Big Data Practice – Network Measurements

The objective of this laboratory is to apply big data processing tools you learned in previous lectures to a practical problem in network measurements. You are given datasets of HTTP and TCP measurements collected in the network of Polito. Details and formats of the data are described below. You are expected to answer the following questions:

(i) What are the most popular "websites" contacted from Polito network?
   a. Define "website" as the FQDNs in TCP Logs

(ii) How is the network usage varying per hour of the day? And per day of the week?
   a. Consider the fraction of HTTP, TLS/SSL, IMAP, SMTP volume in terms of bytes, and show how it varies over time

(iii) **Bonus:** HTTP usage in the campus:
   a. What is the fraction of HTTP requests fired from Polito that matches a response?
   b. Which "websites" generate most errors (4xx or 5xx Response Codes)?

Data Input Format

Two types of logs are provided as input, both captured simultaneously from a router of the university.

1. TCP Log

C2S	S2C	Short description	Unit	Long description
1	**15**	Client/Server IP address	-	IP addresses of the client/server
2	**16**	Client/Server TCP port	-	TCP port for the client/server
3	**17**	Packets	-	total number of packets observed form the client/server
4	**18**	RST sent	0/1	0 = no RST segment has been sent by the client/server
5	**19**	ACK sent	-	number of segments with the ACK field set to 1
6	**20**	PURE ACK sent	-	number of segments with ACK field set to 1 and no data
7	**21**	Unique bytes	bytes	number of bytes sent in the payload
8	**22**	Data pkts	-	number of segments with payload
9	**23**	Data bytes	bytes	number of bytes in the payload, including retransmissions
10	**24**	rexmit pkts	-	number of retransmitted segments
11	**25**	rexmit bytes	bytes	number of retransmitted bytes
12	**26**	Out seq pkts	-	number of segments observed out of sequence
13	**27**	SYN count	-	number of SYN segments observed (including rtx)
14	**28**	FIN count	-	number of FIN segments observed (including rtx)
	29	First time abs	ms	flow first packet absolute time (epoch)
	30	Last time abs	ms	flow last segment absolute time (epoch)
	31	Connection type	-	bitmap stating the connection types (1 = HTTP; 1024 = SMTP; 4096 = IMAP4; 8192 = TLS/SSL)
	32	FQDN	-	Fully Qualified Domain Name recovered using DNHunter

Where:

- C2S stands for “client to server”
- S2C stands for “server to client”

Each line in this log file refers to a TCP connection. The input file is tabular, and each field is separate by a *blank space*. Entries in the log are sorted by the time the flow meter (Tstat) expired the record from its flow cache. Thus, do not assume any particular ordering regarding start and end times of records in the logs. Note that columns with IP addresses are anonymized.

2. **HTTP Log:**

C2S	S2C	Short description	Unit	Long description
1	1	Client IP address	-	IP address of the client (sending the request/receiving the response)
2	2	Client TCP port	-	TCP port for the client
3	3	Server IP address	-	IP address of the server (receiving the request/sending the response)
4	4	Server TCP port	-	TCP port for the server
5	5	Segment time abs	s	absolute time [s] (epoch) of the request/response
6		Request method	-	request method (GET/POST/HEAD)
7		Hostname	-	Value of the "Host:" HTTP request field
	6	Response string	-	response identifier (always "HTTP")
	7	Response code	-	HTTP response code (2xx/3xx/4xx/5xx)
	8	Content len	bytes	value of the "Content-Length:" HTTP response field

Column 6 can be used to distinguish between *S2C responses*, for which there is always the string "HTTP", and *C2S requests*, identified by methods GET/POST/HEAD. Again, IP addresses are anonymized in this dataset, but connection IDs (client IP address, server IP address, client TCP port, server TCP port) are consistent with TCP Logs. The input file is tabular, and fields are separate by *TAB characters* in this case.

You have no particular requirements about which tool or language to use (MapReduce, Spark, etc). Think about which option would suit best the tasks below, and design the solution accordingly.

Data Location

The data is located at the following folder in HDFS:
/data/students/net_measurements/HTTP
/data/students/net_measurements/TCP

The two folders contain a set of zipped textual files. Spark automatically decompresses the zip files while reading them. Hence, you can use the standard approach to read the input files.

Pay attention that the first two lines of each file are comments.

Exercises:

1) What are the most popular websites contacted from Polito network?

In this exercise, summarize the information in the TCP Logs. Using the column FQDN as key, calculate the number *of bytes* sent and received by the *10 most popular websites* (in terms of number of TPC connections, i.e., number of lines of the TCP Logs) contacted from Polito network. Use the column "connection type" to filter the logs, keeping only HTTP (connection type=1) or HTTPS traffic (connection type=8192 and server TCP port 443).

2) How is the network usage varying per hour of the day? And per day of the week?

Use the information in the "first time abs" column to calculate the number of bytes sent/received from Polito network. Calculate the number of bytes sent/received:

- For each hour in the trace for HTTP, HTTPS, IMAP(S), POP(S), SMTP(S) protocols
- For each day of the week in the trace for HTTP, HTTPS, IMAP(S), POP(S), SMTP(S) protocols

Note that "first time abs" is in millisecond epochs. For long connections, consider that all bytes are accounted when the flow starts, but remember that they are logged when the flow actually ends in the trace. Assume HTTPS, IMAPS, POPS, SMTPS run on server TPC ports: 443, 993, 995 and 465, respectively (associated with connection type = 8192).

Given a time expressed in milliseconds since the Unix epoch, you can use the java.util.Calendar class to extract hour and day of the week from it.

The following example shows how to extract hour and day of the week by using a Calendar object.

```
import java.util.*;
public class TestDate {
        public static void main(String[] args) {
                // create a Calendar object
                Calendar cal = Calendar.getInstance();
                // Set the time of the Calendar object to
                // the date associated with the value 1459462711296.270020
                // (in millisecond since the Unix epoch).
                // Only the integer part must be considered
```

```
                String millisecondsEpoch="1459462711296.270020";
                cal.setTimeInMillis(Long.parseLong(millisecondsEpoch.replaceAll("\\..*$", "")));
                // Print the hour
                System.out.println("Hour: "+cal.get(Calendar.HOUR_OF_DAY));
                // print the day of the week (1=SUNDAY       , .... 7=SATURDAY)
                System.out.println("Day of the week: " + cal.get(Calendar.DAY_OF_WEEK));
        }
}
```

Bonus Exercises:

3) HTTP requests fired from Polito always match a response?

Each HTTP request in the HTTP Logs should have a HTTP response in the HTTP Logs as well (column 6 = HTTP).

In this exercise you have to match requests and responses in the HTTP Log. Use as key: <client IP address, client TCP port, server IP address and server TCP port>. For simplicity, assume that a client never reuses the same port to open two connections to the same server. Recall that in the HTTP Logs, requests and responses appear chronologically sorted, i.e., as they appear in the network.

Calculate the number of HTTP requests that are not matched to a response, as well as the number of responses that are not matched to a request. Those cases happen mostly because of packet loss during the capture. Are they significant?

4) Which websites generate most errors (4xx or 5xx Response Codes)?

HTTP status codes 4xx and 5xx indicate errors - e.g, when a page is not found, the client receives a HTTP response with error code 404. The HTTP Logs have the status code of every HTTP response in the column “response code”.

What is the percentage of *requests* receiving error codes 4xx or 5xx? Report the top-10 FQDNs generating errors.