IEICE TRANS. COMMUN., VOL.E86–B, NO.12 DECEMBER 2003

PAPER

Cached Shortest-Path Tree: An Approach to Reduce the Influence of Intra-Domain Routing Instability

Shu ZHANG[†], **Katsuyoshi IIDA**[†], *and* **Suguru YAMAGUCHI**[†], ***Regular Members***

SUMMARY Because most link-state routing protocols, such as OSPF and IS-IS, calculate routes using the Dijkstra algorithm, which poses scalability problems, implementors often introduce an artificial delay to reduce the number of route calculations. Although this delay directly affects IP packet forwarding, it can be acceptable when the network topology does not change often. However, when the topology of a network changes frequently, this delay can lead to a complete loss of IP reachability for the affected network prefixes during the unstable period. In this paper, we propose the Cached Shortest-path Tree (CST) approach, which speeds up intra-domain routing convergence without extra execution of the Dijkstra algorithm, even if the routing for a network is quite unstable. The basic idea of CST is to cache shortest-path trees (SPTs) of network topologies that appear frequently, and use these SPTs to instantly generate a routing table when the topology after a change matches one in the caches. CST depends on a characteristic that we found from an investigation of routing instability conducted on the WIDE Internet in Japan. That is, under unstable routing conditions, both frequently changing Link State Advertisements (LSAs) and their instances tend to be limited. At the end of this paper, we show CST's effectiveness by a trace-driven simulation.

key words: *intra-domain, routing instability, convergence time, link-state routing protocol, OSPF, Dijkstra algorithm*

1. Introduction

The Internet has greatly increased its influence during the last decade. Nowadays, people use the Internet to obtain information from the WWW, send e-mails, exchange files, and so on. The Internet has become an indispensable tool in our daily lives. However, the current Internet is not perfect. Compared with the traditional Public Switched Telephone Network (PSTN), the Internet excels at its ability to provide a variety of services, but it lacks reliability. End users often find that sometimes they must face extremely poor performance while using the Internet. There are many factors that lead to poor performance, including link bandwidth, the efficiency of application software and protocol robustness. In this paper, we focus on the influence of routing instability because it directly affects the efficiency of IP packet forwarding.

Link-state routing protocols have been used for years because of their capability to achieve relatively fast routing convergence and flexible routing. Generally, link-state routing protocols use the Dijkstra algorithm to calculate shortest-path trees (SPTs). The Dijkstra algorithm is classified as an $O(n^2)$ algorithm and requires many CPU cycles when used in large-scale networks (usually with hundreds of nodes). For this reason, implementors often introduce an artificial delay to reduce the number of route calculations. However, because all routers schedule their route calculations with different timing, such a delay may lead to packet loss and routing loops due to inconsistent routing tables. If the network topology does not change frequently, for example, if it only occurs several times a day, the inconsistency is generally considered to be acceptable because it lasts only on the order of seconds. But when the network topology changes frequently and persistently, the total time of inconsistency amounts to minutes or even hours. Under these conditions, the performance of applications will be extremely poor with respect to data transmission. Consequently there are great demands to ameliorate this situation.

One resolution for this problem is to reduce the number of routers in one area by splitting the whole routing domain into more areas and reduce the time interval of route calculations to the order of milliseconds. However, since the Internet is still expanding at a great rate, it is difficult to always maintain a small number of routers in an area. In addition, using more areas for a routing domain introduces extra network operation costs, thus making this approach difficult to be adopted by most ISPs.

In this paper, we first present the results of a case study on intra-domain routing to show how frequently routing instability can occur on a network used daily. Based on the analytical results, we propose the Cached Shortest-path Tree (CST) approach, which not only speeds up intra-domain routing convergence, reduces packet loss, but also decreases router loads. Although CST can also be applied to other link-state routing protocols, in this paper most of our discussion is focused on OSPF [1].

So far, there has been significant research on the issues of routing instability and route calculation for link-state routing protocols. An analysis of how routing messages can be lost in a congested network is shown in [2]. In [3], the synchronization of periodic routing messages and the avoidance of inadvertent synchronization are focused. In [4], a mechanism is proposed to achieve

Manuscript received December 10, 2002.
Manuscript revised March 31, 2003.

[†]The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630-0192 Japan.

faster routing convergence by counting the frequency of changes of the advertised routing information for each network node. In [5], [6], some dynamic SPT algorithms are proposed to reduce the overhead of route calculation. We will show why it is difficult to adopt these algorithms in routing software in Sect. 2.4.

We have organized this paper as follows: in Sect. 2 we present our findings on intra-domain routing instability during an investigation conducted on the WIDE [7] Internet. In Sect. 3 we explain the proposed CST algorithm, discuss how to implement it efficiently and show our simulation results. Finally, we conclude our work in Sect. 4.

2. An Investigation on Intra-Domain Routing

In this section, we present the results of an intra-domain routing investigation that we conducted on the WIDE Internet. After introducing relevant information about OSPF, we show how frequently intra-domain routing instability occurs on the WIDE Internet by analyzing the OSPF data we collected. In the end, we discuss the problem of route calculation of link-state routing protocols.

2.1 Related Information

We chose the WIDE Internet, the largest academic network in Japan, as the target network for measurement. The WIDE Internet consists of about 300 routers (in the routing domain) and connects hundreds of organizations. Because the WIDE Internet uses OSPF as its main routing protocol, we used the OSPF routing messages for our analysis. We collected raw data from a network being utilized by many organizations but not from an experimental network because we thought the results of a real workload would be more convincible.

Currently, the following five kinds of packets are defined in the OSPF specifications:

- Hello packet: Hello packets are sent periodically on all interfaces in order to establish and maintain neighbor relationships.
- Database Description packet: Database Description packets are exchanged when an adjacency is being initialized. They describe the contents of the link-state database (LSDB).
- Link State Request packet: Link State Request packets are used to request the portions of a neighbor's database that are more up-to-date.
- Link State Update (LSU) packet: Link State Update packets are used to flood Link State Advertisements (LSAs). Each LSU packet carries a collection of LSAs one hop farther from their origin.
- Link State Acknowledgment packet: Link State Acknowledgment packets are used to explicitly acknowledge the received LSAs in order to make the flooding of LSAs reliable.

Although we collected all five of these kinds of OSPF packets, we only analyzed the LSU packet for this paper because it is the only kind of packet which includes routing information concerning route calculations.

The LSA is used to disseminate the routing information of each network node. In the OSPF specification, the following five kinds of LSAs are defined:

- Router-LSA: The router-LSA is the Type 1 LSA. In an OSPF routing domain, each router originates a router-LSA for the area to which it belongs. The router-LSA describes the collected states of the router's links attached to the area and is flooded throughout the specific area. By analyzing all router-LSAs, we can learn the changes of link states for all routers in an area.
- Network-LSA: The network-LSA is the Type 2 LSA. It is used to describe the states of broadcast networks or non-broadcast multi-access (NBMA) networks. The network-LSA is originated by the designated router (DR) of the network. As is true for the router-LSA, a network-LSA is specific to one area. So, by analyzing network-LSAs, we can monitor routers that form or break OSPF adjacency in a network.
- Summary-LSA: The summary-LSAs are Type 3 and Type 4 LSAs. They are originated by area border routers and are used to describe inter-area destinations. Type 3 summary-LSA is used when the destination is an IP network and Type 4 summary-LSA is used for the destination of the Autonomous System (AS) boundary router.
- AS-external-LSA: The AS-external-LSA is the Type 5 LSA. This LSA is originated by AS boundary routers and is used to describe destinations external to the AS.

Because OSPF uses all these LSAs to calculate the routing table, by monitoring these LSAs we can figure out to what extent the routing instability is occurring.

OSPF divides the entire network into a number of areas to achieve hierarchical routing. Because the backbone area is generally the most crucial part of a network, the analysis in this paper is based on the backbone area of the WIDE Internet, which consists of approximately 50 backbone routers.

We collected OSPF routing messages at the NARA-NOC of the WIDE Internet, located at the Nara Institute of Science and Technology in Ikoma, Japan. We placed a FreeBSD box on a 100Base-T ethernet which is configured as the backbone area. The tool we used to collect data is *tcpdump* [8], a widely used tool for collecting traffic over shared links such as ethernet and FDDI. In order to facilitate the analysis, we slightly modified the tool so that it could record data on a daily basis.

The data collection began in August 2000 and is

still being conducted. The results we are going to show are based on data from October 2001 to August 2002 (minus 13 days of data which was lost due to disk failure). The collected data of this period amounts approximately to 6.8 GB at an average of 21.1 MB per day.

We analyzed all the data by *ospfanaly*, a tool we wrote in C language. *Ospfanaly* uses *libpcap* [8] to read data recorded by *tcpdump* and outputs the changes of each LSA. We also wrote some Perl scripts for use in the data processing.

2.2 Statistical Results of the Investigation

In this section we show the statistical results of the OSPF LSAs and summarize some oscillation patterns of these LSAs.

2.2.1 Router-LSA

The number of router-LSAs that appeared in the backbone area each day ranges from 43 to 54. In general, because there should not be many changes in network topology, we had thought that these routers would not originate many changing router-LSAs. However, during our investigation, we found just the opposite to be true. Figure 1 is the graph showing the total number of router-LSAs and their changes each day. From this graph we can see that, although in most days the total number of changes is not big, there did exist days in which a lot of changes occurred and sometimes the number of changes in a single day reached 2,000 times.

If these changes are originated by most of the backbone area routers, we may consider them as normal topology changes due to network maintenance. But, after our analysis, we found that this also was not the case: the changes tend to be originated by only a few routers. For example, on November 20, 2000, a total of 11,380 changes occurred, but 98.6% of these changes were due to two router-LSAs. On April 22, 2001, 1093 of 1097 changes were caused by only one router-LSA.

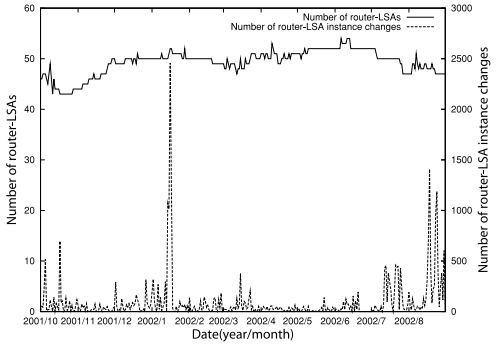

Fig. 1 Number of router-LSAs and their instance changes in the backbone area.

2.2.2 Network-LSA

Figure 2 shows the number of network-LSAs and their changes each day. In total, there are 8-17 network-LSAs that appeared in the backbone area of the WIDE Internet during the 11-month period. Although the network-LSA does not change as frequently as the router-LSA, we still found that on some days the network-LSA can oscillate for hundreds of times.

2.2.3 Oscillation Pattern

We classify the observed LSA changes into two main categories by the characteristics of the changes.

1. Changes of broadcast and NBMA network interfaces
   When a router on a broadcast or NBMA network finds that other OSPF routers exist on the same link, it describes this network as a transit network (Type 2) in its router-LSA. Otherwise, this network is treated as a stub network (Type 3).
   We show the typical changes of a broadcast interface during a 10-minute period in Fig. 3. During this period, the router-LSA changes a total of 32 times, or 3-4 times per minute.
   Figure 4 is part of the output generated by *ospf-*

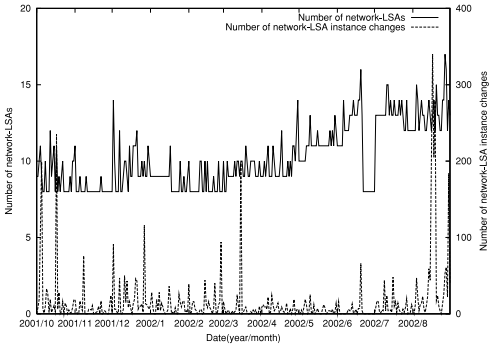

Fig. 2 Number of network-LSAs and their instance changes in the backbone area.

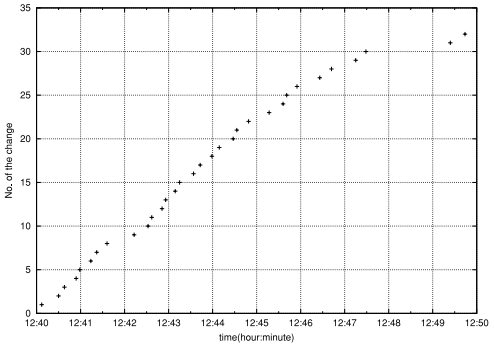

Fig. 3 Typical changes of broadcast and NBMA network interfaces.

```
12:42:32 +link=203.178.137.64 type=3
         -link=203.178.137.77 type=2
12:42:37 +link=203.178.137.77 type=2
         -link=203.178.137.64 type=3
12:42:51 +link=203.178.137.64 type=3
         -link=203.178.137.77 type=2
12:42:56 +link=203.178.137.77 type=2
         -link=203.178.137.64 type=3
12:43:09 +link=203.178.137.64 type=3
         -link=203.178.137.77 type=2
12:43:15 +link=203.178.137.77 type=2
         -link=203.178.137.64 type=3
12:43:34 +link=203.178.137.64 type=3
         -link=203.178.137.77 type=2
```

Fig. 4 Changes of the broadcast and NBMA network interface.

analy for the 10-minute period. '+' indicates the addition of a link compared with the last instance of the LSA and '-' indicates the opposite. We can see that most of the oscillations are the repeated declarations of interface up or down.

2. Changes of point-to-point network interfaces
   In the current Internet, point-to-point connections are often used to connect distant places. When a router detects a link-down or does not receive its peer's hello packet within a certain time (RouterDeadInterval) over a point-to-point interface, it originates a new LSA, in which the point-to-point link is removed. Then, the router floods this new LSA to tell other routers (in the same area) that the link has become unavailable. This kind of change is different than that of the broadcast or NBMA networks in that the router on the other side will also detect the failure and originate a new LSA. Thus, when a point-to-point connection fails, we see two LSA changes originated by the two endpoints at about the same time. In our investigation, this kind of change was also frequently observed.
   Figure 5 shows an example of the typical changes of point-to-point interfaces in a 30-minute period. Figure 6 is the output of *ospfanaly* for that period.

Because the results of the other two kinds of intra-domain LSAs (network summary-LSAs and ASBR summary-LSAs) are similar to the results of the router-LSAs and network-LSAs, we show them in Appendix, but not here. For further results of our routing instability investigation, please refer to [9].

2.3 The Similarity of the WIDE Internet with Commercial Networks

Although we conducted our investigation only on the WIDE Internet, we think that similar phenomena can also be observed in commercial networks. Compared with commercial networks, the WIDE Internet has the following similarities:

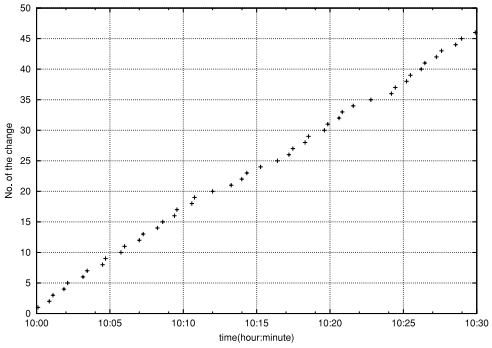

Fig. 5 Typical changes of the point-to-point network interface.

```
10:01:52 -link=203.178.136.22 type=1
10:02:07 +link=203.178.136.22 type=1
10:03:10 -link=203.178.136.22 type=1
10:03:27 +link=203.178.136.22 type=1
10:04:30 -link=203.178.136.22 type=1
10:04:42 +link=203.178.136.22 type=1
10:05:45 -link=203.178.136.22 type=1
10:06:00 +link=203.178.136.22 type=1
10:07:00 -link=203.178.136.22 type=1
10:07:16 +link=203.178.136.22 type=1
10:08:14 -link=203.178.136.22 type=1
```

Fig. 6 Changes of the point-to-point network interface.

- WIDE Internet is a large-scale network.
  As we stated in Sect. 2.1, the WIDE Internet connects hundreds of organizations and the number of users is almost one million. So, we consider the scale of this network to be similar to commercial networks.
- WIDE Internet is well-administrated.
  Although the WIDE Internet is an academic network, it operates under the supervision of many network experts. These experts include network researchers, network engineers and graduate students who are doing research on the network. So, we believe that the WIDE Internet is administrated as well as commercial networks.

2.4 The Problem of Route Calculation of Link-State Routing Protocols

Currently, the route calculation procedure of OSPF can be divided into the following steps:

1. Calculating SPTs
   In this step, a router calculates the SPTs for all areas to which it is associated. Router-LSAs and

network-LSAs are used in this step. Based on the calculated SPTs, the router generates intra-area routes.

2. Calculating inter-area routes
   In this step, a router examines all of its summary-LSAs to calculate inter-area routes.
3. Calculating AS external routes
   In this step, a router examines all of the AS-external-LSAs to calculate the routes to networks in other ASes.

In all three steps, the calculation of the SPT for an area is the most complex. At present, most link-state routing protocols use the Dijkstra algorithm to calculate the SPT. Because the Dijkstra algorithm is an $O(n^2)$ ($O(n * \log(n))$ for heap Dijkstra algorithm) procedure, most implementations introduce an artificial delay to avoid using too many CPU cycles. To our knowledge, the following two kinds of delay are being used:

- SPF Delay: The time that a router needs to wait after receiving an LSA with different content until the next calculation.
- SPF Hold-Time: The time that a router needs to wait after the previous calculation until the next calculation.

Table 1 lists the values of the SPF Delays and SPF Hold-Times used by some of the most popular OSPF implementations [4].

Because of the delay introduced into route calculation, when receiving a changed LSA, routers usually have to wait for several seconds before they can revise their routing tables and start forwarding packets based on them. During this period, routers either simply drop the packets with destinations to the affected network prefixes or send them in the wrong direction. When the topology of a network is relatively stable and there are not many LSA changes, the duration of inconsistent routing can be considered acceptable because it does not last very long. However, with the statistical results we discussed in Sect. 2.2, we know that sometimes the network tends to be unstable and the network topology changes frequently and persistently. Under such conditions, the duration of route inconsistency reaches an unacceptable level and we need to reduce the inconsistent time in order to further improve network reliability.

Table 1 The two kinds of delay used on some popular implementations of OSPF.

Implementation	SPF Delay (s)	SPF Hold-Time (s)
Cisco	5	10
Foundry	5	10
Fujitsu	5	10
Extreme	Unknown	3
River Stone	Unknown	5
Hitachi (Gated)	0-5	5
Zebra	5	10
Gated	0	5

Several dynamic SPT algorithms [5], [6] are proposed to speed up the calculation of SPTs. However, it is difficult for vendors to adopt these algorithms for the following reasons:

- Each dynamic algorithm needs to determine how the network topology has changed before computing the updated SPT. In OSPF, because the attached links in the router-LSA or the attached routers in the network-LSA are not sorted, the comparison of a newer LSA instance with a previous instance is a big task if the number of attached links or routers is relatively large.
- The efficiency of the dynamic algorithms depends on the change in the topology. For example, if the states of a large number of nodes or links change, the execution of dynamic algorithms could be very inefficient.
- Dynamic algorithms tend to be complex and are difficult for the manufacturer to implement.

To the best of our knowledge, none of these dynamic SPT algorithms have been implemented in any commercial or non-commercial routing software so far.

3. CST Algorithm

In this section, we discuss the details of the CST algorithm, which can not only speed up intra-domain routing convergence but also reduce router loads, even under unstable routing conditions. After that, we describe the procedures of implementation, discuss the issues that need to be considered when implementing CST and show the evaluation results.

3.1 Design

The basic idea of CST can be divided into 2 steps:

- Cache a calculated SPT and the corresponding LSA set which consists of router-LSAs and network-LSAs.
- When the LSA set of a new network topology matches one in the caches, instantly create a new routing table based on the cached SPT.

Through the typical changes of LSA we listed in Sect. 2.2.3, we can see that, although an LSA can change frequently, it tends to vary in limited instances. For the most time it just repeats declaring the up/down status of a single interface or link. Further study tells us that this characteristic is especially outstanding when there are frequent and persistent LSA changes.

Because the SPT is determined by the set of router-LSAs and network-LSAs, it is not difficult to imagine that the instances of all the SPTs are also limited. So by caching all frequently appearing SPTs, it is possible to bypass many executions of the Dijkstra

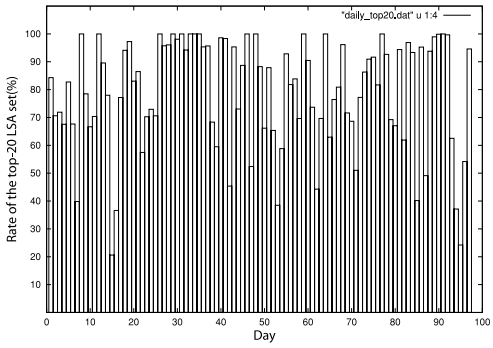

Fig. 7 Percentage of the top-20 frequently appearing instances of SPTs.

algorithm by simply using the cached SPT when doing route calculations.

Figure 7 shows the percentage of the top-20 frequently appearing SPTs found in all SPTs for each day. Here we only show data for the days in which the SPT was calculated more than 50 times. We can see that on most days, the top-20 instances comprise more than 70% of the whole number. In total, 20,213 out of 23,810 (about 85%) are due to the 20 most frequently appearing instances.

Consequently, we do not need to record all SPTs because it requires too much memory; we only need to record the SPTs of frequently appearing LSA sets. To decide whether an LSA set is a frequently appearing one, the following two factors are considered separately:

- The cache's age: If an LSA set has appeared at least once in the last 12 hours, its SPT has a higher priority for continued caching than those which did not appear in the same period.
- The cache's hit number: If the SPT of an LSA set has a higher hit number than another one, it has a higher priority for continued caching.

In our design, the age factor is considered before the hit number because we found that frequently appearing LSA sets tend to change on different days.

To implement CST, two kinds of changes should be made in the current procedure of route calculation: the cache and the search for an SPT.

3.1.1 The Cache of an SPT

When a new SPT is calculated by normal execution of the Dijkstra algorithm, check whether the maximum cache number (this should be a parameter configurable by the network operator) has been reached.

- If the maximum cache number has not been reached, cache the SPT and the corresponding LSA set.
- Otherwise, cache the SPT and the corresponding LSA set after removing an old or less frequently appearing cached SPT as well as its LSA set based on the criteria described in the last section.

3.1.2 The Search of an SPT

When receiving an OSPF LSU packet, compare all of the included LSAs with those in the LSDB.

- If there are new LSAs, obsolete LSAs, or LSAs whose content is different than the ones in the LSDB, update the LSDB. Check whether the present LSA set is the same as any in the caches.
  - If the LSA set and its SPT are already cached, instantly generate a new routing table based on the cached SPT.
  - If the present topology is not cached, wait for the next LSU packet.
- If there are no changed LSAs (including new and obsolete LSAs) in the LSU packet, wait for the next LSU packet.

3.2 Implementation Issues

Because one of our goals is to replace the redundant execution of the Dijkstra algorithm with the switching of a cached SPT, we must make sure that the procedure of finding the right SPT is much simpler than the Dijkstra algorithm. To achieve this, the following three issues need to be considered during the implementation of CST:

- Cache an LSA set efficiently
  Since we need to compare the LSA sets when looking up the cache of an SPT, recording the LSA set for an SPT is necessary. However, recording all LSAs of an LSA set as well as their contents will not only take considerable memory, but will also make the comparison of LSA sets a big task when searching for a cache. We recommend saving a relatively stable LSA set (SLS) and only recording the different parts (from the SLS) of each LSA set. In our design, SLS is defined as follows:
  1. (For the router that boots up) The LSA set immediately after the LSDB exchange.
  2. (For the router in operation) The LSA set which does not change within a certain time (the default is 10 minutes).

  The differences between an LSA set and the SLS can be structured by the following three lists:
  - New list: Record new LSAs which are not included in the SLS.
  - Change list: Record LSAs which are included in the SLS but whose contents have changed.
  - Obsolete list: Record LSAs which are included in the SLS but declared obsolete.

- Search SPT caches quickly
  When looking for an SPT in the caches, comparing the present LSA set with the ones in the caches can be a big task if there are many LSAs in the three lists (new list, change list, obsolete list). For this reason, we recommend using a hash number for each LSA set. The hash number should be generated when the SPT is cached. Thus, we can speed up the search by first comparing the hash number of the two LSA sets and only perform further comparison when the hash numbers are identical.
- Restructure the caches efficiently when the SLS changes
  Because for all LSA sets in the caches only the differences from the SLS are saved, when the SLS changes, the differences of the LSA sets need to be restructured. We recommend dividing this procedure into the following two steps:

  1. Compare the present SLS with the previous one.
  2. Apply the difference of the two SLSes to each saved LSA set.

  Because the first step needs to compare all the LSAs in the two SLSes, and the number of recorded LSAs can be relatively large, we recommend recording all LSAs in a sorted order in advance so that we can limit the complexity of this step to $O(n)$, where n is the number of all LSAs in an SLS. For the second step, the respective comparison of all LSAs in the new list, change list and obsolete list is necessary. Since the numbers of LSAs in the three lists are relatively small, both sorted LSA sets and unsorted sets are suitable.

3.3 Evaluation

We evaluate CST by *ospfsim*, a tool we wrote in C language.

3.3.1 The Effectiveness of CST

In Fig. 8, we show the number of executions of the Dijkstra algorithm needed by a router for both traditional and CST-enabled implementations. We also show the total hit numbers of the cached SPTs in the same figure. To simplify matters, we assume the SPF Delay is 0 and the SPF Hold-Time is 10 seconds. The number of caches used in this simulation is 20.

We can see from this figure that, for most months, the number of executions of the Dijkstra algorithm are greatly reduced in the CST-enabled implementation. This trend is especially outstanding in the months when there were frequent LSA changes, such as January 2002, July 2002 and August 2002. The hit numbers of the cached SPTs in these months are very high, too. This indicates that, while CST works well when the routing is relatively stable, it is especially effective under conditions of unstable routing. There are also months (such as December 2001) in which the effect of the CST approach is not as outstanding as other months. This is because there are more instance numbers of network topology in these months than in other months. But, overall, we can say that CST is an effective approach.

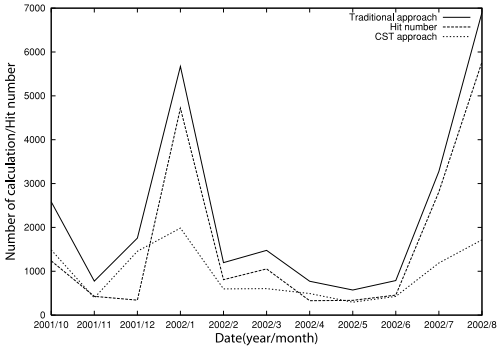

Fig. 8 Number of executions of Dijkstra algorithm in traditional and CST-enabled route calculations.

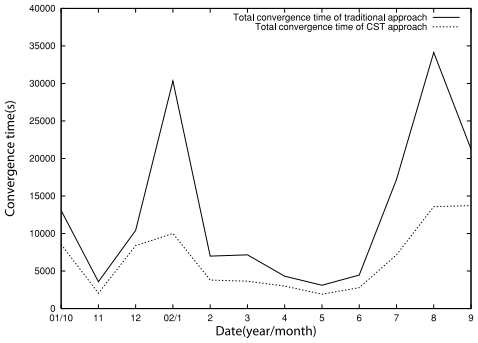

Fig. 9 Monthly convergence time of traditional case and CST-enabled case.

3.3.2 Convergence Time

Figure 9 shows the monthly convergence time of the traditional case and the CST-enabled case, respectively. The cache number used is 20, which is the same as that in the previous simulation. We can see that the convergence time is greatly reduced in the months when the routing tends to be unstable.

Figure 10 shows the daily convergence time in August 2002, when the CST effects were outstanding. We can see that, while in some days the convergence time is largely reduced, there are also days in which the convergence time of the CST approach does not change much compared with the traditional approach. This is because of the diversity of SPTs for those days.

3.3.3 Cache Number

The cache number is the number of SPT caches for

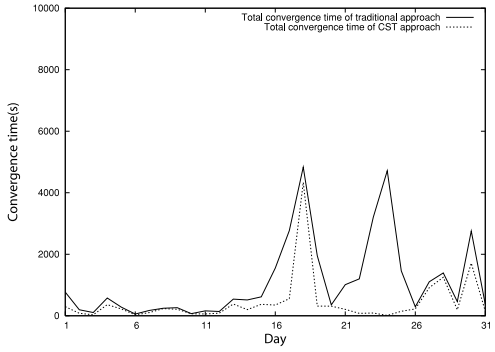

Fig. 10 Daily convergence time of traditional case and CST-enabled case.

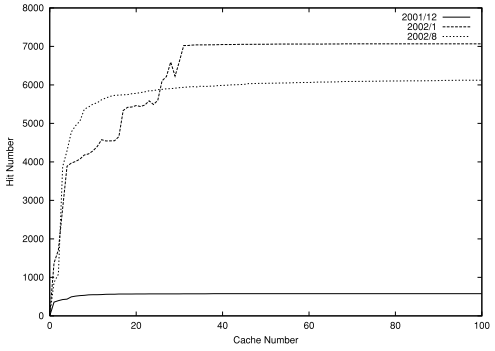

Fig. 11 The number of SPT caches and the cache hit number.

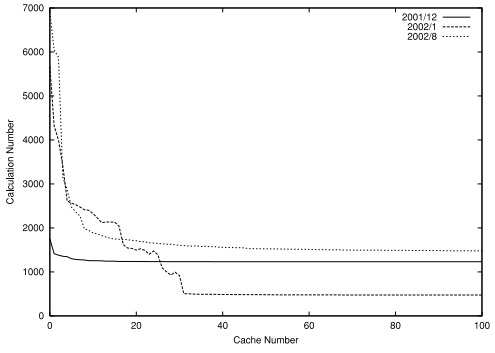

Fig. 12 The number of SPT caches and the number of route calculations.

which a router enabling CST needs to allocate memory. In essence, when the cache number increases, the number of SPT calculation decreases and the hit number increases. However, because the memory of a router is limited, it is not realistic to select a large number as the maximum cache number. Here, we show how the cache number affects the efficiency of CST.

In Fig. 11 and Fig. 12, we can see that in January 2002 and August 2002, when the cache number is around 4 or 5, the number of SPT calculations decreases most steeply and the cache hit number increases most sharply. In these two months, we can also see that when the cache number is more than 30, the effect of

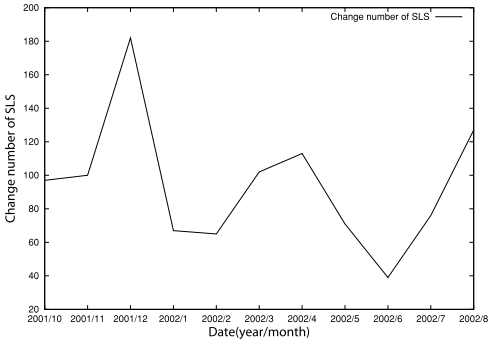

Fig. 13 The change number of SLS.

the increasing cache number becomes less clear. December 2001 is a month in which the CST effect is not as outstanding as January 2002 and August 2002. We can see that in December 2001, the number of route calculations and the cache hit number change much more slowly.

3.3.4 Memory

In this section we evaluate the amount of memory needed by SPT caches. In Zebra, a route entry in the SPT costs 40 bytes for IPv4 and 52 bytes for IPv6. So, for a router running OSPF on IPv4, the necessary memory can be calculated by

$$M = n * (40 * (r + s)) \text{ bytes},$$

where n is the number of SPT caches, and r, s are the numbers of router-LSAs and network-LSAs, respectively.

For the WIDE Internet (48 router-LSAs and 12 network-LSAs assumed), if we use 5 as the maximum SPT cache number, CST will cost $5 * (40 * (48 + 12) = 12,000$ bytes. If we use 100 as the maximum SPT cache number for a routing domain with 500 nodes and 200 networks, it costs $100 * (40 * (500 + 200)) = 2.8$ Mbytes, which is not very much for a router used in a large-scale network.

3.3.5 SLS

The criterion we use to define the SLS is this: while each cached LSA set should not differ much from the SLS, we must also make sure that the SLS does not change often, so that there will not be much restructuring needed. Figure 13 shows the change number of SLSes in each month from October 2001 through August 2002. We can see that, with the definition in Sect. 3.2, SLS does not change often.

3.4 Applying CST to Other Link-State Routing Protocols

All link-state routing protocols have the following two characteristics:

1. The router has its own LSDB, which holds the states of all other routers in the same area.
2. The router uses the Dijkstra algorithm to compute the SPT.

So, it is possible to apply CST to other link-state routing protocols. Here, we briefly show how to enable CST on IS-IS [10], [11], another widely used intra-domain routing protocol.

IS-IS was developed by OSI to facilitate the interconnection of open systems. IS-IS is different than OSPF in that IS-IS uses LSP (the link-state protocol data unit) to constitute its LSDB. So, by caching LSP sets that appear frequently and their corresponding SPTs, it is possible to quickly construct the routing table in a way similar to OSPF when the network topology after a change matches an LSP set in the caches.

4. Conclusion

In this paper, we first presented the results of an investigation on the WIDE Internet to show how frequently routing instability can occur in a network that is used daily. We found that, although most users do not notice, routing instability can happen frequently on a well-administrated network. Such instability not only directly harms IP reachability, but it also increases router loads, thus making additional instability more likely to occur. We then proposed the CST approach, which can speed up intra-domain routing convergence without extra execution of the Dijkstra algorithm. The basic idea of CST is to cache SPTs that appear frequently, and instantly generate a new routing table based on the cached SPT when the network topology after a change matches one in the caches. From our evaluation, we can see that CST is effective most of the time for the WIDE Internet, and the effect is especially outstanding when the routing tends to be unstable.

CST depends on a characteristic of networks under unstable routing conditions. That is, although the LSAs change frequently, most of the resulting LSA sets are limited to some frequently appearing ones. We confirmed this characteristic on the WIDE Internet and believe it should also apply to commercial networks.

Because the essence of our approach is to replace the redundant executions of the Dijkstra algorithm by directly using the cached SPTs when calculating routes, it is important to make sure that the work of finding and switching a cached SPT does not need as many CPU cycles as is needed for the execution of the Dijkstra algorithm. We discussed all of the main issues that need to be considered when implementing CST and showed that the overhead of implementing CST can be minimized with the introduction of SLS, the use of hash numbers, and so on. Although the discussion in this paper focused on OSPF, CST can also be applied to other link-state routing protocols, such as IS-IS.

References

[1] J. Moy, "OSPF version 2," RFC 2328, April 1998.
[2] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma, "Routing stability in congested networks: Experimentation and analysis," Proc. ACM SIGCOMM'00, pp.163–174, 2000.
[3] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," IEEE/ACM Trans. Networking, vol.2, no.2, pp.122–136, 1994.
[4] N. Yoshikawa, A mechanism for faster convergence on link state routing protocol, Master's Thesis, Nara Institute of Science and Technology, 2002.
[5] P.G. Franciosa, D. Frigioni, and R. Giaccio, "Semi-dynamic shortest paths and breadth-first search in digraphs," Symposium on Theoretical Aspects of Computer Science, pp.33–46, 1997.
[6] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Incremental algorithms for single-source shortest path trees," Proc. Foundations of Software Technology and Theoretical Computer Science, pp.113–224, Dec. 1994.
[7] Widely Integrated Distributed Environment Project, http://www.wide.ad.jp.
[8] http://www.tcpdump.org.
[9] S. Zhang, Y. Kadobayashi, and S. Yamaguchi, "An analysis on intra-domain routing instability of WIDE Internet," WIT2001, Sept. 2001.
[10] D. Oran, "OSI IS-IS intra-domain routing protocol," RFC1142, Feb. 1990.
[11] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments," RFC1195, Dec. 1990.

Appendix: Statistical Results of Summary-LSAs

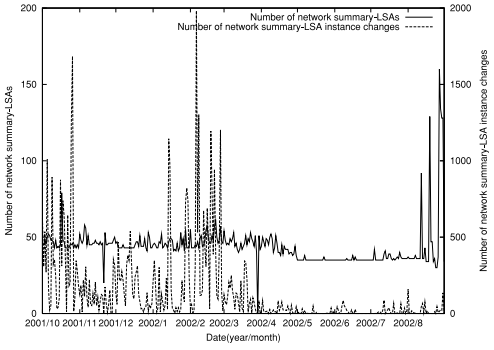

Fig. A· 1 Number of network summary-LSAs and their instance changes in the backbone area.

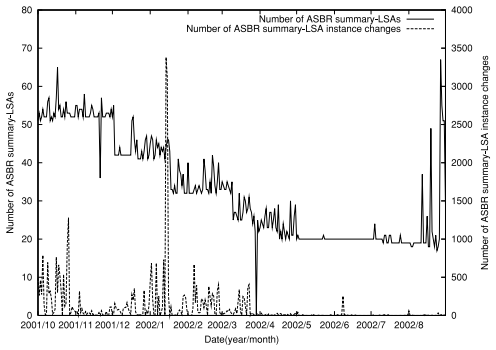

Fig. A· 2 Number of ASBR summary-LSAs and their instance changes in the backbone area.

Shu Zhang received the B.E. degree from Waseda University in 1996, and the M.E. and Ph.D. degrees from Nara Institute of Science and Technology in 1999 and 2003, respectively. He joined the Communications Research Laboratory in April 2003. His research interests include routing, MPLS and network management. He is a member of the WIDE Project.

Katsuyoshi Iida received the B.E., M.E., Ph.D. degrees in Computer Science and Systems Engineering from Kyushu Institute of Technology (KIT), Iizuka, Japan in 1996, in Information Science from Nara Institute of Science and Technology (NAIST), Ikoma, Japan in 1998, and Computer Science and Systems Engineering from KIT in 2001, respectively. Since Oct. 2000, he has been an Assistant Professor in the Graduate School of Information Science, NAIST. Beginning in Sept. 2001 and ending in Aug. 2002, he spent seven months in total at the Department of Information and Computer Science, University of California, Irvine, on leave of absence from NAIST. His research interests include performance evaluation of networking systems, Internet telephony and mobile networks. Dr. Iida is a member of the WIDE project in Japan, Association for Computing Machinery (ACM), and the Institute of Electrical and Electronics Engineers (IEEE). In 2003, he received the 18th TELECOM Sytem Technology Award, the Telecommunications Advancement Foundation, Japan.

Suguru Yamaguchi received the M.E. and D.E. degrees in computer science from Osaka University, Osaka, Japan, in 1988 and 1991, respectively. From 1990 to 1992 he was an Assistant Professor in Education Center for Information Processing, Osaka University. From 1992 to 1993, he was with Information Technology Center, Nara Institute of Science and Technology, Nara, Japan, as an Associate Professor. From 1993 to 2000, he was with Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan, as an Associate Professor. Currently, he is a Professor with the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. He has been also a member of WIDE Project, since its creation in 1988, where he has been conducting research on network security system for wide area distributed computing environment. His research interests include technologies for information sharing, multimedia communication over high speed communication channels, network security and network management for the Internet.