Introduction to Web Science

Assignment 2

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Koblenz-Landau

Submission until: November 9, 2016, 10:00 a.m.
Tutorial on: November 11th, 2016, 12:00 p.m.

The main objective of this assignment is for you to use different tools with which you can understand the network that you are connected to or you are connecting to in a better sense. These tasks are not always specific to "Introduction to Web Science". For all the assignment questions that require you to write a code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

Group Name : Yankee
Candidates :

1. Sabin Bhattarai - 216203590
   sbhattarai@uni-koblenz.de
2. Biplov K.C. - 216203865
   biplov@uni-koblenz.de

1 IP Packet (5 Points)

Consider the IPv4 packet that is received as:

`4500 062A 42A1 8001 4210 XXXX C0A8 0001 C0A8 0003`

Consider `XXXX` to be the check sum field that needs to be sent with the packet.

Please provide a step-by-step process for calculating the ”Check Sum”.

Answer 1

Let us take the following IPV4 packet with xxxx being the checksum as underlined

`4500 062A 42A1 8001 4210 XXXX C0A8 0001 C0A8 0003`

As we know the values are in hexadecimal. To calculate the checksum, we first calculate the sum of each byte value within the packet, skipping only the checksum field.

`4500 + 062A + 42A1 + 8001 + 4210 + C0A8 + 0001 + C0A8 + 0003`
= 2D130 (equivalent to 184,624 in decimal)

Next, we convert the value 2D130 to binary:
`0010 1101 0001 0011 0000`

The first 4 bits act as a carry and is added to the rest
`0010 + (1101 0001 0011 0000) = 1101 0001 0011 0010`

Next, we get the ones complement i.e. we flip every bits, to obtain the checksum:
`1101 0001 0011 0010`
becomes:
`0010 1110 1100 1101`
This is equal to 2ECD in hexadecimal which is our checksum.

Testing

We add in similar way including the checksum this time.

`4500 + 062A + 42A1 + 8001 + 4210 + C0A8 + 0001 + 2ECD+ C0A8 + 0003`
= 2fffd

Adding the carry bits: `fffd + 2` = ffff

Now, Getting the ones complement i.e. we flip every bits, gives us 0000 which tells there was no error detected.

2 Routing Algorithm (10 Points)

UPDATE. The bold fonted numbers have been updated on Monday Nov. 7th. (If you already have done so feel free to use the old numbers. But the solution with the old version will be more complex than the solution with the updated numbers.)

You have seen how routing tables can be used to see how the packets are transferred across different networks. Using the routing tables below of Router 1, 2 and 3:

1. Draw the network `[6 points]`
2. Find the shortest path of sending information from 67.68.2.10 network to 25.30.3.13 network `[4 points]`

Table 1: Router 1

Destination	Next Hop	Interface
67.0.0.0	67.68.3.1	eth 0
62.0.0.0	62.4.31.7	eth 1
88.0.0.0	88.4.32.6	eth 2
141.**71**.0.0	141.**71**.20.1	eth 3
26.0.0.0	141.71.26.3	eth 3
156.3.0.0	141.71.26.3	eth 3
205.**30.7**.0	141.71.26.3	eth 3
25.0.0.0	88.6.32.1	eth 2
121.0.0.0	88.6.32.1	eth 2

Table 2: Router 2

Destination	Next Hop	Interface
141.**71**.0.0	141.71.26.3	eth 3
205.**30.7**.0	205.**30.7**.1	eth 0
26.0.0.0	26.3.2.1	eth 2
156.**3**.0.0	156.3.0.6	eth 1
67.0.0.0	141.**71**.20.1	eth 3
62.0.0.0	141.**71**.20.1	eth 3
88.0.0.0	141.**71**.20.1	eth 3
25.0.0.0	205.30.7.2	eth 0
121.0.0.0	205.30.7.2	eth 0

Table 3: Router 3

Destination	Next Hop	Interface
205.**30.7**.0	205.30.7.2	eth 0
88.0.0.0	88.6.32.1	eth 1
25.0.0.0	25.30.1.2	eth 2
121.0.0.0	121.0.3.1	eth 3
156.**3**.0.0	205.**30**.7.1	eth 0
26.0.0.0	205.**30**.7.1	eth 0
141.0.0.0	205.**30**.7.1	eth 0
67.0.0.0	88.4.32.6	eth 1
62.0.0.0	88.4.32.6	eth 1

Answer 2.1

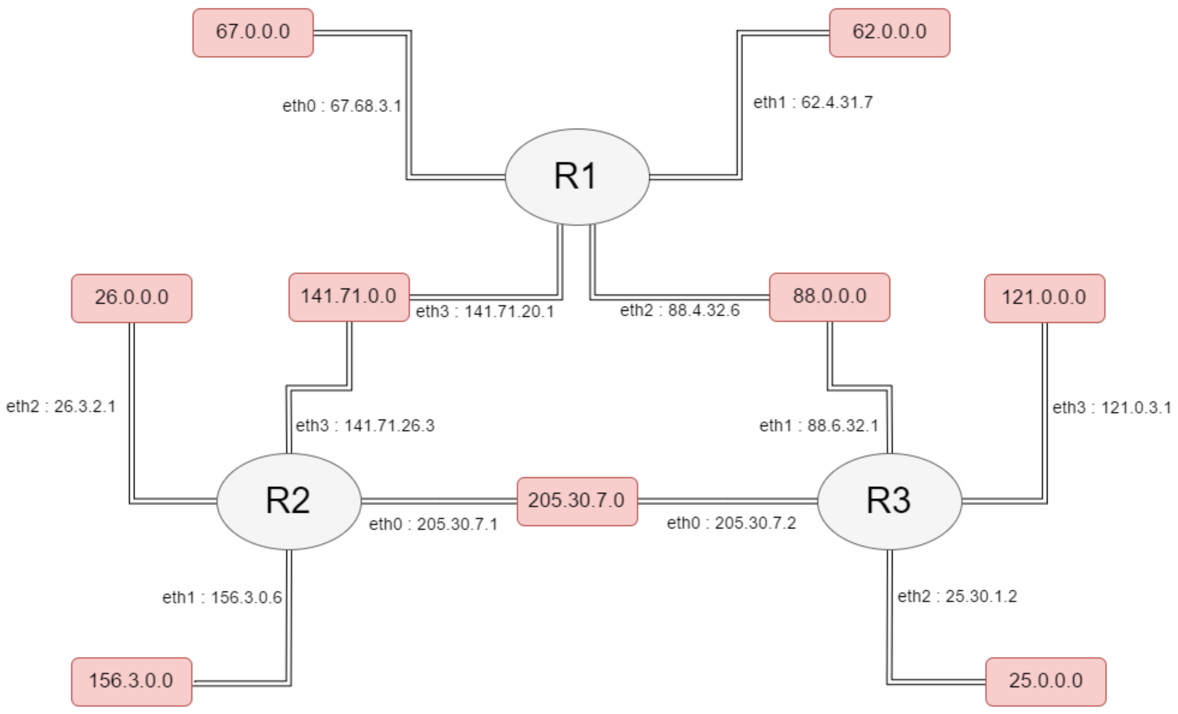


Figure 1: Network Diagram

Answer 2.2

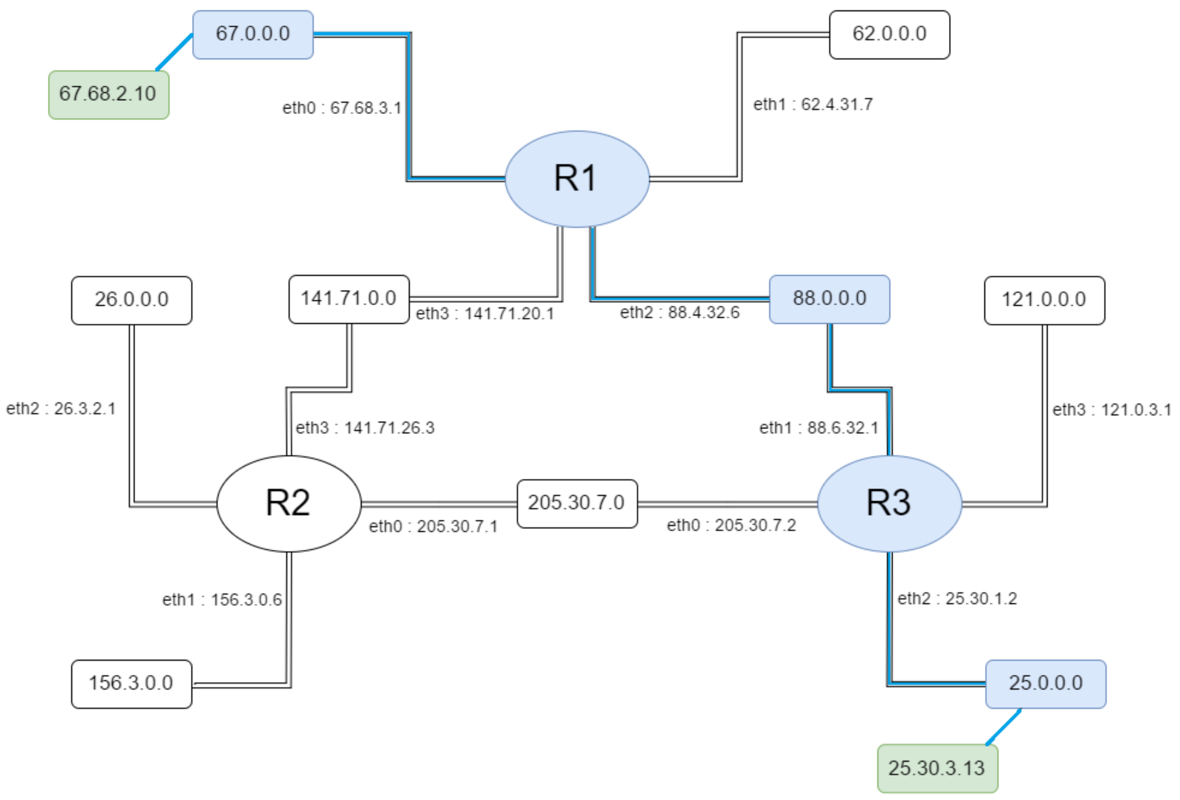


Figure 2: Shortest path to send information from 67.68.2.10 to 25.30.3.13

From the above Figure 2, we can deduce that shortest path from 67.68.2.10 to 25.30.3.13 is : Next hop to Router 1 via network card (IP 67.68.3.1 in eth0). Then Router 1 redirects through its network card (IP : 88.4.32.6 in eth 2) to Router 3 's network card (IP:88.6.3.2.1 in eth1) and redirects via Router 3's network card (IP: 25.30.1.2 eth2) to the destination IP: 25.30.3.13.

3 Sliding Window Protocol (10 Points)

Sliding window algorithm, which allows a sender to have more than one unacknowledged packet "in flight" at a time, improves network throughput.

Let us consider you have 2 Wide Area Networks. One with a bandwidth of 10 Mbps (Delay of 20 ms) and the other with 1 Mbps (Delay of 30 ms) . If a packet is considered to be of size 10kb. Calculate the window size of number of packets necessary for Sliding Window Protocol. [`5 points`]

Since you now understand the concept of Window Size for Sliding Window Protocol and how to calculate it, consider a window size of 3 packets and you have 7 packets to send. Draw the process of `Selective Repeat Sliding Window Protocol` where in the 3rd packet from the sender is lost while transmission. Show diagrammatically how the system reacts when a packet is not received and how it recuperates from that scenario. [`5 points`]

Answer 3 Part A

General formula

Let us assume the transmission delay :

$$T_t$$

Let us assume the propagation delay :

$$T_p$$

That means we have

$$T_t + 2 * T_p$$

seconds until we can have space in the buffer. Because we will have received an acknowledgment of at least 1 packet

We Know

In Tt seconds we can have 1 complete packet in the network
In 1 second we can have 1/Tt packets
In

$$T_t + 2 * T_p$$

seconds we can have

$$1/T_t * (2T_p/T_t)$$

i.e.

$$1 + (2T_p/T_t)packets ---------- equation1$$

which gives us the Window size

Now,

1st Case : Bandwidth of 10 Mbps and delay 20ms

$$T_t = TotalPacketSize/Bandwidth$$

$$= (10 * 10^3)bits/(10 * 10^6)bitsPersec$$

$$= 10^-3sec$$

Also,

$$T_p = 20ms$$

$$= 20 * 10^-3sec$$

Thus, using the equation 1 from General formula and substituting Tt and Tp,

Window Size =

$$1 + (2 * 20 * 10^-3)\ (10^-3)$$
$$= 41packets$$

2nd Case : Bandwidth of 1 Mbps and delay 30ms

$$T_t = TotalPacketSize/Bandwidth$$

$$= (10 * 10^3)bits/(1 * 10^6)bitsPersec$$

$$= 10^-2sec$$

Also,

$$T_p = 30ms$$

$$= 30 * 10^-3sec$$

Thus, using the equation 1 from General formula and substituting Tt and Tp,

Window Size =

$$1 + (2 * 30 * 10^-3)\ (10^-2)$$
$$= 1 + 6$$
$$= 7packets$$

Answer 3 Part B

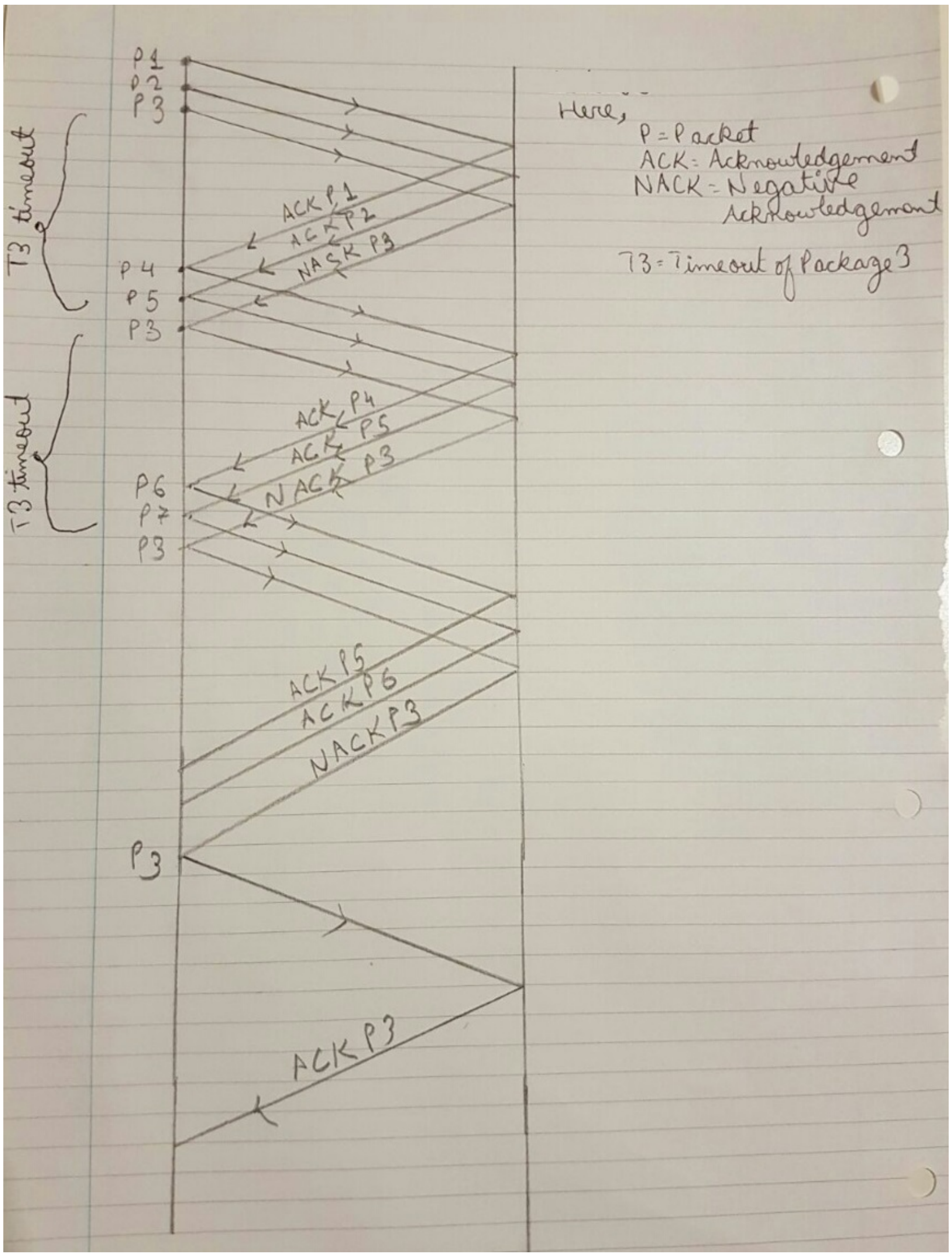


Figure 3: Selective Repeat Sliding Window Protocol

As Shown in the above diagram,

Since window size is set to 3, Packet 1 2 and 3 are sent initially in this order. For each packet sent there will be a time out set to it. Firstly, ACK1 ACK2 i.e acknowledgment 1 and 2 is received at which point another packet 4 and packet 5 is sent. However the ACK for 3 is not received and lets assume time out set to it is not over yet. At this instance according to the Selective Repeat Sliding Window Protocol a Negative acknowledgment is sent (This is because the time out could be longer and thus optimization was done with NACK sent by receiver). The packet 3 is sent again.

Right now we have packet 4 5 and 3 sent to receiver simultaneously. Again ACK for packet 4 5 is received and timeout for packet 3 occurred. This means we send packet 6 7 and 3 again.

Again ACK for 6 7 is received and time out for 3 occurred.

Finally packet 3 is sent and an ACK is received. Thus all packets are received by the Receiver.

4 TCP Client Server (10 Points)

Use the information from the socket documentation and create: [`4 points`]

1. a simple TCP Server that listens to a
2. Client

Note: Please use port `8080` for communication on `localhost` for client server communication.

Given below are the following points that your client and server must perform: [`6 points`]

1. The *Client* side asks the user to input their name, age & *matrikelnummer* which is then sent to the server all together.
2. Develop a protocol for sending these three information and subsequently receiving each of the information in three different lines as mentioned in the below format. Provide reasons for the protocol you implemented.
3. Format the output in a readable format as:

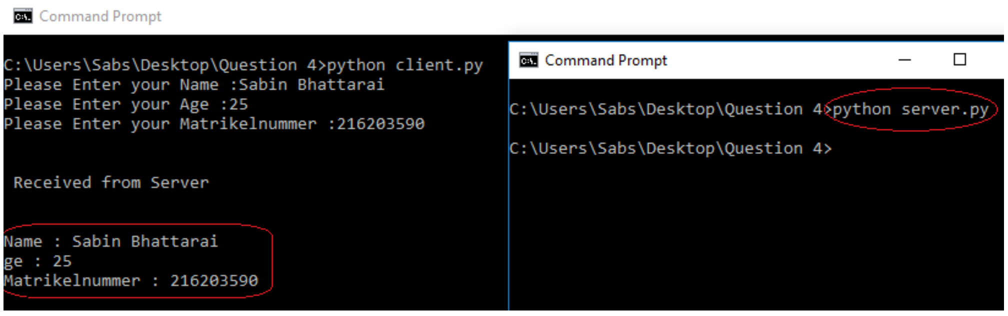
   ```
   Name: Korok Sengupta;
   Age: 29;
   Matrikelnummer: 21223ert56
   ```

Provide a snapshot of the results along with the code.

Answer 4

The protocol we have used in sending three information and subsequently receiving each of the information involves JSON over TCP. We create a dictionary with keys being Name, Age , Matrikelnummber mapped to its result obtained from user input. This is converted to JSON which is then manipulated by server and client side and finally returned to Client by the server as expected result. We then process the obtained result in three lines and display accordingly from client side.

Output



```
C:\Users\Sabs\Desktop\Question 4>python client.py
Please Enter your Name :Sabin Bhattarai
Please Enter your Age :25
Please Enter your Matrikelnummer :216203590


 Received from Server


Name : Sabin Bhattarai
ge : 25
Matrikelnummer : 216203590
```

```
C:\Users\Sabs\Desktop\Question 4>python server.py

C:\Users\Sabs\Desktop\Question 4>
```

Figure 4: Result for Server Client communication

Class Library for Client and Server

```
import socket


# Class for Server
class Server(object):

  backlog = 5
  client = None

  def __init__(self, host, port):
    self.socket = socket.socket()
    self.socket.bind((host, port))
    self.socket.listen(self.backlog)

  def __del__(self):
    self.close()

  def accept(self):
    # if a client is already connected, disconnect it
    if self.client:
      print('3')
      self.client.close()
    self.client, self.client_addr = self.socket.accept()
    return self

  def send(self, data):
    if not self.client:
      raise Exception('Cannot send data, no client is connected')
    self.client.send(data)
    return self

  def recv(self):
    if not self.client:
      raise Exception('Cannot receive data, no client is connected')
    return self.client.recv(30000)

  def close(self):
    if self.client:
      self.client.close()
      self.client = None
    if self.socket:
      self.socket.close()
      self.socket = None



#Class for Client
class Client(object):

  socket = None

  def __del__(self):
    self.close()

  def connect(self, host, port):
    self.socket = socket.socket()
    self.socket.connect((host, port))
    return self

  def send(self, data):
    if not self.socket:
      raise Exception('You have to connect first before sending data')
    self.socket.send(data)
    return self

  def recv(self):
    if not self.socket:
      raise Exception('You have to connect first before receiving data')
    return self.socket.recv(30000)

  def recv_and_close(self):
    data = self.recv()
    self.close()
    return data

  def close(self):
    if self.socket:
      self.socket.close()
      self.socket = None
```

Server Side Code

```python
from jsonSocket import Server
import json

host = 'localhost'
port = 8080

server = Server(host, port)  # Server Class is obtained from jsonSocket Library
server.accept()
#Receives data from client
data = server.recv()
data = json.loads(data.decode())

#Editing the obtained Json formatted data from client
formattedText = "\n"
formattedText = formattedText + 'Name' + " : "
+ data['Name'] +"\n" + 'Age'  +" : "
+ data['Age'] +"\n" + 'Matrikelnummer' + " : "
+ data['Matrikelnummer'] +"\n";

# Please note the following code could be done for hard coding removal. However
# with it the order was not achieved while printing

#formattedText = ""
#for key, value in data.items():
#     formattedText = formattedText + key + ":" + value +"\n";


# Now we encode the string and send to client for printing
server.send(formattedText.encode()).close()
```

Client Side Code

```
from jsonSocket import Client
import json

host = 'localhost'
port = 8080

# Client code:
client = Client()  # Client Class is obtained from jsonSocket Library
data ={}
name = input("Please Enter your Name :")
data['Name']= name
age = input("Please Enter your Age :")
data['Age']= age
Matrikelnummer = input("Please Enter your Matrikelnummer :")
data['Matrikelnummer']= Matrikelnummer

#collects data and converts to Json and sends to the Server
client.connect(host, port).send(json.dumps(data).encode())

#Gets response from server
response = client.recv()

print ("\n \n Received from Server \n \n" , response.decode())
client.close()
```

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment2/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use `UTF-8` as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent indentation.
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

LaTeX

Currently the code can only be build using LuaLaTeX, so make sure you have that installed. If on Overleaf, go to settings and change the LaTeXengine to `LuaLaTeX` in case you encounter any error