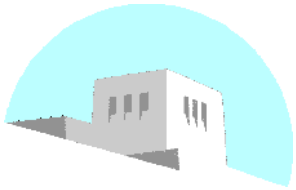
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING



SCHOOL OF ENGINEERING
UNIVERSITY OF NEW MEXICO

Towards a Taxonomy of Inter-network Architectures

Joud Khoury Chaouki Abdallah [1][2]

UNM Technical Report: EECE-TR-08-008

Report Date: June, 2008

[1]Joud Khoury and Chaouki Abdallah are with the Department of Electrical and Computer Engineering, of the University of New Mexico, Albuquerque NM 87131, {*jkhoury, cabdallah*}*@ece.unm.edu*.

[2]The work presented in this report is partially funded by the National Science Foundation NSF under the Future Internet Design (FIND) Grant CNS-0626380.

Abstract

Over the past decade, research on network architecture design has intensified. However, contributions to the field have mainly been idiosyncratic and architectural descriptions remain idiomatic. This state of affairs has led to the emergence of a large body of network architecture proposals with no clear indication of their compatibility points, their cross similarities, and their differences. Thus, a taxonomy of network architectures that provides a framework for better understanding, organizing, and thinking about the complex architecture design space would be a timely contribution. This paper presents a first step in that direction by attempting a classification based on the architecture's information model. The taxonomy is applied to a special network architecture highlighting its descriptive and classification powers.

1 Introduction

Computer networks are continuously evolving into smarter and more complex systems that can better accommodate the nomadic, ubiquitous, and pervasive computing lifestyles of the diverse users. As a result, new internetwork architectures are continuously being proposed, rendering the network architecture space larger and more diverse. Such architectures may be broadly categorized into either incremental or radical efforts. Incremental architectures, such as [38, 44, 36], generally aim at addressing particular limitations of the current Internet architecture through patching, while radical architectures, such as those supported by NewArch [3] and FIND [4] initiatives, tend to adopt a clean-slate approach to designing a "better" Internet, without being necessarily restricted by the current Internet model.

Unfortunately however, the majority of the architectural work remains idiosyncratic and descriptions of network architectures are mostly idiomatic. This current state of affairs is expected to worsen as we start designing and deploying a future Internet, an effort already initiated by NSF's FIND [4], and GENI [2] initiatives. In fact, after surveying the literature, it became obvious that the majority of the recent architectural work is either aimed at exploring novel usage models that adhere to a class of applications, or at directly addressing a set of limitations of the current Internet [1]. Ostensibly, there seems to be a growing consensus in the community about the need for designing a smarter network that is more than just a transparent "bit-plumbing" medium. While such evolution into a smarter and more complex Internet is bringing new potentials and service models, the community generally lacks a unified framework or a taxonomy for thinking about such models and their design implications. In this sense, we believe that a network architecture taxonomy is a timely contribution that can potentially frame the architectural work, clarifying the problem and the solution spaces. Additionally, such a taxonomy provides a unified framework for networking researchers: (1) to better reason about their work at the architectural level, (2) to clearly compare the different proposals and better understand their similarities and differences, and (3) to explore new dimensions for contributing to the field.

This paper presents an attempt towards a taxonomy of inter-networking architectures. Our taxonomy defines a network architecture based on the information model. The latter operates on top of the substrate structure and characterizes the underlying addressing structure, the data objects and the functionality attached to them, and the relative control structure. Prior to defining the taxonomy, we present a high-level classification of some of the existing literature based on the broad service models provided by the network. Such classification offers useful insights regarding the architectural landscape, which are leveraged by the taxonomy. It is worthwhile mentioning that several classes of our taxonomy may be further elaborated. Additionally, we fully expect that several new classes and properties will be added by other researchers. We would like to note that the current taxonomy is not intended for evaluating the performance of architectures and for determining whether one architecture is better than another. Any such effort would require a thorough understanding of the design space (design parameters, relationships, cost structures, etc.), an effort that we believe is more likely to succeed at narrower scopes than the one at hand. In addition, we would like to mention that the literature is replete with network proposals that correspond to the different taxons discussed throughout the paper. The examples we provide throughout are solely meant to help the reader assimilate our ideas rather than provide an exhaustive list of the related work.

As we started studying the taxonomy problem, it seemed that the body of network architecture work is difficult to classify due to the independent nature of the many contributions to the field. However, we have noticed that modern networks are becoming increasingly intelligent, and the intelligence is being manifested by introducing more processing and storage elements, and by providing the users with richer instruction sets instead of the simple static IP packet. Interestingly, such evolving network architectures resemble the computer architecture field, in the sense that a network architecture is currently being designed to provide a general purpose computing platform to its diverse users. Consequently, it is our belief that the modern network architecture and the computer architecture converge conceptually at the architectural level, despite the fact that they significantly diverge otherwise, primarily due to the distributed and large-scale nature of network architectures. We shall leverage this idea to directly apply some useful taxonomical notions from the computer architecture field to our work, particularly from [26, 23].

[1] Those limitations are mainly the lack of information, security, management, troubleshooting, mobility and QoS support, and the economic conflicts as acknowledged by the community [8, 21].

The remainder of the paper is structured as follows: Section 2 explains our classification approach, and the intuition behind it. We start by classifying some modern network architectures, from a high-level perspective, based on their supported service model. Building on the high-level perspective, a taxonomy based on the information model is then discussed in section 3. We demonstrate the descriptive power of the taxonomy by applying it to the Data-Oriented Networks Architecture (DONA) [30]. Related work is then presented in section 5 before concluding with a discussion of the value and limitations of our work in section 6.

2 Classifying Network Architectures

Before discussing our classification approach, we recall some general definitions. A *computer network* is an interconnection of computers over which information [2] flows. The *network architecture* is the conceptual design and the fundamental operation structure of a computer network. Based on these definitions, one may clearly recognize the obvious defining structures of a computer network: computers and inter-connections, communication, and information structures.

2.1 Classification Approach

How to approach the classification problem given the complexity of the design space? In other words, what should the defining element(s) of our classification model be? We start by recognizing that every design is intended to support a set of goals, which generally encapsulate the pressing needs/requirements of users [3]. Generally speaking, the design process then involves converging on a set of defining structures, and proceeding to optimize those. The outcome is an architectural design that is comprised of the following abstraction levels: 1) the *outer-architecture* represents what the network user can see. This is analogous to the network service interface or *Instruction Set Architecture (ISA)* which defines the addressing modes, the data object semantics, and the available operations; and 2) the *inner-architecture* represents the internal operation structure of the network including the low level substrate structure and the functional aspects to support the outer-architecture.

We believe that both abstraction levels provide useful and complementary insights regarding the architectural landscape. Hence, to answer our question of what the defining element(s) of the taxonomy should be, we found it useful to start by devising a high-level classification of some of the existing literature based on their supported service model (or the types of services the network provides to its users). This view has helped us in understanding the underlying goals behind an architectural design, and has additionally highlighted the information model as the main defining element of our taxonomy. The high-level classification, which we refer to as the service-model perspective, is briefly discussed next.

2.2 Service model perspective

Classifying architectures from this perspective is motivated by several factors. First, the service model approach implicitly accounts for the needs of the users relative to a network, which is the ultimate goal of any network design. For example, the Internet's simple "best-effort delivery" service model came about to satisfy a set of goals, as explained in [16], primarily allowing multiplexed utilization of resources (which led to packet switching, domain, gateways), survivability (which led to end-to-end state), etc. Second, most network architectures tend to be naturally categorized and described relative to their service models. For example, we find in the literature the "data-oriented" network architecture [30], the "delay-tolerant" architecture [1], the "differentiated services (diffServ)" architecture [10], and so on. Finally, such a classification could enable future reasoning about - and

[2] Information, content, and data are used interchangeably within this paper, unless otherwise specified, to represent data abstractions recognized by the network.

[3] Within the discussion, a *user* is the general term used to abstract any entity that utilizes the network services.

evaluation of - the degree to which a particular architecture satisfies the service requirements of the users. One such evaluation methodology based on utility was proposed in [41] [4].

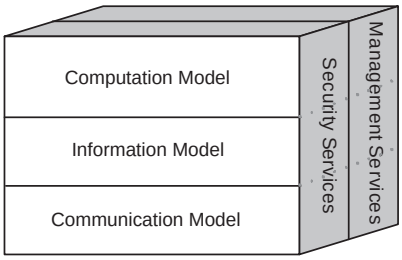


Figure 1: Generalized service model view

The generalized service-model perspective is depicted in Figure 1. The communication, information, and computation [5] models represent the building blocks that collectively define, together with the security and management services, the general service model of any network architecture. By building blocks we mean that every architecture must provide these three models, whether explicitly or implicitly [6]. On the other hand, security and management services are not building blocks (since one can easily come up with architectures that do not provide any security or management services), and they operate across the communication, information and computation models.

- *Communication model*: This service model represents the communication and control services offered by the network. For instance, delivery services whether “best-effort", QoS-aware ([10]), aware of disruption ([1]), and/or geographic location ([25]) all belong to this model. Communication paradigms whether connection-oriented (e.g. ATM) or connectionless (e.g. SMDS, X.25) are classified under this model as well.

- *Information model*: This model deals with the information services that the network provides to its users. The networking community currently recognizes the need for network built-in information services (naming, searching, security, and analysis services) to support a multitude of applications and their requirements (archiving, distribution, etc.) [5].

- *Computation or Programmability model*: This model represents the level of programmability support within the network. The programmability services might potentially span all the other service models, allowing for example the programmability of the communication model and/or the information model etc. Programmable networks [14, 43], for example, provide an explicit computation model.

The security and management services provided by the network are generally, but not necessarily, offered in-band with the rest of the service models. For example, secure communication services include secure end-to-end tunneling and transport (IPSec or SSL), secure identity (HIP [35]) etc. In general, the approach to security and management in traditional newtorks has been incremental, and not accounted for by design.

In fact, it is possible (and maybe convenient) to fully classify the literature based on the service model view depicted in Figure 1 if each of the constituent service models is further divided into its defining elements. However, and as we are mainly interested in the inner-architecture, the major focus of this section is to illustrate some of the prominent architectural work that represents critical points of the aforementioned service spectrum. Additionally, we believe that the independent contributions to the field are converging, and this section aims to highlight such phenomenon by means of a survey. Table I presents, in matrix form, such a survey of the literature, limited to general inter-network architectures. Hence, we do not consider overlays, scoped architectural

[4] In [41], Shenker defines utility as the degree to which a network service model matches the needs of the network users i.e. how good an architecture is, is measured by the happiness of its users.

[5] We abuse terminology referring to the terms computation and programmability interchangeably hereafter.

[6] For example, the Internet provides an information model implicitly (the datagram and information transparency) but not explicitly.

		Year	Comm	Info	Comp	Description
Independent Proposals	**Internet** [16]	1970s	x			providing best-effort delivery of datagrams among globally identified endpoints
	Active Nets [43]	late 90's	x		x	provide a framework for dynamic creation and deployment of network services at runtime
	TRIAD [24]	2000-1	x	x		exposing a "content-layer" that provides transparent access and distribution of named content
	Plutarch [19]	2003	x			provide a communication model that inherently allows inter-operation of semantically disparate domains without mandating uniformity across them
	FARA [17]	2003	x			provide an abstract network model that builds on the Internet's "best effort" service model adding clean separation of endpoint names from network addresses
	TurfNet [40]	2004	x			similar to Plutarch service model, but with global naming
	DONA [30]	2007	x	x		providing data-access (locating and retrieving data) independent of location as well as providing data distribution from multiple locations
FIND [4]	**Postcards** [48]	2006	x	x		providing reliable delivery (push/pull) of content (large data units or files) to mobile/stationary endpoints using in-network storage/caching
	USwarm [45]	2006	x	x		providing multipoint-to-point bulk data transfer/distribution among hosts (endpoints+intermediaries) with in-network storage/caching
	ITDS [46]	2006	x	x	x	providing information transfer in response to user (endpoint) specified service expressions through in-network processing/data handling
	WiKI [11]	2006	x	x	x	providing a network query interface to users for expressing intent and implementing operations through a declarative framework for managing in-network information and state (router and host state, and data streams)
	TNA [28]	2006	x	x		provides a transient network substrate that enables identification and communication among entities based on global, and persistent (location-independent) identifiers
	PostModern [9]	2006	x		x	providing a tussle-resistant communication service, delivery of functional datagrams, that equips providers with usage control over their networks through policy enforcement, and users with policy-aware control over their traffic forwarding

Table I: Matrix view classification of inter-network architectures based on their explicit service model classes.

work (such as naming, or routing architectures) and we do not provide an exhaustive list of inter-network technologies which is not the goal of this section [7]. The work is divided into two parts. The first part overviews some of the independent contributions to the field, while the second part is solely concerned with the FIND [4] work illustrating the community's view of what the future Internet should look like. Note that Table I marks the service models only as those are made **explicit** in the architectural description, and consequently it does not contradict our previous claim of the communication, information, and computation models being building blocks. Several insights may be gathered by observing the matrix. First, there seems to be a growing consensus about the need for expanding the network's service model beyond the communication space, especially as researchers start thinking of designing a future Internet. Additionally, and most importantly, one can clearly notice the emphasis on information services, which is intuitive given the prevailing information-centric usage models with the current Internet. *Hence, while the communication structure is necessary for defining and representing a modern network architecture, it is in general insuffcient*. Information and computation structures are other building blocks that need to be properly understood within modern networks. We shall leverage this observation in the next section to present a taxonomy that revolves around the architecture's information model.

[7]The majority of inter-network technologies (ATM, X.25, XNS, DECnet etc.) would be classified in our matrix as communication-oriented. We only reference those technologies when they directly serve the goals of our taxonomy. For a comprehensive list of the inter-network technologies, we refer the reader to [15].

3 Taxonomy

Our taxonomy is based on the network's information model, as it aims to clarify the following questions:

- What are the types of data objects recognized by the network?; and
- How does the network operate on those objects? In other words, how is the network capable of manipulating the objects?

Towards this end, the taxonomy defines architectures starting with the underlying substrate structure (the topology, the functional units, and their interconnection structure) over which the information model operates, and ending with the information model itself (addressing structure, types of data objects, and control structure).

3.1 Substrate Structure

The network substrate is comprised of the underlying physical network elements over which the information model is defined. The substrate structure, hence, describes the network topology, the functional units, and their interconnection structure.

Topology

Assumptions regarding the inter-network topology are crucial to our analysis. We assume the inter-network is composed of *zone*s. A zone forms an autonomous part of the inter-network and represents a logical region with explicit boundaries. We intentionally define the notion of a network zone to be abstract enough to encapsulate the various definitions proposed in the literature, including the Internet Autonomous Systems (AS), Contexts in Plutarch [19], Turfs in Turfnet [40], and so on. The zone has an explicit "boundary", a logical construct that can take various forms, such as administrative, physical, protocol, or even social boundaries. Within the remainder of the discussion, the notions of "global" and "local" are to be interpreted relative to the zone. For example, a global function (examples of functions are addressing, naming, and forwarding) is one that operates across zones, whereas a local function is to be interpreted as zone-local.

Of particular interest to our taxonomy are the following topological properties:

- *Structure*: is an important property that can take any of the values: *hierarchical*, *flat*, or *special* (e.g. ring) topology. An inter-network that is composed of hierarchical zones will topologically include a root set of zones, generally referred to as "Tier-1". A flat topology on the other hand does not necessitate a topological root.
- *Composition*: The topological structure depends on how the zones are composed. Composition can take three forms as follows: 1) *controlled-overlap* means that part of the topology is shared by multiple zones, 2) *integration* is when one zone subsumes another resulting in an integrated data/control plane for the composed zone (sometimes referred to as horizontal composition), and 3) *direct peering* is when zones, generally heterogeneous, directly connect through dedicated elements (sometimes referred to as vertical composition). Note that from a physical viewpoint, direct peering encapsulates the Internet AS relationships, whether customer-provider, or peering.

Components and Interconnections

We isolate the following components types or functional units:

- Storage Elements (SEs) which may be of two types:

  - *Memory Element*s (ME) are abstract elements that store information within the network, such as content servers/providers; and
  - *Cache Element*s (CE) are memory elements that provide faster access to their information, either by being physically closer to the user and/or because they are connected to the user by a higher bandwidth link than the ME. Examples include proxies or caches used in content distribution networks (CDNs).

- Processing Elements (PEs) which perform information processing and may be further divided into:
  - *Data Processing Element* (DPE); and
  - *Instruction Processing Element* (IPE).
    More details on instructions and data are presented later in the information model. However, for now, one may envision an IPE instance to be a router, or a proxy that operates on packets [8], while a DPE instance might be a content transcoding element inside the network.

- Switching Elements (SWEs) are abstract elements that switch information between SEs and PEs.

Having introduced the abstract component types, we proceed to describe their properties that are of interest to this taxonomy, as follows:

1. The *Dispersal* property/factor is specified for each of the above component types. It describes the required physical distribution/placement of an element type relative to the topology, with a number n to mean one element (or a constant set of elements) per n zones. Values for n may be: 1 (to mean an element exists for each zone), k (to mean an element exists for a group of k zones, such as a Tier-1 ISP provider hierarchy or the set of edge domains on the Internet), and Z (to mean an element exists for all the zones, such as in the case of a centralized global service), where Z is assumed to be the total number of zones in the topology.

2. The *Interconnection* property describes the logical interconnection structure among the component types. Two combinations of element interconnections are of interest to us, mainly those specifying the $PE-PE$, and the $PE-SE$ element interconnections. The different types of interconnections are depicted in Figure 2, and those may be: 1) *dedicated* to mean that the i^{th} component of the first type is connected to the i^{th} component of the second type; 2) *meshed* to mean that every component of the first type is directly connected to every component of the second type; and 3) *switched* to mean that a switching element connects components of the first type to those of the second type.

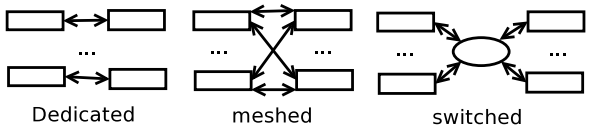


Figure 2: Interconnection types; A square represents an abstract element (SE or PE), while an ellipse represents a switching element (SWE).

Consequently, the tuple $(Dispersal, Interconnection)$ fully describes the component interconnection structure. Additionally, it directly relates to scalability by exposing the bottleneck infrastructure elements. We briefly present some examples related to the current Internet substrate structure to better illustrate the aforementioned properties. Internet routers are IPEs (that process implicit forward instruction) with dispersal factor $n = 1$, and for which their IPE-IPE interconnections is meshed. DNS infrastructure elements, and particularly domain DNS servers are IPEs (that process resolve instructions) with $n = 1$ and may simultaneously be CEs (that cache query results) and MEs (that serve the domain's zone files) with $n = 1$. The DNS root servers, however, are MEs (root database) with $n = Z$ [9]. Additionally, the IPE-ME interconnection is generally switched since resolutions have to pass by the root servers that act as the switch between the IPEs and the MEs.

[8] A packet is a form of a static instruction.
[9] Assuming not replicated.

3.2 Information Model

The information model is defined based on three classes of data objects that encapsulate the information abstractions recognized by the network. At the core of the information model is the notion of data objects that are bound to and accessed from network 'locations' relative to some addressing structure. Consequently, before delving into the details of the data objects, we discuss the first defining element of the network information model, namely *the addressing structure*.

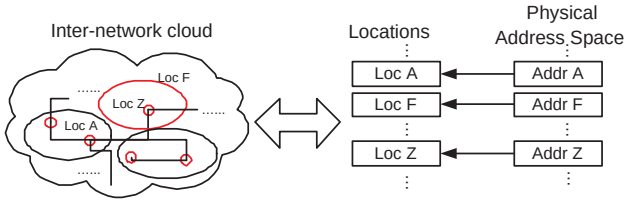


Figure 3: Abstracting network locations (red circles) and visualizing a physical address space.

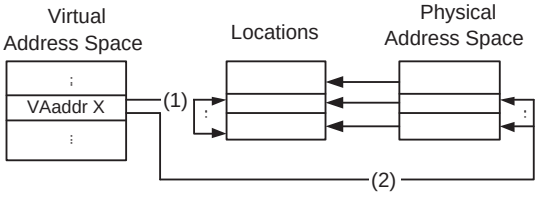


Figure 4: Virtual address space is either integrated (1), or disconnected (2).

Addressing Structure

The information model starts by assuming the existence of *locations* within the network, where the term 'location' may have topological or geographical connotation. When the locations are addressable, we obtain the notion of the network *physical address* belonging to the *physical address space*. The latter consists of all the possible addresses of all the addressable locations on the network and is denoted by $\mathcal{S}_p$. Consequently, the physical address is defined as a location identifier. Figure 3 depicts our visualization of the locations and of the address space constructs. Examples of Addr A in Figure 3 may be an IP address, a path, a set of coordinates, etc. (as long as there exists an underlying control that can link the physical address to the network location it points to).
On the other hand, when the objects on the network are being addressed, we obtain the notion of the *virtual address* belonging to a *virtual address space*, $\mathcal{S}_v$. A virtual address is a location independent identifier which is ultimately mapped to some location(s). Some examples of systems that instantiate virtual spaces are naming/directory systems, metadata registries, and trackers [10]. According to our definitions, there is a conceptual difference between the physical address and the virtual address in terms of what is being identified. The latter generally identifies some high level information abstraction (such as a host, or a content object) in contrast to identifying a location with the former. Data objects, which we shall characterize shortly, are always bound to the locations, and hence every access to an object on the network will require an address (physical or virtual) to succeed. We would like to note that our definitions of location, objects, physical and virtual addresses conform to Saltzer's RFC [39] but are less restrictive in the sense that we are solely interested in highlighting the boundary relationship between location and object abstractions. To eliminate confusion, in [39] 'networked objects' is a general term used to refer to services, nodes, and locations (or attachment points), while in our definition 'objects' are anything bound to locations (which includes nodes, services, or information abstractions in general).

[10] Other terms used in the literature to refer to the virtual address are the name or the label.

Throughout the rest of the discussion, an *address* is to be interpreted as either a physical or a virtual address unless otherwise specified.

As far as the taxonomy is concerned, the following set of properties characterizes the addressing structure. The first property, *address spaces*, describes whether the addressing structure explicitly defines and makes available as part of the *ISA* a physical address space ($\mathcal{S}_p$), a virtual address space ($\mathcal{S}_v$), or both.

- **Physical**: Solely providing a $\mathcal{S}_p$ implies that objects are only addressable (and generally accessible) by location. In addition to necessitating prior knowledge of location, this model falls short of supporting information binding volatility (such as in the case of mobility, re-homing, and disconnections). The IP addressing architecture [16] is one example in which only the IP physical space is made available as part of the *ISA* addressing structure.
- **Virtual**: On the other hand, solely providing a virtual space implies that objects are addressable independent of location. Since no physical space is provided, there is no embedded notion of location on the network from the user's perspective. In this sense, the virtual address space is directly *integrated* with locations i.e. an access to a virtual address will automatically result in accessing the location(s) to which the virtual address points. Figure 4 (1) illustrates this addressing style. Some example architectures that support this style are [13] and [28].
- **Both**: When both spaces are provided, it is necessary to characterize their relationship defined with *space-correlation*, as follows:
  - **Independent**: $\mathcal{S}_v \cap \mathcal{S}_p = \emptyset$
    This is the general case of current addressing architectures in which the spaces are semantically and syntactically independent, and are only related through the mapping/search function. Examples of $\mathcal{S}_v$ could be a space of flat hashes (e.g. DHT approaches) or human-readable strings (e.g. DNS) which is independent of an underlying physical address space (e.g. IP, or topology labels in labeled compact routing [22]).
  - **Correlated-partitioned**: $\mathcal{S}_v \subset \mathcal{S}_p$
    In this model (and the following one), the spaces are consolidated (generally syntactically) and the semantic distinction between the spaces is made statically (i.e. known a priori) or dynamically (i.e. at runtime). The model has the feature that the mapping function from physical addresses to locations (otherwise referred to as routing) is inherently aware of $\mathcal{S}_v$ and could be reused for virtual address search.
  - **Correlated-embedded**: $\mathcal{S}_p \subset \mathcal{S}_v$
    If we are allowed to think of the IPv6 space as virtual address space, then an example of this model would be the embedding of the IPv4 space in the IPv6 space.
  - **Partial-overlap**: $\mathcal{S}_p \cap \mathcal{S}_v \neq \{\emptyset, \mathcal{S}_p, \mathcal{S}_v\}$
    We are not aware of any addressing architecture that supports this model.
  - **Equivalent**: $\mathcal{S}_v = \mathcal{S}_p$
    In this model, from the user's perspective, locations and objects are indistinguishably addressed. An example of such model could be a DHT that is restricted to being matched to the underlying physical topology i.e. in which all pointers (such as successor or finger pointers in Chord) must be actual graph links and hence nodes can only point to direct neighbors. In this setup, nodes could be though of as locations in the graph and objects that are published on nodes take addresses from the same space. Again, we are not aware of any such proposals or technologies.

Second, for each of the physical and the virtual address spaces, two structural properties are defined:

- *Space structure*: may be **hierarchical**, **flat**, or **special**; and

- *Addressing scope*: defines the scope over which the address is valid. Values for scope are based on the topology structure defined in the previous section, and those include: **local** (per zone), **global** (to all zones), and **partial** (to a set of zones).

To gain a better insight into the taxonomy's descriptive powers, consider the following examples of addressing structures described with the properties just introduced:
1) *Physical space* is $(hierarchical, global)$: The Internet and NIRA [47] are two architectures that explicitly rely on a global, strictly hierarchical addressing scheme. Some of the advantages of this scheme are scalable routing and small routing table sizes [11]; 2) *Virtual space* is $(flat, global)$: ROFL [13] and SFS [7] are architectures that utilize this flat, DHT-style addressing scheme. Some of its advantages include semantic-free, flat, and location independent-addressing. On the other hand, some of the disadvantages include global maintenance overhead, consistency issues, and scalability concerns; and 3) Either space is $(*, local)$ [12]: This is the case for example when each network has its own private address space. Plutarch [19], Turfnet [40] are some examples. Some of the advantages of such mode are provider-independent addressing and easier re/multi-homing. However, some disadvantages include extensive translation, complex routing, and larger global routing tables.

Having discussed the addressing structure, we proceed to identify the different classes of network data objects that comprise the second defining element of the information model.

Data Objects

The data objects are characterized by the ordered tuple (C, S, F), where C denotes the object class, S denotes the scope or context within which the object is meaningful, and F denotes the set of functions applicable to the object. We isolate three classes C of data objects [13]: 1) *primitive objects*, 2) *group objects*, and 3) *complex objects*, and their respective functions as follows:

Primitive Data Objects - are further categorized as either *carrier objects* or *consumable objects*. The former set represents the information carriers that are stored or processed within the network but are generally not consumed i.e. neither bound to nor accessed from locations. We have identified the following set of carriers: 1) physical address; 2) virtual address; 3) instruction, represents a functional expression to be executed by the network. Instructions may range in their functional expressiveness. For example, the IP packet is an instance of an instruction data object that is inherently static i.e. the packet implicitly instructs the network to deliver a payload from source to destination. On the other hand, [46] recognizes a more expressive instruction set for dynamic service composition; 4) Data Unit, is the unit of communication and could include control and data; 5) stream, is an aggregation of Data Units; and 6) status block, may be of different types. A status block encapsulates the internal network state as well the status of operations performed within the network.

As to the consumable objects, those represent data objects that are explicitly bound to the network locations and consumed from their locations. The following two styles of consumable objects are identified:

- Raw Information Bit Stream (RIBS): represents an untyped consumable data object that appears to the network as a bit stream. All RIBS objects must be self-descriptive i.e. knowledge about the RIBS data (e.g. typing, and interpretation) is encapsulated within the data itself. The Internet, for example, solely supports RIBS objects and is hence transparent of information types (i.e. all typing intelligence is end-to-end);

- Typed Abstract Content Object (TACO): represents any typed consumable data. Interpretation of TACOs is generally part of the network's control structure. Several examples of possible typed content objects may be recognized from today's applications, including: static and dynamic content (such as files and web

[11] Scalability depends on efficient address aggregation, however. The de-aggregation practice on the current Internet's routing system has driven the latter to be unscalable [34].

[12] The symbol '*' is the wildcard character.

[13] Recall that we are solely concerned with the internal network information model, and hence the end-to-end data abstractions which are transparent to the network are irrelevant to this taxonomy.

pages), continuous content (such as multi-media or live sensor feeds), interactive content (such as the case with online gaming), and metadata objects.

The object scope S defines the scope within which the object is valid, taking values: *local*, *global*, or *partial* relative to the substrate structure.

Several classes of functions F are applicable to primitive data objects. While we overview these classes, we simply focus on characterizing the binding, access, and transfer functions that the network makes available for manipulating the consumable data objects.

- **Binding Functions**: In its simplest form, binding is the process of assigning a data object to some location on the network. Assume X and Y denote network addresses, we isolate the following forms of binding:
  - **Direct-value binding** has the form $X = value$. Binding a data object to an IP address, or a host join in [13] are some examples;
  - **Processed value binding** has the form $X = f(.)$. Some processing is performed before assigning the object. For example, $f(.)$ might be a query whose result is assigned to an address;
  - **Multiple-value binding** [14] has the form $X = Y$. The data object in Y is replicated to X. In general, this form of binding requires some form of COPY/MOVE instruction as part of the architecture's *ISA*. Data replication as in the case of CDNs is an instance of such binding style;
  - **Shared-value binding** [15] has the form $X = \&Y$ to mean that X points to the same data object as Y. Multihoming is an instance of this style;

  While instances of such binding styles are present in the literature and were noted above, the majority of the styles are still not explicitly supported by the current architectures' instruction sets.

- **Access Functions**: Accessing information on the network may be characterized by the following properties: 1) *access type* specifies whether primitive, group and/or complex object access is supported; 2) *access paradigm* specifies **synchronous** and/or **asynchronous** access. For example, publish-subscribe architectures provide an asynchronous access paradigm; 3) *access mode* dictates whether **read**, **write**, and/or **read-modify-write** are supported; ; and finally 4) *addressing mode* which is characterized with:
  - **Direct/absolute addressing** (physical and/or virtual): The absolute address of the object to be accessed is known, whether physical (e.g. IP addressing) or virtual (e.g. [40]).
  - **Indirect addressing (physical and/or virtual)** represents the indirection style addressing, where the absolute address of the object is unknown, but an alternative address (pointer) is used for indirect access. Physical-to-physical indirection (e.g. [27]), and virtual-to-physical indirection (e.g. [42]) are some flavors of indirect addressing.
  - **Associative addressing** (physical and/or virtual) is analogous to intentional addressing, in which the sought object's address is unknown, but some of its attributes are known and are employed for addressing. Distributed searching, whether in the physical space (e.g. [19]) or the virtual space (e.g. [45]), is generally utilized to locate the objects of interest.
  - **Group addressing** (physical and/or virtual) involves addressing a group of locations (e.g. geocast and multicast addressing), or a group of objects. Addressing a group of objects is equivalent to addressing a group object type (to be discussed shortly).
  - **K-preference addressing** (physical and/or virtual): is similar to group addressing except that k elements of the group are addressed instead of the whole group. Anycast addressing, for example, is a special case of this mode in which $k = 1$ and the preference is 'any' (e.g. IP anycast, [30]).

[14] This is similar to assignment by value.

[15] This is similar to assignment by reference.

- **Transfer/Delivery Functions**: Two properties of the delivery function are essential to this taxonomy, while several others may be deduced from other properties of the information model (such as the addressing modes). First, the *information recognition* describes whether the delivery is cognizant of information types. For example, delivery of continuous content objects (such as multi-media stream) requires time-sensitive transport mechanisms to preserve the real-time nature of the data. Second, the *transfer multiplicity* denotes the multiplicities at both ends of the transfer pipe and can take the following forms starting with the multiplicity of the information source:
  - **1-1** is single source, single destination transfer similar to unicast delivery;
  - **1-N** is single source, multiple (or group) destination transfer similar to multicast delivery; and
  - **N-1** represents multiple source delivery as is the case with swarming architectures (e.g. Bittorrent and USwarm [45]);
  - **N-N** is a multiple source, multiple destination delivery model. This model is probably the most intriguing. An example of such model would be a swarm-like information distribution to a multicast group.
- **Discovery/Search Functions**: Almost all of the aforementioned virtual addressing modes require searching for location of sought objects (which generally requires mapping of addresses to location(s)). The *search model* is characterized as follows:
  - **Physical-Only**: is the basic search model for locations (or alternatively routing on physical addresses), as with the Internet for example;
  - **Virtual-Decoupled** is the search model that necessitates searching for an information object before attempting to access (or perform other operation on) the latter. An explicit search instruction is hence necessary (for example LOOKUP operation in [40]). Other examples include some endpoint mapping (or search) in HIP [35], GSE [37] etc.
  - **Virtual-Embedded** is the search model that is generally associated with virtual addressing modes. In other words, the search is performed in-network alleviating the need for users (or endpoints) to explicitly perform a search prior to executing operations on some sought object (e.g. [20] and [33]);

  The form of the search function is largely dependent on the correlation of the address spaces.

The rest of the functions apply to primitive data objects in general. We simply distinguish those and we leave their characterization for a future work. To start with, **Transformation Functions** convert the data objects from one representation to another. Some examples include interstitial functions [19], and NAT boxes that perform address/instruction/protocol translation, and transcoding. **Decoding Functions** interpret the data objects and generate control vectors as a result. Interpretation of the data objects follows from their representation. On the current Internet, for example, every router interprets the instruction in the same way and generates a forwarding control vector that determines the next hop. **Construction Functions** produce new objects and their values. Information fusion/integration, aggregation, joining, splitting etc. are some examples of manufacturing functions. Finally, **Status Functions** get/set the various status blocks within the network, whether those involve internal network state or operation status.

Group Data Objects - A group data object is a collection of primitive objects that generally share some properties such as their type or their access control/policy. The constituent elements of the group belong to the same address space. While the group as a whole is an addressable entity, its constituent elements might not be individually addressable. Elements are identified by a combination of the group object identifier and the element identifier within the group. Group addressing is thus required for group object access. One example of a group object on the current Internet is the multicast group. Additional functions that apply to group data objects include: creating the group object, adding elements (group joining), removing elements (group leaving), and removing the group object.

Complex Data Objects - Complex data objects are simply data structures that the architecture makes explicit. It is intuitive that such complex data structures will emerge in the future, but it is hard at this stage to anticipate their properties. One may envision an explicit distributed stack data structure for example that is tailored to some architecture with an explicit push/pop usage model.

The third and final defining element of the information model is *the control structure* which defines the underlying control to support the information model. Almost every aspect of the information model discussed so far requires its dedicated control protocols and algorithms. For example, control for mapping characterizes the control required for mapping from virtual/physical address spaces to location, and for maintaining the pointers. Control for data access defines the control to support the addressing model and modes, etc. Clearly, such control structures (and others) represent a significant body of the networking research, where each aspect stands alone as a research topic by itself. Consequently, characterizing the control structure is beyond the scope of this paper and is left for a future work.

3.3 Towards a complete taxonomy

Our approach towards a complete network architecture taxonomy is syntactically defined (using a BNF metasyntax) in Table II. We have decided to represent the taxonomy textually rather than graphically since the textual representation is clear and compact. We clarify the following notation: ',' means concatenation; (x,y) means grouping in which terms x, and y are separated by any whitespace character; $\{x\}$ means a set of elements of x; $< x >$ means term x is left unspecified; and $[x]$ means optional term.

network_arch :=	('ARCH', id, 'begin', *substrate_struct*, *info_model*, 'end')
substrate_struct :=	('SUB_STRUCT', 'begin', *topology*, {*component*}, {*interconnection*}, 'end')
topology :=	(*top_struct*, *top_composition*), ";"
component :=	([id], *component_type*, *dispersal_factor*), ";"
interconnection :=	([id], *ic_type*, *ic_link*), ";"
top_struct :=	"hierarchical" | "flat" | "special"
top_composition :=	"controlled_overlap" | "integration" | "direct_peering"
component_type :=	"SE" | "ME" | "CE" | "PE" | "IPE" | "DPE" | "SWE"
dispersal_factor :=	"1" | "k" | "Z"
ic_type :=	"PE-PE" | "PE-SE"
ic_link :=	"dedicated" | "meshed" | "switched"
id :=	character, {character | digit | "_"}
info_model :=	('INFO', 'begin', *addr_struct*, {*data_type*}, $<$*control_struct*$>$, 'end')

addr_struct :=	(‘ADDR_STRUCT’, ‘begin’, *physical_space*, *virtual_space*, *pv_space*, ‘end’)
data_type :=	(‘DATA’, ‘begin’, *data_class*, {*function*}, ‘end’)
physical_space :=	(‘P_SPACE’, *space_structure*, *addr_scope*), “;”
virtual_space :=	(‘V_SPACE’, *space_structure*, *addr_scope*), “;”
pv_space :=	(‘PV_SPACE’, *space_correlation*), “;”
data_class :=	(*class_type*, {*data_object*})
function :=	(*binding_fcn* | *access_fcn* | *transfer_fcn*) | *search_fcn*), “;”
space_structure :=	“hierarchical_addr" | “flat_addr" | “special"
addr_scope :=	“local_scope" | “global_scope" | “partial_scope"
space_correlation :=	“independent" | “partitioned" | “embedded" | “overlap" | “equivalent"
class_type :=	(“primitive" | “group" | “complex"), “:"
data_object :=	(“paddress" | “vaddress" | “instruction" | “data_unit" | “stream" | “status_block" | “RIBS" | “TACO" | *group_object* | *complex_object*, *object_scope*), “;”
binding_fcn :=	(“FN_BINDING", {*assign_mode*})
access_fcn :=	(“FN_ACCESS", *access_type*, *access_paradigm*, {*access_mode*}, {*addressing_mode*})
transfer_fcn :=	(“FN_TRANSFER", *info_cognizant*, {*s2d_multiplicity*})
search_fcn :=	(“FN_SEARCH", *search_model*)
group_object :=	‘group_’, id
complex_object :=	‘complex_’, id
object_scope :=	*addr_scope*

assign_mode :=	“direct-value"	“processed-value"	“multiple-value"	“shared-value"
access_type :=	{*class_type*}
access_paradigm :=	“synchronous"	“asynchronous"
access_mode :=	“read"	“write"	“read-modify-write"
addressing_mode :=	“direct_paddr"	“direct_vaddr"	“indirect_paddr"	“indirect_vaddr"	“associative_paddr"	‘associative_vaddr"	“group_paddr"	“group_vaddr"	“k-preference"
info_cognizant :=	{*class_type*}
s2d_multiplicity :=	“1-1"	“1-N"	“N-1"	“N-N"
search_model :=	“physical_only"	“virtual_decoupled"	“virtual_embedded"

Table II: A BNF syntax for taxonomical specification of network architectures.

4 Applying the taxonomy

To illustrate the applicability of our taxonomy in terms of its classification powers, we have applied it to a rather special network architecture, the Data Oriented Network Architecture (DONA [30]) [16]. The textual description is listed below. In the listing below, ‘%’ stands for comment, “NA” means the term is irrelevant.

```
ARCH DONA
begin
 SUB_STRUCT begin
   % -topology structure
   hierarchical  NA; %Internet ASes
   % -components
   RH  IPE  1; %Resolution Handlers
   ROUTER IPE 1; %traditional BGP routers
   PROVIDER ME k; %content providers
   CACHE  CE  k; %content caches, extended RH
   % -interconnection structure
   PRVIDERS ME-ME meshed;
    %exploits hierarchical topology
   RH_RH PE-PE  meshed;
 end
 INFO_MODEL begin
  ADDR_STRUCT begin
  % -addressing structure
   P_SPACE hierarchical_addr global_scope;
        %IP addressable locations, but global
        % addressing is not necessary
   V_SPACE flat_addr global_scope;
                 %HIP style identification
```

[16] DONA’s description is based on our understanding of the architecture, which may well be incomplete.

```
   PV_SPACE independent;
  end %ADDR_STRUCT end
  DATA begin
  % -data type(s)
  primitive:
   paddress global; %IP, or maybe src route
   vaddress  global;% name is (P:L) tuple
   instruction  global; %find,lookup packets
   data_unit global; stream global;
   RIBS  global; % datum-metadata, service
      %e2e type intelligence
   % -functions
   FN_BINDING direct-value;
               %REGISTER(.) func
   FN_ACCESS  primitive:group:
     synchronous
      direct_paddressing direct_vaddressing
       k-preference; %anycast FIND(.) func
   FN_TRANSFER  NA 1-1 1-N; % for multicast
   FN_SEARCH  virtual_decoupled;
            % FIND required
   end %DATA
 end %INFO end
end
```

A significant amount of knowledge about the architecture is conveyed by simply observing such a compact taxonomical representation. Additionally, architectures are easily compared along their convergence and divergence points by observing their respective representations side by side. For example, it easy to notice the significant similarity, from our taxonomy point of view, between DONA and TRIAD [24] by representing the latter. Aside from the differences in terms of the control structure and name semantics which we do not consider in the taxonomy, their main other difference is the virtual space structure.

5 Related Work

The major differentiator of our work is its generality in understanding networks at the architectural level rather than being confined to the communication/switching properties, or to the computational properties, or to particular scoped network architectures that focus on naming, or routing, or content delivery. This section overviews the related work. To start with, some recent work has particularly focused on creating a taxonomy for overlay (or virtualized) networks, relative to the current Internet. Clark et. al [18], presents a taxonomy of overlays that helps thinking about their motivations and their implications. Augusto [6] classifies networks based on their application-specific or purpose-specific nature. Moreover, [32] presents a simple taxonomy of Network Computing (NC) systems (or overlays) based on their applications, platforms, and management. Additionally, classifying a particular type of overlay, the Content Delivery Networks (CDN), has been the subject of some recent work [12, 31]. Again, while all this work is related (and complementary) to ours, our work addresses the general architecture classification problem.

Other recent work has focused on modeling and reasoning about the communication aspects (mainly switching and binding properties) of networks [29, 49]. Such network modeling work is complementary to ours in trying to better understand and formally reason about the network architecture space.

Classifying programmable networks has also been addressed in the literature [14, 43]. Reference [14] provides a generalized model for programmable networks that explicitly includes a computational model relevant to

such networking environments. In the same way, our general service model perspective acknowledges a computational model as a building block for modern network architectures.

6 Discussion: Value and Limitations

The main contribution of our work is a taxonomy that helps organizing and thinking about the architectural space. The taxonomy is based on a bottom up characterization of a network starting with the underlying physical substrate (the topology, components, and their interconnections) and ending with the information model (the addressing structure, the data objects and the operations allowed on them, and the control structure). In terms of value, in addition to offering a comprehensive overview of the set of architectural possibilities as well as a tutorial for introduction to the field, our taxonomy helps organizing and thinking about the architecture space beyond the communication model (switching/delivery characteristics of networks) which has been the major approach adopted in the literature for taxonomizing architecture (as we have seen with connection-oriented vs connectionless models). The latter approach has weak discriminatory power and hence fails to distinguish the different (and particularly modern) architectures as to their information structures. Our taxonomy additionally helps in highlighting gaps in the design space for exploring new contributions to the field by identifying unexplored research areas. Examples of some gaps that were highlighted include the *ISA* support for binding models and addressing modes, the N-N delivery, and address *space correlation*. Moreover, the descriptive nature of our taxonomy helps in comparing modern network architectures along their convergence and divergence points. Finally, the taxonomy helps set the stage to for attempting to answer the question of whether intelligence in the network is useful, and what is the minimal set of functionality that could be part of an architecture while maintaining the elegance of end-to-end design. In this regard, we lack the expertise necessary to take any position in trying to answer those questions.

In terms of its limitations, the paper in its current form falls short of providing tangible outcomes beyond the descriptive one. Additionally, by no means do we claim that our taxonomy is complete. The taxonomy does not provide characterizations for the control structure, for security in general, and for the timeliness of information. Each of these missing structures spans multiple aspects of the information model, and their treatment is left for a separate work due to lack of space. Despite those limitations at this point, which we plan to address in a future work, we believe that this paper is helpful to the community.

Finally, the authors would like to note that the evaluation of the FIND proposals throughout the discussion are solely based on the preliminary architectural descriptions provided on the FIND website [4]. Although not complete, we still believe that incorporating the FIND clean-slate architectural work is crucial to achieving the goals of this paper. However, we do apologize in advance for any misunderstandings of the architectural descriptions.

7 Acknowledgements

The authors would like to thank KC Klaffy, Jon Crowfort, and Mark Handley for their valuable comments.

References

[1] Delay tolerant networking research group. http://www.dtnrg.org/.

[2] GENI: Global environment for network innovations. http://www.geni.org/.

[3] NewArch: Future generation internet architecture. http://www.isi.edu/newarch/.

[4] NSF Nets FIND initiative. http://www.nets-find.net/.

[5] NSF nets FIND: Breakout on networking at the information layer, 4th pi meeting. [online]: www.nets-find.net/Meetings/FourthPIMeeting/ BreakOut/InformationLayer.pdf, November 2007.

[6] M. Augusto, L. Silva, J. Quaini-Sousa, F. Redigolo, T. C. Carvalho, H. C. Guardia, W. Ruggiero, and B. Ohlman. A proposal for a taxonomy for virtual networks. Real Overlays And Distributed Systems (ROADS) workshop, Poland, 2007.

[7] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A layered naming architecture for the internet. In *Proceedings of ACM SIGCOMM 2004*, pages 343–352, Portland, Oregon, USA, August 2004. ACM.

[8] S. M. Bellovin, D. D. Clark, A. Perrig, and D. Song. A clean-slate design for the next-generation secure internet, March 2005. This is the report of an NSF workshop held in July, 2005.

[9] B. Bhattacharjee, K. Calvert, J. Griffioen, N. Spring, and J. Sterbenz. Postmodern internetwork architecture. NSF Nets FIND Initiative.

[10] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475, 1998.

[11] Z. I. Boon Thau Loo, Jonathan Smith. Wireless knowledge infrastructure (wiki). NSF Nets FIND Initiative.

[12] R. Buyya, M. Pathan, and A. Vakali. *Content Delivery Networks*, volume 9 of *Lecture Notes in Electrical Engineering*. Springer-Verlag, Germany, 2008.

[13] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. Rofl: routing on flat labels. In *Proceedings of SIGCOMM 2006*, pages 363–374, New York, NY, USA, 2006. ACM Press.

[14] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela. A survey of programmable networks. *SIGCOMM Comput. Commun. Rev.*, 29(2):7–23, 1999.

[15] Cisco Systems Inc. Internetworking technologies handbook. Indianapolis, IN: Cisco Press, 2004.

[16] D. Clark. The design philosophy of the darpa internet protocols. In *SIGCOMM '88: Symposium proceedings on Communications architectures and protocols*, pages 106–114, New York, NY, USA, 1988. ACM Press.

[17] D. Clark, R. Braden, A. Falk, and V. Pingali. Fara: reorganizing the addressing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 313–321, New York, NY, USA, 2003. ACM Press.

[18] D. Clark, B. Lehr, P. Faratin, R. Sami, and J. Wroclawski. Overlay networks and future of the internet,. *Communications and Strategies*, (63):1–21, 2006.

[19] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield. Plutarch: an argument for network pluralism. *SIGCOMM Comput. Commun. Rev.*, 33(4):258–266, 2003.

[20] D. Farinacci, V. Fuller, D. Oran, and D. Meyer. Locator/id separation protocol (lisp). http://tools.ietf.org/html/draft-farinacci-lisp-03.

[21] A. Feldmann. Internet clean-slate design: what and why? *SIGCOMM Comput. Commun. Rev.*, 37(3):59–64, 2007.

[22] C. Gavoille. Routing in distributed networks: overview and open problems. *SIGACT News*, 32(1):36–52, 2001.

[23] W. K. Giloi. Towards a taxonomy of computer architecture based on the machine data type view. In *ISCA '83: Proceedings of the 10th annual international symposium on Computer architecture*, pages 6–15, Los Alamitos, CA, USA, 1983. IEEE Computer Society Press.

[24] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, pages 4–4, Berkeley, CA, USA, 2001. USENIX Association.

[25] M. Gruteser. A geometric stack for location-aware networking. NSF Nets FIND Initiative.

[26] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[27] D. Johnson, C. Perkins, and J. Arkko. RFC 3775: Mobility support in ipv6, June 2004.

[28] R. Kahn, C. Abdallah, H. Jerez, G. Heileman, and W. Shu. The Transient Network Architecture (TNA). NSF Nets FIND Initiative.

[29] M. Karsten, S. Keshav, S. Prasad, and M. Beg. An axiomatic basis for communication. In *Proceedings of SIGCOMM 2007*, pages 217–228, New York, NY, USA, 2007. ACM Press.

[30] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *Proceedings of SIGCOMM'07*, Kyoto, Japan, August 27-31 2007. ACM.

[31] H. Kung and C. Wu. Content networks: Taxonomy and new approaches. In *The Internet as a Largescale Complex System*, Santa Fe Institute Series. Oxford Press, 2002.

[32] M. Maheswaran and S. Ali. A taxonomy of network computing systems. *Computer*, 37(10):115–117, 2004.

[33] D. Massey, L. Wang, B. Zhang, and L. Zhang. A scalable routing system design for future internet. In *ACM SIGCOMM workshop on IPv6 and the Future of the Internet*, New York, NY, USA, 2007. ACM Press.

[34] D. Meyer, L. Zhang, and K. Fall. Report from the iab workshop on routing and addressing. Internet RFC 4984, Sep 2007.

[35] R. Moskowitz, P. Nikander, and P. Jokela. Host identity protocol architecture. RFC 4423, May 2006.

[36] T. S. E. Ng, I. Stoica, and H. Zhang. A waypoint service approach to connect heterogeneous internet address spaces. In *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, pages 319–332, Berkeley, CA, USA, 2001. USENIX Association.

[37] M. OÕDell. An alternate addressing architecture for ipv6. IETF Draft. February 1997.

[38] P. F. Ramakrishna. Ipnl: A nat-extended internet architecture. In *Proceedings of SIGCOMM 2001*, pages 69–80, New York, NY, USA, 2001. ACM Press.

[39] J. Saltzer. On the naming and binding of network destinations. RFC 1498.

[40] S. Schmid, L. Eggert, M. Brunner, and J. Quittek. TurfNet: an architecture for dynamically composable networks. In *WAC*, pages 94–114, 2004.

[41] S. Shenker. Fundamental design issues for the future internet. *IEEE Journal on Selected Areas in Communication*, 13(7), September 1995.

[42] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. *IEEE/ACM Transactions on Networking*, 12(2):205–218, Apr. 2004.

[43] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[44] Z. Turányi, A. Valkó, and A. T. Campbell. 4+4: an architecture for evolving the internet address space back toward transparency. *SIGCOMM Comput. Commun. Rev.*, 33(5):43–54, 2003.

[45] A. Venkataramani and D. Towsley. A swarming architecture for internet data transfer. NSF Nets FIND Initiative.

[46] T. Wolf. Service-centric end-to-end abstractions for network architecture. NSF Nets FIND Initiative.

[47] X. Yang. Nira: a new internet routing architecture. In *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pages 301–312, New York, NY, USA, 2003. ACM Press.

[48] R. Yates, D. Raychaudhuri, S. Paul, and J. Kurose. Postcards from the edge: A cache-and-forward architecture for the future internet. NSF Nets FIND Initiative.

[49] P. Zave. Compositional binding in network domains. In *FM*, volume 4085 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 2006.