LOUP: Who's Afraid of the Big Bad Loop?

Nikola Gvozdiev, Brad Karp*, Mark Handley
University College London

ABSTRACT

We consider the intra-AS route dissemination problem from first principles, and illustrate that when known route dissemination techniques propagate even a single external routing change, they can cause transient anomalies. These anomalies are not fundamental; they are artifacts of the *order* in which existing proposals disseminate routes. We show that carefully ordering route updates avoids transient looping and black holes. Perhaps surprisingly, this ordering may be enforced in a completely distributed fashion, while retaining familiar correctness, scalability, and convergence properties.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.1 [**Computer-Communictaion Networks**]: Network Architecture and Design; C.2.2 [**Computer-Communictaion Networks**]: Network Protocols—*Routing protocols*

GENERAL TERMS

Algorithms, Design

1. INTRODUCTION

Transient loops and black holes in a routing system cause grief for users of real-time applications. Kushman *et al.* [9] measure the quality of VoIP calls across the wide-area Internet, and observe that drops in call quality correlate closely with BGP routing changes. They conclude, perhaps surprisingly, that call quality suffers much more from BGP routing changes than congestion. Moreover, Shand *et al.* [12] note that even *intra-domain* routing introduces transient unreachability, as when routers' forwarding tables become inconsistent while a flooded link-state update propagates.

Maintaining end-to-end reachability for real-time traffic when routes change is challenging, as it requires robust behavior from each of the multiple components of the Internet's routing system. The community has devoted attention to unreachability due to delayed convergence of inter-domain routing [4], to fast failover in inter-domain routing [10], and to avoiding transient loops in intra-domain routing [5, 12]. Yet the transient behavior of *intra-domain dissemination of external routes* during routing changes has been, to our knowledge, unexamined.[1]

In this paper, we consider the problem of transient routing loops and black holes introduced by intra-domain route dissemination. Our ultimate goal is to design a new intra-domain route dissemination protocol that does not exhibit these pathologies, and thus moves a step closer toward a wide-area Internet better capable of carrying real-time traffic. To that end, we contribute:

- analysis of how known intra-AS route dissemination mechanisms for BGP give rise to transient black holes and loops;
- novel ordering constraints on the application of route updates from the *same* border router and route updates from *different* border routers that prevent transient black holes and loops;
- the Link-Ordered Update Protocol (LOUP), a design for an intra-domain route dissemination protocol that enforces these ordering constraints;
- evidence that LOUP prevents transient black holes and loops, by first-principles enumeration of possible routing events, and through simulation.

2. INTRA-AS ROUTE DISSEMINATION

Internet routing is composed of three components: *External BGP* (eBGP) distributes routes between routing domains and is the instrument for policy routing. An *Interior Gateway Protocol* (IGP) such as OSPF or IS-IS keeps track of reachability within a routing domain. Finally, *Internal BGP* (iBGP) distributes external routes received by border routers (BRs) both to all other BRs, so they can be redistributed to other domains, and also to all participating internal routers. In this paper we are primarily concerned with iBGP: when a route changes somewhere out there in the Internet, how does this change propagate across a routing domain?

When a BR receives a route change from a neighboring routing domain or *Autonomous System* (AS), it sanity checks it and applies a policy filter. This policy filter can modify or drop the route. Common modifications include setting the Local Preference attribute (akin to a priority field) and setting Community attributes which can be used to determine how the route is redistributed to other ASes by other BRs.

After policy filtering, the BR runs its decision process, determining whether it prefers this route to other routes it may hold for the same IP address prefix. The decision process is specified in the BGP standard, and consists of successive rounds of eliminating candidate routes based on different cri-

*Supported by EU FP7 grant 287581 (OpenLab), FP7 grant 257422 (Change) and by the Cisco University Research Program

Hotnets '12, October 29–30, 2012, Seattle, WA, USA.

[1] The instability of iBGP's route reflector mechanism has been studied [6, 11], but not the *fundamental* behavior of dissemination.

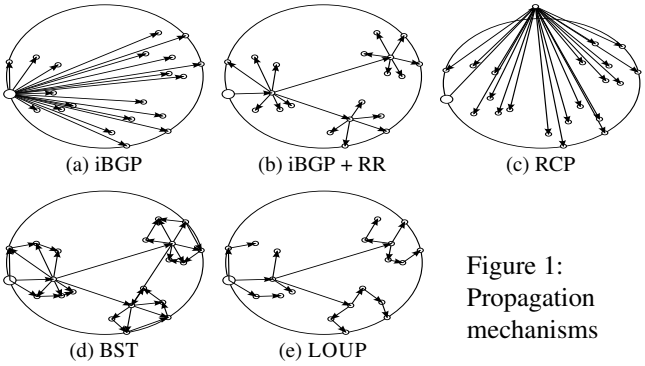

Figure 1: Propagation mechanisms

teria until only one remains. First in the decision process is *Local Preference*, so configured policy trumps all else. Lower down the list come AS Path length, and below that IGP distance (the distance to the BGP Nexthop—usually either the announcing BR itself, or its immediate neighbor in the neighboring AS).

When a BR receives a route announcement for a new prefix, if it is not dropped by policy, the BR distributes it to all the other routers in the domain, so they can reach this destination. If a BR already has a route to that prefix, the new route is only sent to the other routers if the BR prefers the new route. Similarly if a BR hears a preferred route from another BR to one it previously announced, it will withdraw the previously announced route.

Having decided to announce or withdraw a route, the change must be communicated reliably and quickly through the AS. BGP routing tables are large—currently over 400,000 prefixes—and multiple BRs can receive different versions of each route. Periodic announcement of routes doesn't scale well, so dissemination needs to be reliable—once a router has been told a route, it will hold it until it is withdrawn or superseded. Route dissemination also needs to be fast—otherwise inter-AS routing can take a long time to converge.

3. DISTRIBUTION MECHANISMS

Fig. 1 illustrates different styles of propagation used by route dissemination protocols. The simplest solution depicted in Fig. 1a is to open connections from every BR direct to everyone else. This is the approach adopted by full-mesh iBGP. When an update needs to be distributed, a BR just sends it down all its connections. Using TCP ensures reliability and in-order delivery *within* each session. However as each recipient applies the update as soon as it can, this results in an arbitrary ordering *between* recipients. Such arbitrary ordering can cause transient loops and black holes until all the routers have received and processed the update.

Route Reflectors (Fig. 1b): Full-mesh iBGP scales poorly ($O(n^2)$ connections; too much fanout) so Route Reflectors (RRs) were introduced. RRs introduce hierarchy; they force propagation to happen over a tree.[2] Updates are sent by a BR to its reflector, which forwards them to its other clients and to other reflectors. Each other reflector forwards on to its own clients. Reflectors are manually placed according to guidelines that say "follow the physical topology"; not doing so can cause suboptimal routing[11]. This imposes a limited ordering constraint (RR clients receive a route after their RR processes it), but except in trivial non-redundant topologies this is insufficient to prevent loops and black holes.

[2]Actually, often two overlaid trees for redundancy.

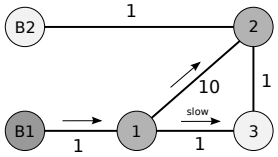

Figure 2: A simple topology exhibits looping

Centralized Processing (Fig. 1c): With this approach, when an external update arrives the BR first sends it to the routing control platform (RCP[2]), which is in charge of running the decision process for the whole domain and distributing the results to all routers. By centralizing the decision process we gain improved scalability so long as the RCP router is provisioned appropriately, and the ability to do fine-grained policy routing. In principle it is also possible to apply updates in any desired order across a domain, but to do so requires a synchronous update approach which, given the RTTs to each router, will be slower than distributed approaches.

Hop-by-Hop Flooding: (Fig. 1d) An alternative is to flood: all routers send the messages they receive to all neighbors. Flooding needs to be done over one-hop reliable sessions to ensure messages are not lost. This is the approach taken by BST[8]. Flooding imposes a topological ordering constraint, guaranteeing that at all times, a contiguous region of routers have processed an update. Essentially an update propagates out across the domain as a *wave-front*; this is a necessary (though not sufficient) condition to avoid transient loops. None of the other mechanisms above have this property.

To see why this condition is not sufficient, consider Fig. 3. BR B had previously received a route to prefix P, and distributed it to all the routers in the domain. BR A then receives a better route to P, and this is in the process of flooding across the domain, forming a wave-front ① which is flowing outwards from A. All the routes in the light-gray region now forward via A; the remainder still forward via B. Unfortunately flooding does not ensure that the wave-front remains convex—that a forwarding path only crosses the wave-front once. As a result transient loops ③ can occur.

Fig. 2 shows one way that such non-convexity can occur. Initially all routers forward to some prefix via B_2, but then B_1 receives a better route. Link 1-2 would not be normally used because of its high metric. If, however, router 1 floods the update from B_1 over this link, then receiving router 2 may direct traffic towards towards B_1 via router 3. As router 3 has not yet heard the update, it directs traffic towards B_2 via router 2, forming a loop. The loop will clear when router 3 eventually hears the update, though network conditions or variable CPU processing delays may delay this.

Reverse Forwarding Tree Dissemination: BST loops when a better route is propagating, replacing an older route. A sufficient condition for avoiding such loops is that *a router does*

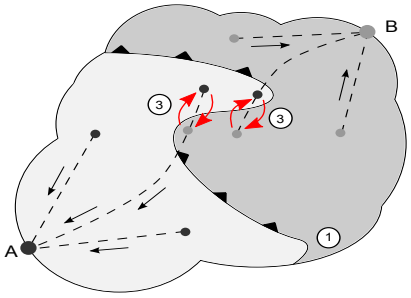

Figure 3: Transient loop when flooding an update.

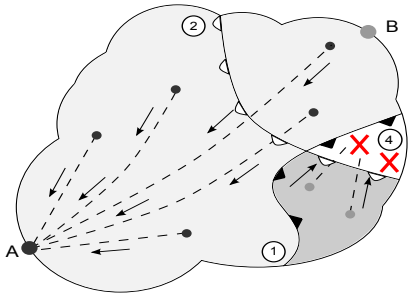

Figure 4: Racing update and withdrawal cause black hole

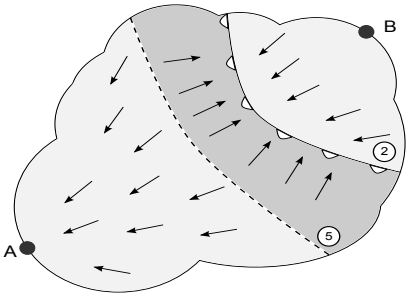

Figure 5: Withdrawal causes loop when alternate exists

not adopt the new route until the next hop for that route has also adopted the route. This transitively ensures a packet forwarded using the new state will not meet a router still using the old state. One way to meet this condition is for a router to only forward a route to routers that will forward via itself. Thus routes flow from a BR along the reverse of the forwarding tree that packets take to get to that BR.

In this paper we discuss a new routing mechanism we call Link-Ordered Update Protocol (LOUP) that is built around this principle . It works by propagating messages over a hop-by-hop tree (Fig. 1e). Unlike the Route Reflector tree, LOUP uses one tree per BR, rooted at that BR. This tree is dynamically built, hop-by-hop, and follows the underlying IGP routes to get to that BR from everywhere in the domain. The hop-by-hop nature preserves the wave-front property, and by distributing down the reverse of the forwarding tree (RFT), it adds additional desirable ordering constraints that eliminate transient loops of the form detailed above.

4. INTER-ORIGIN ORDERING

To avoid loops, routers must apply route updates in the correct order. Hop-by-hop propagation of a route along the reverse of the forwarding tree to the exit router specified in the route is one such an ordering constraint. It imposes an *intra-origin ordering*: each router receives (and hence applies) its own update only after all the routers between it and the exit router have also done so. This is similar to what oFIB does on the IGP level [12].

Unfortunately, to avoid loops and unwanted black holes, intra-origin ordering is not the whole story. In the iBGP ecosystem, one route change can trigger another. There are two common cases. First, a route can be withdrawn, and another route must take over. Second, a new winning route can trigger the withdrawal of the previous winning route. In both cases the order in which the two route changes are applied can lead to transient loops or black holes. As two (or more) changes are involved, distribution down the RFT, which only ensures intra-origin ordering is not sufficient, and we need to also consider inter-origin ordering constraints.

An example of an inter-origin problem is shown in Fig. 4. As before, a new better update flows out from BR A, depicted by wave-front ①. However, this time it has already reached BR B. B now withdraws its own route because it has been obsoleted by the new route. The withdraw spreads out as wave-front ②, and the scene is set for a race. Some routers have not yet heard the new update. As there is no enforced inter-origin ordering, those routers are exposed. Routers ④ hear the withdrawal before the update, apply it and temporarily black-hole any traffic destined for that prefix.

For LOUP to be free from unwanted transient effects, it must enforce both inter and intra-origin ordering.

4.1 Ordering and Triggered Races

Table 1 shows all the possible external route changes that can occur, and their effect on a network containing either zero, one, or two existing routes for a prefix. $X > Y$ indicates route X beats route Y in the decision process at all routers; $X = Y$ implies the two routes tie-break on IGP distance—some routers use one exit and some use the other. We assume changes are propagated down the RFT tree ensuring intra-origin ordering is maintained. The entries highlighted in gray show all the cases where intra-origin ordering is insufficient, and races resulting in transient loops or unnecessary black holes can occur. Although there are multiple causes, only three distinct ordering problems emerge:

⋆ **UW-race**: an update/withdrawal race

† **W-order**: a withdrawal ordering problem

‡ **WA-race**: a withdrawal/announce race

The *UW-race* (⋆) is the case discussed above, and shown in Fig. 4: a transient black hole can occur due to a race between an *update* (or new announcement) of a route and the *withdrawal* of an alternative route that it triggers.

For a loop of type *W-order* (†) to happen, all routers must hold more than one route to the same destination prefix. Typically this is when more than one BR originates a route, but they are all equally preferred. Each router chooses the route to the closest exit, with some routers making one choice and some another. Such hot-potato routing is common when two ISPs peer in multiple places.

A *W-order* loop can then occur when one of those routes is withdrawn, as in Fig. 5. The routers behind withdrawal wave-front ② have already switched to an alternative route via A. Routers further away have not yet heard the withdrawal and still forward to B. Traffic loops at wave-front ②.

For a loop of type *WA-race* (‡) to happen, two or more BRs must hold different routes to the same prefix, with one

routes before event	event that occurs							
	withdraw X	withdraw Y	announce $Z = X$	announce $Z < X$	announce $Z > X$	X gets worse than Y	Y gets better than X	X gets better
no route	n/a	n/a	fwd Z on tree			n/a	n/a	n/a
X	any order	n/a	fwd Z on tree	no-op	fwd Z on tree, withdraw X $\star$	fwd X on tree	n/a	fwd X on tree
$X = Y$	only apply withdrawal when parent does †		fwd Z on tree	no-op	fwd Z on tree, withdraw X,Y $\star$	withdraw X †	fwd Y on tree, withdraw X $\star$	fwd X on tree, withdraw Y $\star$
$X > Y$	fwd Y on tree ‡	no-op	fwd Z on tree	no-op	fwd Z on tree, withdraw X $\star$	withdraw X fwd Y on tree ‡	fwd Y on tree, withdraw X $\star$	fwd X on tree

Table 1: Effect of route changes on network with zero, one or two existing routes X and Y.

route being preferred. In this case, the BRs holding the less preferred routes will have withdrawn them, and all the other routers will only hold the best route.

When the best route is withdrawn a *WA-race* loop can occur. The withdrawal reaches the BR holding the route that previously lost, and this is now re-announced. The withdrawal and announcement now race. This is almost the inverse of Fig. 4: some routers hear the withdrawal first and some hear the announcement first. Routers hearing the announcement first will not apply it, as they still prefer the better route, but they can forward the announcement to routers that have already heard the withdrawal. Packets can then loop between the two.

Interestingly, we observe that *UW-race* and *WA-race* cannot occur with BST. Its hop-by-hop reliable flooding ensures that no router ever receives the triggered effect of an event before it receives the triggering event itself. Succinctly, it ensures that *cause* always happens before *effect*. By forwarding updates along the RFT tree, we can ensure intra-origin ordering, but at the expense of inter-origin ordering. What we would really like is the best of both. There are two ways this might be achieved:

- Use flooding to distribute changes, but add ordering constraints on when an update or withdrawal can actually be applied to avoid the loops in Fig. 3 and 5.
- Use RFT tree-based distribution of changes, but add ordering constrains on the application of withdrawals to avoid *W-order*, *WA-race* and *UW-race* conditions.

The latter is less chatty, so LOUP adopts this strategy.

5. LOUP

With LOUP we demonstrate that it is possible to design a scalable and robust routing protocol that is free from the transients already described. LOUP propagates information over the RFT to maintain intra-origin ordering of updates. The construction of the tree does not incur much computational overhead because we leverage information readily available from the IGP.

As with BST, each router automatically establishes sessions to its immediate neighbors and changes propagate hop-by-hop. Using TCP ensures hop-by-hop reliability, but this only gives end-to-end reliability if the tree does not change. Unlike iBGP with RRs, LOUP does not maintain redundant trees for reliability. It actively maintains the RFT using messages sent from a child to its parent every time IGP routes change. An originating BR sequences updates before sending, and all routers store changes in log-like structures to deal with message loss which may occur in the presence of failures. We have tested this mechanism exhaustively [7].

Note that LOUP is only concerned with distribution of routing information within the AS. It is entirely compatible with eBGP, does not change the decision process and works with any attribute, including MED.

Some IGPs perform equal-cost multi-path (ECMP) routing, so there may be more than one equally preferred next hop to a BR. With ECMP, the RFT is in fact a DAG, and a LOUP router will receive an update from multiple parents. To avoid loops, LOUP must only forward via parents from which it has received the route update. Eventually it will hear from them all, but there may be a brief interval when only a subset of possible paths are used.

While using the tree ensures intra-origin ordering, we still need a way to enforce inter-origin ordering where needed. In essence we need mechanisms to solve the three problems previously outlined in Table 1.

5.1 Predicated Withdrawals (*UW-race*$\star$)

The triggered withdrawal in Fig. 4 is only there to clean up unneeded old state, but it can cause transient black holes due to the *UW-race*. These are best avoided.

In LOUP, the BR that originates such a withdrawal attaches a predicate to it to prevent anyone applying the withdrawal without having first seen the update that caused it. A predicated withdrawal is stored by routers as it travels down the tree, but is not applied until the predicate is met. In this case, the predicate specifies that the update that caused the withdrawal must have been heard. This ensures that no router can end up without a route. For example in Fig. 4 all the routers in ④ will wait until ① reaches them and only then withdraw the route.

5.2 "Tell me when..." (*W-order*†)

When updates flow down the tree, their propagation path follows the order in which they should be applied. Often this is the opposite of the desired application order of withdrawals, leading to the *W-order* loop in Fig. 5.

To avoid this, LOUP uses a second variant of delayed withdrawal. The withdrawal propagates along the tree as usual,

but routers do not use it immediately. Instead, each router runs the decision process to select the new exit router (*A* in Fig. 5), and determines the neighbor towards that exit. It then asks that neighbor to tell it when the neighbor has stopped using the route being withdrawn. Upon receiving a reply, a router knows that it is now safe to apply the withdrawal because all routers between it and the new exit point are already using the alternative route.

In Fig. 5, the withdrawal is not applied until it reaches ⑤. The first hop to the left of ⑤ triggers the reverse activation of the withdrawal by sending a reply. This causes the next router to apply the withdrawal, which causes it to reply, and so on back down the RFT tree from *A*.

5.3 Decision Modifiers (*WA-race‡)*

Delayed withdrawals using "tell-me-when" only work if the routers already have an alternative route. If one route is globally better, other alternatives will have been withdrawn from the domain, leading to the *WA-race* when the best route is withdrawn causing the next-best route to be re-announced.

In LOUP we ensure that the re-announcement will always win, irrespective of whether a recipient has heard the withdrawal of the previously-best route. When the new-best route is re-announced it is tagged with a decision modifier. This causes all recipients to exclude the withdrawn route from the set of routes they use when running the decision process. This causes them to behave as if they have already heard the withdrawal. Since everyone will reliably receive the withdrawal via its RFT at some point, this approach is safe.

5.4 Completeness

Table 1 can be extended to cover three pre-existing routes (and by induction, an arbitrary number) by adding four more rows to cover possible partial orderings of three route preferences. When we do this, we find that three of the four reduce to cases already covered in the table. The remaining row ($X < Y < Z$) results in a more complex chain when Z is withdrawn: both X and Y may be re-announced, and then X is withdrawn again. The two announcements racing cause no problem, but there is the potential for both *WA-race* and *UW-race* simultaneously. Existing LOUP mechanisms are sufficient to handle this.

The table only covers races that are *inherent* in route dissemination, as it only covers a single triggering event. Both races can also occur due to unlucky timing of external events. Decision modifiers can also prevent loops in an externally triggered *WA-race*, but no router has enough information to specify the predicate for an externally triggered *UW-race*, where the old route is externally withdrawn before the new route has fully propagated. There is no loop, but a transient black hole may result.

In general if updates propagate faster than the information horizon it is possible to encounter brief transients, but those should not occur in practice as there is damping on the eBGP side (MRAI) which prevents such fast churn. The only thing any protocol can do to address such transients would be to trade loops for black holes (an approach taken by DUAL [5] on the IGP side, as we discuss in §7), but we do not believe such optimization is worth the complexity.

In LOUP we choose to use only predicated withdrawals, and not predicated updates. This choice ensures that as long as messages are reliably delivered, the system will not deadlock, as there can be no cyclic dependency.

6. EVALUATION

To evaluate whether LOUP prevents the transients seen with iBGP-RR and BST, we compare the protocols in simulation. We also briefly comment on LOUP's scalability. To build confidence in LOUP's correctness, we ran extensive tests on a wide variety of synthetic topologies. In all cases, LOUP achieved the same solution as full-mesh iBGP.

In the results we present here, we simulate a network based on that of Hurricane Electric (HE), an ISP with a global network. We use publicly available data: HE's complete topology including core router locations and iBGP data [1] that reveals all next hops in the backbone. These next hops are the addresses of either customer routers or HE's customer-facing routers. We assume that there is an attachment point in the geographically closest POP to each distinct next hop. We create a router for each attachment point and redundantly connect it to the closest backbone router. The resulting topology is large enough to be representative of real transit networks—it contains a little over 3000 routers.

Route update packets experience a delay on each link equal to the sum of speed-of-light propagation delay and a uniform random delay in [0, 10] ms, which conservatively models processing delay at a router. Because we underestimate update processing delay significantly (*e.g.,* we do not account for long queues of updates that may form in practice), our simulations should produce *fewer* transients than would be seen in real backbones.

When simulating iBGP-RR, we assume that all core routers are RRs, all attachment points are clients, and all RRs run full-mesh iBGP between them. Recent studies suggest this model is not unrealistic [3].

6.1 Transients

We compare LOUP, BST, and iBGP-RR in two scenarios involving a single prefix: the announcement of a new "best" route for a prefix, and the withdrawal of one route when two routes tie-break on IGP distance. Different prefixes are completely separate and do not affect LOUP's, iBGP's or BST's ability to cause transients, hence we only look at the distribution of a single prefix. We also verified that adding additional prefixes does not change the results significantly.

Fig. 6 shows the protocols' behavior when one BR sends an update, and all routers prefer that update to a route they were already using for the same prefix. As a result, this update triggers a withdrawal of the old route. Fig. 7 shows the protocols' behavior in the tie-break withdrawal case.

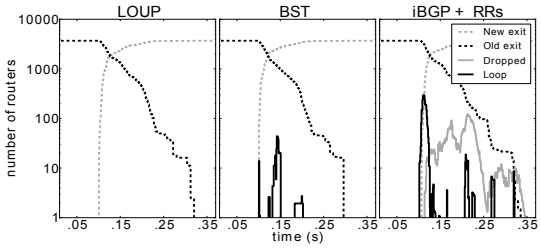

Figure 6: Transients on update

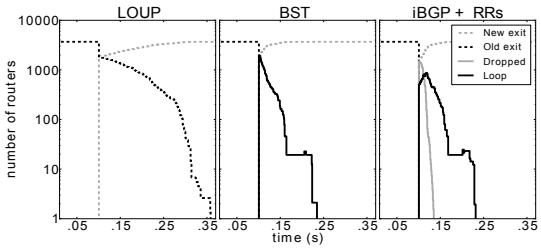

Figure 7: Transients on withdrawal

We are interested in how the prefix's path from each router evolves over time. Define the correct BR before the change occurs as the *old exit* and the correct BR after the change occurs and routing converges as the *new exit*. In these two figures, we introduce the initial change at time $t = 0.1$ seconds. Every 100 μs, each router attempts to route to the prefix; we categorize the state of each router as old exit, new exit, loop, or dropped. The last two categories refer to packets that encounter a loop and those that encounter a black hole, respectively. The y-axis shows the number of routers in each state. We plot the mean of 100 such experiments, each with randomly chosen BRs as the old and new exits.

Fig. 6 confirms that LOUP incurs no transients, neither during the update nor during the following withdrawal. BST and iBGP-RR perform as expected; BST does not cause black holes as predicted in §4.1. LOUP's convergence time is comparable to that of the other protocols. On this topology, there is limited opportunity for races to propagate far, so BST incurs relatively few loops. When it does loop, many paths are affected—the BST results have high variance. The more redundant the network, the more opportunity there is for BST to cause loops.

Fig. 7 shows the necessity of enforcing ordering on withdrawals. LOUP does not cause loops, but it takes longer to converge because the withdrawal first must propagate to the "tie" point and then be activated along the reverse path. All other protocols loop transiently because the BR immediately applies the withdrawal, as explained in §5.2.

6.2 Resource Utilization

LOUP routers store similar numbers of routes to a route reflector, but memory usage is well within the limits of today's hardware. Space constraints preclude our presenting measurements of LOUP's memory consumption. Essentially, the redundancy in iBGP routes allows LOUP to benefit handsomely from compression. More importantly, we find that LOUP, BST, and iBGP-RR incur similar numbers of FIB updates: when an active route changes, the change must be reflected by everyone, regardless of the protocol used.

7. DISCUSSION

We have explored the causes of transient loops and black holes in protocols performing the iBGP role, and have demonstrated a distributed approach to avoiding such problems.

oFIB [12] and DUAL [5] address loop avoidance for topology discovery and path choice in graphs. Trying to reliably disseminate alternative external routes to all routers in an AS is a different problem—we actually believe it to be simpler because removing information from the system is significantly easier (e.g. we do not have to deal with count-to-infinity like DUAL does).

Hybrid centralized/distributed approaches are also possible. By using a central controller that tells BRs when to announce and withdraw routes, but using LOUP for the dissemination itself, we can reduce the exploration of alternative routes, such as occurs when multiple BR's race each other to provide the best alternative to a withdrawn route. Unlike in RCP [2], the central controller in such a hybrid would only be an optimization: if it failed, BRs could fall back to independent operation.

Many ISPs use MPLS to provide VPNs and perform traffic engineering. Some go further, only using BGP in the BRs, with a BGP-free core. A BGP-free core avoids the intra-ordering problems when announcing a new route, because the tunnel exit BR always knows the route before the tunnel ingress BR. It does not however eliminate loops and black holes when routes are withdrawn or become less preferred. LOUP can be used with MPLS to fix this problem.

Finally, a nice property of LOUP is that it is configuration free, so the potential for outages due to configuration errors is greatly reduced.

8. REFERENCES

[1] Hurricane electric's looking glass server, July 2012.
[2] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a routing control platform. In *Proc. ACM/USENIX NSDI*, 2005.
[3] J. Choi, J. H. Park, P. chun Cheng, D. Kim, and L. Zhang. Understanding BGP next-hop diversity. *IEEE INFOCOM*, Apr. 2011.
[4] A. Fabrikant, U. Syed, and J. Rexford. There's something about MRAI: timing diveristy can exponentially worsen bgp convergence. In *Proc. IEEE INFOCOM '11*, 2011.
[5] J. Garcia-Luna-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking*, 1(1), Feb. 1993.
[6] T. Griffin and G. T. Wilfong. Analysis of the MED oscillation problem in BGP. In *Proc. ICNP '02*, pages 90–99, Washington, DC, USA, 2002.
[7] N. Gvozdiev, B. Karp, and M. Handley. Loup: The principles and practice of intra-domain route dissemination. Technical Report RN/12/10, University College London, Oct 2012.
[8] V. Jacobson, C. Alaettinoglu, and K. Poduri. BST - BGP scalable transport. *Presentation at NANOG 27*, Feb. 2003.
[9] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be BGP. *ACM CCR*, Apr. 2007.
[10] N. Kushman, S. Kandula, D. Katabi, and B. Maggs. R-BGP: staying connected in a connected world. In *Proc. USENIX NSDI '07*, 2007.
[11] J. H. Park, R. Oliveira, S. Amante, D. McPherson, and L. Zhang. BGP route reflection revisited. *IEEE Communications Magazine*, July 2012.
[12] M. Shand, S. Bryant, S. Previdi, C. Filsfils, P. Francois, and O. Bonaventure. Loop-free convergence using ofib. *IETF Internet Draft draft-ietf-rtgwg-ordered-fib-06*, June 2012.