Computer Networks
Fall 2015/16

Lab 2 – Web Crawler

Lab1 must be submitted using the course website by **17-1-2016.**
Submission file name should be lab2-ID1-ID2.zip
For example lab1-123-678.zip for ID 123 and ID 678 (and nothing else)
>>> No late submissions will be accepted.

Building Multi-threaded Web Crawler

Goals:

- Multi-threaded Web Crawler.
- Port scanner
- Results page

Major Milestones (Web Server):

- config.ini change:
  - maxDownloaders: The maximum number of threads being using to request and receive pages (**we will use 10**).
  - maxAnalyzers: The maximum number of threads that extract new links from downloaded pages (**we will use 2**).
  - imageExtensions: The extensions considered as images (**we will use “bmp, jpg, png, gif, ico”**)
  - videoExtensions: The extensions considered as videos (**we will use “avi, mpg, mp4, wmv, mov, flv, swf, mkv”**)
  - documentExtensions: The extensions considered as documents (**we will use “pdf, doc, docx, xls, xlsx, ppt, pptx”**)

- Web-based GUI (graphic user interface):
  - The **default page** contains controls to control the crawler:
    - **GUI** contains HTML form:
      - Textbox to enter *Domain* to crawl (i.e. www.idc.ac.il or http://www.google.com/).
      - “Perform full TCP port scan” checkbox
      - “disrespect robots.txt” checkbox (http://tools.seobook.com/robots-txt/)
      - “Start crawler” submit button

        - **Notice:** If the crawler is running the text "crawler already running…" should appear instead of the HTML form.
    - Below HTML form there's a list of links to all past results pages. Each line contains a link to a results page.
        - The link text has the format: [domain] – [date crawl started] – [time crawl started]
- Clicking the submit button starts the crawler and returns execResult.html:
    - If crawler started successfully, the text "crawler started successfully" Should appear instead of HTML form.
    - If crawler did not start successfully (for instance, the given URL is invalid or cannot be resolved), the HTML form should appear and also the text "Crawler failed to start because: [error description]".
    - At the bottom of the HTML page there are links to result pages of previous executions.

- [domain name_date_time].html:
    - The page contains the following statistics that the crawler gathered during its execution:
        - Did the crawler respect robots.txt or not.
        - Number of images (from config.ini)
        - Total size (in bytes) of images
        - Number of videos (from config.ini)
        - Total size (in bytes) of videos
        - Number of document (from config.ini)
        - Total size (in bytes) of documents
        - Number of pages (all detected files excluding images, videos and documents).
        - Total size (in bytes) of pages
        - Number of internal links (pointing into the domain)
        - Number of external links (pointing outside the domain)
        - Number of domains the crawled domain is connected to
        - The domains the crawled domain is connected to
            - **If the linked domain has been crawled**, the text should be a link to this domain's page.
        - If requested, the opened ports.
        - Average RTT in milliseconds (Time passed from sending the HTTP request until received the HTTP response, **excluding** reading the response).
    - A link to the main page (the default page)

- Multi-threaded web crawler:
    - The crawler starts at the root page of the given domain (e.g. google.com, yahoo.com, etc)
    - New links can be detected only in HTML links. (http://www.w3schools.com/html/html_links.asp)

- The crawler crawls only pages inside the given domain.
- Use producer-consumer design pattern (from operating system course) to synchronize between downloaders and analyzers.
    - Do not forget to make the data structures you are using *thread-safe*.
- If requested, port scanning should be done before the crawling of the give domain.
- Once the crawler finished its job, it should save the results into [domain name_date_time].html, when the date and time are the beginning time of the execution.

- The result pages must not be available to direct surfing, only if the user came from the default page (the main page of the application). If a client does surf directly to a result page, or not from the default page, it should receive **403 Forbidden** HTTP response code.

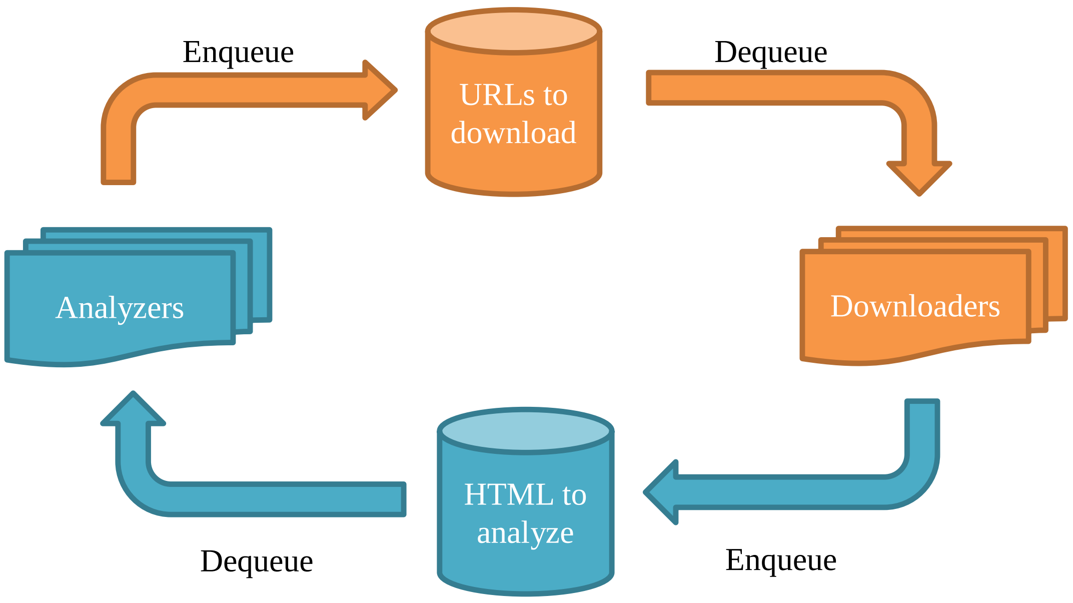
Goals in Detail:

1. Multi-threaded web crawler:

A crawler is an application that traverse through some kind of data (e.g. data structure, file system etc.) to perform different tasks. In particular, a *web crawler* is an application that "crawls" through all the pages and resources of a website. Web crawlers can do many tasks while crawling a website, like indexing for search engine (like Google's GoogleBot), download the website for off-line use, or like the current lab, extract statistics regarding the website. Crawlers and in particular web crawlers are a very basic types of application to traverse data to extract information and perform tasks.

Web crawling is a very time consuming process, therefore, web crawler are usually multi-threaded to increase its efficiency and reduce the time of execution. By using multiple threads it is possible to download many pages in the same time and by that reduce the crawling time.

Tip: In some cases (like images, videos and documents) you need to use HEAD HTTP requests instead of get, because you can tell upfront that the data does not contain HTML links, therefore shouldn't be analyzed.
The crawler should use producer-consumer design pattern (recall from operating systems course):



The crawling ends when both queues are empty.

2. Robots.txt:

Web crawlers traverse all the pages the crawler's analyzers are able to extract; therefore they can also reach web pages that the webmaster (website administrator) does not want the crawlers to reach. In order to enable the webmaster a way to "tell" a certain crawler or all crawlers not to traverse into a certain pages "robots.txt" file has been invented, to give the crawler instructions of what it is allowed or not allowed to do. The robots.txt file always located in the root of the website (for example, http://www.google.com/robots.txt).

Your crawler must support robots.txt (with wildcards). More information about robots.txt can be found at: http://tools.seobook.com/robots-txt/.

If robots.txt file does not exist in a certain website, or it is not well formatted, it should be ignored.

In case "disrespect robots.txt" was selected, crawl **on purpose** to all the links robots.txt reveals.

Exception Handling:

The program must not crash!

Use exception handling to recover. If you cannot recover, print what happened to the console and close the application gracefully.

Notice that crashing will drop **many** points!

Traces (a.k.a. print to console):

In addition to already existing traces, the following traces are **required**:

- Downloader starts downloading URL (and the URL itself)
- Downloader ends downloading the URL (and the URL itself)
- Number of link Analyzer extracted from a given URL (and the URL itself)
- Every time a Downloader or Analyzer dequeues from their queue, trace the number of items left in the queue.

You are **encouraged** to use additional traces in your code!
Traces will help you debug your application and perform a better QA.
Make sure your traces actually mean something that will help to understand better what is going on during runtime. **That can and will save you a lot of time!**

Bonus:

Feel free to add more functionality to the application as you see fit!
Good ideas will be **rewarded** with points and world-wide fame (score may be over a 100)!!!

Please write and explain all the implemented bonuses in a file named bonus.txt.
Notice that bonuses that will not be mentioned in bonus.txt will not be checked, hence they will be ignored!

Important: We don't guarantee in advance that for extra-functionality, extra-points will be rewarded. Also, the idea is to ***add*** a new functionality ***not replace a requested functionality*** – that will lead to dropping of points! In other words, do not ignore things we require, and implement other things instead and call it a bonus! We cannot make it any clearer than that!

Discussion Board:

Please, use the forum of the course to ask questions when something is unclear.

You must make sure that you check the discussion board. If there are unclear matters about the lab that were raised in the discussion board, **our answers are mandatory**, and apply to **everyone**!

You can register to receive e-mail notification from the forum.

A word from your checker:

- **Do not** create your own packages, all your classes **must be in the same package**, and compile while all sources in the same directory
- If you are coding using a mac, make sure you remove all hidden directories generated by your IDE
- You can implement the lab using up to JDK1.7
- Your code will be tested **without an IDE**. Test your code without an IDE and make sure it works!
- Compilation error will drop a huge amount of points

Remember - A happy checker is a merciful checker!

Submission:

- You **must** submit a batch file called "compile.bat" - The batch file compiles your code successfully. If the batch fails to compile, points will be dropped.
- You **must** submit "run.bat" – The batch file starts your application

Submit the lab using a zip that contains the following files:

- Sources
- config.ini
- compile.bat
- run.bat
- server root directory
- bonus.txt with the bonuses you implemented – if applicable.
- readme.txt that explains what exactly you have implemented and how.

Notice: not submitting one of the files needed requested (exclude bonus.txt) will lead to dropping of points!

That's it ☺ !

Good Luck!