A Brief Overview of the NEBULA Future Internet Architecture

Tom Anderson[1] Ken Birman[2] Robert Broberg[3] Matthew Caesar[4]
Douglas Comer[5] Chase Cotton[6] Michael J. Freedman[7] Andreas Haeberlen[8]
Zachary G. Ives[8] Arvind Krishnamurthy[1] William Lehr[9] Boon Thau Loo[8]
David Mazières[10] Antonio Nicolosi[11] Jonathan M. Smith[8] Ion Stoica[12]
Robbert van Renesse[2] Michael Walfish[13] Hakim Weatherspoon[2]
Christopher S. Yoo[8]

[1]University of Washington [2]Cornell University [3]Cisco Systems [4]University of Illinois
[5]Purdue University [6]University of Delaware [7]Princeton University [8]University of Pennsylvania
[9]Massachusetts Institute of Technology [10]Stanford University [11]Stevens Institute of Technology
[12]University of California, Berkeley [13]University of Texas, Austin

ABSTRACT

NEBULA is a proposal for a Future Internet Architecture. It is based on the assumptions that: (1) cloud computing will comprise an increasing fraction of the application workload offered to an Internet, and (2) that access to cloud computing resources will demand new architectural features from a network. Features that we have identified include dependability, security, flexibility and extensibility, the entirety of which constitute *resilience*.

NEBULA provides resilient networking services using ultra-reliable routers, an extensible control plane and use of multiple paths upon which arbitrary policies may be enforced. We report on a prototype system, Zodiac, that incorporates these latter two features.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.6 [**Computer-Communication Networks**]: Internetworking

Keywords

Internet, network architecture, security, routing, extensibility

1. INTRODUCTION

The Internet architecture [8, 23, 9] is an obvious success, with interesting applications continuing to emerge at a rapid pace. However, certain applications categories remain a concern.

Imagine, for example, a healthcare application that might use a future Internet: a diabetic wears both an insulin pump and a continuous glucose monitor (CGM). Data from the CGM are sent over the network to a data center every 5 minutes, while other data such as images of meals, accelerometer readings of activity, *etc.*, are sent as needed. These data are logged, and analyzed against both historical data from the individual and anonymized masses of data from other data sources. Machine learning algorithms are used to estimate appropriate micro-dosages of insulin to be delivered by the pump, as well as to detect anomalies that might can be forwarded to human experts who can ensure that no medical problem has occurred. Dosage values are downloaded using the network into the patient's insulin pump.

Such a healthcare application has clear dependability and security requirements. Such needs are clearly shared by other applications, including teleoperation of vehicles, telemanufacturing tasks such as remote 3-D printing, and remote feedback scenarios such as telesurgery.

The challenge in designing an *architecture*, as opposed to a solution to a specific problem, is that one must anticipate the emergence of unanticipated applications. Any future Internet must preserve the existing Internet's flexibility and extensibility while accommodating important new classes of applications, such as those sketched above. A key question is whether to attempt to enumerate many possible futures and accommodate all of them, or to pick a likely future and do research towards enabling that choice.

We chose the latter strategy, and focused NEBULA on cloud computing, as we discuss further in Section 3. We report on our architectural choices in Section 4, discuss integration in Section 5, an initial prototype in Section 6, some initial reflections on the project in Section 7 and conclude in Section 8.

2. BACKGROUND / RELATED WORK

Network design has many dimensions, but history has shown that extensibility to meet unanticipated application needs is extremely important. Telephony achieved extensibility by refining services offered using programmable switches [7], but premises equipment evolved more slowly.

The architecture of the Internet [8, 9] is based on both (1) an elegant interoperability model based on a packet-switching overlay using disparate subnets, and (2) well-chosen "rules of thumb" such as pushing the primary locus of evolution to the endpoints (hosts) [23]. This latter point has been key to exploiting the continuing exponential improvements in computer performance due to Moore's Law.

The rapid evolution of endpoint services possible with computers attached to an Internet is clear, but it is also clear that the Internet model makes advanced in-network services, such as the desirable capability for IP-layer multicast, more difficult to deploy. Consequently, the Internet architecture makes it harder for the *network itself* to evolve.

Attempts have been made to synthesize the evolvability advantages of telephony's switches and the Internet's end-hosts, perhaps most notably the approach of Active Networks [25] – networks that allowed both users and providers to dynamically deploy new services to support their applications. Interestingly, in fits and starts [13], elements of this approach have made their way into today's Software-Defined Networks.

Other approaches are possible. For example, *Content-Centric Networking* (CCN) [15] posits that the Internet architecture should evolve to focus on content, and routes named units of content rather than packets. In addition to the work originated by Jacobson, some additional proposals for Future Internets based on CCN have emerged. The eXtensible Internet Architecture (XIA) [2] project is targeted at a content-centric architecture, but also at architectural extensibility. Named Data Networking [26] almost exactly follows Jacobson's proposal.

A different approach, more in line with that of NEBULA's choice of one particular future, is MobilityFirst [24], which posits a future Internet driven by billions of mobile devices such as smartphones.

3. TARGETING THE CLOUD

The ubiquity of the Internet has given rise to a new form of computing, cloud computing [4], where services are made available using networked access to one or more large data centers with shared computing and storage resources – in effect, a distributed form of the 1960s "computing utility" [12] vision. The economic advantages of sharing resources are clear, and additional benefits accrue from the computational and storage resources available. Decision-making, for instance, can be improved with access to archives of user, historical, and logistical data. Global coordination and forecasting or planning are often much more effective than distributed coordination. Today, cloud computing services are increasingly the coordination point among always-on mobile devices, such as tablets and smartphones.

For all of these reasons, we believe that cloud computing will play a central role in the Internet of the future. The requirements of cloud-centric services have several implications for a future Internet and the connection properties it provides to end hosts, distributed sites, and data centers:

1. If cloud-based storage, computation, and control/coordination are to replace the local storage and computation facilities we have today, access to the cloud must be highly *dependable* to avoid a loss of availability or integrity, or to avoid fluctuations in timing.

2. Mission-critical data and infrastructure hosted on the cloud means the network must be *secure* to prevent data and control from being corrupted or falling into the wrong hands.

3. The cloud is still in its infancy, and new applications continue to be invented. The network must be sufficiently *flexible* and *extensible* to provide connections meeting their needs.

The four properties in italics are thus essential for a future Internet architecture.

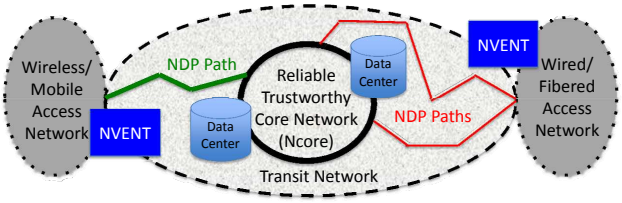

Figure 1: Architectural overview of NEBULA

4. NEBULA OVERVIEW

The NEBULA Future Internet Architecture project [3, 19] has been investigating a new Internet architecture that supports cloud computing [10] by providing the properties discussed at the end of Section 3.

Figure 1 shows the high-level architecture of NEBULA. NEBULA consists of three tiers: the *network core* (NCore) that connects data centers to each other, the *NEBULA data plane* (NDP) that connects the data centers to the access (edge) networks, and the NEBULA Virtual and Extensible Networking Techniques (NVENT) that offers users a dynamic and flexible spectrum of connectivity choices — including, for instance, paths with HIPAA assurances that can be used for protected health information, or high-reliability paths.

NEBULA Core Architecture (NCore): NCore is based on a model [1] of high-performance core routers, as well as richer interconnection topologies for both data center attachment and NCore router interconnection [16]. It uses ideas from distributed systems fault tolerance to achieve high reliability. Research has resulted in new, ultra-reliable router architectures [1, 11], as well as interconnection architectures [21, 16] for data centers that can leverage such ultra-reliable routers.

NEBULA Data Plane (NDP): NDP incorporates new data-plane technologies for resilient access, allowing communication only when all involved parties, such as endpoints and transit networks, have agreed to participate (this is desirable for reasons of confidentiality, integrity and availability).

NDP contains as its key element a *path verification mechanism* called ICING [18]. ICING ensures that, before packets are sent over any network path, each domain along the path has (explicitly or through delegation) consented to the use of the path. A domain's consent is embodied in a cryptographic token called a *proof of consent* (PoC), which the sender embeds in each packet that she launches along the path. As the packet traverses the path, it is incrementally marked with *proofs of provenance* (PoPs), which essentially certify that the packet has indeed traveled through each domain on the path, in the correct order.

The requirement for explicit consent is a major difference from the current Internet architecture, and substantially improves security: for instance, since all traffic must be explicitly authorized and strong cryptographic mechanisms thwart spoofing, denial-of-service attacks are much harder to carry out. ICING also enables NEBULA to enforce a much richer set of policies—*e.g.*, a domain can refuse to carry traffic that has not yet traversed a firewall that is located in another domain.

NEBULA extensible control plane (NVENT): NVENT embodies new control-plane technologies that focus on policy specification, policy-based path setup [6] and service naming [20].

NVENT uses *declarative networking* [17], based on Network Datalog. This declarative approach lets administrators provide high-level *specifications* of their routing policies, without having to

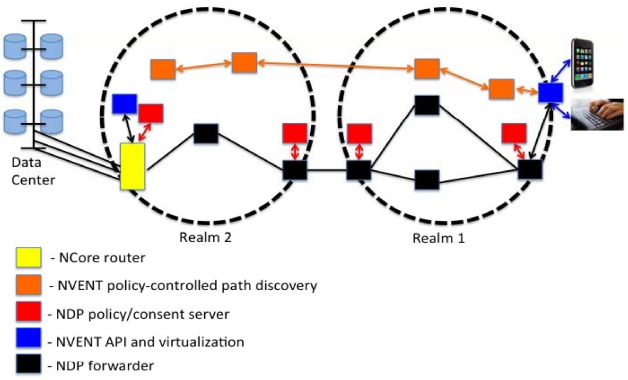

Figure 2: Integration of NEBULA Components

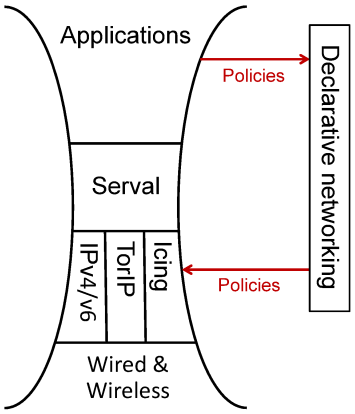

Figure 3: NEBULA revision of the IP "hourglass"

worry about implementation details (and getting them right). The resulting specifications tend to be very concise: complex policies can often be specified with just a handful of rules. This makes it easier for administrators to update and evolve their policies over time. Just as BGP in the current Internet, NVENT provides a set of default paths to ensure global reachability, but it also provides an interface to NDP, which is available to users for requesting custom paths, *e.g.*, for applications that require high reliability. These custom paths are negotiated and set up on demand.

5. PUTTING NEBULA TOGETHER

Figure 2 illustrates how the three tiers work together to negotiate a custom end-to-end path (*e.g.*, for sensitive health data) from a cell phone to a data center. The cell phone contacts NVENT and requests a path to NCore. NVENT looks for a suitable path that complies with the policies of each network, and it contacts the NDP policy server in each network to obtain the necessary proofs of consent (PoCs), which it then returns to the phone. The phone can use the PoCs to send packets via NDP to the nearest NCore router, which inspects the proofs of provenance (PoPs) to check that the negotiated path has been followed, and then uses its NCore links to forward the packets to the correct data center.

A policy server will have zero or more policies. The default policy is to drop traffic, sometimes called "deny by default". Policies are assumed to be dynamic (changeable) but we assume they are changed infrequently, and thus are cacheable. In our initial architecture, we expect that users and prototype applications will want easy to state policies, *e.g.*, a policy indicating HIPAA compliance would be stated as "`HIPAA=yes`". A policy server's policies can be queried by clients or consent servers. A path is constructed from consenting servers.

A user or application specifies policy requirements, *e.g.*, `NEBULAPATH=HIPAA`. The application specifies a destination or service. When this specification is received, the system checks a cache for a cached compliant path to the destination or service. If such a path is available, NEBULA tries to get consent to use the path, perhaps with cached proofs of consent if obtaining consent is expensive. If nothing is cached, or there is no consent for a cached path, the system iterates requests for consent to consent servers. The end result is that NEBULA will either establish and cache a path, or will fail with an error.

Packets carry secure "markings" of consent. This might be the cryptographic seal implied by Onion Routing in TorIP. These "marks" are updated at "realm" (*e.g.*, ISP) boundaries. There are checks to see whether a packet is "permitted".

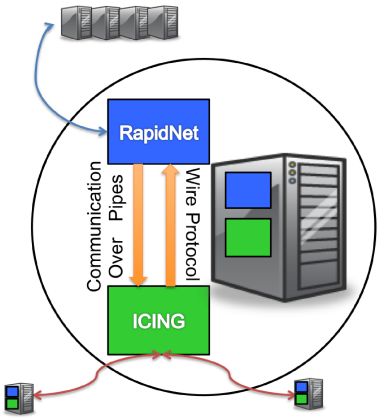

Figure 4: A NEBULA node from Zodiac [5]

6. ZODIAC NVENT+NDP PROTOTYPE

Figure 3 shows how the architectural elements are layered. Note that as functionality is added "in-network", the waist of the hourglass must broaden beyond the packet format and addressing required to be standardized by IP. The multiple verticals for NDP indicate that the NEBULA project was exploring multiple visions for NDP. In the integration and prototyping effort described next, we have used Serval [20] and Declarative Networking [17] to constitute NVENT, and ICING [18] as a choice for NDP.

We have built a preliminary prototype of an integrated NEBULA control plane and NEBULA data plane called Zodiac [5] that combines elements of NVENT and NDP, thus integrating several elements of Figure 3. In the following, we provide a brief description of this prototype.

6.1 Overview

Figure 4 illustrates the internal structure of a network node in our prototype design. Not unlike a router in the current Internet, the node has a "data plane" and a "control plane": the former consists of the NDP path verification mechanism, which is based on ICING, while the latter consists of NVENT's routing and policy mechanism and is based on the RapidNet [22] declarative networking engine.

Each administrative domain can install its own policies to describe what kinds of paths it permits in its network. The policies are written in Network Datalog (NDlog), a declarative language;

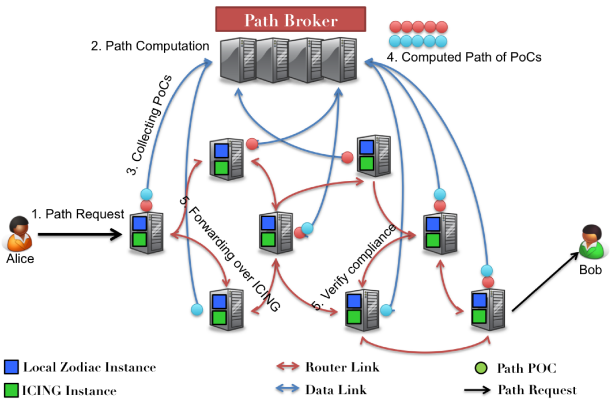

Figure 5: PoC construction in Zodiac [5]

this enables administrators to state policies very concisely, in just a few lines of code, and it facilitates the process of writing and updating the policies. In addition, our prototype design contains the notion of a *path broker*; this is a special kind of node that collects information about policies and locally available paths. Figure 5 illustrates how these components are interconnected.

During normal operation, the path brokers generate a set of default best-effort paths that provide basic connectivity, just as in the current Internet. However, networks and end users with the appropriate credentials can submit *queries* for paths with specific properties. For instance, a hospital that is about to perform telesurgery on a remote patient might request three redundant paths between the hospital and the patient's location that each have sufficient bandwidth and traverse only domains that advertise compliance with HIPAA (and are perhaps certified by an appropriate industry consortium, similar in spirit to 'privacy seal' programs like TRUSTe's or BBBOnLine's). The path broker may have to contact other path brokers to process this query – *e.g.*, to find paths that do not share any interior nodes. Once a set of suitable paths is found, the path broker contacts the domains along the path and requests the appropriate NDP credentials – cryptographic proofs of consent (PoC) – which are then returned to the user that made the request.

6.2 Query processing

Figure 5 illustrates the PoC creation process in some more detail. The process begins when the sender (shown on the left) attempts to transmit a packet to a destination for which the user has formulated a special policy. NDP detects that no PoC is available for that destination yet, and therefore contacts NVENT (1), which issues a query to the path broker (2). Once the path broker has identified a candidate path (the three green nodes), it contacts the NVENT instances along the path (3), which check compliance with their local policies and then ask the local ICING policy server to issue a PoC for the local segment of the path (4). These individual PoCs are then returned to the path broker, which assembles them into an end-to-end PoC and returns it to the NDP instance on the sender (5). From that point on, the sender can generate cryptographic tokens for the packets it wants to send along the path. In our prototype, end nodes can "stripe" their traffic across multiple paths for redundancy; an alternative approach would be to detect path failures and to fail over to an alternative path. In either case, the failure of a single path, or even a small number of paths, does not interrupt the user's connection.

```
materialize(link,infinity,infinity,keys(1:
    str,2:str)).
materialize(datalink,infinity,infinity,keys
    (1:str,2:str)).
materialize(routerIP,infinity,infinity,keys
    (1:str,2:str)).
materialize(pendingPing,infinity,infinity,
    keys(3:str)).
materialize(pathRequest,infinity,infinity,
    keys(1:str,2:str)).
materialize(minPathRequest,infinity,infinity
    ,keys(1:str,2:str)).
materialize(hipaaPathRequest,infinity,
    infinity,keys(1:str,2:str)).
materialize(localPOC,infinity,infinity,keys
    (1:str,2:str,3)).
materialize(replyPOC,infinity,infinity,keys
    (1:str, 2:str,3)).
materialize(replyPOCCount,infinity,infinity,
    keys(1:str,2:str)).
```

Figure 6: NDlog Relations for Maintaining State at Routers, from Zodiac [5] integration prototype

6.3 Network state

To illustrate how policies are implemented in practice, we show some of the relations that our prototype maintains in Figure 6. The `link` table stores the topology of the network, as well as the attributes of the available links. (Like all the other tables, this table is not stored anywhere in its entirety; each node stores the entries that pertain to its local links.) In our prototype, an entry in the `link` table is of the form (A, B, c, d, h), where A and B are NEBULA nodes, c is the capacity of the link, d is the propagation delay, and h indicates HIPAA compliance; other properties would not be difficult to add. `datalink` is a similar table that describes the connections to each node's path broker(s). `routerIP` is an artifact of our prototype, which is implemented as an overlay over an existing IP network; it maps our internal node identifiers to IP addresses. `pendingPing` is used by a simple link failure detection mechanism.

`pathRequest`, `minPathRequest`, and `hipaaPathRequest` store three types of path queries that can be issued in our prototype. Adding more query types would not be difficult, thanks to NDlog's flexibility; each request type should require only a few more lines of NDlog code.

`localPOC` is used to store the proofs of consent (PoCs) that the node has generated locally; `replyPOC` stores PoCs that have been received from the path broker; and `replyPOCCount` keeps track of the number of PoCs that have been received for a particular query. We note that paths could fail or be withdrawn because a network along the path no longer consents to their use; this could be handled by submitting the path request as a continuous query, which would enable the path broker to automatically replace failed paths once the number of working paths becomes too low.

6.4 Interdomain paths

Figure 7 illustrates how the path broker discovers suitable interdomain paths in a setting with multiple path brokers. The process works by chaining together suitable peering links, taking bandwidth constraints and HIPAA requirements into account. Notably, even this complex process can be described with just a few lines of declarative code.

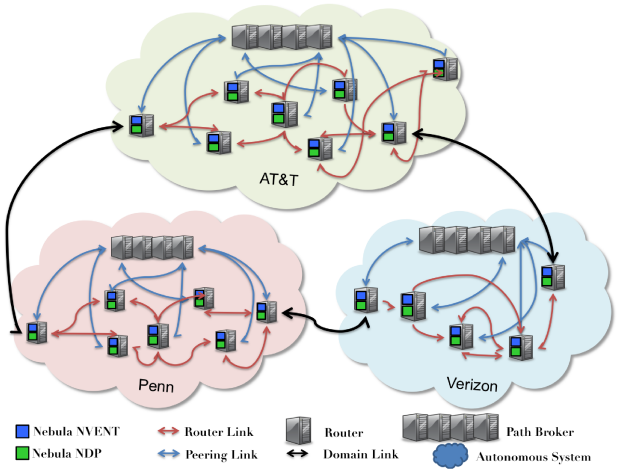

Figure 7: NEBULA interdomain routing, from Zodiac [5] integration prototype

6.5 Status

Our implementation is based on a software-only implementation of ICING [14] and the RapidNet declarative networking engine [22]. A clip of a demo is available at the project web site [19], showing a video being streamed over three redundant NEBULA connections; when faults are injected into some of the paths, the video is unaffected and continues to play.

7. DISCUSSION

As might be expected of a complex project with a large team of researchers, significant management effort was required to bring the architecture to fruition. One surprising problem that created difficulty for us was a lack of clarity on exactly what an "architecture" was. For example, is an architecture defined by the design goals? The components and the interaction of components? An abstract description of a vision and its realization? Or something else entirely? In the end, we tried to loosely follow the model of Clark [9].

As a management mechanism, and because the members of our team brought significant research and implementation experience to the table, we decided to loosely organize our research around the NVENT, NCORE and NDP designs in the first year of NEBULA, and then gradually integrate. For example, a subgroup interested in router reliability [1, 11] focused their energies on bringing fault-tolerance strategies from distributed systems research to the context of core routers that comprise hundreds of line cards and processors. Thus, even if the more radical proposals for policy enforcement did not transition immediately, we were hopeful that our results could influence the router vendor community.

As designs for the elements began to emerge, we realized that postponing the integration of the architecture was a significant strategic error. In retrospect the deepest questions were not in the component research, interesting as it was. Rather, they were in the integration of the components into an architecture that achieved the NEBULA agenda of resilience for new applications — those that would not use an Internet without novel features such as NDP's policy enforcement. One example is the challenge of specifying and enforcing interrealm/interdomain policies. Other examples include policies for path discovery in a federation (today addressed with BGP), and the API used for application specification of policies.

Design Goal	NEBULA
Communication must continue despite loss of networks, links or gateways	NEBULA uses multiple dynamically allocated paths and reliable transport
Allow host attachment and operation with a low level of effort	NVENT/NDP is as easy to automate and use as DHCP/IP
Support secure communication (authentication, authorization, integrity, confidentiality) among trusted nodes	Mutually-suspicious NDP nodes self-select paths exhibiting cryptographic proofs of properties required for security
Provide a cost-effective communications infrastructure	NCORE places resources where architecturally needed, and benefits from regulatory and policy research
Implement network and user policies	Policies implemented with NDP and NVENT
The architecture must accommodate a variety of networks	NDP send packets using encapsulation; NVENT accommodates multiple network types with a service-oriented API
The architecture must permit distributed management of its resources	NDP path establishment satisfies a decentralized federation of realms

Table 1: NEBULA architectural goals and design solutions.

There are perhaps too many of these questions for realizing a complete integrated NEBULA architecture before the project ends. Nonetheless, we have done some experimental work with our Zodiac prototype that indicates that the components can be composed into a functioning system. This piecewise integration reinforces our belief that a NEBULA realization is feasible, albeit one requiring additional thought and engineering to fully instantiate.

8. CONCLUSION AND NEXT STEPS

NEBULA is a novel future Internet architecture that addresses networking challenges for cloud computing. Table 1 summarizes the architectural choices made in NEBULA, following Clark's [9] similar summary.

NEBULA is focused on resilience, and incorporates routers hardened with distributed systems technology, an extensible control plane and a flexible approach to policy enforcement. These latter two elements have been combined in the Zodiac prototype discussed in Section 6.

We continue to work towards integrating more of the research generated in the NEBULA project into operational software. For example, a prototyping effort to integrate Serval and ICING has recently been completed.

9. ACKNOWLEDGMENTS

The NEBULA project was supported by the U.S. National Science Foundation. We also appreciate the support of Cisco Systems.

10. REFERENCES

[1] Andrei Agapi, Ken Birman, Robert M. Broberg, Chase Cotton, Thilo Kielmann, Martin Millnert, Rick Payne, Robert Surton, and Robbert van Renesse. Routers for the Cloud: Can the Internet achieve 5-nines availability? *IEEE Internet Computing*, 15(5):72–77, 2011.

[2] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, Wenfei Wu, Aditya Akella, David Andersen, John Byers, Srinivasan Seshan, and Peter Steenkiste. XIA: An architecture for an evolvable and trustworthy Internet. In *Proc. ACM HotNets-X*, 2011.

[3] Tom Anderson, Ken Birman, Robert Broberg, Matthew Caesar, Douglas Comer, Chase Cotton, Michael J. Freedman, Andreas Haeberlen, Zachary G. Ives, Arvind Krishnamurthy, William Lehr, Boon Thau Loo, David Mazières, Antonio Nicolosi, Jonathan M. Smith, Ion Stoica, Robbert van Renesse, Michael Walfish, Hakim Weatherspoon, and Christopher S. Yoo. *The NEBULA Future Internet Architecture*, volume 7858 of *LNCS*. Springer Verlag, 2013.

[4] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A Berkeley view of Cloud computing. Technical Report UCB/EECS-2009-28, EECS, U. C. Berkeley, Feb. 10 2009.

[5] Dhruv Arya. Zodiac: A Control Plane for Nebula. Master's thesis, U. Penn., Phila., PA, April 2013.

[6] Matvey Arye, Robert Kiefer, Kyle Super, Erik Nordström, Michael J. Freedman, Eric Keller, Tom Rondeau, and Jonathan M. Smith. Increasing network resilience through edge diversity in NEBULA. *ACM SIGMOBILE Mobile Computing and Communications Review*, 16(3), December 2012.

[7] Bell Communications Research. *AIN Release 1 Service Logic Program Framework Generic Requirements*. FA-NWT-001132.

[8] Vinton G. Cerf and Robert E. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, COM-22(5):637–648, May 1974.

[9] David D. Clark. The design philosophy of the DARPA internet protocols. In *Proc. SIGCOMM*, pages 106–114, 1988 .

[10] Douglas Comer. A future Internet architecture that supports Cloud Computing. In *Proc. 6th International Conference on Future Internet Technologies (CFI)*, June 2011.

[11] Douglas Comer and Salman Javed. Applying open resilient cluster management (ORCM) to a multi-chassis core router. In *Proc. 27th International Conference on Computers and Their Appliactions (CATA)*, March 2012.

[12] Robert M. Fano. The MAC system: The computer utility approach. *IEEE Spectrum*, 2:56–64, Jan. 1965.

[13] Nick Feamster, Jennifer Rexford, and Ellen Zegura. The Road to SDN. *ACM Queue*, 11(12):20, 2013.

[14] ICING source code. `http://www.cs.stevens.edu/~nicolosi/projects/icing/src/icing-1.1.tar.gz`.

[15] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. Networking named content. In *Proc. ACM CoNEXT*, 2009.

[16] Vincent Liu, Daniel Halperin, Arvind Krishnamurthy, and Thomas Anderson. F10: A fault-tolerant engineered network. In *Proc. NSDI*, April 2013.

[17] Boon Thau Loo, Tyson Condie, Minos Garofalakis, David E. Gay, Joseph M. Hellerstein, Petros Maniatis, Raghu Ramakrishnan, Timothy Roscoe, and Ion Stoica. Declarative networking. *Communications of the ACM*, 52(11):87–95, November 2009.

[18] Jad Naous, Michael Walfish, Antonio Nicolosi, David Mazières, Michael Miller, and Arun Seehra. Verifying and enforcing network paths with ICING. In *Proc. CoNEXT*, 2011.

[19] NEBULA project web page. `http://nebula-fia.org/`.

[20] Erik Nordström, David Shue, Prem Gopalan, Robert Kiefer, Matvey Arye, Steven Y. Ko, Jennifer Rexford, and Michael J. Freedman. Serval: An end-host stack for service-centric networking. In *Proc. NSDI*, 2012.

[21] Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. FairCloud: Sharing the network in cloud computing. In *Proc. HotNets*, 2011.

[22] RapidNet project web page. `http://netdb.cis.upenn.edu/rapidnet/`.

[23] Jerome H. Saltzer, David P. Reed, and David D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[24] Ivan Seskar, Kiran Nagaraja, Sam Nelson, and Dipankar Raychaudhuri. MobilityFirst Future Internet Architecture. In *Proc. ACM Asian Internet Engineering Conference (AINTEC)*, 2011.

[25] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[26] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobson, James D. Thornton, Diana K. Smetters, Beichuan Zhang, Gene Tsudik, kc claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulous, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. Named data networking (NDN) project. `http://www.named-data.net/techreport/TR001ndn-proj.pdf`, October 2010.