FCoE Target Functional Specification

Version 1.05

Sep 3 2008

1 Project Description

Fibre Channel over Ethernet (FCoE) is a mapping of Fibre Channel over selected full duplex IEEE 802.3 networks. "Selected" means:

- The network must be lossless by supporting 802.3x PAUSE;
- The network must support Jumbo frame to encapsulate the 2KB FC frames;
- The NICs must support multiple unicast MAC addresses, required by FPMA (Fabric Provided MAC Address) defined in FCoE Specification.

The purpose of this project is to provide FCoE target functionality to Solaris.

1.1 Definition

Although FCoE transports FC frames over Ethernet, it still has the same FC-2 and above layers as standard FC. This means we could make use of all common FC features in Solaris and only implement FCoE specific functions.

FC target capability has already been integrated into Solaris with COMSTAR project. In order to provide FCoE target capability, we need to define two new entities :

- An FCA driver which exports FC target port to COMSTAR FCT module (Fibre Channel Port Provider), but talks to FCoE transport instead of a physical FC HBA.
- An FCoE transport which talks to MAC layer to access Ethernet frames directly (VLAN is supported).

In Solaris, most Ethernet controller drivers are implemented under GLDv3 framework where they all register to a common MAC service module. So we are designing the FCoE transport as a GLDv3 MAC client, hence the FCoE transport could run on top of any GLDv3 NIC drivers theoretically.

The management of FCoE target is through fcadm utility. New sub-commands will be added to fcadm to create, delete and list FCoE target ports. These operations are accomplished through interfaces to new library libfcoe.

HBAAPI will be able to handle FCoE target ports just like normal FC target ports.

1.2 Motivation, Goals, and Requirements

FCoE is a newly proposed T11 standard and is advocated by all leading IT vendors. With FCoE, customers can achieve I/O consolidation by leveraging their existing FC investment and knowledge with the 10 Gigabit Ethernet infrastructure. I/O consolidation results in less adapters, less power consumption, less cooling requirement and less cabling. Adding support of FCoE will enrich Solaris

open storage stacks, make Solaris competitive to other Operating Systems in terms of storage connection capability.

The purpose of this project is to provide FCoE target capability for Solairs. This is a software FCoE solution which means it will run with normal Ethernet NICs. It's supposed to run on both x86/x64 and SPARC.

1.3 Changes From the Previous Release

This is a new product.

1.4 Program Plan Overview

1.4.1 Development

Development is ongoing in Beijing. The fcoe driver, fcoet driver and fcadm changes have been prototyped.

1.4.2 Quality Assurance/Testing

Functional tests.

Performance tests: Basic I/O performance over Neptune PCI-E 10G NIC.

Interoperability tests: Will be tested with Cisco Nexus 5000 FCoE switch, normal FC initiators and FCoE initiators (including Open-FCoE software initiator, QLogic CNA and Emulex CNA).

1.4.3 Documentation

Two new manpages fcoet(7d) and fcoe(7d) will be created.

fcadm(1M) will be updated.

Solaris SAN Configuration and Multipathing Guide (820-1931) will be updated.

1.4.4 Release Cycle

An alpha release of the project will be put on OpenSolaris by July 2008. Integration to Solaris Nevada is targeting build 112 (Q3 FY09).

1.4.5 Technical Support

TBD

1.4.6 Training

TBD

1.5 Related Projects

1.5.1 Dependencies on Other Sun Projects

PSARC/2007/523 COMSTAR: Common Multiprotocol SCSI Target

PSARC/2004/291 Fibre Channel HBA Port Utility

PSARC/2004/571 Nemo - a.k.a. GLD v3

PSARC/2006/357 Crossbow - Network Virtualization and Resource Management

1.5.2 Dependencies on Non-Sun Projects

Part of fcoet driver source is ported from Open-FCoE (a Linux based software FCoE implementation) under BSD license. Homepage is at http://www.open-fcoe.org

1.5.3 Sun Projects Depending on this Project

None

1.5.4 Projects Rendered Obsolete by this Project

None

1.5.5 Related Active Projects [Describe the relationship.]

None

1.5.6 Suggested Projects to Enhance this Program

Software Fabric Services running on Solaris systems.

This project will eliminate the requirement of FCoE switches in case that all nodes in the Data Center Ethernet are FCoE nodes.

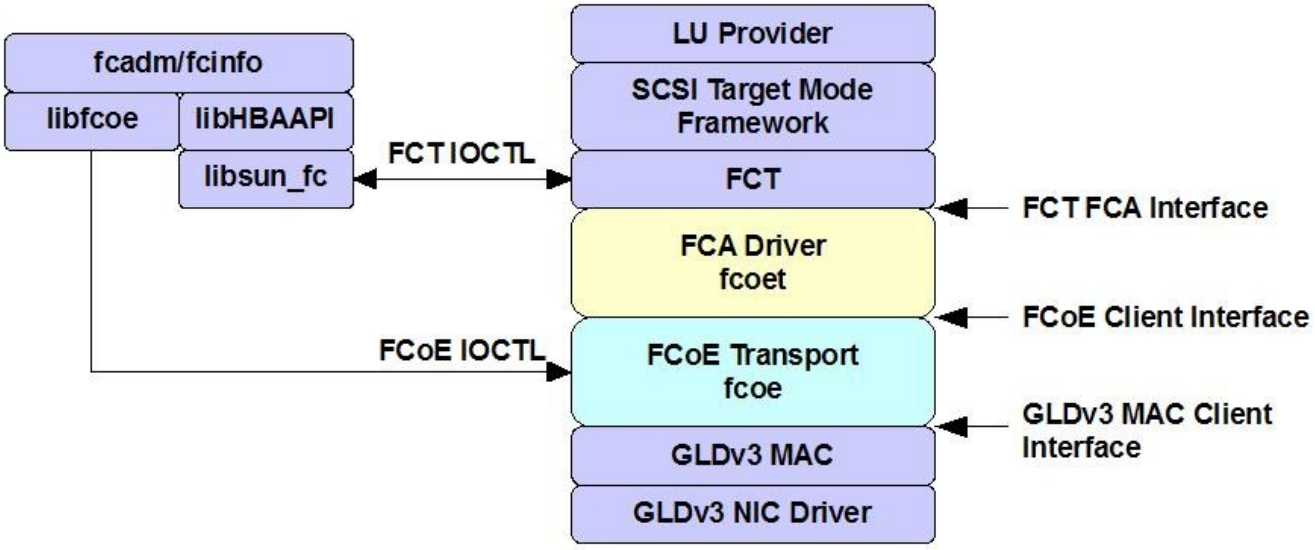
1.6 Competitive Analysis

Intel is maintaining an open-source FCoE target stack on Linux which has both initiator and target capability. Both QLogic and Emulex have announced CNAs (Converged Network Adapter) which are hardware FCoE HBAs. So far no FCoE target devices have been announced by storage vendors.

This project is a software solution, instead of inventing a new FC target mode stack, the existing COMSTAR FCT framework is used. This project may enable Solaris based storage devices (especially SunFire X4500) to be among the earliest FCoE target devices/disk arrays in the market.

2 Technical Description

2.1 Architecture



Two new kernel drivers are introduced in this product.

- The FCA driver is designed as a new pseudo kernel driver "fcoet". It's also an implementation of FCoE VN_Port entity as defined in FCoE Specification.
- The FCoE transport is designed as another pseudo kernel driver "fcoe". It's an implementation of FCoE LEP entity as defined in FCoE Specification.

fcoe and fcoet work together in a service provider/client manner. There is only one instance of fcoe driver across the system which provides FCoE transport services (MAC interface management, FCoE frame encapsulation/de-encapsulation, FCoE frame TX/RX). Multiple fcoet driver instances could co-exist within the same system and consume FCoE transport services provided by fcoe through FCoE Client Interface.

Part of fcoet code is ported from www.open-fcoe.org under BSD license. The inbound OSR and outbound OSR have been approved.

The management of FCoE target is through fcadm utility. New sub-commands will be added to fcadm to create, delete and list FCoE target ports. fcadm will use libfcoe to issue ioctls to fcoe driver to accomplish this.

A new SMF service will create FCoE ports upon boot according to the port configuration information stored in SMF.

2.2 Interfaces

2.2.1 Exported Interfaces

Interface Name	Proposed Stability Classification	Specified in What Document?	Former Stability Classification or Other Comments
FCoE client interface	Project Private	See R.3.1	
fcoe driver IOCTLs	Project Private	See R.3.2	
libfcoe	Committed	See R.3.3	Library interfaces for managing FCoE ports.
fcadm command line	Committed	fcadm.man	
fcadm output	Committed Not an interface	fcadm.man	

2.2.2 Imported Interfaces

Interface Name	Proposed Stability Classification	Specified in What Document?	Former Stability Classification or Other Comments
T11 FCoE Specification	Standard	T11 web site	See Appendix A for exact titles
FCT FCA Interface	Project Private	writing_fca.txt	FCA programming guide for COMSTAR FCT
GLDv3 MAC Client Interface	Consolidation Private	PSARC/2004/571 PSARC/2006/357	Crossbow project will change GLDv3 MAC Client Interface, but is not integrated yet
libscf	Committed		Used for storing FCoE port configurations

2.3 User Interface

There will be extensions to existing CLI user interfaces.

fcadm will allow creating and deleting FCoE ports.

```
fcadm create-fcoe-port [-ito] [-n <Node_WWN>] [-p <Port_WWN>] <mac i/f>
fcadm delete-fcoe-port <mac i/f>
```

See fcadm.man for more details.

2.4 Compatibility and Interoperability

2.4.1 Standards Conformance

FCoE is a technology being developed with TC T11 as part of the FC-BB-5 project. The current documents are at http://www.t11.org/fcoe

2.4.2 Operating System and Platform Compatibility

Solaris Nevada on both X86 and SPARC platforms will be supported.

Ethernet controllers meets the following requirements are supported:

- Has a GLDv3 driver in Solaris;
- Supports 802.3X PAUSE;
- Supports Jumbo frame;
- Supports Multiple Unicast MAC Address (However if this feature is not supported by the hardware, FCoE can still function by setting Promiscuous mode. In this case, an explicit option must be given at the fcadm command line when creating FCoE target ports).

2.4.3 Interoperability with Sun Projects/Products

FCoE Target will support Leadville FC initiators (across an FCoE switch) and Solaris FCoE Initiator (PSARC/2008/311).

2.4.4 Interoperability with External Products

FCoE Target will interoperate with any FC initiators with the existence of an FCoE switch in the SAN.

FCoE Target will interoperate with any FCoE initiators, either in back-to-back mode, or with connections to an FCoE switch.

2.4.5 Coexistence with Similar Functionality

No known overlap in functionality.

Both Emulex and QLogic CNAs provide FCoE connectivity, but so far without target capability.

2.4.6 Support for Multiple Concurrent Instances

There will be only one instance of fcoe driver. There could be multiple instances of fcoet driver.

Version mismatch between fcoe and fcoet driver will stop FCoE target ports from being created.

2.4.7 Compatibility with Earlier and Future Releases

This will be the first release of the product. All exported interfaces will maintain backward compatibility in the future.

2.5 Performance and Scalability

2.5.1 Performance Goals

When running on systems that have enough CPU power, the FCoE Target is supposed to saturate a 10G Ethernet link provided that the Ethernet controller is able to fan out to multiple tx/rx rings and load balance FCoE frames so that it can reach full line speed throughput. SCSI I/O throughput will be lower than line rate due to overheads.

2.5.2 Performance Measurement

The performance will be measured by dd, vdbench and other SCSI benchmarking tools.

2.5.3 Scalability Limits and Potential Bottlenecks

Any MAC interface in the host could be configured as an FCoE target port as long as system resources are available. Hardware limitation like CPU power, PCI-E resources, NIC RX/TX channel configuration, LUN backing stores are potential bottlenecks.

2.5.4 Static System Behavior

fcoe and fcoet driver footprints are small (no more than 500 kbytes).

FCoE port configuration file will be small, approximately 100 bytes per FCoE port.

2.5.5 Dynamic System Behavior

This is a software FCoE solution, there are code paths which are CPU intensive (e.g. FC-CRC calculation). Depending on the CPU power, when I/O traffic to FCoE target ports is heavy, CPU utilization could go very high.

2.6 Failure and Recovery

2.6.1 Resource Exhaustion

FCoE target port creation will fail when system resource is exhausted.

2.6.2 Software Failures

No planned software failures.

2.6.3 Network Failures

FCoE network failures are managed in the same way as FC SAN. Link failures between NIC port and switch are reported by MAC layer, link failures between switch and remote devices are reported through RSCN ELS to the local FCoE port. All failures will be handled accordingly by determining the nature of the change.

2.6.4 Data Integrity

FCoE data integrity is protected via 32-bit FC-CRC which is calculated and checked in fcoe driver.

2.6.5 State and Checkpointing

N/A

2.6.6 Fault Detection

TBD

2.6.7 Fault Recovery (or Cleanup after Failure)

N/A

2.7 Security

A physically secured and controlled network is mandatory for FCoE to run on. However FC-SP (Fibre Channel Security Protocol standard by T11) can be used with FCoE to enforce more security. FC-SP is intended as a secure protocol which includes authentication and encryption, but it is not widely implemented in the industry and is not implemented in Solaris yet

2.8 Software Engineering and Usability

2.8.1 Namespace Management

Common prefix for fcoe driver source is fcoe_.

Common prefix for fcoet driver source is fcoet_.

New package for fcoe driver is named "SUNWfcoe".

New package for fcoet driver is named "SUNWfcoet".

New package for libfcoe is named "SUNWfcoeu".

2.8.2 Dependencies on non-Standard System Interfaces

N/A

2.8.3 Year 2000 Compliance

N/A

2.8.4 Internationalization (I18N)

No change in existing I18L level. fcadm will follow current convention.

2.8.5 64-bit Issues

This product will be 64-bit clean.

2.8.6 Porting to other Platforms

No.

2.8.7 Accessibility

No new CLI or GUI, only additions to existing CLI.

3 Release Information

Note: Some of the packaging and installation details may not be available at design time. Describe expected solutions, and augment the description as the details are decided.

3.1 Product Packaging

This product is bundled with Solaris.

3.1.1 Package Overview

New package "SUNWfcoe"

- package for fcoe driver
- default installation root is /kernel/
- required

New package "SUNWfcoet"

- package for fcoet driver
- default installation root is /kernel/
- required

New package "SUNWfcoeu"

- package for libfcoe.so and libfcoe.h
- default installation root for libfcoe.so is /usr/lib/
- default installation root for libfcoe.h is /usr/include/
- required

3.1.2 (Default) Installation Locations

fcoe and fcoet drivers will be installed at /kernel/ directory structure.

libfcoe.so will be installed at /usr/lib/ directory structure.

libfcoe.h will be installed at /usr/include/ directory structure.

3.1.3 Effect on External Environment

SUNWfcoe and SUNWfcoeu will be shared with a future project "FCoE Initiator" PSARC/2008/310

3.2 Installation

3.2.1 Installation procedure

Standard Solaris installation, no change.

3.2.2 Effects on System Files

None.

3.2.3 Boot-Time Requirements

fcadm will be called at boot time (from SMF) to create FCoE ports.

3.2.4 Licensing

N/A

3.2.5 Upgrade

FCoE client interface is versioned. In case of version mismatch between fcoe and fcoet, creating FCoE target port will abort and an error message will be printed.

3.2.6 Software Removal

Standard package tools.

3.3 System Administration

Only CLI administration tools are supported:

- fcadm is used to create and delete FCoE target ports.
- fcinfo is used to report status of FCoE target ports.

4 FCoE Target Driver Architecture

4.1 Description

fcoet is a pseudo driver that:

- registers to fcoe driver as client of FCoE transport;
- registers to FCT as local port;
- implements FCT local port functions through FCT FCA interfaces;

- dispatches incoming FC frames to FCT routines for processing;
- implements FC exchange/sequence management.

fcoe is a pseudo driver that:

- loads on demand (when creating FCoE ports via fcadm);
- has only one instance across the system;
- registers to GLDv3 MAC layer as a MAC client;
- maintains the FCoE-enabled MAC interface table;
- provides IOCTLs for fcadm to create or delete FCoE ports and attaches/detaches fcoet drivers accordingly;
- manages WWNs for each FCoE ports;
- encapsulates FC frames to FCoE frames, de-encapsulates FC frames from FCoE frames;
- transmits/receives FCoE frames via the MAC interface;
- maintains global data structure to track registered fcoet instances.

fcoet driver supports FC HBA API. HBA attributes and port attributes are populated from fcoet driver to FCT when libsun_fc issuing FCTIO_GET_###_ATTRIBUTES commands. Port statistics are tracked by fcoet driver and sent upon FCTIO_GET_ADAPTER_PORT_STATS command issued to FCT.

4.2 Interfaces

4.2.1 User-visible

None.

4.2.2 Internal (optional for ARC review)

fcoe driver is exporting FCoE client interface for fcoet driver.

4.3 Operation

The first time fcoe driver gets loaded is when "fcadm create-fcoe-port" is issued. Upon rebooting with FCoE ports configured, service /lib/svc/method/fcoeconfig will call fcadm to create the ports at boot time, and fcoe driver will be loaded then.

fcadm will issue ioctl to fcoe driver, and fcoe driver will open the designated MAC interface, then load fcoet driver. fcoet driver registers itself to FCT and creates an FCT local port. When the initialization is done, FCT takes control of login process and a FCoE target port is accessible to initiators.

When FCoE target port is online, FCoE frames received at the MAC interface are de-encapsulated in fcoe driver, the FC frames are then passed to fcoet driver to be dispatched to different FCT functions according to their R_CTL fields. FC data sent from FCT are passed to fcoet to build valid FC frames, then encapsulated into FCoE frames and transmitted to MAC layer with minimum overhead.

After successful "stmfadm offline-target", an FCoE target port could be deleted by issuing "fcadm delete-fcoe-port -t". fcoe driver will offline fcoet and close the associated MAC interface. Both fcoet and fcoe driver could be unloaded then.

5 fcadm and fcinfo CLI

5.1 Description

Two new subcommands will be added to fcadm to create and delete FCoE target ports. fcinfo will be able to list FCoE target port attributes and link statistics.

5.2 Interfaces

5.2.1 User-visible

See proposed fcadm man page and additions for fcinfo.

5.2.2 Internal (optional for ARC review)

fcadm will open fcoe driver and send IOCTLs to fcoe driver to create and delete FCoE target ports.

5.3 Operation

fcadm will be used to create and delete FCoE target ports.

fcinfo will be used to return status of FCoE target ports.

FCoE target port data will be stored and retrieved with libscf functions. The configuration is read upon boot by fcadm to create FCoE target ports automatically.

Appendix A: Standards Supported

dpANS - Fibre Channel - Backbone - 5, T11/08-352v0

http://www.t11.org/ftp/t11/pub/fc/bb-5/08-352v0.pdf

Latest FCoE Specification and all documents are available at http://www.t11.org/fcoe.

References

R.1 Related Projects

PSARC/2007/523 COMSTAR: Common Multiprotocol SCSI Target

PSARC/2004/291 Fibre Channel HBA Port Utility

PSARC/2004/571 Nemo - a.k.a. GLD v3

PSARC/2006/357 Crossbow - Network Virtualization and Resource Management

R.2 Background Information for this Project or its Product

OpenSolaris project page

http://opensolaris.org/os/project/fcoe/

R.3 Interface Specifications

R.3.1 FCoE Client Interface

fcoe_client_t

Structure used to manage FCoE Client (fcoet or fcoei).

Format

```
typedef struct fcoe_client {
      uint32_t    ect_eport_flags;
      uint32_t    ect_max_fc_frame_size;
      uint32_t    ect_private_frame_struct_size;
      char        ect_channel_name[32];
      void        *ect_client_port_struct;
      void        (*ect_rx_frame)(fcoe_frame_t *frame);
      void        (*ect_port_event)(fcoe_port_t *eport, uint32_t event);
      void        (*ect_release_sol_frame)(fcoe_frame_t *frame);
} fcoe_client_t;
```

Fields

ect_eport_flags

Flags for FCoE port. This is a bitfiled with the following values

```
#define     EPORT_FLAG_N2N              0x01
#define     EPORT_FLAG_TGT_MODE         0x02
#define     EPORT_FLAG_INI_MODE         0x04
```

ect_max_fc_frame_size

Maximum FC frame size in bytes.

Default is 2136.

ect_private_frame_struct_size

Size of fcoet_frame_t or fcoei_frame_t. Depending on the client type (fcoet or fcoei).

ect_channel_name

Null ended MAC inteface name string.

ect_client_port_struct

Pointer to fcoet_soft_state_t or fcoei_soft_state_t. Depending on the client type (fcoet or fcoei).

ect_rx_frame

Function pointer to fcoet or fcoei. Called by fcoe when FCoE frames are received from MAC layer.

ect_port_event

Function pointer to fcoet or fcoei. Called by fcoe when port event occurs.

ect_release_sol_frame

Function pointer to fcoet or fcoei. Called by fcoe in its watchdog routine to release all solicited frames.

R.3.2 fcoe driver IOCTL

3 IOCTLs are defined in fcoe driver.

FCOEIO_CREATE_PORT

This IOCTL is used to create an FCoE port on the given MAC interface.

Input arguments: MAC interface name, FCoE port type (Initiator or Target), FCoE target port NWWN, FCoE target port PWWN.

If the NWWN and PWWN arguments are not specified, fcoe driver will generate one pair based on the MAC address of that interface.

If MAC interface is successfully opened, fcoe driver will load fcoet driver.

FCOEIO_DELETE_PORT

This IOCTL is used to delete an FCoE port from the given MAC interface.

Input argument: MAC interface name.

The precondition is that the FCoE target port be offlined by stmfadm first, otherwise an error will be returned to the userland.

If the FCoE target port has been successfully offlined, fcoe driver will offline fcoet devnode. Upon success the both fcoe and fcoet driver instance could be unloaded from memory.

FCOEIO_GET_PORT_LIST

This IOCTL is used to get an array of FCoE port attributes.

R.3.3 libfcoe interfaces

FCOE_PORT_ATTRIBUTE

Structure used to retrieve FCoE port attributes.

Format:

```
typedef struct fcoe_port_attr {
        FCOE_PORT_WWN   port_wwn;
        FCOE_UINT8      mac_link_name[32];
        FCOE_UINT8      mac_factory_addr[6];
        FCOE_UINT8      mac_current_addr[6];
        FCOE_UINT8      port_type;
        FCOE_UINT32     mtu_size;
} FCOE_PORT_ATTRIBUTE;
```

FCOE_CreatePort

Function used to create FCoE ports.

Format:

```
FCOE_STATUS
FCOE_CreatePort(
      const FCOE_INT8   *macLinkName,
      FCOE_UINT8        portType,
      FCOE_PORT_WWN     pwwn,
      FCOE_PORT_WWN     nwwn,
      FCOE_UINT8        promiscuous
);
```

FCOE_DeletePort

Function used to delete FCoE ports.

Format:

```
FCOE_STATUS
FCOE_DeletePort(
      const FCOE_INT8 *macLinkName
);
```

FCOE_GetPortList

Function used to retrieve the list of FCoE port attributes.

The actual number of FCoE ports and a pointer to an array of fcoe_port_attr_t will be returned upon success. The caller is responsible to free the memory block pointed by *portlist.

Format:

```
FCOE_STATUS
FCOE_GetPortList(
      FCOE_UINT32            *port_num,
      FCOE_PORT_ATTRIBUTE    **portlist
);
```

R.3.4 FCoE SMF Service

Content of /lib/svc/method/fcoeconfig

```
/usr/sbin/fcadm create-fcoe-ports
```

Content of fcoe_config.xml TBD

R.3.5 Suggested manpage change for fcadm

See fcadm.man.

R.3.6 COMSTAR FCT FCA Programming Guide

See writing_fca.txt.