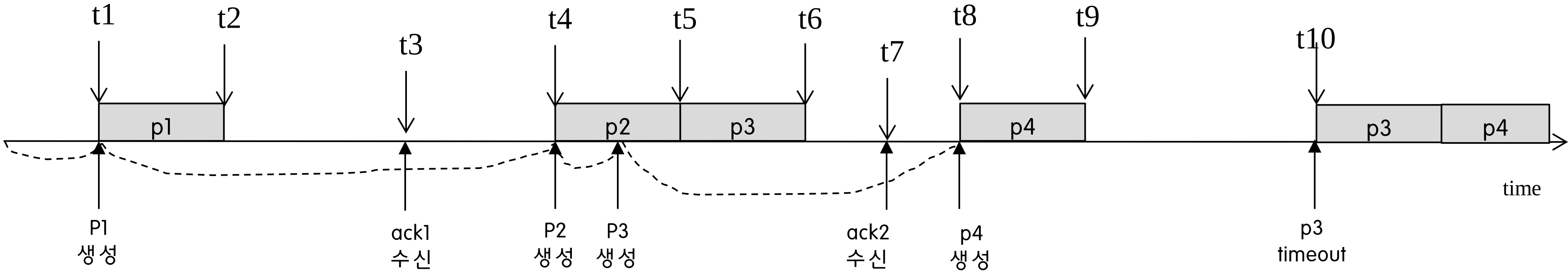
Homework-2

Error and Flow Control Simulation on Point-to-Point Link

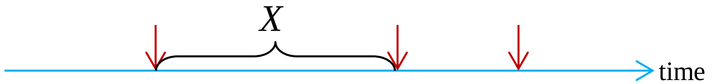
Go-Back-N + Sliding-Window

- One-way transmission
  (from one sender to one receiver)
- ACK(success)
- Timeout (failure)
- Assumption:
  - no processing time
  - no ACK error

Event-driven Simulation



Simulation Parameters



$\Pr\{X \le t\} = 1 - e^{-\lambda t}$

$e^{-\lambda t} = x$, which is a random number between 0 and 1

$-\lambda t = \log_e(x)$

$t = -\frac{1}{\lambda}\log_e(\text{random()})$

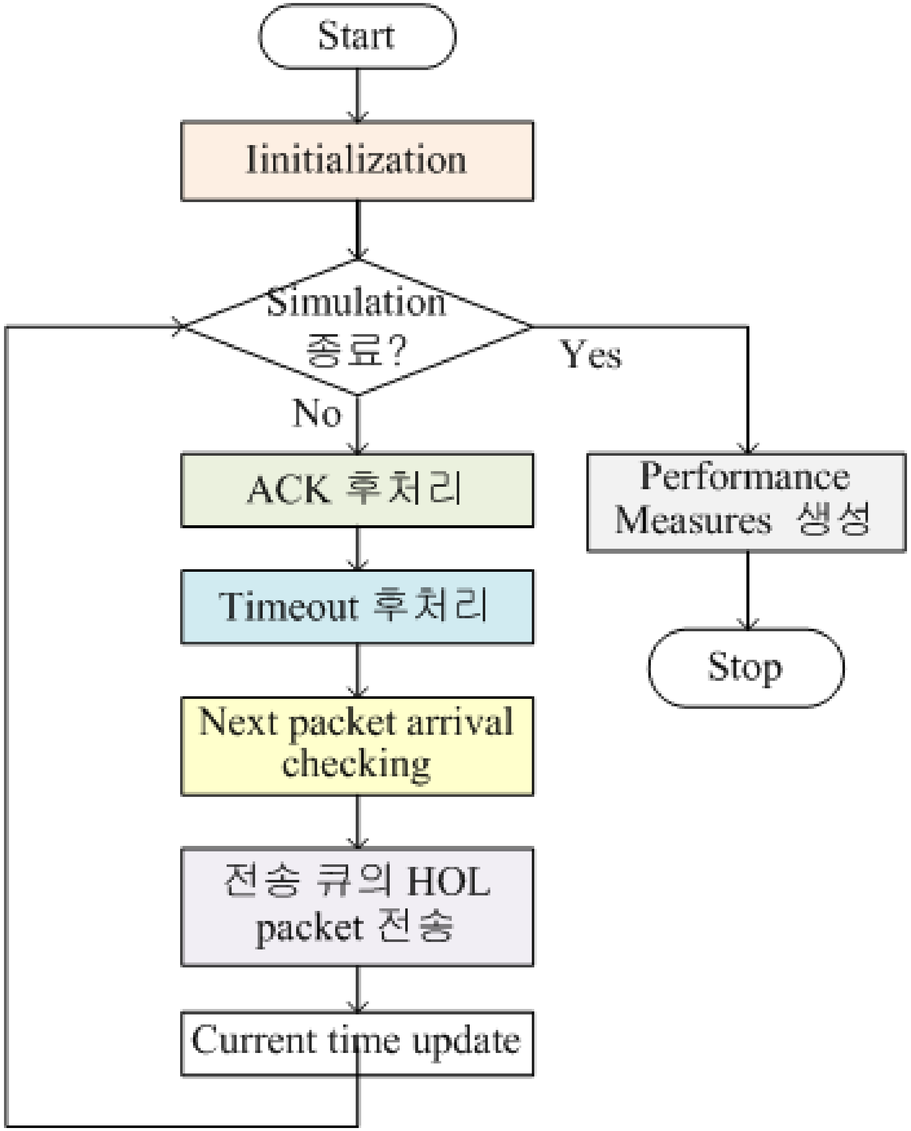
- Input Parameters
  - Sliding-Window Size: W
  - Packet arrival process: Poisson with rate λ

    packet inter-arrival time: exponential distribution

    $-\frac{1}{\lambda}\ln(x)$, where x is a random number between 0 and 1.
  - Packet transmission time: t_pk
  - Packet(*i.e.*, frame) transmission error probability: p
  - Ratio of link propagation time to packet transmission time: a

    (Link propagation delay: t_pro = $a \times$ t_pk)
  - Under load condition: W < 2a+1
- Performance Measures (Outputs)
  - Packet transmission delay
  - Utilization

Simulation Flow Chart

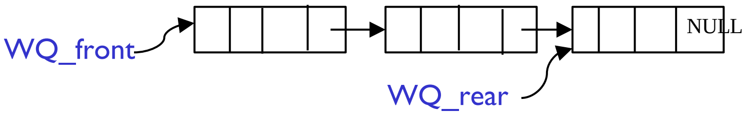


Data-Packet Queue Structure

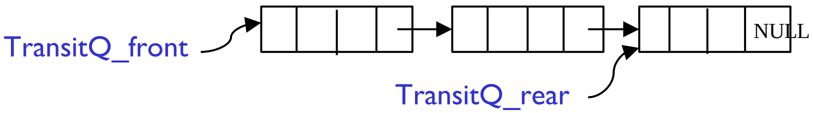
sn	gentm	t_out	link

- sn: sequence number
- gentm: generation (arrival) time of a packet
- t_out: timeout

[전송되기를 기다리고 있는 패킷]



[전송했지만 ACK를 아직 받지 못한 패킷]



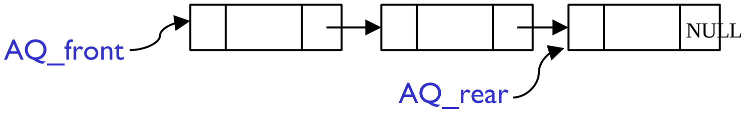
```
struct pk_list{
   long  sn;
   double gentm;
   double  t_out;
   struct pk_list  *link;
}
typedef struct pk_list   DataQue;
DataQue   WQ_front, WQ_rear;
DataQue   TransitQ_front, TransitQ_rear;
```

ACK Queue Structure

sn	ack_rtm	link

- sn: sequence number
- ack_rtm: reception time of an ACK at sender

[수신측에서 보냈지만 아직 송신측에서 처리 되지 않는 ACK]



```
struct ack_list{
   long  sn;
   double  ack_rtm;
   struct ack_list  *link;
}
typedef struct ack_list   AckQue;
AckQue   AQ_front, AQ_rear;
```

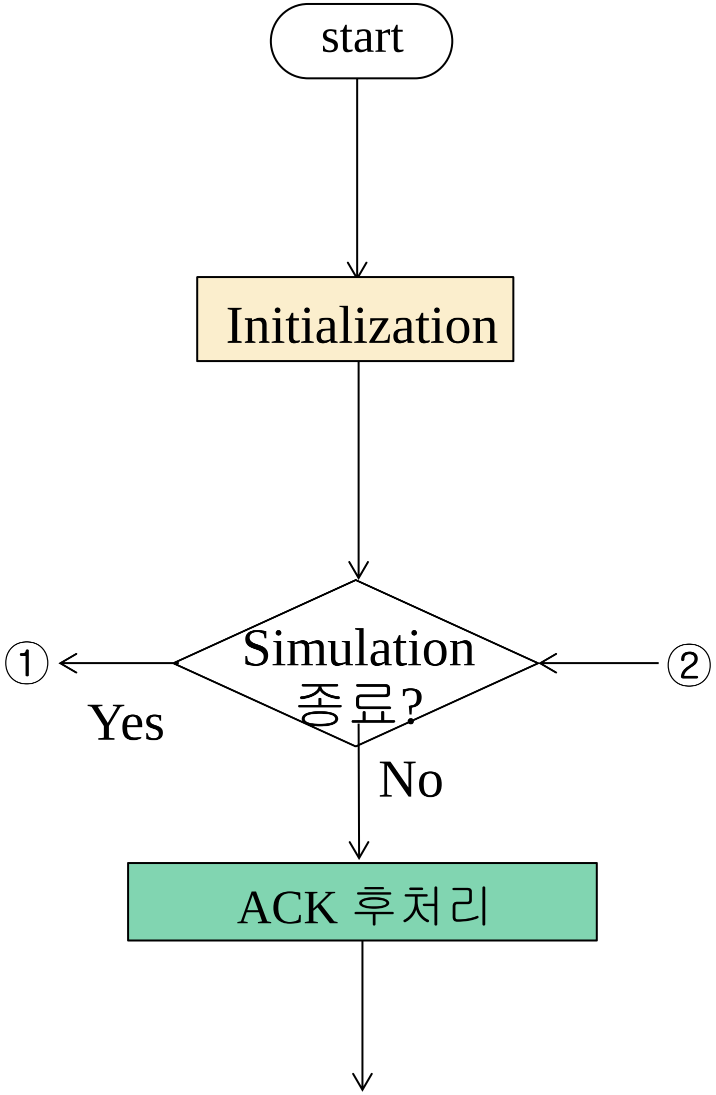
```
#include <sidio.h>
#include <std.lib>
#include <math.h>
       ⋮

struct pk_list{
   long  sn;
   double gentm, timeout;
   struct pk_list  *link;
}
typedef struct pk_list   DataQue;
DataQue  WQ_front, WQ_rear;
DataQue TranitQ_front, TransitQ_rear


struct ack_list{
   long  sn;
   double  ack_rtm;
   struct ack_list  *link;
}
typedef struct ack_list   AckQue;
AckQue   AQ_front, AQ_rear;
```

```
long  seq_n=0; transit_pknum=0;
long  next_acksn=0;
double  cur_tm, next_pk_gentm;
double  t_pknum=0, t_delay=0;

long  N;
double  timeout_len;
int W;
float a, t_pk, t_pro;
float lamba, p;
```

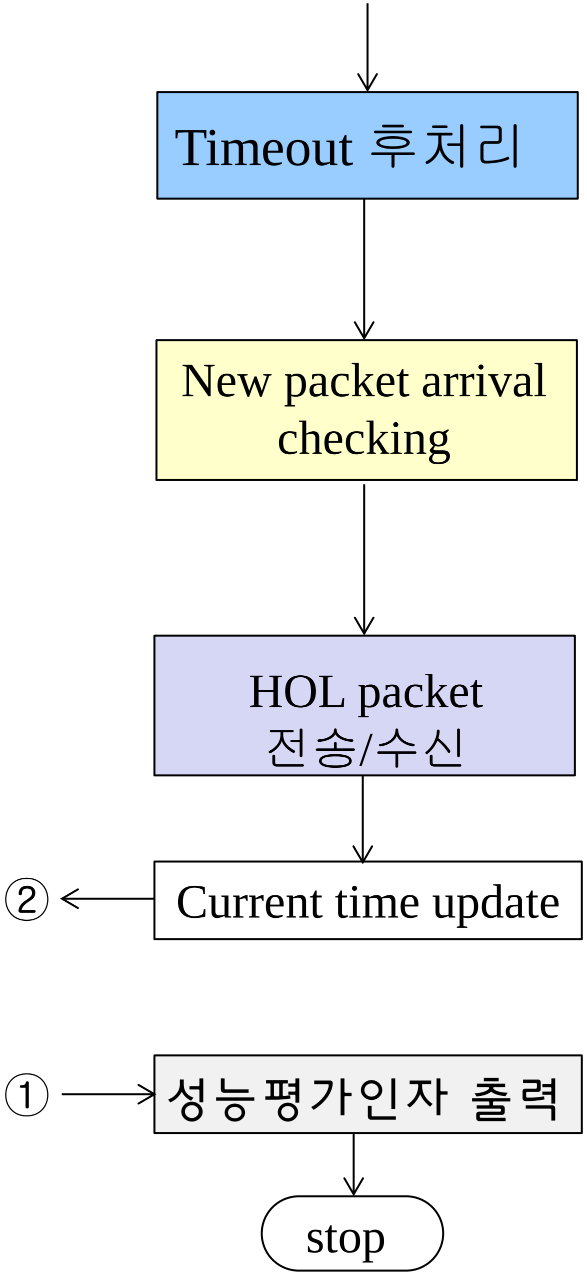
시뮬레이션 시간: 처리되는 패킷 수

Input Parameters

```
float random(void);
void pk_gen(double);
void suc_transmission(long);
void re_transmit(void);
void transmit_pk(void);
void receive_pk(long, double);
void enque_Ack(long)
void cur_tm_update(void);
void print_performance_measure(void);

                 ⋮
```

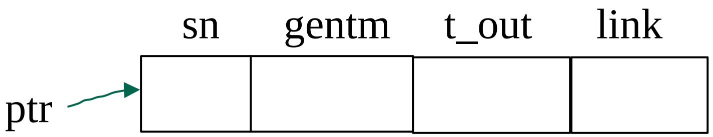


```
void main(void)
{
     /* input parameter setting  */
          ⋮
     WQ_front = WQ_rear = NULL;
     TransitQ_front=TransitQ_rear=NULL;
     AQ_front = AQ_rear = NULL;
                                        0과 1 사이의 난수발생함수
     cur_tm = -log(random())/lambda;
     pk_gen(cur_tm);
     next_pk_gentm = cur_tm -log(random())/lambda;
                                         packet inter-generation time
     while (t_pknum<=N)   {

          while (AQ_front != NULL)
               if (AQ_front->ack_rtm <=cur_tm) {
                    suc_tranmission(AQ_front->sn)
                    deque_Ack();
               }
               else break;
```



```
if (TransitQ_front != NULL)
    if (TransitQ_front->t_out <= cur_tm)
       re_transmit();

 while (next_pk_gentm <= cur_tm) {
     pk_gen(next_pk_gentm);
     next_pk_gentm += -log(random())/lambda;
  }

  if ((WQ_front != NULL) && (transit_pknum <W))  {
      transmit_pk();
      receive_pk(TransitQ_rear->sn, TransitQ_rear->gentm);
  }

   cur_tm_update();

}    /*   simulation loop   */

print_performance_measure();

}
```

```
void pk_gen(double tm)
{
        DataQue    *ptr;

        ptr = malloc(sizeof(DataQue));
        ptr->sn = seq_n;
        ptr->gentm = tm;
        ptr->link = NULL;
        seq_n++;

        if (WQ_front == NULL)
                WQ_front = ptr
        else WQ_rear->link = ptr;
        WQ_rear = ptr;
}
```



- 생성된 패킷을 WQ의 맨 뒤에 삽입

```
void suc_transmission(long sn)
{
      DataQue   *ptr;
      AckQue    *aptr;

      ptr = TransitQ_front;
      if (ptr->sn == sn) {
           TransitQ_front = TransitQ_front->link;
           if (TransitQ_front == NULL)
                 TransitQ_rear = NULL;
           free(ptr);
           transit_pknum--;
      }

      aptr = AQ_front;
      AQ_front=aptr->link;
      if (AQ_front == NULL) AQ_rear = NULL;
      free(aptr);
}
```

[ACK 수신: 패킷의 성공적 전송을 의미]

- ack를 받은 패킷: Transit_Q에서 제거
- Transit_Q에 있는 패킷 수: 1 감소

- 수신한 ACK: AQ에서 제거

```
void re_transmit(void)
{
     TransitQ_rear->link=WQ_front;
     if (WQ_front==NULL)
             WQ_rear=TransitQ_rear;
     WQ_front=TransitQ_front;
     TransitQ_front = TransitQ_rear=NULL;

     transit_pknum=0;
}
```

- Transit_Q의 모든 패킷을 WQ의 앞에 삽입
- Empty Tansit_Q
  - transit_pknum=0

```
void transmit_pk(void)
{
      DataQue  ptr;
      cur_tm+=t_pk;
      WQ_front->t_out=cur_tm+timeout_len;
      ptr=WQ_front;
      WQ_front = WQ_front->link;
      if (WQ_front==NULL) WQ_rear=NULL;
      if (TransitQ_front==NULL)
            TransitQ_front=ptr
      else TransitQ_rear->link=ptr;
      ptr->link=NULL;
      TransitQ_rear=ptr;
       transit_pknum++;
}
```

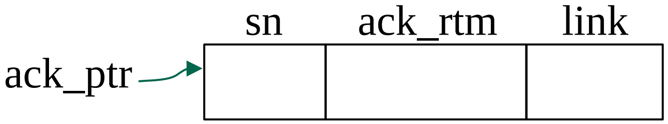
[WQ의 첫 패킷 전송]
- current time update
- 막 전송한 패킷의 timeout 시간 설정
- 전송한 패킷을 WQ에서 Tranit_Q의 맨 뒤로 이동
- Transit_Q에 있는 패킷 수: 1 증가

```
void receive_pk(long seqn, double gtm)
{
     if (random() > p)    //  전송성공?
        if (next_acksn == seqn) {
            t_delay += cur_tm+t_pro –gtm;
            t_pknum++;
            next_acksn++;
            enque_Ack(seqn);
        }
}
```

[Receiver 작업]

- 수신된 패킷: error 발생 유무 check
- 순서에 맞는 패킷인지 check
- 누적패킷 수: 1증가
- 누적 패킷지연: 수신 패킷의 지연시간 추가
- Ack 생성하여 AQ의 뒤에 삽입

```
void enque_Ack(long seqn)
{
      AckQue    *ack_ptr;

      ack_ptr = malloc(sizeof(AckQue));
      ack_ptr->sn = seqn;
      ack_ptr->ack_rtm = cur_tm + 2*t_pro;
      ack_ptr->link = NULL;

      if (AQ_front == NULL)
                AQ_front = ack_ptr;
     else AQ_rear->link = ack_ptr;
     AQ_rear = ack_ptr;
}
```



- Ack 패킷을 생성
- AQ의 맨 뒤에 삽입

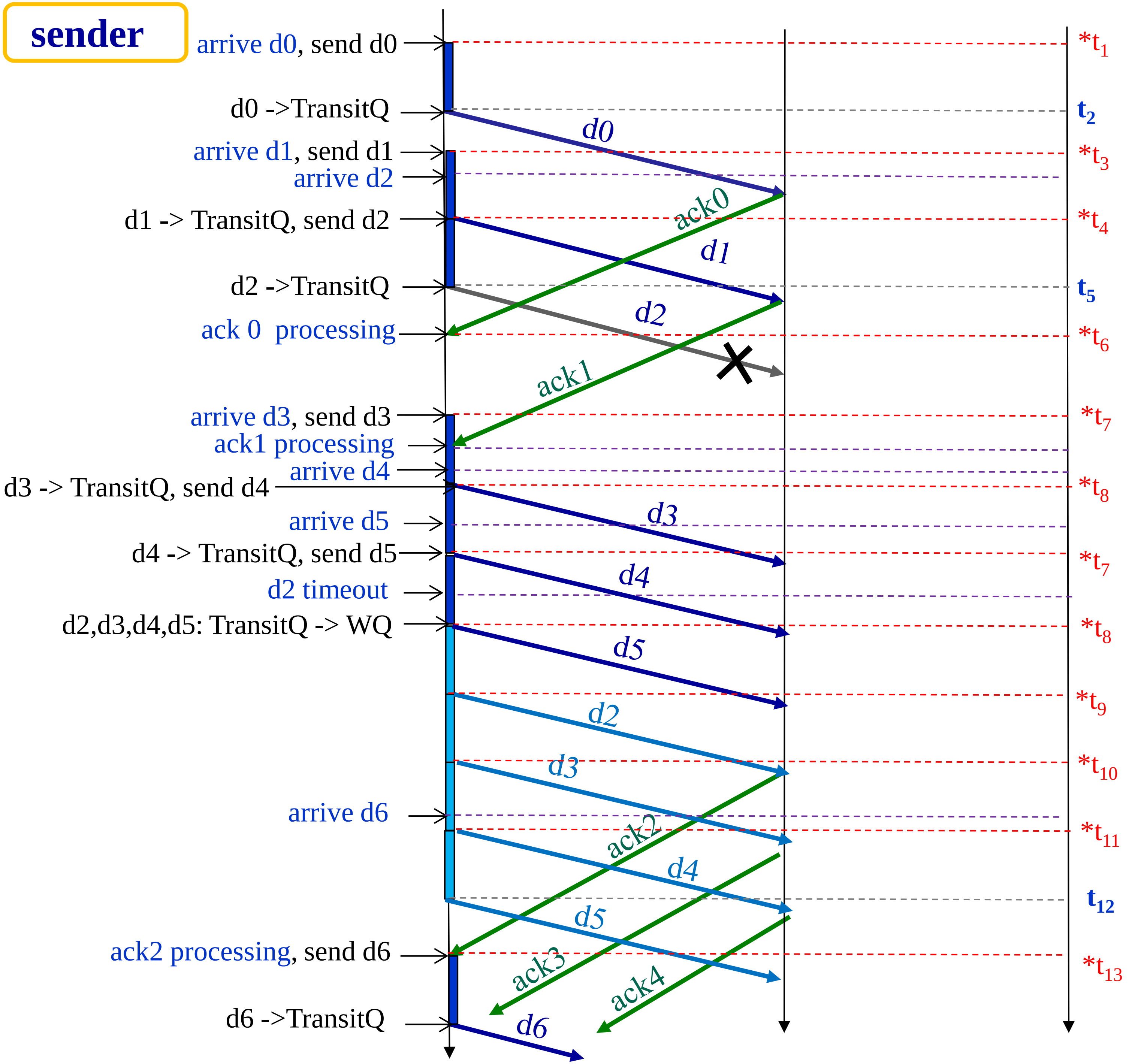
```
void cur_tm_update(void)
{
    double tm;

    if  ((WQ->front !=NULL) &&| (transit_pknum<W)) return;
    else
    {
        if (AQ_front == NULL)
              tm=next_pk_gentm
        else  if (AQ_front->ack_rtm<next_pk_gentm)
              tm=AQ_front->ack_rtm
        else  tm=next_pk_gentm;

        if (TransitQ_front != NULL)
            if (TransitQ_front->t_out<tm)
                tm=TransitQ_front->t_out;

        if  (tm>cur_tm) cur_tm=tm;
    }
}
```

이미 생성되어 전송을 기다리고 있는 패킷 존재하고
window가 닫히지 않았다면: 현재 시간을 그대로 유지

Ack 수신, new packet 생성, timeout 중
가장 일찍 발생한 event 시간: tm

```
void print_performance_measure(void)
{
        double util;
        double m_delay;

        m_delay = t_delay/t_pknum;
        util = (t_pknum*t_pk)/simul_tm;

      /*  print  input parameters and
          performance measures   */

                  ⋮

}
```

```
float random(void)
{
      float   rn;

     /* random number generation
        between 0 and 1   */

                  ⋮

      return(rn);
 }
```



sender

arrive d0, send d0

d0 ->TransitQ

arrive d1, send d1

arrive d2

d1 -> TransitQ, send d2

d2 ->TransitQ

ack 0 processing

arrive d3, send d3

ack1 processing

arrive d4

d3 -> TransitQ, send d4

arrive d5

d4 -> TransitQ, send d5

d2 timeout

d2,d3,d4,d5: TransitQ -> WQ

arrive d6

ack2 processing, send d6

d6 ->TransitQ

Simulation time update

- Loop 시작 때
- Loop 내:
  패킷전송 후

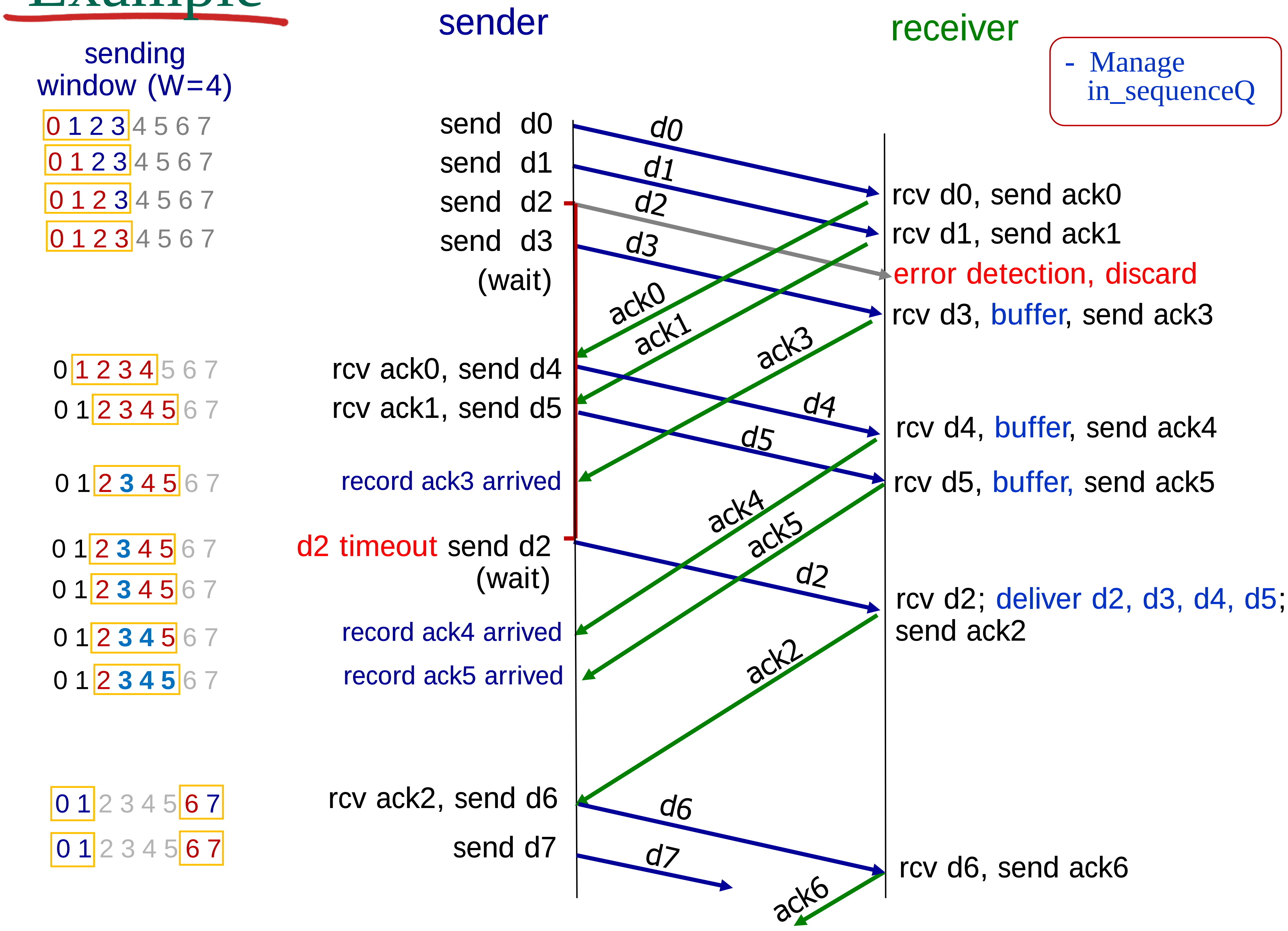
Comparison Scheme

Selective repeat
+ Sliding-Window

- One-way
- ACK + Timeout (without ACK)
- Assumption: no processing time, no ACK error

Example

- Manage in_sequenceQ



sending window (W=4)

sender

receiver

0 1 2 3 4 5 6 7 send d0

0 1 2 3 4 5 6 7 send d1

0 1 2 3 4 5 6 7 send d2

0 1 2 3 4 5 6 7 send d3

(wait)

rcv d0, send ack0

rcv d1, send ack1

error detection, discard

rcv d3, buffer, send ack3

0 1 2 3 4 5 6 7 rcv ack0, send d4

0 1 2 3 4 5 6 7 rcv ack1, send d5

rcv d4, buffer, send ack4

0 1 2 3 4 5 6 7 record ack3 arrived

rcv d5, buffer, send ack5

0 1 2 3 4 5 6 7 d2 timeout send d2

0 1 2 3 4 5 6 7 (wait)

rcv d2; deliver d2, d3, d4, d5; send ack2

0 1 2 3 4 5 6 7 record ack4 arrived

0 1 2 3 4 5 6 7 record ack5 arrived

0 1 2 3 4 5 6 7 rcv ack2, send d6

0 1 2 3 4 5 6 7 send d7

rcv d6, send ack6

Performance Comparison by Simulation between Go-back-N and Selective Repeat

- Report
  - Introduction
  - Scheme description
  - Performance parameters (W, p, a, λ)
  - Performance comparison
    - performance tables
    - discussion
- 기한: 5월 4일

Performance Tables

- For three load conditions (low, medium, heavy)

	Packet delay	channel utilization
Go-back-N		
Selective Repeat		