Contents

1 Description and modeling

The general context is that of the modeling of the N-Green network. Starting from an overall description of the model, state specifications of each component so we set the foundation for what deserves special attention and critical parameters to make it a simulation

1.1 Description

In the description of optical ring we consider node's two types. The first one is **bridge** and the second is **switches**.
Indeed, the bridges are simple nodes of the ring capable of being the source or recipient of a packet within the ring. All their operations are internal to the ring. Switches are special nodes that delimit the size of the ring; They are also used for interconnecting the second ring and also they are able to empty containers without being recipient or source.

About distribution modes we mention the mode **Unicast** and **Broadcast-and-Select** that attract our attention.
In Unicast mode, each source node sends packets for a specific destination. This means that each container contains information for an exact destination and the same for waiting. It's called one-to-one
For the Broadcast-and-Select mode, each source node uses the same slot to address a recipient set, and when they receive packets, they make a copy and then extract what is intended. It's named one-to-many. The constraints are posed by the type of traffic and the requirements on slot's padding.

On the ring there are two types of services: **premium** and **best effort**. The premium means the high priority of packets or critical limits on which the delay must be as minimal as possible while the best effort is to satisfy as soon as possible. Even if it's a best effort, it will also be necessary to minimize the routing time as much as possible.

For packet's insertion, we have also two mode to explore: **opportunistic** and **reservation**.
The opportunistic insertion means that when a slot is empty, a node can use it to transmit some packets; the node may decide not to use it if it considers that none of the packets currently being build is ready for transmission.
About reservation mode, a slot may be empty but reserved for another node, or another class of service.
When a node is allowed to insert data, and there are several packets ready for insertion, the node has to select which one to insert: this is **scheduling**. The scheduling is internal to each node.

As regards the reception on ring, a node decides to receive packet when it's label with receiver number. First case, the node may receive the packet and

free the corresponding slot by erasing the packet; this is typical for the unicast mode. The second case, the node get an copy of the packet to check which data is for him exactly. Only the source of packet can erase and release slot.

1.2 Model of Ring

In this section we present the framework of defining our work, some specifications, problem and the approach of our observations.

1.2.1 Slotted time model

In our model the time is discrete. The unit of time is one slot. Let's note **ST** this time. During ST click, each node sees a container with a capacity **c**. During the ST, two basics operations are possible: **Writing** and **Reading**.
In reading, it is a matter of making a copy of the packets to extract data; for writing, put the packet to be sent. The packet's weight can be noted as w_p and add that the waiting time α_{max} can be associate to packet; this means the maximum time during packet may be waiting for padding.

1.2.2 Latency

Here we define latency as the difference between the time a request was received and the time it should be received if it was sending since the moment it was created. Indeed we can get a deterministic estimate of the time separating the path of a request between source and its destination noted as "best case". So the latency is mainly due to time during requests waiting α_i for packet's get full and sending . Let's note latency as **l**, the estimation sending time as t_e and the reel time of receive as t_r .

Two major problems stand out: minimizing α_i and maximizing containers for message transport.

1.2.3 Problem and Constraint

On this part we summarize the constraint that can be used for the ring observation.
Let's note **R** the set of request, $R_p \subset$ R the set of request premium and
$R_b \subset$ R the set of best effort request

$\forall$ r $\in$ **R** ; t_r=t_e+α_i and if r $\in R_p \Rightarrow \alpha_i \to 0$

α_{max}=max(α_i)

thus, min ($\sum \alpha_i$) and min α_{max}

For slot using, Let's note q_i length of request; q_i=const with const=max(q_i)

min $\sum$(C- W_p)
W_p=$\sum$(q_i) and β =$\frac{W_p}{C}$ where $\beta \to 1$

1.2.4 About nodes

On a node, to manage request's list for preparation of packets to be sent. Each request has a arrival time t_a which depend to period **p** of distribution on the ring.
Whatever the type of request, it is necessary to be able to minimize the waiting time that's we note α_i request for each node. Specially for premium request, the goal is to have no waiting time. Here It's important to explain how to create request on each node. On the ring we'll define period p for premium request creation on each node and quantity q. This implies that if we have request on time t on a node, the next request on the same node should be at t+p
Let's note **MAX** maximum request production of the node

Algorithm 1 creation of request

```
Require: start_time; p
int δ =0;
while ( ¬MAX) do
    if Current_time == start_time+δ.p then
        create request;
        δ++;
```

For creating best effort request we'll use hyper-exponential distribution around each node of ring. Best effort come on the node as a flow. So there is sometimes lot of best effort more than usual case. That's why we use hyper-exponential for generation.
About hyper-exponential, we use combination of random and poisson distribution. let's note λ the parameter of poisson distribution and δ for uniform.

Algorithm 2 Poisson generator

```
Require: λ
x=0; k=1;
k=k*Random;
while k>exp(-λ) do
    x++;
    k=k*Random;
return x;
```

Algorithm 3 Best effort creation

Require: $\delta \setminus \delta \rightarrow 1$;
if ($\neg$MAX) & Uniform(0;1)$\geq \delta$ **then**
call **Poisson generator** with $\lambda \rightarrow$ big-value;
else
call **Poisson generator** with $\lambda \rightarrow$ small-value;

1.2.5 Insertion

About emission problem we use the opportunistic strategy but the goal is to make a good use of resources. During ST, each node must use the container to put a packet as big as it's possible in order to avoid using too much resources than it's necessary. This is to avoid waste of resources and leave the charge of the ring very low. This seems to be somewhat in contradiction with the fact of minimizing the wait times of packets especially those of the service "premium"; But to restore the equilibrium, we will put a mechanism of right of access in emission on the containers in such a way that certain containers can be used only for priority packets. This is a good alternative to improve the time wasting by idea of filling the containers to be full at maximum. Another criteria we can add is to define an maximum waiting time which after this, the packets that we want to be full before sending must be sent even if it's not. This time can be noted α_{max}.

Algorithm 4 Packet's insertion

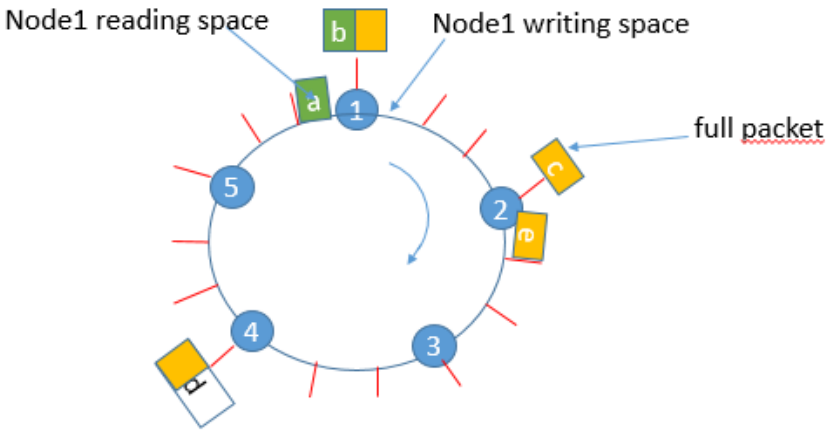
Require: α_{max}; $\beta \in [0;1]$
if $Creation_{time} - Current_{time} == \alpha_{max}$ | | $\frac{W_p}{C} \geq \beta$ **then**
send packet
else
wait for packing

1.3 Presentation of Model 1

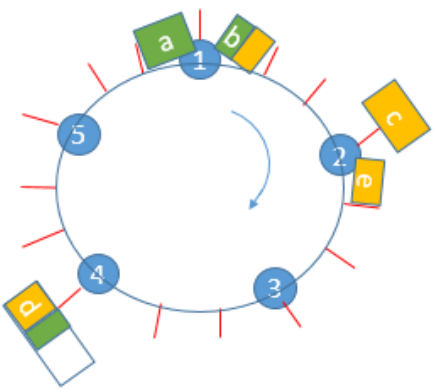
This part concerns the general problem that we study without any specification of the type of request that the node must send. This means that all is Best effort.
Let's note R={ r_1, r_2, r_3,.....,r_i }, the list which can be request or best effort on this model.

Algorithm 5 Model 1

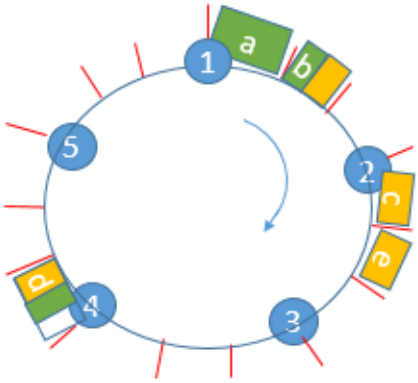
Require: R
$\forall$ r $\in$ R try removing from R to the sending packet's;
call **Packet's insertion**



(a) Picture 1



(b) Picture 2



(c) Picture 3

Figure 1: Basic model1 view

Figure 1: presents the model that we describe above and show how is the ring.

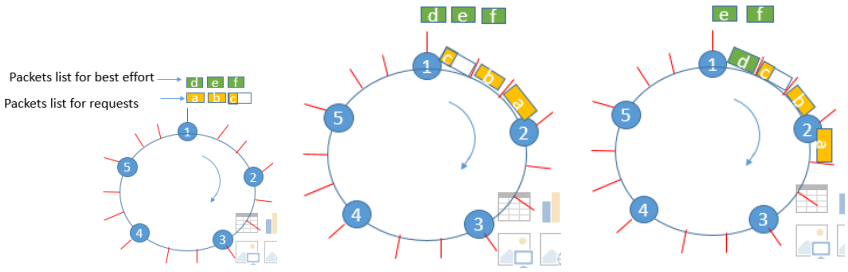
- On Picture 1 the nodes 1,2 and 4 have respectively packets b, c and d on emission. Noticed that each node can read on left and read on right. So node 1 can read the packet a and puts its packet b on right.
- On picture 2 we can see the result of what is done during ST. Node 1 writes on the slot at right and read on left. Node 2 can't write on the slot because the slot is used by packets e sending by another node. Node 4 doesn't write because the packet d isn't full enough for sending.
- On picture 3, the next ST show that node 2 sends his packet by founding empty container and node 4 sends it too because packets d is full enough and ready to be sent .

1.4 Presentation of Model 2

According to this model, node distinguishes the difference between request and best effort. So, for the node we model two packet's list: the first for requests and second for best effort. For sending, main solution is to empty the priority list before trying sending best effort. The goal for this model is to observe and compare with model 1 the gain for request versus loss for best effort.
Let's note LP_r the list of packets for request sending; LP_b the list of packets best effort.

Algorithm 6 Model 2 for priority

Require: LP_r, LP_b
while (LP_r ! ϕ) **do**
  call **Packet's insertion** for LP_r
call **Packet's insertion** for LP_b



(a) Picture 1 (b) Picture 2 (c) Picture 3

Figure 2: model2 view

Figure 2: presents the sample model2 operation.

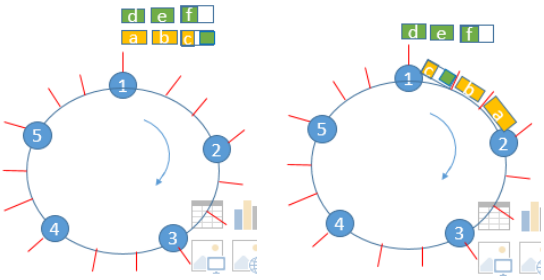
- On Picture 1 the nodes 1 has two packets lists on emission. Request list contains a, b, c while best effort contains d, e, f.
- On picture 2 we can see the result of what is done after three ST. Node 1 sends all its priority packets before best effort. Noticed even if the packets c isn't full enough it was sent before best effort.
- On picture 3, the next ST show that node 1 sends it's first best effort after having it's priority packet list empty.

1.5 Presentation of Model 3

This model contains what we want improve after seeing mode 1 and 2. At first, get very fast as possible for priority packets like on model2; Avoid waste in the use of slot like model2, and improve waiting time for best effort and get it like in model1.
So, let's note
$v_r \subset LP_r \setminus v_r$={ request } and $v_b \subset LP_r \setminus v_b$={ request } $\bigcup$ { best effort } ; $|v_r|$=max($|v_{ri}|$) $\setminus$ $v_{ri} \subset LP_r$) and $|v_b|$=$|LP_r|$–$|v_r|$ For sending, algorithm is the same with model2; but we change best effort insertion.

Algorithm 7 Best effort insertion

Require: b: best effort; v_r.Length; v_b.Length
find=false;
while ¬find **do**
  if |b| $\geq$ v_b.Length **then**
    check place in the next v_r;
if find **then**
  put b in v_r where place was found;
else
  make insertion on LP_b;



(a) Picture 1 (b) Picture 2

Figure 3: model3 view

Figure 4: shows amelioration we think for model3 by using the same picture as model2.

- On Picture 1 the nodes 1 has two packets lists on emission. It's important to notice that the packets c which is not full is used by best effort. In this, the best effort what we put in is the head of packet d. d is looks full again because we put the e in it and f in queue of e. That's why f looks less full of best effort.
- On picture 2 we can see the result of what is done after three ST.

1.6 Observation

on this part we'll show how the model's implementation we describe above runs, note the strengths and the critical cases for each of them.

1.6.1 Model 1

On model1 the observations is about how the initial state and slot capacities or the load of ring during simulation.

1.6.1.1 Starting at same moment: on this this paragraph show what happened when all node start emission at the same time.

1.7 Model for interconnection between two ring

The second part of our work is about ring's interconnection. The problem that we have to solve is about lost of synchronization. When packets need to change node, difference between the slot's time click on the rings. we must point out our attention on the **switch** node.
The node switch can erase container if it gets packets for another rings. So it means that he is a new transmitter of the packets. It's now important to think about how to use the container on the switch nodes. We can do it by using more ways.

1.7.1 Priority to external packets

In this case, it is considered that on the switch node, priority will be given to the packets coming from an external node.The purpose of this is to minimize the additional waiting time for the packets on the waiting line of switch node. This could have very good results especially if inter-ring communications are even more numerous.

1.7.2 Priority to internal packets

In this case, it is considered that all priority is given to the packets inside the same ring. This results can be used when there isn't more packets between two rings.

1.7.3 Mixed Priority

With this model, we combine the two previous. So it's we can define an probabilistic aspect which can define when we get priority for internal packets or not. For sample, let's define $\Delta \in [0;1]$;

Algorithm 8 choose priority

Require: $\Delta \in [0;1]$
if $\Delta < 0.5$ **then**
    return Priority to internal packets
else
    return Priority to external packets

1.7.4 Observations

In this part we show the main problem we explain in interconnection case and describe what we exactly do.

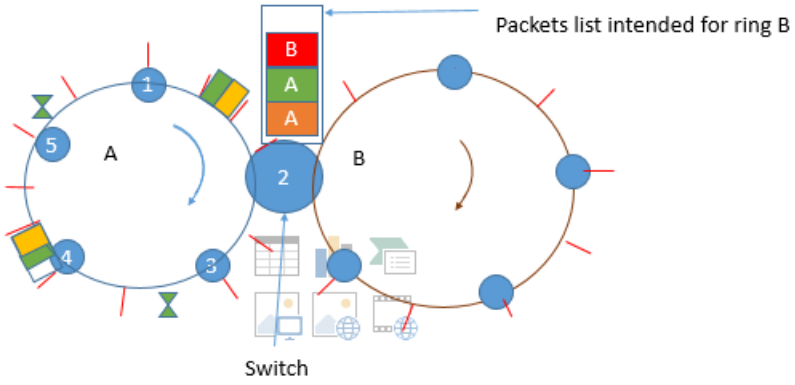
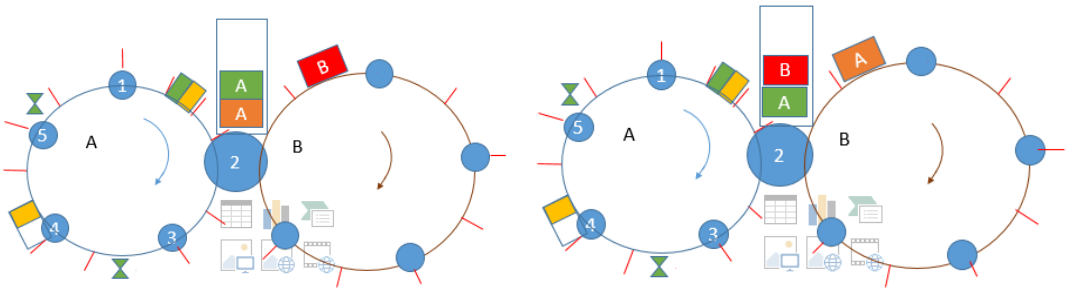


Figure 4: Rings interconnection

Figure 3: presents the model that we describe above and show what happen on switch node. Here we show on node 2 a sample of move on this node. Node 2 packet's waiting list. On the list we have as much packets coming from ring A to B as packets which are introduced for ring B.



(a) Internal priority (b) external priority

Figure 5: Priority choice

Figure4: show us the different possibility for the switch to manage the list. In (a) packets which source is the ring B is chosen to be put on the slot but in (b), the stranger pass at first.

Let's consider T_A , T_B the time that need packets turn around ring A and B respectively. And also consider T_R as real time for packets transmission. The goal to obtain: $T_R \leq T_A + T_B$.