**The Health Checking Problem**

At Uber we have many microservices (A, B & C below), written in different languages. To avoid writing & maintaining the discovery & routing logic in multiple languages we offload this work to an on-host proxy, P.

When Service A needs to communicate with Service B, it sends the request to the local proxy which then forwards the request to an instance B. Similarly for B communicating with C.

Physical hosts (the grey boxes below) may run any combination of the microservices. If there is an instance of B on the same machine as A, no local preference will be given. That is, all instances are equal in the eyes of the proxy.

Services are scheduled via a Cluster Manager. They may come and go from any physical host at any time.

P

P

P

P

P

A

A

B

B

B

P

C

P

C

The proxies are an open source component (HAProxy). They are configured via a file, generated from a cron script. The file contains:

B:

ip:port

ip:port

ip:port

C:

ip:port

ip:port

Proxy

A

cfg file

To avoid instances that may have failed, each Proxy performs health checks in the form: GET /health. If an instance fails the health check the proxy will avoid sending it requests until it passes again.

As Uber has grown, we’re at the point where the number of health checks is starting to overload the service instances. It’s particularly bad for languages like Python.

It’s your first week at Uber and you’ve been asked to solve this problem. Peak traffic is 2 months away.

What would you do? We’ll need both a short & a long term solution.

Clarification:

The health check is hosted at the service level. E.g in addition to all the endpoints, each service will implement a /health endpoint.

Functional Requirements

1. No of health checks should not overload the service instances .

Drawing [link](https://app.codesignal.com/live-interview/WRsYhrFccXmSZsCAr?userRole=candidate)

