

Network Scalability in the Data Centre

CPGS Report

Malcolm Scott

Contents

1	Introduction	1
1.1	Data Centre Network Design Challenges	1
1.2	Scalability Limitations of Ethernet	4
1.3	Related Work	6
2	Scalable Ethernet Addressing: MOOSE	8
2.1	Hierarchical Addressing	8
2.2	MOOSE Architecture	9
2.3	Interoperability Considerations	15
2.4	Implementation and Evaluation	17
2.5	Ongoing Work	17
3	Broadcast Traffic Optimisation	19
3.1	The Broadcast Problem	19
3.2	Address Directory Service: ELK	20
3.3	IPv6 Neighbour Discovery	22
3.4	Ongoing Work	24
4	Layer-3 Virtual Machine Migration	26
4.1	Transparent IPv6 Address Persistence	27
5	Conclusion and Thesis Plan	30
5.1	Thesis Structure	30
5.2	Timeline	31
	Bibliography	32

List of Figures

1.1	Effect of RSTP on a 2D mesh topology	5
2.1	Assignment of MOOSE addresses by switches	10
2.2	Sequence diagram	12
2.3	Mobility options	14
3.1	Broadcast traffic breakdown by protocol	21
3.2	Rate of ARP messages on a test network with and without ELK	23
4.1	Virtual network topology	28

CHAPTER 1

Introduction

My research is on improving the scalability of data centre networks, in a manner which could realistically be deployed in the near future. In order to explain the motivation for this work, I will describe in this chapter the current situation in data centre networking, and why scalability issues are a real and current concern. The outcomes of my research will, I hope, also apply to other scenarios.

1.1 Data Centre Network Design Challenges

Traditionally, data centres have contained disparate clusters of servers dedicated to specific applications which operate independently. The servers purchased for a particular application would largely communicate either within the cluster or with clients in the outside world—not with other server clusters. The network infrastructure was (and still is) almost invariably Ethernet and IPv4 due to their ubiquity elsewhere, but network scalability was rarely an issue as the network could be subdivided arbitrarily. In any case the entire data centre was not likely to be excessively large. The network would generally form a simple tree topology comprising top-of-rack switches, a switched or routed aggregation layer and a routed core; so long as the servers of each cluster were in the same part of the tree and within the same IP subnet, internal and external communication would be reasonably efficient.

This no longer applies to modern data centres. Three main factors have changed.

1.1.1 Virtualisation

From the perspective of the network, a virtual machine is no different to a physical one; each virtual machine is addressable separately via its own Ethernet and IP addresses [Barham et al., 2003]. However, virtualisation introduces a layer of abstraction between physical server and operating system which allows any virtual server to run on any

physical machine, often with the ability to migrate between physical machines whilst running [Clark et al., 2005]. Live migration is beneficial as it allows for more resilience and flexibility; physical servers can be added in order to increase capacity or removed for maintenance at any time without affecting application availability.

Supporting live migration is a significant challenge for network designers. The premise behind current virtual machine migration systems is that the process must be entirely transparent to the guest operating system and application; applications keep running and are not even aware that they have been migrated, and any TCP connections to clients or other servers remain open. In order for TCP connections to survive the migration, the virtual machine’s IP address must not change and therefore—assuming the nonexistence of mitigating technologies such as IP Mobility [Perkins, 2002], which is not widely implemented—live migration can only occur within a broadcast domain and a virtual machine cannot cross any routers.

Consequently, if live migration is to be supported across the entire data centre—or indeed between data centres—the network must be a single large Ethernet. IP routers cannot be used to subdivide the network, and Ethernet is required to scale on its own. As I will describe in Section 1.2, Ethernet is already unable to scale to this degree.

A common workaround is to constrain live migration to certain subsets of the physical servers by dividing the data centre’s servers into pools, each on a separate Ethernet broadcast domain. This provides some of the benefits of live migration in that individual machines can still be taken offline assuming there is spare capacity in every pool, but in some situations this will hinder rather than help: if all servers in a particular region of the data centre have to be shut down, for example due to network or power maintenance, this may leave one pool with insufficient physical servers to run all its virtual machines. Amazon EC2 does not use live migration at all; this decision was likely motivated at least in part by the network engineering challenges. Cold migration does occur in EC2—when a virtual machine instance is booted, it will be allocated to a physical server according to available capacity—but even then the network is subdivided into “availability zones” between which instances will never move.

1.1.2 Larger Data Centres

With the rising popularity of cloud computing, major service providers are building ever-larger data centres. The details are generally considered commercially-sensitive, but the occasional snippet of information hints at the scale involved. Microsoft, for example, opened a high-density data centre in Chicago in 2009 which can hold 300,000 servers [Chrapaty, 2008]; they also run or are building slightly-smaller facilities in Dublin, Texas and Washington. In 2008 their infrastructure was expanding by 10,000 servers every month. These numbers represent *physical* servers; virtualisation adds another order of magnitude of addresses as discussed above.

In such an environment, supporting live migration within and between data centres would require a single broadcast-domain to be able to handle millions of virtual machines across several hundred thousand physical servers in geographically-diverse locations.

A further case study requiring live migration between several interconnected data centres is provided by the Intelligent Energy-aware Networks (“INTERNET”) project, alongside which I am working. One of the aims of the INTERNET project is to develop the ability to move computational loads near to sources of renewable power in order to minimise electrical transmission losses. Data centres would be built near to, for example, wind farms around the country—but since different wind farms generate power at different times according to weather patterns, entire data centres would be turned on and off automatically with virtual servers migrated accordingly between data centres located hundreds or perhaps thousands of miles apart.

1.1.3 Resilient Topologies

Traditional data centre network architectures are based on tree topologies. Each server rack, containing on the order of 20 to 40 servers, has a top-of-rack Ethernet switch with a single connection to each server. (With virtualisation, the server may logically contain several hosts, which is achieved by the means of a software Ethernet switch running in the virtualisation layer.) Connections from around 20–40 top-of-rack switches are aggregated back to a single end-of-row switch. The end-of-row switch is in turn connected into a core network; in a small data centre the core may simply be a single large IP router. As live virtual machine migration becomes a necessity, this router will have to be replaced by a switch.

Ethernet switching works best on a tree topology, and so this arrangement has worked well in the past. In larger networks, however, trees start to have significant problems. For instance, as soon as three or more geographically-diverse data centres are interconnected on a single Ethernet network, direct links between data centres become desirable in order to minimise multi-hop paths across long-distance links. Unless a single data centre acts as the hub through which traffic between any other pair of data centres passes, the overall topology of the multi-data-centre network can no longer be a tree.

A tree network also offers no resilience; if a single component or link fails, everything downstream of the failure will lose connectivity. Resilience is generally added by duplicating components: by deploying two end-of-row switches per row of racks and supplying each top-of-rack switch with an uplink to each, alternate paths exist if anything upstream of the top-of-rack switch fails. Such a topology is termed a *fat tree*. This is just the tip of the iceberg in improved topologies; Abu-Libdeh et al. [2010] propose a 3D torus, CamCube, which works well where there is a high degree of inter-server

communication. Unfortunately, Ethernet does not make efficient use of any resilient topology.

1.2 Scalability Limitations of Ethernet¹

Ethernet has lasted well since its inception in the 1970s [Metcalfe and Boggs, 1976] with Ethernet frame-structure and addressing remaining ubiquitous in the data centre environment as in many others. However, Ethernet exhibits scalability issues when used to build broadcast domains of more than a few thousand devices, such as costly and energy-dense address table logic and storms of broadcast traffic.

The traditional method of avoiding such problems is the artificial subdivision of a network, but this introduces an administrative burden, requires significant routing equipment and with current protocols also precludes live migration.

1.2.1 Forwarding Databases

Ethernet's scalability is limited primarily by the forwarding database that every switch in an Ethernet network must maintain [IEEE Computer Society, 2004a, §7.8–7.9]. A switch's forwarding database contains one entry per source address seen in any frame passing through that switch, and stores that MAC address together with the learnt location of that address—the port on which packets from that address were last seen. This is later used to determine on which port to transmit frames destined for that address. Devices frequently broadcast frames throughout the network (for example ARP queries) so active devices on the network are listed in most switches' forwarding databases most of the time.

In modern switches the capacity of this database is generally of the order of 16,000–64,000 entries. Higher-capacity forwarding databases exist but are generally constrained to very high-end, power-hungry and expensive switches or to low-speed switches such as those implemented in software. On a moderately large network, full databases are a serious risk. If the database becomes full, entries will be discarded; frames for unknown addresses are flooded to all ports and the resulting traffic storm could cause major problems, especially in the presence of low-capacity edge links which could become saturated with other hosts' traffic.

Traditionally the forwarding database has been stored in a content-addressable memory (CAM) as lookups must be very fast, particularly as 10 Gbit/s Ethernet becomes ubiquitous. As networks grow, the number of entries in a switch's forwarding database must naturally increase; however, increasing the capacity of CAMs without

¹This section and the next are adapted from corresponding sections of a previous workshop paper [Scott et al., 2009].

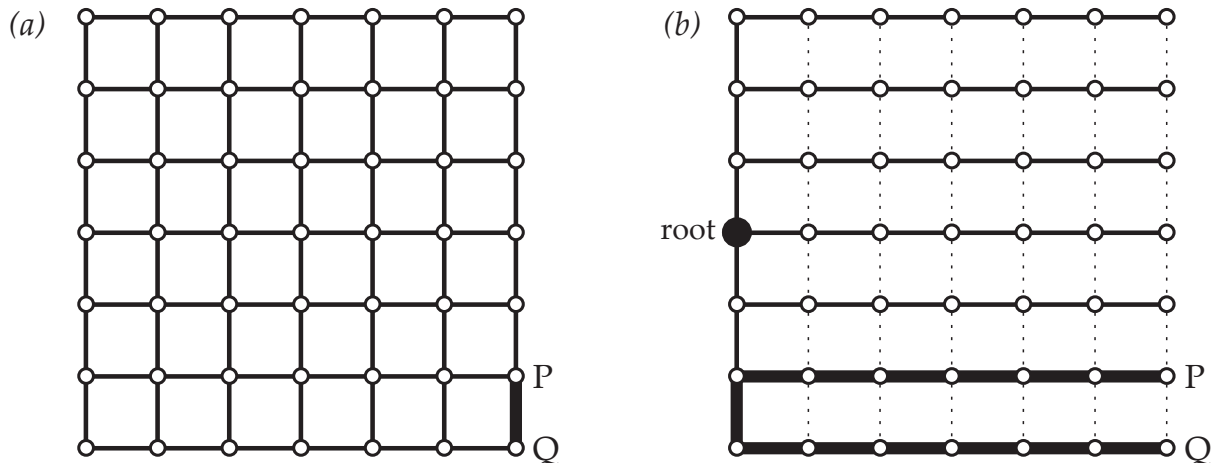


Figure 1.1: A 2D mesh topology, (a) before and (b) after RSTP has removed loops, highlighting the path taken by frames transmitted between two adjacent nodes P and Q

sacrificing speed whilst constraining energy consumption is proving to be challenging [Yu et al., 2005; Pagiamtzis and Sheikholeslami, 2006]. Cheaper switches use DRAM in place of a CAM, but this is likely to remain slower especially for large tables.

1.2.2 RSTP

Ethernet's inability to handle networks containing loops also presents a significant scalability problem. In the presence of loops, frames are forwarded around the loop indefinitely due to the lack of a hop counter; in the presence of broadcast frames this will result in a *broadcast storm* in which all capacity on all links will be consumed by an infinite stream of duplicate frames. The Rapid Spanning Tree Protocol, RSTP [IEEE Computer Society, 2004a, §17], exists to ensure that any loops are broken by disabling any redundant links, which means that any multipath capability of the topology cannot be exploited. Furthermore, on a dense mesh network RSTP must disable a large proportion of links; this considerably lengthens paths across such topologies and may introduce a bottleneck at the root of the spanning tree (see Figure 1.1). In a data centre environment, this potentially amounts to a large proportion of capacity being wasted wherever redundant links are installed, which may be expensive long-haul fibres.

1.2.3 Broadcast

The original Ethernet was a shared-medium network, where every frame was broadcast and no switching took place. Modern-day wired Ethernet-based networks instead consist almost entirely of point-to-point links; as a result of this, the distinction between unicast, broadcast and multicast has become more important.

Not only does Ethernet flood frames destined for unknown hosts, but it also uses—and encourages higher-layer protocols to use—broadcast for control messages. In particular, ARP [Plummer, 1982] performs address resolution via broadcast queries and advertisements, and DHCP [Droms, 1997] uses broadcast messages for automatic configuration. It is impractical to replace these protocols entirely as this would require software upgrades to every device, but it would be desirable for the network to minimise the forwarding of broadcast traffic to unnecessary destinations.

1.3 Related Work

It is well-known that traditional Ethernet scales poorly, and there have been various attempts in recent years to rectify this. The most widely-used of these in real-world networks is MPLS-VPLS (Multiprotocol Label Switching—Virtual Private LAN Service) [Rosen et al., 2001]. This connects Ethernet islands together through tunnels across a MPLS cloud. MPLS works by adding one or more labels to the start of every frame, i.e. encapsulating the frame inside its own protocol.

In MPLS-VPLS, the label edge routers (LERs) must determine the frame’s initial label(s) based upon the destination address via a lookup table. Frames follow prenegotiated label-switched paths (LSPs) that, unlike Ethernet, are not constrained to follow a spanning tree; LSPs are precomputed at connection setup time and the relevant next hop is stored in a lookup table on each intermediate switch. Each switch must hence use each frame’s label to index into this lookup table to determine how to switch the frame.

The effect, once the connection has been negotiated, is to provide what appears to be one or more large Ethernet networks, transparently overlaid on the MPLS cloud. Whilst this solves effectively the problem of shortest-path routing across the MPLS cloud, the overlay Ethernets are still susceptible to the usual scalability problems—and in fact VPLS adds further large lookup tables on every switch that can in some configurations scale even worse than Ethernet’s forwarding databases. LERs must map every MAC address to a LSP; label switch routers (LSRs) must store the next hop for every LSP in which they participate, which in the core of the network could scale as $O(\text{hosts}^2)$.

A similar scheme is proposed by Hadžić [2001], with the difference that Ethernet-inside-Ethernet encapsulation is used rather than a new protocol. This has the advantage that less processing is required on intermediate switches in the backbone network. However, routes across the backbone are constrained to a spanning tree, and encapsulating switches must obtain a new destination address for every frame using a lookup table that—like Ethernet’s forwarding database—must contain every transmitting MAC address. Due to its heavy basis on Ethernet, this shares many of Ethernet’s

scalability problems.

SmartBridge [Rodeheffer et al., 2000] and Rbridges [Perlman, 2004] both encapsulate Ethernet frames in a new inter-switch protocol, and run a link-state routing protocol between switches. The link state graph includes the location of every MAC address—necessary because the address space remains flat and any address could appear anywhere—i.e. it again contains every host. Furthermore, switches must perform expensive computation to update routing tables whenever a MAC address joins or leaves the network.

Myers et al. [2004] suggest that Ethernet’s main failing is its broadcast service, and propose a new architecture in which hosts make explicit use of directory services operated by switches rather than broadcasting queries. It is clear that switches’ participation is necessary in order to deal with the broadcast problem; however the suggested modifications to Ethernet are not backwards-compatible and would require at least software modifications to all connected devices. Ethernet is, perhaps unfortunately, too widespread for this to be practical; transparent interception of broadcast frames and subsequent local handling or redirection via multicast or unicast remains the only practical solution.

SEATTLE [Kim et al., 2008] takes a more scalable approach. A routing protocol is operated between switches, but in contrast to the approaches described above, the routing protocol only propagates switch location information, rather than every MAC address on the network. Flat MAC addresses are still used, and thus a mechanism is required to look up the switch to which a given address is connected. This is achieved by using a distributed hash table (DHT) operating on participating switches with local caching to alleviate load. This is certainly a step in the right direction but introduces considerable complexity to switches, since they now must maintain and update the DHT continually, and it is clear that a SEATTLE switch would have a significant software component in the data path.

1.3.1 Internet Engineering Task Force ARMD Working Group

The IETF has recently formed a working group, entitled “Address Resolution for Massive numbers of hosts in the Data center” (ARMD), to investigate “the impact of changing workloads and existing protocols on datacenter network performance” [Dunbar and Schliesser, 2011]. In the formative stages of this working group, the chairs expressed interest in my contributing to their work; I believe that working with a standards body is worthwhile and I intend to pursue this. Unfortunately, due to internal disagreement within the IETF, the ARMD working group has been directed not to investigate potential protocol enhancements until at least mid-2012.

Nevertheless, during its first year, ARMD will work with data centre operators and may produce useful data to quantify the problem.

Scalable Ethernet Addressing: MOOSE

MOOSE—Multi-level Origin-Organised Scalable Ethernet—is my enhanced switching and addressing architecture for Ethernet. I originally developed it during my time as a research assistant on the Intelligent Network Airport (TINA) project; the problems encountered in the data centre environment are remarkably similar and MOOSE is expected to apply equally well here. I have continued to develop MOOSE over the past year in this new context, and it will form a part of my thesis.

This work was originally described in a workshop paper [Scott et al., 2009].

2.1 Hierarchical Addressing

Ethernet’s poor scalability arises in various guises, as discussed in Section 1.2. It would seem at first glance that these are entirely distinct and unrelated. However, there is a common underlying cause: that *MAC addresses provide no location information*.

Globally-unique MAC addresses are structured such that the first three bytes of a device’s address contain an organisationally unique identifier (OUI) allocated to the device’s manufacturer by the IEEE, with the remaining three bytes allocated by the manufacturer. This hierarchy exists solely for the purpose of allocating unique addresses in a decentralised fashion, and is of no use to Ethernet switches, which must treat the unicast address space as flat.

A flat address space has the advantage that no configuration of devices is required; a device can use its unique, manufacturer-assigned MAC address anywhere on any network. However, this leaves each switch with the task of discovering and storing the location of every addressable device individually.

If the MAC address space were not flat, but instead contained enough information to locate the owner of an address, two major advantages would be gained. Firstly, large forwarding databases would no longer have to be maintained on every switch.

Address location data could instead be distributed across the network so that frames are directed towards their destinations according to successive stages of a hierarchy.

Secondly, a hierarchical MAC address space would also make the addition of shortest-path routing considerably easier. Flat addressing does not lend itself to easy routing: any address can be situated anywhere on the network, which would necessitate advertising every host's MAC address via the routing protocol or location service; this scales very poorly. The use of hierarchical addresses, with each switch handling a block of sequential addresses akin to an IP subnet, would reduce the routing problem to one which is already solved by existing routing protocols.

The facility for network administrators to assign locally administered addresses (LAAs) to devices has existed for as long as Ethernet. However, configuring and maintaining the LAA on every device based upon where they are connected would be a considerable and unwelcome administrative overhead. I am therefore developing MOOSE, a system for applying hierarchical addressing to an Ethernet transparently and without any configuration to edge devices.

2.2 MOOSE Architecture

The basic operation of MOOSE is to assign a hierarchical MAC address to each host on the network, allocated dynamically and automatically from the unicast LAA space. This address is referred to as a *MOOSE address* to avoid confusion with hosts' existing, static, manufacturer-assigned MAC addresses.

Every frame entering the network has its source address rewritten in-place to the sending host's MOOSE address by the first MOOSE-aware switch it traverses. The switch that performs address rewriting for a host—i.e. the closest MOOSE switch to that host—is the host's *home switch* and is responsible for assigning a MOOSE address to that host. (If non-MOOSE switches or hubs are in use, a host may have more than one "closest" MOOSE switch simultaneously, in which case an election protocol must be used to select a home switch for each edge segment.)

The destination address is left intact in the expectation that it already is a MOOSE address. Hosts' ARP caches will already contain the MOOSE addresses of any hosts being communicated with as any packet received will already have had its source address rewritten; a host's manufacturer-assigned MAC address is never seen outside of the segment containing that host. This is a crucial point since encapsulation-based technologies such as MPLS do not reveal to the destination host the address used for routing; as a result, switches must also convert destination as well as source addresses of frames entering the network. In other words, once again switches must maintain large tables of remote hosts on the network. The only destination rewriting that MOOSE switches perform is of frames destined for local hosts, which must be ad-

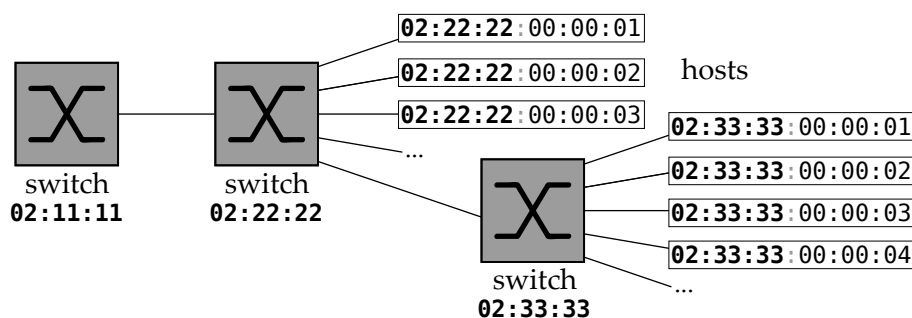


Figure 2.1: Assignment of MOOSE addresses by switches

dressed to the host's manufacturer-assigned MAC addresses in order to pass the NIC's MAC address filter; this is simple as the required information is already known by that switch.

A MOOSE address consists of a *switch identifier* followed by a *host identifier*. The former indicates the location of this address on the network, as illustrated in Figure 2.1. Since these two identifiers when concatenated must form a unicast LAA, the settings of two bits in the first byte of the switch identifier are fixed: the least significant bit must be 0 to indicate a unicast address, and the second-least significant bit must be 1 to indicate a LAA. It is hence possible to route frames through the network to remote hosts by simply inspecting the switch identifier in the destination address, and ignoring the host identifier until the frame reaches the destination host's home switch. Switches no longer need to keep a table of all MAC addresses; they only need store the locations of other switches and of any directly-connected hosts.

In the simple case, the switch identifier and host identifier can both be three bytes in length. However, as part of his undergraduate project to implement MOOSE, Wagner-Hall [2010] proposed an extension which would allow switches to automatically acquire unique, variable-length switch identifiers using a class-based scheme in which the first three bits of the address indicate how many of the following 5-bit blocks make up the switch prefix. Depending on requirements, the switch identifier may itself be a hierarchical address—for example six bits to identify a network area followed by two bytes to identify a switch within that area—which could then be used to aid routing decisions.

Each host is assigned a host identifier by its home switch from the pool of identifiers available to that switch. Only a host's home switch ever bases a switching decision on the host identifier, so the detail of how these are allocated can vary from switch to switch. Suitable schemes include:

- sequential assignment;
- the port number followed by a sequential portion;

- a hash of the host's real MAC address.

The latter two approaches are preferable to a simple sequential assignment, as they better isolate certain kinds of denial-of-service attack in which a malicious host attempts to use up all available host identifiers on the switch. They also require less state to be shared between ports.

As well as reducing the amount of data that must be consulted in order to make switching decisions, MOOSE provides extra resilience by making this much more predictable. The number of MAC addresses in a network can increase unexpectedly in the event of an address flooding attack [Sipes, 2000] or even under normal operation in some cases such as an open wireless network; relying on the MAC address database for forwarding leads to some of the vulnerabilities of Ethernet. The set of switch identifiers participating in MOOSE switching, on the other hand, can be kept predictable and manageable by ensuring that neighbouring switches are authenticated. This authentication could be achieved at layer 3 using the security features found in most popular routing protocols or at layer 2 using 802.1X [IEEE Computer Society, 2004b]. As the switch identifier is the only address consulted for forwarding decisions, a MOOSE switch is likely to remain reliable in the face of attacks that could have brought down a traditional Ethernet. Furthermore, attacks based upon MAC address spoofing cannot function on a MOOSE network as the user-provided MAC address is translated immediately.

2.2.1 Shortest Path Routing

As described so far, MOOSE switches must still forward frames along a spanning tree. The foundations are in place to do much better than this using shortest-path routing.

For the purpose of frame forwarding, a MOOSE switch can be considered akin to a layer 3 router; it has one locally-connected subnet—containing all addresses starting with its switch identifier—and delivers frames to other subnets by passing them to an appropriate neighbouring switch. Bearing this in mind, the switch can run a routing protocol of the kind normally used for IP, such as a variant of OSPF [Moy, 1998]. This allows frames to be routed along the shortest available path, rather than being constrained to a spanning tree. OSPF-OMP [Villamizar, 1999] may be particularly desirable due to its ability to make use of multiple equal-cost routing paths in order to improve performance [Schneider and Nemeth, 2002].

2.2.2 Broadcast

Since Ethernet does still need to support arbitrary existing protocols, broadcast frames must still be forwarded along a spanning tree in order that they reach each host without

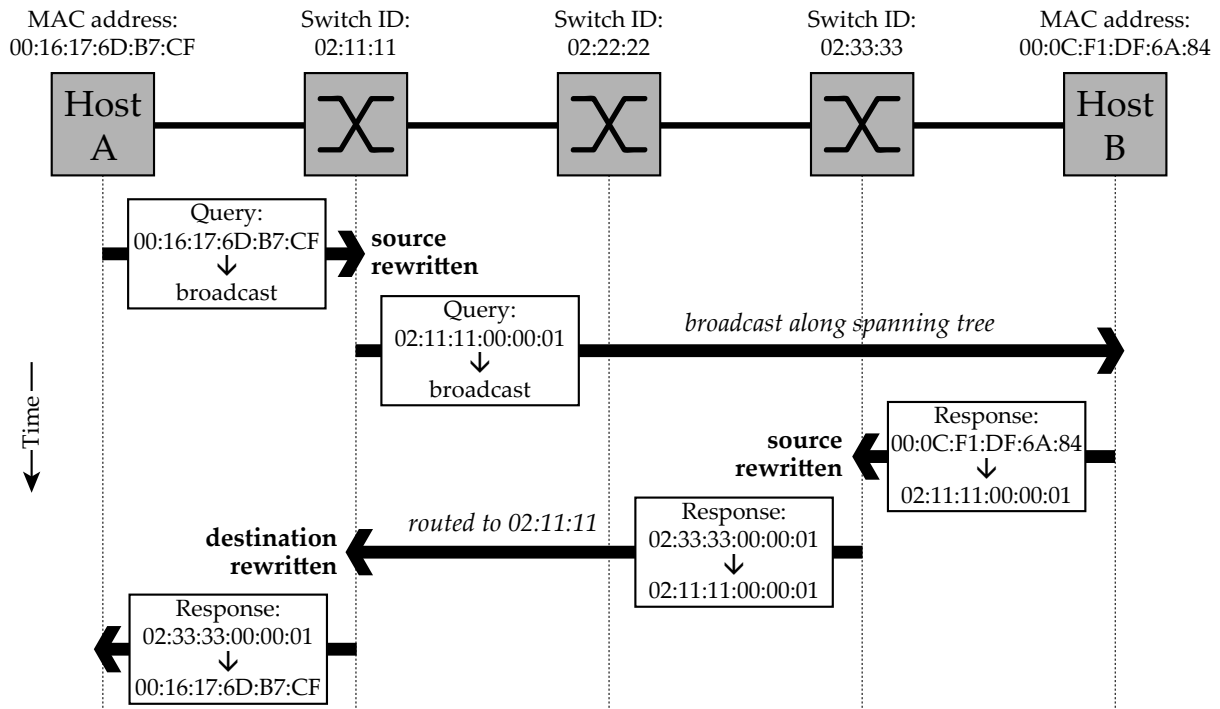


Figure 2.2: Sequence diagram of a broadcast query and subsequent unicast response

causing a broadcast storm. An explicit spanning tree protocol such as RSTP is not required however, as the tree can be deduced from the routing information.

In order to avoid the overhead of computing a spanning tree on every switch, this could use reverse path forwarding in a similar manner to Protocol-Independent Multicast [Adams et al., 2005]. In his evaluation of MOOSE, Whitehouse [2011] concluded that this results in considerably increased broadcast traffic compared with RSTP since rather than reaching every switch exactly once, broadcast frames traverse every link with duplicate copies discarded by the receiver. Instead, then, an explicit spanning tree will be used but this can be computed from the routing protocol's link state graph so long as the root is deterministically chosen.

2.2.3 Example

To illustrate the basic behaviour of MOOSE switches, I will describe the steps involved in forwarding a broadcast frame containing a query in some higher-layer IPv4-based protocol, and subsequent unicast frame containing the response, between two hosts A and B via three MOOSE switches 02:11:11, 02:22:22 and 02:33:33; see Figure 2.2.

Query

1. Host A transmits the broadcast query frame as it would on any Ethernet network, with its own manufacturer-assigned MAC address in the Ethernet header's source field and the broadcast address (FF:FF:FF:FF:FF:FF) as the destination.
2. The frame is received by switch 02:11:11, which observes the non-MOOSE address in the frame's source field, and rewrites the source field into a MOOSE address containing the switch identifier and the appropriate host identifier. As this is Host A's first frame, the switch must allocate a host identifier (in this case 00:00:01, making Host A's complete MOOSE address 02:11:11:00:00:01).
3. The three switches broadcast the frame using reverse path forwarding away from Host A.
4. The frame is received by Host B (and any other hosts on the network) in its current form; no further rewriting is performed.

Response

1. Host B looks up Host A's IP address in its ARP cache to determine a suitable destination address for the response frame. Since the rewritten query frame arrived at Host B with the source field containing the MOOSE address 02:11:11:00:00:01, this is the address returned by the cache lookup.
2. As above, switch 02:33:33 assigns a MOOSE address to Host B (02:33:33:00:00:01) and rewrites the source address of the frame.
3. The frame is now routed through the network based solely on the destination switch identifier—the host identifier is ignored for now. The routing table is consulted for the location of switch 02:11:11 and the frame is forwarded accordingly.
4. On receiving the frame, switch 02:11:11 observes that it is destined for a directly-connected host (02:11:11:00:00:01). It prepares the frame for transmission along its final hop by rewriting the destination address to Host A's manufacturer-assigned MAC address. The source field of the frame is again left as the MOOSE address of Host B in order that this address is used for any further communication with Host B.

2.2.4 Mobility

A consequence of introducing location-based hierarchy into MAC addresses is the need to explicitly handle host mobility. In a traditional Ethernet, hosts can migrate

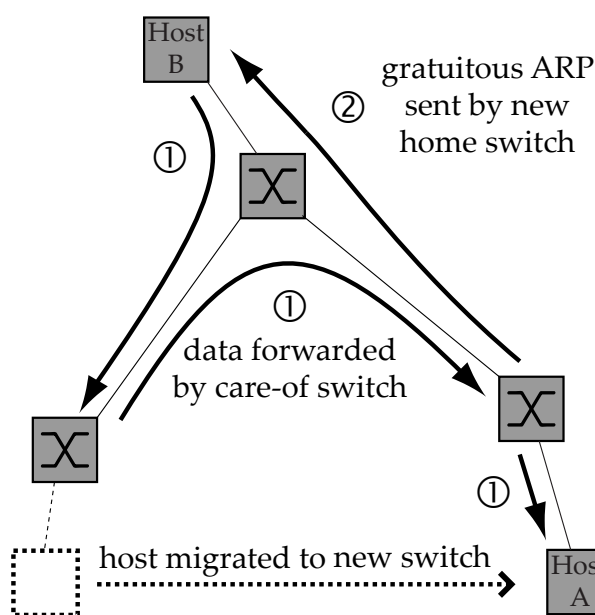


Figure 2.3: Mobility options: two ways to handle a host A roaming onto another switch whilst maintaining communication with another host B

between switches as the switches will learn the host’s new location as soon as it sends a frame. With MOOSE, if a host relocates to a new switch its address changes and any ARP cache entries on other hosts pertaining to the migrated host become incorrect; frames will continue to be sent to the host’s old location for a while. There are two strategies for dealing with this, as illustrated in Figure 2.3, which can be used separately or in conjunction:

1. The previous home switch of the migrated host can forward frames sent to the host’s old address until outdated ARP cache entries expire. This is similar to IP Mobility [Perkins, 2002]: the previous home switch essentially becomes a care-of agent for the host. However, unlike IP Mobility, it requires no host support. A handover protocol is necessary for the old and new home switches to set up such forwarding: on the arrival of a new host at a switch, that switch would ask all other switches (via multicast) whether any had seen this host before, identifying it using its manufacturer-assigned MAC address, and would instruct such switches to redirect frames.
2. A broadcast ARP announcement (or “gratuitous ARP”) can be sent by the new home switch to immediately update remote ARP caches (and the ELK directory—see Section 3.2) with the new MOOSE address. This is the technique used by Xen when migrating live virtual machines [Clark et al., 2005]. Unlike the previous approach, this works even if the previous switch is no longer reachable, for example

if this host migration was as a result of a switch failure. This is a simpler approach as a handover protocol is not required, but results in additional broadcast traffic. It also introduces a requirement for switches to violate layer boundaries and track IP addresses, but many commercial switches already have this capability.

Unless the frequency of host migrations is very high, the additional load introduced by either mobility approach is expected to be negligible.

2.3 Interoperability Considerations

2.3.1 Layer-violating Protocols

In an ideal world, free from layering violations, all layer 3 protocols would operate correctly on top of MOOSE as on Ethernet with no higher-layer rewriting necessary in the switch. In reality, however, protocols abound which use hosts' MAC addresses for purposes other than layer 2 addressing: the MAC address serves as a convenient unique host identifier in protocols such as DHCP. ARP, the glue between IP and MAC addresses, must naturally be handled specifically—especially since it places MAC addresses in the frame payload. Conveniently, the rewriting required in order to have DHCP and ARP function correctly in the presence of MOOSE rewriting is trivial.

Of concern however are recent standards for layering on top of Ethernet protocols which were previously used solely on dedicated hardware interconnects, such as Fibre Channel over Ethernet (FCoE) [T11 FC-BB-5 working group, 2009]. In order to support FCoE and similar protocols on a MOOSE network, each edge switch will need to be able to interpret and rewrite individual protocols that are in use. A production MOOSE switch would, therefore, need to be implemented such that it is possible to add rewriting support for additional protocols after manufacture.

Ultimately, in the general case, this problem could be addressed more satisfactorily by extending the Ethernet standard to provide a protocol-agnostic method for a layer 2 network to inform hosts of their own addresses; LLDP [IEEE Computer Society, 2009] would make a good basis for this extension. This would allow the use of dynamic MAC addresses with any protocol, with some rewriting performed either partially (within the frame payload) or fully by the host itself, and furthermore would allow higher-layer protocols to respond to changes of the host's network-assigned address. This is, however, a very long-term solution, and protocol-specific rewriting on the switch is likely to be required for the foreseeable future.

FCoE is particularly unusual as it already does its own dynamic allocation of MAC address to devices. It is conceivable that an extension to FCoE could be developed which allows a network-wide dynamic address assignment scheme such as MOOSE to be exploited to provide addresses directly to fibre channel devices.

2.3.2 Edge Virtual Bridging

The rise of virtualisation has caused a proliferation of software switches, usually in the host operating system or hypervisor which provides network connectivity to multiple virtual machines. Since software switches are rarely as manageable or as performant as hardware switches, there are efforts elsewhere—Port Extension [Pelissier and Raeber, 2010], Edge Virtual Bridging [Jeffrey et al., 2009] and VEPA [Congdon et al., 2010]—to create a means of making these software switches act merely as additional ports which are logically part of a more central hardware switch. This reduces the work required by a virtual edge switch: frames from local virtual edge ports can be forwarded straight out via the uplink to a physical switch without consideration, and frames from the uplink will arrive simply tagged with a virtual edge port identifier.

(The scope of Port Extension in particular is greater than this, and allows for physical port extenders to replace switches where a large number of ports is required, but virtualisation is likely to be the most significant use case.)

These extensions would require very little adaptation to be implemented on a MOOSE switch. It is unlikely, although too early in the standardisation process to say for certain, that the virtual bridge will need to be MOOSE-aware. A virtual-bridging-aware physical MOOSE switch will thus simply need to take into account the possibility that one physical port may hide a large number of virtual ports when allocating host identifiers, as it would if it had an Ethernet switch connected on that port. If, however, the virtual bridge is made MOOSE-aware, the hierarchical addressing of MOOSE could be exploited to allow the virtual bridge to allocate host identifiers itself, given that it is likely to be aware of the exact nature of virtual edge ports. The parent MOOSE switch would accordingly delegate an address prefix to each child virtual bridge.

2.3.3 Other uses of LAA space

MOOSE is not the only application to make use of locally-administered MAC addresses (LAAs). For example, Citrix XenServer assigns by default a randomly-generated LAA to each virtual machine. This practice is not universal across virtualisation products, however; VMware instead has an IEEE-assigned OUI which allows it to assign globally-administered MAC addresses.

Unfortunately, it may not be possible for two conflicting uses of LAA space to co-exist. If MOOSE is used with XenServer, switches would be unable to differentiate between MOOSE addresses and those which had not yet been rewritten. Solving this problem may require the involvement of standards bodies; if VMware's use of an OUI is considered best practice, the current behaviour of XenServer could be deprecated. It may also prove necessary to reserve one or more OUIs for MOOSE, in order to ensure conflicts do not occur, although this would result in a smaller number of bytes

available for the hierarchical address.

2.4 Implementation and Evaluation

There are currently three implementations of MOOSE in various stages of completeness. The most complete—the result of an undergraduate project by Wagner-Hall [2010]—is based on the OpenFlow platform [McKeown et al., 2008] with a NOX controller and will operate on commercial switches which support OpenFlow. This implementation has gained some interest from switch vendors at the IETF, and although its original author is no longer continuing to develop it I intend to keep it up-to-date with the current state of MOOSE as I enhance it. There is also a separate implementation for the ns3 network simulator due to an undergraduate project by Whitehouse [2011] which will be used for evaluation and a Python-based software implementation for rapidly prototyping new features.

The implementations have been tested for functional correctness on simple topologies running representative IPv4 traffic and paying particular attention to the effect on unmodified hosts' operation. MOOSE was found to be transparent with the only visible effect being the presence of MOOSE addresses in place of manufacturer-assigned MAC addresses in hosts' ARP caches. Inspection of switches' internal state has validated the expectation that the storage requirement has been reduced from $O(\text{hosts})$ to $O(\text{switches})$, assuming that the number of locally-connected hosts is a small constant; this is a significant improvement.

Whitehouse presented preliminary simulation data which validates the prediction that the amount of state required in MOOSE switches is considerably less than in Ethernet switches on a large topology; the simulation is able to operate using much larger topologies than a real test rig, even one using virtualisation.

A full evaluation, likewise based primarily on the ns3 implementation, is ongoing work and will be presented in my thesis. I intend to compare MOOSE quantitatively with Ethernet in as many dimensions as are relevant. It may also be possible to compare MOOSE in simulation with other proposed extensions to Ethernet, such as SEATTLE [Kim et al., 2008].

2.5 Ongoing Work

The foundations of MOOSE are largely complete. However, various parties including the chairs of the IETF ARMD working group have expressed an interest in working with me to develop MOOSE further. This will be an opportunity to gain input from switch vendors and data centre operators, and I expect that this will lead to architectural adaptations to MOOSE to fit real rather than idealised scenarios.

The IETF operates on long timescales, so ongoing development of MOOSE is likely to proceed slowly. Evaluation and some further development can continue independently of any collaborations however.

Broadcast Traffic Optimisation

3.1 The Broadcast Problem

Large Ethernet networks suffer from an excess of broadcast traffic. It is normal for all IPv4 hosts to routinely emit broadcast packets for a variety of reasons. First and foremost, the standard behaviour of ARP [Plummer, 1982] in resolving the MAC address for a given IPv4 address is to emit a broadcast query. Each host must perform at least one such resolution for each address to which it intends to transmit unicast packets, and for the default gateway for destinations outside the local subnet.

A wide range of other protocols, common and obscure, also make use of the broadcast facility. For example:

DHCP: used for automatic configuration of hosts with IPv4 addresses and other details necessary for connection to a network;

NetBIOS, SMB: used by Windows filesharing and domains;

Common UNIX Printing System (CUPS): a printer client and server which can advertise and discover printers via broadcast messages;

Dropbox: a proprietary file synchronisation tool with a broadcast-based host discovery protocol.

Furthermore, there are non-IP sources of broadcast traffic, even on current networks, such as NetWare (which traditionally used IPX) and various protocols layered on top of Logical-Link Control (LLC).

Myers et al. [2004] have predicted that on a million-node network, ARP traffic alone may peak at 239 Mbps. Although this number was reached from an extrapolation of dubious validity from an experiment involving 2,456 hosts, this does illustrate the scale of the problem; this volume of traffic could entirely saturate slower links

(100 Mbps Ethernet is still commonplace, and 802.11a/b/g wireless has even-lower capacity) which would make a network effectively unusable.

Modern service discovery protocols such as Apple mDNS and Microsoft UPnP SSDP use multicast instead of broadcast. This is considerably more efficient than broadcast as packets of a particular protocol will only reach hosts which have explicitly expressed an interest in that protocol, assuming the Ethernet switches support IGMP snooping [Christensen et al., 2006]. However it is not feasible to entirely replace existing widely-deployed broadcast protocols with multicast or unicast versions as suggested by Myers et al..

3.1.1 Experiment

I have undertaken an experiment to illustrate some of the current uses of broadcast.

A sample of slightly over 20,000 broadcast packets was obtained from each of two small campus networks over the course of a few minutes at an off-peak time, during which time around 25 broadcast packets per second were received by the measuring hosts. The aggregate per-protocol packet counts are presented in Figure 3.1.

Several conclusions can be drawn from this data. Firstly, despite the brief sampling window and small sample networks, several broadcast protocols were detected including some which could not be decoded by Wireshark. Secondly, and quite alarmingly, it is clear that not all broadcast traffic consists of legacy protocols. Dropbox was a major contributor of broadcast traffic on both measured networks; this is a recently-developed product (initially released in 2008) and yet its authors chose to make substantial use of broadcast despite the industry's trend towards multicast. This suggests either that the problems of broadcast are not well-understood in the industry or that multicast was determined to be unreliable for this purpose. Since Dropbox is a proprietary system, it could hypothetically switch to a multicast-based protocol as part of a software update, but this may be a sign of a wider lack of understanding of best practice.

The majority of broadcast packets are ARP; removing this source of broadcast would substantially improve the situation.

3.2 Address Directory Service: ELK

I propose a directory service, Enhanced Lookup (ELK), which runs with switches' involvement to learn mappings from IPv4 to MAC addresses and subsequently handle ARP queries from hosts in a broadcast-free manner. Similarly to MOOSE, no modification to hosts is required; switches intercept ARP packets broadcast by hosts and convert them into unicast queries into the ELK directory service.

Protocol	Packets		Bytes	
ARP	63.92 %	25 614	44.21 %	1 529 716
IPv4	26.20 %	10 500	40.69 %	1 407 928
UDP	26.20 %	10 500	40.69 %	1 407 928
Dropbox LAN sync Discovery	10.13 %	4 058	19.48 %	674 073
NetBIOS Name Service	7.44 %	2 983	8.05 %	278 594
NetBIOS Datagram Service	1.54 %	619	4.25 %	146 976
CUPS Printer Browsing	0.59 %	238	1.55 %	53 554
DHCP / BOOTP	0.16 %	65	0.66 %	22 851
<i>other</i>	6.33 %	2 537	6.70 %	231 880
Logical-Link Control (LLC)	6.82 %	2 731	6.54 %	226 381
IPX in LLC	0.58 %	234	0.41 %	14 136
Service Advertisement Protocol	0.34 %	138	0.24 %	8 280
IPX Routing Information Protocol	0.24 %	96	0.17 %	5 856
<i>other</i>	6.32 %	2 497	6.13 %	212 245
IPX	3.05 %	1 221	8.55 %	295 656
Service Advertisement Protocol	1.98 %	793	7.61 %	263 452
IPX Routing Information Protocol	0.93 %	372	0.78 %	26 940
NetBIOS over IPX	0.14 %	56	0.15 %	5 264
<i>other</i>	0.01 %	3	0.01 %	180
Total	100.00 %	40 069	100.00 %	3 459 861

Figure 3.1: Broadcast traffic breakdown by protocol, aggregated across two samples of a few minutes each on separate small campus networks

ELK then acts in a manner akin to a caching proxy ARP gateway [Carl-Mitchell and Quarterman, 1987]: if the MAC address sought by the querier is already known, the reply can be sent immediately via unicast, otherwise ELK emits a broadcast query. When the reply is received from the target, the address mapping is cached before being repackaged for the original querier; if another host sends a similar query, it can now be answered directly from the cache. Although this method does not entirely eliminate broadcast ARP, it is expected to reduce it considerably, especially if the cache entries are periodically renewed by means of unicast queries so that the cache will likely contain all active hosts.

In its simplest form, the ELK directory service can run as an application on a single ordinary host. On larger networks it may be preferable to distribute it on multiple synchronised servers in order both to reduce the load on a single server and to reduce the number of hops which must be traversed between a querier and the directory server. If ELK is used in combination with MOOSE, features of the MOOSE routing protocol can be leveraged: assuming a suitable modern routing protocol is used, multicast and anycast features may be available at layer 2. ARP queries could then be converted

by switches from broadcast into anycast, and thus reach the nearest server participating in ELK. ARP replies and gratuitous ARP announcements could be converted into multicast updates directed at the entire herd of ELK simultaneously.

The reduction in ARP traffic, and hence the benefit to the network from ELK, is proportional to the size of the network [Aggarwal, 2011]. Let H be the number of hosts, S the number of switches and L the number of links; given the number of ports per switch is fixed by the manufacturer (assuming a wired network, and that if virtualisation is in use there is a fixed maximum number of virtual machines per physical machine), $S \propto H$. On a simple topology $L \propto S + H$, and thus $L \propto H^2$. The total amount of broadcast traffic generated throughout the network when every host is trying to resolve the address of every other host would therefore be $O(H^2L) = O(H^4)$ for ARP and $O(H^2)$ for ELK (after the learning phase): there are inevitably $O(H^2)$ broadcast queries emitted by hosts but on ELK they only transit a single hop whereas with traditional ARP they each traverse every link.

As a proof of concept, Aggarwal has implemented a single-server version of ELK using OpenFlow and a NOX-based controller to instruct switches to intercept the relevant packets and forward them to a ELK server written in Java. His evaluation (summarised by Figure 3.2) validates the expected behaviour, that ELK significantly reduces the volume of ARP traffic and in particular on the network tested entirely eliminates broadcast ARP traffic once the cache has been established.

3.2.1 Related Work

SEATTLE [Kim et al., 2008] also includes an ARP optimisation; since their switching architecture is based upon a distributed hash table (DHT), this DHT is also used to store ARP mappings with switches intercepting and answering ARP messages directly if the answer can be found from a DHT query. A DHT may indeed be a practical way to distribute such mappings and could be used in a multi-server ELK environment; however I feel that adding a DHT to switches will, as previously discussed, be too costly. Thus I prefer an approach which keeps the ARP processing logic and directory separate from the switch hardware, requiring only OpenFlow support on the switches.

3.3 IPv6 Neighbour Discovery

The undesirability of broadcast traffic was already apparent by the time IPv6 was being developed, and (despite ARP being a general-purpose protocol, not tied to IPv4 nor to Ethernet) the IPv6 authors replaced ARP with a new, multicast-based protocol: Neighbour Discovery (ND) [Narten et al., 2007]. IPv6 hosts join at least one *solicited-node multicast group* [Hinden and Deering, 2006] with an address formed from the low-

3. BROADCAST TRAFFIC OPTIMISATION

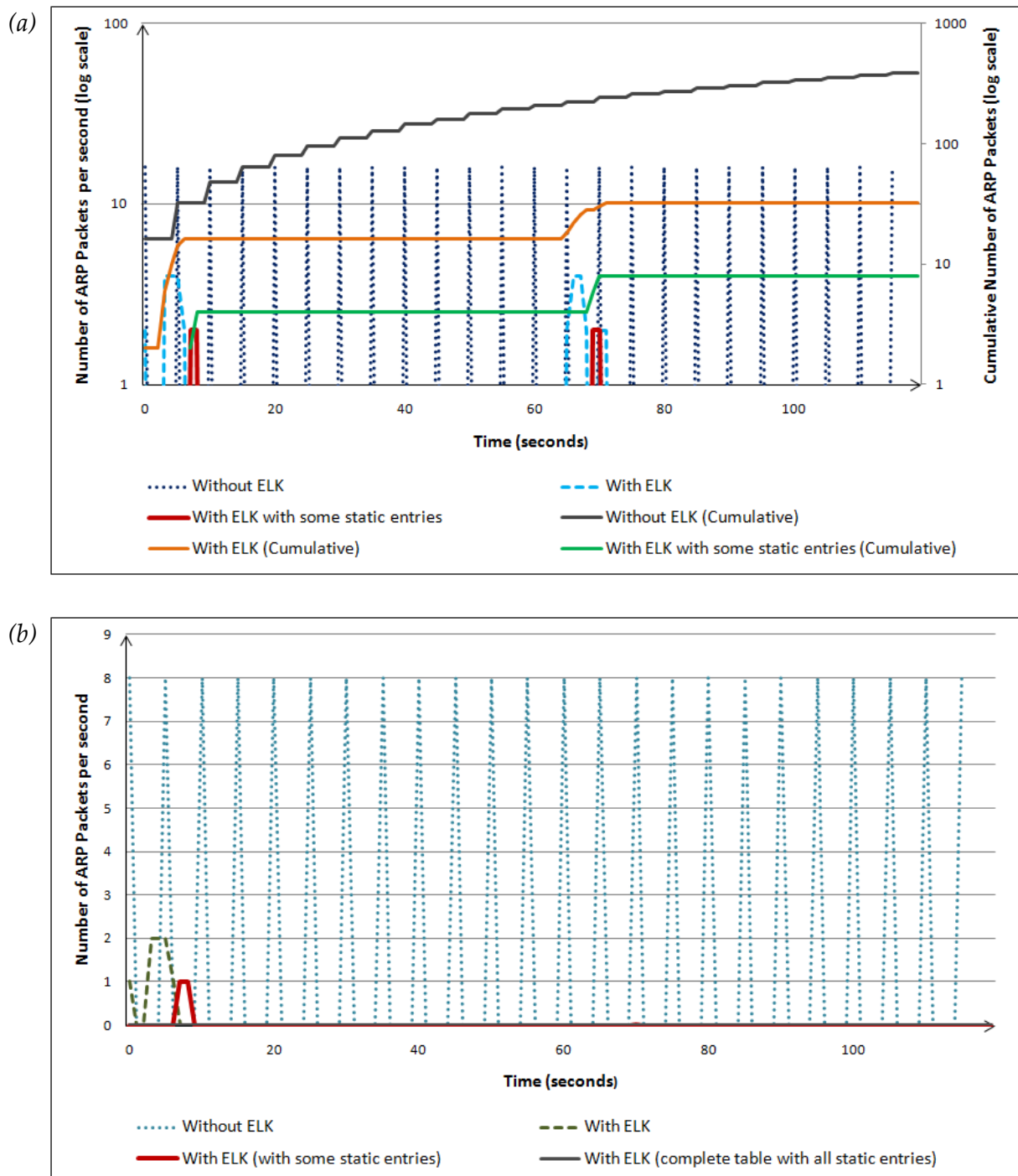


Figure 3.2: Rate of ARP messages exchanged by a host on a small test network with and without ELK: (a) all ARP messages exchanged; (b) broadcast ARP messages received

order 24 bites of each of their IPv6 addresses, which are in turn mapped onto layer-2 multicast addresses. Other hosts querying for an address will send the query to the solicited-node multicast address formed from their query. Thus the intention is that only small groups of hosts will receive each other's queries. This is expected to be more scalable than ARP [Mack-Crane et al., 2010].

However, this only helps at all when the switches support Multicast Listener Discovery (MLD) snooping (the IPv6 equivalent to IGMP snooping, and not yet widely supported); furthermore since it potentially imposes a large number of multicast groups on the network, if several addresses are in use on the same physical system due to virtualisation the capacity of the NIC's MAC filter table may be exceeded, causing layer-2 multicast filtering at the NIC level to break down and potentially requiring the use of promiscuous mode.

3.4 Ongoing Work

ELK is an ongoing development and I intend to extend it for my thesis. Once the infrastructure to gather mappings from IPv4 to MAC addresses is in place, this data may be useful for other purposes such as administrative monitoring for duplicate addresses or misconfigured devices. Furthermore, it would be logical to integrate ELK with DHCP: since DHCP hands out IPv4 address leases to specific MAC addresses, the DHCP server could prepopulate the ELK directory with this mapping at the point of allocation so that no broadcast query ever has to be made for any IPv4 address in the DHCP pool. Indeed, it may prove worthwhile to build a combined ELK and DHCP server so that broadcast DHCP messages can be converted into unicast through the ELK infrastructure.

Aggarwal is no longer developing his ELK implementation (the implementation and evaluation formed a standalone undergraduate project) but he and I will coauthor a paper describing ELK in more detail for publication and for my thesis. ELK also aligns directly with the remit of the IETF ARMD working group (see Section 1.3.1) and I intend to write a draft standard for the working group's consideration.

It should be noted that ARP is by no means the only source of broadcast traffic; whilst ELK makes a significant contribution to reducing broadcast traffic, an optimisation restricted to ARP (and perhaps DHCP) may not be sufficient for larger networks. I will continue to research other methods for reducing broadcast traffic more generally.

Kim et al. propose breaking the current 1:1 link between subnets and broadcast domains by using a modification of VLANs termed *groups*; hosts are divided up between multiple groups (broadcast domains) within a single IP subnet or multiple, in a manner which allows unicast traffic to pass between groups within a subnet and so does not limit reachability. Broadcast traffic within a group is forwarded as if it were multi-

cast, using reverse-path forwarding along a spanning tree containing only the switches currently participating in that group; as noted in Section 2.2.2 reverse-path forwarding proves to be inefficient. However there may be some merit in developing this into a practical solution.

One further avenue towards generalised broadcast optimisation would be to attempt to infer multicast groups automatically by tracking sources of different kinds of broadcast traffic. For example, in the case of Dropbox it is trivial to identify which hosts are running Dropbox by watching for the regular broadcast announcements. There is no need for these announcements to reach hosts which are not running Dropbox, and so a layer-2 multicast group could be formed containing only those hosts which have been observed to emit Dropbox packets. Any subsequent broadcast messages from the Dropbox application could be transparently converted into multicast and delivered only to interested parties. It may be possible, using simple heuristics, to perform this conversion without any specific knowledge of the behaviour of Dropbox; applications can be identified by their protocol headers—in the case of Dropbox and several similar protocols, the UDP port number can be used.

CHAPTER 4

Layer-3 Virtual Machine Migration

It is well-known that large subnets are problematic, and accepted wisdom amongst network operators in the IETF—seemingly stemming from the days of shared-medium Ethernets on which *all* traffic was broadcast—is that a good network design will involve small broadcast domains [Hinden and Carpenter, 2010] and that anything else is “wrong”. There is a surprising prevalence of the opinion that all current virtualisation platforms must be fundamentally broken since they require large subnets. However, philosophical arguments aside, virtualisation and live migration are extremely popular; the unfortunate fact is that the only way to enable live migration given current network protocols is to use large broadcast domains.

The same reasoning can lead people of this mindset towards rejection of the enhancements to Ethernet switching that I have proposed in previous chapters, since they are seen as further encouraging large subnets which have always been “wrong”. The reality, of course, is that large subnets were only ever wrong due to specific failings of Ethernet and IP, and that MOOSE and ELK go some way towards fixing a long-standing problem in a manner which could be deployed in a backwards-compatible way on real (OpenFlow-compatible) switches in the near future.

Nevertheless, it seems prudent to investigate the possibility of a higher-layer solution to the problem of addressing a large collection of migratable virtual machines—one which could be deployed as a software modification to a virtualisation platform such as XenServer or VMware in order to add live migration on top of an existing routed network with small subnets.

There is, furthermore, an understandable prevailing attitude in the IETF against further development of IPv4-based technologies and for promotion of advantages of IPv6 over IPv4. It is certainly the case that since the exhaustion of IANA’s pool of IPv4 addresses, IPv6 is being deployed on a wide range of networks; it will be in the interest of content providers—and therefore of data centre operators—to operate dual-stacked IPv6 and IPv4 services in the near future in order to best serve clients whose

IPv4 connectivity is encumbered by multiple layers of NAT. With this in mind, and considering the improvements which IPv6 brings, I believe it is reasonable for this solution to be built on IPv6.

Such a solution appears to be feasible, and I am in the early stages of an experimental implementation. If this is a success, it will form a part of my thesis. I will describe the current design, although the details are liable to change depending on the outcome of the investigation.

4.1 Transparent IPv6 Address Persistence¹

When a host is moved across a subnet boundary, it will gain a new address. This in itself is not a problem; if the old address remained reachable and the host is capable of correctly operating on multiple addresses (which is handled by IPv6 multihoming) then the host's connections would not be interrupted and the host would have successfully migrated between layer-2 networks. The challenge therefore is in ensuring that packets addressed to a host's old location are forwarded on to its new address in a scalable manner—that is to say, without maintaining a list of migrated addresses in any one host or router.

IPv6 Mobility exists to address a similar problem of maintaining a static address whilst moving between locations. However, two issues make it impractical in this case:

1. Each host would have to be assigned permanently to, and obtain an address from, a home agent. The naïve implementation would contain a single home agent; however in such a scenario, the home agent would have to maintain bindings and forward packets for every mobile node which is not scalable. If instead a collection of several home agents were used, another abstraction layer would likely be needed to automatically configure mobile nodes with the details of a home agent with sufficient spare capacity. This system would rapidly become overcomplicated.
2. IPv6 Mobility requires support in the IPv6 stack of the node being migrated (the mobile node). Despite this being a core part of the IPv6 standard, this has been implemented in few operating systems to date. There are two known implementations IPv6 Mobility in Linux, the first an implementation of a very early draft of the protocol by Lancaster University as a patch for Linux 2.1.90 (current circa 1998) and the second by Helsinki University of Technology which appears to have been abandoned since 2006. The websites of both implementations are no

¹Since this is very early work, it has not yet gained an antlered acronym.

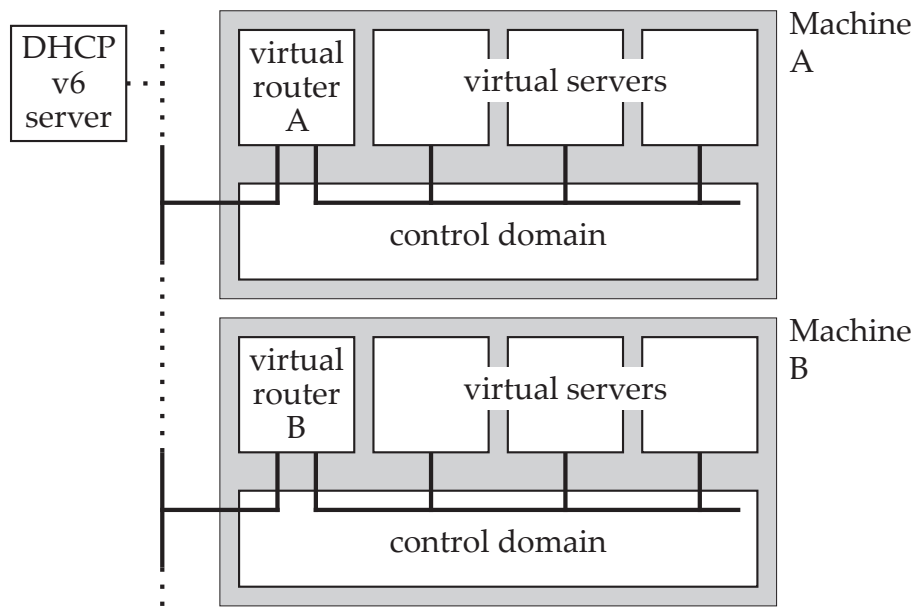


Figure 4.1: Virtual network topology

longer accessible, although the final state of the HUT implementation has been preserved by the Internet Archive.²

A better solution would be decentralised and would not require any modification to virtual machines' operating systems.

4.1.1 Components and Interconnections

The prototype implementation is being built on the Citrix XenServer platform, but for ease of maintenance and to aid transfer to other platforms the platform itself has not been modified. Instead, the intelligence is being implemented in lightweight *virtual routers*, minimal Linux virtual machines instantiated once per physical server.

Virtual networks are used to place all other virtual machines behind the virtual router. Logically, the virtual routers are interconnected with each other, either directly at layer 2 or via intermediate routers. Traffic to a virtual server on host A passes through virtual router A; the virtual router has an uplink towards the internet and a downlink to an entirely-virtual layer-2 network for the virtual servers currently running on that host. This is illustrated in Figure 4.1.

The virtual routers also communicate with a configuration host (another virtual server, running anywhere in the cluster). This runs a DHCPv6 server [Droms et al., 2003] and assigns a unique /64 subnet to each virtual router for its local virtual servers

²<http://web.archive.org/web/20080509070021/http://www.mobile-ipv6.org/>

using prefix delegation [Troan and Droms, 2003]. Linux DHCPv6 client support has proven immature, but workable with some bug-fixing.

In turn, the virtual routers are responsible for configuring the virtual servers currently residing locally. This is achieved using the simpler stateless autoconfiguration mechanism, SLAAC [Thomson et al., 2007] as the stateful features of DHCPv6 are not needed in this case; SLAAC client support is mature in every mainstream operating system so clients do not need to install any additional software.

4.1.2 Actions upon Migration

When a virtual server migrates from machine A to machine B, virtual routers A and/or B must become aware of this. This mechanism is not yet finalised and multiple options exist, including:

Integration with the virtualisation control plane: The virtual routers, or the configuration server, could maintain a connection to the virtualisation layer (for example using XAPI if running on XenServer) and detect the migration of virtual machines. Virtual machines can be identified by MAC address, which remains persistent.

Detection of packet from wrong subnet: If the migrated host has connections open it will likely send a packet using its old address as the source. This will be received by virtual router B which can refer to the upper 64 bits of the address to determine the host's previous location, i.e. that this host was previously the responsibility of virtual router A.

Virtual router A must then be instructed to set up a forwarding rule: any packets which arrive for the migrated host's old address must be redirected to the host's new location instead of to the local virtual network. This will most likely involve virtual router B impersonating A to the migrated host, with packets arriving from the real virtual router A via a tunnel. Outbound packets from the migrated host can be delivered directly, without having to pass through A again, if the routers explicitly do not attempt to perform source address filtering.

Conclusion and Thesis Plan

I have described my three current avenues of research into data centre network scalability:

- MOOSE, a scalable addressing and switching architecture for Ethernet;
- Optimisations for reducing broadcast traffic, including ELK;
- A wholly-software, layer-3 virtual machine migration enabler.

These were presented in descending order of completeness at the time of writing; all three will be developed further for conference papers and ultimately for my thesis. It is likely that further avenues will be added to the list as they are discovered. I expect to have completed this work within the time available.

5.1 Thesis Structure

My thesis will approximately follow the structure of this report, with the following chapters:

1. Introduction
2. Data Centre Networking: Background and Related Work
3. Scalable Ethernet Addressing: MOOSE
4. Broadcast Traffic Optimisation: ELK and Inferred Multicast
5. Layer-3 Virtual Machine Migration: Transparent IPv6 Address Persistence
6. Conclusions and Future Work

Chapters 3, 4 and 5 will be derived from separate papers and will each contain a separate evaluation section, as my three major contributions address different parts of the problem space and can be implemented and used separately.

It is quite likely that my work on transparent IPv6 address persistence will extend into the realm of IPv6 transition technologies in a data centre context. If this work produces any significant findings, it may warrant an additional chapter.

5.2 Timeline

Period	Primary goal	Secondary goals
Summer 2011	Complete design and implementation of Transparent IPv6 Address Persistence	Write preliminary ELK paper; review recent developments, including IETF ARMD work
Michaelmas 2011	Write paper on Transparent IPv6 Address Persistence	Extend ELK to integrate with DHCP; preliminary investigation of inferred multicast
First half 2012	Design and implement inferred multicast	Evaluate MOOSE; complete ELK paper
Second half 2012	Write paper on broadcast optimisation	Depending on status of ARMD, prepare updates to MOOSE and initial internet draft of ELK
First half 2013	Thesis	Continue working with IETF if appropriate

Bibliography

- H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly. Symbiotic routing in future data centers. *ACM SIGCOMM Computer Communication Review*, 40(4): 51–62, Aug. 2010.
- A. Adams, J. Nicholas, and W. Siadak. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). RFC 3973 (Experimental), Jan. 2005. URL <http://www.ietf.org/rfc/rfc3973.txt>.
- I. Aggarwal. Implementation and evaluation of ELK, an ARP scalability enhancement. Computer Science Tripos Part II, Corpus Christi College, University of Cambridge. Dissertation, May 2011.
- P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. SOSP*, pages 164–177, New York, NY, USA, 2003. ACM. ISBN 1-58113-757-5. doi: 10.1145/945445.945462.
- S. Carl-Mitchell and J. Quarterman. Using ARP to implement transparent subnet gateways. RFC 1027, Oct. 1987. URL <http://www.ietf.org/rfc/rfc1027.txt>.
- D. Chrapaty. The reality of the cloud: Microsoft's data center strategy. Microsoft Management Summit keynote address, May 2008.
- M. Christensen, K. Kimball, and F. Solensky. Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches. RFC 4541, May 2006. URL <http://www.ietf.org/rfc/rfc4541.txt>.
- C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. USENIX NSDI*, 2005.
- P. Congdon, A. Fischer, and P. Mohapatra. A case for VEPA: Virtual Ethernet Port Aggregator. In *ITC 22 Second Workshop on Data Center – Converged and Virtual Ethernet Switching (DC CAVES)*, Sept. 2010.

- R. Droms. Dynamic Host Configuration Protocol. RFC 2131 (Draft Standard), Mar. 1997. URL <http://www.ietf.org/rfc/rfc2131.txt>. Updated by RFCs 3396, 4361.
- R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315 (Proposed Standard), July 2003. URL <http://www.ietf.org/rfc/rfc3315.txt>. Updated by RFC 4361.
- L. Dunbar and B. Schliesser. Address resolution for massive numbers of hosts in the data center (arnd). IETF Working Group charter, Mar. 2011. URL <http://www.ietf.org/dyn/wg/charter/arnd-charter>.
- I. Hadžić. Hierarchical MAC address space in public Ethernet networks. In *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, volume 3, 2001.
- B. Hinden and B. Carpenter. Discussion at ARMD BoF. IETF 79, Beijing, Nov. 2010.
- R. Hinden and S. Deering. IP Version 6 Addressing Architecture. RFC 4291 (Draft Standard), Feb. 2006. URL <http://www.ietf.org/rfc/rfc4291.txt>.
- IEEE Computer Society. Std 802.1D: Standard for local and metropolitan area networks: Media access control (MAC) bridges, 2004a.
- IEEE Computer Society. Std 802.1X: Port based network access control, 2004b.
- IEEE Computer Society. Std 802.1ab: Station and media access control connectivity discovery, 2009.
- A. Jeffree, P. Congdon, and J. Pelissier. P802.1Qbg: Edge virtual bridging. PAR, Dec. 2009.
- C. Kim, M. Caesar, and J. Rexford. Floodless in SEATTLE: a scalable Ethernet architecture for large enterprises. In *Proc. SIGCOMM*, pages 3–14, 2008. doi: 10.1145/1402958.1402961.
- B. Mack-Crane, L. Dunbar, and S. Hares. IPv6 Neighbour Discovery scalability for large data centers. draft-mackcrane-arnd-ipv6-nd-scaling-00, Oct. 2010. URL <http://tools.ietf.org/html/draft-mackcrane-arnd-ipv6-nd-scaling-00>. (Expired).
- N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.
- R. M. Metcalfe and D. R. Boggs. Ethernet: distributed packet switching for local computer networks. *Commun. ACM*, 19(7):395–404, 1976. ISSN 0001-0782. doi: 10.1145/360248.360253.

- J. Moy. OSPF Version 2. RFC 2328 (Standard), Apr. 1998. URL <http://www.ietf.org/rfc/rfc2328.txt>.
- A. Myers, E. Ng, and H. Zhang. Rethinking the service model: Scaling Ethernet to a million nodes. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, Nov. 2004.
- T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbour Discovery for IP version 6 (IPv6). RFC 4861, Sept. 2007. URL <http://www.ietf.org/rfc/rfc4861.txt>.
- K. Pagiamtzis and A. Sheikholeslami. Content-Addressable Memory (CAM) circuits and architectures: a tutorial and survey. *IEEE Journal of Solid-State Circuits*, 41: 712–727, 2006.
- J. Pelissier and R. Raeber. Introduction to Port Extension (IEEE P802.1Qbh). In *ITC 22 Second Workshop on Data Center – Converged and Virtual Ethernet Switching (DC CAVES)*, Sept. 2010.
- C. Perkins. IP Mobility Support for IPv4. RFC 3344 (Proposed Standard), Aug. 2002. URL <http://www.ietf.org/rfc/rfc3344.txt>. Updated by RFC 4721.
- R. Perlman. Rbridges: transparent routing. In *Proc. IEEE INFOCOM*, volume 2, 2004.
- D. C. Plummer. Ethernet Address Resolution Protocol. RFC 826, Nov. 1982. URL <http://www.ietf.org/rfc/rfc826.txt>.
- T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. SmartBridge: a scalable bridge architecture. In *Proc. SIGCOMM*, 2000.
- E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), Jan. 2001. URL <http://www.ietf.org/rfc/rfc3031.txt>.
- G. M. Schneider and T. Nemeth. A simulation study of the OSPF-OMP routing algorithm. *Computer Networks*, 39(4):457–468, 2002. ISSN 1389-1286. doi: DOI:10.1016/S1389-1286(02)00231-1.
- M. Scott, A. Moore, and J. Crowcroft. Addressing the scalability of Ethernet with MOOSE. In *ITC 21 First Workshop on Data Center – Converged and Virtual Ethernet Switching (DC CAVES)*, Sept. 2009.
- S. Sipes. Why your switched network isn't secure. In *Intrusion Detection FAQ*. The SANS Institute, Sept. 2000. URL http://www.sans.org/resources/idfaq/switched_network.php.

- T11 FC-BB-5 working group. Fibre channel backbone – 5, June 2009.
- S. Thomson, T. Narten, and T. Jinmei. IPv6 stateless address autoconfiguration. RFC 4862, Sept. 2007. URL <http://www.ietf.org/rfc/rfc4862.txt>.
- O. Troan and R. Droms. IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6. RFC 3633 (Proposed Standard), Dec. 2003. URL <http://www.ietf.org/rfc/rfc3633.txt>.
- C. Villamizar. OSPF optimized multipath (OSPF-OMP). IETF Internet Draft, Feb. 1999. URL <http://tools.ietf.org/html/draft-ietf-ospf-omp-02>.
- D. Wagner-Hall. NetFPGA implementation of MOOSE. Computer Science Tripos Part II, Homerton College, University of Cambridge. Dissertation, May 2010.
- R. Whitehouse. Implementation of data link layer protocols for a network simulator. Computer Science Tripos Part II, Homerton College, University of Cambridge. Dissertation, May 2011.
- F. Yu, R. H. Katz, and T. V. Lakshman. Efficient multimatch packet classification and lookup with TCAM. *IEEE Micro*, 25(1):50–59, Jan. 2005. ISSN 0272-1732. doi: 10.1109/MM.2005.8.