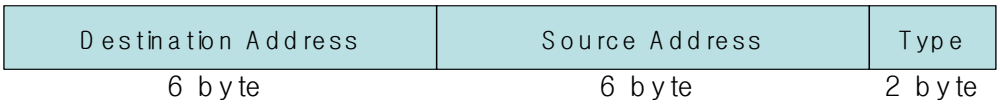
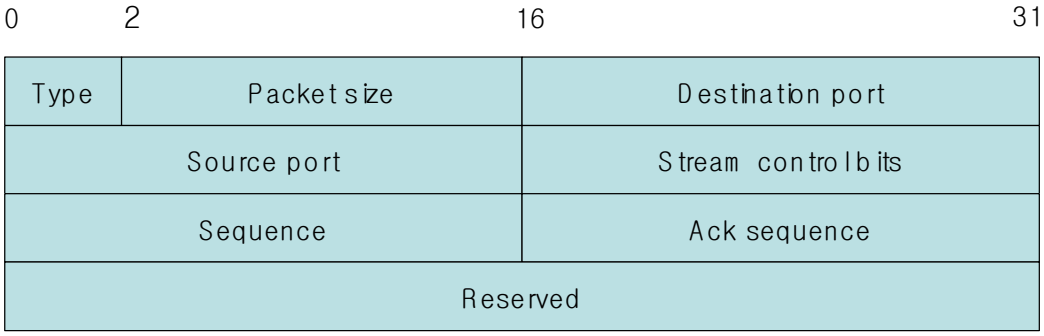
LPX(Lean Packet eXchange) Protocol Specification

* Ethernet Header



Type : 0x88ad(Ximeta Technology)

* LPX_TYPE_STREAM Header



Type(2bit) : LPX protocol의 Type을 지정한다.

```
#define LPX_TYPE_RAW            0x0     /* not implemented */
#define LPX_TYPE_DATAGRAM       0x2     /* not completed */
#define LPX_TYPE_STREAM         0x3
```

위의 3가지 type이 있지만 현재 NetDisk가 동작하는 Protocol은 LPX_TYPE_STREAM이다.

Packet size(14bit) : 위의 Header를 포함한 전체 LPX packet의 size를 나타낸다. 이 size는 byte 단위이며 Header와 Data영역을 구분하기 위해 사용된다. LPX는 ethernet상위에서 동작하는 protocol이기 때문에 ethernet의 max ethenet frame size인 1514 까지만 표현할 수 있으면 되기 때문에 14bit(2^{24} = 4096)이면 충분하다.

Destination port : packet을 전달할 상대방의 port를 지정한다.
Source port : packet을 전달하는 자신의 port를 지정한다.

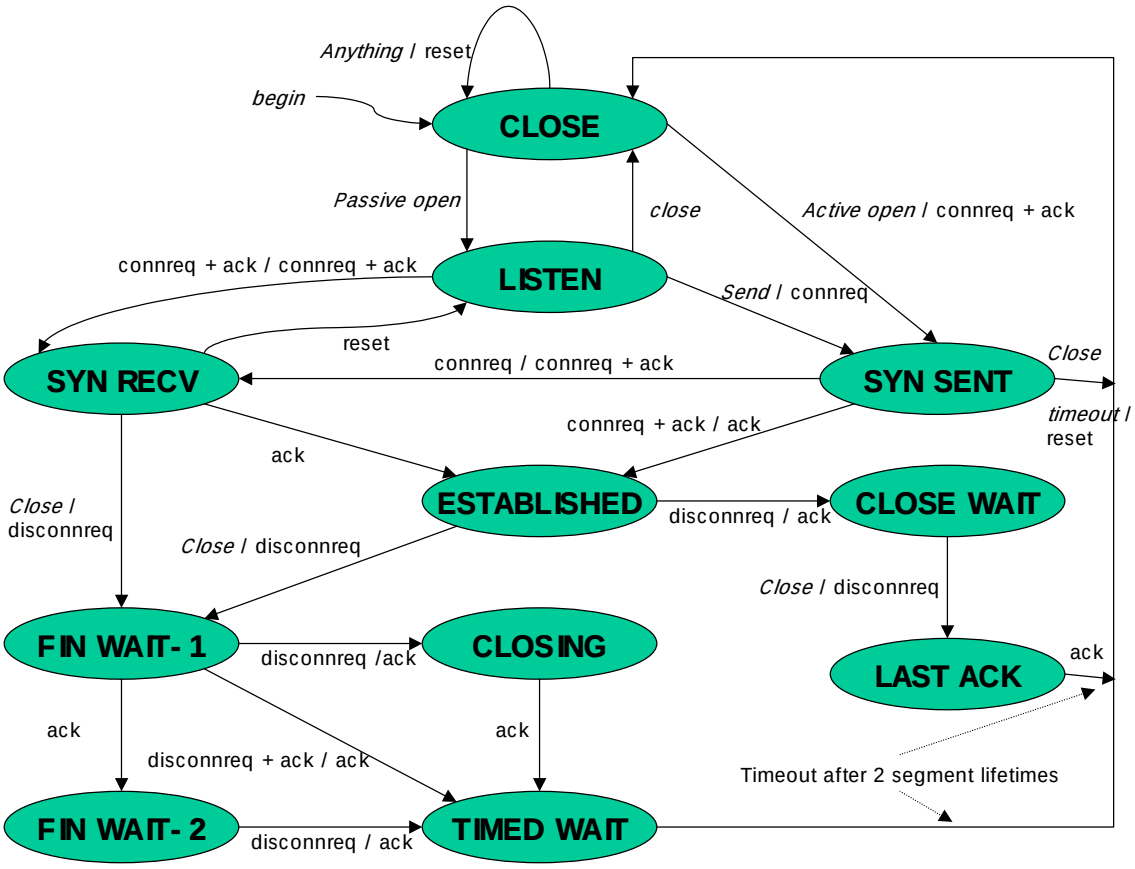
Stream control bits : LPX packet의 type을 결정한다.

CO(LSCTL_CONNREQ) = 0x0001
DA(LSCTL_DATA) = 0x0002
DC(LSCTL_DISCONNREQ) = 0x0004
AR(LSCTL_ACKREQ) = 0x0008
위의 Flag는 같이 Setting되지는 않는다. AR을 제외하고 CO, DA, DC를 포함한 packet은 보내진 후 다음 packet의 sequence가 1증가하는 영향을 미친다.

AC(LSCTL_ACK) = 0x1000
이 Flag는 CO, DA, DC, AR 과 함께 Setting될 수 있다.

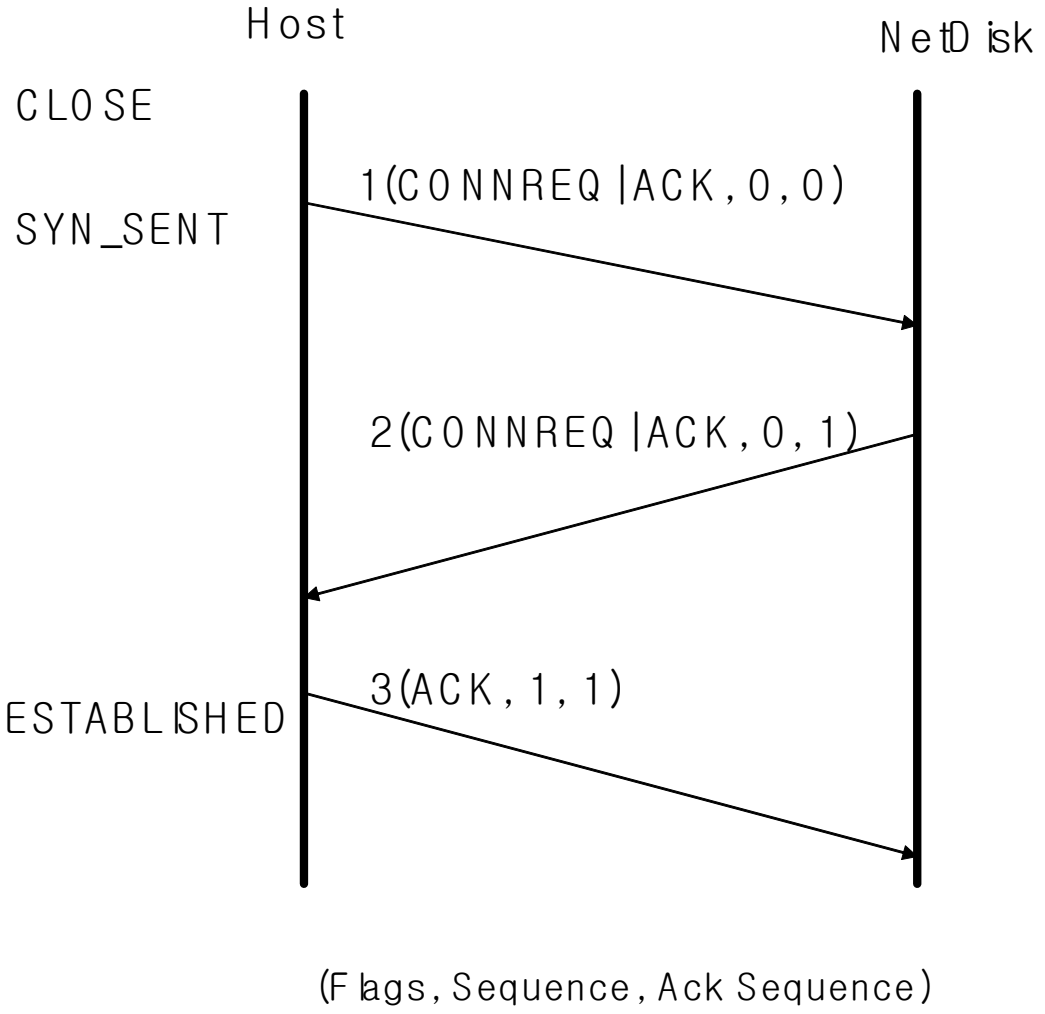
Sequence : 현재 packet의 Sequence
Ack sequence : 상대방으로 부터 받아야 하는 packet의 Sequence(상대방으로 부터 받은 packet의 sequence + 1)

* LPX Stream Protocol 상태 전이도



Host가 항상 NetDisk를 향해 Active Open을 하며, NetDisk는 Connection이 설립되기 전에는 항상 Passive Open을 위해 LISTEN 상태에 있다고 볼 수 있다. 하지만 위의 상태 전이도는 일반적인 LPX의 Host 쪽의 상태 전이도 이며, NetDisk에서는 위의 상태 전이도와는 다르게 동작할 수 있다.

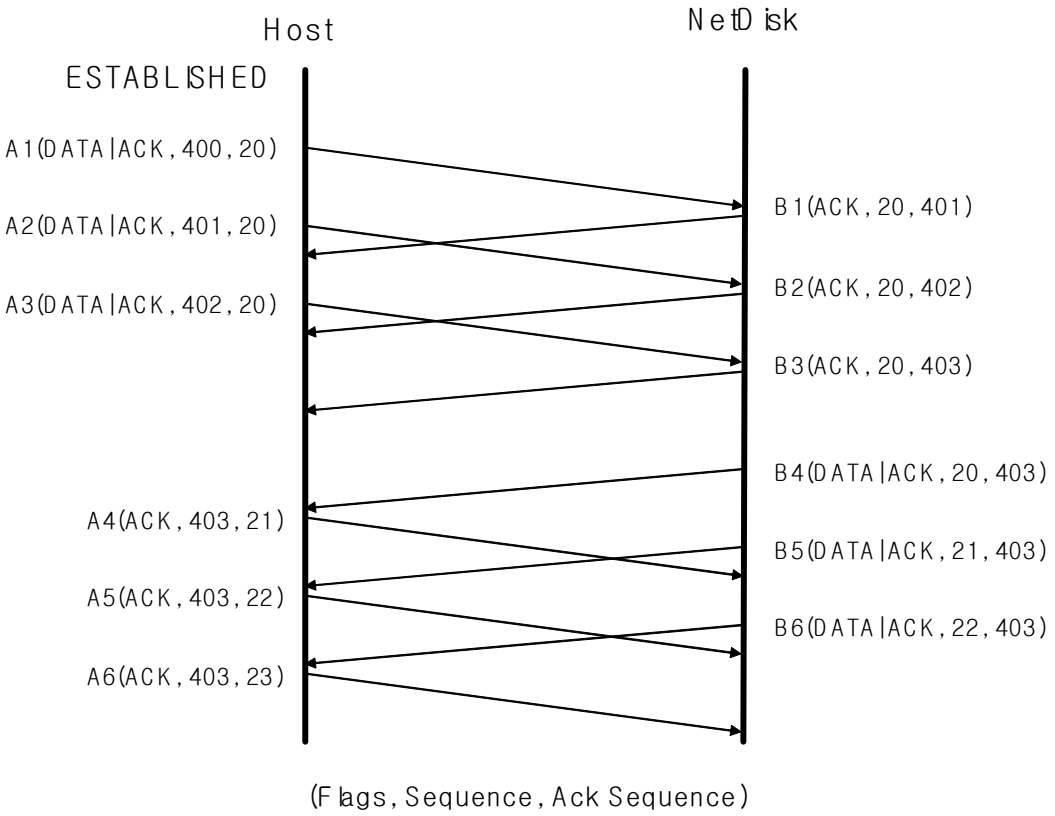
* Connect



LPX Stream Protocol에서 connection을 설립하는 방식은 TCP와 유사하게 3-way hand-shaking을 한다. 위의 그림은 Host가 NetDisk에 connection 하는 것을 나타낸다.

1(CONNREQ|ACK, 0, 0) : Host가 NetDisk에 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, 이전에 packet을 받은 적이 없으므로, ack sequence = 0 이다. Host는 SYN_SENT상태로 된다.
2(CONNREQ|ACK, 0, 1) : 1의 응답으로 NetDisk가 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, 1을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1이 된다.
3(ACK, 1, 1) : 2의 응답으로 ack를 NetDisk로 보낸다. 1을 보냈었으므로 sequence = 1 이 되고, 2를 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1 이 된다. Host, NetDisk는 ESTABLISHED 상태가 된다.

* Data Transfer



LPX Stream Protocol은 connection-oriented Protocol이므로 Connection이 설립된다음에 비로서 Data를 주고 받을 수 있다. 위의 그림은 Data Transfer가 진행되고 있는 상황을 보여주는 것으로 Host가 3개의 Data packet을 보낸 다음, NetDisk가 3개의 Data packet을 보내는 것을 보여주고 있다. Host, NetDisk 모두 sequence 가 어긋나는 data packet (즉, 연속된 sequence가 아닌 packet)은 drop한다.

A1(DATA|ACK, 400, 20) : Host가 보내는 packet의 sequence = 400 이며, NetDisk로 받아야 하는 packet의 sequence를 나타내는 ack sequence = 20임을 나타낸다.

A2(DATA|ACK, 401, 20) : A1에 의해서 sequence가 증가하였기 때문에 Host가 보내는 packet의 sequence = 401 이며, NetDisk로 부터 받은 packet이 없으므로 그대로 ack sequence = 20인 것을 볼 수 있다.

A3(DATA|ACK, 402, 20) : A2에 의해서 sequence가 증가하였기 때문에 Host가 보내는 packet의 sequence = 402 이며, NetDisk로 부터 받은 packet이 없으므로 그대로 ack sequence = 20인 것을 볼 수 있다.

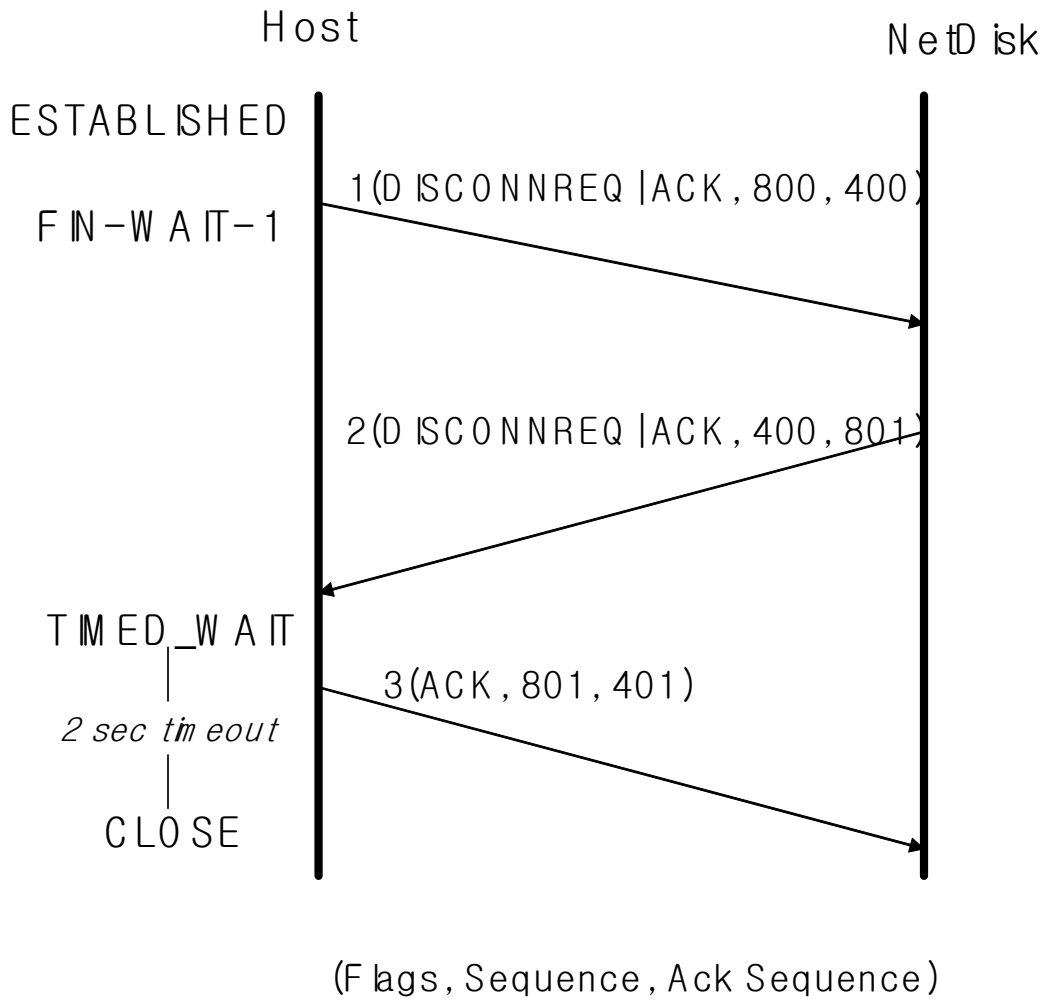
B1(ACK, 20, 401) : **A1**에 대한 ack로써, sequence = 20 이고, Host로 부터 A1을 받았으므로 Host로 받아야 하는 packet의 sequence를 나타내는 ack sequence = 401이 된다.

B2(ACK, 20, 402) : **A2**에 대한 ack로써, NetDisk에서 보낸 packet의 없으므로 그대로 sequence = 20 이고, Host로 부터 A2을 받았으므로 Host로 받아야 하는 packet의 sequence를 나타내는 ack sequence = 402이 된다.

B3(ACK, 20, 403) : **A3**에 대한 ack로써, NetDisk에서 보낸 packet의 없으므로 그대로 sequence = 20 이고, Host로 부터 A3을 받았으므로 Host로 받아야 하는 packet의 sequence를 나타내는 ack sequence = 403이 된다.

B4, B5, B6, A4, A5, A6는 위의 scenario의 역이다.

* Disconnect

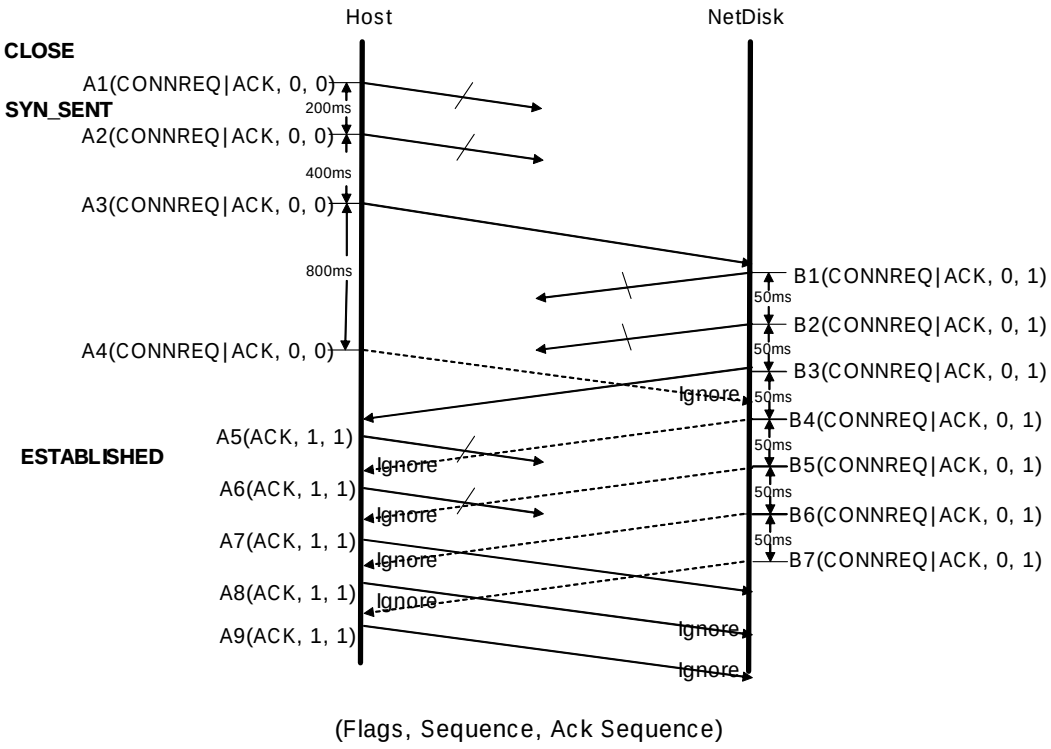


1(DISCONNREQ|ACK, 800, 400) : Host가 NetDisk에게 Disconnect를 request한다. sequence = 800이고, 다음에 받을 packet의 번호인 ack sequence = 400임을 나타낸다.

2(DISCONNREQ|ACK, 400, 801) : 1의 응답으로 NetDisk가 Disconnect request를 한다. sequence = 400 이고, 1을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 801이 된다.

3(ACK, 801, 401) : 2의 응답으로 ack를 NetDisk로 보낸다. 1을 보냈었으므로 sequence = 801이 되고, 2를 받았으므로 다음에 받을 packet의 번호인 ack sequence = 401 이 된다.

* Connect Retransmit



A1(CONNREQ|ACK, 0, 0) : Host가 NetDisk에 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, 이전에 packet을 받은 적이 없으므로, ack sequence = 0 이다. Host는 SYN_SENT상태로 된다.

A2(CONNREQ|ACK, 0, 0) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

A3(CONNREQ|ACK, 0, 0) : A1, A2에 대한 ack를 받지 못했고 2번째 retransmit이기 때

문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다. 이 packet의 경우 NetDisk에 제대로 전달되었기 때문에 NetDisk가 B1을 하게 한다.

A4(CONNREQ|ACK, 0, 0) : A1, A2, A3에 대한 ack를 아직까지 받지 못했고 3번째 retransmit이기 때문에 retransmit timer에 의해서 A3를 보내고 800ms 후에 retransmit 된다. packet의 내용은 A1, A2, A3와 같다. 이 packet 의 경우는 이미 A3가 NetDisk에 전달되었었기 때문에 무시된다.

B1(CONNREQ|ACK, 0, 1) : A3의 응답으로 NetDisk가 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, A3을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1이 된다.

B2(CONNREQ|ACK, 0, 1) : B1에 대한 ack를 받지 못했기 때문에 B1을 보내고 50ms 후에 retransmit된다. packet의 내용은 B1과 같다.

B3(CONNREQ|ACK, 0, 1) : B1, B2에 대한 ack를 받지 못했기 때문에 B2를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2와 같다. 이 packet은 Host에 전달되어 받아 들여지고 A5의 ack를 보내게 한다.

B4(CONNREQ|ACK, 0, 1) : B1, B2, B3에 대한 ack를 받지 못했기 때문에 B3를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A6의 ack를 보내게 한다.

B5(CONNREQ|ACK, 0, 1) : B1, B2, B3, B4에 대한 ack를 받지 못했기 때문에 B4를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3, B4와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A7의 ack를 보내게 한다.

B6(CONNREQ|ACK, 0, 1) : B1, B2, B3, B4, B5에 대한 ack를 받지 못했기 때문에 B5를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3, B4, B5와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A8의 ack를 보내게 한다.

B7(CONNREQ|ACK, 0, 1) : B1, B2, B3, B4, B5, B6에 대한 ack를 받지 못했기 때문에 B6를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3, B4, B5, B6와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A9의 ack를 보내게 한다.

A5(ACK, 1, 1) : B3의 응답으로 ack를 Host로 보낸다. A3를 보냈었으므로 sequence = 1이 되고, B3를 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1 이 된다.

Host, NetDisk는 ESTABLISHED 상태가 된다.

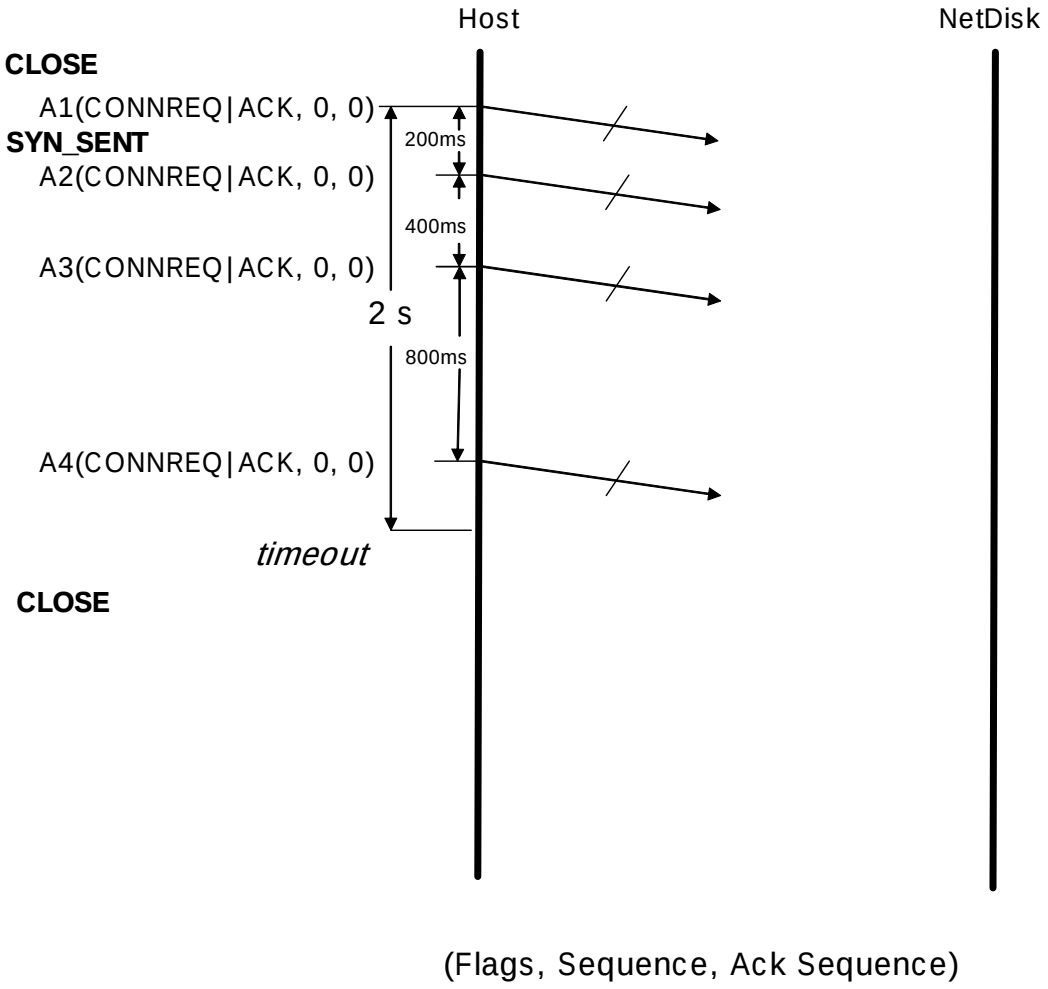
A6(ACK, 1, 1) : B4의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B4는 무시한다. packet의 내용은 A5와 같다.

A7(ACK, 1, 1) : B5의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B5는 무시한다. packet의 내용은 A5, A6와 같다.

A8(ACK, 1, 1) : B6의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B6는 무시한다. packet의 내용은 A5, A6, A7과 같다.

A9(ACK, 1, 1) : B7의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B7은 무시한다. packet의 내용은 A5, A6, A7, A8과 같다.

- Host가 CONNREQ/ACK 를 보내다가 timeout 되는 경우



A1(CONNREQ|ACK, 0, 0) : Host가 NetDisk에 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, 이전에 packet을 받은 적이 없으므로, ack sequence = 0 이다. Host는 SYN_SENT상태로 된다.

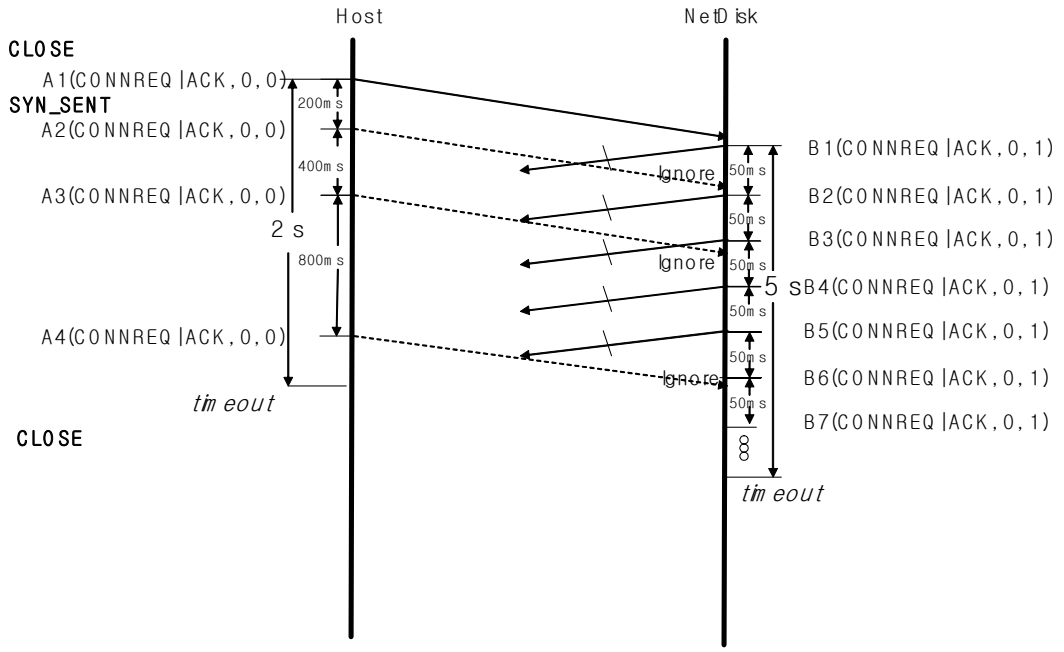
A2(CONNREQ|ACK, 0, 0) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

A3(CONNREQ|ACK, 0, 0) : A1, A2에 대한 ack를 받지 못했고 2번째 retransmit이기 때문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다.

A4(CONNREQ|ACK, 0, 0) : A1, A2, A3에 대한 ack를 아직까지 받지 못했고 3번째 retransmit이기 때문에 retransmit timer에 의해서 A3를 보내고 800ms 후에 retransmit 된다. packet의 내용은 A1, A2, A3와 같다.

위와 같이 MAX_CONNECTION_TIME(2초) 동안 request에 대한 ack가 오지 않으면 timeout되어 CLOSE 상태로 된다.

- NetDisk가 CONNREQ/ACK 를 보내다가 timeout 되는 경우



(Flags, Sequence, Ack Sequence)

A1(CONNREQ|ACK, 0, 0) : Host가 NetDisk에 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, 이전에 packet을 받은 적이 없으므로, ack sequence = 0 이다. Host는 SYN_SENT상태로 된다.

A2(CONNREQ|ACK, 0, 0) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

A3(CONNREQ|ACK, 0, 0) : A1, A2에 대한 ack를 받지 못했고 2번째 retransmit이기 때문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다.

A4(CONNREQ|ACK, 0, 0) : A1, A2, A3에 대한 ack를 아직까지 받지 못했고 3번째 retransmit이기 때문에 retransmit timer에 의해서 A3를 보내고 800ms 후에 retransmit 된다. packet의 내용은 A1, A2, A3와 같다. B1(CONNREQ|ACK, 0, 1) : A3의 응답으로 NetDisk가 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, A3을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1이 된다.

B1(CONNREQ|ACK, 0, 1) : A1의 응답으로 NetDisk가 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, A1을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1이 된다.

B2(CONNREQ|ACK, 0, 1) : B1에 대한 ack를 받지 못했기 때문에 B1을 보내고 50ms 후에 retransmit된다. packet의 내용은 B1과 같다.

B3(CONNREQ|ACK, 0, 1) : B1, B2에 대한 ack를 받지 못했기 때문에 B2를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2와 같다.

B4(CONNREQ|ACK, 0, 1) : B1, B2, B3에 대한 ack를 받지 못했기 때문에 B3를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3와 같다.

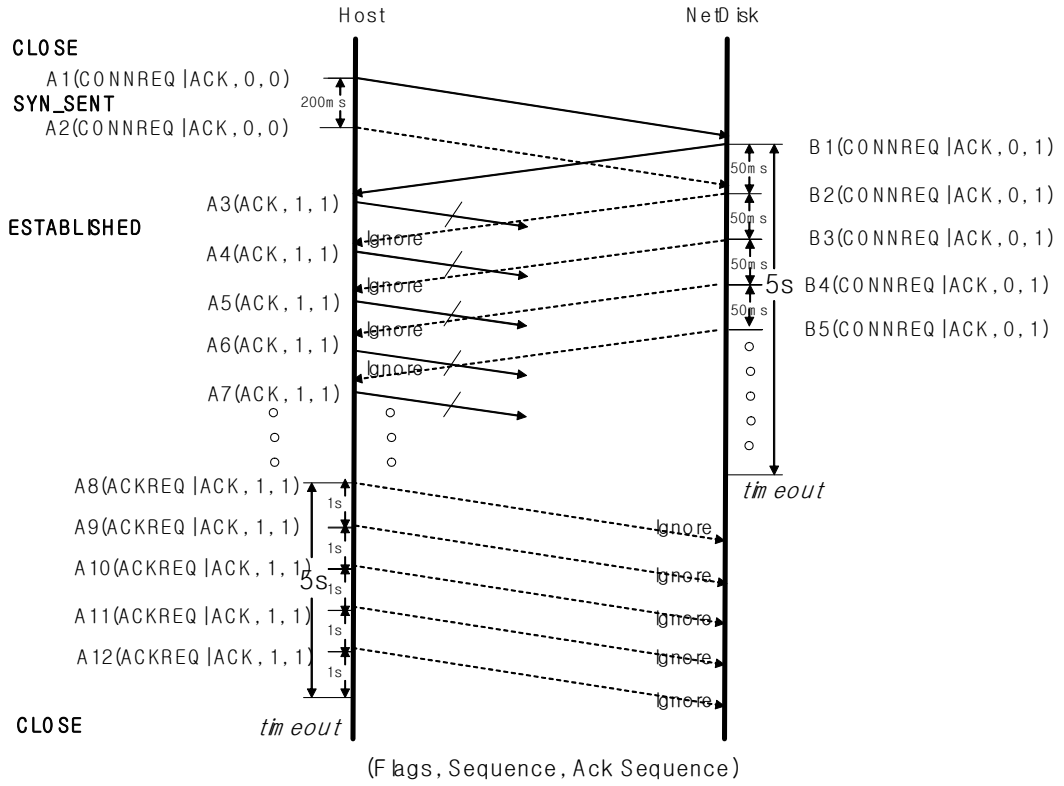
.

.

.

계속 보내다가 NetDisk쪽의 MAX_CONNECTION_TIME(5초) 가 지나면 NetDisk 쪽에서도 connection을 cleanup한다. Host쪽의 MAX_CONNECTINO_TIME은 2초 이므로 이미 clear된 상태이다.

- NetDisk의 CONNREQ|ACK에 대한 ACK가 사라져서 timeout 되는 경우



A1(CONNREQ|ACK, 0, 0) : Host가 NetDisk에 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, 이전에 packet을 받은 적이 없으므로, ack sequence = 0 이다. Host는 SYN_SENT상태로 된다.

A2(CONNREQ|ACK, 0, 0) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다. A1이 받아들여 졌기 때문에 A2는 무시된다.

B1(CONNREQ|ACK, 0, 1) : A1의 응답으로 NetDisk가 connection request를 한다. 처음 보내는 packet이므로 sequence = 0 이고, A1을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1이 된다.

B2(CONNREQ|ACK, 0, 1) : B1에 대한 ack를 받지 못했기 때문에 B1을 보내고 50ms 후에 retransmit된다. packet의 내용은 B1과 같다.

B3(CONNREQ|ACK, 0, 1) : B1, B2에 대한 ack를 받지 못했기 때문에 B2를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2와 같다.

B4(CONNREQ|ACK, 0, 1) : B1, B2, B3에 대한 ack를 받지 못했기 때문에 B3를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3와 같다.

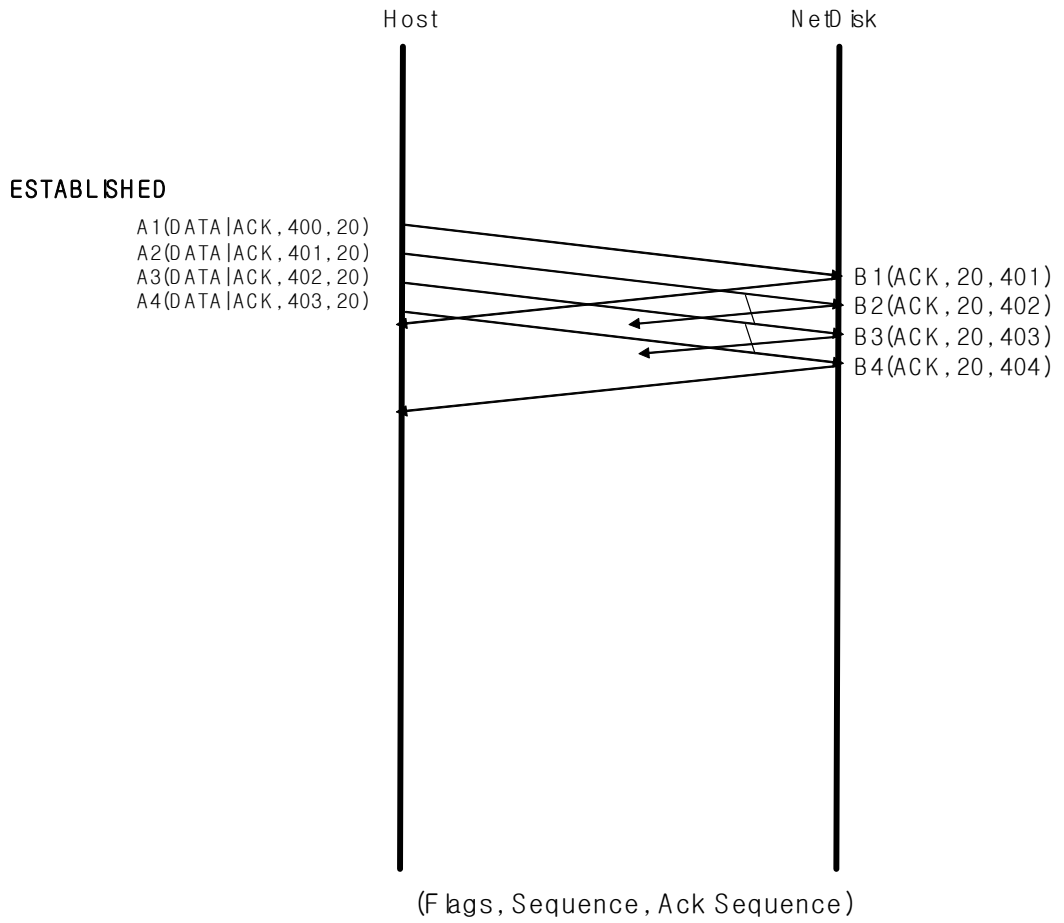
.
.
.
계속 보내다가 NetDisk쪽의 MAX_CONNECTION_TIME(5초) 가 지나면 NetDisk 쪽에서도 connection을 cleanup한다.

A3(ACK, 1, 1) : B1의 응답으로 ack를 Host로 보낸다. A1을 보냈었으므로 sequence = 1이 되고, B1를 받았으므로 다음에 받을 packet의 번호인 ack sequence = 1 이 된다.

A4(ACK, 1, 1) : B2는 B1을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기위해 NetDisk에게 ack를 보낸다.
A5(ACK, 1, 1) : B3는 B1을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기위해 NetDisk에게 ack를 보낸다.
A6(ACK, 1, 1) : B4는 B1을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기위해 NetDisk에게 ack를 보낸다.
A7(ACK, 1, 1) : B5는 B1을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기위해 NetDisk에게 ack를 보낸다.

A8, A9, A10, A11, A12 : ESTABLISHED 상태이므로 마지막 packet이 도착한 시점 부터 ALIVE_INTERVAL(1초) 간격으로 ACKREQ|ACK를 보내고 ALIVE_COUNT(5)가 되면 timeout된다.

* Data Transfer and ack



위의 그림은 sequence가 400인 것부터 403 인것 까지 연속으로 보내고 그것에 대한 ack를 받는 것을 나타낸 것이다. B2, B3가 사라져도 B4를 받으면, sequence 가 404 이전의 것은 모두 받은 것으로 간주하고 더이상 retransmit하지 않는다.

A1(DATA|ACK, 400, 20) : Host가 보내는 packet의 sequence = 400 이며, NetDisk로부터 받아야 하는 packet의 sequence를 나타내는 ack sequence = 20임을 나타낸다.

A2(DATA|ACK, 401, 20) : A1을 보냈으므로 sequence가 증가 되어 401이 된다.
A3(DATA|ACK, 402, 20) : A2를 보냈으므로 sequence가 증가 되어 402이 된다.
A4(DATA|ACK, 403, 20) : A3를 보냈으므로 sequence가 증가 되어 403이 된다.

B1(ACK, 20, 401) : A1에 대한 ack로, A1을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 401이 된다.

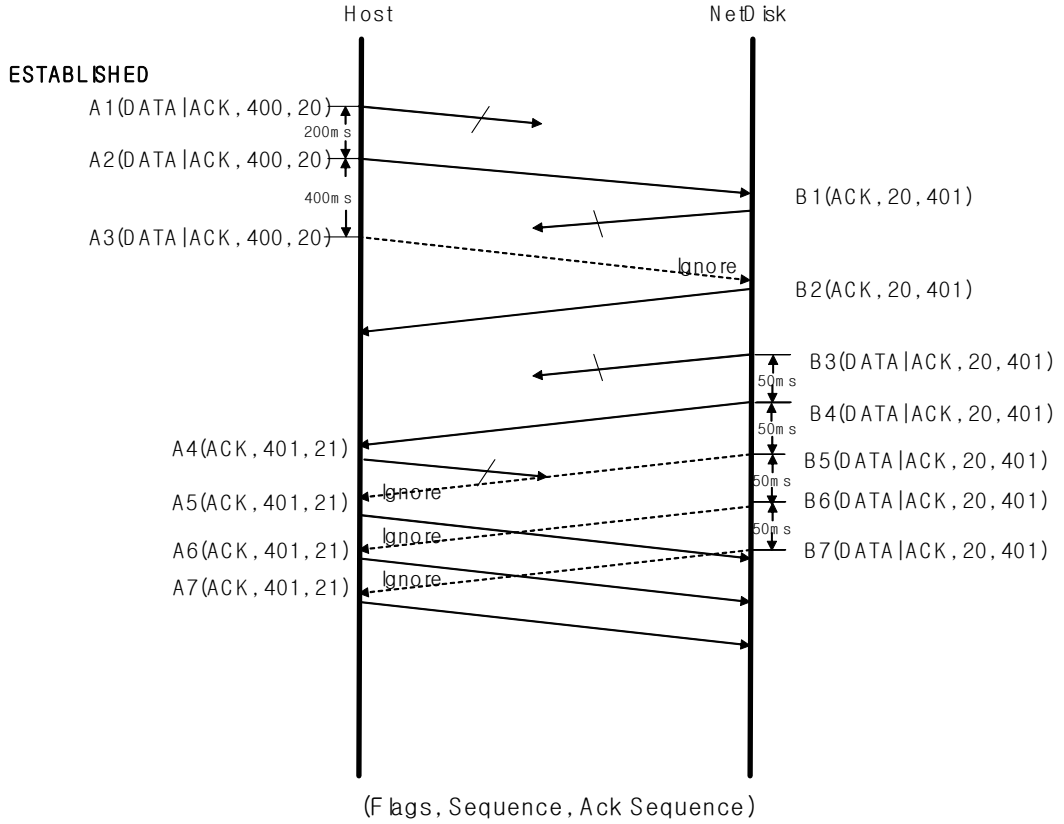
B2(ACK, 20, 402) : A2에 대한 ack로, A2을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 402이 된다. 이 packet은 drop된다.

B3(ACK, 20, 403) : A3에 대한 ack로, A3을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 403이 된다. 이 packet은 drop된다.

B4(ACK, 20, 404) : A4에 대한 ack로, A4을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 404이 된다. 이 packet은 Host에 전해지게 되며, 이로 인하여 A2, A3도 NetDisk가 받았음을 Host에게 알리게 된다.

NetDisk에서 Host로 Data를 보낼 때에도 마찬가지로 동작한다. ACK가 보내다가 사라지더라도 ACK는 retransmit되지 않으며, 나중에 가는 ack가 도착한다면 그 이전에 도착하지 않은 ack를 대신할 수 있다.

* Data Transfer and Retransmit



위의 그림은 Host가 sequence = 400의 packet을 보내고, NetDisk가 sequence = 20의 packet을 보냈을 때 일어나는 상황을 묘사한 것이다.

A1(DATA|ACK, 400, 20) : Host가 보내는 packet의 sequence = 400 이며, NetDisk로부터 받아야 하는 packet의 sequence를 나타내는 ack sequence = 20임을 나타낸다.

A2(DATA|ACK, 400, 20) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다. A1이 사라지고 A2가 전달되었기 이 packet에 의해서 B1이 NetDisk로 부터 보내지게 한다.

A3(DATA|ACK, 400, 20) : A1, A2에 대한 ack를 아직 못했고 2번째 retransmit이기 때문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다. A2가 받아 들여 졌기 때문에 A3는 무시되고, B2가 NetDisk로 부터 보내지게 한다.

B1(ACK, 20, 401) : A2의 packet을 받아들이고, A2의 응답으로 ack를 한다. A2를 받아들였기 때문에 Host로 부터 받아야 하는 packet의 sequence를 나타내는 ack sequence = 401이 된다.

B2(ACK, 20, 401) : A2를 받아들였기 때문에, A3는 무시되고, 받은 packet이라는 것을 알리기위해 Host에게 B1과 같은 ack를 보낸다.

B3(DATA|ACK, 20, 401) : NetDisk가 보내는 packet의 sequence = 20 이며, NetDisk로 부터 받아야 하는 packet의 sequence를 나타내는 ack sequence = 401임을 나타낸다. 이 packet은 Host로 전달되지 않는다.

B4(DATA|ACK, 20, 401) : B3에 대한 ack를 받지 못했기 때문에 B3을 보내고 50ms 후에 retransmit된다. packet의 내용은 B3과 같다. 이 packet이 전달되어 A4를 보내게 한다.

B5(DATA|ACK, 20, 401) : B3, B4에 대한 ack를 받지 못했기 때문에 B4를 보내고 50ms 후에 retransmit된다. packet의 내용은 B3, B4와 같다. 이 packet이 전달되어 A5를 보내게 한다.

B6(DATA|ACK, 20, 401) : B3, B4, B5에 대한 ack를 받지 못했기 때문에 B5를 보내고 50ms 후에 retransmit된다. packet의 내용은 B3, B4, B5와 같다. 이 packet이 전달되어 A6를 보내게 한다.

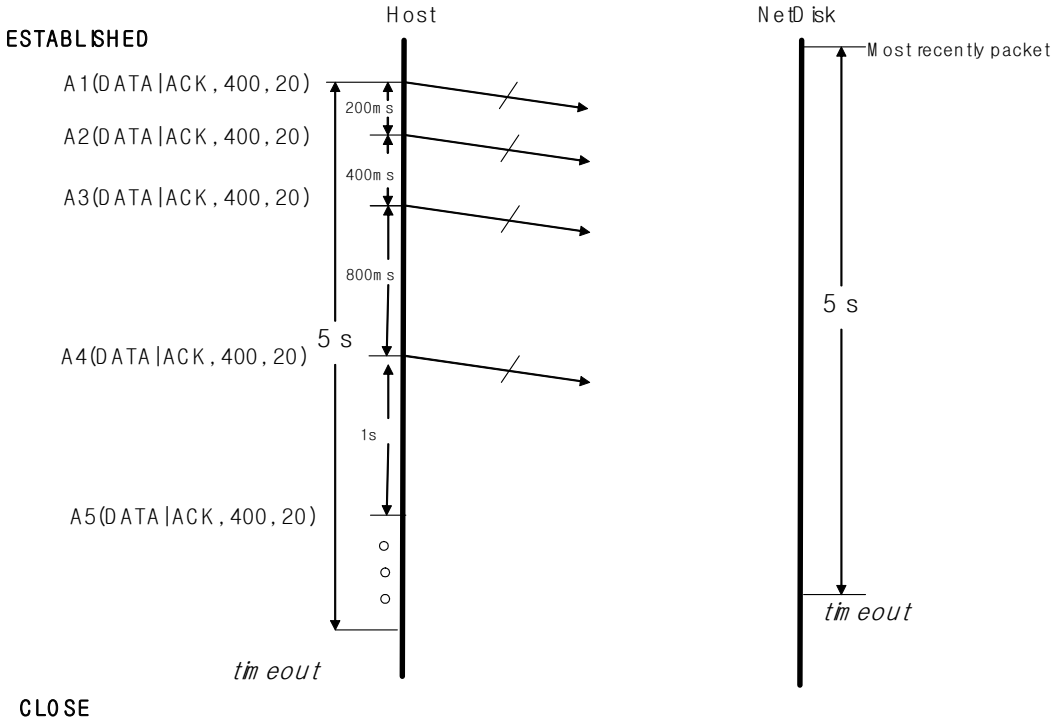
B7(DATA|ACK, 20, 401) : B3, B4, B5, B6에 대한 ack를 받지 못했기 때문에 B6를 보내고 50ms 후에 retransmit된다. packet의 내용은 B3, B4, B5, B6와 같다. 이 packet이 전달되어 A7를 보내게 한다.

A4(ACK, 401, 21) : B4의 packet을 받아들이고, B4의 응답으로 ack를 한다. B4를 받아들였기 때문에 NetDisk로 부터 받아야 하는 packet의 sequence를 나타내는 ack sequence = 401이 된다.

A5(ACK, 401, 21) : B5는 B4을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기위해 NetDisk에게 ack를 보낸다.

A6(ACK, 401, 21) : B6는 B4을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기 위해 NetDisk에게 ack를 보낸다.
A7(ACK, 401, 21) : B7는 B4을 받았기 때문에 무시하고, 받은 packet이라는 것을 알리기위해 NetDisk에게 ack를 보낸다.

- Host가 NetDisk로 Data를 보내다가 timeout되는 경우

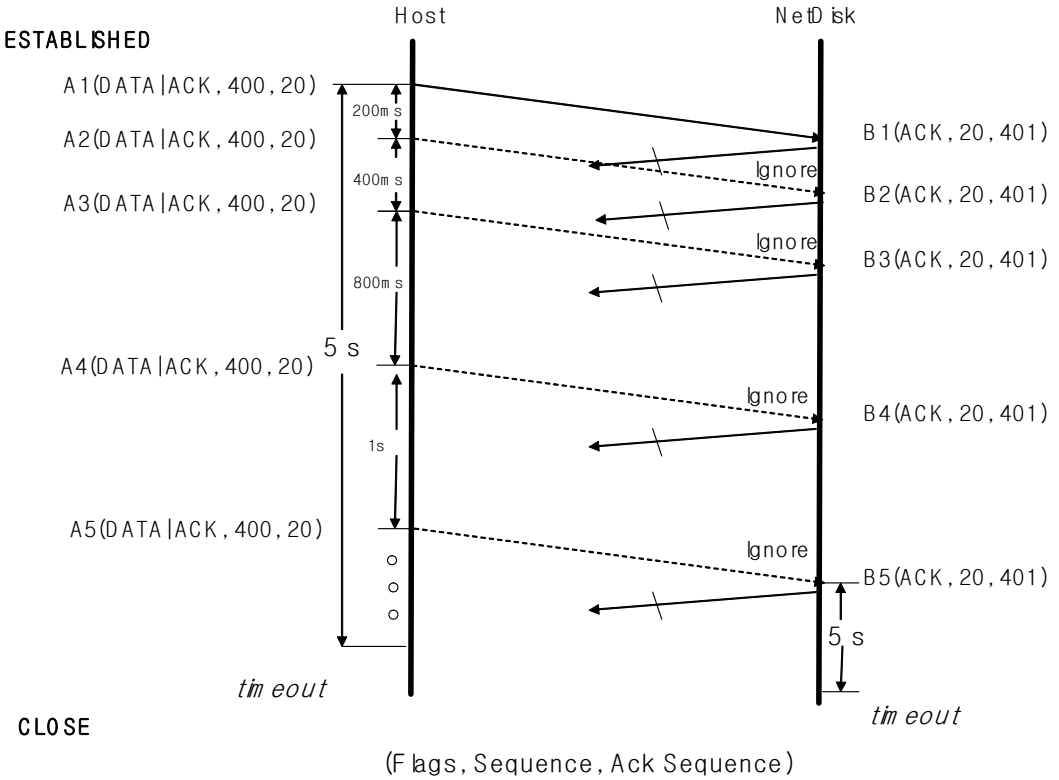


(Flags, Sequence, Ack Sequence)

Host쪽에서는 sequence가 400인 data packet에 대한 응답의 없으므로 retransmit timer가 작동하여 retransmit한다. retransmit time은 200ms, 400ms, 800ms 로 증가하다가 1s를 넘지 않으며, 이 합이 5s가 되면 끊겼다고 생각하고 connection을 CLOSE한다.

NetDisk쪽에서는 최근에 packet을 받은 시간으로 부터 그 connection으로 5초 동안 아무런 packet이 도착하지 않으면 그 connection을 cleanup한다.

- NetDisk가 Host로 ack를 보내다가 timeout되는 경우



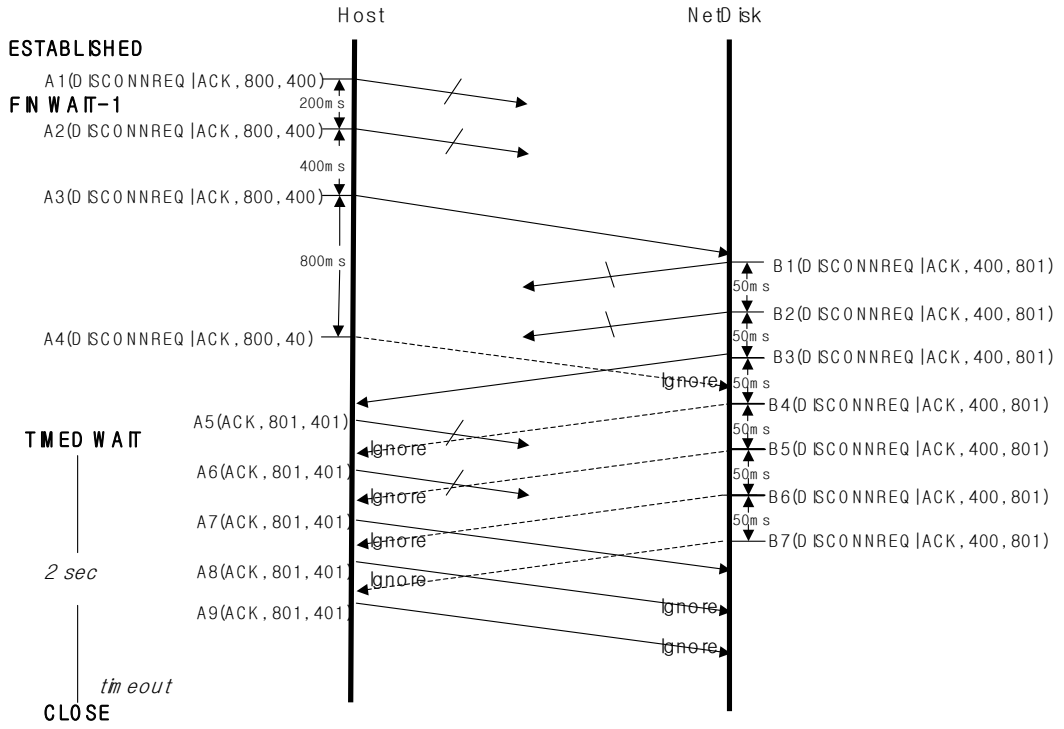
위의 그림은 같이 Host의 sequence = 400인 packet이 5초 동안 retransmit 되는 상황에서 NetDisk의 ack가 계속해서 사라지는 경우를 나타낸다. NetDisk쪽에서는 retransmit된 sequence = 400 인 packet이 올 때마다 Host로 ack를 한다.

Host쪽에서는 5초 동안 ack가 오지 않았기 때문에 connection을 CLOSE한다.
NetDisk쪽에서는 A5가 도착한 후 그 connection으로 5초 동안 아무런 packet이 도착하지 않을 것이기 때문에 connection이 cleanup 된다.

- Host가 NetDisk로 Data를 보내다가 timeout되는 경우
- NetDisk가 Host로 Ack를 보내다가 timeout되는 경우

위의 두 경우는 앞에서 보여 주었던 2가지 경우의 역이다. 단지 retransmit timer가 timeout되는 시간이 Host 쪽은 200ms, 400ms, 800ms, 1s, 1s ... 인데 반해서 NetDisk쪽은 50ms으로 일정하다는 것이다.

* Disconnect Retransmit



A1(DISCONNREQ|ACK, 800, 400) : Host가 NetDisk에 disconnect request를 한다. sequence = 800이고, 다음에 받을 packet의 번호인 ack sequence = 400임을 나타낸다. Host는 FIN WAIT-1상태로 된다.

A2(DISCONNREQ|ACK, 800, 400) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

A3(DISCONNREQ|ACK, 800, 400) : A1, A2에 대한 ack를 받지 못했고 2번째 retransmit이기 때문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다. 이 packet의 경우 NetDisk에 제대로 전달되었기 때문에 NetDisk가 B1을 하게 한다.

A4(DISCONNREQ|ACK, 800, 400) : A1, A2, A3에 대한 ack를 아직까지 받지 못했고 3번째 retransmit이기 때문에 retransmit timer에 의해서 A3를 보내고 800ms 후에 retransmit 된다. packet의 내용은 A1, A2, A3와 같다. 이 packet 의 경우는 이미 A3가 NetDisk에 전달되었었기 때문에 무시된다.

B1(DISCONNREQ|ACK, 400, 801) : A3의 응답으로 NetDisk가 disconnection request를 한다. A3 을 받았으므로 다음에 받을 packet의 번호인 ack sequence = 801이 된다.

B2(DISCONNREQ|ACK, 400, 801) : B1에 대한 ack를 받지 못했기 때문에 B1을 보내고

50ms 후에 retransmit된다. packet의 내용은 B1과 같다.

B3(DISCONNREQ|ACK, 400, 801) : B1, B2에 대한 ack를 받지 못했기 때문에 B2를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2와 같다. 이 packet은 Host에 전달되어 받아 들여지고 A5의 ack를 보내게 한다.

B4(DISCONNREQ|ACK, 400, 801) : B1, B2, B3에 대한 ack를 받지 못했기 때문에 B3를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A6의 ack를 보내게 한다.

B5(DISCONNREQ|ACK, 400, 801) : B1, B2, B3, B4에 대한 ack를 받지 못했기 때문에 B4를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3, B4와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A7의 ack를 보내게 한다.

B6(DISCONNREQ|ACK, 400, 801) : B1, B2, B3, B4, B5에 대한 ack를 받지 못했기 때문에 B5를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3, B4, B5와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A8의 ack를 보내게 한다.

B7(DISCONNREQ|ACK, 400, 801) : B1, B2, B3, B4, B5, B6에 대한 ack를 받지 못했기 때문에 B6를 보내고 50ms 후에 retransmit된다. packet의 내용은 B1, B2, B3, B4, B5, B6와 같다. 이 packet은 Host에 전달되어 이미 B3에 의해서 받아들여 졌기 때문에 무시되고 A9의 ack를 보내게 한다.

A5(ACK, 801, 401) : B3의 응답으로 ack를 Host로 보낸다. A3를 보냈었으므로 sequence = 401이 되고, B3를 받았으므로 다음에 받을 packet의 번호인 ack sequence = 801 이 된다. Host, NetDisk는 TIMED WAIT 상태가 된다.

A6(ACK, 801, 401) : B4의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B4는 무시한다. packet의 내용은 A5와 같다.

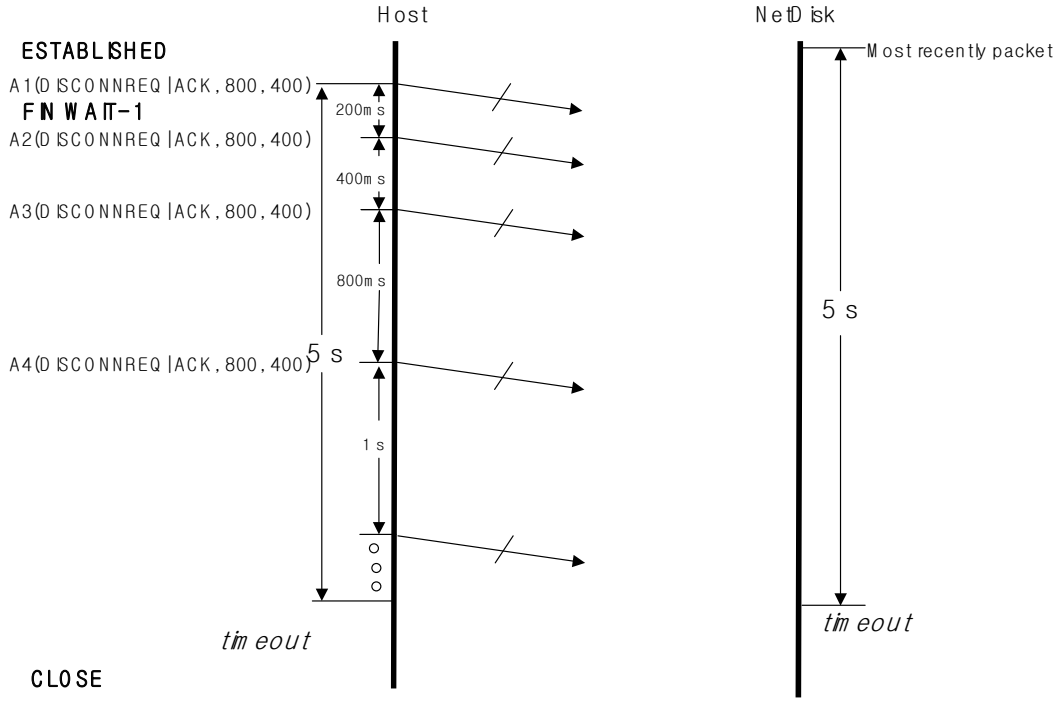
A7(ACK, 801, 401) : B5의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B5는 무시한다. packet의 내용은 A5, A6와 같다. 이 packet이 NetDisk에 전달되어 connection을 cleanup하게 한다.

A8(ACK, 801, 401) : B6의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B6는 무시한다. packet의 내용은 A5, A6, A7과 같다. 이 packet은 이미 A7에 의해서 connection이 cleanup되었으므로 무시된다.

A9(ACK, 801, 401) : B7의 응답으로 ack를 Host로 보낸다. 이미 B3를 받았으므로 B7은 무시한다. packet의 내용은 A5, A6, A7, A8과 같다. 이 packet은 이미 A7에 의해서 connection이 cleanup되었으므로 무시된다.

Host 쪽은 A5의 ack를 보내고 DISTROY_TIMEOUT(2초) 후에는 connection이 CLOSE된다.

- Host가 DISCONNREQ/ACK 를 보내다가 timeout 되는 경우



(Flags, Sequence, Ack Sequence)

A1(DISCONNREQ|ACK, 800, 400) : Host가 NetDisk에 connection request를 한다. sequence = 800이고, 다음에 받을 packet의 번호인 ack sequence = 400임을 나타낸다. Host는 FIN WAIT-1상태로 된다.

A2(DISCONNREQ|ACK, 800, 400) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

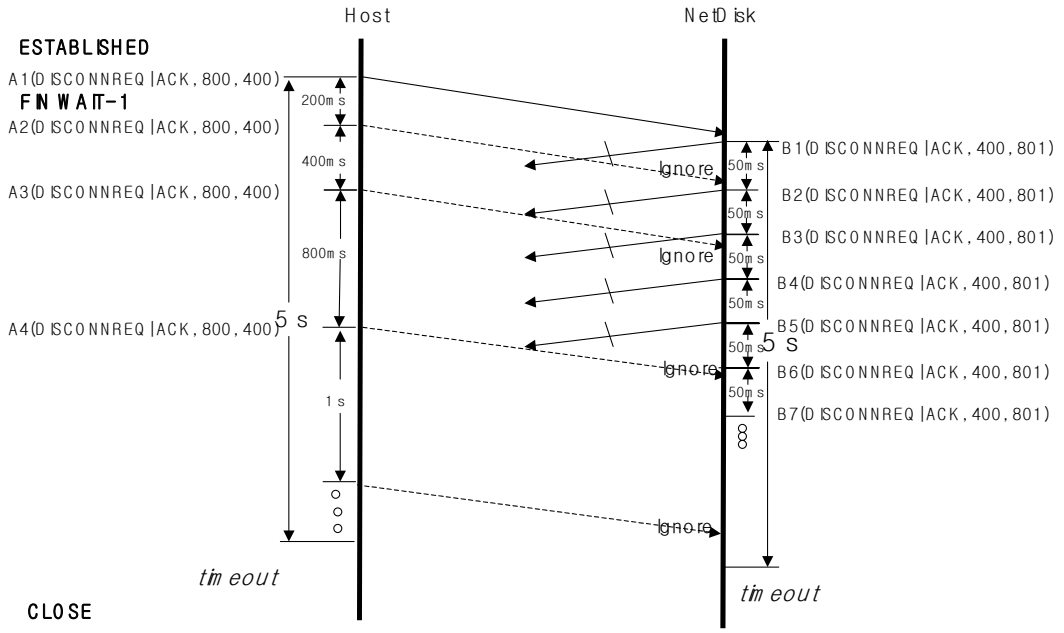
A3(DISCONNREQ|ACK, 800, 400) : A1, A2에 대한 ack를 받지 못했고 2번째 retransmit이기 때문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다.

A4(DISCONNREQ|ACK, 800, 400) : A1, A2, A3에 대한 ack를 아직까지 받지 못했고 3

번째 retransmit이기 때문에 retransmit timer에 의해서 A3를 보내고 800ms 후에 retransmit 된다. packet의 내용은 A1, A2, A3와 같다.

Host쪽의 retransmit time은 1초를 넘지 않으며, Host는 A1을 보내고 MAX_RETRANSMIT_IME(5초)까지 ack가 오지 않으면 connection을 CLOSE한다. NetDisk쪽에서는 connection인 설립되고, 그 connection으로 MAX_CONNECT_TIME(5초) 동안 아무런 packet도 오지 않으면 connection을 cleanup한다.

- NetDisk가 DISCONNREQ/ACK 를 보내다가 timeout 되는 경우



(Flags, Sequence, Ack Sequence)

A1(DISCONNREQ|ACK, 800, 400) : Host가 NetDisk에 connection request를 한다. sequence = 800이고, 다음에 받을 packet의 번호인 ack sequence = 400임을 나타낸다. Host는 FIN WAIT-1상태로 된다.

A2(DISCONNREQ|ACK, 800, 400) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

A3(DISCONNREQ|ACK, 800, 400) : A1, A2에 대한 ack를 받지 못했고 2번째 retransmit이기 때문에 retransmit timer에 의해서 A2를 보내고 400ms 후에 retransmit 된다. packet의 내용은 A1, A2와 같다.

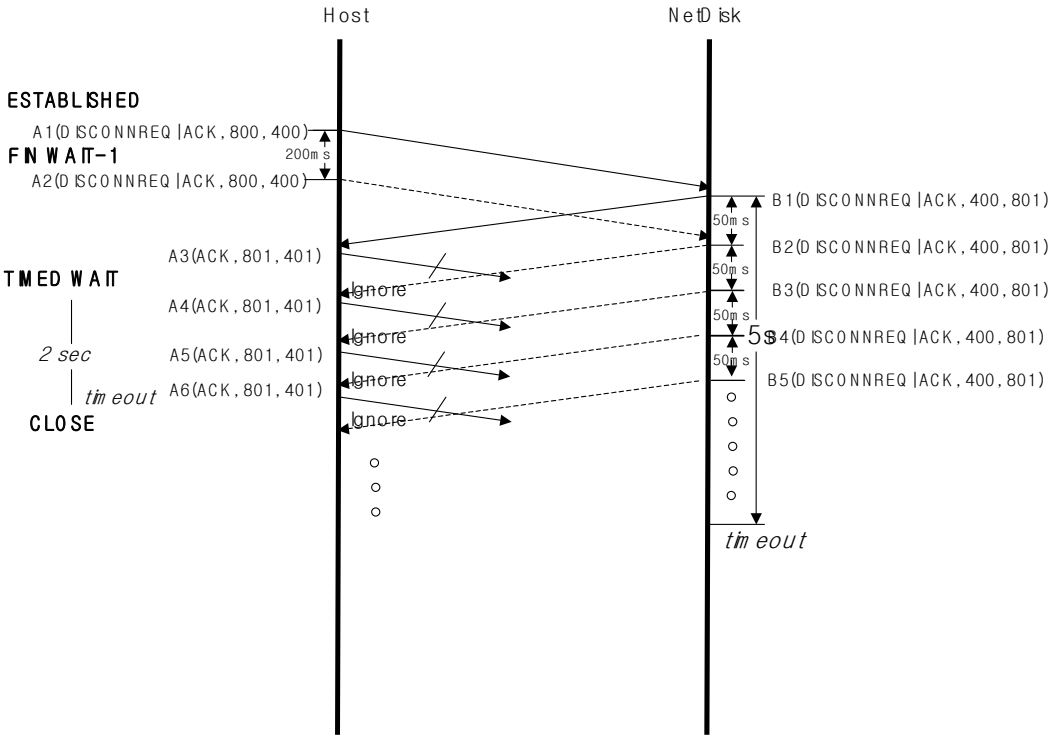
A4(DISCONNREQ|ACK, 800, 400) : A1, A2, A3에 대한 ack를 아직까지 받지 못했고 3번째 retransmit이기 때문에 retransmit timer에 의해서 A3를 보내고 800ms 후에

retransmit 된다. packet의 내용은 A1, A2, A3와 같다.

Host쪽의 retransmit time은 1초를 넘지 않으며, Host는 A1을 보내고 MAX_RETRANSMIT_IME(5초)까지 ack가 오지 않으면 connection을 CLOSE한다.

NetDisk쪽에서는 connection인 설립되고, A1의 disconnect request를 받아서 B1을 보냈지만, B1에 대한 ack를 받지 못했기 때문에, MAX_CONNECT_TIME(5초) 동안 50ms 단위로 retransmit 하다가 ack를 받지 못하면 그 connection을 cleanup한다.

- NetDisk의 DISCONNREQ|ACK에 대한 ACK가 사라져서 timeout 되는 경우



A1(DISCONNREQ|ACK, 800, 400) : Host가 NetDisk에 connection request를 한다. sequence = 800이고, 다음에 받을 packet의 번호인 ack sequence = 400임을 나타낸다. Host는 FIN WAIT-1상태로 된다.

A2(DISCONNREQ|ACK, 800, 400) : A1에 대한 ack를 받지 못했기 때문에 retransmit timer에 의해서 A1을 보내고 200ms 후에 retransmit 된다. packet의 내용은 A1과 같다.

B1, B2, B3, B4, B5 ... 은 (DISCONNECREQ|ACK, 400, 801)에 대한 ack를 받지 못했기 때문에 50ms 단위로 retransmit한다.

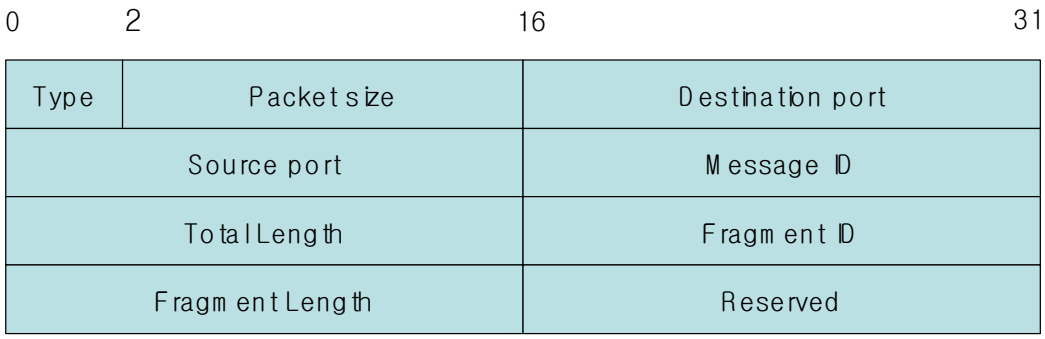
A3(ACK, 801, 401) : B1에 대한 ack로 보내고, TIMED_WAIT 상태에 들어간다.

DISTROY_TIMEOUT(2초) 동안에 NetDisk로 부터 오는 retransmit packet에 대해서는 ack를 해주며, 이후 timeout 후에는 응답하지 않는다.

위에서 여러 가지 상황에 대해서 살펴보았다. 하지만, 위에서 살펴 본 경우 말고도 다양한 상황이 벌어 질 수 있다. 그래서 다음은 기본 적인 원리와 timer 값에 대해서 알아본다.

1. 연속적으로 온 sequence만 받아들인다. packet의 type이 CONREQ, DATA, DISCONNREQ이고, sequence number가 연속되지 않은 것은 무시한다.
2. 이미 받아들인 sequence를 가진 packet을 받았을 경우는 이에 대해 ack를 하고, 무시한다.
3. CONREQ, DATA, DISCONNREQ는 보낼 때 sequence number가 증가하며, 이에 대한 ACK를 받지 못했을 경우에는 retransmit한다.
4. ACKREQ와 ACK는 sequence number를 증가하지 않으며, ACKREQ는 ESTABLISHED 상태에서 retransmit할 packet이 없는 경우에만 하며, 5초 동안 1초 단위로 한다.
5. ack sequence가 항상 유효하도록, ACK flag를 setting하므로 piggy-bag 한다.
6. Host 쪽에서의 retransmit은 200ms, 400ms, 800ms, 1s, 1s... 해서 5초 동안하며 (CONNREQ의 경우는 2초 동안), NetDisk 쪽에서의 retransmit은 50ms 단위로 5초 동안 한다.

* LPX_TYPE_DATAGRAM Header



Type(2bit) : LPX protocol의 Type을 지정한다.

```
#define LPX_TYPE_RAW            0x0     /* not implemented */
#define LPX_TYPE_DATAGRAM       0x2     /* not completed */
#define LPX_TYPE_STREAM         0x3
```

위의 3가지 type이 있지만 현재 NetDisk가 동작하는 Protocol은 LPX_TYPE_STREAM이다.

Packet size(14bit) : 위의 Header를 포함한 전체 LPX packet의 size를 나타낸다. 이 size는 byte 단위이며 Header와 Data영역을 구분하기 위해 사용된다. LPX는 ethernet상위에서 동작하는 protocol이기 때문에 ethernet의 max ethenet frame size인 1514 까지만 표현할 수 있으면 되기 때문에 14bit(2^{24} = 4096)이면 충분하다.

Destination port : packet을 전달할 상대방의 port를 지정한다.
Source port : packet을 전달하는 자신의 port를 지정한다.

Message ID = Message 고유번호
Total Length = Datagram 전체 크기
Fragment ID = Message가 여러 조각으로 나누어졌을 때 순차적으로 붙는 번호
Fragment Length = 조각난 패킷의 크기

이 Protocol은 현재 NetDisk가 다음과 같은 목적을 위해서 Broadcast한다.

첫째, 자신의 Network의 link change가 있을 때, switch에게 이 link change에 대해서 알리기 위해.
둘째, Host에게 자신의 존재를 알리기 위해서.

Broadcast packet의 구성

-Header
Type = LPX_TYPE_DATAGRAM
Packet size = 16(Header size) + 2
Destination Port = 10002
Source Port = 10001
Message ID = 0
Total Length = 2
Fragment ID = 0
Fragment Length = 0

-Data

Type = 0x0
Version = 0x0