International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 22 (2015) pp 42839-42845

Compression and Construction of Optimal IP Routing Using Optimal Route Table Construction Algorithm

Rupa Madhuri Pattanaik

College of Engineering and Technology, Ghatikia, Bhubaneswar rupamadhuri.mr@gmail.com

J. Chandrakanta Badajena

College of Engineering and Technology, Ghatikia, Bhubaneswar chand.cet@gmail.com

Chinmayee Rout

Ajay Binay Institute of Technology, Sector-1, CDA, Cuttack chinu123.abit@gmail.com

Shatabdinalini

College of Engineering and Technology, Ghatikia, Bhubaneswar Shatabdi.ira88@gmail.com

Abstract

Generally Internet routers stores paths to destinations. This information is stored in memory which is expensive, and it requires to be updated continuously as internet use is increasing day by day. But the existing routing protocols are not efficiently handling the situation when memory becomes full and increasing memory due to increasing routing table size. The objective of this paper is to primarily focus on improving ORTC algorithm which will optimize the routing table in an efficient way that will reduce number of prefixes present in the older table without affecting the criteria.

For this we have taken publicly available routing data from internet and evaluate through our algorithm to produce optimized output.

Keywords-RIB, FIB, ORTC, MMS

Introduction

As the Internet grows to all every corner of the world, the demands on the Internet backbone routers keep increasing. One of the major problems facing the backbone routers today is the increasing number of routing entries or prefixes that they have to handle. The number of routes in the Internet backbone has been growing by 10,000 per year for the last several years. 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 disaggregation(advertising more-specific routes) arising from load balancing and security concerns, 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

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. To protect against this, network operators are forced to repeatedly upgrade their equipment at enormous expense due to the large cost of doing an infield 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.

Optimal Route Table Construction (ORTC) is most vital algorithm which helps to find a optimal routing table. 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 takes approximately 500 milliseconds 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, 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.

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. 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. 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.

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.

Routing across ISPs

The Internet is composed of a collection of *Autonomous Systems (ASs)*, 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 ASs 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 ASs. Paths are newly advertised or withdrawn by exchanging update messages containing reachability information. 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 1. 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 disaggregated 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.

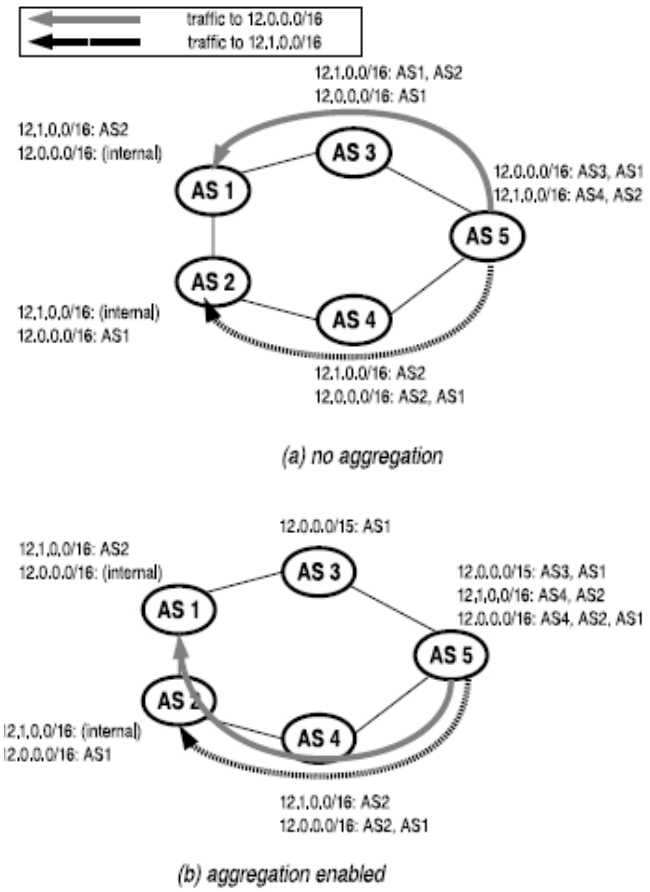

Fig.1 Aggregation can have unintended consequences (a) Suppose AS 1 originates 12.0.0.0/16 and AS 2 originates 12.1.0.0/16. When no ASes perform aggregation, AS 5 can route traffic to 12.1.0.0/16 via AS 3, and traffic to 12.0.0.0/16 to AS 4. (b) However, if AS 3 decides to aggregate 12.1.0.0/16 and 12.0.0.0/16 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.

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 or IS-IS 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 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. They require additional work for 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).

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.

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.

Routing table compression

Figure 2 shows the general process steps involved in compressing a routing table. At step 50, the compression program 48 converts the routing table 30 into a binary tree structure that represents the address prefixes in the routing table. The tree is stored in volatile memory 44 during the compression process.

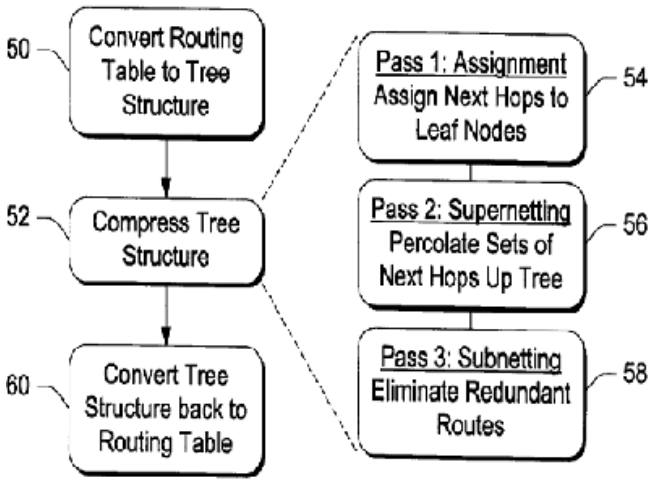

Fig.2 Flow diagram showing steps for compressing routing table

Initial work on routing for large networks established that hierarchical routing produces routing tables logarithmic in the number of network hosts with negligible increase in message path lengths. This is important for achieving scalability as network sizes increase. Internet routing today takes advantage of this principle. Internet address lookup would be simple if we could lookup a 32-bit IP destination address in a table that lists the output link for each assigned Internet address. However, each router would have to keep an entry for every Internet host millions of entries. To reduce database size and routing update trafic, an Internet router database consists of a much smaller set of *prfixes*. This reduces database size, but at the cost of requiring a more complex lookup called *longest matching prefix*. Each prefix P has an associated *next hop* or *output link* information, which specifies where a packet is to be forwarded if its longest matching prefix is P. We will write prefixes as bit strings of up to 32 bits followed by a `*'. For example, the prefix 01* matches any address that begins with the bits 01. The prefix * matches every address. Thus if the destination address begins with 01000 and we had only two prefix entries (01*→ 1; 0100* → 2), the longest-matching prefix would be 0100* a the packet would be directed to nexthop 2.

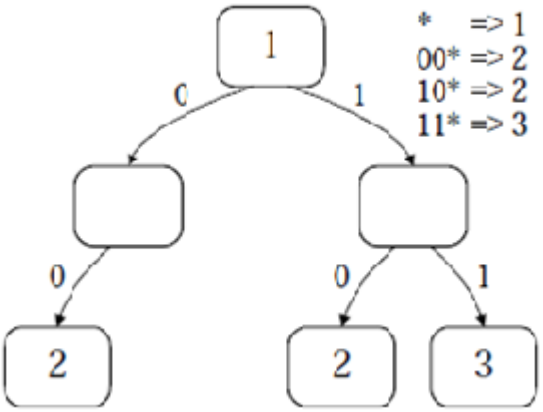

Fig.3 Binary tree Representation of a binary table

The Internet initially used a simple hierarchy in which 32-bit addresses were divided into a network prefix and a host number, so that routers would only store entries for networks. For flexible address allocation, the network prefixes came in three sizes: Class A (8 bits), Class B (16 bits), and Class C (24 bits). Organizations that required more than 256 hosts were given class A or B addresses; these organizations could further structure their addresses for internal routing with *subnetting.*

For example, if P1=00* is a network, then P2=0011* is a *subnet* under P1. However, the Class A and B spaces did not scale to handle the Internet's growth. This led to the invention of Classless Inter-Domain Routing (CIDR). CIDR can give organizations multiple contiguous network prefixes that can still be aggregated by a common prefix, which reduces backbone router table size. *Supernetting* denotes the aggregation of adjacent prefixes that have the same next-hop information.

Constructing optimal routing table

The algorithm for reducing number of entries in routing table generalizes the concept of subnetting and supernetting. Here we have represented an initiative explanation of the algorithm operation. To simplify the explanation, the initial description relies on having a single next hop for a prefix and having a default route for the null prefix. I remove these restrictions in the final subsection.

To graphically depict a set of prefixes, we use a binary tree representation. Each successive bit in a prefix corresponds to a link to a child node in the tree, with a 0 corresponding to the left child and a 1 corresponding to the right child. Note that the binary tree generally contains more nodes than there are prefixes, since every successive bit in the prefix produces a node. We labeled nodes with next-hop information, typically a small integer or a set of small integers.

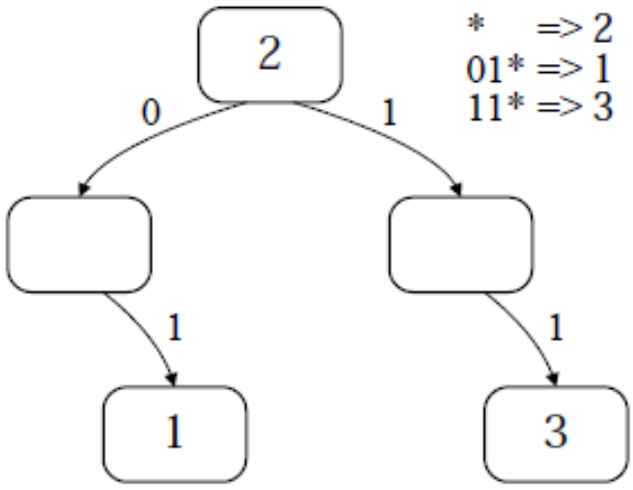

Fig.4 Example of routing table after ORTC

Figure 3 and 4 shows an example with four routes. For instance, the root node in the tree represents the null prefix, with a default route to next-hop 1. The lower left tree node

represents the prefix 00*, and the next hop associated with this prefix is 2. Figure 6.1 shows the output of the algorithm on this example. By changing the default route at the root of the tree from 1 to 2, ORTC reduces the number of routes from four to three. Note that the optimized routing table encodes forwarding behavior equivalent to the original routing table. Using the longest matching prefix algorithm, both routing tables forward 00 to next-hop 2, 01 to next-hop 1, 10 to next-hop 2, and 11 to next-hop 3.

In Figure 3, next-hop 2 is most prevalent in the original routing table: it accounts for half of the possible destinations. Hence ORTC in its simplest form, ORTC optimizes a routing table using three passes over the binary tree representation. (The next subsection describes optimizations to the basic algorithm, including combining the first two passes.) The first pass propagates routing information down to the tree's leaves. The second pass finds the most prevalent next hops, by percolating information (sets of next hops) from the leaves back up towards the root. Finally, a third pass moves down the tree, choosing a next hop from the set of possibilities for a prefix and eliminating redundant routes. This description of the algorithm makes two simplifying assumptions about routing tables, which do not hold for real backbone routing tables. First, we assume that every routing table has a default route, or equivalently, that the null prefix at the root of the tree has a next hop. Second, we assume produces a smaller routing table by moving it to the root of the tree and using longer prefixes to represent routes to next-hops 1 and 3.

Pass One

The first pass “normalizes" the binary tree representation of the routing table, in preparation for the second and third passes. It enlarges the tree so that every node has either zero or two children. It does this by creating new leaf nodes and initializing the next hop for a new node with the next hop that the new node inherits from its nearest ancestor that has a next hop. Once the tree is fully populated with leaf nodes, the next-hop information for interior nodes is no longer needed and may be discarded. In preparation for the second pass, which uses sets of next hops, the first pass converts the next hop for each prefix to a singleton set.

Pass Two

The second pass calculates the most prevalent next hops at every level of the routing table by percolating sets of next hops up the tree. An implementation of the second pass could use a post-order traversal of the tree or a traversal by levels from the bottom up towards the root. At each parent node visited in the traversal, a set of next hops is calculated.

Pass Three

The third pass moves down the tree selecting next hops for prefixes and eliminating redundant routes via subnetting. An implementation could use either a pre-order traversal of the tree or a traversal by levels from the root down. Each node visited will have a set of possible next hops, computed in the second pass. Except for the root node, the node will inherit a next-hop from the closest ancestor node that has a next hop. If this inherited next hop is a member of the node's set of potential next hops, then the node does not need a next hop of its own: it is inheriting an appropriate next hop. However, if the inherited next hop is not a member of the node's set of potential next hops, then the node does need a next hop. Any member of the node's set of potential next hops may be chosen as the node's next hop.

The pseudo-code algorithm operates on a binary tree. The symbol N denotes a node in the tree. *nexthops*(*N*) denotes a set of next hops associated with the node N. If the routing table does not assign next hops to N, then *nexthops*(*N*) is defined to be the empty set ∅ ;. We assume *nexthops*(*root*) *!*=∅ ;. For nodes with children, *left*(*N*) and *right*(*N*) denote the left and right child nodes. Similarly, we define *parent*(*N*) for all nodes except the root. The operation *choose*(*A*) picks an element from the non-empty set *A*.

A#B operation on two sets of next hops can be defined as:

$$A\#B = \begin{cases} A \cap B & \text{if } A \cap B \neq \emptyset \\ A \cup B & \text{if } A \cap B = \emptyset \end{cases}$$

The function *inherited*(*N*) on nodes other than the root is defined as:

$$inherited(N) = \begin{cases} nexthops(parent(N)) & \text{if} \neq \emptyset \\ inherited(parent(N)) & \text{otherwise} \end{cases}$$

The first and third passes perform a traversal from the tree's root down to its leaves. This can be either a pre-order traversal or a traversal by levels. Similarly, the second pass performs a traversal from the leaves up to the root, using either a post-order traversal or a traversal by levels.

Pass One.

for each node *N* (root to leaves) {
if *N* has exactly one child node,
create the missing child node
if *nexthops*(*N*) = ∅;
nexthops(*N*) ← *inherited*(*N*)
}

Pass Two.

for each node *N* (leaves to root) {
if *N* is a parent node,
nexthops(*N*) ←
nexthops(*left*(*N*))#*nexthops*(*right*(*N*))
}

Pass Three.

for each node *N* (root to leaves) {
if *N* is not the root and
inherited(*N*) ∈ *nexthops*(*N*)
nexthops(*N*) ←∅ ;
else
nexthops(*N*) ← *choose*(*nexthops*(*N*));
}

Algorithm

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**
// 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 element in **N.prev set**

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. Now let us check how it is possible to further improve the compression results by allowing the MMS to modify the forwarding behavior. The BGP decision process 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. The router

(1) first chooses the routes with the highest LocalPref (a numericvalue assigned by the operator to indicate which next-hops are most preferred),
(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),
(3) the routes with the lowest origin type (a flag indicating whether the route originated internally or externally to the ISP),
(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),
(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),
(6) the router chooses the closest exit point (or shortest internal route) to reach the destination prefix,
(7) to break ties, if multipleoptions 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.

Network Simulation

In ORTC algorithm, if root node makes a optimized route for a particuler router and its interfaced network then the packets to that particuler router will go through a specific route only which is somewhat similar to a dedicate path, hence there will be no packet fragments to that router from other neigbhouring routers for a prticuler transaction.This indicates due to selection of specific paths there is less interaction between other links

When a new router is added to the network, it has to be configured by specific ports and routing rules according to the networking. The figure 13 is a overview of these configuration packets forwarded without ORTC algorithm.

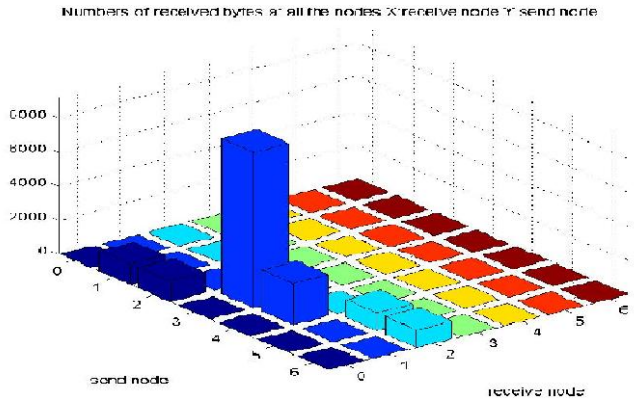

Fig.5 due to selection of specific paths there is less interaction between other links.

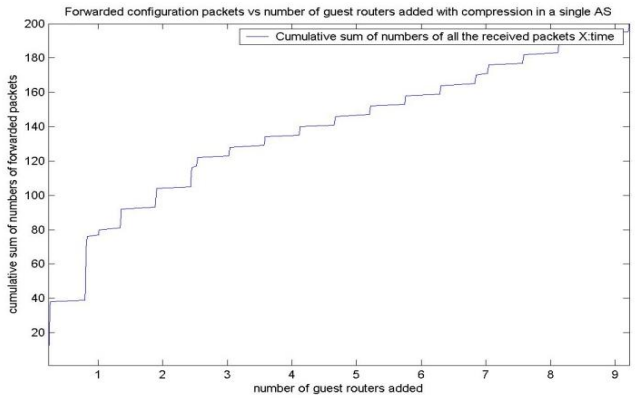

Fig.6 Forwarded packets vs Number of Routers added without compression

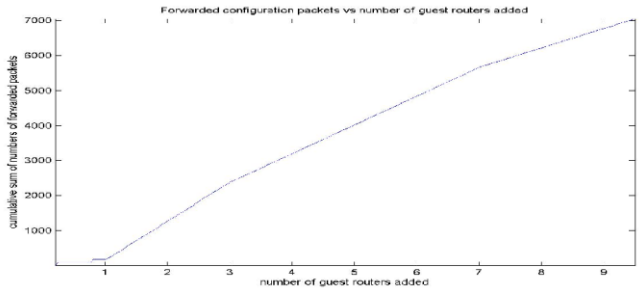

Fig.7 Forwarded packets vs Number of routers added with compression

Conclusion

The compression techniques are described above in the context of compressing routing tables in routers. More generally, the same techniques may be applied to optimally reduce the size of a set of prefixes, where the prefixes are being selected according to the longest matching prefix algorithm. 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.

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.