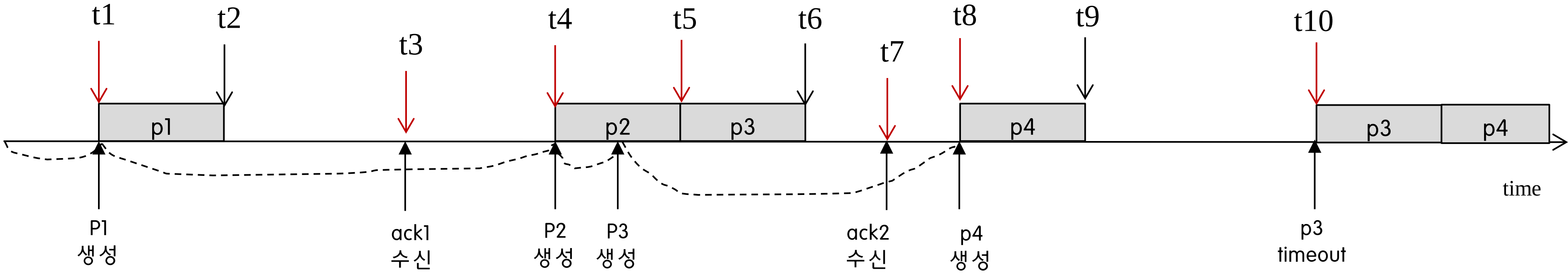
Homework-2

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

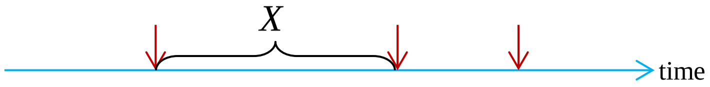
One-way Go-Back-N + Sliding-Window

- Success: ACK
- Failure: Timeout
- 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

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

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

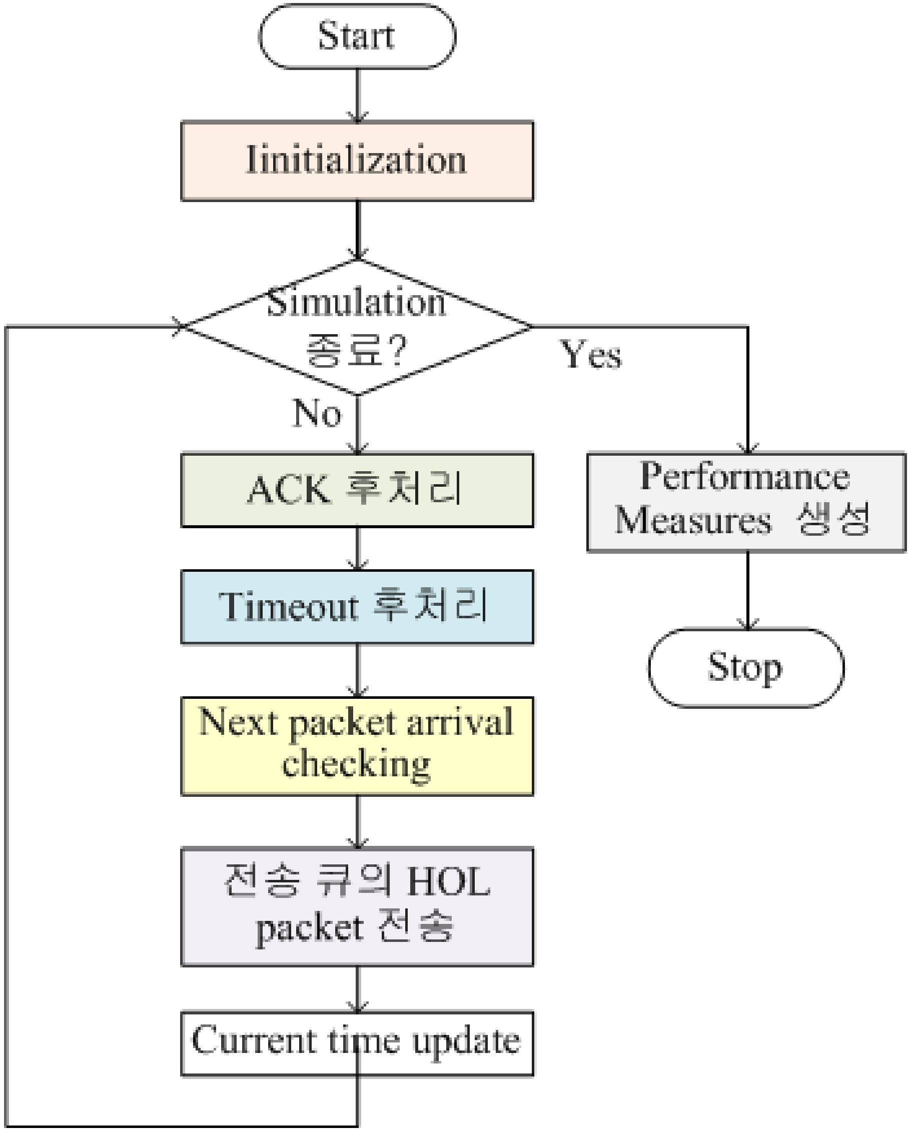
- Input Parameters
  - Sliding-Window Size: W
  - Packet arrival process: Poisson with rate λ

    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 × t_pk)
  - Under load condition: W < 2a+1
- Performance Measures (Outputs)
  - Packet transmission delay
  - Utilization

Simulation Flow Chart

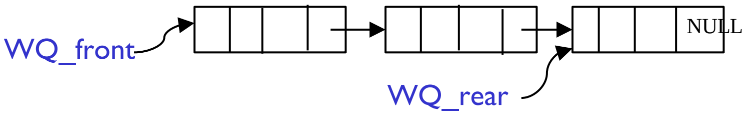


Data-Packet Queue Structure

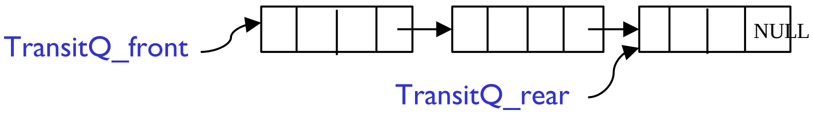
sn	gentm	t_out	link

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

[전송되기를 기다리고 있는 패킷]



[전송했지만 ACK를 아직 받지 못한 패킷]



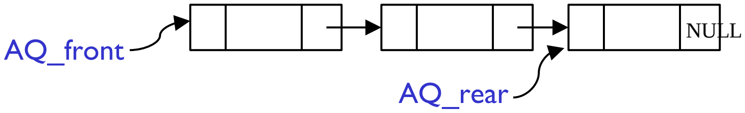
```
struct pk_list{
   long  sn;
   double gentm;
   double  t_out;
   struct pk_list  *link;
}
typedef struct pk_list   DataQue;
DataQue   WQ_front, WQ_rear;
DataQue   TransitQ_front, TransitQ_rear;
```

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;
```

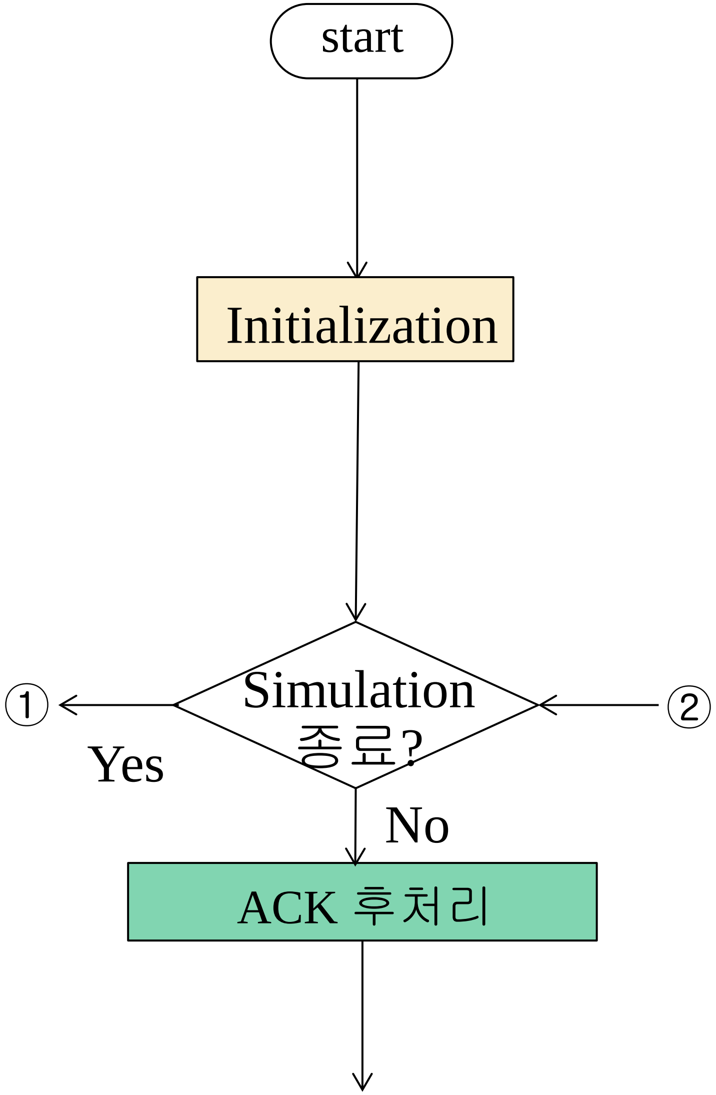
```
long  seq_n=0; transit_pknum=0;
long  next_acksn=0;
double  cur_tm, next_pk_gentm;
double  t_pknum=0, t_delay=0;
```

◀----시뮬레이션 시간: 처리되는 패킷 수

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

Input Parameters

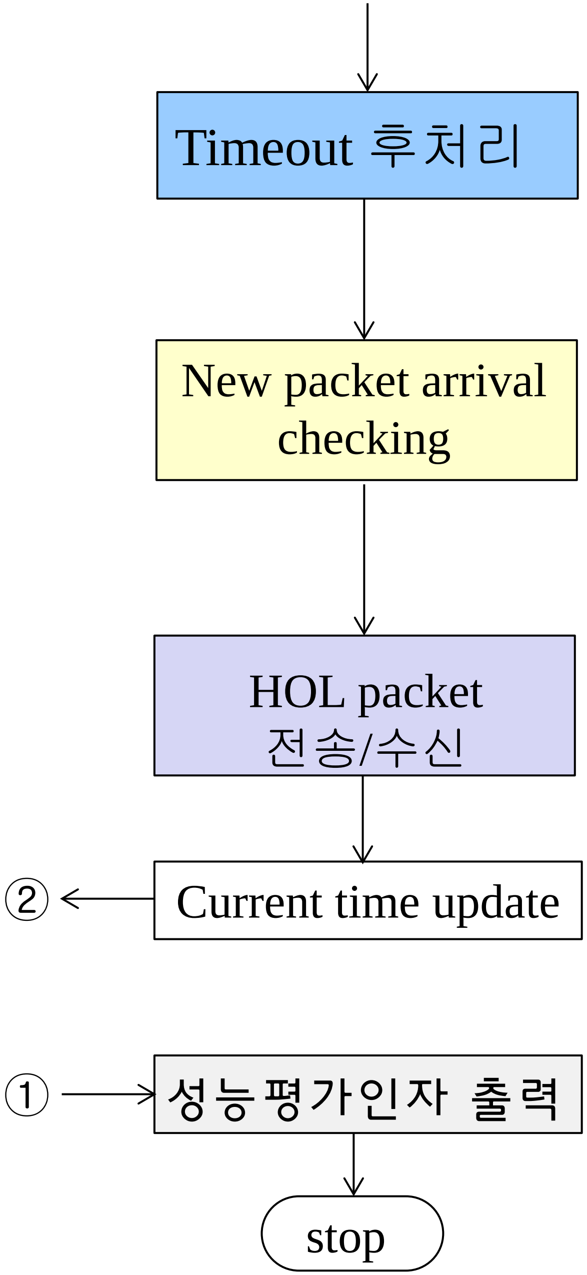
```
float random(void);
void pk_gen(double);
void suc_transmission(long);
void re_transmit(void);
void transmit_pk(void);
void receive_pk(long, double);
void enque_Ack(long)
void cur_tm_update(void);
void print_performance_measure(void);

          ⋮
```



```
void main(void)
{
    /* input parameter setting  */
          ⋮
    WQ_front = WQ_rear = NULL;
    TransitQ_front=TransitQ_rear=NULL;
    AQ_front = AQ_rear = NULL;

    cur_tm = -log(random())/lambda;
    pk_gen(cur_tm);
    next_pk_gentm = cur_tm -log(random())/lambda;


    while (t_pknum<=N)   {

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

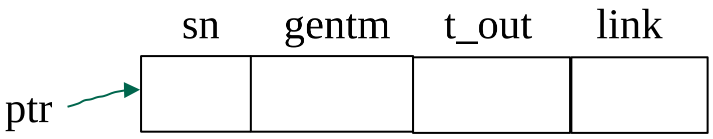
0과 1사이의 난수발생함수

packet inter-generation time



```
    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
- Tansit_Q: empty
- 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]

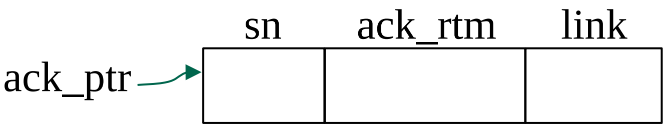
- current time update
- 막 전송한 패킷의 timeout 시간 설정
- 전송한 패킷을 WQ에서 Tranit_Q의 맨 뒤로 이동
- Transit_Q에 있는 패킷 수: 1 증가

```
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
- 누적 패킷지연: 수신 패킷의 지연시간 추가
- 누적패킷 수: 1증가
- Ack 생성하여 AQ의 뒤에 삽입하는 함수 call

```
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

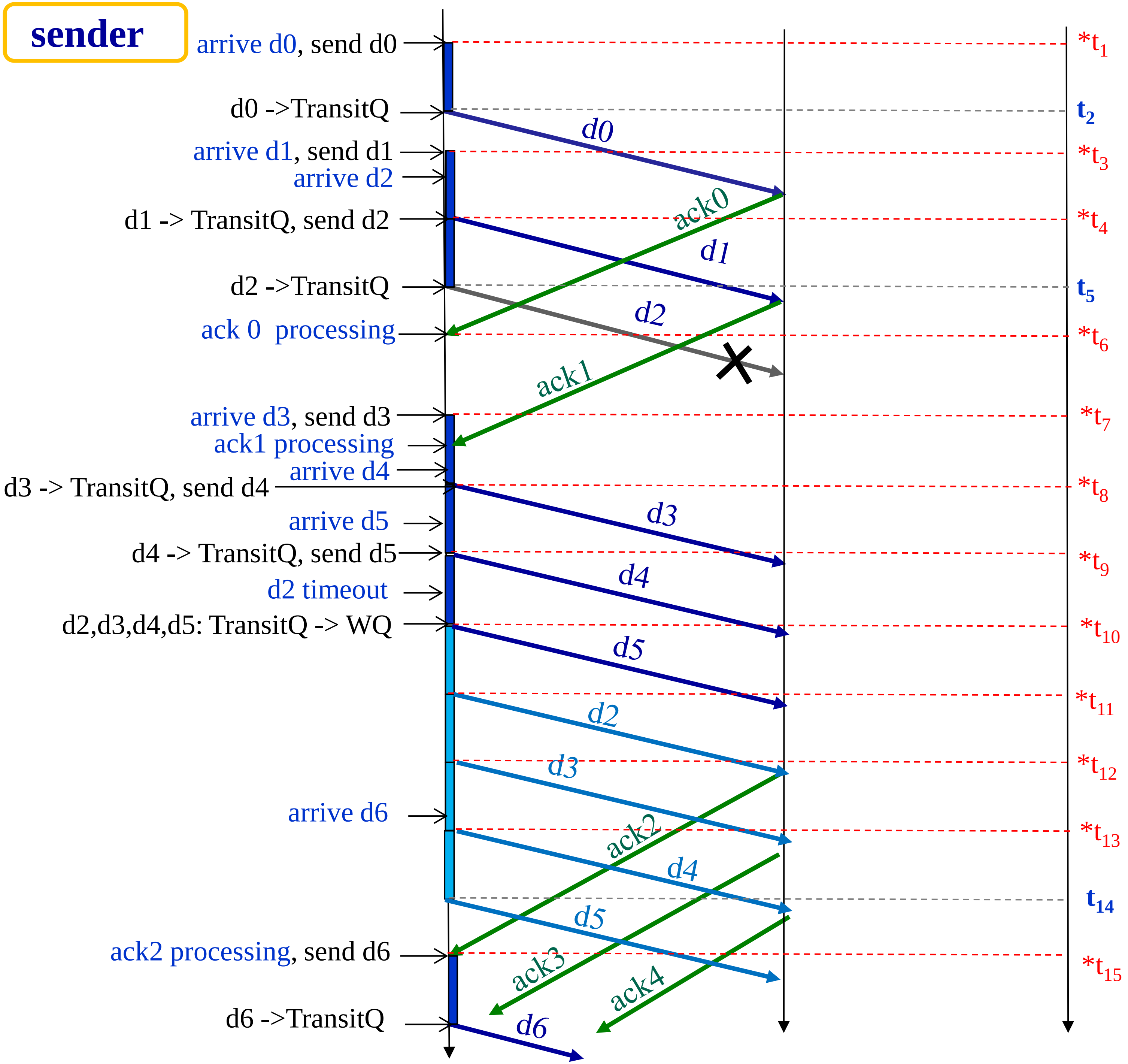
```
void cur_tm_update(void)
{
     double tm;

     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 *t_1

d0 ->TransitQ t_2

arrive d1, send d1 *t_3

arrive d2

d1 -> TransitQ, send d2 *t_4

d2 ->TransitQ t_5

ack 0 processing *t_6

arrive d3, send d3 *t_7

ack1 processing

arrive d4

d3 -> TransitQ, send d4 *t_8

arrive d5

d4 -> TransitQ, send d5 *t_9

d2 timeout

d2,d3,d4,d5: TransitQ -> WQ *t_{10}

*t_{11}

*t_{12}

arrive d6

*t_{13}

t_{14}

ack2 processing, send d6 *t_{15}

d6 ->TransitQ

Simulation time update

- Loop 시작 직전
- Loop 내: 패킷전송 후

Homework 2

Performance comparison between Go-back-N and Selective Repeat by simulation

- When successfully receiving a data frame, the receiver sends ACK (nothing for erroneous data frame)
- Transmission failure of data frame or ACK frame: the sender does not receive ACK until the timer expires
- Assumption:
  - no processing time

Homework-2

- Report
  - Introduction
  - Scheme description
  - Performance parameters (W, a, λ, p_{data}, p_{ack})
  - Performance comparison
    - performance tables
    - discussion

- 기한:

Performance Tables

- For load conditions (low, medium, heavy)

	Packet delay	channel utilization
Go-back-N		
Selective Repeat		

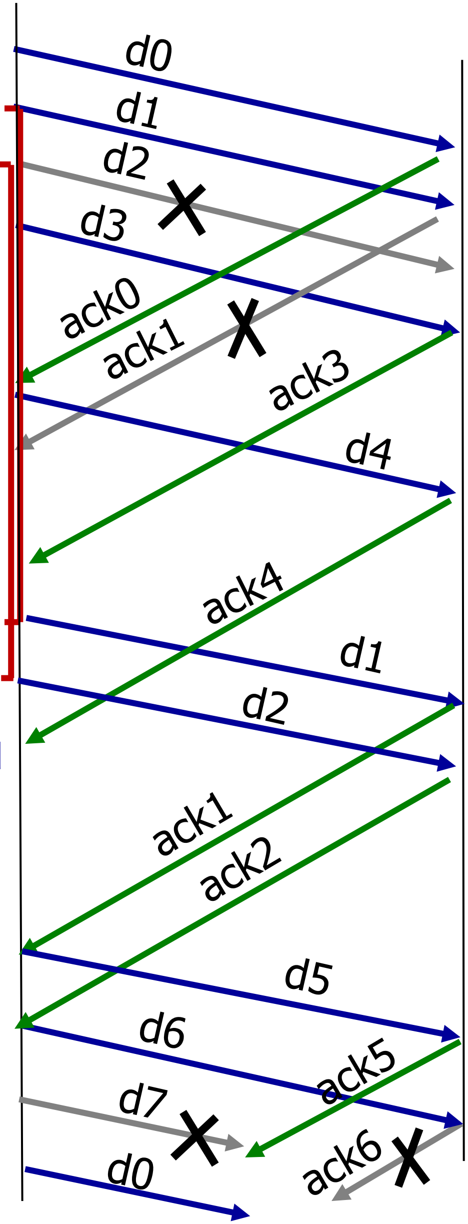
SR (ACK error) Example

- 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)

0 1 2 3 4 5 6 7 — rcv ack0, send d4

0 1 2 3 4 5 6 7 — record ack3 arrived

0 1 2 3 4 5 6 7 — d1 timeout send d1

0 1 2 3 4 5 6 7 — d2 timeout send d2

0 1 2 3 4 5 6 7 — record ack4 arrived

0 1 2 3 4 5 6 7 — rcv ack1, send d5

0 1 2 3 4 5 6 7 — rcv ack2, send d6

0 1 2 3 4 5 6 7 — send d7

0 1 2 3 4 5 6 7 — send d0

rcv d0, send ack0

rcv d1, send ack1

error detection, discard

rcv d3, buffer, send ack3

rcv d4, buffer, send ack4

send ack1, discard d1

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

rcv d5, send ack5

rcv d6, send ack6