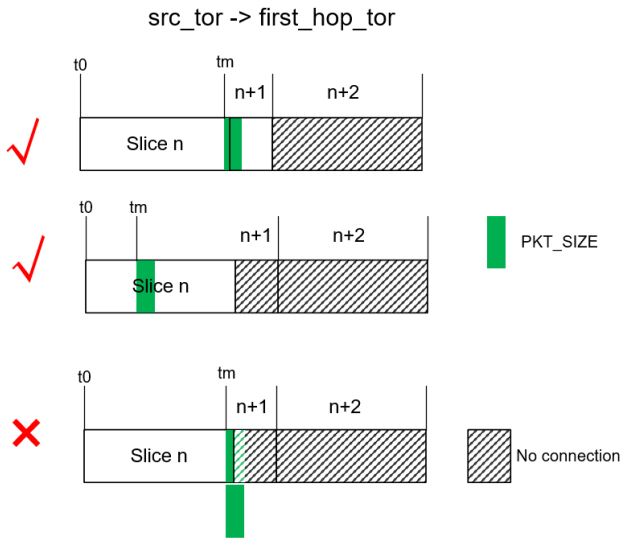
Real-Time Routing logic in C++

The off-line routing gives us the best path when sending a packet at the beginning of a slice. But in simulation, a packet might arrive at any time. We need to realize real-time routing in the simulation.

Instruction:

Image a packet ***pkt*** arrives its source tor ***src_tor*** at time ***tm***. And ***tm*** belongs to slice ***n*** ($n \in [0, 215]$) .Its destination tor is ***dst_tor***. Using (***src_tor, dst_tor, n***), we could get the first hop of the best path. Let's say it is ***first_hop_tor***. However, the best path is calculated assuming ***pkt*** is sending at the beginning of slice ***n (t0)***. We need to check if we could send ***pkt*** to ***first_hop_tor*** in slice ***n*** with the arriving time ***tm*** (***pkt*** might span two or three slices. We say slice ***n*** could transmit ***pkt*** if the first bit of ***pkt*** is in slice ***n***). See the following figure.



Part I: Main logic

If ***pkt*** could be transmitted in slice ***n***:

Transmit ***pkt*** to ***first_hop_tor;***

Now we have a new source tor ***src_tor'*** and new slice ***n'***. Repeat the whole logic again until ***pkt*** arrives ***dst_tor;***

Else:

Wait for next slice ***n+1***. Calculate the best path using (***src_tor, dst_tor, n+1***) and get the ***new_first_hop_tor***. Transmit ***pkt*** to ***new_first_hop_tor*** and repeat the whole logic**.**

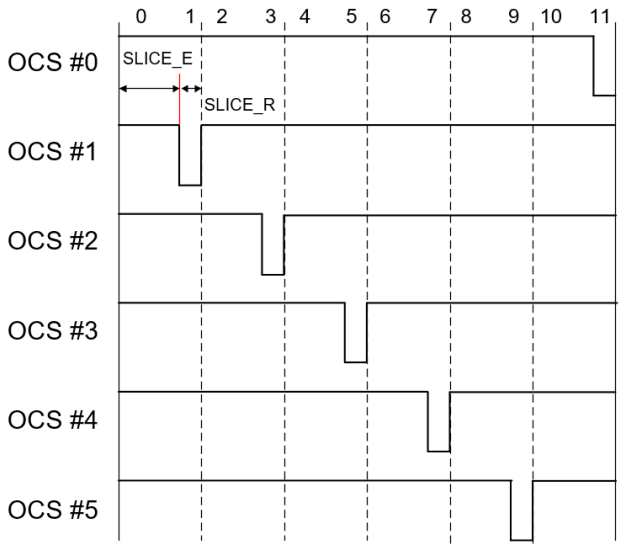
Part II: How to know if *pkt* could be transmitted in slice *n.*

(This part is quite complex and not sure if we need such logic implementations in C++)

Preliminaries:

(1) In the following figures, ***E*** and ***R*** stand for **SLICE_E** and **SLICE_R,** respectively.

(2) A tor to tor connection accounts for 6 superslices, which means 12 slices in total (6 **SLICE_E** + 6 **SLICE_R)**. During 12 slices, active time is 6 **SLICE_E** + 5 **SLICE_R** and one **SLICE_R** is for reconfiguration. See the following figure.



(3) We assume that SLICE_E should be enough to transmit a whole packet (PKT_SIZE = 1200ns). SLICE_E >= 1200ns. SLICE_R could be any value.

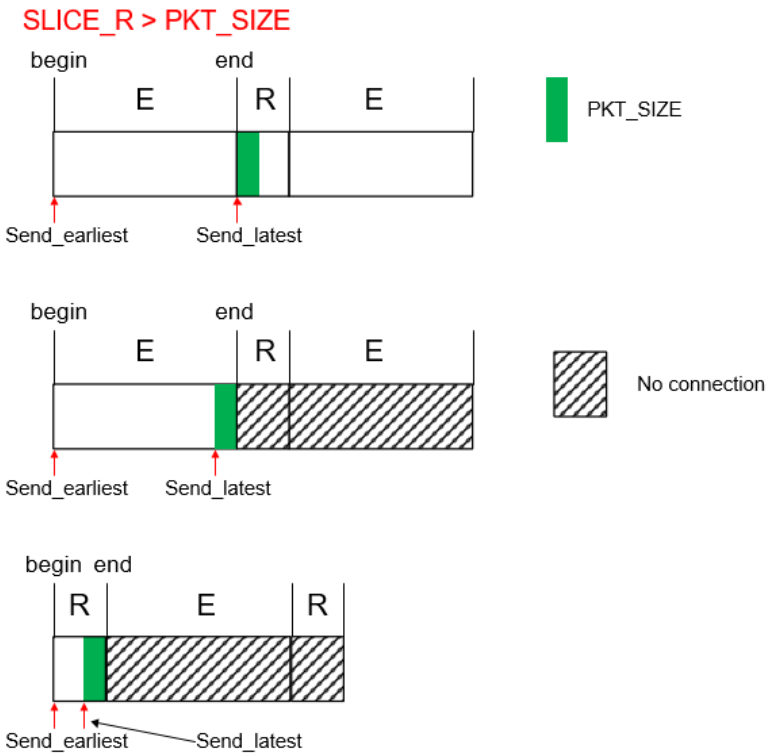
Since ***pkt*** might span two or three slices, we need to know if src_tor to first_hop_tor has connections in next two slices.

is_next_1 = is_have_next_slice(***n, src_tor, first_hop_tor***)

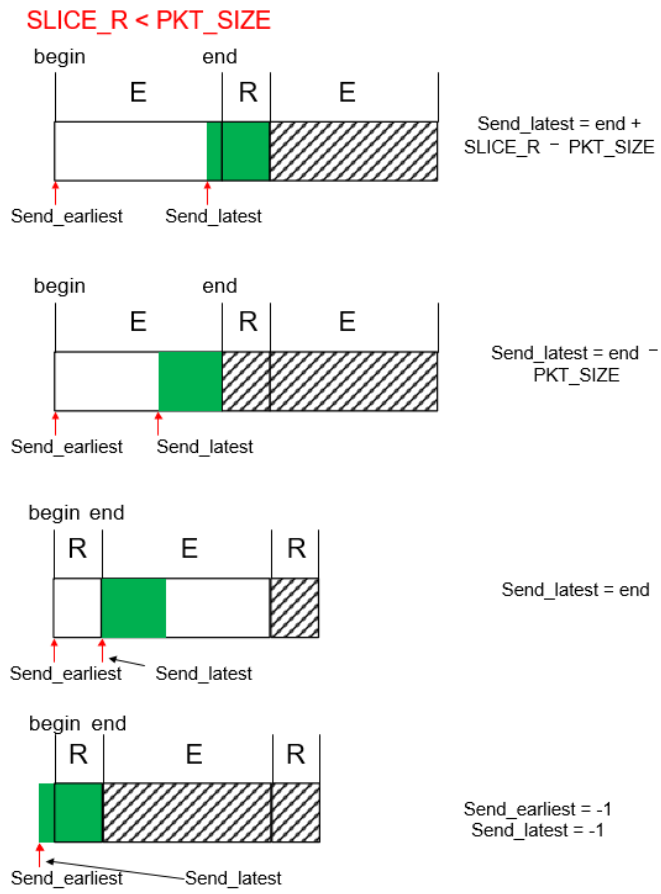
is_next_2 = is_have_next_slice(***n+1, src_tor, first_hop_tor***)

In the following, we will calculate the earliest sending and latest sending time during slice ***n***. If arriving time ***tm*** <= latest sending time, then we could use slice ***n*** to transmit ***pkt***. Otherwise NOT.

Case I: SLICE_R >= PKT_SIZE



Case II: SLICE_R < PKT_SIZE



Code on calculating earliest/latest sending time (early_send/ late_send)

```
begin, end = cal_slice_begin_end(slice_n)
if SLICE_R > PKT_SIZE:
    early_send = begin
    if is_next_1:
        late_send = end
    else:
        late_send = end - PKT_SIZE
else:   # SLICE_R < PKT_SIZE
    if slice_n % 2 == 0:        # SLICE_E
        early_send = begin
        if is_next_1:
            if is_next_2:       # E R E
                late_send = end
            else:               # E R
                late_send = end + SLICE_R - PKT_SIZE
        else:                   # E
            late_send = end - PKT_SIZE
    else:       # SLICE_R
        if is_next_1:           # R E
            early_send = begin
            late_send = end
        else:                   # R
            early_send = -1
            late_send = -1
return early_send, late_send
```