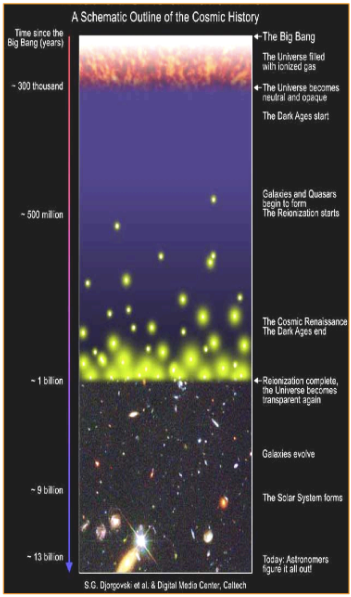
Challenges in Data Sensing, Transmission and Access posed by New Radio Astronomy Telescopes

Charles Smith
CSIRO
charles.smith@csiro.au

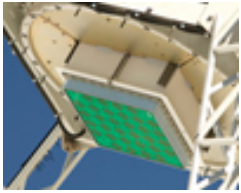
Exploring the Universe

- The first stars and galaxies in the Universe
  - Emergence of structure
- Large scale structure of the Universe
  - "Dark energy"
- Origin and evolution of cosmic magnetic fields
  - "The magnetic Universe"
- Gravity in the strong field case
  - Gravitational wave detection
- Planet formation
  - Including search for extra-terrestrial intelligence (SETI)
- EXPLORATION OF THE UNKNOWN



New Radio Sensor Types

Low Frequency Dipole
70MHz-450MHz



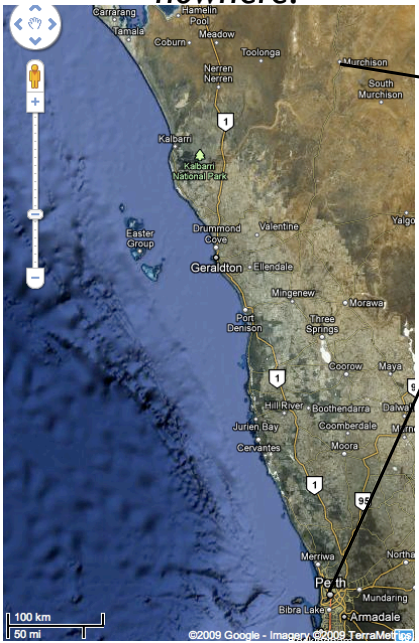
Phased Array Feed
450MHz-3GHz

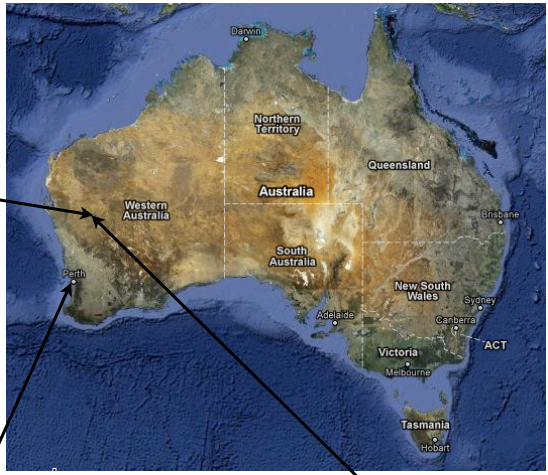
Aperture Array
500MHz - 1GHz

H1 (21cm line) Freq = 1420.4 MHz

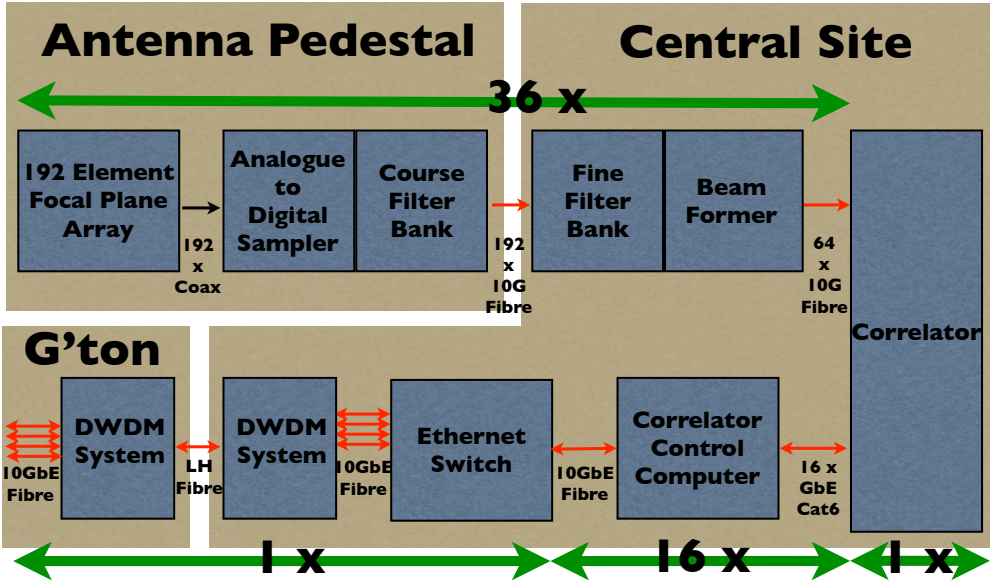
MRO Site Network & Computing Infrastructure Architecture

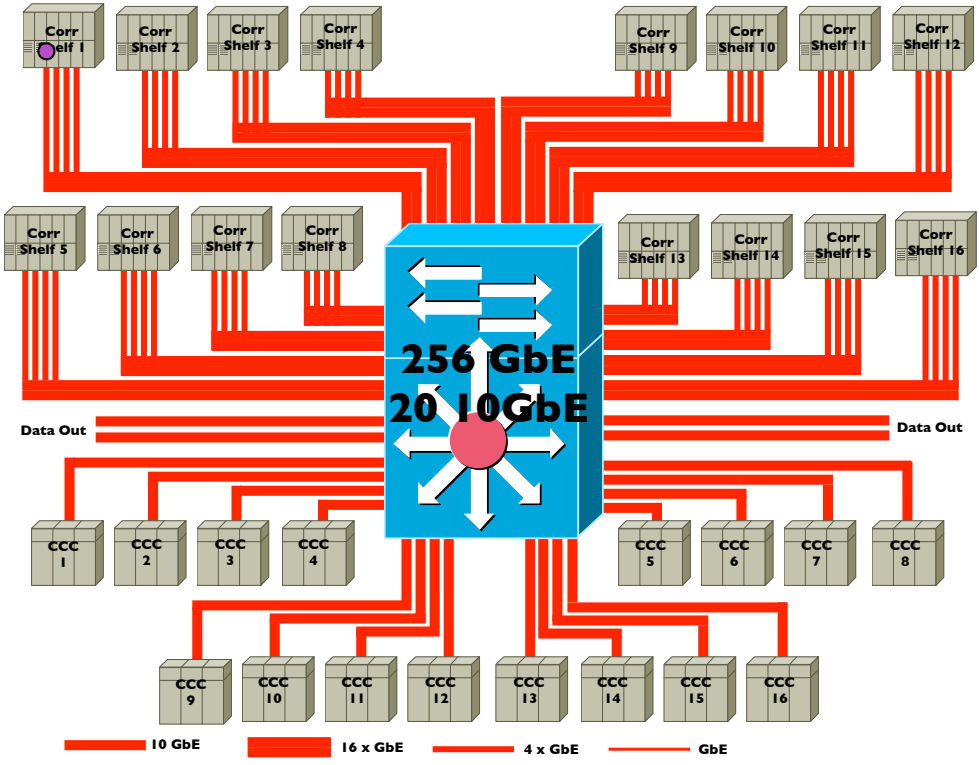
MRO - Yes, this is in the middle of *absolutely nowhere*.



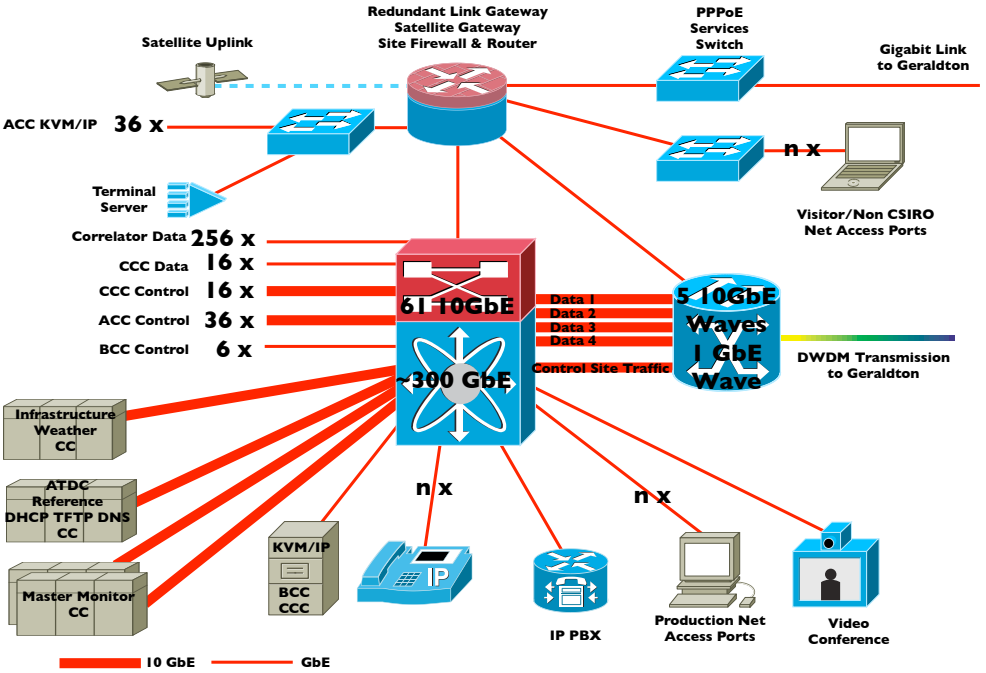


ASKAP - Data Plane Architecture

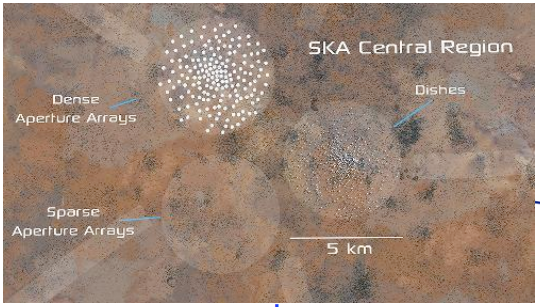



Corr Shelf 1
Corr Shelf 2
Corr Shelf 3
Corr Shelf 4
Corr Shelf 9
Corr Shelf 10
Corr Shelf 11
Corr Shelf 12
Corr Shelf 5
Corr Shelf 6
Corr Shelf 7
Corr Shelf 8
Corr Shelf 13
Corr Shelf 14
Corr Shelf 15
Corr Shelf 16
256 GbE
20 10GbE
Data Out
Data Out
CCC 1
CCC 2
CCC 3
CCC 4
CCC 5
CCC 6
CCC 7
CCC 8
CCC 9
CCC 10
CCC 11
CCC 12
CCC 13
CCC 14
CCC 15
CCC 16
10 GbE
16 x GbE
4 x GbE
GbE

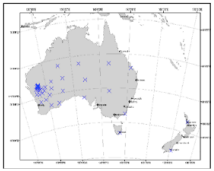
ASKAP - Site Network

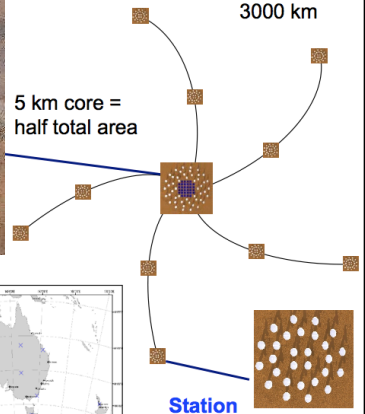


Square Kilometer Array - SKA



Inner core

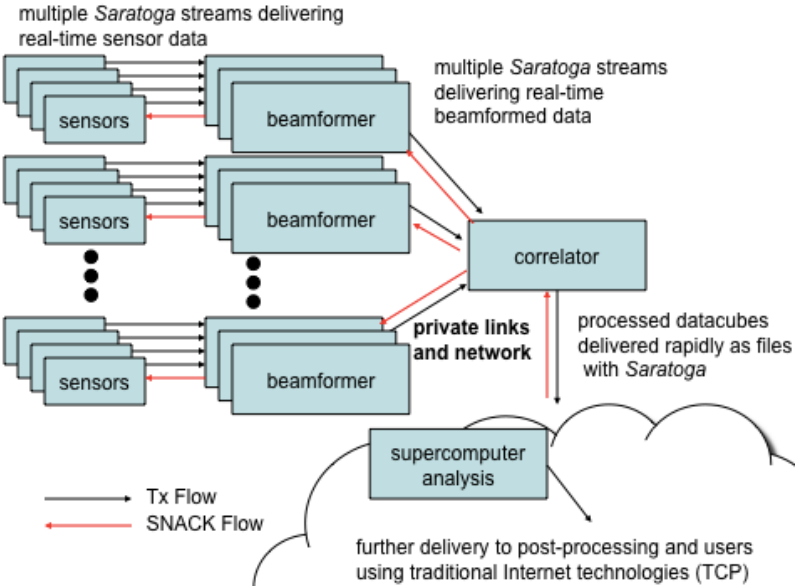




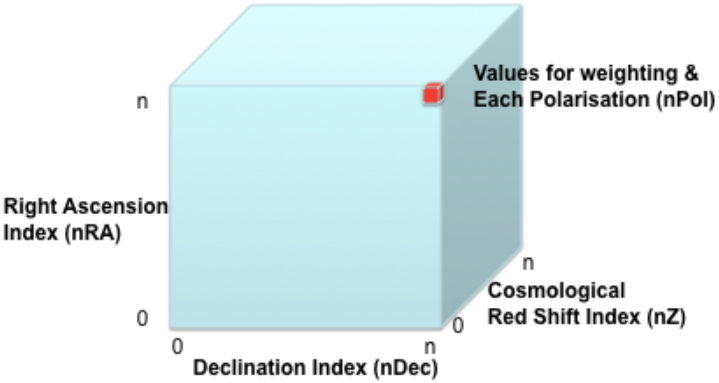
Moving Data Sets Around

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

Sensing & Processing in an Array



Output of an Array Observation Image Data Cube

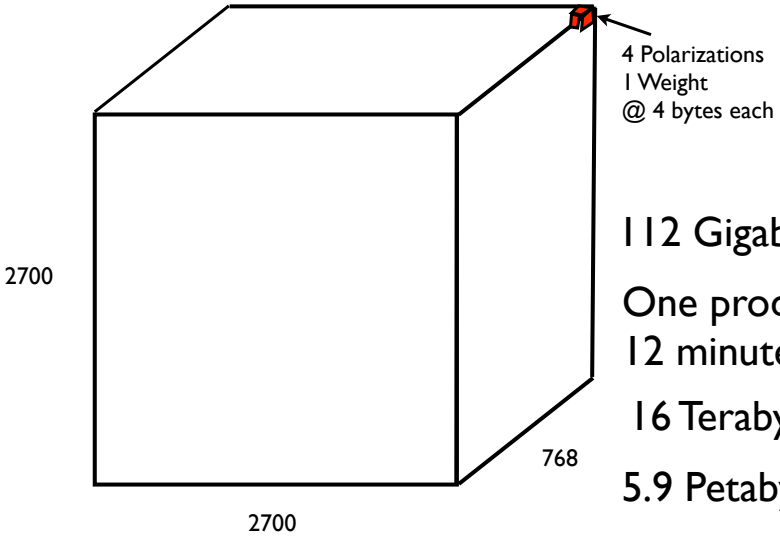


Data Image Cube Size (bytes) =

(nRA + 1) x (nDec+1) x (nZ+1) x nPol x fsize

fsize = single precision floating point number (4 bytes)

Image Data Cubes for MWA



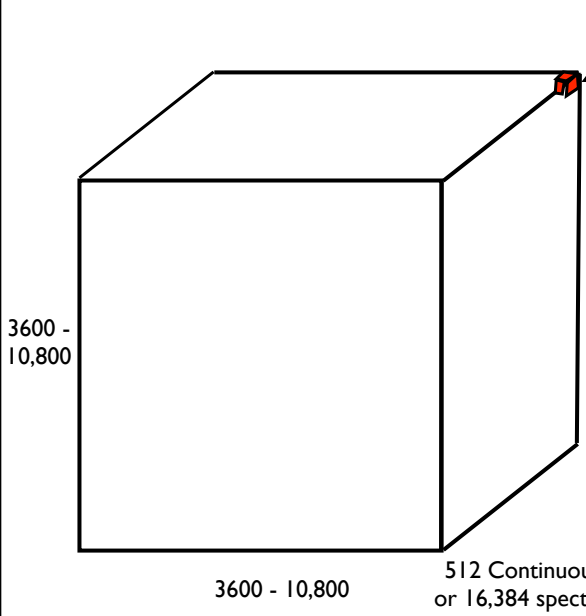
112 Gigabytes / Cube

One produced every 12 minutes

16 Terabytes / Day

5.9 Petabytes / Year

Image Data Cubes for ASKAP



53 Gigabytes
to 30 Terabytes/Cube

Wallaby all sky survey
1000 Cubes - 3.4PB

Dingo (Deep Focus)
2 x 2500 hrs (50 cubes)
5 x 500 hrs (250 cubes)
Total 2.6PB

Data Image Cubes Transport

- Given a 10 Gigabit Ethernet Connection
  - 3.4-Terabyte image takes ~45 minutes to transport
  - 71-Terabyte visibility image takes ~15 hours to transport

- A reliable, *high-speed* transport protocol is required.
- TCP-based transports just cannot fill a 10-Gigabit/s pipe.

CSIRO

Saratoga

A reliable, UDP-based, file/stream/bundle tranport protocol

What is *Saratoga* ?

CSIRO

- *Saratoga* is a high-speed, UDP-based, peer-to-peer protocol, providing error-free guaranteed delivery of files, or streaming of data.
- Flood data packets out as fast as you can. No specified congestion control is required, since data is usually only going one hop over a private link, or across high-speed, low-congestion private networks.
- No specified timers means no timeouts, so Saratoga is *ideal* for very long propagation delay networks (like deep space).
- Every so often the transmitter asks for an acknowledgement from the file receiver. The receiver can also send acks if it thinks it needs to, or to start/restart/finish a transfer.
- Acks are Selective Negative Acknowledgements (SNACKs) indicating received packets, and any gaps to fill with resent data, including information so that intelligent sender rate control or congestion control can be provided if needed.
- Any multiplexing of flows is done by the *Saratoga* peers.
- *Saratoga* is an excellent protocol to use in asymmetric network topologies.

Saratoga is a reliable transport over UDP

- The UDP checksum is used per packet to cover both the header and payload. It is consistent, but not that strong (one's complement), and does not provide end-to-end guarantees.
- An optional end-to-end checksum, using one of CRC32/ MD5/SHA-1, over the entire file being transferred, increases confidence that a reliable copy has been made, and that fragments have been reassembled correctly.
- The HOLESTOFILL list on the receiver requests the transmitter to re-send frames that have not been properly received (a SNACK).
- The transmit window does not move until the holes have been acknowledged by a HOLESTOFILL frame with an advanced offset. The receive window only advances when offsets are contiguous.

Optional Features of *Saratoga*

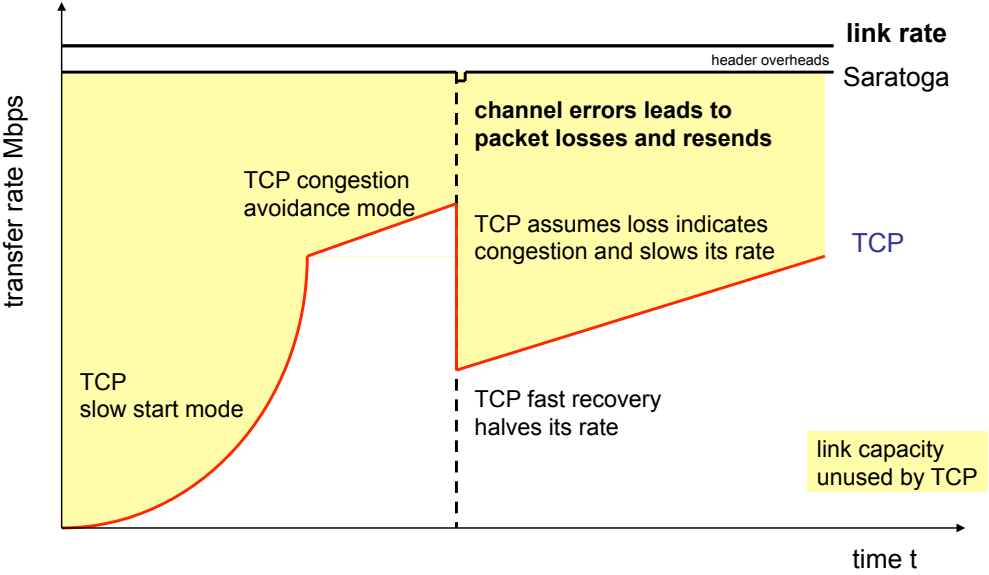
- Uses link-local multicast to advertise presence and discover peers.
- Data can be sent to local multicast addresses for multiple peers to receive. This is ideal for simultaneous distribution of file updates to multiple peers.
- Streaming of data is supported. This allows *Saratoga* to be used for real-time delivery outside the file-based paradigm.
- UDP-Lite is supported when errors in data received can be tolerated. The UDP-Lite checksum covers a minimum of IP/UDP-Lite/*Saratoga* headers. The header content is always checked so that the information *about* the data is error-free.
- Saratoga can also support “DTN bundle” delivery, and can act as a “bundle convergence layer”. This was shown on the UK-DMC satellite.
- Given reliable delivery in streaming mode, then *Saratoga* could be used as a transport for higher-layer protocols such as HTTP.
- A far higher-performing solution than TFTP (the trivial file transfer protocol) when delivery and reliability is required for system remote booting or updates.

Why *Saratoga* instead of FTP/TCP ?

CSIRO

- For high throughput and link utilization on dedicated links, where a single TCP flow cannot fill the link to capacity.
- For links where TCP's assumptions about loss/congestion/competition simply don't hold. i.e. High speed bulk transfer.
- There is no such thing as "slow-start" in *Saratoga*.
- Able to cope with high forward/back network asymmetry (>850:1).
- Long path-delay use – eventually TCP will fail to open a connection because its SYN/ACK exchange won't complete. TCP has many unwanted timers.
- Simplicity. TCP is really for a conversation between two hosts; needs a lot of code on top to make it transfer files. A focus on just moving files or streams of data makes sequence numbering simpler.
- Having SNACKs means that handling a sequence number wraparound when in streaming or bundling mode becomes easy.

Why *Saratoga* instead of FTP/TCP ?



What Saratoga does not do

- There is no MTU discovery mechanism, so you have to know the maximum packet size your network can transmit at. i.e. dictated by the frame size. This is okay for your own private network, but would be troublesome if used across the Internet.

- There is no such thing as “slow-start” or congestion control. That is considered bad and unsociable behaviour on the Internet. *Saratoga* just blasts away on a link with no regard for other flows - which is the exact behaviour that makes it desirable in these environments!

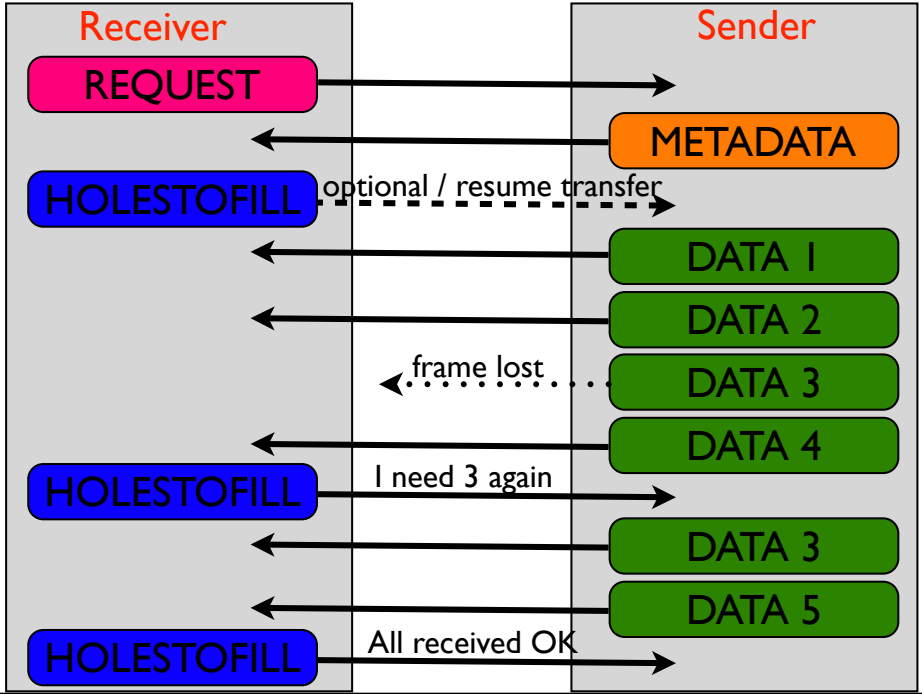
Saratoga Transactions

GET	Get a named file from the peer
GETDIR	Get a directory listing of files from the peer
DELETE	Delete a named file from the peer
PUT	Put a named file or stream data to the peer
PUTDIR	Put a directory listing of local files to the peer

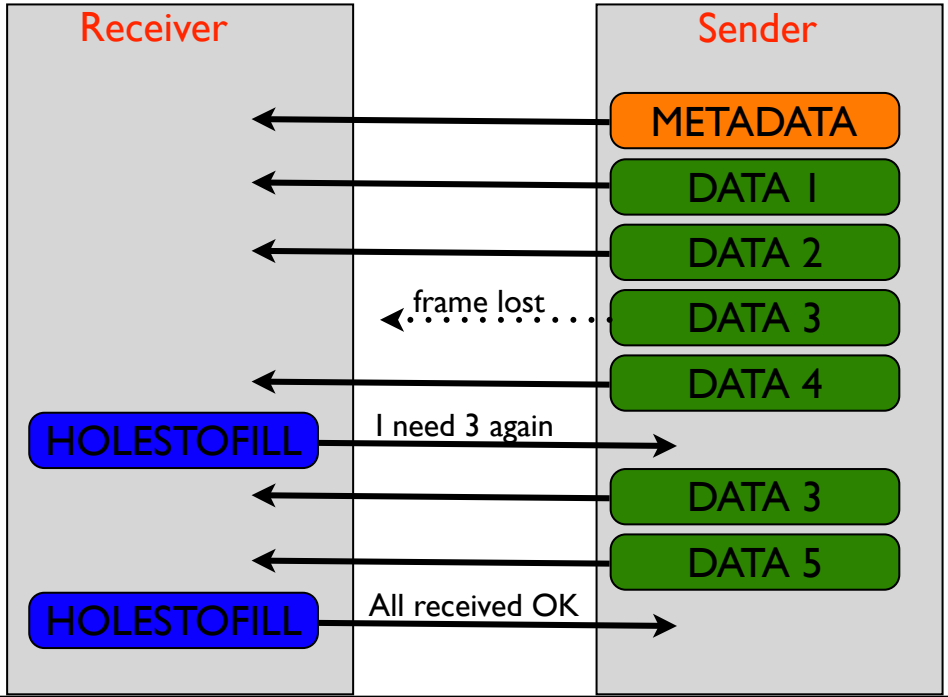
Saratoga Frame Types

BEACON	Sent periodically. Describes the *Saratoga* peer: Identity (e.g. EID) capability/desire to send/receive packets. max. file descriptor handled: 16/32/64/128-bit.
REQUEST	Asks for a file initiating 'get' transaction get file get directory listing delete a file.
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.)
DATA	Actual Data. Uses descriptor of chosen size to indicate offset for data segment in the file/bundle or stream.
HOLESTOFILL	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.

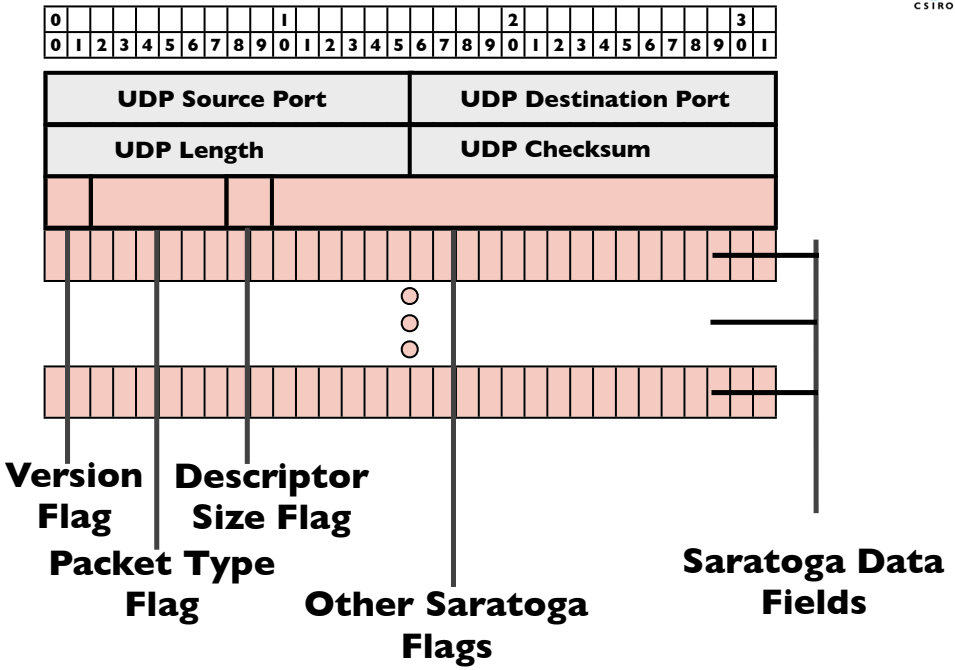
Transaction GET or GETDIR



Transaction PUT or PUTDIR



Saratoga Header Frame Format



Conclusion

- If you have a high-speed private network and you want to get as much data reliably between peers as quickly as possible, then *Saratoga* is a good choice. (That's why it's used to download images from SSTL's DMC satellites.)

- Radio astronomy has high-speed private networks, and needs to move a massive amount of data around. So we're implementing *Saratoga* for radio astronomy.