

PROJECT 2: Implementation of a technical, social networking service using a distributed system

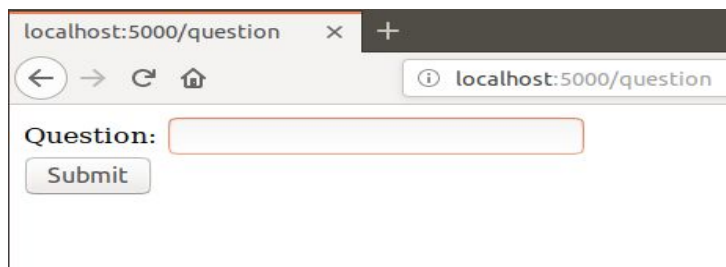
Project Partners: Shaunak Pagnis and Sumeet Menon

Service Discovery: Service discovery is handled using a service registry. The service registry is a cluster of Redis, which is a key-value store. Whenever a service goes up, we update the service information in Redis along with a timestamp. Each update also comes up with a expiry time. After the expiry time is elapsed, the service is responsible for keeping the registry up-to-date using a simple ping request.

Server and Client: For the end-user, it is a simple web application with the ability to question, provide answers and comment on answers. This is implemented using the Flask framework for python.

Note: We're using Flask to ease the implementation of the web UI, the actual distributed system does not use any libraries or frameworks.

For eg:



A screenshot of a web browser window with the address bar showing 'localhost:5000/question'. The page contains a form with a label 'Question:' followed by a text input field. Below the input field is a 'Submit' button.

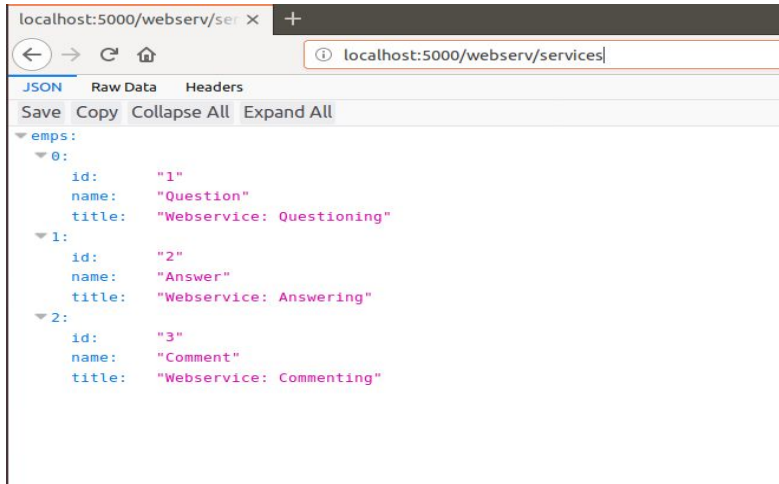
On clicking the submit button the question would reflect as shown below. Similarly, we would be making services to answer and comment.



A screenshot of a web browser window showing the submitted question. The text 'The question is: Which is the most widely used framework for web development?' is displayed on the page.

The question is: Which is the most widely used framework for web development?

At this point, we're using a single RDMS instance for persistence, but in the next phase, this too shall be distributed. The end user sends the questions, answers, and comments as a URL-encoded form. Communication between services themselves will be using gRPC.



Load Balancer: The Load balancer itself is a level 7 load balancer which works like a dispatcher in a server cluster. The load balancer listens for requests on a port and upon receiving a request, it performs the following operations:

- Identify the service required
- Find out the server for that service with least load
- Spin up a new thread that forwards this request to the identified server
- Send an acknowledgment to the client.

Requests from clients:

```

...Socket_Render=/private/tmp/com.apple.launchd.rP7nYpALV/Render ...TMPDIR=/var/folders/2k/w3ctn7zd07d3xw4z00sd25k00000gn/T/
127.0.0.1 - - [21/Nov/2018:19:57:10 -0500] "GET / HTTP/1.1" 200 3 0.0015
127.0.0.1 - - [21/Nov/2018:19:57:10 -0500] "GET /perform HTTP/1.1" 200 20 0.0010
127.0.0.1 - - [21/Nov/2018:19:57:17 -0500] "GET / HTTP/1.1" 200 3 0.0015
127.0.0.1 - - [21/Nov/2018:19:57:17 -0500] "GET /perform HTTP/1.1" 200 20 0.0015
127.0.0.1 - - [21/Nov/2018:19:57:18 -0500] "GET / HTTP/1.1" 200 3 0.0014
127.0.0.1 - - [21/Nov/2018:19:57:18 -0500] "GET /perform HTTP/1.1" 200 20 0.0010
127.0.0.1 - - [21/Nov/2018:19:57:18 -0500] "GET / HTTP/1.1" 200 3 0.0016
127.0.0.1 - - [21/Nov/2018:19:57:18 -0500] "GET /perform HTTP/1.1" 200 20 0.0013
127.0.0.1 - - [21/Nov/2018:19:57:19 -0500] "GET / HTTP/1.1" 200 3 0.0022
127.0.0.1 - - [21/Nov/2018:19:57:19 -0500] "GET /perform HTTP/1.1" 200 20 0.0013
127.0.0.1 - - [21/Nov/2018:19:57:19 -0500] "GET / HTTP/1.1" 200 3 0.0016
127.0.0.1 - - [21/Nov/2018:19:57:19 -0500] "GET /perform HTTP/1.1" 200 20 0.0009

```

Worker threads for sending requests to specific services:

```

...Socket_Render=/private/tmp/com.apple.launchd.rP7nYpALV/Render ...TMPDIR=/var/folders/2k/w3ctn7zd07d3xw4z00sd25k00000gn/T/
2018-11-22T00:57:10.560Z 46071 TID-owtc6d6a7 Balancer::Worker JID-7a0861450b9a887e7e6a0335 INFO: start
body: Performed operation! code: 200 message: OK
2018-11-22T00:57:10.569Z 46071 TID-owtc6d6a7 Balancer::Worker JID-7a0861450b9a887e7e6a0335 INFO: done: 0.009 sec
2018-11-22T00:57:17.563Z 46071 TID-owtc5ai0z Balancer::Worker JID-c164f9a65719bcf2c3f364ed INFO: start
body: Performed operation! code: 200 message: OK
2018-11-22T00:57:17.568Z 46071 TID-owtc5ai0z Balancer::Worker JID-c164f9a65719bcf2c3f364ed INFO: done: 0.005 sec
2018-11-22T00:57:18.201Z 46071 TID-owtc6d6a7 Balancer::Worker JID-c3e5ffc48d72b01347f1eca0 INFO: start
body: Performed operation! code: 200 message: OK
2018-11-22T00:57:18.205Z 46071 TID-owtc6d6a7 Balancer::Worker JID-c3e5ffc48d72b01347f1eca0 INFO: done: 0.003 sec
2018-11-22T00:57:18.762Z 46071 TID-owtc5ai0z Balancer::Worker JID-f20a47523127a7a080d35d89 INFO: start
body: Performed operation! code: 200 message: OK
2018-11-22T00:57:18.767Z 46071 TID-owtc5ai0z Balancer::Worker JID-f20a47523127a7a080d35d89 INFO: done: 0.004 sec
2018-11-22T00:57:19.317Z 46071 TID-owtc5aeyf Balancer::Worker JID-7f86892563791027c7e4962e INFO: start
body: Performed operation! code: 200 message: OK
2018-11-22T00:57:19.321Z 46071 TID-owtc5aeyf Balancer::Worker JID-7f86892563791027c7e4962e INFO: done: 0.004 sec
2018-11-22T00:57:19.859Z 46071 TID-owtc6d8lf Balancer::Worker JID-6b011b18e20a66c2e54deb7f INFO: start
body: Performed operation! code: 200 message: OK
2018-11-22T00:57:19.862Z 46071 TID-owtc6d8lf Balancer::Worker JID-6b011b18e20a66c2e54deb7f INFO: done: 0.003 sec

```

Changes from the previous report:

- Removed RAFT based consensus after discussion in class.
- Added WSDL service information to each service as its mandatory.
- Reduced the scope of actual services after discussion in class.

Remaining work:

- Replicate the load balancer
- Replicate the service registry
- Evaluate between event-machine and callbacks for sending responses to the client
- Inter-service communication using gRPC
- Node failures and health checks
- Managing expiry of a service in the service registry

Proposed Timeline

21st November	Completed the service discovery and the load balancer
29th November	Complete the inter-service communication.
7th December	Complete the replications for balancers and registries.
10th December	Testing and finishing touches.