         Cisco Systems UniDirectional Link Detection (UDLD) Protocol

Status of This Memo

IESG Note

   This RFC is not a candidate for any level of Internet Standard. The
   IETF disclaims any knowledge of the fitness of this RFC for any
   purpose and in particular notes that the decision to publish is not
   based on IETF review for such things as security, congestion control,
   or inappropriate interaction with deployed protocols. The RFC Editor
   has chosen to publish this document at its discretion. Readers of
   this document should exercise caution in evaluating its value for
   implementation and deployment. See RFC 3932 for more information.

Abstract

   This document describes a Cisco Systems protocol that can be used to
   detect and disable unidirectional Ethernet fiber or copper links
   caused, for instance, by mis-wiring of fiber strands, interface
   malfunctions, media converters' faults, etc. It operates at Layer 2
   in conjunction with IEEE 802.3's existing Layer 1 fault detection
   mechanisms.

   This document explains the protocol objectives and applications,
   illustrates the specific premises the protocol was based upon, and
   describes the protocol architecture and related deployment issues to
   serve as a possible base for future standardization.

Table of Contents

1. Introduction

   Today's Ethernet-based switched networks often rely on the Spanning
   Tree Protocol (STP) defined in the IEEE 802.1D standard [1] to create
   a loop-free topology that is used to forward the traffic from a
   source to a destination based on the Layer 2 packet information
   learned by the switches and on the knowledge of the status of the
   physical links along the path.

   Issues arise when, due to mis-wirings or to hardware faults, the
   communication path behaves abnormally and generates forwarding
   anomalies. The simplest example of such anomalies is the case of a
   bidirectional link that stops passing traffic in one direction and
   therefore breaks one of the most basic assumptions that high-level
   protocols typically depend upon: reliable two-way communication
   between peers.

   The purpose of the UDLD protocol is to detect the presence of
   anomalous conditions in the Layer 2 communication channel, while
   relying on the mechanisms defined by the IEEE in the 802.3 standard
   [2] to properly handle conditions inherent to the physical layer.

2. Protocol Objectives and Applications

   The UniDirectional Link Detection protocol (often referred to in
   short as "UDLD") is a lightweight protocol that can be used to detect
   and disable one-way connections before they create dangerous
   situations such as Spanning Tree loops or other protocol
   malfunctions.

   The protocol's main goal is to advertise the identities of all the
   capable devices attached to the same LAN segment and to collect the
   information received on the ports of each device to determine if the
   Layer 2 communication is happening in the appropriate fashion.

   In a network that has an extensive fiber cabling plant, problems may
   arise when incorrect patching causes a switch port to have its RX
   fiber strand connected to one neighbor port and its TX fiber strand
   connected to another. In these cases, a port may be deemed active if
   it is receiving an optical signal on its RX strand. However, the
   problem is that this link does not provide a valid communication path
   at Layer 2 (and above).

   If this scenario of wrongly connected fiber strands is applied to
   multiple ports to create a fiber loop, each device in the loop could
   directly send packets to a neighbor but would not be able to receive
   from that neighbor.

   Albeit the above scenario is rather extreme, it exemplifies how the
   lack of mutual identification of the neighbors can bring protocols to
   the wrong assumption that during a transmission the sender and the
   receiver are always properly matched. Another equally dangerous
   incorrect assumption is that the lack of reception of protocol
   messages on a port unmistakably indicates the absence of transmitting
   protocol entities at the other end of the link.

   The UDLD protocol was implemented to help correct certain assumptions
   made by other protocols, and in particular to help the Spanning Tree
   Protocol to function properly so as to avoid the creation of
   dangerous Layer 2 loops. It has been available on most Cisco Systems
   switches for several years and is now part of numerous network design
   best practices.

3. Protocol Design Premises

   The current implementation of UDLD is based on the following
   considerations/presuppositions:

   o The protocol would have to be run in the control plane of a
      network device to be flexible enough to support upgrades and
      bug fixes. The control plane speed would ultimately be the
      limiting factor to the capability of fast fault detection of
      the protocol (CPU speed, task switching speed, event processing
      speed, etc.). The transmission medium's propagation delay at
      10 Mbps speed (or higher) would instead be considered a
      negligible factor.

   o Network events typically do not happen with optimal timing, but
      rather at the speed determined by the software/firmware
      infrastructure that controls them. (For psychological and
      practical reasons, developers tend to choose round timer values
      rather than determine the optimal value for the specific
      software architecture in use. Also, software bugs, coding
      oversights, slow process switching, implementation overhead can
      all affect the control plane responsiveness and event timings.)
      Hence it was deemed necessary to adopt a conservative protocol
      design to minimize false positives during the detection
      process.

   o If a fault were discovered, it was assumed that the user would
      want to keep the faulty port down for a predetermined amount of
      time to avoid unnecessary port state flapping. For that
      reason, a time-based fault recovery mechanism was provided
      (although alternative recovery mechanisms are not implicitly
      precluded by the protocol itself).

4. Protocol Background

   UDLD is meant to be a Layer 2 detection protocol that works on top of
   the existing Layer 1 detection mechanisms defined by the IEEE
   standards. For example, the Far End Fault Indication (FEFI) function
   for 100BaseFX interfaces and the Auto-Negotiation function for
   100BaseTX/1000BaseX interfaces represent standard physical-layer
   mechanisms to determine if the transmission media is bidirectional.
   (Please see sections 24.3.2.1 and 28.2.3.5 of [2] for more details.)
   The typical case of a Layer 1 "fault" indication is the "loss of
   light" indication.

   UDLD differs from the above-mentioned mechanisms insofar as it
   performs mutual neighbor identification; in addition, it performs
   neighbor acknowledgement on top of the Logical Link Control (LLC)

layer and thus is able to discover logical one-way miscommunication
between neighbors even when either one of the said PHY layer
mechanisms has deemed the transmission medium bidirectional.

5. Protocol Architecture

5.1. The Basics

   UDLD uses two basic mechanisms:

      a. It advertises a port's identity and learns about its neighbors
         on a specific LAN segment; it keeps the acquired information on
         the neighbors in a cache table.

      b. It sends a train of echo messages in certain circumstances that
         require fast notifications or fast resynchronization of the
         cached information.

   Because of the above, the algorithm run by UDLD requires that all the
   devices connected to the same LAN segment be running the protocol in
   order for a potential misconfiguration to be detected and for a
   prompt corrective action to be taken.

5.2. Neighbor Database Maintenance

   UDLD sends periodical "hello" packets (also called "advertisements"
   or "probes") on every active interface to keep each device informed
   about its neighbors. When a hello message is received, it is cached
   and kept in memory at most for a defined time interval, called
   "holdtime" or "time-to-live", after which the cache entry is
   considered stale and is aged out.

   If a new hello message is received when a correspondent old cache
   entry has not been aged out yet, then the old entry is dropped and is
   replaced by the new one with a reset time-to-live timer. Whenever an
   interface gets disabled and UDLD is running, or whenever UDLD is
   disabled on an interface, or whenever the device is reset, all
   existing cache entries for the interfaces affected by the
   configuration change are cleared, and UDLD sends at least one message
   to inform the neighbors to flush the part of their caches also
   affected by the status change. This mechanism is meant to keep the
   caches coherent on all the connected devices.

5.3. Event-driven Detection and Echoing

   The echoing mechanism is the base of UDLD's detection algorithm:
   whenever a UDLD device learns about a new neighbor or receives a
   resynchronization request from an out-of-synch neighbor, it
   (re)starts the detection process on its side of the connection and
   sends N echo messages in reply. (This mechanism implicitly assumes
   that N packets are sufficient to get through a link and reach the
   other end, even though some of them might get dropped during the
   transmission.)

   Since this behavior must be the same on all the neighbors, the sender
   of the echoes expects to receive (after some time) an echo in reply.
   If the detection process ends without the proper echo information
   being received, and under specific conditions, the link is considered
   to be unidirectional.

5.4. Event-based versus Event-less Detection

   UDLD can function in two modes: normal mode and aggressive mode.

   In normal mode, a protocol determination at the end of the detection
   process is always based on information received in UDLD messages:
   whether it's the information about the exchange of proper neighbor
   identifications or the information about the absence of such proper
   identifications. Hence, albeit bound by a timer, normal mode
   determinations are always based on gleaned information, and as such
   are "event-based". If no such information can be obtained (e.g.,
   because of a bidirectional loss of connectivity), UDLD follows a
   conservative approach based on the considerations in Section 3 and
   deems a port to be in "undetermined" state. In other words, normal
   mode will shut down a port only if it can explicitly determine that
   the associated link is faulty for an extended period of time.

   In contrast, in aggressive mode, UDLD will also shut down a port if
   it loses bidirectional connectivity with the neighbor for the same
   extended period of time mentioned above and subsequently fails
   repeated last-resort attempts to re-establish communication with the
   other end of the link. This mode of operation assumes that loss of
   communication with the neighbor is a meaningful network event in
   itself and is a symptom of a serious connectivity problem. Because
   this type of detection can be event-less, and lack of information
   cannot always be associated to an actual malfunction of the link,
   this mode is optional and is recommended only in certain scenarios
   (typically only on point-to-point links where no communication
   failure between two neighbors is admissible).

6. Packet Format

   The UDLD protocol runs on top of the LLC sub-layer of the data link
   layer of the OSI model. It uses a specially assigned multicast
   destination MAC address and encapsulates its messages using the
   standard Subnetwork Access Protocol (SNAP) format as described in the
   following:

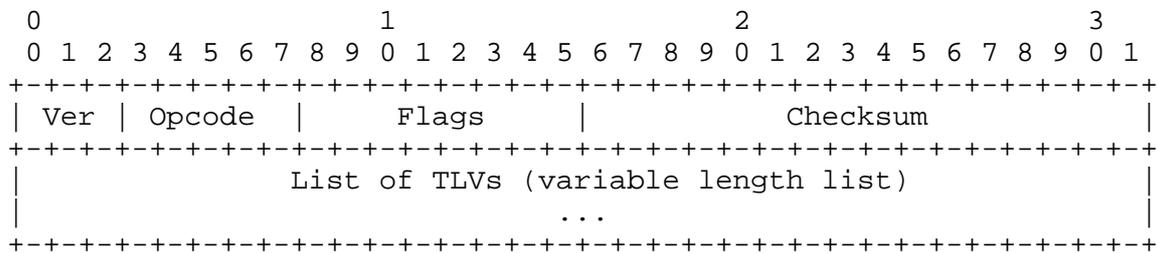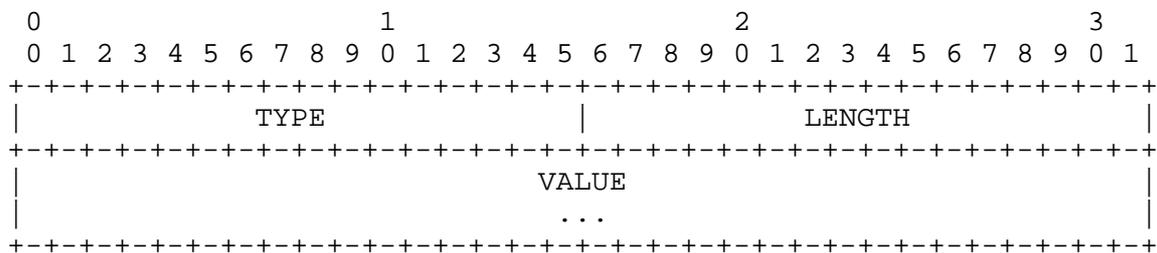   Destination MAC address   01-00-0C-CC-CC-CC

         UDLD SNAP format:
           LLC value   0xAAAA03
           Org Id   0x00000C
           HDLC protocol type   0x0111

   UDLD's Protocol Data Unit (PDU) format is as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Ver | Opcode  |     Flags      |           Checksum           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                List of TLVs (variable length list)            |
   |                              ...                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   The TLV format is the basic one described below:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              TYPE             |            LENGTH            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                            VALUE                             |
   |                              ...                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   Type (16 bits): If an implementation does not understand a Type
         value, it should skip over it using the length field.

   Length (16 bits): Length in bytes of the Type, Length, and Value
         fields. In order for this field value to be valid, it should
         be greater than or equal to the minimum allowed length, 4
         bytes. If the value is less than the minimum, the whole packet
         is to be considered corrupted and therefore it must be
         discarded right away during the parsing process. TLVs should
         not be split across packet boundaries.

Value (variable length): Object contained in the TLV.

The protocol header fields are defined as follows:

Ver (3 bits):
     0x01: UDLD PDU version number

Opcode (5 bits):
     0x00: Reserved
     0x01: Probe message
     0x02: Echo message
     0x03: Flush message
     0x04-0x1F: Reserved for future use

Flags (8 bits):
     bit 0: Recommended timeout flag (RT)
     bit 1: ReSynch flag (RSY)
     bit 2-7: Reserved for future use

PDU Checksum (16 bits):
     IP-like checksum. Take the one's complement of the one's
     complement sum (with the modification that the odd byte at the
     end of an odd length message is used as the low 8 bits of an
     extra word, rather than as the high 8 bits.) NB: All UDLD
     implementations must comply with this specification.

The list of currently defined TLVs comprises:

Name	Type	Value format
Device-ID TLV	0x0001	ASCII character string
Port-ID TLV	0x0002	ASCII character string
Echo TLV	0x0003	List of ID pairs
Message Interval TLV	0x0004	8-bit unsigned integer
Timeout Interval TLV	0x0005	8-bit unsigned integer
Device Name TLV	0x0006	ASCII character string
Sequence Number TLV	0x0007	32-bit unsigned integer
Reserved TLVs	> 0x0007	Format unknown. To be skipped by parsing routine.

6.1. TLV Description

   Device-ID TLV:

      This TLV uniquely identifies the device that is sending the UDLD
      packet. The TLV length field determines the length of the carried
      identifier and must be greater than zero. In version 1 of the
      protocol, the lack of this ID is considered a symptom of packet
      corruption that implies that the message is invalid and must be
      discarded.

   Port-ID TLV:

      This TLV uniquely identifies the physical port the UDLD packet is
      sent on. The TLV length field determines the length of the
      carried identifier and must be greater than zero. In version 1 of
      the protocol, the lack of this ID is considered a symptom of
      packet corruption that implies that the message is invalid and
      must be discarded.

   Echo TLV:

      This TLV contains the list of valid DeviceID/PortID pairs received
      by a port from all its neighbors. If either one of the
      identifiers in a pair is corrupted, the message will be ignored.
      This list includes only DeviceIDs and PortIDs extracted from UDLD
      messages received and cached on the same interface on which this
      TLV is sent. If no UDLD messages are received, then this TLV is
      sent containing zero pairs. Despite its name, this TLV must be
      present in both probe and echo messages, whereas in flush messages
      it's not required.

   Message Interval TLV:

      This required TLV contains the 8-bit time interval value used by a
      neighbor to send UDLD probes after the linkup or detection phases.
      Its time unit is 1 second. The holdtime of a cache item for a
      received message is calculated as (advertised-message-interval x
      R), where R is a constant called "TTL to message interval ratio".

   Timeout Interval TLV:

      This optional TLV contains the 8-bit timeout interval value (T)
      used by UDLD to decide the basic length of the detection phase.
      Its time unit is 1 second. If it's not present in an
      advertisement, T is assumed to be a hard-coded constant.

Device Name TLV:

    This required TLV is meant to be used by the CLI or SNMP and
typically contains the user-readable device name string.

Sequence Number TLV:

    The purpose of this optional TLV is to inform the neighbors of the
sequence number of the current message being transmitted. A
counter from 1 to $2^{32}-1$ is supposed to keep track of the sequence
number; it is reset whenever a transition of phase occurs so that
it will restart counting from one, for instance, whenever an echo
message sequence is initiated, or whenever a linkup message train
is triggered.

    No wraparound of the counter is supposed to happen.

    The zero value is reserved and can be used as a representation of
a missing or invalid sequence number by the user interface.
Therefore, the TLV should never contain zero. (NB: The use of
this TLV is currently limited only to informational purposes.)

7. Protocol Logic

UDLD's protocol logic relies on specific internal timers and is
sensitive to certain network events.

The type of messages that UDLD transmits and the timing intervals
that it uses are dependent upon the internal state of the protocol,
which changes based on network events such as:

    o Link up
    o Link down
    o Protocol enabled
    o Protocol disabled
    o New neighbor discovery
    o Neighbor state change
    o Neighbor resynchronization requests

7.1. Protocol Timers

UDLD timer values could vary within certain "safety" ranges based on
the considerations in Section 3. However, in practice, in the
current implementation, timers use only certain values verified
during testing. Their time unit is one second.

During the detection phase, messages are exchanged at the maximum
possible rate of one per second. After that, if the protocol reaches

a stable state and can make a certain determination on the
"bidirectionality" of the link, the message interval is increased to
a configurable value based on a curve known as M1(t), a time-based
function.

In case the link is deemed anything other than bidirectional at the
end of the detection, this curve is a flat line with a fixed value of
Mfast (7 seconds in the current implementation).

In case the link is instead deemed bidirectional, the curve will use
Mfast for the first 4 subsequent message transmissions and then will
transition to an Mslow value for all other steady-state
transmissions. Mslow can be either a fixed value (60 seconds in some
obsolete implementations) or a user-configurable value (between Mfast
and 90 seconds with a default of 15 seconds in the current
implementations).

8. Comparison with Bidirectional Forwarding Detection

   Similarly to UDLD, the Bidirectional Forwarding Detection (BFD) [3]
   protocol is intended to detect faults in the path between two network
   nodes. However, BFD is supposed to operate independently of media,
   data protocols, and routing protocols. There is no address discovery
   mechanism in BFD, which is left to the application to determine.

   On the other hand, UDLD works exclusively on top of a L2 transport
   supporting the SNAP encapsulation and operates independently of the
   other bridge protocols (UDLD's main "applications"), with which it
   has limited interaction. It also performs full neighbor discovery on
   point-to-point and point-to-multipoint media.

9. Security Considerations

   In a heterogeneous Layer 2 network that is built with different
   models of network devices or with devices running different software
   images, the UDLD protocol should be supported and configured on all
   ports interconnecting said devices in order to achieve a complete
   coverage of its detection process. Note that UDLD is not supposed to
   be used on ports connected to untrusted devices or incapable devices;
   hence, it should be disabled on such ports.

10. Deployment Considerations

   Cisco Systems has supported the UDLD protocol in its Catalyst family
   of switches since 1999.

11. Normative References

   [1]   IEEE 802.1D-2004 Standard -- Media access control (MAC) Bridges

   [2]   IEEE 802.3-2002 IEEE Standard -- Local and metropolitan area
         networks Specific requirements--Part 3: Carrier Sense Multiple
         Access with Collision Detection (CSMA/CD) Access Method and
         Physical Layer Specifications

12. Informative Reference

   [3]   Katz, D., and D. Ward, "Bidirectional Forwarding Detection",
         Work in Progress, March 2008.

Author's Address

   Marco Foschiano
   Cisco Systems, Inc.
   Via Torri Bianche 7,
   20059 Vimercate (Mi)
   Italy

   EMail: foschia@cisco.com