iADT Discussions

Socket basics; Socket lifetime issues;
Service Responses; $Sense_a$ inconsistency

Purpose

- Prepare common ground for discussions
- This shows recent improvement of my understanding of Ethernet
- Want to make sure we all understand the very most basics the same way
- Then cover some issues

Socket Basics

Server is listening

Client does the connect

Sockaddress = IP Address + Port

TCP = Socket stream

- Has a connection

UDP = Socket datagram

- Send and forget

Connection allows two-way connection. Either side can begin conversation

Basic API calls

- Socket()
  - Returns a handle to an internal structure
- Bind()
  - IP_addr + port gets bound to the socket
  - Still only internal
  - I relate this to the ‘I’ in I_T nexus
- Once bound then ready to do:
  - Connect(); or
  - Listen()

Basic API calls (cont.)

- Listen()
  - Allows a bound socket to accept a connection from an external entity
- Connect()
  - Requests a connection between an internal socket and an external ip_addr+port
  - Sends the SYN frame
- Accept()
  - Accepts a connect request
  - Once completed then I think of an I_T nexus existing

Basic API calls (cont.)

- Recv()
  - Parameters
    - Socket, buf, len
  - Receives data into buf
- Send()
  - Parameters
    - Socket, buf, len
  - Queues the data to send
    - Push() will flush the queue when allowed

Basic API calls (cont.)

- Close()
  - Sends FIN
  - After close can only ACK
  - Connection is not fully closed until other side sends FIN;ACK

Connection lifetime

- Long-lived vs. short-lived
  - We have said that either should be allowed and nothing prohibits short-lived but that most of us have been thinking in terms of long-lived connections.
    - I agree we need to allow either
    - We need to understand consequences of each
  - The definition of long-lived might be something like, “does not close at the end of an exchange”
- Does login persist across multiple connections or does it logout (explict or implicit) on connection loss

Connection lifetime (cont.)

- Automation
  - Has to accept a connection at anytime from any drive
    - Could have hundreds of drives in one library
  - If long-lived connections library must be able to manage up to hundreds of concurrent connections
  - If short-lived connections and login is limited to a single connection, then there will be a lot of overhead due to login's

Connection lifetime (cont.)

- When connection is short-lived and one side intends to use it for a single command then close it, how do we keep the other side from using it for a different command?
  - E.g., Automation opens a connection and sends a Mode Select. DTD receives a RES on LUN 1.
  - Issue if login does not persist across connections

Connection lifetime (cont.)

- If library is doing LUN 2 command and the DT device receives a LUN 1 command, does it create a new connection, use the existing connection?
  - If it creates a new connection does it login? If so does it destroy the session created for the LUN2 connection?

Login persistence

- Assume login is not persistent
  - Receiver does not know when sender closes a port unless LOGO received
  - If something is queued up and a connection needs to be opened how can we login? The PLOGI would be queued behind whatever is in the queue.
- Login should be persistent across connections

Actions to service responses

- What happens if connection cannot be opened?
  - i.e., CONNECTION REFUSED
- Need to define explicit behavior by port that receives a non-good service response.

$Sense_a$

- $Sense_a$ indicates that a DT device is in an automation device. It has no meaning related to the presence of an Ethernet connector.
  - Listen() & Connect() show the following which does not make sense in light of this
    - NO PHYSICAL CONNECTION: The request failed because the Sense connection was not asserted.
- We have no indication that Ethernet connector is attached – this is independent of DT device being in a library

Questions received internally

- "Since iADT is not specifying a connector, is there any reason to force 10/100 Ethernet? Everything else is just IP (which can be sent over Ethernet, Token Ring, serial, FC, etc). So why limit this to 10/100 Ethernet? It seems better to just define to the IP layer and then let the rest take care of itself. This would allow expansion to Gigabit Ethernet without a change to the spec. I realize that they may not need gigabit ethernet, but it might be easier to use that in some systems - especially if the drive/library has another port which is gigabit ethernet. For example, if figure 7 in the 07-469r7.pdf stopped at the IP layer, then this would work over any interface that supports IP."
  - How do we ensure interoperability without specifying any physical layer. As long as the vendor in their procurement spec specified iADT over 10/100 ethernet or iADT over Gigabit ethernet, or iADT over FC then I think that would be complete.

Questions received internally

- "I was thinking about the drive location issue some and had a strange thought. If they added another sense line (or redefined an existing sense line), so that the line could be toggled by the drive on demand, then the drive could default the line to off, and turn it on or off when the library requested. This would provide a way for the library to know which drive was connected where by requesting the drive to drive the sense line on and seeing which line went on. Normally, I would just consider this to be an implementation issue (an extra line in addition to any that iADT specified), except that it would be easier if iADT specified the command to toggle the signal line at the ADT layer instead of a special SCSI command (perhaps a new link service or fast access IU type?)."
  - Potential signals would by either $Sense_{aux}$ or $Signal_{aux}$