CMPSC250 Final Project Progress Report I

Aubrey Collins and Hawk Weisman

April 11, 2016

Due to difficulties with our original project topic, we have chosen to change direction to one which is better suited to our experience and background knowledge. Rather than continuing to work on our finger tree implementation, we now intend to work on an implementation of destributed graph algorithms for finding efficient paths in an onion-routed network.

Onion routing is a process by which intermediary nodes in a network do not know the original source, final destination, or contents of the messages they are relaying, allowing for anonymous communication. This works by encapsulating each message in an "onion", a data structure characterized by multiple layers of encryption.[1] Each routing node between source and destination decrypts a layer, revealing information about where to send the onion next. It pads the onion back to its original size, so that future nodes cannot estimate how many times the onion has already been forwarded. It then sends the onion to the next node in the path, and the process continues until the message arrives at the destination.[2]

[1] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. "Onion Routing". *Communications of the ACM* 42.2 (Feb. 1999); Michael G. Reed, Paul F. Syverson, and David M. Goldschlag. "Anonymous Connections and Onion Routing". *IEEE Journal on Selected Areas in Communications* 16.4 (Sept. 2006).

[2] Goldschlag, Reed, and Syverson, "Onion Routing"; Reed, Syverson, and Goldschlag, "Anonymous Connections and Onion Routing".

In onion routing software such as TOR, a client typically picks a random path to the destination rather than the shortest path. This feature is intended to increase the security of the network.[3] However, this makes onion routing unusable for applications which require low latency, such as voice chatting. We will simulate an onion-routed network in Scala, with the added goal of reducing latency. To this end, we will use a distributed algorithm to minimize the number of times an onion must be forwarded before arriving at its destination, while still maintaining anonyminity. We will then analyze our network to evaluate the tradeoff between security and latency, along with typical computational complexity analysis.

[3] Goldschlag, Reed, and Syverson, "Onion Routing"; Reed, Syverson, and Goldschlag, "Anonymous Connections and Onion Routing".

Dijkstra's algorithm and other centralized shortest-path algorithms are not ideal for this use case. The first reason for this is security. Dijkstra's algorithm would require each peer to know every edge in the graph, but this could enable an attacker to make reasonable guesses as to where a particular message is coming from or where it is going. For example, consider the case where a node A is connected to node B and no other nodes. Then it becomes clear that node A is the final destination of any message it receives from B, and the originator of any message which it sends to B.

Additionally, if a routing algorithm requires that each peer is aware of every other peer, an attacker compromising one peer deanonymizes the entire network. If, however, the set of peers which each peer must be aware of is restricted, then the system can often continue to be anonymous even if some of some of its members are compromised. Limiting information shared between peers improves the system's tolerance to attack by ensuring that a single peer cannot compromise the entire network.

Furthermore, Djikstra's algorithm suffers from scalability issues does scale well for this application. In networks with large amounts of peers, each peer would have to devote a large amount of memory to keeping track of the network state, much of which would be irrelevant to the messages it is trying to send or receive.

A distributed implementation could help alleviate these issues so that all nodes are not necessarily aware of the entire network geography. Several distributed shortest-path algorithms have been published, and we are currently exploring these options.[4] While in some cases, it may not be possible to find an optimal solution (*viz.* a short*est* path) without complete knowledge of the network, it is still possible for a greedy algorithm which finds only local optimums to produce an optimized solution (*viz.* a short*er* path). Even if a given routing solution cannot conclusively be shown to be the shortest possible path, producing an optimized solution rather than a randomly chosen one can still have measurable[5] effects on network latency.

[4] William T. Zaumen and J. J. Garcia-Luna Aceves. "Dynamics of Distributed Shortest-path Routing Algorithms". *ACM SIGCOMM Computer Communication Review* 21.4 (Aug. 1991); Pierre A Humblet et al. *An adaptive distributed Dijkstra shortest path algorithm*. Citeseer, 1988; Pierre A Humblet. "Another adaptive distributed shortest path algorithm". *IEEE Transactions on Communications* 39.6 (1991); R. G. Gallager, P. A. Humblet, and P. M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees". *ACM Transactions on Programming Languages and Systems* 5.1 (Jan. 1983).

[5] And more importantly, *noticeable*.

WE HAVE ALREADY COMPLETED much of a general-purpose implementation of multiple graph data structures in Scala. Our implementation provides concrete classes for representing edge-weighted and unweighted directed and undirected graphs, with the hopes that this graph structure may also be useable in future work. We make use of multiple abstract traits for types of graphs, such as weighted and unweighted, allowing us to maximize code sharing between the various graphs that our system models. Additional necessary work will involve extending and testing our graph data structures, implementing our routing algorithm, and using our graph implementation to construct a simulation demonstrating the use of our routing techniques. We will then conduct experimental analysis of our solution using this simulation.

References

Gallager, R. G., P. A. Humblet, and P. M. Spira. "A Distributed Algorithm for Minimum-Weight Spanning Trees". *ACM Transactions on Programming Languages and Systems* 5.1 (Jan. 1983), pp. 66–

77. ISSN: 0164-0925. DOI: 10.1145/357195.357200. URL: http://doi.acm.org/10.1145/357195.357200.

Goldschlag, David M., Michael G. Reed, and Paul F. Syverson. "Onion Routing". *Communications of the ACM* 42.2 (Feb. 1999), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/293411.293443. URL: http://doi.acm.org/10.1145/293411.293443.

Humblet, Pierre A et al. *An adaptive distributed Dijkstra shortest path algorithm*. Citeseer, 1988.

Humblet, Pierre A. "Another adaptive distributed shortest path algorithm". *IEEE Transactions on Communications* 39.6 (1991), pp. 995–1003.

Reed, Michael G., Paul F. Syverson, and David M. Goldschlag. "Anonymous Connections and Onion Routing". *IEEE Journal on Selected Areas in Communications* 16.4 (Sept. 2006), pp. 482–494. ISSN: 0733-8716. DOI: 10.1109/49.668972. URL: http://dx.doi.org/10.1109/49.668972.

Zaumen, William T. and J. J. Garcia-Luna Aceves. "Dynamics of Distributed Shortest-path Routing Algorithms". *ACM SIGCOMM Computer Communication Review* 21.4 (Aug. 1991), pp. 31–42. ISSN: 0146-4833. DOI: 10.1145/115994.115997. URL: http://doi.acm.org/10.1145/115994.115997.