Jake Luoma
Network Systems Project Proposal
3/5/2020

Summary: I propose to work on drone swarm communication. At a high level, the project can be described with the following series of steps. Each step is described in greater detail below.

1. Set up simulation
2. Implement a neighbor detection algorithm
3. Implement a path selection algorithm
4. Implement data exchange within the swarm
5. Implement data exchange external to the swarm
6. Scalability improvements

Set up Simulation
One aspect of the simulation involves specifying which drones can communicate with each other. This may be specified as a list of drone pairs, where each pair can communicate. It can be modelled in a simulation software as a set of trees of routers and hosts. Alternatively, it could be modelled in Linux such that all drones are modelled as hosts all connected to a router whose forwarding behavior is controlled by iptables.

Another aspect of the simulation is the time element. The network topology should change over time, so a clock is required to determine when the topology changes. The network can transition between multiple topologies simply by using multiple lists of drone pairs and changing from one list to another after a certain amount of time has passed.

Implement neighbor detection
I propose that drones ping hello messages every x seconds with a sequence of Hello, Ack, Ack. Each drone must also be capable of running BGP to identify whether it can connect to the internet. Separate routing tables should be used for destinations within the swarm and destinations exterior to the swarm.

Implement path selection
I'm not sure how I want to do this yet. Congestion-awareness will be very important, but I haven't decided whether I want drones to make locally optimal decisions (ie. a BGP-like protocol with congestion awareness a la DCTCP) or globally optimal decisions (ie. full swarm awareness with congestion awareness based on RTT and/or available bandwidth). I may start with the locally optimal approach. To deal with changing topology, it will be necessary to either store multiple paths to a destination or to run neighbor detection and path selection again when multiple timeouts are encountered.

Implement data exchange within the swarm
Drones must be able to store another drone's sent data if the topology changes and a path to the destination drone is no longer available. I propose that an application should be dedicated to data exchange within the swarm so that data can be stored in user space until a path becomes available. A drone would send its data to the next hop's data handling application via FTP, along with the final destination IP address and socket. The data handling application would then attempt to forward the data. In the event of a failure to send due to timeout, the application would occasionally try again to send the data. If the drone is the final destination, the application will forward the data to the correct

socket. The performance can be improved through use of DPDK, XDP, or PF_RING. I propose to initially use the default TCP/IP stack.

Implement data exchange external to the swarm

I propose that drones with internet access advertise it as part of path selection. Then, similar to data exchange within the swarm, data will be sent to a specific application running on each host via FTP which is responsible for holding onto the data in case of the path to the destination becomes unavailable. The data is sent to the next hop, along with the destination IP address and a flag indicating it is external to the swarm. If the drone has internet access, it sends it to the internet using FTP.

Scalability improvements

The path selection, congestion awareness, and data exchange can all be significantly improved over what I've suggested. Once the system is working, I will solicit feedback on what I might focus on. I will also stress test it (ie. change the number of drones in the swarm, change the time between network changes, change the neighbor detection frequency, etc.) to try to identify failure points.