A Network-Agnostic Data Framework and API for CCN

Glenn Scott
Palo Alto Research Center, Palo Alto, CA 94304
Glenn.Scott@parc.com

Christopher A. Wood
University of California, Irvine, CA 92617
woodc1@uci.edu

ABSTRACT

We introduce the CCN Information and Data Framework (IDF). The IDF provides a Create, Read, Update, and Delete (CRUD) interface for reading and manipulating data "in the network." We show how to use the IDF API through the UNIX file I/O API, thereby placing file I/O and network communication under a single, standard interface. The IDF effectively creates a name-based layer of abstraction upon which *network-agnostic* applications can be built.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*

Keywords

Content-centric networking; data framework; API

1. INTRODUCTION

The suite of protocols that exist in the current TCP/IP Internet model do not adequately satisfy modern application use cases which are heavily *based on data*. All of the abstractions provided by these protocol interfaces are inherently location-dependent and provide little more than packet transmission and receipt functionality. In our increasingly content-centric world, abstractions for content are needed to build modern applications and services. Therefore, in addition to reconsidering the Internet architecture, we must also consider the interfaces by which applications make use of it.

Content-Centric Networking (CCN) is an approach to (inter-)networking designed for efficient, secure, and scalable content distribution [1, 4]. In CCN, named content, rather than named interfaces or hosts, are treated as first-class entities that are explicitly requested via interest messages. By decoupling data from its location, content can be cached within the network to optimize bandwidth use, reduce latency, and use multiple network interfaces simultaneously.

ICN'15, Sept.30–Oct. 2, 2015, San Francisco, CA, USA.
ACM 978-1-4503-3855-4/15/09.
DOI: http://dx.doi.org/10.1145/2810156.2812606 .

Apart from the aforementioned performance improvements, an important benefit of the content-centric approach to networking is that it introduces an abstraction layer in the form of requests and content responses between applications and the location where content is stored. In CCNx [1], this layer is used with the CCN Portal. The Portal API enables applications to be constructed in a *location-agnostic* way using the discrete interest and content object messages.

Many modern systems and applications rely on a file as a high-level abstraction upon which other data access mechanisms, e.g., video streams, databases, messages, and key-value stores, are implemented. While interests and content objects serve as a vehicle for transferring *raw* data, they cannot be directly used as these various abstractions without another layer of indirection. These realizations led us to the main contribution of this work: the Information and Data Framework (IDF). This component builds on top of the Portal API to provide a Create, Read, Update, Delete (CRUD) API – the Information and Data Interface (IDI) – for applications to read and manipulate network data. Internally, it interfaces with the operating system processes and remote services needed to compose content objects into these abstractions. To illustrate the efficacy of the IDF, we designed a mapping from the Unix File API to the IDI, which effectively places file I/O and network communication (for content retrieval) under a single interface.

2. THE CCN PORTAL BEDROCK

In CCN, interests and content objects are the primary elements of discourse in CCN. Consumers issue interests for data and receive content objects in response; they do not need to know the details of the transport stack to communicate. The actual contents, semantics, and representation of both interests and content objects within the network stack are entirely dependent on its internal components. Consequently, a natural abstraction for interfacing with CCN is a single interface through which discrete interest and content object messages flow. In CCN, this interface is called the **Portal**. The Portal is a minimal interface used to communicate with the **transport stack** and the network. The Portal API provides a simple interface to the transport stack allowing the application to compose, use, and maintain transport stacks and to perform discrete message operations (e.g., send interests and content objects) through the stack.

3. INFORMATION AND DATA FRAMEWORK

Unix files are an extremely powerful abstraction upon which many different applications (e.g., grep, sed, awk),

Table 1: IDI: The Information and Data Interface API.

Function Signature	Description
`(success, manifest) = IDFCreate(lciName, ACS)`	Make a file available with the given LCI encoded name and access information encoded in a CCN `AccessControlSpecification`.
`buffer = IDFRead(lciName, numbytes=0, offset=0)`	Read data (of numbytes size) from the specified name starting at the given offset.
`count = IDFUpdate(lciName, buffer, count, offset=0)`	Update the file with the given name and offset (default 0) with the specified data.
`success = IDFDelete(lciName)`	Make a file that was previously available, unavailable.

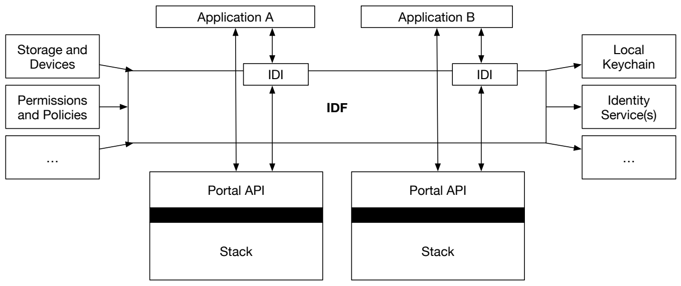

Figure 1: Component-level diagram of the IDF.

APIs, and large systems (e.g., databases) have been built. We borrow from its success to build the CCN **Information and Data Framework** (IDF), a framework that collates other services such as network communication, access control and permissions, and concurrency synchronization to provide a single, simple, file-like API. The framework is an *extension of the Portal and transport stack* that uses external services and local operating system resources to perform a variety of useful functions and provide a lightweight CRUD-like API to upper-level layers in the stack. As illustrated in Figure 1, some of these local system resources may be storage and media devices, system permissions and policies, and the local keychain. The IDF will coordinate interaction with these resources to determine whether upper-level clients can perform simple read and write behaviors, for example. The IDF may also be extended to communicate with external identity services, e.g., via LDAP [3] or OAuth [2], to obtain authentication information and other related data. These services may also provide cryptographic secrets necessary to access content protected under some form of access control, e.g., by encryption.

The goal of the IDF is to free developers from devising and implementing new protocols and mechanisms to publish and retrieve content, similar to the DOT data transfer architecture presented in [5]. The IDF API – the Information and Data Interface (IDI) – follows the CRUD (create, read, update, delete) model with ACID guarantees. Instead of file handles or network sockets, for example, each of these CRUD functions use only the name of the content in question. This name-based abstraction frees developers from the details of network communication and other content stores (i.e., filesystems). A high-level description of this API is summarized in Table 1 to illustrate its simplicity.

From the application's perspective, the IDF is meant as a more useful abstraction beyond interest and content objects. For example, applications may instantiate file instances using the framework and then read and write to them, much like any normal file. From the system's perspective, the IDF coordinates many working services to give further meaning to the atomic interest and content object messages provided by CCN, such as: object reconstruction and access control, local operating environment access, content publication, remote authentication, and content state management.

4. NETWORK-AGNOSTIC CONTENT

The IDI enables clients to interact and treat network data as if it were *persistent* – a paradox in today's world. It enables an application to switch – on the fly – from local file based access to network based access through the same interface, much like the Juno middleware platform that enabled location and protocol agnostic content retrieval [6]. As previously mentioned, persistent data access has traditionally been access via the Unix File API. Since the IDF provides a layer of abstraction for location-agnostic persistent data, it may also be used by the very same Unix File API. To do this, we designed and specified an adapter that maps the Unix File API operations to underlying IDF functions. This adapter enables multiple benefits, including: (a) existing applications can migrate to CCN with minimal development effort, (b) richer APIs can be built upon the concept of random access files, e.g., key-value store, messaging, and data streaming APIs, and (c) security concerns such as encryption and access control are handled beneath the API.

5. REFERENCES

[1] CCNx 1.0 Protocol SpeciïňĄcations Roadmap. `http://www.ietf.org/mail-archive/web/icnrg/current/pdfZyEQRE5tFS.pdf`.

[2] Eran Hammer-Lahav. The oauth 1.0 protocol. 2010.

[3] Timothy A Howes, Mark C Smith, and Gordon S Good. *Understanding and deploying LDAP directory services.* Addison-Wesley Longman Publishing Co., Inc., 2003.

[4] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, pages 1–12. ACM, 2009.

[5] Niraj Tolia, Michael Kaminsky, David G Andersen, and Swapnil Patil. An architecture for internet data transfer. In *NSDI*, 2006.

[6] Gareth Tyson, Andreas Mauthe, Sebastian Kaune, Paul Grace, Adel Taweel, and Thomas Plagemann. Juno: A middleware platform for supporting delivery-centric applications. *ACM Transactions on Internet Technology (TOIT)*, 12(2):4, 2012.