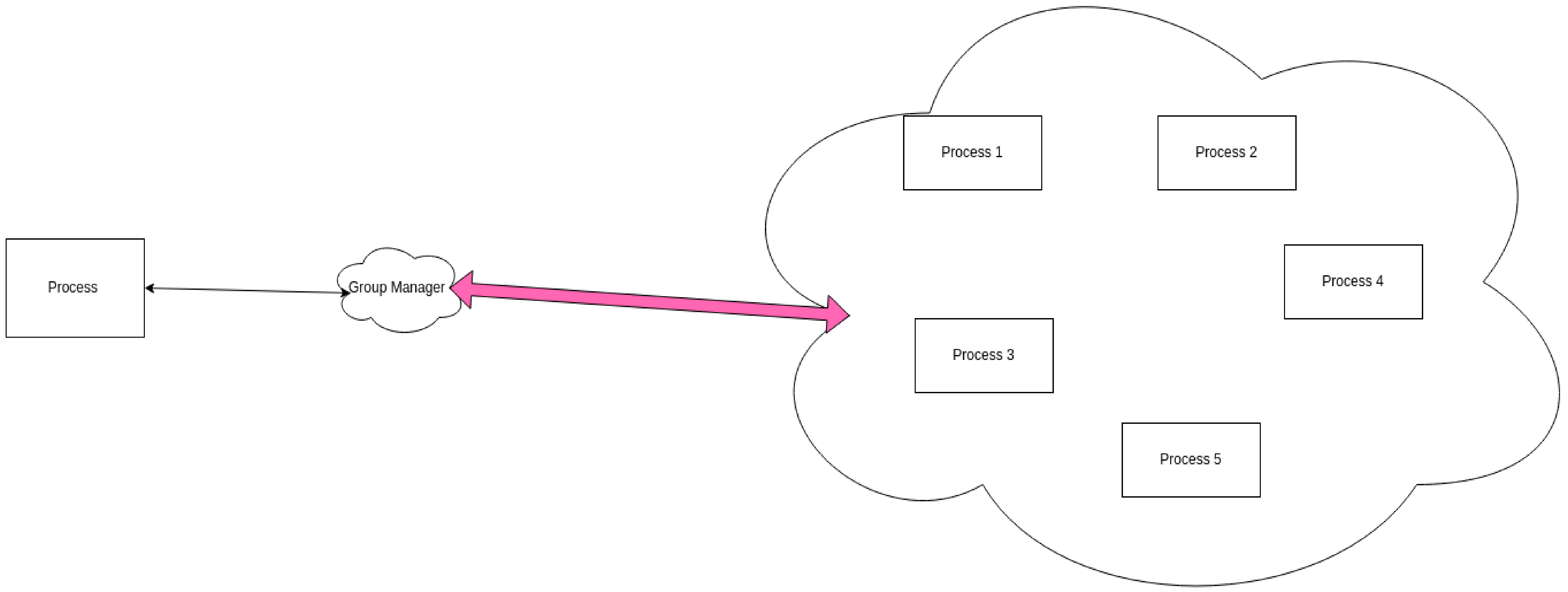
Homework 2

Team 1

Apostolopoulou Ioanna
Toloudis Panagiotis

Start

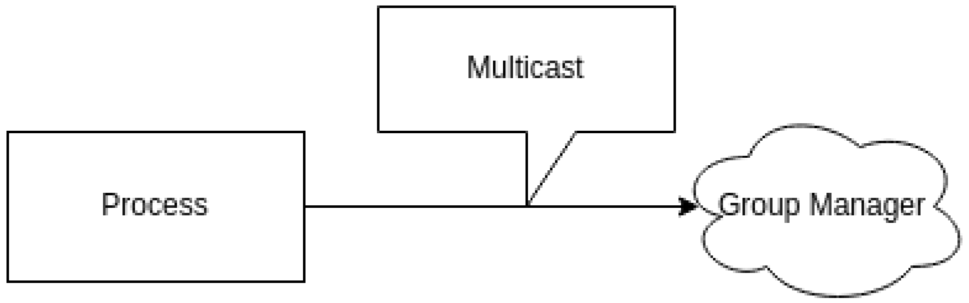

Process
Group Manager
Process 1
Process 2
Process 4
Process 3
Process 5

Basic Stuctures

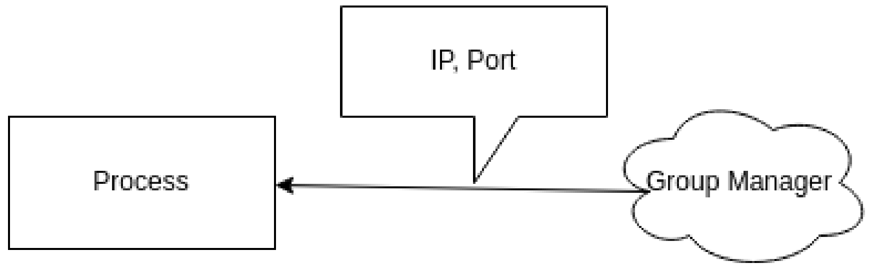
- When a new process is created and wants to join a group, it uses multicast to discover the Group Manager.
- Processes have one TCP socket to communicate with the Group Manager and one UDP socket to discover the Group Manager and exchange messages in the Group.
- Group Manager has also one TCP socket to communicate with processes who want to join or leave the Group and one UDP socket to reply to Multicast done for discovery.

MULTICAST

Multicast Send

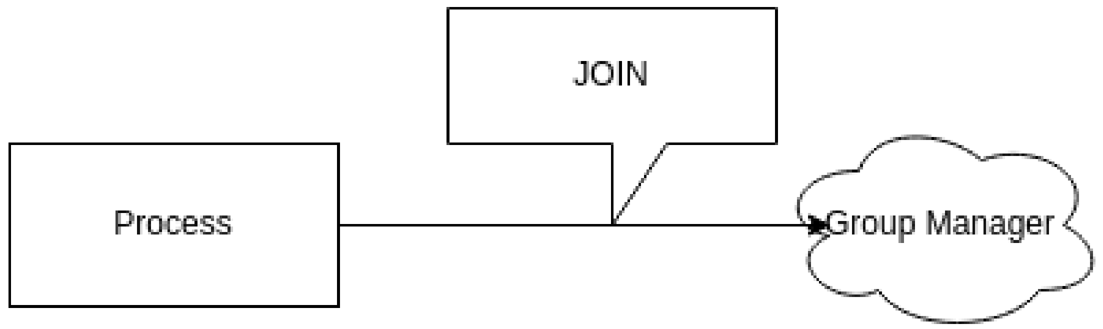


Unicast Resive

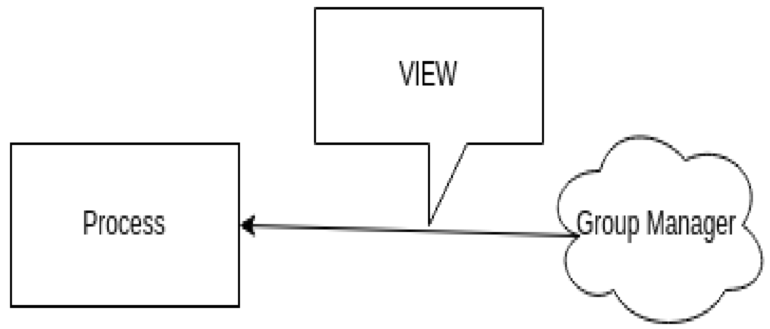


Join in Group

Join in Group Request

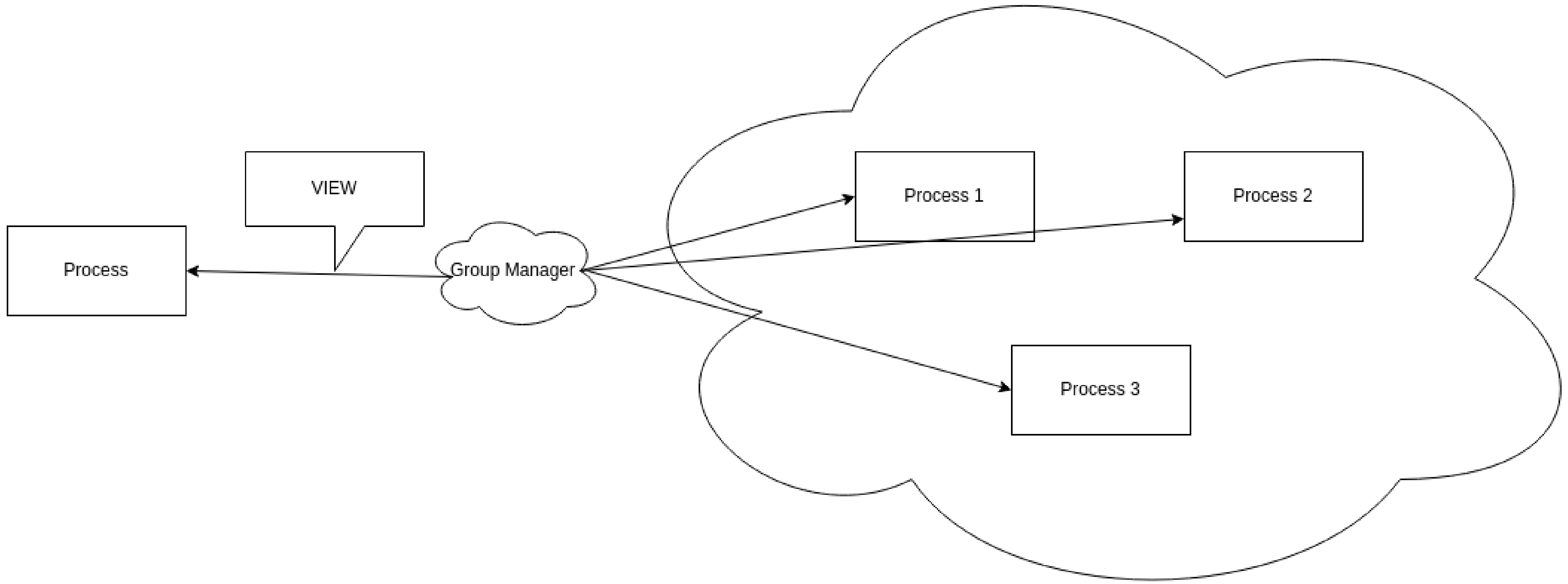


GM Send View



VIEW:
- If Join: Send IP, Port and Name from all in the Group
- Else: NOT

Send Update in Group



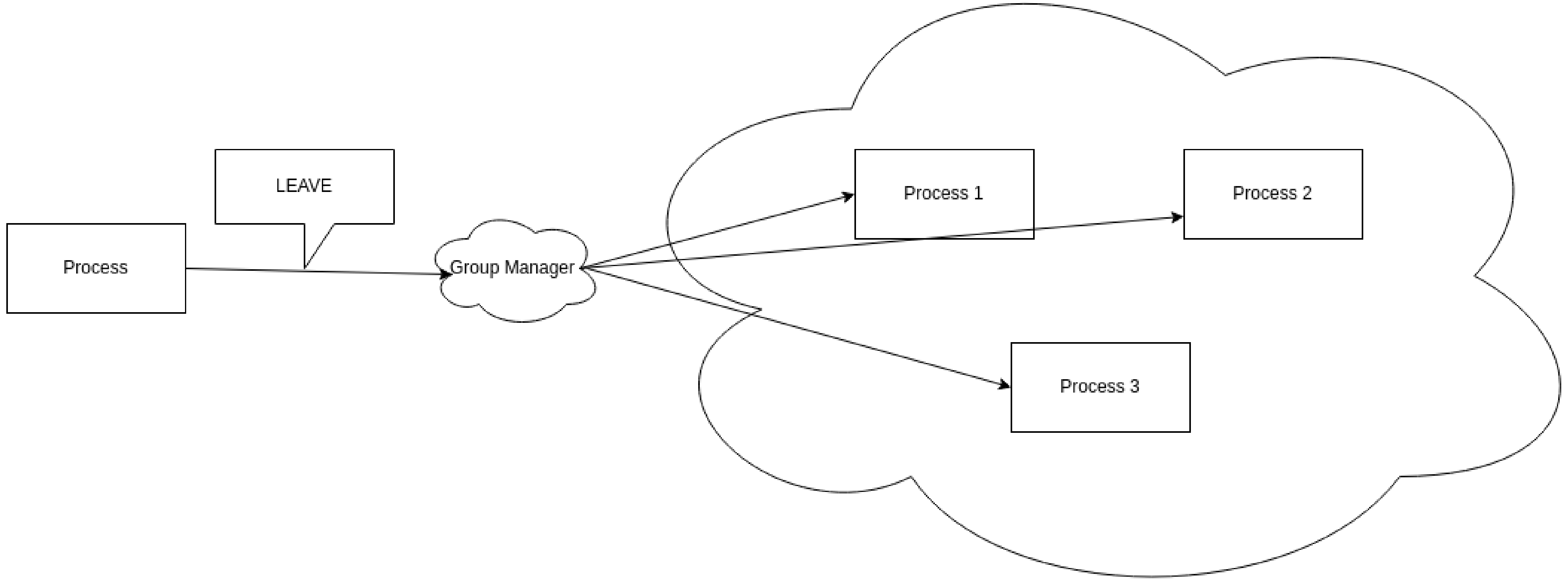
Join & Update Group

- When process finds Group Manager from Multicast, a communication between Process and Group Manager begins by using a TCP/IP connection.

- Process sends a join request at Group Manager, that contains the Process Name, its IP adrress, its Port and the Group Name to join.

- Group Manager receives the request and sends to process a reply containing the view of the group and informs all other processes in group by sending the contact information for the newly joined process.

Leave in Group



Leave & Update Group

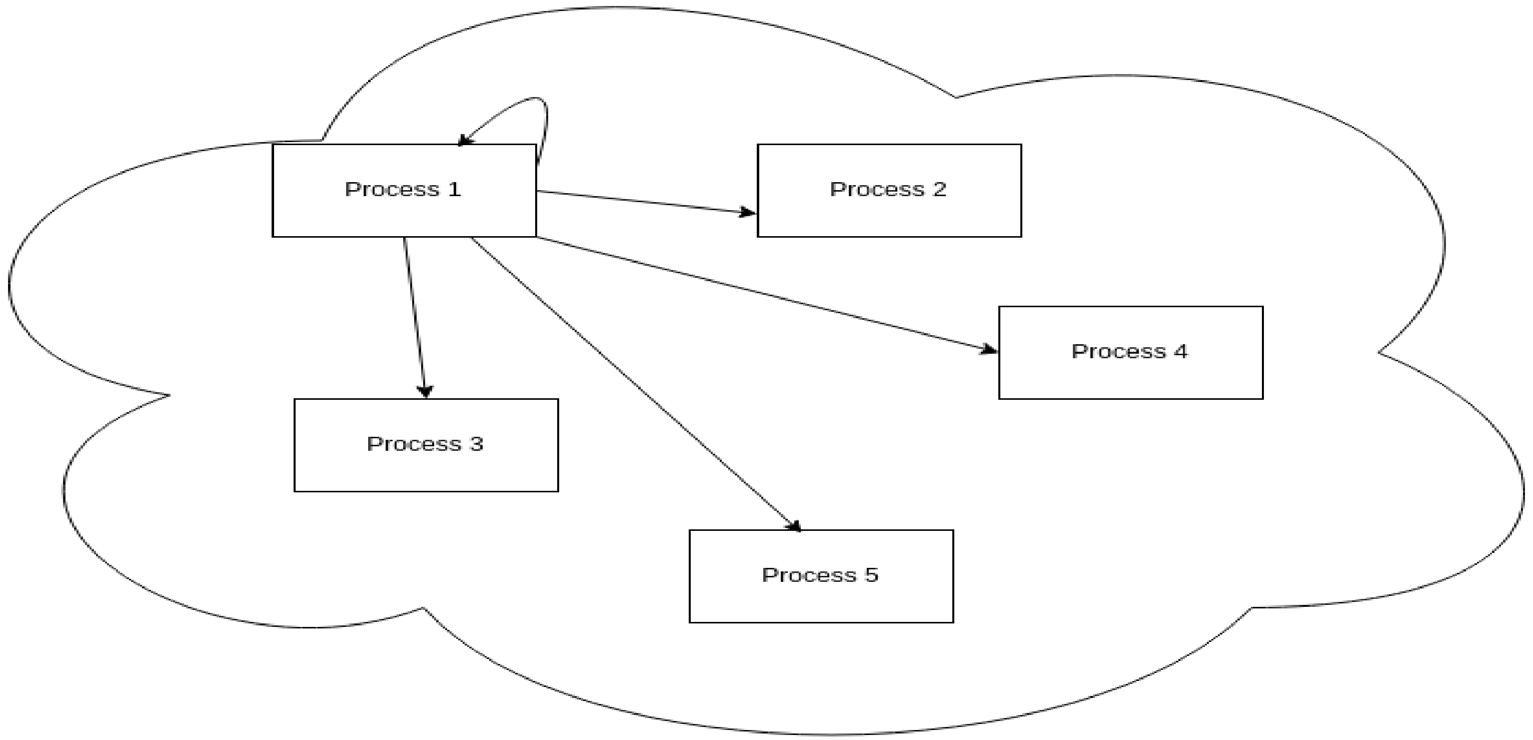
- When a Process wants to leave the Group or a node has a damage, the Group Manager and Process reuse their previous TCP/IP connection for communication.

- Process sends a request to leave Group at the Group Manager.

- Group Manager removes process from Group and informs the other processes in Group for the Changes in the dynamics of the Group.

FIFO Order

Send with FIFO

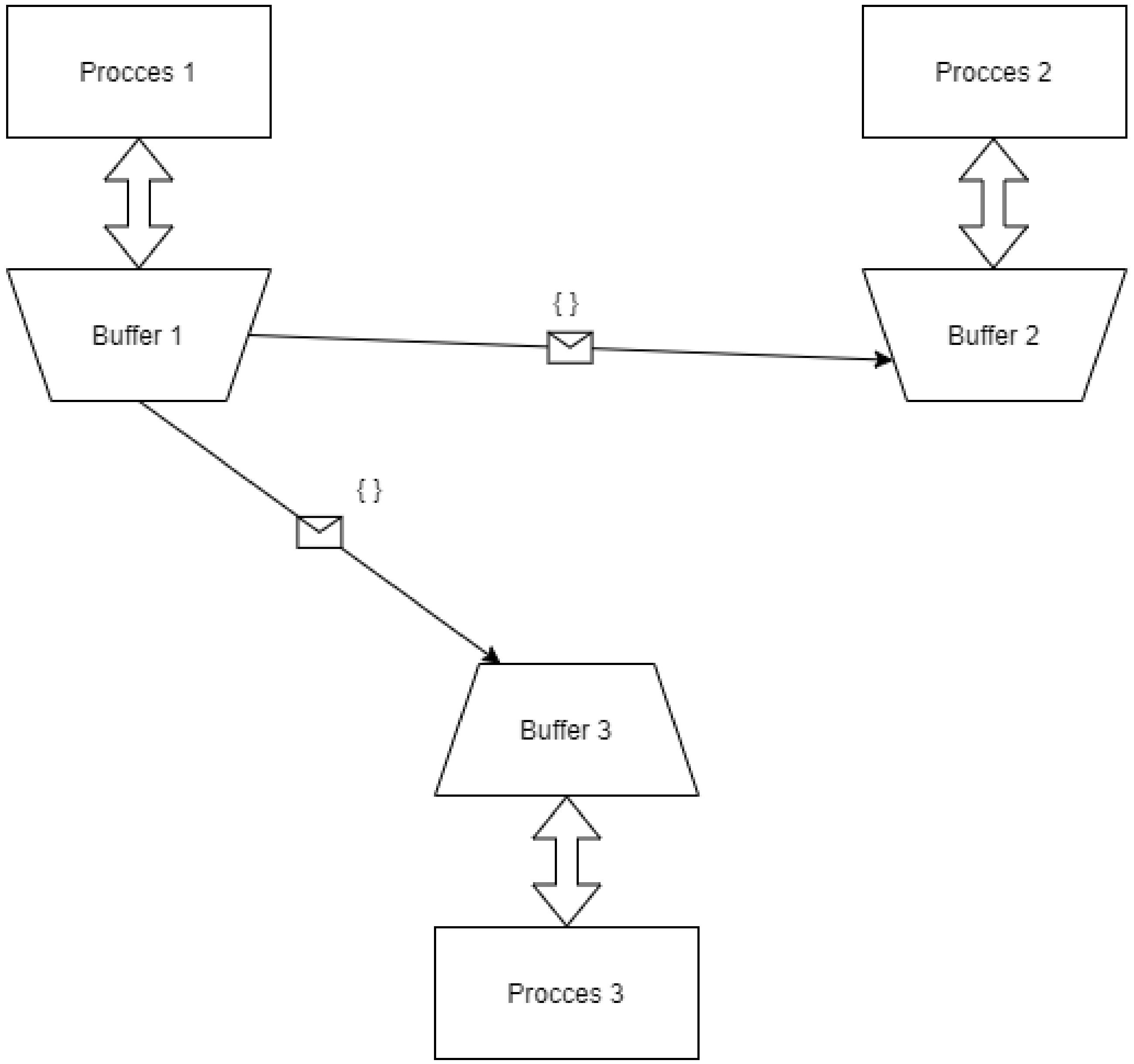


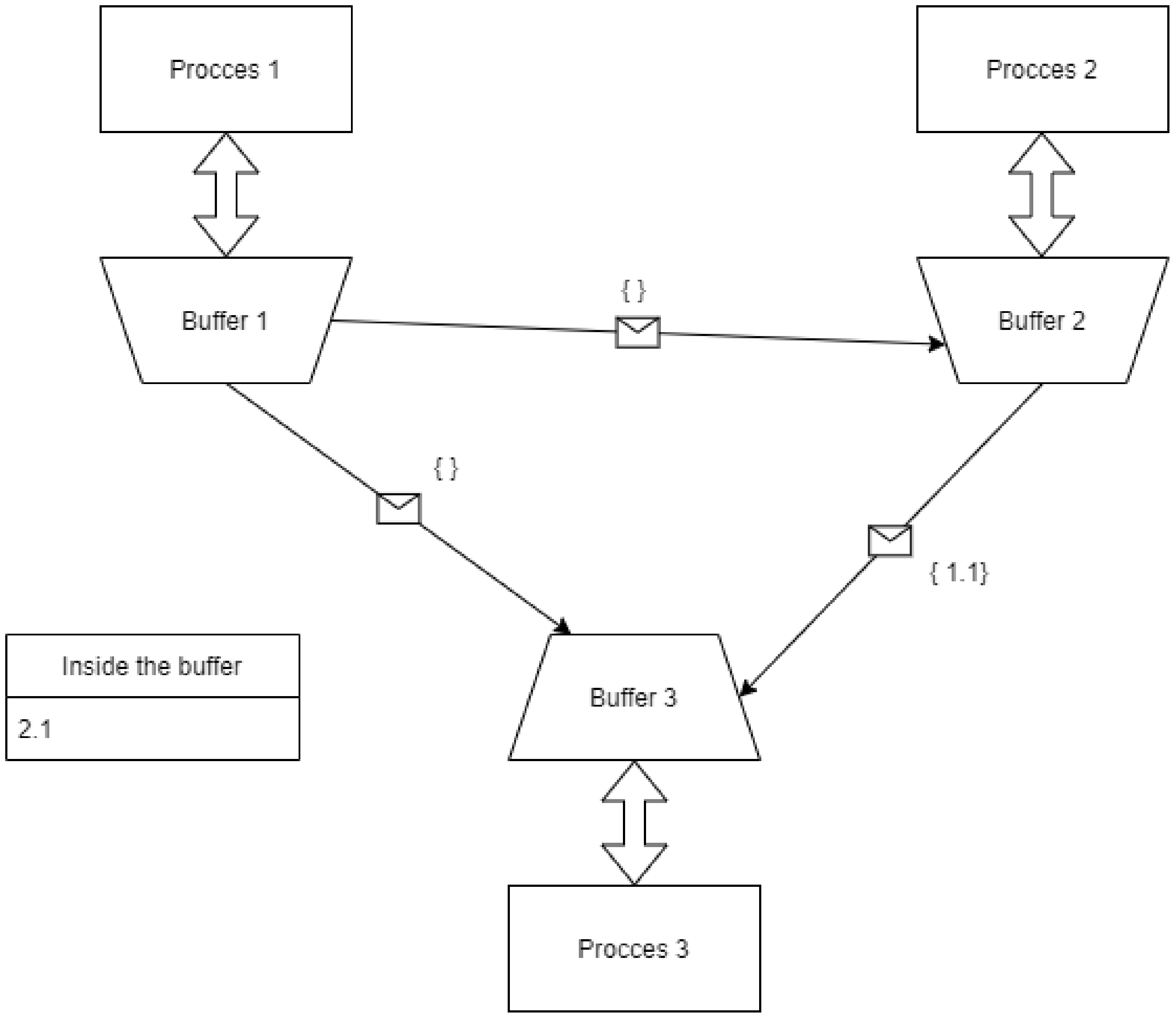
Fifo Order

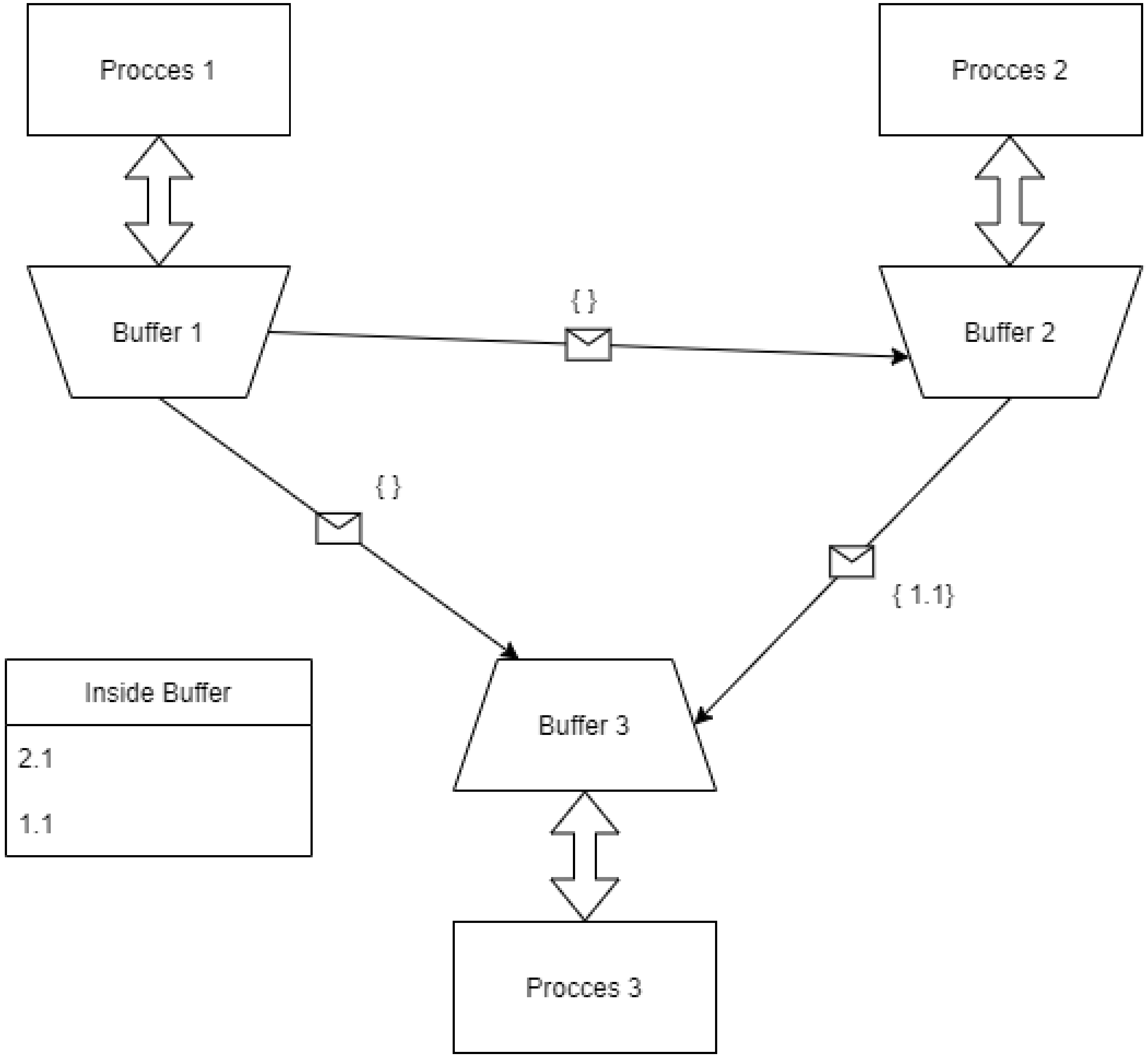
- When Process in Group sends a message using catoc parameter as 0, messages in group are delivered in Fifo Order.

- When message arrives in process it is stored in a buffer. Process sends the first stored message into application and continues to send next messages by compairing their sequence number.

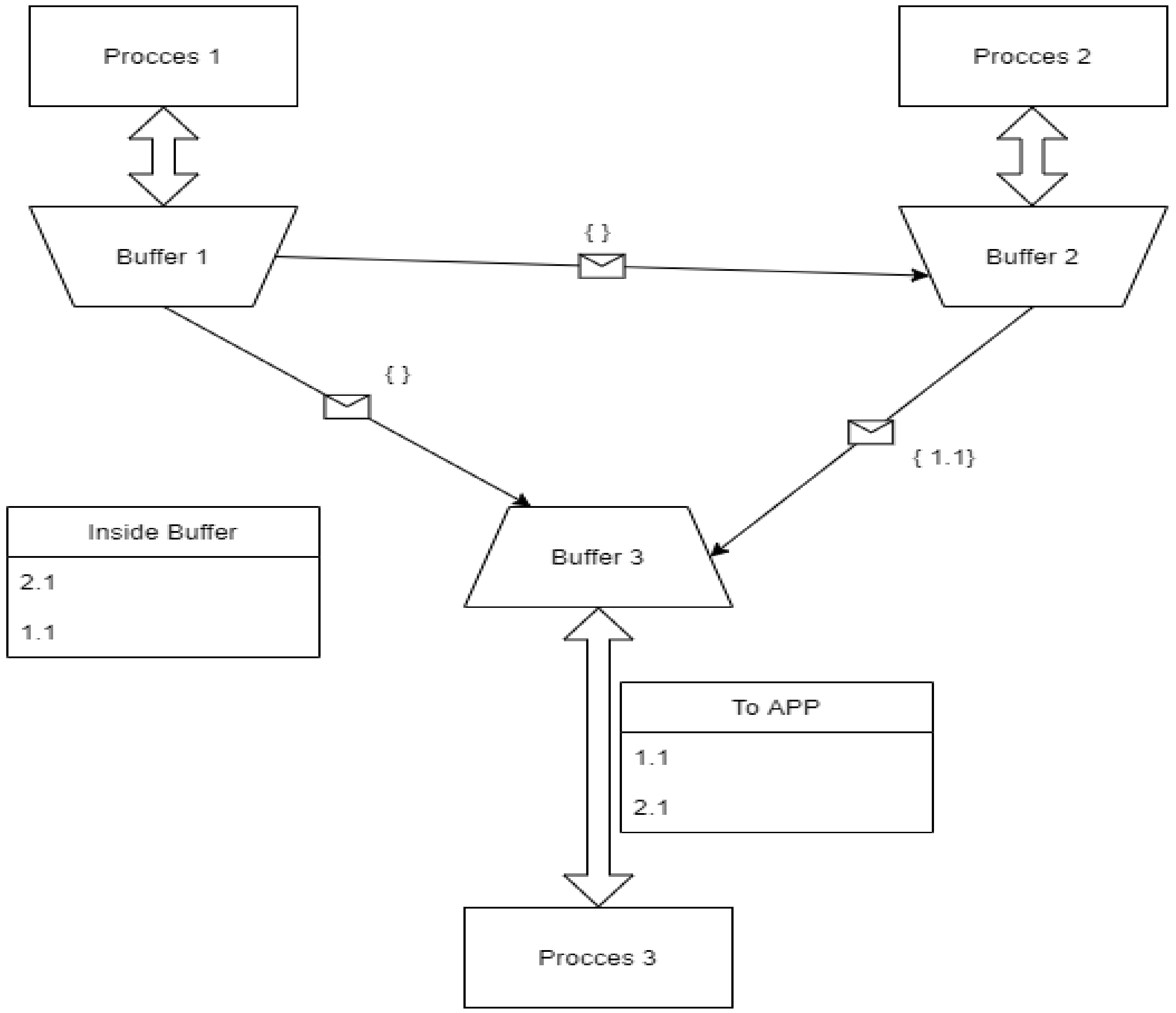
Example Fifo Order


Procces 1
Procces 2
Buffer 1
Buffer 2
{ }
{ }
Buffer 3
Procces 3


Procces 1
Procces 2
Buffer 1
Buffer 2
{ }
{ }
{ 1.1}
Inside the buffer
2.1
Buffer 3
Procces 3



Inside Buffer
2.1
1.1

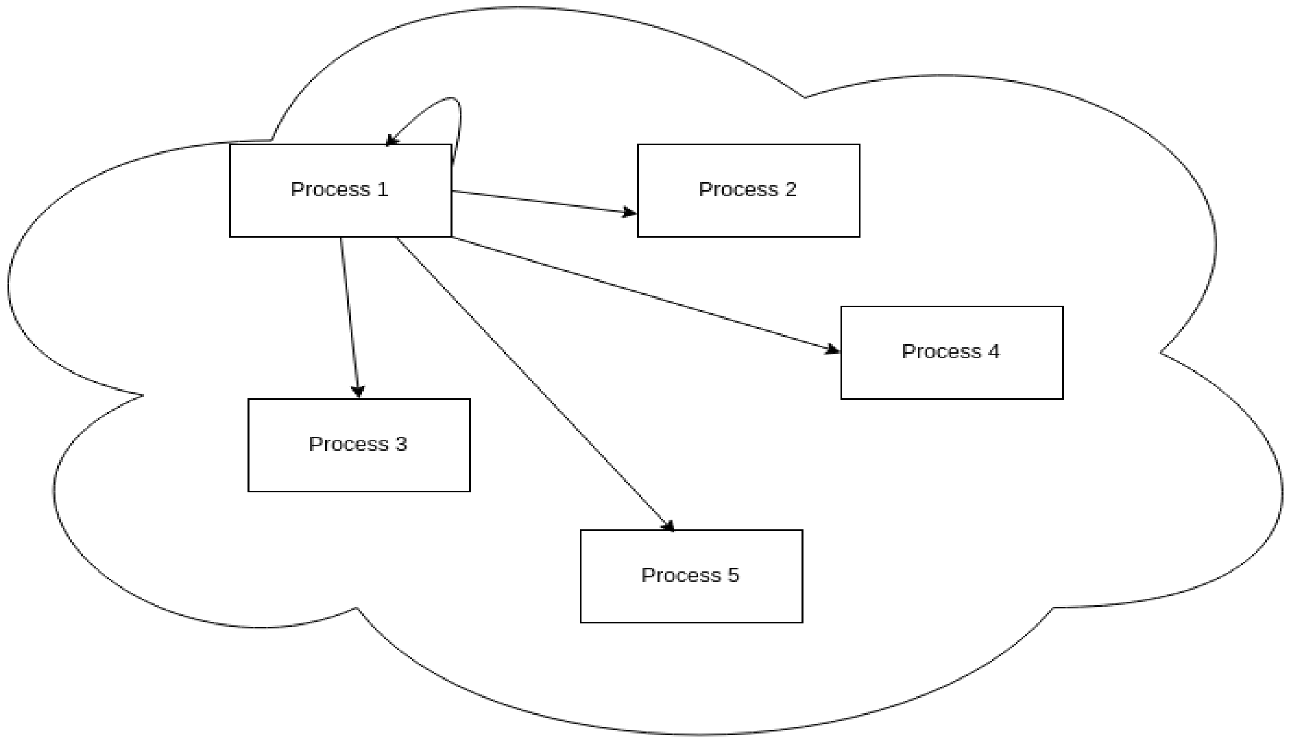


Inside Buffer
2.1
1.1

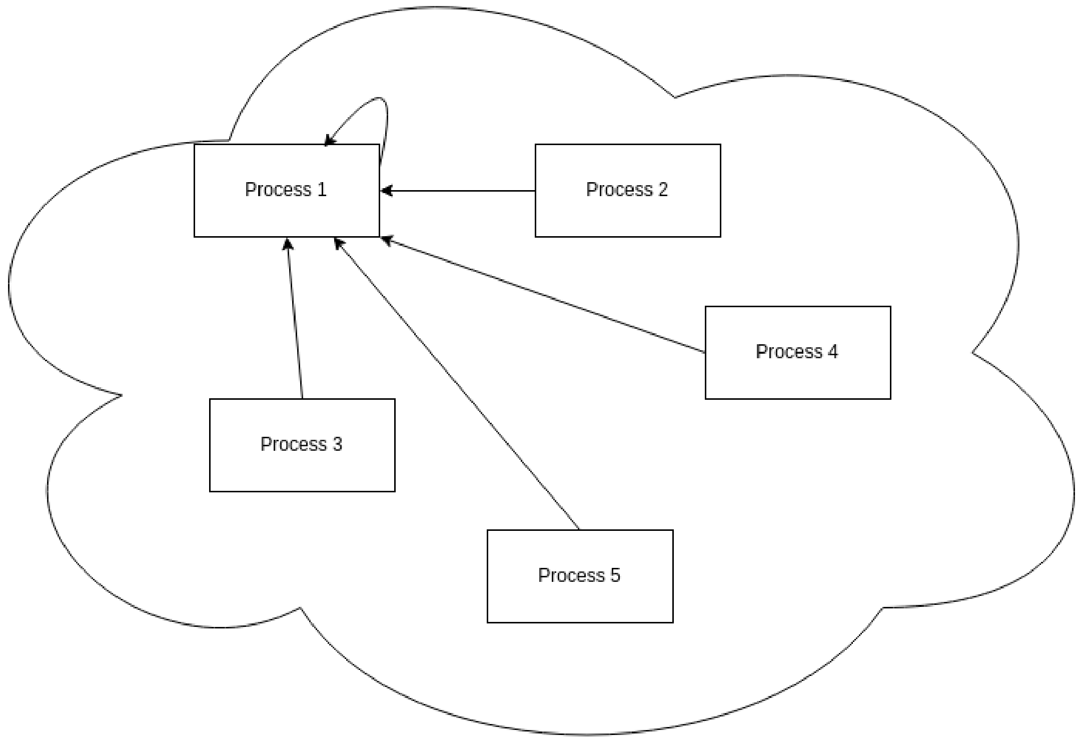
To APP
1.1
2.1

Causal-Total Order

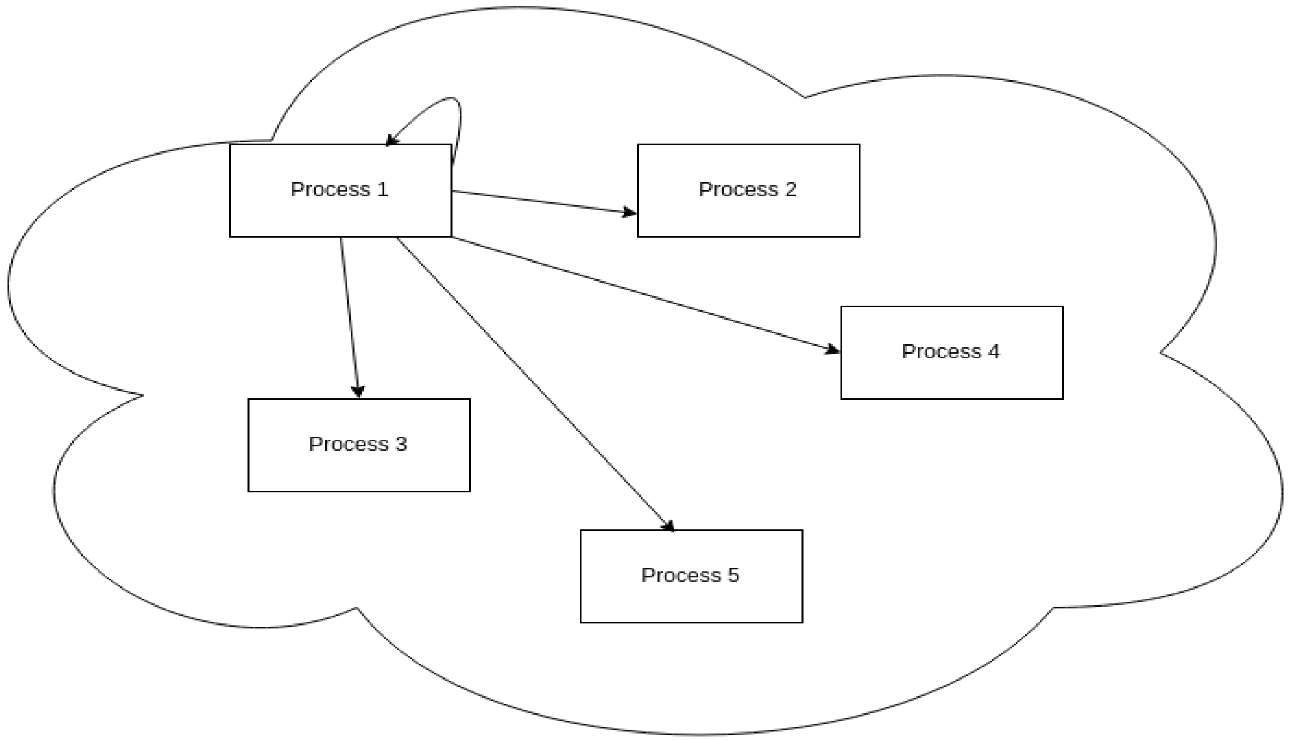
Send with CATOC



VOTE RECIVE



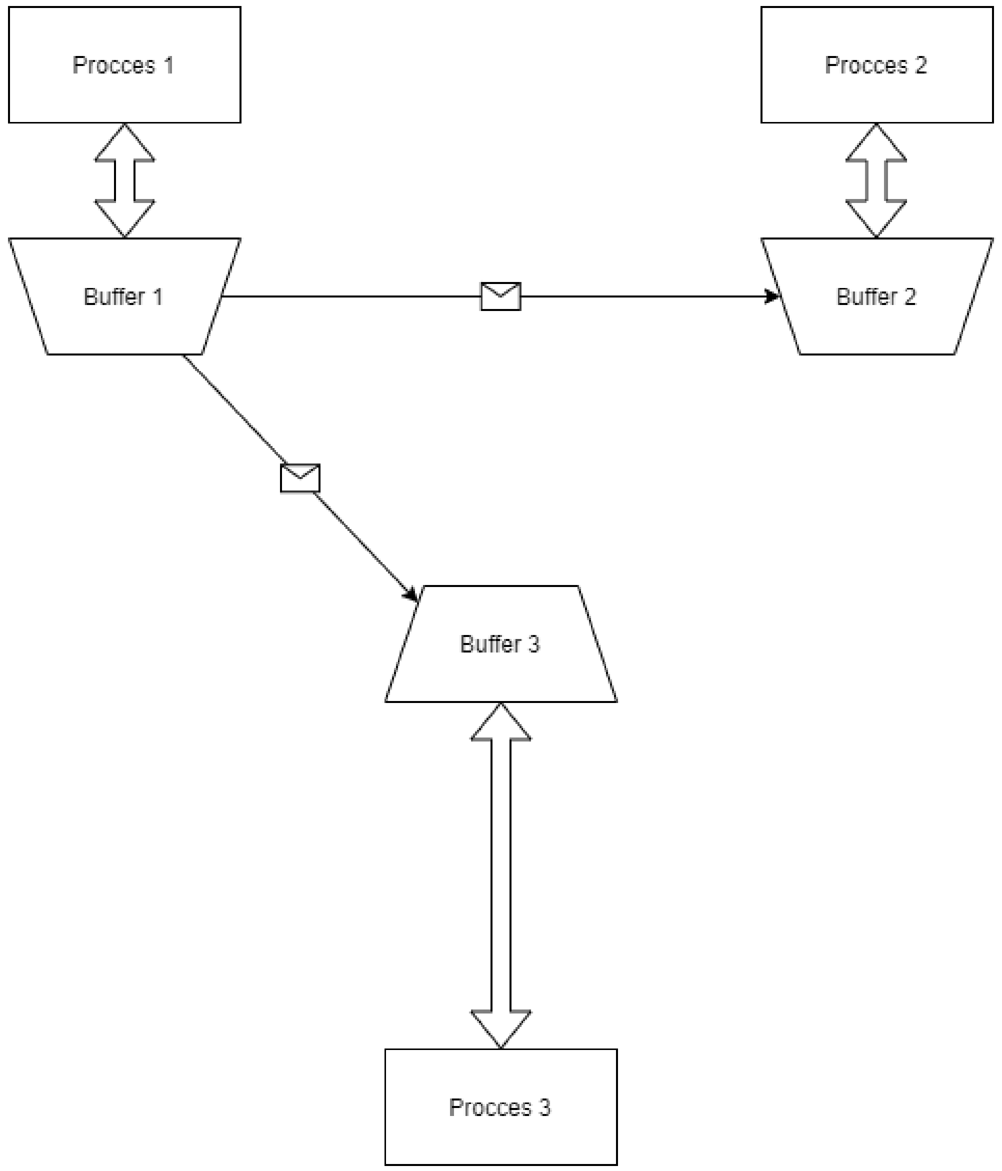
Max Vote

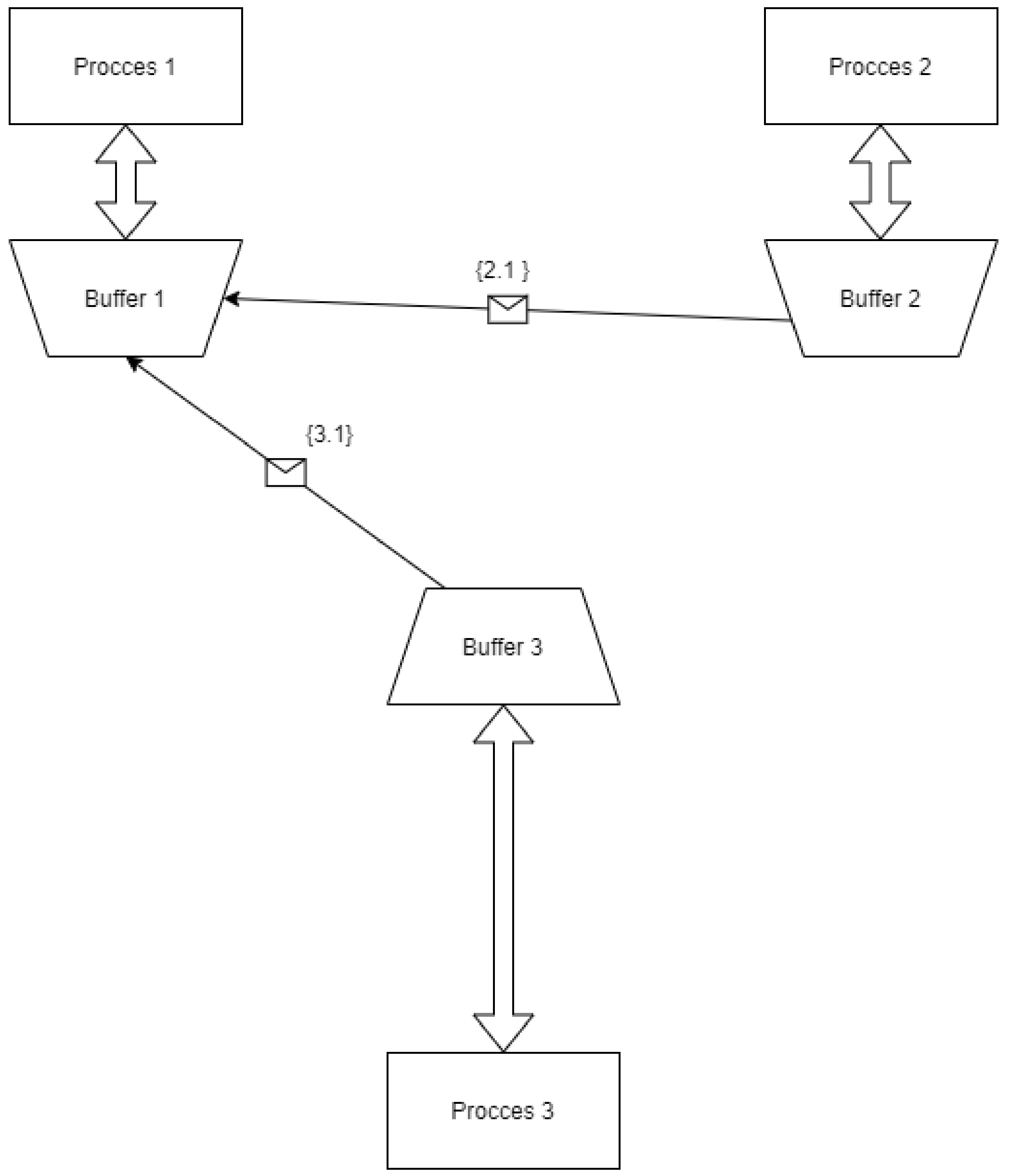


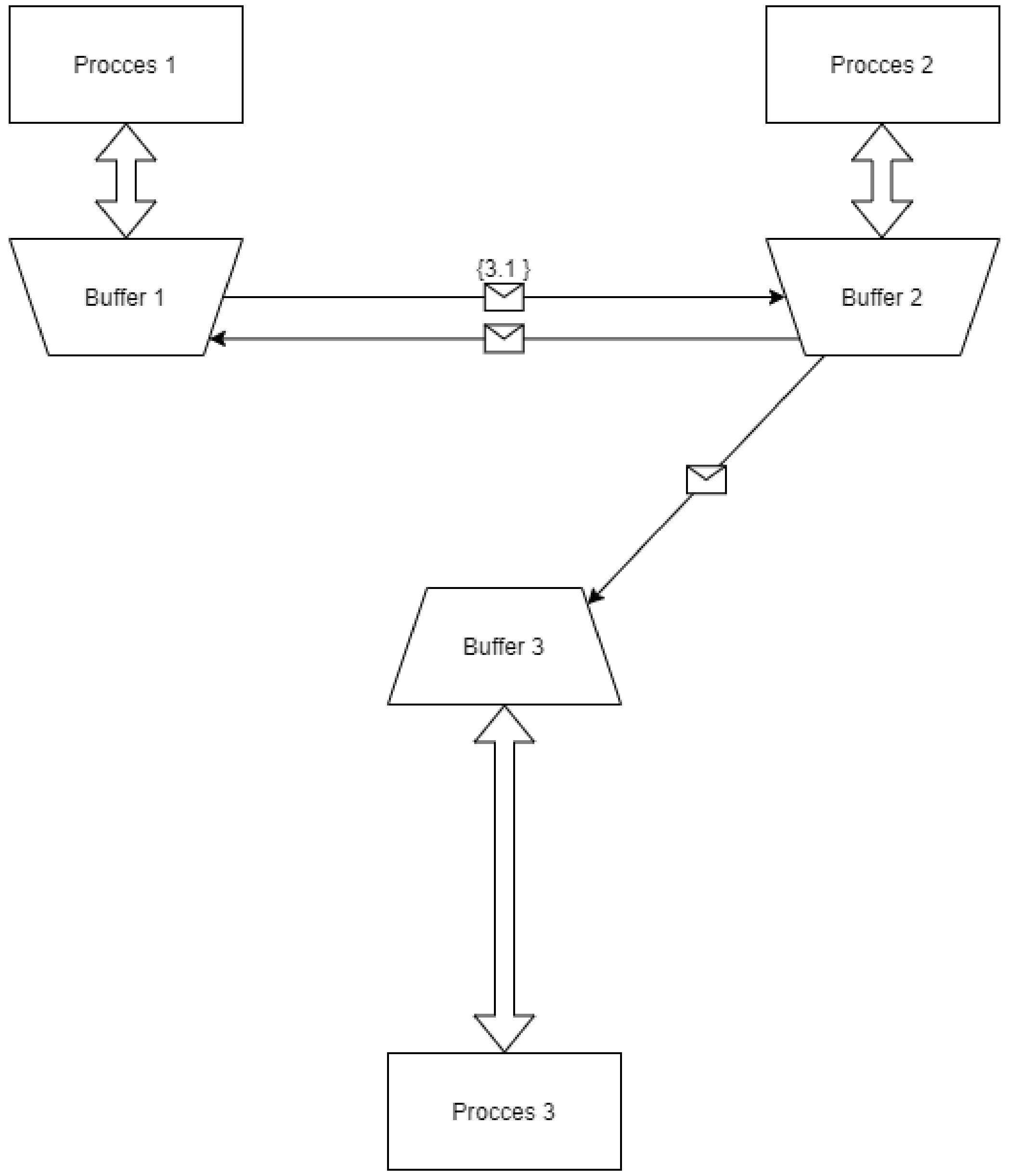
Causal Total Order

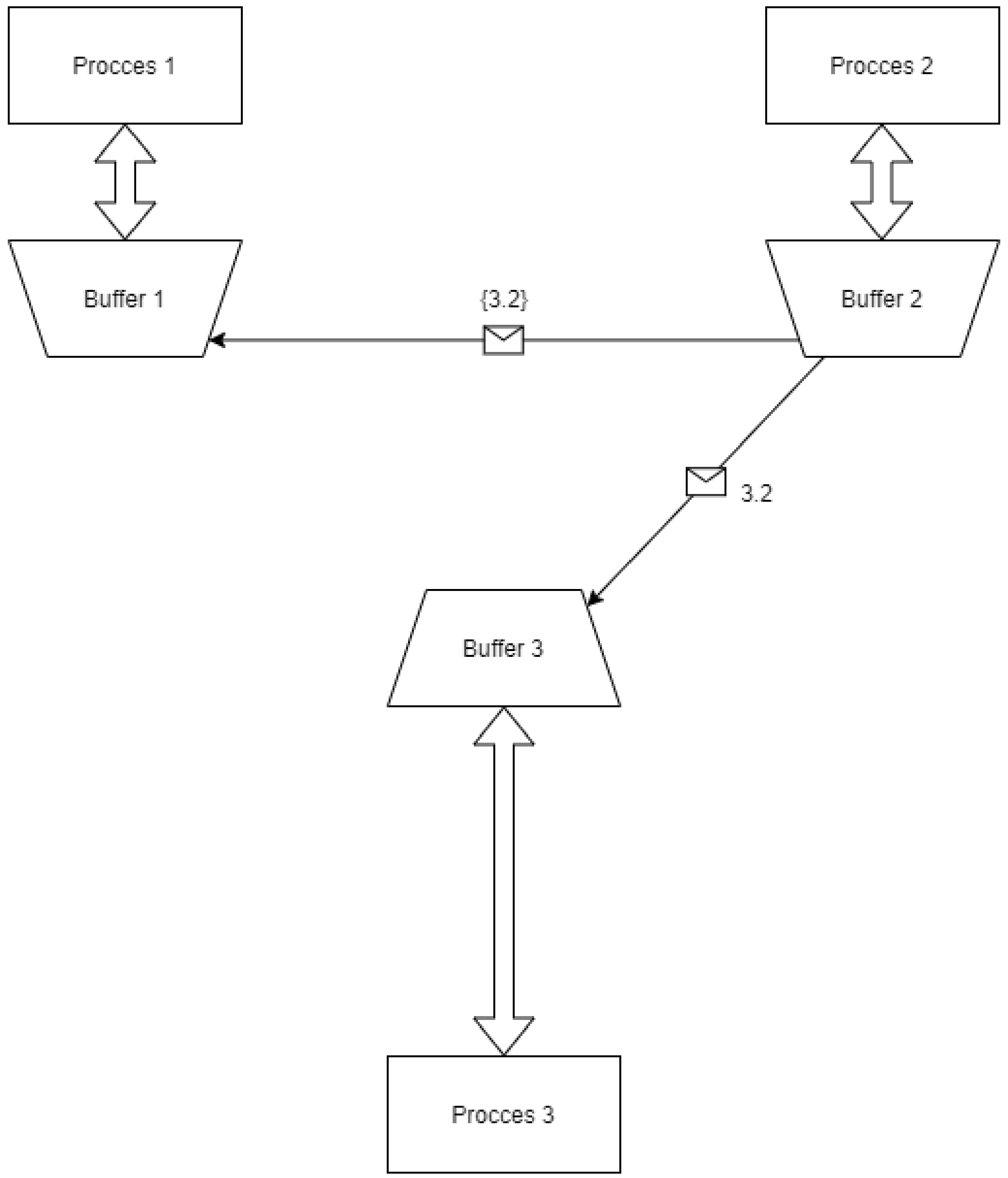
- When a process wants to send message in group with CATOC Order should give catoc parameter the number 1.
- When message arrives in process it is stored in a buffer.
- We are using a Symmetric protocol that each process in Group when a message is stored in its buffer send back to sender a vote to determine the priority each message has in buffers.
- Each message is delivered to the applcation in order determined by the voting process, so the causality of the messages is ensured.
- When one process sends simultaneously n messages causality is broken so we are using a mutex to block each process from sending n messages before the first is delivered to the application.

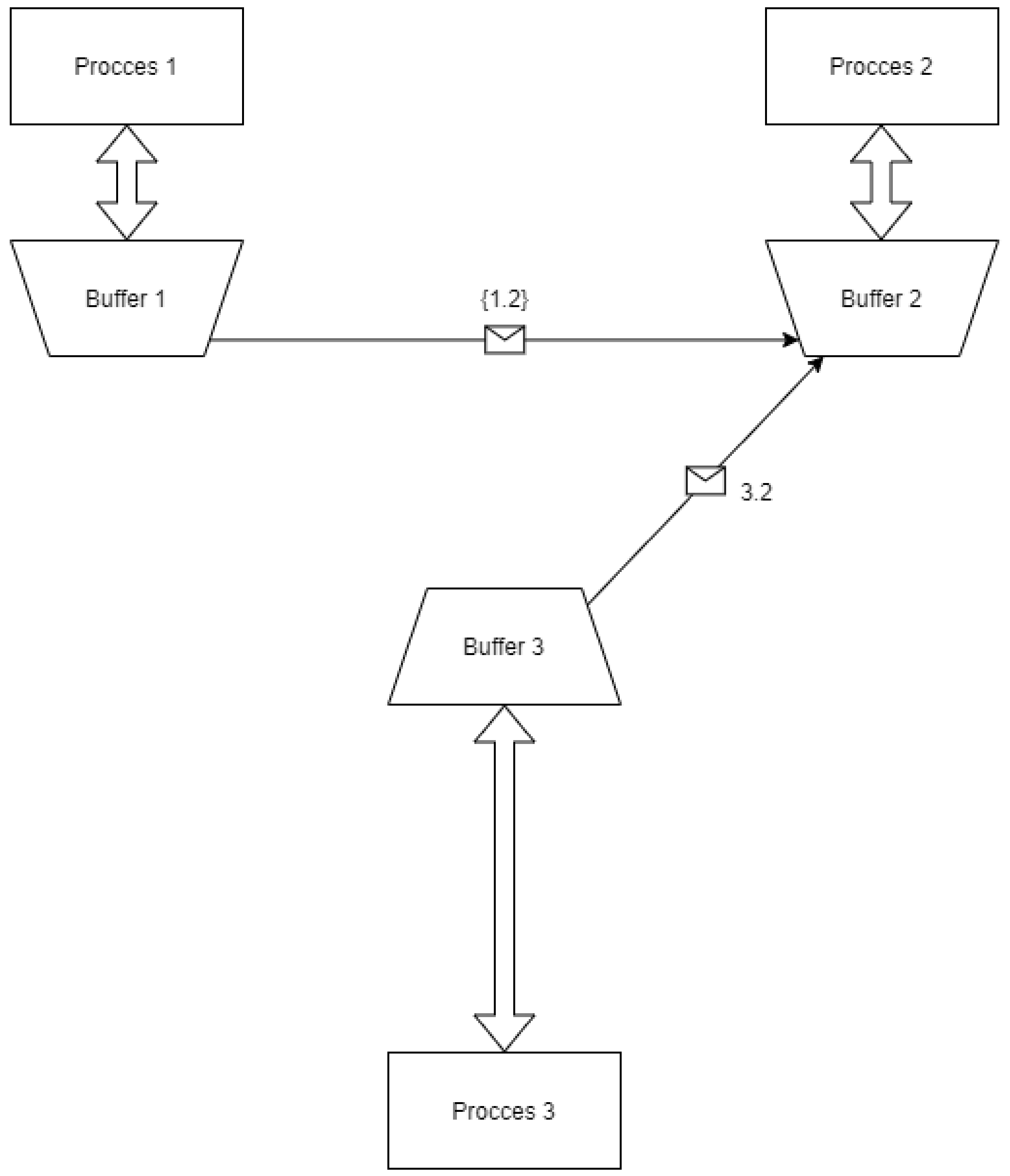
Example CATOC Order

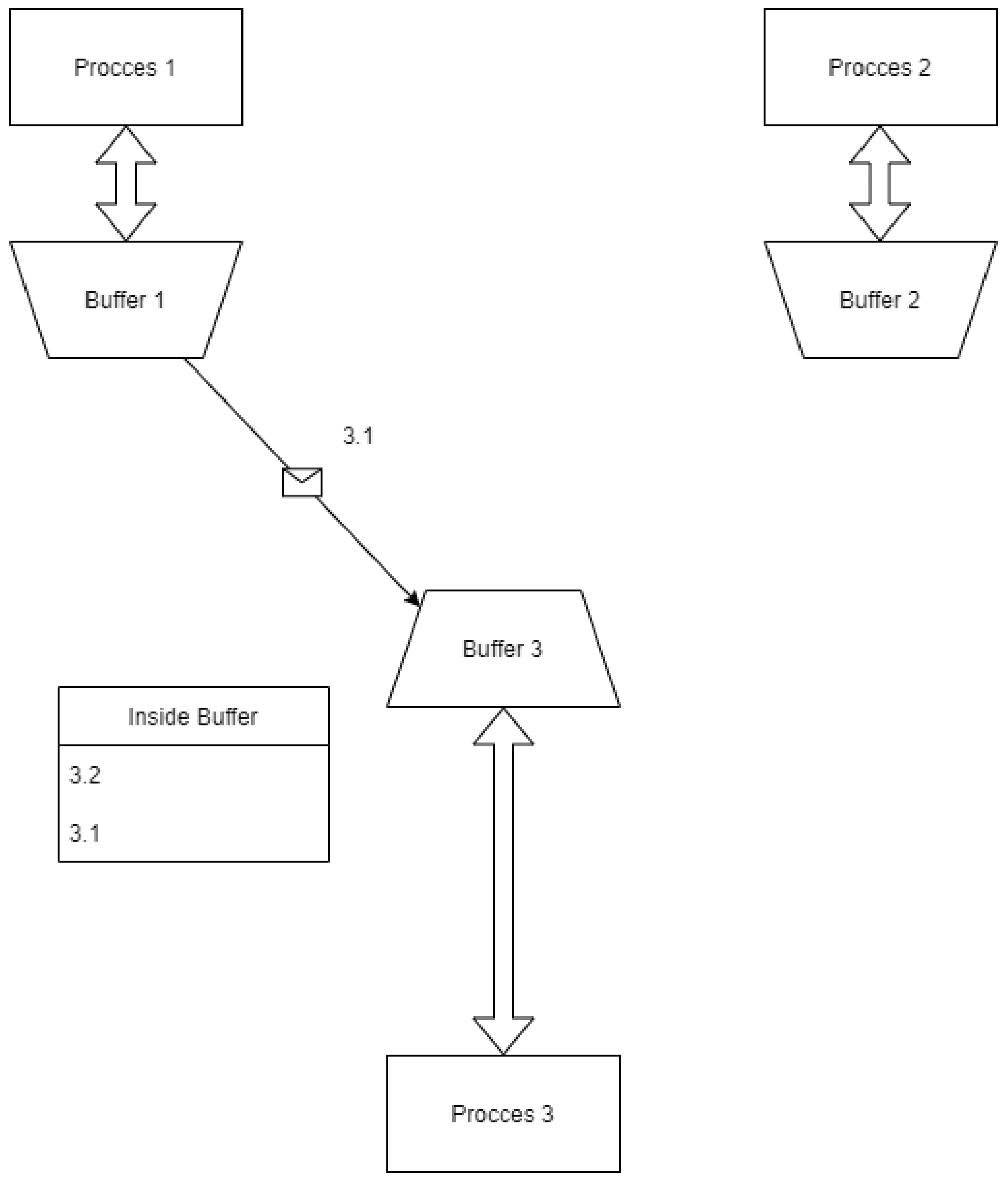

Procces 1
Procces 2
Buffer 1
Buffer 2
Buffer 3
Procces 3

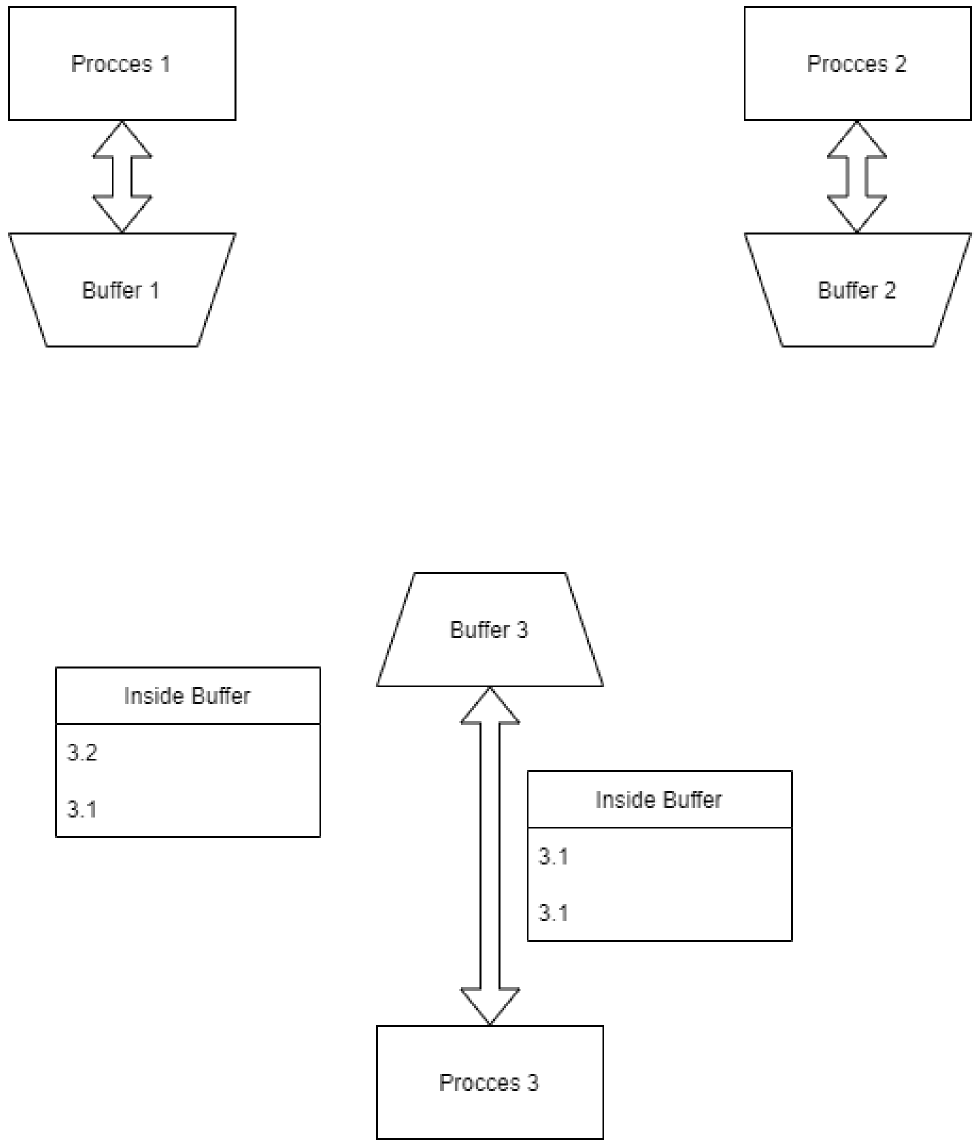

Procces 1
Procces 2
Buffer 1
Buffer 2
{2.1 }
{3.1}
Buffer 3
Procces 3


Procces 1
Procces 2
Buffer 1
{3.1 }
Buffer 2
Buffer 3
Procces 3


Procces 1
Procces 2
Buffer 1
{3.2}
Buffer 2
3.2
Buffer 3
Procces 3


Procces 1
Procces 2
Buffer 1
{1.2}
Buffer 2
3.2
Buffer 3
Procces 3


Procces 1
Procces 2
Buffer 1
Buffer 2
3.1
Buffer 3
Inside Buffer
3.2
3.1
Procces 3


Procces 1
Buffer 1
Procces 2
Buffer 2
Buffer 3
Inside Buffer
3.2
3.1
Inside Buffer
3.1
3.1
Procces 3

The End !!!