Named Data Networking: Motivation & Details

NDN is an entirely new architecture, but one whose design principles are derived from the successes of today's Internet, reflecting our understanding of the strengths and limitations of the current Internet architecture, and one that can be rolled out through incremental deployment over the current operational Internet.

Table of Contents

Context and Vision

In the 1960s and 70s when the core ideas underlying the Internet were developed, telephony was the only example of successful, effective, global-scale communications. Thus while the communication *solution* offered by TCP/IP was unique and groundbreaking, the *problem* it solved was telephony's: carrying a point-to-point conversation between two entities. IP has exceeded all expectations for facilitating ubiquitous interconnectivity and so enabled dramatic changes in the world that we associate with the Internet.

> Information-intensive business like travel, banks and financial services long ago moved onto the Internet. Today almost anything is available online as the Internet becomes the world's storefront.
>
> Moore's-Law-driven hardware advances have made it feasible to connect *everything* to the Internet: not just supercomputers and workstations but also factories, municipal infrastructure, phones, cars, appliances, even light switches. With the rapid growth in hand-held mobile devices such as smart phones and tablets, the Internet is also becoming increasingly mobile.
>
> Digital coding advances have turned not just text but voice, images and video into strings of bits so an ever increasing range of content is now distributed digitally. Moreover, since the Web made it easy for anyone to discover, consume, and create content, exabytes of new content are being produced and distributed yearly.

As a result of these changes, IP, despite being designed for conversations between communicating endpoints, is now overwhelmingly used for content distribution, both to stationary hosts and increasingly to mobile devices. Just as the telephone system would be a poor vehicle for the broadcast content distribution done by TV and radio, the Internet architecture is a poor match to its primary use today. In addition, malicious attacks, attracted by the tremendous economic value of Internet applications, have become daily events. Following the conversational model of IP communications, many efforts have been devoted to securing communication channels, yet security breaches continue to increase.

The “conversational” nature of IP is embodied in its datagram format: IP datagrams can only name communication endpoints (the IP source and destination addresses). As our project title suggests, we propose to generalize the Internet architecture by removing this restriction: the names in NDN datagrams are hierarchically structured but otherwise arbitrary data identifiers. They can be used to name a chunk of data in a conversation, as the TCP/IP transport connection identifier plus sequence number does today, but they can also name a chunk of data from a YouTube video directly, rather than forcing it to be embedded in a conversation between the consuming host and youtube.com. This simple change to the hourglass model, allowing the Internet’s thin waist to use data names instead of IP addresses for data delivery, makes data rather than its containers the first-class citizens in the Internet architecture.

This conceptually simple change creates an abundance of new opportunities:

Today’s applications are typically written in terms of *what* data they want rather than *where* it is located, then application-specific middleware is used to map between the application model and the Internet’s. With NDN the application’s *what* model can be used directly in data delivery, removing all the middleware and its associated configuration and communication inefficiencies.

Since conversations are ephemeral and can be about anything, the current security approach is the one-size-fits-all model of armoring the channel between two IP addresses, which rarely meets the end-to-end security needs of applications. In NDN, all data is signed by data producers and verified by the consumers, and the data name provides essential context for security. For example, NDN can tell if all the data on the web page one is viewing was produced and signed by one’s bank; IP cannot.

Since every chunk of data is uniquely named, an NDN data packet is meaningful independent of where it comes from or where it may be forwarded to, thus it can be cached inside the network to satisfy future requests. In addition, unique data packet names enable routers to maintain data plane state, which opens the door to a number of important functions that today’s IP routers are incapable of supporting. One of those functions is data looping elimination, which allows any node to freely use all of its connectivity to solicit or distribute data, and removes the information asymmetries that give today’s dominant providers disproportionate control over routes and thus over smaller, local providers.

The change also introduces significant intellectual challenges:

Using application names at the thin waist for data delivery not only can bring the great benefits mentioned above, but also brings up new design opportunities as well as open issues. Because NDN removes the translation from application data names to IP addresses, it raises the question of how an application should best select its data names to facilitate *both* the application development and network data delivery.

Name-based routing also raises a scalability question. IP addresses have a finite name space, while the NDN name space is unbounded. However we believe that by using hierarchical names, much like the URLs used to name today’s web content, NDN can effectively control the name space in global routing, in a way similar to how today’s Internet routing state scales via IP prefix aggregation. In addition, NDN’s delivery model allows routing and forwarding to operate with approximate state rather than IP’s exact state, which can potentially reduce NDN’s routing state burden even below that of IP.

Yet another dimension of scaling challenge is packet forwarding speed. Decades of research have proven it possible to engineer ASICs to forward IP packets at wire rate, even for the fastest wires. We think that much of that research plus some new techniques can be used to achieve wire rate forwarding of NDN’s longer and variable length data names.

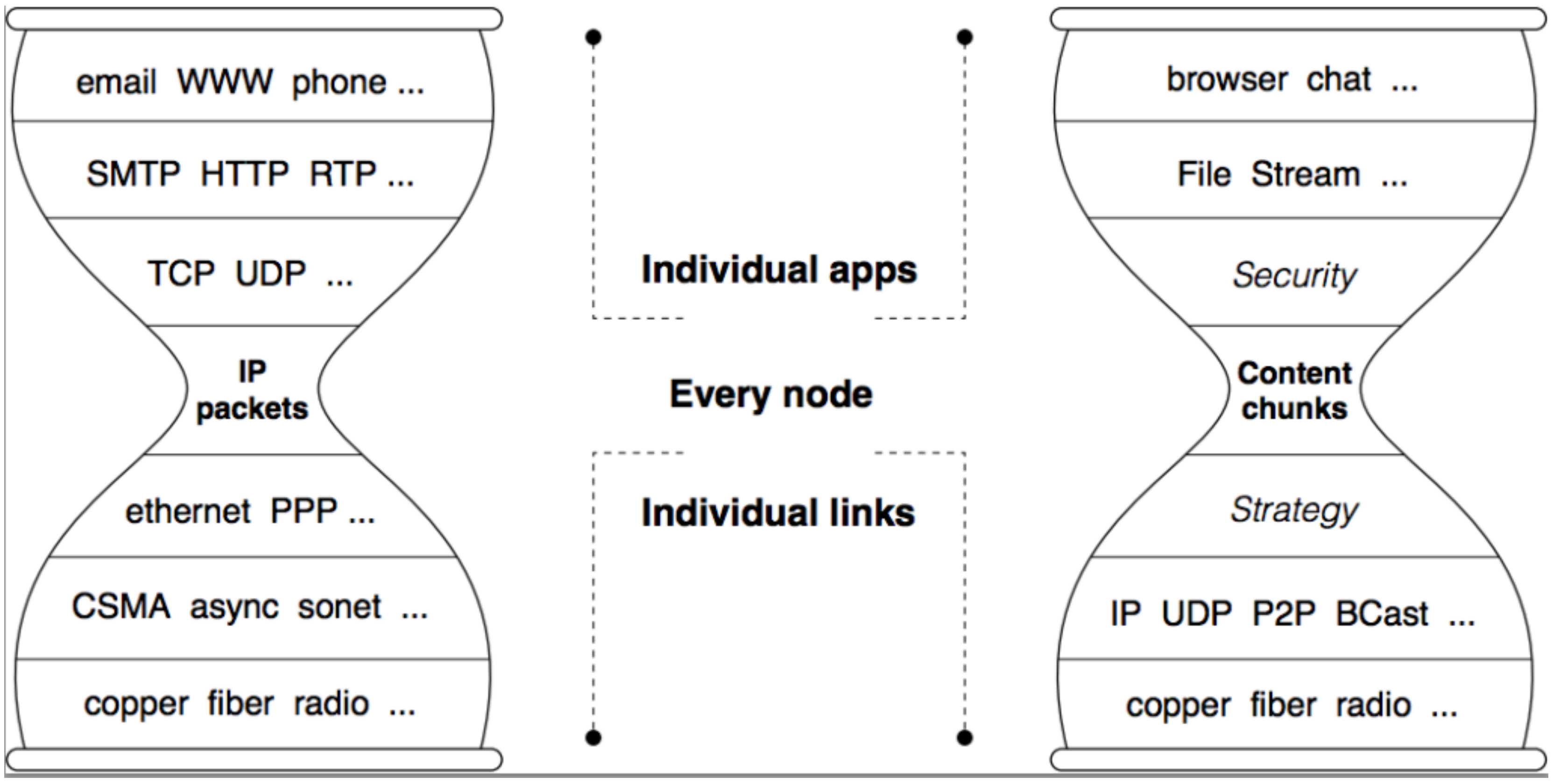
Our fundamental, information-theoretic framework for understanding communications is based on the capacity of a point-to-point *channel*. We believe this model can be extended to describe a communication system where memory has a larger and more central role, an intellectually challenging and novel direction.

Up to now communications security has been divorced from the data it tries to secure. Securing named data potentially allows the security to be much more user-centric, expressed in terms of the user's data model and application context. Finding effective, automatic and transparent mechanisms to implement and manage security of named data will be a new and more promising research trajectory than most IP security research has followed for the last two decades.

Although NDN represents a brand new architecture proposal, its hourglass shape makes it compatible with today's Internet and leads to a clear, simple evolutionary strategy. Like IP, NDN is a "universal overlay": NDN can run over anything, including IP, and anything can run over NDN, including IP. IP infrastructure services that have taken decades to evolve, such as DNS naming conventions and namespace administration or inter-domain routing policies and conventions, can be readily used by NDN. Indeed, because NDN's hierarchically structured names are semantically compatible with IP's hierarchically structured addresses, the core IP routing protocols, BGP, IS-IS and OSPF, can be used as-is to deploy NDN in parallel with and over IP. Thus NDN's advantages in content distribution, application-friendly communication, robust security, and mobility support can be realized incrementally and relatively painlessly.

Architectural Principles

We apply the following six architectural principles to guide our design of the NDN architecture. The first three are derived from Internet's successes and the last three from the lessons learned over the years.



(http://named-data.net/wp-content/uploads/hourglass.png)

Figure 1: Internet and NDN Hourglass Architectures

The *hourglass architecture* is what makes the original Internet design elegant and powerful. It centers on a*universal* network layer (IP) implementing the *minimal* functionality necessary for global interconnectivity. This so-called "thin waist" has been a key enabler of the Internet's explosive growth, by allowing lower and upper layer technologies to innovate without unnecessary constraints. NDN keeps the same hourglass-shaped architecture as shown in Figure 1.

The *end-to-end principle* (http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf) enables development of robust applications in the face of network failures. NDN retains and expands this design principle.

Routing and forwarding plane separation has proven necessary for Internet development. It allows the forwarding plane to function while the routing system continues to evolve over time. NDN sticks to the same principle to allow the deployment of NDN with the best available forwarding technology while we carry out new routing system research in parallel.

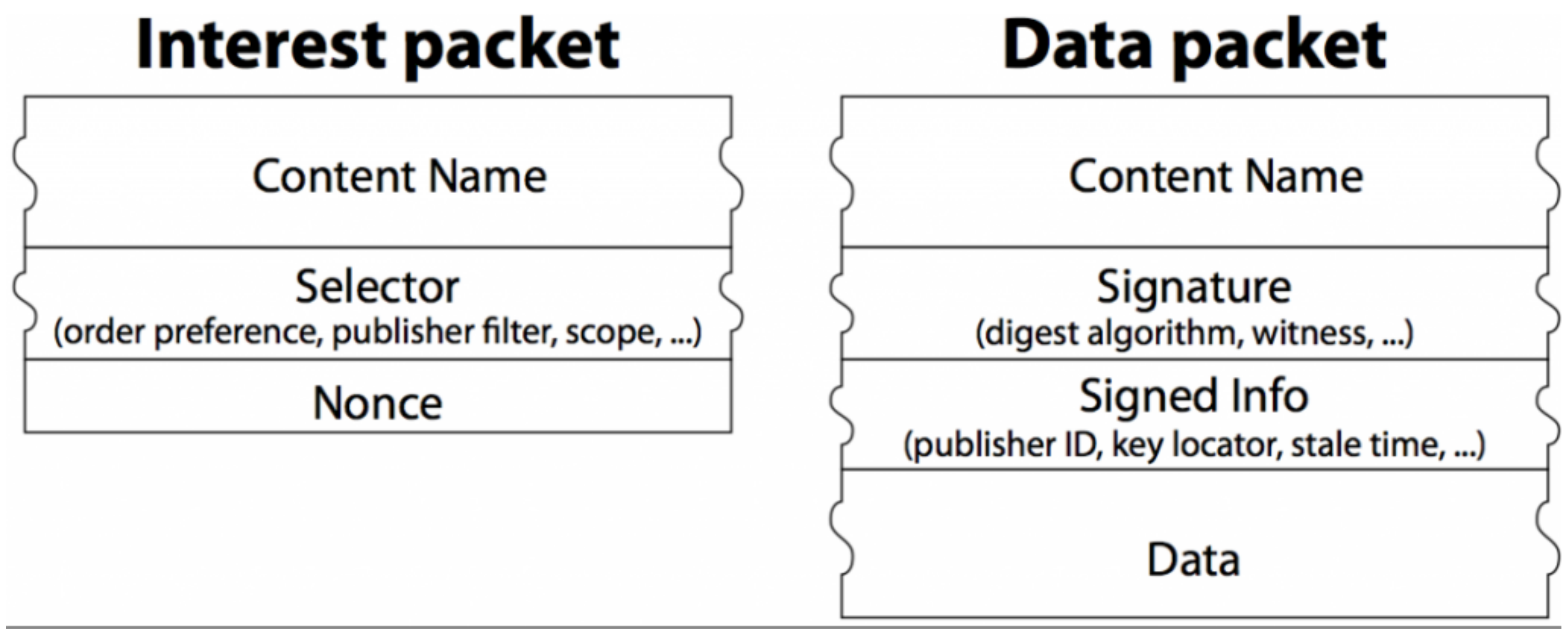
Security must be built into the architecture. Security in the current Internet architecture is an afterthought, not meeting the demands of today's increasingly hostile environment. NDN provides a basic security building block*right at the thin waist* by signing all named data.

Network traffic must be self-regulating. Flow-balanced data delivery is essential to stable network operation. Since IP performs open loop data delivery, transport protocols have been amended to provide unicast traffic balance. NDN designs flow-balance into the thin waist.

The *architecture should facilitate user choice and competition where possible.* Although not a relevant factor in the original Internet design, global deployment has taught us that "architecture is not neutral" (http://groups.csail.mit.edu/ana/Publications/PubPDFs/Tussle2002.pdf). NDN makes a conscious effort to empower end users and enable competition.

The NDN Architecture

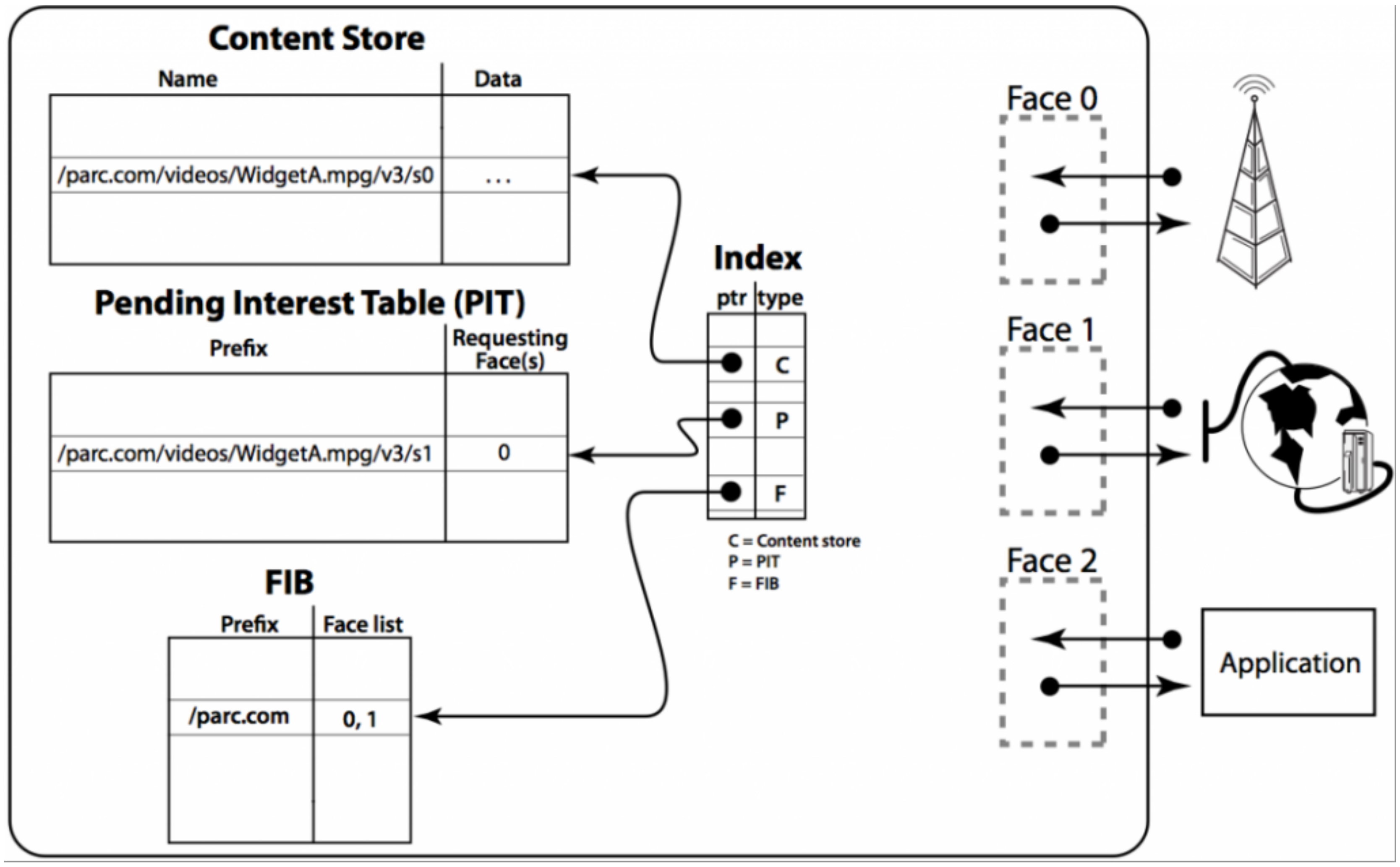
Similar to today's IP architecture, the thin waist is the centerpiece of the the NDN architecture. However because NDN's thin waist uses data names instead of IP addresses for delivery in order to offer a new set of *minimal* functionality, this seemingly simple change leads to significant differences between IP and NDN in their operations of data delivery. In this section, we first give a brief sketch of the basic concepts in NDN data delivery, then explain each element and its role in the overall architecture.


Interest packet
Content Name
Selector
(order preference, publisher filter, scope, ...)
Nonce
Data packet
Content Name
Signature
(digest algorithm, witness, ...)
Signed Info
(publisher ID, key locator, stale time, ...)
Data

(http://named-data.net/wp-content/uploads/ndn_packet.png)

NDN Packets

Figure 2: Packets In the NDN Architecture



(http://named-data.net/wp-content/uploads/node.png)
NDN Node

Figure 3: Internet and NDN Hourglass Architectures

Communication in NDN is driven by the receiving end, i.e., the data consumer. To receive data, a consumer sends out an ***Interest*** packet, which carries a name that identifies the desired data (see Figure 2). A router remembers the interface from which the request comes in, and then forwards the Interest packet by looking up the name in its ***Forwarding Information Base (FIB)***, which is populated by a name-based routing protocol. Once the Interest reaches a node that has the requested data, a ***Data*** packet is sent back, which carries both the name and the content of the data, together with a signature by the producer's key (Figure 2). This *Data*packet follows in reverse the path taken by the Interest to get back to the consumer. Note that neither Interest nor Data packets carry any host or interface addresses (such as IP addresses); Interest packets are *routed*towards data producers based on the names carried in the Interest packets, and *Data* packets are returned based on the state information set up by the Interests at each router hop (Figure 3).

The router stores in a ***Pending Interest Table (PIT)*** all the Interests waiting for returning Data packets. When multiple Interests for the same data are received from downstream, only the first one is sent upstream towards the data source. Each PIT entry contains the name of the Interest and a set of interfaces from which the Interests for the same name have been received. When a Data packet arrives, the router finds the matching PIT entry and forwards the data to all the interfaces listed in the PIT entry. The router then removes the corresponding PIT entry, and caches the Data in the ***Content Store***. Because an NDN Data packet is meaningful independent of where it comes from or where it may be forwarded to, the router can cache it to satisfy future requests. Because one Data satisfies one Interest across each hop, an NDN network achieves hop-by-hop flow balance.

Names

NDN design assumes hierarchically *structured* names, e.g., a video produced by PARC may have the name`/parc/videos/WidgetA.mpg`, where '/' indicates a boundary between name components (it is not part of the name). This hierarchical structure is useful for applications to represent relationships between pieces of data. For example, segment 3 of version 1 of the video might be named `/parc/videos/WidgetA.mpg/1/3`. The hierarchy also enables routing to scale. While it may be theoretically possible to route on flat names (see ROFL (http://dl.acm.org/citation.cfm?id=1159955)), it is the hierarchical structure of IP addresses that enables aggregation, which is essential in scaling today's routing system. Common structures necessary to allow programs to operate over NDN names can be achieved by*conventions* agreed between data producers and consumers, e.g., name conventions indicating versioning and segmentation.

Name conventions are specific to applications but *opaque* to the network, i.e., routers do not know the meaning of a name (although they see the boundaries between components in a name). This allows each application to choose the naming scheme that fits its needs and allows the naming schemes to evolve independently from the network.

To retrieve dynamically generated data, consumers must be able to *deterministically* construct the name for a desired piece of data without having previously seen the name or data. Either (1) a deterministic algorithm allows the producer and consumer to arrive at the same name based on data available to both, and/or (2) consumers can retrieve data based on partial names. For example, the consumer may request `/parc/videos/WidgetA.mpg` and get back a data packet named `/parc/videos/WidgetA.mpg/1/1`. The consumer can then specify later segments and request them, using a combination of information revealed by the first data packet and the naming convention agreed upon by the consumer and producer applications.

Not all the names need to be *globally unique*; only those names that are used to retrieve data globally require global uniqueness. Names intended for local communication may be heavily based on local context, and require only local routing (or local broadcast) to find corresponding data. In fact, individual data names can be meaningful in various specific scopes and contexts, ranging from "the light switch in this room" to "all country names in the world". How to develop efficient strategies to forward data within the intended scope is a brand new research area.

The name space management is not part of the NDN architecture, just as IP networks deliver packets using IP addresses but the IP address space management is not part of the IP architecture. However, data naming is the most important piece in the NDN design. Named data enables NDN to automatically support various functionality including content distribution (many users requesting the same data at different times), multicast (many users requesting the same data at the same time), mobility (users requesting data from different locations), and delay-tolerant networking (users retrieving data over intermittent connectivity). At the same time, we are still at an early stage of understanding how best applications should choose names that can facilitate *both* the application development and network delivery. We obtain that understanding through the development of and experimentation with a variety of pilot applications, so that we can extract a set of basic principles and guidelines for naming in NDN networks. We expect to convert these principles and guidelines into naming conventions that can be implemented in system libraries for consistent reuse to simplify future application development.

Fortunately not all naming questions need be answered immediately; the opaqueness of names to the network — and dependence on applications — means that design and development of the NDN architecture can, and must, proceed in parallel with research into name structure, name discovery and namespace navigation in the context of application development.

Data-Centric Security

In NDN, security is built into data itself (http://dl.acm.org/citation.cfm?id=2063204), rather than being a function of where, or how, it is obtained. Each piece of data is signed together with its name, securely binding them. Data signatures are mandatory — applications cannot "opt out" of security. The signature, coupled with data publisher information, enables determination of data *provenance*, allowing the consumer's trust in data to be decoupled from how (and from where) data is obtained. It also supports fine-grained trust, allowing consumers to reason about whether a public key owner is an acceptable publisher for a particular piece of data in a specific context.

However, to be practical, this fine-grained and data-centric security approach requires some innovation. Historically, security based on public key cryptography has been considered inefficient, unusable and difficult to deploy. Besides efficient digital signatures, NDN needs flexible and usable mechanisms to manage user trust. Preliminary investigations show that NDN offers a promising substrate for achieving these security goals. Since keys can be communicated as NDN data, key distribution is simplified. Secure binding of names to data provides a basis for a wide range of trust models, e.g., if a piece of data is a public key, a binding is effectively a public key certificate. Finally, NDN's end-to-end approach to security facilitates trust between publishers and consumers. This offers publishers, consumers and applications a great deal of flexibility in choosing or customizing their trust models.

NDN's data-centric security can be extended to content access control and infrastructure security. Applications can control access to data via encryption and distribute (data encryption) keys as encrypted NDN data, limiting the data security perimeter to the context of a single application. Requiring signatures on network routing and control messages (like any other NDN data) provides much-needed routing protocol security. We are working on efficient signatures, usable trust management, network security, content protection and privacy.

Routing and Forwarding

NDN routes and forwards packets on names, which eliminates four problems that addresses pose in the IP architecture: address space exhaustion, NAT traversal, mobility, and address management. There is no address exhaustion problem since the namespace is unbounded. There is no NAT traversal problem since a host does not need to expose its address in order to offer content. Mobility, which requires changing addresses in IP, no longer breaks communication since data names remain the same. Finally, address assignment and management is no longer required in local networks, which is especially empowering for embedded sensor networks.

The well understood and well tested core IP routing protocols, BGP, IS-IS and OSPF, can be used more or less as-is as routing protocols in NDN networks. Instead of announcing IP prefixes, an NDN router announces *name prefixes* that cover the data that the router is willing to serve. Routers simply treat names as a sequence of opaque components and do component-wise longest prefix match of the name in a packet against the FIB. However, an unbounded namespace raises the question of how to maintain control over the routing table sizes. Another important question is whether looking up variable-length, hierarchical names can be done at line rate.

NDN can greatly improve routing security. First, signing all data, including routing messages, prevents them from being spoofed or tampered with. Second, multipath routing, together with intelligent data plane as we describe next, can effectively mitigate prefix hijack because routers can detect the anomaly caused by prefix hijacking and retrieve the data through alternative paths. Third, the fact that NDN messages can talk only about data, and cannot be addressed to hosts, makes it difficult to send malicious packets to a particular target. To be effective, attacks against NDN must focus on denial of service, a problem we are actively working on.

Intelligent Data Plane

At NDN's data plane, the PIT records all pending Interests and their incoming interfaces. Each PIT entry indicates the expectation and permission for a Data packet, and is removed after matching Data is received or a timeout occurs. This per-packet state is a fundamental change from IP, where the data plan is stateless. The state information makes NDN's data plane adaptive in handling network failures and effective in utilizing network resources.

First, based on PIT and returning Data (or timeout), an NDN node can monitor the packet delivery performance of different interfaces and detect packet loss if any occurs, all at the time scale of a round-trip time. Second, based on PIT and a random nonce in the Interest packet, an Interest that comes back to the same node is easily identified and discarded. Thus Interest packets do not loop, nor do Data packets. With data-plane feedback and loop-freedom, individual NDN routers can make local decisions on forwarding Interests through multiple interfaces for service selection and load balancing, and can detect problems quickly and choose alternative paths to get around the failures. We are carrying out research efforts on *Forwarding Strategy*, the decision process that determines which Interest is forwarded to which particular interface as well as how many of the available communication interfaces to use to forward Interests, how many unsatisfied Interests should be allowed, the relative priority of different Interests, etc.

The PIT state at each router also serves several other important purposes. Since it includes the set of interfaces over which the Interests for the same data name have arrived, it naturally supports multicast delivery. Since each Interest retrieves one Data packet, a router can control the traffic load by controlling the number of pending Interests to achieve flow balance. The PIT state can also be used to effectively mitigate DDoS attacks. Because the number of PIT entries is an explicit indicator on the router load, an upper bound on this number sets the ceiling on the effect of a DDoS attack. PIT entry timeouts offer relatively cheap attack detection (see LADS (http://www.cs.berkeley.edu/~sekar/papers/lads.pdf)); and the arrival interface information in each PIT entry gives information to implement a push-back scheme (http://www.cs.columbia.edu/~smb/papers/pushback-impl.pdf).

Caching

Automatic in-network caching is enabled by naming data. Since each NDN Data packet is meaningful independent of where it comes from or where it may be forwarded to, a router can cache it in its content store to satisfy future requests. Upon receiving a new Interest, the router first checks the Content Store. If there is a data whose name falls under the Interest's name, the data will be sent back as a response. The Content Store, in its basic form, is just the buffer memory in today's router. Both IP routers and NDN routers buffer data packets. The difference is that IP routers cannot reuse the data after forwarding them, while NDN routers are able to reuse the data since they are identified by the data names. For static files, NDN achieves almost optimal data delivery. Even dynamic content can benefit from caching in the case of multicast (e.g., teleconferencing) or packet retransmission after a packet loss. Cache management and replacement is subject to ISP policies and is one of our research topics.

Caching named data may raise privacy concerns. Today's IP networks offer weak privacy protection. One can find out *what* is in an IP packet by inspecting the header or payload, and *who* requested the data by checking the destination address. NDN explicitly names the data, arguably making it easier for a network monitor to see what data is being requested. One may also be able to learn what data is requested through clever probing schemes to derive what is in the cache. However NDN removes entirely the information regarding who is requesting the data. Unless directly connected to the requesting host by a point-to-point link, a router will only know that someone has requested certain data, but will not know who originated the request. Thus the NDN architecture naturally offers privacy protection at a fundamentally different level than the current IP networks.

Transport

The NDN architecture does not have a separate transport layer. It moves the functions of today's transport protocols up into applications, their supporting libraries, and the strategy component in the forwarding plane. Multiplexing and demultiplexing among application processes is done directly using names at the NDN layer, and data integrity and reliability are directly handled by application processes where the appropriate reliability checking, data signing and trust decisions can be made.

An NDN network is designed to operate on top of unreliable packet delivery services, including the highly dynamic connectivity of mobile and ubiquitous computing. To provide reliable delivery, Interest packets that are not satisfied within some reasonable period of time must be retransmitted by the final consumer (the application that originated the initial Interest) if it still wants the data. Such functionality is common to many NDN applications, and is provided by NDN common libraries.

NDN routers manage traffic load through through managing the Interest forwarding rate on a hop-by-hop basis; when a router is overloaded by incoming data traffic from any specific neighbor, it simply slows down or stop sending Interest packets to that neighbor. This also means that NDN eliminates the dependency on end hosts to perform congestion control. Once congestion occurs, data retransmission is aided by caching since the retransmitted Interest will meet the Data right above the link the packet was lost, not the original sender. Thus NDN avoids congestion collapse that can occur in today's Internet when a packet is lost at the last hop and bandwidth is mostly consumed by repeated retransmissions from the original source host.

Traditional transport services provide point-to-point data delivery and most of today's distributed applications, including peer-to-peer applications, heavily rely on centralized servers. To aid the development of robust and efficient distributed applications, we envision a fundamentally new building block for distributed systems that we are calling *Sync*. Built on top of NDN's basic Interest-Data communication model, Sync utilizes naming conventions to enable multiple parties to synchronize their datasets by exchanging data digests, so that individual parties can discover and retrieve new and missing data in a most efficient and robust manner. We expect that Sync's role in the NDN architecture will evolve to one similar to TCP's in the IP architecture.

This research is partially supported by NSF (Award CNS-1040868 (http://www.nsf.gov/awardsearch/showAward?AWD_ID=1040868))