Secure Naming and Addressing Operations for Store, Carry and Forward Networks

Wesley Eddy*, William D. Ivancic†, Dennis C. Iannicca†, Joseph Ishac†, and Alan G. Hylton†
*MTI Systems †NASA Glenn Research Center

***Abstract*—This paper describes concepts for secure naming and addressing directed at Store, Carry and Forward (SCF) distributed applications, where disconnection and intermittent connectivity between forwarding systems is the norm. The paper provides a brief overview of store, carry and forward distributed applications followed by an in depth discussion of how to securely: create a namespace; allocate names within the namespace; query for names known within a local processing system or connected subnetwork; validate ownership of a given name; authenticate data from a given name; and, encrypt data to a given name. Critical issues such as revocation of names, mobility and the ability to use various namespaces to secure operations or for Quality-of-Service are also presented. Although the concepts presented for naming and addressing have been developed for SCF, they are directly applicable to fully connected systems.**

***Index Terms*—Internetworking, Mobile ad hoc networks, Information Security**

I. Introduction

INTERNET technology has become pervasive and is now present in many types of devices that are deployed in the field for use in scenarios where they do not have good (or any) actual Internet connectivity. The devices support data transfer during episodes of connectivity, and the applications and protocol are configured to avoid reliance on many typical infrastructure services (e.g. Domain Name System (DNS)). These devices may be only intermittently connected to other devices, and are used to support data flows where the source and ultimate destination might never be fully connected to one another at any time. Applications operate highly asynchronously, with incalculable constraints on their communication. Often, there are intermediate relaying nodes (or "agents") that must "carry" the data while waiting for connectivity to develop. The systems and applications that are of concern are primarily operating with a much higher level of asynchrony between the data producers, individual relays, and eventual data consumers. We call these "Store, Carry, and Forward" (SCF) systems to distinguish them from typical Store and Forward (SF) systems, which generally operate over a better-connected infrastructure [1][2].

SCF distribute applications can be thought of as an extreme case in Mobile Ad Hoc Network (MANET) because disconnection and intermittent connectivity is the assumed condition whereas in mobile ad hoc networking hop-by-hop connectivity is generally assumed. Fig. 1 illustrates a generic SCF network architecture, with the SCF agents (labeled "SCF") frequently

Corresponding author: W. Ivancic (email: william.d.ivancic@nasa.gov)

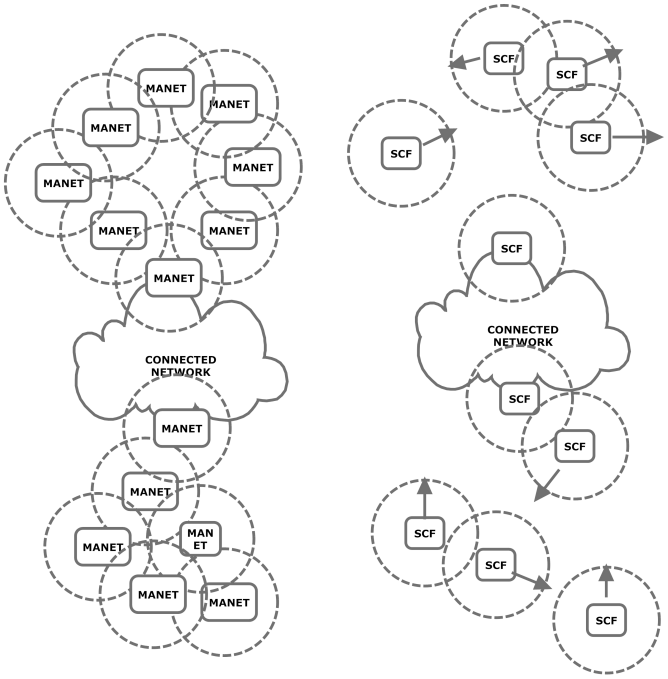

Fig. 1. MANET and Store, Carry and Forward Networks

partitioned into time-varying disconnected subsets. Depending on specifics of an individual scenario, it may be likely that some SCF agents are permanently attached to a connected network providing stable gateways to the other SCF agents. However, in general, the system should be considered to consist of a number of primarily intermittently connected SCF agents at any point in time.

There are numerous lessons to be learned from previous deployments of MANETs and store and forward networks such as Delay/Disruption/Disconnection Tolerant Networking (DTN) [3][4][5][6]. Since SF and DTN networks have no real bounds relative to the maximum time an identified data or control unit can exist within a routed network, SF and DTN are really distributed applications [7]. Regardless, some of the more critical items are:

- SCF systems are generally connected via radio networks. Some radio systems may take far less power to listen than to transmit, though this varies by individual link technology. Unnecessary transmission wastes power on a wireless system and can quickly drain a battery. The problem is compounded for devices whose entire lifetime is determined by their battery (e.g. non-rechargeable sensor

nodes). Thus, reducing the number of transmissions is very important.

- It is highly desirable for the sender to know early in a transmission whether or not the receiver will accept the data, and likewise for the receiver to be able to make this decision early within a transfer. This permits a savings in power and optimization of network capacity usage. For instance, in DTN experiments with large bundles, an entire large bundle may be sent, only to be discarded due to receiver policy for security, resource scarcity, or other issues.
- Disconnected and intermittently connected networks are difficult, if not impossible, to globally synchronize state across - particularly achieving even rough global time synchronization is a challenge. Timer based mechanisms can be used without requiring global time synchronization. Tight time synchronization is seldom necessary and should be avoided in any distributed system as it introduced a single point of failure.
- It is highly desirable for a receiving agent to determine early within a transfer whether or not to accept the data. Large data sets utilize significant processing and storage resources for data that may end up being discarded due to security, resource constraints, or other policy issues.
- It is highly desirable to have some way to establish single-copy routes rather than flooding entire networks with multiple copies of the same data.
- Communications and mobility is not completely random even for ad hoc networks.
- As one moves father from the core (backbone) of the network, nodes generally have less connectivity and capability.

A. Terminology

To aid in discussion within this document it is useful to develop and define some terminology specific to our concepts of SCF networks.

Container	The application/user data to be transported over the network as well as a checksum of that information (the payload).
Shipping Label	Metadata describing the characteristics of a container and its forwarding requirements (the header).

II. Namespaces (Naming and Addressing)

The conclusion goes here We draw much of our concepts for naming and addressing from three source: "Patterns in Network Architectures" [8], "A note on Inter-Network Naming, Addressing, and Routing" [9], and "On the Naming and Binding of Network Destinations" [10]. In particular, Saltzer [10] provides a summary of services, nodes, and attachment points that, if strictly followed, enables: services (a.k.a. applications) to be distributed and/or move, multi-homing of nodes, and mobility.

1) *A given service may run at one or more nodes, and may need to move from one node to another without losing its identity as a service.*
2) *A given node may be connected to one or more network attachment points, and may need to move from one attachment point to another without losing its identity as a node.*
3) *A given pair of attachment points may be connected by one or more paths, and those paths may need to change with time without affecting the identity of the attachment points. [sic]*

Saltzer also points out that three sets of bindings must be maintained and must be discovered in order to send information between services:

- The binding between the service and the node it at which it resides;
- The binding between the node and the network attachment point (or points, if multi-homed); and,
- The path from source attachment point to destination attachment point (routing)[1].

For our discussions, we are not concerned with the bindings of attachment points (i.e. routing). Rather, we consider two basic forms for names: locators and identifiers.

Locators (a.k.a. addresses) are hierarchical, at least that is highly desirable in order to aid in routing as agents need some clue about where to send containers in order to get "closer" even if they do not know the best direct path. For example, to send information from 1600 Pennsylvania Avenue NW Washington, DC to 10 Downing Street, London, England, United Kingdom one knows that sending the information to somewhere in the United Kingdom is getting that information closer to the final destination. Likewise, in a tree-based hierarchical numbering system, if information is to be transferred from 1.2.3.4 to 1.2.100.87, sending the information towards a grandparent node, 1.2, should be getting the information "closer" to the destination or at least to a node that likely has a better idea of where 1.2.100.87 is.

Identifiers are not necessarily hierarchical, and may or may not be human readable. Identifiers should be unique and are used to identify applications or services. Identifiers are bound to locators and discovered via some type of directory service. This binding may change over time. In SCF distributed applications where disconnection is assumed to be normal, distribution and synchronization of these directories required for discovery must be well thought out. Directory services are discussed further in section 11.

III. Philosophy of Multiple Namespaces

In the Internet, there is one namespace, Internet Protocol (IP) addresses for routing. The World Wide Web contains Uniform Resource Locator (URL) for high for higher-level identifiers. The Domain Name System (DNS) directory provides a directory service for mapping computers, services, or any resource (e.g. email, Unique Resource Locator for Web

[1]Whereas, here, Salzer defines routing as between attachment points, we consider routing between source node and destination node as a node may have multiple points of attachment i.e., multi-homing.

services, etcetera) connected to the Internet. (Arguably, Internet Protocol version 6 (IPv6) can support multiple namespaces e.g. Global Unique Address (GUA) for normal routing and ORCHID [11] for higher-layer identifiers, but this facility has not been strongly used, nor will it be easy to, given the way that existing software and hardware works, basically only supporting their known subsets of existing type prefixes, and not new prefixes). For SCF, we are proposing a system of unlimited namespaces, which can be used to construct either pools of application identifiers without mandated structure, or pools of addresses with hierarchical structure. Thus, here, the only difference between addresses and other identifiers is their hierarchical nature.

The limitation of one namespaces, and the global visibility of that namespace to applications, is a root cause of many complexities and fragilities within today's Internet architecture, including within: the interdomain routing system, the DNS, IP neighbor discovery, and other aspects. This has led to a multitude of security issues related to not being able to verify ownership of particular identifiers or addresses, and not being able to authenticate the bindings between particular identifiers and addresses. These issues have, to some extent, been attempted to patch over with BGPSEC/SIDR [12], DNSSEC [13], SeND [14], and other extensions, but these have shifted the security issues to issues of increased operational and infrastructure complexity. Both of the namespaces still have centralized (though hierarchical) allocation and management at the top (e.g. IANN, ICANN, RIRs). There are no real mechanisms available for creating new namespaces, as even with IPv6, the 128-bit fields have been fixed and follow formats with prefixes that Internet Assigned Numbers Authority (IANA) defines.

One of the most significant new facets of this SCF proposal for namespace security is that rather than living within existing namespaces, or subsets of them, we are allowing the creation of an infinite number of new namespaces, and to do so with minimum effort and quickly. Communication is only possible between nodes that have consented to join a given namespace, so though a node may create its own namespace, this will be worthless unless other nodes have policies that allow them to become enrolled within the new namespace. Although similar in concept to Virtual Private Networks (VPNs), the SCF namespaces are more powerful for several reasons, including (1) wider scope compared to VPN prefixes, (2) less brittle configuration and potential for negative interaction with other portions of a host Operating System (OS) networking stack, and (3) better integrated with identifier-address resolution mechanisms, preventing issues of confused scope that occur in VPNs.

SCF's multitude of namespaces also differs very significantly from the Internet, as nodes that do not participate in IP addressing are completely unreachable, and nodes have a relatively poor and unclear granularity in terms of whether they're privately reachable [15] versus using globally routable addresses. Furthermore, the lack of security in the IP namespace, allows visible and invisible proxies, Network Address Translators (NATs), and other middleboxes to subvert the roles and identities of end-nodes in communication flows, without explicit consent, and this brutality is really the only way to grow the Internet and add new features because of the limitation of the single IP namespace.

In summary, the traditional approach to networking in today's Internet is to build one big layer-3 network and then deploy firewalls and VPNs throughout until one deems the network secure. Unfortunately, the configuration becomes so baroque that it will almost certainly break eventually. Our approach is to use credentials to build pair wise relations with neighbors or end-to-end peers, and to verify hosts and data prior to committing resources. No firewalls, VPNs, etc. are required in order to implement the policies and security postures desired. Rather, the architecture is actually just secure by design.

IV. Creating a Secure Namespace

To mitigate potential threats to network, data, and application security SCF needs ways for:

- Applications (end-applications and agents) to validate received data
- End-applications to protect transmitted data
- Agents to validate end-applications that attempt to utilize them
- Agents to validate one another when in contact

Application of namespaces will enable these capabilities.

In a secure namespace, a root server exists somewhere in order to keep a database of registered names within the managed namespace, and to issue certificates when names are allocated from the namespace. Once allocated, a name should never be de-allocated or reused, since the lifetime of containers/labels with the name may be unbounded (however, names may be revoked). The root certificate for namespace X, called the Name Space Identifier (NSI) certificate, needs to be installed on systems hosting applications that will use or (securely) process containers/labels with names from namespace X. The NSI contains a public key for the root, and optionally a description of the valid name formats within the namespace (e.g. via a regular expression), along with optional metadata. The root certificates are the only trusted components of the system.

Given that SCF supports a multitude of namespaces, in order to be implementable and deployable, the format needs to be bounded. We propose to uniquely indicate namespaces through the use of Universally Unique Identifierss (UUIDs) created by the "namespace owner". These UUIDs can follow the format defined in RFC 4122 [16], which supports 128-bit UUID constructed from a timestamp, sequence number, and spatially unique node identification.

Since we recognize that time synchronization in SCF networks is difficult, and that even remembering the current time across boot-ups may be difficult for some nodes, we are initially using RFC 4122's version 1 form of UUIDs, where the timestamp is made robust to such issues via scoping it within the other fields. In this form, the sequence number can either be recorded between boots, or generated randomly (or pseudo-randomly), and where the node identification comes from either Institute of Electrical and Electronics Engineers

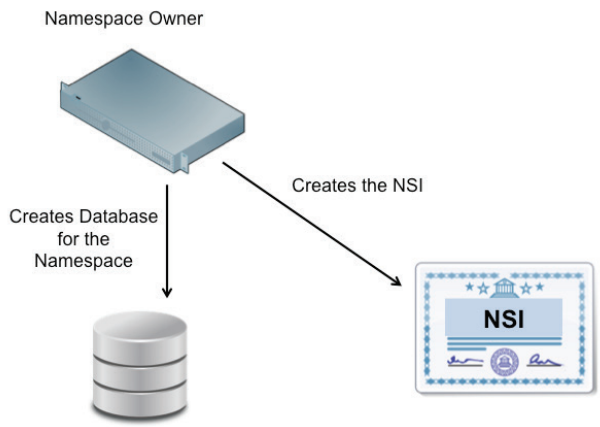

Fig. 2. Creating a Namespace

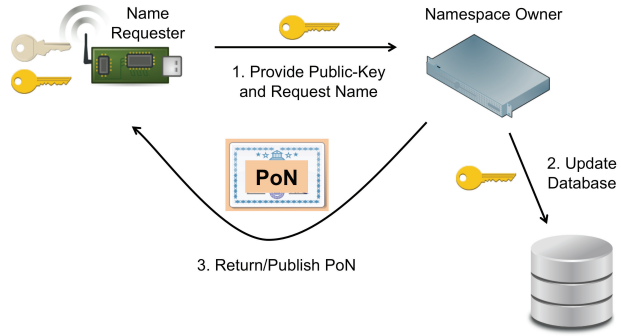

Fig. 3. Creating Proof-of-Names

(IEEE) Media Access Control (MAC) addresses or a self-generated value. One downside to this form of UUIDs is that they are not human-readable or otherwise indicative of the namespace's purpose. Whether this is a downside in practice or not needs to be determined through further experimentation and deployment experience with SCF-based networks. We suspect it may not be an issue, as a database service mapping UUIDs to human-meaningful strings could be created, as well as preconfiguring applications with the UUIDs of namespace they need to operate within so that the UUIDs themselves are not user-visible.

Once the UUID has been selected, the namespace owner will associate it with a public/private keypair by creating a certificate called the Name Space Identifier (NSI) [Fig. 2]. This certificate holds fields for the UUID and public key, and is signed using the private key. Of course, the details of the certificate format and the cryptographic algorithms chosen are of interest, and those are addressed in section 6, Certificate Details.

The namespace owner is now responsible for managing a database recording any names that it has granted. The basic schema for this database needs to include a sequence number, the allocated name itself (an arbitrary string of bits), a public key from the node that the name was allocated to, and potentially timestamps associated with the name creation and/or expiration.

At this point, the namespace has been created, and the namespace owner can serve requests for allocating names, as described in the next section.

It is imperative to understand that in order to be a user of this namespace, the user must obtain a copy of the NSI certificate. This could be done in a number of ways. The key question is how is this bootstrapped, how does the initial creation and distribution of the NSI work in a practical deployment? One method that is highly likely - particularly for SCF networks consisting of sensors - is that an entity is populated with at least one NSI during pre-deployment or even as part of the manufacturing process. For other types of applications (that build overlays for instance), the NSI could be installed when the application is installed. Also, application software could support importing NSIs retrieved from a web server or in some other way, similar to the way the Peer-to-Peer Session Initiation Protocol (P2PSIP) Distributed Hash Tables (DHTs) are configured [17]. Without an NSI, a system cannot validate any names within that particular namespace associated with that particular NSI.

V. Allocation of Names

We need a mechanism to secure and validate names and applications. We propose to support this by using simple certificates called Proof-of-Name (PoN) certificates, related to NSI certificates. How an application receives its names is highly dependent on the operational environment. In some cases, this may be totally pre-configured and statically setup, requiring no direct real-time contact with the root of the namespace. In other cases, applications may be able to dynamically receive PoN certificates during a time of connectivity to the root. The following describes the procedures to obtain and allocate validated names from the perspective of the name requester.

The name requester wishes to obtain a name from the namespace owner to be used as a secure identifier. In order to do so, the requester needs to obtain a PoN certificate from the namespace owner. The requester either asks for a particular name (identifier) explicitly or allows for an owner-selected name. The requester supplies its public key (described in 5.1). The namespace owner either checks its database to see if the specific name is available or generates an unambiguous name per the request. The namespace owner enters the name into the database and marks it as in-use, storing the public key and returning a PoN certificate for the name, signed by the namespace owner [Fig. 3]

Names may be hierarchically assigned by the owner, supporting addressing as just another type of namespace that happens to have structure. A request can also be issued to request a batch of names (a delegated-subnetwork-namespace); this allows for secure prefix-delegation in an addressing system from the namespace owner. In this case, there is a slight modification to the basic operations using a Hierarchical Proof-of-Name (HPoN) certificate:

- The HPoN certificate's name field needs to indicate a range of names that have been allocated, rather than a single name.

- The namespace owner's database needs to handle ranges of names.
- All HPoN certificate holders become namespace owners and need to hold their own database of any PoNs or HPoNs they grant within the delegated subset of the namespace.
- When HPoN certificates are given out from the subsets of the namespace (below the top-level), they include a copy of the upper-level delegated-subnetwork-namespace owner's PoN as well. This is needed in order to validate the HPoNs using (and trusting) only the NSI certificate.

A. User Key Pairs for Requesting Names

The public key used for requesting a name could be from an existing keypair, or one that is generated just for the purpose of use with that name. It all depends on the situation and operational environment. For instance, if privacy/anonymity is a concern, a brand new keypair could be generated to use with an ephemeral name, and everything would be disposable. If access control to the namespace is an issue, keys that are already in-use and vetted somehow (e.g. through being present in a Personal Identify Verification (PVI) card Public Key Infrastructure (PKI) system) could be used.

In general the source of key material should not matter to the naming system. However, there will definitely be some expectations on the sources of key material for specific applications creating and using the namespaces.

VI. Certificate Details

The certificates in our secure naming system are not X.509 certificates [18]. Rather, they need to be much simpler in order to only support the profile of fields that is required for secure naming and reduce processing requirements and code footprint, as well as certificate size.

NSI certificates include the following information

- Namespace UUID
- Public key of namespace-holder
- Signature from namespace owner
- Optional Fields:
  - Cryptographic Algorithm(s) used
  - Additional Serial (Sequence) Number
  - Regular Expression for names within the namespace
  - Creation Timestamp (rather coarse to be useable in a SCF network)
  - Expiration Timestamp (rather coarse to be useable in a SCF network)

PoN certificates only need to include the following information:

- Namespace UUID (matching the NSI)
- Name granted
- Public key of name-holder
- Signature from namespace owner
- Optional Fields
  - Serial (Sequence) Number
  - Creation Timestamp (optional and likely not readily useable in a SCF network)
  - Expiration Timestamp (optional and likely not readily useable in a SCF network)

Numerous cryptographic algorithms are available for generating the needed keypairs, digital signatures, etc., as well as specifications for certificate encoding and other aspects. The NSI certificate can indicate which cryptographic algorithms are to be used for operations within the namespace. This provides the namespace owner with the freedom to pick any sets of cryptographic algorithms, and optionally include them within the NSI. This information is only optional because in some highly embedded systems it may be fixed to the limited capabilities of the particular devices and statically preconfigured or otherwise known rather than a matter of choice. Due to nature of SCF and design principles of SCF, the need to Keep It Simple, in initial experiments we are using only one public key crypto suite (Elliptic Curve Cryptography (ECC) per National Security Agency (NSA) Suite B with 256-bit prime modulus Elliptic Curve Diffie-Hellman (ECDH) and Elliptic Curve Digital Signature Algorithm (ECDSA); one block cipher (Advanced Encryption Standard (AES)-128 Counter encryption mode (CTR)); and, one hash algorithm (Secure Hash Algorithm (SHA)-256), but other deployments can pick different algorithms while supporting the same concepts.

Names and namespaces could have an expiration date, but supporting this goes back to the time synchronization requirement that SCF needs to avoid. However, for SCF distributed applications, time-synchronization could be much 'rougher' than today's connected systems.

The serial number and timestamp fields in the NSI are optional they may be redundant with the fields within the UUID, if the lengths there are sufficient.

The secure namespace concepts presented here are agnostic to the concrete encoding of certificates as they are stored and exchanged. However, in any practical use of these concepts, concrete formats need to be defined. For our experiments, involving small systems such as in space exploration and sensor nodes, with no tolerance for extraneous code, we believe that X.509 certificates carry far too much baggage that isn't strictly necessary. We are instead experimenting with JavaScript Object Notation (JSON) objects that hold the necessary fields, and are rather easy to parse and generate with very small amounts of code.

A. Certificates and Name Revocation

Once issued, an attacker that obtains the corresponding private key could maliciously use an SCF PoN certificate. This is obviously a problem for distributed applications operating over intermittently connected and disconnected networks, as is the time to notify is unbounded and in the extreme is infinite. However, that do not mean one cannot attempt to mitigate the problem.

Two ways that this can be mitigated are through flooding of Certificate Revocation Lists (CRLs) when the compromise of the public key is suspected, and through using lifetimes on the certificates designed to expire before the private key is likely to be compromised. The downsides to flooding CRLs is that it takes memory, network capacity, and time which will all be at

a premium in the use cases SCF is desired for. The downside to expiration times is that using them requires at least rough synchronization of distributed system clocks.

As stated previously, it is difficult to synchronize state across SCF - particularly time. Because of this, traditional PKI techniques for revoking certificates (and names) cannot be used. However, to provide some benefits, time-synchronization may only need to be to a coarse granularity of, for instance, a day. Even that may be non-trivial, in some systems (e.g. across reboots). Regardless, we suggest that other methods are possible.

Without needing other nodes to understand an absolute expiration time, the namespace owner can simply revoke certificates when it unilaterally decides the expiration time has been reached. Because the NSI and PoN certificates have serial numbers, and because certificates within the same namespace should typically be expiring in sequence, this can be exploited in a sort of CRL compression method. For example, a rather small revocation message could be flooded containing only the serial number of the lowest unexpired PoN within a namespace or NSI generated by the owner. This would be signed with the owner's private key. On reception, nodes would be able to store only this sequence number and know that any certificates below it are no longer valid. This implements a sort of rolling window of valid certificates advanced by the owner.

In exceptional cases where the namespace owner needs to revoke certificates prior to natural expiration (e.g. in the case of compromise), a set of additional revoked sequence numbers can be appended to the flooded message. As such incidents will hopefully be significantly more rare than natural expiration, and as once natural expiration is reached, these special case revocations become subsumed by the advancing minimum valid sequence number, we believe this stands a good chance of working quite well in practice.

Note that having the namespace owner announce revocations in this way does not prevent further mechanisms from being incorporated into implementations in order to support more timely responses to incidents known within disconnected pockets of the network. For instance, it may be useful in some environments to be able to blacklist given names if there's confidence that they've been compromised through some other means (like localized Host or Network Intrusion Detection Systems), even prior to a CRL being obtained that covers them.

It is important to note the following two items regarding SCF certificates:

- Since time-synchronization cannot be assumed, the certificates do not strongly support non-repudiation; and,
- A namespace owner destroys the namespace if it revokes its own NSI certificate, only if notice of that revocation reaches all nodes, and is remembered by them (e.g. not forgotten about after a reboot).

In this secure naming system, it is currently much easier to create namespaces and names than it is to effectively destroy them. This may be a fruitful area of future work.

VII. Discovering and Querying Names

Creating a namespace and allocating names within it are necessary but not sufficient to enable communications. There needs to be a way for the names of reachable nodes and applications to be discovered and mapped into lower-layer protocol details in order to establish communications. This is provided through a generic directory service. This does not assume or imply that the addresses are exposed to the applications. Rather, this is the binding between the service e.g., application, and the node it at which it resides. That directory service can be implemented in a number of different ways, all optional for a given use of secure namespaces, and all requiring some further concrete details. These discovery mechanisms are specific to the given lower-layer protocols that secure namespaces are being built on top of in an application.

The namespace owner already maintains a database of the granted certificates, so it seems natural at first to also use that database for directory services by enhancing it with lower-layer locators for the granted names. This clearly has scalability issues, since we have not yet defined a way to distribute the namespace owner role within a namespace. It also would only be useful in scenarios where nodes have frequent connectivity that allows communications with the namespace owner in order to query and update records as their lower-layer locators change. Clearly it is not a complete workable solution for SCF distributed applications.

Another approach is to define neighbor discovery mechanisms similar to those used in IPv6., which will make use of lower-layer multicast/broadcast capabilities in order to learn about the nodes and applications that are available within the local scope of the lower-layer protocols. This is relatively easy to do by adapting the formats, timers, and algorithms that IPv6 neighbor discovery uses, and simply replacing the address fields with secure name fields. In contrast to IPv6 Secure Neighbor Discovery (SeND), however, our secure namespace concept allows much easier proof of ownership to be demonstrated (see section 9). Establishment of neighbor relationships allows communications to be secured with the obtained credentials, optionally providing authentication and/or privacy services for future exchanges.

A neighbor discovery based approach for learning name bindings is likely to work better in most SCF scenarios than a centralized database. However, the neighbor discovery only works within the scope of a single lower-layer hop. It does not support multi-hop forwarding or discovering the bindings for names that are owned by nodes that are multiple hops away within the underlying network. For this, multiple approaches can be made to work, including adaptation of existing routing algorithms and protocols such as Trickle [19] or IPv6 Routing Protocol (RPL) [20], adaptation of resource-locating protocols like Application-Layer Traffic Optimization (ALTO) [21], or developing a gossiping query protocol. In fact, different SCF scenarios that we have defined are certain to drive alternative approaches for this part of instantiating the secure naming concepts within a concrete system. This is an area where the most future work is needed in the near term; however, we believe it can be done largely using existing protocols as models or frameworks. In section 11, Directory Services, we provide some notional deployment scenarios for Directories.

VIII. Validating Name Ownership

Given that the NSI for a namespace is generated by the owner and distributed to any applications that will be working within that namespace, all applications are guaranteed to have the public key of the namespace owner, and be able to check signatures generated using the owner's private key. Since the owner's private key is used to sign the PoN certificates, any PoN certificates received from other applications can be easily validated, without resorting to cumbersome certificate chain operations normally involved in PKI-based systems.

This only proves that the PoN certificate is legitimate and that the name has been issued; it does not prove that the application providing the PoN indeed holds the private key associated with the public key, nor does it prove that the name has not been revoked for some reason.

Proving ownership of the name within the PoN can be done in two ways:

1) Via a challenge-response exchange, in which the verifying party encrypts a puzzle with the public key from the PoN, and awaits a response that could only be generated through decrypting the puzzle, thus demonstrating possession of the private key.
2) Via a signature using the corresponding private key and covering the PoN plus some nonce like a timestamp, sequence number, or other freshness indicator that is bootstrapped out-of-band in a way that prevents replay attacks.

The first method is relatively straightforward but requires both parties to be "online" or with direct low-latency communication, otherwise much time and the corresponding opportunities for communication may be wasted.

The second method is more complex, and requires some help or support from the lower-layer protocols in order to provide the means to indicate freshness of a signature; possibly requiring time synchronization. The significant advantage of this method is that it can potentially be done one-way (without bidirectional exchange) and thus may be more amicable to scenarios where there is only unidirectional connectivity, high-delays, or lack of concurrent end-to-end paths.

IX. Authenticating and Encrypting

Authenticating and encrypting data to a given name is a relatively straightforward process. To authenticate data (the container), the sender signs the data using the private key corresponding to the public key in the sender's PoN. The receiver then uses the PoN public key to check the signature and (if necessary) validates the PoN using the appropriate NSI certificate [Fig.4]. Similarly, to encrypt data, the sender uses the public key in the destination's PoN to encrypt data. The receiver then uses the private key portion of the keypair identified in its PoN in order to decrypt the data it receives.

For SCF networks, it is highly desirable for a receiving agent to determine early within a transfer whether or not to accept the data in order to maximize resource utilization (e.g. bandwidth, storage, computation, battery). Thus, the ability to authenticate the data source is imperative. If the SCF protocol is designed in a matter to allow the shipping label to be processed separately from the container body, the label can be authenticated efficiently within the network precisely in the same manner as the complete container's data.

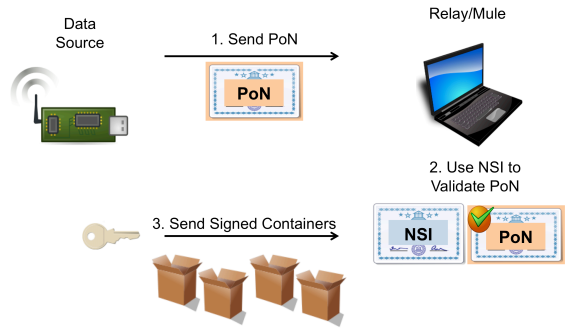

Fig. 4. Authenticating Names

X. Applications and Namespaces

All applications need to have an Application Process Name (APN) that identifies them. Some APNs can be Distributed Application Namess (DANs) in order to support multicast style delivery, but in the basic case, an APN uniquely identifies a single process, and DANs are an advanced topic, beyond the scope of this paper.

SCF agents are applications that parse labels and relay containers for other applications. SCF agents have APNs drawn from a namespace that identifies them as relaying applications. It is assumed that the applications (including SCF agents) share the same set of NSIs in order to be able to communicate within a namespace.

How the application receives an APN, was covered in section 5, Allocation of Names. For now, assume the application knows about its APN, and has a certificate to prove that the APN was assigned from a root for the namespace. The application should internally posses the private key, which corresponds to the public key within its PoN certificate. This allows the application to prove ownership of the APN to any SCF agents or other SCF applications within the same namespace.

A. Suggested Application Program Interface (API) based on APNs

Applications use SCF via an API that can be system/vendor dependent. SCF agents can be within the same platform as applications or remote; the API is all that matters. An example API is shown below:

- Poll for any SCF agents or SCF applications directly known to the local system. The SCF agents in the network may be using a beacon process to broadcast their presence, may be statically configured on systems, or may be discovered through some other type of dynamic process. It does not matter to the application. When polling, the application's APN should be provided, since some SCF agents may only have access controls that permit specific APNs to utilize them, and are not generally available to relay for all applications. This polling should return a

list of APNs that identify the SCF agents. There might be two flavors of polling; one that returns immediately with currently known information, and one that blocks while some on-demand results are collected by the local system; not all systems need to support both.
- Register the application's APN with a particular SCF agent. This should block and return success/failure. Registration may allow the application to reserve space on the agent for incoming/outgoing containers.
- Send a container via a SCF agent the application is registered with. The send call should include some way of 'signing' the request, so that the SCF agent can authenticate it before committing resources for the container.
- Receive a notification from a SCF agent the application is registered with that a container has arrived for the APN, giving relevant label material to the application.
- Request a given container's contents from the SCF agent.
- Withdraw/destroy a registration with a SCF agent. This needs to be authenticated.

It is important to note that the secure namespace operations allow all of these functions to be performed in a robust manner that protects both the network infrastructure and resources (buffers, bandwidth, etc), as well as the nodes and applications themselves. This is a significant difference from other store-and-forward systems that have been built (e.g. based on DTN) with similar APIs between relays and applications, but without the strength of any security to the namespaces involved.

B. Addressing and Routing Application

The following demonstrates how naming is used by applications to communicate with one another and with SCF agents, without having addressing information visible.

Advertising reachability of APNs between SCF agents can be done securely, if, when registering, the application provides a copy of its APN ownership certificate embedded in another certificate that indicates delegation to the SCF agent's APN and is signed using the application's private key. Other SCF agents can then use the public key from the embedded certificate to check that signature, and can use the root certificate for the application's namespace in order to check the inner certificate proving that the application itself really owns the APN initially.

Full routes between SCF agents can be securely advertised by further nesting the certificates this way. This mechanism can be used to prove contacts have existed at one point in time or another, and that transitive sets of contacts have taken place over time, but does not show current or future proof of reachability. That is part of the routing/addressing system.

XI. Directory Services

In order to illustrate how name-to-address (N2A) binding directory services could operate in SCF networks we provide two examples. The first example is an army deployment. This is used to show an SCF with high degree of disconnection. The second example is the use of namespaces for aeronautics. The purpose of the aeronautics example is to show how distributed N2A directories are: updated, enable mobility, and enable use of common infrastructure while simultaneously securing critical infrastructure.

A. Army Field Operations

Figure 5 illustrates a conceptual field deployment for the army. Army communications is highly structured - particularly the closer one gets to the core network. In addition, connectivity and bandwidth increase as one moves from the soldier to the core. The field army hierarchy shown is of the form, Division (DI), Brigade (BR), Battalion (BA), Company (CO), Platoon (PL), and Squad (SQ). Each upper echelon is composed of multiple lower echelons. For example, there are 8 to 16 soldiers in a squad, 2 to 4 squads in a platoon and 3 to 5 platoons in a company. In our example, Companies have full connectivity to Battalions; Battalions have full connectivity to Brigades; and Brigades have full connectivity to Divisions.

In figure 5, each rectangle from Division to Squad represents a SCF routing agent. For convenience, the identities of these SCF routing agents are provided by hierarchical names. The upper rectangle is D1 for Division 1. The lower middle rectangle as squad echelon level is SQ4.PL1.CO5.BA3.BR2.DI1, i.e. <Sqaud4> <Platoon1> <Company5> <Battalion3> <Brigade2> <Division1>. Such a naming system could be use as addressing, but care should be taken to not use application identifiers as the point of attachment locator (address) otherwise multi-homing and mobility problems will result (see the following aeronautics example for clarification). In the army example, we use a hierarchical numbering system for addressing with the alphanumeric names for identities.

The Division is responsible for allocating addresses (location names) in the namespace 1.0. When the Brigade routing, BR2, attaches to Division router, DI1, BR2 sends and empty request for names signifying that it is requesting an address, or, in this case, a set of addresses. The Division allocates the locator name 1.2 and the Delegated-Subnetwork-Namespace (DSN) 1.2.* to the Brigade router, BR2. BR2 is now responsible for that DSN and passes a fraction of that down to the Brigade 3 router, BR3. BR3 in now responsible for DSN 1.2.3.*. As echelon routers connect to the system, they request and are allocated sub-address space. Note, prior to time, T1, the Platoon and Squad routers have not been allocated delegated-subnetwork-namespace (addresses). At time, T1, the Platoon routers receive their address allocations and at time, T2, the Squad routers have delegated-subnetwork-namespace

Two soldiers are represented by their identities, ID123 and ID199. They have no addresses until time, T3, at which time they can communicate up and down within the 1.0 namespace. At time, T4 they become disconnected. At time, T5, they connect to each other and can communicate over a link local address on the wireless connection. They can only communicate via applications that have been allocated and validated. Validation occurs using the common NSIs for those particular applications (see the following aeronautics example for clarification). At time, T6, soldier ID123 can communicate with and across echelons within the 1.0 namespace using an entirely new location identifier. Note, rebinding of location to identity occurs from bottom up. Thus, those nearest to the

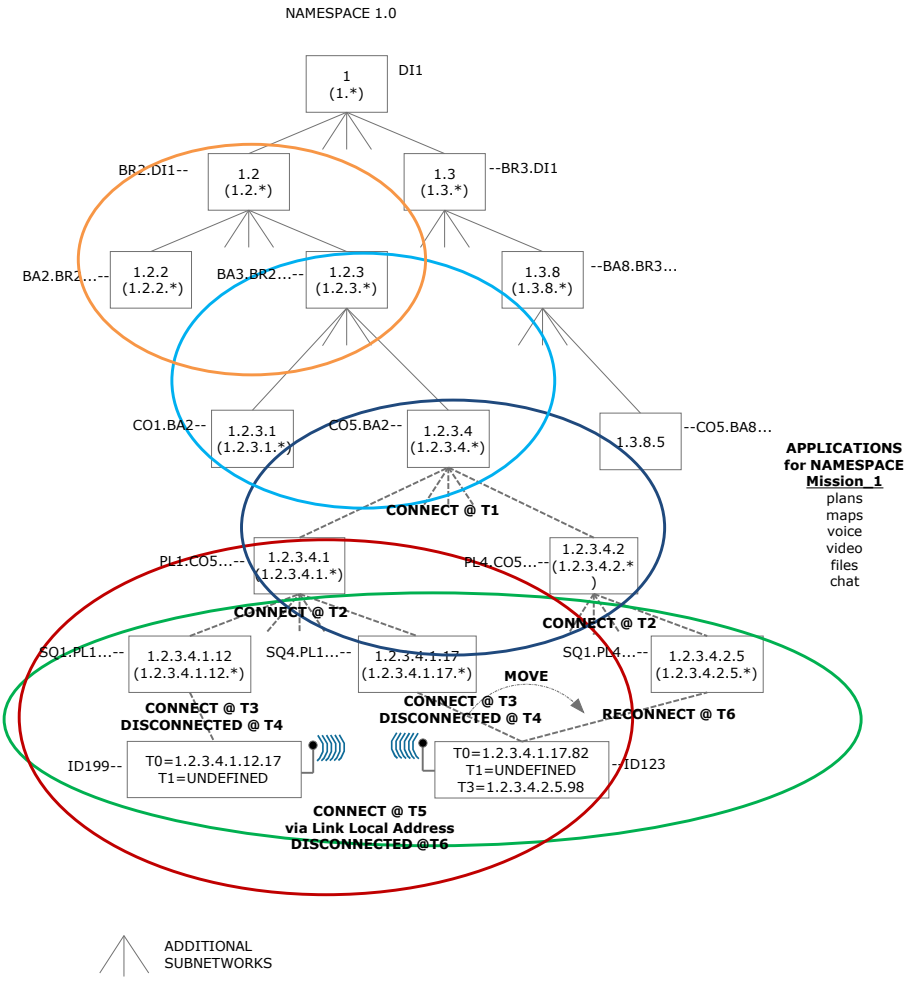

Fig. 5. Notional Field Army Naming and Addressing

mobile node will perceive the updates more quickly than those topologically further away. This is exactly what we want in a SCF network. Also, during times of disconnection, when, for instance, ID123 cannot find or connect to ID199, sending the containers up the tree is perfectly reasonable as one would expect the location of ID199 to eventually propagate to the upper echelons.

B. Aeronautical Mobile Networks

Figure 6 illustrates and aeronautical mobile network and table 1 shows the B2A directory updates. This example is used to show: how the B2A tables get updated; how mobility is accommodated; and, how namespaces can be used to enable shared infrastructure while securing critical infrastructure.

For this aeronautics network, we have a number of domains; each can have their own set of namespaces for applications. We also have a global routing namespace for addressing (location). In aeronautic networks, Air Traffic Control (ATC) is a critical communication system for safety of flight and safety of life. Airline Operation Control (AOC) is used for passenger information, fuel, weather, electronic flight bags and other applications often specific to the airlines. In future networks it is envisioned that ATC and AOC may be permitted to share the same radio links. However, ATC is always given priority over AOC. The other system on an aircraft is the Passenger Internet and Entertainment Services (PIES). This is generally and open network. We also have the open Internet services on the ground as well as the various passengers' corporate networks (private networks).

In figure 6 we show eight different N2A directories. Directory 2 is a local, on aircraft directory. The aircraft ID is NX211. We assign the aircraft router the same ID. In this example, assume there are 5 computing systems onboard, ATC, AOC, and three passengers' computers (e.g. smart phones, pads, laptops, etc.). ATC has one application with a UUID of NX211(atc). AOC has three applications: NX211(efb), NX211(fuel), and NX211(weather). The local onboard router is providing pong and chess as entertainment applications to the passengers. Chuck and Kim have registered to play pong. Chuck and Larry have registered to play chess as well as access to the Internet. Kim will be using her corporate email system.

While on the ground at the gate, all systems are connected via the AeroMAX link. AeroMAX is a shared, high-speed wireless link used on the airport tarmac for communication to

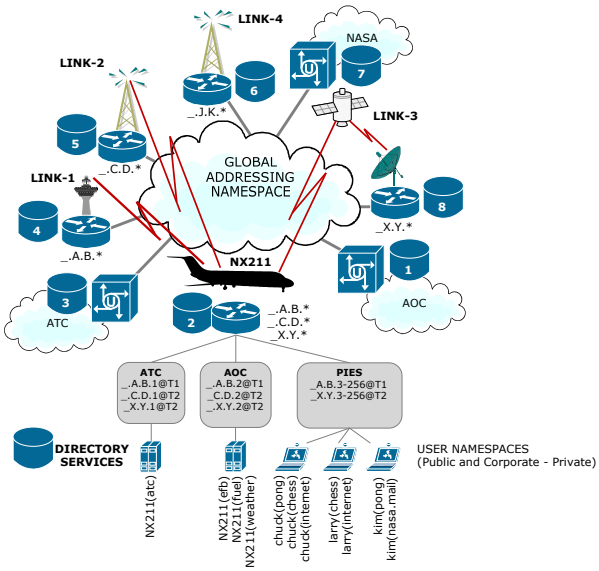

Fig. 6. Aeronautical Network

multiple entities. Once in the air, enroute, the aircraft ATC and AOC can use Link-2 back to the FAA Control Center. Link-2 is a highly reliable, low-rate link. This link is not available to passengers. NX211 happens to have satellite service. This link is available to passengers and (let us assume) it is also available to ATC and AOC services. At some point in the flight, there is a handover from link-2 (Cleveland Control Center) to link-4 (Chicago Control Center).

Table I shows the N2A binding updates that occur during various stages of flight. At time, T1, the onboard systems update their binding with the local directory, D2. Also, all systems are permitted to use the AeroMAX. Thus, all systems send binding information to Directory 4. Directory 4 then updates the AOC and ATC directories. At time, T2, Links 2 and 3 are active. ATC and AOC are permitted to use both links with PIES is only permitted to use Link-3. The corresponding directly connected directories, D5 and D8 receive binding updates. Note that ATC and AOC are now multi-homed (i.e. have two or more N2A binding entries). In addition, at time, T2, ATC, AOC and PIES have all moved topologically. Finally, at T3, Link 2 is inactive and link 3 is active. Thus ATC and AOC binding updates show mobility from the Cleveland Control Center to the Chicago Control Center.

XII. Quality-of-Service

In the Aeronautics Networking example, we show that specific networks can be separated via namespaces. In this manner we can restrict use of various links such as links 2 and 3 to various namespaces (here ATC and AOC. This is one aspect of (Quality of Service (QoS)).

An important aspect of QoS regarding SCF networks is the ability to manage resources (e.g. storage, computation, bandwidth and power - battery life). This is critical for SCF systems as resources are precious. Furthermore, and inability to properly manage resources opens the system to denial-of-service (DOS) attacks. Namespace can be used in SCF firewalls to control resource allocations such as:

- What namespaces are permitted to use any of the system resources at all;
- What links may be used by particular namespaces;
- How much storage will be allocated to a particular namespace; and
- The size of the container that may be accepted for reception.

Note, since we can prove that containers were sent by the name-holder, QoS using namespaces has authentication unlike what the IP world offers. It is also much stronger than what Bundle Authentication Block (BAB) offers for DTN [22] since it gives proof all the way back to the source, not just to the previous hop. Thus, it is robust to having compromised agents in the middle of the network generating bogus containers.

TABLE I
Name-to-Address Bindings

T1 - Link 1 (WIMax)						
T2 - Link 2 (Cleveland Control Center), Link 3 (KuBand Satellite)						
T3 - Link 4 (Atlanta Control Center, Link 3 (Ku-Band Satellite)						
	T1		T2		T3	
Directory	Application	Address	Application	Address	Application	Address
1	NX211(efb)	_.A.B.2	NX211(efb)	_.C.D.2	NX211(efb)	_.J.K.2
AOC			NX211(efb)	_.X.Y.2	NX211(efb)	_.X.Y.2
	NX211(fuel)	_.A.B.2	NX211(fuel)	_.C.D.2	NX211(fuel)	_.J.K.2
			NX211(fuel)	_.X.Y.2	NX211(fuel)	_.X.Y.2
	NX211(weather)	_.A.B.2	NX211(weather)	_.C.D.2	NX211(weather)	_.J.K.2
			NX211(weather)	_.X.Y.2	NX211(weather)	_.X.Y.2
2	chuck(pong)	_.A.B.3	chuck(pong)	_.X.Y.3	chuck(pong)	_.X.Y.3
Local	chuck(chess)	_.A.B.3	chuck(chess)	_.X.Y.3	chuck(chess)	_.X.Y.3
	larry(chess)	_.A.B.4	larry(chess)	_.X.Y.4	larry(chess)	_.X.Y.4
	kim(pong)	_.A.B.5	kim(pong)	_.X.Y.5	kim(pong)	_.X.Y.5
3	NX211(atc)	_.A.B.1	NX211(atc)	_.C.D.1	NX211(atc)	_.J.K.1
ATC			NX211(atc)	.X.Y.1	NX211(atc)	.X.Y.1
4	NX211(atc)	_.A.B.1				
AeroMAX	NX211(efb)	_.A.B.2				
	NX211(fuel)	_.A.B.2				
	NX211(weather)	_.A.B.2				
	chuck(internet)	_.A.B.3				
	larry(internet)	_.A.B.4				
	kim(internet)	_.A.B.5				
5			NX211(atc)	_.C.D.1		
Cleveland			NX211(efb)	_.C.D.2		
Control			NX211(fuel)	_.C.D.2		
Center			NX211(weather)	_.C.D.2		
6					NX211(atc)	_.J.K.1
Atlanta					NX211(efb)	_.J.K.2
Control					NX211(fuel)	_.J.K.2
Center					NX211(weather)	_.J.K.2
7	kim(nasa.mail)	_.A.B.5	kim(nasa.mail)	_.X.Y.5	kim(nasa.mail)	_.X.Y.5
NASA						
8			chuck(internet)	_.X.Y.3	chuck(internet)	_.X.Y.3
Internet			larry(internet)	_.X.Y.4	larry(internet)	_.X.Y.5
Public			kim(internet)	_.X.Y.5	kim(internet)	_.X.Y.5

XIII. Conclusions

The secure naming system presented provides a light-weight method for allocating and validating application names and locators (addresses) that could be deployed in a Store, Carry and Forward, normally disconnected networks. The technique can also be applied to fully connected networks. By ensuring that the application names separate from the location names, the system readily handles multi-homing and mobility.

Our system could be an enabling technology for the aeronautics networks vastly simplifying operations and management. For instance, every infrastructure provider can maintain its own namespaces for management of its equipment. Since these are not exposed to the users, most security threats to the infrastructure instantly disappear.

Infrastructure providers that wish to confederate for the purposes of creating a routable address space between them

can do so, and those routable addresses still do not expose their management and control planes to one another. Mobile users sharing NSI certificates for that address space, can roam to any provider that's also part of it, without any pre-existing trust relationships, and obtain addresses. If they need to be globally reachable themselves, they can use their own namespaces above, created for specific domains (ATC, AOC, PIES) and allowing applications from all domains to utilize the same infrastructure yet be completely isolated from one another except for sharing bandwidth. Such techniques also apply to securing "Critical Infrastructure Networking". There will be no fear of accidentally leaking routes, because the namespaces have been factored out, access to names is secured, and proof of ownership is verified.

References

[1] W. Ivancic, W. Eddy, D. Iannicca, and J. Ishac, "Store, Carry and Forward Problem Statement," Internet Engineering Task Force, Internet-Draft draft-ivancic-scf-problem-statement-00, Jul. 2012, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ivancic-scf-problem-statement-00.txt

[2] W. Ivancic, W. WesleyEddy, D. Iannicca, and J. Ishac, "Store, Carry and Forward Testing Requirements," Internet Engineering Task Force, Internet-Draft draft-ivancic-scf-testing-requirements-00, Jul. 2012, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ivancic-scf-testing-requirements-00.txt

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay-Tolerant Networking Architecture," RFC 4838 (Informational), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[4] K. Scott and S. Burleigh, "Bundle Protocol Specification," RFC 5050 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5050.txt

[5] L. Wood, W. Ivancic, W. Eddy, D. Stewart, J. Northam, C. Jackson, and A. da Silva Curiel, "Use of the delay-tolerant networking bundle protocol from space," in *Proceedings of the 59th Astronautical Congress, Glasgow. IAC*, 2008.

[6] W. Ivancic, P. Paulsen, D. Stewart, W. Eddy, J. McKim, J. Taylor, S. Lynch, J. Heberle, J. Northam, C. Jackson *et al.*, "Large file transfers from space using multiple ground terminals and delay-tolerant networking," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. IEEE, 2010, pp. 1–6.

[7] R. Watson, "Timer-based mechanisms in reliable transport protocol connection management," *Computer Networks (1976)*, vol. 5, no. 1, pp. 47–56, 1981.

[8] J. Day, *Patterns in network architecture: a return to fundamentals*. Prentice Hall, 2007.

[9] J. Shoch, "A note on inter-network naming, addressing, and routing," *Xerox Palo Alto Research Center, IEN*, vol. 19, 1978.

[10] J. Saltzer, "On the Naming and Binding of Network Destinations," RFC 1498 (Informational), Internet Engineering Task Force, Aug. 1993. [Online]. Available: http://www.ietf.org/rfc/rfc1498.txt

[11] P. Nikander, J. Laganier, and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)," RFC 4843 (Experimental), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4843.txt

[12] G. Huston and R. Bush, "Securing bgp with bgpsec," in *The Internet Protocol Forum*, vol. 14, no. 2, 2011.

[13] [Online]. Available: http://www.dnssec.net/

[14] J. Arkko, J. Kempf, B. Zill, and P. Nikander, "SEcure Neighbor Discovery (SEND)," RFC 3971 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFCs 6494, 6495, 6980. [Online]. Available: http://www.ietf.org/rfc/rfc3971.txt

[15] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear, "Address Allocation for Private Internets," RFC 1918 (Best Current Practice), Internet Engineering Task Force, Feb. 1996, updated by RFC 6761. [Online]. Available: http://www.ietf.org/rfc/rfc1918.txt

[16] P. Leach, M. Mealling, and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace," RFC 4122 (Proposed Standard), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt

[17] C. CullenJennings, B. Lowekamp, E. Rescorla, S. Baset, and H. HenningSchulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-p2psip-base-23, Nov. 2012, work in progress. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-p2psip-base-23.txt

[18] R. Housley, W. Polk, W. Ford, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 3280 (Proposed Standard), Internet Engineering Task Force, Apr. 2002, obsoleted by RFC 5280, updated by RFCs 4325, 4630. [Online]. Available: http://www.ietf.org/rfc/rfc3280.txt

[19] P. Levis, N. Patel, D. Culler, and S. Shenker, *Trickle: A self regulating algorithm for code propagation and maintenance in wireless sensor networks*. Computer Science Division, University of California, 2003.

[20] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," RFC 6550 (Proposed Standard), Internet Engineering Task Force, Mar. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6550.txt

[21] (2013, January) Application-layer traffic optimization (alto). [Online]. Available: http://datatracker.ietf.org/wg/alto/

[22] S. Symington, S. Farrell, H. Weiss, and P. Lovell, "Bundle Security Protocol Specification," RFC 6257 (Experimental), Internet Engineering Task Force, May 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6257.txt

Appendix

List of Acronyms

AES	Advanced Encryption Standard
AOC	Airline Operation Control
API	Application Program Interface
APN	Application Process Name
ATC	Air Traffic Control
CRL	Certificate Revocation List
CTR	Counter encryption mode
DAN	Distributed Application Names
DHT	Distributed Hash Table
DNS	Domain Name System
DSN	Delegated-Subnetwork-Namespace
DTN	Delay/Disruption/Disconnection Tolerant Networking
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GUA	Global Unique Address
HPoN	Hierarchical Proof-of-Name
IANA	Internet Assigned Numbers Authority
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPv6	Internet Protocol version 6
JSON	JavaScript Object Notation
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
N2A	name-to-address
NAT	Network Address Translator
NSA	National Security Agency
NSI	Name Space Identifier
OS	Operating System
PIES	Passenger Internet and Entertainment Services
PKI	Public Key Infrastructure
PoN	Proof-of-Name
PVI	Personal Identify Verification
QoS	Quality of Service
SCF	Store, Carry and Forward
SHA	Secure Hash Algorithm
SF	Store and Forward
URL	Uniform Resource Locator
UUID	Universally Unique Identifiers
VPN	Virtual Private Network