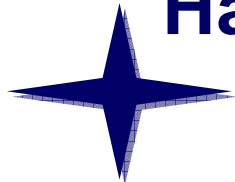
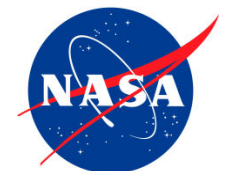


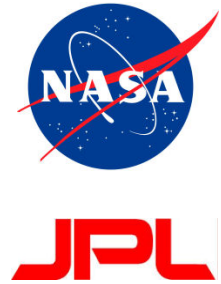
# D-RATS 2011 RAFT Protocol Overview

Hans Utz



- Did his doctoral research on the deployment of distributed system middleware on autonomous mobile robots
- Is working at the Intelligent Robotics Group at NASA Ames Research Center as the Rover-Software Lead
- Member of the inter-center RAPID (Robot API Delegate) team working on a common high-level APIs for interoperability of robots from different NASA centers
- Lead architect of the RAFT (RAPID Advanced File-Transfer) protocol

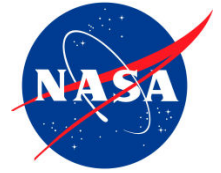




# ★ RAFT Target Scenario

- ◆ Planetary robot(s) sending data back to ground-control
- ◆ More data products available than bandwidth
- ◆ Data downlink most likely over a comm-relay
- ◆ Two-hop topology
  - ◆ High latency, fairly reliable space link
  - ◆ Low latency planetary link(s) with big fluctuations in reliability
  - ◆ Differing/varying bandwidth availability on each hop
- ◆ Medium latency: 1.7 – 50 seconds (Lunar/NEO)
- ◆ Online adjustments to data-product prioritization

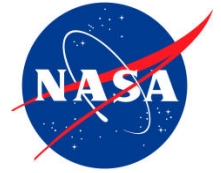




# ★ RAFT Requirements

- ◆ Optimal/full utilization of allocated bandwidth
- ◆ High flexibility in data-product prioritization
- ◆ High degree of automation
- ◆ Latency-tolerant interface
- ◆ (x)GDS integration
- ◆ Observability of the system/data-flow

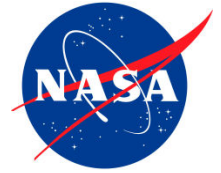




# RAFT Design



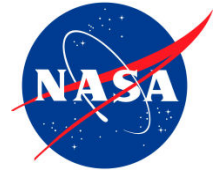
- ◆ Multiple channels with token-based bandwidth allocation
- ◆ Priority queue on each channel
- ◆ Pre-fetching on the planetary network (pull-model)
- ◆ Data-products are announced on the planetary network by data-producers
- ◆ (Remote) commands:
  - ◆ Put, Remove
  - ◆ Pause, Resume
  - ◆ Set bandwidth
- ◆ Auto-queuing
  - ◆ Uses file-name and meta-data based rule-set to select default priorities and queues



# RAFT Communication

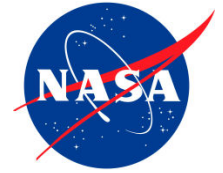


- ◆ Shared information space (DDS)
  - ◆ White-board style: instances of data organized by topic
  - ◆ Updated through a pub/sub architecture
  - ◆ QoS dimensions per topic: reliability, durability, history, ...
- ◆ RAFT file-announce topic (reliable, durable)
  - ◆ Availability, location and size of files
- ◆ RAFT state topics (reliable, durable)
  - ◆ Queue state: file and data volume
  - ◆ Sample state: queued, pre-fetching, active, done
  - ◆ Receiver state: file and data volume, cache size (best effort)
- ◆ RAFT file sample topic, reliable
  - ◆ Chunks of files with id



# ★ RAFT Implementation

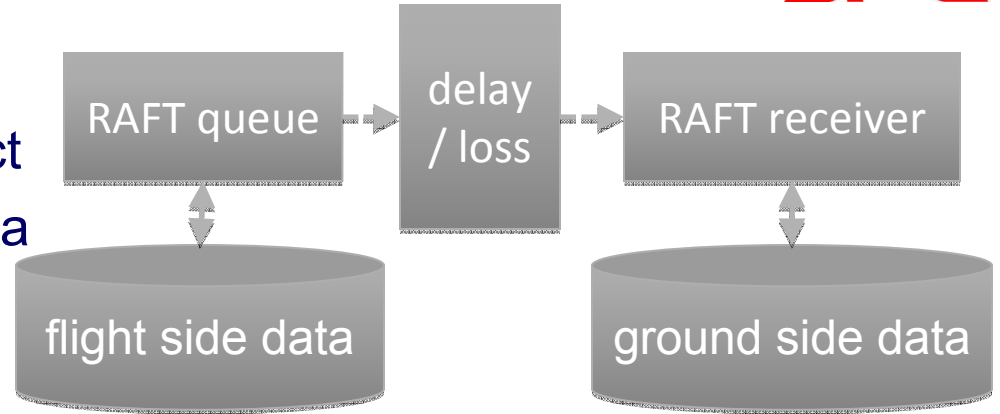
- ◆ DDS/RTSPS2 used as distributed systems middleware
  - ◆ reliability protocol (ack/nack based) over UDP
  - ◆ extended the protocol to work with huge bandwidth-delay product ( $50 \text{ s} * 10 \text{ MBit/s}$ )
  - ◆ used for file-transfer as well as meta-data (RAFT state) communication
    - ◆ Token-bucket based flow-control for bandwidth management
- ◆ C++ implementation (file queue and file receiver)
- ◆ libCurl (scp) used for pre-fetching
  - ◆ can specify bandwidth-limits in the library
  - ◆ kernel-level limits would be more appropriate
- ◆ Java/Eclipse front-end
- ◆ Web-frontend



# RAFT Results: Lab

- ◆ Reliable communication over varying bandwidth, latency & packet-loss

- ◆ 100ms, 1.7s, 50s delay
- ◆ 0%, 1% loss, disconnect
- ◆ 250 MB to 1.5GB of data

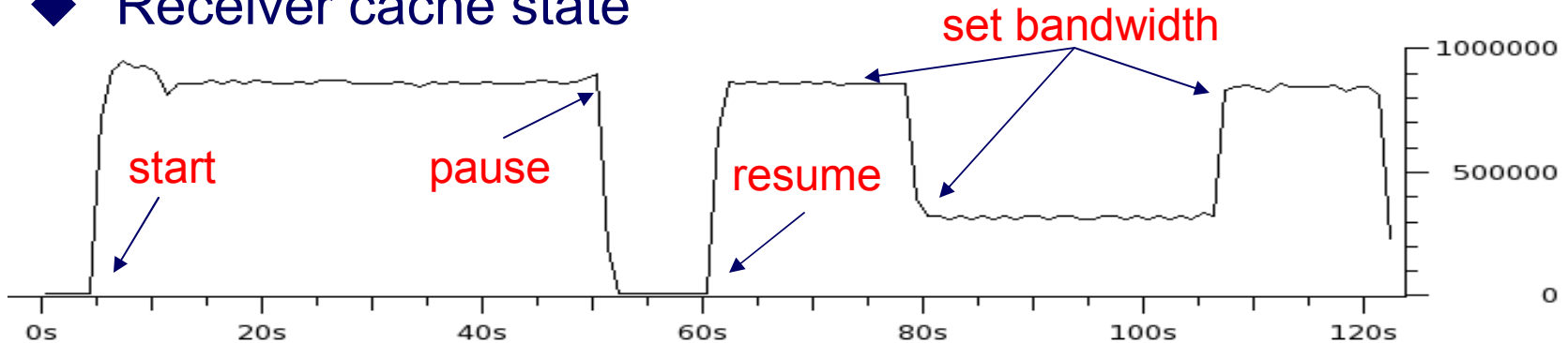


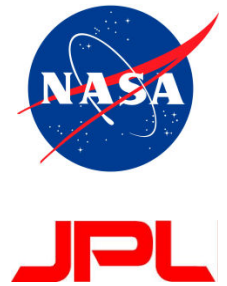
- ◆ Uniform bandwidth utilization

- ◆ Adherence to bandwidth limits

- ◆ Burstiness (application space) due to total ordering

- ◆ Receiver cache state





# ★ RAFT Results: D-RATS 2011

- ◆ Extensively used for data-download
  - ◆ 800 data-products
  - ◆ 12.500 files
  - ◆ 9.5 GB
- ◆ Full xGDS integration
  - ◆ Automated science data processing
  - ◆ Web based RAFT status-display
- ◆ Data from 8+ data producers
  - ◆ 2 robots
  - ◆ 4 suits (2 laptops)
  - ◆ 2 GigaPan Voyage installations

