Implementation and Evaluation of Latency Efficient Overlay Network (LEON)

Yudai Yamagishi† Katsuhiro Horiba† Keisuke Uehara† Jun Murai‡
†Graduate School of Media and Governance, Keio University
‡Faculty of Environment and Information Studies, Keio University
{yummy, qoo, kei, jun}@sfc.wide.ad.jp

Many Internet services today use multiple geographically distributed locations to provide a service with fast response time and fault tolerance. When using multiple locations to provide a service, we require the latency between the locations to be as low as possible to prevent impacts to the service response time. To realize this, we propose Latency Efficient Overlay Network, which is a Layer 2 network extension technology capable of forwarding Ethernet frames using the lowest latency path. LEON measures the latency of the locations and uses Dijkstra's algorithm to select the lowest latency path. By using LEON, the performance of latency dependent components improves. This allows faster processing of the requests to a service and a service to be provided with better response time.

1 Background

Many Internet services today use multiple geographically distributed locations to provide a service. However, as the paths on the Internet are not selected using parameters such as latency, the latency is high in some cases. High latency between the components of the service cause slower processing of requests. This results in a longer response time visible to the service's clients.

1.1 Internet Services Today

The services on the Internet today need to be responsive and fault tolerant in order to achieve user satisfaction. To achieve this, many services are now using multiple locations to provide a service.

The Internet users' satisfaction is greatly affected by the service's response time. According to the research done by Zona Research in 2001, the response time level at which users felt satisfied was 8 seconds [1]. However, according to research done by Akamai 5 years later, in 2006, it was 4 seconds [2]. Furthermore, the same research was done by Akamai another 3 years later, in 2009, and it was 2 seconds [3]. The response time level at which users feel satisfied can be predicted to become even lower in the future. To meet this demand, the response time of a service should be as low as possible.

Also, service downtimes today can cause great damage to a service. In recent years, there have been many news stories about natural disasters and power failures taking down a whole datacenter, some taking over a week to recover [4]. According to research done in 2009, the financial losses caused by a one-hour downtime of a famous online shopping service can be over 2.8 million dollars [3]. To prevent such damage, the services on the Internet today are designed to be fault tolerant.

Multiple locations are often used to provide a service with fast response time and fault tolerance. An example of a service which is provided using multiple locations is a Content Delivery Network (CDN) service provided by Akamai [5]. When a user accesses the service, the service is provided from the server closest to the user. This can achieve faster response time of a service by providing faster data transfer. Also, service downtimes due to a failure in a location can be prevented when a service is provided from multiple locations. In case of a failure in one location, another location will continue to provide service.

To take advantage of using multiple locations, the locations of a service are often geographically distributed and are connected to different networks. Geographically distributed locations can prevent a service failure due to a failure of multiple locations in the same geographic region caused by natural disasters. Also, by having the locations geographically distributed and connected to different networks, the service will have better probability of having a server near the user.

1.2 Routing on the Internet

Traffic on the Internet may not always be routed using the path with the lowest latency. The paths on the Internet are learned and determined using the BGP protocol and the traffic is usually routed using the best path. In BGP, the best path is the path with the shortest AS-path, passing through the fewest independently managed networks. The best path is not determined by the latency or the saturation levels of the link. Also, business matters are often involved in determining the best path. In the past, there have been business disputes which involved peering disconnection of two major networks. This resulted in longer latency between the two networks [6]. Therefore, depending on the destination, the traffic may be routed over the Internet using a path with high latency.

The latency to the destination may be reduced by routing the traffic via another location in order to reach the final destination [7]. If the path relayed by another location has lower latency than the direct Internet path, we should use the lower latency path instead of the direct Internet path. Such routing overlays can improve latency and availability of the destination [8]. By using routing overlays, we can communicate with the destination using the lowest latency path.

1.3 Problem of using Multiple Locations on the Internet

Services on the Internet are often made up of multiple components. For example, WIDE Cloud, which is an IaaS cloud service operated by the WIDE Project, is made up of three components [9]. The first component is the hypervisor, which provides CPU and memory resources to the virtual machine. The second component is the storage server, which provides storage areas for the virtual machines. Lastly, the third component is the network gateway, which handles the network traffic of the virtual machines. The components are composed to provide a service.

When a service is provided using multiple locations, a Layer 2 network is often extended to enable interaction of the components. By extending the same Layer 2 network to all the locations, the service provider can enforce the same security policy easily. Also, the service providers can prevent backend servers from having Internet access. Therefore, some service providers extend the Layer 2 network to the locations to enable interaction of the components.

In the WIDE Cloud, a Layer 2 network is extended to all the locations. The WIDE Cloud consists of multiple locations, mainly inside Japan with some outside of Japan. The storage server and the network gateway are set up in one of the locations. Hypervisors are set up on all the locations. The hypervisors interact with the storage server and the network gateway over the extended Layer 2 network.

The latency between the locations is very high compared to the latency inside a single location. When the components interact over a network with high latency, the performance of the components greatly decreases in some cases. For example, the WIDE Cloud uses NFSv3 [10] to share virtual machines' disk images to all hypervisors. However, the performance of NFSv3 decreases greatly as the latency between the components grow higher [11]. To prevent a slow down of request processing due to a decrease of component performance from impacting service response time, the latency between the components should be as low as possible.

The latency between the components can be reduced by making the Layer 2 network extension technology aware of the latency between the locations. By making the Layer 2 network extension technology be able to detect and select the lowest latency path used to forward Ethernet frames, we can make decrease in performance of the components due to the latency as small as possible.

2 Related Work

Layer 2 network extension technologies are used to extend a Layer 2 network to multiple geographically distributed locations over the Internet. A tunnel end point setup on each location is connected to a local Layer 2 network. When the tunnel end point receives an Ethernet frame, it encapsulates the Ethernet frame and forwards them over the Internet. By using Layer 2 network extension technologies, we can virtually create a wide area Layer 2 network over the Internet.

There are two types of Layer 2 network extension technologies. One is a point-to-point Layer 2 network extension technology and the another is a multi-point Layer 2 network extension technology. They are de-

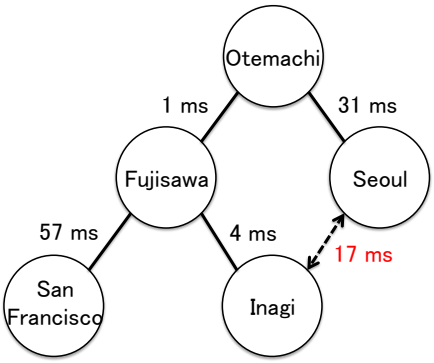

Fig. 1: Current Layer 2 over IP based Layer 2 network extension technologies limit the topology to a tree

scribed in the following sections.

2.1 Point-To-Point Layer 2 Network Extension Technology

Point-to-point Layer 2 network extension technologies such as L2TP [12] are used to extend a Layer 2 network between two locations. A Layer 2 network can be extended to multiple locations by using multiple instances of tunnel end points to connect all locations. A loop in a Layer 2 network topology will cause problems such as broadcast storm and multicast storm. This will saturate the link between the tunnel end points and cause high load on the tunnel end point. Therefore, the topology of a Layer 2 network extended to multiple locations using point-to-point Layer 2 network technologies must be a tree topology.

However, the latency of forwarding Ethernet frames over the Internet is very high compared to forwarding Ethernet frames inside a single location. An example of an extended Layer 2 network's topology is illustrated in Figure 1. In this topology, for an Ethernet frame to reach from Seoul to Inagi, it will take 36 ms. However, direct Internet path between Seoul and Inagi is 17 ms. In order to reduce the latency, the Ethernet frame should be forwarded directly between Seoul and Inagi in this case. However, if the topology is changed to connect Seoul and Inagi directly, the latency to other locations will become higher as the traffic from Inagi to other locations will be forwarded via Seoul. Therefore, a Layer 2 network extended using point-to-point Layer 2 network extension technologies will always have a high latency path.

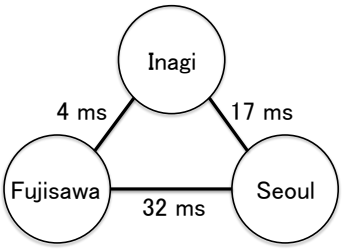

Fig. 2: Example of lower latency via relayed path

2.2 Multi-point Layer 2 Network Extension Technology

Multi-point Layer 2 network extension technologies are Layer 2 network extension technologies which are designed to extend a Layer 2 network to multiple locations. With multi-point Layer 2 network extension technologies, Ethernet frames are directly forwarded to the destination tunnel end point. Therefore, the latency between the locations will be that of a direct Internet path.

One example of a multi-point Layer 2 network extension technology is VXLAN [13]. VXLAN is proposed mainly by Cisco Systems and VMware. VXLAN uses IP multicast to forward broadcast frames and find other tunnel end points. However, the cost of deploying the same IP multicast group over the Internet to multiple locations is very high. Also, the troubleshooting in case of a problem will become harder when multiple technologies are involved. From these points, we can say that VXLAN is not designed to extend a Layer 2 network to multiple locations over the Internet.

Another example is N2N [14]. N2N is a Layer 2 network extension technology developed by NTOP. N2N is designed to extend a Layer 2 network to multiple locations over the Internet. In N2N, there is a central server called a supernode which is used to manage the tunnel end points and forward broadcast frames. However, in N2N, the supernode is a single point of failure. A service provider often uses multiple locations to provide fault tolerance. In N2N, if the location running the supernode fails, the Layer 2 network will stop working.

2.3 Problem with Current Technologies

Current technologies have problems when extending a Layer 2 network to multiple locations over the

Internet. With point-to-point Layer 2 network extension technologies, there will be a high latency path when extended to multiple locations. With multi-point Layer 2 network extension technologies, there are problems such as high operation cost and single point of failures. These problems prevent us from constructing a stable, low latency Layer 2 network extended to multiple locations.

Also, all the technologies do not use the lowest latency path when forwarding Ethernet frames. As described in Section 1.2, a path relayed by another location may have lower latency compared to the direct Internet path. An example of the described case is illustrated in Figure 2. A direct path from Fujisawa to Seoul is 32 ms. By having the traffic relayed via Inagi, the latency can be reduced to 21 ms. When forwarding Ethernet frames, if the described path exists, the tunnel end point should forward it using that path. However, none of the current technologies can detect and forward using the lowest latency path.

3 Proposal

In this section, we propose a multi-point Layer 2 network extension technology which can forward Ethernet frames using the lowest latency path and run in a distributed manner.

3.1 Requirements

The purpose of this research is to reduce the response time of a service running on the Layer 2 network extended to multiple geographically distributed locations over the Internet. Multiple locations are used by the service provider to provide a service with fast response time and fault tolerance. We assume the number of locations to be about a dozen locations. We assume the locations to be distributed inside a same geographic region such as Asia and North America.

The requirements for the Layer 2 network extension technology to be used in the described environment are the following:

- Extend Layer 2 network to multiple locations
- Select next-hop based on Ethernet frames' destination
- Forward using the path with the lowest latency
- Run in a distributed manner

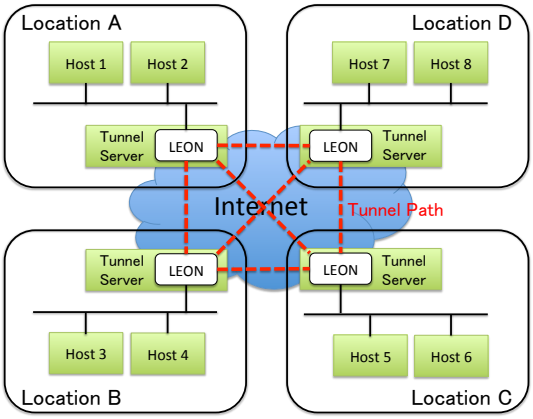

Fig. 3: Layer 2 Network Topology of LEON

The Layer 2 network extension technology must be designed to extend a Layer 2 network to multiple locations distributed over the Internet. When forwarding Ethernet frames, the Layer 2 network extension technology must be able to look at the destination host of the Ethernet frame and determine the destination tunnel end point. Upon forwarding of the Ethernet frame, it must be able to select the lowest latency path in order to make the latency between the locations as low as possible. These are the requirements for improving the performance of components and the response time of a service.

Also, the Layer 2 network extension technology must run in a distributed manner. Since the service providers use multiple locations to provide fault tolerance, a failure of any location should not effect the Layer 2 network. To realize this, the Layer 2 network extension technology must not require any kind of central server and run in a distributed network model.

3.2 Latency Efficient Overlay Network

We propose Latency Efficient Overlay Network (=LEON), a multi-point Layer 2 network extension technology which can forward Ethernet frames using the lowest latency path and run in a distributed manner. By using LEON, a service provider can extend a Layer 2 network to multiple geographically distributed locations over the Internet. The Ethernet frames are forwarded using the lowest latency path, so the response time of the service running on top of the Layer 2 network will be as fast as possible. Also, the Layer 2 network will continue to run in case of complete failure in multiple locations.

An example topology of a Layer 2 network extended to multiple locations over the Internet using LEON is illustrated in Figure 3. In each location, there is a server running LEON. The server is connected to the Internet and to a local Layer 2 network with hosts which will be connected to the extended Layer 2 network. LEON will act as a gateway when a local host and a host in another location communicate.

In LEON, the Ethernet frames can be relayed by another location. Current multi-point Layer 2 network extension technologies only forward Ethernet frames directly to the destination. However, in order to forward Ethernet frames using the lowest latency path, we require the tunnel end points to be able to relay the Ethernet frames received from other locations. When LEON receives an Ethernet frame from another location, the destination of the Ethernet frame is checked. If the destination host is not connected to the tunnel end point, it is relayed to the next hop of the path.

To forward Ethernet frames using the lowest latency path, LEON uses latency as a metric to select the path. LEON measures the latency between all the tunnel end points and share the measurement results with all the tunnel end points. Then, the measurement results are used to select the lowest latency path using Dijkstra's shortest path first algorithm.

Also, LEON runs in a distributed manner to prevent single point of failures in the Layer 2 network. LEON manages the tunnel end points connected to the Layer 2 network by making the tunnel end point send control messages such as join and leave to all the tunnel end points one by one. The tunnel end point also sends broadcast frames to all the tunnel end points in the same manner. This removes the need for IP Multicast or a central server. Therefore, the Layer 2 network extended using LEON will not be affected from failure of a location.

By using LEON, a service provider can construct a latency efficient and fault tolerant Layer 2 network consisting of multiple locations over the Internet. This improves the performance of service's components, which will allow faster processing of requests.

4 Implementation

We implemented LEON on Linux using C. The implementation is an userland application. It creates a TUN/TAP device used to send and receive Ethernet frames. It has been tested to run on Debian GNU/Linux 6.0.6 and Fedora 17. Also, it has been tested to run on the Linux Kernel versions 2.6.43, 3.7.2 and 3.7.4.

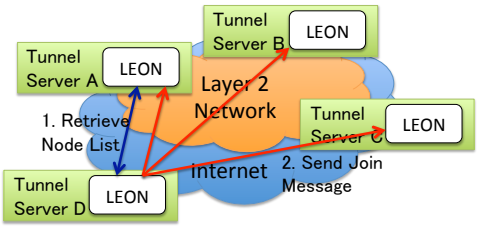

Fig. 4: Join process of a tunnel end point

There are three functions in the implementation: 1. Distributed Node Management, 2. Lowest Latency Path Selection and 3. Ethernet Frame Forwarding. Each of the functions will be explained below.

4.1 Distributed Node Management

To realize the distributed management of the tunnel end points, every tunnel end point manages its own list of all the tunnel end points and uses control messages to update the list.

When a tunnel end point joins the Layer 2 network, it broadcasts a join message to all the tunnel end points. The joining process is illustrated in Figure 4. First, the tunnel end point joining the Layer 2 network retrieves the current list of tunnel end points from one of the tunnel end points. Then, the new tunnel end point sends join messages to all the tunnel end points on the list. The tunnel end point will add the new tunnel end point to the list and the new tunnel end point joins the Layer 2 network.

When the tunnel end point leaves the Layer 2 network, it broadcasts a leave message to all the tunnel end points. When the tunnel end points receive the message, it removes the leaving tunnel end point from the list. Also, it removes all the records related to the tunnel end point such as list of hosts connected to the leaving tunnel end point. After the broadcast of the leave message, the tunnel end point is completely detached from the Layer 2 network.

Also, in order to detect failure of a tunnel end point, status of the tunnel end points are periodically checked. Each tunnel end point sends a ping request at regular intervals to all the tunnel end points. If

the tunnel end point doesn't reply to multiple ping requests, the tunnel end point is considered as failed. The tunnel point gets removed from the list and all the records related to the tunnel end point is removed.

4.2 Lowest Latency Path Selection

When forwarding Ethernet frames, LEON selects the path with the lowest latency and forwards Ethernet frames using that path.

In order to select the path with the lowest latency, a latency database is created on all the tunnel end points. By querying the latency database, the tunnel end point can learn the latency between any of the two tunnel end points connected to the Layer 2 network.

In order to keep the latency database updated, the latencies between the locations are regularly measured by running the measurement process on all the tunnel end points. The process sends a ping request to all the tunnel end points on the tunnel end point list. On receiving ping reply from a tunnel end point, it will store the latency data in its latency database. Also, it broadcasts the latency data to all the tunnel end points in order to update other tunnel end points' latency database. The same copy of the latency database is created on all the tunnel end points.

After the measurement process is finished, the path selection process will run on all the tunnel end points. The path selection process calculates the lowest latency path to all the tunnel end points on the tunnel end point list. The selection process uses Dijkstra's algorithm with latency as a metric to calculate the lowest latency path using the latency database. If the direct Internet path is the lowest latency path, the direct path will be selected. If a path relayed by another location is the lowest latency path, the relayed path will be selected.

4.3 Ethernet Frame Forwarding

LEON forwards Ethernet frames using the lowest latency path selected using the process described in Section 4.2. LEON uses a forwarding database and the path selection results to forward Ethernet frames.

A forwarding database (FDB) is a database of hosts connected to the Layer 2 network. The FDB keeps track of which tunnel end point a host is connected to. From the FDB, the tunnel end point can determine which tunnel end point the Ethernet frame should be forwarded to. Each tunnel end point has an FDB and manages it.

The FDB is updated using incoming Ethernet frames from other tunnel end points of the Layer 2 network. When a tunnel end point receives an Ethernet frame from another tunnel end point, it checks the source host of the Ethernet frame and the source tunnel end point. If the source host is not registered in the FDB or if the Ethernet frame came from a tunnel end point which is different from the one on the FDB, the FDB is updated. This will keep the FDB updated on which tunnel end point a host is connected to.

When forwarding an Ethernet frame, the tunnel end point checks the destination MAC address of the Ethernet frame and decides how to handle it. If the destination MAC address is either a broadcast MAC address or a multicast MAC address, it is handled as a broadcast frame. If it's not either of them, the destination MAC address of the Ethernet frame is looked up in the FDB. If the destination MAC address is found in the FDB, it is handled as an unicast frame. If the destination MAC address is not found in the FDB, it is handled as a broadcast frame.

Unicast frames are encapsulated and forwarded to the destination tunnel end point using the lowest latency path. The tunnel end point identifies the destination tunnel end point by looking up the destination MAC address of the Ethernet frame in the FDB. Then, the shortest latency path to the destination is selected. After the path is selected, the Ethernet frame is encapsulated in an IP header and a LEON header. The LEON header contains the forwarding path information for the Ethernet frame. All the relaying tunnel end point's IP address and port number and the destination tunnel end point's IP address and port number is listed in the LEON header. Lastly, the encapsulated Ethernet frame is forwarded to the next hop of the selected path.

Broadcast frames are encapsulated and forwarded to all the tunnel end points. The tunnel end point selects each tunnel end point in the tunnel end point list as a destination. The shortest latency path to the destination is selected. Then, the Ethernet frame will be encapsulated and forwarded to the destination as it would do for unicast frames. This process is repeated for all the tunnel end points in the list.

When a tunnel end point receives an encapsulated Ethernet frame, it checks its LEON header. If the tunnel end point it is the destination tunnel end point, it will forward the Ethernet frame to the local Layer 2 network and updates FDB if necessary. If the tunnel end point is not the destination tunnel end point, it will relay the Ethernet frame to the tunnel end point listed after itself in the LEON header.

5 Evaluation

LEON forwards Ethernet frames using the lowest latency path. In cases where a path with latency lower than a direct Internet path exists, LEON will use the lower latency path to forward the Ethernet frame. By using LEON in such cases, the performance of the components improves, which allows faster processing of the requests to a service. This will result in better response time of a service.

For evaluation, we assumed WIDE Cloud as a service constructed on top of the Layer 2 network extended using LEON. One of the components in WIDE Cloud which is affected by high latency is the storage server. The storage server provides the storage area of the virtual machines to the hypervisor using the NFSv3 protocol. All the virtual machine's disk transactions are processed over the Layer 2 network. By using LEON, the performance of the storage server is expected to improve. This enables faster access to the data required to provide a service inside the virtual machine resulting in better response time of the virtual machine.

To evaluate this, we measured the following:

- File system performance of NFSv3
- Linux Kernel compile time inside a VM

First, we measured the file system performance of NFSv3. By using LEON in an environment with a lower latency path than the direct Internet path, LEON will forward Ethernet frames using the lower latency path. In such an environment, we expect to see an increase in performance of NFSv3 compared to using it directly over the Internet. We evaluate this by comparing the performance of NFSv3 when used over LEON and when used directly over the Internet.

Secondly, we measured the compile time of the Linux Kernel on a virtual machine using the same setup. If the performance of NFSv3 is improved, the disk access performance will improve, allowing faster processing of the compile request.

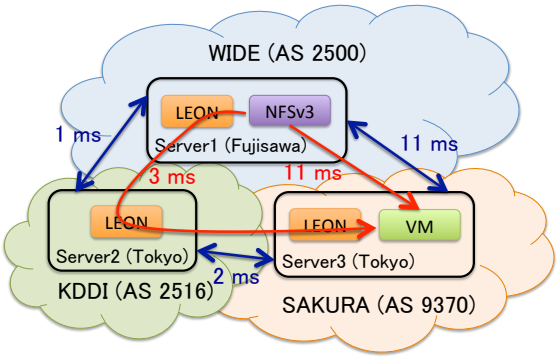

Fig. 5: Environment used for evaluation

Host	CPU	RAM	NIC
Server 1 (Fujisawa)	Intel Xeon L5520 2.27GHz (8 cores)	12GB	NetXtreme II BCM5709
Server 2 (Tokyo)	AMD Phenom 9550 2.20GHz (2 cores)	2GB	virtio
Server 3 (Tokyo)	Intel i7 870 2.93GHz (4 cores)	16GB	Intel 82574L

Table 1: Specification of the servers

5.1 Environment

For evaluation, we measured the performance of a component running on top of LEON and the performance of a service using that component. LEON forwards Ethernet frames via another location when the path relayed by another location has lower latency than the direct Internet path. As an evaluation, we evaluate whether the performance of a service and its component improves by using the relayed path. To evaluate this, the following are required in the evaluation environment:

- Measurement on the real Internet
- A relay path with lower latency than the direct Internet path

As an evaluation environment, we deployed a copy of WIDE Cloud with the components connected over a Layer 2 network extended using LEON. The topology

Host	OS	Kernel
Server 1 (Fujisawa)	Debian GNU/Linux 6.0.6 (Squeeze)	3.7.2
Server 2 (Tokyo)	Fedora 17	3.7.4-204
Server 3 (Tokyo)	Fedora 15	2.6.43.8-1

Table 2: Software versions of the servers

of the deployed evaluation environment is illustrated in Figure 5. We deployed 3 servers in 3 different different locations. All the servers are connected to different AS networks. The specifications of the servers are shown in Table 1. Also, the software versions used on the servers are shown in Table 2. Server 1 acts as the storage server and Server 3 acts as the hypervisor of WIDE Cloud. A virtual machine runs on Server 3 using the disk image located inside Server 1's storage area, which is exported using NFSv3.

A lower latency path than the direct Internet path exists in this evaluation environment. Server 1 is a server connected to WIDE Project's network (AS2500) [15]. Server 2 is a virtual machine provided by KDDI Web Communications [16]. It is connected to KDDI's network (AS2516). Server 3 is a server connected using EditNet's Internet connection service over NTT East's Flet's Hikari Next access line [17]. EditNet uses Sakura Internet (AS9370) [18] as their backbone [17]. Although all the servers are located in the same geographic region, since AS2500 and AS9370 are peered in Osaka, Japan, the latency between Server 1 and Server 3 is high. However, as AS2500 and AS2516 are peered in Tokyo, Japan and AS2516 and AS9370 are also peered in Tokyo, Japan, by having the traffic relayed by Server 2, the latency between Server 1 and Server 3 can be reduced. By running LEON on these 3 servers, this path will be detected and Ethernet frames from Server 1 to Server 3 will be forwarded via Server 2.

Using the 3 servers described above, we constructed an environment which meets the requirements.

5.2 Experimental Procedure

For evaluation, we measured the filesystem performance of NFSv3 and compile time of Linux Kernel inside a virtual machine. We used the environment described in Section 5.1 for the measurement. We ran the measurement in two different situations. One is a situation where storage server is directly mounted over the Internet. The second is a situation where storage server is mounted over a Layer 2 network extended using LEON. We compared the performance in these two situations.

We measured the filesystem performance of NFSv3 using Bonnie++ [19]. Bonnie++ is a filesystem benchmarking utility created by Russel Coker. We mounted Server 1's storage area on Server 3 using NFSv3. Then, we ran Bonnie++ on Server 3 to measure sequential access performance and random access performance of the exported storage area.

Next, we measured the compile time of Linux Kernel. We ran a virtual machine on Server 3 using Kernel-based Virtual Machine (=KVM) [20] as a hypervisor technology. The disk image of the virtual machine is stored on Server 1's storage area and the disk image is accessed using NFSv3. We compiled Linux Kernel inside a virtual machine and used the time command to measure compile time.

5.3 Experimental Results

First, we measured the filesystem performance of NFSv3. The measurement results of sequential access performance is shown in Table 3. Also, the measurement results of random access performance is shown in Table 4. Each measurement has been done 3 times and the results shown are the average values.

	read (KB/sec)	write (KB/sec)
Direct	4537	5372
LEON	4726	5198

Table 3: Performance of sequential access file operations in Bonnie++ over NFSv3

	seek (/sec)	create (/sec)	info (/sec)	delete (/sec)
Direct	2985	23	46	46
LEON	8417	65	132	132

Table 4: Performance of random access file operations in Bonnie++ over NFSv3

	Compile Time (min)
Direct	78.9
LEON	58.8

Table 5: Compile time of Linux Kernel

As we can see from the results of sequential access performance, there was a little difference between the two situations. In a situation where the storage server was directly mounted, the performance of sequential read access was 4537KB/sec and the performance of sequential write access was 5372KB/sec. On the other hand, in a situation where the storage server was mounted over a Layer 2 network extended using LEON, the performance of sequential read access was 4726KB/sec and the performance of sequential write access was 5198KB/sec. From these results, we can see that the performance of sequential access is not greatly effected by using LEON.

Meanwhile, from the results of random access performance, there was a large difference between the two situations. In a situation where the storage server was directly mounted, the performance of seek operations was 2985 ops/sec, the performance of create operations was 23 ops/sec, the performance of info operations was 46 ops/sec and the performance of delete operations was 46 ops/sec. On the other hand, in a situation where the storage server was mounted over a Layer 2 network extended using LEON, the performance of seek operations was 8417 ops/sec, the performance of create operations was 65 ops/sec, the performance of info operations was 132 ops/sec and the performance of delete operations was 132 ops/sec. From these results, we can see that by using LEON, we can achieve about 3 times the performance of the direct mount case in the evaluation environment.

Next, we measured the compile time of Linux Kernel inside a virtual machine with its disk image on the storage server. The results are shown in Table 5. In a situation where the storage server was directly mounted, it took 78.9 minutes to compile. On the other hand, in a situation where the storage server was mounted over a Layer 2 network extended using LEON, it took 58.8 minutes to compile. From the result, we can see that by using LEON, compile time was reduced by 20 minutes.

5.4 Discussion

From the results of NFSv3's filesystem performance, we saw an improvement in random access performance by using LEON. However, we could not see an improvement in sequential access performance. We assume that this is because the sequential access performance is dependent of throughtput and not the latency. In the evaluation environment, the Internet connection of the server connected using the Flet's Hikari Next access line was narrow compared to the other 2 servers. This Internet connection became a bottleneck in throughput and the sequential access performance did not improve. On the other hand, random access performance improved as the number of operation requests a server can send in a second has increased due to lower latency between the servers. From the results, we can say that the performance of a component has improved by using LEON.

Also, from the results of compile time of Linux Kernel, we saw an improvement in compile speed. We assume that there are many random access to the disk image during the compile process. As the random access performance improved, it enabled faster processing of the compiling process. This resulted in improvement of the compile time. From this result, we can say that the performance of a service has improved by using LEON.

From the experiments conducted, we can say that the response time of a service improves by using LEON. The performance of a component running on a Layer 2 network extended using LEON improves when there is a lower latency path than the direct Internet path. By using a lower latency path, the performance of latency dependent components inside a service improves. This enables faster processing of requests and as a result, the response time of a service improves.

6 Conclusions

In this research, we implemented and evaluated LEON, which is a multi-point Layer 2 network extension technology capable of extending a Layer 2 network to multiple geographically distributed locations on the Internet. On the Internet, there are many situations where a path relayed by another location has lower latency than the direct Internet path. LEON measures the latency between the locations and uses

the latency data to select the path with lowest latency. LEON uses the selected path for forwarding Ethernet frames. By using LEON, we saw an improvement in performance of latency dependent components in environments with a lower latency path than the direct Internet path. The improvement of the component's performance enabled faster processing of the request and improved service's response time.

References

[1] Zona Research. The Need For Speed II. *Zona Market Bulletin, Issue 05*, 2001.

[2] Akamai Press Release. Akamai and Jupiter Research Identify '4 Seconds' as the New Threshold of Acceptability for Retail Web Page Response Times. Press Release, November 2006. `http://www.akamai.com/html/about/press/releases/2006/press_110606.html`.

[3] Erik Nygren and Ramesh K. Sitaraman. The Akamai Network: A Platform for High-Performance Internet Applications. *ACM SIGOPS*, pages 2–19, July 2010.

[4] Huffington Post. Websites Scramble As Hurricane Sandy Floods Data Centers. News Paper Column, October 10, 2013.

[5] Akamai Technologies. `http://www.akamai.com/`, August 2013.

[6] PC World. Sprint-Cogent Dispute Puts Small Rip in Fabric of Internet, October 31, 2008. `http://www.pcworld.com/article/153123/sprint_cogent_dispute.html`.

[7] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A Survey of Network Virtualization. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, pages 862–876, April 2010.

[8] Hariharan Rahul, Mangesh Kasbekar, Ramesh Sitaraman, and Arthur Berger. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. Technical Report MIT-LCS-TR-1009, Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, 2005.

[9] WIDE Cloud Controller. `http://wcc.wide.ad.jp/`, accessed August 2013.

[10] B. Callaghan, B. Pawlowski, and P. Staubach. NFS Version 3 Protocol Specification. RFC 1813, IETF, June 1995.

[11] Peter Radkov, Li Yin, Pawan Goyal, Prasenjit Sarkar, and Prashant Shenoy. A Performance Comparison of NFS and iSCSI for IP-Networked Storage. *FAST*, pages 101–114, 2004.

[12] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer Two Tunneling Protocol "L2TP". RFC 2661, IETF, August 1999.

[13] 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. Internet Draft, IETF, May 2013.

[14] Luca Deri and Richard Andrews. N2N: A Layer Two Peer-to-Peer VPN. In *Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security: Resilient Networks and Services*, AIMS '08, pages 53–64, 2008.

[15] WIDE Backbone. `http://two.wide.ad.jp/`, accessed Auguest 2013.

[16] CloudCore VPS Service. `http://www.cloudcore.jp/vps/`, accessed August 2013.

[17] EditNet Operation Information. `http://intereddy.edit.ne.jp/disclosure-detail.php`, accessed August 2013.

[18] SAKURA Internet, Inc. `http://www.sakura.ad.jp`, accessed August 2013.

[19] Bonnie++: Disk and File System Performance Benchmark Utility. `http://www.coker.com.au/bonnie++/`, accessed August 2013.

[20] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori. kvm: the linux virtual machine monitor. In *Proceedings of the Linux Symposium*, volume 1, pages 225–230, 2007.