Speculative Route Invalidation to Improve BGP Convergence Delay under Large-Scale Failures

Amit Sahoo
Dept. of Computer Science
Univ. of Califonia, Davis
Davis, CA 95616
Email:asahoo@ucdavis.edu

Krishna Kant
Intel Corporation
Hillsboro,OR 97124
Email: krishna.kant@intel.com

Prasant Mohapatra
Dept. of Computer Science
Univ. of Califonia, Davis
Davis, CA 95616
Email:pmohapatra@ucdavis.edu

Abstract— **The Border Gateway Protocol (BGP) has been known to suffer from large convergence delays after failures. We have also found that the impact of a failure rises sharply with the size of the failure. In this paper we present and evaluate a speculative route invalidation scheme aimed at reducing the convergence delays for large-scale failures. Our scheme collects statistics from BGP updates received at a router and identifies Autonomous Systems (ASes) that are likely to be "unstable". Routes that contain these ASes are marked as invalid and not propagated further. This cuts down the number of invalid routes during the convergence process and results in a significant improvement in the convergence delay.**

I. Introduction

The Internet is composed of a large number of independently administered networks (also known as autonomous systems). Packets are routed between different autonomous systems (ASes) by the Border Gateway Protocol [1]. BGP has been used as the primary inter-domain routing protocol in the Internet over the last ten years or so and the excellent scalability of BGP has been a major facilitating factor for the explosive growth of the Internet. However BGP does have its share of shortcomings and the most important one is the long convergence delay (or recovery time) after failures in the network. The failure of network elements can cause BGP to suffer from extended periods of instability during which numerous BGP routers modify their routing tables.

BGP employs the path-vector routing algorithm in which each router sends the best route for each destination to each of its BGP neighbors. The router also stores all the routes that are sent by its neighbors and selects the best route for each destination according to some policy. If it receives a notification that the best route for a destination has failed, then it switches to the next best route and advertises this route to the neighbors. However there is no guarantee that the backup route is still valid, and the process has to be repeated all over again if the backup route is later found to have failed as well. This can result in a considerable delay before the cycle of withdrawals/advertisements comes to an end and all BGP nodes have a valid and stable path to each destination.

Numerous studies [2], [3], [4], [5], [6] have been carried out to study the fault tolerance and recovery characteristics of BGP. In particular, it was shown by Labovitz et al. [3] that the convergence delay for isolated route withdrawals can be > 3 minutes in 30% of the cases. They also found that packet loss rate can increase by 30x and packet delay by 4x during recovery. Furthermore, we have shown in our previous work [7] that multiple simultaneous failures can cause the convergence delay to increase significantly. Though large-scale failures might be rare, they have the potential to cause widespread disruptions to e-commerce as well as critical services such as emergency response, banking, infrastructure monitoring etc. Hence the importance of a short convergence delay after failures (especially large ones) cannot be overstated.

In this paper we present a scheme aimed at decreasing the convergence delay of large-scale failures by reducing the number of invalid routes explored during the convergence process. We do this by collecting statistics about withdrawn/replaced routes and by using this information to estimate whether a route is valid or not. If a route is suspected to be invalid, then it is not propagated further. Our scheme is designed to run independently at each BGP router/AS. Hence coordination between various ASes is not required and performance gains can be achieved with partial deployment. Our experiments show that we can reduce the convergence delay substantially for large-scale failures without causing any significant deterioration for other scenarios.

In this section we have talked about BGP and the need for reducing the convergence delay incurred by BGP after a failure. In the next section, we talk about schemes that try to identify the cause of BGP route updates, and their suitability for real time application. We present and discuss our speculative invalidation scheme in Section III. In Section IV we describe our experimental setup followed by the results and analysis in Section V. We wrap up with the conclusion in Section VI, and the references.

II. Related Work

There have been a few proposals that attempt to identify topological changes using BGP updates. Chang et al. [8] collected BGP updates from multiple vantage points and used a two level clustering scheme to group the updates into events. The first clustering scheme groups updates that were received at the same observation point, refer to the same prefix and are closely spaced in time. The second clustering scheme takes the first-level clusters from multiple vantage points and groups those that might be caused by the same event. The authors studied various characteristics of BGP events, such as the duration of the instability, the number of prefixes affected, etc. Caesar et al. [9] presented a scheme to identify the cause of "major" BGP events. Whenever a BGP route for a prefix is changed, suspect elements (ASes or links) and the corresponding event at that element are identified. An AS or link is considered to be of either major or minor significance and marked accordingly; based on the number of prefixes for which the route might have changed because of an event at this element. After all the elements have been marked, a set of rules are used to estimate the location. Finally the intersection of the results from various views can be computed to narrow down the set of responsible network elements.

Feldmann et al. [10] and Lad et al. [11] also proposed schemes similar to the ones above. All these schemes attempt to identify the cause of the BGP updates. Once the cause has been identified, we can just avoid the invalid routes and thus the BGP convergence delays can be reduced greatly. However, none of these schemes are suitable for real time application because of the high processing overhead associated with them. As BGP updates keep coming in, the algorithms would have to be run repeatedly and that is not feasible in real time. Furthermore, all the schemes except the one proposed by Caesar et al. require data to be collated from multiple views; and that complicates things further. Our scheme employs a approach similar to the ones above, but it has a much lower overhead and operates independently at each BGP router.

III. Speculative Invalidation Scheme

As we mentioned earlier, the primary cause of long convergence delay is the absence of information in BGP about the validity of routes. Our goal is to expedite the BGP convergence process for large-scale failures by identifying invalid routes and reducing their effect on the convergence process. We identified the following constraints that must be satisfied by any candidate scheme:

- All the analysis has to be done independently at a BGP router. Communication between routers will lead to additional delays and the scheme will have to deal with lost packets. The deployment would also have to be coordinated between multiple ASes/organizations.
- The scheme should have a low processing overhead, so that the timely processing of BGP updates in not affected.
- The convergence delay for situations other than large-scale failures should not be increased.

Our scheme leverages the fact that large-scale failures generate a large number of updates that can be used to identify the failed/affected ASes. It is also based on the assumption that failed/affected ASes will be present in a large number of the generated updates. There can be scenarios where this assumption does not hold true [12]. However, these scenarios are known to be rare [13], and it is very unlikely that a large number of updates will be generated due to these events. Similarly, addition of new links or peerings, and policy changes lead to BGP updates, but again the volume can be expected to be low.

Our scheme maintains a *failCount* for each AS at each BGP router. This value is incremented when a route containing that AS is withdrawn or replaced. It is decremented when we receive a new route containing that AS. When an AS suffers total or partial failure, a large number of routes containing that AS are likely to be withdrawn. Hence, we consider the *failCount* to be a measure of the likelihood that an AS has failed. We then select the ASes that have the largest *failCount*s and consider all routes that contain those ASes to be "invalid". As the overhead of examining all the stored routes will be too high, we install route filters which reject new route advertisements that contain any of the "suspect" ASes. We only want to consider the recent history for selecting the "suspect" ASes. Therefore we divide the time into slots and use a configurable parameter, *history*, to determine how many slots of data we want to consider. BGP updates are sent periodically, and the period is determined by the MRAI [1]. We select the length of the time slot to be equal to the MRAI (after adjusting for jitter [1]), so that we receive one cycle of update messages in each slot. A larger slot length will only increase the response time for our scheme. A smaller value will decrease the response time but there could be a lot of variation in the number of messages received in two contiguous slots and that might lead to instability.

The *failCount*s for all the ASes are stored in a two dimensional array *failCounts[numASes][history]*. At the beginning of each time slot, we add up the *failCount*s for the last *history* slots for each AS, and then use the *avgFailedPathLength* parameter to determine the number of ASes that will be "invalidated". *avgFailedPathLength* is the average path length of the routes replaced/withdrawn during the last *history* time slots. If we assume conservatively that each replaced/withdrawn route contains one failed/unstable AS; then after the replacement/withdrawal of a route, the *failCount* of one impaired AS and (*avgFailedPathLength*-1) stable ASes is incremented on average. If the sum of the *failCounts* for all ASes at an observation point is *X*, then the sum of the *failCounts* of the failed/unstable ASes can be expected to be close to *X/avgFailedPathLength*. Therefore, starting from the front (highest *failCountSums*), we identify the smallest set of ASes for which the sum of *failCountSums* is greater than this value (*X/avgFailedPathLength*), and consider all the ASes in this set to be "suspect" for the duration of the time slot. The pseudocode for our scheme is listed in Algorithm 1. As it might be difficult to identify the failed/unstable ASes correctly if the amount of data available for analysis is small, we execute the invalidation scheme only if the number of destinations for which the route has changed is greater than *largeFailureThresh*.

Our scheme manages to keep both the storage and processing overhead low. The overhead of incrementing/decrementing the *counts* and of running the new routes through the filters can be expected to be much lower than the overall processing overhead for a BGP update. The overhead of sorting the *counts*

Algorithm 1 Invalidation Scheme

```
1:  if (Number of destinations for which the routes has changed in the last history slots) < largeFailureThresh then
2:      Stop
3:  end if
4:  for $i = 1$ to $numASes$ do
5:      failCountSums[i] ← $\sum_{j=1}^{history} failCounts[i][j]$
6:  end for
7:  sumFailCounts ← $\sum_{i=1}^{numASes} failCountSums[i]$
8:  Sort failCountSums in descending order
9:  failThreshold ← sumFailCounts / avgFailedPathLength
10: currentSum ← 0
11: for $i = 0$ to $numASes$ do
12:     if currentSum < failThreshold then
13:         Add a filter to deny routes containing the AS at position i
14:         currentSum ← currentSum+ failCountSums[i]
15:     end if
16: end for
```

and installing the filters will also have little effect because these activities are only carried out once every time slot. The storage overhead will only be about 100 KB, with a *history* of two, as the *failCount* for each timeslot can be safely stored in two bytes per AS. The storage requirements can be decreased further at the cost of computational complexity if we use dynamic instead of static storage.

Our scheme does have a few issues that we need to work around. A new route is marked invalid if it contains a "suspect" AS. We need a way to remove the "invalid" flag if the AS in question is no longer considered to be "suspect". For this purpose, we need to maintain a list of routes that have been invalidated because of a particular AS. If this AS is not "suspect" anymore, then we can retrieve the corresponding list and validate the routes. We believe that the overhead for this process can be restricted to a manageable level, as we only need to do this once every time slot. Another issue is the possibility that all the routes for a destination are marked invalid. In such a scenario, we store the destination in a list and check for the existence of a valid route at the beginning of the next time slot. As the scheme runs independently at each router, the set of "suspect" ASes at two BGP routers in the same AS might be different, which in turn might cause them to select different routes for the same destination. To solve this problem, we consider the routes received from internal BGP peers to be always valid. This means that all the routers in an AS will have the same set of candidate route for each destination, and hence will select the same "best" route as long as they employ the same policy.

IV. Evaluation Methodology

A modified version of BRITE [14] was used for topology generation and BGP simulations were carried out using SSFNet [15].

A. Topology Generation

BRITE can generate topologies with a configurable number of ASes and with multiple routers in each AS. BRITE supports a couple of schemes that try to generate a power-law degree distribution, however the results are generally not satisfactory if the number of nodes (ASes) is less than a thousand. We therefore modified BRITE to allow more flexible degree distributions. This also enabled us to work in more controlled settings (e.g. uniform degree).

We used 200 node topologies for our experiments. This was dictated in part by memory restrictions on the 32 bit machines that we used. For our experiments, we predominantly used topologies with a "realistic" degree distribution, derived from the actual degree distribution for inter-AS links [16]. The average measured inter-AS degree from the Internet AS-level topology is about 8.0 [16]. However, the Internet has over 22000 ASes and the maximum inter-AS degree is in the thousands. For our 200 AS network we decided to restrict the maximum degree to 50. The resultant degree distribution decays as a power law with an exponent of about -1.85. and has an average degree of about 4. For all links, we used a one way delay of 25 ms.

Although large-scale failures could be scattered throughout the network, many failure scenarios (e.g., those caused by natural and man-made disasters) are expected to be geographically concentrated. We randomly placed all the routers on a 1000x1000 grid and then considered failures in contiguous areas of the grid (usually the center of the grid to avoid edge effects). We assumed that all routers and links in the failed area become inoperative. We experimented with failure magnitudes of 1 to 20%. While we are primarily interested in the 1 to 10% range, we included 20% failures to emphasize the trends.

B. BGP Simulation

We used the SSFNet simulator for our experiments because it has been used extensively in the research community for large-scale BGP simulations. In the simulations, the *path length* (i.e., number of hops along the route) was the only criterion used for selecting the routes and there were no policy based restrictions on route advertisements. All the timers were jittered as specified in RFC 4271 [1] resulting in a reduction of up to 25%. In our experiments, the MRAI timer was applied on a per-peer basis (as is commonly done in the Internet) rather than a per-destination basis. The MRAI timer was set to 30 seconds for external BGP peers and 0 for internal BGP peers. We simulated processing delays for BGP updates, but the delays were were much lower than the external BGP MRAI and hence can be expected to have little effect on the recovery time.

V. Results

In this section we present and analyze the results of our experiments. We used 200 node/AS (1 router per AS) topologies with "realistic degree" distributions (average degree 4) and a *history* of one time slot for our experiments, unless stated otherwise. We found that our scheme performs best

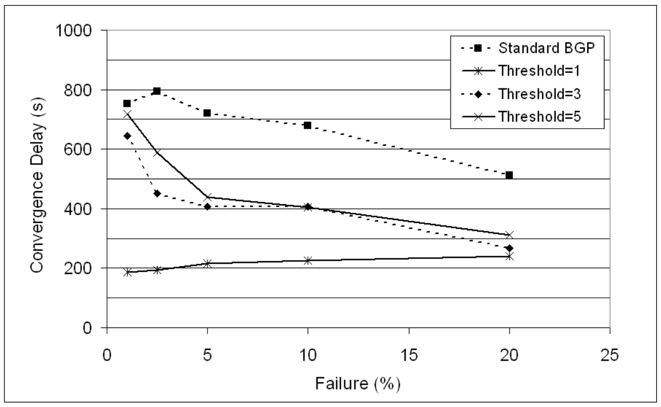

Fig. 1. Convergence delay with invalidation scheme

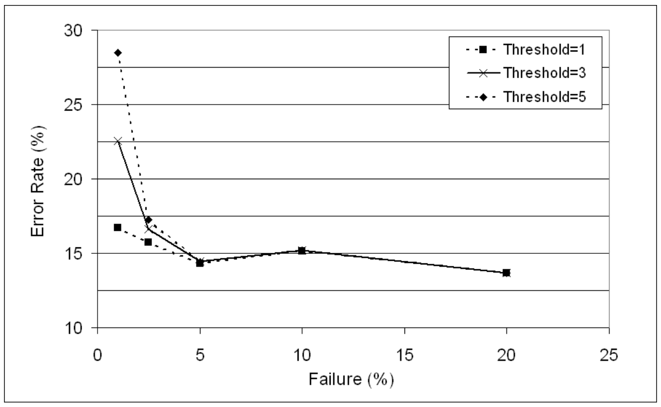

Fig. 2. Number of valid routes marked as invalid

when we implement it at "high" degree nodes only. By default, the results that we present here were obtained with the scheme implemented at the nodes/ASes with a degree of greater than 4. We study the consequences of this type of partial deployment of our scheme later on in this section.

We first compare the convergence delay for standard BGP and our invalidation scheme in Fig. 1. We experiment with *largeFailureThresh* values of 1, 3 and 5. We plot the magnitude of the failure (in terms of the fraction of routers failed) on the x-axis and the convergence delay (in seconds) on the y-axis. We can see that there is a significant improvement in the convergence delay for each failure magnitude when we use the invalidation scheme. We also see that the performance improves as *largeFailureThresh* is decreased. A lower *largeFailureThresh* means that our algorithm will be executed more often and more ASes will be marked as "suspect", which in turn causes a higher fraction of invalid routes to be removed. This leads to a quicker convergence process. For large failures, routes are changed for a large number of destinations, and hence the difference in the number of invocations of our scheme (for different values of *largeFailureThresh*) is decreased. Thus the convergence delays start to converge (for different values of *largeFailureThresh*) as the size of the failures is increased.

There is no guarantee that our scheme will identify the failed/unstable ASes correctly. In case that we mark a stable AS as "suspect", we will deny any routes that contain that AS (for that time slot). In order to study the correctness of the decisions made by our scheme, we kept track of: the number of routes that are denied, the number of valid routes are denied, and the total number of invalid routes. We plot the *error rate* (valid routes denied/total routes denied) of our scheme in Fig. 2. For the lowest possible value of *largeFailureThresh* (=1), we see that the *error rate* declines slowly. This happens because the amount of data available for analysis increases with the size of the failure and thus the validity of the deductions is improved. The actual number of valid routes that are denied ranged from about 2 (1% failure) to about 10 (20% failure) per AS over the convergence period. As we increase the *largeFailureThresh*, we were surprised to see that the *error rate* went up for the smaller failures. The reason for this behavior is clear from Fig. 3, in which we plot the fraction of invalid routes that are denied (*efficiency*). As we can see, the *efficiency* for "threshold=1" is pretty much constant for different failure sizes. If we increase the threshold, the amount of data analyzed per invocation (of our scheme) is increased; however the total amount of data analyzed across the network goes down because our scheme is executed less often. Thus for smaller failures, the effectiveness of our scheme is decreased leading to lower *efficiency* and a higher *error rate*. From these results it is clear that we get the best performance when we set *largeFailureThresh* to 1, and that is what we used for the rest of the results in this section.

The performance of our scheme is dependent on the number of updates received at a router, which is determined in part by the degree of the node/AS. We evaluate the performance of our scheme when it is implemented only at those nodes whose degree exceeds a certain value (*highDegreeThreshold*), in Fig. 4. We see that for the degree distribution we used for the experiments, the convergence delay is decreased by up to 75% with our scheme running at just 20% of nodes (*highDegreeThreshold*=4) by upto 50% with the scheme running at just 10% of nodes (*highDegreeThreshold*=8). This is significant because we can get sizeable benefits without implementing our scheme at smaller ASes that typically have limited connectivity. The convergence delay increases with the *highDegreeThreshold* because the fraction of the invalid routes that we are able to identify (*efficiency*) decreases (in general) when we run it at fewer nodes. For example, *efficiency* is equal to about 45% when *highDegreeThreshold* is equal to 4, but the *efficiency* drops to the 30 to 35% range when *highDegreeThreshold* is increased to 8. However on the brighter side, increasing the *highDegreeThreshold* from 0 to 8, drops the maximum *error rate* from 20 to 10%. Thus, a lower *highDegreeThreshold* increases the probability that we mistakenly consider some routes as invalid. In order to study that effect, we measure another parameter that we call "lost connectivity".

After a failure, connectivity to some destinations is truly lost. However other destinations might not be accessible

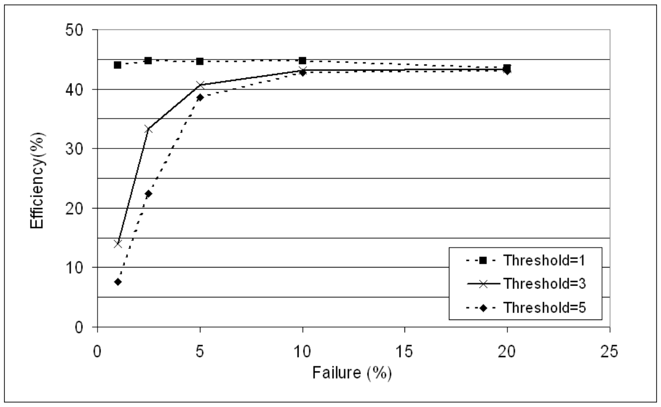

Fig. 3. Number of correctly identified invalid routes

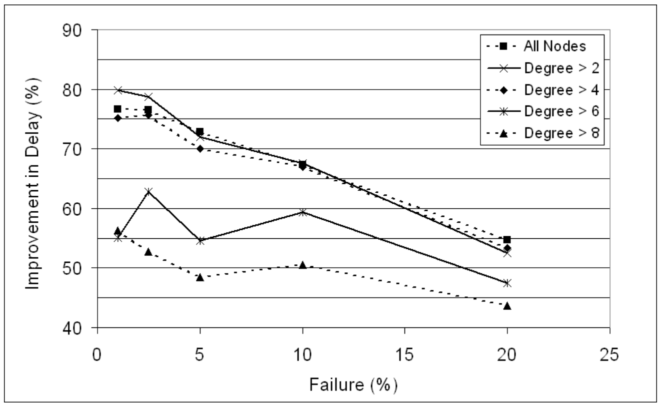

Fig. 4. Effect of deploying the invalidation scheme on a subset of nodes

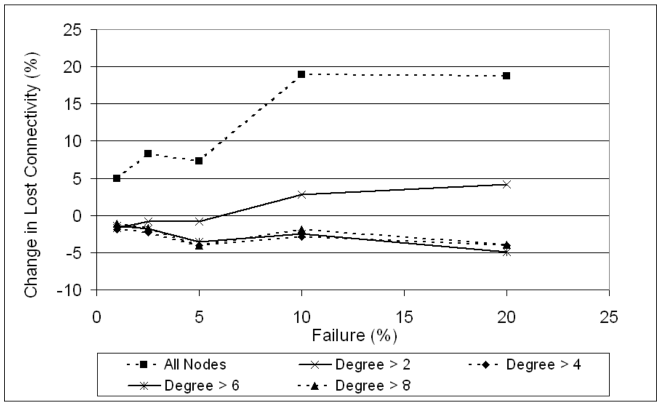

Fig. 5. Effect of *highDegreeThreshold* on “Lost Connectivity”

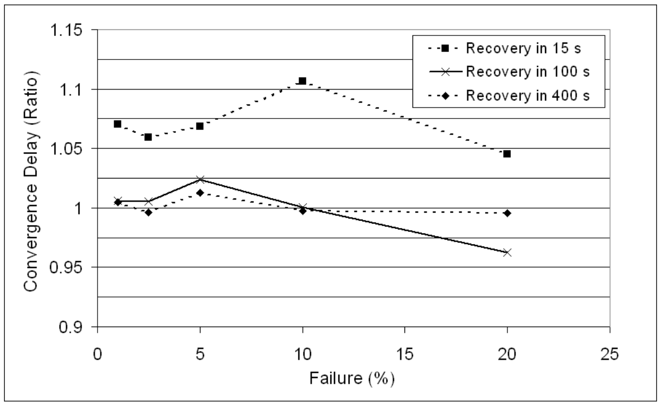

Fig. 6. Convergence delay for recovery scenario

for some time even though there exists a valid path from the source. For example, standard BGP might prefer shorter invalid routes over longer but valid routes. Similarly in our scheme, a router might not have a path to an AS because all the candidate routes are mistakenly considered to be invalid. The “lost connectivity” at a router is the cumulative duration of the transient “loss” of connectivity. We measure the “lost connectivity” for all routers in the network and compute the average. We show the difference in the “lost connectivity”, as compared to standard BGP in Fig. 5. We observe that with a *highDegreeThreshold* of 4 or higher, the “lost connectivity” is improved marginally. However, for *highDegreeThreshold* equal to 0, there is a significant increase from the standard case. That is because at low degree nodes, the probability of mistakenly marking a neighbor as “suspect” is high since each neighbor is present in a large number or routes (and withdrawals). However we see that the convergence delay can be improved significantly without increasing the “lost connectivity” if we run our invalidation scheme at the high degree nodes only. For the remaining results in this section, we use a *highDegreeThreshold* of 4.

We also evaluated the performance of our scheme on some other topologies. We used topologies with linearly decreasing degree distributions and average degrees of 4 and 8. We again found that with a *highDegreeThreshold* of 4, our scheme provided big improvements in convergence delay without any significant increase in the “lost connectivity”. We would like to verify the effectiveness of our scheme for a larger number of topologies and study the factors, if any, that affect the ideal value for the *highDegreeThreshold*. That work is ongoing.

We have seen that our scheme works well for large scale failure scenarios. However this improvement in convergence delay should not come at the cost of worse performance in scenarios where the failed routers and links “recover” after some time. All the routes are valid after the recovery. However our scheme might still consider some routes to be invalid, because of the length of the time slot or because a large number of updates are still flying around. We show the ratio of the convergence delays for our scheme and standard BGP, for different recovery scenarios in Fig. 6. Markopolou [17] et al. measured the “time to repair” for router failures and found the median value to be about 400 seconds. If we recover the failed nodes/ASes after 400 seconds, the convergence delay for our scheme is effectively equal to that for standard BGP. For a recovery time of 100 seconds, the difference is within a few percent. For a recovery time of 15 seconds (highly unlikely for large-scale failures), the convergence delays are moderately greater ($\sim$10% at the higher end) than the base

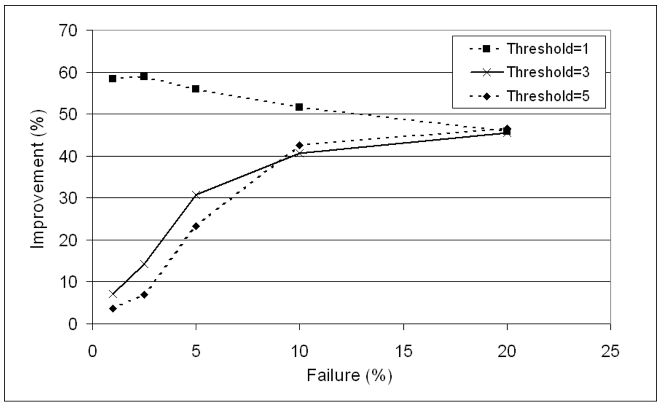

Fig. 7. Convergence delay for multi-router ASes

case. However even in that scenario, the deterioration in the convergence delay is significantly less than the improvement that we observed in Fig. 1.

All the results that we have looked at till now, used a *history* of 1 time slot. It makes the most sense to have a history of 1 time slot, because we will be basing our decisions on the most recent data that we have. A longer *history* does provide us more data, but that data could be stale. We experimented with other values of *history* and found that the performance went down as *history* is increased. We also found that the *error rate*, which we talked about previously, went up with the *history*.

Finally we evaluated the performance of our invalidation scheme for topologies with multiple routers per AS. We selected the number (1-100) of routers in an AS from a heavy tailed distribution. The details of the topology generation can be found in our previous work [7]. In Fig. 7 we plot the improvement in the convergence delay for the multirouter-AS topology. The trends are similar to what we observed in Fig. 1, with the lowest *largeFailureThresh* providing the best results. We also see that the improvement in the delay is somewhat less than what we observed in Fig. 1. This can be attributed in part to the possibility of partial AS failures. A failure could affect the capability of an AS to provide conduit to one destination but not to another. However, our scheme is not sophisticated enough to make such distinctions. Despite this issue, we are able to reduce the convergence delays by more than 50%.

VI. Conclusion

In this paper we proposed and evaluated a speculative invalidation strategy, designed to reduce the convergence delay for large-scale failures. We identify failed/unstable ASes by analyzing the BGP updates received at a router. Routes containing these "suspect" ASes are then removed from contention. This action goes a long way in reducing the uncertainty, about the validity of routes, faced by BGP during the convergence process. As a result the convergence delays are reduced greatly. At the same time, there is no significant increase in the convergence delay for realistic recovery scenarios. As in any speculative scheme, errors do occur. However the errors can be minimized and all the benefits of our scheme can be obtained if we implement it at "high degree" nodes only. We also introduced a new parameter, "lost connectivity", to measure the impact of a failure. This parameter is a measure of the total loss of connectivity (between all possible source-destination pairs) as a result of the failure.

Currently we are looking at ways to improve the performance of our invalidation scheme further. The algorithm that we used to identify the "suspect" ASes is fairly lightweight. It is also very general as we do not make any assumptions about the preference functions used to select the best routes. We are considering more complex (both in terms of time and space) schemes that might be able to provide even better results, possibly by independently analyzing updates sent by a neighbor or updates sent for a group of prefixes. We are also exploring strategies to reduce the "lost connectivity" during a failure.

References

[1] Y. Rekhter, T. Li, and S. Hares, "Border Gateway Protocol 4," RFC 4271, Jan. 2006.

[2] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet Routing Instability," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 515–528, Oct. 1998.

[3] Labovitz, C., Ahuja, et al., "Delayed internet routing convergence," in *Proc. ACM SIGCOMM 2000*, Stockholm, Sweden, Aug. 28–Sep. 1, 2000, pp. 175–187.

[4] Dan Pei, B. Zhang, et al., "An analysis of convergence delay in path vector routing protocols," *Computer Networks*, vol. 30, no. 3, Feb. 2006, pp. 398–421.

[5] T.G. Griffin and B.J. Premore, "An experimental analysis of BGP convergence time," in *Proc. ICNP 2001*, Riverside, California, Nov. 11–14, 2001, pp. 53–61.

[6] D. Obradovic, "Real-time Model and Convergence Time of BGP," in *Proc. IEEE INFOCOM 2002*, vol. 2, New York, Jun. 23–27, 2002, pp. 893–901.

[7] A. Sahoo, K. Kant, and P. Mohapatra, "Characterization of BGP recovery under Large-scale Failures," in *Proc. ICC 2006*, Istanbul, Turkey, June 11–15, 2006.

[8] Di-Fa Chang, R. Govindan, and J. Heidemann, "The Temporal and Topological Characteristics of BGP Path Changes," in *Proc. ICNP 2003*, Atlanta, Georgia, Nov. 4–7, 2003, pp. 190-199.

[9] M. Caesar, L. Subramanian, R. Katz, "Towards localizing root causes of BGP dynamics," U.C. Berkeley Technical Report UCB/CSD-04-1302, November 2003

[10] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating Internet Routing Instabilities," in *Proc. ACM SIGCOMM 2004*, Portland, Oregon, Aug. 30–Sep. 3, 2004, pp. 205-218..

[11] M. Lad, A. Nanavati, D. Massey, and L. Zhang, "An algorithmic approach to identifying link failures," in *Proc. IEEE PRDC*, Papeete, Tahiti, Mar. 3–5, 2004, pp. 25-34.

[12] R. Teixeira and J. Rexford, "A measurement framework for pinpointing routing changes," in *Proc. ACM SIGCOMM workshop on Network troubleshooting*, Portland, Oregon, Sep. 3, 2004, pp. 313–318.

[13] F. Wang, L. Gao, "On inferring and characterizing internet routing policies," in *Proc. IMC 2003*, Miami, Florida, Oct. 27–29 2003, pp. 15–26.

[14] A. Medina, A. Lakhina, et al., "Brite: Universal topology generation from a user's perspective," in *Proc. MASCOTS 2001*, Cincinnati, Ohio, August 15–18, 2001, pp. 346-353.

[15] "SSFNet: Scalable Simulation Framework". [Online]. Available: http://www.ssfnet.org/

[16] B. Zhang, R. Liu, et al., "Measuring the internet's vital statistics: Collecting the internet AS-level topology ," *ACM SIGCOMM Computer Communication Review*, vol. 35, issue 1, pp. 53–61, Jan. 2005.

[17] A. Markopoulou, G. Iannaconne, S. Bhattacharrya, C-N. Chuah, and C. Diot, "Characterization of Failures in an IP Backbone," in *Proc. IEEE INFOCOM 2004*, vol. 4, Hong Kong, Mar. 7–11, 2004, pp. 2307-2317.