2012

Patterns in network security: an analysis of architectural complexity in securing recursive inter-network architecture networks

BOSTON UNIVERSITY

METROPOLITAN COLLEGE

Thesis

PATTERNS IN NETWORK SECURITY: AN ANALYSIS OF ARCHITECTURAL COMPLEXITY IN SECURING RECURSIVE INTER-NETWORK ARCHITECTURE NETWORKS

by

JEREMIAH SMALL

B.S., University of Massachusetts, Amherst, 2000

Submitted in partial fulfillment of the

requirements for the degree of

Master of Science

2012

Approved by

First Reader

John Day
Lecturer in Computer Science

Second Reader

Lou Chitkushev, PhD
Associate Professor of Computer Science

Third Reader

Ibrahim Matta, PhD
Professor of Computer Science

"For now, you should concentrate on shifting your mindset from the traditional networking model of custom static layers to thinking in terms of distributed applications that provide IPC recursively. This isn't as easy as it sounds."

– John Day (*Patterns in Network Architecture*, Chapter 7)

"You must unlearn what you have learned."

– Master Yoda

PATTERNS IN NETWORK SECURITY: AN ANALYSIS OF ARCHITECTURAL COMPLEXITY IN SECURING RECURSIVE INTER-NETWORK ARCHITECTURE NETWORKS

JEREMIAH SMALL

ABSTRACT

Recursive Inter-Network Architecture (RINA) networks have a shorter protocol stack than the current architecture (the Internet) and rely instead upon separation of mechanism from policy and recursive deployment to achieve large scale networks. Due to this smaller protocol stack, fewer networking mechanisms, security or otherwise, should be needed to secure RINA networks. This thesis examines the security protocols included in the Internet Protocol Suite that are commonly deployed on existing networks and shows that because of the design principles of the current architecture, these protocols are forced to include many redundant non-security mechanisms and that as a consequence, RINA networks can deliver the same security services with substantially less complexity.

Contents

1 INTRODUCTION **1**

1.1 Thesis 1

1.2 Method 2

1.3 Security Requirements 3

1.3.1 Security Services 4

1.3.2 Basic Requirements 6

1.3.3 Layer Security 9

1.3.4 Taking Stock 9

1.4 Networking Mechanisms 9

1.5 Phases of Communication 13

1.6 Overview of the current architecture (the Internet) 14

1.6.1 Architectural Overview 14

1.6.2 Example 17

1.6.3 Summary 21

1.7 Overview of RINA 21

1.7.1 Architectural Overview 21

1.7.2 Example 25

1.7.3 Summary 30

1.8 Configurations 31

1.8.1 The Internet 31

1.8.2 The RINA Internet 32

2 ANALYSIS **33**

2.1 Security protocols of the current architecture 33

2.1.1 Data-link Layer 35

2.1.2 Network Layer 36

2.1.3 Transport Layer 38

2.1.4 Application Layer 40

2.2 RINA Security Modules 42

2.2.1 Module Descriptions 42

2.2.2 RINA's Security Story 44

2.3 Security Comparison of RINA and the Internet 44

2.3.1 Differences in Approach to Security 45

2.3.2 Measuring Flows, Protocols, and Mechanisms 45

2.3.3 Observations 58

3 CONCLUSION **66**

3.1 Thesis Review 66

3.2 Recommendations and Topics for Further Study 66

References **69**

About the Author **73**

List of Tables

2.1 Network Mechanism Detail 34
2.2 Sample of Internet Security Protocols 35
2.3 RINA Security Modules 42
2.4 Internet Results for Secure HTTP Traffic 46
2.5 RINA Results for Secure HTTP Traffic 53

List of Figures

1·1 Internet Layers 15

1·2 Example Network 18

1·3 RINA Layers 23

2·1 Network Mechanism Summary 33

2·2 Internet Model Basic Data Flow Layers 47

2·3 Internet Link Security Layers 48

2·4 Internet Network Layer Security 50

2·5 Internet Full Security Stack 51

2·6 RINA Basic Data Flow Layers 53

2·7 RINA Link Security Layers 55

2·8 RINA with Organization Backbone Layer 56

2·9 RINA Full Security Stack 57

2·10 Flows, Protocols, and Mechanisms 59

2·11 Additional Flows, Protocols, and Mechanisms to Add Security 61

2·12 Flows, Protocols, and Mechanisms (No Link Security) 63

2·13 Additional Flows, Protocols, and Mechanisms to Add Security (No Link Security) 64

2·14 Adjustment for Dedicated RINA Directory 65

List of Abbreviations

AH	.....	Authentication Header
AP	.....	Application Process
APE	.....	Application Process Entity
API	.....	Application Programming Interface
APN	.....	Application Process Name
ARP	.....	Address Resolution Protocol
BGP	.....	Border Gateway Protocol
CDAP	.....	Common Distributed Application Protocol
DAN	.....	Distributed Application Name
DHCP	.....	Dynamic Host Configuration Protocol
DIF	.....	Distributed IPC Facility
DNS	.....	Domain Name System
DTLS	.....	Datagram Transport Layer Security
EAP	.....	Extensible Authentication Protocol
EAPOL	.....	EAP Over LAN
EFCP	.....	Error and Flow Control Protocol
ESP	.....	Encapsulating Security Payload
FA	.....	Flow Allocator
FAI	.....	Flow Allocator Instance
FTP	.....	File Transfer Protocol
HDLC	.....	High-level Data Link Control protocol
HMAC	.....	Hash-based Message Authentication Code
HTTP	.....	Hyper-Text Transfer Protocol
ICV	.....	Integrity Check Value
IEEE	.....	Institute of Electrical and Electronics Engineers
IETF	.....	Internet Engineering Task Force
IP	.....	Internet Protocol
IPC	.....	Inter-process Communication
LAN	.....	Local Area Network
LDAP	.....	Lightweight Directory Access Protocol
MAC	.....	Media Access Control
MD5	.....	Message Digest Algorithm

OSPF	.....	Open Shortest Path First protocol
PCI	.....	Protocol Control Information
PDU	.....	Protocol Data Unit
PKI	.....	Public Key Infrastructure
PNA	.....	*Patterns in Network Architecture*
PPP	.....	Point to Point Protocol
PS	.....	Processing System
PSTN	.....	Public Switched Telephone Network
QoS	.....	Quality of Service
RADIUS	.....	Remote Authentication Dial In User Service
RFC	.....	Request For Comment
RIB	.....	Resource Information Base
RINA	.....	Recursive Inter-Network Architecture
RMT	.....	Relaying and Multiplexing Task
SA	.....	Security Association
SCTP	.....	Stream Control Transmission Protocol
SDU	.....	Service Data Unit
SIP	.....	Session Initiation Protocol
SMTP	.....	Simple Mail Transfer Protocol
SNMP	.....	Simple Network Management Protocol
SPD	.....	Security Policy Database
SSH	.....	Secure SHell
SSL	.....	Secure Sockets Layer
TCP	.....	Transport Control Protocol
TLS	.....	Transport Layer Security
UDP	.....	User Datagram Protocol
VLAN	.....	Virtual Local Area Network

Chapter 1

INTRODUCTION

1.1 Thesis

The global network, commonly known as "The Internet", that millions of organizations and individuals rely upon daily for communication, commerce, and entertainment has its roots in a network architecture designed over 40 years ago. (Crocker, 1969) The Internet as we know it today had its genesis in roughly 1977 when TCP was officially split from IP. (Cerf and Postel, 1978) The Internet was not originally designed with security in mind and unsurprisingly it has proven to be hard to secure.[1] One of the reasons that the internet may be so hard to secure is that there are so many different protocols to manage. (Braden, 1989b; Braden, 1989a) This is unsurprising given that the Internet was not designed from the system view, but rather as a suite of independently developed protocols that each provide a distinct function. Section 2 will show that these functionally independent protocols are forced to include redundant mechanisms. Likewise each one of these protocols either provides its own security or else relies upon a separate protocol for security. Currently each security solution uses different distinct mechanisms even though the problems being solved at each level are largely the same.

Recursive Inter-Network Architecture (RINA) networks take a different approach to layers based on a basic model of Inter-Process Communication (IPC) and use

[1] See the National Vulnerability Database (NVD) at `http://nvd.nist.gov/` and the Common Vulnerabilities and Exposures List (CVE) at `http://cve.mitre.org/cve/`

recursive deployments called Distributed IPC Facilities (DIF) to achieve large scale networks. (Day, 2008; Day et al., 2008) DIFs are natural security domains and are securable containers. (Small, 2011) One of the guiding principles of RINA is the reduction of redundancy by separating mechanism from policy. This has lead to an architecture that uses fewer overall protocols. Unlike the Internet protocols, the RINA protocols have very few redundant mechanisms. So it seems plausible that networks built upon these principles should be able to provide security with less complexity by using fewer overall mechanisms to deliver an equivalent amount of security compared with what the Internet protocol suite provides today. Put another way, the Internet has several different types of layers to secure and it uses different mechanisms to secure each one. RINA has only one repeating homogeneous layer to secure and requires only one set of mechanisms to do so. RINA should therefore be able to provide security with less overall complexity. If this is the case, then to meet a set of security requirements against a set of security domains, there should be no RINA configuration that requires more distinct networking mechanisms than does the Internet. This thesis will endeavor to show just that.

1.2 Method

We will set about showing that RINA networks require fewer mechanisms to deliver security services than the Internet protocol suite using the following method. In order to compare RINA networks to the Internet, a basis of comparison will be needed. The goal is to measure the complexity of the two network architectures in delivering security mechanisms. One way to do this is to measure the number of distinct mechanisms required to deliver on security requirements. To inform us as to what services should be deployed we will consider common security requirements for networks. Each architecture delivers security services in a different way. The Internet uses additional

security protocols. RINA provides security modules that plug into the architecture at appropriate points. These must be examined and cataloged in terms of what mechanisms they include. In addition to cataloging the mechanisms, they will be aggregated into two groups: security mechanisms and non-security mechanisms. The catalog of mechanisms can then be used to measure the complexity of the target networks in delivering on the security requirements. The target networks will be measured for complexity in terms of the number of flows, the number of protocols, the number of distinct mechanisms, and the number of instances of those mechanisms as each set of requirements is applied. If the hypothesis is correct, the RINA network should never require more distinct mechanisms to deliver security than does the equivalent network built upon the Internet Protocol Suite.

When selecting Internet protocols, we will choose methods that are commonly used. If there is no common solution, then a protocol with the fewest mechanisms will be chosen. On the RINA side, whenever a new DIF is needed, it will be treated as a black box and all mechanisms will be counted even if fewer may be needed. In doing so, the analysis will attempt to err on the side of being favorable to the Internet model rather than RINA. In addition, although we are counting distinct mechanisms, if a given protocol has more than one distinct version of a mechanism we will count the protocol as having only one. For example, if a protocol is broken up into multiple sub-protocols that each provide their own delimiting mechanism to account for record length, like the Transport Layer Security (TLS) protocol does (Dierks and Rescorla, 2008), delimiting will be counted only once for the whole protocol.

1.3 Security Requirements

In this section we will define the set of security requirements that we will then measure the two architectures against. The requirements to be discussed are defined in terms

of security services. Before describing specific requirements we will first define the security services.

1.3.1 Security Services

In order to meet the requirements of deploying secure network communication, several security services must be provided. There are many models of security that describe the various security services. For the purposes of this thesis we will refer to the following security services: Authentication, Authorization, Integrity, Confidentiality, Availability, and Accountability. They are defined as follows.

Authentication

The New Oxford American Dictionary gives the following definitions:

> **Authenticate** – prove or show (something, esp. a claim or an artistic work) to be true or genuine
>
> **Subject** – a person or thing that is being discussed, described, or dealt with

Authentication is the process by which we verify a subject's claimed identity. There are three types of authentication: Peer-Entity, Data Origin, and User. Authentication techniques can range from simple shared secrets (passwords), to asymmetric cryptographic techniques, to biometric measurement.[2] Peer-entity authentication is used by communicating entities to verify the identity of a communicating entity mutually or one-way. This is generally accomplished with asymmetric cryptographic techniques. Data origin authentication is used to verify the source that is sending data. Symmetrical cryptographic techniques such as a keyed digest[3] are typically used. User

[2]Biometric authentication being only useful for human authentication.

[3]A Keyed digest is a cryptographic mechanism where by the contents of a message are combined with a secret key and passed through a cryptographically secure one-way hash function.

authentication is used to verify the identity of a human. (Jacobs, 2011)

Authorization

> **Authorize** – give official permission for or approval to (an undertaking or agent)

Authorization is the process by which we determine if a given subject has permission to access a piece of information (often referred to as an "object") or initiate an action on an object. It is also referred to as Access Control.

Integrity

> **Integrity** – internal consistency or lack of corruption in electronic data

Integrity protection is most often done using a keyed or signed digest. The digests can vary in strength and generally use symmetric or asymmetric cryptographic techniques along with a cryptographically secure one-way hash function.

Confidentiality

> **Confidential** – intended to be kept secret

Confidentiality is maintained in networks by using cryptography to transform the data into a form that looks like random noise. Only entities with the appropriate keys can reverse the transformation to recover the original unencrypted data.

Availability

> **Available** – able to be used or obtained; at someone's disposal

Accountability

> **Accountable** – (of a person, organization, or institution) required or expected to justify actions or decisions; responsible

1.3.2 Basic Requirements

In his book *Engineering Information Security* (Jacobs, 2011), Stuart Jacobs describes the following basic set of security objectives:

Customer

1. Ensuring confidentiality and integrity of all customer information entrusted to the organization and any information the organization obtains as a result of customer usage or organization services
2. Maintaining customer contracted for service availability in compliance with those service level agreements between the customer and the organization
3. Enforcing customer access to only authorized features so that the actions of one customer cannot interfere with service availability to, or information about, other customers
4. Ensuring error-free and nonmalicious interactions between customers and the organization infrastructure

Peering Service provider

1. Ensuring confidentiality and integrity of all peering services provider information entrusted to the organization, such as routing, subscriber, billing information, and video content
2. Maintaining peering service provider contractually obligated availability in compliance with those service level agreements between the peering service provider and the organization
3. Enforcing peering service provider access to only authorized features so that their actions cannot interfere with service availability to, or

information about, organization customers

4. Ensuring error-free and nonmalicious interaction between peering service providers and the organization infrastructure.

Infrastructure

1. Maintaining the confidentiality and integrity of system information be it signaling and control or related to operations, administration, maintenance and provisioning (OAM&P)
2. Limiting operations personnel access to system attributes based on an authorized "need-to-know" basis
3. Providing error-free and nonmalicious interaction between operations personnel and infrastructure components consistent with provisions to customers and peering service providers.

Network Provider Security Requirements

Using the above objectives as a guide, let us define the security requirements for a network provider that we can measure our networks against. Since this thesis is about network architectures, we will focus on requirements that apply to the data being transmitted across these networks as well as the data needed to operate the network. Organizations have many more security requirements than just these, but they fall outside the scope of this discussion.

A network provider has two types of external entities that interact with its network. It has customers that purchase network service from it, and peering providers which it may have data sharing agreements with or from which it may have purchased lower level network service. Providers also have to consider internal personnel who interact with its network. Based on this we can say providers want to:

1. Prevent unauthorized systems from accessing the network.[4] This means systems accessing the network must be authenticated and be subject to access control and possibly non-repudiation.

2. Protect user traffic, traffic originating outside the network, from interception, interruption, modification, or fabrication. This generally means traffic must be protected with the confidentiality, integrity, and potentially non-repudiation mechanisms.

3. Protect management traffic, traffic originating within the network, from interception, interruption, modification, or fabrication. This also implies the use of confidentiality, integrity, and non-repudiation mechanisms.

Or to summarize, protect data as it comes into the network, and protect any data that goes out of the network whether that be over a physical link, or transiting a peer or lower level provider network. Internally generated data should be prevented from leaving the network or else protected as it leaves the network.

Non-provider Network Security Requirements

At first the network security requirements of an organization whose primary business is not delivering network services seem somewhat different than a network provider. However if we look at this architecturally and treat applications as customers of the network, then the requirements start to look the same. Protect data when it comes into the network, protect data as it leaves the network and either prevent management data from leaving the network, or protect it as it leaves the network.

[4]This does not include preventing unauthorized access to those systems, though that is also of great importance.

1.3.3 Layer Security

Even if every layer of a network may not require all security services to be enabled for a given deployment, the network architecture being used must be capable of delivering the following security services at each "layer" of the network.

1. Identify entities accessing resources – implies authentication.
2. Protect resources from eavesdropping while in transit – implies confidentiality
3. Prevent unauthorized access – implies authorization and confidentiality
4. Ensure reliable unmodified delivery of requested resources – implies integrity
5. Ensure authorized resources are available when requested – implies availability
6. Provide remediation when one of the other mechanisms fails – implies accountability / non-repudiation

1.3.4 Taking Stock

From an architectural perspective, it looks like each layer of the network has the same requirements. Therefore we should be able to measure the amount of complexity required to provide security over the entire network by measuring the complexity at each layer. We will measure the complexity in delivering the security mechanisms to each layer by measuring the number of flows, protocols, and networking mechanisms required to deliver on the security requirements at each layer. The networking mechanisms will be discussed in the next section.

1.4 Networking Mechanisms

There are 21 mechanisms that will be used to catalog the security protocols and modules of the architectures being compared. Sixteen of those are non-security mecha-

nisms; they are: Delimiting, Initial State Synchronization, Policy Selection, Addressing, Flow or Connection Identifier, Relaying, Multiplexing, Ordering, Fragmentation / Reassembly, Combination / Separation, Data Corruption, Lost & Duplicate Detection, Flow Control, Retransmission / Acknowledgement, Compression, and Activity. There are also five security mechanisms: Authentication, Access Control, Integrity, Confidentiality, and Non-repudiation. A short description of each mechanism is given here; for complete details see *PNA* chapter 2. (Day, 2008)

Delimiting The mechanism used to indicate the beginning and ending of a Protocol Data Unit (PDU). There are generally two types of delimiting, internal and external. If a protocol relies on the layer below for delimiting it is called external, if delimiting is included in the protocol via a flag, length field, etc., it is referred to as internal. For the purposes of this analysis, only internal delimiting will be counted as part of a protocol.

Initial State Synchronization The function provided by a protocol or mechanism that prepares for transfer of data. This includes binding local ports, synchronization or handshake messages, initialization of timers, etc. It also refers to the resynchronization of state during data transfer such as rekeying. All of the protocols examined in this analysis include this mechanism.

Policy Selection Many protocols have options that can be selected during initialization including algorithms, timer limits, maximum PDU size, etc. The process of negotiating these options is called policy selection.

Addressing In all but point to point connections, addressing is required to identify to which application process a given PDU is destined, and in many cases which

application process it came from. The synonyms used for this purpose are called addresses.

Flow or Connection Identifier Any protocol providing multiple flows for a given address must include a way to distinguish those flows from one another. This is done by assigning an identifier to each flow and including it in the Protocol Control Information (PCI).

Relaying Relaying is a mechanism used to forward PDUs on networks that are not fully connected graphs. If two stations are not directly connected, but are both connected to a third station, messages can be relayed across that third station.

Multiplexing Multiplexing refers to the bundling of multiple flows onto a single lower level connection. For a sparsely connected network of systems (or processes), multiplexing is often required.

Ordering This is a mechanism by which the order in which PDUs are delivered to the destination is guaranteed to be the exact order in which they were sent by the source.

Fragmentation / Reassembly This mechanism is needed when the maximum size of lower layer PDUs is smaller than the PDUs being passed to it from the N+1 layer. The N+1 PDUs are broken into smaller fragments so that they fit within the N layer PDUs. The fragments are then reassembled at the destination before being passed across the N+1 layer boundary. Protocols can fragment PDUs for other purposes as well.

Combination / Separation This mechanism is used to increase the utilization of a given lower layer flow. When multiple N+1 PDUs can fit within a single N layer PDU they can be sent together. Relaying stations can separate the N+1 PDUs if they must be forwarded to different destinations. If more than one N+1 PDU for the same destination is included in a single N layer PDU then each N+1 PDU is individually delivered across the N+1 layer boundary.

Data Corruption Data corruption mechanisms refer to the means by which protocols detect the occurrence of and possibly repair transmission errors.

Lost & Duplicate Detection This mechanism is used to detect when a PDU has already been received and processed and also the detection of the condition that one or more PDUs has been lost in transmission.

Flow Control Flow control is a mechanism by which a source can avoid sending PDUs faster than the destination can process them, or otherwise control the rate at which PDUs are sent across the network.

Retransmission / Acknowledgement A property of guaranteed delivery, this is a mechanism that deals with resending PDUs that never reached their intended destination and also confirming to the source that a PDU has arrived.

Compression Mechanism by which the number of bits required to convey a piece of information is reduced in a fashion that can be reversed at the receiving end. This is done for efficiency.

Authentication This is a mechanism used by end points of communication to verify the identity of the other end points.

Access Control This is a mechanism by which an application process determines if a requesting entity is authorized to access a given resource.

Integrity Similar to data corruption, the integrity mechanism is a means to detect if a PDU has not arrived as intended by the sender. Unlike data corruption which deals with unintentional errors, integrity mechanisms attempt to detect deliberate modification, fabrication, or interruption of PDUs between the source and destination.

Confidentiality This mechanism prevents the data from being divulged to unintended parties. This involves the use of cryptography.

Nonrepudiation This mechanism is intended to prevent entities from denying participation in a transaction, often through the use of cryptographic signatures based on asymmetric key pairs.

Activity This is the mechanism by which long running connections are maintained even across long periods of inactivity. These mechanisms are often called "keep alive" mechanisms.

1.5 Phases of Communication

Any form of communication passes through three different phases: *enrollment*, *allocation*, and *data transfer*. Communication begins with enrollment, then passes to allocation, and then to data transfer. When the data transfer phase completes, the instance of communication returns to the allocation phase. Likewise when the allocation phase completes, communication returns to the enrollment phase. The operation of moving from the allocation phase back to the enrollment phase is called *de-allocation*. The operation invoked to complete the enrollment phase is called *de-enrollment*. This thesis will refer to these phases and so a brief description of each is given here.

Enrollment According to *PNA* (Day, 2008): "The enrollment phase creates, maintains, distributes, and deletes the information within a layer that is necessary to create instances of communication." The data managed by enrollment includes: addressing, directory entries, routing entries, available policies for security and Quality of Service (QoS), etc. Enrollment has traditionally been a mostly manual affair. For example, the assignment of a telephone number on a Public Switched Telephone Network (PSTN) or learning someone's name.

Allocation *PNA* also refers to this phase as the *establishment* and *synchronization* phase. It defines the phase thusly: "The synchronization phase creates, maintains, and deletes the shared state necessary to support the data transfer phase." The data managed by allocation includes: initial state, selected policies, and final synchronization data. This phase is roughly analogous to the process of placing a telephone call or a salutation in vocal communication.

Data Transfer Communication reaches the data transfer phase when data unrelated to (but supported by) the earlier two phases begins to flow between two entities in at least one direction. In most cases instances of communication are in the data transfer phase for a majority of their existence. It is the data transfer phase that we typically think of as "communication".

1.6 Overview of the current architecture (the Internet)

1.6.1 Architectural Overview

As John Day points out in *PNA*, the global network commonly known as "The Internet" is not actually an inter-network, that is a network of networks, but rather

a concatenation of networks sharing a single global address space.[5, 6] The Internet Engineering Task Force (IETF) is the governing body for the design of the Internet. The Internet is defined by a series of documents known as Requests for Comments (RFC). At the time of writing, there have been 6725 such RFCs published.[7]

The architecture of the Internet is based on the classic model of functional layers. That is a set of logical divisions of protocol machines by function. The Internet has five such layers: physical, data-link, network, transport, and application.[8] See Figure 1·1. In actuality, what we think of as "The Internet" wasn't designed as is, but has evolved over time through constant revision. The organization of the protocols into layers happened later in the process. What the IETF was designing is often referred to as a "Protocol Suite" rather than an architecture. The architecture that we refer to in this thesis as "The Internet" began around 1977 at the point when TCP was separated from IP giving us the five layers to discuss. (Cerf and Postel, 1978)

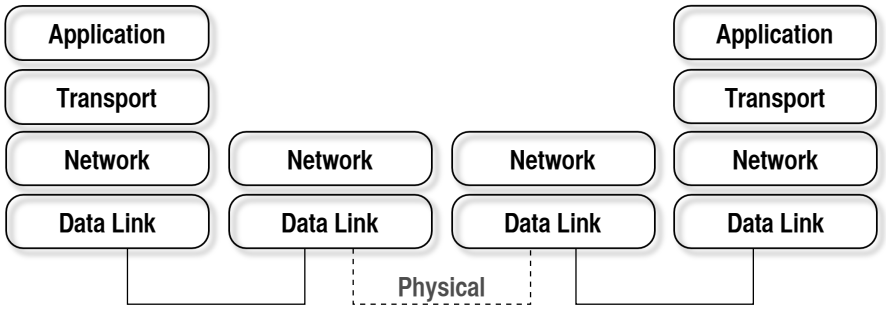

Figure 1·1: Internet Layers

The physical layer models the physical transmission of bits across a medium, be it conductor, optical, wireless, or otherwise. In essence, this covers the hardware that sends and receives signals that are interpreted as bits. The scope of this layer is

[5]The Internet Protocol address space

[6]The collection of private networks sitting behind a Network Address Translating (NAT'ing) firewall not withstanding.

[7]See `http://www.ietf.org/rfc/rfc-index.txt`

[8]According to the IETF in RFC 1122, there are only 4, dropping the physical layer.

usually either point to point, or over a small group of processing systems in the case of a shared medium like Ethernet or wireless.

The data-link layer directly above the physical layer is the domain of flow control over the physical medium. Data-link protocols like the High-level Data-link Control protocol (HDLC) provide error and flow control functions like integrity protection (detection of errors) and retransmission (if an error is detected and cannot be repaired). In shared mediums data-link protocols also detect and handle transmission collisions (when more than one sender is trying to transmit at the same time). The scope of this layer is usually over a small group of processing systems (e.g. a Local Area Network or LAN). Common data-link protocols include: HDLC, PPP, and IEEE 802.3.

The Internet's network layer provides a means for any connected processing system (PS) to communicate with any other connected processing system. It does this by providing a global address space and protocols for routing groups of bits, called packets, from PS to PS by relaying them across as many individual physical transmissions ("hops") as necessary. The scope of the networking layer is all reachable processing systems on the network. Common protocols include: IP, ARP, DHCP, BGP, and OSPF.

The transport layer provides a similar function to the data-link layer but at a larger scope. The Transport Control Protocol (TCP) handles flow-control, guaranteed delivery, retransmission, and other functions while the User Datagram Protocol (UDP) provides unreliable best-effort packet delivery. On the Internet, the transport layer is tightly coupled with the network layer and thus its scope is the same, e.g. the global network. Common Protocols include: TCP, UDP and SCTP.

The application layer is at the top of the stack and includes those processes that use the lower layers to communicate with other processes connected via the global

network. There are many application protocols that fit into the application layer such as the Hyper-text Transfer Protocol (HTTP), Secure SHell protocol (SSH), Simple Mail Transfer Protocol (SMTP), etc. The scope of the application layer is different in nature than the lower layers. Each application protocol could be considered its own "layer" sitting above transport. The scope of each of those are the number of reachable processes that understand the specific application protocol. Each of these theoretical application "layers" share the same rank. That is, they all communicate over the transport layer. Common application protocols include: HTTP, FTP, SSH, SIP, SMTP, DNS, and LDAP.

An important attribute of this architecture is that on each processing system, each layer communicates only with the layer directly above and below it. It does this via an Application Programming Interface (API). Between processing systems (or hosts), it appears that communication is happening within the layer, indeed, within the layer is where any shared state is "stored". Messages are sent between processes within the layer.

1.6.2 Example

It is difficult to understand how the layered architecture works just from the description of how each of those layers works. Let us illustrate the combined use of the layers via a common example of an HTTP browser and server communicating over "The Internet". For simplicity sake, we will consider 6 processing systems. A host or processing system running the HTTP browser, a host running the HTTP server, a host running a Domain Name Resolution (DNS) server, and three hosts acting as routers each connected to one of the non-router hosts and to the other two routers. Please refer to Figure 1·2. In this case the routers take the place of the entire Internet.

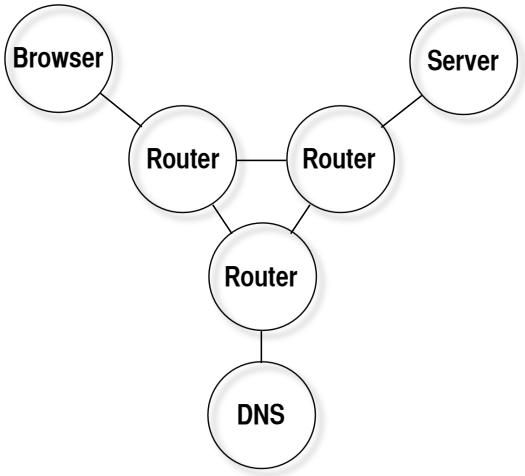

Figure 1·2: Example Network

Enrollment Before communication can happen, each of the systems in question must be enrolled. That is each must be given one or more addresses. In this case we will assume manual, static enrollment meaning that each of the systems in our example have been provided configuration data by a human or manufacturing process. The browser, server, and DNS systems (we'll call these the application hosts) each get a single address (note that the addresses actually label the points of attachment). Each of the routers get 3 addresses (one for each point of attachment). The application hosts also need to be configured with the address of their gateway router (the address through which all traffic must flow through to reach the other application hosts). The browser and server hosts also need to be provided with the address of the DNS server.[9] We will assume that the DNS server already has a entry mapping the HTTP server hostname to its IP address.

Allocation Before data can be transfered between the browser and server, the browser must first determine the address of the server. In this case, the user of the

[9]In practice, these hosts would not likely share the same DNS server, but we will set aside that point for now.

browser knows the name of the desired server, e.g. `www.bu.edu`. Since the internet protocol only understands addresses, the browser must first translate the name `www.bu.edu` into an IP address. It does this by using a different protocol – DNS. In this case, the browser host already knows the address of the DNS server (because it was provided directly) and can make a DNS request to fetch the address associated with `www.bu.edu`. DNS is a connectionless protocol that operates over the Universal Datagram Protocol (UDP). The browser asks its underlying operating system to allocate a UDP port to be used to communicate with the DNS server. The browser AP then creates a DNS request and passes it to the transport layer which wraps the DNS request in a UDP packet (including the source and destination port)[10] and passes it down to the network layer with the address of the DNS server. The network layer wraps the UDP packet in an IP packet that includes (among other things) the source and destination IP address. The network layer must then decide which interface to send the packet to. In this case, each host is configured with a "default gateway" address. The network layer code then uses the Address Resolution Protocol (ARP) to translate the IP address into a link-layer address (often called a Media Access Control address or MAC address). Once it has the MAC address of the gateway, it passes the IP packet and the MAC address of the gateway to the Data-link layer (sometimes called the "driver") and the networking hardware then transmits the bits onto the medium.

The networking hardware at the gateway router receives the bits transmitted by the browser host and passes them up to the driver which strips off the MAC address (and other link-layer PCI) and passes the remainder of the bits up to the networking layer. The router then reads the destination address and sees that it is not its own address. Each router keeps a table of forwarding rules that tell it for each destination (sometimes aggregated), on which interface to send the packet out so that it can reach

[10]Interestingly the name of the DNS application is a well known port number – 53.

its final destination. The router then uses ARP to determine the MAC address of the next physical hop and passes the IP packet and the new MAC address to the link layer (driver) which then transmits the bits across the second medium. This process repeats until the packet reaches the router that has a direct connection to the DNS server. Each layer on the DNS host strips off the PCI that it used and passes the remaining bits to the next layer up until it finally reaches the DNS server process. The DNS server takes the name `www.bu.edu` and looks up the corresponding IP address, builds a response and sends it back to the browser using the port number provided in the UDP header. The packet follows the same process of arriving and departing from each router that it did on the way to the DNS server, but not necessarily via the same routers. When the response reaches the browser host and all the lower layer PCI is stripped from the response, the browser then reads the IP address and can proceed with opening a TCP connection to the HTTP server.

The browser now asks its underlying operating system to allocate a port and open a TCP connection to the HTTP server system using the IP address from the DNS response. The TCP protocol machine establishes a "connection" with the remote server by sending a series of SYN, and SYN-ACK PDUs via the lower layers which travel through the network just like the DNS request discussed earlier. Once the connection (shared state) is established, the browser can make an HTTP request for the desired resource using the HTTP protocol.

Data Transfer Once a connection has been established, the browser and server can exchange data, initially an HTTP-GET request specifying `www.bu.edu/index.html`. This request is sent across the network in the same fashion as the DNS request and the TCP SYN and SYN-ACK messages before it.[11] Once the request is received by the server it will stream the bytes making up the document `index.html` back to the

[11] Actually these constitute data transfers at the network layer and below also.

browser. It is important to note that the stream of bytes is actually broken up into chunks (packets) in many cases and those chunks are sent independently through the network. Since they may arrive out of order, as each chunk of data arrives at the browser system, the TCP protocol machine must keep track of the order of the packets received and send the data within them back to the browser application process in the correct order. As each chunk of data is passed up to the browser a TCP-ACK message is sent back to the server again following the same router to router transfer described earlier.[12]

Deallocation Once all of the data for `index.html` has been transferred from the server to the browser, the browser will ask its operating system to close the TCP connection.[13] This is done by sending a TCP-FIN message to the server, and deleting local state.

1.6.3 Summary

The preceeding description is a gross simplification of how the internet protocol stack works, but it should suffice for our purposes. For complete details consult the RFCs published by the IETF. This architecture has been refined over the past 40+ years, but it continues to suffer from problems.[14]

1.7 Overview of RINA

1.7.1 Architectural Overview

The Recursive Inter-Network Architecture is the reference model of the network architecture suggested in John Day's book *Patterns in Network Architecture: A Return*

[12]TCP involves more complexity, but this simpler view will suffice for our discussion.

[13]Some browsers and servers support sending multiple resources (images, style sheets, etc.) over a single connection.

[14]See the National Vulnerability Database (NVD) at `http://nvd.nist.gov/` and the Common Vulnerabilities and Exposures List (CVE) at `http://cve.mitre.org/cve/`

to Fundamentals (PNA).(Day, 2008) It centers around the realization that networking is IPC and *only* IPC.(Day et al., 2008) PNA is an effort to investigate the fundamental principles of networking. It walks through the minimal mechanisms required to transfer data between processes starting from two processes on the same system and scaling up to two separate systems separated by some number of intermediate relaying systems. From this exercise emerges a recursive structure of layers called Distributed IPC Facilities (DIF) that provide functionality similar to the data link, network, and transport layers of the Internet model. Unlike the Internet however, RINA is a true inter-network architecture, where routing and addressing have scope within the DIF and layers above and below have no need of this information. DIFs can be stacked as deep as necessary generally increasing in scope as one moves "up" the stack. RINA is an architecture based on the principles explored in PNA and is currently under active development by an organization known as the Pouzin Society. Along with PNA, the Pouzin Society has been producing detailed specifications of the RINA architecture[15]; these are roughly equivalent to the collection of RFCs maintained by the IETF. (Day, 2009a; Day, 2009b; Day, 2009c; Bunch, 2010; Day, 2010c; Day, 2010a; Day, 2010b; Day, 2011a; Day, 2011b)

Each DIF consists of one or more IPC processes containing the following components: a Resource Information Base (RIB), IPC Management Tasks (Enrollment, Routing, Directory, Resource Allocation, Security Management), Relaying and Multiplexing Task (RMT), a Service Data Unit (SDU) Delimiter task, an SDU protection task, a flow allocator, and the Error and Flow-Control Protocol (EFCP) machine. Each IPC process may also have one or more Application Processes (APs) running "above" it that will rely upon the IPC process to communicate with other APs that have access to the same DIF. The IPC Process itself is an AP and relies on DIFs "below" it for communication and so on down to the physical transmission medium.

[15]See `http://www.pouzinsociety.org`

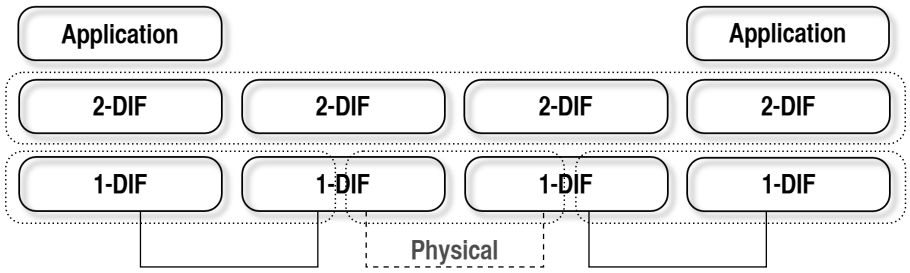

Figure 1·3: RINA Layers

When exchanging messages, a DIF is similar to an Internet layer in that the protocol machine on one node (in this case an IPC process) communicates only with protocol machines in the same layer (DIF), usually on other nodes. For example in TCP/IP, TCP protocol machines "talk" to other TCP protocol machines on other nodes, but not to IP protocol machines. Similarly IP only "talks" to IP. Communication between the layers occurs via an API on a single host. This aspect is the same in RINA in that IPC processes in a DIF only communicate with other IPC processes in the same DIF. A key difference between RINA and TCP/IP is that all the networking mechanisms found in TCP/IP (which are different layers) are in the *same* layer in RINA. The messages that get passed between IPC processes are called Protocol Data Units (PDUs), these consist of Protocol Control Information (PCI)[16] and a payload which RINA calls user data. User data can consist of a single Service Data Unit (SDU), an SDU fragment, or multiple SDUs. An SDU is a PDU from the layer above. An IPC process is permitted to fragment or combine SDUs as needed to meet the requested Quality of Service (QoS). Operationally, the distinction between PDU and SDU may not matter much because of RINA's inherent recursiveness. However it is useful to make a distinction when considering the security between layers. For the remainder of this thesis we will use the term PDU to refer to an *(n+1)-DIF* SDU

[16]sometime referred to as header and trailer

with *n-DIF* PCI.

Also like the Internet model, RINA is organized into layers, however, RINA layers (DIFs) are only deployed as needed; for small networks one or two layers are sufficient. Unlike Internet layers which are separated by function, each DIF provides data transfer, routing, directory, and management functions. RINA also provides components (referred to as modules) for authentication, authorization, and PDU protection (integrity and confidentiality). An address in RINA names an IPC process[17] and only has scope within the layer. Unlike the Internet, DIF addresses are not visible outside of the DIF and application processes (APs) request flows to other APs via their name. The IPC manager is responsible for mapping that name to a DIF address and forwarding the request via the appropriate lower level DIF. The only data visible to the AP (or higher level DIF) is the name of the Application Process Entity (APE) that it requested and the port id (handle) given to it by the local IPC manager to access the requested flow.[18] The port number is independent of the connection-id used by the IPC manager to identify the outbound flow. This has several important aspects. Firstly, since the N+1 AP only has a port number / handle to the flow, the underlying IPC manager is free to change the lower level connection-ids or even to use multiple connection-ids[19] to provide the requested QoS without the N+1 AP knowing anything about it. This allows among other things mobile clients to change points of attachment (cell towers) without affecting the flow between the communicating APs and without the traffic having to take a circuitous route back through a point of origin. Secondly, it provides security via the principle of least privilege (need-to-know). The AP has no visibility into the address or indeed even the address space being used by the DIF. It has no need to know this, unlike IP where the applications must know the address of the interface and port number on which the remote process is listening

[17] This is in contrast to an IP address which names a point of attachment.
[18] A *port id* in layer N is a *connection endpoint id* in layer N+1.
[19] multi-homing

in order to connect to it (even if it has to use DNS to translate from a name).

RINA DIFs use two protocols to provide its services: the Error and Flow-Control Protocol (EFCP), and the Common Distributed Application Protocol (CDAP). EFCP is composed of two sub-protocols, DTP and DTCP, that work in tandem to get bits (PDUs) from one IPC process to another and also to establish flows between DIF members. This is roughly equivalent to the combination of IP, TCP, and UDP. EFCP is based on a transfer protocol called Delta-T that was a contemporary with TCP in the early days of the Internet and is based around the idea that flow control can be done by simply binding on 3 timers: maximum packet lifetime, maximum retransmission time, and maximum acknowledgement time. (Watson, 1981) CDAP is used to communicate at a higher level between APs, in this case IPC Management APs. CDAP is the protocol used by IPC processes to operate on configuration objects involved in enrollment (like DHCP), allocation, directory queries (like DNS), routing updates (like BGP and OSPF), and management (like SNMP). CDAP connections are authenticated. Data managed by IPC processes, which is stored in a Resource Information Base (RIB), is subject to access control. CDAP is a generic application protocol and could be used by any distributed application. It is designed around the concept that there are only 6 operations that can be done on distributed objects: create, delete, read, write, start, stop.[20](Day, 2010b)

1.7.2 Example

Again, it may be difficult to understand how RINA networks work just from a description of its components, so let us now walk through our example from the previous section involving the 6 processing systems: one browser, one server, one directory, and three routers. See figure 1·2. The first thing to notice is that a dedicated di-

[20]Examination of other popular application protocols such as HTTP should show that each could be reduced down to the same set of operations.

rectory server (DNS) is no longer required.[21] Directory services are provided by the IPC-process and used during allocation which is why addresses need not be exposed outside of the DIF. We will keep the router that was directly connected to the DNS server to maintain the multi-path nature of the example.

Enrollment For simplicity, we will assume that the three routers and the web server have already enrolled in the DIF, and that the browser Processing System (PS) has been configured with the Distributed Application Name (DAN) of the DIF. In this case we'll say it is "RINA-Internet.DIF". We will also assume for brevity that the Application Process Name (APN) of the web server PS is already known to the IPC-process on the server's gateway router. That is, its APN, let's call it `URL.www.bu.edu`, is already mapped to the address of the IPC-process servicing the web server AP.

The browser AP wants to get an HTML document from the server.[22] The browser AP must find the DIF in which the server is available. To do this the browser AP will make an Allocate API call to its IPC Manager including the destination APN (in this case `URL.www.bu.edu`) and the QoS parameters. The IPC manager must then select the DIF over which to allocate a flow. If the local IPC process is not a member of the DIF that the server is available on it may enroll in it at this point.[23]

To enroll in the DIF the IPC process will make an Allocate API call to its (N-1) DIF (in this case the driver of the hardware network device) including the DAN of the DIF as well as the destination APN and QoS parameters. The source APN (in this case the IPC process on the browser PS, we'll call it "IPC.Browser") is provided by the OS/driver. The driver responds with a handle for the N-DIF IPC process to use to send PDUs to the IPC process on the browser's gateway router.

[21] In very large DIFs, dedicated name resolvers may be deployed, but they are neither required nor essential, it is simply a matter of policy.

[22] Presumably because it was requested by a user.

[23] It may be the case that IPC processes will attempt to enroll in the DIF on system startup, specially if the IPC process is part of the underlying operating system (OS).

Now that PDUs can be exchanged between "IPC.browser" and the IPC process on the gateway router, we'll call it "IPC.browser.gateway", we can proceed with the enrollment of "IPC.browser" into the DIF "RINA-Internet.DIF". "IPC.browser" sends a CDAP CONNECT including the DAN ("RINA-Internet.DIF"), authentication information, and other CDAP connection parameters. If the authentication succeeds, and "IPC.browser" is authorized to access "RINA-Internet.DIF", then a CDAP CONNECT_R message is sent indicating success. After receiving the CONNECT_R, "IPC.browser" sends an M_START Enrollment including the address it thinks it should have (which could be null). "IPC.browser.gateway" then assigns an address (possibly the one sent by "IPC.browser" and sends an M_START_R Enrollment with that address and some additional information. "IPC.browser.gateway" also sends zero or more M_CREATE messages to provide additional information needed to communicate with the DIF, such as policies, etc., to "IPC.browser" who responds to each with an M_CREATE_R. Once "IPC.browser" has enough state to participate in the DIF, "IPC.browser.gateway" sends an M_START Operation and "IPC.browser" responds with an M_START_R Operation. "IPC.browser" is now enrolled in the DIF "RINA-Internet.DIF".

Allocation Unlike the "driver-level" DIF (1-DIF) which has only a single pre-allocated flow, "IPC.browser" must allocate flows across the network. Now that "IPC.browser" is a member of the DIF, it can proceed with servicing the allocation request API call made by the browser AP. The browser AP has requested a flow to `URL.www.bu.edu`. "IPC.browser" creates a Flow Allocator Instance (FAI) to handle this flow and passes it the allocation request. The identifier of this FAI is unique within "IPC.browser" and will be passed back to the browser AP to use as a handle for further interaction with this flow. We will call it *browser-port*. The FAI builds a CDAP create request for the flow that includes the local address of "IPC.browser",

the local port *browser-port*, the APN of the server `URL.www.bu.edu`, authentication information,[24] requested QoS parameters, and proposed policies for the connection.[25] An EFCP-AE is also created to handle the flow once it is established. The create flow request must be sent to the server AP, so the FAI consults the local RIB to look up its address. The address for `URL.www.bu.edu` is not in the local RIB, so the FAI must follow the local search rules. In this case it means that we will send the request to the only other known member of "RINA-Internet.DIF", "IPC.browser.gateway". So the request is sent via 1-DIF to "IPC.browser.gateway". "IPC.browser.gateway" is notified of the incoming request and checks its RIB for a mapping of `URL.www.bu.edu` to an IPC process address. None is found so the gateway follows its own search rules and forwards the request again. In this case it will forward the request to the gateway router of the server. At "IPC.server.gateway", the mapping is found in the RIB and the request is forwarded directly to the IPC process "IPC.server". When the request finally reaches "IPC.server", and it sees that the requested AP is available locally and that the browser AP has access to it, an FAI and EFCP-AE will be created to service this flow. The identifier of the local FAI will serve as the port id that the server AP will use for further interaction with this flow should it accept the request. We will call it *server-port*. IPC then passes the request received from the browser AP to the server AP via the Allocate API call. It is important to note that unlike server processes on the Internet, the server APE need not be running or listening in order for it to accept flow allocations. The AP may be started by IPC prior to passing the allocation request. This is similar to how processes are created by name when running from a command shell.[26]

If the server AP accepts the request from the browser AP, an M_CREATE_R is generated that includes the local port id, *server-port*, the agreed upon policies, etc.

[24] the specific information is dictated by policy

[25] In this case we will require in-order guaranteed delivery similar to what TCP provides.

[26] This is similar to how the *inetd* process works on Linux systems.

This response is then forwarded back across the network to "IPC.browser". When the request reaches "IPC.browser", IPC binds the port-id *browser-port* to the browser AP and to the flow identifier which is the concatenation of the two port-ids *browser-port* and *server-port*. The browser AP can now begin sending application PDUs to the remote server AP.

Data Transfer Now that a flow has been successfully allocated between the browser and the server, the browser AP may make requests. In this case, we will assume that the browser and server continue to use HTTP to communicate.[27] The browser will now make an HTTP GET request for `http://www.bu.edu/index.html`. To do this it simply constructs the HTTP request and makes a Write API call using *browser-port* to the local IPC manager, the buffer of data containing the HTTP GET request becomes an SDU from the perspective of "IPC.browser" and it is delivered to the EFCP Application Entity Instance (AEI) bound to *browser-port*. The EFCP-AEI generates the appropriate PDU by prepending the SDU with the flow-id and other N-DIF PCI and hands it over the Relaying-Multiplexing Task (RMT). The RMT then places it on the outbound queue to be delivered to the 1-DIF.[28] The 1-DIF treats the "RINA-Internet.DIF" PDUs as SDUs and delivers them across the physical medium to the gateway router as described earlier.

The 1-DIF PDUs are received by the 1-DIF IPC process (driver) at the gateway router, the 1-DIF PCI is stripped from the SDU and is passed up to the IPC process named "IPC.browser.gateway". "IPC.browser.gateway" interprets the 1-DIF SDU as a PDU and it sees that the PDU's destination is not itself. It consults its forwarding table and sees that it should forward the PDU along the N-1 port connected to

[27]It should be noted that the functionality provided by HTTP overlaps substantially with that of CDAP. We will use HTTP to illustrate that RINA does not impose any restrictions on the choice of application protocol.

[28]If the browser PS had more than one hardware networking interface, the RMT would have to select one of them based on policy and QoS.

the server gateway router. The process repeats at the server gateway router and eventually the "RINA-Internet.DIF" PDU arrives at the IPC process "IPC.server". At this point "IPC.server" recognises the destination of the PDU is local and that it should deliver SDUs for the indicated flow to the local N+1 port *server-port*. The PCI is stripped from the PDU and the resulting SDU is passed via API call to the server AP.

The server AP processes the request and then transmits the bytes from the file `index.html` to the browser as one or more PDUs. These PDUs follow the same pattern of transfer back to the browser AP. Since HTTP operates on documents, it is likely that the server will send a large PDU to the DIF below and the RMT of that IPC process will likely fragment the SDU into several parts sending them as separate PDUs across the DIF.[29] Once all of the "RINA-Internet.DIF" PDUs reach "IPC.browser" it will forward them to the browser AP. The browser may continue to request further documents from the server, images, etc., until it can display the web page.

Deallocation When the browser has no need to communicate with the server any longer it will make a deallocate API call to the IPC manager with the local port-id *browser-port*. The FAI will then tear down the EFCP-AEI and send a CDAP Delete request to "IPC.server". When this is complete the FAI can be torn down. The ports *browser-port* and *server-port* are now available for reuse in their respective IPC processes.

1.7.3 Summary

Like the previous section, the preceding description is a simplification of how the RINA architecture works. For further details see PNA and the reference specifications

[29]Note that it can only do this if it has established an in-order guaranteed delivery policy on one of the underlying flows.

maintained by the Pouzin Society.[30] Let us now note some similarities and differences between RINA and the Internet.

Like the Internet, RINA is organized into layers. PCI is added and removed when crossing layer boundaries. Communication between layers happens only on individual processing systems via API calls and communication within the layer happens via exchange of PDUs.

Some differences between the architectures include the following. Addresses on the Internet name points of attachment to the layer below while addresses in RINA name IPC processes within the layer. This means that routing between addresses on the internet is creating a path across N-1 layer ports while in RINA routes are paths within the layer allowing multi-homing (more than one point of attachment) and also changing or multiplexing traffic across points of attachment. Another key difference between the Internet and RINA is that Enrollment, Allocation, and Data Transfer use exactly the same mechanisms at each layer, the only difference is policy. We will see in chapter 2 that this is an advantage in delivering a secure network with less complexity.

1.8 Configurations

Now that we have a basic understanding of the security requirements that we will be measuring against and a basic understanding of how the two architectures work, we can go about describing the network configurations that will be used to compare the architectural complexity required to deliver security.

1.8.1 The Internet

The layer model used by The Internet allows only one high-level configuration and this is what will be measured. The configuration is described in Section 1.6. While the

[30] See `http://www.pouzinsociety.org/`

physical transmission technologies differ, the layers of functionality are fixed. Many separate physical links managed by separate data link layers all connected via a global network and transport layer. See Figure 1·1.

1.8.2 The RINA Internet

Though the RINA design does not limit the configuration of layers, for the purpose of close comparison, we will consider RINA deployed to mimic the existing Internet. The physical links represent the lowest layer and will be referred to as *0-DIF*. Above the physical layer will be a logical link DIF. In our example network, there is one of these per link. They will be referred to as *1-DIF*s. Above the collection of 1-DIFs will be a single global network, the *2-DIF*. The 2-DIF is roughly equivalent to layers 3 and 4 of the Internet model. See Figure 1·3.

Chapter 2

ANALYSIS

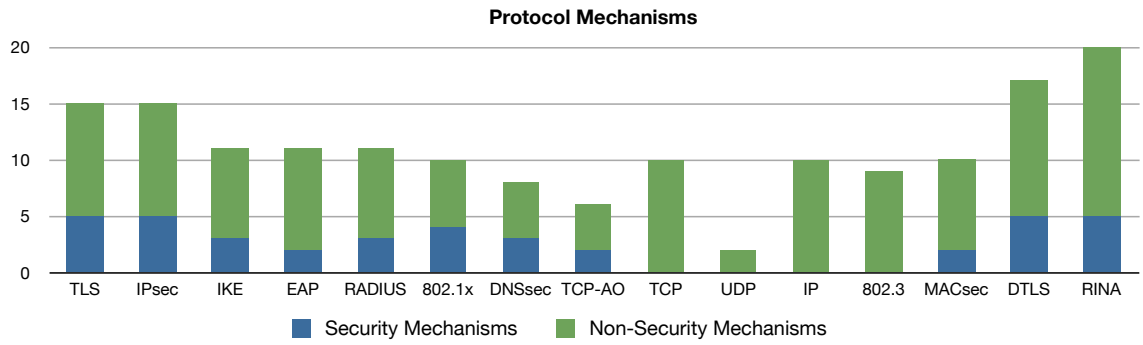

Figure 2·1: Network Mechanism Summary

2.1 Security protocols of the current architecture

Each layer of the internet is intended to be self-contained. Each layer provides security mechanisms in many cases as a separate sub-layer that includes a set of distinct security mechanisms. Physical Layer security (physical shielding, spread spectrum radio transmission, physical access to networking equipment, etc.), while important, is outside of the scope of this analysis. Each of the physical networking technologies rely upon different security protocols in their data-link layer implementations: IEEE 802.1x, 802.11i, RADIUS, EAP, etc. The network layer includes: IPSec, IKE, etc. The transport layer relies on: TLS, DTLS, etc. Applications rely on additional protocols for authentication such as: RADIUS, EAP, DNSsec, etc. Many Application

Mechanism	TLS	IPsec	IKEv2	EAP	RADIUS	802.1x (EAPOL)	DNSsec	TCP-AO	TCP	UDP	IP	802.3	802.1ae (MACsec)	DTLS	RINA
Delimiting	1	1	1	1	1	1	1		1	1	1	1	1	1	1
Initial State Synchronization	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Policy Selection	1	1	1	1	1	1	1	1	1		1	1	1	1	1
Addressing		1				1					1	1	1		1
Flow or Connection Identifier	1	1	1	1	1	1			1		1	1	1	1	1
Relaying				1	1		1				1				1
Multiplexing	1	1		1	1	1					1			1	1
Ordering									1					1	1
Fragmentation / Reassembly	1	1							1		1	1		1	1
Combination/Separation	1										1			1	1
Data Corruption	1	1	1				1	1	1		1	1	1	1	1
Lost & Duplicate Detection	1	1	1	1	1			1	1			1	1	1	1
Flow Control			1	1					1						1
Retransmission / Acknowledgement			1	1	1				1			1	1	1	1
Compression	1	1												1	1
Authentication	1	1	1	1	1	1	1	1						1	1
Access Control	1	1	1	1	1	1								1	1
Integrity	1	1				1	1	1					1	1	1
Confidentiality	1	1	1		1	1							1	1	1
Nonrepudiation	1	1					1							1	1
Activity															
Total Mechanisms	15	15	11	11	11	10	8	6	10	2	10	9	10	17	20
Security Mechanisms	5	5	3	2	3	4	3	2	2	0	0	0	2	5	5
Non-Security Mechanisms	10	10	8	9	8	6	5	4	10	2	10	9	8	12	15

Table 2.1: Network Mechanism Detail

Layer	Protocol
Application	DNSsec, EAP, RADIUS
Transport	TLS, TCP-AO, DTLS
Network	IPsec, IKEv2
Data Link	802.1x (EAPOL), 802.11i (WPA2), 802.1ae (MACsec)

Table 2.2: Sample of Internet Security Protocols

protocols even roll their own security: SNMPv3, HTTP, SSH, etc. Let us now examine the more prominent protocols in some detail. Each of the protocols will be examined for both the security and non-security mechanisms that it provides.

2.1.1 Data-link Layer

As already mentioned, there are numerous specific link layer security protocols. In our comparison we will be using IEEE 802.1x and IEEE 802.1ae.[1]

IEEE 802.1x

Overview IEEE 802.1x is a standard that defines the protocols and mechanisms that should be used to provide port based access control for IEEE 802 LANs. (IEEE, 2010) The standard specifies the use of the Extensible Authentication Protocol (EAP) along with IEEE 802.1ae (MACsec) and the Remote Authentication Dial in User Service (RADIUS) protocol. (Aboba et al., 2004; IEEE, 2006; Ringney et al., 2000) Specifically it defines the EAP method EAP Over LAN (EAPOL). Ports controlled by 802.1x mechanisms can restrict access to physical 802 LAN ports to authorized hardware and facilitate negotiation of cryptographic keys for use with MACsec. This is done by defining several Virtual LANs (VLAN)s, an unauthenticated VLAN used as a staging point to authenticate connecting hardware and assigning those devices to one or more other VLANs after authentication and evaluation of access control rules.

[1] IEEE 802.11i is similar to IEEE 802.1x but for wireless links.

Mechanisms IEEE 802.1x uses the following networking mechanisms: Delimiting, Initial State Synchronization, Policy Selection, Addressing, Flow or Connection Identifier, Multiplexing, Authentication, Access Control, Integrity, and Confidentiality.

IEEE 802.1ae (MACsec)

Overview The IEEE 802.1ae standard defines the necessary mechanisms required to deliver confidentiality and integrity protection to 802 LAN technologies.(Aboba et al., 2004) The standard defines additional PCI that can be used within 802 LAN frames in a transparent way so that protected traffic can flow across hardware that is not MACsec capable. Confidentiality and integrity protection can be provided on point-to-point links as well as for broadcast traffic.

Mechanisms MACsec uses the following networking mechanisms: Delimiting, Initial State Synchronization, Policy Selection, Addressing, Flow or Connection Identifier, Data Corruption, Lost & Duplicate Detection, Retransmission / Acknowledgement, Integrity, and Confidentiality.

2.1.2 Network Layer

Our comparison will include the use of IPsec and its associated key exchange protocol IKE.

Internet Protocol Security (IPsec)

Internet Protocol Security (IPsec) is intended to add peer-entity and data-origin authentication, integrity, payload confidentiality, and partial traffic-flow confidentiality functionality to the network. (Kent and Seo, 2005) It is defined by the IETF in a series of RFCs, the main ones being RFC 4301 Architecture, RFC 4302 Authentication Header, and RFC 4303 Encapsulating Security Payload. IPsec also relies upon the

Internet Key Exchange (IKE) protocol defined in RFC 5996 which will be examined separately in a later section. IPsec is technically a layer 3 protocol as it is intended to protect IP traffic directly.

IPsec is based around simplex associations between endpoints. Two way communication requires two security associations (SA). The protocol has two security payloads, *Authentication Header* AH, and *Encapsulating Security Payload* ESP. (Kent, 2005a; Kent, 2005b) AH provides only data origin authentication and can provide some integrity assurance if used with an HMAC. ESP can provide the same services as AH, but can also include confidentiality services. ESP used with a null cipher is effectively AH reducing the need for deploying AH. ESP can also provide tunnel mode where all traffic between two endpoints is encapsulated with an additional set of IP headers (between the tunnel endpoints) and IP packets traveling within the tunnel get additional protection because their IP headers can be encrypted along with their payloads which cannot happen when ESP is employed directly to those individual flows. Keys for confidentiality and integrity based authentication are exchanged using the Internet Key Exchange (IKE) protocol which in turn relies upon Public Key Infrastructure (PKI) certificates.

Mechanisms IPsec provides the following eleven networking mechanisms: Delimiting, Initial State Synchronization, Policy Selection, Addressing (tunnel mode), Flow or Connection Identifier, Multiplexing (in tunnel mode), Data Corruption Detection, Authentication, Access Control, Integrity, and Confidentiality.

Internet Key Exchange Protocol (IKE)

Overview: The Internet Key Exchange protocol, defined in RFC 5996, is the IPSec mechanism for exchanging keys to be used for both encryption and authentication.(Kaufman et al., 2010) The protocol is extensive and includes many features

and mechanisms traditionally provided by other layers like error and flow control. The protocol has its own packet format and runs over UDP. In addition to key exchange, it also natively provides authentication services via PKI or pre-shared secret. IKE can also integrate with the Extensible Authentication Protocol (EAP) for authentication. The purpose of IKE is to automate the establishment of IPSec Security Associations (SAs). IKE is a request/response protocol. It is engaged when a packet arrives that matches one of the policies in the Security Policy Database (SPD), and no SA has yet been established. Once IKE establishes the appropriate SA, the incoming packet is then forwarded via the newly established SA. IKE essentially implements enrollment for IP.

Mechanisms IKE makes use of the following networking mechanisms: Delimiting, Initial State Synchronization, Policy Selection, Flow or Connection Identifier, Data Corruption, Lost & Duplicate Detection, Flow Control, Retransmission / Acknowledgement, Authentication, Access Control, and Confidentiality.

2.1.3 Transport Layer

Several security protocols map to the Internet transport layer. Transport Layer Security (TLS), Transmission Control Protocol - Authentication Option (TCP-AO), and Datagram Transport Layer Security (DTLS) will be examined here.

Transport Layer Security (TLS)

Overview: The Transport Layer Security protocol, defined in RFC 5246, runs on top of TCP and provides authentication (peer-entity, data-origin), confidentiality, and message integrity services.(Dierks and Rescorla, 2008) It also includes a key exchange protocol for negotiating cipher suites, hash and signature algorithms. It is a formalization of the Secure Sockets Layer (SSLv3) protocol. TLS uses its own record

protocol consisting of individual records that can be encrypted, integrity protected, and authenticated. TLS has two sub-layers, an internal transport layer that uses the record protocol, and a layer above in which the handshake protocol, the alert protocol, the change cipher spec protocol, and the application data protocol all run. Both server and client authentication via public key certificates is supported. Only server authentication is required for a baseline secure connection. Completely anonymous connections can be protected only from casual eavesdropping and cannot reasonably be considered secure.

Mechanisms TLS makes use of the following networking protocols: Delimiting, Initial State Synchronization, Policy Selection, Flow or Connection Identifier, multiplexing, Fragmentation / Reassembly, Combination / Separation, Data Corruption, Lost & Duplicate Detection, Compression, Authentication, Access Control, Integrity, Confidentiality, and Nonrepudiation.

Transmission Control Protocol - Authentication Option (TCP-AO)

Overview TCP-AO is an extension to TCP defined in RFC 5925 and RFC 5926. TCP-AO is intended to add data origin authentication to TCP PDUs. (Touch et al., 2010; Lebovitz and Rescorla, 2010) The standard is intended to replace the older TCP-MD5 option by introducing a stronger message authentication code (MAC) definition that can help protect against interruption and replay attacks on TCP connections. The defined MACs make use of a shared symmetric signing key. While the distribution of the master keys is largely handled out of band, TCP-AO does provide some mechanisms for use of multiple keys during a connection as well as a way to derive new keys from a master key. TCP-AO introduces a new optional TCP header that includes a keyID and the MAC that protects the sequence number, the pseudo-IP header (subset of IP header fields), and the TCP segment data. Different

MAC algorithms are available and are defined in RFC 5926. The algorithms available for deriving keys are also defined in RFC 5926.[2]

Mechanisms TCP-AO uses the following networking mechanisms: Initial State Synchronization, Policy Selection, Flow or Connection Identifier, Data Corruption, Lost & Duplicate Detection, Authentication, and Integrity.

Datagram Transport Control Protocol (DTLS)

Overview DTLS is defined in RFC 6347 and can be succinctly summarized as TLS with ordering and retransmission over a datagram protocol (such as UDP). (Rescorla and Modadugu, 2012) The major differences between DTLS and TLS are that DTLS runs over protocols with unreliable delivery, and that it does not support stream ciphers (due to lack of guaranteed in-order delivery).

Mechanisms DTLS includes all the same mechanisms as TLS and adds Ordering and Retransmission / Acknowledgement to the list.

2.1.4 Application Layer

Extensible Authentication Protocol (EAP)

Overview EAP is defined in RFC 3748. It is a protocol and framework for supporting authentication operations. (Aboba et al., 2004) EAP is not restricted to use in any specific layer, but is typically used over a data link layer. EAP provides authentication via pluggable modules known as EAP Methods. The success or failure of authentications is delivered via EAP response packets. EAP can run directly over link layer protocols, so it includes several other networking mechanisms not related to authentication such as the detection of data corruption, retransmission, etc.; it

[2] Alternatively, IPsec with ESP-NULL could be used.

even has its own internal multiplexing model. EAP also participates in Access Control by virtue of authentication mechanisms returning failure when a subject can be successfully authenticated but does not have access to the requested resource.

Mechanisms Delimiting, Initial State Synchronization, Policy Selection, Flow or Connection Identifier, Relaying, Multiplexing, Lost & Duplicate Detection, Flow Control, Retransmission / Acknowledgement, Authentication, and Access Control

Remote Authentication Dial in User Service (RADIUS)

Overview As its name implies, the RADIUS protocol was designed to manage access to dial in equipment, though its use has expanded beyond the modem bank and has been deployed for use in general network access solutions. The protocol is defined in RFC 2865 and provides authentication, access control, and configuration services. RADIUS servers are often deployed with a separate authentication server to which they relay requests. (Ringney et al., 2000)

Mechanisms The RADIUS protocol includes the following networking mechanisms: Delimiting, Initial State Synchronization, Policy Selection, Flow or Connection Identifier, Relaying, Lost & Duplicate Detection, Retransmission / Acknowledgement, Authentication, Access Control, and Confidentiality.

Domain Name Resolution Security (DNSsec)

Overview RFC 4034 describes a set of extensions to the Domain Name System that provide a way to sign entries allowing DNS clients to validate the authenticity of those records and discard them if the signatures are invalid or have expired. (Arends et al., 2005b)

Mechanisms DNSsec provides the following networking mechanisms: Delimiting, Initial State Synchronization, Policy Selection, Relaying, Data Corruption, Authentication, Integrity, and Nonrepudiation.

Border Gateway Protocol Security

Overview There are several proposals and solutions for securing BGP. (Kent et al., 2000; van Oorschot et al., 2007) They all seem to boil down to using some sort of authenticated and optionally encrypted tunnel between the peer BGP routers. One solution is to use the authentication option for TCP (TCP-AO) defined in RFC 5925. This is the mechanism that will be used when evaluating BGP security against the test network.

2.2 RINA Security Modules

Module	Mechanisms
RINA Authentication	Authentication, Nonrepudiation*
RINA Access Control	Access Control
RINA SDU Protection	Confidentiality and Integrity

** on connection events if public key authentication methods are used*

Table 2.3: RINA Security Modules

2.2.1 Module Descriptions

RINA aims to provide each of the networking mechanisms only once per layer and the security mechanisms are no exception. RINA provides the 5 security mechanisms across 3 different modules: the authentication module, the access control module, and the SDU protection module. The key is that each of these mechanisms fit into the architecture of the layer at a single point. While different policies may be employed at each of those points, their place in the architecture remains the same. The authentication and access control modules work closely with CDAP while the SDU protection

module is employed after EFCP in the processing chain. (Day, 2009c; Bunch, 2010) The modules work together to achieve security. For example SDU protection policy data is exchanged using CDAP as part of enrollment. That key material can then be used by the SDU protection module to provide confidentiality, data integrity, or both.

Authentication Module

RINA's Authentication Module plugs into CDAP. Section 4.2.3 of the CDAP spec mentions methods from simple name and password to stronger methods that may use asymmetric cryptographic techniques. It also indicates that private methods can be defined. This extensibility allows CDAP to meet authentication requirements for any given configuration. For example a vendor could supply an authentication module that makes use of one-time password tokens. The section also suggests that applications should generate a session key and use it to secure all or part of CDAP messages including authentication message exchanges. (Bunch, 2010)

Access Control Module

Access Control signaling in RINA is communicated via CDAP. The access control mechanism itself is done by the IPC manager. According to PNA this will typically be done via capabilities which will be evaluated against the requesting subject and the requested object. (Day, 2008) The access control mechanism provides some of the functionality that an application firewall might in the Internet model. Local processes can be prevented from allocating outbound flows just as remote processes can be prevented from allocating inbound flows.

SDU Protection Module

The SDU protection module, sometimes referred to as the PDU protection module, is where confidentiality and integrity mechanisms are applied. SDU protection can be applied to the N-DIF PCI, the payload of one or more N+1 SDUs, or both. It is important to point out that when SDU protection is enabled on the entire PDU, it is completely opaque to the N-1 DIF IPC processes; they cannot even see the N-DIF address where the PDU is to be delivered. All any N-1 IPC process knows is that it must transmit the PDU (now an SDU) to the other end of the flow bound to the port-id the SDU was received on.

2.2.2 RINA's Security Story

Management messages that are passed within the DIF are done via the CDAP protocol. CDAP connections are authenticated according to the policy selected in the authentication module. Requests made via CDAP are subject to access control enforced by the IPC process servicing the request. SDU protection happens just before the PDU is passed down to layer N-1 (which could be the physical layer) and can include confidentiality, data integrity, or both for PDUs. Key material is also exchanged via CDAP. In concert these modules contribute to DIFs being securable containers. (Small, 2011) It is important to note that RINA requires no additional non-security mechanisms to deliver these functions within a DIF.

2.3 Security Comparison of RINA and the Internet

In this section a brief comparison of the approach to security will be given for each of the architectures. The results of measuring the test network for flows, protocols, distinct networking mechanisms, and number of total networking mechanism instances will be reviewed. Following this some additional analysis of the results will be pre-

sented showing that the data set supports the thesis that RINA can deliver security mechanisms with less complexity than the Internet Protocol Suite.

2.3.1 Differences in Approach to Security

The difference in approaches to security between the Internet Protocol Suite and RINA can be succinctly described. To add security to an "Internet" network, entire protocols are added to the stack. To add security to a RINA network, policies are changed in the existing protocols.

The Internet protocol suite aims to separate the different functions of networking, including security, into separate protocols that can be layered together as needed to deliver the required networking mechanism. The protocols operate independently and do not (or should not) share state. As a result some protocols must include redundant mechanisms. For example IKE must provide its own flow control because it operates over UDP.

RINA has separated mechanism from policy and so security mechanisms just plug into the existing protocols without requiring additional redundant non-security mechanisms. In addition, RINA protects N-DIF PDUs in their entirety as they cross the N-1 DIF boundary. Even the PCI (addresses, flow-ids, etc.) can be protected from the layer below.

2.3.2 Measuring Flows, Protocols, and Mechanisms

We will now discuss the results of measuring the example network for flows, protocols, and mechanisms as we secure each layer. The network we will be modeling is the same one we used as an example in sections 1.6.2 and 1.7.2. See Figure 1·2. We will look at each architecture separately in eight steps. First we will establish a baseline of a basic data flow with no security explicitly enabled (A). We will then add Dynamic Routing (B), followed by Name Resolution (C). Next we will begin securing the layers

starting with the links (D). After the links we will create an encrypted tunnel between the browser and server gateway routers (to mimic securely connecting two separate campus networks across a provider network) (E). Next up, we will discuss securing the transport traffic between the browser and server (F). Following that we address integrity of routing and directory data (G), capping the list off with the application protocol security (H).

Internet Results

This section will discuss the measurement results of the version of our example network running the Internet Protocol Suite. Table 2.4 gives a summary of the flows, protocols, and mechanisms used at each stage.

	Description	Mechanism Detail	Protocols	Flows	Non-Sec Mech	Non-Sec Inst	Sec Mech	Sec Inst	Total Mech	Total Inst
A	Data Flow (Static Routes)	802.3(6) + IP(1) + TCP(1) + HTTP(1)	4	9	29	74	0	0	29	74
B	Dynamic Routes	IP(3) + TCP(3) + BGP(3)	1	9	0	60	0	0	0	60
C	Name Resolution	IP(1) + UDP(1) + DNS(1)	2	3	2	12	0	0	2	12
D	Link Security (MACsec, EAPOL, RADIUS)	EAPOL(6)+MACsec(6) + RADIUS(6)	3	12	22	132	9	54	31	186
E	Tunnel Between Gateways (IPsec, IKE)	IPsec(1) + IKEv2(1) + UDP(1)	2	3	18	20	8	8	26	28
F	Transport Security (TLS)	TLS	1	1	10	10	5	5	15	15
G	Route (BGP+TCP-AO) and Directory security (DNSsec)	TCP-AO(3) + DNSsec	2	0	9	17	5	9	14	26
H	App Security	HTTP Auth	0	0	0	0	1	1	1	1
	Total		**15**	**37**	**90**	**325**	**28**	**77**	**118**	**402**

Table 2.4: Internet Results for Secure HTTP Traffic

A: Basic Data Flow - no security per se Our example network has six physical links each with IEEE 802.3 protocols running over them. Above that IP, TCP, and HTTP (the application protocol) are in use. We will ignore the specific mechanisms used by HTTP and other application protocols though they do add complexity to the mix. Figure 2·2 shows our starting point.

As Table 2.4 shows, there are six IEEE 802.3 flows, one IP flow, one TCP flow,

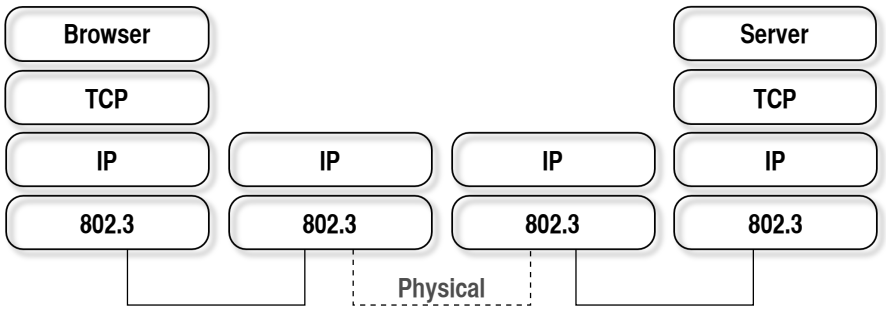

Figure 2·2: Internet Model Basic Data Flow Layers

and one HTTP flow. That's 4 protocols to start out with.[3] Using the results discussed in section 2.1 and summarized in Table 2.1 we see that this tallies up to 29 distinct non-security mechanisms and in total 74 instances of those mechanisms. No security mechanisms are in use.

B: Dynamic Routes So far static routes have been assumed for the routers. Adding dynamic routing on the internet model requires an additional application protocol. In this case BGP was chosen. BGP runs over TCP/IP and therefore additional flows between the three routers are needed. In this case it means three instances of BPG, TCP, and IP respectively, nine flows and 60 additional non-security mechanism instances. Though an additional protocol, BGP, has been added, we will omit its mechanisms as we did for HTTP.

C: Name Resolution Like static routing, the example network has thus far relied upon static name resolution. Adding dynamic name resolution on the Internet requires deploying DNS. The DNS protocol runs over UDP. Only the browser DNS look up of the IP address of the server will be considered, meaning a single additional

[3]Note several protocols have been omitted including ARP and DHCP each of which add additional mechanisms and flows.

flow for DNS, UDP, and IP must be considered. This adds three new flows with 12 additional non-security mechanism instances. IP mechanisms are already covered, but 2 additional distinct non-security mechanisms must be accounted for from UDP. Mechanisms from DNS itself will be omitted.

D: Link Security So far we have not actually added any security services; it is now time to do so. Authentication, access control, confidentiality, integrity, and non-repudiation[4] are added to each of the links in the data-link layer. Of the many available choices, IEEE 802.1x using EAPOL and MACsec (IEEE 802.1ae) with supporting RADIUS server will be deployed. This adds 12 flows, 132 non-security mechanism instances, and for the first time, 54 security mechanism instances. Figure 2·3 illustrates where the additional protocols fit into the stack. The additional protocols bring 22 non-security and 9 security mechanisms to the total of distinct mechanisms being used.

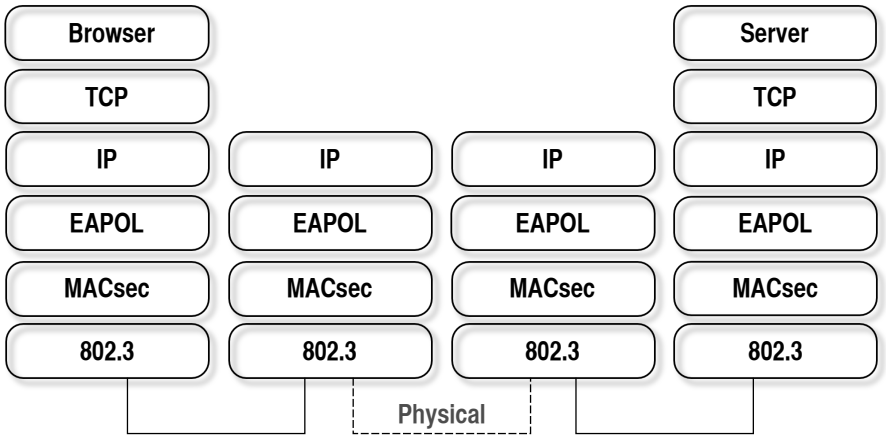

Figure 2·3: Internet Link Security Layers

[4]on connection events

E: Tunnel Between Gateways This step may seem strange on the small example network. An encrypted tunnel is established at the network layer between the gateway router for the browser and the gateway router for the server. In the example network this does not seem necessary because the two routers are connected by a physical link and the link is already protected. For the sake of argument, let us imagine that the two devices are actually connected by several other routers not under our control, for example a protected channel between two geographically separated campuses of a single organization.

To accomplish this in the Internet world we will use IPsec with an ESP tunnel. This requires us to configure a security association between the two gateway routers (a new flow) and since we want this link to be able to rotate keys for maximum protection, we will also deploy IKE. As we see in Table 2.4, this means adding three additional flows, one for the tunnel, and one each for IKE and UDP. This adds 20 non-security mechanisms and 8 security mechanisms to the total instances being employed. IPsec and IKE also add 18 non-security and 8 security mechanisms to the total distinct mechanism count. Figure 2·4 shows the changes in the protocol stack. To illustrate the distinction between the management protocols and the application being secured (HTTP), the diagram shows IKE and UDP separately from HTTP and TCP, but they are actually the same rank. Distinct mechanisms from UDP are not included here since they were previously accounted for in stage C.

F: Transport Security Moving up the stack, we secure the transport connection, the end-to-end channel between the browser and server. For the Internet suite TLS is used.[5] A single TLS flow is added to the mix resulting in a new protocol, ten new non-security mechanisms, and five additional security mechanisms. Since there

[5]An additional IPsec tunnel could have also been chosen, but TLS was selected due to its more common use in this scenario.

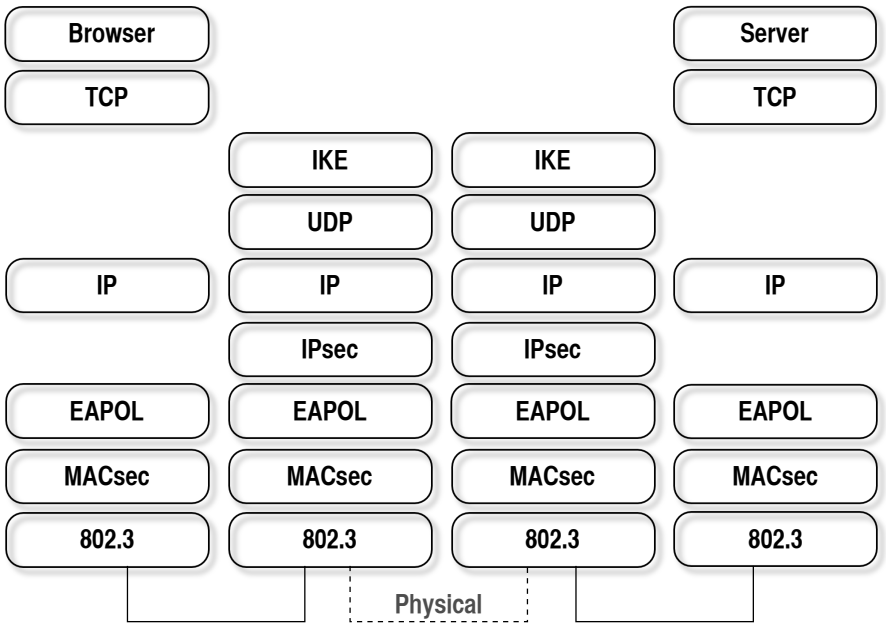

Figure 2·4: Internet Network Layer Security

is only one flow, these numbers are added to the totals for distinct mechanisms and mechanism instances. It is interesting to note that TLS adds nearly a complete additional transport layer between the existing transport and application layers with lack of ordering, flow control or retransmission.[6] This addition is reflected in Figure 2·5.

G: Management Data Security (Routing, Directory, etc). In recent years attention has been paid to securing management data on the net.(Heffernan, 1998; Kent et al., 2000; Arends et al., 2005a; van Oorschot et al., 2007; Touch et al., 2010) Protections for BGP and DNS in particular have been designed and deployed to guard against rogue updates. Adding these into the mix in our model involves enabling the TCP Authentication Option (TCP-AO) and using the techniques defined in DNSsec to authenticate and ensure integrity of the directory data found in the DNS system.

[6]This shouldn't be surprising since TLS is a formalization of the Secure Sockets Layer (SSL).

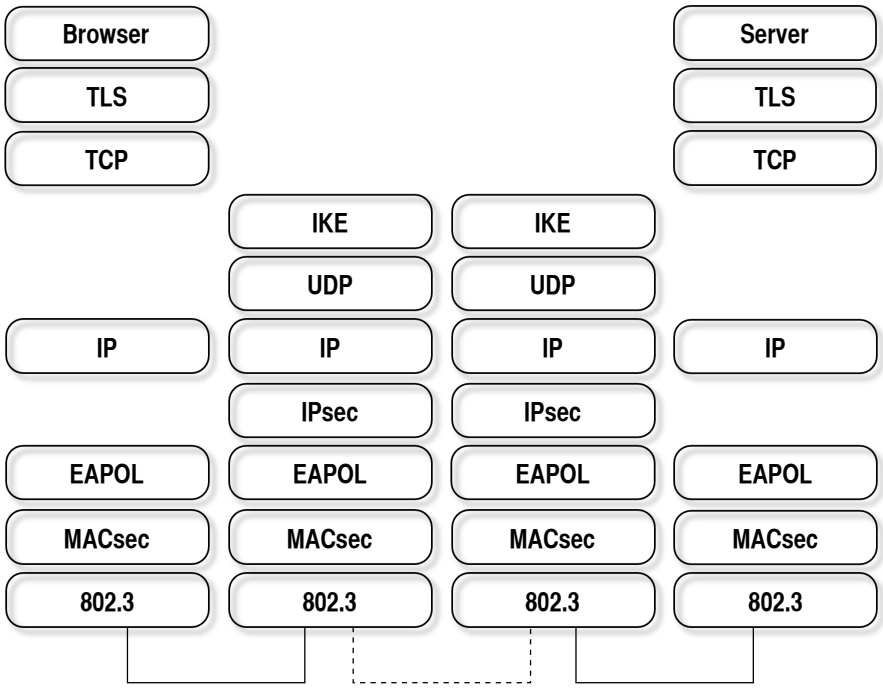

Figure 2·5: Internet Full Security Stack

If confidentiality is omitted, this does not require any additional flows. However, it does cause an additional 17 non-security and 9 security mechanism instances to be added. These 26 mechanisms provide authentication and integrity (for DNS) but not confidentiality. If confidentiality is required then additional mechanisms must be applied via TLS, DTLS, or IPsec. These are not reflected in the current totals. TCP-AO and DNSsec bring with them 9 distinct non-security and 5 distinct security mechanisms.

H: Application Security At the top of our stack is HTTP, the example application protocol. HTTP supports only one security mechanism, authentication. One might balk at this statement saying that HTTPS also supports strong authentication, integrity protection, and confidentiality, possibly access control as well. However, HTTPS is simply a synonym for "use HTTP over SSL/TLS", and TLS has already been considered. To reflect the addition, a single security mechanism is added to the total. Applications may also provide their own security via various techniques, but those are not considered here because they would be the same in the RINA case and so would be a wash in the results.

Totals As Table 2.4 shows, to meet all of the requirements by securing each layer using stages A through H, the Internet model uses: 15 protocols, 37 flows, 90 distinct non-security mechanisms with 325 total instances, 28 distinct security mechanisms with 77 total instances. That is 118 total distinct mechanisms and 402 total instances of those mechanisms.

RINA Results

This section will discuss the measurement results of the version of our example network running RINA. Table 2.5 gives a summary of the flows, protocols, and mecha-

nisms used at each stage.

	Description	Mechanism Detail	Protocols	Flows	Non-Sec Mech	Non-Sec Inst	Sec Mech	Sec Inst	Total Mech	Total Inst
A	Data Flow (Static Routes) (CDAP & EFCP)	1-DIF no sec(6) + 2-DIF no sec(6) + HTTP(1)	3	13	15	195	0	0	15	195
B	Dynamic Routes	*(included in DIF function)*	0	0	0	0	0	0	0	0
C	Name Resolution	*(included in DIF function)*	0	0	0	0	0	0	0	0
D	Link Security (1-DIF)	(1-DIF Security)(6)	0	0	0	0	5	30	5	30
E	Tunnel Between Gateways (2-DIF)	(BackboneDIF)(1)	0	1	0	15	0	5	0	20
F	Transport Security (N/A)	(App Sec DIF)(1)	0	1	0	15	0	5	0	20
G	Signed Route / Directory Data	*(included in DIF function)*	0	0	0	0	1	16	1	16
H	App Security	HTTP Auth	0	0	0	0	1	1	1	1
	Total		**3**	**15**	**15**	**225**	**7**	**57**	**22**	**282**

Table 2.5: RINA Results for Secure HTTP Traffic

A: Basic Data Flow The example network being used has 6 physical links.[7] A DIF will be created to manage each of these links.[8] These are denoted as *1-DIF* in the layer diagram shown in Figure 2·6. Running over those 6 DIFs is a single "Network" or "Internet" DIF. This is denoted as *2-DIF* in the diagram. Finally above that is the application.

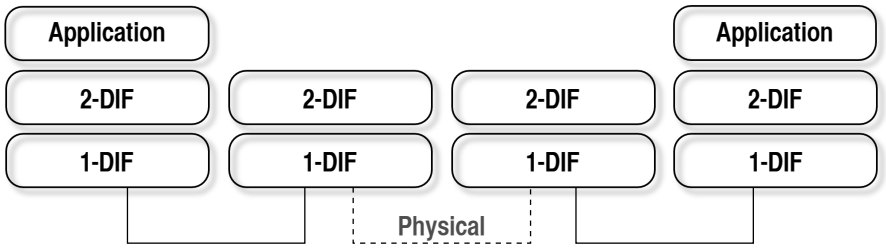

Figure 2·6: RINA Basic Data Flow Layers

There is an application connection with a flow established across each of the links and between each of the nearest neighbors in the 2-DIF. There is also a flow allocated

[7]The dedicated DNS server is unnecessary under RINA, but the additional host and link will be included for closer comparison with the Internet model.

[8]For multi-access media like wireless and shared segment Ethernet only one DIF is needed for all of the links.

in the 2-DIF for the application traffic. That is 13 flows in total. Since RINA encapsulates all of the network services into the two protocols CDAP and EFCP and those are used in all DIFs, with HTTP running over the network that is only 3 protocols in use. Also, since RINA takes care not to repeat mechanisms within the DIF and security is not enabled on these DIFs, we start with only 15 distinct non-security mechanisms. Interestingly RINA begins with a substantially higher number of mechanism instances than was measured on the Internet model. This is largely because of two factors: RINA creates additional flows between the nearest neighbors and as will be discussed in the next sections, is already providing dynamic routing and name resolution. These additional services require additional protocols in the Internet model.

B: Dynamic Routes Under RINA, nothing additional is required for dynamic routing since the existing CDAP connections handle routing updates. IPC processes have a Resource Information Base (RIB) that holds their state and policy data. The RIB is a partially replicated distributed database and its content is requested and pushed between IPC processes as needed using CDAP. No new flows or mechanisms are introduced.

C: Name Resolution Like dynamic routing information, name resolution is provided in the functionality of the DIF. When an N+1 process calls the allocation primitive, it does so with the name of the requested destination, the IPC process does the name resolution and the address of the destination within the N-DIF is not divulged to the calling process. No new flows or mechanisms are introduced. Naming updates are exchanged using CDAP.

D: Link Security To secure the links in the 1-DIFs all RINA must do is change policies to require use of the five security mechanisms over the existing flows. CDAP can start authenticating new connections and control access based on the authenticating entity. If public key cryptography is employed for the authentication, then non-repudiation is provided on the connection event. The SDU protection module provides confidentiality and integrity of PDUs. If the SDU protection policy includes a signed MAC, then non-repudiation is provided for each PDU, but this would likely have performance penalties. More likely the policies will stick with some sort of keyed digest. Unlike the Internet which must deploy an additional pseudo-layer using additional protocols and establish flows within that layer, RINA requires adding only the five security mechanisms to the existing flows. Figure 2·7 illustrates that the structure of the layers is unaffected by the addition of security mechanisms in the 1-DIFs.

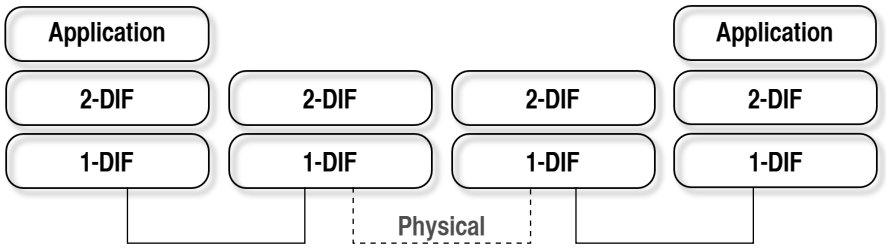

Figure 2·7: RINA Link Security Layers

Enabling security in the 1-DIFs adds five distinct security mechanisms but notably zero non-security networking mechanisms. There are six links so the instance count for the security mechanisms is proportional at 30.

E: Tunnel Between Gateways In order to establish the equivalent of a tunnel between the two gateways, a new DIF will be created to include only the gateway routers in question. With the example of connecting geographically separated cam-

puses, this makes sense. The new DIF is placed between the 1-DIFs and 2-DIF. Figure 2·8 denotes this new DIF as *Backbone-DIF*.

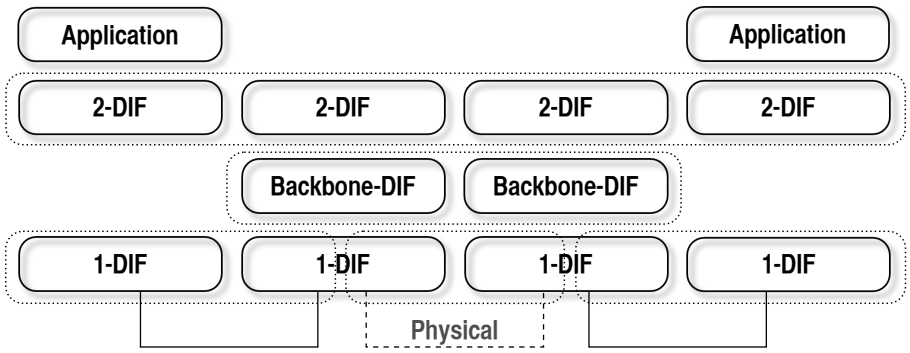

Figure 2·8: RINA with Organization Backbone Layer

Adding the new DIF causes us to add a single flow with the full compliment of 20 mechanism instances (15 non-security and five security), but adds no new distinct mechanisms to the total. Another interesting thing to note here is that to add additional sites to this Backbone-DIF requires only a single flow for each campus; all campuses need not even be directly connected because PDUs will be forwarded accordingly. In the Internet model, IPsec security associations and tunnels may need to maintained between each of the campuses.

F: Transport Security Applications built for RINA may actually be able to handle confidentiality and integrity using the same mechanism as the IPC processes running below them in their supporting DIF, but for comparison sake, we will assume HTTP is unmodified and our network will provide transport security between the browser and server. To accomplish this another DIF will be created that sits above the 2-DIF. The new DIF will only have two members, the IPC processes on the browser and server hosts. The addition of this DIF, denoted as *App Sec DIF*, is depicted in

Figure 2·9.[9]

For simplicity's sake, we will say that this new DIF adds a single flow with the 20 mechanisms, though in actuality fewer may be needed. For example, addressing and relaying are not strictly needed in a DIF that only has two members. Either way, no new distinct mechanisms, security or otherwise, are added to the total.

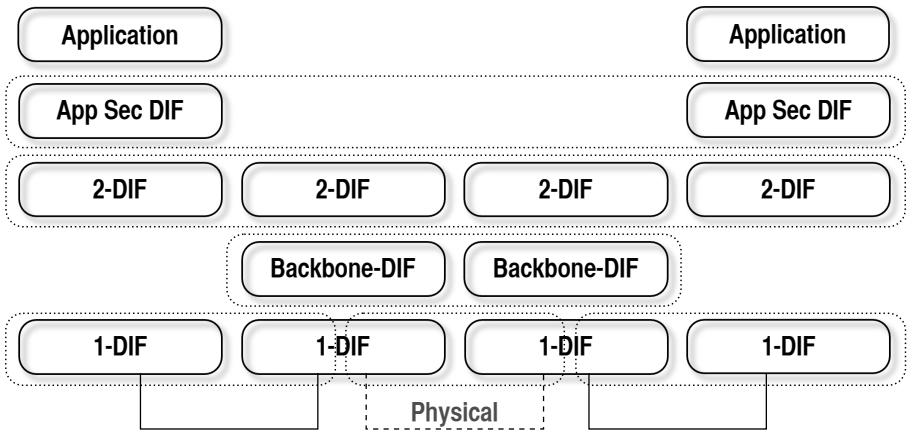

Figure 2·9: RINA Full Security Stack

G: Management Data Security (Routing, Directory, etc). There is only one protocol across which management data flows, CDAP, and CDAP data travels across the established flows. RINA has not yet specified a mechanism for ensuring the integrity of the data while it lives in the RIB, but keep in mind, aside from a rogue piece of code directly running on the router, in order for a malicious directory or routing entry to make it into the RIB, a previously authenticated IPC process will have to have sent it via an encrypted, integrity protected connection. If signing (non-repudiation) is also in effect for messages, the rogue also cannot deny having done so. Even so, a single mechanism (non-repudiation) will be accounted for similar to the

[9]Interestingly unlike TLS, this DIF could be used by multiple browsers and the server where if there were two browsers and a server in the DIF the browsers could actually exchange traffic directly within the DIF without the HTTP server itself having to do relaying.

additional signature records added to DNS as part of DNSsec. Sixteen instances of this mechanism will also be recorded to account for the mechanism running in each of the IPC processes. Note that, unlike the Internet, this includes all layers.

H: Application Security Again, applications using RINA have the opportunity to benefit from using the existing RINA security mechanisms; this would eliminate the need for the *App Sec DIF*. However we are mimicking the Internet model and are using HTTP unmodified, therefore a single distinct mechanism and a single instance will be accounted for.

Totals As Table 2.5 shows, to meet all of the requirements by securing each layer using stages A through H, the RINA model uses: three protocols, 15 flows, 15 distinct non-security mechanisms with 225 total instances, and seven distinct security mechanisms with 57 total instances. That is just 22 total distinct mechanisms and 282 total instances of those mechanisms.

2.3.3 Observations

Figure 2·10 shows the comparison of the results found in Table 2.4 and Table 2.5. This figure shows a comparison of the number of new protocols, flows, mechanisms, or instances, respectively, for the Internet vs. RINA. These are represented by the bar graph elements with the Internet on the left and RINA on the right for each stage, A-H. In addition, line graph elements depict the running total for each of the variables, again, Internet vs RINA. The left column of charts shows protocols and distinct mechanisms broken down by non-security, security, and total. The right column does the same for flows and instances of mechanisms.

The left column of charts in Figure 2·10 supports the hypothesis that RINA can deliver security with less complexity than the Internet model, which requires more

Figure 2·10: Flows, Protocols, and Mechanisms

mechanisms at each stage as the difference in the cumulative total mechanisms is greater at each stage, while the RINA total stays nearly flat. The right-hand column supports the thesis less clearly. This is due to the decision to break up the initial stages being measured into the base data flow and other management protocols (A-C). If we start with stage D when the security mechanisms are applied at the link layer, the right column also supports the claim that RINA is less complex in terms of flows, protocols, distinct networking mechanisms, and instances of those mechanisms. Figure 2·11 shows the same data zoomed into stages D-H. Here the support for the hypothesis is more pronounced.

Now one might notice that link security is a bit of an outlier, especially for the Internet model. The hypothesis is that RINA will always be at most as complex as the Internet when adding security. Figure 2·12 and Figure 2·13 show the results with link security taken out of the picture. Again, the left-hand columns support the thesis somewhat strongly. It seems reasonable to state that RINA has less complexity for delivering security in terms of the number of distinct mechanisms. The answer is not as clear for complexity in terms of the number of running protocol machines. Initially RINA seems to have more mechanism instances than does the Internet. This is due to RINA providing more functionality in a single package. Remember RINA provides functionality such as dynamic address assignment and name resolution in each DIF. The Internet requires the deployment of additional protocols and flows for this such as ARP and DHCP.[10] Again we can account for some of the differences in base operation by taking base mechanisms and just looking at the mechanisms needed to add security. Figure 2·13 does this. Once we account for the differences in initial overhead, the right-hand column starts to support the thesis again with one anomaly. The chart showing the number of additional security mechanism instances

[10]The link layer often relies upon a global address space such as MAC addresses rather than having addresses be dynamically assigned.

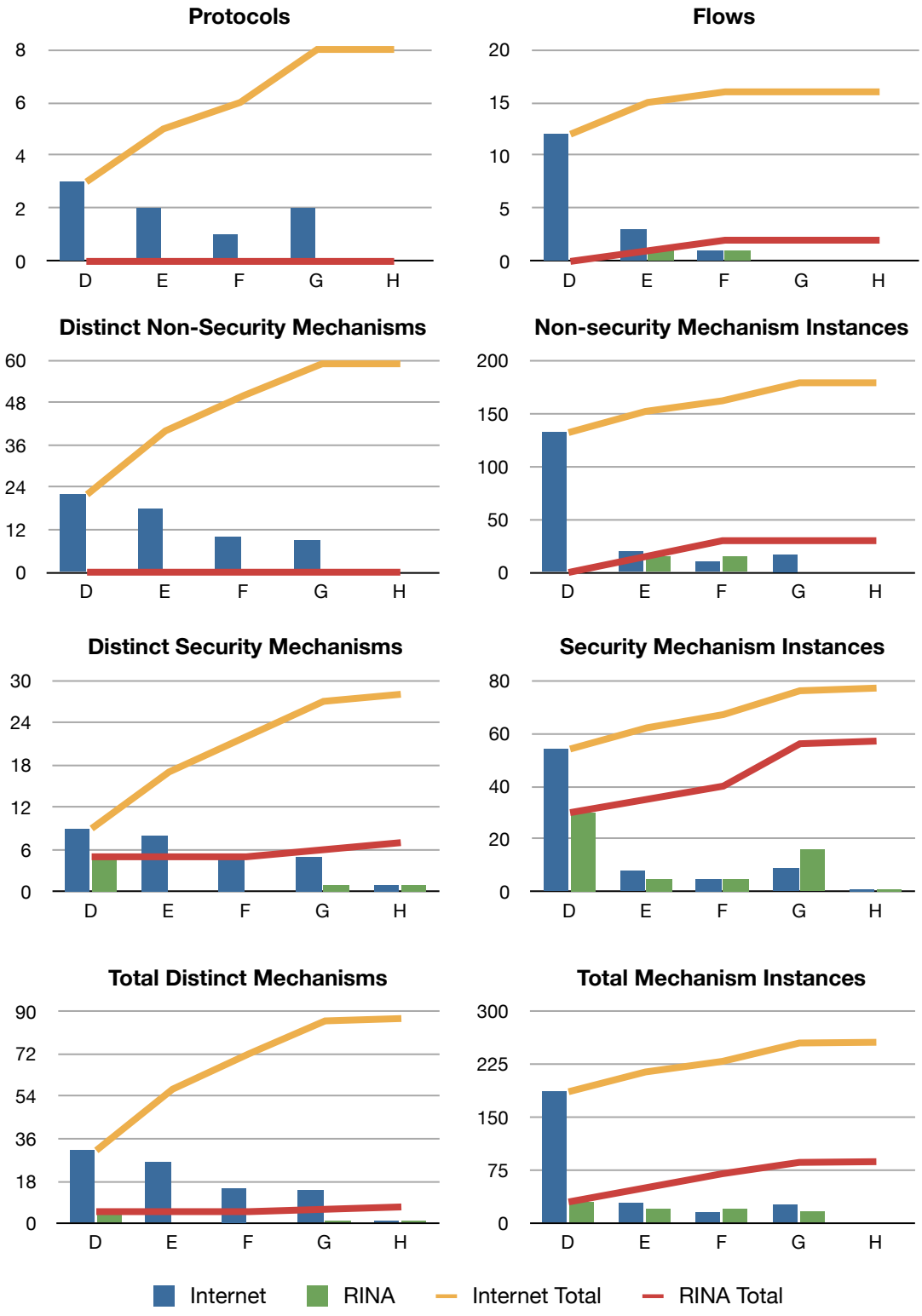

Figure 2·11: Additional Flows, Protocols, and Mechanisms to Add Security

shows RINA requiring more instances than the Internet. Close inspection shows that this is because again, RINA is providing more functionality. The data has RINA security directory entries in all of the IPC processes, specifically that IPC processes are signing entries. If instead the RINA network is deployed so that the directory functionality is only signed at one authoritative IPC process, similarly to how the Internet test network was measured, then this chart looks substantially different as one can see in Figure 2·14, along with the effect on the total mechanism instance chart.

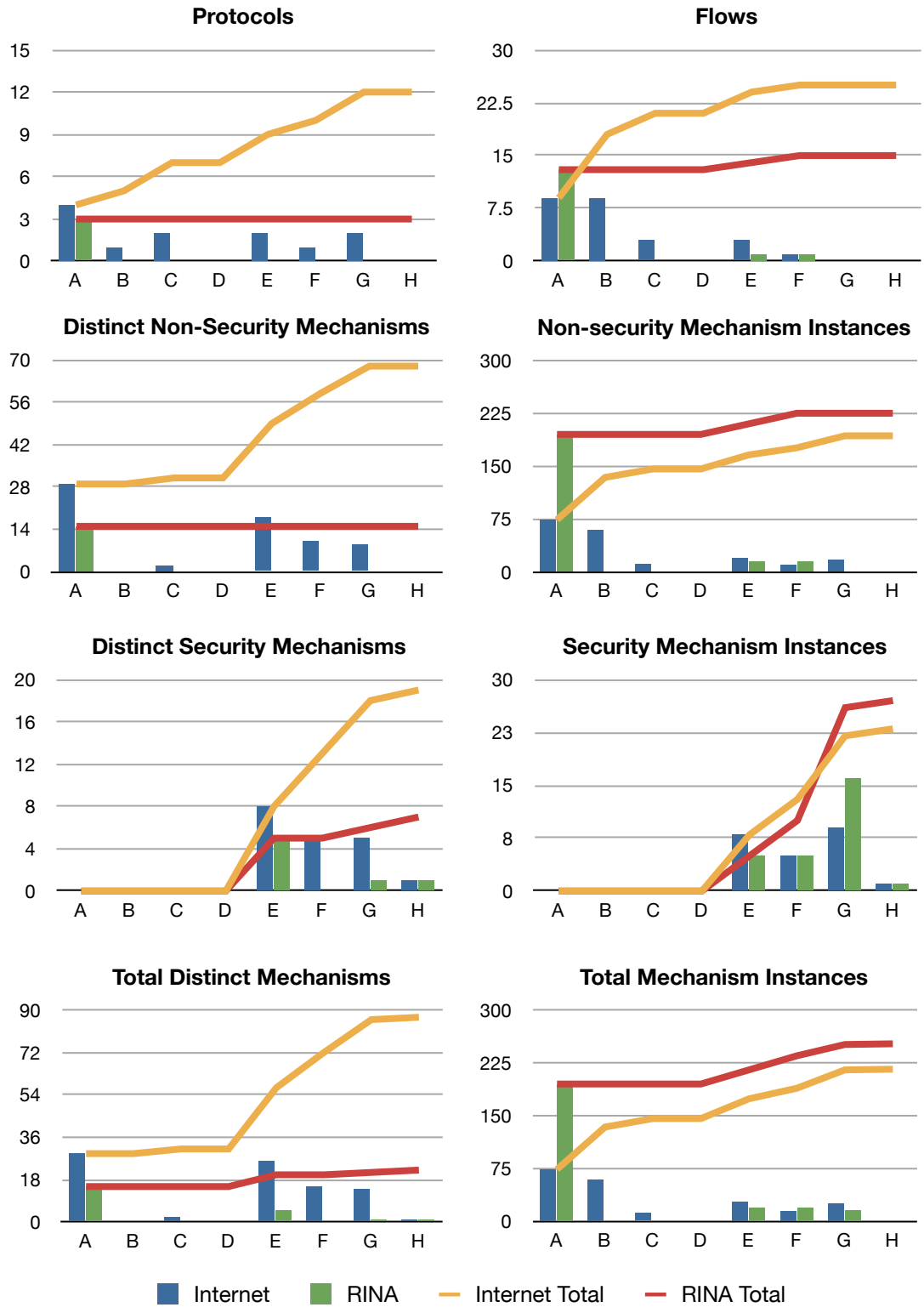

Figure 2·12: Flows, Protocols, and Mechanisms (No Link Security)

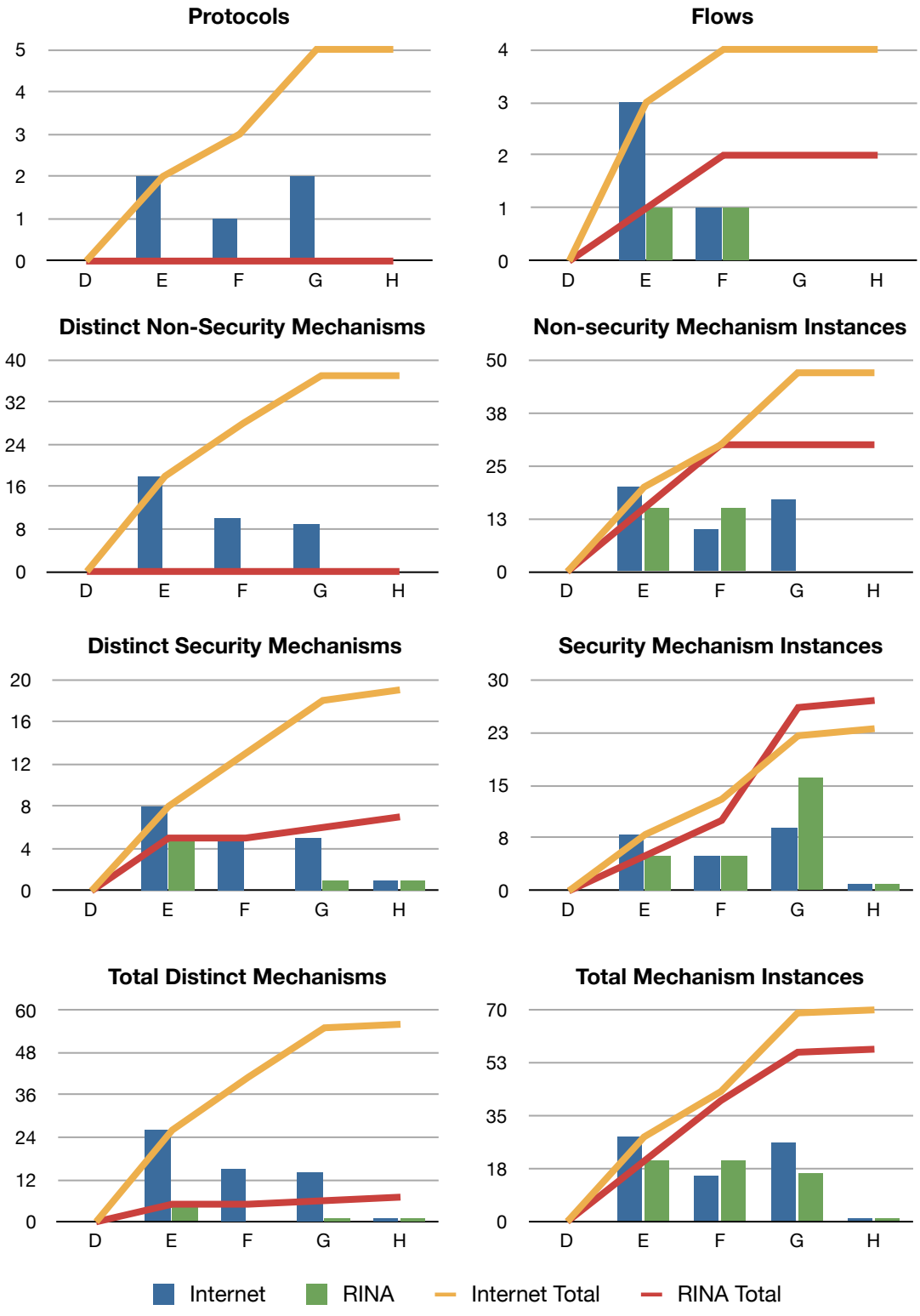

Figure 2·13: Additional Flows, Protocols, and Mechanisms to Add Security (No Link Security)

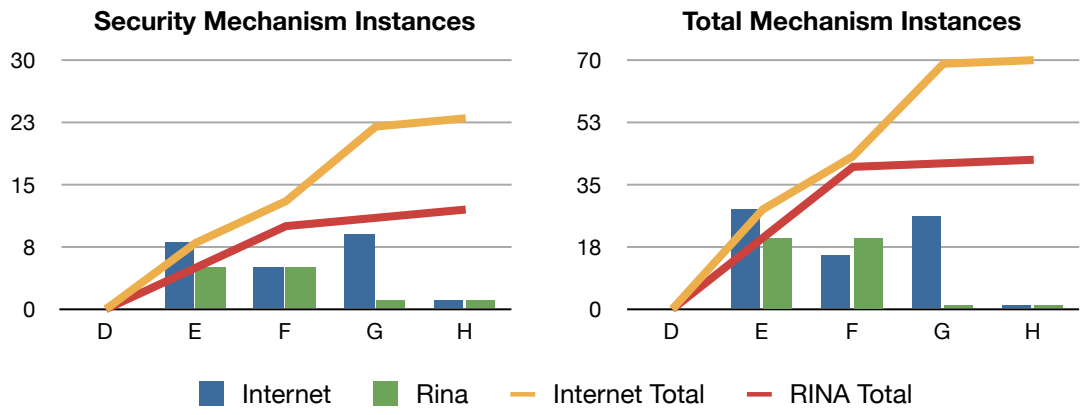

Figure 2·14: Adjustment for Dedicated RINA Directory

Chapter 3

CONCLUSION

3.1 Thesis Review

This thesis sets out to show that RINA networks can deliver on security requirements with less complexity than is currently possible using the Internet Protocol Suite. A selection of commonly used Internet security protocols were reviewed for what security and non-security mechanisms they include. These were tallied and used to measure a test network at several stages as security requirements were applied to each of its layers. This exercise was done once using the Internet Protocol Suite and again using RINA. The results were analyzed and shown to be in strong support of the thesis statement. Based on the evidence presented, RINA seems to be able to deliver on security requirements with substantially less complexity in terms of number of protocols, required number of flows, and especially the required number of distinct mechanisms. RINA is also reasonably less complex in terms of number of active instances of networking mechanisms, especially when the link layer is being secured.

3.2 Recommendations and Topics for Further Study

Key Exchange As was mentioned in section 1.7, the RINA reference model and specification is still under active development. There has been substantial work done in the development of the security protocols for the Internet Protocol Suite. RINA can benefit from much of this work. In particular the key exchange algorithms used by

TLS and IKE may be very applicable when stripped of the overhead of flow control, etc.

SDU Protection The TLS protocol most closely matches the types of flows used between RINA IPC processes. As this thesis has shown, there is a 2:1 ratio of non-security to security mechanisms in the TLS protocol. CDAP and EFCP should be able to provide those non-security mechanisms. It would be interesting to see if the remaining parts of TLS could be reworked to provide the basis of the SDU protection module, and possibly the CDAP authentication module.

Re-keying and Traffic Multiplexing Extensive parts of the Internet protocols that provide integrity and confidentiality deal with how to do in-line re-keying of secure connections. Because of the separation of lower boundary connection-ids from upper boundary port-ids, in-line re-keying may be completely unnecessary under RINA. The flow allocator could simply monitor key lifetime and when needed, open a new flow to the same destination negotiating the key as part of the new allocation. Once a new flow is established and secured, the IPC process can rebind the port-id with the new connection-id and start forwarding the N+1 SDUs across the new flow. The old flow can then simply be deallocated once the Maximum Packet Lifetime timer expires for the last PDU sent across it.

Management Data Integrity RINA DIFs promote reducing scope of networks, but for sufficiently large DIFs, the level of trust in management data may begin to weaken. It seems that some mechanism for integrity protecting routing and directory information may be needed, even if for small DIFs it may be overkill. Two interesting topics arise here: to determine what specific mechanisms for protecting the integrity of RIB records are appropriate, and to determine if there is some measurable threshold

of DIF size where the level of trust in the management data is weak enough to warrant use of further integrity protection.

How many layers need SDU protection? Sufficiently large networks, like major carrier networks, may be organized into many DIFs: Customer, Campus, Metro, Regional, Backbone, etc. Some additional research is needed into whether full SDU protection is needed in all of the layers or if it is sufficient to apply security over a subset of those DIFs. It would also be interesting to explore which are the most effective layers to apply security to (e.g. at the very top and across physical links).

Physical Link Encryption Hardware link encryption/decryption devices exist and are in use by governments and military organizations. It may be interesting to research if RINA reduces the need for these devices, at least for single access media like copper and optical links, since the SDU protection module provides the ability to encrypt and protect the integrity of entire PDUs including addresses, etc.

References

Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and Levkowetz, H. (2004). Extensible Authentication Protocol (EAP). RFC 3748.

Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. (2005a). DNS security introduction and requirements. RFC 4033.

Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. (2005b). Resource records for the DNS security extensions. RFC 4034.

Braden, R. (1989a). Requirements for internet hosts – application and support. RFC 1123.

Braden, R. (1989b). Requirements for internet hosts – communication layers. RFC 1122.

Bunch, S. (2010). CDAP - Common Distributed Application Protocol reference. Unpublished.

Cerf, V. and Postel, J. (1978). *Specification of Internetwork Transmission Control Program. TCP Version 3.* University of Southern California.

Crocker, S. (1969). Host software. RFC 1.

Day, J. (2008). *Patterns in Network Architecture: A Return to Fundamentals.* Pearson Education, Upper Saddle River, N.J.

Day, J. (2009a). Patterns in network architecture: Recursive IPC network architecture: The reference model: Basic concepts and distributed applications. Unpublished.

Day, J. (2009b). Patterns in network architecture: Recursive IPC network architecture: The reference model: Distributed interprocess communication. Unpublished.

Day, J. (2009c). Patterns in network architecture: Recursive IPC network architecture: The reference model: Operations. Unpublished.

Day, J. (2010a). Details of CDAP PDU flow. Unpublished.

Day, J. (2010b). How CDAP works. Unpublished.

Day, J. (2010c). Patterns in network architecture: Recursive IPC network architecture: The reference model: Interesting configurations. Unpublished.

Day, J. (2011a). Patterns in network architecture: Recursive IPC network architecture: Enrollment specification. Unpublished.

Day, J. (2011b). Patterns in network architecture: Recursive IPC network architecture: Flow allocator specification. Unpublished.

Day, J., Matta, I., and Mattar, K. (2008). Networking is IPC: A guiding principal to a better internet. In *Rearch'08 - Re-Architecting the Internet*, Madrid, SPAIN. Co-located with ACM CoNEXT 2008.

Dierks, T. and Rescorla, E. (2008). The transport layer security TLS protocol version 1.2. RFC 5246.

Heffernan, A. (1998). Protection of BGP sessions via the TCP MD5 signature option. RFC 2385.

IEEE (2006). IEEE 802.1ae - IEEE standard for local and metropolitan area networks - media access control (MAC) security.

IEEE (2010). IEEE 802.1x - IEEE standard for local and metropolitan area networks - port-based network access control.

Jacobs, S. (2011). *Engineering Information Security: The Application of Systems Engineering Concepts to Achieve Information Assurance*. Wiley & Sons, Inc., Hoboken, NJ.

Kaufman, C., Hoffman, P., Nir, Y., and Eronen, P. (2010). Internet Key Exchange Protocol Version 2 (IKEv2). RFC 5996.

Kent, S. (2005a). IP Authentication Header. RFC 2402.

Kent, S. (2005b). IP Encapsulating Security Payload (ESP). RFC 4303.

Kent, S., Lynn, C., and Seo, K. (2000). Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4).

Kent, S. and Seo, K. (2005). Security architecture for the internet protocol. RFC 4301.

Lebovitz, G. and Rescorla, E. (2010). Cryptographic algorithms for the TCP Authentication Option (TCP-AO). RFC 5926.

Rescorla, E. and Modadugu, N. (2012). Datagram transport layer security version 1.2. RFC 6347.

Ringney, C., Willens, S., Rubens, A., and Simpson, W. (2000). Remote Authentication Dial In User Service (RADIUS). RFC 2865.

Small, J. (2011). Threat analysis of recursive inter-network architecture distributed inter-process communication facilities. Unpublished.

Touch, J., Mankin, A., and Bonica, R. (2010). The TCP authentication option. RFC 5925.

van Oorschot, P., Wan, T., and Kanakis, E. (2007). On interdomain routing security and pretty secure BGP (psBGP). *ACM Transactions on Information System Security*, 10, 3(11).

Watson, R. (1981). Delta-t protocol specification. Lawrence Livermore National Library.

ABOUT THE AUTHOR

Jeremiah Small was born near the advent of TCPv3 in 1977. He has over 16 years experience as a software designer and engineer. He received his Bachelors of Science in Computer Science from the University of Massachusetts at Amherst in May 2000. His professional career began in 1996 writing embedded system firmware for video editing and home automation controller equipment for Vidan Corporation. Following this, he spent a year building and enhancing large scale web-based learning systems with the Center for Computer Based Instructional Technology at the University of Massachusetts at Amherst. In February of 2000, he joined Concord Communications (later purchased by Computer Associates) to build and enhance key components of *e*Health^{TM}, an industry leading integrated network fault and performance management system. Jeremiah received a graduate certificate in Information Security in 2009 and expects to complete his Masters of Science in Computer Science from Boston University in May 2012. He is currently employed by RSA, The Security Division of EMC.

Jeremiah Small
P.O. BOX 637
Westford, MA 01886
jdsmall@bu.edu