Practical Network-Wide Compression of IP Routing Tables

Elliott Karpilovsky, Matthew Caesar, Jennifer Rexford, Aman Shaikh, Jacobus van der Merwe

***Abstract*—The memory Internet routers use to store paths to destinations is expensive, and must be continually upgraded in the face of steadily increasing routing table size. Unfortunately, routing protocols are not designed to gracefully handle cases where memory becomes full, which arises increasingly often due to misconfigurations and routing table growth. Hence router memory must typically be heavily overprovisioned by network operators, inflating operating costs and administrative effort. The research community has primarily focused on clean-slate solutions that cannot interoperate with the deployed base of protocols.**

This paper presents an incrementally-deployable *Memory Management System* (MMS) that reduces associated router state by up to 70%. The MMS coalesces prefixes to reduce memory consumption and can be deployed locally on each router or centrally on a route server. The system can operate transparently, without requiring changes in other ASes. Our memory manager can extend router lifetimes up to seven years, given current prefix growth trends.

***Index Terms*—Network architecture and design, network protocols, network management.**

I. Introduction

THE rapid and sustained growth of the Internet over the past several decades has resulted in large state requirements for IP routers. In recent years, these requirements are continuing to worsen, due to increased deaggregation (advertising more-specific routes) arising from load balancing and security concerns [1], [2], the fact that routers run multiple routing protocols simultaneously (each with their own routing state), and increasing demand for Virtual Private Networks, which requires multiple routing tables.

Memory growth occurs in two different data structures located on routers, known as the RIB and FIB. The Routing Information Base (RIB) stores the set of routes advertised from neighboring routers. The RIB must store a copy of attributes and reachability information for hundreds of thousands of prefixes, which must be kept up-to-date in the presence of failures and network churn. The Forwarding Information Base (FIB) contains entries that map incoming packets to outgoing links. In the FIB, state must be stored in very fast (and typically very expensive and power-hungry [3], [4], [5], [6]) memory for packet lookups; even though it is much smaller in size than the RIB, the cost per megabyte is considerably higher. RIB and FIB sizes are determined by many factors, but are both impacted by the number of routable *prefixes* (*i.e.*, sets of reachable IP addresses).

Growth of memory requirements presents a serious problem to ISP operators. Routing protocols are not designed to handle scenarios where memory is exhausted, leading to *incorrect* behavior when this occurs. Memory exhaustion leads to highly serious failure modes, such as route oscillations and incorrect forwarding decisions [7]. To protect against this, network operators are forced to repeatedly upgrade their equipment at enormous expense due to the large cost of doing an in-field deployment of new hardware. To avoid repeated field deployments, network operators can aggressively over-provision memory on routers. However, provisioning is itself a highly challenging problem because memory requirements depend on external factors outside the ISP's control. In addition, misconfigurations such as "route-leaks" cause temporary spikes in the number of advertised routes and are hard to predict. When faced with overload conditions, operators can employ route filters to restrict the amount of information learned by a router, but these filters may disrupt connectivity.

There have been many proposals in the research community to solve this problem, but unfortunately these techniques have not been deployed. Many of these solutions are not backwards compatible with current protocols, hindering deployment. While clean-slate design proposals are interesting and worthy of consideration, they often require massive structural changes and new protocols, which may limit their usage for the foreseeable future. As an alternative, our work considers incrementally deployable solutions, Our solutions can be deployed in isolation as a single Autonomous System (AS), without requiring changes to router hardware or software, and without requiring cooperation from neighboring ASes.

One work that serves as our inspiration is Optimal Route Table Construction (ORTC) [8]. The ORTC algorithm operates only on FIB memory, taking a FIB as input and producing a more compact FIB as output. It guarantees that the compact FIB has the exact same forwarding behavior as the input, and given that constraint, that the output FIB has a provably minimal number of entries. Experimental tests conducted in 1998 have shown that it can reduce the number of FIB entries by up to 50%. Despite this benefit, ORTC has not been adopted in practice, as it suffers from several major drawbacks. First, it is computationally expensive: the original implementation

Manuscript received February 4, 2012; revised June 29, 2012. The associate editor coordinating the review of this paper and approving it for publication was D. Medhi.

E. Karpilovsky is with Google.

J. Rexford are with Princeton University.

M. Caesar is with the University of Illinois at Urbana-Champaign (e-mail: mccaesar@gmail.com).

A. Shaikh is with AT&T Labs–Research.

J. van der Merwe is with the University of Utah.

Digital Object Identifier 10.1109/TNSM.2012.12.120246

1932-4537/12/$31.00 © 2012 IEEE

takes approximately 500 milliseconds[1] to run for *every* routing update received; in modern networks routers must process tens of updates per second on average and tens of thousands of updates per second during spikes [9], making it difficult to use this algorithm in practice. Moreover, it is inflexible; it must always produce an output that forwards *exactly* the same as the input. However, there may be times when even a "compressed" FIB will not fit in memory. In this case, it may be preferable to alter forwarding behavior to allow further compression instead of allowing the router to crash. If these two problems were fixed, ORTC could be a useful building block in a larger system that managed memory.

A. Managing ISP Memory with an MMS

The focus of our research is to improve performance of the ORTC algorithm to enable its use in practical settings, to measure its use in modern networks, and to leverage it to design a generic *memory management system* (MMS) to manage the memory usage in the routers of an ISP's network. We apply several techniques to greatly boost the runtime speed of the algorithm (the vast majority of updates from a tier-1 ISP network are processed in less than a millisecond). Moreover, the MMS provides multiple levels of compression, allowing for a trade-off between unaltered routing and "maximal memory compression."

The MMS can be deployed either *locally* on each router or in a logically-centralized system that monitors and compresses state at all routers in the *AS-wide* network. In a local deployment, each router independently performs the operations of an MMS over its own local routing state. This enables our system to run in a completely distributed fashion. However, this does have some drawbacks. It requires router software upgrades and possible hardware upgrades (if CPU power is lacking). Moreover, there are limitations to the potential memory savings, as routers still need to maintain BGP control sessions (and hence cannot compress RIBs, only FIBs) with neighboring routers, and also because each router only has a local view of the network and acts independently.

To circumvent these problems, the MMS can also be deployed in an AS-wide setting, where it runs on a set of servers that collectively assume responsibility for the routing interaction of an AS with neighboring ASes [10], [11], [12]. The MMS receives routing updates from neighboring ASes, preprocesses these updates before sending routes to routers within the MMS-enabled network, and communicates selected routes to neighboring ASes. Neighboring ASes can be configured to send updates directly to the MMS, rather than to the border routers. If neighboring ASes do not wish to perform any re-configuration, border routers can act as proxies and relay BGP messages between the MMS and neighboring ASes. Not only does this deployment enable extra compression, but this approach allows for additional amortization techniques to be applied.

B. The Benefits of the MMS

Our design has several benefits:

[1]We verified this number by running our own experiments on a Pentium-IV, 3.6GHz processor (comparable to control-plane processors in modern routers).

Flexibility: By default, MMS operates in a transparent fashion, with absolutely no changes to the way routes are chosen and packets are forwarded. In this "transparent mode" external networks (e.g., other ASes) need not be aware that an MMS has been deployed. In such a situation, the MMS can still provide about a 50% reduction in router memory across the entire network, without altering forwarding behavior. If more memory savings are desired, the MMS can shift paths to attain additional memory reduction, up to 70%. However, routes selected for forwarding may differ from the "transparent" case. We provide algorithms to automatically perform a small set of routing changes that increase compressibility without operator involvement. It is important to note that even if paths are shifted, the system remains inter-operable with routing protocols and does not introduce any routing loops.

Reduced Operational Cost: The MMS can simplify capacity planning and extend the lifetimes of older routers. We demonstrate this through experimental results conducted within a large tier-1 ISP backbone: using local-mode, FIB memory usage is reduced between 50% to 70%, the rate of increase of table growth is decreased by a factor of 2.2, and variation in table size is reduced by a factor of 2.6 (reducing variability increases the accuracy of future provisioning predictions). Given current levels of routing table growth [13], these reductions can be expected to increase lifetimes of routers needing immediate replacement by up to seven years. In particular, our results collected on the tier-1 ISP backbone indicate an increase in table growth by a factor of roughly 2.2 per year. If we measure the rate of growth of an uncompressed router's FIB, and compared that to a compressed router's FIB, given current routing table sizes and growth rates, it currently takes about 4 years for the size of a routing table to increase by 50%, while it takes 11 years for compressed FIBs to grow the same amount. Moreover, since the MMS can operate in the form of a logically-centralized cluster (or a small redundant set of clusters), it can form a small set of locations where resources may be upgraded, reducing expenses associated with field deployment.

Safety: Routers near their memory limits can use the MMS to increase the amount of available resources. This improves resilience to misconfigurations in neighboring networks. Moreover, given that our compression techniques perform better with increased levels of deaggregation, our approach could enable interdomain routing on fully-deaggregated $/24$ prefixes, which has benefits in terms of routing flexibility and mitigating hijacking attacks. Use of the MMS can *guarantee* that routers will not reach overload conditions (which can trigger reboots) by selectively filtering new prefix advertisements before overload is reached.

Incrementally Deployable: A single ISP can deploy an MMS while maintaining interoperability with existing protocols, and without requiring cooperation from neighboring ASes, in both local and AS-wide deployment modes. In AS-wide deployment mode, our MMS design requires no changes to existing router hardware or software. Furthermore, this deployment may proceed in an incremental fashion (e.g., over a period of time),even within a single AS, by having the MMS only control a limited subset of routers within the ISP. In

```
TIME: 08/27/09 03:30:29
TYPE: BGP4MP/MESSAGE/Update
FROM: 213.144.128.203 AS13030
TO: 128.223.51.102 AS6447
ORIGIN: IGP
ASPATH: 13030 8342 12389 9198
NEXT_HOP: 213.144.128.203
MULTI_EXIT_DISC: 1
COMMUNITY: 13030:1 13030:1016
ANNOUNCE
  88.204.221.0/24
  95.59.1.0/24
```

Fig. 1. BGP update message from Route Views on August 27, 2009, announcing that two prefixes are reachable through the same next-hop.

local deployment mode, the MMS can be loaded as a protocol daemon update to route software, so this deployment approach does not require changes to router hardware (aside from CPU upgrades, which our results on computational overheads indicate should rarely be needed). In this case, the MMS can deployed at a single router, with no changes required to external protocols or neighboring routers.

II. Memory Saving Approaches and Limitations

The primary goal of the MMS is to reduce router memory usage within an ISP. To do this reduction, the MMS performs *route coalescing*, i.e., replacing groups of routes sharing the same next-hop with smaller, equivalent sets. Although this seems like a simple procedure, several operational challenges of ISPs make this process quite complex. In this section we describe the challenges in route coalescing through several examples. We show that naïve approaches can introduce inconsistencies in packet forwarding, and we motivate why our design decisions are necessary.

A. Routing across ISPs

The Internet is composed of a collection of *Autonomous Systems (ASes)*, each of which corresponds to a single ISP, enterprise network, or other organizational entity. Each AS has a set of border routers which communicate to border routers of adjacent ASes through the use of the Border Gateway Protocol (BGP). BGP communicates information about routes and constructs forwarding paths to allow data packets to flow across ASes. Paths are newly advertised or withdrawn by exchanging update messages containing reachability information (shown in Figure 1). The updated routing information replaces old information and is used for forwarding data packets. After processing an update, the router notifies its neighbors if any routing changes occurred.

BGP is a path vector protocol, where routers exchange the entire AS-level path they use to reach the destination. Each AS has a globally unique AS number. When routes are propagated, the current AS adds its AS number to the head of the AS path contained in the routing update. This allows other networks to quickly detect if the path contains routing loops (by scanning for their own AS number in the list) as well as providing a simple metric for determining which routes are shorter than others (by preferring routes with fewer AS-level hops).

BGP propagates routes for prefixes, which denote a collection of host addresses immediately adjacent in the IP namespace. Prefixes are represented by an IP address followed by a mask. For example, the prefix $12.1.0.0/16$ represents all IP addresses whose first 16 bits match 12.1. Prefixes specify reachability on multiple levels of granularity, creating ambiguity in reachability information. For example, a route to $12.0.0.0/8$ could have a next-hop of $1.1.1.1$, while a route to $12.0.0.0/9$ could use $2.2.2.2$. To eliminate this ambiguity, routers select the longest matching prefix when there are multiple choices. However, longest prefix matching significantly complicates aggregation, *i.e.*, the ability to take two prefixes with the same next-hop information and combine them into a single, larger prefix. An example of such a complication with aggregation is shown in Figure 2. To avoid introducing such difficult-to-predict side effects, ISPs are constrained in the types of aggregation they can perform.

Although ISPs cannot aggregate advertised routes (RIB), they can aggregate forwarding entries (FIB). As previously shown, even if two prefixes have the same next-hop, an ISP cannot announce an aggregate route, as it causes problems for other ASes. However, in the case of forwarding, there are no negative effects from such aggregation. Aggregating FIB entries is completely transparent to other routers; an aggregated FIB forwards exactly the same as a deaggregated one. Moreover, if we choose routes from the RIB that have the same next-hop, we can aggregate these entries in the FIB. In other words, our choices of routes in the RIB will determine the compressibility of the FIB.

To summarize, *Autonomous Systems cannot advertise compressed routes to neighboring ASes*. While forwarding entries can be coalesced, routing entries cannot.

B. Routing within an ISP

ISP networks earn revenue by providing transit service, *i.e.*, by forwarding traffic between their neighbors. Hence, ISPs must share reachability information received from one neighbor with the others. This is often done by establishing BGP sessions between border routers (when BGP is run within an ISP, it is referred to as iBGP). Internal reachability between border routers is provided by an intra-domain routing protocol such as OSPF [14] or IS-IS [15]. iBGP sessions are sometimes established in a full-mesh configuration, where each border router maintains a session to *every* other border router. However, since routers must maintain routing state separately for each iBGP session, full-mesh configurations can have very large RIB memory requirements. For example, if there are n border routers, then each border router may need to store and maintain up to $n-1$ internal routes for each of the hundreds of thousands of prefixes in the routing table.

To circumvent this problem, larger networks often deploy route reflectors [16] at strategic locations within their network. Route reflectors act as internal accumulation points, which collect routing updates from a subset of border routers, and only advertise the most preferred route to their iBGP neighbors; as such, border routers only receive the most preferred routes from their associated route reflectors. Unfortunately, the use of route reflectors introduces a set of problems. They can induce persistent forwarding loops and oscillations if deployed improperly [17]. They require additional work for

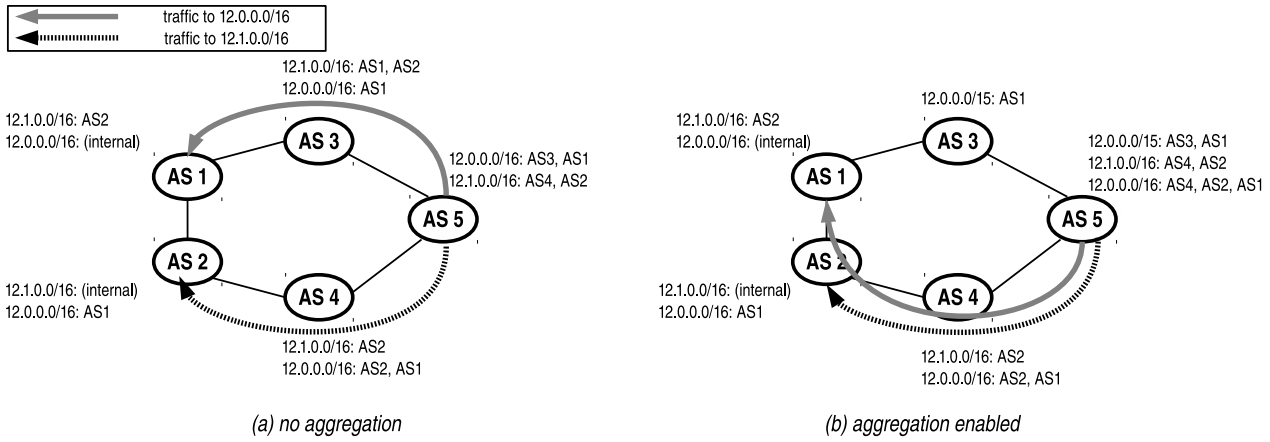

Fig. 2. Aggregation can have unintended consequences: (a) Suppose AS 1 originates 12.0.0.0/16 and AS 2 originates 12.1.0.0/16. When no ASes perform aggregation, AS 5 can route traffic to 12.1.0.0/16 via AS 3, and traffic to 12.0.0.0/16 to AS 4. (b) However, if AS 3 decides to aggregate 12.1.0.0/16 and 12.0.0.0/16 into 12.0.0.0/15, AS 5 can no longer use the route via AS3. The reason is that all of 12.0.0.0/15 is covered by more specific prefixes that are reachable via alternate exit points, and Internet routing always prefers more-specific prefixes.

network operators to maintain, as they must be reconfigured to match changes in the underlying network topology. While route reflectors reduce memory usage, they do *not* reduce the number of prefixes in the routing table. Hence route reflectors do not reduce the size of the router's *forwarding* table (which is commonly stored in expensive, fast memory).

C. Router-Level Routing

Routers are logically divided into a *control plane*, which contains the RIB, and a *data plane*, which contains the FIB. The goal of the control plane is to compute the set of routes the router should use locally, and of these, which should be advertised to neighboring routers. The goal of the data plane is to forward data packets, by selecting from a set of next-hops computed by the control plane. In addition to storing the next-hop and prefix information, the RIB also stores a set of *attributes* that define properties of the route (e.g., the AS-path, cost metrics, where the route was learned from). The RIB also stores multiple routes per prefix—this is done so that if the currently-used route fails, the router may use an alternative route through a different neighbor to circumvent the failure. Unfortunately, when routers run out of memory, they can continuously reboot, crash, or begin behaving incorrectly [7]. Reducing RIB memory is quite difficult. RIB entries contain routing information that may be vital when primary links fail and backup routes are needed. Moreover, routing information is often exchanged between routers and used to determine forwarding paths. As such, care must be taken when attempting to reduce RIB memory – data cannot be simply discarded.

The *FIB* stores the set of routes which will be used to forward packets to individual prefixes. The FIB must perform forwarding lookups very quickly and are hence typically implemented in fast memory with low access times, such as SRAM or TCAM. There are two restrictions regarding FIB memory reduction. First, the contents of the FIB must "match" the RIB (each entry in the FIB should be the most-preferred route in the RIB) to prevent routing loops. Therefore, prefixes can be coalesced if such actions do not change the forwarding behavior advertised by the router. Figure 3 provides an example. Second, FIB reduction techniques must be extremely fast. If an algorithm is too slow, a router may not be able to handle the high rates of updates present in modern networks.

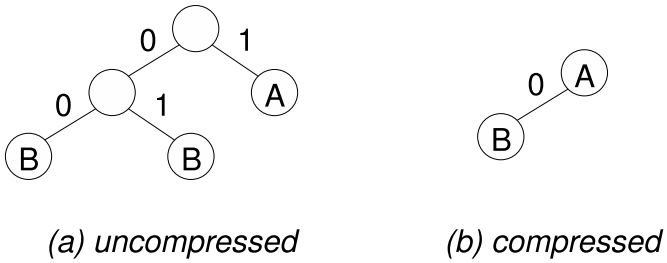

Fig. 3. Example of prefix coalescing over tries. Both FIBs forward packets in the same way, but FIB (a) needs to store three prefixes, while FIB (b) only needs to store two.

III. The MMS in Local Deployment

There are fundamental problems with trying to compress routes: prefixes cannot be coalesced when announced, FIB compression is limited by RIB decisions, compression algorithms must be fast, *etc.* In this section, we discuss how our Memory Management System can circumvent some of these problems when deployed in "local mode" on individual routers. We demonstrate how the MMS can provide flexibility in aggregation for the FIB without introducing network problems. Moreover, we show how techniques such as parallelization and incremental computation can be used to significantly speed-up the ORTC algorithm, which is used as a building block for the MMS. The local mode MMS also serves as a basis for the AS-wide MMS, which is discussed in the next section.

Although the MMS can be used to reduce FIB memory consumption, the RIB cannot be easily compressed in "local mode." A router may need backup routes in case of primary route failure, and may need to advertise information about them to neighboring ASes in such a situation. As such,

we focus on FIB compression. Later, during the discussion of "AS-wide mode," we demonstrate how the RIB can be compressed.

Algorithm 1 Pseudo-code for the ORTC algorithm. Each node represents a different prefix. **rib_info** represents the chosen route for a prefix (as dictated by the RIB). NULL next-hop indicates no FIB entry needed for that prefix.

```
// Normalization: all nodes to have 0 or 2 children.
for node N in t in preorder traversal:
  if N has one child:
    create missing child for N
    child inherits N.rib_info

// Prevalent hop calculation: find the set of
// maximally coalescable next-hops.
for node N in t in postorder traversal:
  if N has no children:
    N.prev_set = {N.rib_info}
  else:
    N.prev_set is the intersection of its
    children's prev_sets

  if N.prev_set == ∅:
    N.prev_set is the union of its
    children's prev_sets

// Next-hop selection.
for node N in t in preorder traversal:
  if N is root of t:
    N.next_hop = arbitrary element
    of N.prev_set
  else:
    clst = closest ancestor of N with
    non-NULL next-hop
    if clst.next_hop ∈ N.prev_set:
      N.next_hop = NULL
    else:
      N.next_hop = arbitrary elem
      in N.prev_set
```

A. A Fast FIB Compression Implementation

Draves *et al.* [8] previously proposed an Optimal Routing Table Construction (ORTC) algorithm, which takes a forwarding table as input, and computes the provably smallest [2] forwarding table that performs forwarding in an equivalent manner. Algorithm 1 outlines their algorithm, which assumes a binary tree representation known as a trie data structure. ORTC works by making three passes over the trie, in steps known as *normalization*, *prevalent hop set calculation*, and *next-hop selection*. The normalization step enlarges the binary tree representing the routing table such that every node has zero or two children. It does this by adding new leaf nodes as necessary, and setting the next hop for new nodes with the next hop of its nearest ancestor. The prevalent hop set calculation step then determines the set of next hops that occur most commonly in the tree, and labels each internal node with the set of possible next hops it could be labeled with. The idea here is that if a next hop occurs very commonly in a particular branch of a tree, we may be able to reorganize that branch such that the most popular next hop appears closer to the root, reducing the size of that branch. This is what the next-hop selection step does. This step moves down the tree selecting next hops for each internal node, eliminating redundant routes along the way. A more detailed description of this approach is given in [8]. The authors of [8] provide several optimizations to speed up this computation. They also extend the algorithm to deal with multiple next-hops per prefix and default routes.

Unfortunately, even with optimizations, ORTC is too slow to use online in modern networks. While the authors were able to optimize run time down to several hundred milliseconds for the smaller forwarding tables that existed when their paper was published, these run times remain too slow in modern networks which can burst to tens of thousands of updates per second. We leverage the techniques of parallelization and incremental updates to augment this algorithm, speeding it up so it can be used with the MMS.

1) Parallelization: Parallel algorithms are becoming increasingly important as chip manufacturers move to multicore designs. Conventional wisdom is now to double the number of cores on a chip with each silicon generation, while the speed of each core grows much more slowly or remains constant [18]. As commercial routers typically use commodity CPUs for control-plane processors, this provides the opportunity to leverage this increased parallelization in our design. Such trends in processor design can be exploited to help compression algorithms keep pace with the increased computational load associated with the growth and churn of Internet routing tables.

There has been substantial previous work on parallel algorithms for graph structures [19]. Our design (Algorithm 2 is loosely based on these techniques and consists of two stages. In the first stage, all nodes associated with $/8$ prefixes are added to a queue. When a thread becomes available to perform work, it selects a node from the head of the queue, and performs compression on the sub-trie rooted at that node. To ensure correctness, it is important that no other threads concurrently process any nodes in that sub-trie. As a result, a thread locks all descendants of that node. In the second stage, a single thread performs the rest of the remaining compression for the nodes that have not been processed. Note that the second stage could be parallelized as well to further decrease computational time.

2) Incremental Computation: The ability to incrementally update data structures is crucial for speed. The benefit of an incremental approach is that changes to a single prefix do not require recomputing the router's entire FIB. However, with ORTC compression, this is no longer true – changing a single prefix may trigger other routes to become coalesced (or to uncoalesce). The naïve way to deal with this would be to rerun ORTC after every received update. However, doing this would be wasteful, as the vast majority of routes would not change after a particular update is received. Furthermore, some

[2]With respect to the number of prefix/next-hop pairs.

Algorithm 2 Pseudo-code for parallel execution.

```
// Build a queue of all /8 prefixes
Q = {∀ /8 nodes in tree}
while Q.size > 0:
  curr=Q.head
  // block until next thread is available
  T=get_next_thread()
  // traverse curr's subtree, locking all nodes
  // starting at and including curr
  for node n in sub-trie rooted at curr
    lock(n)
  T.execute:
    // run Algorithm 3 (compression algorithm) on
    // subtree rooted at curr, in newly available thread
    compress(sub-trie rooted at curr)
```

updates do not require any recomputation (for example, an update that removes a route that is not used by any routers in the network). An alternate way to deal with this would be to periodically process batches of updates at fixed intervals [8]. However, such an approach increases the amount of time needed before a router can respond to a chance in the network. For example, if a new BGP update arrives, we need to wait until the next processing time of updates occurs before the new path advertised in the update can be used by the protocol.

To address this challenge, we developed an *incremental algorithm*, which only processes the portion of the ORTC trie that is affected by a received update. Pseudocode for this algorithm is shown in Algorithm 3. The idea behind this code is the following. When a new update is received, only a subset of the trie needs to be updated. Instead of recomputing the entire trie from scratch, if we can determine the subset of the trie that needs to be recomputed, we can reduce computation time by only recomputing that subset of the trie. Luckily, it turns out that figuring out which subset of the trie is affected is possible to do: we simply need to find the location where the update's prefix would be placed in the trie, and "trace back" all affected nodes in the trie.

To clarify this process, let us walk through the pseudocode in Algorithm 3. This code calls four subroutines. First, *mod_normalize* performs the normalization step, which enlarges the binary trie so that every leaf has either zero or two children, creating new leaf nodes as necessary. This process is the same as normalization in ORTC, except that if a node is not modified by normalization, children of that node are not normalized. Second, *mod_calc_prev_set* computes the set of prevalent next hops (the set of next hops that are most commonly used in the trie, in preparation for moving the most prevalent hops closer to the root of the trie). This step is the same as the prevalent hop calculation step in ORTC, except that if a node N has no children, *N.prev_set* is set equal to *N.rib_info*. Third, all affected ancestors of N are normalized using *mod_ancestor_normalize*. Here, ancestors are processed in ascending order. If a node was not modified by normalization, its ancestors are not normalized. The *highest* variable is updated to refer the highest ancestor normalized. Finally, the *mod_select_next_hop* function computes new next hops as needed. This is the same as the next-hop selection procedure in ORTC, except that if the *next_hop* of a node is unchanged, that node's children are not processed.

To clarify operation of Algorithm 3, we next give an example of its operation. Consider a trie with two (prefix, next-hop) pairs: $(0.0.0.0/0, 1.1.1.1)$, and $(0.0.0.0/1, 1.1.1.1)$. This trie can be compressed to a single (prefix, next-hop) entry: $(0.0.0.0/0, 1.1.1.1)$. Now consider the announcement of a new route: $(128.0.0.0/1, 2.2.2.2)$. Adding this new route does not change forwarding behavior of $0.0.0.0/1$. The forwarding behavior does change for $0.0.0.0/0$, though, and it (along with new nodes) will need to be re-evaluated for compression gains. $0.0.0.0/1$, however, can be unaware of any such computations, as long as $0.0.0.0/0$ still "covers" it by forwarding toward 1.1.1.1. The new compressed trie, $(0.0.0.0/0, 1.1.1.1)$, $(128.0.0.0/1, 2.2.2.2)$, is optimal and does not require a full computation.

While parallelization and incremental updates could be combined, it is unclear how well they would work together. Parallel algorithms work best with large sets of data that can be processed independently, while the incremental algorithm attempts to operate over small sets of nodes that may have inter-dependencies. As such, the overhead from thread locks may outweigh the amount of possible parallelization savings. Thus, we consider these techniques to be complementary and useful for different situations. For example, if an operator enabled compression, a parallelized full computation would be faster than the incremental algorithm (since a full computation is needed anyway, and a parallelized version has additional opportunity for speedups). However, if a router is simply processing updates received in normal BGP communication, the incremental version would most likely be the fastest algorithm to use.

Algorithm 3 Pseudo-code for the incremental update algorithm. **rib_info** represents the set of routes passed to a prefix from the RIB. Incrementally update a trie **t** with an update **u** from neighboring router **u.neighbor**.

```
// Update the node with the new routing information.
N = node in t associated with u.prefix
if u is an announcement:
  N.rib_info -= {old next_hop of u.neighbor}
  N.rib_info += {u.next_hop}
else:
  N.rib_info -= {old next_hop of u.neighbor}

// Normalize all affected children of N.
mod_normalize(sub-trie rooted at N)

// Calculate the prevalent hop set.
mod_calc_prev_set(sub-trie rooted at N)

// Normalize all affected ancestors of N.
highest = N
mod_ancestor_normalize(N, highest)

// Compute new next-hops as needed.
mod_select_next_hop(sub-trie rooted at highest)
```

B. Selecting Routes to Improve Compression

Although ORTC coalesces the prefixes in a FIB, it is bound by the requirement that the forwarding behavior is unchanged. In this section, we demonstrate how it is possible to further improve the compression results by allowing the MMS to modify the forwarding behavior.

As previously mentioned, the BGP decision process, shown in Figure 4, is run over the RIB to select the route to populate into the FIB. This decision process uses a series of rules to pick routes. Each rule eliminates a subset of routes, and rules are applied until a single route remains [20]. The router (1) first chooses the routes with the highest LocalPref (a numeric value assigned by the operator to indicate which next-hops are most preferred), then (2) the routes with shortest AS-path length (the routing update contains the AS-path, which is the sequence of AS-level hops to the destination), then (3) the routes with the lowest origin type (a flag indicating whether the route originated internally or externally to the ISP), then (4) routes with the lowest MED (a numeric value advertised by a neighboring ISP, to indicate which entry point should be used, when the two ISPs peer in multiple locations), then (5) routes learned through eBGP (BGP sessions with neighboring ASes) are preferred over iBGP routes (routes learned through other border routers in the local AS), then (6) the router chooses the closest exit point (or shortest internal route) to reach the destination prefix, then (7) to break ties, if multiple options still exist, the router chooses the route advertised by the router with the smallest router ID. The process is designed around several goals, such as maximizing revenue (through local preference settings), attempting to minimize latency (through shortest AS paths), load balancing (through IGP metrics), and so on.

The BGP decision process constrains the level of compression achievable, as it places constraints on the set of routes that are populated into the FIB. To improve compression further, the MMS allows the operator to select *sets* of routes that are acceptable for use. By allowing the compression algorithms flexibility to choose amongst this set, additional compression can be achieved. In particular, an operator configures the MMS with a *threshold level*. The threshold level determines how many steps of the BGP decision process to execute. All routes that are equally good at a particular level are considered possible routes for the FIB. A route coalescing algorithm is then computed over these possibilities. For example, a "level 0" setting would not run any steps of the decision process, and use all possible routes in the coalescing algorithm. A "level 1" setting would select all routes that remain after applying step 1 of the decision process; a "level 2" setting would select all routes that remain after applying steps 1 and steps 2; and so on.

It is important to note that such flexibility requires the use of tunnels between border routers. Without tunnels, packets may be forwarded in a different manner than expected. For example, consider the network depicted in Figure 5a. Routers A and B both use their external links to reach $1.2.0.0/16$. It is possible for router D to choose A for forwarding to this prefix, while router E chooses B. However, both D and E must go through C. If C decides to forward traffic to $1.2.0.0/16$ through A, then router E's choice is invalidated. Since BGP specifies a single next-hop for a given prefix, this problem is unavoidable. To overcome it, tunnels between border routers can be used. Tunnels have the additional benefit of freeing memory in the core of the network. Such BGP-free cores are feasible to implement (*e.g.*, using GRE or MPLS tunnels) and are often used in practice.

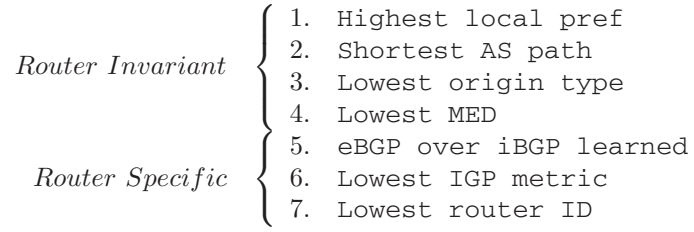

Fig. 4. The BGP decision process.

Flexibility in route selection may cause routes to change from the original forwarding behavior; however, such deviation may be tolerable. If a router is at risk of memory exhaustion, higher levels of compressibility can ensure reachability information is not discarded, even if routing to those destinations is suboptimal. Moreover, the amount of differentiation can be tweaked, offering more differentiation and savings in some situations and less differentiation and savings in others. In addition, this approach can be used as a fallback mechanism that is enabled only if the level 7 (*i.e.*, no differentiation) compressed routing table size would exceed router capacity.

C. Limitations on Route Selection

Care must be taken whenever deviating from the BGP decision process, as routing loops or oscillations could occur. These problems can occur at either the inter-AS level or intra-AS level.

At the inter-AS level, due to the relationships that exist between Autonomous Systems, such problems can be avoided at the inter-AS level as long as step 1 of the BGP decision process is always applied. This is because step 1 is primarily used by ISPs to encode relationships, with customers often receiving higher local preference values than peers, and peers receiving higher local preference values over providers. As long as ASes are routing according to economic incentives, loops and oscillations should not happen [21]. However, sibling ASes (that is, ASes that appear to be separate but are actually owned by the same organization) may not be able to use route selection, depending on their setup. This problem arises because the MMS assumes a BGP-free core, and sibling ASes would need to use BGP to communicate between themselves; since sibling ASes are owned by the same organization, they can be thought of combining to form a super-AS, and the inter-connections of the siblings would require BGP in the "core" of the super-AS.

At the intra-AS level (in local deployment mode), the MMS cannot be overly aggressive with route selection. Oscillations can occur if we are not careful and routers act independently. For example consider router A and router B in a network that both have external routes to the same prefix. If enough BGP decision steps are ignored, router A might decide that it

should simply forward everything to B. In this case, it would withdraw its reachability information from B, since it is using B for routing. Likewise, B would do the same thing for A. If the events are synchronized, the routers may oscillate between using their externally learned routes (and thus re-announcing them) and each other (and thus withdrawing them). Although such oscillations may not be common in practice, it is nonetheless noteworthy.

To solve the oscillation problem, each MMS should be configured so that step 5 (eBGP preferred over iBGP) is always some part of the decision process. This configuration prevents oscillation because, due to the dynamics of iBGP, iBGP learned routes always point to a router that has an eBGP learned route. Thus, all routers in a network fall into one of two cases:

- *The router has at least one eBGP-learned route to choose from after applying the modified BGP decision process*. In this case, the MMS forces the router to pick an external route, preventing intra-AS problems.
- *The router has all iBGP-learned routes after applying the modified BGP decision process*. Using any of these routes will send packets to a router that has at least one eBGP-learned route. The previous case applies to that router, and packets will be forwarded using the eBGP route, preventing intra-AS problems.

In summary, as long as the modified BGP decision process includes step 5 (the eBGP comparison), intra-AS oscillation can be avoided. For example, applying steps 1 through 3 and 5 (while ignoring 4) would be sufficient, but applying steps 1 through 4 would not. The increased flexibility can lead to better compression. In the next section, we discuss AS-wide deployment of the MMS, and discuss a different mechanism to solve this problem.

IV. Optimizations in AS-Wide Deployment

As an alternative to being deployed at a single router, the MMS may be deployed across an AS. The overall architecture of an AS-wide MMS is shown in Figure 5b. It can be implemented through a logically-centralized architecture which offloads memory management functionality to a small set of servers. These servers are completely responsible for disseminating routing information to routers within the ISP. The MMS directly maintains peering sessions with neighboring ASes, offloading the responsibility from its associated border routers. The Memory Management System maintains a network-wide view including the routing preferences of and routing updates received by all border routers. Thus, the MMS can locally maintain a routing table on behalf of each BGP-speaking router in the network. The MMS can compress the routes and send the compressed information to the border routers (while sending the uncompressed information to other autonomous systems). To design the server infrastructure for the MMS, we rely on previous work that shows offloading routing can be deployed at scale and with resilience [10], [11], [12]. For example, when deployed across an AS, the MMS should be replicated to improve fault tolerance. We use an approach similar to a Routing Control Platform (RCP) [10], having one server act as a primary in charge of distributing routes throughout the network, with the rest of servers acting as backups.

This approach has several benefits. First, our centralized approach offloads computation from routers, freeing up computational resources for other protocols or for speeding convergence. Second, as opposed to the local deployment mode, this approach requires minimal changes to existing routers (no changes to protocols or router software is required). Third, common computations across routing tables could be amortized to yield further computational savings.

A. Compressing FIB Entries

In AS-wide deployment mode, the MMS obtains all the FIB compression benefits from the local deployment mode. In its simplest setting, the AS-wide MMS runs an instance of a local MMS for each router, performing all the computation on behalf of the routers. Moreover, because the AS-wide MMS has a complete view of the network, it avoids the problem of routing loops caused by incomplete routing information. The MMS dictates all forwarding decisions such that no routing loops occur.

In addition, the MMS *amortizes* some of the computational steps across the network by performing them once instead of repeating them for each router. For example, before step 5 of the decision process, all routers with the same routing information will make the exact same decisions regarding "equally good" routing sets. This phenomenon occurs because the first four BGP decision process steps are always the same for every router, if given the same set of routes. However, not every router will have the same set. For example, if router 1.1.1.1 has an eBGP learned route r with next-hop 2.2.2.2 and advertises it, all other routers will see r as having next-hop 1.1.1.1. However, if routers share similar sets, the computations can be amortized.

To efficiently compute compressed FIBs (and RIBs) in an amortized fashion, the MMS first computes sets of routes that are equally good according to the first N steps of the BGP decision process, where N is the threshold level. All routers in the network must select a route from this set. A smaller computation is then done to further select routes on a per-router basis (that deviate from this "common" case). This approach consists of two separate stages:

Stage 1, compute common FIB: First, the MMS computes a compressed FIB that all routers in the network share. In particular, the MMS logically creates a virtual *internal* router, which receives all routes from every border router in the network. The MMS then constructs a compressed FIB for this router.

Stage 2, compute router-specific differences: At first glance, it appears that every router in the network should use the common FIB computed in stage one. However, this is not the case. For example, consider a network that picks next-hop 1.1.1.1 for prefix p. If 1.1.1.1 is a border-router in the network, then everyone can route successfully *except for 1.1.1.1*. Since 1.1.1.1's forwarding table would state that the next-hop is 1.1.1.1, the router would forward packets to itself. To avoid this scenario, the MMS computes, on behalf of each router in the network, which outgoing link is best suited to

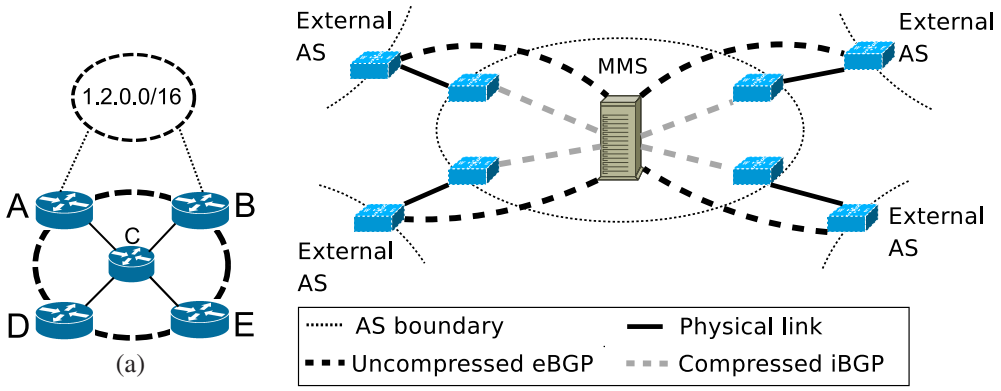

Fig. 5. (a) An example network with four border routers and one internal router. Dashed lines represent AS boundaries, solid lines indicate links, and dotted lines represent paths to the AS that owns $1.2.0.0/16$. (b) An example network showing peering with neighboring domains. Note for backwards-compatibility reasons, the MMS uses BGP to communicate routes to routers (hence routers only need to store a single RIB corresponding to their session to the MMS). Border routers at other autonomous systems speak directly to the MMS, which is permissible as BGP is run over the TCP/IP protocol. The MMS then sends coalesced information to its own routers for forwarding.

TABLE I
TIER-1 RESULTS FOR A SINGLE ROUTER ON A SINGLE DAY. RESULTS WERE SIMILAR FOR OTHER ROUTERS IN THE NETWORK.

Threshold Level	FIB Entries	% of original size
1	110890	35.6%
2	119150	38.3%
7	130842	42.0%
Uncompressed	311466	100.0%

forward traffic for each prefix in its routing table. For example, in the example above, the MMS computes (on behalf of 1.1.1.1) which one of 1.1.1.1's outgoing links is best suited for forwarding traffic to p, and sends that information to 1.1.1.1.

It is important to note that this amortization resolves the oscillation problem mentioned above, since the MMS ensures that the border router responsible for a prefix picks an eBGP route. Further, two separate compression steps do not necessarily produce the smallest possible FIB for each router (unlike ORTC, which is provably minimal in the number of prefixes it produces). However, our results indicate that the MMS compresses well.

B. Compressing RIB Entries

The AS-wide MMS has the opportunity to reduce the amount of redundant routing state in a network. First, instead of maintaining multiple iBGP sessions, each router only maintains one (with the MMS), reducing the number of RIBs that need to be maintained. Second, every time a route is announced and propagated, it may be stored on every router that receives it. Individually, each router may not be able to remove RIB entries, since it may need to transmit the information to neighboring ASes; thus, reducing the redundancy may be difficult. However, the MMS can act as a central database to store all such routes. Only one copy of the route need be stored in this case.

Moreover, if the AS-wide MMS is responsible for routing advertisement, prefixes can be coalesced and supernetted for *both* the RIB and the FIB. Since routers are no longer advertising information, they can compress their RIBs through the same mechanism that FIBs are compressed. Attributes can be stripped (except for prefix and next-hop information), as the MMS would retain an original copy. For example, information such as AS path and community attributes can be removed, both of which have the potential to consume significant amounts of memory relative to the other attributes.

V. EVALUATION

Data used to evaluate the MMS comes from a tier-1 ISP's BGP feeds from January of 2008 to June of 2008. These feeds are live traces, containing failures, configuration changes, and other network events, including effects from both inter- and intra-domain events. These feeds are input into our implementation of the MMS. In order to evaluate how other ASes might benefit from the MMS, and in order to perform a longitudinal study on how compression results may change over time, a public feed from RouteViews [22] is also used from 2002 to 2008. Unfortunately, because the public feed is from a single aggregation point, the data does not indicate network topology or router configuration in the considered ASes. ISPs today are often (understandably) unwilling to share such information due to privacy issues. Hence, we attempt to infer such information from the Route Views feed. We do this by applying the Gao-Rexford rules [21] to compute routes to each prefix from each AS (treating each AS as if it were a single router). While this inference greatly oversimplifies how actual networks operate, the method represents a "best effort" attempt to evaluate the MMS on alternate Autonomous Systems, given the limited source of data. Unless otherwise mentioned, results are for FIB compression, under local-mode, and with threshold 7.

A. Compression Ratio

Table I shows compression achieved across a router within the ISP. Here, the compression techniques were run over a routing table snapshot that was collected on June 1, 2008. The compression gains of this routing table snapshot were compared with other routers in the network; no significance difference was seen. Moreover, the router's compression gains were studied over a two month period from April 15, 2008 to

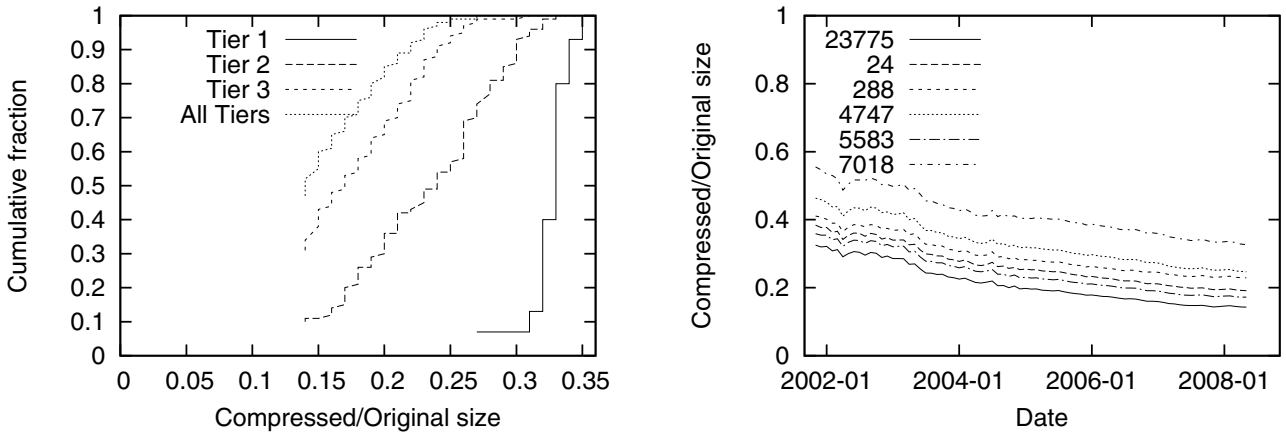

Fig. 6. *Internet simulation results:* AS relationships are inferred, routes are propagated, and compression ratios are calculated, with flexibility level set to "local-preference." Figure (a) shows compression ratios across fifty randomly sampled ASes from each tier, and Figure (b) shows compression ratios over time for a subset of ASes.

June 15, 2008; no significant variance was seen. As such, the data from Table I can be considered representative. With Level 7 compression, routing table size is 42% of its original size, without causing any changes in forwarding behavior. Lower levels increase compression. Finally, additional compression may extend lifetimes of deployed routers, by reducing the need to deploy new hardware to meet increasing table sizes. To study this, we measured the rate of growth of an uncompressed router's FIB, and compared that to rate of growth of the router's FIB if it were compressed in local-mode. We repeated this over all routers in the ISP, and found that router lifetimes could be extended by over 7 years on average, assuming the current rate of growth continues. We repeated this study on the AS-level data and found a similar amount of lifetime extension.

Figure 6a shows a CDF of the compression ratio (compressed size divided by uncompressed size) achieved for each of the 28335 ASes present in the Route Views snapshot collected on June 1, 2008. The algorithm given in [23] is used to classify ISPs into tiers within the AS hierarchy. In this simulation, the MMS reduces routing table size of most tier 1 ISPs to roughly 35% of their original size. Interestingly, lower tier ISPs achieve a greater benefit with our approach, for example 90% of tier 2 ISPs (the customers of tier 1 ISPs) achieve a compression ratio of 25% or better, with 50% of ISPs attaining a compressed routing table only 17% of the original size.

Figure 6b shows the compression ratio for a representative subset of ISPs, sampled monthly from November 1, 2001, to June 1, 2008. The compression ratio for ISPs steadily improves over time. One possible explanation is that ISPs increasingly employ deaggregation to simplify multihoming and to improve load balancing. Route Views traces indicate that the number of more-specific prefixes is increasing at a faster rate than less-specific prefixes, as shown in Figure 7a. For example, between November 1 2001 and June 1 2008, the number of /8s in routing tables did not significantly change, the number of /16s increased by 42%, and the number of /24s increased by 127%.

The variability in routing table size over time was also studied. Router table sizes associated with the tier-1 ISP were studied over a period from May 1, 2008 to June 7, 2008. Both compressed and uncompressed versions of the routing table were analyzed. Overall, route table compression reduced table size variability by an average factor of 2.6. Figure 7b shows table size, sampled after every update, for a 2.5 hour window containing a sudden increase in table size. Compression reduces magnitude of the spike by a factor of 2.1.

B. Runtime

Next, we evaluated the run time behavior of our MMS implementation, executing on a 2004-era Pentium IV 3.6GHz processor with 1GB RAM. Figure 8a shows the speed up results from parallelization. In this experiment, a single threaded version of ORTC was run, and timing information was recorded for processing each node. These results were fed to a simulator that simulated a multithreaded version of the ORTC algorithm. For simplicity, we assumed that the underlying parallelization of the hardware was equal to the number of threads that was run. The simulator used the results from the single threaded run to estimate the time that each thread spends when it processes a node. Based on these results, significant speed up can occur. However, after about 20 threads, speed up becomes negligible. It is important to note that a speedup of up to a factor of 8 (with approximately 20 threads) is significant (current commodity CPUs commonly have 4 cores, with projected doubling every 18 months [18]).

As the number of threads is increased past 20, speedup worsens, due to threading overheads (e.g., locking).

Figure 8b demonstrates the benefits from incremental computation, *i.e.*, only recomputing the portion of the routing table that is affected by a received update. The figure shows a time-series plot of update processing time, for both the incremental algorithm and the traditional non-incremental ORTC algorithm. The incremental computation significantly improves update processing time, both in terms of absolute magnitude, as well as in terms of absolute variance. For example, over

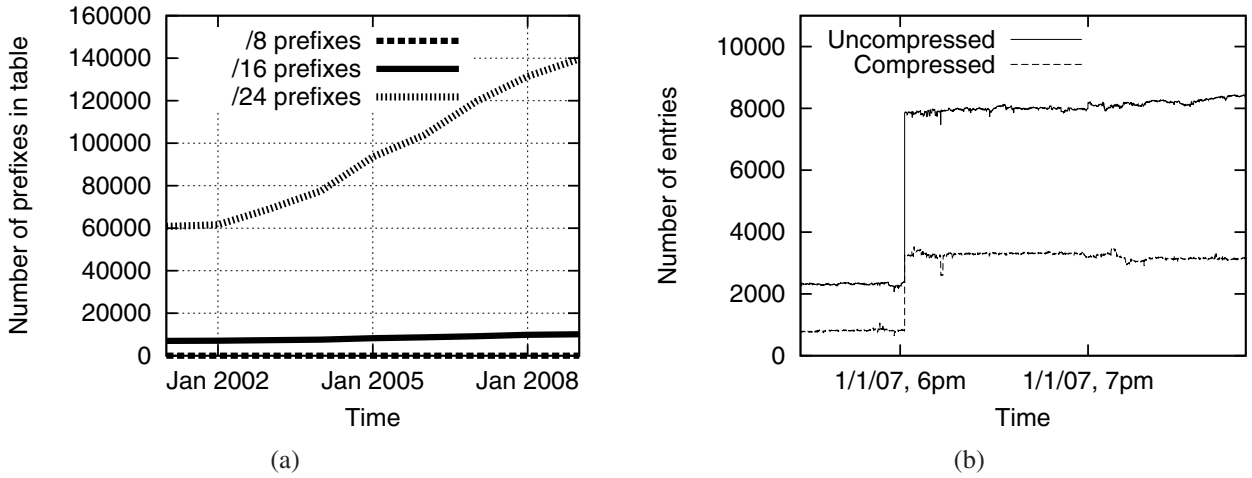

Fig. 7. (a) Number of prefixes making up Internet routing tables, as observed from Route Views. Increasing deaggregation leads to larger numbers of more specific prefixes; (b) compression savings over time during spike in routing table size.

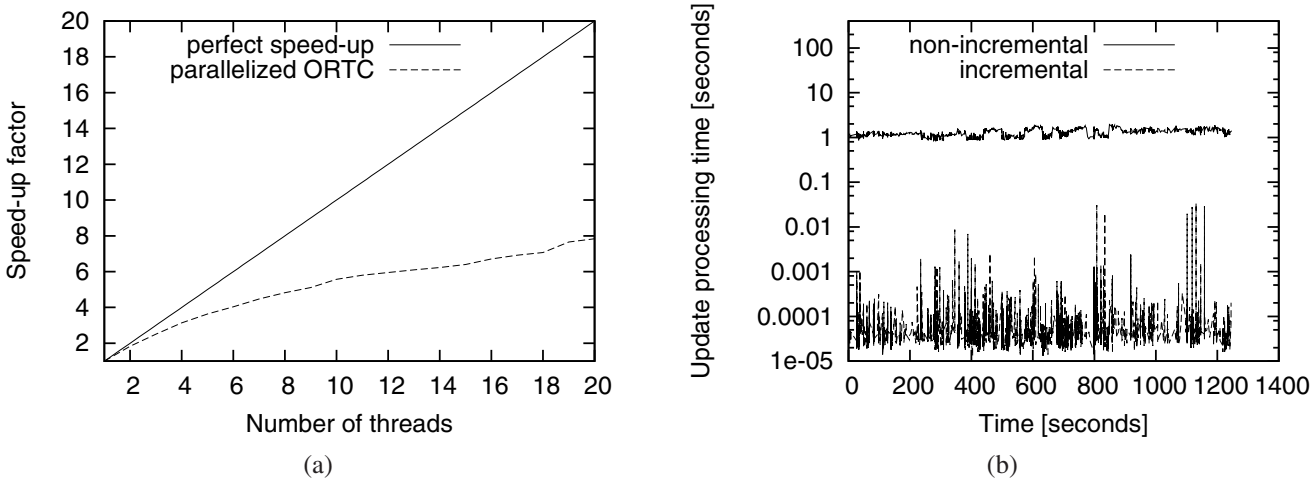

Fig. 8. (a) Parallelization and its effects on speed up; (b) overall run time with incremental computation. The vast majority of updates are processed in less than a millisecond.

the one month period from June 15, 2008, to July 15, 2008, the computation time decreased by a factor of 9344 on average, and the standard deviation in update computation time decreased by a factor of 194. The incremental computation time is a function of the portion of the trie affected by the update, and queuing (as several updates arriving close together in time have to be processed sequentially, as the parallelization optimization was disabled for this test). The trace shown in the figure had a maximum incremental computation time of 33ms (standard deviation of 1.29ms). In practice, overheads greater than update exchange periods (which can be several seconds, for example in the event of a session failure) are not visible, as computation is done at the same time as update exchange.

Lastly, Figure 9 demonstrates how the AS-wide deployed MMS can save additional computational resources through amortization. The MMS can leverage this by performing compression-related computations once for parts of the routing table that are common across routers. The figure shows a CDF, over all 40 border routers in the tier-1 ISP network, of the relative speedup gained by amortization as compared to running ORTC once per router. On average, this simulation indicates that amortization reduces computation time by a factor of 12 on average. Overall, it appears that the run time to compute *all* 40 FIBs in the network was only three times larger than running ORTC for a *single* FIB.

VI. Related Work

Improving network scalability by reducing router memory usage has been widely studied in previous work. Hierarchical routing [14], [24], landmark routing [25], [26], and geographic routing [27] embed topological information in addresses, so as to reduce the number of routes required to be stored at routers. Alternatively, DHTs [28] work by reducing the number of routes maintained by each participant in the system. These techniques are more formally studied by *compact routing* [29], which provides theoretical bounds on the amount of memory that can be saved for a given degree of suboptimal routing. Commonly, such work focuses on minimizing routing table size or control overhead while bounding path inflation. The MMS architecture differs from previous work in these areas, in that it aims to operate within the confines of existing IP routing protocols, rather than replacing them.

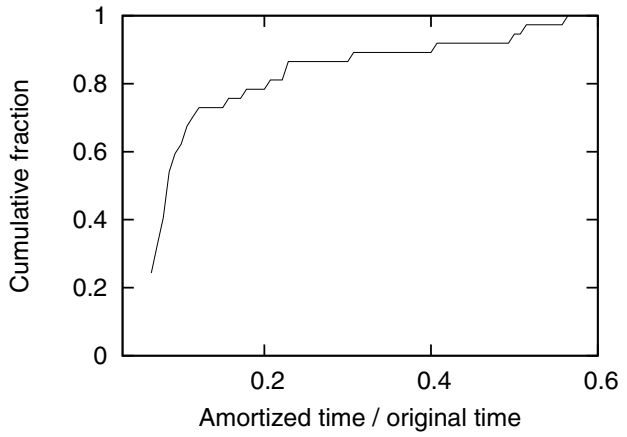

Fig. 9. Cumulative fraction of routers with their ratios of amortized time compared to unamortized time.

One way to reduce memory usage is to use MPLS tunneling in an ISP's core, while deploying route reflectors to exchange routes amongst the edge routers, to construct a "BGP-free core". While this helps the memory requirements for both border routers and ISP's cores, it does require the deployment of route reflectors. As previously mentioned in Section II, route reflectors have their own set of problems and limitations. Alternatively, instead of storing an entire routing table itself, a router may instead use *default routes* to forward traffic to another router that contains the routes. Unfortunately, default routes often require manual effort to construct, and can lead to unexpected behavior during link failures. Moreover, default routes can only be used for routes that can be statically pinned to a certain egress, limiting their applicability in non-stub ISP networks.

There has been other recent research in reducing memory consumption while remaining backwards compatible. The ViAggre [30] work demonstrates how routers can be reconfigured to store a smaller subset of the routing table. ViAggre works by aggregating prefixes into super-prefixes and assigning routers to be responsible for certain super-prefixes. By adding indirection, the BGP table can be effectively split between a set of routers. As compared to our work, Viaggre achieves its memory reduction from a different source, and is hence complementary and orthogonal to our work. Viaggre suffers from additional stretch (increasing worst-case stretch by a factor of four to achieve the 60% gain we acquire in this work), and requires modification to the way routing protocols are configured to operate correctly. Moreover, while ViAggre requires the reconfiguration of multiple routers in a network, the MMS (in local mode) can be deployed on as little as a single router and still provide memory savings. Since our techniques are complementary, they may be applied to a ViAggre router to further increase memory savings. More recently, SMALTA introduced a compression algorithm for FIBs [31]. However, SMALTA does not perform optimization in an AS-wide fashion, and modifies different parts of the FIB tree. That said, it may be possible to incorporate SMALTA's FIB compression algorithm into our MMS (or incorporate the MMS's FIB compression algorithm into SMALTA) if desired.

Another piece of work known as Route Relaying [32] demonstrated a similar technique in the VPN setting, where edge routers forward to traffic to a collection of "hub" nodes that store the full routing table. However, it is worth noting that such deflection techniques can interfere with traffic engineering. In contrast, the MMS can be configured to use IGP weights in the decision process, which are typically used for traffic engineering.

The Routing Research Group (RRG) has also explored the scalability issue with respect to memory [33]. In particular, the work on Locator / ID splitting (LISP) has gathered attention, where the IP address space is divided into separate spaces for end-hosts and for organizations. Substantial memory savings are possible under this scheme [34]. However, this scheme has a deployment problem. A single ISP cannot deploy it and realize the savings unless other ISPs cooperate. This is because LISP uses IP-in-IP tunneling that requires encapsulator and decapsulator routers positioned in each ISP. While it may be considered "incrementally deployable" from the perspective that it builds on top of existing infrastructure, it does require some coordination between ASes. As such, we consider this work complementary to the MMS.

There has also been work on several technologies that enable the MMS design. The Routing Control Platform (RCP), NOX, and 4D [10], [12], [11], [35] provided an architecture for logically centralizing route selection within an RCP. The prototypes in [10], [35], [36] demonstrated that this architecture can scale to the size of a tier-1 ISP backbone, and deal with failure and consistency issues when operating at scale. The RCP aimed to compute and distribute BGP routes to routers within an ISP, and did not aim to reduce table sizes at routers. However, the MMS algorithms may be deployed on top of an RCP-like infrastructure.

Other related work includes Verkaik *et. al.*'s BGP Atoms [37], Forgetful Routing [38], and Draves *et. al.*'s Optimal Routing Table Constructor [8]. BGP Atoms can be used to reduce memory overhead by clustering prefixes based on policy, rather than supernets. Forgetful Routing enables routers to share their RIBs in a distributed fashion, reducing redundancy in a network. The work by Draves *et. al.* served as a primary inspiration for our work. The algorithmic contributions, architecture, and deployment strategies used in the MMS can be viewed as a way to make ORTC practical in a modern-day network environment. Our work also measures compression benefits over modern workloads and a range of of topologies and environments, including a tier-1 ISP network.

VII. Conclusions

Deploying an MMS within an ISP has several benefits. An MMS can prevent router memory requirements from exceeding capacity, as well as extend the lifetime of routers. Moreover, experimental results show substantial reduction of routers' FIBs. Reducing these requirements and safely preventing routers from becoming overloaded reduces the need to upgrade them as often, decreasing operational costs and administrative work. The MMS is designed to be practical and also amenable to partial deployment.

For future work, several items may be interesting to investigate. While the threshold levels are assumed to be fairly

static, a fully-automated "adaptive mode" could be developed; the algorithm would automatically adjust the threshold level to stay within memory bounds while deviating from the BGP decision process as little as possible. Additional savings might be possible by developing protocols to perform memory management across ISPs. Finally, if memory is still scarce after compression, the memory management system could be used to selectively filter less popular routes to ensure that the most popular ones remain available.

REFERENCES

[1] T. Bu, L. Gao, and D. Towsley, "On characterizing BGP routing table growth," *Computer Networks*, vol. 45, pp. 45–54, May 2004.

[2] P. Smith, R. Evans, and M. Hughes, "RIPE routing working group recommendations on route aggregation," http://www.ripe.net/ripe/docs/ripe-399.html, Dec. 2006.

[3] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. 2008 IEEE INFOCOM*.

[4] Q. Dong, S. Banerjee, J. Wang, D. Agrawal, and A. Shukla, "Packet classifiers in ternary CAMs can be smaller," in *Proc. 2006 ACM SIGMETRICS*.

[5] Q. Dong, S. Banerjee, J. Wang, and D. Agrawal, "Wire speed packet classification without tcams: a few more registers (and a bit of logic) are enough," in *Proc. 2007 ACM SIGMETRICS*.

[6] P. Gupta, "Address lookup and classification," course lecture, May 2006, www.stanford.edu/class/ee384y/Handouts/lookup-and-classification-lec2.ppt.

[7] D.-F. Chang, R. Govindan, and J. Heidemann, "An empirical study of router response to large BGP routing table load," in *Proc. 2002 Internet Measurement Workshop*.

[8] R. Draves, C. King, S. Venkatachary, and B. Zill, "Constructing optimal IP routing tables," in *Proc. 1999 IEEE INFOCOM*.

[9] "The BGP instability report," http://bgpupdates.potaroo.net/instability/bgpupd.html, Aug. 2009.

[10] 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. 2005 NSDI*.

[11] A. Greenberg, G. Hjalmtysson, D. Maltz, A. Meyers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4D approach to network control and management," *ACM Computer Commun. Rev.*, Oct. 2005.

[12] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," *ACM Computer Commun. Rev.*, July 2008.

[13] "BGP reports," http://bgp.potaroo.net.

[14] J. Moy, *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.

[15] D. Oran, "OSI IS-IS Intra-domain routing protocol," RFC 1142, Feb. 1990.

[16] T. Bates, R. Chandra, and E. Chen, "BGP Route Reflection—an alternative to full mesh IBGP," Request for Comments 2796, Apr. 2000.

[17] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong, "Route oscillations in I-BGP with route reflection," in *Proc. 2002 SIGCOMM*, pp. 235–247.

[18] K. Asanovic, R. Bodic, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams, and K. Yelick, "The landscape of parallel computing research: a view from Berkeley," in University of California, Berkeley, Technical Report No. UCB/EECS-2006-183, December 2006.

[19] M. Quinn and N. Deo, "Parallel graph algorithms," *ACM Computing Surveys*, Sep. 1984.

[20] M. Caesar and J. Rexford, "BGP routing policies in ISP networks," in *IEEE Network Mag.*, Nov. 2005.

[21] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. Networking*, Dec. 2001.

[22] "Route views project page," http://www.routeviews.org/.

[23] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. 2002 IEEE INFOCOM*.

[24] Internet Engineering Task Force, "Internet protocol," RFC 791, Sep. 1981.

[25] P. Tsuchiya, "The landmark hierarchy: a new hierarchy for routing in very large networks," in *Proc. 2006 ACM SIGCOMM*.

[26] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica, "Beacon vector routing: scalable point-to-point in wireless sensornets," in *Proc. 2004 NSDI*.

[27] H.-T. Kung and B. Karp, "Greedy perimeter stateless routing for wireless networks," in *Proc. 2000 ACM Conference on Mobile Computing and Networking*.

[28] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for Internet applications," in *Proc. 2001 ACM SIGCOMM*.

[29] M. Thorup and U. Zwick, "Compact routing schemes," in *Proc. 2001 ACM Symposium on Parallel Algorithms and Architectures*.

[30] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making routers last longer with ViAggre," in *Proc. 2009 NSDI*.

[31] Z. Uzmi, M. Nebel, A. Tariq, S. Jawad, R. Chen, A. Shaikh, J. Wang, and P. Francis, "SMALTA: practical and near-optimal FIB aggregation," in *2011 CoNEXT*.

[32] C. Kim, A. Gerber, C. Lund, D. Pei, and S. Sen, "Scalable VPN routing via relaying," in *Proc. 2008 ACM SIGMETRICS*.

[33] "Routing research group (RRG)," http://tools.ietf.org/group/irtf/trac/wiki/RoutingResearchGroup.

[34] B. Quoitin, L. Iannone, C. de Launois, and O. Bonaventure, "Evaluating the benefits of the locator/identifier separation," in *Proc. 2007 ACM/IEEE International Workshop on Mobility in the Evolving Internet Architecture*.

[35] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. Snoeren, and J. Van der Merwe, "Wresting control from BGP: scalable fine-grained route control," in *Proc. 2007 USENIX Annual Technical Conference*.

[36] Y. Wang, I. Avramopoulos, and J. Rexford, "Design for configurability: rethinking interdomain routing policies from the ground up," in *IEEE J. Sel. Areas Commun.*, 2009.

[37] P. Verkaik, A. Broido, and K. C. Claffy, "Beyond CIDR aggregation," Tech. Rep., Nov. 2004, http://www.caida.org/outreach/papers/2004/tr-2004-01/tr-2004-01.pdf.

[38] E. Karpilovsky and J. Rexford, "Using forgetful routing to control BGP table size," in *Proc. 2006 ACM International Conference on emerging Networking EXperiments and Technologies*.

Elliott Karpilovsky completed his B.S. in Computer Science from the California Institute of Technology. He completed his Ph.D. in Computer Science from Princeton University in 2009, where he worked with Prof. Jennifer Rexford. He recently joined Google, Inc., in Mountain View.

Matthew Caesar is an assistant professor in the Computer Science department at the University of Illinois at Urbana-Champaign. His research involves simplifying the management of distributed systems and networks through principles of self-organization and self-diagnosis, with an emphasis on Internet architecture. As part of his six-year collaborations with AT&T Labs, his work on the Routing Control Platform has been deployed and is in daily use on their North American IP backbone. He received the NSF CAREER award in 2011 for his work in Internet debugging.

Jennifer Rexford joined the Network Systems Group of the Computer Science Department at Princeton University in February 2005 after eight and a half years at AT&T Research. She received her BSE degree in electrical engineering from Princeton University in 1991, and her MSE and Ph.D. degrees in electrical engineering and computer science from the University of Michigan in 1993 and 1996, respectively. She was the winner of ACM's Grace Murray Hopper Award for outstanding young computer professional of the year for 2004.

Aman Shaikh is a Principal Member of Technical Staff (PMTS) at AT&T Labs (Research) in New Jersey, USA. His current research interest lies in network management and operations. Aman Shaikh obtained his B.E. in Computer Science and M.Sc. in Mathematics from the BITS, Pilani (India) in August 1998; and M.S. and Ph.D. in Computer Engineering from the University of California, Santa Cruz, CA (USA) in June 2000 and December 2003 respectively. His Ph.D. advisor was Prof. Anujan Varma. For his Ph.D. dissertation, he worked on how to improve management related aspects of the routing infrastructure in IP networks.

Jacobus van der Merwe is the Jay Lepreau Professor in the School of Computing at the University of Utah and a director in the Flux Research Group. He joined the University of Utah in August 2012 after more than fourteen years at AT&T Labs - Research. He received a Ph.D in 1998 from the Computer Laboratory at the University of Cambridge and B.Eng and M.Eng degrees in electronic engineering from the University of Pretoria in South Africa in 1989 and 1991 respectively. He received the AT&T Science and Technology Medal in 2010 for his work on Intelligent Route Control.