NAIST-IS-DT9961032

Doctoral Dissertation

Studies on Intra-domain Routing Instability

Shu Zhang

March 5, 2003

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Shu Zhang

Thesis Committee:	Suguru Yamaguchi, Professor
	Hideki Sunahara, Professor
	Youki Kadobayashi, Associate Professor
	Kazutoshi Fujikawa, Associate Professor

Studies on Intra-domain Routing Instability[*]

Shu Zhang

Abstract

Routing instability means the route change or disappearance due to unexpected network trouble. When routing instability happens, IP packets are either forwarded to the wrong direction or simply discarded, and sometimes it leads to routing loops, which seriously waste network resources. As it greatly degrades the efficiency of data transmission over the Internet when occurring frequently and persistently, there are great demands to explore this pathological phenomenon and reduce the bad influence that it brings to the Internet.

In this thesis we first present the result of a routing instability investigation on WIDE Internet, the largest academy network in Japan, to show how frequently routing instability can occur on a daily-used network. We collect the data of OSPF, the intra-domain routing protocol used on the backbone of WIDE Internet, for a period of about two years and analyze the data. The result tells us that although most end-users do not notice, routing instability can occur frequently. We characterize the instability and also present some reasons that lead to such frequent instability.

In the next part of this thesis, we discuss how to minimize the influence of routing instability. Currently, as most of Internet Service Providers(ISP) use link-state routing protocols to do intra-domain routing and link-state routing protocols calculate routes using Dijkstra algorithm, which suffers scalability problem, it is often the case that implementors introduce artificial delay to reduce the number of route calculation. When routing instability occurs frequently and persistently, this delay can lead to complete loss of IP reachability for the affected network prefixes during the unstable period. In order to ameliorate this situation, we

[*]Doctoral Dissertation, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9961032, March 5, 2003.

propose Cached Shortest-path Tree (CST) approach which quickens intra-domain routing convergence without extra execution of Dijkstra algorithm even if the routing for a network is quite unstable. The basic idea of CST is to cache the Shortest-Path Tree (SPT) of network topology that appears frequently, and use the cached SPTs to instantly create routing table when the topology changes to one of the cache. At the end of this thesis, we show CST's effectiveness by a trace driven simulation.

Keywords:

intra-domain, routing instability, convergence time, link-state routing protocol, OSPF, Dijkstra algorithm.

ドメイン内経路の不安定性に関する研究[*]

張 舒

内容梗概

現在、インターネットの信頼性を損なう問題の一つとして経路の不安定性がある。経路の不安定性とは、何らかのネットワーク障害が原因で頻繁に経路が変更し、または消滅する現象を指す。これによりパケット破棄、ルーティングループが発生し、さらにルータの負荷が増加し、ネットワークの信頼性が著しく低下し問題となる。インターネットが将来の人間社会において主な通信手段になると予想それているだけに、このようにインターネットの基盤を直接に影響する問題の実態を解明し、対策を講じることが重要である。

本論文ではまず日本国内最大な学術ネットワークである WIDE Internet における経路の不安定性の調査結果を示す。WIDE Internet のバックボーンで経路制御に使われている経路制御プロトコル OSPF のデータを二年間に渡って収集し、データの解析を通じ、WIDE Internet において経路の不安定性の発生頻度を定量的に評価し、その不安定性を起こす原因について分析する。これにより、エンドユーザが普段ほとんど気づいていないにも関わらず、経路の不安定性が日常的に生じていることを示す。また、経路の不安定性が頻繁かつ持続に起きている場合、経路情報のインスタンスの数が限られていることも示す。それによって全体的でみたネットワークのトポロジの数も限られていることがわかった。

経路の不安定性が頻繁に起こりうる以上、その悪影響を最小限に抑える必要がある。本論文の後半ではキャッシュ型最短木によって、経路の不安定性による悪影響を最小限に抑える Cached Shortest-path Tree (CST) 手法を提案する。CST の基本的な考え方は、ルーティングが不安定な状態において、頻繁に出現するネットワークトポロジおよびその最短木をキャッシュし、ネットワークトポロジがキャッシュされたものと同一になった時、保存された最短木を使って経路表作成時間を短縮することである。これによって経路の収束時間を短縮することできる上、計

[*]奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9961032, 2003 年 3 月 5 日.

算負荷の大きい Dijkstra アルゴリズムの実行回数を大幅に削減できる。CST は上記の調査で得られた、経路が不安定の場合、全体的でみるネットワークトポロジの数が限られているという調査結果に基づいている。WIDE Internet で計測した経路制御情報の履歴データを用いたシミュレーション評価により、CST の有効性を明らかにする。

キーワード

ドメイン内、経路、不安定性、収束時間、リンクステート経路制御プロトコル、OSPF、Dijkstra アルゴリズム

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

The Internet has greatly increased its influence during the last decade. According to NielsenNetRatings ([1]), till the end of year 2001, users of the Internet has almost reached 500 million all over the world. With persistent economical development and education, it is expected that the users will continue to increase. Nowadays, in addition to getting information through the WWW, exchanging e-mails, transferring file, etc., people began to use the Internet on business, medicine, etc., and it is becoming an indispensable tool for our daily lives.

Although there is so much expectation on the Internet, currently the 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 lacks reliability. End users often find that sometimes they get extremely poor performance, if not none, during their use of the Internet. There are many factors that lead to poor performance such as the bandwidth of links, the efficiency of application software and the robustness of protocol. In this paper, we focus on the influence of routing instability because it directly affects the efficiency of IP packet forwarding.

The instability of data transmission derives from the complexity of the Internet. Different with traditional telephone system, which tends to be under the control of telephone companies of limited number, the Internet consists of thousands of Internet Service Providers (ISP). On the network of each ISP, different hardware and software are used. The utilization policies inside or outside the network are diverse. All of these can lead to routing instability. As the Internet is still on its process to expand at a high rate, this complexity is expected to keep on increasing.

In this thesis, we first present the result of a routing instability investigation on WIDE Internet, the largest academy network in Japan, to show how frequently routing instability can occur on a daily-used network. We then propose Cached Shortest-path Tree (CST) approach which not only quickens intra-domain routing convergence but also reduce routers' load. The thesis is organized as follows: in the remainder of this section, we explain what routing instability is and present the related work on it. In Sect. 2, we introduce the current routing architecture of the Internet and the transition of the intra-domain routing protocols. In Sect. 3, we show the result of an intra-domain routing instability investigation on WIDE Internet and analyze the origin of the instability. In Sect. 4, we detail the proposed CST approach and show its effectiveness through a simulation. In Sect. 5, we conclude our work and introduce the future directions of our research.

1.2 Routing Instability

Routing instability, also called route flap, means the route change or disappearance due to unexpected network trouble. As generally routers use their routing tables to decide the next hop to which they should send a packet while forwarding it, when routing information for a network changes or disappears by some network trouble, routing table is recalculated with the wrong information and the next hop for the network is mistakenly decided. Packets for that network will be either sent to the wrong direction or simply dropped. What is worse, when the mistakenly decided next hop happens to be the router from which the packet was received and which has not recalculate the routing table yet, routing loop will occur. The packet will be repeatedly transfered between these two routers until its TTL becomes zero or the route for this packet converges on these two routers. When the number of packets that involved in the same routing loop reaches certain level, the bandwidth of links will be seriously wasted and the load of routers will increase,thus making other oscillation of routing information more likely to happen.

Although in most cases, when a packet is lost, TCP (in the case that the application uses TCP as the transport protocol) will retransmit the packet after its retransmission timer expires without receiving proper acknowledgment, it is explicit that the efficiency of the data transmission will be seriously degraded. Furthermore, when this route change persists, TCP connection will be timed out and disconnected, thus directly affecting end users.

1.3 Route Calculation for Link-state Routing Protocols

As a category of intra-domain routing protocol, link-state routing protocols have been used for years because of its capability to achieve relatively fast routing convergence and flexible routing. Generally, link-state routing protocols use Dijkstra algorithm to calculate shortest-path tree (SPT). Dijkstra algorithm is classified into $O(n^2)$ algorithm and costs much CPU cycle when used on large scale networks (usually with hundreds of nodes). For this reason, implementors often introduce artificial delay to reduce the number of route calculations. However, because all routers schedule their route calculation in different timing, such delay may lead to packet loss and routing loop due to the inconsistent routing tables. If the change of network topology does not happens frequently, for example, it only happens several times a day, generally thinking, the bad influence it brings can be considered acceptable because the consistency lasts only for the order of seconds. But when the network topology changes frequently and persistently, the total time of inconsistent routing table will amount to the order of minutes or even hours. Under such condition, applications will get extremely poor performance on data transmission and 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 interval time of route calculation to the order of millisecond. However, as the Internet is still on its process to expand at a great rate, it is difficult to keep the number of routers of an area small in all time. In addition, using more areas for a routing domain will lead to extra network operation cost, thus makes it difficult to be adopted by most ISPs.

1.4 Related Work

Routing instability has been a hot research topic for years and a quantity of work on it has been done so far. [2] is an early work on routing instability. In this paper, the author measured the instability of NSFNET by analyzing the routing information of 12 hours logged from NSFNET backbone on Tuesday, Aug. 18th, 1992, and showed the unequal distribution of fluctuation cluster sizes and cycle intervals. They showed that most of the fluctuations involved small numbers of networks and cycle intervals were on the order of 10 minutes and most networks experienced only a few transitions while a very small number of nets showed a high degree of volatility. They also showed that the NSFNET backbone update process was sensitive to the size of the update as well as the hop distance between source and destination and larger updates may take many minutes to permeate through the backbone, which caused a high update latency to user packet latency ratio.

A more recent research on routing instability measurement was presented in [3]. In this paper, the authors showed the number of Border Gateway Protocol (BGP) updates exchanged per day was one or more order of magnitude larger than expected and characterized some pathological routing information based on the data collected from the five major Internet exchanges (IX).

In [4], the authors identified the origins of several kinds of pathological Internet routing observed in [3] and showed how their suggest to specific router vendor software changes greatly decreased the volume of Internet routing updates. They also gave a discussion on the trends in the evolution of Internet architecture and policy that may lead to a rise in Internet routing instability.

In [5], the authors examined the latency in Internet path failure, failover and repair due to the convergence properties of inter-domain routing. The measurement showed that inter-domain routers in the packet switched Internet may take tens of minutes to reach a consistent view of the network topology after a fault.

In [6], the author explored routing instability from the aspect of end systems. By the analysis of about 40000 end-to-end route measurements conducted using repeated "*traceroute*"[1], the author analyzed the routing behavior for pathological conditions, routing stability and route symmetry. In addition, the author characterized the prevalence of routing loops erroneous routing, infrastructure failure and temporary outages.

In [7], the authors described an enhancement to OSPF, which is called "I'll Be Back" (IBB) capability and enables other routers to use a router whose OSPF process is inactive for forwarding traffic for a certain period of time. The IBB capability is designed to avoid route flaps that occur when the OSPF process is brought down for the purpose of protocol software upgrade, operating system

[1]A tool used to list all necessary routers to forward a packet.

upgrade, router ID change, AS and interface renumbering, etc. The authors provided an analysis of how and when loops or the inconsistency of the routing table and the forwarding table are formed and proposed solutions to prevent them. In addition, the authors developed an IBB extension to OSPF incorporating these solutions using the GateD platform.

In [8], the authors presented some methods to investigate the behavior of OSPF. By the proposed methods, they measured the time of a router to process LSAs, perform SPF calculations, update FIB and flood LSAs for a variety of routers of Cisco Inc. In addition, they validated the methodology by comparing the measurement with GateD.

In [9], the results of a simulation on the election and the flooding protocols are presented. The authors showed that for the election Protocol: (a) The DR can be elected in constant time. (b) If a router has a limited number of input buffers, a competition for buffers between the election time and causes an oscillatory behavior. (c) At each router, the Router-ID of the DR continuously changes causing instability. (d) In the worst case, when the DR and the BDR fail at the same time, the DR-agreement-time is bounded above by twice the HelloInterval. The authors simulated OSPF flooding protocol by using 20, 50 and 80 router point-to-point networks and showed: (a) For the 50 router network, as link speed exceeds 4000 Kbps, the probability of overflowing the input buffers increases causing retransmissions. The increase in bootup-convergence-time from retransmissions is bounded by two and three times the RxmtInterval for link speeds of 4000 to 6000 Kbps and above 50 Mbps respectively. (b) For 20 and 50 router networks, the input buffer size has little impact on the bootup-convergence-time. For the 80 router network, a small change in the input buffer size drastically change the bootup-convergence-time. (c) Reducing the value of the RxmtInterval lowers the bootup-convergence-time at high link speeds.

In [10], the authors studied the effects of traffic overload on routing protocols by quantifying the stability and robustness properties of OSPF and BGP. They developed analytical models to quantify the effect of congestion on the robustness of OSPF and BGP as a function of the traffic overload factor, queuing delays and packet sizes. Then they use the analytical framework to investigate the effect of factors that are difficult to incorporate into an experimental setup, such as a wide range of link propagation delays and packet dropping policies. The authors demonstrated the importance of selective treatment of routing protocol messages from other traffic by using scheduling and utilizing buffer management policies in the routers to achieve stable and robust network operation.

In [11], the authors argued that it is possible for link state routing protocols to converge in the order of tens of milliseconds. They presented some analysis of IS-IS convergence by showing its behavior upon link/router failures and repairs, and its scaling properties to large networks both in terms of number of nodes and links. They then explored changes needed in the IS-IS specification and implementation in order to reach IGP convergence in milliseconds.

In [12], the author provided a case study on the characteristics and dynamics of LSA traffic for a large enterprise network. For this network, they focused on LSA traffic and analyzed: (a) the class of LSAs triggered by OSPF's soft-state refresh, (b) the class of LSAs triggered by events that change the status of the network, and (c) a class of duplicate LSAs received due to redundancy in OSPF's reliable LSA flooding mechanism. They also derived the baseline rate of refresh-triggered LSAs automatically from network configuration information and investigated finer time scale statistical properties of this traffic, including burstness, periodicity, and synchronization. In addition, the authors discussed root causes of event-triggered and duplicate LSA traffic, as well as steps identified to reduce this traffic.

In [13], a mechanism is proposed to achieve faster routing convergence by counting the changing frequency of advertised routing information for each network node. The authors suggest: (a) When the routing information for one network node does not change frequently, calculate routes without delay to reduce the time of inconsistent routing. (b) When the node's routing information changes frequently, calculate routes with longer delay to reduce router's workload. This approach works fine with cases in which only one node's routing information changes but has difficulty to deal with cases in which routing information of several nodes change at the same time.

As currently each route calculation of OSPF needs a full execution of Dijkstra algorithm from scratch, several dynamic SPF algorithms are proposed to generate the new SPT from a precalculated one in order to reduce CPU cycle needed. In

[14], the authors provided an output bounded resolution to compute the SPT of a directed or an undirected graph with positive real edge weights. They conclude that with their proposition, the computation needs $O(\sqrt{m}\log n)$ amortized time per vertex update for a general graph, where n is the number of the vertices and m is the number of edges.

In [15], the authors proposed an algorithmic framework that yields dynamic versions of some well-known static SPT algorithm such as Dijkstra, Bellman-Ford and D'Esopo-Pape. In particular, they proposed two different incremental methods to transform static algorithms into new dynamic algorithm. They argued that with the two methods, the resulting dynamic algorithm will make the minimum number of changes to the SPT topology following a link state update, thus improving routing stability.

In [16], they authors gave an experimental analysis on the dynamic algorithms in [14] and [17]. They compared the result of these algorithms with Dijkstra algorithm and argue that dynamic algorithm is more efficient than static SPT algorithms.

Although the proposed dynamic algorithms has less complexity than running Dijkstra algorithm each time from scratch, it is difficult for the manufacture to adopt them. We show the reason in Sect. 3.5.

There are also some other research on routing instability. In [18], an algorithm is proposed to conduct BGP route flap damping. This algorithm uses the change frequency of a route to decide whether or not to advertise a route through BGP. [19] focuses on the synchronization of periodic routing messages and offer guidelines on how to avoid inadvertent synchronization. In [20], the result of an intra-domain routing instability investigation on a IX of Japan is presented and a method of selecting unstable AS is proposed.

2. Internet Routing

Internet routing has been a hot topic ever since the beginning of the interconnection of different networks. Throughout the 40 years' development of the Internet, new routing technology has been deployed and obsoleted one by one. In this section, in addition to giving a review to the history of the routing technology of the Internet, we introduce the current routing architecture. We describe the overall routing architecture of the Internet in Sect 2.1, introduce the transition of IGP in Sect. 2.2 and explain link state routing protocol in Sect. 2.3. We then introduce the two most widely used link state routing protocols: OSPF and IS-IS respectively in Sect. 2.4 and Sect. 2.5.

2.1 The Transition of Internet Routing Architecture

The global Internet's progenitor was the Advanced Research Projects Agency Network (ARPANET) of the U.S. Department of Defense. From 1969 through 1983, the routing protocol used on the Internet was Gateway Gateway Protocol (GGP). GGP uses Bellman-Ford algorithm and was first implemented by BBN [21] and COMSAT [22] in tiny PDP11 computers.

As the ARPANET kept growing, it became necessary to adopt a hierarchical structure for routing. The Internet was thus split into a set of Autonomous Systems (AS). The concept of AS is very loose and commonly it can be considered as a set of routers and networks under the same administration.

The first routing protocol that was used for the purpose of inter-AS routing was Exterior Gateway Protocol (EGP). EGP is composed of three separate procedures:

- The "neighbor acquisition" procedure determines whether two adjacent gateways agree to become neighbors.
- The "neighbor reachability" procedure monitors the link between the neighbors.
- The "network reachability" procedure organizes the exchange of reachability information.

Although EGP can conduct basic inter-AS routing, it is not capable of avoiding false information, conducting policy routing or avoiding topology and routing loops. With the Internet keeps growing, there came a strong user demand for more robust and flexible routing. This led to the standardization of the Border Gateway Protocol (BGP) by the BGP working group of the IETF.

The design of BGP underwent several stages. BGP1 [23], BGP2 [24], BGP3 [25] were published respectively in June 1989, June 1990 and October 1991. BGP4 [26] [27] [28] [29] was published in March 1995 and currently it is the most widely deployed inter-domain routing protocol.

The most outstanding characteristic of BGP is that it is a path vector routing protocol. In BGP, each routing update carries the full list of transit ASes traversed between the source and the destination. As routing loop occurs only if one AS was listed twice in the update, it can be simply detected and avoided. When receiving a route advertisement, the exterior router will check whether or not its own AS is not already listed in the path. If it is, the router can refuse to use the advertised path.

In BGP, the advertised paths in a route update are described by a set of attributes. These attributes help a router to choose a better route when multiple paths exist. Currently, following attributes are defined in BGP4:

- ORIGIN

  ORIGIN is a well-known mandatory attribute that defines the origin of the path information.

- AS_PATH

  AS_PATH is a well-known mandatory attribute that is composed of a sequence of AS path segments.

- NEXT_HOP

  NEXT_HOP is a well-known mandatory attribute that defines the IP address of the border router that should be used as the next hop to the destinations listed in the Network Layer Reachability field of the UPDATE message.

- MULTI_EXIT_DISC

  MULTI_EXIT_DISC is an optional non-transitive attribute that is a four octet non-negative integer. The value of this attribute may be used by a BGP speaker's decision process to discriminate among multiple exit points to a neighboring autonomous system.

- LOCAL_PREF

  LOCAL_PREF is a well-known discretionary attribute that is a four octet non-negative integer. It is used by a BGP speaker to inform other BGP speakers in its own autonomous system of the originating speaker's degree of preference for an advertised route.

- ATOMIC_AGGREGATE

  ATOMIC_AGGREGATE is a well-known discretionary attribute of length 0. It is used by a BGP speaker to inform other BGP speakers that the local system selected a less specific route without selecting a more specific route which is included in it.

- AGGREGATOR

  AGGREGATOR is an optional transitive attribute of length 6. The attribute contains the last AS number that formed the aggregate route (encoded as 2 octets), followed by the IP address of the BGP speaker that formed the aggregate route (encoded as 4 octets).

BGP is still being improved by the Inter-Domain Routing working group of IETF.

2.2 The Transition of IGP

In the early time of the Internet, mostly the Routing Information Protocol version 1 (RIPv1) [30] was used as IGP. RIPv1 is a kind of hop count routing protocol and is very simple. With the advent of Classless Inter-Domain Routing (CIDR) [31], RIPv1 was gradually replaced by RIPv2, which was based on RIPv1 and can be used for classless routing. Although RIPv2 is just as simple as RIPv1, network researchers begin to prefer link state routing protocol which provides more flexible routing and faster routing convergence.

2.3 Link State Routing Protocol

Link state routing protocols are based on the "distributed map" concept: all nodes have a copy of the network map, which is regularly updated. Each node generates its routing table based on the network map, thus avoiding routing inconsistency.

2.3.1 The Characteristics of Link State Routing Protocols

All link state routing protocols consist of the following components:

- The Link State Database

  For link state routing protocols, all routers maintain a same link state database. This database holds the link status of all routers in the routing domain. As all routers generate their routing tables based on the link state database, generally routing tables of all these routers are consistent.

- The Flooding Protocol

  The flooding protocol is used to distribute the link state of routers throughout the network. In order not to install old routing information into the link state database, all routing information flooded is marked with sequence number so that routers can determine whether the received information is fresh.

- Bringing Up Adjacencies

  Link state routing protocol uses adjacency to maintain reachability information with neighbors. When a router loses or recovers its connection to the Internet, the information is flooded throughout the routing domain.

- Securing the Map Updates

  Link state routing protocol periodically floods the routing information of all routers so that corrupt information will be detected.

2.3.2 The Advantages of Link State Routing Protocol

Compared with RIP, the link state routing protocol has following advantages:

- Fast, loopless convergency

  Unlike RIP which executes a distributed computation using the Bellman-Ford algorithm, the link state routing protocol uses a rapid transmission of the new information through the flooding protocol and creates routing table by a local computation. This makes routing convergence of the link state routing protocol much faster. In addition, in the case of link state routing protocol, after successful flooding and computation, all routes in the network are consistent, thus assure that there is not persistent routing loop and counting to infinity.

- Support of multiple metrics

  In modern Internet, network topology tends to be complex and network operators may want to decide which link should be used with higher priority. As the concept of link cost is introduced into link state routing protocol, it becomes possible to set a lower cost for the favorable link.

  In addition, the link state routing protocol supports TOS routing. Each routing record can contain a set of metrics so that the route can be selected by the following criteria:

  - The largest throughput
  - The lowest delay
  - The lowest cost
  - The best reliability

- Multiple path

  In complex networks, several equivalent routes may exist toward a same destination. In the case of RIP, only one route can be selected. But with the link state routing protocol, it is possible to split the traffic over the equivalent paths so that data transmission becomes more efficient.

2.3.3 Dijkstra Algorithm

Currently, the link state routing protocol uses Dijkstra algorithm (named after its discover, E.W. Dijkstra) to compute SPT. Dijkstra algorithm solves the problem of finding the shortest path from a point (the source) in a graph to a destination. As one node can find the shortest paths from a given source to all points in a graph in the same time, this problem is sometimes called the single-source shortest paths problem.

Here we describe this famous algorithm. For a graph

$$G = (V, E)$$

where

- V is a set of vertices and
- E is a set of edges.

  Dijkstra's algorithm keeps two sets of vertices:

- S: the set of vertices whose shortest paths from the source have already been determined and
- V-S: the remaining vertices.

The other data structures needed are:

- d: an array of best estimates of shortest path to each vertex.
- p_i: an array of predecessors for each vertex.

The basic mode of operation is:

1. Initialize d and p_i.
2. Set S to empty.
3. While there are still vertices in V-S,

(a) Sort the vertices in V-S according to the current best estimate of their distance from the source.

(b) Add u, the closest vertex in V-S, to S.

(c) Relax all the vertices still in V-S connected to u.

The relaxation process updates the costs of all the vertices, v, connected to a vertex, u, if we could improve the best estimate of the shortest path to v by including (u, v) in the path to v. This sets up the graph so that each node has no predecessor ($p_i[v] = nil$) and the estimates of the cost (distance) of each node from the source ($d[v]$) are infinite, except for the source node itself ($d[s] = 0$). The relaxation procedure checks whether the current best estimate of the shortest distance to v ($d[v]$) can be improved by going through u (i.e. by making u the predecessor of v) as follows:

```
relax( Node u, Node v, double w[][] )
    if d[v] > d[u] + w[u,v] then
       d[v] := d[u] + w[u,v]
       pi[v] := u
```

The algorithm itself is now:

```
shortest_paths( Graph g, Node s )
    initialise_single_source( g, s )
    S := { 0 }        /* Make S empty */
    Q := Vertices( g ) /* Put the vertices in a PQ */
    while not Empty(Q)
        u := ExtractCheapest( Q );
        AddNode( S, u ); /* Add u to S */
        for each vertex v in Adjacent( u )
            relax( u, v, w )
```

The complexity of Dijkstra algorithm is $O(n^2)$ and it can be reduced to $O(n * log(n))$ with the use of a binary heap.

2.4 OSPF

OSPF is a kind of link state routing protocols developed by IETF to run over IP. It consists of the Hello Protocol, the Exchange Protocol and the Flooding Protocol. At present, OSPF is the most widely deployed link state routing protocol.

2.4.1 The Three Consistent Protocols

- The Hello Protocol

  In OSPF, the Hello Protocol is used to check whether or not other routers are operational. It ensures that the communication between neighbors is bidirectional. On broadcast or NBMA network, it is also used to select designated router (DR). DR is responsible for maintaining reachability information of all routers on the same link. Backup DR is also selected by the Hello Protocol to play the role of DR when the DR becomes unreachable.

  The Hello Protocol works differently on broadcast networks, NBMA networks and Point-to-MultiPoint networks. On broadcast networks, each router advertises itself by periodically multicasting Hello Packets. This allows neighbors to be discovered dynamically. These Hello Packets contain the router's view of the Designated Router's identity, and the list of routers whose Hello Packets have been seen recently.

  On NBMA networks some configuration information may be necessary for the operation of the Hello Protocol. Each router that may potentially become Designated Router has a list of all other routers attached to the network. A router, having Designated Router potential, sends Hello Packets to all other potential Designated Routers when its interface to the NBMA network first becomes operational. This is an attempt to find the Designated Router for the network. If the router itself is elected Designated Router, it begins sending Hello Packets to all other routers attached to the network.

  On Point-to-MultiPoint networks, a router sends Hello Packets to all neighbors with which it can communicate directly. These neighbors may be discovered dynamically through a protocol such as Inverse ARP [33], or they may be manually configured.

After a neighbor has been discovered, bidirectional communication ensured, and (if on a broadcast or NBMA network) a Designated Router elected, a decision is made regarding whether or not an adjacency should be formed with the neighbor. If an adjacency is to be formed, the first step is to synchronize the neighbors' link state databases.

- The Exchange Protocol

  The Exchange Protocol is used to synchronize link state databases of two routers which have established two-way connectivity. By the Exchange Protocol, the two routers exchange the description of their link state databases first, and then exchange the detailed information that one of the routers does not hold. After the exchange procedure, the two routers will get the same link state database of the routing domain.

- The Flooding Protocol

  The Flooding Protocol is used to disseminate routing information throughout the routing domain. In OSPF, the Flooding Protocol is a reliable one, which means all received routing information must be acknowledged. The acknowledgment can be conducted in either explicit or implicit way.

  Link State Update packets are used for flooding LSAs. A Link State Update packet may contain several distinct LSAs, and floods each LSA one hop further from its point of origination.

  When a router encounters two instances of an LSA, it must determine which one is more recent. As an LSA is identified by its LS type, Link State ID and Advertising Router, for two instances of a same LSA, the LS sequence number, LS age, and LS checksum fields are used for the determination. The comparison must also be done during the Database Exchange procedure which occurs during adjacency bring-up.

2.4.2 Hierarchical Routing

OSPF allows collections of contiguous networks and hosts to be grouped together. Such a group, together with the routers having interfaces to any one of the included networks, is called an area. Each area has its own link state database and runs a separate copy of the basic link state routing algorithm.

The topology of an area is invisible from the outside of the area. Conversely, routers internal to a given area know nothing of the detailed topology external to the area. This isolation of knowledge enables the protocol to effect a marked reduction in routing traffic as compared to treating the entire Autonomous System as a single link state domain.

Routing in the Autonomous System takes place on two levels, depending on whether the source and destination of a packet reside in the same area (intra-area routing is used) or different areas (inter-area routing is used). In intra-area routing, the packet is routed solely on information obtained within the area; no routing information obtained from outside the area can be used. This protects intra-area routing from the injection of bad routing information.

The areas in OSPF can often be classified into two categories by its function: backbone area and non-backbone area. The backbone area is a special OSPF area(often referred to as Area 0.0.0.0 or Area 0, since OSPF Area ID's are typically formatted as IP addresses). The OSPF backbone always contains all area border routers and is responsible for distributing routing information between non-backbone areas. In addition, the backbone must be contiguous(with the exception in which the Virtual Link is used).

2.4.3 OSPF Messages

The following five kinds of packets are defined in the OSPF specification [34] [35] [36] [37]:

- Hello packet: Hello packets are sent periodically on all interfaces in order to establish and maintain neighbor relationships. In addition, Hello Packets are multicast on those physical networks having a multicast or broadcast capability, enabling dynamic discovery of neighboring routers.

All routers connected to a common network must agree on certain parameters (network mask, HelloInterval and RouterDeadInterval). These parameters are included in Hello packets, so that differences can inhibit the forming of 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. Multiple packets may be used to describe the database. For this purpose a poll-response procedure is used. One of the routers is designated to be the master, the other the slave. The master sends Database Description packets (polls) which are acknowledged by Database Description packets sent by the slave (responses). The responses are linked to the polls via the packets' DD sequence numbers.

- Link State Request packet: Link State Request packets are used to request the pieces of the neighbor's database that are more up-to-date. A router that sends a Link State Request packet has in mind the precise instance of the database pieces it is requesting. Each instance is defined by its LS sequence number, LS checksum, and LS age, although these fields are not specified in the Link State Request Packet itself. The router may receive even more recent instances in response.

- Link State Update (LSU) packet: Link State Update packets are used to flood Link State Advertisement (LSA). Each Link State Update packet carries a collection of LSAs one hop further from their origin. Several LSAs may be included in a single packet.

  Link State Update packets are multicast on those physical networks that support multicast/broadcast. In order to make the flooding procedure reliable, flooded LSAs are acknowledged in Link State Acknowledgment packets. If retransmission of certain LSAs is necessary, the retransmitted LSAs are always sent directly to the neighbor.

- 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. This acknowledgment is accomplished through

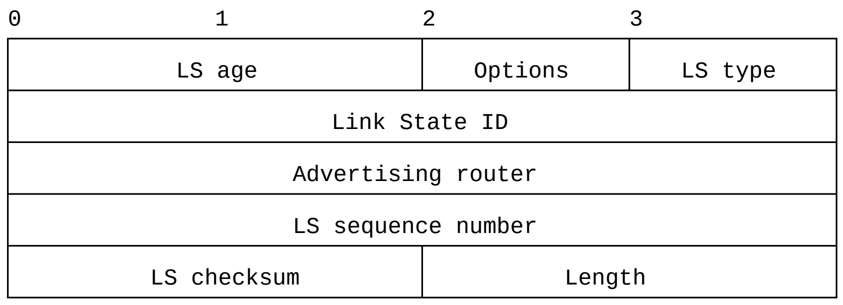

Figure 1. LSA common header

the sending and receiving of Link State Acknowledgment packets. Multiple LSAs can be acknowledged in a single Link State Acknowledgment packet.

Depending on the state of the sending interface and the sender of the corresponding Link State Update packet, a Link State Acknowledgment packet is sent either to the multicast address AllSPFRouters, to the multicast address AllDRouters, or as a unicast.

2.4.4 LSA

LSA is used to disseminate the routing information of each network node. In the OSPF specification, five kinds of LSAs are defined. All of the five kinds of LSA uses a common header showed in Fig. 1.

The five kinds of LSA are:

- Router-LSA: Router-LSA is the Type 1 LSA. In an OSPF routing domain, each router originates a router-LSA for the area that it belongs to. Router-LSA describes the collected states of the router's links attached to the area and is flooded throughout the specific area.

  The router-LSA then describes the router's working connections (i.e., interfaces or links) to the area. Each link is typed according to the kind of attached network. Each link is also labeled with its Link ID. This Link ID gives a name to the entity that is on the other end of the link. Table 1 summarizes the values used for the Type and Link ID fields.

Table 1. The link ID description

Link type	Description	Link ID
1	Point-to-point link	Neighbor Router ID
2	Link to transit network	Interface address of DR
3	Link to stub network	IP network number
4	Virtual link	Neighbor Router ID

In addition, the Link Data field is specified for each link. This field gives 32 bits of extra information for the link. For links to transit networks, numbered point-to-point links and virtual links, this field specifies the IP interface address of the associated router interface (this is needed by the routing table calculation). For links to stub networks, this field specifies the stub network's IP address mask. For unnumbered point-to-point links, the Link Data field should be set to the unnumbered interface's MIB-II [38] ifIndex value. Finally, the cost of using the link for output is specified. The output cost of a link is configurable. With the exception of links to stub networks, the output cost must always be non-zero.

- Network-LSA: Network-LSA is the Type 2 LSA. Network-LSA is used to describe states of broadcast network or non-broadcast multi-access networks. It is originated by designated router of the network. As same as the router-LSA, a network-LSA is specific to one area. So by analyzing network-LSAs, we can monitor routers that form OSPF adjacency or break it on a network.

  A network-LSA is generated for every transit broadcast or NBMA network (A transit network is a network having two or more attached routers). The network-LSA describes all the routers that are attached to the network.

  The Designated Router for the network originates the network-LSA. The Designated Router originates the LSA only if it is fully adjacent to at least one other router on the network. The network-LSA is flooded throughout the area that contains the transit network, and no further. The network-LSA lists those routers that are fully adjacent to the Designated Router; each fully adjacent router is identified by its OSPF Router ID. The Designated Router includes itself in this list.

The Link State ID for a network-LSA is the IP interface address of the Designated Router. This value, masked by the network's address mask (which is also contained in the network-LSA) yields the network's IP address.

- Summary-LSA: Summary-LSAs are the Type 3 and 4 LSAs. They are originated by area border routers and 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 AS boundary router.

  The destination described by a summary-LSA is either an IP network, an AS boundary router or a range of IP addresses. Summary-LSAs are flooded throughout a single area only. The destination described is one that is external to the area, yet still belongs to the Autonomous System.

  Summary-LSAs are originated by area border routers. The precise summary routes to advertise into an area are determined by examining the routing table structure. Only intra-area routes are advertised into the backbone, while both intra-area and inter-area routes are advertised into the other areas.

- AS-external-LSA: AS-external-LSA is the Type 5 LSA. This LSA is originated by AS boundary routers and used to describe destinations external to the AS.

  AS-external-LSAs describe routes to destinations external to the Autonomous System. Most AS-external-LSAs describe routes to specific external destinations; in these cases the LSA's Link State ID is set to the destination network's IP address (if necessary, the Link State ID can also have one or more of the network's "host" bits set). However, a default route for the Autonomous System can be described in an AS-external-LSA by setting the LSA's Link State ID to DefaultDestination (0.0.0.0). AS-external-LSAs are originated by AS boundary routers. An AS boundary router originates a single AS-external-LSA for each external route that it has learned, either through another routing protocol (such as BGP), or through configuration information.

AS-external-LSAs are the only type of LSAs that are flooded throughout the entire Autonomous System; all other types of LSAs are specific to a single area. However, AS-external-LSAs are not flooded into/throughout stub areas. This enables a reduction in link state database size for routers internal to stub areas.

The metric that is advertised for an external route can be one of two types. Type 1 metrics are comparable to the link state metric. Type 2 metrics are assumed to be larger than the cost of any intra-AS path.

If a router advertises an AS-external-LSA for a destination which then becomes unreachable, the router must then flush the LSA from the routing domain by setting its age to MaxAge and reflooding.

2.4.5 Route Calculation of OSPF

Here we detail the OSPF routing table calculation. Using its attached areas' link state databases as input, a router runs the following algorithm, building its routing table step by step. At each step, the router must access individual pieces of the link state databases (e.g., a router-LSA originated by a certain router).

The routing table build process can be broken into the following steps:

1. The present routing table is invalidated. The routing table is built again from scratch. The old routing table is saved so that changes in routing table entries can be identified.

2. The intra-area routes are calculated by building the shortest-path tree for each attached area. In particular, all routing table entries whose Destination Type is "area border router" are calculated in this step. This step is described in two parts. At first the tree is constructed by only considering those links between routers and transit networks. Then the stub networks are incorporated into the tree. During the area's shortest-path tree calculation, the area's TransitCapability is also calculated for later use in Step 4.

3. The inter-area routes are calculated, through examination of summary-LSAs. If the router is attached to multiple areas (i.e., it is an area border

router), only backbone summary-LSAs are examined.

4. In area border routers connecting to one or more transit areas (i.e, non-backbone areas whose TransitCapability is found to be TRUE), the transit areas' summary-LSAs are examined to see whether better paths exist using the transit areas than the ones found in Steps 2-3 above.

5. Routes to external destinations are calculated, through examination of AS-external-LSAs. The locations of the AS boundary routers (which originate the AS-external-LSAs) have been determined in steps 2-4.

2.4.6 SPT calculation

This calculation yields the set of intra-area routes associated with an area. A router calculates the shortest-path tree using itself as the root. The formation of the shortest path tree is done here in two stages. In the first stage, only links between routers and transit networks are considered. Using the Dijkstra algorithm, a tree is formed from this subset of the link state database. In the second stage, leaves are added to the tree by considering the links to stub networks.

The first stage of the procedure (i.e., the Dijkstra algorithm) concerns only the transit vertices (routers and transit networks) and their connecting links. It can be summarized as follows. At each iteration of the algorithm, there is a list of candidate vertices. Paths from the root to these vertices have been found, but not necessarily the shortest ones. However, the paths to the candidate vertex that is closest to the root are guaranteed to be shortest; this vertex is added to the shortest-path tree, removed from the candidate list, and its adjacent vertices are examined for possible addition to/modification of the candidate list. The algorithm then iterates again. It terminates when the candidate list becomes empty.

The stub networks are added to the tree in the procedure's second stage.

2.5 IS-IS

IS-IS is another link state routing protocol developed by OSI to facilitate the connection between open systems. It is similar to OSPF and can be used over many network layer protocols such as IP and ATM. As same as OSPF, the IS-IS protocol also contains several subprotocols such as the Hello Protocol and the Flooding Protocol.

3. An Investigation on Intra-domain Routing

In this section, we present the result of an intra-domain routing investigation that we conducted on WIDE Internet, the largest academic network in Japan. After introducing some related information of OSPF, we show how frequently intra-domain routing instability occurs on WIDE Internet by analyzing OSPF data we collected from WIDE Internet during the investigation. In the end, we discuss problems of route calculation for link state routing protocols.

3.1 Related Information

We choose WIDE Internet, the largest academic network in Japan, as the target network of our measurement. WIDE Internet consists of about 300 routers (in the routing domain) and connects hundreds of organizations. As WIDE Internet uses OSPF as its main routing protocol, OSPF routing messages become our target to analyze. We collected raw data from the network being utilized by many organizations but not from experimental networks because we think the result of real workload is more persuasive.

Although we collected all of the five kinds of OSPF packets, we only analyze Link State Update packets in this thesis because it is the only kind of packets which includes routing information concerning route calculation.

As OSPF uses LSAs to calculates intra-domain routes, by monitoring the LSAs carried by Link State Update packets we can figure out to what extent routing instability is occurring.

As we introduced in Sect. 2.4.2, OSPF divides the whole network into several areas to achieve hierarchical routing. Because generally the backbone area is the most crucial part of a network, we mainly focus on the analysis of the backbone area of WIDE Internet which consists of about 50 backbone routers.

The place we collect OSPF routing messages is NARA-NOC (Fig. 2) of the WIDE Internet, located in Nara Institute of Science and Technology, Ikoma-city of Japan. We placed two FreeBSD boxes on the two 100Base-T ethernet segments respectively. Although we collect OSPF routing messages only from NARA-NOC, we think it is sufficient to get most routing information of the whole network because the WIDE Internet is well designed and it is not likely to happen that

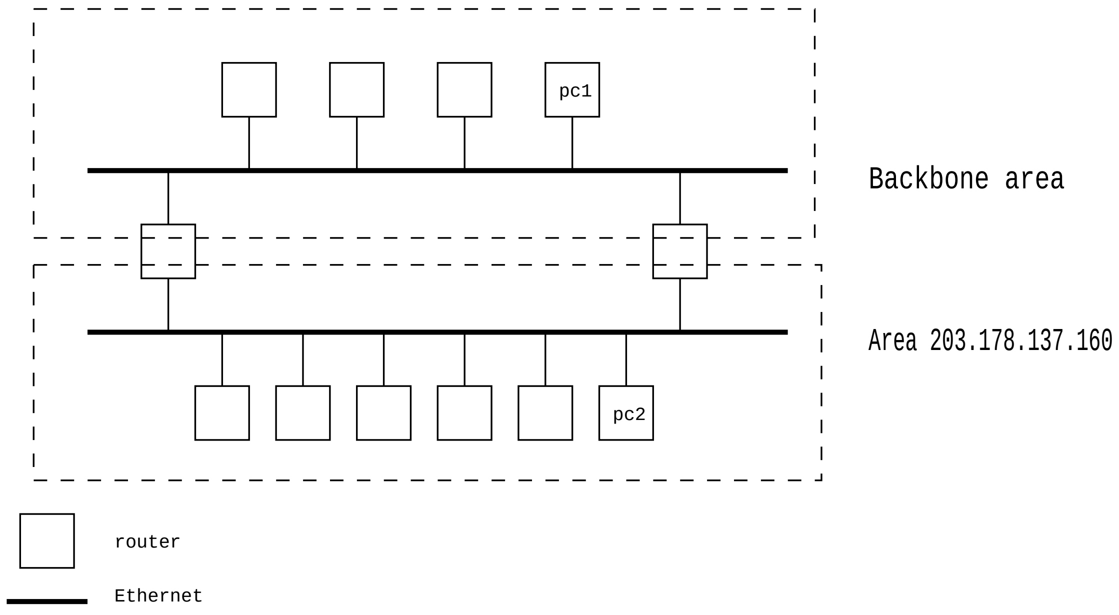

Figure 2. The topology of NARA-NOC network

the NARA-NOC is completely separated. To further indicate this argument, we show the statistics result of the data collected by another research group from the SFC-NOC in the appendix.

The tool we use to collect data is *tcpdump* [39], a widely used program for collecting traffic over shared links such as Ethernet and FDDI. In order to ease the analysis, we changed the tool a little so that it can record the data at daily basis.

The data collection begins on August 2000 and it is still being conducted. The result we are going to show is based on the data from October 2001 to September 2002 (the data of 13 days is lost due to disk failure).

We analyzed all the data by *ospfanaly*, a tool we wrote in C language. *Ospfanaly* uses *libpcap* [39] to read data recorded by *tcpdump* and output the changes of each LSA. Some other self-made Perl scripts are also used in the data processing.

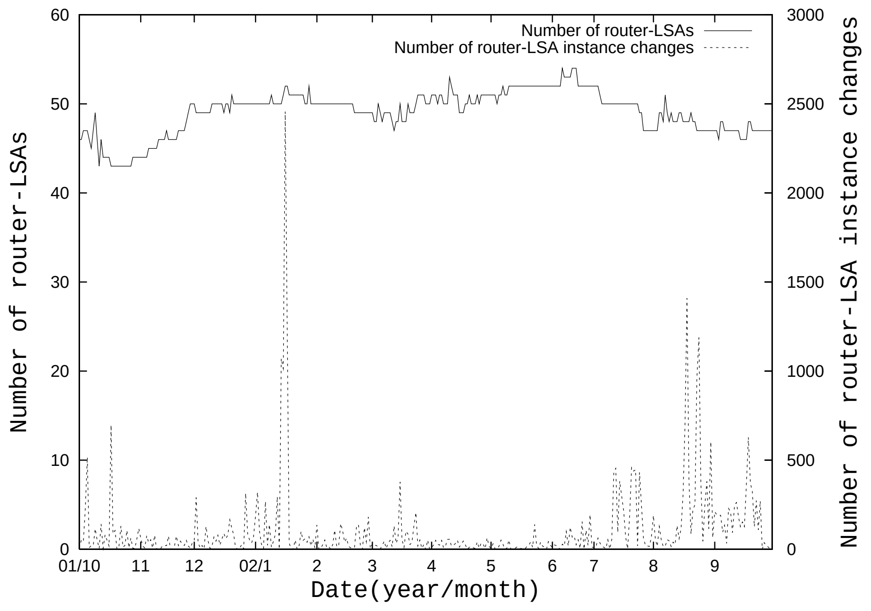

Figure 3. Number of router-LSA and their instance changes in backbone area

3.2 Statistics Result of the Investigation

Here we show the statistics results of OSPF LSAs.

3.2.1 Router-LSA

The number of router-LSAs that appeared in the backbone area each day ranges from 43 to 54. In general, as there should not be much change of network topology, we had thought that these routers would not originate many changing router-LSAs. However, during our investigation, we find the fact is just opposite. Figure 3 and Fig. 4 show the number of total router-LSAs and their changes. From this graph we can see although in most days the number of total changes is not big, there did exist days in which a lot of changes occur and sometimes the number of changes in a single day reaches 2,000 times.

If these changes are originated by most of the backbone routers, we may consider them as normal topology changes due to network maintenance. But after our analysis, we find it is not the case: the changes tend to be originated by only a few routers in relatively short period, such as a few hours. For example, on

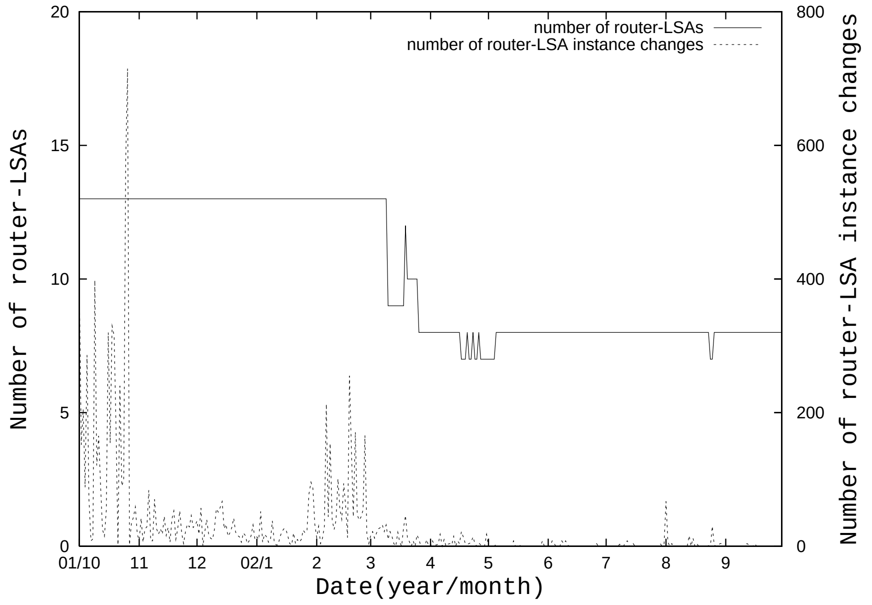

Figure 4. Number of router-LSA and their instance changes in non-backbone area

20th November 2000 a total of 11,380 changes occurred, but 98.6% of the changes were due to two router-LSAs. On 22nd April 2001, 1093 of total 1097 changes were caused by only one router-LSA.

Figure 4 is the statistics result of router-LSAs in non-backbone area. We can see that although in most of the days there are not many changes, there did exist days in which the changes frequently occurred.

3.2.2 Network-LSA

Figure 5 shows the number of network-LSAs and their changes everyday. In total, there are 8-17 network-LSAs that appeared in the backbone area of the WIDE Internet during the 12-month period. Although network-LSAs do not change as frequently as router-LSAs, we still find in some days a network-LSA can oscillate for hundreds of times.

Figure 6 is the statistics result of network-LSAs in non-backbone area. Compared with network-LSAs in backbone area, the ones in non-backbone area are

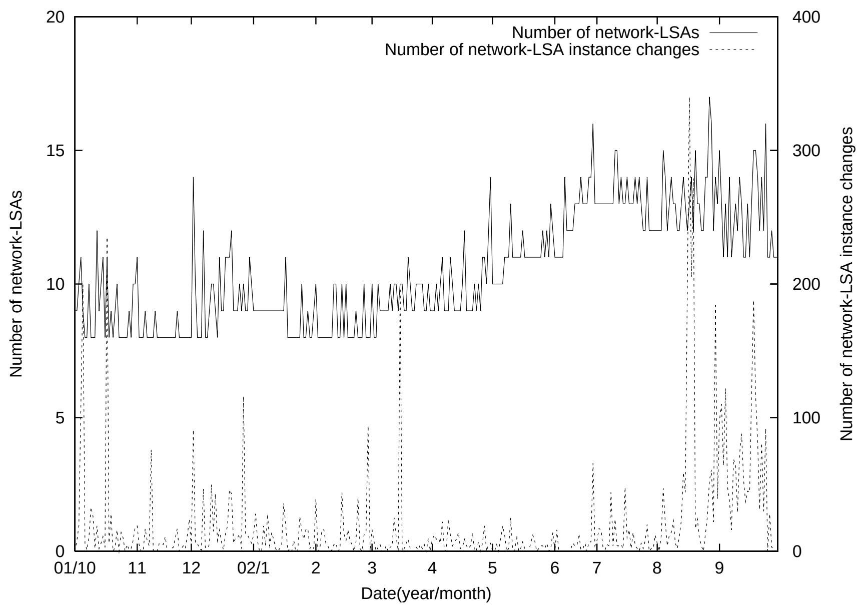

Figure 5. Number of network-LSAs and their instance changes in backbone area

more stable because the number of networks in non-backbone area is relatively small.

3.2.3 Summary-LSA

Summary-LSAs are originated by area border routers in order to let one area know the routing information specific to other areas. They describe destinations of either IP networks, AS boundary routers or range of IP addresses. Summary-LSAs are generated by examining the routing table structure of area border routers. According to the destination they describe, summary-LSAs are grouped into network summary-LSAs and ASBR summary-LSAs.

Figure 7 and Fig. 8 respectively show the numbers of network summary-LSAs and ASBR summary-LSAs that appeared in the backbone area along with the numbers of their changes. Although these two kinds of summary-LSAs show relatively high instability sometimes, the number of changes is low for the most time.

But things get a little different with Fig. 9 and Fig. 10, in which non-backbone

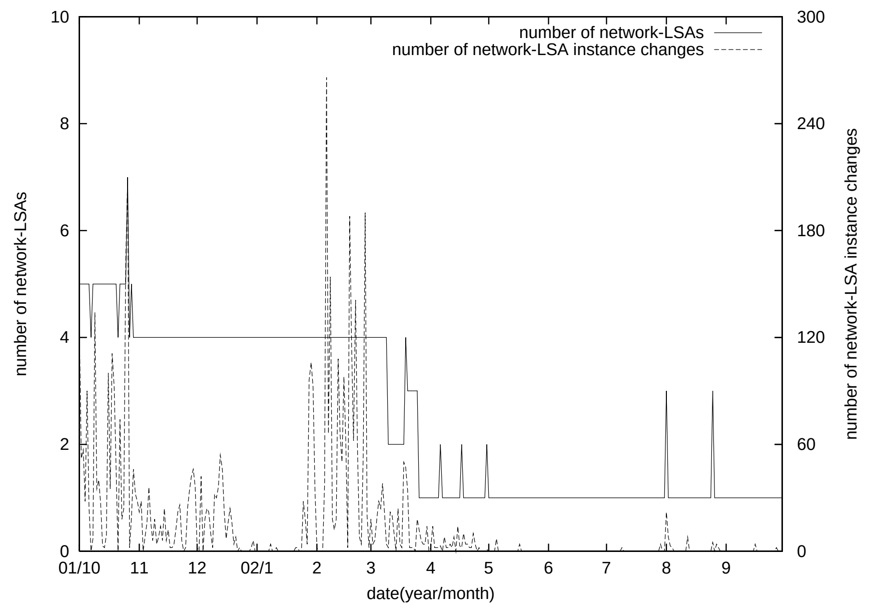

Figure 6. Number of router-LSA and their instance changes in non-backbone area

area statistics results of network summary-LSAs and ASBR summary-LSAs are showed. We can see the two kinds of LSAs in our target non-backbone area showed very high instability and sometimes it goes over 10,000 times a day.

3.2.4 AS-external-LSA

AS-external-LSAs describe routes to destinations external to the Autonomous System. They are the only kind of LSAs flooded throughout the whole routing domain (except the stub area). Usually, AS-external-LSAs are generated by going through all external routes in the routing table. Generally thinking, most of the routes should be quite stable and there should not be much oscillating LSAs. But to our surprise, we find more than half of the AS-external-LSA showed instability to some extent in our investigation. We show the number and the rate of oscillating AS-external-LSAs in Fig. 11 and Fig. 12 respectively. As AS-external-LSA is flooded through the whole routing domain, the results of backbone and non-backbone area are similar. So we only present the statistics result of data

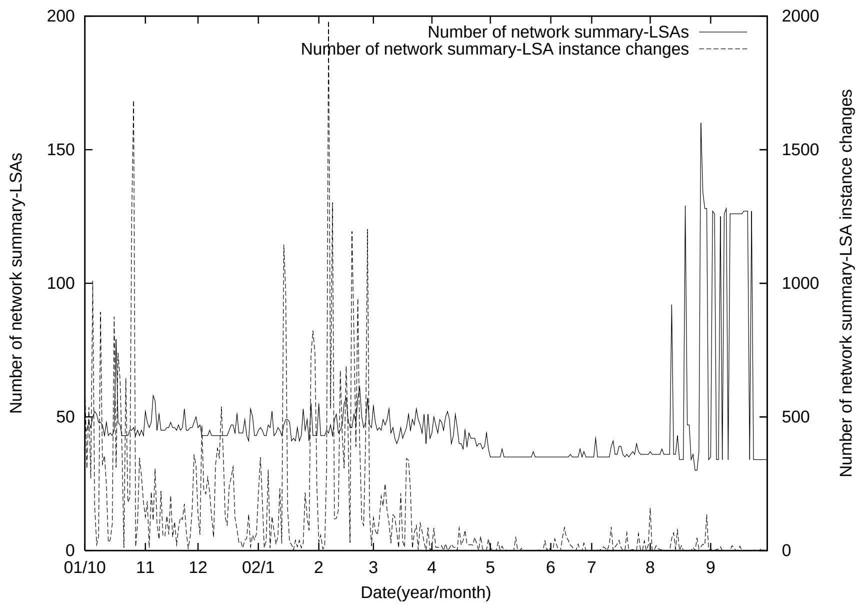

Figure 7. Number of Network Summary-LSA and their instance changes in backbone area

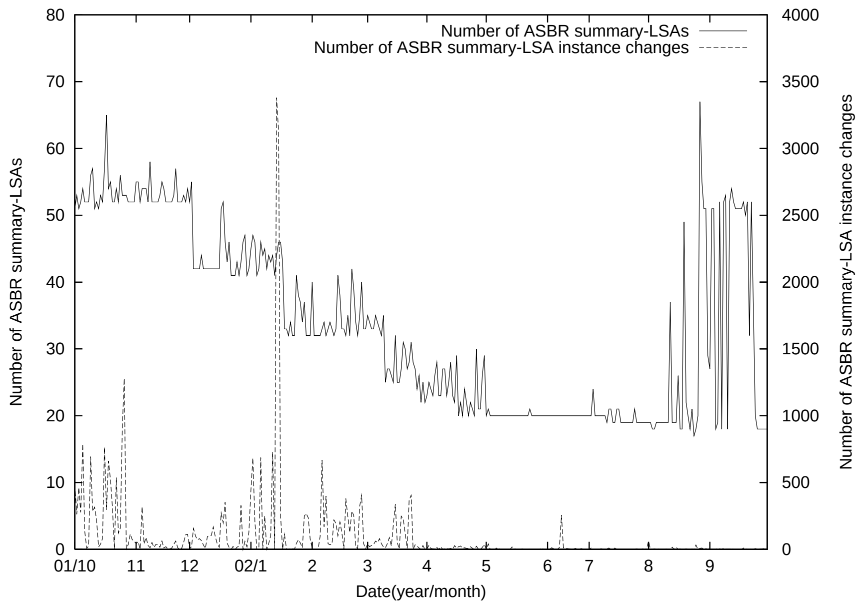

Figure 8. Number of ASBR Summary-LSA and their instance changes in backbone area

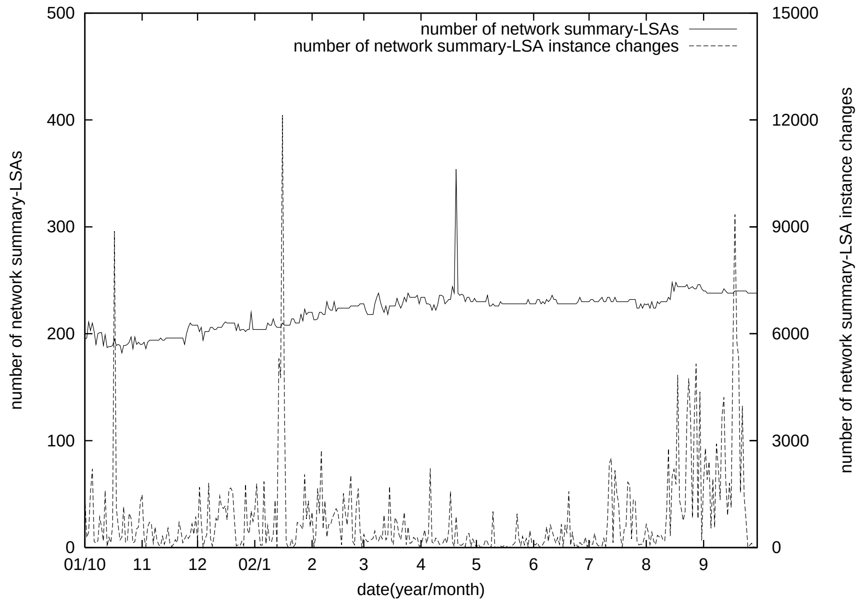

Figure 9. Number of network summary-LSAs and their changes in non-backbone area

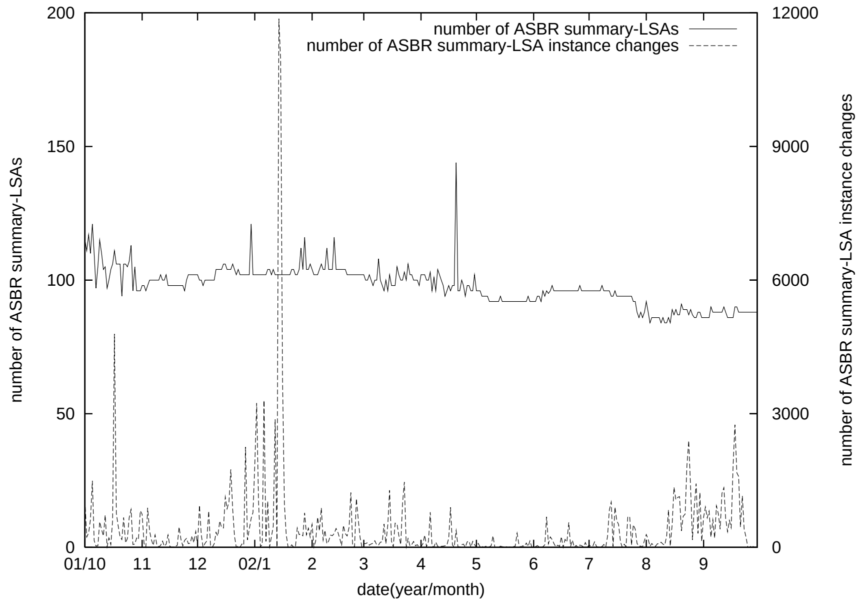

Figure 10. Number of ASBR summary-LSAs and their changes in non-backbone area

collected in the backbone area here.

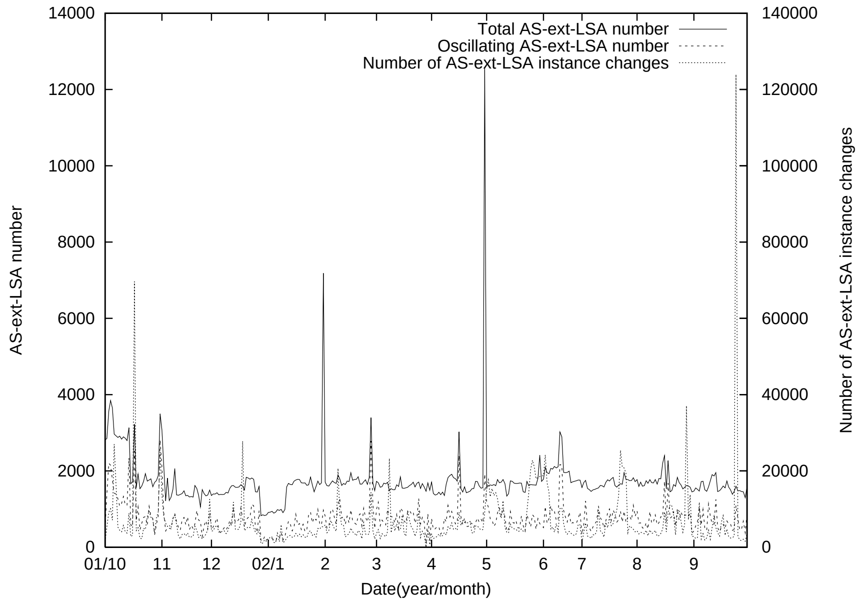

Figure 11. Number of total AS-external-LSAs and oscillating ones

3.3 Oscillation Pattern

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

1. Changes of broadcast and NBMA network interface

   When a router on a broadcast or NBMA network finds that there exist other OSPF routers on the same link it describes this network as a transit network (type 2) in its router-LSA. Otherwise, this network will be treated as a stub network (type 3). In our investigation of oscillating router-LSA links, most cases are repeated changes between transit network and stub network. Many of this kind of change occurs several times per minute.

   We show typical changes of a broadcast interface of 10-minute period in Fig. 13 . During this period, the router-LSA has changed for 32 times in total and 3-4 times per minute.

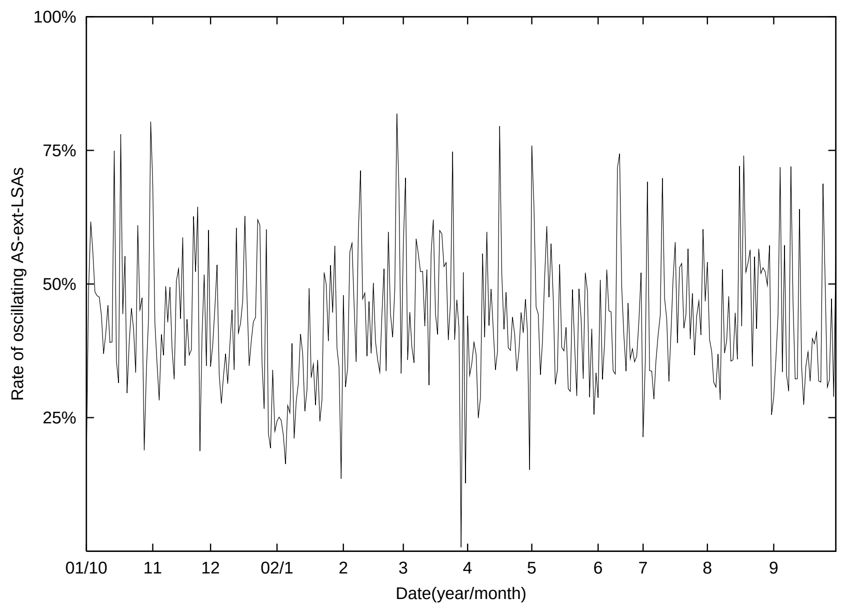

Figure 12. Rate of oscillating AS-external-LSAs

Figure 14 is part of the output generated by *ospfanaly* for the 10-minute period. '+' means the addition of a link and '-' means the opposite.

2. Changes of point-to-point network interface

   In the current Internet, point-to-point connection is often used to connect distant places. When the interface of such connection detects link-down or does not receive its peer's hello packet in certain time (RouterDeadInterval), it originates a new LSA in which the point-to-point link is erased and floods this new LSA to tell other routers (in the same area) that the link has become unavailable. This kind of change is different with the one of broadcast or NBMA network in that the router in the other side will detect the failure and originate a new LSA either. Thus, when point-to-point connection fails, we can see two LSA changes originated by the two endpoints at about the same time. In our investigation, this kind of change was frequently observed too.

   Figure 15 shows an example of the typical changes of point-to-point interface in 30-minute period. Figure 16 is the output of *ospfanaly* for that period.

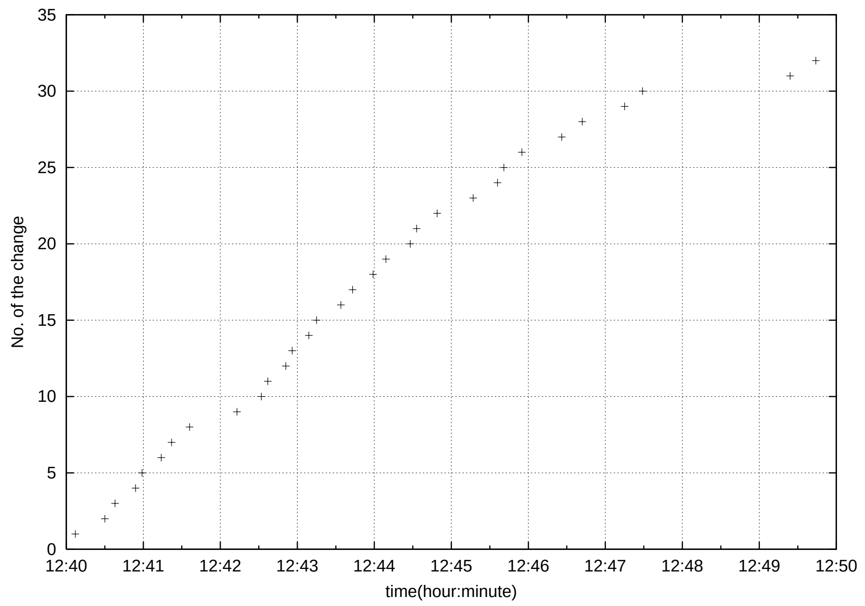

Figure 13. Typical changes of broadcast and NBMA network interface

3.4 Analysis

By presenting the statistics results of oscillating LSAs in Sect. 3.2 we showed that although most network administrators did not notice, route flaps did occur frequently on WIDE Internet, which has been thought quite stable because it is under the operation of many network experts. Here we analyze the oscillation of the intra-domain routing messages. We discuss causes of the oscillation in Sect. 3.4.1 and argue how to improve the robustness of routing protocols in Sect. 3.4.3.

3.4.1 Causes of the Oscillation

We summarize the causes of observed oscillations as network congestion, misconfigurations and bugs of the routing software.

First we consider network congestion as the most factor that accounts for the oscillations. As we described in Sect. 3.2, most changes that have been observed in our investigation are the ones of forming and/or breaking OSPF adjacency. Usually adjacency breaks when link failure occurs or a router does not

```
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
```

Figure 14. Changes of broadcast and NBMA network interface

receive hello message from its adjacent peer in period of RouterDeadInterval(40 seconds normally) because of network congestion. At first, we had thought link failure may be the most factor but meanwhile we found when we detected unusual oscillation reports of congestion came either at almost the same time. The second reason that convince us is that we find oscillation of different LSAs tend to happen at the same time. It is well-known that link failures tend to occur accidentally.

The second factor that we consider to affect LSA stability is the misconfigurations due to network operators. In fact, we observed persistent LSA oscillation caused by a mistaken Router-ID configuration.

The bug of routing software is another factor. During our investigation on the non-backbone area, we found for two times a router running zebra[40] sent all of its link state database every five seconds to another router which did not form adjacency relationship with it. Fortunately the receiving router was awake to the fact and ignored all of the received OSPF LSU packets.

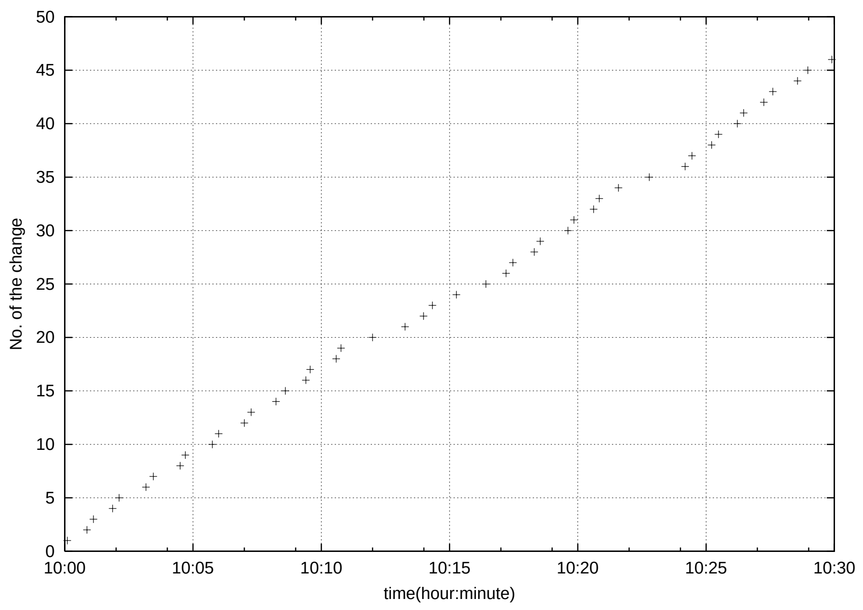

Figure 15. Typical changes of point-to-point network interface

3.4.2 The Similarity of WIDE Internet with Commercial Networks

Although we conducted the investigation only on WIDE Internet, we consider similar phenomena can also be observed on commercial networks. Compared with commercial networks, the WIDE Internet has the following similarity:

- The WIDE Internet is well administrated.

  Although the WIDE Internet is an academic network, it is under the operation of many network experts. These experts include network researchers, network engineers and graduate students who are doing research on network. So we considered the WIDE Internet is administrated as well as commercial networks.

- The WIDE Internet is a large-scale network.

  As we introduced in Sect. 3.1, the WIDE Internet connects hundreds of organizations and the number of users is near a million. So, we consider it has similar scale with commercial networks.

```
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
```

Figure 16. Changes of point-to-point network interface

3.4.3 Toward Faster Routing Convergence

As we saw in the previous sections, routing failures can occur on a daily-used network frequently. When such failures occur, new LSAs are generated and flooded, and all routers in the routing domain need to recalculate their routing tables. However, as all these calculations are conducted in different timing, routing inconsistency will happen. The more routing failure occurs, the longer the inconsistency will last. During the inconsistent period, packets sent to the affected networks are either sent to the wrong direction or simply dropped. As such phenomena directly affect the efficiency of data transmission and have negative impact on the reliability of Internet, there are great demands to reduce the bad influence that routing instability brings to the data transmission.

3.5 The Problem of Route Calculation for Link-state Routing Protocols

As we introduced in Sect. 2.4.5, currently, the route calculation procedure of OSPF can be divided into following steps:

1. Calculating SPT

Table 2. The two kinds of delay used on some popular implementation for OSPF

Implementation	SPF Delay (second)	SPF Hold-Time (second)
Cisco	5	10
Foundry	5	10
Fujitsu	5	10
Extreme	Unknown	3
River Stone	Unknown	5
Hitachi (Gated [41])	0-5	5
Zebra	5	10
Gated	0	5

2. Calculating inter-area routes

3. Calculating AS external routes

In all these three steps, the calculation of SPT for an area is the most complex part. At present, most link state routing protocols use Dijkstra algorithm to calculate SPT. As Dijkstra algorithm is a $O(n^2)$ ($O(n * log(n))$ for heap Dijkstra algorithm) procedure and has scalability problem when used on large-scale networks, most implementation of the protocol introduces artificial delay to avoid using too much CPU cycle to perform it. 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 next calculation.

- SPF Hold-Time: The time that a router needs to wait after previous calculation until next calculation.

Table 2 lists the value of SPF Delay and SPF Hold-Time being used by some most popular OSPF implementations [13].

Because of the introduced delay of route calculation, when receiving a changed LSA, usually routers have to wait for several seconds before they get right routing tables and start forwarding packets based on it. During this period, routers either simply drop packets sent to the affected network prefixes or send them in wrong direction. When the topology of a network is relatively stable and there is not much LSA change, this can be considered acceptable because the time of inconsistent routing will not be very long. However, through the statistical result we got in Sect. 3.2, we know that sometimes the network tends to be unstable and the network topology changes frequently and persistently. Under such condition, the time of route inconsistency totals up to an unacceptable level and we need to reduce such time for further improving the reliability of a network.

Several dynamic SPF algorithms [42] [43] are proposed to quicken the calculation of SPT. However, it is difficult for the vendors to adopt these algorithms because:

- Each dynamic algorithm needs to determine how the network topology changed before computing the updated SPT. In OSPF, as the attached links in Router-LSA or the attached routers in Network-LSA is not sorted, when comparing a newer LSA instance to the previous one, it can be a heavy task when the number of attached links or routers is relatively large.

- The efficiency of the dynamic algorithms depends on the change of the topology. For example, if a large number of nodes or links changed their states, the execution of dynamic algorithms could be very inefficient.

- The dynamic algorithms tend to be very complex and are difficult for the manufacturer to implement.

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

4. CST Algorithm

In this section, we detail CST algorithm which can not only quicken intra-domain routing convergence but also reduce routers' load even under unstable routing condition. After that, we describe the procedures of implementation, discuss issues that need to be considered when implementing CST and show the evaluation result.

4.1 Design

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

- Cache a calculated SPT and its LSA set (Type 1 and 2 LSAs, which decides the network topology).
- When the present LSA set changes to one of the caches, instantly create new routing table based on the cached SPT.

Through the typical changes of LSA we listed in Sect. 3.3, 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 of a single link. Further study tells us that this characteristic is especially outstanding when the LSAs change frequently and persistently.

As the SPT is determined by the set of type 1 and 2 LSAs, it is not difficult to imagine that the instances of all SPTs are also limited. So by caching all frequently appearing SPTs, it is possible to bypass most execution of Dijkstra algorithm by simply using the cached SPT when doing route calculation.

Figure 17 shows the percentage of the top-20 frequently appearing SPTs in all SPTs each day. Here we only show days in which the SPF was calculated more than 50 times. We can see that for most days, the top-20 instances occupy more than 70% of the whole number. In total, 20,213 out of 23,810 (about 85%) are the 20 most frequently appearing instances.

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

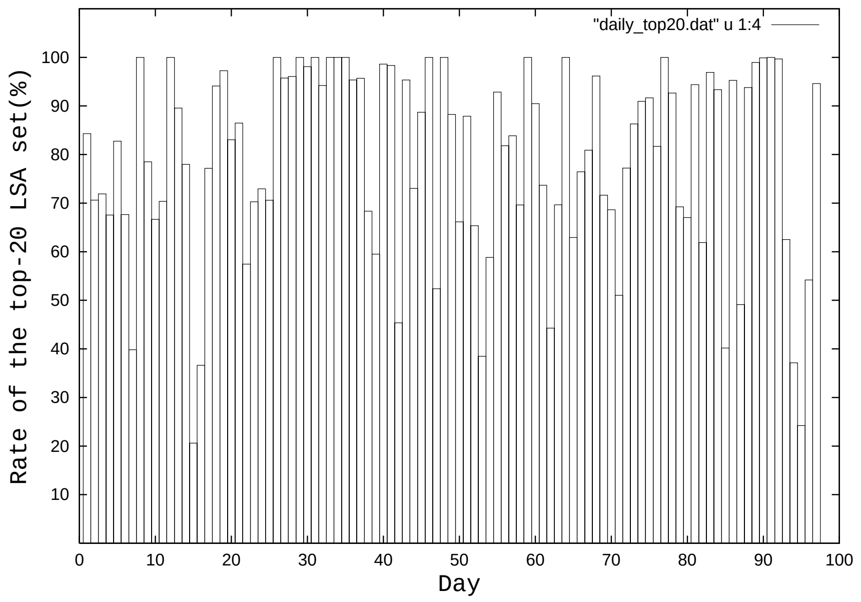

Figure 17. Percentage of the top-20 frequently appearing instances of SPT

- The cache's age: If an LSA set has appeared at least once in the last 12 hours, its SPT has higher priority to be 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 higher hit number than another one, it has higher priority to be continued caching.

In our design, the factor of age is considered before the hit number because we find the frequently appearing LSA set tends to change in different days. The hit number is used to compare two LSA sets both of which have appeared at least once in the last 12 hours.

To implement CST, two kinds of change should be made to the current style of route calculation: the cache and the search of an SPT.

4.1.1 The Cache of an SPT

When a normal calculation of SPT by Dijkstra algorithm is finished, check whether the present LSA set is one that appeared frequently in recent time (the default is 12 hours). If it is, cache the calculated SPT and the corresponding LSA set.

4.1.2 The Search of an SPT

When receiving an OSPF LSU packet, compare all included LSAs with the ones in the LS database.

- If there are new LSAs, obsolete LSAs or LSAs whose content are different with the ones in LS database, update the content of LSAs in LS database. Check whether the present LSA set is the same with any one in the cache.
  - If the LSA set and its SPT are already cached, instantly generate new routing table based on the cached SPT.
  - If the present topology is not cached, wait for the next LSU packet.
- If there is not any changed LSA (including new LSAs and obsolete ones) in the LSU packet, wait for the next LSU packet.

4.2 Implementation Issues

As one of our goals is to replace the redundant execution of Dijkstra algorithm with the switching of SPT, we must make sure that the procedure of finding right SPT is much more simple than Dijkstra algorithm. The following three issues need to be considered during the implementation of CST:

- Cache the LSA set efficiently

  As 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 cost much memory but also make the comparison of LSA sets a heavy task when searching a cache. We recommend to save a relatively stable LSA set and only record the different part of every changed LSA set for an SPT. We call this relatively stable LSA set as stable LSA set (SLS) and define it as follows:

1. (For router that boots up) The LSA set immediately after link state database exchange.
2. (For router in operation) The LSA set which does not change in certain time (the default is 10 minutes).

The difference between an LSA set and the SLS can be stored with the following three lists:

- New list: List to record new LSAs which are not included in SLS.
- Change list: List to record LSAs which are included in SLS but the contents have changed.
- Obsolete list: List to record LSAs which are included in SLS but declared as obsolete.

- Search SPT caches quickly

  When looking up for an SPT in the caches, comparison between the present LSA set and the ones in cache can become a heavy task if there are many LSAs in the three lists (new list, change list, obsolete list). For this reason, we recommend to use a hash number for each LSA set. The hash number should be generated when the SPT is cached. Thus we can quicken the search by first comparing the hash number of the two LSA sets and only doing further comparing when their hash numbers are identical.

- Restructure the cache efficiently when SLS changes

  As all LSA sets are saved in the difference with SLS, when SLS changes, the LSA set difference of each cache needs to be restructured. We recommend to divide 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.

  As the first step needs to compare all LSAs in the two SLSes and the number of recorded LSA can be a relatively large one, we recommend to record all LSAs in 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 New List, Changed List and Obsolete List is necessary. As the number of LSAs in the three list is relatively small, both sorted LSA set and unsorted one are fine.

4.3 A Sample Implementation

Here we show a sample implementation.

4.3.1 New Data Structure

To enable CST we introduce following new structures.

```
struct _sp {
    void *sp_table;     /* the pointer to the shortest tree */
    struct slsa *new_lsa_head;
    struct slsa *old_lsa_head;
    struct slsa *chg_lsa_head;
    u_int addlist_num, oldlist_num, chglist_num;
    u_int hash;
    int hit_num;
    struct _sp *next;
    struct timeval last_time;
};
```

```
struct slsa
{
    struct in_addr ls_id, adv_rtr;
    u_short type, len;
    u_int hash;
    void* data;
    struct slsa *next;
} *ssp_lsa_head=NULL;
```

The structure of _sp is used to store SPT cache and the structure of slsa is used to maintain the LSAs in the three LSA lists of _sp structure.

4.3.2 Initiation

The step of initiation comes after the first SPT calculation. It includes the initiation of SLS and some other related variables.

It can proceed as follows:

```
struct _sp sls;
int sls_flag = 0;

cst_init()
{
    for (lsa = ospf->router_lsa; lsa; lsa = lsa->next)
        copy_lsa();/* copy each Router-LSA to SLS */

    for (lsa = ospf->network_lsa; lsa; lsa = lsa->next)
        copy_lsa();/* copy each Network-LSA to SLS */

    sls_flag = 1;/* to indicate that SLS has been initiated */

    sls->spt = ospf->spt; /* set the SPT pointer of SLS to the first
                             calculated SPT */
}
```

4.3.3 Processing LSU packet

The procedure of processing LSU packet can be implemented as follows:

```
int any_lsa_changed = 0;

while (new_lsu_packet){
    foreach (new_received_lsa){
        if (has_changed(new_received_lsa)){
            install(new_received_lsa);
            calculate_hash(current_lsa_set);
            any_lsa_changed = 1;
        }
    }

    foreach (spf_cache){
        if (spf_cache->hash == current_lsa_set->hash){
             switch_spf(spf_cache->spf);
             generate_routing_table();
             any_lsa_changed = 0;
        }
    }
}
```

4.3.4 Caching SPT

The procedure of caching SPT can be implemented in the route calculation procedure as follows:

```
calculate_spt()
{
    u_int min_hit_num = -1;
    struct _sp spf_cache, temp_cache

    if (!any_lsa_changed)
        return;
    execute_dijkstra();
    generate_routing_table();

    if (spf_cache_num >= max_spf_cache_num){
        foreach (spf_cache){   /* to decide which cache should be deleted */
            if (spf_cache->last_access_time - current_time > 12 * 60 * 60){
                temp_cache = spf_cache;
                break;
            }
            if (spf_cache->hit_num < min_hit_num){
                min_hit_num = spf_cache->hit_num;
                temp_cache = spf_cache;
            }
        }
        delete_cache(temp_cache);
        install_cache();
    }
}
```

4.4 Evaluation

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

4.4.1 The Effectiveness of CST

In Fig. 18, we show the execution number of Dijkstra algorithm needed by a router in the case of traditional and CST enabled implementation along with the total hit number of the cached SPT. To make things more simple, we assume SPF Delay is 0 and SPF Hold-Time is 10 seconds. The number of cache used in this simulation is 20.

We can see from this figure that for most months the execution number of Dijkstra algorithm with CST are greatly reduced compared with the traditional implementation. This trend is especially outstanding in the months when there

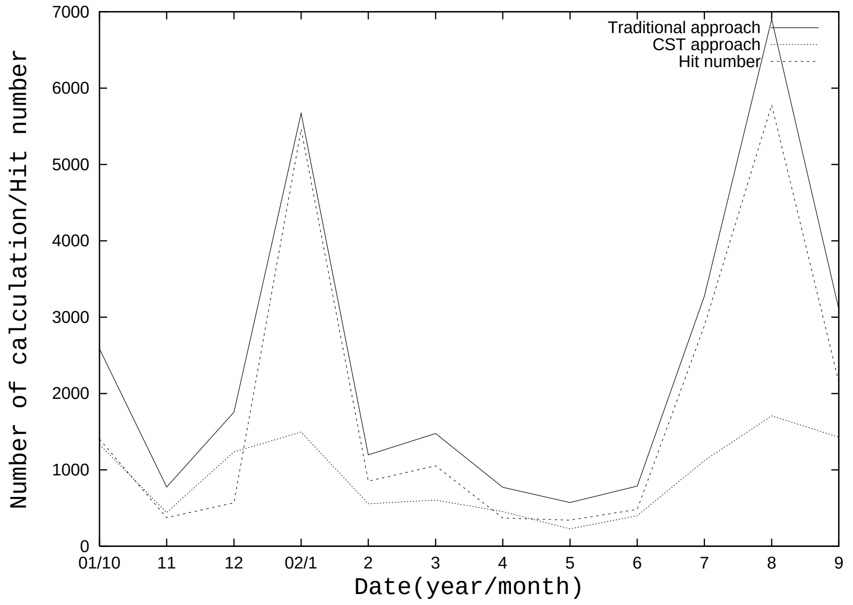

Figure 18. Execution number of Dijkstra algorithm in traditional and CST-enabled route calculation

are frequent LSA changes, such as January 2001 and August 2002. The hit number of the cached SPT in these months are very high either. This indicates that while CST works fine with normal networks, it is especially effective under unstable routing condition. There are also months in which CST algorithm does not show as much effect as other months, such as December 2001. This is because the instance number of network topology in these months are more than other months. But in total, we can say CST is an effective approach.

4.4.2 Convergence Time

Figure 19 shows the monthly convergence time of traditional case and CST-enabled case. The cache number used in this simulation is 20. We can see that the convergence time is largely reduced in the months when the routing tends to be unstable.

Figure 20 shows the daily convergence time in August 2002. We can see that while in some days the convergence time is largely reduced, there are also days

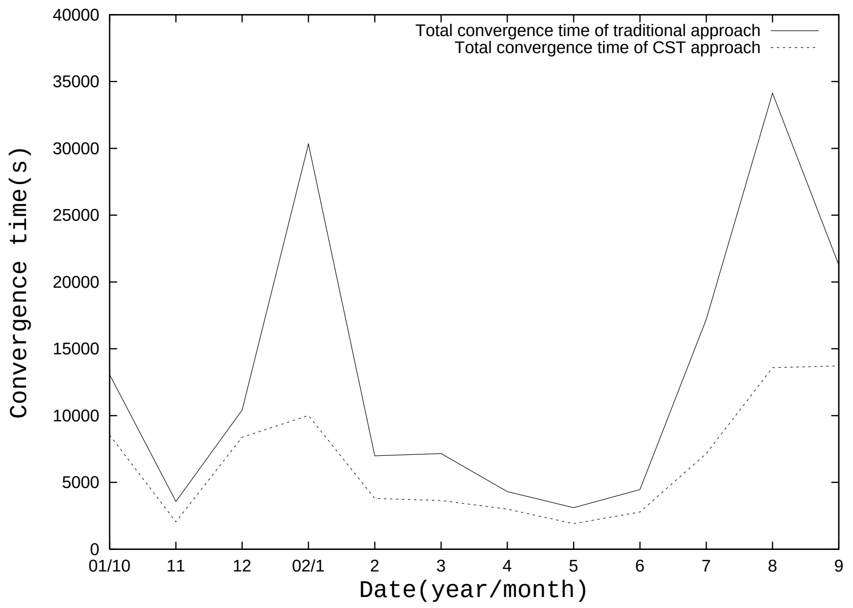

Figure 19. Monthly convergence time of traditional case and CST-enabled case

in which the convergence time of CST approach does not change much compared with the traditional approach. This derives from the diversity of SPTs in those days.

4.4.3 Cache Number

Cache number is the number of SPT cache for which a router enabling CST needs to allocate memory. In essence, when the cache number increases the hit number also increases. However, as the memory of a router is limited, it is not realistic to select a big number as the maximum cache number. We need to find a number with which we can maximize the effect of CST while it does not use too much memory.

In Fig. 21 and Fig. 22 we show how the number of SPT caches affects the efficiency of CST. We can see from these two figures that when the cache number is around 4 or 5, the number of SPT calculation decreases most steeply and the cache hit number increases most sharply in January 2001 and August 2001. We can also see that when the cache number is more than 30, the effect of increasing

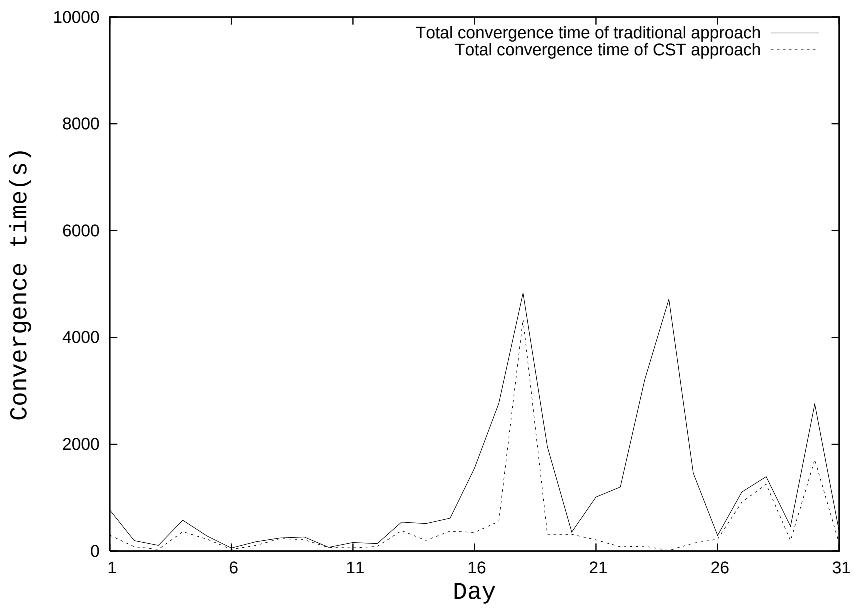

Figure 20. Daily convergence time of traditional case and CST-enabled case

cache number becomes less clear. While in December 2001, the effect is little all along.

4.4.4 SLS

The criterion we use to define SLS is: while every LSA set of the cache should not differ much from SLS, we must also make sure that the SLS is one that does not change often so that there will not be much restructure work. Figure 23 shows the change number of SLS in each month from October 2001 through September 2002. We can see that with our definition in Sect. 4.2 SLS does not change often.

4.4.5 Memory

Here we evaluate the amount of memory that SPT caches need. In Zebra, the structure of an route entry in the routing table is defined as following:

```
struct route_node
```

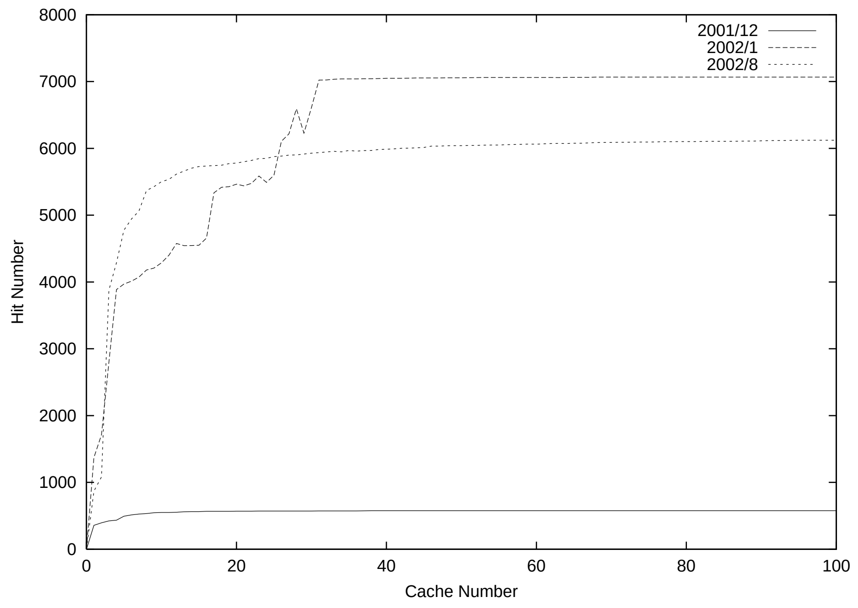

Figure 21. The number of SPT cache and the cache hit number

```
{
  /* Actual prefix of this radix. */
  struct prefix p;

  /* Tree link. */
  struct route_table *table;
  struct route_node *parent;
  struct route_node *link[2];
#define l_left   link[0]
#define l_right  link[1]

  /* Lock of this radix */
  unsigned int lock;

  /* Each node of route. */
  void *info;
```

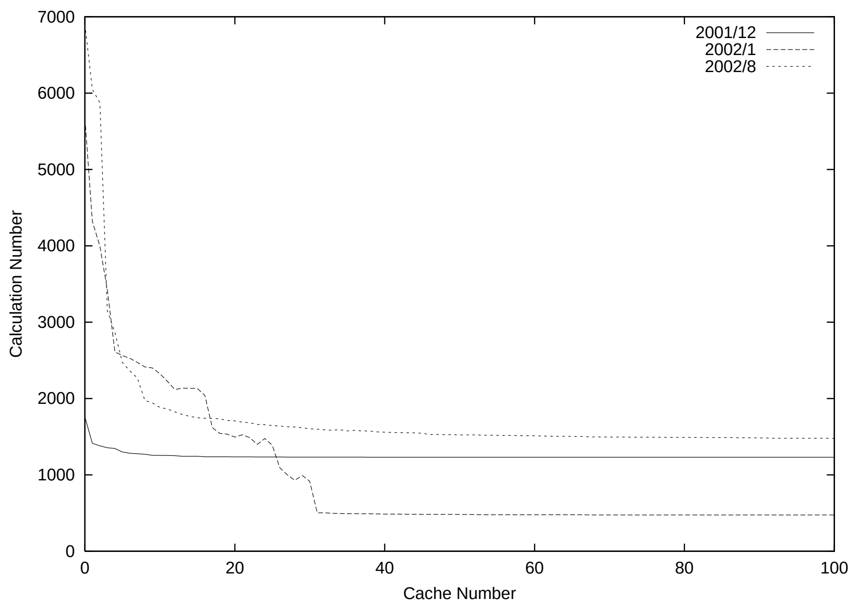

Figure 22. The number of SPT cache and the number of route calculation

```
  /* Aggregation. */
  void *aggregate;
};
```

and the structure of network prefixes is defined as:

```
struct prefix
{
  u_char family;
  u_char safi;
  u_char prefixlen;
  u_char padding;
  union
  {
    u_char prefix;
    struct in_addr prefix4;
```

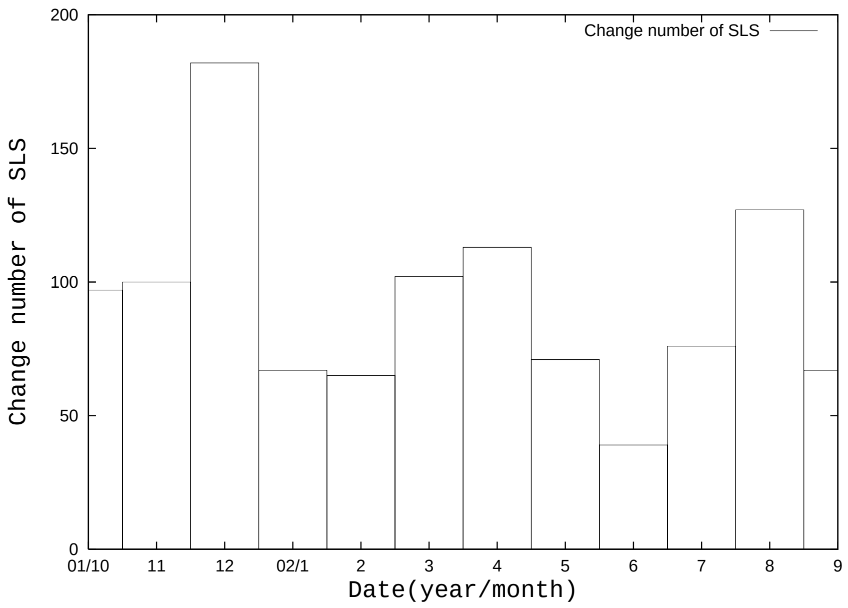

Figure 23. The change number of SLS

```
#ifdef HAVE_IPV6
    struct in6_addr prefix6;
#endif /* HAVE_IPV6 */
    struct
    {
      struct in_addr id;
      struct in_addr adv_router;
    } lp;
    u_char val[8];
  } u;
};
```

Consequently 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 needed memory can be calculated by

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

where n is the number of SPT cache, and r, s are the number of Router-LSA and Network-LSA respectively.

For the WIDE Internet, if we use 5 as the maximum SPT cache number, CST will cost $5 * (40 * (48 + 12) = 12,000$ bytes. If we uses 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.

4.5 Applying CST to Other Link State Routing Protocols

As all link state routing protocols have the following two characteristics:

1. In an area, each router has the same link state database.

2. The router uses Dijkstra algorithm to compute the SPT.

It is possible to apply CST to other link state routing protocols. Here we show how to enable CST on IS-IS [44] [45] [46], another widely used intra-domain routing protocol.

IS-IS is developed by OSI to facilitate the interconnection of open systems. Different with OSPF, IS-IS uses link state protocol data unit (LSP) to constitute its link state database. So, by caching LSP sets that appears frequently along with SPTs of these LSP sets, it is possible to quickly construct routing tables when the network topology changes to one of the cached LSP sets just in a similar way with OSPF.

5. Conclusion and Future Work

Nowadays, no one doubts that the Internet will greatly influence our daily lives in the near future. In addition to gathering information from the WWW, exchanging emails and getting all kinds of digital files, people will use Internet for shopping, education and so on. Other utility of the Internet may also be brought to us. There are every demand of a stable and reliable Internet for all users.

In this thesis, we first presented the result of an investigation on WIDE Internet to show how frequently routing instability can occur on a daily-used network. We found although most of the users do not notice, routing instability can happen frequently on our daily-used network. Such instability not only directly does harm to IP reachability but also increase routers' load, thus make other instability more likely to happen. We summarized some patterns of the LSA oscillation and presented some reason that led to the instability such as traffic congestion.

We then proposed CST approach which can quicken intra-domain routing convergence by bypassing redundant executions of Dijkstra algorithm. The basic idea of CST is to cache SPTs that appears frequently, and instantly generate new routing table based on the cached SPT when the LSA set of the network switches to one of the caches. From our evaluation, we can see CST is effective in most time on 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 condition. 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 WIDE Internet and believe it should also apply to other commercial networks.

As the essence of our approach is to replace the redundant execution of Dijkstra algorithm through cached SPTs, it is important to make sure that the work of finding and switching to a cached SPT do not need as much CPU cycle as the execution of Dijkstra algorithm does. We discussed all main issues that need to be considered when implementing CST and showed the overhead of implementing CST can be minimized with the introduction of SLS and the use of hash algorithm. Although the discuss in this paper was mainly focused on OSPF, CST can also be applied to other link state routing protocols such as IS-IS.

To further indicate the effectiveness of CST, we are going to implement CST on existing routing software such as Zebra and evaluate it on a daily-used network.

Acknowledgements

I would like to thank Prof. Suguru Yamaguchi, Prof. Hideki Sunahara, Associate Prof. Youki Kadobayashi and Associate Prof. Kazutoshi Fujikawa for their great help throughout the work of this thesis. I would also like to thank Dr. Katsuyoshi Iida and Dr. Takeshi Okuda for their valuable advise on my research.

I greatly appreciate all the support from Ms. Natsue Tanida and Ms. Keiko Nakano, the secretaries of the Internet Engineering Laboratory, throughout my research life in Nara Institute of Science and Technology. I would also like to thank other ones in my laboratory for their kindly support throughout my 4-year research life on the doctoral course.

I also greatly appreciate the Association of International Education Japan, which provided economical support from 1997 through 2002 to me. Without its warmhearted help, I could not finish my graduate study.

I would like to thank Mr. Noritoshi Demizu, who helped me much on my research and taught me what the research is when I was a master student.

My sincere appreciation goes to one of my best friends, Jianyuan Huang, who gave me warm help when I was in the most difficult time.

I would also like to thanks Ms. Huizhu Li, the mathematics teacher of my elementary school time, along with all of my other teachers. Without them, I will not think of pursuing a doctoral degree.

Finally, I would like to thank my parents and my sister. Without their support, I would not be able to finish my study in NAIST.

References

[1] Nielsen//NetRatings http://http://www.nielsen-netratings.com.

[2] B. Chinoy, “Dynamics of internet routing information,” SIGCOMM, pp. 45–52, 1993.

[3] C. Labovitz, G. R. Malan, and F. Jahanian, “Internet routing instability,” IEEE/ACM Transactions on Networking, vol. 6, no. 5, pp. 515–528, 1998.

[4] C. Labovitz, G. R. Malan, and F. Jahanian, “Origins of internet routing instability,” no. CSE-TR-368-98, 17, 1998.

[5] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, “Delayed internet routing convergence,” SIGCOMM, pp. 175–187, 2000.

[6] V. Paxson, “End-to-end routing behavior in the Internet,” IEEE/ACM Transactions on Networking, vol. 5, no. 5, pp. 601–615, 1997.

[7] A. Shaikh, R. Dube, and A. Varma, “Avoiding instability during graceful shutdown of ospf,” 2001.

[8] A. Shaikh and A. Greenberg, “Experience in black-box ospf measurement,” ACM SIGCOMM Internet Measurement Workshop (IMW), Nov. 2001.

[9] D. P. Sidhu, T. Fu, S. Abdallah, R. Nair, and R. Coltun, “Open shortest path first (ospf) routing protocol simulation,” SIGCOMM 1993, 1993.

[10] A. Shaikh, A. Varma, L. Kalampoukas, and R. Dube, “Routing stability in congested networks: Experimentation and analysis,” SIGCOMM ’00, Stockholm, Sweden, August 2000.

[11] C. Alaettinoglu, V. Jacobson, and H.Yu, “Towards milli-second igp convergence,” Internet Draft, draft-alaettinoglu-ISIS-convergence-00, November, 2000.

[12] A. Shaikh, C. Isett, A. Greenberg, M. Roughan, and J. Gottlieb, “A case study of ospf behavior in a large enterprise network,” ACM SIGCOMM Internet Measurement Workshop(IMW), November, 2002.

[13] N. Yoshikawa, "A mechanism for faster convergence on link state routing protocol," Master's thesis, Nara Institute of Science and Technology, 2002.

[14] Frigioni, Marchetti-Spaccamela, and Nanni, "Fully dynamic output bounded single source shortest path problem (extended abstract)," SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms), 1996.

[15] P. Narvaez, K.-Y. Siu, and H.-Y. Tzeng, "New dynamic SPT algorithm based on a ball-and-string model," INFOCOM (2), pp. 973–981, 1999.

[16] D. Frigioni, M. Ioffreda, U. Nanni, and G. Pasquale, "Experimental analysis of dynamic algorithms for the single-source shortest-path problem," ACM Journal of Experimental Algorithms, vol. 3, p. 5, 1998.

[17] G. Ramalingam and T. W. Reps, "An incremental algorithm for a generalization of the shortest-path problem," J. Algorithms, vol. 21, no. 2, pp. 267–305, 1996.

[18] C. Villamizar, R. Chandra, and R. Govindan, "BGP route flap damping," RFC 2439, Internet Engineering Task Force, Nov. 1998.

[19] S. Floyd and V. Jacobson, "The synchronization of periodic routing messages," IEEE/ACM Trnasactions on Networking, April 1994.

[20] S. Zhang, N. Demizu, and S. Yamaguchi, "A method for reducing network routing instability," ICDCS2000.

[21] BBN Technologies http://www.bbn.com.

[22] Comsat Corp. (now a part of Lockheed Martin Global Telecommunications) http://www.lockheedmartin.com/.

[23] K. Lougheed and Y. Rekhter, "Border gateway protocol (BGP)," RFC 1105, Internet Engineering Task Force, June 1989.

[24] K. Lougheed and Y. Rekhter, "Border gateway protocol (BGP)," RFC 1163, Internet Engineering Task Force, June 1990.

[25] K. Lougheed and Y. Rekhter, “Border gateway protocol 3 (BGP-3),” RFC 1267, Internet Engineering Task Force, Oct. 1991.

[26] Y. Rekhter and T. Li, “A border gateway protocol 4 (BGP-4),” RFC 1771, Internet Engineering Task Force, March 1995.

[27] Y. Rekhter and P. Gross, “Application of the border gateway protocol in the internet,” RFC 1772, Internet Engineering Task Force, March 1995.

[28] P. Traina, “Experience with the BGP-4 protocol,” RFC 1773, Internet Engineering Task Force, March 1995.

[29] “BGP-4 protocol analysis,” RFC 1774, Internet Engineering Task Force, March 1995.

[30] C. Hedrick, “Routing information protocol,” RFC1058, Junuary 1988.

[31] Y. Rekhter and T. Li, “An architecture for IP address allocation with CIDR,” RFC 1518, Internet Engineering Task Force, Sept. 1993.

[32] C. Huitema, “Routing in the internet,”.

[33] R. Finlayson, T. P. Mann, J. C. Mogul, and M. M. Theimer, “Reverse address resolution protocol,” RFC 903, Internet Engineering Task Force, June 1984.

[34] J. Moy, “OSPF version 2,” RFC 2328, April 1998.

[35] J.Moy, “OSPF standardization report,” RFC2329, April 1998.

[36] J. M. R. Coltun, D. Ferguson, “OSPF for IPv6,” RFC2740, December 1999.

[37] R. Coltun, “The OSPF Opaque LSA option,” RFC2370, July 1998.

[38] F. Baker and R. Coltun, “OSPF version 2 management information base,” RFC 1850, Internet Engineering Task Force, Nov. 1995.

[39] http://www.tcpdump.org.

[40] http://www.zebra.org.

[41] http://www.gated.org.

[42] P. Franciosa, D. Frigioni, and R. Giaccio, “Semi-dynamic shortest paths and breath-first search in digraph,” In Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science, March 1997.

[43] D.Frigioni, A.Marchetti-Spaccamela, and U. Nanni, “Incremental algorithms for single-source shortest path trees,” In Proceedings of Foundations of Software Technology and Theretical Computer Science, December 1994.

[44] D. Oran, “OSI IS-IS intra-domain routing protocol,” RFC1142, February 1990.

[45] R. Callon, “Use of OSI IS-IS for routing in TCP/IP and dual environments,” RFC1195, December 1990.

[46] A. Martey, “Introduction to IS-IS,” NANOG 20, October 2000.

Appendix

The Statistics Result of Data Collected from SFC-NOC

Here we show the statistics result of OSPF routing message collected from SFC-NOC of WIDE Internet. Figure 24 and Fig. 25 show the number of LSA and LSA changes of Router-LSA and Network-LSA respectively. Compared with the figures of NARA-NOC, we can see there are not many major differences.

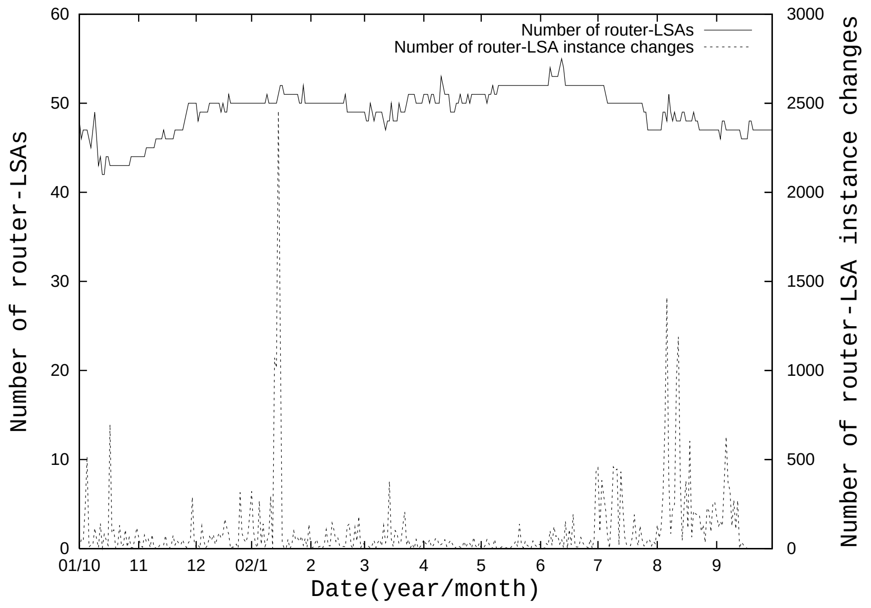

Figure 24. Number of router-LSA and their instance changes in backbone area (SFC-NOC)

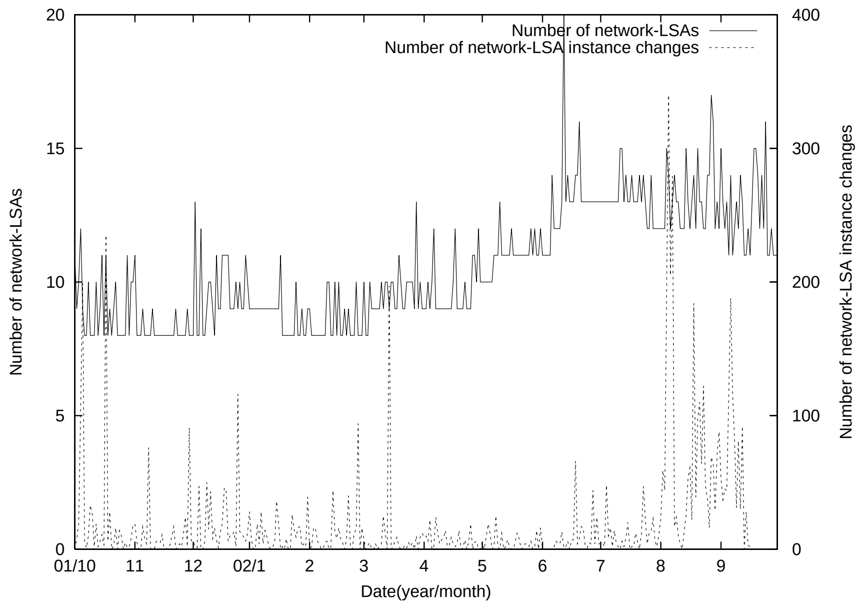

Figure 25. Number of network-LSA and their instance changes in backbone area (SFC-NOC)

Achievements

Journal

1. Shu Zhang, Katsuyoshi Iida and Suguru Yamaguchi, "Cached Shortest-path Tree: An Approach to Reduce the Influence of Intra-domain Routing Instability", conditional acceptance by IEICE transactions on Comm.

International Conference

1. Shu Zhang, Noritoshi Demizu and Suguru Yamaguchi, "A Method for Reducing Network Routing Instability", proceedings of the 20th IEEE ICDCS, International Workshop on Internet, April 2002, Taipei.

2. Shu Zhang, Youki Kadobayashi and Suguru Yamaguchi, "An Analysis on Intra-Domain Routing Instability", poster, ACM SIGCOMM Internet Measurement Workshop 2001, November 2001, San Francisco.

Domestic Conference

1. Shu Zhang, Youki Kadobayashi and Suguru Yamaguchi, ”An Analysis on Intra-Domain Routing Instability of WIDE Internet”, proceedings of The Fourth Workshop on Internet Technology (WIT2001, sponsored by Japan Society for Software Science and Technology), September 2001, Wakkanai

Presentation

1. Shu Zhang, Youki Kadobayashi and Suguru Yamaguchi, ”An Analysis on Intra-Domain Routing Instability”, IEICE Communications Society General Conference 2001, September 2001, Tokyo