AIN: A Blueprint for an All-IP Data Center Network

Vasileios Pappas Hani Jamjoom Dan Williams
IBM T. J. Watson Research Center, Yorktown Heights, NY

Abstract

With both Ethernet and IP powering Data Center Networks (DCNs), one should wonder if their coexistence is necessary or an unquestioned legacy. Especially in cloud DCNs, the role of layer-2 is diminishing rapidly as the vast majority of applications only require layer-3 connectivity. At the same time, cloud workloads are demanding that DCN architectures better support network scalability, multitenancy, address virtualization, and end-host mobility. This paper argues that today's DCN architectures have a conjoined layer-2 and layer-3 design that is not only unnecessary, but is counter productive. We present *AIN*, a blueprint for scalable all-IP DCN. AIN implements virtual routers inside hypervisors, eliminating the need for virtual switching. It leverages the proven scalability of routing protocols and avoids unnecessary packet encapsulation, while supporting both multitenancy and end-host mobility. Finally, AIN is compatible with existing applications and is fully realizable with current protocols and hardware.

1 Introduction

Both Ethernet and IP were initially developed in the 1970s in two uncoordinated efforts. Ethernet was developed for networking of local area networks (LANs) with minimal configuration; IP was developed for internetworking of different LANs in a scalable and extensible manner. Today, both technologies are conjoined in state-of-the-art DCN deployments.

Recently, next-generation DCN architectures [9, 2, 4, 12, 8] focus on achieving (1) network scalability, (2) multitenancy, (3) address virtualization—MAC, IP, or both—and (4) end-host mobility. They typically achieve these goals by (a) encapsulating packets to mask the source—IP or MAC—address, (b) routing the encapsulated packets through optimized network fabrics that blend layer-2 and layer-3 features, and (c) minimizing unnecessary overheads (like ARP and broadcast) through a distributed directory service. While these new designs do improve DCN scalability, they come at the expense of increased design complexity.

Part of the DCN design complexity arises from the *blurring* of roles between Ethernet and IP, especially for the purposes of addressing, routing, mobility, and multitenancy. In this paper, we show that a simple and robust DCN can be achieved by cleanly separating the roles of Ethernet and IP. Ethernet is only used in point-to-point links, providing features like framing, synchronization, and medium access control. IP is used for addressing, routing, mobility, and multitenancy.

Following this principle, we present *All-IP Network (AIN)*, a blueprint for an all-IP DCN. AIN implements virtual IPv6 routers inside end-host hypervisors and eliminates the need for virtual switching altogether. AIN avoids unnecessary packet encapsulation by mapping addresses to a large address space, while supporting both multitenancy and end-host mobility. AIN provides the following benefits:

- *Scalability:* By using existing IP routing protocols, such as OSPF [7], IS-IS [1] or even I-BGP [10], DCNs can scale to the size of the whole Internet.
- *Mobility:* By leveraging IP-based mobility solutions (e.g., Proxy Mobile IP [3]) DCNs can support VM mobility at rates similar to that of LTE cellular networks.
- *Performance:* By using existing DCN hardware and with the use of IP equal-cost and multi-path routing [5], DCN forwarding performance can be better than of Ethernet.
- *Address Virtualization:* By exploiting the vast IPv6 address space, DCNs can provide true VM address virtualization for both IPv4 and IPv6 addresses.

We demonstrate that AIN can be implemented with existing hardware and protocols and is backward compatible with applications that require broadcast semantics.

2 AIN: All-IP Network

To demonstrate the feasibility of scalable all-IP DCNs, we present AIN, a fully realizable architecture using current hardware and existing network protocols. AIN is designed primary for multitenant data center environments, where each server hosts multiple virtual machines. For simplicity, we assume that hypervisors are installed on all servers.[1]

In AIN, hypervisors include virtual IPv6 routers that implement the following functionalities: *(a)* map, one-to-one, the VM network address space into the AIN network address space, *(b)* advertise the routes of the AIN network addresses space, i.e., the hypervisor prefixes, into the DCN, *(c)* implement a network-based mobility solution on behalf of VMs, and *(d)* support multitenancy. Next, we describe each of these functions in detail.

2.1 Addressing

AIN uses IPv6 exclusively for routing between hypervisors, while providing IPv4 and/or IPv6 addresses to hosted VMs. By utilizing the vast IPv6 address space, AIN allows hosting of multiple tenants within the same physical infrastructure while avoiding unnecessary packet encapsulation. This is achieved by managing two address spaces in tandem:

- *Host Address Space*: This is an address space, either IPv4 or IPv6, where VMs get their addresses from; it is visible only to the VMs and the hypervisors. The same host address space can be reused within the same DCN. In fact, multiple VMs, belonging to different or even the same tenant, can be assigned the same address, while avoiding address clashes.
- *Network Address Space*: This is an IPv6 address space that is visible only to the hypervisors and the network equipment that interconnects them. The network address space can be either private or public. AIN uses globally routable IPv6 addresses, enabling seamless VM mobility across data centers.

To enable address virtualization, AIN takes advantage of the interface identifier part of the IPv6 addresses, i.e., the last 64 of the 128 bit addresses. These bits can take any value, as long as the whole IPv6 address stays unique when combined with the first 64 bits of the addresses. With 2^{64} addresses in its disposal, AIN maintains a one-to-one mapping between the host and the network address spaces, allowing it to address a practically unlimited number of uniquely identified VMs.

A VM with a host address has a corresponding *home hypervisor*. The home hypervisor has ownership of the network address that maps to the VM host address. A VM can be hosted by any hypervisor, including its home hypervisor. In the case that a VM has multiple host addresses, it can potentially have multiple home hypervisors as well. When a packet, generated by a VM, is received by its hosting hypervisor, it gets its source and destination host addresses mapped to their respective network addresses. Then, based on the network addresses, the packet gets routed to the appropriate hypervisor or gateway, where it is mapped back to the original host addresses.

[1]The same architecture applies to data centers with non-virtualized servers by placing all network functionality in Top-of-Rack switches.

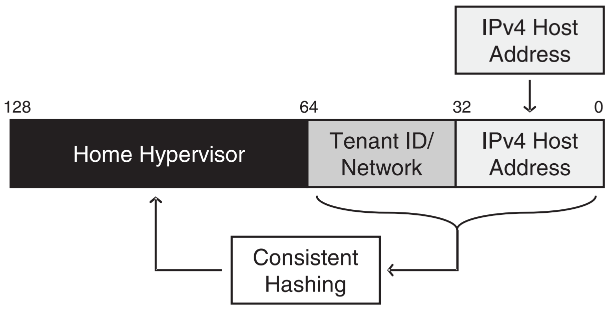

Figure 1: Host to Network Address Translation

Next, we describe the AIN mapping schemes for IPv4 and IPv6 host addresses. These schemes are implemented by the hypervisors and the Internet gateways, and do not rely on a directory service as in the case of [9, 2, 8].

IPv4 Address Virtualization. AIN enables hosting of any IPv4 address—private or public addresses—belonging to the cloud provider or to the tenants. As shown in Figure 1, all 32 bits of an IPv4 address are mapped, one-to-one, to the last 32 bits of the interface identifier of the IPv6 network address. The remaining 32 bits of the interface identifier are used for enabling multiple tenants to have the same IPv4 address (e.g., for reuse of private addresses). These 32 bits are divided in the following two tags: the tenant and the network tag. For example, the first 22 bits can be used to identify the tenant, while the remaining 10 bits can be used to identify isolated networks of the same tenant. The exact number of tenant and network tag bits can vary depending on hosting requirements. The remaining 64 bits of the network address come from the first 64 bits of the home hypervisor address.

A network address can be trivially mapped to a host address by extracting out the last 32 bits of the IPv6 address. The reverse mapping of a host address to a network address is more involved. It first requires constructing the interface identifier part of the IPv6 address, by combining the tenant tag, the network tag and the IPv4 address of the VM (information available at the hosting

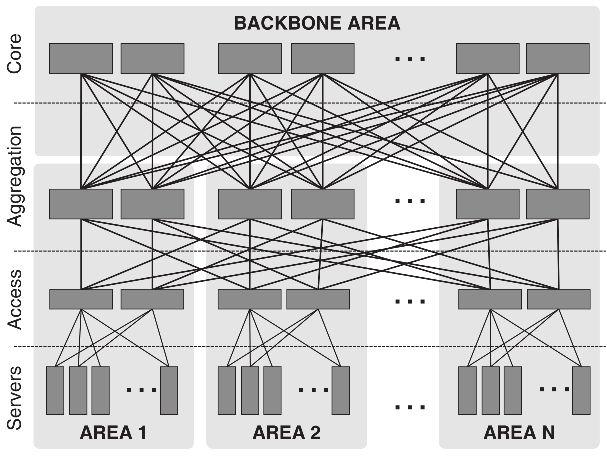

Figure 2: AIN Routing Architecture

hypervisor). Then, it requires constructing the first 64 bits of the IPv6 address, i.e., the bits that identify the home hypervisor of the VM. For this purpose, AIN uses a consistent one-way hash function that maps the 64 bits of the interface identifier to the first 64 bits of a hypervisor address (Figure 1). As such, the home hypervisor of a VM is pseudo-randomly chosen.

IPv6 Address Virtualization. AIN enables hosting of any IPv6 address as well. Depending on who owns the addresses, AIN handles the following two cases differently: *(a)* the address belongs to the DCN and is given out to the hosted VM, and *(d)* the address belongs to the tenant of the hosted VM. In the former case, a VM gets an address from the pool of available IPv6 addresses belonging to its home hypervisor; in the latter case, a VM gets an address that is directly given by its tenant (while routing is done by using mobile IPv6, as described in Section 2.3).

2.2 Routing

Figure 2 shows AIN's routing architecture. Following existing DCN terminology, the network is divided into three domains: the access, the aggregation, and the core. Servers connect to one or more access switches. Access switches connect to two or more aggregation switches, and aggregation switches connect to each other through the core switches. All switches and hypervisors run an instance of the OSPF routing process.[2]

To improve scalability, the DCN is divided into multiple OSPF *routing areas*. They consist of a *backbone area* that includes all core and aggregation switches. They also consists of multiple *point-of-delivery (PoD) areas* that include non-overlapping subsets of access switches

[2]We describe the whole architecture using OSPF as the routing protocol, but other ones, such as IS-IS or even I-BGP, can be used as well.

	Servers	Access	Aggregation	Core
Node Count	262144	4096	128	128
Port Count	2	132	256	128
Speed (Gbps)	10	10&100	100	100

Table 1: Data Center Network Size

and servers, as well as overlapping subsets of aggregation switches. Each PoD area maps to a distinct part of the data center physical infrastructure, e.g., a set of racks.

Following OSPF terminology, aggregation switches are area border routers (ABRs). They isolate different areas in terms of link state advertisements, while enabling routing across areas through the backbone. With each hypervisor originating one IPv6 prefix, all prefixes from the same PoD area get aggregated to a single IPv6 prefix by the aggregation switches. Consequentially, the number of prefixes in the backbone area is equal to the number of PoD areas.

Routing within a single PoD area happens through the access switches and possibly by using one aggregation switch. Routing across PoD areas involves the backbone area, and thus it always happens through one or more aggregation switches and possibly through one core switch. In essence, routing paths are directly mapped to the shortest paths of the physical topology.

Multipath Support. By implementing an all-IP network, AIN can take advantage of redundant end-to-end paths between hypervisors. Such paths are available, as OSPF supports equal cost multipath routing (ECMP). Then by using ECMP [5], traffic between any two VMs can be equally split across all available end-to-end paths. This enables load balancing of DCN links without requiring any traffic engineering. Due to the one-to-one mapping between host and network addresses, VM addresses are in effect directly exposed to ECMP, thus enabling load-balancing based on VM addresses rather than hypervisor addresses.

Broadcast Support. By default, and in order to suppress any broadcast traffic, IPv4 addresses come from /32 prefixes, i.e., they do not have a broadcast domain. Legacy applications that require IPv4 broadcast can request host addressees from prefixes larger than /32. We call these prefixes *broadcast-capable* prefixes. A broadcast-capable prefix requires an Ethernet broadcast domain to carry layer-2 broadcast traffic. The broadcast domain is 'emulated' by all hypervisors hosting VMs with IPv4 addresses belonging to that prefix. Each hypervisor intercepts all layer-2 broadcast traffic generated by any hosted VM and encapsulates it in IPv6 packets, which are then sent to all other hypervisors hosting VMs in the same broadcast domain. Depending on the size of the broadcast-capable prefixes, the Ethernet-in-IPv6 en-

Pod Areas	Servers	Access	Aggregation	Core
4096	521	521	17433	16912
1024	1316	1316	22176	16912
256	4496	4496	88848	16912
32	34176	34176	1085696	16912
0	1085696	1085696	1085696	1085696

Table 2: Link State Database Size

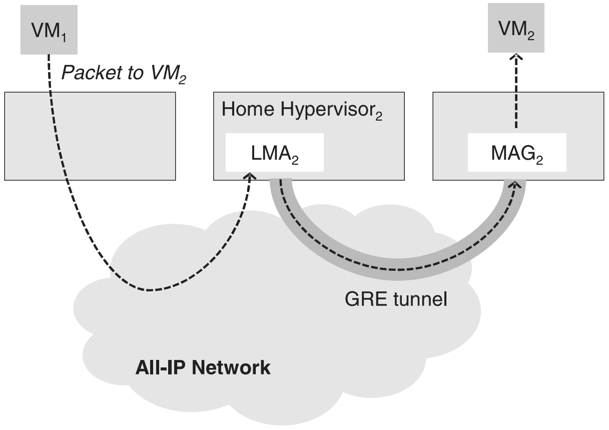

Figure 3: Proxy Mobile IP in AIN

capsulation is done using either unicast IPv6 packets, in which case broadcast packets are replicated at the source hypervisor, or using multicast IPv6 packets, in which case the replication happens in the network.

Link State Database Scalability. To understand AIN scalability, we consider the data center of Table 1, which provides full bi-sectional bandwidth. Table 2 shows the total number of OSPF link state database entries, for different number of PoD areas. Most of the entries on the core switches are due to the links between aggregation and core switches, while most of the entries in the aggregation switches are due to their participation in multiple PoD areas. Using 1024 or even as low as 256 PoD areas, the link state databases are reasonably sized, as they introduce at most 12 and 50 OSPF update messages per second, respectively (when using the default 30 minute OSPF update timer). The update rates at the servers and access switches are one order of magnitude lower. Overall, the OSPF routing overhead is at least one order of magnitude lower compared to an Ethernet-based DCN of the same size, when considering the overhead introduced by flooding of broadcast and unicast Ethernet frames (due to MAC forwarding entries expiring by default in 5 minutes).

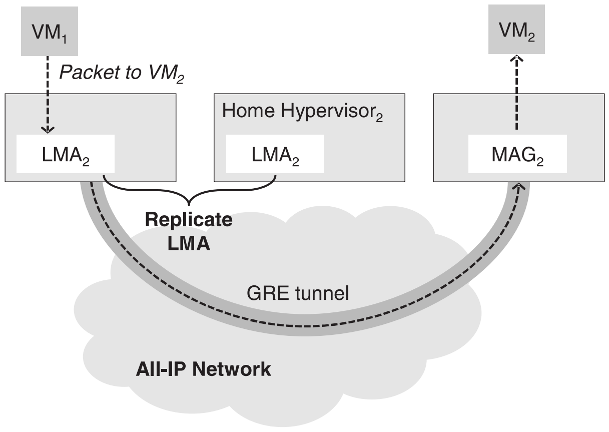

Figure 4: Distributed Proxy Mobile IP in AIN

2.3 Mobility

AIN provides VM mobility at the network layer to enable routing of VM addresses between any hosting hypervisor. More specifically, it adopts the mobility management solution used in LTE and 4G cellular networks, called Proxy Mobile IP (PMIP) [3]. PMIP enables end-host mobility without requiring any end-host modifications. Similarly, in AIN, VM mobility is enabled without any VM support, as all mobility operations are performed by the hypervisors and the core routers on behalf of the VMs. Furthermore, PMIP does not introduce any routing changes, as all traffic redirections are based on GRE tunneling.[3]

Distributed Proxy Mobile IP. As depicted in Figure 3, the home hypervisor of a VM becomes, using PMIP terminology, the Local Mobility Anchor (LMA). In other words, it is responsible for redirecting traffic that is destined to a mobile VM towards the hypervisor that hosts the VM. The latter hypervisor is called the Mobile Access Gateway (MAG). All VM traffic between the LMA and MAG hypervisors is carried using GRE tunneling. Whenever a VM moves to a new hypervisor, the LMA binding between the VM and its MAG hypervisor changes, which results in redirecting the traffic to the new hypervisor. Unfortunately, the PMIP redirection scheme requires all VM traffic to always flow through its home hypervisor, which results in non-direct paths between mobile VMs.

To address this issue, AIN uses a distributed version of PMIP,[4] shown in Figure 4. More specifically, the LMA of a mobile VM is replicated across all the hypervisors

[3]Note that multipath routing property is maintained even with mobility, as ECMP is aware of GRE protocol headers.

[4]Our version does not change the protocol between PMIP entities. It only affects the implementation of the MAG entity. In AIN, MAGs need to maintain, for each mobile, bindings with multiple LMA entities, instead of just one.

(or core routers) that communicate with the VM. The default LMA hypervisor of a VM is always its home hypervisor. If a second VM wants to communicate with the mobile VM, the hypervisor of the second VM also becomes the LMA of the mobile VM. This is done by first contacting the default LMA hypervisor which notifies the second hypervisor about the MAG hypervisor of the mobile VM. Consequentially, the MAG hypervisor establishes different GRE tunnels with all LMA hypervisors. This enables direct communication (i.e., one that does not involve home hypervisors) between any two VMs, even when both of them are mobile.

Mobility Across Data Centers. Using PMIP not only provides scalable mobility management within a data center, but it also enables VM mobility across data centers. With traffic redirection based on GRE tunnels, a host address, either IPv4 or IPv6, can be hosted by any hypervisor in any data center, as long as the data centers have network addresses that are globally routable.

2.4 Multitenancy

AIN uses IP layer firewall rules, enforced by the hypervisors, to prevent unauthorized direct communication between VMs. The default firewall policy is to drop all incoming and outgoing traffic. When a hypervisor hosts a new VM, it adds a set of accept rules into its firewall. The first rule allows only outgoing traffic with a source address belonging to the hosted VM, preventing IP spoofing attacks. The rest of the outgoing rules specify the VMs and the Internet gateways which the new VM is allowed to communicate with. Incoming rules are also enforced to block any traffic generated outside of the DCN.

To improve scalability, the internal traffic rules are expressed based on the tenant and/or network tags, rather than using VM IP addresses. Depending on whether a host address (IPv4 or IPv6) is public or private, there are two different default firewall rules. For private addresses, the default rule is drop, i.e., any traffic between two private addresses is dropped unless both of them belong to the same tenant network. For public address, the default rule is accept, i.e., any traffic between public addresses is allowed, independently of their tenant and network. Tenants can add new rules to enable additional traffic between private addresses or to disable specific traffic between public addresses.

3 Related Work

TRILL [12] provides an alternative mechanism for Ethernet forwarding, that utilizes link state routing to enable layer-3 style forwarding in layer-2 topologies with redundant links. AIN shares the same goal, but it achieves that at layer-3. Unlike TRILL, AIN makes use of network-based IP mobility solutions, enabling VM mobility in large scale data centers, as well as across data centers.

PortLand [9], BCUBE [4] and VL2 [2] describe new DCN architectures, which, among other things, provide scalable layer-2 fabrics, enabling VM mobility anywhere in the data center. AIN shares some similarities with VL2, as both make use of IP forwarding and equal-cost multipath routing in the DCN core. Though, like PortLand and BCUBE, VL2 is a layer-2 architecture, given that it encapsulates all Ethernet traffic, both unicast and broadcast, over the IP network. In contrast, AIN is a layer-3 architecture, as its all-IP network is extended all the way to the hypervisors.

VXLAN [6] (and similarly NVGRE [11]) provides the means of encapsulating Ethernet traffic over an IP network. Consequentially, it shares many similarities with VL2, with the primary difference being that VXLAN is fully implementable by the hypervisors and does not require any support from network equipment. AIN uses the VXLAN Ethernet-in-IP encapsulation concept to support broadcast addresses for legacy applications. While in AIN all other traffic is IP traffic, any traffic between different VXLAN domains is encapsulated over IP.

4 Conclusion

AIN is a new DCN architecture that shifts the emphasis of scalable network fabric from layer-2 to layer-3. AIN implements an all-IP network that leverages IP's core strengths all the way to host servers, such as equal-cost multipath forwarding, scalable routing, network-based mobility management, etc. AIN handles broadcast addresses as an exception, by emulating Ethernet broadcast semantics over IP only when it is explicitly requested. AIN has a number of advantages compared to other DCN architectures. By dropping all layer-2 semantics, it enables VM mobility that scales across multiple data centers. It also simplifies DCN management by reducing unnecessary complexities due to layer-2 network configurations.

References

[1] ISO/IEC 10589:2002 Second Edition.

[2] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. In *Proc. of ACM SIGCOMM*, Spain, Aug. 2009.

[3] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy Mobile IPv6. RFC 5213, Aug. 2008.

[4] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A High Performance,

Server-centric Network Architecture for Modular Data Centers. In *Proc. of ACM SIGCOMM*, Spain, Aug. 2009.

[5] C. Hopps. Analysis of an Equal-Cost Multi-Path Algorithm. RFC 2992, Nov. 2000.

[6] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. `http://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-00`, Aug. 2011.

[7] J. Moy. OSPF Version 2. RFC 2328, Apr. 1998.

[8] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary. NetLord: A Scalable Multi-Tenant Network Architecture for Virtualized Datacenters. In *Proc. of ACM SIGCOMM*, Toronto, Canada, Aug. 2011.

[9] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In *Proc. of ACM SIGCOMM*, Aug. 2008.

[10] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4. RFC 4271, Jan. 2006.

[11] M. Sridharan, K. Duda, I. Ganga, A. Greenberg, G. Lin, M. Pearson, P. Thaler, C. Tumuluri, N. Venkataramiah, and Y. Wang. NVGRE: Network Virtualization using Generic Routing Encapsulation. `http://tools.ietf.org/html/-draft-sridharan-virtualization-nvgre-00`, Sept. 2011.

[12] J. Touch and R. Perlman. Transparent Interconnection of Lots of Links (TRILL). RFC 5556, May 2009.