Design and Implementation of a Framework for Persistent Identification and Communication in Emerging Networks

Joud S. Khoury[*] jkhoury@ece.unm.edu

Henry Jerez[†] hjerez@cnri.reston.va.us

Luca De Cicco[‡] ldecicco@gmail.com

ABSTRACT

The Internet Protocol (IP) is currently used to provide inter-networking among heterogeneous access networks. However, the evolution of and the innovation within these networks is greatly hindered by the geographical and topological rigidness of the protocol implementation that hinders the support for flexible unstructured communication paradigms. To broaden the user's innovation space and to efficiently embrace the characteristics of these emerging unstructured networks, clean-slate architectural approaches are being pursued. In this paper, we present the Persistent Identification and NeTworking research framework (PINT); an implementation of the Transient Network Architecture (TNA) currently being developed between the University of New Mexico and the Corporation for National Research Initiatives. PINT provides the research community with a modular and extensible set of networking components and primitives that enable novel research and experimentation atop a persistently identified networking platform. This technology provides a ground for inter-networking of heterogeneous communication networks where novel networking primitives are exposed through the Persistent Identification and Networking Layer (PINL), allowing mobile and stationary entities to communicate securely based on persistent identifiers that are location independent.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.6 [**Computer-Communication Networks**]: Internetworking; C.5 [**Computer System Implementation**]

General Terms

Design, Experimentation

Keywords

Persistent identification, mobility, testbed

1. INTRODUCTION

The emergence of key wireless technologies, the proliferation of mobile devices, and the nomadic user and computing lifestyles have redefined the basic characteristics of the internet. Wireless mesh networks (WMNs), wireless sensor networks (WSNs), mobile ad-hoc networks (MANETs), and vehicular area networks (VANs) are examples of self-organizing unstructured networks that have local communication paradigms and are optimized to perform under particular physical constraints. Traditionally the Internet Protocol (IP) has been used to provide inter-networking among heterogeneous access networks. IP unifies the underlying forwarding mechanisms and the routing identifiers providing end-to-end tranparency. In other words, the whole intermediate network appears to be homogeneous with a well-defined topology as far as the endpoint is concerned. This abstraction based on the original Internet design has been very successful and scalable so far. However, with the emergence of more diverse heterogeneous access technologies, and with the continuous adoption of wireless communication, maintaining the end-to-end IP abstraction is becoming harder resulting in more strain on the evolution of the network. Additionally, unifying the addressing scheme (IP address) has led to inefficiencies within emerging networks. Such networks must support IP addressing with the added administration requirements despite the fact that the topological IP address has little physical significance as a routing directive within these networks. Part of our recent work [11, 12, 10] has demonstrated that a persistent identifier can be used as a routing identifier (forwarding directive) within a local mesh network that implements a multi-hop routing protocol, hence replacing IP. Additionally, work by Kim et al. [13] shows that ethernet bridging can be made scalable and efficient enough to route based on MAC addresses within an enterprise network eliminating the need for internal IP subnetting and administration. Again, in this scenario, IP is only useful for external reachability and application interoperability. Furthermore, even when IP is implemented within such networks, the majority of communications between the endpoints requires a high level naming system and an indirection mechanism, whether hierarchical (eg. DNS) or flat

[*]University of New Mexico, USA.

[†]Corporation for National Research Initiatives, Reston VA, USA.

[‡]Politecnico di Bari, Bari, Italy.

(eg. DHT [22]), which endpoints can use.

Several research test-beds have been recently proposed to enable experimentation with next generation networks, coexistence of heterogeneous systems, mobile networking, and wireless environments [21, 6, 1, 23]. This paper revisits TNA concepts [10, 5] that have been implemented in the Persistent Identification and Networking research framework (PINT) and its research test-bed deployment at the University of New Mexico and the Corporation for National Research Initiatives. PINT may either coexist as a deployment on top of readily available test-beds to provide the identification framework, or it may be deployed into a separate test-bed for scoped research with persistent identification. Briefly, PINT exposes to the research community a modular and extensible set of networking components and primitives, which enables novel research and experimentation atop a persistent identification and networking framework along the lines of the TNA architecture. The framework is designed to support the following key concepts of the architecture:

- Intrinsic support for unstructured networks;
- persistent identification and certification of network entities;
- distributed control-plane fucntionality provisioning using mobile agents; and
- seamless mobility.

The framework components include 1) entities that represent the communicating endpoints, 2) areas of influence that abstract sets of entities sharing a common communication protocol, 3) a virtualization model for agent based provisioning of control-plane functionality, and 3) a network substrate virtualization. Novel networking primitives are exposed through the Persistent Identification and Networking Layer (PINL), allowing mobile and stationary entities to communicate securely based on persistent identifiers that are location independent. The paper presents the modular, extensible, and portable implementation of components and primitives within PINT. It then discusses our experiences with the framework so far, based on a first deployment on wireless mesh and traditional ethernet networks.

The remainder of this paper is structured as follows. Section 2 discusses the principal TNA concepts that guided the development of PINT. Section 3 describes the PINT framework and test-bed, and the reseach opportunities enabled thereof. A deployment over mesh neworks is then illustrated in section 4. Section 5 overviews our current and future work and concludes.

2. TRANSIENT NETWORK ARCHITECTURE

PINT is an instantiation of the Transient Network Architecture (TNA) [10]. This Architecture re-conceptualizes the Internet as a set of persistently identified nodes that can self-organize into networks that automatically interconnect with each other as needed and are capable of providing end to end communication based on those persistent identifiers. TNA builds on the original logical model of the Internet to form a logical network that allows the effective merging of heterogeneous networks without forcing them to modify their communication protocol but rather their logical coordination mechanism. The architecture that embraces mobility, security, and identity persistence by design allows any or all of the TNA nodes and self-organized networks, to be in motion or disconnected at any point in time, while still maintaining basic secure global connectivity and functionality. In this section, we layout the principal TNA design guidelines that guided the development of PINT.

2.1 Area of Influence - AoI

The self-organized basic networks in TNA are called Areas of Influence (AoI), i.e. the AoI captures the scope of "local" communications. Briefly, an AoI is a local communication community that defines its own communication protocols and network architecture implementation. Examples of these implementations include, but are not limited to, LANs, Cellular networks, MANETs, sensor nets, and mesh networks. These networks implement their own communication mechanisms and protocols and can survive independently of the global system. A sketch of how currently available networks can fit into the AoI framework, is shown in Figure 1. The figure shows how the nodes of a mesh network, for instance, may assemble into an AoI. The AoIs themselves define their own local communication implementation such as Ethernet, RF or Bluetooth, and even their own local identification mechanisms. The basic constituents of AoIs are network entities which we formally define next.

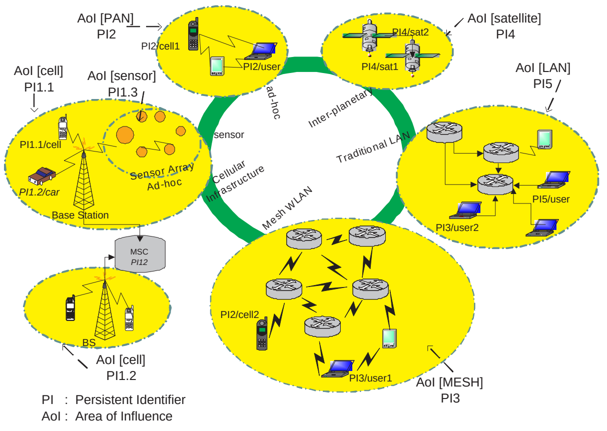


Figure 1: Examples of different Areas of Influence that form TNA

2.2 Entities and Communication

Based on the definition in [7], an *entity* is the end-point of communication. It is an abstract construct that can represent different network elements including, but not limited to, a process, a thread, a device, a cluster of devices, or a service. The entity is the smallest element on the network that can be mobile. In TNA each entity has its own Persistent Identifier (PI) that is globally unique, and secure by design. Security, as we shall see later, results from the direct association of the PI with a set of credentials that can be challenged by the network at any point in time.

This approach is different form the traditional internet and actually improves it. Traditionally, the Internet and particularly IP has taken a *location-oriented* paradigm to identifying entities, i.e. the most basic entity identifier expressed as a tuple {*IP address*, *port number*} is directly dependent on the topological IP address. So far, the IP address has performed well as a location identifier since it inherently embeds topological information and thus fosters routing scalability. However, when mobility is introduced as in the case of wireless networks, IP looses any meaning of identity reference and degenerates into a pure routing identifier. Coupling the entity identifier to its topological location hinders mobility and poorly identifies the actual entity, which should exist independent of its network location or state. Several proposals have focused on solving the mobility problem by decoupling the host identity from the attachment point [16, 25, 24, 18]. Most of these proposals share an overlay approach on top of IP whereby a high level address is translated to an IP address by early binding and routing is an end-to-end, IP-based mechanism. This translates into *the current Internet architecture design making it inefficient to initiate communication with an arbitrary entity on the current Internet*, unless that entity has a public IPv4 (or an IPv6) address. Several architectures have been proposed to solve the Internet addressing issue as in [17, 26, 20]. However, in these approaches and even when a public address is available, inefficient mobility management schemes prevail requiring centralized infrastructure and continuous end-to-end negotiations between the endpoints over a simple "core".

TNA's *entity-oriented* approach to identification and communication recognizes the entity as a first class network component and unlike the current Internet approach, associates it directly with a globally unique PI that is independent of any topological information. We do so by creating an underlying network engineered to seamlessly and securely incorporate those entities and their persistent identifiers as the global logical interconnection mechanism.

2.3 Persistent Identification

Persistence and global uniqueness are two attractive characteristics of the PI. Persistence of the identifier is essential when the attributes (e.g., state and location information) of the identified entity change, but the identifier itself persists. Global uniqueness is necessary to avoid identifier conflicts especially when the identified entity is highly mobile. Traditionally, achieving this has required significant centralization; that is why TNA introduces a new distributed certification and resolution approach.

2.3.1 *Certification and Resolution*

An identifier is used by the entity for interaction with the rest of the system provided the identifier can be challenged and certified within the environment of communication whenever necessary. The architecture calls for three certification realms, as follows:

- *Instance (Red Realm)* is defined relative to a particular certifying entity that is frequently certified by other entities itself. It represents the authoritative domain of the entity that certifies and/or identifies a set of other entities.

- *Local (Yellow Realm)* is defined relative to the local network, AoI. This realm represents the authoritative domain of an AoI that represents also a certification and resolution network for the different entities that compose it.

- *Global (Green Realm)* represents the collection of globally trusted nodes responsible for certifying yellow and red realms. This realm that could be composed of nodes at any level has to simultaneously guarantee global certification and scalability. Note that at this level, many globally trusted authorities can co-exist and inter-operate avoiding the pitfalls of a single trust system as is the case with the current Internet.

The colors of the realms indicate the level of trust within the system. This way, certification by the *Green Realm* represents the highest level of trust with respect to the overall system. Certification, or the flow of trust, is "top-down", from *Green* to *Yellow* to *Red Realms.* Hence, an identifier certified by the *Green Realm* is globally trusted, while an identifier certified by the *Yellow Realm* may only be used for secure interactions within the AoI.

The PI is resolved into information useful for the interaction between the communicating entities such as an AoI PI for routing or and actual hardware address for final transmission. The result of the PI resolution implies certification by the answering realm. Resolution is performed in a "bottom-up" fashion: First, try to resolve against the *Red Realm.* A failure here will percolate the resolution one level up against the *Yellow Realm* and then against the *Green Realm.* The mechanisms for certification and resolution are closely coupled and their details will depend on the particular architecture implementation.

2.4 Distributed control-plane functionality provisioning using the Ghost/Shell model

TNA defines two new abstractions, as follows:

Ghost or Generic Host is a persistently identified abstraction of an entity that in the case of this testbed is a service that provides control-plane functionality.

Shell is a virtualization environment that represents the abstraction of the platform/infrastructure over which the Ghosts execute.

The combination of these technologies enables the migration of network functionality and services across different platforms while maintaining referential integrity and connectivity due to the use of location independent identifiers. This adds a great deal of flexibility and resiliency to the network and allows network managers and users to approach it as a logical abstraction, where each node is characterized by its persistent identifier and its core functionality abstracted into a Ghost. Our implementation uses the concept of mobile agents in distributed systems [8] to instantiate Ghosts. This improves network utilization and reduces human intervention [9]. Ghosts can execute custom business logic, move freely across the network, terminate or spawn new agents using the resources provided by the architecture.

3. PINT FRAMEWORK

Before delving into the details of the PINT implementation, we summarize the key features of our framework:

- *Intrinsic support for unstructured networks*: the framework is designed with emerging networks in mind, especially wireless environments. WMNs, WSNs, MANets, VANs, and traditional structured networks should all be able to participate and seamlessly inter-network, while respecting each of the networks' local communication paradigm and protocol implementation.

- *Persistent identification and certification of network entities*: The advantages of using the PI as the network address are several, including:

  – Mobility: The independence of the PI from its attributes is an attractive property for a network layer identifier. The direct advantage of persistence is mobility since an entity that is persistently addressed by the network layer is reachable on that address at all times. Consequently, mobility occurs natively eliminating the network layer indirection introduced by other proposals [16, 25, 24, 18]. In other words, the indirection from a persistent name to a forwarding address (e.g. DNS name [15] to IP address) is eliminated in our framework, since the PI is itself the forwarding address.

  – Security: The PI address is stamped i.e. it is inherently associated with security information (e.g. public/private keys) which can be used at all times by the communicating parties (and the network if necessary) for accountability, identifier authentication, and confidentiality.

  PINT allows the experimentation with different persistent identification technologies. The framework is oblivious of the particular semantics of the PI, or the mechanisms attached to it including authentication, resolution, and registration. Consequently, several current technologies can be experimented with. For example, the PI implementation might be hierarchical as in the case of the current Handle System [4] and the Domain Name System (DNS), or flat as in the case of hashes whether self-certifying (e.g HIP [16]) or not (e.g. Chord, Pastry).

- *A novel approach to dynamic and extensible network control-plane service provisioning* using mobile agents; Routing as well as identification are essential network services that provide control plane functionality. The abstraction of each such service is what we have previosuly referred to as a Ghost (section 2.4). Within the PINT framework, Ghosts are implemented as mobile agents, of which we isolate the following:

  – Identification Ghost: This agent is particularly disseminated into the network with the goal of implementing the identification service for the AoI. Managing the namespace including creating, removing, and updating persistent identifiers within the AoI are operations of the identification service which this Ghost implements. The entities within the AoI are oblivious of the actual implementation specifics of the identification service. For example, upgrading the identification service model from a centralized system to a P2P system requires simply upgrading the identification Ghosts within the the AoIs and the upgrade is transparent to the AoI entities. The same is true with the routing Ghost.

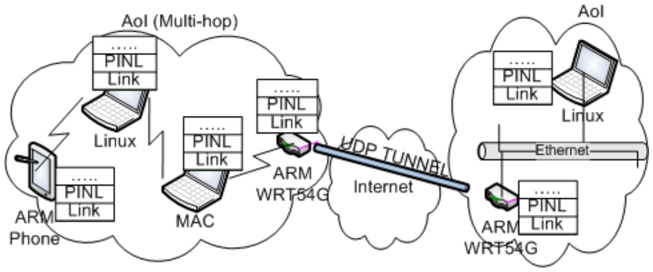


Figure 2: PINT components and primitives in sample test-bed showing a multi-hop ad-hoc AoI connected virtually to a traditional ethernet AoI. The PINL layer running on all nodes is able to deliver packets to persistently identified entities.

  – Routing Ghost: It is similar to the identification Ghost except for its functionality. The routing Ghost implements the actual PI routing protocol that delivers packets to their correct destination(s).

  Ghosts[1] may register for providing a discovery service that allows for their automatic discovery by other entities within the network. Note that the Ghosts do not represent infrastructural components within the AoI, but instead provide dynamic on-the-fly services for the rest of the entities in the AoI. For example, in an emergency (first responder) network, we envision a set of nodes rapidly forming into an AoI with the necessary Ghosts automatically initializing the AoI and relocating to optimize the network utility. The routing Ghost, for instance, locates a node with Internet connectivity bridging the emergency network to the Internet. Optimizing the placement of Ghosts for maximum network utility is a topic we re investigating in parallel [19].

- *Seamless entity mobility*: directly results from the network being PI-aware. Entities, whether devices, services, or processes can relocate and re-bind while still being reachable on their PI. PINT is generic enough to allow the deployment and experimentation with various mobility mangement schemes.

3.1 Components and networking primitives

Figure 2 shows the basic components within the PINT framework. First, a set of nodes is abstracted into an AoI. AoIs are allowed to inter-connect either through dedicated links, or through virtual UDP tunnels that abstract the Internet link. Nodes implement the Persistent Identification and Network Layer (PINL) as part of a modified networking stack. Entities attach to the network through PINL, either directly, or through transport layers that can add reliability and/or security to the end-to-end communication. We start by describing the entity identification assumptions and continue to discuss the details of the PINL layer and the primitives and interfaces it exposes to upper layers.

[1]Note here that the Ghost is a logical entity, and it might be that both the identification and the routing Ghosts are implemented as one physical entity.

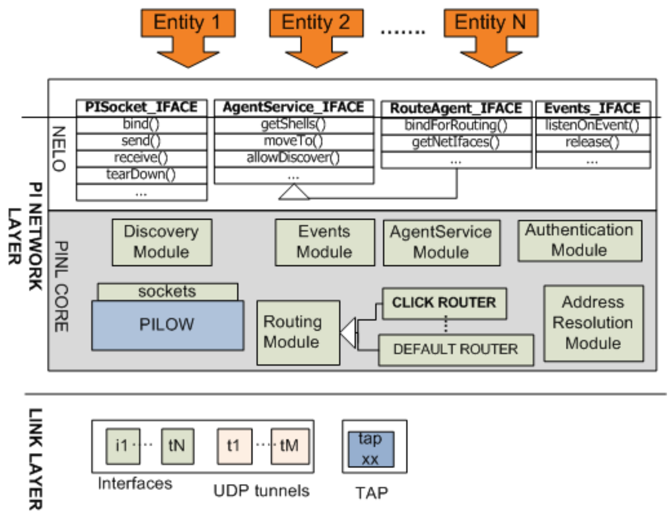


Figure 3: PINL layer building blocks

3.1.1 *Entity identification*

With the proliferation of mobile devices and the anticipated large scale of the network, comes the challenge of how to design a system that is capable of identifying individual entities at a large scale. PINT makes some assumptions in this regard in order to organize entities within the system. First, in order to participate in the system, an entity must acquire a stamped PI, i.e., a PI associated with a stamp. The latter is a credential acquired from a certification authority (CA) to authenticate the owner(s) of the PI. Second, and for scalability reasons, we allow the aggregation of a set of processes into a single entity by assigning a different *type* to each. An aggregated process set appears as a single entity with respect to the rest of the network. Hence, in the case that one of the processes intends to become mobile (for example to migrate), that process must obtain a valid globally unique PI that identifies the process itself. Our *type* analogy is similar to the application port number in current TCP/IP stack and is useful for local demultiplexing.

3.1.2 *Persistent Identification and Networking Layer*

PINL provides the necessary network services to foster the evolution of the network. Services and protocols belonging to this layer mainly handle the initialization of entities within the AoI, and packet delivery between persistently identified entities that may be challenged and authenticated based on their PIs. Presented with a PI, this layer is intelligent enough to deliver a packet to its destination(s). Reliable and/or secure delivery mechanisms are part of a separate upper layer, and motivate an interesting future research effort.

Figure 3 shows the componentized architecture of PINL. The PINL layer includes a set of modules that implement the basic layer services, and exposes an extensible neutralizing interface, which we refer to as the NELO inerface[2], to the upper layers. Entities may directly interface with PINL through the NELO interface. The details of the modules and the interface follow:

[2]NELO stands for Neutral Environment Language for Operation.

- PILOW: The main responbilities of PILOW is switching incoming requests between modules and maintaining layer state such as PI tables and ARP tables. Additionally, it implements the RouterEngine interface which is employed to route packets to their destination. We ship a default implementation of the RoutingEngine interface in order to provide basic routing functionalities, i.e. routing packets whithin the AoI. The default routing algorithm may be overriden at runtime when a new entity (a routing Ghost) assumes routing responsibilities through the use of the NELO interface. A more complex router can thus be implemented on-the-fly as we shall see later in the discussion. PILOW additionally implements the PI_Socket_IFACE which provides the traditional socket primitives to entities based on a connectionless transport mechnism that simply demultiplexes incoming packets to resident entities. For example, to use this interface, an entity implements the following code:

```
socket.bind(pi, type);
//bind entity with PI 'pi' to a socket
...
socket.send(pi_packet);
//sends a PIPacket out
...
while(true){
PIPacket pi_packet = socket.receive();
//listen for incoming packets
}
```

- AgentService module: provides service to Ghosts in general. Since Ghosts are abstracted as entities, the Ghost must bind to the layer and authenticate itself before executing. This module implements the AgentService_IFACE, which might be extended to add particular agent functionality. The RouteAgent_IFACE for example extends AgentService_IFACE, introducing functionality specific to routing agents. Through this interface, for example, a routing Ghost can securely bind to PINL overriding the routing service.

- Discovery module: provides a discovery service to Ghosts and entities in general. An agent may invoke *allowDiscover()* on the AgentService_IFACE to enable external entities to discover it. Upon invocation of the *allowDiscover()* function, the discovery module will answer discovery requests destined to the registering agent entity. For example, as we shall see section 4, a routing agent within an AoI may assume the role of routing beyond the AoI, hence acting as a default gateway that can be discovered by all the AoI entities.

- Routing module: accepts packets from PILOW for routing, based on PI. Within our framework, the routing service can be easily extended to support various routing implementations. The actual router implementation is determined at runtime for extensibility. A simple device will normally utilize a default router, while a gateway will need a more complex router implementation (e.g. Click router [14]). PILOW is oblivious of the router type and will forward packets to whatever router currently active on the node. The flexibility of this implementation is better explained by

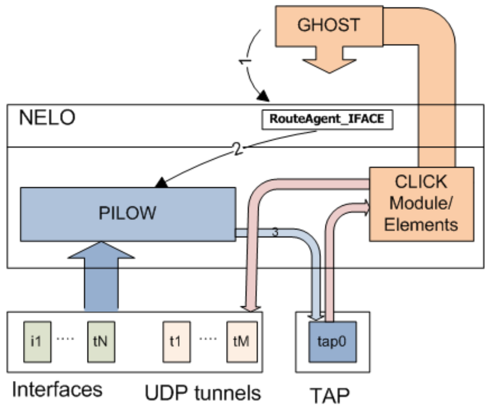


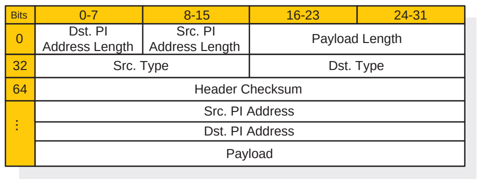
Figure 4: Click router asks for agent binding from NELO interface.

introducing a simple example in which a Click router asks PINL to replace the default routing algorithm at run time. Figure 4 shows how a Click routing Ghost is able to override the default gateway implementation: 1) the Click entity asks for router binding issuing *RouteAgent_IFACE.bindForRouting* primitive; 2) the PINL AgentService module (see Figure 3) will then authenticate the Ghost; 3) when authenticated, PILOW sets up a `tap` interface through which the PINL daemon and the Click entity communicate.

- Authentication module: The authentication module defines the primitives that allow identifier authentication mechanisms, validates certificates, signatures, etc. The complexity of this module will eventually depend on the actual PI implementation technology.
- Events module: enables entities or upper layers to listen on network events through an extensible Events_IFACE. The module takes care of propagating registered event callbacks to upper layers (transport protocols or entities),

3.1.3 Protocols

1. Simple Persistent Identification Protocol (SPIP): is the basic networking protocol used for communication. The format of the SPIP packet is illustrated in Figure 5. This is the most basic unit of communication that all entities within the PINT framework currently use to communicate. The source and destination PI ad-



Bits	0-7	8-15	16-23	24-31
0	Dst. PI Address Length	Src. PI Address Length	Payload Length	
32	Src. Type		Dst. Type	
64	Header Checksum			
⋮	Src. PI Address			
	Dst. PI Address			
	Payload			

Figure 5: PI packet format.

dresses are variable length with a max size of *32* bytes. We allow a variable size identifier to support different implementations of the PI, such as a string (a *handle* in the Handle System implemetation [4]) or a hash (HIP [16])[3]. In order to send a packet, the sender entity addresses the packets to the 2-tuple $\{PI,\ type\}$ identifier of the destination entity.

Regarding the ARP and the Discovery protocols, we have simply extended current implementations of those and introduced a PI ethernet frame type specific to our implementation.

3.2 Implementation Details

The implementation of the Persistent Identification and Networking Layer (PINL) consists of a daemon running at the user-space, and a client library which is used by the entities in order to utilize the functionality provided by the daemon. The layer code is written in object oriented C by exploiting the facilities provided by the portable GLib library [3]. Additionally, a java JNI interface is provided for the NELO to allow Java entities to communicate with PINL.

The C code has been designed with portability in mind, and at the same time targetting embedded devices with very limited RAM and CPU resources like mobile phones, PDAs and routers. In this sense, we have designed the code base to depend only on highly portable libraries such as libpcap, libglib and libgnet which are known to run on Unixes, Windows, Mac OS X and on different architectures like x86, ARM, Mipsel, and SPARC. The PINL has been compiled and tested on the following platforms and devices without the need for patching the code base: Linux (Debian/Ubuntu), Mac OS X, neo1973 using openmoko, n770/n800 using maemo platform, and on the router WRT54G using the openwrt distribution.

The daemon is the fundamental building block of PINL, implementing all the functionality that we have described so far in the previous sections and exposing the NELO interface to the entities. In particular, we have used libpcap to bypass the IP layemr both when sending and receiving PI packets; libpcap listens on all the interfaces that are configured and captures all inbound frames that are either PI packets, Discovery packets or ARP packets by inspecting the MAC type field within a frame. The captured frame is then received by the PILOW module by issuing a callback function (see Figure 3) that, based on the MAC type, forwards the frame payload to the right module. When a module needs to send a packet on the network it will use the inject feature which is available using the pcap library. Using pcap as an interface to the link layer is very convenient as we are able to send and receive frames bypassing the IP layer and providing a service that is oblivious of the underlying link layer.

In order to be able to communicate with the PINL daemon the entities must link their code to the *pientity* library. The latter internally utlizes sockets to implement inter process communication. The library exports a very simple API through which the entities can communicate in a transparent way with the NELO interface.

3.3 Research Impact

PINT provides a research framework and test-bed for emerging networks such as wireless mesh networks, wireless sensor

[3]Despite the expensive header size, we have deliberately chosen a 32 byte max PI size to allow experimantation with various PI technologies. WSNs, for example, are expected to utilize a significantly smaller PI size.

networks, MANets, as well as traditional networks to interconnect and communicate beyond the limitations of the traditional Internet Protocol (IP), which was not designed for wireless and mobile environments. Those networks can experiment with a novel persistent identification framework locally and globally, exploiting the novel identification and networking primitives. PINT may either coexist as a deployment on top of the readily available test-beds such as ORBIT [21] to provide the identification framework, or it may be deployed into a separate test-bed for scoped research with persistent identification.

We are currently pursuing several interesting research topics that are based on the TNA architecture. Some of the prominent topics involve:

- Inter-AoI routing implementaions based on PIs; routing based on PIs is a critical research challenge and is essential for our framework to function properly;
- Transport protocols that provide reliability and/or security; currently, as part of the PINL implementation, we provide a simple connectionless transport protocol that demultiplexes incoming packets based on PI and PI-type combinations. We envision different transport protocols emerging on top of PINL, which can add reliability, congestion control, and security to communication while respecting the wireless and mobile nature of the communication;
- Efficient mobility management schemes.

PINT provides the framenwork to experiment with the feasibility, efficiency, and scalability of possible solutions to the above topics. A preliminary deployment of mesh/ethernet AoIs is discussed next.

4. MESH/ETHERNET DEPLOYMENT

In this section, we describe a mesh/ethernet deployment over PINT at the ECE department building at the university of new mexico. The goal of the deployment is to validate the operation of the components and primitives rather than to measure their performance. Performance measurements will directly depend on the inter-AoI routing mechanisms, PI technology, and mobility management schemes that we will end up adopting and this is part of our future work.

We setup two distinct multi-hop mesh networks with SSIDs *mesh1* and *mesh2*, repectively, and an ethernet network as shown in Figure 6. *mesh1* is comprised of 4 nodes dispersed across the first floor of the building, while *mesh2* is comprised of 4 nodes dispersed across the third floor, and the ethernet network is comprised of 3 nodes deployed in the second floor. As part of each network is a special WRT54G router node that runs the PINL layer. The three networks are connected through the routers with UDP tunnels that traverse the local IP network internal to the building. All the nodes run the PINL layer at the user level on top of an Ubuntu7.04 OS. Within the mesh networks, PINL attaches to the AWDS mesh link state routing protocol [2] which exports a virtual layer 2 interface ("awds0"). As to the nodes within the ethernet network, PINL attaches to layer 2 through interface "eth0". A sketch of the complete deployment is shown in Figure 6.

All nodes employ the default routing engine that ships with PINL, except for the WRT54G nodes that are running

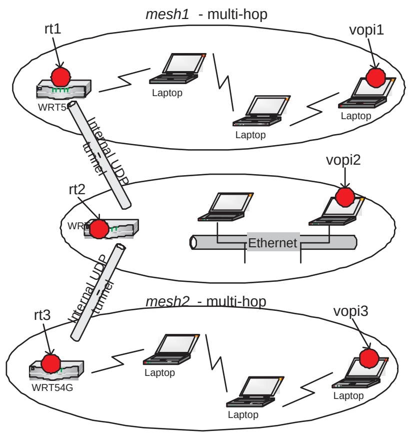


Figure 6: Mesh/Ethernet deployment of 3 AoIs (2 mesh and 1 ethernet network). Red circles represent entities.

a virtualized Click router to handle the inter-AoI PI based routing i.e. acting as a gateway. The Click entities are represented by *rt1*, *rt2*, and *rt3* in Figure 6 Internal AoI nodes use the discovery protocol to discover the gateway, and forward all traffic that is not local to the latter. To know whether a destination PI is local, the default routing engine employs an extended ARP mechnism for local resolution of PIs. Additionally, entities utilize the discovery protocol to proactivly announce their presence to the gateway, which in turn maintains soft state about the local network. Finally, and most importantly, the gateways implement a simple PI propagation protocol periodically exchanging their local state. This inter-AoI PI propagation mechanism is not scalable; however, it is just a proof of concept implementation that enables the gateways to locate external entities and hence, to route inter-AoI traffic correctly. As mentioned previously, part of our current research is targeted at examinig efficient and scalable PI propagation, caching, and replication mechanims.

Aside from the deployment, our experience with the PINL layer particularly shows that PINL is performing as good as UDP/IP over a local mesh network, and that both PINL and IP are constrained by the underlying physical and link layer characteristics.

5. CONCLUSIONS AND FUTURE WORK

This paper introduced the PINT framework an instantiation of TNA implemented and deployed at the university of New Mexico. PINT exposes to the research community a modular and extensible set of networking components and primitives, which enables novel research and experimentation atop a persistent identification and networking platform. The framework may either coexist as a deployment on top of readily available test-beds to provide a novel identification framework, or it may be deployed into a separate test-bed for scoped research with persistent identification and networking. PINT is still a work in progress and the implementation is constantly changing. Aside from investigating the identification, routing, and mobility mechanisms as discussed in section 3.3, we are currently enhancing our implementation to allow easy deployment of the framework within the ORBIT test-bed [21] and the bridging of external networks to the ORBIT deployment. Briefly, ORBIT is the Open Access Research test-bed for Next Generation Wireless Networks. It is a radio grid (20x20 APs) developed for scalable evaluation of next generation wireless network protocols. The grid allows multiple simultaneous experiments specified using scripts and uses virtualization of APs for that purpose. This is essential for broad participation of the research community especially with the recent bridging of ORBIT and PlanetLab [6].Finally,we intend to port the PINL code to kernel space by targetting first the Linux operating system once the API becomes stable.

6. ACKNOWLEDGEMENTS

The work reported in this paper is part of the Transient Network Architecture project funded by the National Science Foundation NSF under the Future Internet Design (FIND) Grants CNS-0626380, and CNS-0627067. The actual framework and more details are available at the project website http://www.tna.cnri.net.

7. ADDITIONAL AUTHORS

Additional authors from The University of New Mexico: Jorge Crichigno (`jcrichigno@ece.unm.edu`), Chaouki T. Abdallah (`chaouki@ece.unm.edu`), Wei W. Shu (`shu@ece.unm.edu`) and Gregory L. Heileman (`heileman@ece.unm.edu`) .

8. REFERENCES

[1] The WHYNET project, http://pcl.cs.ucla.edu/projects/whynet.

[2] Ad-hoc wireless distribution service - awds. http://awds.berlios.de/.

[3] Glib library. http://www.gtk.org/.

[4] The handle system. http://www.handle.net.

[5] The transient network architecture. http://hdl.handle.net/2118/tna.

[6] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: realistic and controlled network experimentation. In *Proceedings of SIGCOMM '06*, pages 3–14, New York, NY, USA, 2006. ACM Press.

[7] D. Clark, R. Braden, A. Falk, and V. Pingali. Fara: reorganizing the addressing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 313–321, New York, NY, USA, 2003. ACM Press.

[8] G. F. Coulouris and J. Dollimore. *Distributed systems: concepts and design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1988.

[9] T. C. Du, E. Y. Li, and A.-P. Chang. Mobile agents in distributed network management. *Commun. ACM*, 46(7):127–132, 2003.

[10] H. Jerez, J. Khoury, and C. Abdallah. A mobile transient network architecture. 2006. eprint arXiv:cs/0610100 available at http://arxiv.org/pdf/cs/0610100v1.

[11] J. Khoury, J. Crichigno, H. Jerez, C. Abdallah, W. Shu, and G. Heileman. The intermesh architecture (student demo). In *MobiCom 07*, Montreal, Canada, September 2007.

[12] J. Khoury, J. Crichigno, H. Jerez, C. Abdallah, W. Shu, and G. Heileman. The intermesh network architecture. Technical Report EECE-TR-07-007, University of New Mexico, April 2007. [online]: http://hdl.handle.net/1928/3052.

[13] C. Kim and J. Rexford. Revisiting ethernet: Plug-and-play made scalable and efficient. to appear in Proc. of IEEE LANMAN Workshop 2007.

[14] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, 2000.

[15] P. Mockapetris. RFC 1035: Domain names implementation and specification, November 1987.

[16] R. Moskowitz, P. Nikander, and P. Jokela. Host identity protocol architecture. RFC 4423, May 2006.

[17] T. S. E. Ng, I. Stoica, and H. Zhang. A waypoint service approach to connect heterogeneous internet address spaces. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 319–332, Berkeley, CA, USA, 2001. USENIX Association.

[18] C. E. Perkins. RFC 3220: Ip mobility support for ipv4, January 2002.

[19] J. Piovesan, C. Abdallah, H. Tanner, H. Jerez, and J. Khoury. Resource allocation for multi-agent problems in the design of future communication networks. Technical Report EECE-TR-07-001, University of New Mexico, April 2007. [online]: http://hdl.handle.net/1928/2973.

[20] P. F. Ramakrishna. Ipnl: A nat-extended internet architecture. In *Proceedings of SIGCOMM 2001*, pages 69–80, New York, NY, USA, 2001. ACM Press.

[21] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 3, pages 1664–1669 Vol. 3, 2005.

[22] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: a public dht service and its uses. In *Proceedings of SIGCOMM '05*, pages 73–84, New York, NY, USA, 2005. ACM Press.

[23] J. Rodriguez, A. Gameiro, C. Politis, G. Kormentzas, and N. Ibrahim. Virtual distributed testbed for optimisation and coexistence of heterogeneous systems. In *TRIDENTCOM*, 2006.

[24] A. C. Snoeren and H. Balakrishnan. An end-to-end approach to host mobility. In *Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 2000.

[25] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. *IEEE/ACM Transactions on Networking*, 12(2):205–218, Apr. 2004.

[26] Z. Turányi, A. Valkó, and A. T. Campbell. 4+4: an architecture for evolving the internet address space back toward transparency. *SIGCOMM Comput. Commun. Rev.*, 33(5):43–54, 2003.