Component-based Active Networks for Mobile Multimedia Systems

S. Schmid, J. Finney, A.C. Scott, and W.D. Shepherd

Distributed Multimedia Research Group
Computing Department
Lancaster University, UK

{sschmid, joe, acs, doug}@LandMARC.net
http://www.LandMARC.net/

Abstract

New network technologies are often restricted by the limitations of today's network devices. This lack of flexibility and extensibility hinders or delays the evolution and deployment of new mechanisms. We believe that active networks have the potential to overcome these limitations by replacing current network devices with programmable nodes.

This work-in-progress report introduces a novel component-based active router architecture designed to provide support for emerging network technologies, and demonstrates the flexibility and extensibility of this architecture by addressing some of the most pressing handoff problems present in next generation IP based mobile multimedia systems.

1. Introduction

Multimedia services in mobile environments often have very strict and extended QoS requirements [1]. QoS properties such as throughput, latency and error rate are more critical since such environments are usually built upon low bit-rate wireless networks. These issues are widely discussed within the QoS research community, and solutions to the problems are evolving from modifications to network protocols and extended services (for example, DiffServ, RSVP, and IntServ).

However, when the provision of QoS support to mobile devices is also considered, additional QoS issues become apparent – most notably the latency introduced by network handoffs of mobile devices. This is of particular relevance to Mobile IPv6 [2] (the IETF standardized mobile routing protocol for the IPv6). These issues are only now being addressed within the research community.

We believe that current networks are not flexible and extensible enough to cope with this problem in an efficient and elegant manner. This work-in-progress report introduces a novel component-based approach to active networks providing a highly flexible and dynamically extensible architecture for programmable network nodes. In particular, we will show how this architecture can be deployed to resolve some of the QoS problems associated with Mobile IPv6 enabled networks.

1.1. Previous Work

The distinction between programming paradigms of network devices and end systems has led to an ever growing gap between capabilities of end nodes and intermediate devices. While for many years end systems (which are totally programmable) have had sophisticated support for multimedia streaming, multimedia support for network devices is still in the process of standardization, and deployment is held back by the problems of upgrading significant amounts of router software.

As a response, we started to investigate the performance, scalability, safety and security issues of active networking. Based on the results, we developed a novel router architecture especially for active networks, called *Lancaster Active Router Architecture (LARA)* [3]. Our main design focus was to develop a scalable high-performance active router. *High-performance* here refers to the performance of active processing. LARA achieves high-performance for the active processing by means of two concepts: (1) LARA employs high-end processor units on a per interface basis, which guarantee sufficient processing power for active computation. (2) LARA implements efficient zero copy mechanisms for the data path processing. Expensive copy operations are avoided and context switches minimized. *Scalability* is achieved through dedicated active processor units on a per interface basis. Consequently, adding new interfaces to the router increases the processing power proportionally.

Building on lessons learned from the development of LARA, our first generation active router, we are currently in the process of developing its successor, LARA++, which has a major focus on supporting emerging network technologies through flexible and extensible active router programmability.

The remainder of this document first motivates our work by introducing the QoS problems associated with Mobile IPv6 handoffs. We then continue by presenting our component-based active router architecture, LARA++, and the components we have designed to optimise the handoff performance for mobile devices. Finally, we conclude the paper by reviewing how active networking can improve mobile network handoffs and discuss future work.

2. Motivation

We have recently investigated several performance problems regarding multimedia streaming systems in Mobile IPv6 environments. In particular, we consider the two dominant factors that govern the performance of a network handoff – the speed of address acquisition and mobile routing convergence time. We believe that the efficiency of solutions to these problems can be addressed more elegantly by means of active network technology.

The following sections outline the issues that affect these factors, and how they can be improved by active networking.

2.1. Address Acquisition

IPv6's stateless address auto-configuration mechanism [4] is an excellent means for mobile devices utilising the Mobile IPv6 protocol to dynamically allocate IP addresses, as this greatly simplifies the process of network detection and care-of address acquisition. However, to prevent multiple devices from accidentally using the same IP address in a network environment, a *Duplicate Address Detection (DAD)* mechanism has been mandated [5] for IPv6. This demands that devices first verify that a newly acquired address is not already in use before actively utilising that address. Experiments have shown that while addresses can be acquired using stateless auto-configuration within a few milliseconds, the DAD verification process demands (according to the specification) up to five seconds to complete [6]. Network outages of this order are, however, not tolerable for multimedia applications.

By using active networks, routers could be dynamically programmed to monitor the availability of a mobile node's potential care-of addresses within a network domain. Thus, when a mobile device roams to a new network within the domain, it could simply ask a router on the link if the corresponding care-of address is still available. If so, the mobile node can immediately utilize the address without incurring the vast latency introduced by the DAD verification mechanism, and still remain confident that no IPv6 address collision exists on the network.

Although the aim of this approach, namely assigning unique IP addresses to network devices, might seem to be already addressed by the *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)* [7], nevertheless, both approaches are complementary. While the approach outlined here only addresses link-local and stateless address configuration, DHCP manages stateful address allocation. DHCP requires a node first to configure a link-local or stateless address, before it can acquire a stateful address. Since the DAD algorithm must be performed on all addresses, independent of whether they are obtained via stateless or stateful auto-configuration, the DAD must be performed twice when DHCP is used.

2.2. Mobile IPv6 convergence time

The second inefficiency we identified, while analysing mobility support protocols, is that Mobile IPv6 is purely a routing protocol for end systems; network routers (apart from the home agent) are not involved.

As a result, handoff times can be unnecessarily high due to the fact that handoffs require a correspondent node (any IPv6 device conversing with a mobile device) to receive a *Binding Update*[1] message notifying the correspondent node about the network change before the route change can take place. This adds at least the delay of one round trip time (between the mobile node and the correspondent nodes) to the handoff latency. As typical latencies of wide area links can be as high as 400 ms, this can result in a handoff performance that is unacceptable for multimedia applications. Figure 1 illustrates this inefficiency and proposes a solution involving network routers.

Although several research groups are already addressing this inefficiency [8, 9, 10] (and there is limited support within the Mobile IPv6 standard), all these solutions require extensions to the Mobile IPv6 standard, and/or additional control messaging to operate. By utilising active networks, this problem can be *transparently* resolved, minimising the impact on network utilisation, protocol complexity and service interruption.

Active routers can be programmed to reroute any data traffic incorrectly routed to the mobile device's old location to its new point of attachment without delay. This provides a means to repair the incorrect routing locally, until the Mobile IPv6 protocol responds to the route change.

[1] A Mobile IPv6 control message sent to the home agent and correspondent nodes to indicate a network handoff.

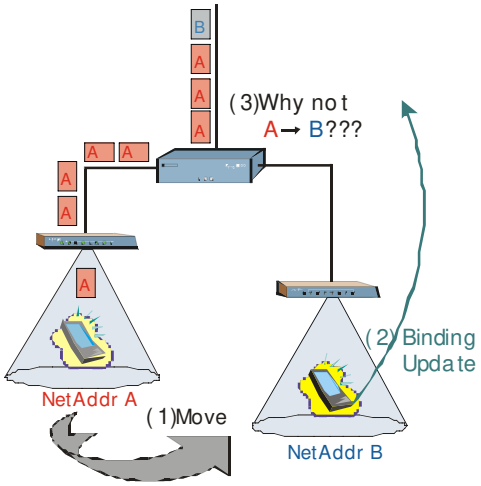

Figure 1. *Mobile IPv6 convergence time:* After move of mobile device (1), it sends binding update to inform correspondent node (2). It takes ≥ RTT until data flow reaches the new destination. Why should not the router reroute the stream immediately (3)?

2.3. Rational for Active Network Use

Solutions to both mobile handoff problems outlined above are based on active network technologies. Although it is clear that these mechanisms can also be built directly into future router software releases (thus negating the need for active networks), we propose the use of active network technologies for several reasons:

First, we believe that today's networks lack a common platform for the development and deployment of network services such as firewalls, proxies, media gateways. Active network technology has the potential to separate network services (software) from the underlying router devices (hardware) by introducing a programmable layer in between.

Second, experimentation with our mobile handoff solutions relies on support from network routers. However, since today's routers are still closely integrated systems (where only manufacturers can introduce new functionality), we would rely on support from a router manufacturer or would be forced to build our own. In the first case, the lengthy process of standardization and software development by router manufacturers would delay our research. The second case, however, complements our research in other fields. Since we need an extensible router in order to do network research anyway, why should we not invest in a programmable network router?

And finally, the deployment of network services in the absence of active and programmable technologies implicates a very longwinded process of upgrading large amounts of routers in the network before the service can be used. Naturally, the same problem occurs when active network technology needs to be introduced. However, when active routers are in place, new services can be rolled out on-the-fly.

Moving the problem from the end stations into the network is clearly not a panacea for all problems. Additional delay and other problems arising from the active processing within network routers must be considered. However, solutions to the mobile handoff problems rely only on active processing support in routers close to the mobile device's location. Thus, scalability and deployment of such active services is less critical.

3. Component-based Active Node Architecture

LARA++, the next generation active router architecture of LARA [3], provides a programmable platform for active services development based on the composition of many small components. Unlike first-generation active router architectures, LARA++ implements a very flexible and extensible programmable router platform based on the principles of decomposition.

LARA++ components are dynamically loadable onto LARA++ active routers where they provide additional or extended services for individual data streams or even for whole protocol families. Figure 1 provides a conceptual view of the component-based approach used for LARA++.

LARA++ components can be either *active* or *passive*. Active components perform the active computations on a node based on one or more execution threads, whereas passive components provide merely static functionality, similar to support libraries.

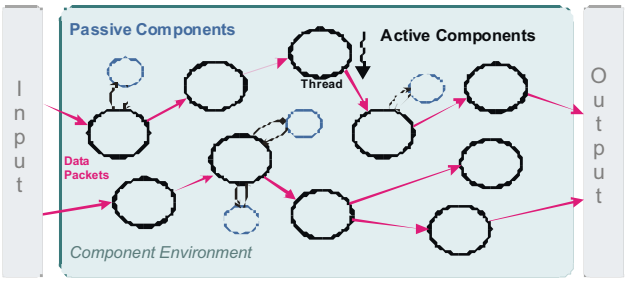

Figure 2. A conceptual view of the Active Component Space

Components can be further differentiated by their function into *user components* and *system components*. User components are those active programs or libraries that are injected by users of the active network. System components, in contrast, are the components that constitute LARA++ and expose the API to low-level resources and system calls.

The remainder of this section provides a short introduction to LARA++, its design goals and component-based architecture. Due to the very restricted space in this report, we limit the description to the basics. Further details on the component architecture and an in-depth discussion of LARA++ internals such as the loading and registration of components, component APIs, inter-component communication, service composition, policing, as well as the security and safety framework are freely available in the LARA++ design specification [11].

3.1. Design Goals

The motivation for our component-based approach arises from the fact that conventional active node architectures provide one of two paradigms – either fixed execution environments, which can "only" be programmed through in-band active programs (also referred to as active capsules), or a programmable switch platform, which enables users to download and install active programs. The former approach is limited by the programming capabilities offered by the execution environments and relies on active support by every node along the transmission path, whereas the latter is usually restricted by the support the programmable switch provides for interaction between single active software components and integration of active programs.

LARA++ tries to resolve the limitations of the programmable switch approach by providing a sophisticated composition framework for active components. This enables active services to be split into many simple and easy to develop tasks (each processed by a single active component). The services are then built from the individual components, as they are needed. This "divide and conquer" approach simplifies extensibility of the router functionality. Individual software components can be upgraded more easily and new components can be gradually added.

As a result, our component-based active node architecture is clearly advantageous over conventional implementations in terms of reusability of active code and customisability of user tailored services. The critical aspect of the design is to provide a sufficiently flexible composition method, which is further discussed in later sections.

Although LARA++ is designed to be a very generic platform for active network services, some of its design features were motivated by analysing the mobile handoff problems and their requirements for active routers in order to solve these problems in an elegant and yet efficient manner. The following items describe a number of relevant LARA++ features for mobile network handoff:

- *Dynamic extensibility of router functionality:* LARA++ routers are programmable by network users, which enables mobile devices to upgrade the network services as they enter a new domain.
- *Flexible packet filtering and insertion mechanisms:* LARA++ supports packet filtering and (re)injection mechanisms in order to react upon certain events in the network (for example, binding update messages).
- *Provision to carry out routing tasks:* Active LARA++ components have the ability to route individual flows or even protocol families (for example, to temporarily reroute a data stream to a mobile device's new address).
- *Support for active programs to "program" active routers:* LARA++ enables active components to control (i.e. load, instantiate and remove) other active components as long as security policies are not violated (for example, a component identifying a handoff should be able to instantiate a routing component).
- *Efficient processing of active programs:* LARA++ is designed such that the management overhead for the execution of active programs is small. Since the latency of the active processing increases the overall end-to-end delay of data streams, only very small delays are tolerable.
- *Lightweight instantiation and removal mechanisms for active programs:* Since active programs are often very short lived (for example, the flow routing optimisation for mobile handoffs compensates only until the binding update notifies the corresponded node), instantiation and removal of active components is designed to be lightweight and fast.

3.2. Node Architecture

A key difficulty in designing active networks is to allow nodes to execute user-defined programs while providing reliable network services for everyone. An active node must therefore protect co-existing network protocols and services from each other and securely control shared resources.

As a result, LARA++ executes active code only within restricted processing environments that limit access to low-level service routines and shared resources. The framework for our safety model is based on a layered architecture, where higher levels are limited and controlled by lower level policies.

The active node is assembled out of the following four layers[2]:

(i.) The *NodeOS* provides a set of low-level service routines and system policies to access and manage local resources and device configurations.

(ii.) *Execution Environments (EE)* form the management unit for resource access and security policies, which are enforced on every active program executed in that environment.

[2] This layered architecture extents the original active network node architecture specified by the DARPA Active Networking Group.

(iii.) *Processing Environments (PE)* provide the code space where trusted components are processed. Trust relations among components are defined upon the code producers (i.e. code signatures) and the network users (i.e. user authentication) instantiating components.

(iv.) *Active Components (AC)* comprise the active programs, which are executed within PEs alongside with other trusted components. The active code is executed by means of one or several threads.

Figure 3 illustrates the layered node architecture.

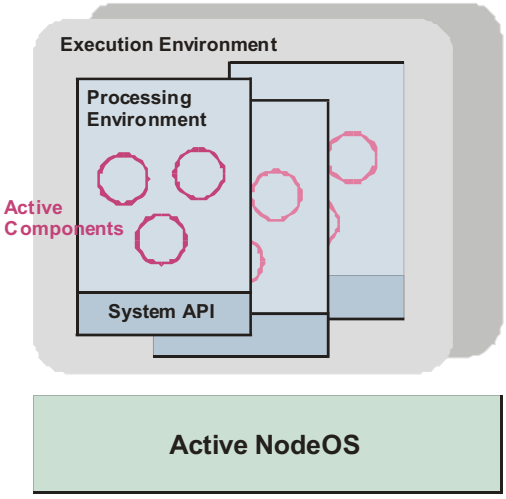

Figure 3. The layered architecture of LARA++

In contrast to other active network architectures, LARA++ introduces the concept of processing environments as an extra protection layer between the actual active programs and the execution environment. Although this might seem to make the execution of active code more complex and heavyweight, PEs were introduced for exactly the opposite reason. PEs allow components with trust relationships to be processed very efficiently in a single process-like environment. This unique characteristic of the LARA++ architecture makes use of the performance benefits arising from trust relations.

Trusted active components, which are either precompiled executables or "on-the-fly" compiled programs, are directly loaded into the PE's address space and then executed by creating one or more active threads. Trusted active programs can be executed within the same protection environment (i.e. address space) without creating security threats. The performance boost results from the fact that user-level thread scheduling is remarkably faster since only a lightweight context switch is involved. Early experiments with our LARA++ implementation have shown that user-level thread scheduling is approximately one order of magnitude faster than the scheduling of conventional active processes.

The intermediate processing environment on top of the EE has besides the obvious performance boost the advantage that the concept of *module thinning*[3] can be applied on a fine-grain basis. Each PE can provide a specially customized system API depending on the trust relation between its components and the NodeOS.

3.3. Service Composition

The structure and make-up of active services is defined by the composition method. It determines the set of software components used for an active service and the bindings linking the components [12]. Active service composition on LARA++ nodes is based on a sophisticated packet classification mechanism as described here.

LARA++ active services are composed through insertion of packet filters into the *packet classifier*. Active components typically register their filters with the packet classifier at instantiation time (and if necessary again at run-time). Thus, packets received on a physical interface undergo a classification procedure which identifies the active components to process a packet as it passes the active node. According to the order determined by the classifier, packets are "routed" through the active component space to obtain the respective active computations of the filtered components in the correct order.

Due to the classification-based service composition method, LARA++ composition is based on the *conditional binding* of active components. Since the bindings of active components rely on the data in a packet, a binding is only in force if the filter for the binding matches the packet.

The basic structure for the component bindings is defined by the TCP/IP layer model, which ensures that active components providing low-level support are executed before components dealing with higher-level computations (for example, network protocol options must be processed prior to transport protocol operations). In addition to the ordering imposed by the layer model, the individual protocol stacks also need to process their headers, options and so forth in a protocol specific order (for example, extension headers in IPv6 must be processed in a order).

In order to enable active components to flexibly extend the functionality of active services, LARA++ requires an "elastic" means to describe at which point in the processing chain a component must be inserted. For this purpose, we chose a *Management Information Base (MIB)* style representation, referred to as the *classification hierarchy*, which is sufficiently flexible to extend current protocol stacks and has provision to incorporate new protocols.

[3] Module thinning secures programmable system by individually tailoring the programming interface exposed to a software process based on the privileges of the user or program.

For performance reasons, the packet classifier maintains the packet filters in a tree style representation, called the *classification graph*. The nodes of the graph represent the packet filters for the active components currently instantiated on the active node. The graph is built according to the structure imposed by the classification hierarchy. Figure 4 illustrates a simplistic representation of the classification graph.

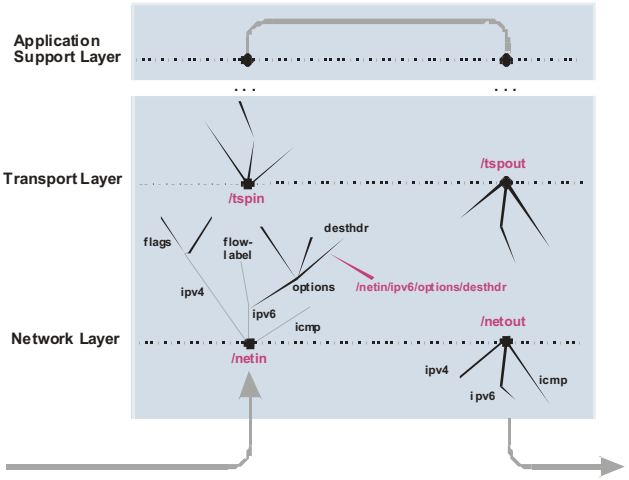

Figure 4. *The classification graph.* The grey arrow represents the general route data packets take when passing through a LARA++ node. The classification graph on the network layer part of the input and output paths show how the classifier manages the packet filters of the corresponding active components.

The packet classifier filters incoming and outgoing data packets based on the packet filters in the classification graph. The *n-nary* tree structure is parsed according to a left-to-right depth-search approach. If a filter matches a packet, it is "forwarded" to the active component associated with the filter. When the component has completed the active processing, it either discards the packet or reinserts it at the same point or any other point of the graph (depending on the system policies and privileges defined for the component). The classifier continues the classification process until it runs out of filter branches.

4. Components for Mobile Multimedia Systems

This section briefly describes the active components we have designed to resolve the inefficiencies of the Mobile IPv6 handoff mechanism as introduced in earlier sections.

4.1. Optimised Stateless Address Auto-configuration

The handoff latency introduced by duplicate address detection (DAD) is solved by means of an *Address Placeholder Component (APC)*, which actively monitors the state of mobile nodes' potential care-of addresses within a network domain and performs DAD on behalf of mobile nodes. Thus, mobile devices using this active service can avoid the handoff delays introduced by DAD.

Upon entering a new "access domain" or roaming to a new network within the domain, a mobile device invokes an APC in the domain router by means of a domain local or predefined multicast. The invocation message consists of the active code for the APC or simply a component reference and the mobile's network interface identifier (i.e., the EUI-64 number [4] or another unique identifier) as parameter. Note that if a referenced component is not yet loaded, the active router retrieves the code automatically [11]. For each APC invocation LARA++ routers instantiate a new active component in a processing environment that trusts the active code. Since the active code for the APC is most likely to originate from the same code producer, all APC are executed within the same PE.

Upon instantiation, the APC performs the DAD mechanism for the respective IPv6 address (created from local network address and the mobile's interface identifier). If the DAD fails (i.e. the address is already in use) then the component immediately terminates. Otherwise, the APC monitors the ongoing state of the address by actively listening for the neighbour solicitation [5] messages sent by "new" devices on the network that are performing DAD. This is achieved by the APC installing a LARA++ packet filter that traps these solicitation messages. If the APC receives such a solicitation message for one of the addresses it is monitoring, it transmits an *address-available* control message to the source of that solicitation (indicating that DAD has been performed for this address, and it is safe to immediately use), then terminates. A mobile device performing DAD that receives an address-available message can therefore safely bypass DAD and the associated delay.

This approach has the advantages that it is based on a very simple soft-state mechanism that circumvents complex signalling protocols. The positive acknowledgement technique indicating address availability ensures – even in the exceptional case of active router reboot, active code termination, or packet loss – that no duplicate address allocation takes place. If the mobile device does not receive the *address-available* message, it simply performs the DAD as obligated by the IPv6 standard.

Considering the scalability of this mechanism reveals that active routers on large access networks (with many mobile devices) have to cope with large number of APCs (one APC for each roaming mobile device). However, the component-based architecture of LARA++ allows LARA++ routers to maintain only a single code copy for

each active component and still manage multiple instantiations. Nevertheless, it is clear that the maximum number of APCs and respectively the maximum number of mobile devices are limited.

4.2. Optimising Handoff over High Latency Links

The network handoff latency of mobile devices can be minimized by means of a generic LARA++ *Flow Routing Component (FRC)* which introduces Mobile IPv6 functionality into active routers, such that they actively take part in mobile routing as pointed out in Figure 1.

Upon network handoffs, mobile devices are mandated to inform the home agent and correspondent nodes about the network change by sending binding update messages. In order to reduce the mobile routing convergence time, the mobile device injects (or simply invokes) an FRC into active LARA++ routers along the reverse transmission paths[4]. After instantiation on a LARA++ router, the FRC immediately re-routes any packets misrouted to the mobile node's old location to its real location by means of network address translation [13]. The FRC re-routes the data streams by registering a packet filter specifying the relevant filter information (i.e. old care-of address, home address) with the classifier. Since the flow routing optimisation is only required temporarily (until all relevant correspondent nodes have received a binding update), the FRC is programmed to timeout and remove itself after a period of inactivity (typically within a few seconds of instantiation).

This approach has again the advantage that it is based on a simple soft-state mechanism, which falls back into conventional operation mode if the active code accidentally stops. Since the FRCs are very short lived, the impact of network level processing in the active routers and scalability issues are less critical. Apart from the additional filtering expense in the active router, this solution only introduces active processing cost for flows that are misrouted and only for the short time a FRC is active. The fact that the FRCs are very short-lived, which confines the number of FRC instantiated at any given point in time, makes the proposed approach scalable.

5. Related Work

On the one hand, many early active router architectures are built on the principles of "active capsules" [14], where data packets are replaced by so-called capsules which include small active code fragments. Although this approach to active networks changes the communication model compared with today's networks more radically (thus might be more interesting from a research point of view), it is for several reasons not suited for the mobile handoff solutions discussed in this document:

- The active capsule approach requires all the routers along the transmission path and the mobile devices to be active (which might never happen in large-scale networks such as the Internet), although conceptually only the routers of the mobile access networks require active support.
- Transparency can only be achieved, if the packet format of the media streams and the end systems do not require modifications. However, the active capsule approach demands active code to be insert in the data packets, which implies extension of the end systems.

On the other hand, active router architectures that are like LARA++ based on the "programmable switch" approach [14] to active networks are more appropriate to implement the mobile handoff solutions proposed in this paper. A number of relevant features to implement these handoff components are discussed in section 3.1.

The remainder of this section compares two recently developed programmable switch architectures with LARA++.

(1) The Router Plugin project [15] proposes router programmability based on software modules that can be plugged in at specific *gates* in the IP processing path. This enables network users to customize a router avoiding the performance penalties of layered active network protection. Consequently, Router Plugins rely on external safety and security mechanisms. Since plugin processing is integrated in the router OS, Router Plugins do not introduce the thread scheduling and queuing overheads of the LARA++ processing path. Although Router Plugins provide limited scope for extensibility (plugins can only be injected at specific points in the IP forwarding path) and programmability is less flexible compared to LARA++ (it is optimised for packet forwarding rather than more general purpose processing), the architecture would be suitable to implement the proposed mechanisms.

(2) The Bowman active Node OS [16] together with the CANEs execution environment [17] provides similar flexibility and extensibility based on a composition framework for active code modules. While LARA++ uses a flexibly extensible classification hierarchy in order to describe at what point in the processing chain an active component needs to be processed, a generic "underlying program" defining the slots is used to customize services within CANEs. As a result, this active network architecture would also fit the requirements to implement the proposed mobile handoff components. However, the Bowman design does not account for run-time safety and security mechanisms such that only trusted code can be executed. LARA++, in contrast, ensures safe execution of active components by means of PEs.

[4] Note that although active network support might be restricted to the local network domain, the handoff optimisation is nevertheless effective, since mobile route changes occur primarily close to the mobile's network location.

6. Conclusion

We believe that the lack of active support in current IP networks has hindered the development and deployment of novel network services and advanced solutions to overcome the limitations of today's network technologies.

As a result, we have designed and are currently developing a component-based active network node architecture, called LARA++. We demonstrated its flexibility and extensibility properties by introducing two simple to develop and yet efficient active components, which have the potential to significantly reduce the network handoff of roaming Mobile IPv6 devices when loaded and executed on active LARA++ nodes in the mobile network.

The first component, also referred to as APC, supports the Mobile IPv6 protocol by providing a means to monitor the availability of unused network addresses. This enables roaming mobile devices to evade the long-winded duplicate address detection that is required for stateless address configuration when entering a new network.

The second component, called FRC, complements Mobile IPv6 routing. While the routing convergence time in normal Mobile IPv6 network takes at least one RTT (between the mobile device and a correspondent node) in the case of a network handoff, the FRC enables active routers to fix the routing to the mobile's new location immediately (within a couple milliseconds) until the Mobile IPv6 routing protocol adapts.

Further work in the LARA++ area will involve completing the implementation of the active node architecture and experimentation with the proposed Mobile IPv6 handoff enhancing components. The work will be based around the LandMARC implementation of Mobile IPv6 for MS Windows 2000/WinCE [18].

In the longer term, research is also planned into the benefits of LARA++ component technology in end-stations, in addition to network routers.

7. Acknowledgements

The LARA++ work documented was undertaken as part of the Microsoft funded LandMARC collaborative project between Lancaster University, U.K. and Microsoft Research, Cambridge U.K. [18]. The LARA platform was developed under U.K. EPSRC (grant no. GR/L59603).

8. References

[1] N. Davies, et al., "Supporting Adaptive Video Applications in Mobile Environments", IEEE Personal Communications, Special Issue on Video Over Wireless, September 1999.

[2] D. Johnson, C. Perkins, "Mobility Support within IPv6", IETF Internet draft draft-ietf-mobileip-ipv6-11.txt, March 2000. Work in progress.

[3] R. Cardoe, et al., "LARA: A Prototype System for Supporting High Performance Active Networking", Proc. IWAN'99, June 1999.

[4] S. Thomson, "IPv6 Stateless Address Autocofiguration", T. Narten, Internet RFC 2462, December 1998

[5] T. Narten et al., "Neighbor Discovery for IP version 6 (IPv6)", Internet RFC 2461, December 1998.

[6] J. Finney, "Supporting Continuous Multimedia Services in Next Generation Mobile Systems", Ph.D. Thesis, Lancaster University, September 1999.

[7] J. Bound, M. Carney, and C. Perkins, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", IETF Internet draft draft-ietf-dhc-dhcpv6-14.txt, May 2000. Work in progress.

[8] R. Caceres, et al., "Fast and Scalable Handoffs for Wireless Internetworks", ACM Mobicom '96, pp. 56-66, November 1996.

[9] A. Campbell et al., "Cellular IP", IETF draft draft-ietf-mobileip-cellularip-00.txt, January 2000. Work in progress.

[10] K. El Malki, H. Soliman, "Hierarchical Mobile IPv4/v6 and Fast Handoffs" IETF draft draft-elmalki-soliman-hmipv4v6-00.txt, March 2000. Work in progress.

[11] S. Schmid, "LARA++ Design Specification", Lancaster University DMRG Internal Report, MPG-00-03, January 2000.

[12] E. Zegura, "Composable Services for Active Networks", Active Network Composable Services Working Group Draft, May 1998. Work in progress.

[13] K. Egevang, et al. "The IP Network Address Translator (NAT)", Internet RFC 1631, May 1994.

[14] D.L. Tennenhouse et al., "A Survey of Active Network Research", IEEE Communications, January 1997.

[15] D. Decasper et al., "Router Plugins: A Software Architecture for Next Generation Routers", SIGCOMM '98, Vancouver, CA, September 1998.

[16] S. Merugu et al., "Bowman: A Node OS for Active Networks", IEEE Infocom '00, Tel Aviv, March 2000.

[17] S. Merugu et al., "Bowman and CANEs: Implementation of an Active Network", 37th Allerton Conference on Communication, Control and Computing, September 1999.

[18] "The LandMARC Project", available via the Internet at http://www.LandMARC.net, Mar. 2000.