Network Working Group
Internet Draft
Expiration Date: May 1998

Noritoshi Demizu
SonyCSL, Inc.

Hidetaka Izumiyama
Japan Satellite Systems, Inc.

November 1997

Dynamic Tunnel Configuration Protocol

<draft-demizu-udlr-dtcp-00.txt>

Status of this memo

Abstract

As the Internet grows, the importance of Uni-Directional Links such as satellite links has been increasing. However, most of existing routing protocols assume that datalinks are bi-directional. To incorporate UDLs into the Internet, an IP tunneling approach has been proposed, where tunnels are set up as a virtual back-channel of a UDL to carry routing information between Feeders and Receivers.

Dynamic Tunnel Configuration Protocol (DTCP) supports to setup tunnels dynamically between Feeders and Receivers in such environments.

1. Introduction

To use Uni-Directional Link (UDL) such as satellite links as an IP route, an IP tunneling approach[1] has been prpoposed. In this approach, tunnels are set up as a virtual back-channel of a UDL to carry routing information between Feeders and Receivers. Some of such UDLs might have only one Feeder, others might have more than one.

Dynamic Tunnel Configuration Protocol (DTCP) supports to setup tunnels dynamically between Feeders and Receivers in that environments. The main roles of DTCP are:

- to notify the availability of the Feeder to Receivers.
- to exchange node information that is necessary to transfer packets through a UDL (e.g. MAC addresses of Receivers to forward unicast packets)
- to establish and to release tunnels dynamically, including dynamic IP address allocation for Receivers if necessary.

Here is the outline of the procedure to start using a UDL as an IP route with DTCP:

1. Feeders send a "DTCP Announcement" message to a UDL periodically.
2. When a Receiver hears this message and it wants to use the UDL, it sends a "DTCP Request" message to a Feeder to establish a tunnel between the Feeder and the Receiver. Receiver MAC address is included in this message.
3. The Feeder sends back a "DTCP Reply" message to the Receiver to notify of acceptance or rejection. If the Feeder accepts the request, a tunnel is established between the Feeder and the Receiver. A UDL IP address is dynamically allocated to the Receiver, if it does not have a statically allocated UDL IP address.
4. The Feeder and the Receiver exchange routing information over the tunnel and the UDL to use the UDL as an IP route.
5. Packets start to go through the UDL.

As figured above, DTCP helps the procedure 1. through 3. The procedure 4. and 5. are out of the scope of this document. Feeder selection algorithm by Receivers for the case where multiple Feeders exist on a UDL is also out of the scope of this document.

This document specifies the format and semantics of DTCP. Section 2 outlines how DTCP works in the environment of [1]. Section 3 specifies the DTCP format and the meanings of each field. Section 4 depicts DTCP state transition diagram. Section 5 recommends default values of constants used in this document.

2. The outline of DTCP

DTCP has four types of messages. These messages can be categorized into following two:

1. DTCP Announcement message: "Announcement"
   - This message is periodically sent from each Feeder to Receivers to notify the avalability of the Feeder.

2. DTCP Tunnel Setup messages: "Request", "Reply", "Release"
   - These messages are triggered by a DTCP Announcement message. The aims of these messages are to establish and to release tunnels dynamically.

In the environment of [1], there can be many Receivers per one Feeder. To avoid that a flood of messages comes to a Feeder from many Receivers at the same time, each node SHOULD obey the following rules when sending messages:

1. DTCP Announcement message:
   - A Feeder SHOULD send DTCP Announcement messages at the interval rate as the Feeder specifies in the "interval" field of DTCP Announcement messages. The interval rate SHOULD be fixed during the Feeder keeps sending the messages.

2. DTCP Tunnel Setup messages:
   - A Receiver MUST wait for random seconds between 0 to DTCP_ANN_WAIT before sending a message triggered by a DTCP Announcement message.
   - Only Receivers that are specified by the pattern field of a trigger DTCP Announcement message can send a response message. Other Recievers SHOULD NOT respond to the DTCP Announcement message.
   - When a Feeder is willing to send messages to all Receivers that has a tunnel with the Feeder, the Feeder SHOULD wait for random seconds between 0 to DTCP_ANN_WAIT before sending a message to each Receiver.
   - In other cases, any messages can be sent any time.

If a Receiver has not heard any DTCP Announcement message after its boot, or if a Receiver does not hear DTCP Announcement messages for more than DTCP_ANN_TIMEOUT from a Feeder, or when a Feeder does not send a DTCP Announcement message periodically, a Receiver and a Feeder SHOULD assume DTCP_ANN_INTERVAL as the announced interval rate.

Since Feeders and Receivers can connect to multiple UDLs at the same

time, UDL network addresses SHOULD be unique among those UDLs to allow Feeders and Receivers to distinguish UDLs. That is, UDL network address space SHOULD be either a part of global address space or a part of private address space that are well coordinated among UDL service providers so that UDL IP addresses are uniquely allocated.

The following subsections describe DTCP messages. State names used in these subsections are the one used in the state transition diagram in Section 4.

2.1 DTCP Announcement Message

The DTCP Announcement Message is periodically sent from a Feeder to Receivers through a UDL. The roles of a DTCP Announcement Message are:

- to notify the availability of a Feeder to Receivers.
- to give a clock to Receivers to determine a timing to send a DTCP Tunnel Setup message to the Feeder.
- to announce Feeder's information that is necessary to send a DTCP Request message.

A Feeder SHOULD send DTCP Announcement messages periodically if the Feeder is working and a UDL is administratively available. The interval rate SHOULD be notified in the interval field of this message. The recommended interval rate is DTCP_ANN_INTERVAL. The code field of this message indicates the availability of the Feeder and tunnels.

If the code field of a DTCP Announcement message received by a Receiver is DTCP_CODE_OK, it means the Feeder is available and a new tunnel can be set up with the Feeder. If the Receiver does not have a tunnel yet and the Receiver wants to have one, the Receiver can set up a tunnel by using Tunnel Setup messages (See 2.2). If the Receiver already has a tunnel with the Feeder, the Receiver can keep using the tunnel and the UDL. If the valid time of a tunnel with the UDL is almost expired, the Receiver can request to extend the valid time of the tunnel.

If the code field of a DTCP Announcement message received by a Receiver is DTCP_CODE_FULL, it means the Feeder is available but a new tunnel cannot be setup no more with the Feeder at the moment. If the Receiver does not have a tunnel yet and the Receiver wants to set up a tunnel, the Receiver SHOULD wait for another DTCP Announcement message with code value DTCP_CODE_OK. If the Receiver already has a tunnel with the Feeder, the Receiver can keep using the tunnel and the UDL. If the valid time of a tunnel with the UDL is almost

expired, the Receiver can request to extend the valid time of the tunnel.

If the code field of a DTCP Announcement message received by a Receiver is DTCP_CODE_GOINGDOWN, or the Receiver does not hear a DTCP Announcement message from a Feeder for DTCP_ANN_TIMEOUT, it means the Feeder is not available and tunnels should be released if the Receiver has. If the Receiver wants to use the UDL, the Receiver needs to establish a tunnel with other Feeder.

If the sequence field of a DTCP Announcement message received by a Receiver changes more than expected, it means that the Feeder has been booted without saving information of Recievers and tunnels. If the Receiver wants to keep using UDL or to start to use UDL, the Receiver needs to send a DTCP Tunnel Setup Request message to a Feeder.

2.2. DTCP Tunnel Setup Messages

The DTCP Tunnel Setup messages consist of three messages: Request, Reply, and Release. These messages are exchanged between a Feeder and a Receiver through a BDL on demand. They have the same formats, but op field has a different value for each message.

The roles of DTCP Tunnel Setup Messages are:

- to establish and to release a tunnel dynamically between a Feeder and a Receiver. Recievere's UDL IP address can be allocated with these messages in the case where the Receiver does not have a statically allocated UDL IP address.
- to notify Receiver's node information that is necessary for communication to a Feeder. (e.g. Receiver's MAC address)

The outline of the procedure follows. State names here are the state transition diagram in Section 4.

0. A Receiver hears a DTCP Announcement message from a Feeder. (See 2.1) Receiver's state becomes OPEN.

1. The Receiver sends a DTCP Request message to the Feeder. Receiver's MAC address is carried on the DTCP options field if necessary. Receiver's state becomes REQ_SENT. If Feeder receives this message, Feeder's state for the Receiver becomes REQ_RECV. A Feeder has a tunnel state per Receiver.

2. The Feeder sends back a DTCP Reply message to the Receiver. If the request is accepted, the code field of the reply message is DTCP_CODE_OK. The Feeder SHOULD configure the tunnel before

sending back the message. After the Receiver prepares for the tunnel, the tunnel can be used. Both Feeder's state and Receiver's state become ESTABLISHED.

at a Feeder:

- To avoid a flood of request messages to extend the valid time of tunnels from Receivers, the valid time assigned to each Receiver SHOULD be varied between DTCP_TUN_VALIDTIME plus-minus DTCP_TUN_REFRESH.
  If the Feeder does not want assign such a long period, the period can be shorten, but with random variation (plus-minus DTCP_TUN_REFRESH). If the Receiver wants to be assigned shorter period than DTCP_TUN_VALIDTIME, there is no need of variation.

at a Receiver:

- The Receiver SHOULD use the dynamically allocated UDL IP address if allocated.
- The Receiver can send routing information only when the Receiver is allowed to do it.

If the Receiver receives a reply message with its code value DTCP_CODE_NO, or the Receiver does not get any response from a Feeder, the Receiver should wait for another DTCP Announcement message with code value DTCP_CODE_OK. Both Feeder's state and Receiver's state becomes OPEN.

3. If the rest of valid time becomes less than DTCP_TUN_REFRESH, the Receiver sends a DTCP Request message to the Feeder to extend the valid time. By this request, the Receiver can extend the valid time of both a tunnel and the dynamically assigned UDL IP address at the same time. All fields including DTCP options of the DTCP Request message SHOULD be filled as if it is a request message when the Receiver has a statically allocated UDL IP address. Receiver's state becomes REFRESHING, but Feeder's state does not change from ESTABLISHED.

   The Feeder sends back a DTCP Reply message. All fields including DTCP options of the DTCP Reply message SHOULD be filled as if it is a reply message to a request that is to set up a new tunnel. If the Receiver does not receive any reply message, the Receiver SHOULD wait for another DTCP Announcement message with code value DTCP_CODE_OK. If the code field of the reply message is DTCP_CODE_NO, it means the Receiver cannot extend the valid time of the tunnel. After the valid time is expired, both the Feeder and the Receiver unconfigure the tunnel, and MUST NOT keep using the tunnel.

   If the request is accepted, Receiver's state becomes ESTABLISHED. If the request is denied, Receiver's state becomes NOREFRESH. Feeder's state does not change in either case.

4. If a Receiver finishes using a tunnel, the Receiver sends a DTCP release message to the Feeder to release the tunnel. Either Feeder or Receiver can send a DTCP Release message first. The tunnel between the Feeder and the Recever is released after the exchange of DTCP Release messages between them.

   - If a Receiver (or a Feeder) sends a DTCP Release message first, its state becomes REL_WAIT. If the Receiver (or the Feeder) does not receive a response release message, the Receiver (or the Feeder) SHOULD send a DTCP Release message again after a next DTCP Announcement message. After receiving a response release message, its state becomes OPEN.
   - If a Feeder (or a Receiver) receives a DTCP Release message for an existing tunnel, the Feeder (or the Receiver) SHOULD release the tunnel and send back a DTCP Release message. Feeder's state becomes OPEN.
   - If a Feeder or a Receiver receives a DTCP Release message for an unknown tunnel, it SHOULD send back a DTCP Release message for the unknown tunnel.
   - If the releasing is successful, Both Feeder's state and Receiver's state become OPEN.

3. The format of DTCP messages

DTCP is a protocol over UDP. Its port number is (TBD).

3.1. Announcement Message Format

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |       Op      |      Code     |           Interval            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |           Sequence            |Pattern Length |  RecvID type  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                            Pattern                            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                           Feeder ID                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          UDL Netmask                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Feeder BDL IP addr                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |       DTCP options...                                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

src IP addr of the IP header:
   - Feeder's UDL IP address

dst IP addr of the IP header:

```
          - 255.255.255.255 (link-local broadcast address of IPv4)
     dst port of the UDP header:
          - (TBD)

     Op:
          0 = DTCP Announcement message
     Code:
          0 = DTCP_CODE_OK
              This Feeder is active and new tunnels can be set up
              with this Feeder.
          1 = DTCP_CODE_FULL
              This Feeder is active but new tunnels cannot be set
              up with this Feeder.  e.g. address pool is empty,
              or a tunnel table is full.
          2 = DTCP_CODE_GOINGDOWN
              This Feeder will be down soon.
     Interval:
          - Interval time in second to send a DTCP Announcement
            message.  The recommended interval is DTCP_ANN_INTERVAL.
            This field SHOULD not be changed during the Feeder
            keeps sending DTCP Announcement messages.
     Sequence:
          - Sequence number of DTCP Announcemenet messages.
            A Feeder SHOULD assign an initial random number when
            it boots so that Receivers can recognize this Feeder'
            rebooting from the big change of this field.
            The next sequence number of 2^16-1 is 0.
     Pattern:
          - This field is to limit the number of Receivers that
            can respond to a DTCP Announcemnet message to avoid
            a flood of DTCP messages.  Only Receivers who
            satisfies following expression are allowed to respond.
               (Reciever_ID) & MASK == pattern & MASK,
               where MASK = (1 << Pattern_Length) - 1
          - Unused pattern bits SHOULD be zero.
     Pattern Length:
          - See the explanation for pattern field.
     RecvID type: (Receiver-ID type)
          - See the explanation for pattern field.
             0 = Receiver BDL IP address
             1 = Receiver MAC address
     Feeder ID:
          - Identity information of the Feeder.  This field will
            be used to select a Feeder by Receivers.  --(TBD)--
     UDL Netmask:
          - the netmask of the UDL
     Feeder BDL IP addr:
          - Feeder BDL IP address
```

3.2. Request, Reply, Release Messages Format

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Op      |      Code     |           Valid Time          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Request Flags | Accept Flags  |  Tunnel Proto |    Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       src UDL IP addr                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       dst UDL IP addr                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       DTCP options...                                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
src IP addr of the IP header:
     - Source node's BDL IP address
dst IP addr of the IP header:
     - Destination node's BDL IP address
dst port of the UDP header:
     - Default port number is (TBD).

Op:
      8 = Request
          - To request to establish a tunnel, to extend
            the valid time of a tunnel, or to change the
            parameters sent before.
          - Known information SHOULD be put into fields.
      9 = Reply
          - To respond to a request message.
          - Known information SHOULD be put into fields.
     10 = Release
          - To release a tunnel.
Code:
     - In a Request message, MUST be zero.
     - In a Reply message,
         0 = DTCP_CODE_OK
          The request is Accepted.
         1 = DTCP_CODE_FULL
          The request is denied, because tunnel is
          unavailable, e.g. the address pool is empty,
          or tunnel table is full.
         2 = DTCP_CODE_NO
          The request is denied because of
          administrative reasons.
     - In a Release message, MUST be zero.

Valid Time:
     - Valid time of a tunnel in second.  If Receiver's
```

UDL IP address is allocated dynamically, this field also means its valid time.

- In a Request message, the source node puts a valid time it wishes. Zero means source node does not want to use tunnel no more. Zero would be used to release tunnels gracefully.
- In a Reply message, the source node put a valid time it allows. If the source node does not allow to use a tunnel, this field MUST be zero. The source node can tell longer period than the destination wishes, especially when the destination wishes zero seconds.
- In a Release message, MUST be zero.

Request Flags: (Request flags from a source node)
Accept Flags: (Acceptable flags by a destination node)

- In a Release message, all bits MUST be zero.
- In a Request and a Reply message, each bit has following meanings:

(R=Request Flags, A=AcceptFlags, MSB=7)

7: unicast routing info

- R: If source node wants to send unicast routing information, then 1, else 0.
- A: If source node accepts unicast routing information, then 1, else 0.
- If a node says 1 in R, and the opposite node says 1 in A, then unicast routin information can be sent in the direction.

6: multicast routing info

- R: If source node wants to send multicast routing information, then 1, else 0.
- A: If source node accepts multicast routing information, then 1, else 0
- If a node says 1 in R, and the opposite node says 1 in A, then multicast routing information can be sent in the direction.

5: IGMP (join, leave)

- R: If source node wants to send IGMP join/leave, then 1, else 0.
- A: If source node accepts IGMP join/leave, then 1, else 0.
- If a node says 1 in R, and the opposite node says 1 in A, then IGMP join/leave can be sent in the direction.

4: reserved: (MUST BE zero)
3: reserved: (MUST BE zero)
2: reserved: (MUST BE zero)
1: unnumbered or not

- R: If source node wants unnumbered tunnel,

then 1, else 0.

- A: reserved (MUST BE zero)
- If both nodes say 1, then the tunnel is unnumbered.

0: TTL treatment in a tunnel

- R: If source node wants TTL field to be decremented by the number of hops, then 1. If source node wants TTL field to be decremented only by 1, then 0.
- A: reserved (MUST BE zero)
- If both nodes say 1, then TTL is decremented by the number of hops.

- If a node wants to change flags, the node sends a Request message.

Tunnel Proto (Tunneling Protocol):

- Selected tunneling protocol.
- In a request message, if a node can support multiple tunneling protocol, this field SHOULD be zero and a candidates list SHOULD be appeared in DTCP options. If a node can support only one tunneling protocol, this field is filled with the protocol.
- If the opposite node sends a tunneling protocol list in a DTCP options of a request message, this field of a reply message should be filled with the selected protocol. If any of the protocols are not supported, the request SHOULD be rejected and this field MUST be zero.

Reserved:

- (MUST BE zero)

src UDL IP addr:

- Source node's UDL IP address
- If source node knows its UDL IP address, this field SHOULD be filled with the address. For example, a UDL IP address is statically allocated, or it has been already allocated dynamically by a previous request.
- If source node does not know its UDL IP address, this field MUST be zero. For example, in the case where requesting a UDL IP address in a Request message.

dst UDL IP addr:

- Destination node's UDL IP address
- If source node knows destination's UDL IP address, this field SHOULD be filled with the address.
- If source node does not know destination's UDL IP address, this field MUST be zero. For example, in

the case where notifying a failure of a tunnel setup request by a Reply message.

DTCP options:

- All the options that was necessary to establish tunnel SHOULD appear in all Request and Reply messages. This would avoid troubles if opposite side has been rebooted after last exchange of DTCP messages.

3.3. DTCP options:

```
  - No Operation
     +-------+
     | type  |
     +-------+
     type: 0 = NoOp

  - No Operation
     +-------+-------+-------+-------+
     | type  | len   | padding...    |
     +-------+-------+-------+-------+
     type: 1 = NoOp, len = N
     - padding MUST be zero.

  - UDL Netmask
     +-------+-------+
     | type  | len   |
     +-------+-------+-------+-------+
     |       UDL Netmask             |
     +-------+-------+-------+-------+
     type: 2 = UDL Netmask, len = 6
     - to notify UDL Netmask.

  - Acceptable Tunneling Protocol option
     +-------+-------+-------+-------+
     | type  | len   |list of proto..|
     +-------+-------+-------+-------+
     type: 3 = acceptable tunnel proto, len = N
     - To notify acceptable tunneling protocol.  Protocols are
       listed in the order of precedence.
     - Each protocol is denoted by 1 byte.
          - 1: ipproto=4 (IP in IP)
          - 2: ipproto=94, raw packet format (IP within IP)
     - Selected protocol is notified by using the tunnel protocol
       field of DTCP Tunnel Setup messages.
```

```
  - Authentication option (Certification?)
     +-------+-------+-------+-------+
     | type  | len   |    ........   |
     +-------+-------+-------+-------+
     type: 4 = authentication info, len = N
     - (TBD)
          - basic authentication
          - otp
          - etc.

  - Source node's MAC address
     +-------+-------+-------+-------+
     | type  | len   |   MAC(high)   |
     +-------+-------+-------+-------+
     |           MAC(low)            |
     +-------+-------+-------+-------+
     type: 5 = src MAC addr, len = 8
     - to notify source node's MAC address (or EUI-48).
     - Typically, to notify Receiver's MAC address to a Feeder.
```

4. State Transition Diagram

State Transition Diagram for DTCP Tunnel Setup Messages:
(Event/Action style)

```
                            +-------------+ ---+ REQ0/REL
                            |   CLOSED    |    | REL/REL
                            +-------------+ <--+
                                   A
                                   |
                                   V
                            +-------------+
     +--------------------->|    OPEN     |<---------------------------+
     |                      +-------------+                            |
     |                        /         \                              |
     |                       /           \                             |
     |                      /(Next+wait)  \REQUEST/[C]                 |
     |                     / /REQUEST      \                           |
     |                    /                 \                          |
     |(LinkDown),        v                   v                         |
     |NO/        +-----------+ REQUEST/  +-----------+                 |
     +<----------| REQ_SENT  |---------->| REQ_RECV  |                 |
     |           +-----------+           +-----------+                 |
     |         A   |     \                   /                         |
     |         |   V      \                 /                          |
     |         +---+       \               /                           |
     |    (Next+wait)/REQ   \             /                            |
     |                       \           /                             |
+-----------+ ---+(Next+wait) |         |                              |
| REL_WAIT  |    |/REL        |OK/      |/OK                           |
+-----------+ <--+            |         |                              |
     A                        |         |          RELEASE/RELEASE     |[!T]
     |[!T]                    |[T]      |[T] +------------------------>+
     |                        |         |   /                          |
     |(LinkDown),REQ0,        v         v  /                           |
     |(EndOfComm)/RELEASE   +-------------+ ---+                       |
     +<---------------------| ESTABLISHED |    |REQUEST/OKorNO         |
     |                      +-------------+ <--+                       |
     |                        A         |                              |
     |                        |(VT>x),  |(ValidTime<x&&cont)/REQUEST   |
     |(LinkDown),REQ0,        |(!cont)  |(stop)                        |
     |(ValidTime==0),         |         v                              |
     |(EndOfComm)/RELEASE   +-------------+ ---+(Next+wait)/REQUEST    |
     +<---------------------| REFRESHING  |    |OK/                    |
     |                      +-------------+ <--+                       |
     |                        A         |  \                           |
     |                        |REQ/OK   |   \    RELEASE/RELEASE       |
     |(LinkDown),REQ0,        |OK/      |NO/ +------------------------>+
     |(ValidTime==0),         |         v                              |
     |(EndOfComm)/RELEASE   +-------------+      RELEASE/RELEASE       |
     +<---------------------|  NOREFRESH  |--------------------------->+
                            +-------------+
```

Next: DTCP Announcemenet with DTCP_CODE_OK
wait: random wait
LinkDown: TimeOut or DTCP Announcement with DTCP_CODE_GOINGDOWN

[T]: Tunnel up
[!T]: Tunnel down
[C]: check whether the request is acceptable

5. Recommended constant values or expressions

```
  DTCP_ANN_INTERVAL:     10 (sec)
     - Interval time of sending DTCP Announcement messages

  DTCP_ANN_WAIT:    interval / 2  (5 sec by default)
     - The maximum waiting time to respond after receiving a DTCP
       Announcement message.

  DTCP_ANN_TIMEOUT: interval * 6  (60 sec by default)
     - The valid time of a DTCP Announcement message.

  DTCP_TUN_VALIDTIME:    1200 (sec) (= interval * 120)
     - Tunnel valid time.

  DTCP_TUN_REFRESH: interval * 12  (120 sec by default)
     - Threshold value to start extending tunnel valid time.
```

Security Considerations

Security issues are not discussed in this document.

Acknowledgements

The concept and the basic design of DTCP is a result of discussions by the members of Wish-WG of WIDE Project. Noboru Fujii and Mikiyo Nishida give members technical valuable input from their independent experience of implementation. Hitoshi Asaeda, Akira Kato, Kazuhiro Hara, Jun Takei, and Akihiro Tosaka give members technical valuable comments.

References

[1] H. Izumiyama, A. Tosaka, A. Kato,
"An IP tunneling approach for Uni-directional Link routing",
<draft-ietf-udlr-wide-tunnel-00.txt>, Nov 1997

[2] H. Izumiyama, A. Tosaka,
"Dynamic Tunneling Path Configuration for Uni-directional Link Routing",
<draft-ietf-udlr-dtpc-00.txt>, Nov 1997

Authors Information

Noritoshi Demizu
Sony Computer Science Laboratory, Inc.
Takanawa Muse Bldg.
3-14-13, Higashigotanda,
Shinagawa-ku, Tokyo, 141 Japan
Phone: +81-3-5448-4380
E-mail: demizu@csl.sony.co.jp
E-mail: nori-d@is.aist-nara.ac.jp

Hidetaka Izumiyama
Japan Satellite Systems Inc.
Toranomon 17 Mori Bldg.5F
1-26-5 Toranomon,
Minato-ku, Tokyo 105 Japan
Phone: +81-3-5511-7568
Fax: +81-3-5512-7181
E-mail: izu@jcsat.co.jp