Improving BGP Convergence Through Consistency Assertions

Dan Pei[1], Xiaoliang Zhao[2], Lan Wang[1], Daniel Massey[3], Allison Mankin[3], S. Felix Wu[4], Lixia Zhang[1]

***Abstract*— This paper presents a new mechanism for improving the convergence properties of path vector routing algorithms, such as BGP. Using a route's path information, we develop two consistency assertions for path vector routing algorithms that are used to compare similar routes and identify infeasible routes. To apply these assertions in BGP, mechanisms to signal failure/policy withdrawal, and traffic engineering are provided. Our approach was implemented and deployed in a BGP testbed and evaluated using simulation. By identifying and ignoring the infeasible routes, we achieved substantial reduction in both BGP convergence time and the total number of intermediate route changes.**

I. Introduction

This paper presents an approach for improving the convergence time of Internet routing. The Internet is composed of thousands of Autonomous Systems (ASes), loosely defined as networks and routers under the same administrative control. BGP[1] is the *de facto* inter-AS routing protocol, and ideally BGP should quickly converge on a new set of stable routes after any topological or policy changes. However when a topological change occurs, BGP routers often take a long time to explore a large number of transient routes before converging on a new stable route. Measurements in [2] found that the delay in Internet inter-AS path failover averages to 3 minutes, and some non-trivial percentage of failovers trigger routing table oscillations that may last up to 15 minutes. The transient route changes that occur during such a long convergence period result in delayed packet delivery or even losses, as well as added overhead to BGP routers. Lengthy BGP convergence is a problem for the Internet today and threatens to become a larger problem as the Internet continues to grow in size.

Figure 1 shows an example of BGP slow convergence as observed from a single router's view. This example, taken from [3], actually occurred in the Internet. As Figure 1 shows, the router first received a BGP route withdrawal message from AS 2129 to report that it lost its route to a destination prefix within its own domain. AS 2117 apparently believed that it had found, and announced, 6 different routes to the destination, but all these 6 routes end in AS 2129. Since AS 2129 had lost its route to the destination, all these 6 routes were invalid and were eventually discarded. This illustrates the delayed convergence problem that occurs after a route failure; similar problems can also occur when an AS switches to an alternate route, *i.e.*, *route failover*.

TIME	BGP Message/Event
10:40:30	Route Fails/Withdrawn by **AS2129**
10:41:08	2117 announce 5696 **2129**
10:41:32	2117 announce 1 5696 **2129**
10:41:50	2117 announce 2041 3508 3508 4540 7037 1239 5696 **2129**
10:42:17	2117 announce 1 2041 3508 3508 4540 7037 1239 5696 **2129**
10:43:05	2117 announce 2041 3508 3508 4540 7037 1239 6113 5696 **2129**
10:43:35	2117 announce 1 2041 3508 3508 4540 7037 1239 6113 5696 **2129**
10:43:59	2117 sends withdrawal

Fig. 1. Slow convergence example in the Internet

In this paper we will show how routers can detect and ignore invalid routes by applying a set of protocol assertions, thus allowing BGP to converge substantially faster. We have developed a set of assertions for path vector routing protocols in general, as well as a set of enhancements to the general assertions that allows BGP to properly handle special cases such as policy withdrawals and traffic engineering. In our network testbed, our enhancements improved BGP convergence time for a failure withdrawal from 30.3 seconds to 0.3 second and the convergence time after a route change improved from 64.9 seconds to 0.1 second. In simulation tests with a 60-AS network topology, the convergence time after a failure withdrawal improved from 337.0 seconds to 19.5 seconds and the convergence time after a route failover improved from 471.2 seconds to 93.9 seconds.

II. Previous Work

Current BGP implementations explore a potentially large number of backup routes when a failure occurs and many of these backup routes are already invalid. Analysis in [2] showed that, in the theoretically worst case, a fully connected n-AS system might explore all ($n!$) possible paths before BGP converged on a new set of stable routes. Since during this procedure, a changed route is likely to change again in a brief in-

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No DABT63-00-C-1027. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the DARPA.

[1]Dan Pei, Lan Wang and Lixia Zhang are with UCLA. E-mail: {peidan, lanw, lixia}@cs.ucla.edu [2]Xiaoliang Zhao is with North Carolina State University. E-mail:xzhao@unity.ncsu.edu [3]Daniel Massey and Allison Mankin are with Information Sciences Institute. E-mail: {masseyd, mankin}@isi.edu [4]S. Felix Wu is with UC Davis. E-mail: wu@cs.ucdavis.edu

terval[4], BGP includes an update rate-limiting timer to allow BGP routers to 'pack" consecutive updates [4]. The BGP specification [1] requires that a minimum amount of time, denoted ***MinRouteAdver***, must elapse between route advertisements for a specific destination. Therefore, if a route changes multiple times during a $MinRouteAdver$ period, only the last change should be announced. Simulation results by Griffin *et al* [5] illustrated the necessity of having the $MinRouteAdver$ timer: setting $MinRouteAdver$ to be 0 would lead to not only unacceptably large number of updates but also unacceptably long convergence time.

Despite of deployment of $MinRouteAdver$ timer, Labovitz *et al*[2] found that route failovers and route failures still resulted in a significant delay. This delay averaged three minutes and some changes took up to 15 minutes in the current Internet. Further experiments and analysis by Labovitz *et al* [6] showed that the time complexity of Internet failover convergence is upper bounded by $(p \times MinRouteAdver)$, where p is the length of the longest possible backup AS path between the source and destination node.

One previously reported effort to improve BGP convergence time is the *sender side loop detection* approach proposed by [**?**]. Their simulation results show that in a 7-node fully connected topology, this approach improved the convergence time from 120 seconds to 30 seconds. However, Griffin *et al* [5] observed that in general, this approach reduces the convergence time only by a limited amount. They also observed that for each specific network topology they simulated, there is an optimal value for $MinRouteAdver$ that minimizes the convergence time. However, this optimal value varies from network to network, therefore searching for optimal $MinRouteAdver$ cannot be a general mechanism for improving BGP convergence.

In this paper, we present a new technique that can substantially reduce BGP convergence time. It should be noted that, because BGP allows each AS to independently formulate its routing policies, BGP might never converge on any stable routes [7], [8]. In this paper we assume that any route change will eventually result in a new stable route if one exists, or result in the destination being declared unreachable.

III. Assertions for Improving Routing Convergence in Simple Path Vector Protocols

We define a *route convergence period* as the period that starts when a previously stable route to some destination D becomes invalid and ends when the network has obtained a new stable route for D (or when D has been correctly declared unreachable). We measure the length of the convergence period and the number of intermediate route changes that occur during the convergence period. Due to factors such as processing and propagation delay, any failure or route change will take some time to propagate through the network, and at each router there will be at least one route change since the previous (and now invalid) route must be removed. We say that a slow convergence problem occurs if any invalid routes are adopted by some router during a route convergence period; an invalid route is defined as a route that does not reach the destination, and this definition is formalized in Definition 2.

In [9], [10], [11], [12], the convergence properties of distance vector routing algorithms were improved by exploiting the relationships between routes learned from different neighbors and using this information to detect invalid routes. Similarly, we look for relationships between path vector routes and use these relationships to detect invalid routes. The resulting approach allows a router to discard (invalid) transient routes that might occur during the route convergence period. This approach reduces both the convergence time and the total number of route update messages.

In order to clearly present the basic concept of our approach, we will first introduce a *Simple Path Vector Protocol* model.

Definition** 1: **Simple Path Vector Protocol(SPVP) A Simple Path Vector Protocol (SPVP) is the path vector protocol in which each node selects and uses *only one* of its available paths to each destination and advertises to its neighbors *only* the route it is using. That is, in SPVP, a node will advertise to its neighbors only one single path to each destination. The latest path received from each neighbor node replaces the previous path sent by the same neighbor and is kept as a candidate for path selection. If this new path results in a route change, then the newly selected path is sent to neighbors. When a node loses all the paths to the destination, it sends an empty path to its neighbors to withdraw the path that it sent before.

An AS might be roughly viewed as a node in SPVP, thus the $path$ advertised by SPVP is roughly equivalent to the AS Paths in BGP. However, as we will show in Section IV, SPVP differs from BGP in a number of important aspects. To handle these differences, Section IV develops a set of enhancements to the general assertions that we derive in this section.

A. Consistency Theorem for SPVP

To illustrate the relationship between different paths in $SPVP$, let us assume that node R has learned two paths to destination D. Neighbor N_1 is advertising the path (N_1, A, B, C, D) and neighbor N_2 is advertising (N_2, X, B, Y, Z, D). By comparing these two paths, one can conclude that they are not consistent. If one believes N_1, then B's path to D should be (B, C, D). If one believes N_2, then B's path to D should be (B, Y, Z, D). Since B can only advertise one path to D, either N_1 or N_2 (or even both) must be advertising an invalid path to D. We formalize the above idea in the following.

In the discussion of this section, let $path(N_1, D) = (N_1, P_1, P_2, ..., P_n, D)$ be the last path to D reported by N_1 and let $path(N_2, D) = (N_2, Q_1, Q_2, ..., Q_m, D)$ be the last path to D reported by N_2. According to the definition of $SPVP$, each node P_i can only use one path, denoted $path(P_i, D)$. For each $i, 1 \leq i \leq n$, $path(N_1, D)$ can be written as $path(N_1, P_i) + path_{N_1}(P_i, D)$, where $path_{N_1}(P_i, D)$ reflects N_1's last reported

view of the path from P_i to D, and it is correct only if it equals to $path(P_i, D)$.

***Definition** 2: **Valid Path.** $path(N_1, D)$ is *valid* if and only if $path_{N_1}(P_i, D) = path(P_i, D)$ for all $i, 1 \leq i \leq n$. Otherwise, it is *invalid*.*

In other words, $path(N_1, D)$ is valid if and only if its view of the path from each P_i to D is correct. The empty path, $path(N_1, D) = NULL$, is always valid since it only reflects that N_1 does not know a path to D.

***Definition** 3: **Path Consistency.** $path(N_1, D)$ and $path(N_2, D)$ are **consistent** if one of the following consistency conditions is satisfied:*

1) Two empty paths are consistent.
2) Two non-empty paths with no common node (not counting the destination) between them are consistent.
3) Empty $path(N_1, D)$ is consistent with non-empty $path(N_2, D)$ if $N_1 \neq Q_i$, for all $i, 1 \leq i \leq m$.
4) Two non-empty paths $path(N_1, D)$ and $path(N_2, D)$ intersect at nodes $P_j = Q_i$ are consistent if $path_{N_1}(P_j, D) = path_{N_2}(Q_i, D)$.

***Theorem** 1:* If two paths are both valid, then they must be consistent.

Proof: The theorem is a natural result of the definitions of consistent paths and valid paths. Suppose $path(N_1, D)$ and $path(N_2, D)$ are both valid paths.

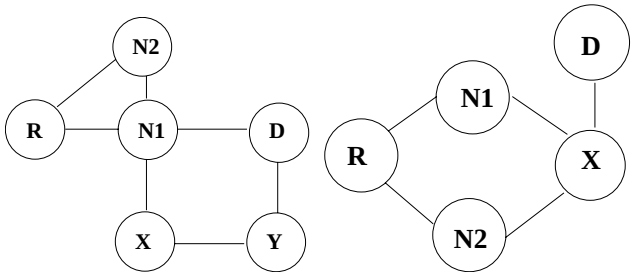
1) If both of them are empty or neither is empty but they do not intersect, they are consistent according to the first and the second consistency conditions.
2) Consider the case that $path(N_1, D) = NULL$ and $path(N_2, D) \neq NULL$. Because $path(N_2, D)$ is valid, $path_{N_2}(Q_i, D) = path(Q_i, D) \neq NULL$ is the correct view of the path from Q_i to D. On the other hand, $path(N_1, D) = NULL$ is also the correct view of the path. Therefore, $N_1 \neq Q_i$, because otherwise $path(Q_i, D) = path(N_1, D) = NULL$ leads to contradicts. $path(N_1, D)$ and $path(N_2, D)$ are consistent according to the third consistency condition.
3) In the case that neither of two paths is empty and they intersect at nodes $P_j = Q_i = M$, then it must be true that $path_{N_1}(P_j, D) = path(M, D) = path_{N_2}(Q_i, D)$ since $path(N_1, D)$ and $path(N_2, D)$ are valid paths. They are consistent according to the fourth consistency condition.

Since in all possible cases two valid paths are consistent, the theorem holds.

In case of two paths being inconsistent, Theorem 1 offers no indication of which one is invalid or whether both are invalid. In the following section, we present a practical approach for consistency checking by applying this theorem to certain specifically restricted cases.

B. Restricted Case of Consistency Theorem

We only apply Theorem 1 to the restricted case where node N_1 appears in $path(N_2, D)$. In this particular case, we claim that information received directly from N_1 should take precedence



(a) N_1 in $path(N_2, D)$ (b) N_1 not in $path(N_2, D)$

Fig. 2. Example network topologies

over information about N_1 that was received indirectly via N_2 if they are inconsistent. If the paths are not consistent, we mark the path from N_2 as an *infeasible* path.

***Definition** 4: **Infeasible Path.** If $path(N_1, d)$ and $path(N_2, d)$ are not consistent and $N_1 = Q_i$ for some $i, 1 \leq i \leq m$, $path(N_2, D)$ is **infeasible**.*

An infeasible path is not necessarily an invalid path, but we require that an infeasible path not be selected as the best path to D, as explained below.

Take Figure 2(a) as an example, suppose the link between N_1 and D just went down, and R's neighbor N_1 is advertising the path (N_1, X, Y, D) and another neighbor N_2 is advertising the path (N_2, N_1, D). These paths are $inconsistent$ because they intersect each other at N_1 and $(N_1, X, Y, D) = path_{N_1}(N_1, D) \neq path_{N_2}(N_1, D) = (N_1, D)$. The intuition behind our approach says that the path (N_1, X, Y, D) that was learned directly from N_1 should take precedence over the path (N_1, D) that was learned indirectly from N_2. Therefore, the path (N_2, N_1, D) is marked as infeasible and should not be selected as the best path to D. If later R receives from N_2 a path (N_2, N_1, X, Y, D), which is consistent with $path(N_1, D)$, the infeasible mark is cleared.

Note that even though the information from N_1 is usually preferred, it is still possible that the information from N_2 is valid. Suppose in Figure 2(a), both N_1 and N_2 have converged on the new paths after the link failure between N_1 and D. So $path(N_1, D) = (N_1, X, Y, D)$, and $path(N_2, D) = (N_2, N_1, X, Y, D)$, and they are consistent. Now suppose the link between N_1 and D comes up, so N_1 advertises (N_1, D) to both R and N_2. Suppose that the link between N_1 and R are so slow that N_2 receives this new path and its advertisement of the new path $path(N_2, D) = (N_2, N_1, D)$ reaches R before the update from N_1 does. Now, R has $path(N_2, D) = (N_2, N_1, D)$ and $path(N_1, D) = (N_1, X, Y, D)$, and this leads to the same inconsistency of the previous example. However, in this example path (N_2, N_1, D) is valid, so it should not be removed although it is marked as infeasible.

This example gives a scenario where indirectly learned information reflects the latest path changes, but this can only occur if there is also a pending update from N_1 that would correct the

conflict with N_2 and clear the infeasible mark. The maximum amount of time that N_2's path will be marked infeasible and ignored is bounded by the time required for the pending update to arrive from N_1. Therefore, it will not be marked as infeasible for an indefinitely long period of time.

An infeasible path is ignored during the path selection process. On one hand, if the path advertised by N_2 was invalid, then ignoring this path avoids an incorrect path change and prevents an invalid path from being advertised to other neighbors. On the other hand, even if a valid path advertised by N_2 is marked as infeasible and is being ignored, its impact on routing convergence is still positive as long as all the nodes follow the shortest path length rule in selecting paths. The temporarily ignored path from N_2's is not the best path to D after the network stabilizes, because $path(N_1, D) = (N_1, D)$ is shorter than $path(N_2, N_1) + path(N_1, D)$. By ignoring N_2's path, the router is ignoring a path that would change as soon as the pending update from N_1 arrives.

Note also that some invalid paths might be not invalidated by this approach. Take Figure 2(b) as an example, and suppose the link between X and D fails, thus D is not reachable for X, N_1, N_2 and R. X will send withdrawals to both N_1 and N_2. After receiving X's withdrawals, N_1 and N_2 will also send out withdrawal since all of their paths go through X. Now suppose N_1's withdrawal reaches R before N_2's does. R cannot mark $path(N_2, D) = (N_2, X, D)$ as infeasible although $path(N_2, D)$ is already invalid, since N_1 does not appear in $path(N_2, D)$. Then $path(N_2, D)$ is selected as the best path by R, and if R has peers other than N_1 and N_2, $path(N_2, D)$ may be further propagated. R will recognize D is not reachable after receiving N_2's withdrawal.

The above approach is formalized into the following two *Consistency Assertions* for $SPVP$.

C. Route Withdrawal and Route Change Assertions

In $SPVP$, assume node R has neighbors $N_1, N_2, ..., N_n$, $path(N_i, D)$ is the last path to D reported by neighbor N_i.

Definition** 5:* ***Route Withdrawal Assertion. If the latest update is that N_{lost} withdraws its path to D ($path(N_{lost}, D) = NULL$), then mark $path(N_i, D)$ as infeasible if N_{lost} appears in $path(N_i, D)$.

That is, upon receiving an withdrawal, we check whether any existing path to D is inconsistent with this withdrawal. If so, we mark the existing path as infeasible.

Note also that the Withdrawal Assertion is not applied as the result of link failure. For example, suppose the link between R and N_1 in Figure 2(a) fails. R will then remove $path(N_1, D)$, but this is not equivalent to a withdrawal from N_1. After this link failure, $(R, N2, N1, D)$ is a valid backup route that still indirectly goes through N_1.

Definition** 6:* ***Route Change Assertion. If the latest update is that N_{change} advertises $path(N_{change}, D)$, do the following:

- If N_{change} appears in $path(N_i, D)$ and $path_{N_i}(N_{change}, D) \neq path(N_{change}, D)$, then mark $path(N_i, D)$ as infeasible.
- If N_i appears in the $path(N_{change}, D)$ and $path_{N_{change}}(N_i, D) \neq path(N_i, D)$, then mark $path(N_{change}, D)$ as infeasible.

Upon receiving a new path, one uses the new path to check the feasibility of existing paths. Second, the existing paths are used to check the feasibility of the new route. Details on how to implement these two assertions are presented in Section V.

IV. Enhanced Assertions for BGP

Theorem 1 and Route Withdrawal/Change Assertions in Section III provide mechanisms for improving the convergence of $SPVP$. However, BGP does not fit completely into the $SPVP$ model. One may consider an AS as a node in $SPVP$, but in BGP there is typically more than one BGP router in an AS. Neighboring routers in the same AS are called *iBGP peers* and neighboring routers in other ASes are called *eBGP peers*. BGP peers differ from the SPVP model due to *traffic engineering, AS partition, policy withdraw*. Details on how to address these issues are discussed in the rest of this section.

A. Logical AS: Signal Traffic Engineering

One AS, through multiple BGP routers, may advertise multiple routes to one single destination in the Internet. For example the Oregon Route Views Server [13] shows that on 6/8/2001, AS 701 announced two different routes to prefix 169.131.0.0/16, in an attempt to better engineer the traffic to this destination. As a result, AS1 had learned the route (701 6079 4527) and AS1740 had learned the route (701 6347 4527). Further analysis of the data shows that 56,081 out of 121,602 prefixes, and 125 out of 11,514 ASes in the Internet are involved in traffic engineering from 07/10/2001 to 07/18/2001.

***Assumption** 1:* One single BGP router can only advertise to its peers *one single route* to one destination.

While AS traffic engineering does occur, Assumption 1 remains true in the Internet[1] and is the basis of the enhanced consistency assertions for BGP. To signal AS traffic engineering, we attach an additional attribute to the route: the ID of the *Entry Router*, i.e., the router who receives the route from eBGP peers or originates the route by itself.

In general, within one AS X there are multiple routes available to a destination d. When there is traffic engineering for d in AS X, different BGP routers may select and advertise different best routes. In this case, AS X is *virtually* divided into multiple ***logical ASes*** according to the $EntryRouter$ (RID) of the selected route. Each logical AS is uniquely identified in the Internet by the 2-tuple (X, RID). All the BGP routers in AS X that select the best route from the Entry Router RID belong to the logical AS $< X, RID >$. The logical AS is defined in the context of one particular destination, therefore one real AS may

have different divisions of the logical ASes for different destinations. No *Entry RouterID* is attached if a route is not traffic engineered, and the logical AS is the same as the real AS.

Assume that every router attaches the $Entry\ RouterID$ to the route if its AS is performing traffic engineering on the destination. According to Assumption 1, one BGP router can only advertise to its iBGP peers one route to one destination. Therefore, all the routers within a logical AS can only use (and advertise to their peers) one single route to one destination. Here we can find that the logical AS concept fits well into $SPVP$ model.

By taking a logical AS as a node in $SPVP$, we can apply the ***enhanced definitions of Valid Path, Consistent Paths, Feasible Path***. Similarly, we can apply ***enhanced Theorem 1 and Route Change Assertion***. Implementation details are discussed in Section V-B and V-C.

B. Failure Withdrawals and Policy Withdrawals

Section III defined an empty path to be valid in $SPVP$ since it reflects the neighbor node's latest view that it cannot reach the destination. However, this is not always true in BGP and there are two distinct causes for a BGP withdrawal message. A *failure withdrawal* occurs if an AS has lost its route to the destination. Failure withdrawals can occur due to the failure of a route imported from IGP, the close of the peering session with the upstream peer advertising the route, or a withdrawal received from the upstream peer. In all of these of cases, the existing route to the destination is no longer valid and the failure withdrawal conveys topology information that can be used to *invalidate* other routes(mark them as infeasible).

A *policy withdrawal* occurs if a change in route policy causes an AS to stop advertising a route to some of its neighbors. In this case, the upstream router still has its existing route to the destination but the upstream router no longer makes this route available to some peer(s). To determine whether a backup route is feasible, one must distinguish between failure withdrawals, which convey new topology information, and policy withdrawals, which must not be used to invalidate backup routes.

The ***enhanced version of the Route Withdrawal Assertion*** is obtained by taking a logical AS as a node in SPVP and limiting Route Withdrawal Assertions so they only apply to failure withdrawal. However, the BGP specification does not differentiate a failure withdrawal from a policy withdrawal so we extend the BGP protocol to signal failure/policy withdrawals. Details on how to do this are presented in Section V-D.

C. Addressing the AS Partitions

In some scenarios, an AS may become partitioned into several parts due to failure of internal links. As a result, routers in different partitions could choose different routes to one destination, or some routers could lose the route to the destination while others still have the route. This does not fit into the $SPVP$ model. Internet AS partitions should be rare and be fixed quickly, but in order to guarantee the correctness of our assertions, one should not apply the assertions to any withdrawals or new route changes that resulted from AS Partition. We assume there is already a mechanism to detect AS partition.

In an AS that is doing traffic engineering on the route to destination considered, one of its routers may lose a route due to the loss of the connectivity to the entry router of the route. In this case, the withdrawal it sends should be set as a policy withdrawal, since other routers in the same logical AS may still have the route. When a router advertises a new route due to the loss of the connectivity to the *Entry Router* of the previous route, the new route should have been already attached *Entry RouterID* of the new route.

In an AS that is not performing traffic engineering on the route to the destination, one of its routers may also lose a route due to AS partition. In this case, a policy withdrawal is sent. When a router advertises a new route due to the loss of the connectivity to the *Entry Router* of the previous route, the router should attach the *Entry RouterID* of the new route. The local logical AS in the new route will be treated as a different one from the logical AS in former route(with no $Entry\ RouterID$ attached), thus will not invalidate the former routes sent by other iBGP peers.

V. Implementation in BGP

This section shows how the enhanced assertions from Section IV can be implemented in BGP. In order to signal policy/failure withdrawal and traffic engineering, we need to modify the BGP UPDATE message format in a backward compatible way. We achieve this by defining and using new community attributes [14]. The community attribute is a 32-bit value, normally associated with route advertisements and used to convey routing policy information. For example, including a community the value of 0xFFFFFF02 with a route advertisement indicates that the route should not be advertised to other peers.

A. The BGP Routing Process

Neighboring BGP routers exchange messages using long lasting TCP[15] connections. The use of TCP insures the reliable delivery of messages and periodic BGP KEEPALIVE messages verify that the TCP connection is functioning properly. To announce a route to some destination, a BGP router sends an UPDATE message. A route advertisement UPDATE message includes the destination network (NLRI) and a number of *path attributes*, most notably the AS Path attribute that lists the path of ASes used to reach the destination. Additional UPDATE messages for this destination are sent only if the route's attributes change[1] or if the route is withdrawn. BGP withdrawal messages are sent by listing the destination in the withdrawn routes section of an UPDATE message.

A BGP router records the routes received from each of its peers in a table. The table for peer p is denoted $AdjRIB[p]$ and entry $AdjRIB[p][d]$ indicates the route peer p uses to reach

[1] A route refresh capability[16] has been added so a router may request the re-advertisement of a route.

destination d. After receiving a route advertisement for destination d (or a withdrawal for d), the corresponding entry in the $AdjRIB$ table is updated and the *BGP Decision Process* is run to determine the new route to d. For each peer p_i, the router calculates a preference for $AdjRIB[p_i][d]$. If no routes to d are available, the router will declare the destination unreachable. Otherwise, the best route to d is installed in the router's routing table. If the BGP Decision Process resulted in a new route to d (or if d has become unreachable), the router applies its routing policies and sends the appropriate UPDATE messages listing the new route to d.

As discussed in Section II, the BGP standard requires a minimum amount of time, *MinRouteAdver*, must elapse between route advertisements for a particular destination. The value of *MinRouteAdver* is recommended to be 30 seconds with a random jitter. In order to avoid long-lived black holes, this rate-limiting timer does not apply to withdrawals[1].

B. Implementation of Logical AS

To signal that a local AS is performing traffic engineering on the routes to a destination, each Entry BGP router should attach an $Entry\ RouterID$ community attribute to the route, whose format is defined as the following.

ASN	F	E=0	RID

where ASN is the 2-Byte AS number of local AS, F is the 1-Byte flag which will take a specific value to indicate that this community attribute will include the $Entry\ RouterID$ Information. If the 1-bit extension flag $E = 0$, the left 7-bit RID field will be the $RouterID$ of the router who is creating this community attribute. When 7-bit RID field is not enough to contain the $RouterID$, two consecutive $Entry\ RouterID$ community attributes are attached in the following format.

ASN	F	E=1	H-RID
ASN	F	E=0	L-RID

The first community attribute with Flag $E = 1$ gives the higher 7 bits of the $RouterID$, and the second one with $E = 0$ gives the lower 7 bits of the$RouterID$, allowing up to 16,256 distinct Router IDs.

When a BGP router receives an route with $Entry\ RouterID$ community attribute and selects the route, it should not modify this attribute and should propagate it when advertising the route to eBGP peers.

C. Implementing the Route Change Assertion

In order to implement the Route Change Assertion, the UPDATE processing and route selection algorithms must be changed. In the discussion below, the *logical AS number* of peer p_i is denoted as $AS(p_i)$ and the *logical AS path* associated with the route from p_i to d is denoted as $aspath(p_i, d)$.

A new table entry, $conflicts[p][d]$, is added into $AdjRIB$ to indicate whether a route is infeasible. If $conflicts[p][d] = NULL$, then this route does not conflict with the routes from any other peer. Otherwise, this route is $infeasible$ and $conflicts[p][d]$ lists the AS numbers of the conflicting peers. The infeasible routes cannot be selected as the preferred route to the destination.

After receiving a route advertisement for destination d from peer p_{change}, the $AdjRIB[p_{change}][d]$ table entry is updated and $conflicts[p_{change}][d]$ is initially set to $NULL$ and the following modified BGP decision process is run.

First, the new route is used to check the feasibility of existing $AdjRIB[p_i][d]$ entries. If $AS(p_{change}) \in aspath(p_i, d)$ and $aspath(p_i, d)$ does not end with $aspath(p_{change}, d)$, then the $AS(p_{change})$ is added to the set $conflicts[p_i][d]$. If $aspath(p_i, d)$ does end with $aspath(p_{change}, d)$ and $AS(p_{change}) \in conflicts[p_i][d]$, then $AS(p_{change})$ is removed from $conflicts[p_i][d]$.

Second, the existing $AdjRIB[p_i][d]$ entries are used to check the feasibility of the new route. If $AS(p_i) \in aspath(p_{change}, d)$ and $aspath(p_{change}, d)$ does not end with $aspath(p_i, d)$, then $AS(p_i)$ is added to the set $conflicts[p_{change}][d]$.

Finally, for the peers with $conflicts[p_i][d] = NULL$, the most preferred route is selected as the route to d and the BGP process continues in the normal way.

D. Implementing the Route Withdrawal Assertion

After receiving a withdrawal for destination d from peer p_{lost}, the $AdjRIB[p_{lost}][d]$ table entry is cleared. If the withdrawal is a failure withdrawal, and $AS(p_{lost})$ appears in any $aspath(p_i, d)$, then $AS(p_{lost})$ is added to $conflicts[p_i][d]$. $AS(p_{lost})$ is removed from any $conflicts[p_j][d]$ that contains it but $AS(p_{lost})$ doesn't appear in $aspath(p_i, d)$. If the withdrawal is a policy withdrawal, $AS(p_{lost})$ is removed from any $conflicts[p_i][d]$ that contains it.

To signal failure/policy withdrawal, a simple 1-bit withdrawal type flag would have been enough, but there are no reserved bits left in the BGP UPDATE message. Instead, a router signals a failure withdrawal by including a *failure withdrawal community attribute(FWCA)* (0x88888888, for example)in the BGP UPDATE message. If the *FWCA* is not present, then any withdrawn routes are assumed to be policy withdrawals.

This approach is compatible with existing implementations. If a router does not implement our approach, it will not include $FWCA$ and its withdrawals will always be considered policy withdrawals. The BGP capability negotiation process[17] is also used so that $FWCA$ is only sent to those routers who negotiate to receive it. Failure withdrawal UPDATEs consist of only a withdrawn routes part and $FWCA$, a valid UPDATE format. Route announcements are never included in failure withdrawal UPDATEs in order to avoid falsely associating $FWCA$ with the announced routes.

E. Example

A slightly simplified version of the example in Figure 1 is given below to illustrate how our enhanced BGP implementation

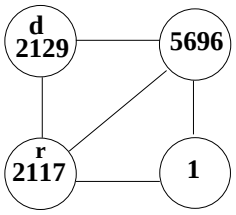


Fig. 3. An example to illustrate how the Route Withdrawal Assertion works

peer	AS Path	Conflicts
p2129	**2129**	NULL
p5696	5696 **2129**	NULL
p1	1 5696 **2129**	NULL

(a) Initial $AdjRIB$ values

peer	AS Path	Conflicts
p2129	NULL	NULL
p5696	5696 **2129**	2129
p1	1 5696 **2129**	2129

(b) After a failure withdrawal by AS 2129

peer	AS Path	Conflicts
p2129	NULL	NULL
p5696	5696 **2129**	NULL
p1	1 5696 **2129**	NULL

(c) After a policy withdrawal by AS 2129

Fig. 4. $AdjRIB$ values of router r after failure and policy withdrawals

would improve BGP route convergence. This example does not involve traffic engineering or AS partition, but does consider the policy withdrawal. In a network with topology shown in Figure 3, AS 2117 learned a route to d from AS 2129. When AS 2129 sent a withdrawal for d, AS 2117 first tried the route with AS path (5696, 2129). When that route was withdrawn, AS 2117 tried the route with AS path (1, 5696, 2129). For simplicity, let r be a router in AS 2117 and assume that r has three peers: peer $p2129$ in AS 2129, peer $p5696$ in AS 5696, and peer $p1$ in AS 1. Initially, r uses $p2129$ to reach destination d and the corresponding $AdjRIB$ entries are shown in Figure 4(a).

Now suppose that $p2129$ sends a failure withdrawal for d. This failure withdrawal creates conflicts for the (invalid) backup routes since both of the routes rely on AS 2129. The resulting $AdjRIB$ table is shown in Figure 4(b). Since the two backup routes contain conflicts, neither can be selected and router r declares d to be unreachable. Since AS 2129's route to d has failed, eventually AS 5696 and AS 1 will withdraw their routes to d and the conflicts will be removed. With only one route change (current path to unreachable) and virtually no delay, router r has correctly determined that d is unreachable.

Now suppose that AS 2129 implements a policy change and no longer advertises the route for d to AS 2117. In this case AS 2129 can still reach d, but the link between AS 2117 to AS 2129 can no longer be used to reach d. The policy withdrawal will not generate any conflicts and router r can switch to the (valid) backup route via peer $p5696$. The resulting $AdjRIB$ table is shown in Figure 4(c).

VI. Testbed Deployment and Simulation Results

To test the BGP convergence assertions, the assertions were implemented in MRTD[18] routing software and deployed in the FNIISC project's BGP testbed[19]. In addition, simulations were used to explore large topologies that could not be created in the testbed. The results show a substantial reduction in both convergence time and number of update messages exchanged.

A. Deployment in the Testbed

The testbed topology is shown in Figure 5 and each router belongs to a different AS. Routers A, B, C and D ran the enhanced MRTD that implements our approach. Routers H, I, and J ran the original MRTD. Currently MRTD applies the $MinRouteAdver$ timer to the withdrawals and uses *sender side loop detection* approach described in Section II. Neither of these two approaches is commonly implemented in commercial routers[2]. Therefore, we turned off both of them in all routers in the testbed in order to clearly compare our approach with results achieved using commercial routers.

To inject route failures, R first announced a route to a destination r. Two minutes later, the route was withdrawn. To inject route changes, R first announced a short path, then announced a much longer backup routes. The experiments were repeated many times to get the average results, which are summarized in the Figure 6.

In these experiments, as soon as B, C and D received the failure withdrawals from A, they would find that all the backup routes contain conflicts and declare the destination as unreachable. Similarly, as soon as B, C, and D received the route changes from A, they would find that all the backup routes contain conflicts and conclude the best routes are the new routes they received. Because the invalid routes were immediately marked as infeasible after receiving the route withdrawal or route change, the $MinRouteAdver$ timer did not affect the enhanced BGP in this experiment and there was a substantial reduction in the convergence time.

In order to further test the compatibility of our enhanced BGP and other BGP routers, we deployed the enhanced MRTD on the CAIRN Testbed[20]. The CAIRN testbed peers with the research Internet and no deployment problems were encountered during a week long test.

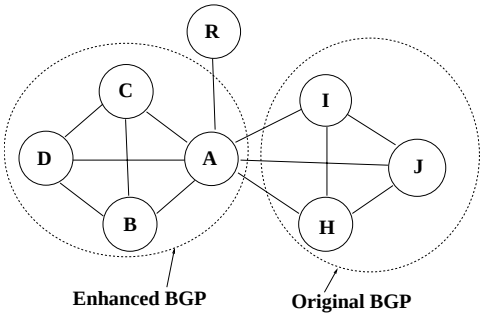


Fig. 5. Experiment Testbed Topology

	Original BGP	Enhanced BGP
Convergence time:	30.3 seconds	0.3 second
Number of Messages:	24	12

(a) Results for route failure

	Original BGP	Enhanced BGP
Convergence time:	64.9 seconds	0.1 second
Number of Messages:	24	12

(b) Results for route failover

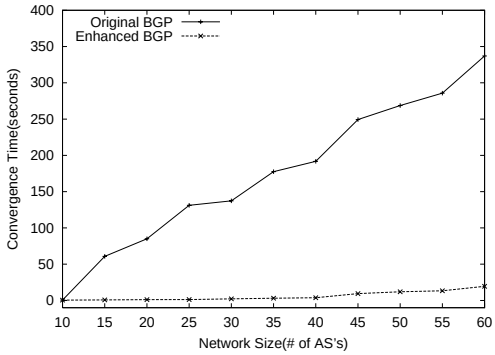
Fig. 6. Testbed Experiment Results

B. Simulation Results

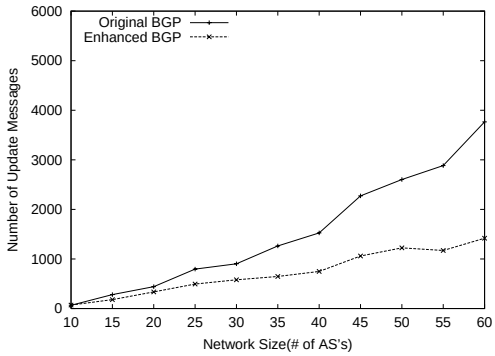
1) Simulation Setup: We implemented our assertions in the BGP protocol of SSFnet simulator[21]. We configured all $MinRouteAdver$ timers with 30 seconds and set the link delays to be 0.01 second. The CPU processing time of each message was randomly generated during simulation to be between 0.01 and 0.05 second. The total delay for processing one message was its CPU processing time plus the sum of delays of all the other messages that arrived before this message.

To generate the topologies used in the simulation, we first obtained a BGP routing table from the Oregon Route Views server [13]. Then we inferred BGP peering relationship based on the AS Path attributes in the BGP routes. For example, if a route to a prefix p has the AS Path (1239, 6453, 4621), we consider AS6453 to have two BGP peers, AS1239 and AS4621. We also mark AS6453 as a *Transit AS* since packets to and from AS4621 may traverse through it (note that AS1239 is also a transit AS). If an AS does not appear to be a transit AS in any of the routes, we consider it a *Stub AS*. Transit ASes are usually ISPs (e.g. AS1239 is Sprint), while stub ASes are networks at the edges of the Internet such as small organizations and universities. Next, we randomly selected $x\%$ of the stub ASes and constructed a topology containing these stub ASes and their peers, with the peering relationship among them completely preserved. We pruned transit ASes with too few peers to get the final topology. For the simulation, we modeled only one router in each AS and thus no traffic engineering or AS partition occurred.

We ran simulations for network topology sizes 10, 15, 20, $\cdots$, 60. For each network size, we randomly generated 5 topologies; for each topology, we randomly chose one stub AS as the *origin AS* of the destination 3 times. Therefore, each data point in the results is the average of 15 simulation runs. We simulated route failure and route failover, and measured both the convergence time and the number of update messages. It should be noted that both the Route Withdrawal Assertion and the Route Change Assertion apply to all the simulations.



(a) Convergence Time

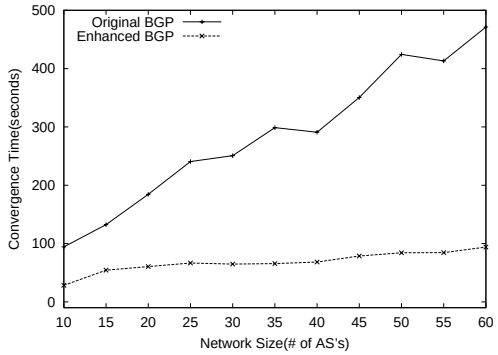


(b) Number of Update Messages

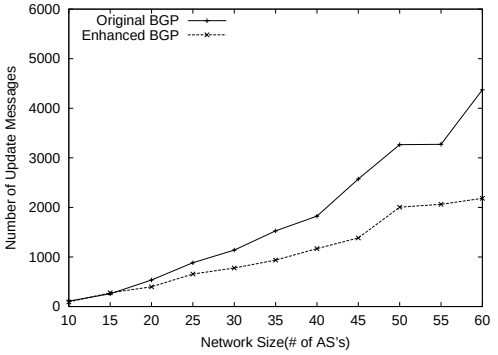
Fig. 7. Comparison of original and enhanced BGP for route failures

2) Route Failure: To simulate route failure, we randomly selected one stub AS with degree 1 (*i.e.* a stub AS with only one peer connecting it to network). This AS first announced a route to its peer, and after the whole network had converged on this route, the stub AS withdrew the route. We compared the convergence time and number of updates of original BGP with the enhanced BGP, as shown in Figure 7(a) and 7(b).

The convergence time and the number of update messages for original BGP increases greatly as the network size increases. Original BGP explores all the backup routes before convergence. Therefore, as the network size increases, more backup routes become available and the number of route changes also grows. A new route change has to wait for the $MinRouteAdver$ timer to expire before it can be sent out, therefore more route change



(a) Convergence Time



(b) Number of Update Messages

Fig. 8. Comparison of original and enhanced BGP for route failovers

leads to much longer convergence time. The convergence delay is exaggerated by the fact that more messages will incur longer CPU processing time thus longer delay.

As shown in Figure 7(a), the enhanced BGP improves the convergence time substantially. For example, in a 60-AS topology, the convergence time is reduced from 337.0 seconds to 19.5 seconds. However, as the network size increases, the convergence time of enhanced BGP does increase from 0.48 second in 10-AS networks to 19.5 seconds in 60-AS networks. We attribute this to two factors. First, the network diameter also increases as the network size increases, resulting in longer propagation delay from the origin AS to the farthest AS. The second factor is that more update message incurs longer delay and the number of messages does increase a lot for the reason described below.

Figure 7(b) shows that the number of messages also improves in enhanced BGP compared to original BGP. However, this improvement is not as dramatic as the reduction in convergence time. For example, in 60-AS networks, the number of messages is reduced from 3766 to 1419. One major reason is that removing a single update may cut convergence time by 30 seconds. A second reason is that $MinRouteAdver$ timer may have already queued and then reduced the number of updates substantially as described in section II, in both original and enhanced BGP. A third reason is that even with Withdrawal Assertion, it is still possible that an invalid route is selected and propagated as shown in Section III-B. Finally note also that our assertions only reduce the number of invalid update messages, but many valid update messages(e.g. withdrawals in route failures) still need to be propagated to achieve and confirm convergence. We confirmed above analysis by studying simulation log files.

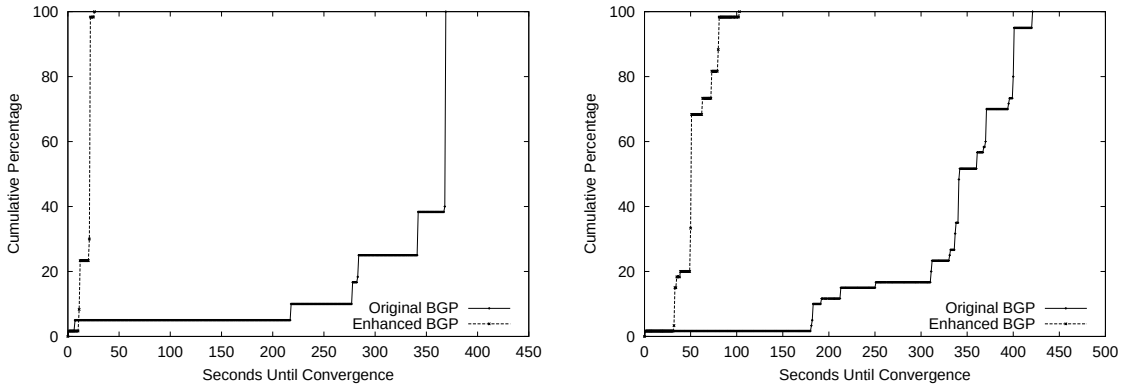
3) Route Failover: In order to measure the route change assertion, we simulated current Internet multihoming practice and created a route failover similar to the one described in [2]. We first randomly selected one stub AS with degree 2. This multihomed stub AS announced one short (primary) route to one of its two peers and announced a much longer (backup) route to the other peer. The backup route is created by prepending the stub AS's number 30 times, making the route long enough to ensure that every AS in the network would always prefer the primary route. When the primary route was withdrawn, only the backup path remained available and the routers converge to the backup path. We compared the convergence time and number of updates of original BGP with the enhanced BGP, as shown in Figure 8(a) and 8(b).

Substantial reductions of convergence time and number of messages are achieved in enhanced BGP. For example, the convergence time is reduced from 471.2 seconds to 93.9 seconds and number of messages is reduced from 4732 to 2183 in a 60-AS network. While the convergence time for original BGP grows rapidly as the network size increases, the convergence time for the enhanced BGP grows slowly and is usually between 30 seconds and about 90 seconds. Study of the simulation log files shows that convergence time for one single simulation run is usually about a multiple of 30 seconds (note this is due to the 30 second $MinRouteAdver$ timer).

These convergence times are longer than those seen in route failure experiments. In route failover, ASes have to propagate route announcements as opposed to propagating withdrawals. This is an important distinction since route updates are limited by the $MinRouteAdver$ timer, but the withdrawals are not. Therefore, the $MinRouteAdver$ timer plays more important role in route failovers and reduces the improvement of convergence time.

This is illustrated by Figure 9, in which a point (x, y) represents that at time x, totally y percent of the ASes have converged. A jump of the curve means that a large number of ASes converge in a short period. The jumps for route failures in Figure 9(a) are more significant than those of route failovers in Figure 9(b), for both original BGP and enhanced BGP. In route failures, as soon as an AS converges, it can send out a withdrawal to its peers, leading to more convergence and resulting the huge jump of the curve. However in route failovers, even though an AS has converged, it cannot send the update out until $MinRouteAdver$ timer expires. This could delay the convergence of its peers by up to 30 seconds.

4) Discussion: There are several simplifications in our simulations. We use only one router in one AS and thus do not



(a) Route Failure

(b) Route Failover

Fig. 9. Comparison of cumulative convergence percent for route failure and route failover

include the impact of IGP, traffic engineering, and AS partition. The simulated link delays may not a realistic approximate of the link delays in the Internet. During simulation, we also observed that different configurations of CPU processing delay lead to slightly different results, thus a better estimate model for it could be studied. Finally, although the network topology is derived from BGP routing table, its resemblance to the actual Internet could be studied more closely.

VII. Summary

Instead of blindly accepting all BGP UPDATEs, our basic approach is to let a router check route consistency using the information it has learned from previous updates and from other neighbors. In particular, in this work we used the information provided in the AS path to define route consistency assertions and used these assertions to identify infeasible routes. By taking this broader view and exploiting the relationships between routes, we were able to substantially reduce the BGP convergence time, as shown in our simulation results.

In this paper we showed how to implement our assertions in BGP and verified our design by developing a backward compatible implementation of the MRTD routing software. We demonstrated, through both simulation and measurement over our BGP testbed, that our approach reduces the BGP convergence time by up to 1 to 2 orders of magnitude.

Future work includes deriving traffic engineering information from Oregon Route Views Server and using this information in simulations and in experiments on testbeds with larger and more complex topologies.

VIII. Acknowledgments

We would like to express our appreciation to the people who reviewed an early version of this paper and pointed the need to accommodate traffic engineering and AS partition while solving the routing convergence problem. We also would like to thank the anonymous reviewers of this paper for their valuable comments.

References

[1] Y. Rekhter and T. Li, "Border Gateway Protocol 4," RFC 1771, SRI Network Information Center, July 1995.
[2] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet Routing Convergence," in *Proceedings of ACM Sigcomm*, Aug. 2000.
[3] C. Labovitz, "Delayed Internet Routing Convergence," http://www.research.microsoft.com/ labovit/, Aug. 2000.
[4] Christian Huitema, *Routing in the Internet*, Prentice-Hall, 2000.
[5] T. Griffin and B. Premore, "An Experimental Analysis of BGP Cvergence Time," in *Proceedings of ICNP*, Nov. 2001.
[6] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The Impact of Internet Policy and Topology on Delayed Routing Convergence," in *Proceedings of IEEE INFOCOMM*, Apr. 2001.
[7] K. Varadhan, R. Govindan, and D. Estrin, "Persistent Route Oscillations in Inter-Domain Routing," Tech. Rep. 96631, SRI Network Information Center, Feb. 1996.
[8] T. Griffin and G. Wilfong, "An Analysis of BGP Convergence Properties," in *Proceedings of ACM Sigcomm*, 1999.
[9] J.J Garcia-Lunes-Aceves and S. Murthy, "A Loop-Free Path-Finding Alogirthm: Specification, Verification and Complexity," in *Proceedings of the IEEE INFOCOM*, Apr. 1995.
[10] Pierre A. Humblet, "Another Adaptive Distributed Shortest Path Algorithm," *IEEE Transactions on Communications*, vol. 39, no. 6, pp. 999–1003, 1991.
[11] Z. Xu, S. Dai, and J.J. Garcia-Luna-Aceves, "A More Efficient Distance Vector Routing Algorithm," in *Proceedings of IEEE MILCOM*, Nov. 1997.
[12] Y. Afek and A. Bremler, "Self-Stabilizing Unidirectional Netowrk Alogirhtms by Power-Supply," in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, Jan. 1997.
[13] "The Route Views Project," http://www.antc.uoregon.edu/route-views/.
[14] R. Chandra, P. Traina, and T. Li, "BGP Communities Attribute," RFC 1997, SRI Network Information Center, Aug. 1996.
[15] Jon Postel, "Transmission Control Protocol," RFC 793, SRI Network Information Center, Sept. 1981.
[16] E. Chen, "Route Refresh Capability for BGP-4," RFC 2918, SRI Network Information Center, Sept. 2000.
[17] R. Chandra and J. Scudder, "Capabilities Advertisement with BGP-4," RFC 2842, SRI Network Information Center, May 2000.
[18] "MRTD: The Multi-Threaded Routing Toolkit," http://www.mrtd.net.
[19] "The FNIISC Project," http://fniisc.nge.isi.edu.
[20] "The CAIRN Testbed," http://www.cairn.net.
[21] "The SSFNET Project," http://www.ssfnet.org.