COMP 4601

Assignment: Distributed Search Engine Functionality

The Requirement

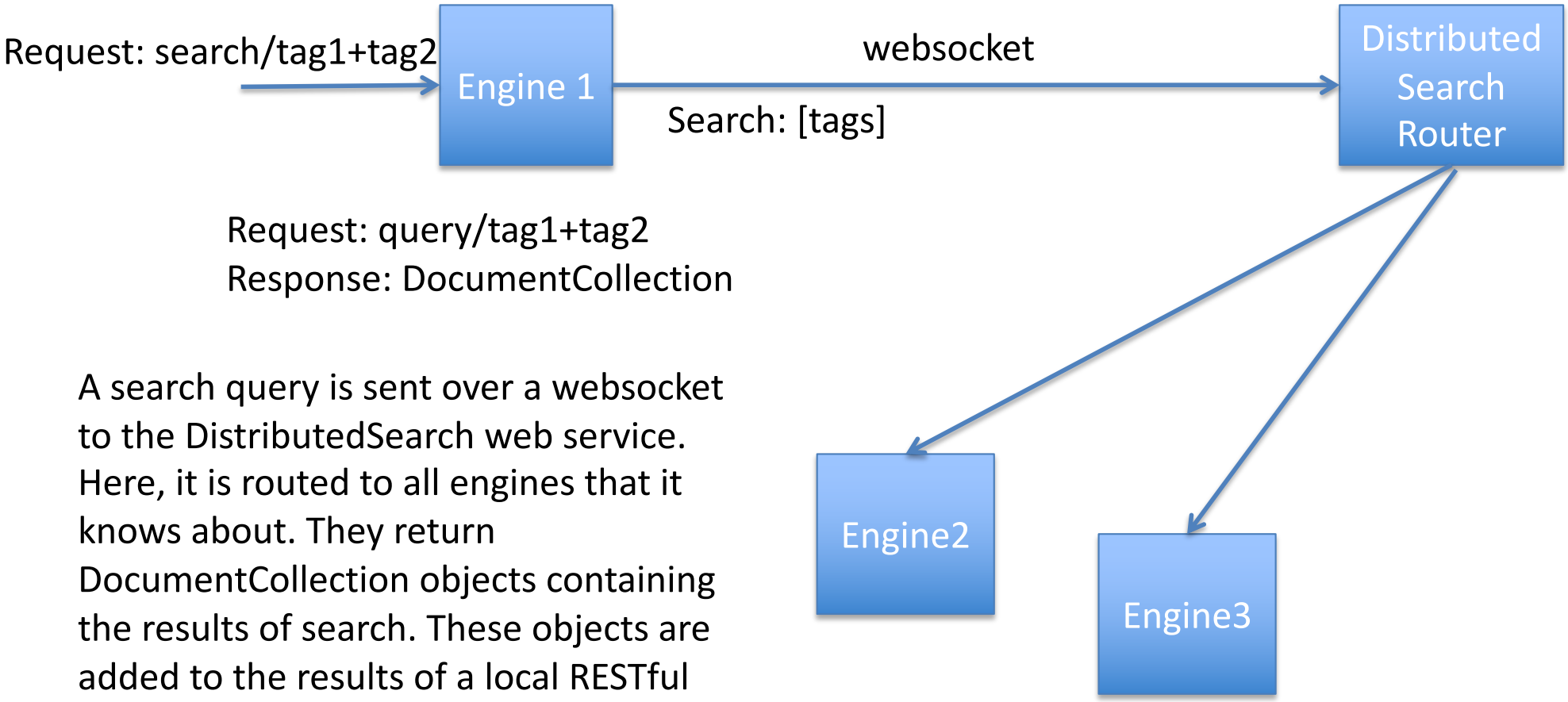
- You are to support search distribution.
- Search engines are registered using the Generic Directory RESTful web service.
- This service is hosted off of:
  - wss://sikaman.dyndns.org:8443/DistributedSearch/router
- Slides 5 onwards are the ones to consult
- The sda-1.1.jar contains all required software. See assignment web page for link.

Software dependencies

- tyrus-standalone-client-1.13.1.jar
- gson-2.8.5.jar

- Jars for support of:
  - Lucene
  - Jersey

Mechanism



A search query is sent over a websocket to the DistributedSearch web service. Here, it is routed to all engines that it knows about. They return DocumentCollection objects containing the results of search. These objects are added to the results of a local RESTful web search.

SEARCH SERVICE MANAGER: A SIMPLE INTERFACE BETWEEN RESTFUL SEARCH ENGINES

SearchServiceManager (SSM)

- This class is your interface to distributed search functionality. It provides APIs to interact with the DistributedSearch service (DS) hosted on my server.
- The SSM accesses the DS to perform a distributed search.
- It uses the WSS (secure websocket) protocol.

IMPORTANT SEARCH ENGINE SOFTWARE MODIFICATIONS: YOU NEED TO READ SLIDES 8-14 CAREFULLY!

Software Changes

- In order to integrate distributed search functionality in your engine you must add code in 3 places.
- One: In your RESTful search API. See slide **11**.
- Two: You must support a query/{tags} API as shown in slide **12**.
- Three: To initialize the SSM, slide **9**.

Initialization

- You must add a call to start the SearchServiceManager in initialization code that you provide (e.g., an init() method)

```
SearchServiceManager.getInstance().start()
```

- This call initializes the SSM and connects it to the DistributedSearch service.

The Search API

- You use the SSM through the API:

  SearchResult search(String tags)

- The tags argument is the query passed to your RESTful web service.
- The SearchResult accumulates all document references from engines responding to the search. Searches are performed in parallel. There is a timeout of 10 seconds (by default).

Example

Blue text must be included in your code for distributed search to work!

```
@GET
@Path("search/{tags}")
@Produces(MediaType.TEXT_HTML)
public String searchForDocs(@PathParam("tags") String tags) {
    // Perform the distributed part of the search
    SearchResult sr = SearchServiceManager.getInstance().search(tags);
    // Perform your local search (this is my specific code, yours differs!)
    ArrayList<Document> docs = Documents.getInstance().query(tags);
    // We will wait for up to 10 seconds or until all distributed searches complete
    // (whichever is shorter) but will then take the documents that we have.
    try {
        sr.await(SDAConstants.TIMEOUT, TimeUnit.SECONDS);
    } catch (InterruptedException e) {
    } finally {
        SearchServiceManager.getInstance().reset();
    }
    // Take the state of the documents
    docs.addAll(sr.getDocs());

    // Build the page (not provided here)
    return documentsAsString(docs, tags);
}
```

query/{tags} RESTful API

```
@GET
@Path("query/{tags}")
@Produces(MediaType.APPLICATION_XML)
public DocumentCollection queryAsXML(@PathParam("tags") String tags) {
        DocumentCollection dc = new DocumentCollection();
        // Perform your local search (this is my specific code, yours differs!)
        dc.setDocuments(Documents.getInstance().query(tags));
        // Return the XML version of the DocumentCollection
        return dc;
}

// You must support this RESTful interface in order for the Search Service
// Manager to see your service. Producing the MediaType.APPLICATION_XML is
// critical. You must (as an assignment requirement) support an API that produces
// the MediaType.TEXT_HTML.
```

IMPORTANT!

- Make sure that you provide the services as requested by the assignment.
- Names are critically important!
- You **MUST** support search/{tags} and query/{tags}. Query is local, search supports distribution.

Search Service Manager Logging

- A logger is provided by the SSM

  SearchServiceManager.getInstance().log(Level, String log)

- Logs saved in home directory:
  - ssm-log.html
  - ssm-log.csv

Software

- The required software may be found in the **sda-1.1.jar** file.
- This file can be obtained through the "Assignment Details" section of the Assignment page.