Saratoga: A Scalable File Transfer Protocol
- High Speed Performance Testing -

Will Ivancic
NASA Glenn Research Center
william.d.ivancic@nasa.gov

David Stewart
Verizion Federal Systems
dstewart@nasa.gov

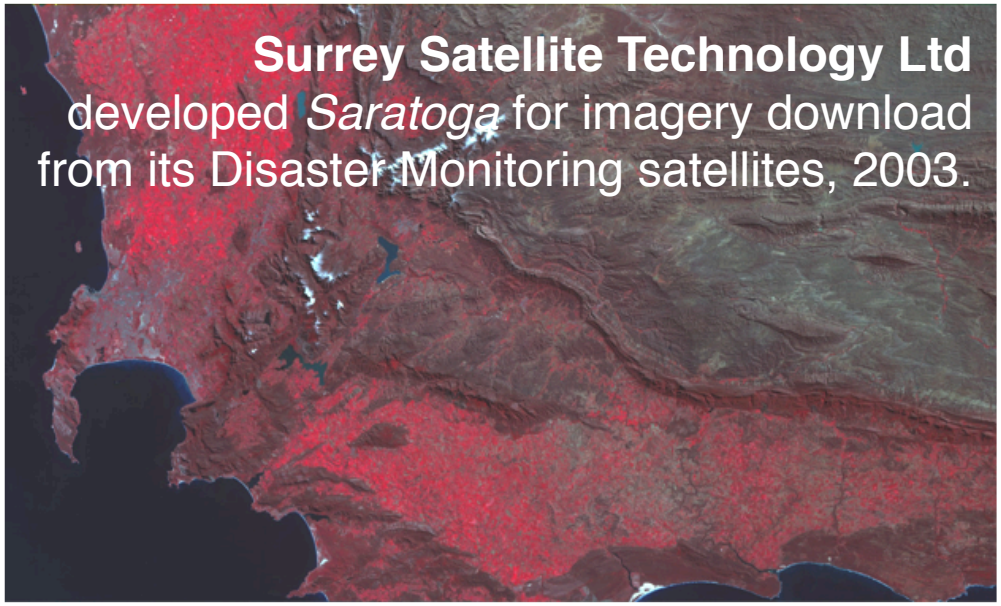
Private sensor networks

- Must deliver sensor data – very quickly.
- Want to use Internet technologies – cheap, reliable, robust.
- Want more speed than TCP can offer.
- Congestion is not a problem; private single-owner managed network with scheduled traffic, single flow per link with no competition. **This is not the shared public Internet!**
- Sensor capabilities are ever-increasing (side-effects of Moore's law). Need to scale for ever-growing data sizes.
- Support for streaming and simultaneous delivery to multiple receivers is also useful.
- ***Saratoga* protocol designed to meet these needs.**

Saratoga in short

- *Saratoga* is a high-speed, UDP-based, peer-to-peer reliable transport protocol, providing error-free guaranteed delivery of files, or streaming of data.
- Send 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.
- Some implementations have a rate-limiting option for restricted downstream links where line rate may not match downstream radio link.
- No specified timers means no timeouts, so *Saratoga* is can be used in very long propagation delay networks.
- *Saratoga* is an excellent protocol to use in highly asymmetric network topologies.

Saratoga's development



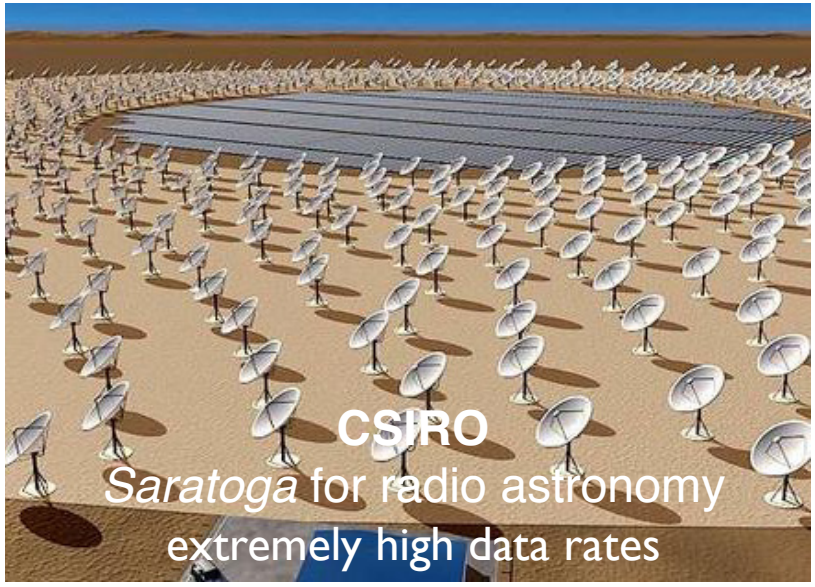
Surrey Satellite Technology Ltd developed *Saratoga* for imagery download from its Disaster Monitoring satellites, 2003.

NASA Glenn Research Center
Saratoga for sensors on UAVs

Saratoga redesigned, specified to the Internet Engineering Task Force, 2007.

NASA Glenn uses *Saratoga* to test DTN and Interplanetary Internet on UK-DMC, 2008.

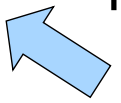
Multiple *Saratoga* implementations in progress with interoperability testing.



CSIRO
Saratoga for radio astronomy extremely high data rates

Research led to new use

- SSTL remote-sensing images grew to cross 4GiB file size, needing >32-bit pointers.
- How to design a *scalable* file transfer protocol able to handle any size file, without requiring separate incompatible implementations for big files?
- Solved this problem with 16/32/64/128-bit pointers and advertising capabilities.



not needed - yet!

- Support for scalability and streaming introduced new users – high-speed networking for radio astronomy in Very Long Baseline Interferometers.

Saratoga operation

Simple sliding window with selective acknowledgments.

- The HOLESTOFILL list on the receiver requests the transmitter to re-send frames that have not been properly received (a SNACK) by sending a STATUS with the list of HOLESTOFILL.
- The receive window only advances when offsets are contiguous. The left edge of the transmit window does not advance until the holes have been acknowledged by a HOLESTOFILL frame with an advanced offset.
- 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 for payloads sent using multiple packets.
- 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.

Features of *Saratoga* v1

Major features

- Scalable to handle large files. 16-bit descriptors for efficiency with small files. 128-bit descriptors cope with *huge* files up to 2^{128} bytes. 32- and 64-bit descriptors most useful.
- Streaming of data is supported. This allows *Saratoga* to be used for real-time delivery outside the file-based store-and-forward paradigm.

Minor features

- Supports link-local multicast to advertise presence, discover peers and for delivery to multiple receivers simultaneously for e.g. file or code image updates. (Will outperform TFTP trivial file transfer.)
- Optional UDP-Lite use for tolerating errors in payloads and minimizing checksum computation overhead. 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.
- Optional “DTN bundle” delivery as a “bundle convergence layer”. Shown with tests from the UK-DMC satellite.

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.
- ***Saratoga* does not include "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 private networks and these environments!
  - Simulations have shown that it is possible to implement congestion control mechanisms in *Saratoga* if desired – see University of Oklahoma paper describing Sender-Based TCP Friendly Rate Control (2010 IEEE Aerospace Conference).
    - Saratoga's timestamp option can be used to implement such *closed-loop* mechanisms.
  - Simple *open-loop* rate-limiting output to *XMbps* can also allow Saratoga to coexist with other traffic.

Saratoga version 1 implementations

C (Charles Smith under contract to Cisco Systems)

- Implementation licensed to CSIRO by Cisco.
- Built for Speed (Raw I/O).
- Streaming to be implemented in FPGA, File transfer may be implemented in FPGA.

C (Surrey Satellite Technology Limited – SSTL)

- Implemented in Real-Time Operating System for high-speed image transfers from Low Earth Orbiting (LEO) satellites over highly asymmetric links.

PERL (NASA Glenn Research Center)

- Sequential file transfer and rate-limiting implemented.

C++ (NASA Glenn Research Center)

- Discovery, multiplexed file transfer, hooks for bundling and streaming and rate-limiting to be implemented.

Wireshark Dissector (Charles Smith)

- http://sourceforge.net/projects/saratoga/files/

We hope to make some of these implementations available to the public.

Testing Strategies

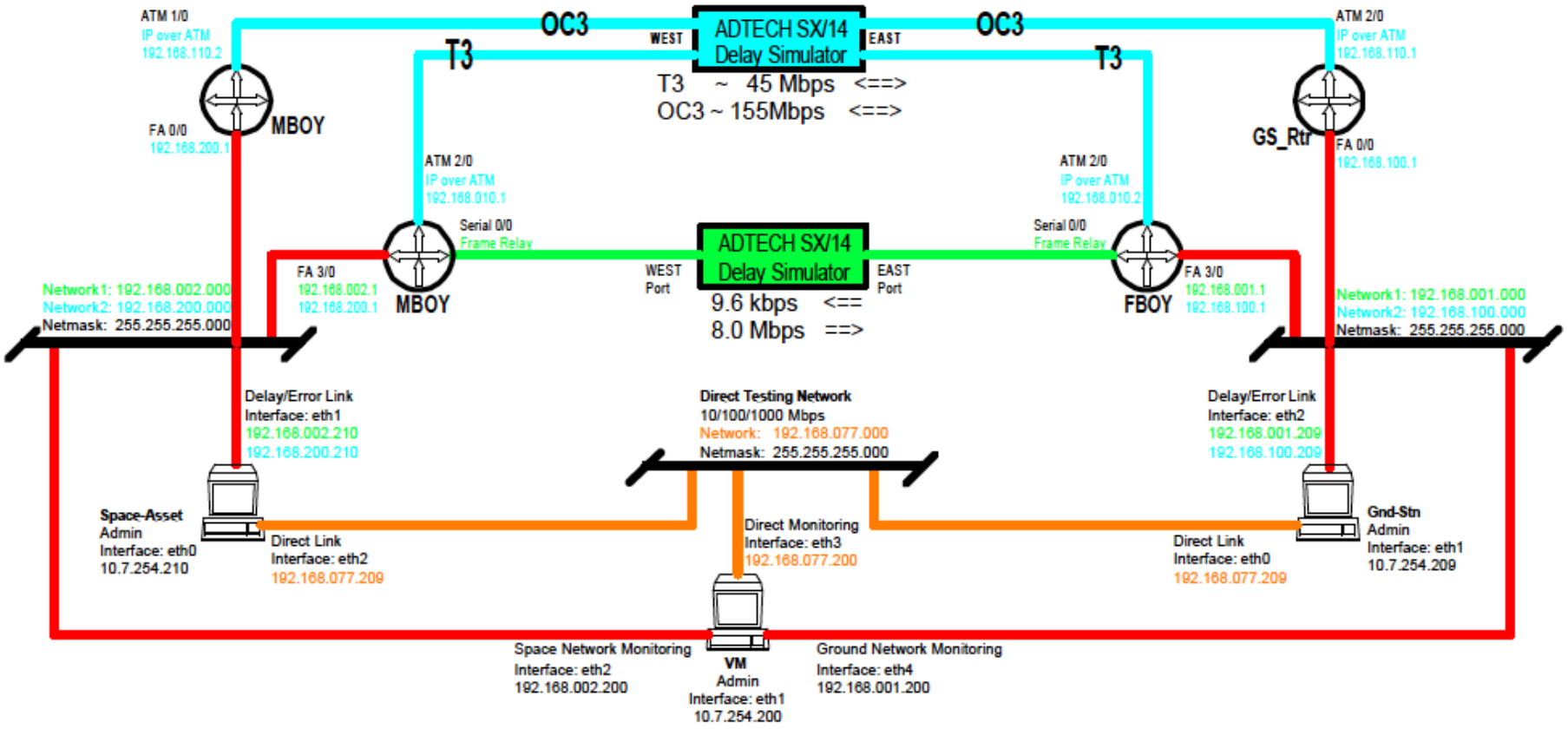
Interoperability Testing

- Remove ambiguities from Internet Draft
- Use NASA Testbed
- GRC PERL implementation directed at Interoperability testing and ease of distribution.

Performance Testing

- Not initially part of NASA Earth Science Task but ANI testbed made this possible within funds allotted.
  - Useful for NASA as we move large data sets from space-to-ground and from ground-to-ground.
- Cisco Implementation in C for high-speed operations
  - Target was Square Kilometer Array (SKA)
  - Pre-Alpha code
  - Cisco permitted NASA to test / debug with understanding the code was incomplete at best with no plans to continue (transport protocols are not part of Cisco's business case)

NASA Interoperability Testbed



Performance Testing

PERL

- Not designed for performance, but useful to get comfortable with ANI testbed.
- Successfully transferred 10G files at 50Mbps with PERL version of Saratoga. This tested the implementation of 64 bit byte counters needed due to the large size of the file. This test was possible due to the large amount of ram of the 100G testbed's hosts.
- Successful transfers of 50M file at a 10% packet loss and 500M file at 1% packet loss.
- Netem was used to simulate RF link conditions.

Performance Testing

C implementation

- Designed for Speed
- Implementation target is moving very large data sets (Terabytes) over low-loss high-rate terrestrial fiber links.
- Pre-Alpha Code
  - Transfers were not operating correctly – really pre-alpha!
  - Debugging ongoing
- Currently stopped performance testing work as Earth Science task ends in May
  - Working Layer-2 triggers (modemLPA and DLEP) to trigger rate adaptation
  - Would like to complete if resources become available

In Conclusion

- Performance testing would not have been attempted without access to DOE-ANI Testbed
  - Cost and time prohibitive to put together such a facility for only NASA use.
- Testbed design and operations is well thought out, novel and quite cleaver
  - Each experimenter controls their own image
  - Keeps different experimenters from accidently effecting another's configurations
  - Good from a security perspective versus multiple accounts on the same machine
  - Relatively easy to use
    - Remote operations are always a bit “interesting” as you just cant walk over to a machine to see what is happening

Thanks to DOE-ANI

We would not have been able to attempt performance testing without the 100 G testbed