

# **Information Protection in Content-centric Networks**

by

**Christopher C. Lamb**

B.S., Mechanical Engineering, New Mexico State University, 1994

M.S., Computer Science, University of New Mexico, 2002

DISSERTATION

Submitted in Partial Fulfillment of the  
Requirements for the Degree of

Doctor of Philosophy  
Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

October, 2012

©2012, Christopher C. Lamb

# Dedication

*This dissertation is dedicated first and foremost to my wife, who dealt with me, bills, kept our children fed and otherwise kept a roof over my head while I was preoccupied with writing this dissertation. She took care of everything while working full time in Santa Fe and enabling my higher education addition. I could never have accomplished this without her support, love, and encouragement. She was key in keeping me on some kind of schedule to finish. Without her gentle reminders regarding the importance of actually **completing** my dissertation, I would likely be submitting this some time in 2014.*

*This is dedicated to you Missy.*

# Acknowledgments

I would like to thank my advisor, Professor Greg Heileman, for reading and re-reading my papers and dissertation, and guiding my research. Truly widely educated with a deep understanding of an astounding number of fields, his guidance and suggestions were invaluable in helping me craft this dissertation.

Thanks to my committee for sitting through my defence, reading this dissertation, and supporting me as well as the larger computer science and computer engineering disciplines at the University of New Mexico (UNM). We indeed have world-class academics here at UNM, and I was so very lucky to have some of them working with me, teaching me, and helping me with my research over the (many) years.

I would also like to thank everyone in the Informatics Lab. Their feedback, support, and encouragement was so very helpful in my completion of this dissertation. Especially Pramod Jamkhedkar, my long time collaborator, and Sinan al-Saffar. It's especially inspiring when you see someone you work with actually complete a Ph.D program.

# **Information Protection in Content-centric Networks**

by

**Christopher C. Lamb**

## **ABSTRACT OF DISSERTATION**

Submitted in Partial Fulfillment of the  
Requirements for the Degree of

Doctor of Philosophy  
Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

October, 2012

# **Information Protection in Content-centric Networks**

by

**Christopher C. Lamb**

B.S., Mechanical Engineering, New Mexico State University, 1994

M.S., Computer Science, University of New Mexico, 2002

Ph.D., Computer Engineering, University of New Mexico, 2012

## **Abstract**

Information-centric networks have distinct advantages with regard to securing sensitive content as a result of their new approaches to managing data in potential future internet architectures. These kinds of systems, because of their data-centric perspective, provide the opportunity to embed policy-centric content management components that can address looming problems in information distribution that both companies and federal agencies are beginning to face with respect to sensitive content. This work addresses the current state of the art in both these kinds of cross-domain systems and information-centric networking in general. It then covers other related work, outlining why information-centric networks are more powerful with regard to usage management. Then, it introduces a taxonomy of types of policy-centric usage managed information-centric network systems and an associated methodology for evaluating the individual taxonomic elements. It finally delves into experimental evaluation of the various defined architectural options and presents results of comparing experimental evaluation with anticipated outcomes. This work is the end

result of four conference papers (IEEE Cloud 2011, ACM DRM 2011, IEEE SoSE '11, and IEEE Cloud 2012), one invited talk (DoDIS 2012), and research performed using this system is currently submitted to two journals for publication (IJ-Closer and Dynamic Work).

# Contents

<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xiv</b>
<b>Glossary</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Confidentiality and Integrity Models . . . . .	4
1.2 National and International Standards . . . . .	6
1.3 Current Solutions . . . . .	7
1.3.1 NSA, Filtered . . . . .	8
1.3.2 NSA, Services . . . . .	9
1.3.3 Raytheon . . . . .	10
1.3.4 Booz   Allen   Hamilton . . . . .	11
1.4 Information-centric Networking . . . . .	14
1.5 Other Related Work . . . . .	16



## Contents

<b>2</b>	<b>Usage Management in Information-centric Networks</b>	<b>18</b>
2.1	Characteristics . . . . .	21
2.2	Taxonomies of Usage Management Overlay . . . . .	22
2.2.1	$\phi$ -level Overlay Systems . . . . .	24
2.2.2	$\alpha$ -level Overlay Systems . . . . .	25
2.2.3	$\beta$ -level Overlay Systems . . . . .	27
2.2.4	$\gamma$ -level Overlay Systems . . . . .	28
2.3	Taxonomic Analysis . . . . .	29
2.3.1	Characteristics of Policy-centricity . . . . .	30
2.3.2	Information-centric Structure . . . . .	33
2.4	Analysis . . . . .	36
<b>3</b>	<b>Experimental Configuration</b>	<b>37</b>
3.1	Overlay Implementation Concerns . . . . .	38
3.2	Initial Prototype Implementation . . . . .	40
3.3	Inter-Provider Cloud Configuration . . . . .	43
3.4	Inter-Cloud Architecture . . . . .	45
3.5	Experimental Structure . . . . .	52
3.5.1	Initial Seed Information . . . . .	55
3.5.2	Policies and Attributes . . . . .	57
3.6	Primary Interfaces and Mappings . . . . .	60

## *Contents*

<b>4</b>	<b>Experimental Results and Conclusions</b>	<b>65</b>
4.1	Hierarchical Networks . . . . .	66
4.2	Non-Hierarchical Networks . . . . .	72
4.3	Removing Network Effects . . . . .	76
4.4	Conclusions . . . . .	77
	<b>References</b>	<b>79</b>

# List of Figures

1.1	NSA Legacy Notional Architecture Model . . . . .	8
1.2	NSA Service-Oriented Model . . . . .	10
1.3	Ratheon Model . . . . .	10
1.4	Booz   Allen   Hamilton Model . . . . .	12
2.1	Taxonomy ( $\phi$ ) . . . . .	24
2.2	Taxonomy ( $\alpha$ ) . . . . .	26
2.3	Taxonomy ( $\beta$ ) . . . . .	27
2.4	Taxonomy ( $\gamma$ ) . . . . .	29
3.1	Simulation Logical Configuration . . . . .	37
3.2	Physical Simulation Configuration . . . . .	42
3.3	Overall System Architecture . . . . .	47
3.4	Node Architecture . . . . .	49
3.5	Router Architecture . . . . .	50

## *List of Figures*

4.1	Hierarchical Results from Amazon . . . . .	68
4.2	Hierarchical Results from Rackspace . . . . .	69
4.3	Hierarchical Results from Comcast . . . . .	70
4.4	Non-Hierarchical Results from Amazon . . . . .	73
4.5	Non-Hierarchical Results from Rackspace . . . . .	74
4.6	Non-Hierarchical Results from Comcast . . . . .	75
4.7	Results from Requests to a Singe Node . . . . .	77

# List of Tables

2.1	Proposed Usage Management Taxonomy . . . . .	23
3.1	Supporting Components . . . . .	44
3.2	All Possible Attributes for Usage Management Decisions . . . . .	58
3.3	Possible Attributes for Usage Management Decisions specific to Users .	59
4.1	Approach Evaluation Summary . . . . .	72

# Glossary

RDFa   Resource Description Framework – in – attributes

XDM   Extensible Metadata Platform

XML   eXtensible Markup Language

CCN   Content-Centric Network

ICN   Information-Centric Network

DONA   Data-Oriented Network Architecture

PSIRP   Publish-Subscribe Internet Routing Project

NetInf   Network of Information

# Chapter 1

## Introduction

Current enterprise computing systems are facing a troubling future. As things stand today, they are too expensive, unreliable, and information dissemination procedures are just too slow. Current approaches to partitioning information in cross-domain scenarios are simply unable to migrate to cloud environments because of reliance on control of physical hardware to enforce information separation. The current approach of controlling information by controlling the underlying physical network is too cost ineffective to continue, and does not scale into shared datacenters. This leaves large government and commercial organizations concerned with avoiding the exposure of sensitive data in a very uncomfortable position, where they cannot continue doing what they have done, and cannot migrate to what everyone else is doing.

Generally, systems handling sensitive information still do not use current commercial resources as well as they could and use costly data partitioning schemes. Most of these kinds of systems are managed in house by the enterprise itself rather than exploiting lower cost cloud-enabled services. Furthermore, many of these systems have large maintenance loads imposed on them as a result of internal infrastructural requirements like data and database management or systems administration. In many cases networks containing

## *Chapter 1. Introduction*

sensitive data are separated from other internal networks to enhance data security at the expense of productivity, leading to decreased working efficiencies and increased costs.

These kinds of large distributed systems suffer from a lack of stability and reliability as a direct result of their inflated provisioning and support costs. Simply put, the large cost and effort burden of these systems precludes the ability to implement the appropriate redundancy and fault tolerance in any but the absolutely most critical systems. Justifying the costs associated with standard reliability practices like diverse entry or geographically separated hot spares is more and more difficult to do unless forced by draconian legal policy or similarly dire business conditions.

Finally, the length of time between when a sensitive document or other type of data artifact is requested and when it can be delivered to a requester with acceptable need to view that artifact is prohibitively long. These kinds of sensitive artifacts, usually maintained on partitioned networks or systems, require large amounts of review by specially trained reviewers prior to release to data requesters. In cases where acquisition of this data is under hard time constraints like sudden market shifts or other unexpected conditional changes this long review time can result in consequences ranging from financial losses to loss of life.

Federal, military, and healthcare computer systems are prime examples of these kinds of problematic distributed systems, and demonstrate the difficulty inherent in implementing new technical solutions. They, like other similar systems, need to be re-imagined to take advantage of radical market shifts in computational provisioning. New approaches to networking and information management present possible solutions to these kinds of problems by providing distributed information-centric approaches to data management and transfer.

Current policy-centric systems are being forced to move to cloud environments and incorporate much more open systems. Some of these environments will be private or hybrid cloud systems, where private clouds are infrastructure that is completely run and operated



## *Chapter 1. Introduction*

by a single organization for use and provisioning, while hybrid clouds are combinations of private and public cloud systems. Driven by both cost savings and efficiency requirements, this migration will result in a loss of direct control of computing resources by involved organizations as they attempt to exploit economies of scale and utility computing.

Robust usage management will become an even more important issue in these environments. Federal organizations poised to benefit from this migration include agencies like the National Security Agency (NSA) and the Department of Defense (DoD), both of whom have large installed bases of compartmentalized and classified data. The DoD realizes the scope of this effort, understanding that such technical change must incorporate effectively sharing needed data with other federal agencies, foreign governments, and international organizations [2]. Likewise, the NSA is focused on using cloud-centric systems to facilitate information dissemination and sharing [5].

Cloud systems certainly exhibit economic incentives for use, providing cost savings and flexibility, but they also have distinct disadvantages as well [44]. How to address these issues is an open research question. Organizations ranging from cloud service providers to the military are exploring how to engineer solutions to these problems, and to more clearly understand the trade-offs required between selected system architectures [3]. The problems themselves are wide ranging, appearing in a variety of different systems. Military and other government systems are clearly impacted by these kinds of trust and security issues, and they also have clear information sensitivity problems. This, coupled with the fact that these organizations have been dealing with these issues in one form or another for decades make them very well suited for prototypical implementation and study.

## 1.1 Confidentiality and Integrity Models

Current models currently in active use to support information confidentiality and integrity include the Bell-LaPadula model, the Biba model, the Clark-Wilson model, and the Brewer and Nash model. Of these, Bell-LaPadula and Brewer and Nash address information confidentiality, while Biba and Clark-Wilson address information integrity.

Bell-LaPadula was developed in the 1970's to address information confidentiality in the centralized, mainframe and minicomputer environments common in military installations of the day. It is essentially a mathematical state machine model that establishes rules with respect to how information can flow in stratified environments. It is established around the *Basic Security Theorem*, which essentially states that a system with a secure initial state and only secure transitions is guaranteed to terminate in a secure final state. It extends this theorem to establish four rules. The first is the *simple security rule*. The simple security property prohibits reading information from security levels higher than that occupied by the reader. The next two are the *\*-property rule* and the *strong \*-property rule*, which prohibit writing data to any security level less than that of the writer. The final property is the *ds-property*, or *Discretionary Security Property*. The ds-property allows a subject to pass permissions on to other subjects at the initial subject's discretion. This model also introduces the concepts of *dominance relations* and *tranquility*. Dominance relations exist within partially ordered sets of classification levels, where higher classification levels are said to dominate lower levels. This means that if a subject is cleared to access top secret material, that subject can also access secret and confidential material, as both secret and confidential material is considered to be less sensitive than top secret material. Tranquility limits object state changes after creation. Essentially, any object is created at a specific sensitivity level and that sensitivity level does not change with time [17].

The Brewer and Nash model is currently widely used in the financial services industry. Also referred to as the Chinese Wall Model, this model changes what subjects can access

## Chapter 1. Introduction

based on a subjects context. In this model, information is continually monitored so that no subject is allowed access to an object that may create a conflict of interest or other moral hazard. More generally, access to objects is dynamically evaluated based on the context of a given subjects previous object accesses. This enforces information confidentiality based on the context of information access [23].

The Biba model, developed in the late 1970's, addresses information integrity using a layered approach similar to that used by Bell-LaPadula, and was in fact developed to complement Bell-Lapadula's focus on confidentiality. Bell-LaPadula, features a hierarchy of information sensitivity that guides access decisions. Biba, on the other hand, is based on a hierarchy of information integrity rather than confidentiality. Biba also has a collection of specific properties that must be adhered to. The first, the *simple integrity axiom*, holds that a subject cannot access an object of a lower integrity level. The second, the *\*-integrity axiom*, likewise states that a subject cannot write information to higher integrity level. The third property, the *invocation property*, states that subjects cannot invoke objects at higher integrity levels either [19].

The David Clark and David Wilson presented their integrity model in 1987 to address business information integrity as Biba was thought to be more suitable for military use. Clark-Wilson uses five essential components. The first, *users*, are active subjects requesting access to objects. The second, *transformation procedures*, are operations allowed over objects. These general operations are any that may change the state of an object, like create, update, and delete operations, as well as simple read access. Systems are also required to have collections of *integrity verification procedures* which validate the state of managed objects. The object themselves have two distinct flavors, *constrained data items* and *unconstrained data items*. Constrained data items are those managed via trusted procedures and monitored via integrity verification procedures. Unconstrained items can be directly accessed. Access and management of data items are constrained by a set of nine rules, divided into sets of *certification* and *integrity* rules. In essence, these rules ensure

that all constrained data item access is through trusted procedures, and that the trusted procedures ensure that information is maintained in a valid state. Subjects are limited to sets of trusted procedures expressed by subject, procedure, object triples, while unconstrained data items can be accessed without limitation [24].

## 1.2 National and International Standards

Once the Bell-LaPadula and Biba models were developed and understood, the Department of Defense started another effort to more clearly categorize the security of information systems and related networks. This effort cumulated in what is now known as the Rainbow Series of books, specifically the *Trusted Computer System Evaluation Criteria*, known as the Orange Book, and the *Trusted Network Interpretation*, or Red Book.

The Orange Book addresses security of individual computer systems. It establishes a taxonomy of security levels at which systems can be classified. These levels range from systems with minimal security characteristics and stretch to formally verified computer systems. These levels establish such characteristics as separate administrator accounts, role-based security systems, data labelling, and the recognition of possible covert communications channels [?].

The Red Book deals with how networks protect the confidentiality, integrity, and availability of information. It addresses encryption, digital signatures, flow protection and confidentiality, data confidentiality, and infrastructure protection. Like the Orange Book, it is a framework established over a set of principles to facilitate network security. It discusses how subjects and objects need to be controlled and accounted for in computer networks just as they are within individual computer systems [?].

Today, the Rainbow Series has largely been supplanted by the International Organization for Standardization's Common Criteria. The Rainbow Series was seen as too rigid, and other

## *Chapter 1. Introduction*

standards too flexible and difficult to implement, leading to the development of the Common Criteria in 1993. The Common Criteria are based on Rainbow Series, the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC), and the Information Technology Security Evaluation Criteria (ITSEC). While the Orange Book examined a system based on a Bell-LaPadula-centric perspective, the Common Criteria evaluates computing systems based on a pre-established protection profile, designed to cover a specific security requirement. The common criteria also use a taxonomy to classify systems. This taxonomy has seven levels, again ranging from minimally tested systems that have some assured functionality to formally verified, designed, and tested information systems [?].

Other standards commonly referenced include publications from the National Institute of Standards and Technology, particularly from their 800-series of special publications. These publications cover everything from cloud computing security [?] to single computer systems [?] to web services [?].

### **1.3 Current Solutions**

The Unified Cross Domain Management Office (UCDMO) supports efforts to develop other specific solutions that have been presented over the past few years to handle this kind of information management. The National Security Agency set the standard in this area initially. In 2009, at a conference sponsored by the UCDMO, Booz Allen Hamilton (BAH) and Raytheon presented alternative notional architectures contrasting with current NSA-influenced approaches [39, 4, 22, 42].

The current standard architectural model in place and governed by the UCDMO to deal with these kinds of issues are guard-centric cross domain architectures. As we show in section 1.3, the thinking behind these system architectures has remained relatively static over the past 20 years. New thinking with regard to future internet architectures and usage

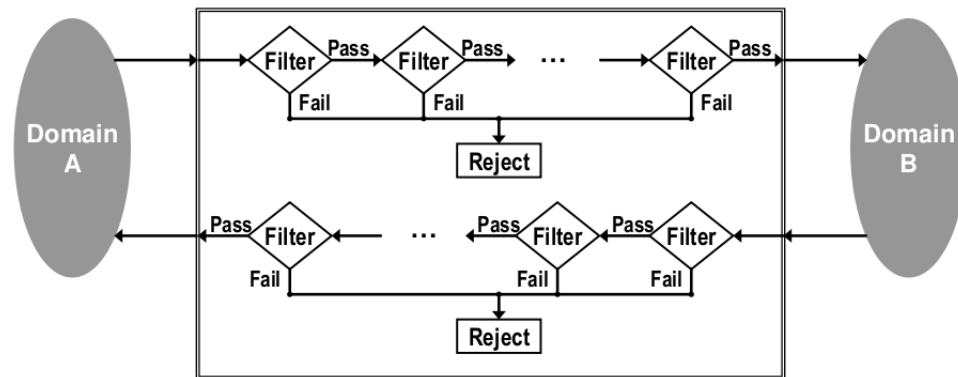


Figure 1.1: NSA Legacy Notional Architecture Model

management provide more powerful approaches to securing information as it flows through dynamic systems.

Current and near-future proposed solutions endorsed by the UCDMO include system architectures assembled by the NSA, Raytheon, and Booz | Allen | Hamilton (BAH). The NSA has been active in this area for decades as a logical extension of their role in signals intelligence collection and processing. Raytheon and BAH have been engaged over the past few years to provide an alternative voice and design approach to these kinds of systems, an effort met with limited success.

These cross-domain solutions are intended to enable sensitive information to easily flow both from a higher sensitivity domain to a lower sensitivity domain, and from lower to higher as well. They generally act over both primary data (say, a document) and metadata over that primary data as well. Note that in these system, in most cases, human intervention is still required to adequately review data prior to passing into lower security domains.

### 1.3.1 NSA, Filtered

The NSA conducted initial work in this area. Their standard-setting efforts culminated in a reasonable conceptual system architecture, using groups of filters dedicated to specific

## Chapter 1. Introduction

delineated tasks to process sensitive information [39].

In the scenario portrayed in Figure 1.1, *Domain A* could very well be a private cloud managed by the U.S. Air Force, while *Domain B* is a public operational network of some kind shared by coalition partners in a joint operation.

A system user attempts to send a *data package* consisting of a primary document and associated metadata from *Domain A* to *Domain B*. At some point, that submission reaches a *guard*, which contains at least one *filter chain*. Each filter chain then contains at least one *filter*. Individual filters can execute arbitrary actions over a submitted data package and have access to any number of external resources as required. At any point, a filter can examine the data package and reject it, at which point it will frequently wait for human review. If a filter does not reject a data package, it passes that package onto the next filter or submits it for delivery to Domain B.

### 1.3.2 NSA, Services

In recent years, the NSA has extended the legacy system architecture for cross-domain information sharing to exploit service-oriented computing styles [39]. Visualized in Figure 1.2, this model incorporates more modern conceptual elements and componentry.

In the view in Figure 1.2, we see on the left the *Global Information Grid*, or *GIG*. On the right, we have the *Distributed Service-oriented Cross Domain Solution*, or *DSCDS*. The GIG is not a truly open system — rather, it is a loosely coupled collection of computational services handing data at a variety of levels of sensitivity, federated to provide stakeholders timely access to relevant information [4]. The DSCDS is essentially the embodiment of the NSA’s cross-domain vision applied to service oriented computing. This model fuses various technology choices with previous cross-domain thinking.

Indicative of this more modern system design thinking, we have a variety of services and

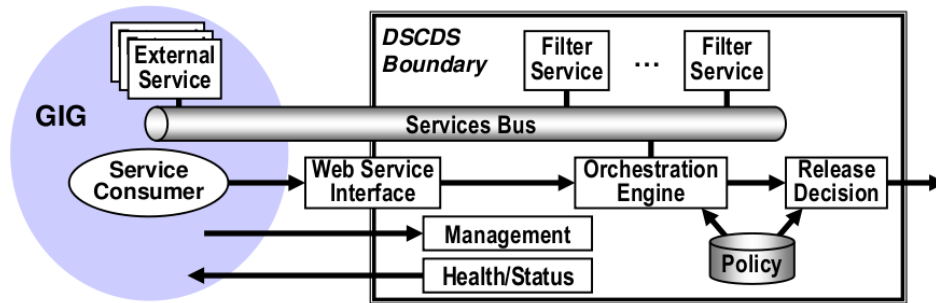


Figure 1.2: NSA Service-Oriented Model

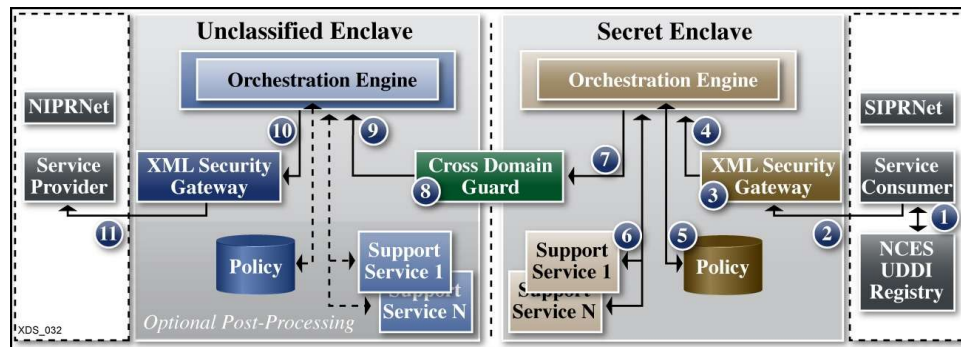


Figure 1.3: Ratheon Model

service consumers attached to a common service bus within the GIG. Within the DSCDS, we have groups of filters implemented as services inspecting transferred data when moved over the bus. Finally, all of this interaction is managed by a management interface and controlled by an orchestration engine accessing a centralized group of policies.

Note that here we have begun to access a common policy repository for various types of security metadata regarding primary data elements.

### 1.3.3 Raytheon

In the past few years, Raytheon has offered a new model for cross domain use influenced by the NSA service-oriented model [42].



## Chapter 1. Introduction

The model in Figure 1.3 is more grounded in the actual technical environment this kind of solution would be embedded within. Here, we have the Non-secure Internet Protocol Router Network (NIPRNet) as one domain, and the Secret Internet Protocol Router Network (SIPRNet) as the other. Here, NIPRNet is the lower security domain (lowside), and SIPRNet the higher security domain (highside). This particular view shows the motion of data from the high side (SIPRNet) to the low side (NIPRNet).

Here, a data request is submitted from SIPRNet first to the *XML Security Gateway* which calls into the *Orchestration Engine* for policy validation. The Orchestration Engine then coordinates calls into a *Policy Repository* as well as to a collection of external *Support Services*. Once rectified against these elements, the request is passed into the *Cross Domain Guard* which routes the request into the *Unclassified Enclave* in NIPRNet. Here, the request is passed directly through the lowside *XML Security Gateway*, without rectification, onto the *Service Provider*. The response from the Service Provider is then passed back to the requester via the inverse path.

This model also begins to use a centralized policy repository, just as the NSA Service Model. It also uses a single cross domain guard to transfer information from both the highside to the lowside, and vice-versa.

### 1.3.4 Booz | Allen | Hamilton

BAH submitted a competing model, also in 2009 [22]. In fact, both Raytheon and BAH presented their models under competitive contract to the UCDMO at the same conference, so the domain application is not coincidental.

Figure 1.4 embodies BAH's thinking with respect to cross domain information management. We have a *Domain A* as a high security domain, and *Domain B* as a low security domain. Here, we again have dataflow from the highside to the lowside through the cross domain management system.

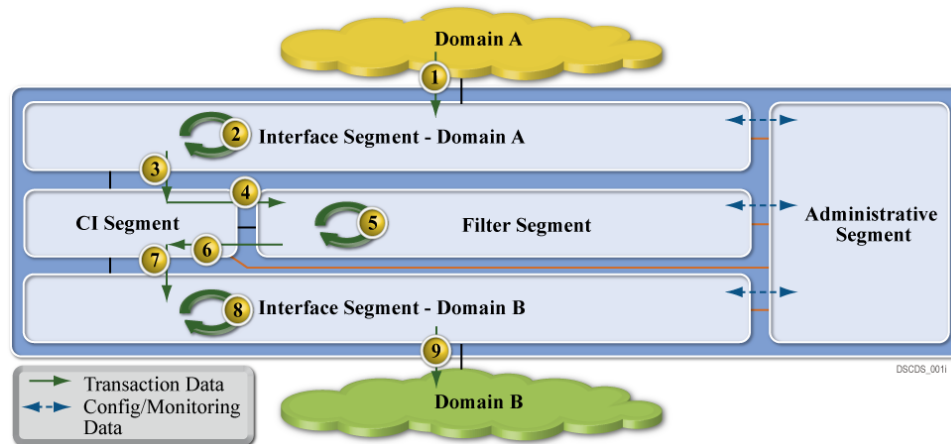


Figure 1.4: Booz | Allen | Hamilton Model

While not as detailed as the Raytheon proposal, this does have similar elements. Here, we data first travels from Domain A into the *Interface Segment for Domain A*, similar to the secret enclave used in the Raytheon model. From there, it moves into the *CI Segment*, which in turn submits the transferring data into the *Filter Segment*. From there, the package is moved into the *Interface Segment for Domain B*, and then onto *Domain B*. The *Administrative Segment* provides managment and oversight of the system as a whole.

Note the absence of specific policy-centric elements. This system is reliant on specific policy-agnostic content filters as well. These kinds of cross-domain solutions still have clear similarities, and in fact have not progressed far beyond the initial notions of how these kinds of systems should work. They still, for example, all use some kind of filter chaining mechanism to evaluate whether a given data item can be moved from a classified to an unclassified network. Both NSA models used filters explicitly, as did the BAH model. They all use a single guard as well, a sole point of security and enforcement, providing perimeter data security, but nothing else. In each of these current system architectures, users are only allowed to exchange one type of information per domain. The physical instantiations of these models are locked by operational policy to a single classification level. Users cannot, for example, have Top Secret material on a network accredited for Secret material. Finally,

## *Chapter 1. Introduction*

these models violate end-to-end principles in large service network design, centralizing intelligence rather than pushing that intelligence down to the ends of the system [26].

End-to-end principles are generally considered core to the development of extreme scale, distributed systems. Essentially, one of the key design decisions with respect to the early internet was to move any significant processing to system end nodes, keeping the core of the network fast and simple. Known as the end-to-end principles, this design has served the internet well, allowing it to scale to sizes unconceived when originally built. Current cross domain systems are placed at key routing points between sensitive networks. These locations are core to information transfer between systems and as a result violate the initial design principles upon which the internet was founded. There does exist some belief that end-to-end principles need to be modified to support future networks, but nevertheless, current cross domain systems still violate the basic ideas behind large, scalable networks by placing complex application-specific logic directly and only in the core of a given sensitive network [20].

Future systems will generally demonstrate decentralized policy management capabilities, infrastructural reuse, the ability to integrate with cloud systems, and security in depth. Policy management will need to be decentralized and integrated within the fabric of the system. The system is both more secure and resilient as a result, better able to control information and operate under stressful conditions. Multi-tenancy can lower costs and increase reliability and is furthermore a common attribute of cloud systems. An appropriately secured system facilitates integration of computing resources into multi-tenant environments. The ability to handle multi-tenant environments and to reliably secure both data at rest and data in motion leads to computational environments deployable in cloud systems. Finally, systems must operate under *all* conditions, including when they are under attack or compromise [48] and provide protection to sensitive data in depth.

## **1.4 Information-centric Networking**

Information-centric networking (ICN) is a new approach to internet-scale networks that shows promise with respect to decentralized, content-centric usage management, addressing scale and availability issues with current systems. In general, it takes extensive advantage of data locality, cache data aggressively, decouple information providers from consumers, and use a content-centric perspective in network design. The overriding goals of this approach include providing higher information availability through better network resilience and implementing systems that more closely reflect today's use, focusing on heterogeneous systems from mobile to static access [16]. Four of the leading projects implementing these ideas are Data-Oriented Network Architecture (DONA) [38], Content-centric Networking (CCN) [13, 9], the Publish-Subscribe Internet Routing Project (PSIRP) [15], and the Network of Information (NetInf) [14]. In general these projects and the thinking behind them is motivated by the belief that the current internet is not well suited to the way it is used today and that in order to efficiently support future use the internet needs to be fundamentally re-examined and perhaps re-implemented.

As expected, given that they are trying to solve similar problems and have taken similar conceptual approaches, the capabilities and features of these projects are similar as well. To begin with, they all use named data objects as a central abstraction. In this paradigm, information elements like videos, web pages, databases, and the like are represented by unique names rather than locations. Current internet systems blur this distinction. Uniform Resource Locators (URLs) were never designed to be used as content names, for example. They are content addresses, describing the server and port on which content resides, the protocol to use to retrieve the content, and the specific location of that content on the identified server [18]. That said, they are still commonly used to name content, particularly in caching systems [30] and content distribution networks [40]. The names of these data objects, since they have unique relationships to content, are tightly bound to the content they represent. These names need to exhibit strong name-data integrity, so that the name can

## *Chapter 1. Introduction*

be trusted to refer to specific content, and the object retrieved must be verifiably authentic. They have very similar programming interfaces. These interfaces are built around acquiring and routing specific data objects from providers to consumers rather than forwarding bits from one system to another [46, 47]. As a result, operations are oriented more toward registering interest in a named data object in some way, either through a specific object request or subscription, and the resulting delivery of that object [16]. ICN systems route information in similar ways as well, depending on the specific naming topology used as well. Some ICN systems use name resolution services to bind specific objects to names [29] while others use direct routing schemes to multiple hosts [31]. Finally, data objects are frequently cached on devices, both on edge devices and in-network. These caches are generic and usable by any other services distributed throughout the network [16].

Contrast these design principles with those used to build the current internet. Where internet-scale networks were originally governed by end-to-end principles and packetization of information [25, 49], ICNs are build with a much more data-centric perspective, focusing on routing information rather than just ones and zeros. This is not a refutation of end-to-end arguments however. Rather, it is an affirmation that some services must be incorporated into the fabrics of these kinds of systems reflecting the ubiquitous need for those functions [21].

Though the various different types of ICNs certainly have similarities, they have significant differences too. To begin with, they are not all synchronous. PSIRP, for example, uses asynchronous publish/subscribe style communication between nodes, where requests and responses can be routed differently. This is very useful for longer-lived sessions on mobile networks where the content is available at some point in the future, for example. Furthermore, while most of these systems use the data object as the primary networking primitive, CCNs do not. Rather, they still use packets. They all use IP as the basic transport layer technology, but they can use other protocols as well. Names may or may not be human-readable, and some systems use public-key infrastructures to enforce name-data

integrity [16].

These differences can provide significant advantages however. Avoiding public-key cryptography in data object names allows for human readable names, for example [16]. Allowing network clients to subscribe to content allows dynamic content to be generated and then routed to a client at some point in the future. This is still an active area of research where the most appropriate solutions have yet to be established.

The fact that these systems have independently concluded that future macro-networks should use a data object as the primary abstraction is significant in that it demonstrates a widespread belief in the approach. Furthermore, application-level overlay networks have used this essential approach to content management as well, though not as pervasively as ICNs [51]. These types of solutions do offer significant advantages and solve many of the problems currently facing large-scale networks [16]. Timelines for adoption, feasibility of migration, and other important issues have yet to be established at this point. With that in mind, commercial organizations have begun to invest research and development budgets in this kind of work [8, 11].

## **1.5 Other Related Work**

This work introduces the notion of usage management embedded in a delivery network itself. It also provides an in-depth analysis of the challenges and principles involved in the design of an open, inter-operable usage management framework that operates over this kind of environment. Besides referencing the material we have covered in depth to portray the current state of the art, the analysis includes application of well-known principles of system design and standards [21, 25, 27], research developments in the areas of usage control [43, 35], policy languages design principles [36], digital rights management (DRM) systems [34], and interoperability [33, 32, 37, 28, 1] towards the development of

## Chapter 1. Introduction

supporting frameworks.

While a large body of work exists on how overlay networks can use policies for *network* management, very little work has been done on using usage policies for *content* management. The primary contribution in this area focuses on dividing a given system into specific *security domains* which are governed by individual policies [45]. This system fits into our proposed taxonomy as an  $\alpha$ -type system as it has domains with single separating guards.

A large body of work currently exists with respect to security in and over overlay networks. These kinds of techniques and this area of study is vital to the production development and delivery of overlay systems, but is outside the scope of this work.

## Chapter 2

# Usage Management in Information-centric Networks

When it comes to managing the usage of information resources, information-centric networks provide capabilities that traditional packetized networks cannot. The basic structure of packet networks facilitates simple and efficient data transfer, but is fundamentally based on certain design assumptions that render network-centric usage management difficult at best and impossible at worst [25, 49]. Information-centric networks, taking a very different approach to network design, are much more amenable to embedded content control based on their different design principles 1.4.

Current packet-based systems share three underlying design principles. Strict layering, in which upper layers only use services that exist in lower layers which in turn have no knowledge of upper layers, end-to-end arguments governing service placement, and limited runtime packet sizes. All three of these increase the difficulty in applying control over information transmitted through networks.

In internet systems, switching and routing traditionally occur in the lower layers of the OSI model [50]. These decisions are made based on a priori knowledge of a given network



topology, by manual or programmatic configuration [6] and are not impacted by transmitted content except in very high-end systems [10]. In fact, access to application content occurs at much higher levels [50]. As a result of strict service layering, the information needed to make content-sensitive routing decisions is simply not available without breaking layer encapsulation on these kinds of devices. Granted, Vendors do provide switches that examine application-level traffic [10]. These intelligent switches are expensive however and as a result are only feasible for large ISPs to deploy.

End-to-end arguments dictate where services should be placed in a network. Services like information distribution control that require access to application layer data should, following these principles, be deployed into the ends of a given network [49]. Admittedly, this does encourage scalable network design, by keeping the core of a network simple, efficient, and fast. It however does not support granular information distribution control based on content rather than topology. In order to control information flow based on content, internal network nodes must be able to access and evaluate transmitted content. The fundamental end-to-end principles when applied to this problem would strictly prohibit that kind of content analysis.

Likewise, policies associated with content can be arbitrarily large. As a result, they can exceed maximum packet sizes defined in packetized networks. Furthermore, as content-sensitive networks must evaluate defined policies prior to routing content, any policy to be evaluated must be completely downloaded into a router and analyzed for suitability for transmission prior to any packet routing, leading to inevitable bottlenecks as content is queued behind the policy elements.

Content analysis of certain kinds of transmitted artifacts may not be possible without a holistic perspective either. For example, if I have an XML document transmitted through a network, that document may very well have content in element  $n$  which is described in more detail in element  $n + 2$ . Here, element  $n$  and element  $n + 2$ , by themselves, are not sensitive. When combined however, they are. When transmitted, these elements would be

in separate packets. For the sake of this argument, assume these packets are built such that element  $n$  is in packet  $m$ , and element  $n + 2$  is in packet  $m + c$ , where  $c$  is some constant, and that packet  $m$  is assembled and transmitted from the source node at some time prior to packet  $m + c$ . In this scenario, packet  $m$  will be passed through an intervening nodes prior to packet  $m + c$ . Even nodes that maintain a history of transmitted content that may be able to determine that information in  $m$  is sensitive when combined with information in  $m + c$  will be unable to undo the earlier transmission of packet  $m$ . In order to circumvent this problem, nodes would need to hold packet for some time  $t$  to check for context. This may help solve the problem, as related information likely has some kind of intrinsic locality, but nevertheless the size of  $c$  can be still be relatively arbitrary. As a result, the size of  $t$  is impossible to set *a priori*. This approach imposes possibly significant performance penalties as well.

Information-centric networks are based on different primitives, as described in Section 1.4. Specifically, they are based on named data objects with strict name-data integrity, as well as other associated principles. This different abstraction makes policy evaluation and content binding simpler, as content can be bound either in-line to policy or via specific naming conventions. In these systems, once content is located by name, it is returned to the requester either via a predefined path mirroring the original request path or a variable response path. In either case however, all content and associated policy is available at each routing node in that return path, and can be evaluated for suitability of transmission.

As a result of these fundamentally different underlying models, information-centric networks in the next-generation internet enable usage management capabilities that are very problematic to implement and enforce in current internet architectures without dedicated application-level overlays.

## 2.1 Characteristics

Examining content at each network node or router point can certainly impact performance and by extension availability. We also need to establish that this kind of dynamic dispatch will actually guarantee delivery along the most secure path. We can help to increase the throughput of the network overall by taking advantage of the volatility characteristics of the network. With respect to delivery, we need to show that by selecting optimum paths at given network points the overall selected path will have the appropriate security characteristics outlined by any policy associated with delivered content.

To begin with, imaging in a given aggregate path between two points if local decisions are made with respect to routing based on specific security criteria at interleaving points the path as a whole will adhere to those security criteria. Essentially, this implies that we can use a greedy algorithm with respect to security and routing and that the algorithm will yield an optimal security path. It is important to recognize that this is key to establishing a secure route between two specific points. Furthermore, in a given route, that route must be viewed temporally as well, in that each link may not be optimal when the delivered data element reaches a destination, but each link was optimal at the time it was selected, and by extension, when the aggregate path is reviewed, it would likewise be optimal with respect to time of traversal. Finally, local nodes may very well have knowledge about the local environment that cannot be known by a centralized routing authority. Allowing local routing decisions with respect to security can help take advantage of this locality.

The idea behind the proof is if a path  $P$  exists consisting of nodes and edges  $\{V, E\}$ , that path was then assembled by choosing the most secure edge  $e \in E$  from a corresponding node  $v \in V$  at some time  $t$ . The path  $P$  consisting of these edges  $e$  is then the most secure path that can be chosen. A proof by contradiction establishes the feasibility of this approach:

1. Assume a path  $P = \{V, E\}$ .

2. Assume that  $P$  is not the most secure, and that a more distinctly secure path  $P' = \{V, E'\}$  exists.
3. If  $P'$  exists, then at some  $v \in V \exists e' \in E$  such that  $e'$  is more secure than the corresponding edge  $e \in E$ .
4. If so, then at all  $v \in V$ ,  $e' = e$  leading to  $E' = E$  and  $P' = P$ , so  $P'$  is not distinct from  $P$ .

This assumes that a path of some kind does exist. If so, and if the most secure edge is chosen at each node, the resulting path will in fact be appropriately secure and policy compliant.

## 2.2 Taxonomies of Usage Management Overlay

A clear taxonomic organization of potential steps in approaching finer-grained policy based usage management helps in describing the difficulties inherent in developing potential solutions as well as aiding in planning system evolution over time. Here, we have four distinct types of integrated policy-centric usage management systems, as shown in Table 2.1. Of these four, only the first two levels are represented in current system models.

In this taxonomy, it is not required that systems pass through lower levels to reach higher ones. This taxonomy represents a continuum of integration of usage management controls. Systems can very well be designed to fit into higher taxonomic categories without addressing lower categories. That said however, many of the supporting infrastructural services, like identification management or logging and tracing systems, are common between multiple levels.

The taxonomy itself starts with the current state, integrating policy evaluation systems into the network fabric gradually, moving away from filters, then by adding policy evaluation

<i>Name</i>	<i>Description</i>
$\phi$	The initial level of this taxonomy, $\phi$ classified systems have a single guard without policy-based control
$\alpha$	$\alpha$ classified systems have a single guard by have begun to integrate policy-based control
$\beta$	Systems that have begun to integrate policy-based control with router elements are in the $\beta$ category
$\gamma$	Systems that have integrated policy-based control with routing and computational elements

Table 2.1: Proposed Usage Management Taxonomy

into the routing fabric, then the computational nodes, and finally by incorporating evaluation directly into content.

The UCDMO, described previously, is focused exclusively on promoting controlled sharing of sensitive information and has specific goals that a clear, realized taxonomy of granular information-centric usage mangement helps fulfil. Those goals include an ideal end state described as a flat network architecture with usage management injected into the distributed system. This is exactly the final  $\gamma$  architecture described within this taxonomy [41, 7]. The UCDMO also has specific goals outlined within its founding charter, including:

- **Optimize Capabilities** — Drive robust and extensible cross domain capabilities to support a secure and integrated information enterprise.
- **Oversee Resources** — Maximize return on cross domain investments, reduce duplication of effort, and increase efficiency of cross domain activities.
- **Mitigate Risk** — Support risk-based decisions by enabling global awareness of cross domain operational connections.
- **Provide Leadership** — Provide leadership across the interagency spectrum to ensure coordinated cross domain governance, oversight and community reciprocity.

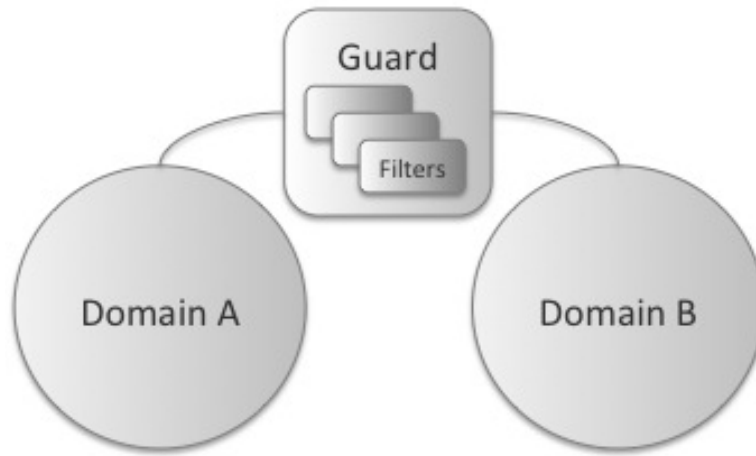


Figure 2.1: Taxonomy ( $\phi$ )

Our work here certainly contributes to these goals, providing robust cross-domain capabilities, helping mitigate risk, and contributing toward advancing the state of the art in this kind of multi-level security environment.

### 2.2.1 $\phi$ -level Overlay Systems

The  $\phi$  classification consists of systems like the initial NSA and BAH notional models. These systems consist of two distinct domains, separated by a filter-centric single guard. The initial NSA system model is clearly of this type, separating two domains with a guard using filter chains. The BAH model is also of this type, using a Filter Segment to evaluate data packages transmitted between interface segments attached to specific domains.

Generally one of the domains supports more sensitive information than the other, but that is not always the case. In the models we have examined this has certainly been true, but classified information for example is commonly stored in *compartments* which are separated by clear *need-to-know* policies enforced by access lists and classification guides. These kinds of compartments contain information at similar levels of classification, but contain distinct informational elements that should not be combined.

In these kinds of systems, specific rules regarding information transfer and domain characterization are tightly bound to individual filter implementations. They are based on *a priori* knowledge of the domains the guard connects, and therefore are tightly coupled to those domains. Furthermore, the filter elements are standalone within the system, in this classification, not availing themselves of external resources. Rather, they examine information transiting through the filter based purely on the content of that information.

The set of filters that could be developed and deployed within the guard are unlimited. Developers could easily create a filter that inspects and possibly redacts the sections within the document, rather than passing or not passing the entire document through the guard. Indeed, if we assume even very limited processing capabilities within the guard, that is, turing completeness, then this guard can be made as powerful as any solution we can derive for implementing a cross-domain solution (CDS). Thus the computational power of the guard is not the issue. The real issues are the benefits that can be gained by distributing the capabilities intelligently within the networked environment as opposed to fixing them programmatically and topologically at the perimeter of a sensitive network.

### 2.2.2 $\alpha$ -level Overlay Systems

The  $\alpha$  overlay classification contains systems that have begun to integrate policy-centric usage management. Both policies and contexts are dynamically delivered to the system. The dynamic delivery of context and policy allows these kinds of systems more flexibility with policy evaluation. The  $\alpha$  category begins to integrate policy-centric management rather than using strict content filtering.

Here, we again have at least two domains, Domain A and Domain B, though we could potentially have more.  $\phi$  type systems require domain specific information to be tightly coupled to the filter implementations. Separating the permissions, obligations, and other constraints from the filters and incorporating them into a specific separate policy entity

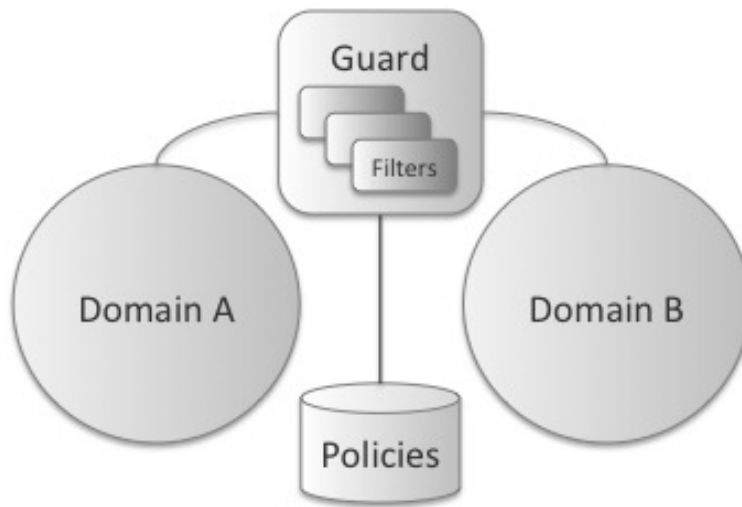


Figure 2.2: Taxonomy ( $\alpha$ )

frees the guard from this coupling and provides additional flexibility to the system.

The guard can continue to use filters to process data. These filters however are now more generic and decoupled from the specific domains the guard manages. The choice of using a specific filtering model rather than some other kind of construct is a design detail level to implementers. That said however, individual filters will be remarkably different and still need to understand the ontologies over which specific licenses are defined rather than specific content semantics.

The policy repository is key to the implementation and differentiation of this taxonomy category. This repository can be implemented as a separate repository keyed into via a data artifact's unique URI, for example. It could also represent a policy sent in tandem with a data artifact in a data package.

The policy repository may be implemented as some kind of external service, and as such, represents the first such external service explicitly used in this taxonomy. Other external services may well exist and be used to adjudicate information transfer decisions as well.



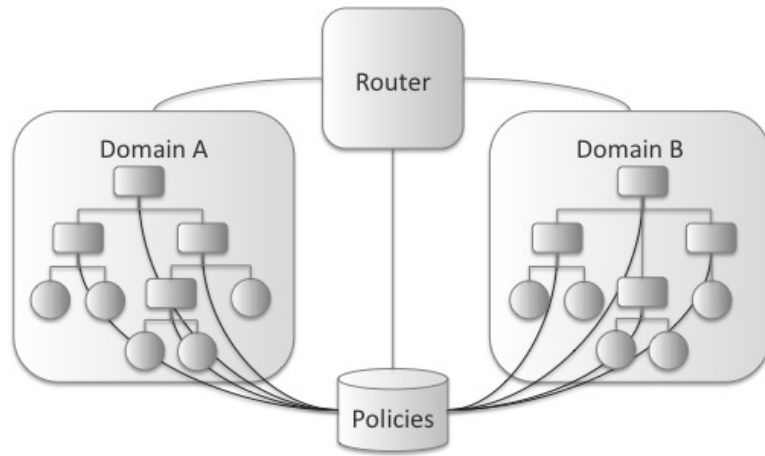


Figure 2.3: Taxonomy ( $\beta$ )

### 2.2.3 $\beta$ -level Overlay Systems

The  $\beta$  taxonomic category begins to integrate policy-centric processing with router elements in a given network.

In this model we can also host multiple domains as a result of flexible policy-based content examination. Each domain hosts a network of some kind, though that hosted network could very well be a degenerate network of a single system. Each network hosted in a domain is hierarchical, with specific computational nodes embodied by workstations, tablet computers or mobile devices, and routing points embodied by routers or switches of some kind.

Note that usage management has started to penetrate into the routing fabric of the network by doing content evaluation at router points. Content-based switching networks have been successful in other domains, and such techniques can be used here to provide policy evaluation capabilities [12].

Certain types of traffic are easier to evaluate than others however. For example, HTTP requests and responses are easier to examine than TCP packets. When examining TCP

packets, systems generally require additional context to select an appropriate packet window (e.g. the number of packets cached for examination). HTTP traffic does not usually require this kind of flexibility. Information-centric networks, due to their shift from previous network models, follow this same kind of pattern by placing the data to be protected at the center of attention.

This migration of policy evaluation into the routing fabric provides for enhanced data security and better network management, especially if part of a network is compromised. Now that policy decisions can be made at the router level in a given network, we are starting to have network security in depth rather than simple perimeter protection. This not only provides the ability for additional information protection, but also allows for different compartments holding information at different need-to-know levels to be created ad-hoc under different routing segments. In cases of network compromise, this kind of dynamic policy enforcement can also allow for quick node excision as well.

#### **2.2.4 $\gamma$ -level Overlay Systems**

The  $\gamma$  compartment has integrated policy evaluation with compute and routing nodes. Here, policies can be evaluated against content at all network levels — nodes emitting requests, nodes fielding requests, and all routing elements in between.

We see that the policy repository is supplying services to all computational elements in both domains. This gives us increased granularity with respect to data compartmentalization by integrating information security into each network element. At this point, the network can create compartments of single nodes, while previously in  $\beta$  level systems compartments could only be created under specific routing elements. At this level, we can also provide services revoking data access based on policy evaluation decisions when needed.

Furthermore, individual node exclusion is possible as well.  $\beta$  classified systems could excise network elements under specific routers by dynamic policy application. Now, we

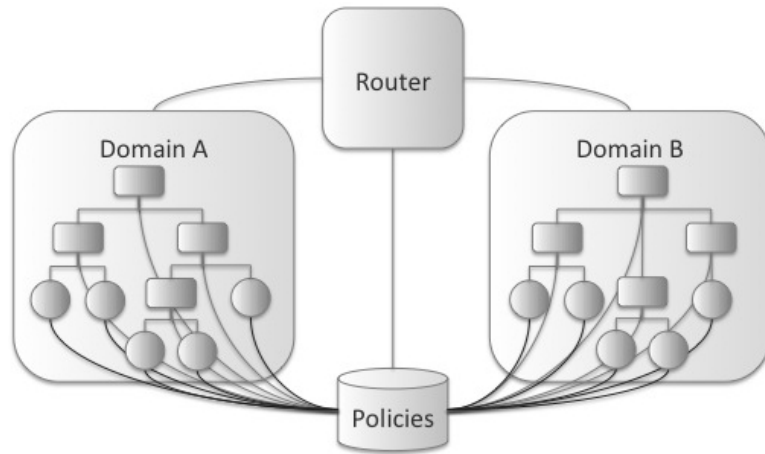


Figure 2.4: Taxonomy ( $\gamma$ )

can apply the same functionality to individual compute nodes. For example, if a networked device like a smart phone is compromised, that device can be removed from access quickly or used to supply mis-information.

## 2.3 Taxonomic Analysis

The various levels of the taxonomy vary primarily with respect to the inclusion of policy-based usage management and information-centric structure.  $\phi$  type systems are not structured with information-centric use in mind, nor do they use policy-centric management. Conversely,  $\gamma$  type systems are both purely policy oriented and completely information-centric.

As systems move through the various levels of the taxonomy they gradually move from one side of the spectrum to another. Information-centric structures, hierarchical or otherwise, gradually migrate into the network beginning with  $\beta$  systems. Policy orientation is injected into the architectures starting with  $\alpha$  systems and moving into the network fabric in parallel with information-centric exploitation.

### 2.3.1 Characteristics of Policy-centricity

In these systems, policy-based management supplies distinct advantages over filter-centric information control. This kind of policy-centric usage management is more content specific than filters, more flexible, and is more expressive than filter-centric systems.

#### Content Specific

Filters, in filter-based systems, are not coupled to the content passing through the system. Rather, they are usually tied to the characteristics of attached networks. For some filters, that is not problematic. Mal-ware filters, for example, are very general and do not need to have an understanding of filtered content and are not sensitive to that content at all, though they can be very sensitive to specific context. This limitation does however prohibit filters from doing anything content specific. Due to their deployment limitations, in that they are deployed to such a system via a process distinct from processing content, they are unable to use presented content or current dynamic context to influence information processing decisions.

Consider content  $c$  impacted by a dynamic context  $d$  where  $d$  is defined in terms of the content itself, the person or system requesting that content, and the environment in which that request is made. Here, only under certain specific environmental conditions is that requesting agent allowed access to the requested content. Ergo, the decision to pass the content to the requester is based upon characteristics of the content related to dynamic changes within the environment. A filter-centric solution contained within the  $\phi$  level of the taxonomy is unable to change filter rules based on changes like new content or environmental alteration as a result of the static nature of the deployed filters. A policy-based system, on the other hand, is able to express the content specific policy easily for more dynamic evaluation.

For example, if  $c$  contains information that can only be accessed for a specific time period, a static filter based on evaluating content only cannot determine that the information in  $c$  is no longer appropriate for dissemination after that time period ends. That kind of evaluation requires meta-data associated with  $c$  that specifically describes these time bounds and a dynamic contextual evaluator able to determine when that window of access has closed, functionality static filters do not demonstrate.

## **Flexibility**

Policy-centric systems are more flexible than filter-based counterparts. In a filter-based solution, the type of content that can be evaluated is tightly coupled to the filters installed. If a given piece of content is new to a given filter-centric solution, that content cannot be appropriately examined and must be submitted for human review. A policy-based system is designed to be more general. When based upon a common ontology, the evaluation system can be very general with respect to its evaluation of a given policy [35]. A general policy engine can handle a great variety of different content as long as the policies associated with that content correspond to known domain ontologies. This generality leads to a greater amount of flexibility with respect to what can be expressed in a specific policy.

A filter is going to have a specific responsibility, like redacting sensitive words from a document, for instance. In order for that filter to redact those sensitive words, it must have access to some kind of list of what those sensitive words are. Remember,  $\phi$  level systems use static filters, so that filter can only be updated when the filter itself is updated. Now a policy-centric system on the other hand can have a policy associating sensitivity with various areas of content in a specific document. In this case, all the system must do is understand the sensitivity described in the policy associated with the content, and can then redact that content if needed. The ontology describing the areas of sensitivity will change more slowly than the possible content itself, leading to a more flexible maintainable system.

This is of course a simple example solvable by creating a dynamic list; the key point of the above example is that the specificity of the filters requires additional complexity in the filter system itself. The generality of the policy-centric system allows the complexity to be more clearly expressed and contained within the policy file.

### **Expressiveness**

While filters can process content at specific perimeter points, its lack of reach into a given network fabric limits the power a given filter can actually have over transmitted content. A policy associated with content, when transmitted with content, can reference much more than the semantics of the protected content. That policy can describe specifically, in detail, how that content can be used. Filters cannot exercise that level of control.

Assume a distributed system with multiple filter points. In this kind of system, information distribution can be controlled via deployed filters at a relatively fine level of granularity. This kind of distribution control cannot influence the use of protected content however — once that content is distributed, possessors are accorded full access.

Policy-enabled systems are not limited in this way. Policies, when coupled with policy evaluation tools, can exercise control not only over distribution and routing, but also over use of distributed content at endpoints.

These advantages accrue in usage management systems as policy capabilities are propagated through the information-centric fabric. Some of these advantages, like expressiveness, appear simply by beginning to use policies instead of filters. The remaining two have more of an impact as additional policy-centric nodes combine to form a system suitable for cloud deployment, increasing their impact as they move from  $\alpha$  to  $\gamma$  types of systems.

### 2.3.2 Information-centric Structure

Information-centric integration exhibits clear advantages over single point perimeter systems as well. Specifically, information-centric systems are more partition-able than perimeter solutions, enable content throttling, provide capabilities for dynamic content control, and allow content to be more traceable.

#### Characteristics of Partition-ability

Administrators typically deploy filter-based perimeter protection at strategic routing points on secure networks. These kinds of networks are designed with specific regions of enhanced sensitivity separated by cross domain management systems regulating information flows [39, 42, 22]. While sensible from the perspective of each protected region as a secure domain, this design thinking begins to fall apart when exposed to the very real threat of the malicious insider. Boundary-centric information flow control is impossible to realistically achieve when the actual boundaries between malicious actors and system users is constantly in flux. When a malicious actor can be anywhere within a system, boundaries are simply too dynamic to be realistically recognized. In order to surmount this fluid system posture, designers must adopt a security in depth mindset.

Information-centric networks enable this kind of defence in depth via the possibility of partitioning. A given information-centric system depending on the level of can partition the user space and by doing so decrease the attack surface available to a malicious insider.  $\phi$  and  $\alpha$  level systems based on perimeter filters simply do not present this ability. Systems beginning with  $\beta$  provide the potential to create need-to-know cells of finer granularity up to  $\gamma$  type systems in which cells can be created at the level of specific nodes. These need-to-know cells serve to help quarantine possible intrusion into the sensitive distribution fabric if that fabric is compromised by helping isolate that system failure within a compromised cell.

## *Chapter 2. Usage Management in Information-centric Networks*

For example, assume a hypothetical system with nine nodes connected along a single data plane within a prototypical secure network. With perimeter defenses, if one of those nodes is compromised, a malicious actor can potentially begin to monitor communications traffic between all network nodes, effectively compromising the entire network. In this same network, if designers partition the system into three cells of three nodes, a similar intrusion in one of those cells will effectively only compromise that cell, leaving the other two cells unaffected. This decrease in possible targets for compromise effectively decreased the network attack surface from any give node by  $\frac{2}{3}$ , correspondingly increasing the security posture of the system.

### **Content Throttling**

Perimeter located filter systems only have the opportunity to control sensitive traffic at that initial boundary. Information located in repositories behind that boundary is not subject to control if it is retrieved by an agent also ensconced behind that same system boundary. Granted, control can be exerted at the repository level, but in a system with more than one repository, this is of limited impact.

A partitioned cell-oriented system, on the other hand, provides greater opportunity for information monitoring and control. The partitions provide additional potential control points requests must cross in order to access needed information. Furthermore, less random cell design provides the capability to unify repositories, providing tight control of information dissemination.

Our hypothetical nine-node system, for example, provides no control over information dispatched from one of the contained nodes to other contained nodes in its initial design form. There are no control points within that nine-node network at which to monitor and control information flow. Partitioning that space into three three-node cells provides at least two potential control points for inter-cell requests at which information flow can be



monitored. In cases where a malicious insider is actively collecting and hoarding data for exfiltration, these additional control points give administrators the ability to automatically throttle the rate at which sensitive material can be accessed by users to increase the cost of data collection and increase the likelihood of agent discovery.

### **Dynamic Content Control**

Singular perimeter solutions due to their lack of internal control points also forego the ability to provide dynamic content control. Once information has traversed a given perimeter access point, it is no longer under the control of that point and can no longer be retrieved, accessed, monitored, or modified. Solutions with internal control points can provide the ability to continually monitor and control disseminated information.

Within a given information-centric system, depending on that system structure, data can be more rigorously controlled.  $\beta$  and  $\gamma$  systems provide the ability to dynamically change information access via contextual changes at a finer grained level than perimeter solutions can.  $\gamma$  systems can in fact provide the ability to retract information access on a per request basis.

This kind of control is especially useful in situations where external partners may temporarily need access to sensitive information for a specific short period of time, say during some kind of joint exercise or activity.  $\gamma$  and  $\delta$  systems can provide that access only during the window of operation, and retract that access when that window closes. This kind of use is common in joint military operations with coalition partners.

### **Traceability**

The singular location of perimeter filter solutions also precludes easy information traceability. Data requests within a given network sans internal controls is more difficult to

trace than an information-centric solution with a partitioned cell structure that is tailored to the specific information requested (say, XML databases or semantic web content). The partitioned information-centric system requires requests to traverse multiple routing nodes at which request and response content can be examined and stored for later analysis and visualization. Perimeter solutions without this kind of structure cannot monitor flows at this finer-grained level.

The strengths of information-centric systems over single perimeter points gradually increase as information-centricity permeate any given system. Some abilities, like content-centric access repudiation, can only occur at the  $\gamma$  level. Others, like traceability or throttling, become more effective as a system architecture traverses from lower to higher levels of capability within the proposed taxonomy.

## **2.4 Analysis**

What do we expect to see when we inject UM into ICN? what are the tradeoffs? what kind of conclusions can we extract?

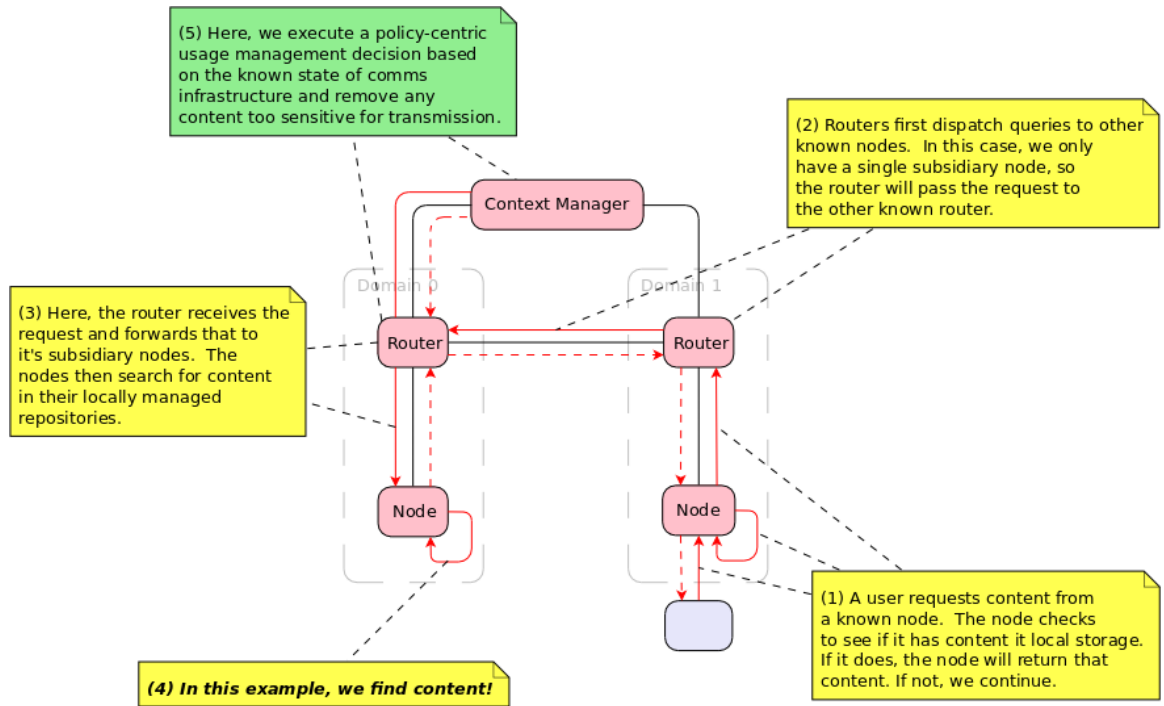


Figure 3.1: Simulation Logical Configuration

## **Chapter 3**

# **Experimental Configuration**

### **3.1 Overlay Implementation Concerns**

A key concept in our current work is the separation of content management from physical communication networks. In the past, content was controlled via partitioning and physical network access management. Physical networks were tightly controlled as a way to manage access to sensitive content. Classified networks in common use today are canonical examples of this kind of approach to content management. Access to these networks is tightly controlled by classification authorities and the ability to transfer content from these networks to more open systems is rigorously managed. Corporate systems have also commonly used this kind of approach, though not usually with so much regulation or rigor.

This kind of approach is not scalable however. It imposes huge costs and infrastructural requirements that are becoming too large to effectively manage. Furthermore, future systems containing sensitive information require similar security features, and simply cannot be developed without custom controlled infrastructure. Health care systems, for example, have huge security needs and a more finely grained level of application than even deployed government systems. These systems will contain exabytes of data, all of which

### *Chapter 3. Experimental Configuration*

needs to be explicitly controlled, managed, and reviewed by those associated with specific managed records.

Separating content networks from physical networks enables network infrastructure virtualization and multi-tenancy. Use the popular file-sharing system BitTorrent as an example. BitTorrent is a content network optimized for download efficiency. It runs over traditional TCP/IP networks, but manages traffic according to specialized algorithms unique to BitTorrent. These algorithms take advantage of the asymmetry between upload and download speeds of typical home-use Internet systems in which upload speeds are regularly an order of magnitude slower than download speeds. By partitioning content into distinct sections and downloading them from multiple clients, a downloading node can effectively use all available download bandwidth and is no longer necessarily constrained by the upload bandwidth of a serving peer system. We use a similar approach, in that our hypothesized systems also overlay TCP/IP traffic, but rather than optimizing download speeds we focus on content usage management.

Just as systems like BitTorrent run over current established protocols, usage management overlay systems could as well. They support multi-tenant cloud computing systems by providing secure compartmentalized access to managed information. They also support the ability to create and use integrated overlay systems between multiple cloud providers, supporting running of overlay components in systems hosted at Amazon while accessing nodes executing on Rackspace infrastructure.

Content networks must deal with situations analogous to those encountered in previous physical systems. Specific examples include cross-domain monitoring and content mashing. Both problems are currently areas of active research within physical networks and need extensive examination in overlay systems as well.

To begin with, in content-specific overlay networks, cross-domain routing can become an even more pervasive issue. Currently, cross-domain data processing guards are installed

### *Chapter 3. Experimental Configuration*

on the perimeter of sensitive networks where they can monitor and manage outgoing and incoming traffic. In content networks, these kinds of systems can begin to multiply within the information transmission fabric. In physical networks, the network topology is fixed and is established when the network is installed. After installation, changes in the essential network topology are cost-prohibitive and correspondingly rare. Overlay systems do not suffer from this high cost of change, and can easily morph from one topology to another. As additional content enclaves appear within a given overlay topology, the need for content usage management between those enclaves increases.

Mashup scenarios become similarly common. As additional sources of accessible data appear, opportunities for inappropriate data combinations increase at best geometrically. Data combinations need to be likewise managed to prevent inappropriate data combinations.

## **3.2 Initial Prototype Implementation**

Our first completed prototype shows that overlay routers can in fact use licenses bundled alongside content to modify transmitted content based on dynamic network conditions. Running on a single host over HTTP, it simulates two content domains and communication between them. The communication link has uncertain security state and changes over time. Note that this prototype currently runs on a single host with varying ports, but it could easily run on multiple hosts as well. The current single host configuration is simply to simplify system startup and shutdown.

License bundles are hosted on the filesystem, though they could be hosted in any other data store. These artifacts are currently XML. They are stored in a directory, and the license file has a LIC extension while the content file has an XML extension. Both the content and the license files have the name of the directory in which they reside (for example, if the directory is named test, the license file is named test.lic and the content file test.xml). In

### *Chapter 3. Experimental Configuration*

this context, the directory is the content bundle. The license and content files are simply documents and port to document-centric storage systems like MongoDB easily. They can certainly be stored in traditional relational databases as well.

The system itself has two domains, Domain 0 and Domain 1. Each domain consists of a client node and a content router node. Requests are initially served to client nodes. If client nodes do not contain the requested content, they forward that request to their affiliated content router. The content router will send that request to all the content routers of which it is aware. Those other routers will then query associated client nodes for content. If the requested content is in fact found, it will be returned to the original requesting router and then to the requesting node. If the content is not found, HTTP status 404 codes are returned to requesting routers and nodes.

All router-to-router content traffic is modified based on security conditions. A Context Manager maintains metadata regarding network paths. If a given network path is only cleared for data of a certain sensitivity level, a transmitting router will remove all license information and content that is associated with higher sensitivities, and then transmit only information at an appropriate sensitivity level over the link.

Figure 3.1 shows the prototypical workflow through the system across the domains, and Figure 3.2 shows the current system configuration of the simulation, with the cross-domain link highlighted in red. The system is current configured to use ports 4567 through 4571.

All content requests are via HTTP GET. Link status can be changed via HTTP POST and we use the CURL command to exercise the network.

This proof-of-concept does implement a simple overlay network for usage managed content over HTTP, easily extensible to HTTPS. Changes in the context of the network dynamically change the format of transmitted content. All source code for this simulation is publically available on GitHub, at <https://github.com/cclamb/overlay-network>, with documentation on how to run the simulation.

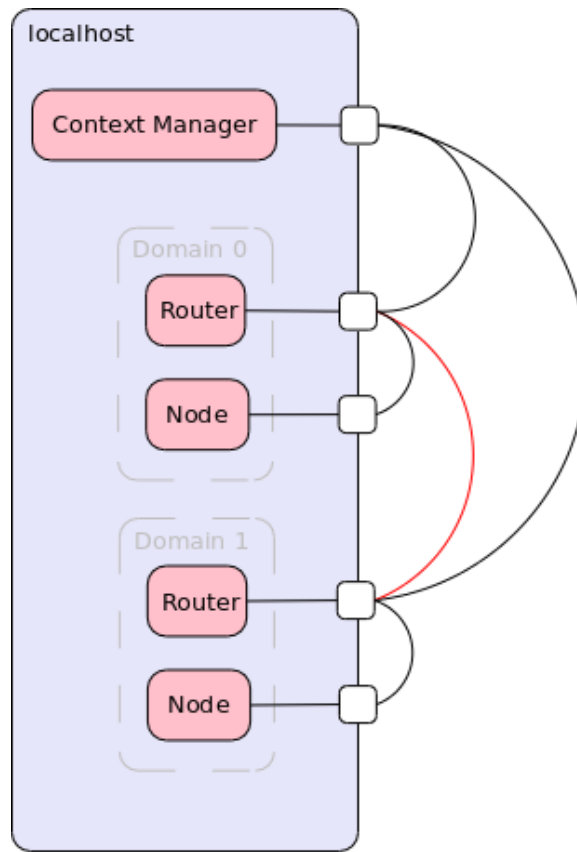


Figure 3.2: Physical Simulation Configuration

Initial results confirmed that the approach was feasible. The network was able to successfully filter content based on policies and dynamic network conditions. The system was also able to deliver both arbitrary content and policies within a single document as well, and it was not prohibitively difficult to extract either data or policies. Furthermore, Ruby with Sinatra effectively supported the required HTTP-centric infrastructure needed to effectively simulate an information-centric network. It seemed clear at this point that extending the prototypical implementation from a single host to a fully nationally distributed network was feasible.



### **3.3 Inter-Provider Cloud Configuration**

At this point, I have created and deployed baseline system images in both Amazon's Elastic Compute Cloud (EC2) and Rackspace Servers infrastructures. I have also created and exercised our deployment, configuration, and logging systems to enable distributed monitoring and centralized reporting. Overall, we currently have 20 nodes running with two distinct providers geographically dispersed across the continental United States. This leads to a distinct requirement for a centralized system with distributed access for both initial configuration information as well as logging and auditing. We have implemented this required infrastructure using Amazon's Simple Storage Service (S3), accessible from both Rackspace and Amazon hosted virtual machines.

The specific technical components are Amazon EC2, Amazon S2, Rackspace Servers, and GitHub. Both EC2 and Rackspace nodes are Ubuntu virtual machines, albeit at different versions, as we run Ubuntu version 11.04 in Rackspace and Ubuntu Version 12.04 in Amazon's infrastructures. These systems are provisioned with Git, Ruby, the Ruby Version Manager (RVM), and supporting libraries. They all run as micro-instances or equivalent, and are bootstrapped with the appropriate project information to begin to participate as an overlay network node. While EC2 and Rackspace Server infrastructures are infrastructure-as-a-cloud (IaaS) offerings supporting virtual machine instances of various types, Amazon S3 is a simple key-value store. Running with REST semantics over HTTP, S3 stores arbitrary documents associated with specific keys in buckets. These documents can be downloaded by any authorized participant, where authorization state is proven by possession of a secret key. In this way, we can store the global configuration of a specific overlay network in a single location from which every node can access information with respect to their pending role and needed configuration information. Likewise, all overlay network state can also be saved to centralized buckets for later analysis. Finally, Github is a centralized source code repository used to share code between all participating nodes. Prior to each content network instantiation, each node checks the repository for updates,

### Chapter 3. Experimental Configuration

<i>Category</i>	<i>Components</i>
<i>Infrastructure</i>	Amazon S3, Amazon EC2, Rackspace Servers
<i>OperatingSystems</i>	Ubuntu 11.04, Ubuntu 12.04
<i>Technologies</i>	Ruby (Sinatra, Capistrano, YAML)
<i>SupportingSystems</i>	Git, Github

Table 3.1: Supporting Components

and downloads them if they exist.

All data saved within S3 is serialized in a text-based data serialization language known as YAML. YAML is a widely supported hierarchical data representation language with support within the Ruby core platform. This enables us easily serialize Ruby-native data structures to text-based representations for storage within S3. More importantly, it simplifies post-experimental data analysis as any information logged to the centralized logging system during a given experimental run can be easily read and analyzed after the fact.

In order to manage and initialize all overlay nodes, we use Capistrano. Capistrano is a distributed deployment system initially used to manage large clusters of Ruby-on-Rails systems. It has since expanded into a general-purpose distributed deployment toolchain, tightly integrated with Git. This allows us to bootstrap different configurations of networks from a single command-and-control node simply and efficiently.

All these infrastructural elements, protocols, and technology components have been successfully tested, allowing for unified control and configuration of large, distributed overlay systems. We have successfully tested our logging systems, and integrated them with the Ruby runtime for ease of access. We have also passed configuration information to both Rackspace and Amazon EC2 systems and verified access from all participating nodes. Finally, we have successfully exercised the ability to dynamically update all participating nodes from Github as well as the capability to manage the system via Capistrano.

## 3.4 Inter-Cloud Architecture

At this point the system is a distributed content network distributed across multiple nodes and domains providing cross-domain managed data access. This network consists of clients accessing information through a user interface subsystem that accesses data from external sources and a distributed cross-domain information network. Queries are submitted through a client, to an application server, then to external services and information nodes.

The unique strength of this system is enabling dynamic distributed content control. This includes information retraction, redaction, protection, and secure routing. Information retraction involves quickly removing a user's access to sensitive data. Redaction addresses simple data removal, while protection would operationally involve applying encryption layers of increasing strength based on operational demands. Finally, secure routing would provide the ability to send data over a more secure link if such a link is available and required.

In this system information retraction involves changing the execution context such that access for a given user, perhaps even on a specific device, is removed. This context then propagates through the information network and attached clients. This is useful when a given user, say a coalition partner, is suddenly considered compromised and can no longer be allowed access to sensitive information. Likewise, a specific user's system may likewise be compromised and be forbidden access to specific information.

Information redaction is generally used when a user simply does not have authorization for a specific section of content, generally within a larger document. In these cases, that information and related policy metadata are simply removed from any query responses. Likewise, information protection also addresses specific subsections of information in a larger document, but unlike redaction, a user is in these cases authorized to access information, but one of the links over which the information must travel is not authorized to transmit specific sensitive information. In these cases that information can be encrypted

### *Chapter 3. Experimental Configuration*

with appropriately strong encryption to allow for more secure information transmission.

Finally, secure routing use directly addresses the ability to select communication links based on information content. In these situations, a network has more than one path over which to return content. Furthermore, these multiple paths have different characteristics providing different levels of service. The system, based on rules contained in a policy and the current context can then select communication links of different security levels when returning content. Likewise, the content network must:

- Support and distribute queries for available content based on submitted constraints including artifact key and hop count.
- Support and distribute queries for specific content based on key.
- Evaluate returned content for suitability for transmission to a requesting node at each transmission step.
- Support partitioning into multiple domains.
- Allow for dynamic information distribution at network start.
- Collect experimental metrics for evaluation.
- Be distributed across multiple nodes.

Overall, the system consists of an HTML 5 based user interface subsystem, external data sources, and a content network, as shown in Figure 1. The user interface layer displays maps and associated metadata to users based on submitted geolocation information and supports two different mobile profiles (tablet and telephone) and a single workstation profile. We use HTML 5 media queries for end device detection, allowing us to format information differently for our three profiles facilitating usability. External data sources could be any data programming interface offered by a third party over which we have no

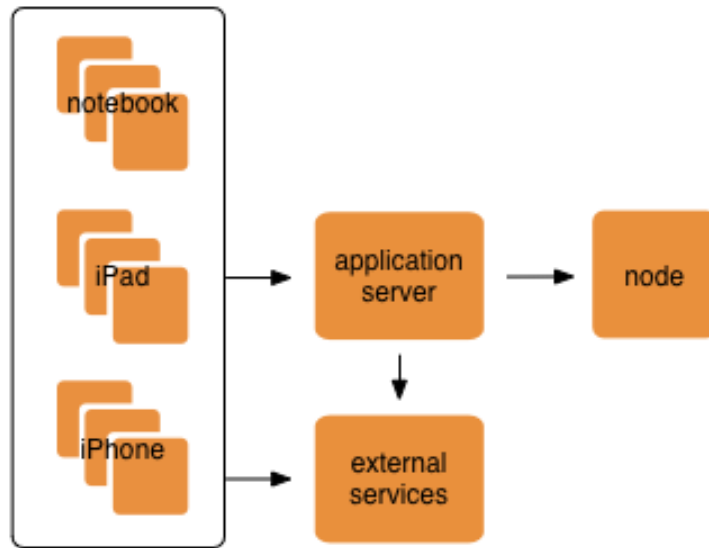


Figure 3.3: Overall System Architecture

direct control. In this system, we use Google Maps to define, download, display, and format maps. Finally, we have a content network configurable either as a hierarchical network or a non-hierarchical network containing geo-tagged information at various sensitivity levels. This content network can be configured arbitrarily, enabling us to create a virtually unlimited number of different information domains.

In this work, the client systems layer will be replaced with a command-line interface and external services will not be accessed, but a typical deployment operationally would have these elements.

The user interface subsystem processes requests and returns information from both Google Maps and the content network based on those requests. Technically, it is based on the latest version of Ruby on Rails (RoR) using standard RoR configuration conventions running on top of Ruby 1.9.\*. We use Rake for deployment, and Gem for component installation. We use Bundler to maintain consistent application dependency state and RVM to manage Ruby virtual machine versions. HTML 5 interface elements are defined using

### *Chapter 3. Experimental Configuration*

#### **SASS and HAML.**

Operationally, typical system use involves query submission, usage management rectification, and result display. We have two distinct types of queries - an initial query for a map of a specific location, generally triggered by entering some kind of geolocation parameters (though potentially using device-generated location information, allowing automatic map alignment with a user's current location) and a query for specific sensitive information. Initial queries have two distinct subqueries, one of map information directed at the Google Maps API, and another of the content network to see what data is available. All content is usage managed to ensure that mashed information is consistent from a data sensitivity perspective prior to display to the user. Currently, no information is cached within the interface subsystem.

The content network can be configured to run as an HTTP overlay system using HTTP routers and nodes or in a peer-to-peer configuration. In either case, queries can be submitted to the network from any one of the constituent nodes - note that routers do not store data; rather, they focus solely on routing queries through a hierarchical network. After initial submission, queries propagate throughout the network based on user-submitted search parameters. The content network physically runs on nodes provisioned from Rackspace Cloud and Amazon Elastic Compute Cloud (EC2). It is built using Sinatra for HTTP processing and uses Capistrano for distributed system deployment and control. We store distributed data in Amazon Simple Storage Service (S3) buckets. We use RVM, Gem, and Bundler in this system as we do with the user interface subsystem.

In both configurations, the common functional flow is built around responding to content queries with information of appropriate sensitivity for a given query context, as shown in Figures 3.4 and 3.5. In general, systems are designed with a layered perspective, with an application layer fielding initial requests, a protocol-agnostic domain layer that manages query responses, and an infrastructure layer that contains specific required libraries and other technical artifacts. In these systems, the application layer handles HTTP protocol

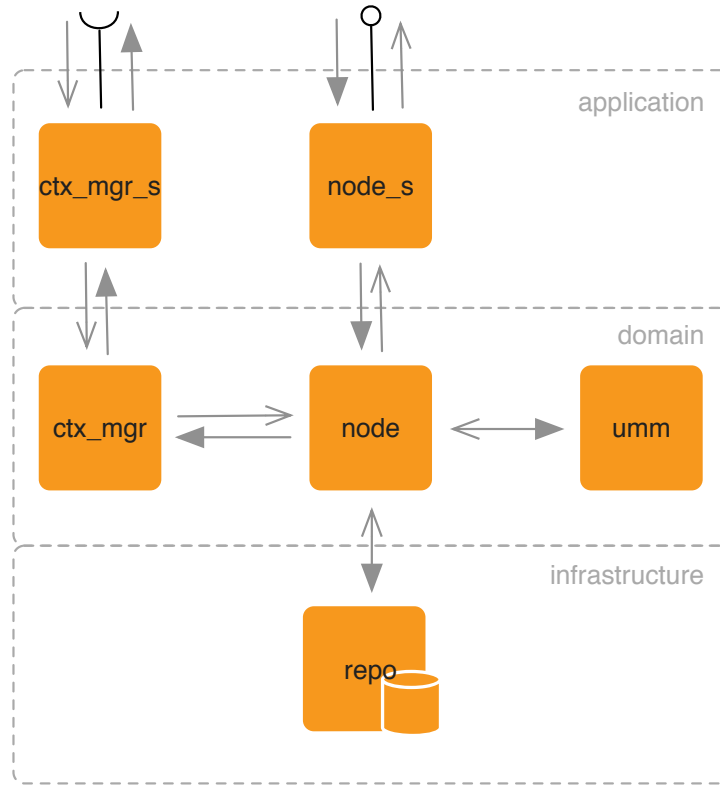


Figure 3.4: Node Architecture

issues, translating requests from the lingua franca of HTTP into the domain language reflected in the domain layer. The infrastructure layer consists of various data management technologies called upon by the domain layer when needed.

Figures 3.4 and 3.5 highlights communication ordering within components in a hierarchical content network and also shows the functional components within the system. From a communication perspective, requests come in through the application layer and are then handed off for processing to the domain layer. The domain layer retrieves the current context and is responsible for query dispatch (in the case of a router) or data responses (in the case of a node) that are managed according to the current environmental context.

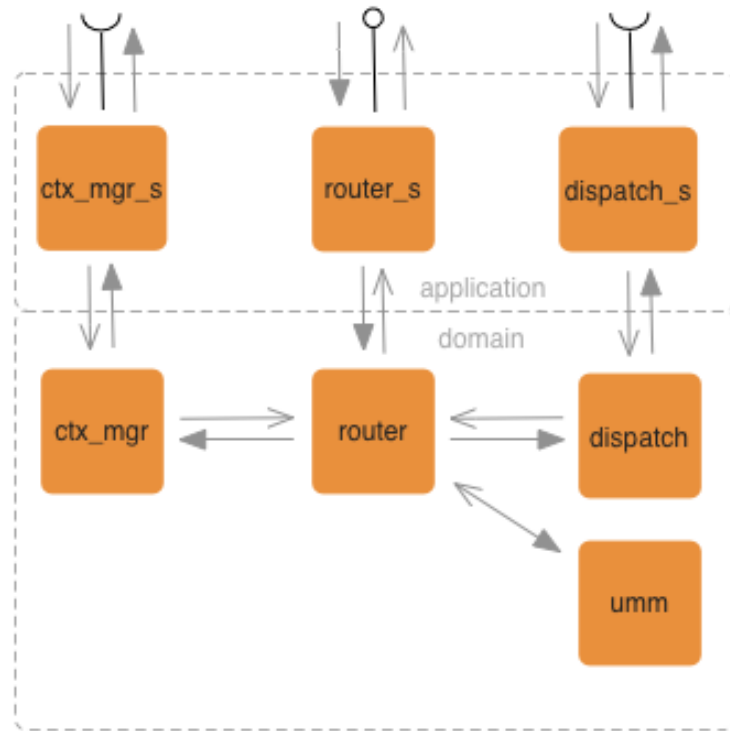


Figure 3.5: Router Architecture

The primary components in the router and node systems’ application layer are small adapters intended to translate between HTTP protocols and domain components. They are:

- **Context Manager Client Service (ctx\_mgr\_s)** — This is an adapter between the domain context manager and the external context service.
- **Node Service (node\_s)** — The node service provides a RESTful interface to external clients. All content requests are initially sent to a known node service. This is essentially the external interface to a given content network. A content network generally contains many distinct nodes as well.
- **Router Service (router\_s)** — The router service is essentially a customized HTTP router that dispatches content requests and responses through a hierarchical con-



### Chapter 3. *Experimental Configuration*

tent network in accordance with established policies and the current environmental context.

- **Dispatch Service (dispatch\_s)** — This service dispatches information requests to known nodes based on known policies and context.

The domain layer components include:

- **Context Manager (ctx\_mgr)** — The context manager client service calls into the context manager service to retrieve the most current contextual information with respect to the content network, attached clients, users, and devices.
- **Node (node)** — The node component contains all logic needed to process and respond to information requests. Nodes manage requests, responses, context evaluation, and usage management mechanism application.
- **Usage Management Mechanism (umm)** — The usage management mechanism will apply rules grouped into policies against a known context to determine the acceptability of an intended action. It will indicate whether or not that action can proceed. It can also make changes to a proposed action so that the alternative action can be executed.
- **Router (router)** — Router domain components manage the distribution of information requests and responses, applying managing information dispersal throughout a content network in accordance with context and policy.
- **Dispatcher (dispatch)** — Dispatchers send requests to known routers or nodes in the larger context network.

Finally, the sole infrastructure component:

- **Information and Policy Repository (repo)** — Unique to nodes, information and policy repositories contain specific network content, organized by key, and associated policies.

We use the same components to assemble non-hierarchical networks, in which nodes have both content and policy storage as well as request response and dispatching responsibilities. Also note that context management and usage management components are shared between all types of content networks as well as all types of component systems within those networks. Non-hierarchical nodes and hierarchical routers and nodes all need these kinds of services.

## 3.5 Experimental Structure

Content-centric networks are generalized constructs supplying the ability to manage distributed content more effectively. This work explores specifically how users can control information security and privacy in a more granular way when data is arbitrarily combined. In order to do this effectively however, we must define a simple protocol that allows connected systems to determine what kind of information is available.

We can use a variety of approaches with respect to information storage in these kinds of networks. In many ways, they exhibit behavior very similar to filesystems. In a content-centric network, rather than asking for content via some kind of address, like a uniform resource locator (URL), we use a specific non-ambiguous name. This is very similar to how content management systems and web caches work today. These kinds of systems treat a URL as a name rather than an address, returning a cached image of the requested content rather than the content actually pointed to by the URL. This requires that consumers and caching agents recognize and manage the possibility of stale data, but that risk is generally worth the performance gain. Content-centric networks can similarly optimize various

### *Chapter 3. Experimental Configuration*

aspects of content retrieval, returning the most local, highest quality, or most reliable data item, for example.

In this content network, we store metadata associated with specific locations as well as the locations themselves. Rather than optimizing with regard to location or quality, this network optimizes security posture. In order to do so, we need to specify a simple data discovery protocol so clients can discover what data is available.

We have proposed two different models for content access in this kind of network. The first, the Cat Model, mimics typical filesystem interaction on unix-centric computers. The second, the Index Model, acts more like a typical website, with a central index providing available options. Both models are can manage hierarchical content, a requirement for managing large volumes of information.

The first system is modeled after a typical filesystem. In this case, a user would have read access to the network via a set of related commands. Filesystems follow a model where you can list the available contents, access specific details of the contents, and then access individual contents themselves. In UNIX and unix inspired systems, these actions correspond to `ls`, for directory listings, and programs like `cat`, to allow access to specific individual content. File details are exposed by options on the `ls` command.

Command-line access to a content network is certainly feasible. Command-line shells are common in a variety of environments, ranging from development environments like Play to software development systems like Ruby and Python.

In our content network, the `ls` command would traverse the network returning information describing contents based on the current security context. This context consists of the environment, the resource requested, and the subject requesting the resource. For example, a user with access to a content network via some kind of shell may list network content from a device at a given physical and network location and receive content listing A, while executing a listing from a different device from the same locations may generate content

### *Chapter 3. Experimental Configuration*

listing B, which can be significantly different from A based on contextual changes.

Another problem arises with listing network contents is the fundamentally different nature of listing a relatively small directory on a local computer as opposed to the contents of a geographically dispersed network. We feel comfortable that the latency involved when reading this kind of local directory is small, and that the number of elements to list is tractable. Networks do not support these kinds of assumptions. The time required to list the available contents on a dispersed content network can be significant.

A cat-like command on a content network suffers from similar problems. As content within an artifact can be marked with different sensitivity, displayed artifact content can change based on context as well. Likewise, large artifacts can take significant time to display on devices because of content dispersion issues.

The proposed Index Model has significant precedent as well. This model is commonly used in world-wide-web systems both large and small. Modified versions have been used to seed BitTorrent networks as well as direct content traffic on early instances of Napster. Here, we have a small index file that lists available content on the network. This index could be associated with a policy and marked for sensitivity, and could contain links to content as well as metadata describing that content. This index would essentially serve the function of the `ls` command in the Cat Model. Selecting a link from an index via a network client would then serve as the Cat Model's `cat` command.

Similar issues with respect to network dispersion exist with showing the contents of artifacts in both models, and the index contents can seem to change with respect to changing context, as they are also associated with policy sets describing the use of content. Both models can also be optimized for project-centric content viewing or to show indicators with respect to expected content retrieval latency. Organizationally, any kind of informational hierarchy within the network would need to be based on the semantics of referenced content rather than external factors. Content-centric networks use keys to location content rather

## Chapter 3. Experimental Configuration

than addresses, so this hierarchical name would in fact be such a key rather than an address for the content.

Latency effects and content surprise are characteristics of the underlying content network rather than a specific interface approach. They effect either approach equally. The Cat Model is more general than the Index Model however. You can in fact implement the Index Model with the Cat Model, but not the inverse. Our work is focused on secure mashing of content in a web-centric environment. As a result, the Index Model fits our needs better than the more general Cat Model.

### 3.5.1 Initial Seed Information

In order to seed network clients, we provide an initial index object that contains location information and associated metadata. This information is classed according to sensitivity and consists of names and latitude/longitude coordinates contained in an XML file, similar to that shown in listing 3.1.

#### Listing 3.1: Seed Information for the Network

```
1 <index>
2   <location>
3     <name>The location name; a city name, for example</name>
4     <lat>The location latitude</lat>
5     <lon>The location longitude</lon>
6     <about>Metadata about the location</about>
7     <key>The detail data object name</key>
8     <key>...</key>
9     ....
10  </location>
11  <location>...</location>
12  <location>...</location>
13  ...
14 </index>
```

### Chapter 3. Experimental Configuration

Any of these XML elements can be marked with an attribute, `policy-set`, which is the name of a policy set contained in the associated policy file. It is contained as an artifact with an associated policy set.

Detail data objects are arbitrary XML documents that support the policy attribute. We support text, images, and shape information. Each different type is ensconced within an XML element corresponding to the type of data contained, and are delivered in a single XML document with associated policy sets.

Listing 3.2: Image	Listing 3.3: Shape	Listing 3.4: Content
<pre>1 &lt;artifact&gt; 2   &lt;policy-set&gt; 3     ... 4   &lt;/policy-set&gt; 5   &lt;data-object&gt; 6     &lt;image type=  &gt; 7       ... 8     &lt;/image&gt; 9   &lt;/data-object&gt; 10  ... 11 &lt;/artifact&gt;</pre>	<pre>1 &lt;artifact&gt; 2   &lt;policy-set&gt; 3     ... 4   &lt;/policy-set&gt; 5   &lt;data-object&gt; 6     &lt;shape type=  &gt; 7       ... 8     &lt;/shape&gt; 9   &lt;/data-object&gt; 10  ... 11 &lt;/artifact&gt;</pre>	<pre>1 &lt;artifact&gt; 2   &lt;policy-set&gt; 3     ... 4   &lt;/policy-set&gt; 5   &lt;data-object&gt; 6     &lt;content type=  &gt; 7       ... 8     &lt;/content&gt; 9   &lt;/data-object&gt; 10  ... 11 &lt;/artifact&gt;</pre>

In these examples, data-objects can be associated with policies contained in the `policy-set` element. Each `policy-set` element can contain zero or more policies. Sections within the `content` element can also be associated with policy sets, and currently `type` can be either `xml` or `txt`. A shape can only be associated with a policy set from the shape element itself. Properties of a shape cannot be associated with a policy set individually. Shape types include `marker`, `circle`, and `polygon`, as shown in listing 3. Data contained within an image element is Base64 encoded and must contain type information to indicate the specific image format. Currently, the supported values are `jpg` and `png`.

Listing 3.5: Marker

```
1 ...
2 <shape type=          >
3   <marker>
4     <lat>...</lat>
5     <lon>...</lon>
6   </marker>
7 </shape>
8 ...
```

---

Listing 3.6: Circle

```
1 ...
2 <shape type=          >
3   <center>
4     <lat>...</lat>
5     <lon>...</lon>
6   </center>
7   </radius>...</radius>
8 </shape>
9 ...
```

---

Listing 3.7: Polygon

```
1 ...
2 <shape type=          >
3   <vertex>...</vertex>
4   ...
5 </shape>
6 ...
```

---

### 3.5.2 Policies and Attributes

This system will use attribute based mechanisms for usage management. The policies defined over content must therefore consist of rules that address usage over an ontology of possible user attributes of concern. We are specifically interested in a user's primary attributes: mission affiliation, clearance levels (both sensitivity and category), organization, and computational environment (consisting of both device and operating system). We also make decisions with respect to usage based on a secondary attribute, need-to-use.

Sets differ from orderings in the above table as sets denote membership with no associated value. Orderings on the other hand have distinct values increasing from left to right in the listed enumerations. For example, a user can be affiliated with a specific mission in Domain A, either `tropic_thunder` or `gallant_entry`, or both. That user is also associated with a sensitivity value, either `unclassified`, `secret`, or `topsecret`, where `topsecret` is the most sensitive and `unclassified` the least.

Need-to-use decisions are based on the current context in tandem with mission and organizational affiliation. We use attribute based control in these scenarios, in which we make access decisions based on the attributes of a requesting user rather than defined roles.

### Chapter 3. Experimental Configuration

<i>Dimension</i>	<i>Type</i>	<i>Required?</i>	<i>Domain A</i>	<i>Domain B</i>	<i>Domain C</i>
<i>Affiliation</i>	Set	Yes	tropic_thunder, gallant_entry	tropic_thunder, gallant_entry	tropic_thunder, curious_response
<i>Sensitivity</i>	Ordering	Yes	unclassified, secret, top_secret	unclassified secret, top_secret	unclassified, secret, top_secret
<i>Category</i>	Set	No	aqua, magenta, vermillion	alpha, beta, gamma	one, two, three
<i>Organization</i>	Set	Yes	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia
<i>Device</i>	Set	No	workstation, tablet, phone	workstation, phone	workstation, tablet

Table 3.2: All Possible Attributes for Usage Management Decisions

User attributes support defined policy elements. Not every policy attribute has a corresponding user attribute as not all policy attributes are associated with users. Some are associated with the user’s environment, like operating system or device.

Policies are evaluated either via direct set membership or via membership in a category in an ordering. Content can be affiliated with multiple sets with regard to set-oriented attributes. Likewise, users can belong to multiple sets as well. Both content and users will be associated with a single value from an ordering element, as that value is inclusive of lower values as well. For example, a user can be affiliated with both the `tropic_thunder` and `gallant_entry` missions, but only one of the clearance values of `uncleared`, `secret`, or `top secret`. In the case of clearance values, `secret` subsumes `uncleared`, so a user with a `secret` attribute set would be able to access any unclassified material.

In the scope of this project, we use a Ruby-based domain specific language (DSL)



### Chapter 3. Experimental Configuration

<i>Dimension</i>	<i>Type</i>	<i>Required?</i>	<i>Domain A</i>	<i>Domain B</i>	<i>Domain C</i>
<i>Affiliation</i>	Set	Yes	tropic_thunder, gallant_entry	tropic_thunder, gallant_entry	tropic_thunder, curious_response
<i>Clearance</i>	Ordering	Yes	unclassified, secret, top_secret	unclassified secret, top_secret	unclassified, secret, top_secret
<i>Category</i>	Set	No	aqua, magenta, vermillion	alpha, beta, gamma	one, two, three
<i>Organization</i>	Set	Yes	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia	Oceania, Eastasia, Urasia

Table 3.3: Possible Attributes for Usage Management Decisions specific to Users

to describe policies. In larger heterogeneous deployments, a standards-based alternative like XACML would be more suitable. This project however is not focused on developing a complete policy specification language, but rather on using one in a very dynamic environment. XACML, for example, is a very large and complete standard that would require a significant investment of effort to implement. It can also tend to be verbose. A simple DSL focused on our specific needs is a more efficient alternative that allows us to focus our time and effort on the goals of this work rather than implementation of a large standard.

#### Listing 3.8: Policy DSL Example

```
1 policy_set {
2   policy(:p1) {
3     match :all
4     rule(:mission_affiliation) { |x| x == :tropic_thunder }
5     rule(:sensitivity) { |x| x == :top_secret }
6   }
7
8   policy(:p2) {
9     include :p1
```

## Chapter 3. Experimental Configuration

```
10     match :all
11     rule(:device) { |d| d == :workstation || d == :phone }
12 }
13
14 policy(:p3) {
15     include :p1
16     match :one
17     rule(:category) { |c| c == :vermillion }
18     rule(:organization) { |o| == :oceania }
19 }
20 }
```

---

This is our simplified DSL supporting a subset of XACML elements. In this example, we have a base policy, p1, that all other policies inherit. That policy requires that all rules evaluate to true. p2 adds another rule based on devices, all of which must evaluate to true as well. Finally, p3 adds two additional rules, only one of which must evaluate to true for the policy to be fulfilled.

### 3.6 Primary Interfaces and Mappings

Each of the defined components have an associated interface defined over domain datatypes. These interfaces are implemented using Representational State Transfer (REST) semantics over Hypertext Transfer Protocol (HTTP), and the datatypes are represented in Extensible Markup Language (XML).

#### Listing 3.9: Key Artifact Datatypes

```
1 typedef policy_set string;
2 typedef artifact string;
3
4 struct artifact_descriptor {
5     policy_set policy_set;
6     artifact artifact;
7 };
8
```

## Chapter 3. Experimental Configuration

```
9 typedef sequence<artifact_descriptor> artifact_descriptor_list;
```

---

As shown in Listing ??, we deal primarily with two key datatypes, *artifacts* and *policy\_sets*. For the purpose of networked data transfer, both of these datatypes are formatted strings of XML and policy DSL data. An *artifact\_descriptor* combines an artifact with its associated set of policies. An *artifact\_descriptor\_list* is an unlimited sequence of *artifact\_descriptors*.

### Listing 3.10: Key Status Datatypes

```
1 enum status { unsecured, confidential, secret, top_secret };
2
3 struct link_status {
4     string name;
5     status status;
6 };
7
8 typedef sequence<link_status> link_status_list;
9
10 struct context {
11     date date;
12     link_status_list network_status;
13 };
```

---

Network status information is contained in *status* elements and grouped into a *context* structure, as shown in Listing 3.10. A *status\_list* is essentially a dictionary of network connection statuses organized by link name, where an edge is named by concatenating the edge nodes in any order. These node names are concatenated and separated by a pipe symbol, so that the edge between *NodeA* and *NodeB* is named *NodeA|NodeB* or *NodeB|NodeA*. This makes searching less efficient, in that a Context can contain a *status\_list* with names in either ordering, in exchange for easier and more terse data exchange.

### Listing 3.11: Key Error Datatypes

```
1 exception error {
2     string message;
```

### Chapter 3. Experimental Configuration

```
3 };  
4  
5 exception client_error : error {};  
6 exception server_error : error {};  
7 exception unknown_response_error : error {};
```

---

Finally, shown in Listing 3.11, the *error* exception is represented by standard HTTP error codes and responses operationally, and is used extensively throughout system interface operations. Other information can be included in exception messages if the errors are not HTTP specific.

The *artifact\_manager* interface described in Listing 3.12. This interface is mapped to a REST style request over HTTP where the argument ordering is preserved when building the URL for accessing artifact content. For example, when accessing a specific artifact, the artifact operation called with a username of 'truchas', on an iphone, for artifact X1234 would map to the URL `http://host/artifact/truchas/iphone/X1234`. Likewise, a similar operation call on the artifacts operation would use the URL `http://host/artifacts/truchas/iphone`. Both Nodes and Routers implement the *artifact\_manager* interface.

#### Listing 3.12: The Node Interface

```
1 typedef string user_name;  
2 typedef user_name subject;  
3 typedef string key;  
4  
5 enum device { tablet, phone, workstation };  
6  
7 interface artifact_manager {  
8     artifact_descriptor get_artifact(in subject s, in device d, in key k) raises (error)  
9     ;  
10    artifact_descriptor_list get_all_artifacts(in subject s, in device d) raises (error)  
11    ;  
12 };
```

---

### Chapter 3. Experimental Configuration

This type of calling convention is used throughout the system. The specific ordering of the URL elements stems from corresponding artifact set relationships. Specifically, the set of all artifacts a user has access to is the same as or larger than the set of all artifacts that a user on a specific device can access and that subset is then the same size or smaller than the set of all available artifacts.

#### Listing 3.13: The Context Manager Interface

```
1 interface ContextManager {  
2   context context() raises (NetworkError);  
3 };
```

---

The *ContextManager* interface defined in Listing 3.13 describes how the network context monitor exposes network state information to requestors. Note, in this case, the defined interface maps to the URL `http://host/context`.

The *usage\_management\_mechanism* makes decisions with respect to proposed activities based on a set of policies and the current dynamic environmental context.

#### Listing 3.14: The Usage Management Mechanism Interface

```
1 enum activity { read, transmit };  
2 typedef string policy;  
3  
4 [Constructor(in ContextManager contextManager);]  
5 interface usage_management_mechanism {  
6   bool can_execute(in policy p, in context c, in activity a);  
7 };
```

---

The *repository* interface shown in listing 3.15 defines how information is stored and retrieved within a given node or router. This is an internal component used for concrete data item storage.

#### Listing 3.15: The Repository Interface

```
1 interface repository {
```

### Chapter 3. Experimental Configuration

```
2 artifact_descriptor get_artifact(in key k);
3 artifact_descriptor_list get_all_artifacts();
4 };
```

---

Finally, the *dispatcher* interface, also used internally within a given node or router, describes how we pass requests to other network participants.

#### Listing 3.16: The Dispatcher Interface

```
1 [Constructor(in string host);]
2 interface dispatcher {
3   artifact_descriptor_list dispatch(in user u, in device d, in key k) raises (error);
4 };
```

---

## **Chapter 4**

# **Experimental Results and Conclusions**

Experiments using this inter-cloud framework yield promising support for this approach. Experiments show only a slight degradation of information availability as a result of this network permeated security approach, with redaction and encryption demonstrating the smallest degradation at a higher impact on delivered information integrity. Rerouting-based approaches have the most performance degradation. Encryption generally has the smallest impact on information integrity. This is most evident when network effects are removed from evaluation. Non-hierarchical and hierarchical networks have very similar performance with respect to content availability as well.

The goal of this experimental work was to characterize the issues around specific confidentiality strategies in these kinds of networks. The specific strategies addressed were redaction, rerouting, and protection (via encryption), and these strategies were evaluated from the perspective of confidentiality, integrity, and availability over hierarchical and non-hierarchical networks, and on standalone nodes. Confidentiality was measured via the control used to protect information. Removing information entirely provided the highest measure of protection but is akin to unplugging a computer to improve its cyber-security posture. Routing information through a more secure channel is the next approach, followed

by sensitive information protection via strong encryption. We use 256-bit AES-CBC encryption scheme in this work. We measured availability by the delivery of information and the time required to ensure information delivery, measured by end-to-end network performance. Integrity is a function of the alterations to the information required for secure delivery in the tested scenario. Unaltered information has the highest integrity, followed by information that is still complete but protected via encryption, information that has been divided and rerouted, and finally information that has had content redacted. Though we can specify combinations of strategies in a given network, as we specify strategies by network node, in these experiments only a single strategy in each network was used to more clearly attribute strategy performance impacts. We used identical policies in each simulation to ensure the same amount of required usage management actions, limiting the effects on availability to the approach rather than differing policy. In each case, a control simulation that did not incorporate any usage management was run to provide a performance baseline.

## **4.1 Hierarchical Networks**

In these tests, a simulated  $\gamma$ -categorized system was examined. This is the kind of system that organizations like the UCDMO have identified as the final goal state of their work, systems that incorporate policy-centric management in the fabric of systems and networks (12). The kind of components required to do this kind of policy-based content-sensitive evaluation do not currently exist, and components of these kinds of systems are only now beginning to emerge. Systems like OpenFlow, when they have stronger hardware support, can begin to provide some of these kinds of capabilities. OpenFlow enabled systems are not yet common or widely used however, and though they do provide the needed control for these kinds of systems, they do not supply the necessary policy interpretation and evaluation. As a result, this experimental work was conducted over an HTTP overlay network, at the application layer. Using a document-focused protocol makes content evaluation simpler as



#### *Chapter 4. Experimental Results and Conclusions*

well, as systems can evaluate all content when it transits a network rather than maintaining a buffer of content required when processing packet-level communications.

In order to develop a stronger perspective on the network performance, we measure delivery times from three separate nodes. One node is hosted in Comcast's infrastructure (a large local Internet Service Provider), one at Amazon, and another at Rackspace. The tested network has four levels. The first level has a single router node. The next level has two routers, both connected to the router in the first level. The third level contains four routers, two attached to each of the routers at the level just above. Finally, the fourth level contains nodes, distributed so that two level three routers have three nodes, one level three router has two nodes, and the last level three router has four nodes. The first three levels are essentially a binary tree. We query the network from five different locations. We query the node that contains the content requested directly (the home node). We then query a node under the same router as the home node (the peer node). Next, we query a node under a different router, but connected to the same second level router (the neighbor node). Finally, we query two nodes on the other side of the network (the distant (1) and (2) nodes). We query each node 50 times in each simulation, for a total of 250 queries per simulation.

Figure 4.1 shows performance results from the Amazon testing node. The access times for the content from the home, peer, and neighbor nodes were by far the smallest. As the testing node was hosted in the same datacenter as these three nodes, that was to be expected. The access times for both distant nodes was, however, surprisingly high. With that in mind, the overall trend for response times is sensible however, with access time increasing as the requesting node is farther away from the content in the information network. Queries from distant nodes need to traverse five information routers, while home, peer, and neighbor nodes only traverse one, two and three, respectively. Also surprising was the finding that rerouting was generally more expensive from an availability perspective than encryption-based approaches. This is likely attributable to the costs associated with attaching to the external SMTP server, hosted at Google, used as the out-of-band communications channel.

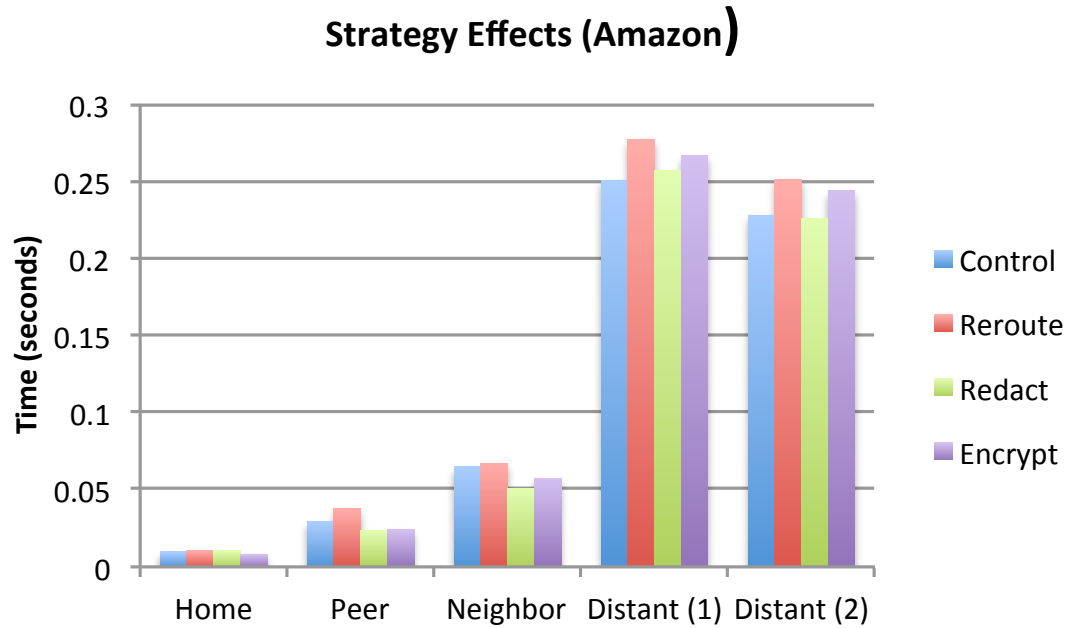


Figure 4.1: Hierarchical Results from Amazon

Also evident is remarkable performance variability. Control data was collected at different times than experimental data, and infrastructural demands seem to have driven the control data availability to be less than that of other, managed approaches. Overall, this evidence of variable performance due to external provider demands leads to the conclusion that overall, the availability costs of the various approaches are in fact negligible.

Figure 4.2 shows similar results to Figure 4.1. Here, the query times are much higher for the home and peer nodes, but actually lower for the distant nodes. In this case, the content is still hosted in Amazon's infrastructure, but the testing node is at Rackspace. As a result, the longer response time for content from the home node is to be expected. Queries to distant nodes are actually shorter than the previous calls into distant nodes from Amazon. This stems from the fact that the distant nodes are both hosted at Rackspace. This locality shortens the round trip distance for a request. Previously, from Amazon, a content request would need to travel from Amazon's east coast data centers to the Rackspace data center in

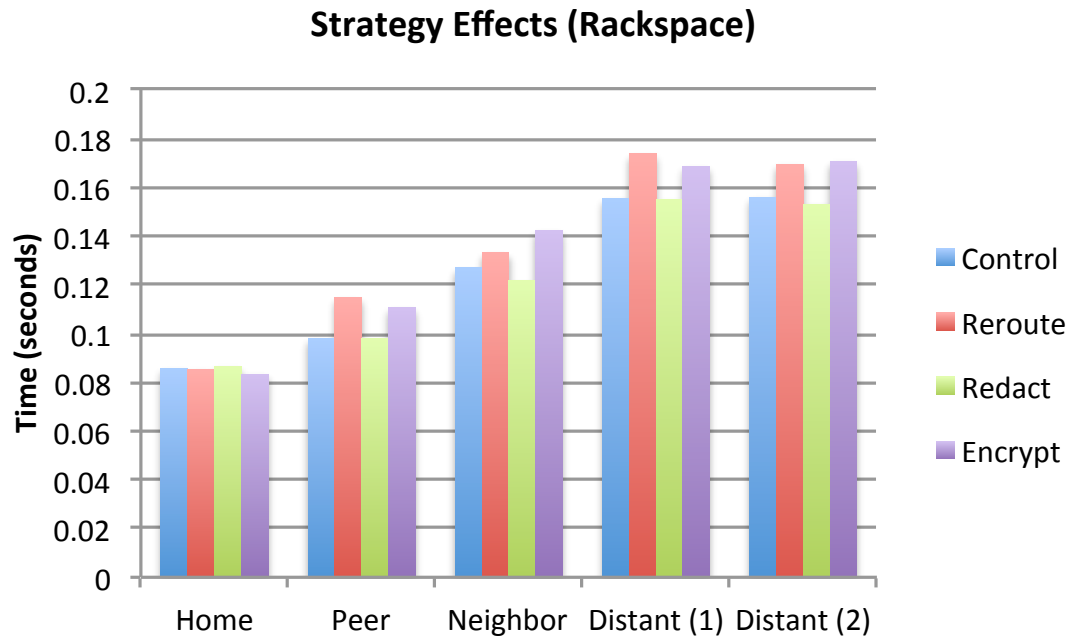


Figure 4.2: Hierarchical Results from Rackspace

Dallas, then back to the east coast for content, then back to Dallas, then back to the east coast. In this test, the request only travels from Dallas to the east coast, and back. Nevertheless, the overall performance profile is sensible, reflecting the expected shorter latency between home, peer, and neighbor nodes when compared to distant nodes. Similar to amazon, we again have cases when the control latency is higher than experimental latency, indicating some amount of infrastructure performance variability. In Figure 4.2 however, we see that overall encryption and rerouting impact performance more than rerouting, as we would expect. Rerouting again has high overall impact, likely as a result of contacting Google's remote SMTP services.

Figure 4.3 Shows performance results measured from Comcast. Interestingly, they show significant variability when accessing nodes hosted at Amazon, and more predictable performance when accessing nodes in Rackspace's infrastructure. The overall variability does not follow the expected pattern of shorter response times when accessing content from

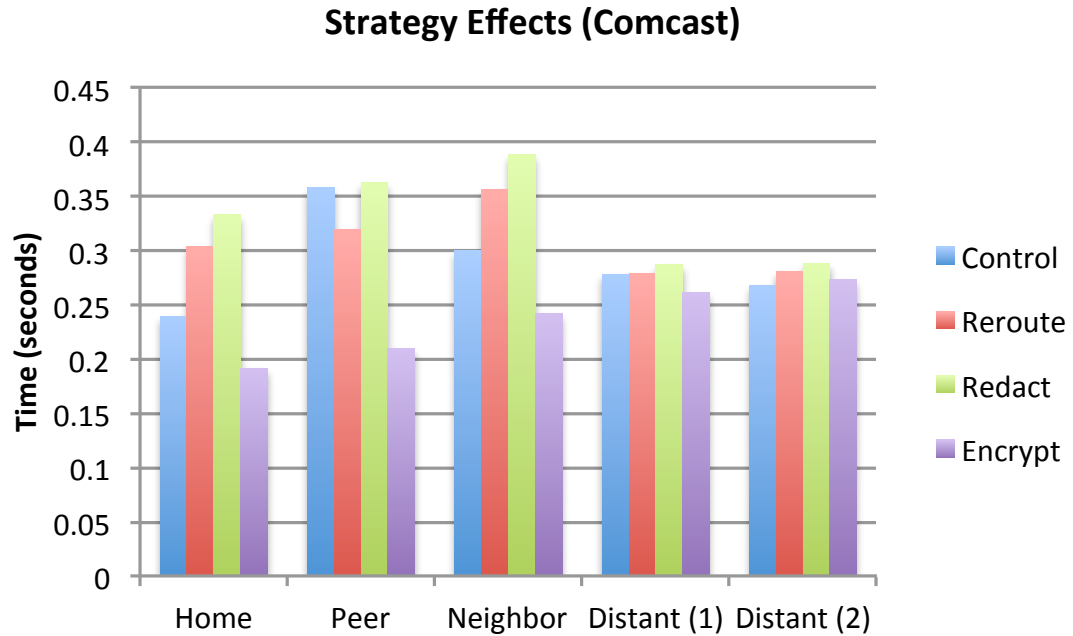


Figure 4.3: Hierarchical Results from Comcast

nodes close to that content, except in a few cases. This illustrates the kind of performance variability one can expect from an external service provider.

Integrity impacts are the result of approach rather than platform. Redacting content destroys information integrity, as information is removed and not delivered to requesters. Encryption maintains integrity the best of the three alternatives as information, even though encrypted, is still delivered, and delivered in the context of the query response at that. Rerouting is better than redaction, in that sensitive information is still delivered, but worse than encryption, as it is not delivered within the response context and is sent out-of-band. Simulations removed sensitive information from the information network and dispatched it to a user's email address via SMTP over TLS when the selected strategy was rerouting. This impacts information availability, as email delivery times can be highly variable. In these experiments, delivery could take anything from a few seconds to a few minutes.

#### *Chapter 4. Experimental Results and Conclusions*

Confidentiality is likewise impacted primarily by approach and not by infrastructure. Redacting sensitive content provides the best confidentiality protection, as sensitive content is simply not exposed. Encryption is likely the worst solution from a confidentiality perspective as content encryption is a delaying tactic against a determined, well-resourced adversary. Rerouting may be better or worse than encryption as an approach, depending on the confidentiality of the out-of-band channel. If the security of that channel can be guaranteed, then it is likely a better approach. If, on the other hand, the security of that channel is more variable or difficult to ascertain, encryption may be a more reliable approach.

Overall, results show that, from a performance perspective, the encryption approach fares the worst, but only slightly, and certainly not in all cases. Both results from Amazon and Rackspace, in Figures 4.1 and 4.2, show encryption as generally taking the largest performance hit, just following rerouting. Furthermore, network effects have a much larger impact on performance than information protection approaches. The query to the home node is an excellent predictor of overall network stability, as content delivered directly from a home node is not subjected to the selected information protection strategy. Note that when queried from Amazon or Rackspace, the home node timing results are very close to uniform. Queries from Comcast, however, are much more varied, indicating more highly variable quality of service within the Comcast network. This is also supported by the gross distribution of response times. Within both the Amazon and Rackspace networks, the farther a queried node is from the content requested, the worse the performance, as expected. Comcast's network has a much more uniform information network response time overall as the processing time of the information network simulation is overshadowed by the highly varied performance of Comcast's physical network. Availability is surprisingly uniform across all confidentiality strategies, showing little impact on end-to-end processing times. Encryption strategies show the most degradation, though that performance degradation is less than general network performance variation.

## 4.2 Non-Hierarchical Networks

In order to test non-hierarchical networks, a simple branching network of participants was used, identical in form to the hierarchical network, though queries could be routed through the network from any point. Queries could come into any node on the network, and would propagate through the network to the requested content, evaluating the returned content as it passes back through the network in response to the initial query.

In these experiments, the node that contains the content was queried, then the node immediately next to that content node, and so on, to a distance of five nodes. The home node again contains content, and the additional nodes are marked by the distance in node count from the home node, starting with Neighbor (1), proceeding through Neighbor (5). We queried the non-hierarchical network from Rackspace, Amazon, and Comcast, for a total of 250 queries per test, testing the system once per each confidentiality strategy.

Figure 4.4 shows the performance of a non-hierarchical network as tested from the Amazon test node. The content response latency is characteristic of moving farther from the source node through the network. The request nodes switch from Amazon infrastructure to Rackspace infrastructure starting with Neighbor (3), and this is reflected in the sudden increase in latency. As the tests originate from Amazon, at the Neighbor (3) node, a request and its response must travel from Virginia to Texas, then back to Virginia, then back to Texas, then back to the original requester in Virginia. The spike in latency at Neighbor (3) when re-routing traffic is caused by SMTP delays with systems hosted at Google. Overall, the distribution is very similar to the hierarchical case. Also evident is a continuation of the previous pattern in which re-routing is the least efficient strategy, followed by encryption, then redaction.

Unlike our previous Amazon-based tests, the Rackspace tests shown in Figure 4.5 latencies seem much more uniform. This again stems from the fact that each content query will always traverse the distance between Amazon and Rackspace data centers at least

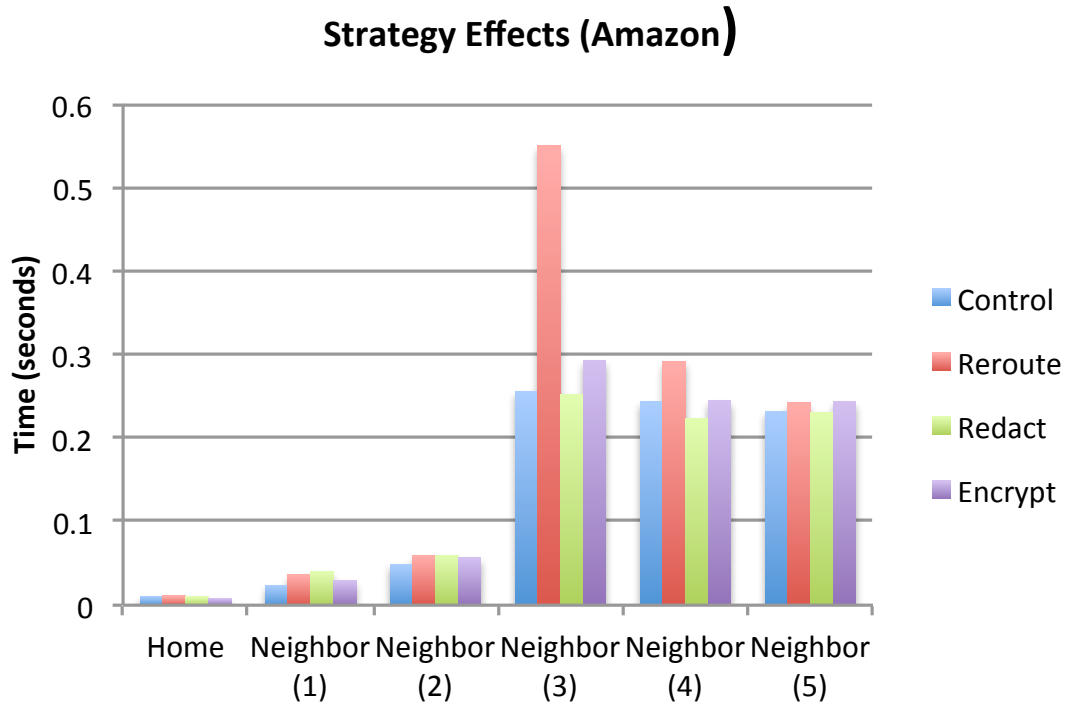


Figure 4.4: Non-Hierarchical Results from Amazon

once. Other than that, the distribution again shows an increase in measured latency as the queried node moves farther and farther away from the home node. Once again, we have a dramatic spike in latency based on SMTP delays when re-routing information. The pattern of re-routing having the highest latency continues here as well.

Results from Comcast, included in Figure 4.6, shows a fairly regular distribution of response latencies overall. In this case, the test node is ensconced within Comcast's network infrastructure. Generally, re-routing is the least efficient approach, but not uniformly. In this case, network effects created by the physical location of the testing node dominate our results.

Non-hierarchical networks behave very similarly to hierarchical networks. This is not surprising — although the nodes are more functionally complex, performing routing and

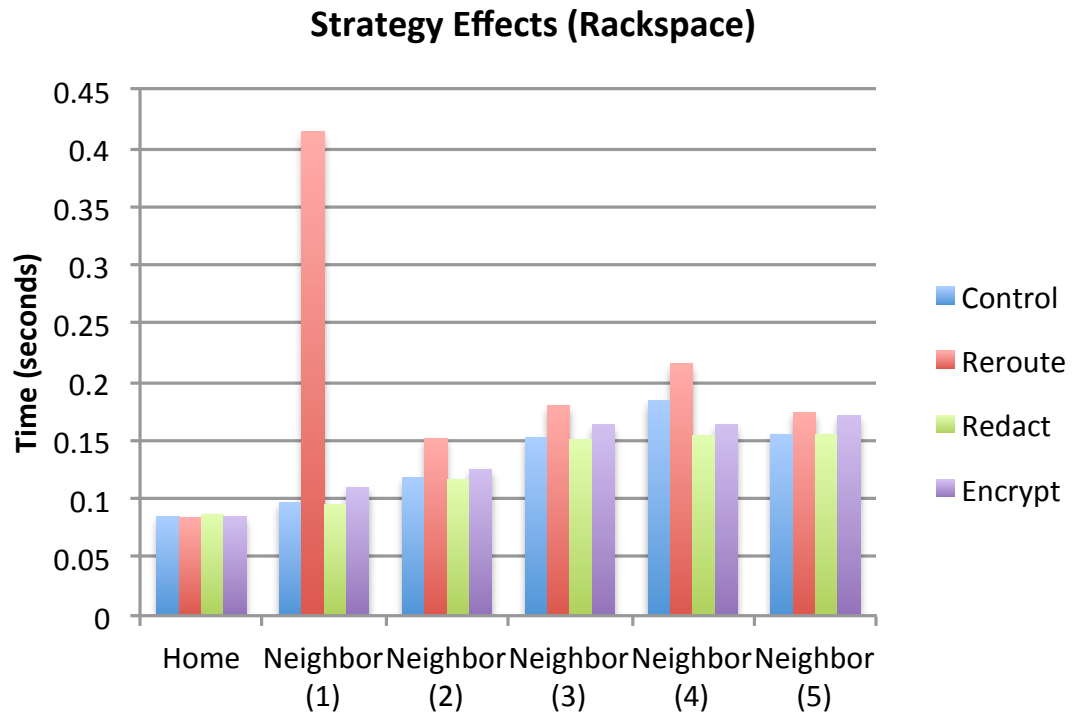


Figure 4.5: Non-Hierarchical Results from Rackspace

repository functions, once the content is found and delivered the roles the nodes fall into mirror those in a hierarchical network. For example, in a typical query, a node will receive a request, check the repository for the requested content, and if the content doesn't exist, pass the request onto the next known nodes. This does differ slightly from the hierarchical case in that the nodes check for content at each routing step, though this is a very fast and simple test. Once content is found, the response is routed back the the requester without any repository checks, just as it would be in a hierarchical system.



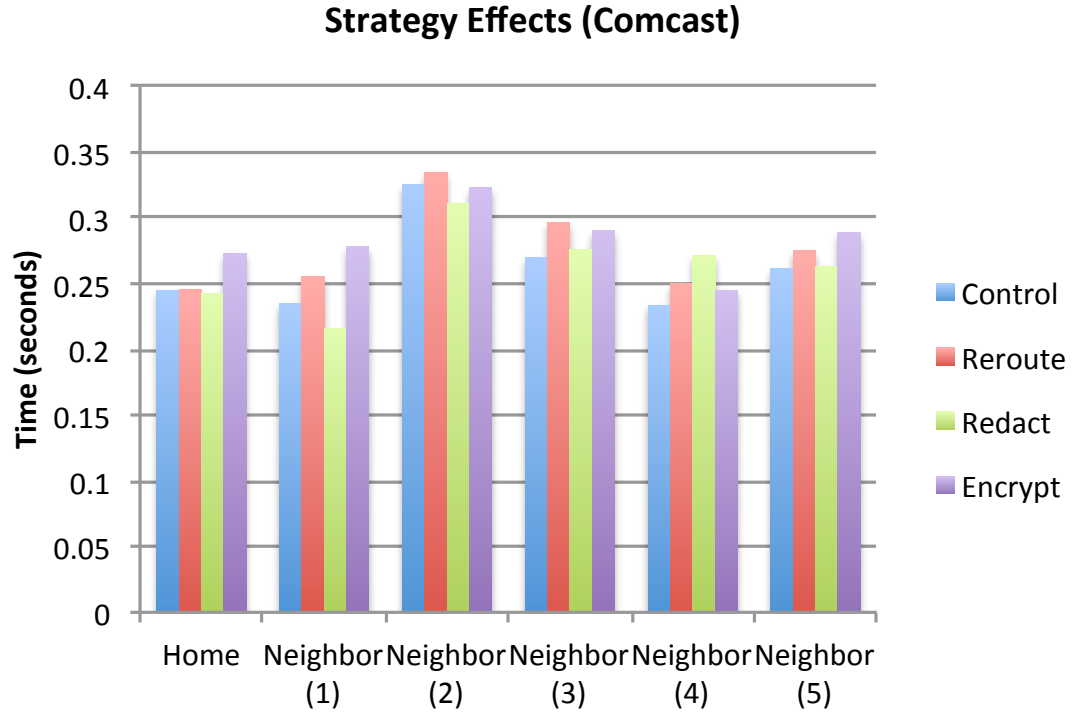


Figure 4.6: Non-Hierarchical Results from Comcast

### 4.3 Removing Network Effects

Having established the parameters under which confidentiality strategies may be chosen, the next immediate area of concern involves the number of filtering events that can occur prior to a given information network suffering from degraded performance. Previous results demonstrated that we do in fact have some kind of degradation of performance in or selected network based on distance from content, but that can also be attributed to the distributed nature of the network itself. Processing performance of a given node must be evaluated free of network effects in order to more clearly understand the availability implications of content filtering itself.

A single node, configured on one of the test nodes in either infrastructure, would

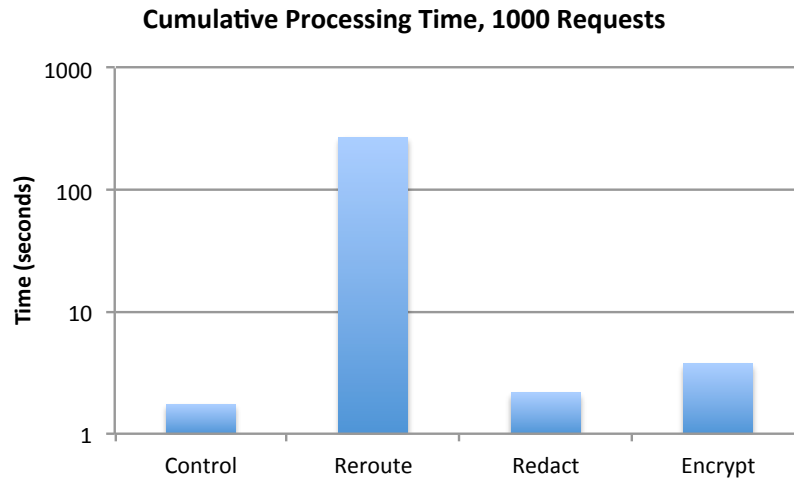


Figure 4.7: Results from Requests to a Singe Node

yield the type of network effect free performance limits needed. A node in Amazon’s environments were configured such that requests were made of the home node itself, under each of the three confidentiality strategies. Requests were also directed to the home node without any usage management systems engaged in order to collect control data. After that initial request, the node was configured with various usage management strategies in order to measure their availability impact.

As shown in Figure 4.7, with information network effects removed, readaction and encryption have very similar performance overall. Redaction, as a strategy, is very simple programmatically, and as we use symmetric encryption, ciphering and deciphering operations are very fast. Rerouting, in this case, is clearly the worst strategy. This is a result of the dependency of this strategy on external systems and system configuration times. Specifically, configuring and using SMTP for each rerouting operation is prohibitively expensive.

## **4.4 Conclusions**

The work described herein presents bounds under which to select specific confidentiality strategies for protecting information in content networks. We first described the state of the art of this kind of information protection in content networks, and introduced the current accepted protection architectures sponsored by the UCDMO. We then presented a related taxonomy of increasing information protection, describing their advantages and disadvantages and how they could be implemented. Next, we described our current customizable experimental framework for evaluating various confidentiality strategies. We closed with a description of and the motivation for our experiments over these networks, the results of these experiments, and analysis of those results. All simulation code is freely available via Github.

Overall, confidentiality strategy had little impact on information availability. Redaction, rerouting, and encryption all performed within similar bounds. Of these three approaches, redaction damaged information integrity the most, followed by rerouting, and then encryption, depending on the security of rerouting infrastructure. Redaction provided the most confidentiality, followed by rerouting, and then by encryption (as encrypted content is generally at best a delaying tactic given enough time for cryptanalysis). Based on these results, rerouting is likely the best general solution, depending on the existence of a secondary secure channel. Less sensitive information can still be delivered via encryption, especially if that information is only sensitive within a given time window. Very sensitive information can be redacted, but due to the related damage to integrity, this is only an attractive option when confidentiality is of the utmost importance.

Non-hierarchical and hierarchical networks performed similarly. There was no significant difference in availability between networks with respect to confidentiality strategies. Different network topologies certainly have different characteristics with respect to reliability as a result of selected architectures however, specifically with respect to the centralization

<i>Property</i>	<i>Redaction</i>	<i>Rerouting</i>	<i>Encryption</i>
<i>Infrastructure</i>	3	1	1
<i>Operating Systems</i>	1	2	3
<i>Technologies</i>	1	1	1

Table 4.1: Approach Evaluation Summary

or decentralization of key functions, but that analysis is outside the scope of this work.

Table 4.1 shows the overall results of experiments and analysis with respect to various possible approaches to securing information transiting content networks, on a scale of one to three, with three the highest and one the lowest scores. Not surprisingly, there is no clear best approach. Rather, decisions with respect to which approach to choose for given content is highly dependent on the sensitivity of the content as well as integrity and availability requirements.

At this point, our information network implementation has integrated three different configurable strategies for information protection, and routes information via an overlay network using HTTP. Longer term, this project will expand to both incorporate public-key encryption protocols and software defined networking (SDN) capabilities to provide physical control of information routing. We intend to provide public-key encryption capabilities via an integrated public key infrastructure providing additional privacy and non-repudiation abilities for the network and SDN capabilities via integration with OpenFlow. Shorter term goals include inclusion of different modes of operation, so that the network can support both request/response and publish/subscribe modes of operation, and more robust development so the system can run as a commercial grade security-on-demand service.

# References

- [1] Marlin architecture overview. Technical report, 2006.  
<http://www.marlin-community.com>.
- [2] DoD Information Sharing Strategy. <http://dodcio.defense.gov/Portals/0/Documents/InfoSharingStrategy.pdf>, May 2007.
- [3] Assured Information Sharing in Clouds. <http://www.zyn.com/sbir/sbres/sttr/dod/af/af11-bt30.htm>, August 2011.
- [4] Department of Defense Global Information Grid Architectural Vision. <http://cio-nii.defense.gov/docs/GIGArchVision.pdf>, 2011.
- [5] NSA Pursues Intelligence-Sharing Architecture. <http://www.informationweek.com/news/government/cloud-saas/229401646>, April 2011.
- [6] Openflow - Enabling Innovation in Your Network. <http://www.openflow.org>, November 2011.
- [7] About the UCDMO. <http://www.ucdmov.gov/about.html>, January 2012.
- [8] Benefits of Content-Centric Networking. <http://www.xerox.com/innovation/news-stories/networking/enus.html>, September 2012.
- [9] CCNx. <http://www.ccnx.org/>, September 2012.
- [10] Cisco Catalyst 6500 Series Switches. <http://www.cisco.com/en/US/products/hw/switches/ps708/index.html>, September 2012.
- [11] Content-Centric Networking - Parc, a Xerox Company. <http://www.parc.com/work/focus-area/content-centric-networking/>, September 2012.

## References

- [12] JBoss ESB. <http://www.jboss.org/jbossesb>, January 2012.
- [13] Named Data Networking. <http://named-data.org/>, September 2012.
- [14] NetInf | SAIL. <http://www.sail-project.eu/about-sail/netinf/>, September 2012.
- [15] PSIRP. <http://www.psirp.org/>, September 2012.
- [16] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *Communications Magazine, IEEE*, 50(7):26–36, july 2012.
- [17] D. E. Bell and L. J. LaPadula. Secure Computer Systems: Mathematical Foundations. Technical report, MITRE Corporation, March 1973.
- [18] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform resource identifier (URI): generic syntax. RFC 3986, Internet Engineering Task Force, January 2005.
- [19] Biba. Integrity Considerations for Secure Computer Systems. *MITRE Co., technical report ESD-TR 76-372*, 1977.
- [20] Marjory S. Blumenthal and David D. Clark. Rethinking the design of the internet: the end-to-end arguments vs. the brave new world. *ACM Trans. Internet Technol.*, 1:70–109, August 2001.
- [21] Marjory S. Blumenthal and David D. Clark. Rethinking the design of the Internet: The end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology*, 1(1):70–109, Aug. 2001.
- [22] Booz, Allen, and Hamilton. Distributed service oriented architecture (soa) compatible cross domain service (dscds). Presented at the Unified Cross Domain Management Office Conference, 2009.
- [23] D. F. C. Brewer and M. J. Nash. The Chinese wall security policy. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 206–214, 1989.
- [24] D. D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *1987 IEEE Symposium on Security and Privacy*, pages 184–194. IEEE Computer Society Press, 1987.
- [25] David D. Clark. The design philosophy of the DARPA internet protocols. In *ACM SIGCOMM*, pages 106–114, Stanford, CA, Aug. 1988.
- [26] David D. Clark. The design philosophy of the darpa internet protocols. *SIGCOMM Comput. Commun. Rev.*, 25:102–111, January 1995.

## References

- [27] David D. Clark, John Wroclawski, Karen R. Sollins, and Robert Braden. Tussle in cyberspace: Defining tomorrow's internet. In *SIGCOMM*, pages 347–356, Pittsburg, Pennsylvania, USA, Aug. 2002.
- [28] Coral consortium whitepaper. Technical report, Feb. 2006. [www.coral-interop.org/main/news/Coral.whitepaper.pdf](http://www.coral-interop.org/main/news/Coral.whitepaper.pdf).
- [29] M. D'Ambrosio, P. Fasano, M. Marchisio, V. Vercellone, and M. Ullio. Providing data dissemination services in the future internet. In *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*, pages 1 –6, 30 2008-dec. 4 2008.
- [30] R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, L. Masinter, P. J. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.
- [31] Ali Ghodsi, Teemu Koponen, Jarno Rajahalme, Pasi Sarolahti, and Scott Shenker. Naming in content-oriented architectures. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking, ICN '11*, pages 1–6, New York, NY, USA, 2011. ACM.
- [32] Gregory L. Heileman and Pramod A. Jamkhedkar. DRM interoperability analysis from the perspective of a layered framework. In *Proceedings of the Fifth ACM Workshop on Digital Rights Management*, pages 17–26, Alexandria, VA, Nov. 2005.
- [33] Pramod A. Jamkhedkar and Gregory L. Heileman. DRM as a layered system. In *Proceedings of the Fourth ACM Workshop on Digital Rights Management*, pages 11–21, Washington, DC, Oct. 2004.
- [34] Pramod A. Jamkhedkar and Gregory L. Heileman. Digital rights management architectures. *Computers Electrical Engineering*, 35(2):376–394, 2009.
- [35] Pramod A. Jamkhedkar, Gregory L. Heileman, and Chris Lamb. An Interoperable Usage Management Framework. In *Proceedings of the Tenth ACM Workshop on Digital Rights Management*, Chicago, Oct. 2010.
- [36] Pramod A. Jamkhedkar, Gregory L. Heileman, and Ivan Martinez-Ortiz. The problem with rights expression languages. In *Proceedings of the Sixth ACM Workshop on Digital Rights Management*, pages 59–67, Alexandria, VA, Nov. 2006.
- [37] Rob H. Koenen, Jack Lacy, Michael MacKay, and Steve Mitchell. The long march to interoperable digital rights management. *Proceedings of the IEEE*, 92(6):883–897, 2004.

## References

- [38] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, Kye Hyun Kim, Scott Shenker, and Ion Stoica. A data-oriented (and beyond) network architecture. *SIGCOMM Comput. Commun. Rev.*, 37(4):181–192, August 2007.
- [39] NSA. Distributed service oriented architecture (soa)- compatible cross domain service (dscds) dscds overview. Presented at the Unified Cross Domain Management Office Conference, 2009.
- [40] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.
- [41] Unified Cross Domain Management Office. Cd101. Presented at the Unified Cross Domain Management Office Conference, 2009.
- [42] Jason Ostermann. Raytheon dscds intro. Presented at the Unified Cross Domain Management Office Conference, 2009.
- [43] Jaehong Park and Ravi Sandhu. The  $U\text{CON}_{ABC}$  usage control model. *ACM Trans. Inf. Syst. Secur.*, 7(1):128–174, 2004.
- [44] S. Pearson and A. Benameur. Privacy, security and trust issues arising from cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 693 –702, 30 2010-dec. 3 2010.
- [45] G.M. Perez, F.J.G. Clemente, and A.F.G. Skarmeta. Building and managing policy-based secure overlay networks. In *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, pages 597 –603, feb. 2008.
- [46] J. B. Postel. Internet protocol. RFC 791, Internet Engineering Task Force, September 1981.
- [47] J. B. Postel. Transmission control protocol. RFC 793, Internet Engineering Task Force, September 1981.
- [48] Ron Ross. Next generation risk management. Presented at the Unified Cross Domain Management Office Conference, 2009.
- [49] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions in Computer Systems*, 2(4):277–288, Nov. 1984.
- [50] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1985.
- [51] Sasu Tarkoma. *Overlay Networks: Toward Information Networking*. Auerbach Publications, Boston, MA, USA, 1st edition, 2010.