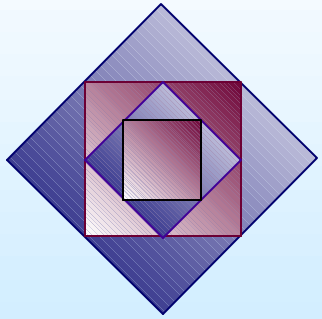
iSCSI – protocol & implementation

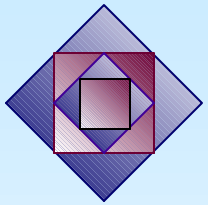
CERN – September 2000

Julian Satran

Kalman Meth

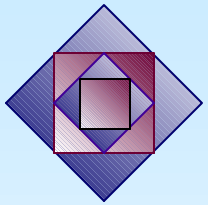
IBM Research Lab in Haifa





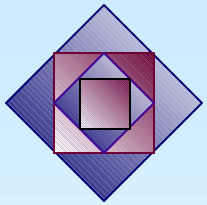
Why?

- Standard networking infrastructure is good enough for storage
- Leverage existing networking infrastructure
  - equipment
  - management
  - skills



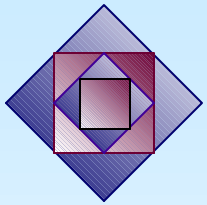
Objective

- Performance
  - interconnect will meet or exceed current storage needs and enable growth
  - high bandwidth, minimum latency
- Availability
  - enable various levels of recovery
- Cost
  - reuse whatever available
- A good network citizen
  - congestion control
  - security



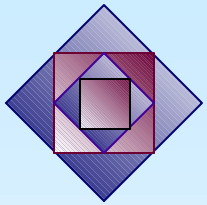
Design guidelines

- Minimalist design
- Enable efficient implementations
- No or minimal options
- Features can be ignored without affecting interoperability – no negotiation or setting beyond that mandated by SCSI for basic functions
- Clear layering of functions
  - SCSI
  - iSCSI
  - transport/delivery network



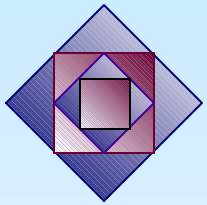
Basic mechanisms

- Sessions and connections
- Login, authentication and security
- Commands, messages, tasks and tags
- Ordered delivery – the numbering scheme
- The response numbering scheme
- Recovery
- Each of the basic mechanisms has a minimal functionality that must be implemented



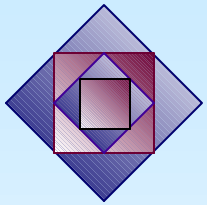
Sessions and connections

- A session is a set of TCP connections linking an initiator and a target
- It is long lived
- It is meant to provide bandwidth and availability
  - several connections = more bandwidth
  - several connections = better availability provided fail over is enabled
- The initiator is supposed to be able to use any connection to execute a command and keep all the command related packets on the connection it started the command (connection allegiance)
- *Minimum requirement – 1 TCP connection/session*



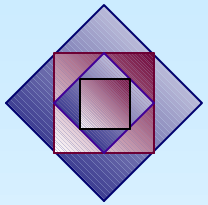
Login, authentication and security

- Login has multiple functions:
  - session building
  - authentication and security
  - some parameter negotiations
- Authentication and security
  - none
  - authentication only
  - authentication & encryption
  - *minimal implementation – session building*



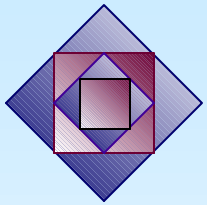
Commands, messages, tasks and tags

- Commands & Responses are mappings of SCSI and carry an initiator-wide unique tag
- The initiator tag helps relate all the elements of a command (command, optional data and response)
- Messages are used to:
  - manipulate tasks or check/set the whole SCSI/iSCSI path (in which case they carry also a tag)
  - check only the iSCSI transport in which case they don't carry a tag
- *A compliant implementation must support all command and message types but can choose to ignore parameters as outlined in the draft or reject them*



3 stages in the life of a SCSI command

- Initiation – delivery from initiator SCSI layer to target SCSI layer
- Execution – optional data transfer and RTT
  - to keep latency at a minimum iSCSI permits unsolicited data transmission as immediate data (attached to the command) or in separate packets
  - *to enable recovery iSCSI mandates honoring target issued RTT*
- Status transmission from target SCSI to initiator SCSI *(status recovery is enabled but not mandated)*

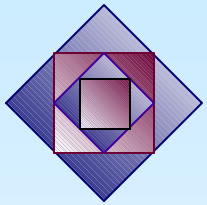


Ordered delivery – the numbering scheme

- iSCSI enables ordered delivery of commands and tagged messages from initiator to target over several distinct TCP connections
- The model used by a target that chooses to implement ordered execution is that of an iSCSI staging-area from which command are delivered to a SCSI device-server only after all preceding commands have been delivered
- A sliding window mechanism is used to keep the staging-area within bounds

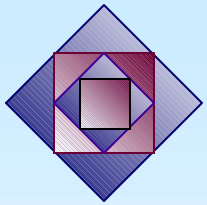
Ordered delivery – the numbering scheme (cont.)

- The command numbers are significant only while commands are within the staged area
- The only unique identifier for the life of a command is the initiator tag
- *Ordered delivery is not mandatory – although initiators supporting several connections per session should implement it*
- *Targets supporting several connections per session should indicate the support/lack-of-support for ordered delivery (how?)*



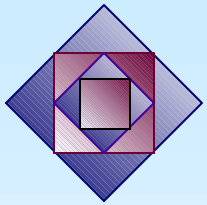
Alternatives considered

- An asymmetric scheme
- 1 control connection + several data connections
- The command connection can fail over on another connection



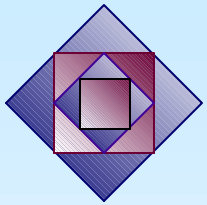
The response numbering scheme

- Responses are also subject to numbering
- The main purpose of response numbering is bulk response acknowledgement
- Response acknowledgement enables a target to discard whatever residual information it has about a task after the response is acked
- *Response numbering and acknowledgement is mandatory*



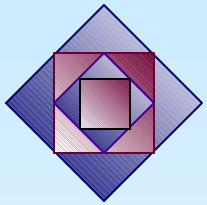
Recovery

- Several levels of recovery are enabled by iSCSI:
  - error notification – broken connections lead to SCSI command failure
  - command restart – commands pending or in progress on a broken connection can be restarted on a new or one of the remaining connections in a session
  - refinements considered:
    - data transmission restart – commands can be restarted from a known point in the data transfer (mainly important for long operations)



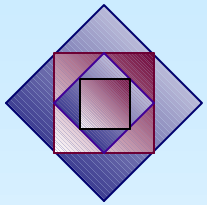
Recovery (cont.)

- failure to handle data and deadlock avoidance – data can be dropped by targets and reacquired by RTT
- the design aim is to enable a session to stay operational as long as a single TCP link can be maintained/established
- *The mandatory recovery support is error notification and data replay on request (RTT)*



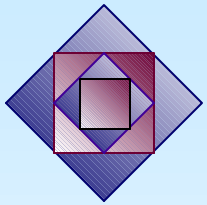
Additional mechanisms

- Text commands
  - enable parameter negotiations and vendor unique extensions
- Mapping
  - an aliasing mechanism for string mapping into 8 byte “normal” SCSI addresses
  - to be used for third party naming and access control
- *Text commands and mapping may be rejected(not implemented)*



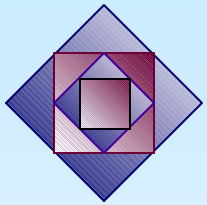
Under construction

- Login / authentication / security
  - Security context defined during a login “phase” (delimited by a login request-response pair but containing several text commands)
- Text parameters
- Host requirements
  - APIs
  - Timers
  - Tag structure
- Rationale/explanations/implementer notes/state diagrams
- RDMA/Synch Recovery



Linux initiator driver

- Follows the general structure of a Linux SCSI driver
  - High-Level SCSI driver (Class Driver)
  - Mid-Level SCSI Driver (Port Driver)
  - New Low-Level SCSI Driver (Mini-Port Driver)
    - HBA specific
    - iSCSI – uses kernel sockets for TCP
- Built as a loadable module



Linux initiator driver (cont.)

- At install time:
  - Reads a configuration file
  - Attempts to establish connections to targets and login
  - If it fails the buses are available but the devices are not
  - Will reattempt connection at SCSI command inception