Address Summarization for the Extended Bellman-Ford Protocol

Erik van Zijst
erik@marketxs.com
MarketXS
September 19th 2003

Abstract

This text describes the application of address summarization to the Extended Bellman-Ford distance vector protocol[i], which is a descendant of the well known distributed Bellman-Ford routing protocol[ii] (implemented initially in ARPANET[iii]). This paper is based on earlier work[iv] and describes the technique we have chosen for our experiments in greater technical detail and specifically how to deal with the head-of-path information that ExBF attaches to each individual path and is used to eliminate the bouncing effect and the counting-to-infinity problem[i,v]. Additionally, this paper presents a wire-level optimization algorithm that could be used between peers to decrease the size of the packet headers significantly by substituting the dotted address strings.

Introduction

In previous work we have introduced two hierarchical addressing schemes[iv] that allow ranges of hosts, subnets or groups of subnets to be substituted by a single wildcard address in a node's distance table or routing table. This is an essential enhancement for traditional distance vector protocols when the network is very large in size or has frequent topology changes. One of the proposed schemes uses variable length dotted strings to represent both unicast and multicast addresses, while the second is a more traditional numeric scheme with a 64 bit address space and fixed domain classes. It is a simple and somewhat limited scheme as it does not support variable subnet lengths as used in Classless Inter-Domain Routing (CIDR)[vi] for IP. For our current lab experiments we have chosen to use the more flexible, easy to understand and rather unconventional address strings instead of the more efficient but complex numeric scheme. The major drawback of using variable length dotted strings is that they require more bytes to serialize than a 32 or 64 bit numeric address. If we consider the string `com.gblx.level3.mxs.gw3` to be a typical unicast host address, it requires at least 23 bytes or 184 bits to include this address in a packet header. Note that most packets will carry at least two addresses, namely a source and a destination address. Also, with the current maximum address length of 127 bytes, that could contribute to 254 bytes of address headers, opposed to 16 bytes per packet when two 64 bit numeric addresses are used. This makes the protocol very bandwidth inefficient when the payload per packet is small. Also, routers will probably require more computing power and storage capacity to process strings instead of doing numeric operations. Nevertheless, we think modern computers are cheap and fast enough to allow for the added computational complexity but instead present network administrators with a more flexible and easy to use addressing scheme.

Head-of-Path Summarization

The Extended Bellman-Ford protocol, that is at the base of our experiments, stores the last hop to each destination as the head of the path to that destination. This head address is included in every distance vector and in each entry in a node's routing table for each neighbor. With this information it is possible to compute the complete path from every reachable destination back to the node itself by recursively using the head of path as a destination itself. This allows nodes to discover routing loops (non-simple paths) and prevent them by not advertising paths to neighbors that would lead to non-simple paths. This approach is comparable to the *split-horizon* or *poisoned-reverse*[1] technique in that they both advertise destinations as being unreachable that would otherwise lead to routing loops. The difference is that where route-poisoning can only prevent loops with only two nodes, while the head-of-path technique detects loops between any number of nodes. The problem can be shown in the network in illustration 1.

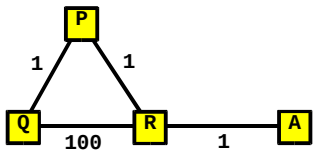

Illustration 1: Example distance vector network that uses poisoned-reverse but still suffers from a routing loop.

When run by the distributed Bellman-Ford

1 Split Horizon consists of omitting routes learned from a peer when sending updates back to that peer. With Poisoned Reverse instead of omitting those routes, they are advertised as unreachable.

protocol, Q will advertise its distance vectors to R. This includes the path to A (via P and R) with distance 3. R will have two routes to node A, one directly to A with distance 1 and one via Q with cost 103. Note that P will not advertise a route to A to R because of the poisoned-reverse mechanism. Now when the link between R and A fails, R's preferred hop for destination A changes from A to Q. This creates a routing loop as well as the counting-to-infinity problem. R advertises its new route to P (not to Q, due to poisoned-reverse). P only had one route to A (via R) and learns the new distance of 104, which it advertises to Q. Q updates its distance from A to 105 and advertises it to R, etc.

ExBF prevents the loop using the head-of-path information. When the routing tables initially converge, every node verifies its neighbor is not in the path it advertises to it. Before Q advertises destination A to R, it backtracks the path from A back to Q and discovers the presence of R. This would create a three-node routing loop and as a result Q does not advertise the route to R.

Applying address summarization on received ExBF distance vectors means also summarizing the head-of-path addresses.

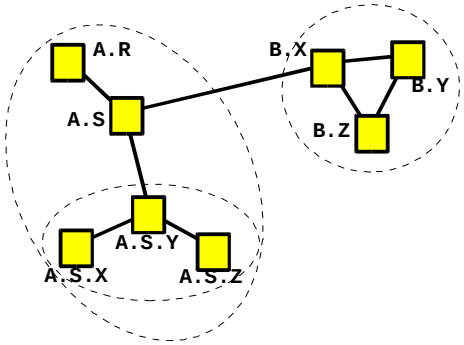

Illustration 2: Example topology to demonstrate head-of-path address summarization in ExBF. We assume all links to have a cost 1.

We use the example network topology as depicted in illustration 2 to demonstrate the exchange of ExBF distance vectors and creation of summarized distance tables as well as the preemptive detection of routing loops. When the network first starts to converge, B.X advertises the following routing information to neighbor A.S: $DV_{B.X,\ A.S}$: {(B.X, *, 0), (B.Y, B.X, 1), (B.Z, B.X, 1)}[2]. When A.S receives the vectors it summarizes the entries. What remains is the single (shortest) vector $DV_{B.X,\ A.S}$: {(B.*, *, 0)}. The distance table of node A.S now contains $DT_{A.S}$: {(A.S, *, *, 0), (A.R, A.R, A.S, 1), (B.*, B.*, A.S, 1)}. Note that all addresses are summarized before processing. This includes neighbor addresses. The consequence of this is that when a node has more than one connection with a foreign domain, both neighbor addresses will be summarized into the same wildcard. This is a known limitation of the current architecture that assigns addresses on a per node basis, rather than per interface and also occurs when configuring the same neighbor twice without summarization. The routing update A.S sends to A.S.Y contains $DV_{A.S,\ A.S.Y}$: {(A.S, *, 0), (A.R, A.S, 1), (B.*, A.S, 1)} and leads to A.S.Y's distance table $DT_{A.S.Y}$: {(A.S.Y, *, *, 0), (A.S.X, A.S.X, A.S.Y, 1), (A.S.Z, A.S.Z, A.S.Y, 1), (A.S, A.S, A.S.Y, 1), (A.R, A.S, A.S, 2), (B.*, A.S, A.S, 2)}. When A.S.Y has finished updating its distance table, it advertises $DV_{A.S.Y,\ A.S}$: {(A.S.Y, *, 0), (A.S.X, A.S.Y, 1), (A.S.Z, A.S.Y, 1), (B.*, *, *), (A.S, *, *), (A.R, *, *)} back to neighbor A.S where the * indicates unreachable in the last three vectors.

Packet-Level Address Substitution

As explained before, the use of large human readable text strings as addresses can introduce a large amount of overhead in packet headers and be wasteful for bandwidth. Favoring bandwidth efficiency over computing power efficiency, we propose a mechanism for substituting the address strings by small 16 bit values when transmitting a packet to a neighbor and keeping a translation table at both sides of the connection that is used to put the original address string back in the packet before processing it further. This optimization works on a per link basis and it totally independent from the communication with other neighbors. When it is used on more than one interface, each interface maintains its own translation table. When a packet is transmitted to a neighbor, all address strings are stripped from the packet header. For each address string, a 16 bit substitute value is generated and stored in the translation table. These 16 bits replace the original addresses and before the packet is transmitted, its protocol version field in the first header is changed to a higher version. This version number is reserved for optimized packets that would be unreadable for routers without support for address substitution. To ensure proper translation by the peer, the interface first sends a special packet that contains the new address substitutions. Such a packet is tagged with the higher protocol version and contains one or more tuples of address string and substituted value. It is

2 The following notation is used to describe a set of distance vectors exchanged by peers:
$DV_{from,\ to}$: {(destination, head-of-path, distance), ...}
When describing a distance or routing table, we use the notation:
DT_i: {(destination, hop, head, distance), ...}

important that this packet is received by the peer prior to the data packet itself, which is something that cannot be guaranteed by every transport layer.

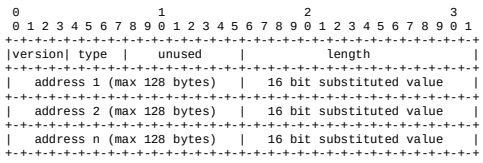

Illustration 3: Packet used to synchronize substitution information between peers. Issuing new substitutions is done with type 0x1 while invalidation of earlier or unknown substitutions is done with type 0x2.

If UDP is used for communication between peers, we must anticipate out-of-order datagram delivery and use a piggybacking technique instead. While this guarantees proper translation by the peer node, it cannot circumvent the situation where the second substituted packet arrives prior to the first packet with the piggybacked translation information. Piggybacking does have the additional advantage of efficiency however. Configuring an interface for substitution can be done manually but can also be done automatically using the special handshake packet of illustration 4.

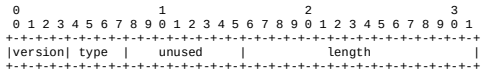

Illustration 4: Handshake packet exchanged by peers to configure their interfaces for address substitution. The version field has the higher protocol version, the type is 0x0 and the length is always 4.

A peer transmits such a packet when the connection is first established. When the packet is echoed back by the peer, the interface is configured for address substitution. When an unreliable transport layer is used such as UDP, it is possible the echoed packet is lost on its way back and the node's interface will not substitute addresses. Introducing an acknowledgment packet does not solve the problem as that packet might also be lost. Instead, we do not anticipate a lost handshake packet, but when the peer node starts transmitting translation information packets together with data packets carrying the higher protocol version, the interface will be enabled after all. After a connection with a peer has been lost, the handshake must be done again since the peer could be replaced by software that does not support substitution. Entries in the translation table each have a time out. After this period the entry is removed from the table and any packet still using the substituted value will not be processed causing the peer to send back a packet (with type 0x2) containing the 16 bit substitution values that were not recognized. Since the invalidation of timed out entries is implicit, it is essential both peers use the same time out to avoid re-use of values by one peer that still represent old addresses in the other peer. When the translation table is full, entries are best removed according to a last recently used scheme.

Conclusions

In this text we have studied the application of summarization on a location-based addressing structure for the Extended Bellman-Ford routing protocol. We have maintained the head-of-path information used for preemptive routing loop detection and demonstrated the use of wildcards. Additionally we have proposed a simple and somewhat naïve mechanism for reducing the size of the address headers in data packets. All in all this should contribute to a scalable and relatively bandwidth efficient dynamic routing protocol with a flexible and easy to understand addressing allocation structure.

i Cheng, Riley and Kumar, *A Loop-Free Extended Bellman-Ford Routing Protocol Without Bouncing Effect*, In Proceedings of the ACM SIGCOMM, pages 224-236, 1989

ii L.R. Ford and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962

iii J. McQuillan and D.C. Walden, *The ARPANET Design Decisions*, Computer Networks, vol. 1, August 1977

iv E. van Zijst, *Address Summarization for Distance Vector Protocols*, MarketXS internal paper, August 2003

v M.S. Sloman and X. Andriopoulos, *A Routing Algorithm for Interconnected Local Area Networks*, Computer Networks and ISDN Systems, pp. 109-130, 1985

vi Fuller, Li, Yu & Varadhan, *Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy*, RFC-1519, September 1993