Verification of the Statistical Model for Multi-Stage Message Distribution

Jedi High Council

July 11, 2013

1 Monte Carlo Simulation

To verify our model we implemented a monte carlo simulation that simulates the behavior of the network given parameters n, m, and k. In particular, the simulation initializes the network of n nodes in an unconnected state (i.e. all nodes are disconnected from each other and there exists a single node, the key manager, who possesses the key). Then, in discrete time steps, the simulation attempts to establish new connections between a pair of nodes u and v with probability p_1. In addition, nodes already in the process of establishing a connection proceed forward through the m stages of message distribution by a single step with probability p_2. At the end of one time step, the overall discrete time is incremented by one, and the process repeats until all nodes have been connected and received the key. This procedure is formalized in Algorithm 1. The source code, written in Matlab, has been made available at ¡LINK¿ for interested readers to experiment with. It is a self contained program that contains all of the required documentation within.

We repeated this general procedure for $T = 10000$ iterations each for different values of n, k, and m, collecting the average time over all program runs, standard deviation of each run, and the estimated error in the simulation. Our estimated error was quite small and well within the expected bounds of precision. We then computed the difference Δ between the output from our model and the average time reported by this monte carlo simulation to verify the correctness. We found that for all configurations considered the delta was always less than 0.2. We attribute this difference to the error introduced in monte carlo simulation.

```
Data: T, k, m, n, p_1, and p_2
Result: Expected time
total ← 0;
for T_i = 0 to T do
    t ← 0;
    A_c ← zeros[1 . . . n][1 . . . n];
    A_m ← zeros[1 . . . k][1 . . . n][1 . . . n];
    n_c ← 0;
    C_l ← zeros[1 . . . n];
    while n_c < n − 1 do
        L ← getReadyChildren(A_c, A_m, C_l, 1 − p_1)
        for r = 1 to n do
            for c = 1 to n do
                if A_m[r][c][m] = 1 then
                    A_m[r][c][m] ← 0
                    A_c[r][c] ← 1
                    A_c[c][r] ← 1
                    C_l[c] ← 1
                    n_c ← n_c + 1
                end
            end
        end
        for m' = 1 to m − 1 do
            for r = 1 to n do
                for c = 1 to n do
                    if A_m[r][c][m'] = 1 and rand() < p_2 then
                        A_m[r][c][m' + 1] ← 1
                        A_m[r][c][m'] ← 0
                    end
                end
            end
        end
        P ← getReadyParents(A_c, A_m, A_l)
        for i ← 0 to min{|P|, |L|} do
            A_m[P[i]][L[i]][1] ← 1
        end
        t ← t + 1;
    end
    total ← total + (t − 1);
end
output total/T
```

Algorithm 1: Monte carlo simulation to verify the statistical model. The functions $getReadyChildren()$ and $getReadyParents()$ uniformly select nodes from the entire group at random to be new children and parents, respectively, by analyzing the current state of the network as represented by the adjacency matrix A_m, message matrix A_m, and connected list C_l. Also, $getReadyChildren()$ uses probability $1 - p_1$ when selecting new children to establish connections with.