Report by Danil Ginzburg. Lab 8.

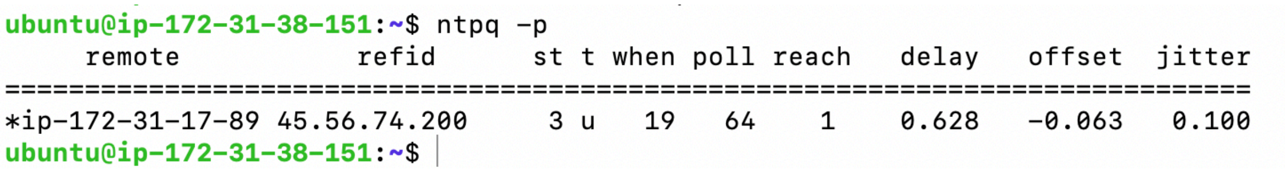
BS17-DS-01

Answers to the questions

Q1: What is a stratum in terms of NTP?

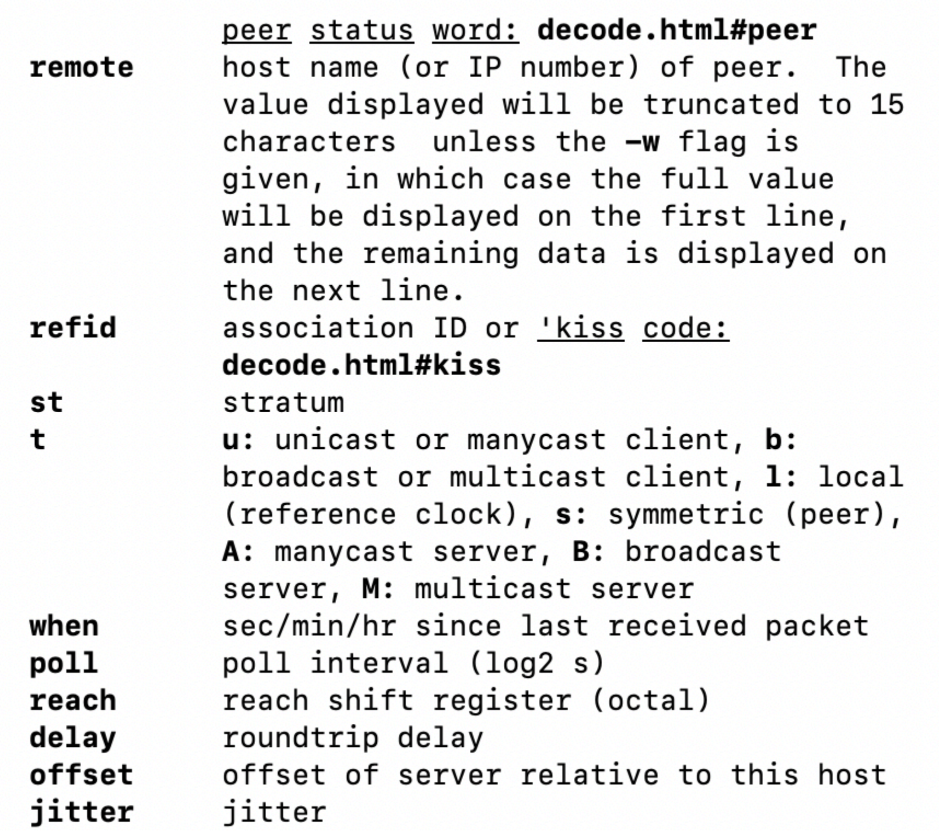
A stratum in terms of NTP is one layer in a hierarchical network of layers that distribute accurate time across a network of devices.

Q2: Provide the output of ntpq -p command and describe the meaning of the following fields: remote , refid , st , t, when, poll, reach, delay, offset , and jitter .



```
ubuntu@ip-172-31-38-151:~$ ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*ip-172-31-17-89 45.56.74.200     3 u   19   64    1    0.628   -0.063   0.100
ubuntu@ip-172-31-38-151:~$
```

The explanation of each field:



```
            peer status word: decode.html#peer
remote      host name (or IP number) of peer.  The
            value displayed will be truncated to 15
            characters  unless the -w flag is
            given, in which case the full value
            will be displayed on the first line,
            and the remaining data is displayed on
            the next line.
refid       association ID or 'kiss code:
            decode.html#kiss
st          stratum
t           u: unicast or manycast client, b:
            broadcast or multicast client, l: local
            (reference clock), s: symmetric (peer),
            A: manycast server, B: broadcast
            server, M: multicast server
when        sec/min/hr since last received packet
poll        poll interval (log2 s)
reach       reach shift register (octal)
delay       roundtrip delay
offset      offset of server relative to this host
jitter      jitter
```

Q3: What are the lacks of using the Lamport's algorithm?

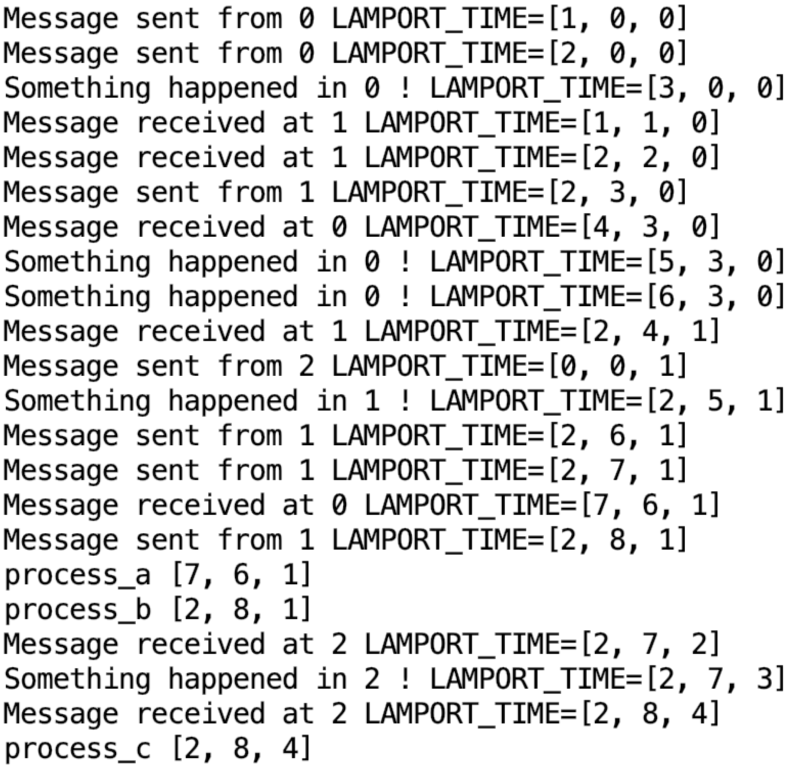
- It can not recognise whether two processes are concurrent or not

- It orders events only partially (in partial order).

The final vector state

This is the final result according to my program:

process a [7, 6, 1]
process b [2, 8, 1]
process c [2, 8, 4]



```
Message sent from 0 LAMPORT_TIME=[1, 0, 0]
Message sent from 0 LAMPORT_TIME=[2, 0, 0]
Something happened in 0 ! LAMPORT_TIME=[3, 0, 0]
Message received at 1 LAMPORT_TIME=[1, 1, 0]
Message received at 1 LAMPORT_TIME=[2, 2, 0]
Message sent from 1 LAMPORT_TIME=[2, 3, 0]
Message received at 0 LAMPORT_TIME=[4, 3, 0]
Something happened in 0 ! LAMPORT_TIME=[5, 3, 0]
Something happened in 0 ! LAMPORT_TIME=[6, 3, 0]
Message received at 1 LAMPORT_TIME=[2, 4, 1]
Message sent from 2 LAMPORT_TIME=[0, 0, 1]
Something happened in 1 ! LAMPORT_TIME=[2, 5, 1]
Message sent from 1 LAMPORT_TIME=[2, 6, 1]
Message sent from 1 LAMPORT_TIME=[2, 7, 1]
Message received at 0 LAMPORT_TIME=[7, 6, 1]
Message sent from 1 LAMPORT_TIME=[2, 8, 1]
process_a [7, 6, 1]
process_b [2, 8, 1]
Message received at 2 LAMPORT_TIME=[2, 7, 2]
Something happened in 2 ! LAMPORT_TIME=[2, 7, 3]
Message received at 2 LAMPORT_TIME=[2, 8, 4]
process_c [2, 8, 4]
```

Source Code

Python realisation is uploaded on github:

https://github.com/thedownhill/Distributed-Systems/blob/master/lab8/logical_clock.py