The beacon

The problem

For most distributed computing, there is a marked non-uniformity in networking access. For example, it is common that all the nodes in a cluster can talk to each other (and normally at full bandwidth), but the connectivity to nodes outside the cluster may be restricted in number and speed.

In our case, we use *myriad* to manage our containers; these containers share a local network. We don't know their addresses before their run, so (in the absence of persistent storage) how do they communicate what they did?

This is a different to the related problem of how these nodes get organised internally (elect a leader, manage services, etc).

Solution 1: the oracle

A simple solution is that of the oracle: a known, unchanging, IP address where you can store a modest amount of information about your cluster. This would typically include a database address and a monitoring port. Often, the database is transient and local for this computation, and can be used (say, using *etcd*) to elect leaders and store information about cluster-wide services. There are existing commercial oracle services; some are free.

Solution 2: the beacon

A simpler form of the oracle is that of the beacon. Typically, the beacon

- sets up a server port at port *addr* (which can be assigned transiently)
- it prints *addr* on standard output and then forks. The original process exits.
- the forked process, or *beacon*, then listens on the address and copies any input it receives to a specified place.
- when the beacon receives some specific string, or *termination message*, it exits.

The normal usage would be to start the beacon, capture its address, and then feed that as a command line argument to every container we start up. We can either arrange for the cluster software to send the termination message when it finishes, or use *curl* (or equivalent) to send the termination message.

Note that the beacon is receive only. And it does not address the problem of how to manage cluster-wide activities, such as elections etc.