Suzuki–Kasami Algorithm

GitHub Link

https://github.com/shobhitka/wilp-sem1/tree/main/DC

Demo video can be seen in the above link as well.

Test Environment

1. Code has been tested using Python 3.8.10, but should work on all versions 3.x. Please do not use Python2. Python2 will not work
2. The test operating system is Ubuntu Linux 20.04. Should work on Windows and MAC but not tested
3. Main modules used by the code are all in-built python modules. No other external modules are utilized
4. Below is the list of modules used. All should be ideally available with default python 3 installation

```
import queue
import sys
import threading
import signal
import socket
from time import sleep
from random import randint
```

5. Communication channel implementation has been done from scratch using Python sockets

Code organization

Filename	Description
algo.py	The main Driver code for testing

channel.py	Communication channel implementation
node.py	Main Site node implementation. Implements the complete algorithm
Readme.md	This Readme file
system.cfg	System configuration input file. Discussed in next section

System definition

The site configurations are defined in the **system.cfg**. It should define the IDs of all the sites in the system with their IP address and port.

Refer to below as a test sample input

```
# System Configuration defining all sites in the distributed system
# Site ID, Site IP Address, Site Port
1, 127.0.0.1, 2022
2, 127.0.0.1, 2023
3, 127.0.0.1, 2024
4, 127.0.0.1, 2025
```

The above define 4 sites, S(i) with i = 1..4

Site IDs should all be sequential starting from 1

Basic steps

1. Run everything from inside the package folder. Ensure that system.cfg is available at the same level as the python files
2. Start all the sites in different terminals using following command

```
python3 algo.py -i <site_id>
```

3. If successfully initialized the application will drop in a command line shell for the site as given below
4. We can type "help" to see what commands are supported

```
❯ python3 algo.py -i 1
Parsed system configuration.
Starting message receiver thread at site id: 1
Initialized params for channel: (1 -> 2)
Initialized params for channel: (1 -> 3)
Initialized params for channel: (1 -> 4)
Site ready: 1, HAS_TOKEN: True
(site:1)$
(site:1)$
(site:1)$ help
help: This help information
exit: quit the application
dump: dump all debug info
enter: enter critical section
(site:1)$
```

5. **Note that Site 1 always starts with the token**
6. We can then type "enter" command to try to enter CS at the site prompt.
7. We can type "enter" on multiple sites one after the other if needed and algorithm will take care of token management
8. When a site enters critical section, it does a random sleep between 5s to 10s and the processing is indicated by increasing number of "." on the terminal
9. All relevant messages and state of RN and Token [LN, Q] will be printed on the console at every state change
10. We can exit the application by "exit" command or by using "CTRL+C"
11. “dump” command can be used anytime to dump the state of LN, RN and the Token Queue
12. We can continue any number of iterations with the running set of applications. No need to restart them. But in case you restart one, remember to restart all
13. Algorithm can be tested with any order of sites trying to enter CS. Feel free to play with the sequence

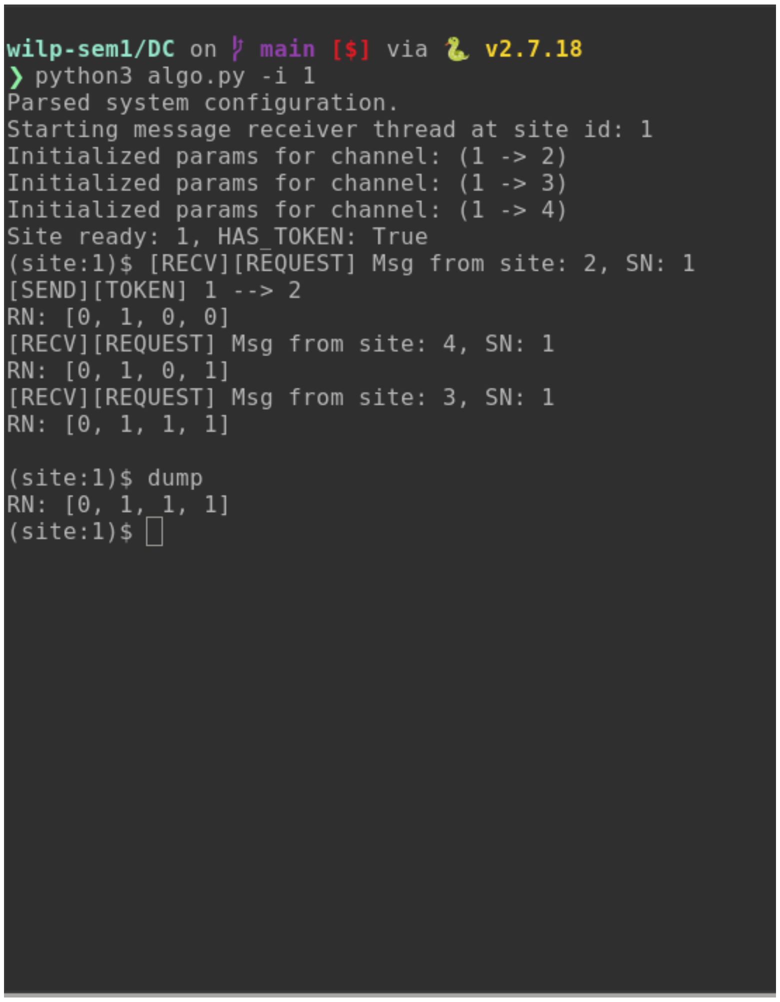
Sample video demo of the application usage

Refer to the MP4 video file in the submitted package.

Sample screenshot at end of one iteration

1. Type "enter" for Site 2 to enter CS
2. Type "enter" on site 4 to try to enter CS while site 2 is in CS
3. Type "enter" on site 3 to try to enter CS while Site 2 is still in CS
4. See the sequence of messages and RN, Ln, Q states in the below screenshot

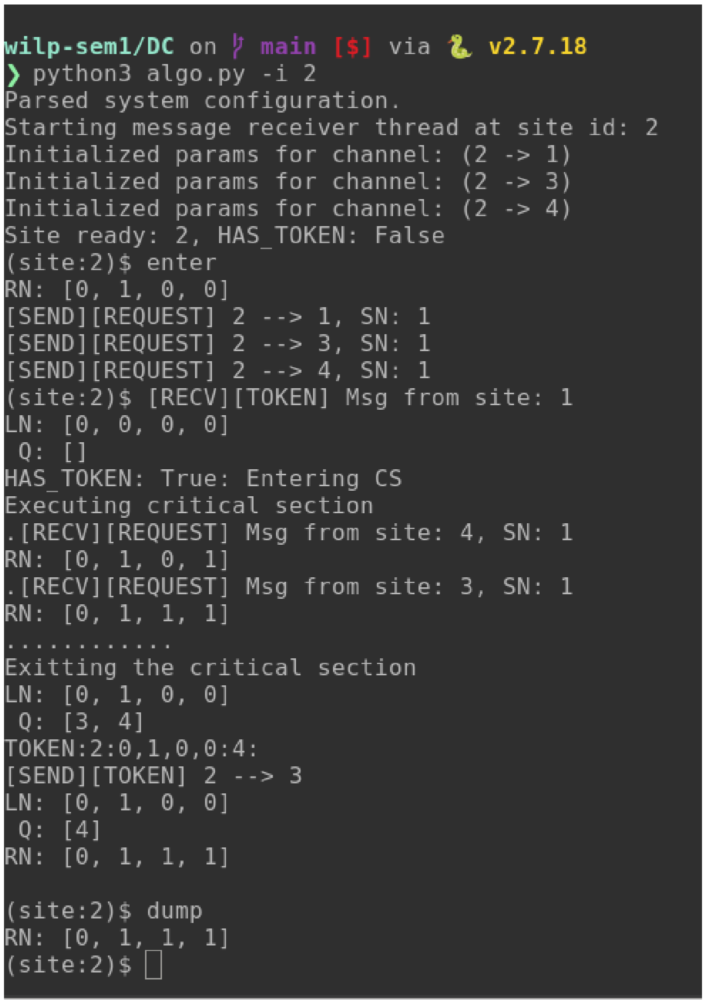
SITE 1



```
wilp-sem1/DC on ᚠ main [$] via 🐍 v2.7.18
❯ python3 algo.py -i 1
Parsed system configuration.
Starting message receiver thread at site id: 1
Initialized params for channel: (1 -> 2)
Initialized params for channel: (1 -> 3)
Initialized params for channel: (1 -> 4)
Site ready: 1, HAS_TOKEN: True
(site:1)$ [RECV][REQUEST] Msg from site: 2, SN: 1
[SEND][TOKEN] 1 --> 2
RN: [0, 1, 0, 0]
[RECV][REQUEST] Msg from site: 4, SN: 1
RN: [0, 1, 0, 1]
[RECV][REQUEST] Msg from site: 3, SN: 1
RN: [0, 1, 1, 1]

(site:1)$ dump
RN: [0, 1, 1, 1]
(site:1)$
```

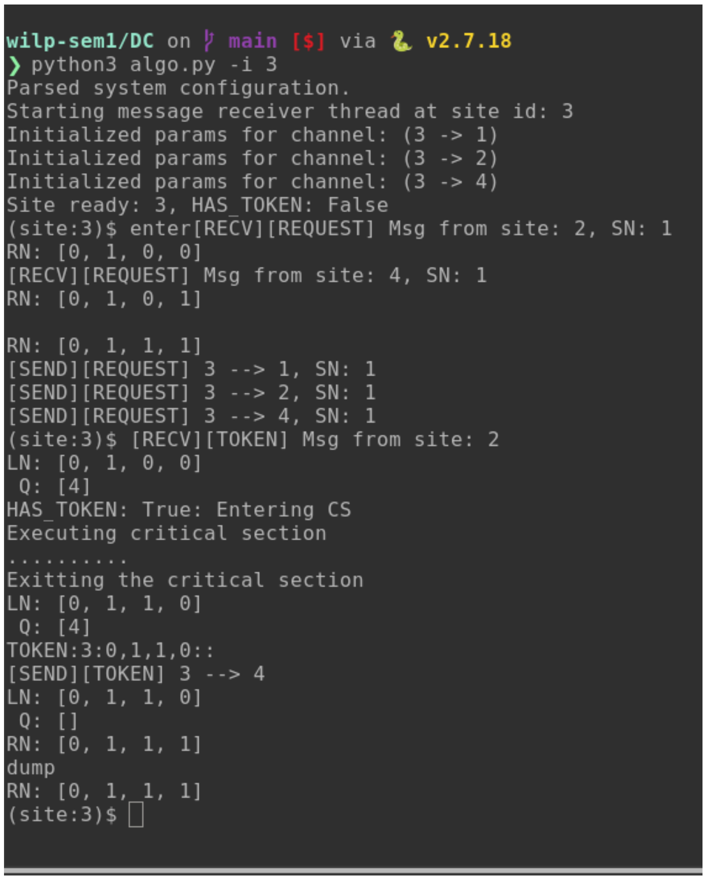
SITE 2



```
wilp-sem1/DC on ᚠ main [$] via 🐍 v2.7.18
❯ python3 algo.py -i 2
Parsed system configuration.
Starting message receiver thread at site id: 2
Initialized params for channel: (2 -> 1)
Initialized params for channel: (2 -> 3)
Initialized params for channel: (2 -> 4)
Site ready: 2, HAS_TOKEN: False
(site:2)$ enter
RN: [0, 1, 0, 0]
[SEND][REQUEST] 2 --> 1, SN: 1
[SEND][REQUEST] 2 --> 3, SN: 1
[SEND][REQUEST] 2 --> 4, SN: 1
(site:2)$ [RECV][TOKEN] Msg from site: 1
LN: [0, 0, 0, 0]
 Q: []
HAS_TOKEN: True: Entering CS
Executing critical section
.[RECV][REQUEST] Msg from site: 4, SN: 1
RN: [0, 1, 0, 1]
.[RECV][REQUEST] Msg from site: 3, SN: 1
RN: [0, 1, 1, 1]
............
Exitting the critical section
LN: [0, 1, 0, 0]
 Q: [3, 4]
TOKEN:2:0,1,0,0:4:
[SEND][TOKEN] 2 --> 3
LN: [0, 1, 0, 0]
 Q: [4]
RN: [0, 1, 1, 1]

(site:2)$ dump
RN: [0, 1, 1, 1]
(site:2)$
```

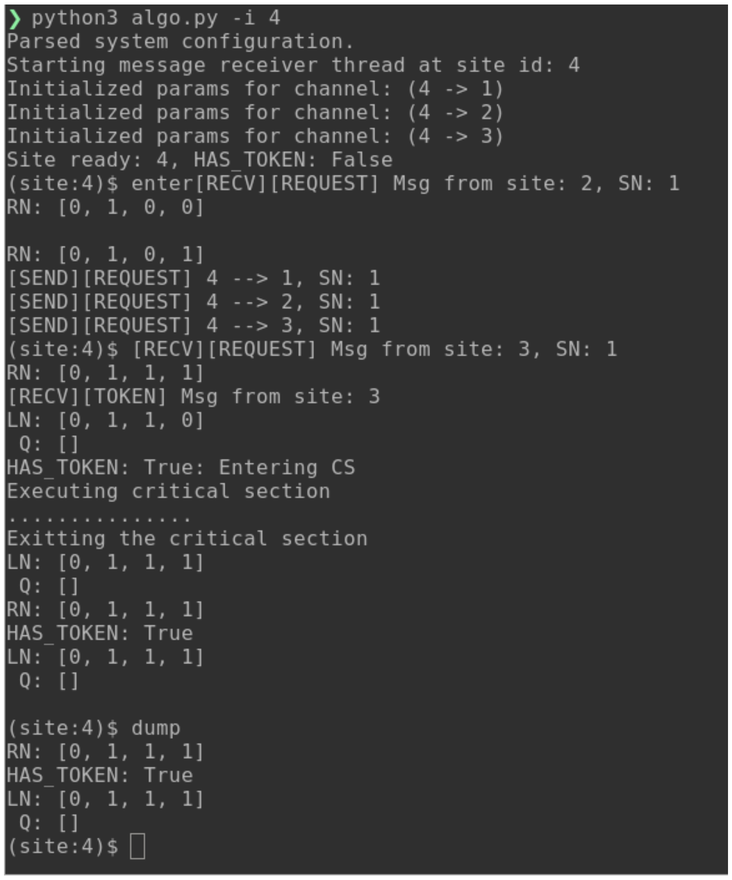
SITE 3



```
wilp-sem1/DC on ᚴ main [$] via 🐍 v2.7.18
❯ python3 algo.py -i 3
Parsed system configuration.
Starting message receiver thread at site id: 3
Initialized params for channel: (3 -> 1)
Initialized params for channel: (3 -> 2)
Initialized params for channel: (3 -> 4)
Site ready: 3, HAS_TOKEN: False
(site:3)$ enter[RECV][REQUEST] Msg from site: 2, SN: 1
RN: [0, 1, 0, 0]
[RECV][REQUEST] Msg from site: 4, SN: 1
RN: [0, 1, 0, 1]

RN: [0, 1, 1, 1]
[SEND][REQUEST] 3 --> 1, SN: 1
[SEND][REQUEST] 3 --> 2, SN: 1
[SEND][REQUEST] 3 --> 4, SN: 1
(site:3)$ [RECV][TOKEN] Msg from site: 2
LN: [0, 1, 0, 0]
 Q: [4]
HAS_TOKEN: True: Entering CS
Executing critical section
..........
Exitting the critical section
LN: [0, 1, 1, 0]
 Q: [4]
TOKEN:3:0,1,1,0::
[SEND][TOKEN] 3 --> 4
LN: [0, 1, 1, 0]
 Q: []
RN: [0, 1, 1, 1]
dump
RN: [0, 1, 1, 1]
(site:3)$ 
```

SITE 4



```
❯ python3 algo.py -i 4
Parsed system configuration.
Starting message receiver thread at site id: 4
Initialized params for channel: (4 -> 1)
Initialized params for channel: (4 -> 2)
Initialized params for channel: (4 -> 3)
Site ready: 4, HAS_TOKEN: False
(site:4)$ enter[RECV][REQUEST] Msg from site: 2, SN: 1
RN: [0, 1, 0, 0]

RN: [0, 1, 0, 1]
[SEND][REQUEST] 4 --> 1, SN: 1
[SEND][REQUEST] 4 --> 2, SN: 1
[SEND][REQUEST] 4 --> 3, SN: 1
(site:4)$ [RECV][REQUEST] Msg from site: 3, SN: 1
RN: [0, 1, 1, 1]
[RECV][TOKEN] Msg from site: 3
LN: [0, 1, 1, 0]
 Q: []
HAS_TOKEN: True: Entering CS
Executing critical section
...............
Exitting the critical section
LN: [0, 1, 1, 1]
 Q: []
RN: [0, 1, 1, 1]
HAS_TOKEN: True
LN: [0, 1, 1, 1]
 Q: []

(site:4)$ dump
RN: [0, 1, 1, 1]
HAS_TOKEN: True
LN: [0, 1, 1, 1]
 Q: []
(site:4)$ 
```