EDA387: Computer Networks - Lab 2.1
Value Discovery in Complete Graphs

Group 5: Haitham Babbili and Olalekan Peter Adare

17th September 2020

Use of Processor Identifiers (PID)

A process identification number (PID) is an identification number that is automatically assigned to each process when it is created on a Unix-like/Linux operating system. This makes the operating system to uniquely identify and keep track of a process, until the process is terminated. A register is a small storage, which maybe read-only, write-only, or both, to hold fast information for the processor. Below is a pseudo-code for this solution:

```
01: Do forever
02:   for i = 0 to n 1
03:   r(i)= write (read s(i+1) mod n && 04:   read s(i 1))
05:   for p(i){r(i)= read s(i+1) mod n && read s(i 1)
06:   secret(i)= r(i){new}    r(i){old}
07: End
```

Based on the assumption that the set of processors are synchronized in time, with P_i being the designated leader. This is to create sort of fair execution. Taking that the processors are activated infinitely often, leading to legitimate configuration eventually. Then, assume that all processors start from individual state 0, with internal local ordering. Also, assuming that the entries in the registries do not change, as well as the secret information. Finally, building on the fact that the process identification numbers do not change for a given process, except it is killed and restarted. The diagram below represents the solution:

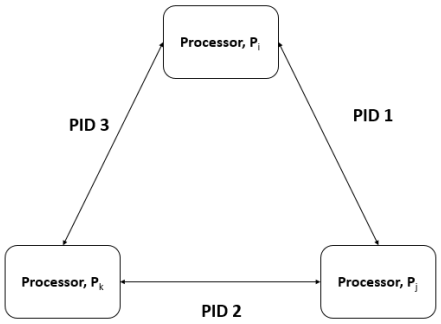

Figure 1:

Given that the number of processors, n, is such that n>2. This means n≥3. Let us consider a case of three (3) processors, P_i, P_j and P_k, each with secret codes C_i, C_j and C_k respectively. Each processor has its r-register and s-register. They can all read and write into their r-registers.P_i does not know the secret code in its s_i. Meanwhile, other processors P_j and P_k can only read the s_i register. Let there be a ring connection of the three (3) processors, with each interconnection representing a bidirectional process assigned a process number. Once the processors are excited, the bidirectional communication of each processor is automatically assigned a PID that is globally unique. This means that the PIDs are not repeated, as long as the process is up and running. PID 1 is assigned for P_i and P_j interconnection, PID 2 is assigned for P_j and P_k interconnection, and PID 3 is assigned for P_k and P_i interconnection, respectively. In the first state, P_i will have in its

r_i register C_j and C_k, according to the PIDs 1 and 3, learnt from its directly connected neighbours. After the second execution, P_i will now have in its r_i register new entries C_j,C_i and C_k,C_i. P_i then performs mutual exclusion between the previous entry and the latest entry, and extract its secret code.This holds accurately for P_j and P_k as well. For any value of n, such that n≥3, this is valid. This shows that the algorithm can self-stabilize and give a legitimate configuration. This is demonstrated below:

Execution 1

P_i	
PID	Secrets
1	$\{C_j\}$
3	$\{C_k\}$

P_j	
PID	Secrets
1	$\{C_i\}$
2	$\{C_k\}$

P_k	
PID	Secrets
2	$\{C_j\}$
3	$\{C_i\}$

Execution 2

P_i	
PID	Secrets
1	$\{C_j\}, \{C_j\ C_i\}$
3	$\{C_k\}, \{C_k\ C_i\}$

P_j	
PID	Secrets
1	$\{C_i\}, \{C_i\ C_j\}$
2	$\{C_k\}, \{C_k\ C_j\}$

P_k	
PID	Secrets
2	$\{C_j\}, \{C_j\ C_k\}$
3	$\{C_i\}, \{C_i\ C_k\}$

Figure 2:

Without Processor identifiers (PID)

Assuming an asynchronous bidirectional relationship exists amongst the processors, which is triggered by a scheduling mechanism, whereby the selection of the process is random and the execution is sequential. Then, we consider a complete graph with edges of $(\frac{n(n-1)}{2})$, for n≥3. Suppose the s-registers contains a value which is the addition of the secret codes learnt from the other processors. All processors perform the same operation with their neighbours. Then,P_i will perform addition of r_j and r_k, minus r_i, in its s-register. The secret code is the final value divided by two (2), since the edges in the graph were counted twice. This shows that the algorithm can give a legitimate configuration. This is demonstrated below:

$$r_i = s_j + s_k = C_j + C_k \tag{1}$$

$$r_j = s_i + s_k = C_i + C_k \tag{2}$$

$$r_k = s_i + s_j = C_i + C_j \tag{3}$$

$$secret_i = (r_j + r_k - r_i)/2 \tag{4}$$

$$secret_i = (C_i + C_k + C_i + C_j - C_j - C_k)/2 = C_i \tag{5}$$