Application (A)	wheels, buttons	location modified cost performance (by example)	cost/performance (user)
Transport (T)	service level mechanisms flow control end-to-end reliability		service level (A)
Network (N)	route cost models encryption routing	modified cost routing info congestion	routing (A - source based)
Data Link (D)	handover compression error correction MAC encryption	cost encryption level data rate congestion	handover (N,A) encryption type (N,A) compression (on/off) (A) congestion (N)
Physical (P)	power	data rate power consumption	data rate/power trade-off (A)

receiver, and congestion control for the network.

The policy of data transfer reliability is already determined by the application through the choice of UDP (unreliable) or TCP (reliable) transfer. In addition to the level of reliability of packet delivery, we can consider service dimensions such as protection level and quality of service. Protection level defines the importance of a particular communication. For example, an emergency channel that identifies when the owner of the mobile device needs emergency help (police, ambulance, etc) should be protected and maintain transmission at any cost. Quality of Service defines performance factors such as preferred data rate, provisioning for burstiness, etc. All of these services affect the priority of different data units for transport across the network. It is important to note that quality of service may indeed conflict with protection level. An important file transfer may have higher priority to the user than a video conference, even though the latter requires a higher quality of service in terms of achieving a data rate that makes the application useful. Users would clearly benefit from the ability to influence the policy of scheduling of data transfers.

The existing congestion control mechanism based on ACKs can easily be circumvented by a misbehaving TCP implementation. For example, such an implementation might send out one or more ACK's in advance, in order to reduce the chance of backoff and maintain or increase the sending rate [11]. The rate of receipt of ACK packets is not definitive evidence of congestion or lack of congestion. The data link layer is the only layer having actual information about the congestion characteristics of a link. This information should be made available to the network layer, to provide a sound basis for determination of the congestion characteristics of a path through the network.

We do not propose pushing *packet* congestion policy up to higher layers, On the contrary, there is strong evidence that management of congestion at the packet level needs to be pushed down to the network layer [12] [13]. However, the fact that congestion is occurring should be accessible to the application layer as the application or user may wish to make congestion policy decisions regarding which applications continue to communicate or what sort of data is sent to best make use of the hosts reduced bandwidth share. Upcalls to the application layer on the change of available bandwidth will allow the application to adapt to these changes: perhaps increasing or decreasing the quality of an image transfer, or adapting communication by some applications to use periods of low congestion.

Application Layer

Along with standard applications, we model user behaviour as another application at the application layer. The user-behaviour application interacts with one or more other applications. Applications get policy input from the user-application via input devices such as wheels and buttons and provide state information via screens and other feedback mechanisms.

The cost of communication may also be modified by the application layer. Negotiation with remote servers may indicate a differential cost between multiple servers which can be considered along with the network com-

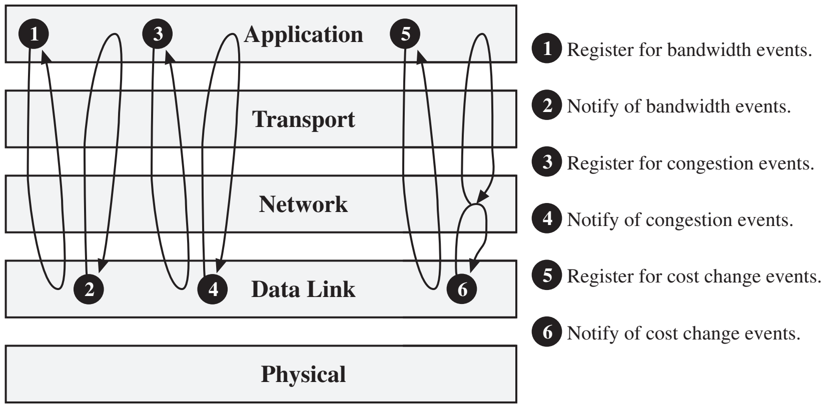

Fig. 2. EIS Support for Controlled Degradation/Recovery

munication costs to each of the servers in determining an overall cost estimate for the communication.

The mechanisms provided by each layer, the policies which are made accessible to higher layers and the state information that needs to be shared to support higher layer policy decisions are summarised in Table I. We identify the layers that are likely to want to specify the policy in parenthesis after the policy. In some cases, the existing TCP/IP implementation supports higher layer policy decisions as is the case with the use of IP source routing which allows the application layer to dictate routing policy (where packets are routed) to the network layer, which provides the routing mechanism.

IV. Controlled Degradation and Recovery in the Embedded Inverted Stack Model

We now return to Joe. The scenario he experiences relies on a level of adaptability not available within the existing network stack architecture. Applications are unable to make adaptation decisions promptly as they are not informed of changes in network state. Instead they rely on direct user estimation of network state or "divining" the state by observing behaviour over longer periods. Multimedia applications, for example, typically rely on streaming. Data is buffered before playback based on a data rate specified by the user, not the network. If the user specified rate differs significantly from the actual data rate, then the performance of the application degrades. An implementation of the described scenario in existing network stacks would likely result in policy changes occurring too slowly in response to network state changes.

In contrast, if Joe's handheld used the embedded inverted stack model, it is far more feasible to provide him with the controlled degradation/recovery he desires. In particular, our model supports quick reaction to rapid and frequent changes in the network. To support Joe's scenario we only need the policies and calls shown in Figure 2.

The link layer provides a mechanism for determining the current bandwidth available on the link. It provides a default policy of using whatever bandwidth is available as needed. The layer also provides the mechanism for detecting changes in available bandwidth. The default policy is the existing link layer policy and to make upcalls only on changes of $\infty\%$ (i.e. make no upcalls). To support user and application policies on bandwidth changes, a downcall is provided for registering interest in bandwidth changes (#1). This registration includes the % change that should trigger an upcall (#2) to the application layer.

The detection of congestion is a related, but separate event from bandwidth changes. The application layer registers for notification of congestion (#3). At the application layer, the user policy (which application should continue) and/or application policy (which parts of the application continue communication) is triggered by the upcall (#4). If there is no registration for the upcall by the application layer, then no upcalls will occur.

Finally, changes in communication cost estimates are also passed to the application layer from the data link via the network layer (which is likely to refine the cost estimates in the light of network layer information) (#6). The user's policy on cost will affect the mechanism for choosing link and network route (#5).

V. Conclusions

The use of upcalls to improve performance of TCP/IP stacks is widely accepted as an optimised implementation technique. However, little work has been published on the use of upcalls to support policy decisions. QoS policies involving the scheduling of packet transmissions have been implemented using upcalls [14], but the focus is on one policy and the mechanism and policy are both implemented within the same layer (transport). We believe that the ability of network stacks to support new technology such as mobile devices can systematically be enhanced by application of the Compliant Systems Architecture discipline. That is, that the existing stack should be examined at *all layers* to identify all areas where policy decisions may usefully be separated from the supporting mechanisms. In this paper we have re-designed the network stack as an instantiation of the CSA and identified the policies and mechanisms needed to support a high level of adaptivity for mobile network devices and applications.

The composition of upcalls and downcalls and the transparency of layers have not been specified in detail in this paper. We recognise that these details are critical to developing a widely accepted layered architecture that defines policies as well as services and allows for different implementations to be interchangeable. We are currently working on an implementation of EIS to support mobile devices. Experience from this work will allow us to refine our set of per layer mechanisms and policies and propose a set of upcalls and downcalls to be used. We hope that the ideas expressed in this paper will also stimulate discussion and debate in the networking research community on what these policies should be and what form they should take.

The mechanism of upcalls is limited to a single stack. The idea of policy/mechanism separation with communication of state information between the layers implementing policy and mechanism is not limited to a single stack, however, and can be usefully extended across multiple communicating stacks. An area of further work will be to consider what mechanisms can be efficiently used to extend the EIS to a multi-stack model.

New network environments are likely to invalidate many of the underlying assumptions encoded into the TCP/IP protocols. The extensive research being done on problem specific work-a-rounds for mobile environments reveal just how strongly embedded these assumptions are. We argue that to evolve the Internet into a flexible network that supports varying network behaviour and dynamic application adaptation, the architecture of TCP/IP stacks needs to be reconsidered in terms of policies and mechanisms rather than services. As we have shown in this paper, the Compliant Systems Architecture provides a sound model for designing a new network stack that meets these requirements.

REFERENCES

[1] R. Morrison, D. Balasubramaniam, M. Greenwood, G. Kirby, K. Mayes, D. Munro and B. Warboys: An Approach to Compliance in Software Architectures. Computing and Control Engineering Journal. **11:4** (August 2000) 195–200

[2] C. Perkins: Mobile networking in the Internet. ACM Journal on Mobile Networks and Applications. **4:3** (December 1998) 319–394

[3] E. Brewer, R. Katz, et al.: A Network Architecture for Heterogeneous Mobile Computing. IEEE Personal Communications. (October 1998)

[4] Improving TCP performance over wireless networks at the link layer Christina Parsa , J. J. Garcia-Luna-Aceves Mobile Networks and Applications March 2000 Volume 5 Issue 1 Pages: 57 - 71

[5] R. Caceres and L. Iftode: Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. IEEE Journal on Selected Areas in Communications. **13:5** (June 1995) 1–10

[6] R. Morrison, D. Balasubramaniam, R.M. Greenwood, G. Kirby, K. Mayes, D. Munro and B. Warboys: A Compliant Persistent Architecture. Software, Practice & Experience, Special Issue on Persistent Object Systems **30:4** (2000) 363–386.

[7] D. Clark: The Structuring of Systems Using Upcalls. ACM SIGOPS Operating Systems Review. **19:5** (December 1985) 171–180

[8] A. Edwards and S. Muir: Experiences implementing a high performance TCP in user space. ACM SIGCOMM Computer Communications Review. **25:4** (October 1995) 196–205

[9] C. Perkins (ed): IP Mobility Support. IETF Request for Comments - RFC 2002. (October 1996) http://www.ietf.org

[10] C. Huitema: IPv6 The New Internet Protocol, Second Edition. Prentice Hall 1998.

[11] S. Savage, N. Cardwell, D. Wetherall and T. Anderson: TCP Congestion Control with a Misbehaving Receiver. ACM Computer Communications Review. **29:5** (October, 1999) 71–78

[12] S. Floyd and K. Fall: Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Transactions on Networking. **7:4** (August 1999) 458–472

[13] S. Floyd: TCP and Explicit Congestion Notification. ACM Computer Communication Review. **24:5** (October 1995) 10–23

[14] R. Gopalakrishnan and G. M. Parulkar: Efficient User-Space Protocol Implementations with QoS Guarantees Using Real-Time Upcalls. IEEE/ACM Transactions on Networking. **6:4** (August 1998) 374–388