CS 4410: Operating Systems
Homework 9

- Homework may be done in pairs, or individually. If doing in pairs, **one** of you should upload to gradescope and add your partner to the group assignment in the upper right corner of the screen. (Do **not** just upload the assignment twice or it will be graded twice, which means grading will take longer.)
- The deadline is Fri, Dec 2 at 11:59AM.
- No late submissions will be accepted.
- You must attribute every source used to complete this homework.
- For some of the problems, you will need two integers. Here is the algorithm for computing these integers:
  - If you are working with a partner, let `var` be the lexicographically smaller of the two NetIDs.
  - Let `varInt` be the integral part of `var`. That is, if `var = rst12`, then `varInt = 12`.
  - If `varInt` is a single digit integer, let `varInt = 13 × varInt`
  - Let `Int1` be the first digit of `varInt`
  - Let `Int2` be the second digit of `varInt`
- Assume the storage unit convention: $1\text{G} = 2^{10} \times \text{M} = 2^{20} \times \text{K} = 2^{30}$ (bytes).
- **For all problems that use `Int1` or `Int2`, please write down the parameters (related variables and settings calculated from your NetID) before answering each question.**

1 Semaphore

Due to some historical reason, VWMare currently does not have condition variable implementation in the kernel of the network operating system, which really upsets both of Harold and you. Since it takes a fairly long time to add such feature directly into the kernel, Harold asked you to first implement the condition variable using the existing synchronization primitive, semaphores.

Your first implementation was written as in the following left pseudo-code. It appeared to be pretty much correct until one day you found developers complaint about the unexpected behavior of the new condition variables.

Algorithm 1 Implement CV using semaphores, ver. 1

```
▷ Semaphore($v$): create a semaphore with initial value $v$
▷ acquire($l$): acquire the mutex lock $l$
▷ release($l$): release the mutex lock $l$
function INIT($c, m$)
    $c.m = m$
    $c.s =$ Semaphore(0)
    $c.x =$ Semaphore(1)
    $c.$waiting $= 0$
end function
function WAIT($c$)
    $P(c.x)$
    $c.$waiting++
    $V(c.x)$
    release($c.m$)
    /* (1) */
    $P(c.s)$
    acquire($c.m$)
end function
function SIGNAL($c$)
    $P(c.x)$
    if $c.$waiting $> 0$ then
        $c.$waiting−−
        $V(c.s)$
    end if
    $V(c.x)$
end function
function BROADCAST($c$)
    $P(c.x)$
    while $c.$waiting $> 0$ do
        $c.$waiting−−
        $V(c.s)$
    end while
    $V(c.x)$
end function
```

Algorithm 2 Implement CV using semaphores, ver. 2

```
function INIT($c, m$)
    $c.m = m$
    $c.s =$ Semaphore(0)
    $c.x =$ Semaphore(1)
    $c.h =$ Semaphore(_)
    $c.$waiting $= 0$
end function
function WAIT($c$)
    $P(c.x)$
    $c.$waiting++
    $V(c.x)$
    release($c.m$)
    $P(c.s)$
    __($c.h$)
    acquire($c.m$)
end function
function SIGNAL($c$)
    $P(c.x)$
    if $c.$waiting $> 0$ then
        $c.$waiting−−
        $V(c.s)$
        ____
    end if
    $V(c.x)$
end function
function BROADCAST($c$)
    $P(c.x)$
    for $i$ in $[0, c.$waiting$)$ do
        $V(c.s)$
    end for
    while $c.$waiting $> 0$ do
        $c.$waiting−−
        ____
    end while
    $V(c.x)$
end function
```

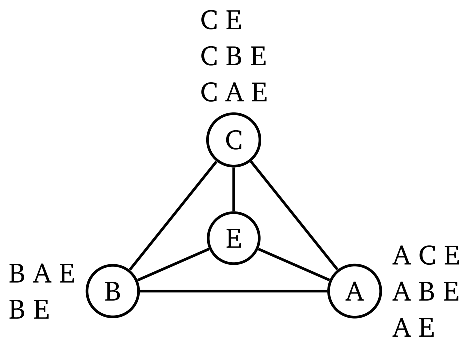
1. Suppose there are only $((\texttt{Int1} \mod 4)+4)$ threads have called `wait(c)` and they are all at position (1) when `broadcast(c)` is called. After the broadcast, how many threads will exit from `wait(c)`? Write down all the possibilities.

2. Following the previous question, if there is another thread which calls `wait(c)` right after `broadcast(c)` is called. How many threads that called `wait(c)` before `broadcast(c)` will exit from `wait(c)`? Write down all the possibilities.

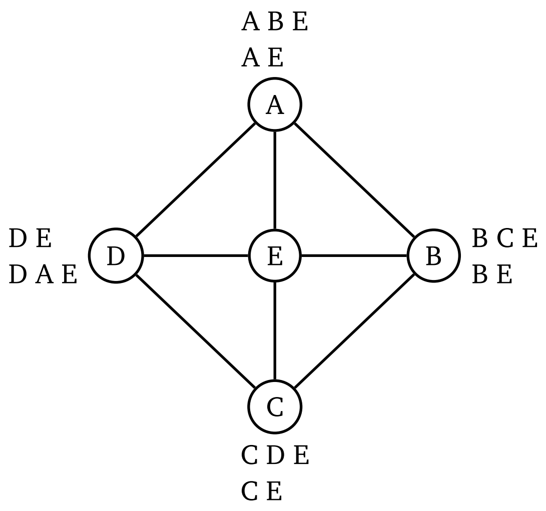
3. Fill in the blanks of the above right pseudo-code to correctly implement a condition variable using semaphores.

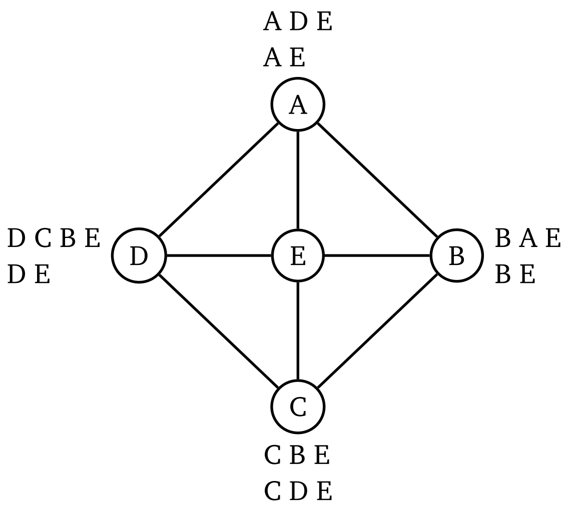
2 Stable Network

One principle for CSMA Co., Ltd. is "Stability matters." You deeply understand all lessons this company has learnt. To better analyze BGP routing, you use an abstract model, Stable Paths Problem (SPP), in which each node keeps a list of preferred paths, from the most (the first in the list) to the least. In a stable solution, given a path assignment for each node, no node can choose a higher-ranked path that is consistent with path assignment for its neighbors.



For example, in the above graph, the solution ACE, BE and CE assigned for A, B and C is stable, while ABE, BE and CBE is unstable because C can choose a higher-ranked path CE.

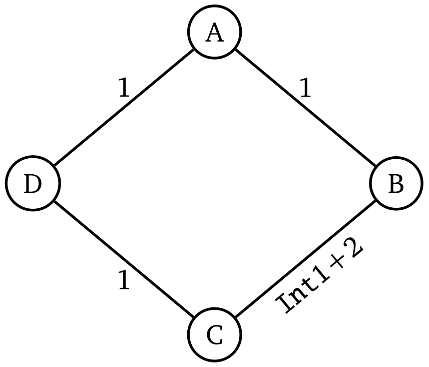




1. Consider the SSP depicted on the left, in which nodes *A*, *B*, *C* and *D* all wants to choose a path to node *E*. Does this SSP have a stable solution? If yes, please indicate the path chosen by each node in your solution. If no, give an example of instability (oscillation).

2. Consider the SSP depicted on the right, in which nodes *A*, *B*, *C* and *D* all wants to choose a path to node *E*. Does this SSP have a stable solution? If yes, please indicate the path chosen by each node in your solution. If no, give an example of instability (oscillation).

3. In a link-state routing protocol, unlike a distance-vector based protocol, a node will only send a message, link-state advertisement identifying the neighbors it has, to its neighbors and pass on the latest link-state advertisement received from neighbors. It calculates the shortest path based on its own knowledge of the whole network state updated by link-state advertisements and makes routing decision. Both the messaging and calculation are done periodically so all the nodes may not necessarily have the same picture of the whole network at any given moment.

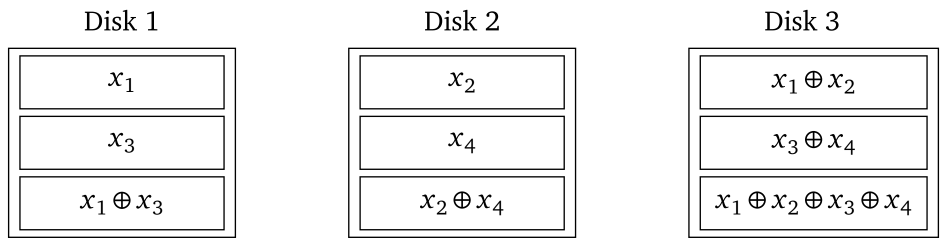


Consider the network depicted above, in which the network runs a link-state routing protocol that calculates the shortest path as the sum of weights of links along the path. The number for each link represents the weight for both directions. Assume at time zero, the link between C and D fails in both directions, immediately after that, nodes A, B and D send packets to C. Which of the nodes could possibly see their packets stuck in a forwarding loop temporarily?

3 New Product

One company tried to sell their RAID system products to VWMare. Being suspicious of the new technology it advertises, Harold wants you to analyze the RAID system that has intra-disk redundancy which can improve the safety. Each disk in this system reserves one parity block for the data blocks in the same disk. With such arrangement, each data block is protected by two parity blocks: one external parity block on a different disk and the other intra-disk parity block on the same disk.

When one block in a disk is lost, it can be recovered first from the intra-disk parity block and then rely on the external parity block on another disk if the intra-disk parity block is also lost.



1. In the arrangement of RAID shown above, how many block failures can the system detect at most for the best case? Write down the example. Detecting the failures means finding out the number and the location of failed blocks.

2. Considering the worst case, how many failures can the system recover at most? Briefly explain the way to recover.