Brief Announcement: An Incremental Algorithm for Calculation of Backup-Paths in Link-State Networks

Pierre Fransson and Lenka Carr-Motyčková
Luleå University of Technology
SE-971 87 Luleå, Sweden
{pierre, lenka}@sm.luth.se

Categories and Subject Descriptors: F.2.2: Routing and layout

General Terms: Algorithms, Performance, Reliability.

Keywords: Routing, link-state networks, backup-paths.

Link-state routing algorithms, such as OSPF (Open Shortest Path First) and IS-IS are popular and widely used intra-domain routing protocols. Link-state routing is generally seen as quite efficient at recalculating routing paths when the network topology changes due to component outages and has been able to cope well with the requirements of traditional Internet applications. However, newer applications (such as Voice-over-IP) have more stringent requirements regarding convergence times. In trying to cope with these new requirements, it can be noted that it is hard to reach sufficiently fast reaction times, i.e., ≈ 100 milliseconds, using the normal link-state mechanism; flooding of LSAs followed by recalculation of the SPT 's. A possible solution when fast reaction times are necessary, is instead to precompute backup paths that can be immediately used by routers as soon as a failure is detected. Herein we present a new algorithm for calculation of backup-paths available through directly connected neighbors, in link-state routing networks. The algorithm can be implemented in today's link-state routing protocols, e.g., Open Shortest Path First (OSPF), with a minimum of modification to such protocols.

The algorithm, hereafter referred to as the *backup-algorithm*, is based on a minimalistic strategy. Only using backup-paths that are available by forwarding packets to other directly connected neighbors and letting those neighbors forward the packets along their ordinary forwarding paths. This approach requires a minimum of modification to link-state routing protocols, since it only relies on a method of detecting a failed neighbor. Other than that there is no kind of signaling that takes place between routers. Keeping modification to a minimum is an important issue when adding functionality to existing routing protocols, since the amount of modification needed to a large extent determines

how easy it will be to deploy the functionality into existing networks.

The backup-algorithm is a derivative of the incremental algorithm and like it, it requires that a shortest path tree (SPT_{init}) has already been produced. In order to ensure that it is possible for source s to reroute a packet destined for destination u , via neighbor m , when neighbor n (on the ordinary shortest path) has failed, the incremental Dijkstra algorithm is augmented/modified in a couple of aspects so that the failed neighbor n is guaranteed to be circumvented. Because the failed neighbor is avoided, it is possible to compute the minimum cost backup-paths that are available by simply forwarding packets to directly connected neighbors.

The most important change compared to the ordinary incremental algorithm, is that the backup-path algorithm uses the knowledge of which subtree (T_n) a vertex belongs to in the original shortest path tree. Instead of just updating the current cost estimate of a path from the source s to a vertex u , as is done in the incremental algorithm, current cost estimates for paths through all subtrees in SPT_{init} are also recorded. The cost estimate used to determine if the vertex is added to the "backup"-shortest path tree, is the minimum current cost estimate of all current cost estimates through all available subtrees. This allows the algorithm to select an alternate path (i.e., through another subtree) when the minimum current cost estimate implies a path that is infeasible, e.g., traversing the failed neighbor.

Using a Fibonacci heap, the running time of Dijkstra's algorithm is $O(V \lg V + E)$ which results in a running time for a brute force algorithm of $O(NV \lg V + NE)$. If the graph is a clique, i.e., $|N| = |V| - 1$, both the backup-algorithm and the brute force algorithm have the same running time $O(V^2 \lg V)$. If $|N| < |V|$ then the running time of the backup-algorithm is better than the brute force algorithm. Finally, if $|N|^2 < |V|$, the running time for backup-algorithm becomes $O(V \lg V) + O(E) + O(V\sqrt{V} \lg(\sqrt{V}))$, which is slightly larger than the running time of Dijkstra's algorithm. The memory consumption of the backup algorithm depends on how the vertices of the initial shortest path tree are distributed amongst the subtrees. The worst case memory consumption arises from a very unbalanced distribution, in which case the asymptotical memory growth becomes $O(N(V - N)) = O(NV - N^2)$. However, if the vertex distribution is balanced, the memory growth becomes $O(N(V/N)) = O(V)$, which is on par with the memory consumption of Dijkstra's algorithm.