iSCSI

iS not an Apple appliance

Daniel Braniss
danny@cs.huji.ac.il

BSDCan 2008, Ottawa
why

- learn a new OS/kernel
- check feasibility of writing a driver knowing little about:
  - SCSI
  - network
- see what iSCSI is about
what's iSCSI

- iINTERNET Small Computer System Interface
- iINSECURE SCSI
- iINTERESTING
SCSI
Small Computer Storage Interface
SCSI ...

/dev/da0
/dev/sa0
/dev/ch0

...
/dev/da0
/dev/sa0
/dev/ch0
...

disk driver (da)
tape driver (sa)

CAM
Common Access Method

iSCSI -HBA
initiator

iscsi target
iscsi target

Virtual

Virtual
from bus to ether

- IP Header: 20 bytes
- TCP Header: 20 bytes
- ISCSI Header: 48 bytes
- ISCSI Data: 0 or more bytes
iSCSI RFC 3720

- 257 pages - 14395 lines
- 7226 lines of code
- first products shipped in 2001 (IBM, Cisco)
- the RFC is from 2003/2004.
iSCSI login

there are 2 types of sessions:

• Discovery:
  • the response is a list of targets

• Normal:
  • goes into Full Feature mode/phase
first steps

1. read RFC for the nth. time
2. write some code
   2.1. translate designers language into programming language.
3. run
4. goto 1
iSCSI login ...

- needed to start a session
- establish authenticity of both/either/neither target and/or initiator based on
  - target name - iqn.1992-08.com.netapp:sn.50405391
  - initiator name - iqn.2005-il.ac.huji.cs:xhost
- if CHAP then also some shared secret
- negotiate some options:
  - header/data CRC
## iSCSI PDU

### Protocol Data Unit

<table>
<thead>
<tr>
<th></th>
<th>byte 0</th>
<th>byte 1</th>
<th>byte 2</th>
<th>byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Basic Header Segment (BHS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Additional Header Segment 1(optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>Additional Header Segment 2(optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Additional Header Segment n(optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>Header-Digest (optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>Data-Segment (optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>Data-Digest (optional)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### iSCSI PDU...

#### Basic Header Segment - BHS

<table>
<thead>
<tr>
<th></th>
<th>byte 0</th>
<th>byte 1</th>
<th>byte 2</th>
<th>byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>.</td>
<td>I</td>
<td>Opcode</td>
<td>F</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>TotalAHSLength</td>
<td>DataSegmentLength</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td>LUN or Opcode-Specific Fields</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td>Initiator Task Tag</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>Opcode-Specific Fields</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
architectural view

iscsi

output handler

input handler

session manager

CAM handler

iscsi
target

iscontrol
connection
link recovery
login
authentication
from RFC 3720

7.1.3. Standard Connection State Diagram for an Initiator

Symbolic names for States:

S1: FREE
S2: XPT_WAIT
S4: IN_LOGIN
S5: LOGGED_IN
S6: IN_LOGOUT
S7: LOGOUT_REQUESTED
S8: CLEANUP_WAIT
iscontrol ...

Globals
#
port = 3260

myiscsi {
    targetname = iqn.1900.com.com:sn.123456
    initiatorname = iqn.2005-01.il.ac.huji.cs:somebody
    targetaddress = iscsi1; tags = 8; port=3333;
}

chaptest {
    targetaddress = iscsi2
    targetname = iqn.1900.com.com:sn.123456
    initiatorname = iqn.2005-01.il.ac.huji.cs:nobody
    authmethod = CHAP
    chapsecret = 0x3713c3336d9a224c2791c873d3d2b174
    tags = 8
}
iscsi_initiator

- Kernel side of the protocol.
- Consists of several threads.
- Handles PDU queues.
- iSCSI <-> CAM.
- About 4500 lines of code.
iscsi_initiator ...

- Session Manager
  - for each `session`:
    - starts a session-manager
      - which starts a receiver on the given connection/socket
    - loops for something to do till:
      - terminated from user land
      - user land died
iscsi_initiator - receiver

- waits till it gets a 48 byte header
- calculates size of PDU
- process the received PDU
  - if it’s a SCSI command pass it to the CAM

will run as long as there is a connection
iscsi_initiator - dispatcher

• goes through the PDU queues for something to do

• PDU queues:
  ● Immediate
  ● R2T(Request to Transmit)
  ● command (SCSI) - only if window size allows.
  ● hold - waiting for ack

one per session
iscsi_initiator Read

iscsi target

iscsi initiator

decapsulates

ISCSI_READ

cam_done

encapsulates ISCSI_CMD

iscsi initiator

CAM XPT_SCSI_IO
iscsi initiator ...

- sync with the shutdown process
- `iscontrol` gets killed.
- we loose the socket/connection
- finally the buffers get flushed.
iscsi_initiator ... 

- Recovery:
numbers ...
conclusions
to do list

- recovery
- header/data digest
- boot from an iSCSI
- make the CAM aware of virtual scsi devices?
- clean up
- target?