|  |
| --- |
| Ripple Technical Challenge Solution |
|  |
| **Prepared By: Jayesh Jain** |
| **16th Aug 2020** |



|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Prepared By** | **Date** | **Comments** |
| 1.0 | Jayesh Jain | 16/08/2020 | Final |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Index

1. Requirement
2. Software requirements/ Pre-requisite
3. Solution Overview
4. Detailed Solution
5. Details on each questions

Requirement

**Preface**: Transactions on the XRP Ledger are communicated and recorded by a network of computers running a software daemon called "rippled." Every few seconds, the network reaches consensus on a new set of transactions which are applied to the old state of the ledger to create a new “validated ledger” that gets broadcast across the network. You can use rippled's server\_info command to gather information about the set of validated ledgers that this rippled has received.

**Task Summary**: Write a script/program that periodically calls rippled’s server\_info command and records the sequence number of the latest *validated ledger* along with the current time. Record this data in a file. Then, use this data to construct a plot (time on the x-axis, sequence number on the y-axis) that visualizes how frequently the ledger sequence is incremented over time (i.e. how often new ledgers are validated). Choose a time span and polling interval that can effectively capture and depict this information.

In a few paragraphs, describe your process and results:

* How does your script work?
* How did you decide on your polling interval?
* What do the results tell you?
* What might explain the variation in time between new ledgers? (this description of the consensus algorithm may help you: <https://developers.ripple.com/consensus-principles-and-rules.html>)

Bonus question #1: Enhance your script to calculate the min, max, and average time that it took for a new ledger to be validated during the span of time captured.

Bonus question #2: There are some other (better) ways that you could use the rippled API to find how long each ledger took to close/validate. Using the API documentation, find and describe one of these methods (you don’t need to actually implement it).

**Notes**

- Use [s1.ripple.com:51234](http://s1.ripple.com:51234/) as your target rippled server

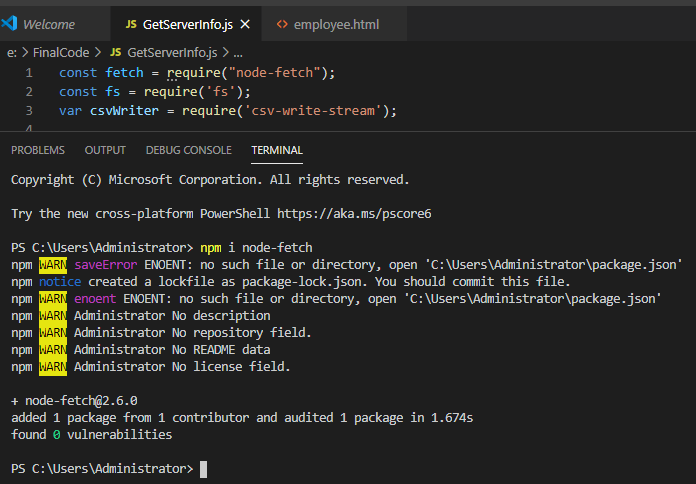
- A description of the server\_info command can be found here: <https://developers.ripple.com/server_info.html>

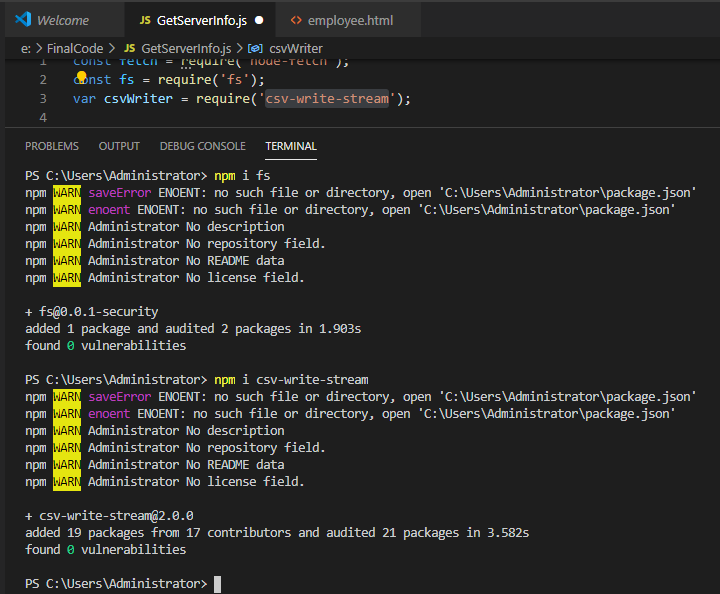
- You can use any plotting software, but we recommend gnuplot. See this page for examples of how to get started: <https://alvinalexander.com/technology/gnuplot-charts-graphs-examples>

- Feel free to get creative. Any additional insights which demonstrate your coding skills or general knowledge are more than welcome!

Software’s requirement / pre-requisite

1. IDE for java script development / html design – Visual Studio Code
2. Install Nodejs along with necessary packages
   1. npm i node-fetch (for calling rippled JSON-RPC method (Server\_info) to get ServerInfo)
   2. node i csv-write-stream (for creating and writing the rippled server info data in csv file.)
   3. node i fs and other packages for nodejs and npm





***Solution Overview***

Technical challenge can be fulfilled using multiple implementations like (Unix Shell scripting, Java Script, Java program).

My implementation is based on Java scripting and Html.

1. Java Script implementation :
   1. It is used to call the rippled server JSON-RPC server\_info method to fetch the details.
   2. Creating the CSV file and appending the XRP ledger info detail (like Time of the ledger, Sequence Number and calculated ledger duration) into the csv file.
   3. Periodically call the main() function to fetch the Server\_info from Rippled server.
2. UI Implementation Using HTML page :
   1. This implementation is to read the information from ServerInfo.csv file.
   2. Dispalying all the entries in the CSV file.
   3. Drawing the graph to plot time duration (x-axis) and sequence number (y-axis).
   4. Calculating the Min , Max and Avg XRP ledger performance statistics.

***Solution in Detail***

My implementation is based on Java scripting and Html.

1. Java Script implementation (GetServerInfo.js):
   1. It is used to call the rippled server JSON-RPC server\_info method to fetch the details :
      * Below code is used to define and assign require values to fetch server info.

const Url = 'http://s1.ripple.com:51234';

const Data = {

"method": "server\_info",

"params": [

{

"api\_version": 1

}

]

};

//optional parameters

const otherPram = {

headers: {

'Accept': 'application/json',

'content-type': 'application:json',

},

body: JSON.stringify(Data),

method: "POST"

};

* + - Below code is used to fetch server info.

fetch(Url, otherPram).then(data => { return data.json() }).then(function (data) {

const ledgerTime = new Date(data.result.info.time);

const ledgerSeqNumber = data.result.info.validated\_ledger.seq;

const preLedgerDuration = data.result.info.last\_close.converge\_time\_s;

* 1. Creating the CSV file and appending the XRP ledger info detail (like Time of the ledger, Sequence Number and calculated ledger duration) into the csv file.
     + Below code is used

if (ledgerLastUpdate == '') {

ledgerLastUpdate = new Date();

ledgerLastUpdate.setSeconds(ledgerLastUpdate.getSeconds() - preLedgerDuration);

console.log(ledgerLastUpdate);

}

var writer = csvWriter({ sendHeaders: false }); //Instantiate var

var csvFilename = "ServerInfo.csv";

// If CSV file does not exist, create it and add the headers

if (!fs.existsSync(csvFilename)) {

writer = csvWriter({ sendHeaders: false });

writer.pipe(fs.createWriteStream(csvFilename));

writer.write({

header1: 'TIME',

header2: 'SEQUENCE NUMBER',

header3: 'PROCESSING TIME'

});

writer.end();

}

if (preLedgerSeqNumber != ledgerSeqNumber) {

// Append some data to CSV the file

if (ledgerLastUpdate != '') {

// var date1 = new Date(ledgerTime);

var date2 = new Date(ledgerLastUpdate);

var Difference\_In\_Time = ledgerTime.getTime() - date2.getTime();

console.log(ledgerTime, date2, Difference\_In\_Time);

}

ledgerLastUpdate = ledgerTime;

const ledgerDuration = Difference\_In\_Time;

writer = csvWriter({ sendHeaders: false });

writer.pipe(fs.createWriteStream(csvFilename, { flags: 'a' }));

writer.write({

header1: ledgerTime,

header2: ledgerSeqNumber,

header3: ledgerDuration

});

writer.end();

preLedgerSeqNumber = ledgerSeqNumber;

}

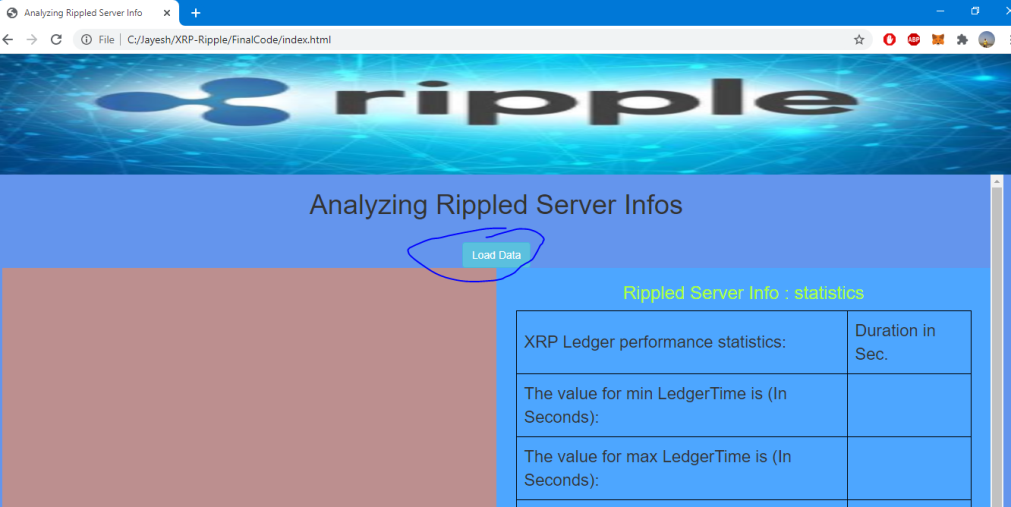
})

* 1. Periodically call the main() function to fetch the Server\_info from Rippled server.
     + Below section of GetServerInfo.js will show the details to periodically (every second) call the API.

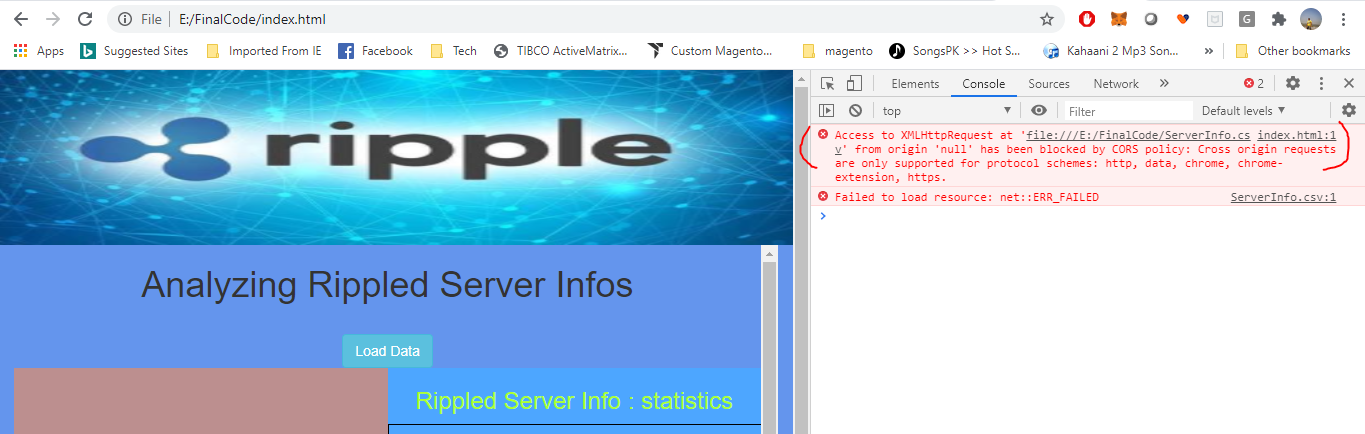
var update\_loop = setInterval(Main, 1000);

Main();

1. UI Implementation Using HTML page :
   1. This implementation is to read the information from ServerInfo.csv file.
      * When we click the Load button on screen, all the data in ServerInfo.csv file will be available on html page.



* 1. Displaying all the entries in the CSV file.
     + In case when you click on load data, if details are no getting populated, this is because of below error.



There is couple of solution for this issue, but we can solve this issue using below setting based on your browser.

**Change local files security policy**

**Firefox:**

* Go to about:config (type the address in the Firefox navigation bar)
* Find security.fileuri.strict\_origin\_policy parameter
* Set it to false

**Chrome:**

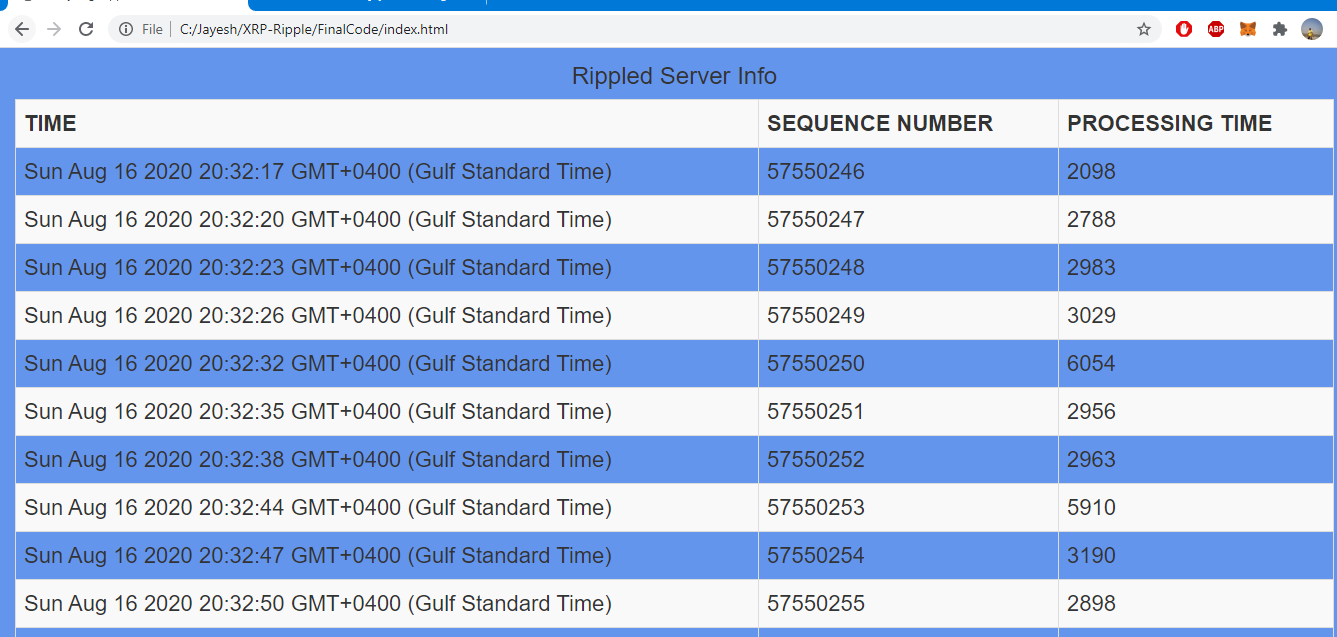
* Close all running Chrome instances first. The important word here is ‘all’. On Windows, you may check for Chrome instances using the Windows Task Manager. Alternatively, if you see a Chrome icon in the system tray, then you may open its context menu and click ‘Exit’. This should close all Chrome instances.
* Start a command prompt / terminal. On Windows, type Command Prompt in the Search box, and then, in the list of results, double-click Command Prompt.
* Find the Chrome executable folder. On Windows, search for the file chrome.exe in a Navigator windows.
* In the command prompt terminal, start the Chrome executable with a command line flag: path\_to\_chrome/chrome --allow-file-access-from-files

chrome --allow-file-access-from-files

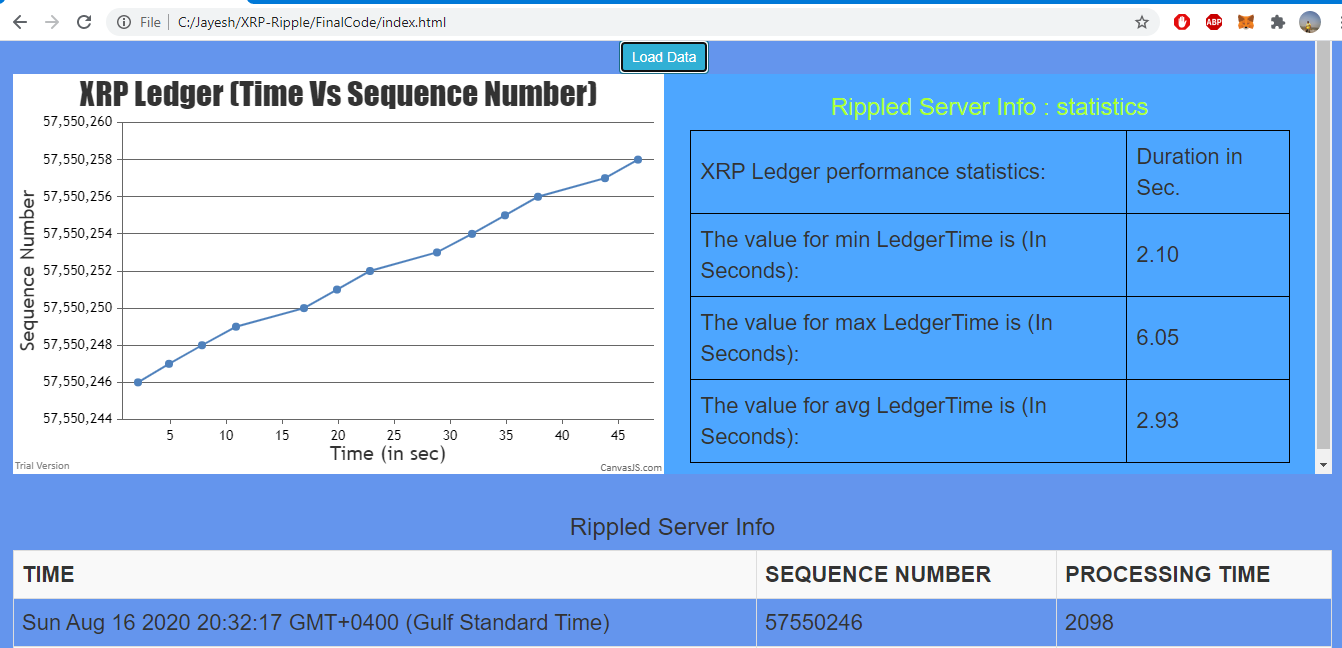
**Safari**

* Enable the develop menu using the preferences panel, under Advanced → "Show develop menu in menu bar".
* Then from the Safari "Develop" menu, select "Disable local file restrictions".
* If you are editing & debugging using Safari, it is advisable to also select the "Disable caches" option in the same menu.

Above setting will solve the issue and data will be loaded without any issue.



* 1. Drawing the graph to plot time duration (x-axis) and sequence number (y-axis).



* 1. Calculating the Min , Max and Avg XRP ledger performance statistics.

***Answer to the question on solution***

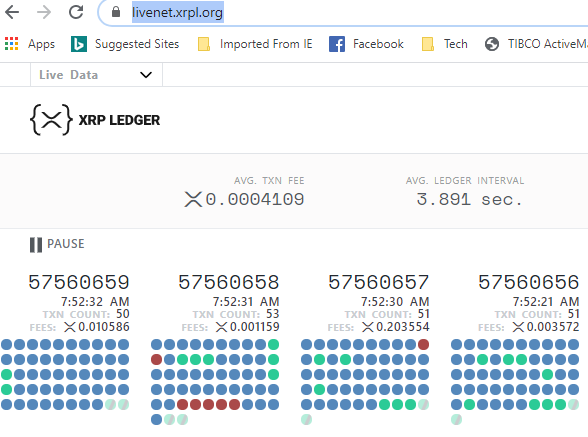
* How does your script work?

Answer: As I explained in the above section of the document.

* + - GetServerInfo.js javascript is used fetch the details for the Server\_info method.
    - Creating the csv file(ServerInfo.csv) and inserting the require details to plot the graph like ledger time, sequence number and calculated duration as well for finding the min, max and average time.
    - Index.html file will load the data from csv file(ServerInfo.csv) and display the data on screen.
    - Plot the graph using the data available in csv file.
    - Enhanced my script to calculate the min, max, and average time that it took for a new ledger to be validated during the span of time captured.
    - Calculated details are displayed on the UI.
    - Every time you click on Load button, html will load the latest data from .csv file and display the data onscreen , create graph , calculate and display the min, max, and average time that it took for a new ledger to be validated during the span of time captured.
* How did you decide on your polling interval?

Answer:

* + - My polling interval = 1 sec (1000 ms).
    - I decided polling interval based on how frequent the new ledger is validated as per the details available in Server\_Info method as well as detail displayed on <https://livenet.xrpl.org/> .



* + - So if polling interval is 1 second then will can capture all the sequence numbers.
    - In case we will get the same record when we are polling, then we can avoid the detail based on comparing the sequence number with previous sequence number.
    - So keeping polling interval 1 sec will be fine, as we are making sure that we are capturing all the sequence numbers of new validated ledger.
* What do the results tell you?

Answer:

* + - Rippled server is very quick in validating the new XRP ledger.
    - Server\_Info method and above implementation details dictate below facts :

AVG. LEDGER INTERVAL ~= 3 sec.

MIN. LEDGER INTERVAL ~= 1 sec.

MAX. LEDGER INTERVAL ~= 6 Sec

* What might explain the variation in time between new ledgers?

**Answer**: Variation in time between new ledgers is influenced by below factors:

* + - **Consensus Rules:** The primary role of consensus is for participants in the process to agree on which transactions are to be processed as a group to resolve the double spend problem. Every participant’s top priority is **correctness**. They must first enforce the rules to be sure nothing violates the integrity of the shared ledger. However, every honest participant’s second priority is **agreement**.
    - A **consensus round** is an attempt to agree on a group of transactions so they can be processed. A consensus round starts with each participant who wishes to do so taking an initial position. This is the set of valid transactions they have seen.
    - Handles Consensus Failure is also add time the time between new ledger.

**Bonus question #1:** Enhance your script to calculate the min, max, and average time that it took for a new ledger to be validated during the span of time captured.

**Answer**: As I explained in the above section of the document.

* + - I enhanced my script to calculate the min, max, and average time that it took for a new ledger to be validated during the span of time captured.
    - I calculated the duration during inserting the details into csv file , whenever I am loading the data on HTML page calculating the min, max, and average time and displaying the same.

**Bonus question #2:** There are some other (better) ways that you could use the rippled API to find how long each ledger took to close/validate. Using the API documentation, find and describe one of these methods (you don’t need to actually implement it).

### Answer: We can implement “[ledger](https://xrpl.org/ledger.html)” Method as well to implement the given problem statement. As it retrieves information about the public ledger.

### As per the response it has all the detail require :

### {

### "id": 14,

### "result": {

### "ledger": {

### "accepted": true,

### "account\_hash": "2227975DBC545C6E6920FF68C62F659C7D874D8578E8D7D2DD53D5F072A8D927",

### "close\_flags": 0,

### "close\_time": 650954550,

### "close\_time\_human": "2020-Aug-17 04:42:30.000000000 UTC",

### "close\_time\_resolution": 10,

### "closed": true,

### "hash": "DE5E95A3468FDCA34BE18DA8C9F6DC48E9E1B4733DDEFC8BF3907186F1A3A091",

### "ledger\_hash": "DE5E95A3468FDCA34BE18DA8C9F6DC48E9E1B4733DDEFC8BF3907186F1A3A091",

### "ledger\_index": "57561424",

### "parent\_close\_time": 650954542,

### "parent\_hash": "9F71609E8EB194904EB1B6C1EE365B1E73F4788C30E843221994F30E0BAB5951",

### "seqNum": "57561424",

### "totalCoins": "99990896349679623",

### "total\_coins": "99990896349679623",

### "transaction\_hash": "7252BBADC52C48871BF90AF4D94B1512B20DCA461C2350419899BE6A81489EED"

### },

### "ledger\_hash": "DE5E95A3468FDCA34BE18DA8C9F6DC48E9E1B4733DDEFC8BF3907186F1A3A091",

### "ledger\_index": 57561424,

### "validated": true

### },

### "status": "success",

### "type": "response"

### }

### Extra implementation Details: Same problem can also be solved using shell script.

### Step 1: Execute below shell script and create .dat file.

### #! /bin/sh

### # Shell script to call the rippled server\_info API to get the ledger related info

### # Created By : Jayesh Jain

### # Date 16th Aug 2020

### # curl command is ued to call the api and fetch the ledger time and validated ledger sequence number and save the details in finalvalues.dat file

### curl -X POST -H 'Content-type:application.json' --data "{\"method\":\"server\_info\",\"params\":[{\"api\_version\":1}]}" http://s1.ripple.com:51234 | jq --raw-output ".result.info.time, .result.info.validated\_ledger.seq" | sed -z '0,/\n/s// /' >> finalvalues.dat

### Step 2: Execute watch command to run the shell script periodically:

### $ watch ‘filelocation’

### Step 3: Plot the graph in Linux using gnuplot , enhance the graph with label and details by loading below file in gnuplot.

### set title " XRP Ledger Analysis"

### set xlabel "Time"

### set ylabel "Sequence Number"

### plot "finalvalues.dat" with lines