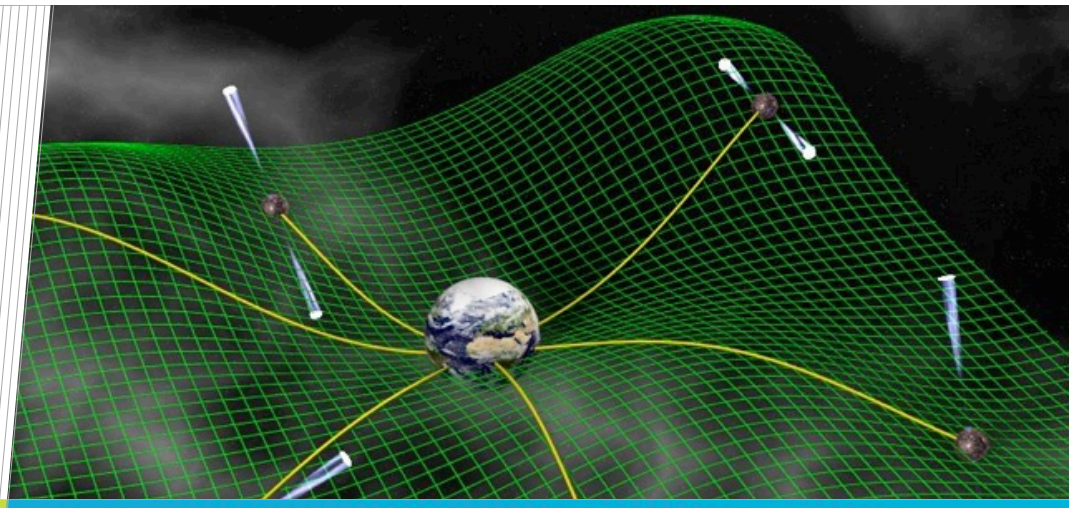


www.csiro.au

Moving Large Data Sets Around

Taking Saratoga from space-based ground sensors to ground based space sensors

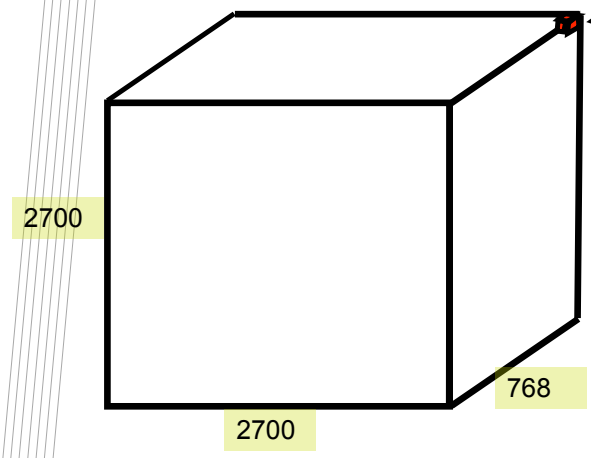
Charles Smith
Network Consulting Engineer
9 September 2010

CSIRO

The need for a high performance file/stream transport protocol in Science

Some Dataset Size Examples from Radio Astronomy

Data Image Cubes for the Murchison Wide Field Array



Element
4 Polarizations, 1 Weight
@ 4 bytes each

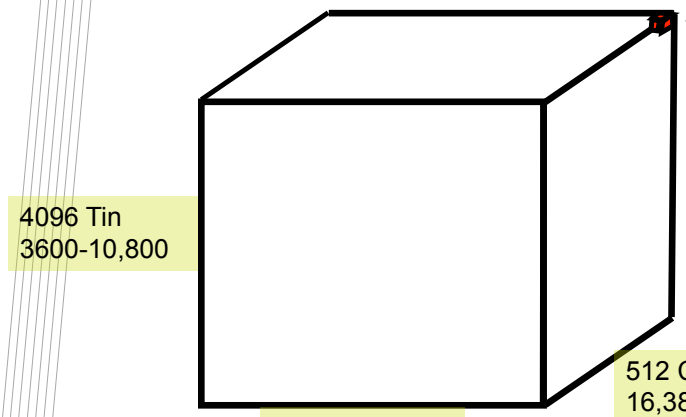
112 Gigabytes Data Cube

Continuous Sampling

One produced every 12 minutes.

16 Terabytes / day
5.9 Petabytes / Year

Data Image Cubes for ASKAP Surveys



Element
4 Polarizations, 1 Weight
@ 4 bytes each

53 Gigabytes to 71 Terabytes / Cube
Wallaby All Sky Survey 1000 Cubes - 3.4 Petabytes
Dingo Deep Focus 2 x 2,500 hours - 50 cubes
Dingo Deep Focus 5 x 500 hours - 250 cubes
Total 2.6 Petabytes

Transport using a 10 Gigabit Carrier Interface

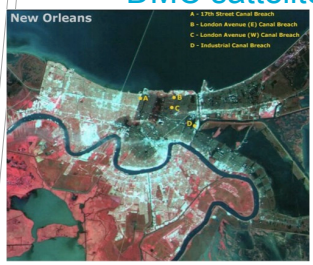
- Given a 10 Gigabit Ethernet Connection at theoretical line rate
  - 1 Terabyte files takes ~14 minutes
  - Single ASKAP 3.4 Terabyte Data Cube Image takes ~46 minutes
  - 71 Terebyte Visibility Data Cube Image takes ~15 hours
- Currently a TCP based single streaming transport cannot fill a 10 Gigabit Interface to capacity due to:
  - Aggresive congestion controls on TCP flows forcing the interface into “slow-start” mode when the link has saturated.
  - Sychronous frame acknowledgement mechanisms to ensure data reliability.
  - Inefficient TCP stacks on general purpose compute platforms not able to operate at 10 Gigabit line rates.
- A single stream asynchronous high speed file/stream transport mechanism is needed.
  - Today high performance file transport is being deployed by custom development in separate Scientific communities using simple UDP based mechanisms. e.g. eVLA, LOFAR, VLBI & ASKAP.

Saratoga

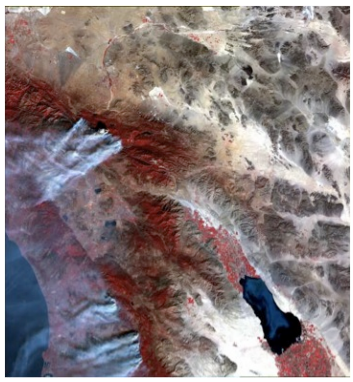
A reliable asymetric file,stream,bundle transport protocol with UDP or UDP-Lite

Saratoga - Short History

- Developed originally by Surrey Satellite Technlogy Ltd (SSTL) to transfer remote sensing imagery from IP-based LEO stellites.
- NASA Glenn cleaned up the Saratoga design to create a new version of Saratoga for file or bundle transfers.
- Deployed on the UK Disaster Monitor Constrellation Satellites (DMC) on RTEMS platform for image transfer to ground stations.
- e.g. Images of fires in Southern California and Hurricane Catrina aftermath in New Orleans delivered by Saratoga from DMC sattelites.



Saratoga is in daily operational use.



How Saratoga is used in operations

- Each DMC satellite has multiple onboard computers (OBC) for image capture & packetised transmission.
  - The Solid State Data Recorders (SSDR's) are interconnected by IP over a 8.1Mbps serial link for data and CANbus for backup control.
  - Newer satellites have a 20/40 Mb/s X-band downlink for hi-res cameras.
  - The uplink is only 9600b/s for command and control.
- The system is very asymmetric 850:1 or worse.
- As much data as possible must be transferred during a pass over a ground station.
  - Passes may be up to 12 minutes, depending on elevation.
  - At 8Mb/s ~650MB of useful data can be transmitted in a high pass.
- Link utilisation really matters!

What is Saratoga

- A high speed UDP based peer-to-peer protocol providing error-free gaurenteed delivery of files, bundles or streams of data.
- Designed for use in asymmetrous network topologies.
  - Flood data packets out as fast as you can.
    - No specified congestion control is required, since data is usualy going one hop over a private link, or across high speed low congestion private links.
    - Multiplexing of flows is via a Saratoga peer EID for each transaction.
  - No timers are specified in the protocol definition.
    - Ideal for very long propogation delay networks and was originally used for communications from Satellites to Earth Stations.
    - Intelligent sender rate or congestion control can be provided as exact timing of each frame can be optionally included in the frame header.
  - The transmitter requests acknowlegement from the receiver on its discretion.
    - The receiver can also send acknowledgements it needs to start/stop/ restart or finish a transfer.
    - Acks are Selective Negative Acknowledements (SNACKs). They indicate the received offset & any gaps to fill.

Why not just use TCP

- TCP has inbuilt assumptions about loss,congestion competition that do not need to be applied on dedicated links.
  - We have a known pipe, lets just fill it...
- It performs very poorly with high link asymmetry (>850:1)
  - This applies to most sensor networks.
- TCP is really for a conversation between two hosts. Saratoga needs to just transfer lots of data from one host/ sensor to another,
  - The transmitter needs to be as simple as possible.
- Long delay use - Eventually TCP will fail to open a connection because of its SYN/ACK exchange won't complete.
  - There is too much conversation going on.
- TCP has many unwanted timers.

A reliable transport over UDP or UDP-Lite

- The normal UDP checksum covers both the header and payload. It is consistent but not that strong (one's compliment), and does not provide end-to-end gaurentees.
- An end-to-end optional checksum using a choice of CRC32, MD5 or SHA-1 over the entire file being transferred increases the confidence that a reliable copy has been made and that fragments have been re-assembled correctly.
- Operation of the HOLESTOFILL mechanism
  - The HOLESTOFILL list on the receiver requests the transmitter to re-send frames that have not been fully received (a SNACK).
  - The transmit window does not move untill the holes have been acknowledged by a HOLESTOFILL frame with an advanced offset.
  - The receive window only advances when offsets are contiguous. Yes you need lots of memory but it is cheap today.
  - Status messages within the HOLESTOFILL header containing error causes or start/stop are used to enforce a flow control capability. Aggresiveness of flow control is under user control.

Accomodation for Large Files

- For the DMC satellites imaging files are up to a few gigabytes at 32m resolution and larger for newer cameras. As shown Astronomy data cubes are huge - terabytes and up. Bundles will also be large - gigabytes and up.
- Ad-hoc sensor nets on the other hand need to transfer small files or bundles. Using a single worst case file size limits the use and performance of a file transport mechanism.
- Saratoga allows a range of file-descriptor pointers to be advertised.
  - 16/32/64 & 128-bit file descriptors are supported.
  - If a file is less than 64kB, a 16 bit offset is used. If the file is larger but less than 4GB, use a 32-bit offset ...
  - The 16-bit offset is always supported. Others are optional.

Optional Features

- A link-local multicast can be used to advertise presence and capabilities of Saratoga peers.
- Data can be sent to a local multicast address for peers to receive.
  - This is ideal for the simultaneous efficient distribution of file updates across a group of hosts.
- Streaming of data is supported.
  - Allows for real-time delivery outside of the file-based paradigm.
  - DTN Bundle delivery is also supportred and Saratoga can be used as a bundle convergence layer.
  - Given reliable delivery in streaming mode Saratoga could be used as a transport for higher level protocols such as HTTP.
- Offers a far more network efficient and reliable solution to TFTP for system remote booting or updates.
  - When coupled with its multicast capability, distribution of a software update across multiple endpoints could be a single transaction.
  - Advanced checksums ensure the file integrity.

What Saratoga does not do

- There is no MTU discovery mechanism.
  - You have to know the maximum packet size your network can transmit.
  - This is OK for private networks but will be troublesome if used across the Internet.
- There is no such thing as “slow-start”.
  - Saratoga uses UDP which just blasts away at a link with no regard to other flows.
  - This is considered bad and unsociable behavior on the Internet and UDP may be restricted for this reason.
- Memory utilisation can be very high
  - Care has to be taken on high-loss links as this could lead to large amounts of memory being used for transaction buffers.
  - Tuning of Saratoga transport parameters will be required over dedicated links to optimise performance and memory.

Saratoga

Inside the Saratoga Transaction

Saratoga Transactions

get	Get a named file from a peer
getdir	Get a directory listing of files from a peer
delete	Delete a named file from a peer, or multicast group peers
put	Send a named file or stream data to a peer, or multicast group peers
putdir	Put a directory listing of local files to a peer, or multicast group peers

Saratoga Frame Types - BEACON

BEACON

Sent periodically.

Describes the *Saratoga* peer Identity (e.g. EID)
capability/desire to send/receive packets.
max. file descriptor size: 16/32/64/128-bit.

Saratoga Frame Types - REQUEST

REQUEST

Initiates a _get_ transaction

get file
get directory listing
delete file

Saratoga Frame Types - METADATA

METADATA

Sent at start of transaction. Initiates a _put_ transaction.

Describes the file, bundle or stream:
set identity for transaction
file name/details, including size.
set descriptor size offsets to be used for this transaction
(16/32/64/128-bit pointer sizes.)

Saratoga Frame Types - DATA

DATA

Sent throughout transaction

Holds Actual Data.
Uses descriptor of chosen size to indicate offset for data segment in the file/bundle or stream.
May request an ‘ack’ (send me a holestofill).

Saratoga Frame Types - HOLESTOFILL

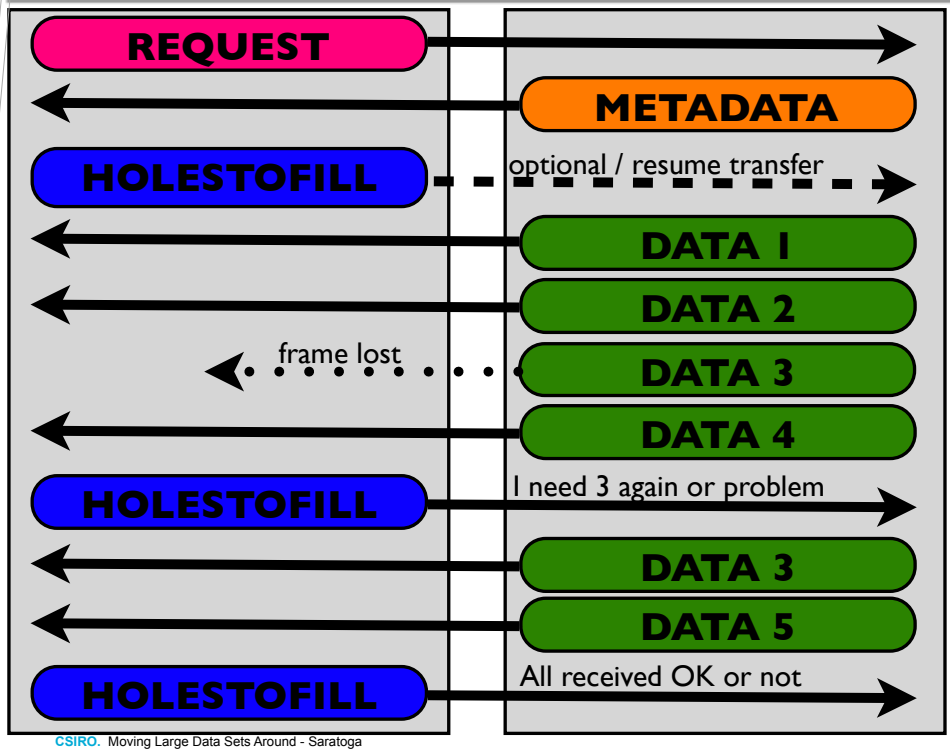
HOLESTOFILL

Sent periodically and at end of transaction.

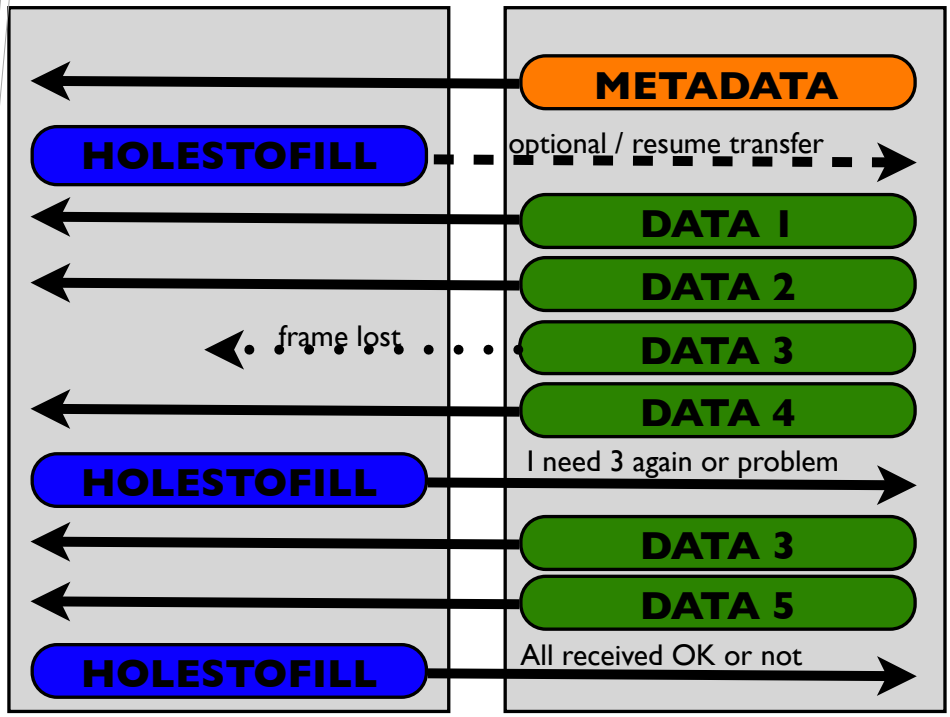
Missing Data Offsets / Error & Status Messages
Selective negative ack ('snack').

Set left window edge for successful transfer so far
List of offsets and lengths indicate missing 'holes' in data.

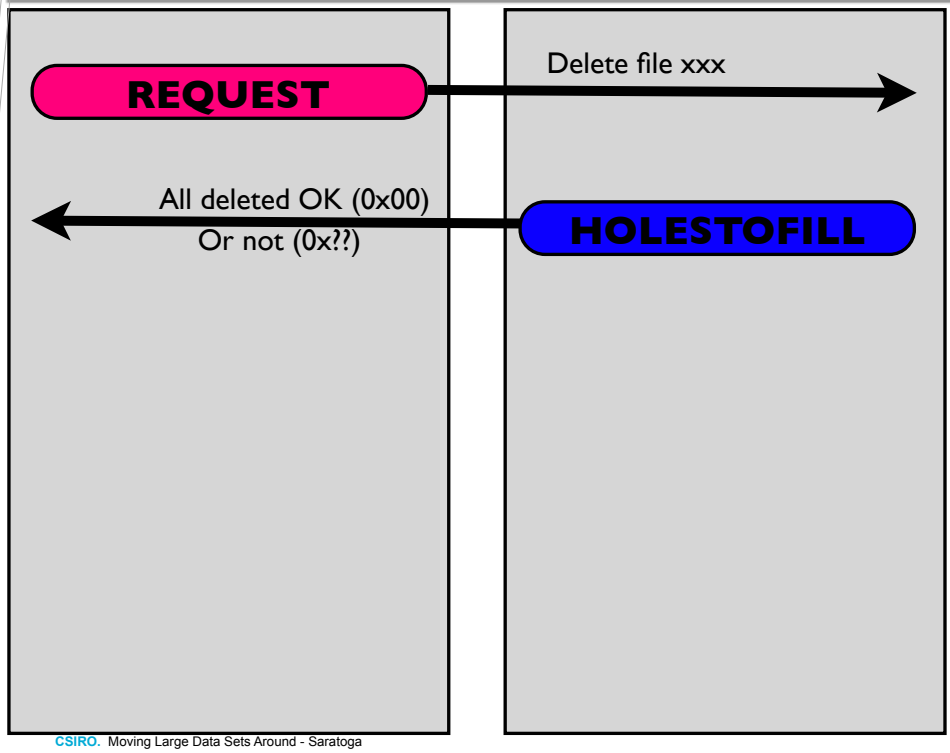
Saratoga Transactions - _get_, _getdir_



Saratoga Transactions - _put_, _putdir_



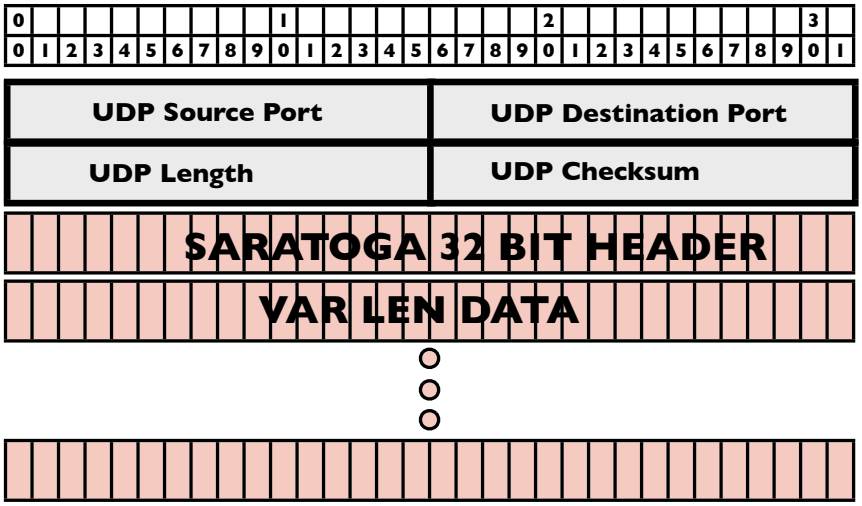
Saratoga Transactions - _delete_



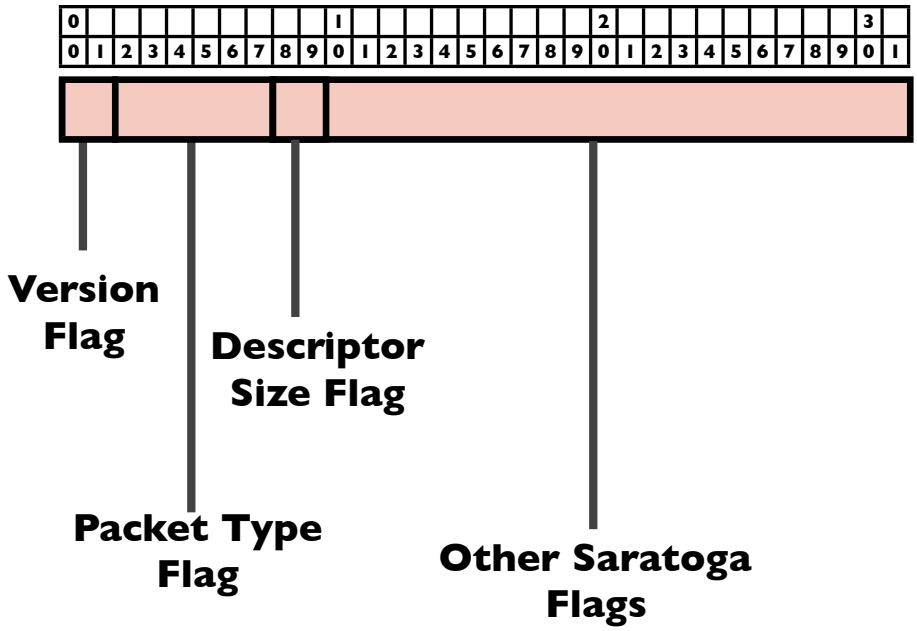
Saratoga

Inside the Saratoga Frame

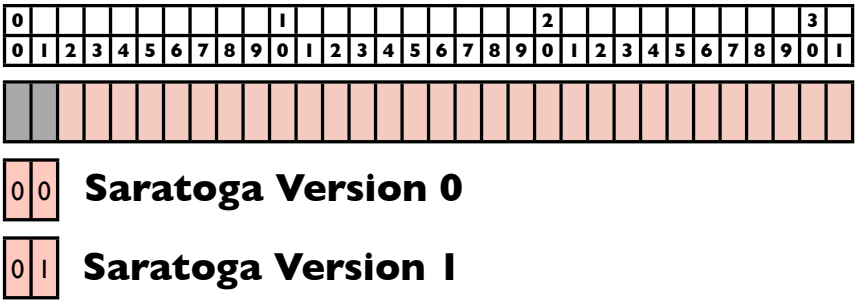
Saratoga UDP Frame Format



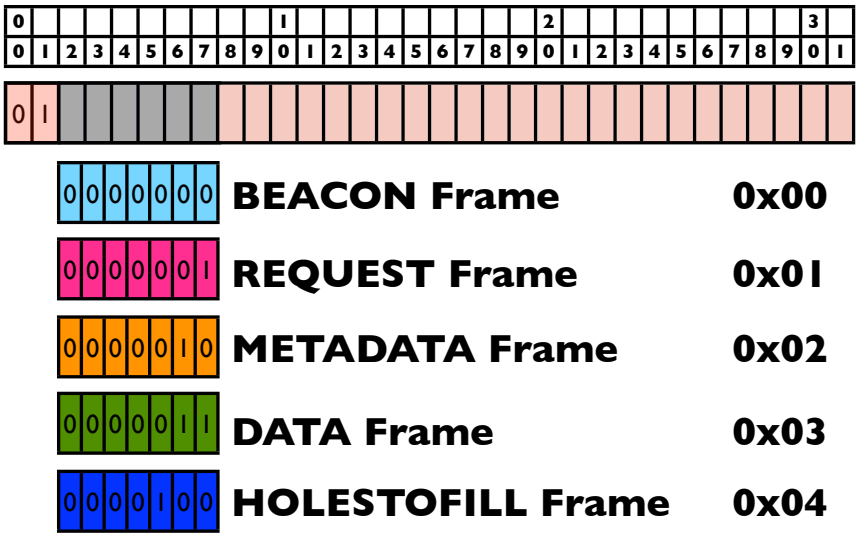
Saratoga - 32BIT HEADER



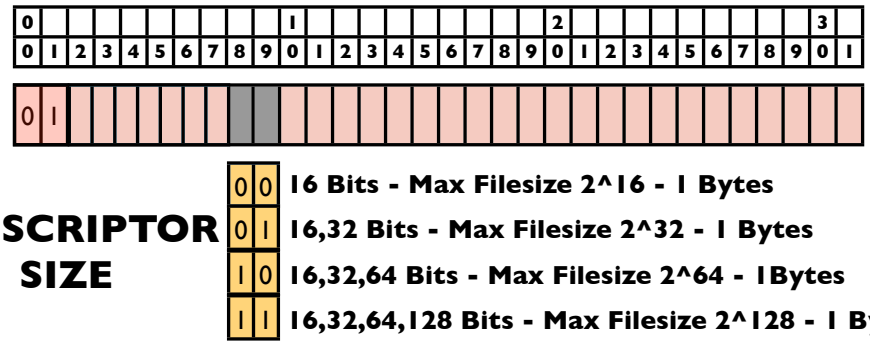
Header - Version



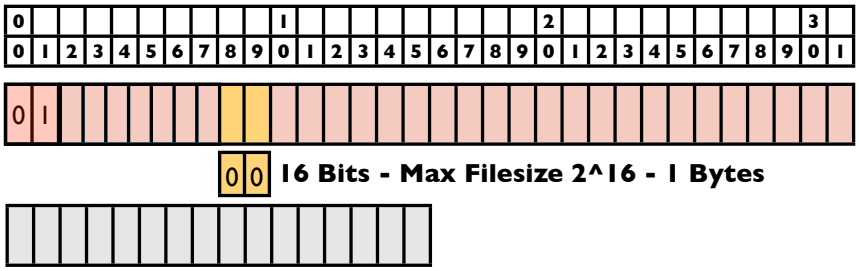
Header - Packet Type



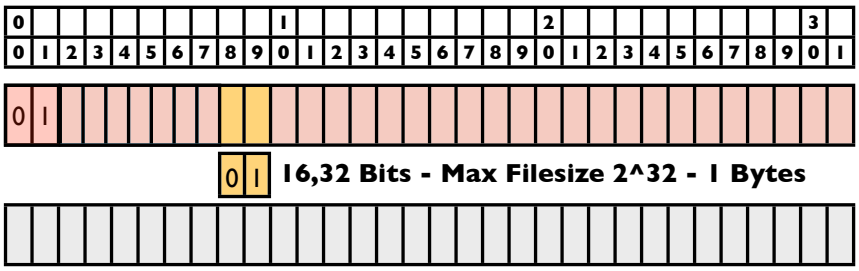
Header - Descriptor Size



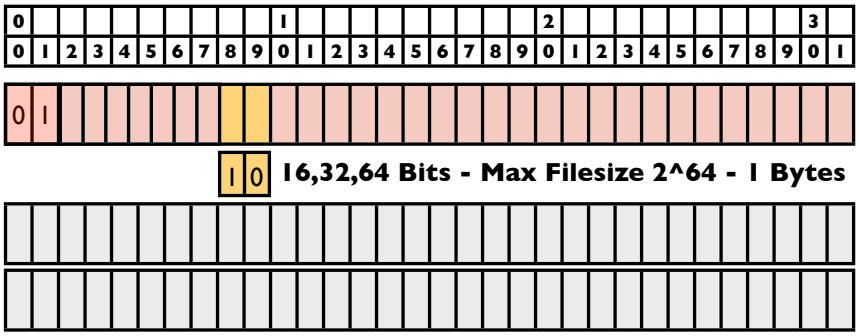
Header - Descriptor Size - 16 bit



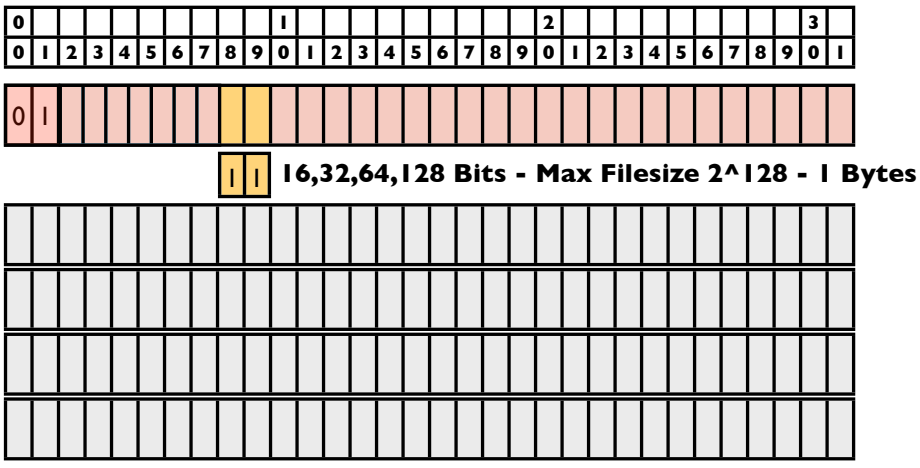
Header - Descriptor Size - 32 bit



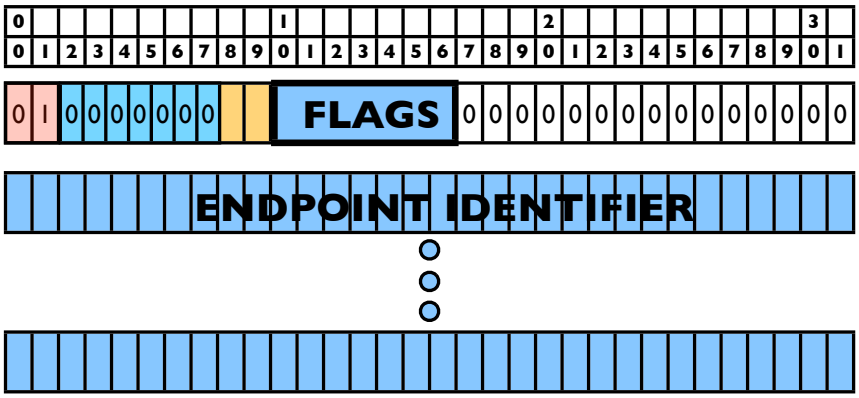
Header - Descriptor Size - 64 bit



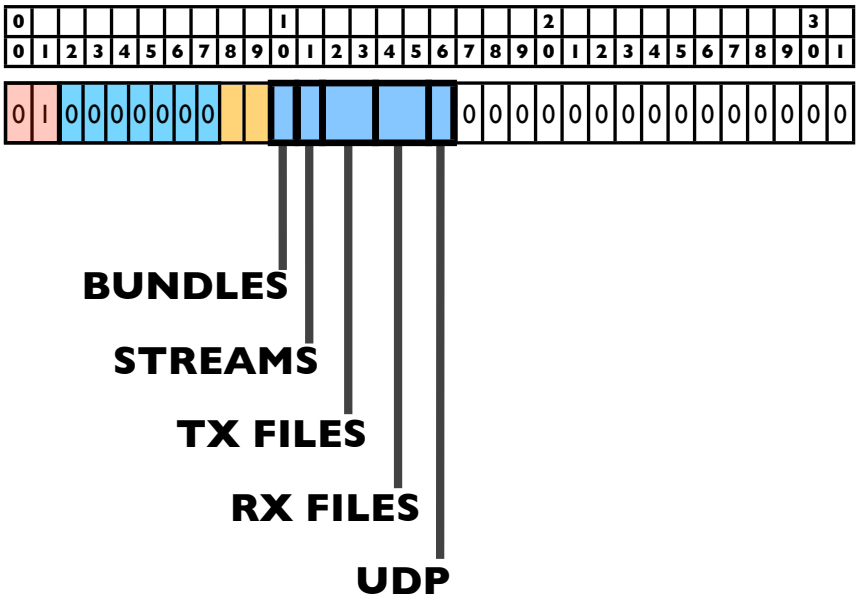
Header - Descriptor Size - 128 bit



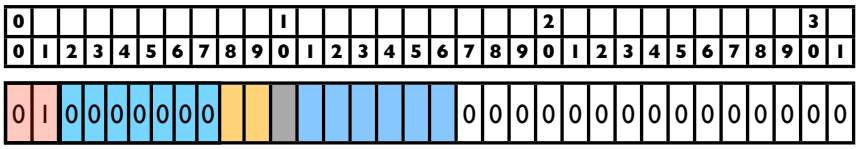
BEACON FRAME



BEACON HEADER FLAGS



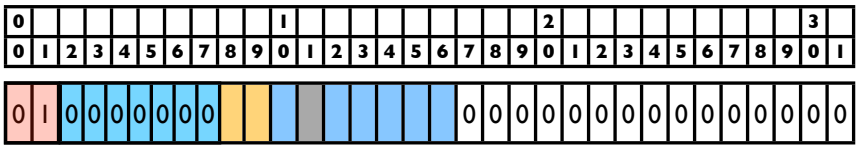
BEACON HEADER FLAGS - BUNDLES



BUNDLES

0	**No Bundles supported, handles files only**
1	**Pass marked bundles to local agent**

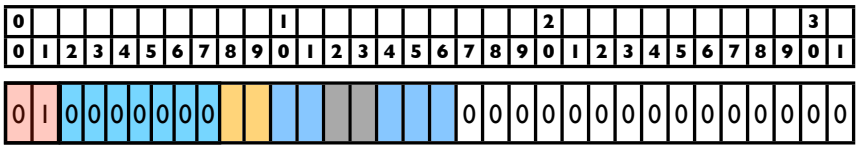
BEACON HEADER FLAGS - STREAMS



STREAMS

0	**No streaming supported**
1	**Streaming supported**

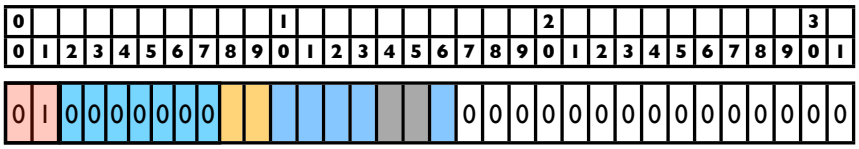
BEACON HEADER FLAGS - TX FILES



TX FILES

0 0	Cannot Send Files
0 1	Invalid
1 0	Can Send but not right now
1 1	Can Send Now

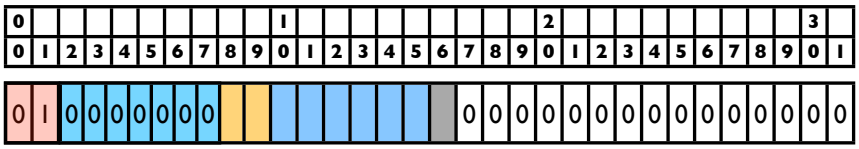
BEACON HEADER FLAGS - RX FILES



RX FILES

0 0	Cannot Receive Files
0 1	Invalid
1 0	Can Receive but not right now
1 1	Can Receive Now

BEACON HEADER FLAGS - UDP

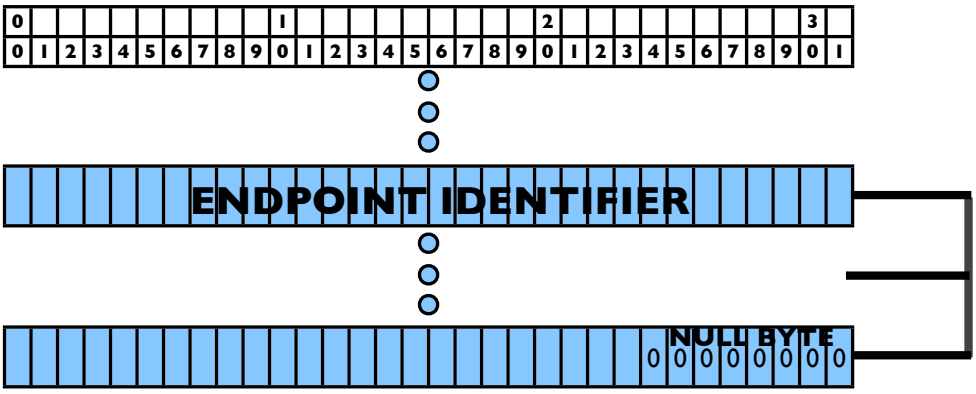


UDP

0 **Supports UDP transfers only**

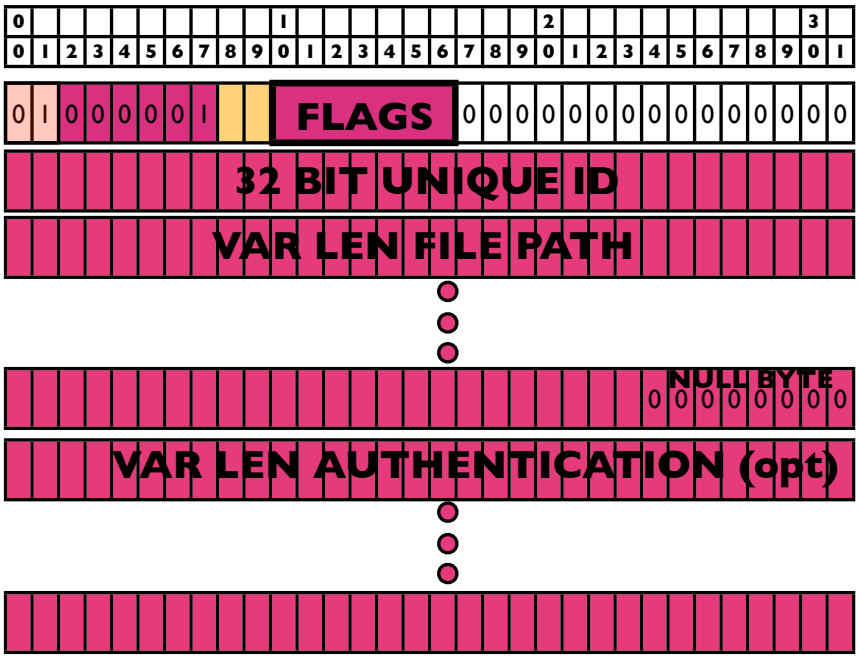
1 **Supports UDP & UPD-Lite Transfers**

BEACON ENDPOINT IDENTIFIER

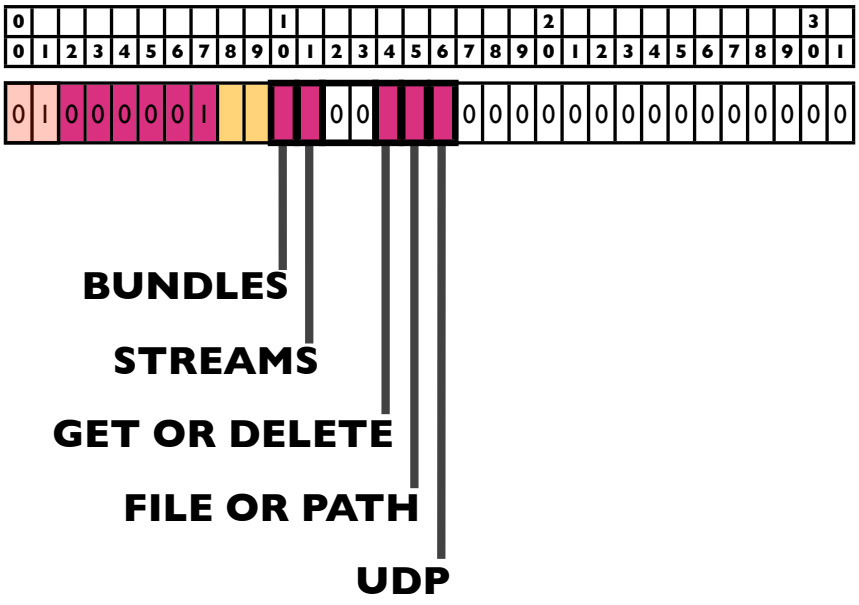


- The endpoint identifier is a unique null terminated ascii string describing the source peer and session.
- An example would be the IP address or fully qualified domain name of the host coupled with the socket number and current process ID.
- If Saratoga is being used as a bundle agent then a Bundle Endpoind ID (EID) can be used.
- The descriptor size bits (8 & 9) identify the maximum supported descriptor size 16,32,64 or 128 bit file size the endpoint supports.

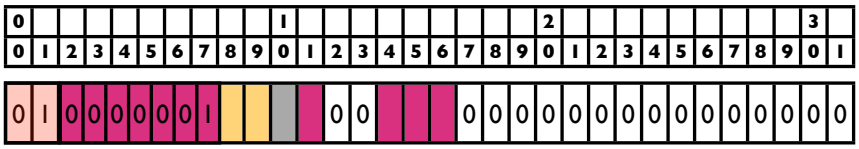
REQUEST FRAME



REQUEST HEADER FLAGS



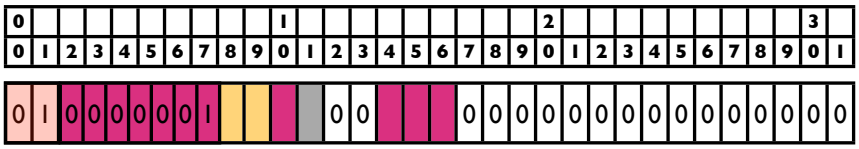
REQUEST HEADER FLAGS - BUNDLES



BUNDLES

0	**No Bundles supported, handles files only**
1	**Pass marked bundles to local agent**

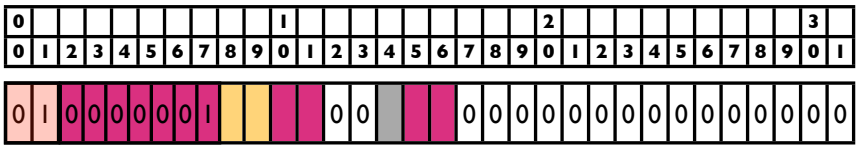
REQUEST HEADER FLAGS - STREAMS



STREAMS

0	**No streaming supported**
1	**Streaming supported**

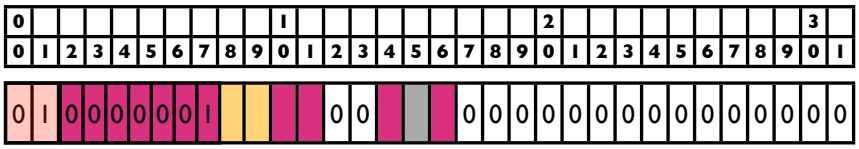
REQUEST HEADER FLAGS - GET OR DELETE



GET OR DELETE

0	_get_ or _getdir_ requested
1	_delete_ requested

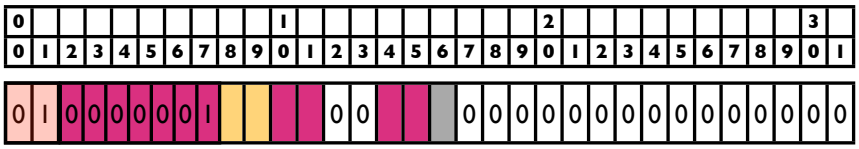
REQUEST HEADER FLAGS - FILE OR PATH



FILE OR PATH

0	Is a File for _get_ or _delete_
1	Is a Directory for _get_

REQUEST HEADER FLAGS - UDP

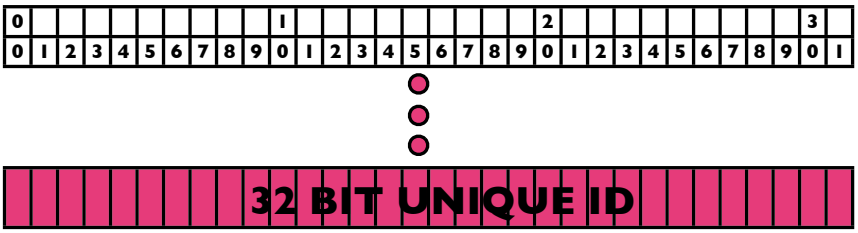


UDP

0 **Supports UDP transfers only**

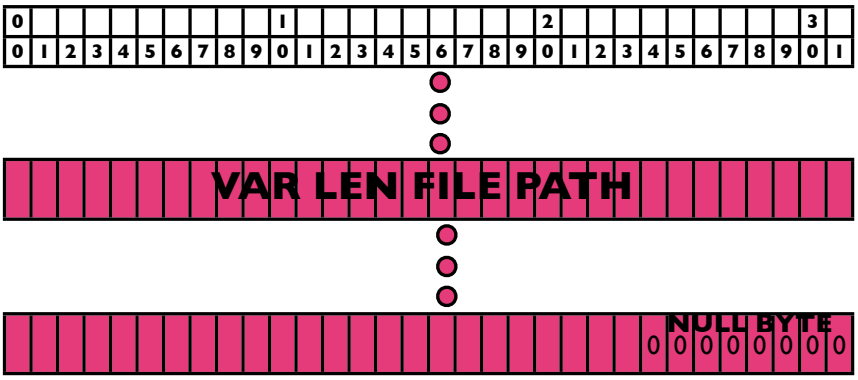
1 **Supports UDP & UPD-Lite Transfers**

REQUEST - UNIQUE ID



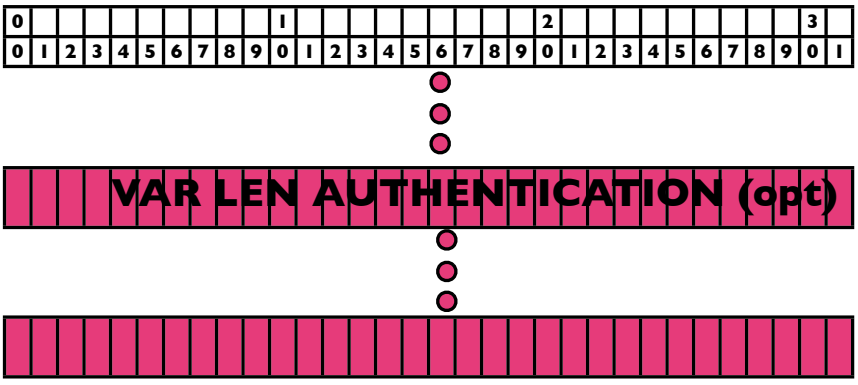
- The unique ID is a 32 bit field used during a transaction to uniquely associate a packet with a particular transaction.
- This allows multiple simultaneous transfers to be identified between peers.
- The ID is selected by the initiator of the transaction, it is unique.
- It could be generated by the properties of the file associated with a unique incrementing transaction counter on the initiating system.

REQUEST - VAR LEN FILE PATH



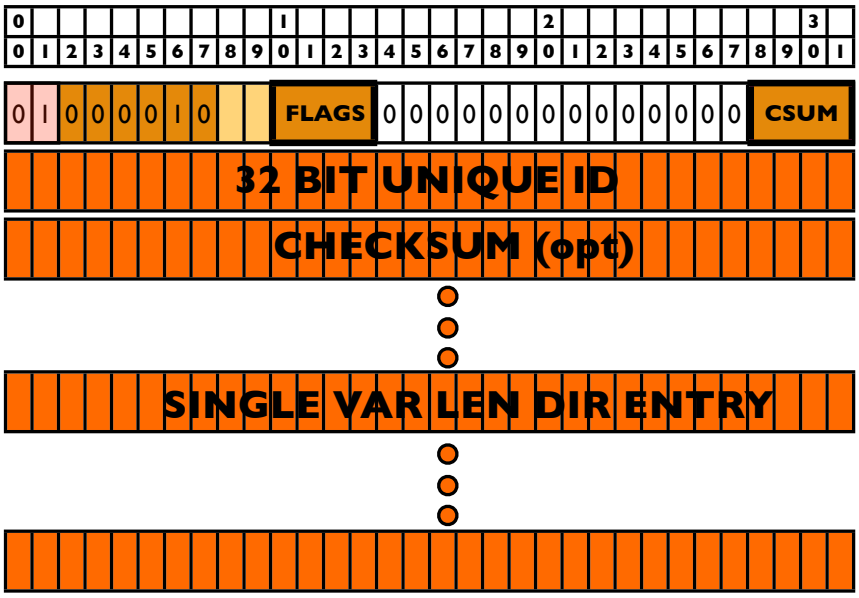
- The File Path is a variable length null terminated string of UTF-8 characters which identifies the path and base name of the file being requested for transfer.
- The maximum length of the file path field is 1024 bytes.
- If the file path is NULL then this indicates that the file reciever is willing to accept any file the file-sender would like to transmit.
- This field is mandatory for and must not be NULL for _delete_ transactions.

REQUEST - VAR LEN AUTHENTICATION

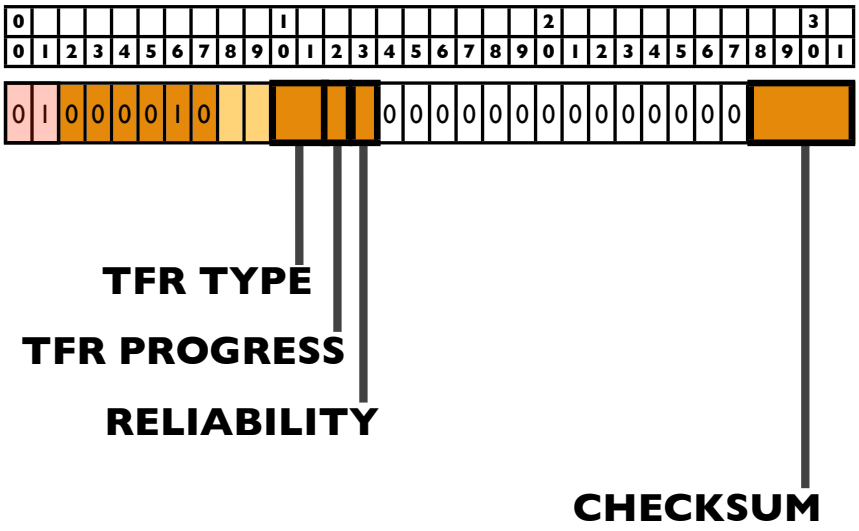


- The optional Authentication is a variable length field computed from the packet contents.
- The calculation should include contents of the IP/UDP header, Saratoga 32 bit Header, 32 bit Unique ID and File Path.
- It is used to validate a request and its origin upon receipt at a peer.

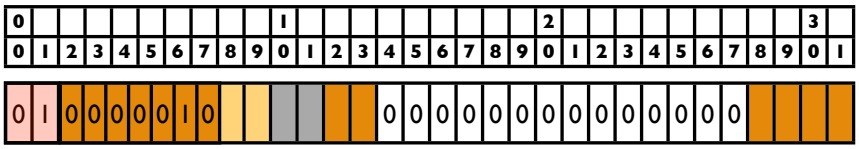
METADATA FRAME



METADATA HEADER FLAGS



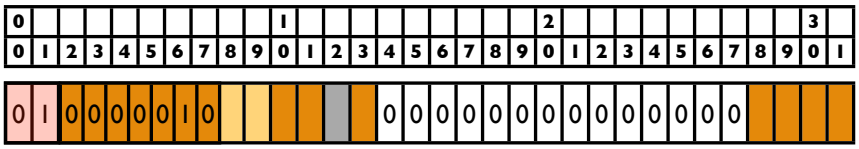
METADATA HEADER FLAGS - TFR TYPE



TFR TYPE

0 0	**Transfer a File**
0 1	**Transfer a Directory**
1 0	**Transfer a Bundle**
1 1	**Transfer an Indefinite Length Stream**

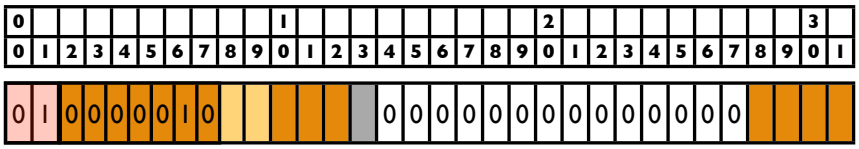
METADATA HEADER FLAGS - TFR PROGRESS



TFR PROGRESS

0	**Transfer in progress**
1	**No longer in progress and terminated**

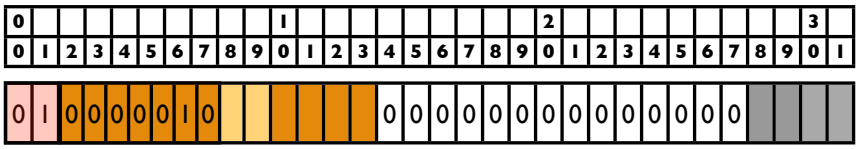
METADATA HEADER FLAGS - RELIABILITY



RELIABILITY

0	Must be delivered reliably by UDP
1	May be delivered unreliably by UDP-Lite

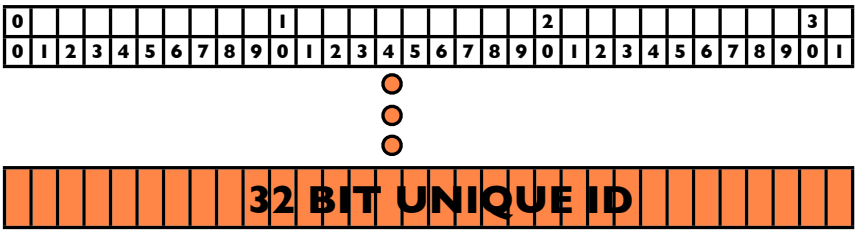
METADATA HEADER FLAGS - CHECKSUM



CHECKSUM

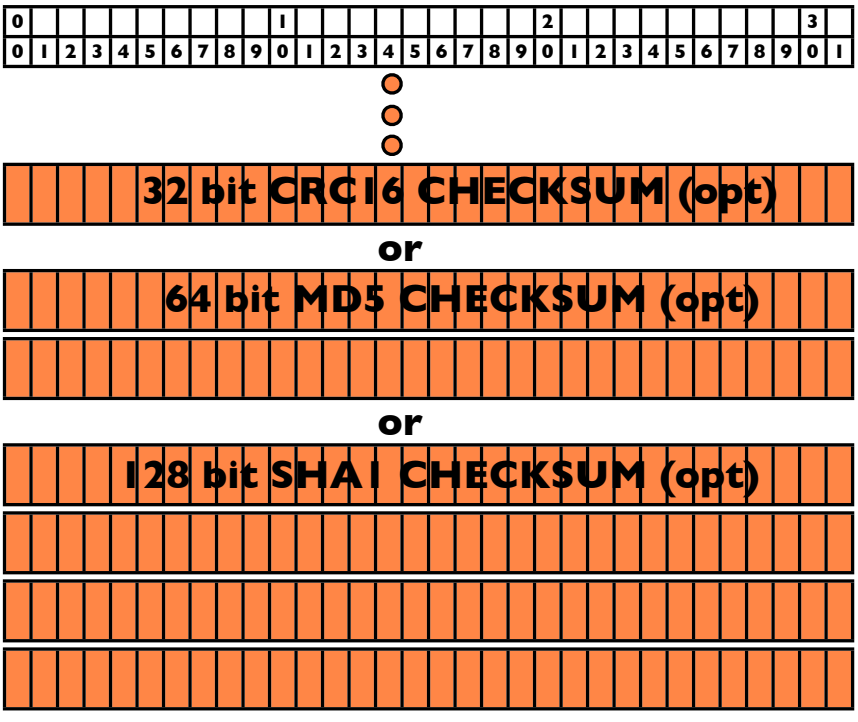
No Checksum	0 0 0 0
32 bit CRC32	0 0 0 1
128 bit MD5	0 0 1 0
160 bit SHA-1	0 0 1 1

METADATA FRAME - UNIQUE ID



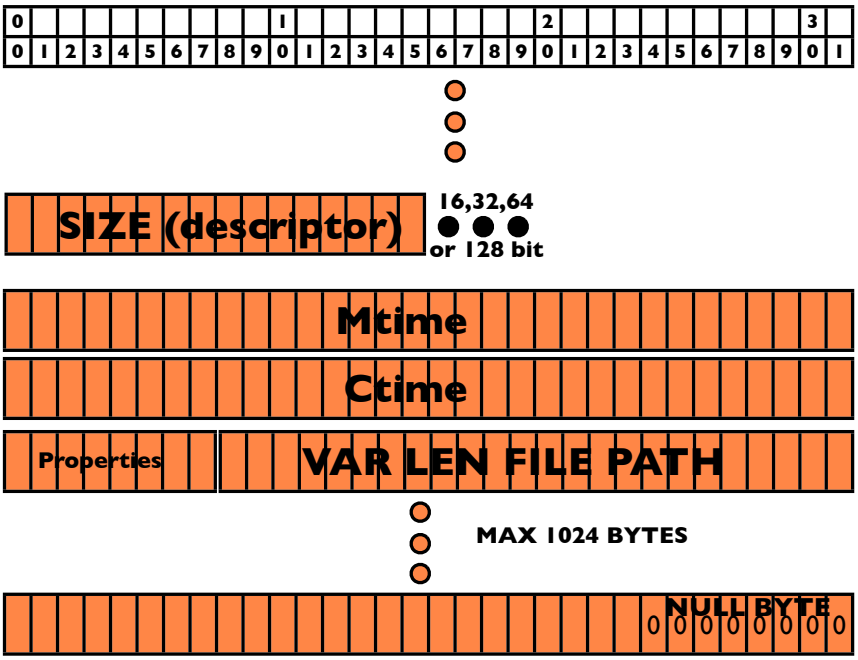
- The unique ID is a 32 bit field used during a transaction to uniquely associate a packet with a particular transaction.
- It is to match the transaction identified by a previously received REQUEST frame for a _get_ or _getdir_ transaction.

METADATA FRAME - CHECKSUM



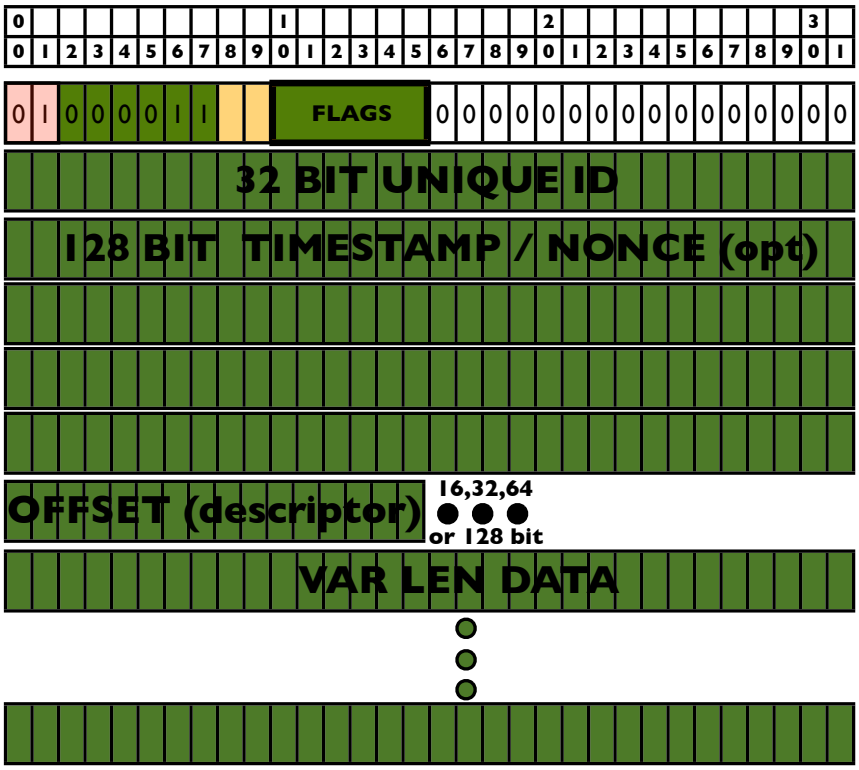
- The optional checksum is the the checksum of the file to be transferred. It is validated upon receipt. A HOLESTOFILL error status is returned if invalid.

METADATA FRAME - DIR ENTRY

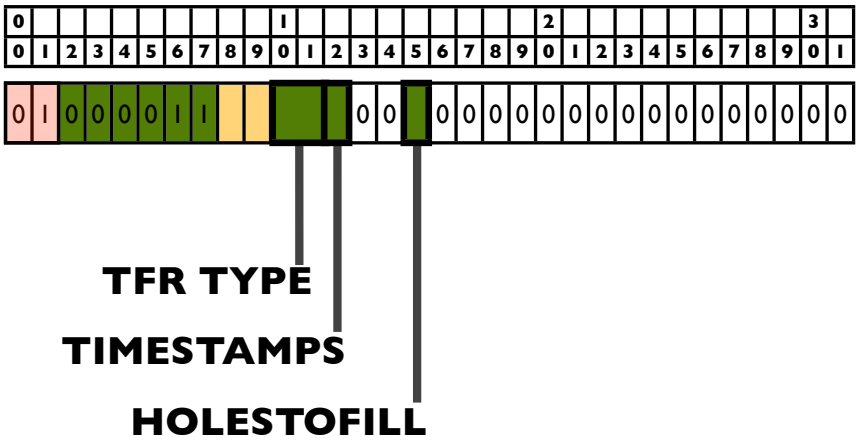


- Bit 7 Set of Properties identifies file is a directory
- Bit 6 Set of Properties identifies file as special file (sym link, named pipe, device ...)

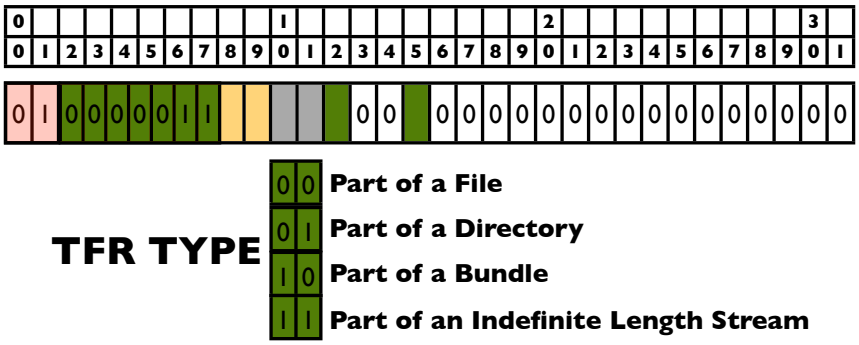
DATA FRAME



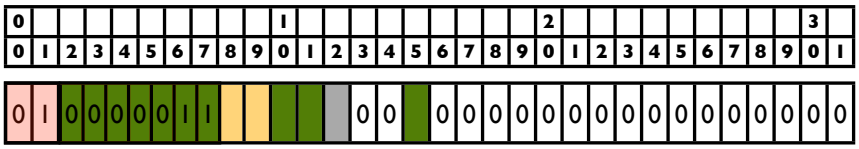
DATA HEADER FLAGS



DATA HEADER FLAGS - TFR TYPE



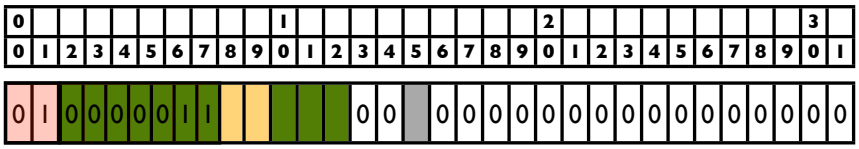
DATA HEADER FLAGS - TIMESTAMPS



TIMESTAMPS

0	**Timestamp not included in header**
1	**Timestamp included in header**

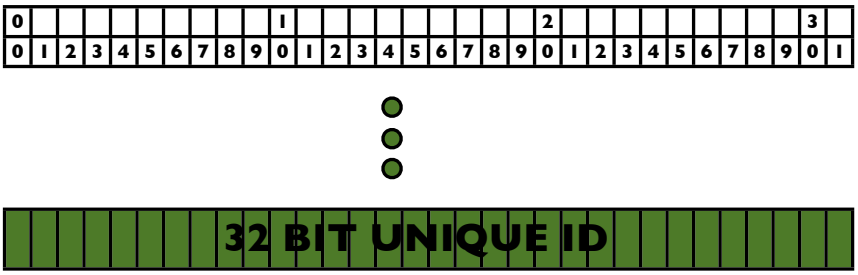
DATA HEADER FLAGS - HOLESTOFILL



HOLESTOFILL

0	**No response requested**
1	**HOLESTOFILL response requested**

DATA FRAME - UNIQUE ID

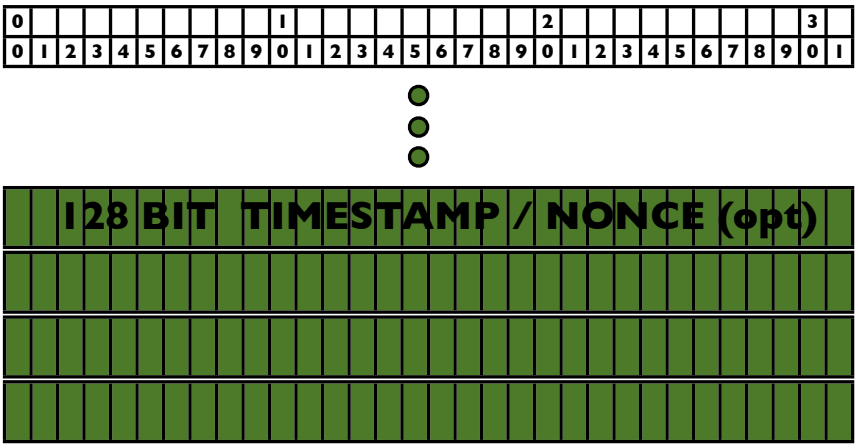


- The unique ID is a 32 bit field used during a transaction to uniquely associate a packet with a particular transaction.
- It is to match the transaction identified by a previously sent METADATA frame for a _put_ transaction

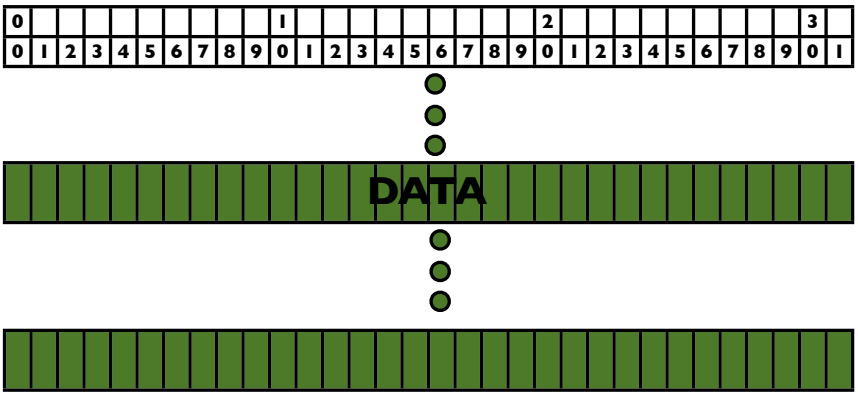
or

- It is to match the transaction identified by a previously received REQUEST frame for a _get_ or _getdir_ transaction.

DATA FRAME - TIMESTAMP / NONCE



DATA FRAME - DATA

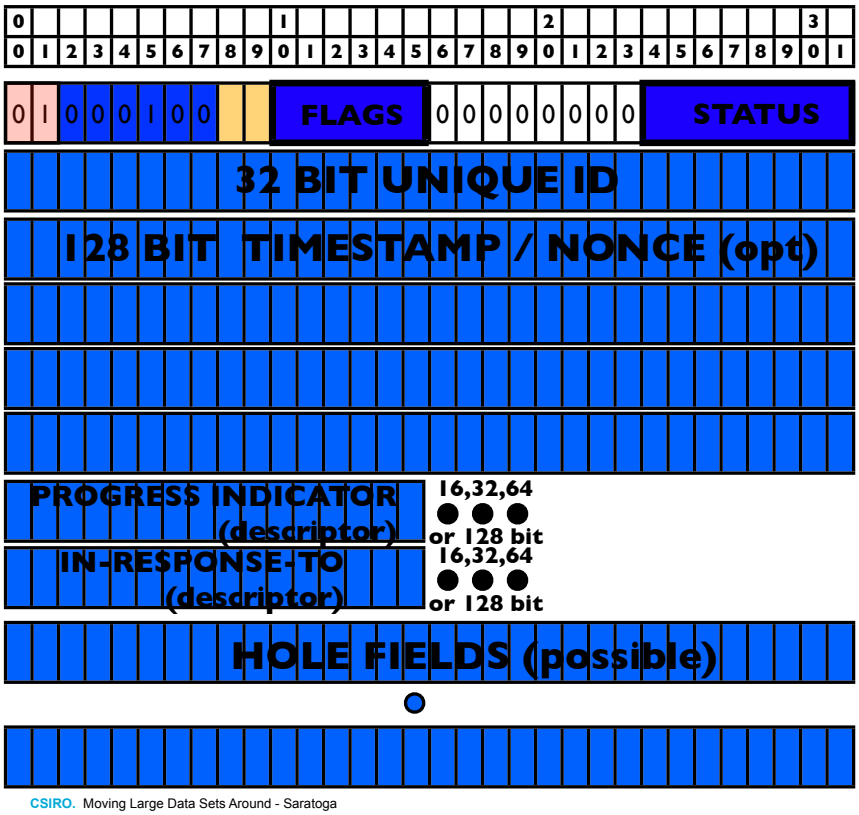


- Raw File Data for _get_, _put_ transactions.

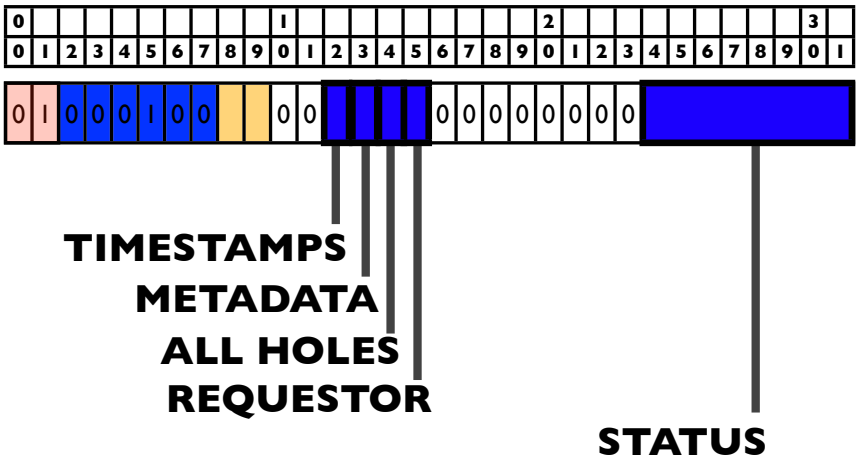
or

- List of DIR ENTRY structures for _getdir_, _putdir_ transactions.

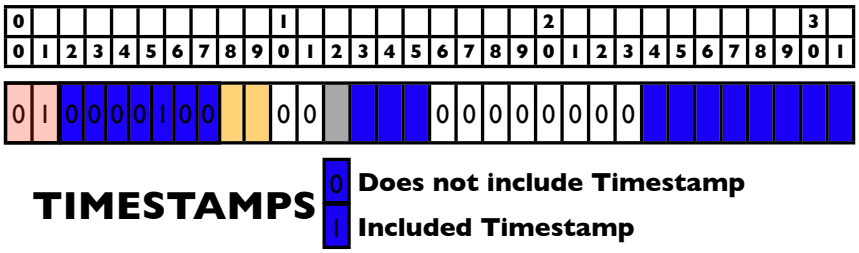
HOLESTOFILL FRAME



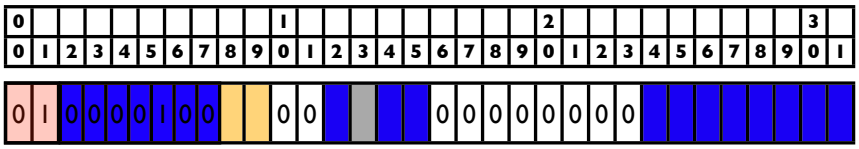
HOLESTOFILL FLAGS



HOLESTOFILL FLAGS - TIMESTAMPS



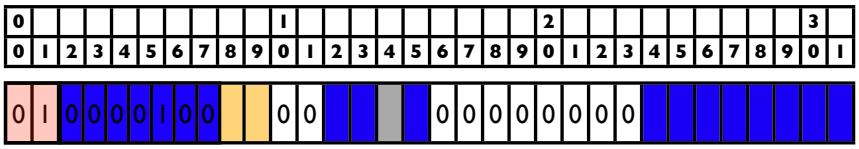
HOLESTOFILL FLAGS - METADATA



METADATA

0	**METADATA has been received for file**
1	**Files METADATA has not been received**

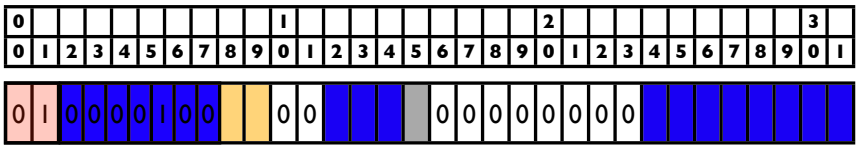
HOLESTOFILL FLAGS - ALL HOLES



ALL HOLES

0	**This packet contains all the holes**
1	**Packet has supplementary holes to be added in subsequent packet(s)**

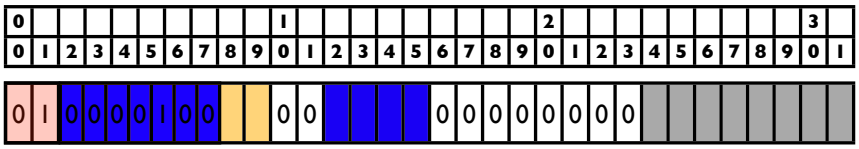
HOLESTOFILL FLAGS - REQUESTOR



REQUESTOR

0	**This HOLESTOFILL requested by sender**
1	**This HOLESTOFILL sent voluntarily**

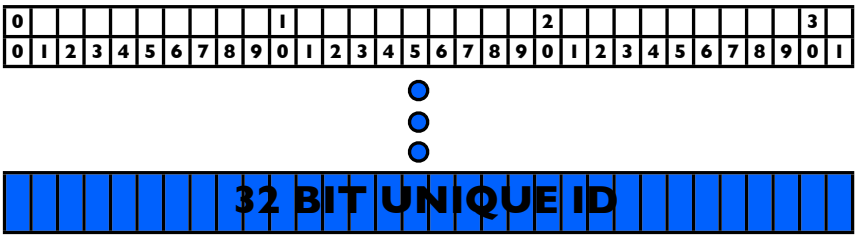
HOLESTOFILL FLAGS - STATUS



STATUS

0x00	Success - No Errors	0 0 0 0 0 0 0 0
0x01	Unspecified Error	0 0 0 0 0 0 0 1
0x02	Can't send - No resources	0 0 0 0 0 0 1 0
0x03	Can't receive - No resources	0 0 0 0 0 0 1 1
0x04	File not found	0 0 0 0 0 1 0 0
0x05	Access denied	0 0 0 0 0 1 0 1
0x06	Unknown ID field	0 0 0 0 0 1 1 0
0x07	Did not delete file	0 0 0 0 0 1 1 1
0x08	File length > REQUEST support	0 0 0 0 1 0 0 0
0x09	File offset descriptors mismatch	0 0 0 0 1 0 0 1
0x0A	Unsupported packet type	0 0 0 0 1 0 1 0
0x0B	DATA flag bits changed	0 0 0 0 1 0 1 1
0x0C	Receiver no longer interested	0 0 0 0 1 1 0 0
0x0D	Receiver request pause	0 0 0 0 1 1 0 1
0x0E	Receiver request resume trasnfer	0 0 0 0 1 1 1 0

HOLESTOFILL FRAME - UNIQUE ID

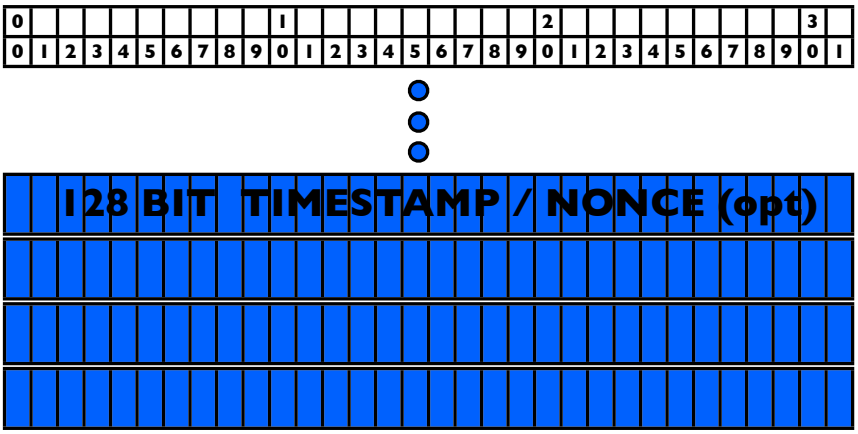


- The unique ID is a 32 bit field used during a transaction to uniquely associate a packet with a particular transaction.
- It is to match the transaction identified by a previously sent METADATA frame for a _put_ transaction

or

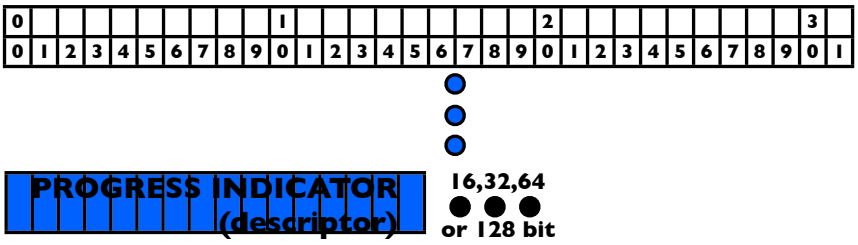
- It is to match the transaction identified by a previously received REQUEST frame for a _get_, _getdir_ or _delete_ transaction.

HOLESTOFILL FRAME - TIMESTAMP / NONCE



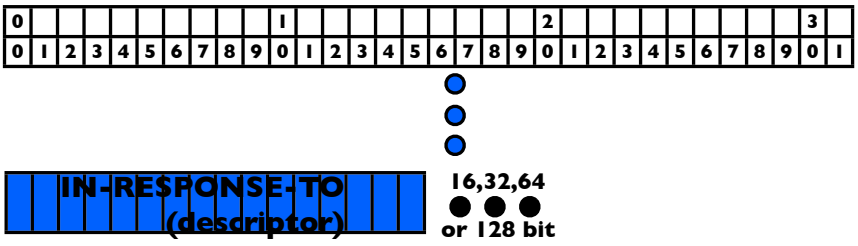
- If the voluntary flag is set, the optional timestamp holding the time when the last DATA frame was received
- If the HOLESTOFILL was generated as a mandatory request, the optional timestamp of the DATA frame that generated the request.
- This timestamp information can be used to calculate latency across the network.

HOLESTOFILL FRAME - PROGRESS INDICATOR



- The progress indicator is an unsigned integer which identifies the offset of the lowest-numbered octet of the file not yet received.
- It tells us how far along we have successfully come so far. Everything before this number has been received OK so we can clear those caches.
- When this number is the same as the file size then we know we are done and can close the transaction.

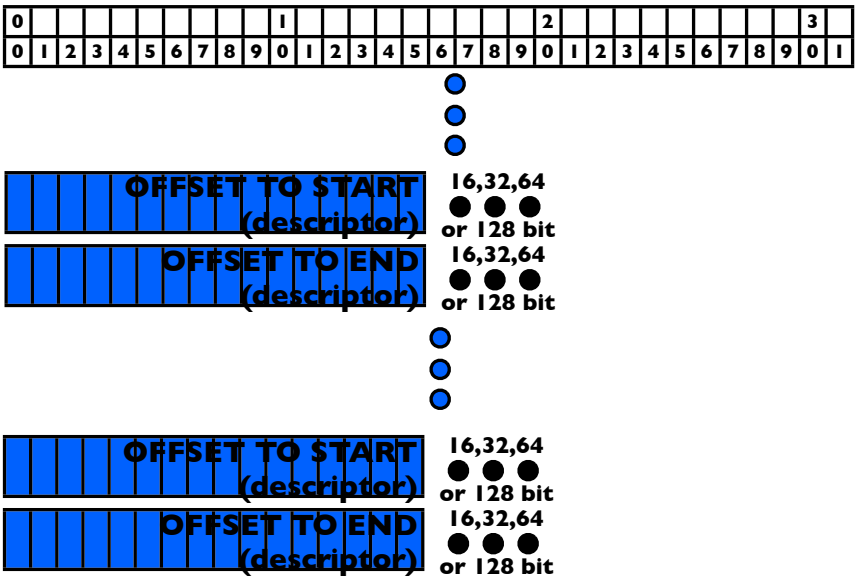
HOLESTOFILL FRAME - IN-RESPONSE-TO



- The in-response-to indicator is either:
  - If the voluntary flag is unset, an unsigned integer which identifies the offset of the highest-numbered octet within a DATA packet received that generated this HOLESTOFILL packet.

  or
  - If the voluntary flag is set, an unsigned integer which identifies the offset the highest number octet seen.
- This is where we are currently in the file when a holestofill was requested. You can use this to give a guide to how “good” our link is as a % of file size.

HOLESTOFILL FRAME - HOLES



- A series of zero or more tuples of descriptors holding the beginning and end offsets for data that needs to be re-transmitted.

Conclusion

- If you have a high speed private network and you want to get as much data reliably between peers as quckly as possible then Saratoga is a good choice.
- We need to run tests on Saratoga over dedicated links at Gigabit & 10 Gigabit speeds to determine its characteristics vs those of traditional file transport applications such as ftp.
- Saratoga can be easily modified to test new algorithms for congestion control with its inherent capabilities for accurate timestamps and a single select() system call handling all I/O multiplexing.
- The current IETF draft is:
  - draft-wood-tsvwg-saratoga-06.txt
- My reference implementation with a GNU public licence for Saratoga V1 in 'C' for Debian LINUX will be available for download from Sourceforge shortly.

Astronomy & Space Science
Charles Smith
Network Consulting Engineer

Phone: +61 404 058974
Email: charles.smith@csiro.au
Web: www.atnf.csiro.au

www.csiro.au

Thank you

Contact Us
Phone: 1300 363 400 or +61 3 9545 2176
Email: enquiries@csiro.au Web: www.csiro.au

CSIRO