Public Review for

YAMR: Yet Another Multipath Routing Protocol

Igor Ganichev, Bin Dai, P. Brighten Godfrey, and Scott Shenker

This paper presents an Interdomain multipath routing protocol. Multipath routing has been shown to be a promising technique to increase reliability of Internet routing and provide greater flexibility for users to choose paths that best suit their needs. In recent years, many schemes have been proposed to support multipath interdomain routing. These existing works demonstrated the feasibility of providing alternative paths in a scalable and policy-compliant manner. In this work, authors propose "Yet Another Multipath Routing Protocol" (YAMR). The contributions of this paper are two-fold. The paper first presents a path calculation mechanism called YPC. YPC constructs a set of policy-complaint interdomain routing paths which tolerant any single interdomain link failure. However, the increase in path resiliency comes at a price: a considerable increase in the control message overhead imposed by alternative path advertisement. The paper then presents a mechanism which reduce control message overhead by localizing routing updates. The idea is that a router does not notify its neighbors failure event if it still has an available path to reach the same destination. This method can effectively reduce the control message overhead. To avoid forwarding loops and disconnectivity due to information hiding, a token is used as a signal to determine if information hiding is needed or not. All reviewers found that this failure hiding mechanism is quite interesting.

The paper uses simulations to evaluate the benefit of YAMR and compares it with BGP. The results show that YAMR not only outperforms BGP in terms of reliability, but also bests BGP in terms of control message overhead (a benefit from failure hiding). These results clearly demonstrate that YAMR is a promising technique for multipath interdomain routing. The current evaluation is based on synthetic topology. It would be interesting to see how YAMR performs on the real Internet topology. In addition, there are many interesting performance tradeoffs that clearly need careful exploration. For example, YPC requires each router to keep a set of alternative forwarding paths. This would incease router FIB size consumption. This can be a concern that impacts the routing scalability. What would be the ideal tradeoff between the increase in path diversity and the increase in router FIB size? The failure hiding mechanism can lead to inconsistency between the control plane and the data plane. To what extend would this inconsistency impact the failure resilience and existing routing dependent systems on the Internet? All of these tradeoffs warrant further exploration.

Public review written by
Jia Wang
AT&T Labs – Research, USA

acm sigcomm

YAMR: Yet Another Multipath Routing Protocol

Igor Ganichev
Computer Science Division
Univ. of California, Berkeley
igor@cs.berkeley.edu

Bin Dai*
School of Computer
National University of
Defense Technology, China
bin.danieldai@gmail.com

P. Brighten Godfrey
Dept. of Computer Science
Univ. of Illinois at
Urbana-Champaign
pbg@illinois.edu

Scott Shenker
UCB & ICSI
shenker@icsi.berkeley.edu

ABSTRACT

Multipath routing is a promising technique to increase the Internet's reliability and to give users greater control over the service they receive. However, past proposals choose paths which are not guaranteed to have high diversity. In this paper, we propose yet another multipath routing scheme (YAMR) for the interdomain case. YAMR provably constructs a set of paths that is resilient to any one inter-domain link failure, thus achieving high reliability in a systematic way. Further, even though YAMR maintains more paths than BGP, it actually requires significantly less control traffic, thus alleviating instead of worsening one of the Internet's scalability problems. This reduction in churn is achieved by a novel hiding technique that automatically localizes failures leaving the greater part of the Internet completely oblivious.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols*

General Terms

Algorithms, Design, Performance, Reliability

Keywords

Routing Protocols, Reliability, Internet

1. INTRODUCTION

In recent years, a growing chorus of researchers have advocated the multipath routing paradigm, in which the routing infrastructure makes multiple paths available, allowing senders to select among them. This approach gives users access to the paths that best suit their needs (low latency, high bandwidth, low loss, low jitter), thereby improving reliability and increasing competition among ISPs ([15], [3]). It is hard enough to design multipath routing algorithms for the intradomain case, but the interdomain case is even more challenging because of policy constraints and scaling requirements. There have been several proposals for interdomain multipath routing (see, for example, [14, 12]), and they have made admirable progress in grappling with these two issues; to wit, they have demonstrated that it is possible to provide a set of alternate interdomain paths in a scalable and policy-compliant manner.

*Author was supported by Major State Basic Research Development Programs of China: No.2009CB320503

The only disquieting aspect of these approaches (and many other multipath proposals in the intradomain case) is that the set of alternate paths is somewhat *ad hoc*; they cannot systematically compute a set of alternate paths that have a high degree of path diversity.[1] That is, while they provide a tunable number of alternate paths, these paths may have significant overlap, thereby leaving the possibility that a single failure could take out the entire set.

In this paper we present the Yet Another Multipath Routing (YAMR) protocol that systematically provides high path diversity. There are two components to YAMR.

(1) An efficient BGP-like mechanism for computing a diverse family of policy-compliant paths: This component of YAMR (which we call YAMR Path Construction, or YPC) computes a set of alternate paths that are deviations from BGP's default path.[2] Each alternate path is computed assuming that a link in the default path is down. Considered as a static set of paths, there is no single failure that can break all the paths simultaneously, unless that failure disrupts *all* policy-compliant paths between the source and receiver. When protocol dynamics are taken into account, the story is more complicated (because when BGP recovers from a link failure, it can break paths that did not contain the failed link). We present simulation results on the actual resilience achieved under full dynamics, which show that YAMR improves the reliability of BGP in single link failures by almost three orders of magnitude.

However, computing this family of paths involves higher control plane messaging overhead than BGP. We therefore added another component to YAMR.

(2) A technique for reducing churn by localizing routing updates: Much of the churn created by BGP is due to the fact that every change in a path must be disseminated to all nodes that use that path. YAMR hides some of these updates, and it turns out that this "update hiding" technique not only reduces YAMR's churn, it also increases (by an order of magnitude) YAMR's resilience, by largely avoiding BGP's problem of recovery causing functioning paths to break.

The rest of the paper is organized as follows. In Section 2 we present the YAMR Path Construction algorithm and,

[1]The theory literature has many such algorithms, but they do not lend themselves to scalable, policy-compliant implementation.

[2]This idea is borrowed from [5], which computes the cost of the cheapest path that avoids each link on shortest path.

in Section 3, we present the hiding technique. We describe simulation results in Section 4 and conclude in section 5.

2. YAMR PATH CONSTRUCTION

The core of YAMR is a policy-based multipath construction mechanism that is very similar to BGP. Paths are described by a series of ASes, such as $[A, B, C, D]$; this also defines a series of interdomain links, such as (A, B), (B, C), and (C, D) in the preceding example. For convenience, we use a failure model where these links are the unit of failure. We can easily generalize the algorithm to cover both link failures and domain failures (where all links $(*, A)$ and $(A, *)$ fail for some A), but our notation is cumbersome enough as is, so we opt for clarity over generality in this short paper.

The goal of YPC is to compute a *default* path p_d (that is identical to what BGP would compute) and for each link L in p_d also compute (if one exists) a policy-compliant *alternate* path p_L that does *not* contain L. It turns out that we can only guarantee this under a set of restrictive conditions.

We say that the network is in a *canonical condition* when the network has converged, all ASes follow *next-hop* [6] and *widest-advertisement* [10] policies (these policies include the customer-peer-provider policy [8]), and there are no *dispute wheels* [9]. Under these conditions, we can show (see [7] for proofs of all theorems in this paper):

THEOREM 1. *Assuming the canonical condition, for any destination D, AS A, and interdomain link e, if there is a policy-compliant path from A to D that avoids e, then YPC computes a policy-compliant e-avoiding path to that destination.*

We need the aforementioned assumptions for the following reasons. Without the "no dispute wheels" assumption, BGP and YPC can oscillate forever and we cannot talk about the converged state of the network. Without the widest-advertisement assumption, one cannot make guarantees about path availability. In fact, without this assumption, even when there is a policy-compliant path from some AS to some destination, BGP can leave this AS disconnected (see [7]). Thus, we need these two assumptions because of BGP properties that our protocols inherit, not because our protocols introduce new restrictions. The last assumption of next-hop policies is needed mainly to talk about policy-compliance in the context of our hiding technique.

We now describe YPC in more detail, starting with the control plane and then moving to the data plane.

2.1 Control Plane

Similar to BGP, ASes in YAMR construct their default and alternate paths from the paths advertised by their neighbors, applying local policies, import and export filters and actions. These multiple paths to the same destination are differentiated using *labels*. Default paths have a special label which we denote by d, while alternate paths are labeled by the link they avoid.

Within this framework, YAMR achieves Theorem 1 by selecting paths as described in Procedure 1, where p_L denotes an L-labeled path, $best_A(U)$ denotes the A's most preferred path from the set U, U_L denotes the set of L-labeled paths A knows from its neighbors, and U_d^L denotes the set of default paths A knows that avoid link L. AS A first selects its default path from the default paths it knows from its neighbors. Because default paths are selected only from default paths, the default paths in YAMR are exactly the same as in BGP. Then, for each interdomain link L on the default path, A selects an L-labeled alternate path from the set of default paths avoiding L and alternate L-labeled paths. To clarify terms, we use *RIB_IN* to describe the set of routes learned from neighbors and *RIB_LOCAL* to describe the set of routes used for forwarding.

Procedure 1: YPC path selection run by AS A.

```
/* select the default path */
p_d := best_A(U_d)
foreach link L in p_d do
    /* select the L-labeled alternate path */
    p_L := best_A(U_d^L ∪ U_L)
end
```

We now walk through a complete run of YPC shown in Figure 1. First, C announces its default path $[C]$ to its neighbors, which then construct their default paths. None of the neighbors is able to construct an alternate path yet. Next, B and D send their default paths to each other. Upon processing these messages each is able to construct the alternate path it needs. Next, B and E send to A the updates to their RIB_LOCALs. A can construct its default paths either from $[B, C]$ or $[E, C]$. A prefers to have $[A, B, C]$ as its default path and now needs to construct alternate paths avoiding links (A, B) and (B, C). For the (A, B)-avoiding path A has the path $[A, E, C]$ as the only choice because the path $[A, B, C]$ goes through (A, B) and the path $[A, B, D, C]$ cannot be considered because of its label (and would be unsuitable anyway, since it does not avoid (A, B)). Finally, A sends updates to its RIB_LOCAL to E, which is now able to pick its alternate path.

Using techniques borrowed from the classic proof for BGP ([9]), we can show:

THEOREM 2. *If there are no dispute wheels, YPC always converges to a unique final configuration that has no loops.*

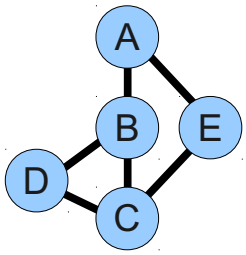
2.2 Data plane

YAMR requires a single addition to the IP header: a 32-bit field for the path label. A packet arriving at AS A destined to D with label L is forwarded along the L-labeled path towards D if A has such a path in its RIB_LOCAL. Otherwise, the packet is forwarded along the default path towards D (without overwriting the label). If A does not have a default path towards D, the packet is dropped. Once the control plane has converged, this algorithm is guaranteed to produce no loops.

For each destination, a YAMR router needs to have a forwarding entry for the default path and for each alternate path whose next-hop is different from the next-hop of the default path. Thus, the state requirements of YAMR are roughly $1 + k$ times that of BGP, where k is the average interdomain path length. Recent measurements suggest that this is around 3.6 [2].

2.3 Discussion

As Theorem 1 shows, YPC is guaranteed to give each AS a policy-compliant path that avoids any given interdomain link (if such a path exists), thus greatly improving reliability. Moreover, users can use all of the paths simply by inserting



This figure presents a simple run of YPC on the topology shown on the left. AS C announces a single prefix and other ASes build their paths to this prefix. In the table below, the first column shows the messages sent by the protocol. The other five columns show the state of the routing tables after all the messages in the first column have been processed. Messages are denoted by "src->dst : msg", where msg contains a number of paths. Each path is denoted by "label:AS-path". In this figure, we denote the default path label by (0,0). Messages that don't result in changes to the routing tables are omitted. Also, note that we picked a particular order of the messages. If another order were picked, intermediate routing tables would have been different.

Messages	A	B	C	D	E
C->D: (0,0):[C] C->B: (0,0):[C] C->E: (0,0):[C]	none	(0,0):[B,C]	local path	(0,0):[D,C]	(0,0):[E,C]
D->B: (0,0):[D,C] B->D: (0,0):[B,C]	none	(0,0):[B,C] (B,C):[B,D,C]	local path	(0,0):[D,C] (C,D):[D,B,C]	(0,0):[E,C]
B->A: (0,0):[B,C], (B,C):[B,D,C] E->A: (0,0):[E,C]	(0,0):[A,B,C] (A,B):[A,E,C] (B,C):[A,B,D,C]	(0,0):[B,C] (B,C):[B,D,C]	local path	(0,0):[D,C] (C,D):[D,B,C]	(0,0):[E,C]
A->E: (0,0):[A,B,C], (A,B):[A,E,C], (B,C):[A,B,D,C]	(0,0):[A,B,C] (A,B):[A,E,C] (B,C):[A,B,D,C]	(0,0):[B,C] (B,C):[B,D,C]	local path	(0,0):[D,C] (C,D):[D,B,C]	(0,0):[E,C] (C,E):[E,A,B,C]

Figure 1: A complete run of YPC on a simple topology.

the appropriate path label into their packets. (The default path lists all the AS links, and so the edge will know which labels will produce different paths; YAMR does not include mechanisms to tell the edge which of these AS links might be providing subpar service.) The paths are constructed and made available to users with moderate increases in the control messaging (or churn) as we will see in Section 4, and in RIB and FIB sizes.

An alternate approach would be to construct a single "optimally disjoint" path. However, we were not able to prove that such a path is always policy compliant (or conversely, that any policy-compliant alternate path is sufficiently disjoint so that Theorem 1 would hold).

BGP scalability is considered a critical challenge [11] and YPC makes it worse. Among the many dimensions of scalability, churn appears to be the most intractable. Indeed, the comparison of technology trends and projected growth of RIB and FIB sizes in [1] suggests that technology advances are expected to satisfy RIB and FIB memory requirements at a constant cost. We now present a method that reduces YAMR's churn below that of BGP. This churn reduction method involves hiding path withdrawals, leaving most of the Internet completely oblivious to the failure.

3. HIDING ROUTE UPDATES

YAMR's hiding technique is a set of distributed mechanisms that can be applied to either YPC or BGP to confine the effects of a link failure to a small neighborhood around the link. Hiding ASes do not propagate information about the failure to their neighbors if they can safely reroute around it. For example, in Figure 1, if link (B, C) fails, B can reroute around this failure by deflecting all traffic onto $[B, D, C]$ without telling A that path $[B, C]$ has failed. We call B a *hiding* AS, path $[B, D, C]$ a *deflection* path, and path $[B, C]$ (the failed path being hidden) a *lame* path.

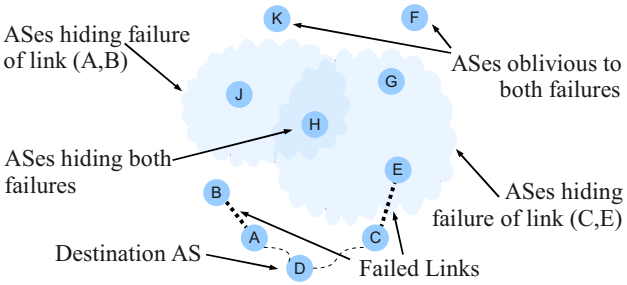


Figure 2: An illustration of hiding bubbles.

In the example above, B is able to completely hide the failure so that all other ASes remain oblivious to it. However, in general topologies and policies, B might be able to hide the failure only from a subset of its neighbors, but can't hide it from others because it doesn't have a suitable path it can export to them. In such a case, B withdraws the failed path from the neighbors for whom it can't hide the failure. These neighbors then try to hide the failure from their neighbors, recursively. This process continues until the failure is completely hidden. In other words, a single failure is hidden by a dynamically determined bubble of hiding ASes (see Figure 2).

Hiding is easy: just pretend a withdrawn path is available. Hiding without creating loops and without damaging connectivity is hard. We don't give the full details of the hiding mechanism here as that would distract the focus of the paper (see [7] for details). Instead, we first state the guarantees our mechanism provides (see [7] for proofs), and then give a high level overview of the design. We can show that when hiding is combined with YPC to produce the full YAMR protocol, the following results hold:

Procedure 2: YAMR default path selection.

```
while U_d is non-empty do
    p_d := best_A(U_d)
    if p_d is lame then
        p_d.defl := best_non_lame_A()
        if p_d.defl is null then
            delete p_d from RIB_IN
            continue
        end
    end
    break
end
```

THEOREM 3. *If there are no dispute wheels, YAMR always converges.*

THEOREM 4. *In the converged state, YAMR has no forwarding loops or dead ends. Moreover, if ASes follow next-hop policies, all forwarding paths are policy-compliant.*

THEOREM 5. *Assuming canonical conditions, for each AS A, if there is a policy-compliant path from A to the destination, A has a policy-compliant path to the destination in YAMR.*

THEOREM 6. *When a failed link recovers, all hiding caused by it stops and routing returns to normal.*

We now present three hiding mechanisms, each of which was introduced to solve a particular problem.

Hiding Path Selection The fundamental mechanism of hiding is to pretend that a withdrawn path is available. When a path currently in the AS A's RIB_IN is withdrawn, A does not delete it from the RIB_IN as it would in BGP. Instead, A marks the path as *lame* and calls path selection. In path selection, if A selects a lame path, it tries to choose a deflection path (from among the set of other default and alternate paths) for it. If there is no suitable deflection path, the lame path is deleted from the RIB_IN and no hiding occurs. YAMR's selection of the default path is presented in Procedure 2. Alternate paths are selected analogously.

After path selection, each lame path in the RIB_LOCAL has a deflection path associated with it. As in BGP, after the RIB_LOCAL has been updated, YAMR announces the changes to its neighbors. Export filters and actions are applied to non-lame paths in the same way as in BGP. However, for lame paths, export filters are applied to the corresponding deflection paths and export actions are applied to the lame paths. If export filters allow the deflection path to be advertised to a neighbor, the lame path is actually advertised. Otherwise, a withdrawal is sent to the neighbor.

In YAMR, withdrawal messages also include the failed link(s), if any, that caused the withdrawal. This failed link information gives the receiving AS "permission" to hide the withdrawal. Withdrawals caused by permanent changes (e.g., a policy change or a prefix withdrawal) don't include the failure information, and are therefore not hidden. When the failure recovers, all failure information is revoked causing all ASes to stop hiding and to return to regular routing.

Hiding Forwarding Forwarding in YAMR is the same as in YPC except that the forwarding entries for lame paths are built based on the corresponding deflection paths. If the lame path's label is different from the deflection path's label, the labels of packets forwarded along the deflection path are replaced by the deflection path's label.

Tokens In the two previous sections, we described that YAMR advertises lame paths, but forwards on deflection paths. When multiple ASes are hiding multiple failures, the "lies" being told by the ASes might result in forwarding loops or leave an AS unnecessarily disconnected. These two problems are solved by introducing a new control message we call a *token*.

The goal of tokens is to determine if there is a problem (either loop or disconnection) and, if so, signal to one of the hiding ASes to stop hiding. A "loop" token is sent whenever a deflection path is installed into RIB_LOCAL. The loop token is sent along this deflection path and, as forwarded, it records all the ASes along the path. If it ever arrives at a hiding AS a second time, that AS stops hiding. A "disconnection" token is sent when an AS finds out that all available routes appear loopy (include itself in the path). The token is sent along one of these paths and causes the first encountered hiding AS to stop hiding.

The token mechanism is designed to be safe, but not necessarily optimal. In other words, tokens will always find and fix a problem if there is one, but they can be overly-cautious, causing more ASes to stop hiding than needed. We believe (but cannot prove) there is a fundamental trade-off between the optimality of the hiding bubble and the overhead of achieving it. Tokens have low overhead, but can produce sub-optimal bubbles.

3.1 Discussion

Recalling the high level picture, YPC is able to effeciently construct a set of paths with provable static diversity, but incurs higher messaging overhead than BGP. To decrease the overhead, we developed a hiding technique that, as we will see in the next section, brings the churn level of YAMR below that of BGP. The surprising result, again to be discussed in the next section, is that hiding also substantially improves *resilience*. Hiding localizes the impact of any routing update, and decreases the chance that the convergence process will interfere with any functioning paths.

However, hiding deprives YAMR of some of YPC's valuable properties. First, the set of YAMR paths might not be one-failure resistant because the set of paths might already be hiding failures. Second, YAMR's advertised paths can be different from the forwarding paths. Because ASes cannot be sure about the paths beyond the first hop, they cannot implement policies beyond next-hop policies with 100% confidence. Furthermore, this inconsistency between the control and data planes can adversely affect routing-dependent systems such as distributed spam detection. We did not evaluate this potential negative impact here; we don't know how often there is a mismatch between the control and data planes, nor how seriously such a mismatch affects various routing-dependent systems. The question is whether the benefits (described in the next section) of YAMR's increased resilience and substantially lower churn compared with YPC are worth these two disadvantages.

4. EVALUATION

Recall that YAMR is composed of the path construction algorithm YPC and a hiding technique. This hiding technique can be applied to BGP, forming what we call HBGP. To understand the contributions of these components to various metrics, we run each experiment for all four protocols: BGP, HBGP, YPC, and YAMR.

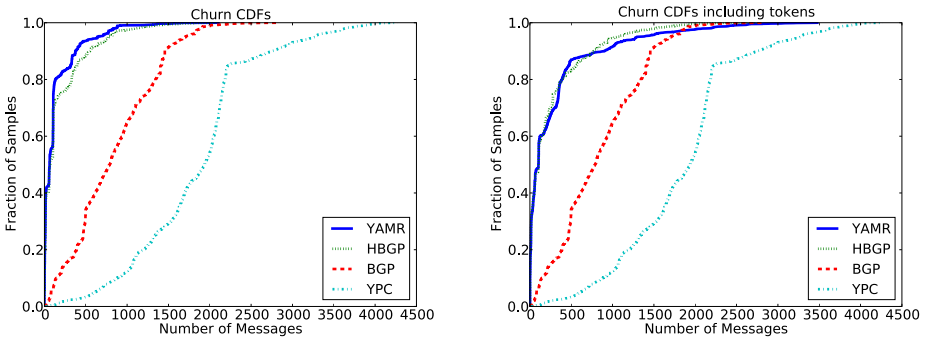


Figure 3: CDFs of number of messages following a link event (either failure or recovery). On the left side, only update messages are included. On the right side, all messages are included. The averages are BGP: 829, YPC: 1828, HBGP: 178 and 249, YAMR: 134 and 286.

	BGP	HBGP	YPC	YAMR
Disconnectivity	9.05	8.43	0.12	0.01
Convergence Time	23.8	16.7	44.9	1.16

Table 1: Average percentage of ASes experiencing transient disconnectivity (top row) and average convergence time in seconds (bottom row) following a single link failure in a 1000 node topology.

4.1 Methodology

We implemented a message-level event driven simulator that includes important features like MRAI timers (with average value of 30 seconds), router processing delay, and message propagation delay. For simplicity, we represent each AS as a single router. We used annotated Internet-like topologies generated using [4].

Our basic experiment is the following. Given a topology and a multihomed stub AS, we make the AS announce a prefix, let the network converge, fail one of the provider links from this AS, and let the network reconverge. This basic experiment is repeated for all multihomed stub ASes and all of their provider links. We use a 1000 node topology for most metrics. To study scalability, we use topologies of sizes from 500 to 5000 in increments of 500.

We selected this initial experiment, which focuses on failures close to the edge, because internal failures are substantially less common and more amenable to recovery, even in BGP. Thus, these edge failures are the most interesting case, and are the dominant case in reality. We also note that this simulation is similar to the live deployment experiment of [13]. Our future work will study a broader class of failure models.

4.2 Results

We present and discuss our simulation results for reliability, churn, path stretch, and forwarding table size.

Reliability Table 1 shows the average number of ASes that experience any disconnectivity during the convergence process. We consider an AS to experience disconnectivity if there is ever a moment when none of the paths in its forwarding table are working. The table shows YAMR is almost 1000 times more resilient than BGP. The table also includes the average convergence time. Note that YAMR converges more than 20 times faster than BGP, because of failure localization. In both cases, the hiding aspect helps YPC far more than BGP. Presumably this is because YPC provides many more potential deflection paths than BGP.

Because the simulation evaluations of other interdomain multipath routing proposals [16, 14, 12] were done with different methodologies, we have not yet been able to accurately compare YAMR's reliability with them; however, we can give a very rough comparison of YAMR to path splicing [12] [3]. Recall that static reliability means that a routing protocol, at the time of the failure, has an alternate path that avoids the failure. This is an easier quantity to measure than what we studied in our dynamic simulations, but it ignores the fact that when a routing protocol is converging to route around the failure, it can disrupt this functioning alternate path. Nonetheless, it does provide some measure of reliability. Eyeballing Figure 7 in [12], we see that path splicing is able to improve static BGP reliability by about a factor of 15 with 5 forwarding entries per router per destination (that is, there are 15 times more unnecessary disconnections in BGP than there are in path splicing). YAMR, on the other hand, has no unnecessary disconnections when a single link fails, and even in our dynamic simulations which allow routing recovery to disrupt these alternate paths, it has almost 1000 times fewer unnecessary disconnections than BGP.

R-BGP [10] is another promising approach, achieving both perfect static and dynamic reliability when a single link fails. However, it is not a canonical multipath algorithm because it does not make multiple paths available to the users; it only invokes them upon network-detected failure. It also does not have perfect policy compliance.

Churn Figure 3 shows the CDFs of the number of messages following a link event. We present two graphs with and without tokens because token processing is a much lighter operation than update message processing and because separating them shows how many updates hiding saved and how much extra communication it introduced.

In both graphs, YAMR and HBGP significantly outperform YPC and BGP, reinforcing the conclusion that hiding is effective in reducing the messaging overhead. If tokens are ignored, YAMR reduces the message overhead by a factor of 6.2 compared to BGP, and by a factor of 2.9 if tokens are counted despite the fact that YAMR constructs more paths.

Note that compared to the protocols without hiding (BGP and YPC), the protocols with hiding (HBGP and YAMR) perform relatively better in the lower percentiles than in the higher percentiles. For example, at 20^{th} percentile, YAMR has only 5 messages while BGP has 388 messages. For every failure, BGP requires many messages to converge. YAMR,

[3]Path splicing constructs multiple paths by running the original routing protocol on several different instantiations (slices) of the underlying topology. For instance, these slices could have modified link weights, so each slice gave a different set of paths.

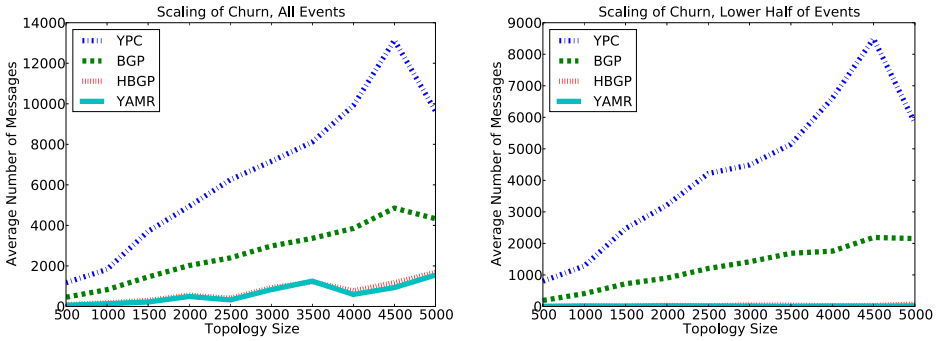


Figure 4: Average number of messages following a link event (either failure or recovery) versus topology size. On the left side, all link events are included. On the right side, only half of the events with lowest number of messages are included, separately for each protocol.

in contrast, is more bimodal: when the failure occurs in a richly connected portion of the network, it recovers with very few messages, but if connectivity is sparse then recovery is expensive (sometimes more so than BGP). Exploring this bimodal distribution, Figure 4 demonstrates that the size of the smaller convergence events are independent of network size, but the average size of all convergence events grows linearly with network size.

Path Stretch After each of our basic experiments with a single provider link failure, we measured the average path stretch for all 3 protocols and found that the average path stretch was negligible. For example, the average path stretch of YAMR was only 1.02. The reason is that in the Internet-like topologies, it is almost always possible to find an alternate path that has the same length as the shortest path.

Forwarding Table Let F be the average number of forwarding entries per router per destination. As noted in Section 2.2, the pessimal F is $1+k$, where k is the average path length, and this occurs when every alternate path is different. In our 1000 node topology the average path length is 2.86, so the pessimal F is 3.86, but in our simulations $F = 2.21$, 43% less than the pessimal value. If this holds for the Internet, F for the Internet would be roughly 2.62.

4.3 Incremental Deployability

The only portion of YAMR that is not easy to deploy incrementally is the token mechanism. Beside the tokens, YAMR is incrementally deployable using an approach very similar to [12]. In the data plane, YAMR's label can be placed in a shim header between the IP and the transport headers so that routers unaware of YAMR can forward packets using just the IP address. In the control plane, YAMR can be implemented as a different version of BGP. A YAMR-aware AS A can ask the neighboring AS what version of BGP it speaks. If the version is non-YAMR, A sends over only the default paths and interprets the incoming ones as such.

The hiding mechanism does not need any modifications to accommodate non-participating ASes because, by design, any AS is free to stop hiding at any time. From the hiding perspective, a non-participating AS is indistinguishable from an AS that knows about hiding, but decides not to hide.

The problem with incrementally deploying tokens is that a non-YAMR AS will not forward them along the path. A possible solution for a YAMR-aware AS A with a non-YAMR neighbor B is to ask the AS after B if it is YAMR-aware. If so, A can forward tokens directly to the latter AS. Otherwise, A can continue asking until it finds the first YAMR-aware AS along the path or the path ends. A detailed design and evaluation of such a mechanism is left for future work.

5. CONCLUSION

In this paper, we presented an efficient mechanism, YPC, to systematically construct a set of paths that is resilient to any one link failure. Because YPC manages more paths than BGP, it has a higher churn and a longer convergence time. However, when YPC is combined with the hiding technique, churn and convergence time fall well below the BGP levels. In our trials, YAMR increased the reliability by almost three orders of magnitude. This improvement is due to the carefully selected alternative paths as well as to the faster convergence realized by the hiding technique. We applied the hiding technique in the context of BGP, but we hope that it will be of independent interest because it is applicable to path-vector routing in general.

6. REFERENCES

[1] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable internet protocol (AIP). In *SIGCOMM*, pages 339–350, 2008.

[2] Routing table report. http://thyme.apnic.net/ap-data/2009/07/23/0400/mail-global.

[3] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in cyberspace: defining tomorrow's internet. *IEEE/ACM Trans. Netw.*, 13(3):462–475, 2005.

[4] X. Dimitropoulos, D. Krioukov, A. Vahat, and G. Riley. Graph annotations in modeling complex network topologies. In *NSDI*, August 2007.

[5] J. Feigenbaum, C. H. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18(1):61–72, 2005.

[6] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. *Distributed Computing*, 18(4):293–305, 2006.

[7] I. Ganichev, B. Dai, P. B. Godfrey, and S. Shenker. Yamr: Yet another multipath routing protocol. Technical Report UCB/EECS-2009-150, EECS Dept, UC, Berkeley, Oct 2009.

[8] L. Gao and J. Rexford. Stable internet routing without global coordination. In *SIGMETRICS*, pages 307–317, 2000.

[9] T. Griffin, F. B. Shepherd, and G. T. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Netw.*, 10(2):232–243, 2002.

[10] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs. R-BGP: Staying connected in a connected world. In *NSDI*, 2007.

[11] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984.

[12] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala. Path splicing. In *SIGCOMM*, pages 27–38, 2008.

[13] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush. A measurement study on the impact of routing events on end-to-end Internet path performance. In *SIGCOMM*, 2006.

[14] W. Xu and J. Rexford. Miro: multi-path interdomain routing. In *SIGCOMM*, pages 171–182, 2006.

[15] X. Yang, D. Clark, and A. W. Berger. Nira: a new inter-domain routing architecture. *Trans. Netw.*, 15(4):775–788, 2007.

[16] X. Yang and D. Wetherall. Source selectable path diversity via routing deflections. In *SIGCOMM*, pages 159–170, 2006.