FARA: Reorganizing the Addressing Architecture*

David Clark† Robert Braden‡ Aaron Falk‡ Venkata Pingali‡

† MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
ddc@lcs.mit.edu

‡ USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA, USA 90292
{braden,Falk,pingali}@isi.edu

ABSTRACT

This paper describes FARA, a new organization of network architecture concepts. FARA (Forwarding directive, Association, and Rendezvous Architecture) defines an abstract model with considerable generality and flexibility, based upon the decoupling of end-system names from network addresses. The paper explores the implications of FARA and the range of architecture instantiations that may be derived from FARA. As an illustration, the paper outlines a particular derived architecture, M-FARA, which features support for generalized mobility and multiple realms of network addressing.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Protocol Architecture*

General Terms

Design

Keywords

Network, Architecture, Model, Instantiation, Modularity, Association, Rendezvous, Mobility, Security

1. INTRODUCTION

This paper suggests a new organization of the concepts of naming and binding [11] that underlie the current Internet architecture. This organization is embodied in a general architectural model named FARA (an abbreviation of: "Forwarding directive, Association, and Rendezvous Architecture."). The FARA model defines an abstract set of components and the modular relationships among them, while it leaves undefined many detailed technical mechanisms and design decisions. Thus, FARA defines a class of architectures from which a variety of specific architectures can be instantiated by adding additional assumptions and defining various mechanisms. To clarify the FARA model, this paper describes one particular instantiation, the M-FARA architecture (Section 4.)

Although many of the individual ideas combined in FARA are not new (see Section 5 on prior work), the synthesis of concepts in FARA provides a conceptual framework for reasoning about protocol architecture that we believe to be new.

On another level, this paper is an illustrative exercise in abstract reasoning about protocol design. It explicitly divides the reasoning process required to create a new architecture into two stages. FARA represents the first stage: the definition of a general high-level model that satisfies a specific set of assumptions (Section 2.2). In the second stage, a complete architecture is derived as an instantiation of the general model. This instantiation reduces the generality of the model by defining specific mechanisms, but the derived architecture satisfies the assumptions of its parent high-level model. It also defines many details that are unimportant to the general architecture, such as protocol fields and headers. The purpose of this two-step process is to make explicit the important relationships among specific design decisions within the derived architecture. The success of FARA or any analogous high-level architectural model can be judged by the clarity as well as the utility of derived instantiation architectures.

It is well known that IP addresses [10] are overloaded, as they indicate both network locations and node identities[11]. This overloading has advantages and disadvantages that are also well-known. In simplest terms, overloading the network address provides some (minimal) security but makes mobility more difficult. Conversely, decoupling of location from identity results in an architecture that facilitates mobility but makes security harder. A central theme of the FARA model is to avoid this overloading by: (1) cleanly decoupling application identity from network-layer forwarding mechanisms while (2) avoiding the (necessary) introduction of a new global namespace and (3) providing security with a range of assurance levels.

By separating location from identity, we free each part of the design from the constraints that would otherwise arise due to interactions and interdependence. As a result, the

*This work was supported by DARPA under grants F30602-00-2-0553 (MIT) and F30602-00-1-0540 (ISI), as part of the NewArch project.

ACM SIGCOMM 2003 Workshops August 25&27, 2003, Karlsruhe, Germany

network designer is free to pick an addressing scheme that meets his needs, and to independently pick a naming scheme. The generality of the FARA design is the confirmation that the ideas of location and identity can in fact be separated.

Section 2 introduces the abstract components and explains the basic assumptions of FARA. These definitions and assumptions determine the range of possible architectures that can be derived from FARA. Section 3 explains the FARA model more completely. Section 4 then describes a specific derived architecture, M-FARA, including a brief description of a prototype implementation. Section 5 briefly surveys prior work, and Section 6 presents conclusions.

2. OVERVIEW OF FARA

2.1 Basic Components

In FARA, host-to-host communication is replaced conceptually by communication between pairs of **entities** via logical linkages that are called **associations**, using packet exchanges over a **communication substrate**. This section summarizes these three architectural components of FARA. Three more components – rendezvous, the FARA directory service, and slots – are introduced later (Section 3.)

- **Entity**

  An *entity* is the generalization of an application that is an end-point of network communication. An entity contains state, both application state and communication state, and an entity is the smallest unit that can be mobile.

  The term "entity" is deliberately chosen to be abstract. An entity might be a process, a thread in a process, a set of processes, an entire computer, a cluster of computers, etc. FARA is intended to encompass any of these forms of entities, although the protocol design details of a FARA instantiation may depend upon the particular entity structures that are supported. A Unix process in a non-mobile host could be a simple example of a non-mobile entity. At the other extreme, a complex entity might have its own internal communication structure that is hidden from FARA, for example it might be a computer cluster. For communication with such a cluster entity, FARA would deliver packet to a portal system, e.g., a load-balancer, which would forward these packets to some server node invisibly to FARA.

- **Association**

  FARA entities communicate with each other using logical communication links called *associations.* An association implies persistent communication state within the linked entities[1]. This state evolves over the life of the association and is synchronized by the association. An association may be considered to be roughly analogous to a transport-layer connection in the current architecture.

  Abstractly, an association is the combination of relevant communication state in each entity and an ongoing sequence of packets that are flowing between the entities. Each packet belongs to exactly one association, and an entity may have multiple concurrent associations. Associations are purely end-to-end; they are known only to their entities and are invisible to the routers.

  Under FARA, each packet carries an **association ID** (**AId**) that enables the receiving entity to demultiplex the message to its association. Since AIds are local to each entity there may be a distinct AId at each end, so it is convenient to carry a (source AId, destination AId) pair in every packet. However, FARA itself does not specify the format of an AId.

- **Communication Substrate**

  Entities live in a world of lower-level supporting systems, including operating systems and networks. FARA assumes some network communication substrate that delivers data on behalf of associations. In particular, FARA assumes a connectionless[2] packet delivery mechanism with appropriate addressing and routing, although it does not restrict the particular choice of mechanism(s). Possible mechanisms might include conventional hop/hop delivery, explicit (source) routing, or label swapping, for example.

  The choice of a particular forwarding mechanism must take into account a range of issues on which the FARA model is silent, for example the performance of network nodes, or the balance of anonymity vs. identity for communicating end-nodes. Just as NAT boxes hide the number and identity of the hosts behind them, so some forwarding schemes may provide a high degree of invisibility at the cost of reduced accountability. The particular case of conventional hop-by-hop forwarding based upon a single globally-unique binary address would provide a mid-point on this anonymity/identity scale, while some schemes might mandate a more robust indicator of identity and hence accountability without anonymity.

  Since we assume only connectionless service from the communication substrate, entities are responsible for end-to-end reliability when required. When an entity wants to send a packet for one of its associations, it hands that packet to its communication substrate with a header field that we call the destination **Forwarding Directive (FD)**. The destination FD contains the information needed to cause eventual delivery of the packet to the desired destination entity, although a particular forwarding mechanism may rewrite the FD in route. In addition to the destination FD, a packet may also contain a *reply FD* that can be used to deliver a return packet to the source entity.

  Note that FARA packet delivery is not to a node or to a destination host's protocol stack, but "all the way" to the entity containing the remote end of the association. This delivery is effected by the network and by the operating system environment within which the

[1]The FARA model is currently defined only for point-to-point communication between pairs of entities. Initial efforts to extend FARA in a completely natural way to multipoint delivery have been less than satisfactory, which could be considered to be a defect of the FARA model.

[2]It would be possible to further generalize FARA to allow a connection-oriented substrate, but we chose not to do so.

destination entity operates. Once a packet is delivered to the destination entity, this entity must interpret the destination AId to find the corresponding association state.

The essential modularity of the FARA model separates the forwarding mechanism of the communication substrate from the end-to-end communication functions performed by entities. We have found it useful to visualize a (metaphorical) "**red line**" demarcating this separation; the communication substrate operates "below the line," while the entity and its associations operate "above the line." A complete architecture instantiated from the FARA model must define an interface specification (API) corresponding to the red line, to maintain modular separation of forwarding function from entities.

This separation allows the independent evolution of forwarding mechanisms and applications, so that the details of the structure and semantics of the communication substrate are hidden from an entity. As discussed above, one of the goals of the FARA design exercise was to demonstrate that this degree of generality does not interfere with the rest of the design.

A further benefit of this separation is the freedom to change the delivery path for packets belonging to a particular association, i.e., to change the entity's logical "point of attachment". This is a logical point of attachment because an entity generally corresponds to an application, not (in general) to a physical box. This freedom is a definition of (generalized) *mobility*, a capability that the current Internet architecture lacks. Generalized mobility may come in different flavors. An entity may move within the same system or to a different system. It may also move because the entire system moves, either physically or logically (e.g., renumbering, provider-based addressing, etc.) FARA covers all these forms of mobility with a common architectural model. FARA itself assumes, but does not define, specific mechanisms to ensure that each entity maintains up-to-date FDs defining the forwarding paths for its ongoing associations. However, an architecture instantiating the FARA model must define such mechanisms; Section 4 describes the M-FARA mechanism in particular.

2.2 Key Assumptions

We now summarize the most fundamental assumptions that together distinguish FARA from other network architectural models.

1. Mobile Entities

   An entity is the unit of mobility in FARA. An entity that moves, moves as a unit, carrying its application state and communication state including the end points of its associations. Of course, an entity may also move because its containing system – the end node or even the attached subnetwork – moves. The FARA model allows this as well.

2. Association Naming

   FARA does not have a global namespace for associations.

   An association ID (AId) must be unique within the destination entity, but it is local to that entity. Since it is local, an AId is unchanged by movement of the entity.

3. Entity Naming

   An association is *not* a name for an entity, and in fact FARA *does not define a global set of entity names.* To communicate with a specific entity, it must be possible somehow to determine its unique location – e.g., by a lookup from some user-friendly character string, or as a process id within a particular system, or via a static FD – but there is no universal name for an entity within FARA. FARA instead provides higher-level services to help users to find remote entities for the purpose of communication (Section 3.1). The location of an entity, as distinct from its identity, is defined by the FD that will forward packets to that entity.

   The underlying philosophical point here is that communication with a remote entity only requires the ability to get a packet to it; you don't necessarily have to know its name. Furthermore, the mechanism used to learn how to get to it may be global, or it may be local.

4. End System Addresses

   FARA *does not require that end systems have addresses chosen from a single global address space.* As described earlier, FARA is intended to allow a range of forwarding mechanisms. Some of these mechanisms may use a single global address space (and in fact such an address space is awfully convenient), but some may not.

To meet FARA's goal of decoupling entity identity from its location *without requiring the introduction of a new namespace of entity names* raises two major issues: (1) how do associations get established when there is no global namespace for entities, and (2) how are packet delivery paths maintained as entities move? Association creation is discussed in Section 3.1. FARA assigns maintenance of packet delivery paths to the communication substrate (see Section 3.3) but does not otherwise define the specifics. We assume that a particular elaboration of FARA will provide a mechanism to keep delivery paths for current associations up to date. Section 4 illustrates one possible mechanism, which is used in M-FARA.

3. FARA FUNCTIONS

We now describe in more detail the basic functions of the FARA model.

3.1 Creating an Association

How do two entities A and B establish an association?[3] This will generally involve some kind of handshake that begins when, say, A sends an initial message to B. Entity A must have an FD to reach B. Subsequent packets from A to B will also carry an AId for the association, but the initial packet cannot; since AIds are local to entities, the destination AId can only be determined during the handshake. Thus, there is a bootstrapping problem whose solution requires two additional FARA components: a rendezvous mechanism and a FARA Directory System (fDS).

- **Rendezvous Mechanism**

  The initial packet in an association must be special: rather than a destination AId, it contains a *rendezvous*

[3]Note that "A" and "B" here are identification tags used for this document; they do not correspond to identifiers within the FARA model.

information (RI) string, which the destination entity can use to establish the association and assign an AId. In addition, if this is the first association of a client-server relationship, the initial packet may need a special FD to reach a dispatcher daemon. The RI string would then supply whatever parameters (e.g., service name) are needed by that daemon to start the entity and create the association.

This discussion assumed that rendezvous takes place on the target (server) system. However, the FARA model admits of more general rendezvous mechanisms. For example, the rendezvous point might be an agent at some central location [14]. Upon receiving the initial packet, this agent could rewrite the initial FD to point to the correct target entity or even to another rendezvous agent.

The rendezvous mechanism has two parts, *discovery* and *initiation*. The discovery part returns an FD and an RI string. The *initiation* part of rendezvous is the handshake required to actually establish the association state, once the (FD, RI string) pair has been discovered.

- **FARA Directory Service**

  Discovery may be accomplished by a variety of high-level mechanisms, including DNS-like directory systems, Web sites, or other programs. The FARA model subsumes these various discovery processes into a single generic *directory service*, leaving the details to an instantiating architecture.

  Note that the service names used by the FARA Directory Service (fDS) are not necessarily global and unique. Private name spaces may exist and some entities may not be named, while the rendezvous process may have to do a final disambiguation among candidate (FD, RI string) pairs. Furthermore, the architecture must allow communication even if the fDS is unavailable, to restart the network after a power failure and to provide low-level services like network management. Like dotted-decimal IP addresses, manually-constructed (FD, RI string) pairs are needed for this purpose.

In simple cases, the (FD, RI string) pair returned by the discovery process can be used directly by the initiating entity. In more complex cases, the initiating system may have to perform a transformation on the pair in order to generate the full FD and the final RI string. Such a mechanism is proposed for M-FARA, whose assumptions allow source-dependent FDs (see Section 4).

3.2 End-to-End Security

The overloading of the IP address in the current Internet architecture gives the receiver some limited assurance about the source of a packet. However, source addresses are easily spoofed, and real assurance can be provided only by cryptographic security. Decoupling FDs from associations removes even this limited assurance and forces the architecture to deal directly with the end-to-end security issue.

Therefore, FARA entities may (and generally should) implement some mechanism to validate themselves to each other, during the handshake that creates the initial association. The FARA model permits two entities to choose whatever means they want for this validation. FARA makes this a private matter between consenting entities, just as end-points under the current architecture may use clear-text passwords, ssh, shared-key crypto, etc. In particular, two entities might use a means of mutual identity that does not reveal their identity to third parties.

It is also generally necessary to verify that the source of each packet is the remote end of the same association. The FARA model does not specify the mechanism used for this packet source verification; it might or might not be the same mechanism used for the initial entity validation. Source verification will generally require some state that is shared between the two ends of an association and established by the handshake that creates the association.

An important FARA goal is to support a range of source verification mechanisms, ranging from a full cryptographic signature on each packet to no security. The Host Identity Protocol (HIP) [8] and the IP Security Protocol (IPsec) [5] (transport mode) are examples of existing crypto-based security mechanisms that could be adapted to this purpose. However, FARA allows intermediate levels of assurance between these two extremes. Some applications may require high assurance that demands authentication of every packet, while others may be satisfied with performing verification only when an association is created or when the remote FD changes due to mobility. Generally, overhead, connection setup complexity, and perhaps setup latency increase as assurance level increases.

3.3 Communication Substrate Mechanisms

Although FARA deliberately tries not to constrain the mechanisms used to implement packet forwarding, we can list the functions that we generally expect to be performed "below the [red] line".

- Packet Delivery

  The basic function of the common substrate is connectionless ("best-effort") packet delivery. Thus, it generally corresponds to the network layer of the current architecture. However, the FARA model allows a wide range of forwarding mechanisms, and a FARA instantiation might allow multiple forwarding mechanisms to coexist in different parts of the network, or even in the same nodes. Different mechanisms will give a different tradeoff between mobility, identity and anonymity, and different schemes to maintain an association will give different degrees of assurance that the relationship has not been corrupted.

- FD Management

  A specific FARA-derived architecture must include a mechanism in the communication substrate called *FD management*, to provide FD manipulation and signaling functions. In particular, the ability of an entity to move, changing its logical attachment point, requires a robust mechanism to update FDs to track these changes.

  This in turn places requirements on the API that realizes the "red line". An entity must be able to notify FD management that it is moving, and FD management needs to notify the entity of the new FD.

- Delivery Failure Notification

There will generally be an ICMP-like function to report packet delivery problems. The FD/forwarding mechanism must in general support this error-reporting function. This puts some constraints on the forwarding mechanisms that can be used. There must be a reply FD in each packet, and it must be rewritten as necessary as the packet is forwarded.

- Resource Control

  Packets consume resources, which is manifested by congestion and its control and possibly by a requirement for QoS. The FARA model does not specify any particular congestion/QoS header, but it is not hard to imagine what this might look like. Such a header would be in the part of the packet that is visible to the network elements (e.g. in the same part of the header as the FD), but there must be an interface that allows an entity to get and set this header.

- Network-Layer Security

  The communication substrate is subject to resource attacks – theft of service, denial of service, etc. Such attacks naturally require "below the line" security measures. Current examples of such mechanisms might include tunnel-mode IPsec, authenticated admission control for QoS, and packet trace-back facilities.

  Such mechanisms may demand the communication-substrate have its own robust notion of identity. We claim that this manifestation of identity is essentially different from the identity used "above the line" to validate entities to each other, as described in an earlier section. The design criteria are very different and the two should not be confounded. If it is required, identity in the communication substrate has to be of a public nature. An end-point is identifying itself to the network, to the "authorities", and so on. We note that such a lower-level identity mechanism may add to the overhead in packets and constrain what sorts of FD dynamics can be supported.

3.4 Forwarding Directives

The FARA model envisions a range of possibilities for the form and function of an FD, corresponding to a range of forwarding mechanisms in the communication substrate. For example, an FD might or might not be inherently reversible; it might be rewritten in flight; it might be independent of the location of the sender in the network, or not.

However, the FARA model implies some general characteristics of FDs.

- FDs must be expressed in a uniform and well-defined syntax.
- FDs should be derived from the network topology, so that forwarding decisions can be efficiently derived by FARA routers.
- An FD may be a generalized source route composed of a series of sub-FDs, each of which has meaning within some scope.
- In particular, every FD will have a network part and a local-delivery (*slot ID*) part. The network part controls delivery of the packet to a protocol stack in the node containing the entity, and the local-delivery part is used to complete delivery to the designated slot. This is analogous to an (IP address, port number) pair in the current architecture.
- FDs may contain identifiers drawn from a global address space. Such identifiers can be relatively static and can be registered in the fDS. However, this does not guarantee global reachability; an FD that uses globally-unique identifiers may still describe a forwarding process that will not result in successful packet delivery when initiated from some part of the network.
- Some FDs may be *reversible*, i.e., there may exist a transformation from a destination FD to a corresponding reply FD. Some reversible FDs will work from anywhere in the network.
- FDs may be transformed in route, generally at boundaries between addressing realms. This transformation may be controlled by state either in the FD ("source routing") or in the network ("label swapping"). Such FD transformations might be undone at the egress from a realm, using a stacking mechanism in the FD; this would provide the equivalent of IP-in-IP encapsulation.

To enable one-way or anonymous communication, FARA does not require that every packet contain a reply FD. However, entities may choose to not accept packets without sender identification; this preference might be expressed in the RI string, for example. Note that the absence of a reply FD will prevent notification of the sender in case of a forwarding or delivery failure.

A FARA objective is to allow changes in the forwarding mechanism without changes to applications, so the internal structure of an FD must be transparent to an entity. On the other hand, entities need to create FDs, to save FDs in their association state, and to update these saved FDs when they change. Furthermore, the change of an FD due to mobility may force an entity to invoke a re-authentication procedure with the remote entity. An FD change may also trigger invalidation of transport-protocol state for the association, e.g., cached RTT measurements. Entities should treat FDs as opaque objects, but the FD management functions provide services and an API that entities can use for creating, analyzing, saving, and updating FDs.[4]

3.5 Slots

The description of an FD given earlier was conceptually incomplete. An FD tells the network how to deliver a packet to a logical location in some system, yet the packet is to be delivered to the abstraction called an entity. To make sense of this, we must add an additional architectural component, the place to which an FD directs packet delivery and in which an entity is located. This "place" is called a **slot**.

[4]An end-system implementation is likely to have an OS-supported function that creates a level of indirection between an association's handle for sending and receiving packets, and the corresponding FDs. Hence, an application program itself normally won't be explicitly aware of FD updates. However, the function of updating the FD in the association state is still logically within the entity abstraction.

- **Slot**

  Strictly, an FD tells the network how to deliver a message to the entity that is currently occupying a particular *slot* in the target system; see Figure 1. Since FDs deliver to slots, a slot is a logical point of attachment of the entity to the network topology. The implementation of a slot is system-dependent.

  When a mobile entity moves to a different slot, the FDs for existing associations must be updated to deliver to the new slot. When a packet is delivered to a slot, a mobile entity may have already departed, leaving the slot is empty; the packet is then discarded. This is a denial of service, but it has no other effect. Mobility may also have moved another entity into the same slot, so entities must be prepared to reject packets that do not belong to their entities. This reduces to the security problem described earlier.

3.6 The Protocol Stack

While the FARA model does not specify a particular packet encoding, we can draw some general conclusions about the protocols to implement FARA. For normal data packets, the following headers will be required.

- The basic forwarding functions "below the line" would be controlled by a *network layer* header. This layer must carry the destination and source FDs. It is likely to include mechanisms for fragmentation of datagrams and for loop prevention analogous to (or exactly) those of IP.

- There would be one or more protocol layers "above the line"; we can think of these collectively as the *association layers*. The association layers must carry destination and source AIds. They will also carry the association state information that is currently in a transport-layer header for reliable delivery, etc. Finally, they will carry the security information needed for source verification and/or (re-)authentication.

3.7 The System Model

Lurking behind every abstract network architecture, there should be some model of how it can be realized in real systems. FARA changes our concept of how an end node would implement the protocol stack. Figure 1 illustrates one entity within a slot of a FARA-based end system. The dotted box at the bottom represents the lower-level operating system functions in the end system, which use the FD to deliver a packet to a slot[5].

In FARA, stored association state (symbolized by vertical rectangles in Figure 1) is logically within the entity. This state might include information for authenticating the remote entity and for reliable delivery, for example. The entity uses the AId to match an incoming packet with the correct stored information, so that it can validate the packet and act on it; this process is implied by the arrows.

It is important to understand that the "red line" will not in general correspond to the user/kernel boundary in a system. Although each entity could have its own code and algorithms for its associations, we expect that there will be

[5]This slot is shown as *multi-homed*, with network attachment points reached with forwarding directives FD1, FD2.

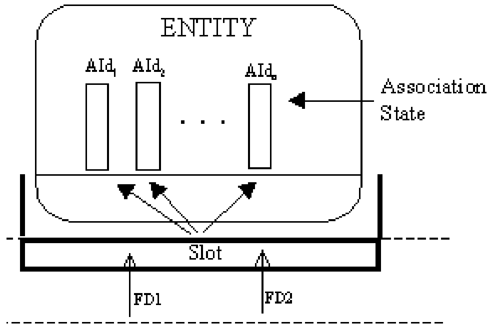

Figure 1: FARA End System

standardized association protocols that can be implemented in common library or system routines available to every entity. Furthermore, although the abstract FARA model associates this state with the entity, it does not follow that all network protocol code and state must be in user space. A particular implementation can still put transport-like functions into the kernel.

The sequence of events when a service entity starts up might be the following. The entity code would request the allocation of a slot, and then query the local routing subsystem (through FD management) to learn its own FD (or FD fragment, from which a complete FD can be computed); this would include its slot ID. If the entity provides a service that is cataloged in the fDS, the entity could now register a ($servicename \rightarrow FD$) mapping with the fDS. To do this, the entity asks the kernel for the FD of the fDS, creates an association with the fDS, and sends the registration request.

4. A FARA INSTANTIATION: M-FARA

The strength of the abstract FARA model is that it provides a consistent framework from which a wide variety of specific architectures can be derived. To illustrate this, we now describe M-FARA, one particular architecture derived from the FARA model.

M-FARA is not at this point a complete architecture; it defines some, but not all, of the mechanisms that FARA leaves unspecified. In particular, M-FARA was designed to explore and illustrate the implications of FARA for mobility and addressing domains, so it provides a specific set of mechanisms for addressing, forwarding, FD management, and security. The important issues of rendezvous and directory service have been deferred for later work.

Furthermore, the features chosen for M-FARA were motivated by a desire to explore the generality in FARA, rather than by a consideration of whether they are the best choice for a real architecture. We do claim that, because it inherits FARA's general separation of naming from addressing, M-FARA provides an interesting mechanism for seamless mobility across diverse address spaces.

4.1 Network Addressing in M-FARA

The choice of packet delivery mechanism depends upon the assumptions about network addressing and forwarding. M-FARA explores the implications of not assuming a single global address space. Instead, it assumes that there are multiple domains, each of which is a distinct *addressing realm*, similar to the world of private address spaces created by

NAT boxes. Within each realm there is a space of unique addresses that can be used, for example, for conventional hop-by-hop forwarding within the realm. As a result, a destination FD is independent of the topological location of the source entity if it is in the same domain[6]. An M-FARA FD then contains a generalized source route of sub-FDs, to traverse each realm along the path. We also assume that the reply FD will be transformed along the path so that it will be meaningful at the destination.

When an entity moves to a new location, it must compute a new FD for the "other end" of each existing association. This implies that the routing subsystem, or some other subsystem, must be able to transform FDs meaningful at the old location into FDs meaningful at the new. If the internetwork consists of a flat network of private domains, this FD translation could be arbitrarily complex. To avoid this, M-FARA assumes a two-level domain hierarchy, with a distinguished *core* domain in the "center". Then a complete FD to deliver a packet may be composed of two FD fragments: (FDup, FDdown). FDup delivers the packet into the core domain, while FDdown forwards a packet from the core domain to the appropriate private routing domain.

We call FDdown a *canonical route* because it is invariant between the various end-to-end routes used to reach the destination entity. We assume that every entity can obtain, using local knowledge, a path FDup from itself to the core. It is then easy to compute an end-to-end FD: the source entity's FDup fragment is concatenated with the canonical route FDdown of the destination entity[7].

Canonical routes are advertised through the fDS and saved as part of association state. They can also be used for constructing more efficient routes between two entities, although this is not necessary for correctness. Once an entity gets a canonical FD from the fDS, it is free to use its local knowledge of topology to transform the canonical route into a more efficient FD that does not take the traffic "up" to the globally-known core domain and "down" again.

4.2 FD Maintenance in M-FARA

M-FARA is intended to support mobility with dynamics that cannot be practically obtained by simply updating the fDS. M-FARA implements this part of FD management with **M-agents** (mobility agents). M-agents act as rendezvous points and as third parties to update FDs to handle mobility.

- Each mobile server entity has an associated M-agent (which is assumed to be non-mobile, with a static FD) with which the server registers itself at startup. The FD of an M-agent in turn may be registered in the fDS in place of the FD of a target entity, to allow clients to find a mobile entity.
- The entity informs its M-agent whenever the entity moves (changes its FD), so the M-agent can keep track of the entity's location.
- The entity sends packets carrying updated reply FDs to the remote entities for which it has associations. In this way, it tells remote entities where it is as it moves.

[6]This strong assumption in M-FARA represents an important and useful case, although the FARA model admits of more generality.

[7]The resulting routes have been called *triangular routes*.

When it receives a packet intended for a entity in its care, an M-agent may either (1) rewrite the packet's destination FD to point to the entity and then re-launch the packet, or (2) send a redirect message to the source entity specifying the new FD. Case (1) supports anonymous one-way communication, while case (2) is more efficient. In case (2), redirection can be triggered by the initial packet that creates the association, so that subsequent packets will bypass the M-agent and go directly to the destination entity. If the connection "breaks", the source entity can obtain the latest FD from the M-agent and perform the re-authentication sequence. In the worst case the initial rendezvous may have to be repeated, e.g., if an M-agent crashes with loss of state.

This brief discussion necessarily leaves many issues unspecified, such as how the entity knows that it has moved, what happens if both ends are moving (both might then use M-agents), and so on. Many of these issues are discussed in detail in [9].

In addition to M-agents, M-FARA requires a set of FD management functions for building, analyzing, and updating FDs. For example, the entity needs downcalls to compose complete FDs from canonical FDs, for analyzing and dissecting FDs, and for computing the entity's own canonical FD. There are also upcalls to the entity to pass a new FD when either its own FD or a remote FD has changed, for example. See [9] for details.

4.3 Associations in M-FARA

M-FARA supports four kinds of association: simple, connected, mobile, and reliable.

- A *simple association* has functionality analogous to UDP: unordered, unreliable, and unauthenticated messages.
- A *connected association* is also unreliable and unordered, but it includes a handshake to establish and destroy the association, to support authorization.
- A *mobile association* builds on the connected association to provide transparent mobility, by performing authentication and resynchronization after every move or in general whenever the status of the remote entity is uncertain.
- A *reliable association* builds on a mobile association by including reliability and ordering, analogous to TCP.

4.4 Source Verification in M-FARA

M-FARA adopts a relatively weak form of security, which provides authentication during the initial packet exchange that establishes the association and re-authentication after mobility or any other event that calls into question the remote state, but which does not verify every packet. It uses an authentication protocol based on that defined for DCCP (Datagram Congestion Control Protocol) [6].

Basically, a pair of tokens exchanged during association creation serve as credentials. Re-authentication is accomplished by presenting an authentication *challenge* to the remote entity, containing some function of the two credentials. The remote entity verifies the function and returns a *challenge response* containing its credential.

4.5 M-FARA Prototype

Element	Implementation
Entity, M-Agent	Unix process
FARA Kernel	Unix process
Association	Datastructure
FD	Multiple forms (see text)
Rendezvous	Association establishment protocol
Rendezvous String	String carrying protocol cookies
Authentication	Original DCCP: XOR
FARA Directory Service	(Not implemented)

Table 1: Prototype Implementation of M-FARA

An M-FARA prototype was built as a platform for testing ideas and for verifying that M-FARA does indeed lead to enhanced flexibility and extensibility of the network. Table 1 lists the M-FARA components and their manifestations in the implementation.

The prototype is written in C++, using Unix processes for entities and communicating via Internet overlays with UDP encapsulation. For ease of implementation, the low-level end-system functions such as delivery to a slot are also implemented in user space as Unix processes; this is known as the FARA "kernel" or fKernel. The entity/fKernel "system call" interface (the slot interface) is implemented using Unix IPC. As a result, a simulated M-FARA end system is represented by an fKernel process and zero or more entity processes. An fDS entity was not implemented, but it would be also be represented by a Unix process.

The prototype supports the M-FARA association types identified in Section 4.3. It supports traditional hop-by-hop forwarding and source routing using IPv4 and IPv6 addresses. Specifically, an FD takes the form:

$$(HopFD1, HopFD2, ..., SlotID),$$

where each $HopFD$ contains an IPv4 or IPv6 address for crossing a domain, and SlotID is used by the fKernel in the target system to deliver to the destination entity.

The prototype implements a reliable association using a trivial subset of TCP protocol, using the header format of TCP and providing reliable delivery but limited to a one-byte window. Each application entity executes some fixed application code. The only applications currently implemented are simple ping and a reliable byte-stream data echo.

Figure 2 shows a typical experimental setup for the prototype. Here there is a non-mobile entity communicating with a mobile entity. There is also a router connecting IPv4 and IPv6 domains. The entities set up a reliable association, and this association persists and is re-authenticated when the mobile entity moves from one domain to the other. The M-agent could be in either domain (although it was not mobile.) In practice, mobility has been simulated by multi-homing the mobile entity into domains and configuring its logical interfaces up and down.

Many more details of the prototype implementation of M-FARA can be found in [9].

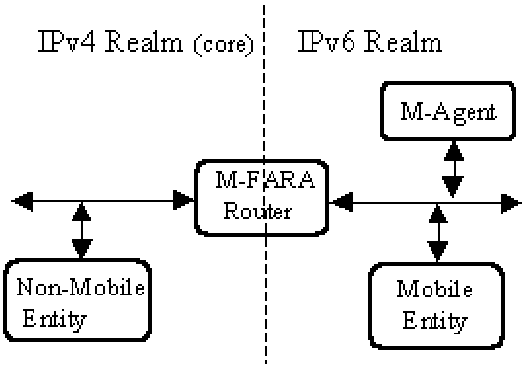

Figure 2: M-FARA Prototype: Experimental Setup

5. PRIOR WORK

In 1982, Jerry Saltzer [11] borrowed the concept of binding from operating system design and applied it to computer networks. He distinguished four kinds of objects that could be named: (1) services/users, (2) nodes, (3) network attachment points, and (4) paths for packet delivery. He related these in turn to the name/address/route distinction of Shoch [12]. He also defined three bindings, mapping (1) to (2), (2) to (3), and (3) to (4).

The present paper reconsiders these relationships. FARA's association and entity correspond very crudely to Saltzer's service/user and node, but the alignment is far from perfect. A FARA entity could be considered to be an abstraction of a service in a node, although Saltzer did not make the distinction between entities and associations (think: process vs. socket). "Below the line", FARA's FD defines Saltzer's delivery path. The FARA model always requires the definition of a path, while it allows but does not require attachment point naming. The path effectively specifies the attachment point without any explicit name. Finally, FARA attempts to avoid requiring a global namespace for service/user objects, envisioning instead a variety of alternative mechanisms, some global and some local, to find the FD for the desired virtual attachment point (slot).

The seminal 1994 effort PIP (Paul's IP Protocol) [2] introduced the forwarding directive concept and, in effect, separated location from identity. A later paper on IPNL (IP Next Layer) [3] extended the FD concept to create a NAT-based solution to the IP address depletion problem.

TRAP (Trivial Routing Architecture Proposal) [1] introduced a single identifier for a port and a network address; packets sent to a TRAP transport layer are delivered to the application. This is similar to FARA's concept of FDs delivering packets all the way to the entity.

The TRIAD architecture [4] introduced a notion called Wide-area Relay Addressing Protocol (WRAP), in which packets contain an ordered list of opaque tokens, or routing hints, forming a source route. This is similar to the forwarding directive function in FARA.

Finally, the HIP (Host Identity Payload)[8] provides a security architecture that is very relevant to FARA. HIP uses a cryptographic key to represent the identity of an end system (not an entity, as in FARA). A hash of this identifier is carried in a shim between the IP and transport layers. Transport and application layers are expected to bind to the hash of the cryptographic identity rather than to the IP address of the end system. HIP thus provides a separation of address and identity. Although HIP was originally designed as an extension of the current Internet architecture, its mechanisms are a candidate for association setup under FARA.

6. CONCLUSIONS

Although the basic concepts of FARA are not new, the reasoned assembly of these concepts into a general model is believed to be new. FARA should provide a framework for proposing and understanding a range of alternative protocol architectures. he FARA model re-modularizes the architecture into two conceptual levels: the abstractions of entities and associations at the upper level and a distinct communication substrate for connectionless packet forwarding in the lower level. The intent is to cleanly separate location from identity, providing support for general mobility as well as independent evolution of mechanisms at the two levels. FARA envisions a clearly articulated "red line", with a corresponding API, between the two levels.

FARA is also an exercise in abstract reasoning about architecture, and part of its value is what it can teach us about the logical structure of protocol architectures. Its assumptions were deliberately pushed to the limit of generality. Indeed, its assumptions (Section 2.2)are mostly negative, specifying what FARA does *not* define. Further development of FARA instantiations will be needed before we can make a judgment on whether the present FARA assumptions are too aggressive to create a practical instantiation.

FARA is a work in progress. The FARA model appears to be self-consistent and M-FARA shows that there exists at least one instantiation of FARA that provides significant functionality beyond that of the current Internet architecture. However, additional work is needed to further explore M-FARA as well as other possible and perhaps useful instantiations of the FARA model.

7. ACKNOWLEDGMENTS

The work reported in this paper was performed under the DARPA-funded New-Generation Internet Architecture (NewArch) project. We are very grateful to the other members of this project, especially Noel Chiappa, Ted Faber, Mark Handley, Karen Sollins, and John Wroclawski, for their creative ideas and patient criticism as FARA and M-FARA evolved.

8. REFERENCES

[1] V. Antonov. Trivial Routing Architecture Proposal (TRAP). IETF Work in Progress, September 1995. `http://gato.kotovnik.com/~avg/old_page/papers.html`.

[2] P. Francis. Pip Near-term Architecture. Internet Request for Comments RFC 1621, May 1994.

[3] P. Francis and R. Gammadi. IPNL: A NAT-extended Internet Architecture. *Proc. ACM SIGCOMM 2001*, pp 69-80, 2001.

[4] M. Gritter and D. Cheriton. An Architecture for Content Routing Support in the Internet. *Proc. USENIX USITS*, March 2001.

[5] S. Kent and R. Atkinson, Security Architecture for the Internet Protocol. Internet Request for Comments RFC 2401, November 1998.

[6] E. Kohler, M. Handley, and S. Shenker. Datagram Congestion Control Protocol (DCCP). IETF Work in Progress, March 2003.

[7] E. Lear and R. Droms. What's In A Name: Thoughts from the NSRG. IRTF Name Space Research Group report, Work in Progress, December 2002.

[8] R. Moskowitz. Host Identity Payload. IETF Work in Progress, February 2001.

[9] V. Pingali, A. Falk, T. Faber, and R. Braden. M-FARA Prototype Design Document. USC Information Sciences Institute, In preparation, 2003.

[10] J. Postel Internetwork Protocol. RFC 791, September 1981.

[11] J. Saltzer. On the Naming and Binding of Network Destinations. In Local Computer Networks, North-Holland Publishing Company, Amsterdam, 1982, pp. 311-317. Reprinted as RFC 1398, August 1993.

[12] J. Shoch. Inter-Network Naming, Addressing, and Routing. *Proc. 17th IEEE Conf. Comp. Comm. Networks*, pp 72-79, September 1978.

[13] A. Snoeren and H. Balikrishnan. An End-to-End Approach to Host Mobility. *Proc. 6th Int. Conf. Mobile Comp.*, 2000.

[14] I. Stoica, D. Adkins, S. Zuhang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. *Proc. ACM SIGCOMM 2002*, pp 73-86, August 2002.