Layering IT Services for a Planetary "Exploration Web"

California Institute of Technology

Norm Lamarra

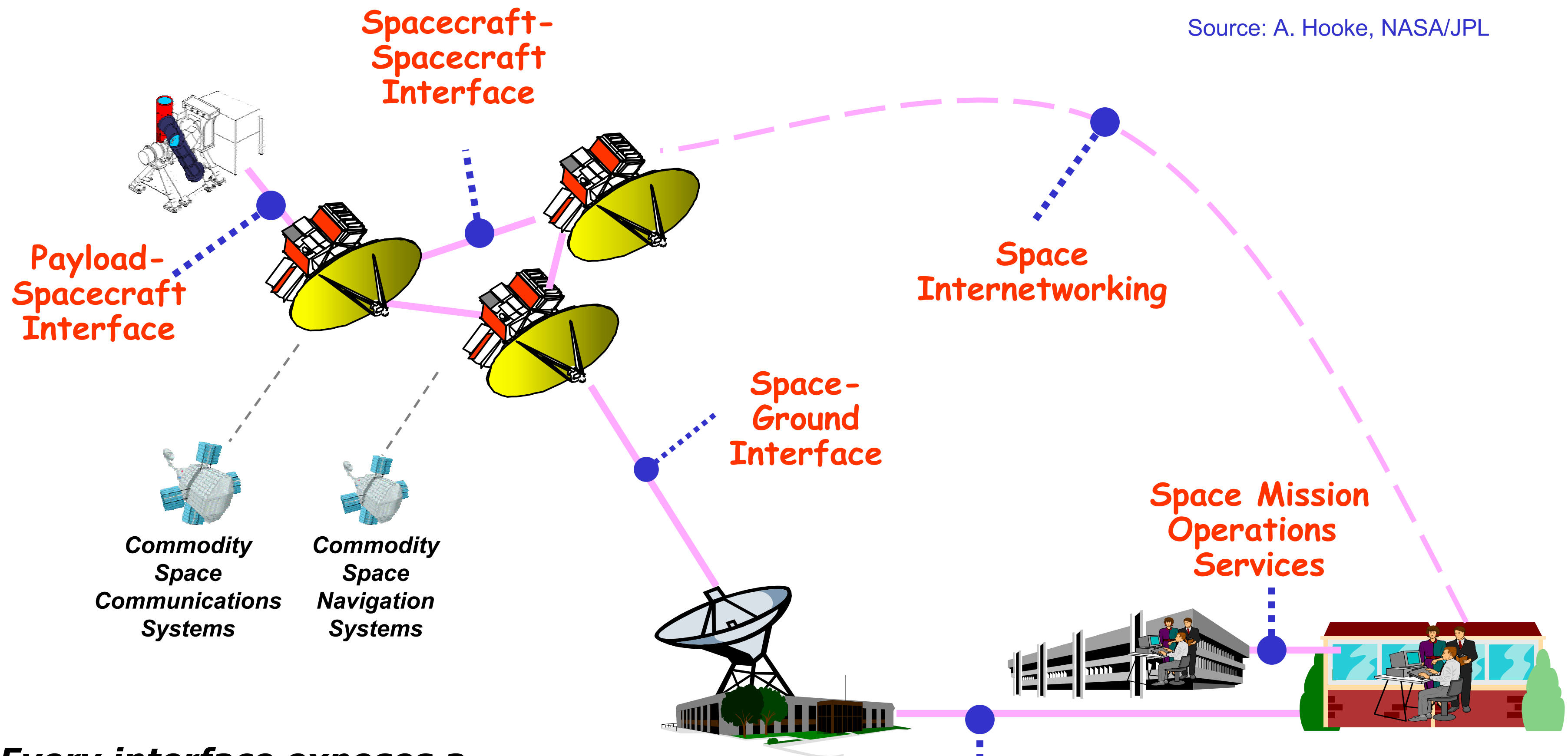
GSAW 2003

Mar 4-6, 2003

Overview

- Background & Problem Domain
  - Example Space Communications Scenarios
  - Problem Areas
- Proposed Middleware Approach
  - Shared Services
  - Layered Middleware View
- Potential Middleware Benefits
- IND Prototypes
- Future Steps

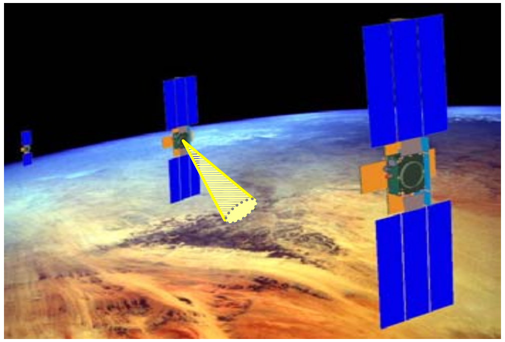
Space Communications Interoperability Points



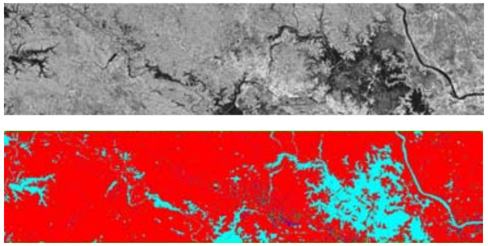
Every interface exposes a catalog of standard services and protocols

Example: ASE Mission Scenario

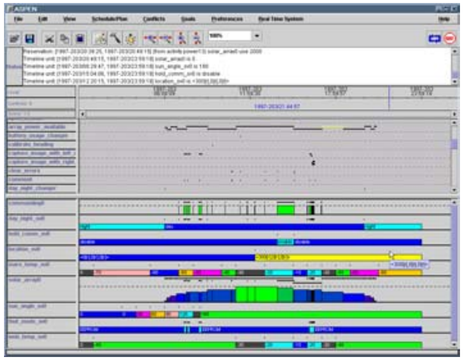
Target Image with Autonomous Sciencecraft



Onboard Science Processing and Event Detection

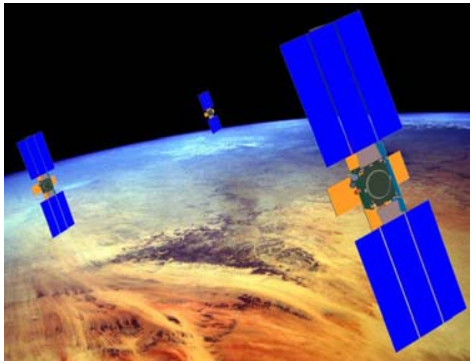


Onboard Replanning



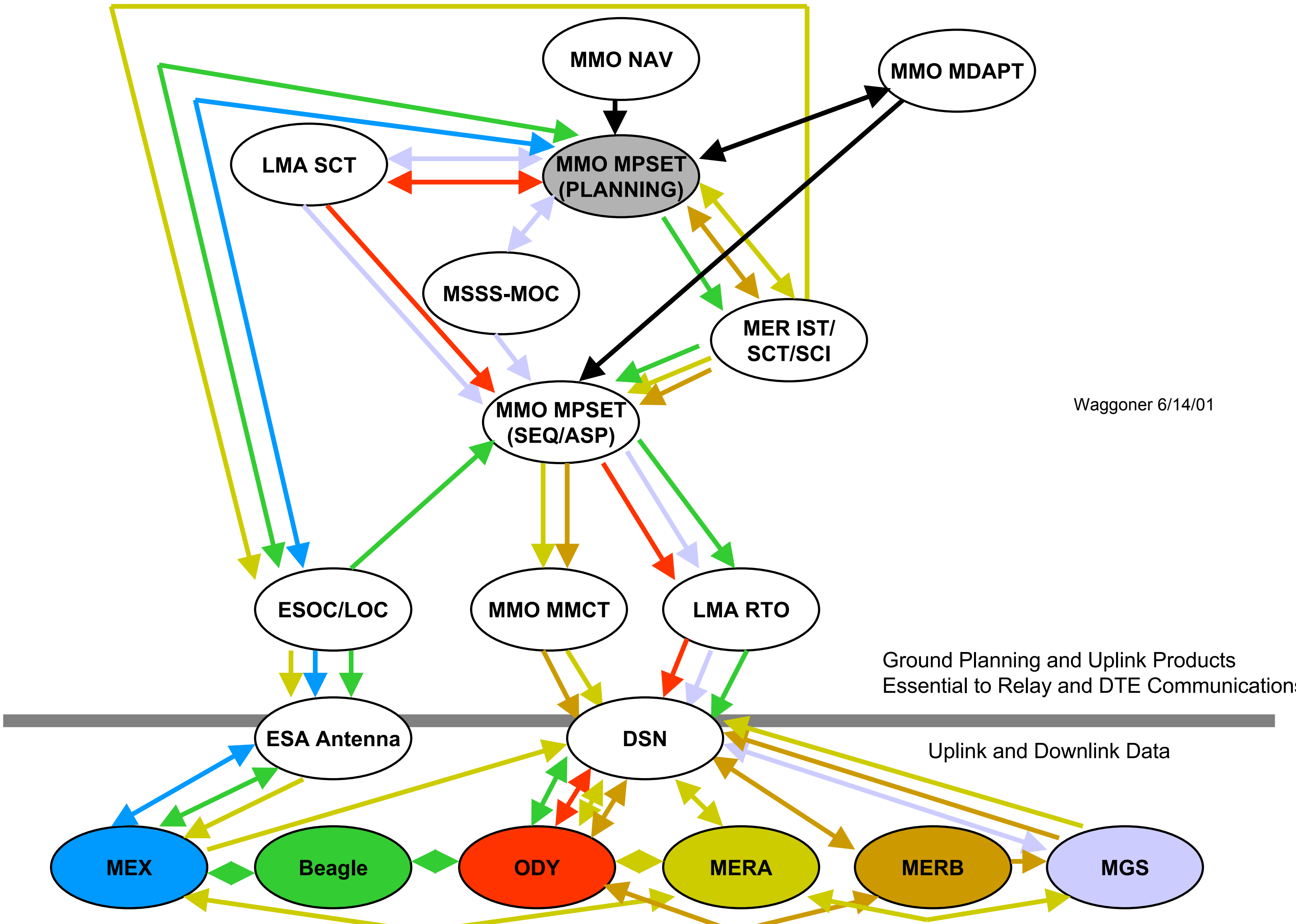
New Science Images

Execute plan to acquire new images



Autonomous Sciencecraft Experiment (on TechSat21)

Example: Mission-Interaction Dataflow



Problem Areas

- Scientist-instrument connection
  - Large effort and cost of coordinating mission plans
  - Coordination among spacecraft
- Value of data
  - Much sensor data can't reach Earth (MGS d/l <1% data)
    - Which bits to d/l? Knowledge vs. Information vs. Data
  - Coordinated measurements
- Operations cost
  - Support for automation (e.g., data management)
  - Support for autonomy
    - On-board reasoning (e.g., vehicle health, science goals, etc.)
- Application development
  - Few standard API's or accepted s/w architecture
  - Difficult to access distributed resources
  - Limited robustness (e.g., failure det'n/recovery, s/w modification)

Proposed Approach

JPL

- Conceptualize a set of standardized "*shared services*"
  - 3 broad categories: Communications, Storage, Processing
  - Distributed client-server model useful for all 3
    - Make object model highly flexible
    - Make clients as lightweight as possible
    - Simplify server replication (when necessary)
  - Build upon "enhanced" internet-style communication
    - Asynchronous messaging has many advantages
    - Publish/subscribe has further advantages
    - Message prioritization and efficiency are crucial
- Deploy "*layered infrastructure*" incrementally
  - Basic services: Messaging, time, events, security
  - Information services: data management, alarms
  - Higher-level services: navigation, weather, etc.
  - Agent interaction infrastructure (far future)
    - e.g., "autonomous" communication vs. "scheduled"

Time

Shared IT Services

- Communications
  - Tolerate delay, disconnection, b/w limitation
    - Buffered, asynchronous, …
  - Tolerate variety of network topologies (near/far)
    - Simplify data relay
  - Provide QoS (guarantees, reserved b/w, etc.)
    - Allow (dynamic) priorities (inc. time-to-live)
  - Allow choice of transport protocol
    - Support standards (e.g., CCSDS)
- Processing (on-board & distributed)
  - Simplify science processing
  - Support fault tolerance (service management)
  - Simplify off-board processing (like "solver service")
- Storage
  - Provide flexible storage type (e.g., image, meas't, stream)
  - Provide query capability
  - Support management functions (e.g., location, access)
  - Simplify transport (e.g., move, replicate)

Layered Service View

Application Services

Services	Applications	Benefits
Information Request	Navigation, Time, Goals, Meas'ts, Resources	Leveraged resources
Information Supply	Raw: sensor output Processed: science	Higher science value
Information Analysis	Science Analysis vs Goals (Dynamic) data prioritization	Higher effectiveness
Decision Output	Plan negotiation	Higher efficiency

Active Middleware Services

Services	Applications	Benefits
Query	Where things are; What is available	Leveraging
Processing	Appl'n modules; Remote & Local	Fault tolerance
Data Management	Hierarchical storage; Remote & Local	Redundancy Robustness
Messaging	Up (event) Down (QoS)	Automation Autonomy

Comm Services

Services	Applications	Benefits
QoS services	Priority, guarantee, B/w reserv'n	N/w efficiency
Network services	Routing, multicast End-points	N/w optimization
Queues	Channels; Buffer mgt	Transparency
Protocols	CCSDS	Directly accessible Onboard & external

Key concept: Shared Object Model

- Example of JPL's SharedNet (SN) Architecture
  - Information distribution with tight constraints
    - Efficiency, bandwidth, priority, QoS
    - Comm layer handles different transport media
  - V6 in field testing now
- Shared Object Model
  - Client works with local objects (vehicle, sensor, etc.)
    - Create, modify events distributed via publish/subscribe
  - Server maintains current value (or history) for distribution
  - Only attributes and object references are transferred (efficient)
- Higher-layer information processes easily constructed
  - Combinations of lower-layer events, values, locations, etc.
- Compare this middleware to "messaging protocols"…

Potential Middleware Benefits

- Simplified communications
  - Improve use of “local” network bandwidth
    - Higher aggregate capacity than typically “scheduled” for use
    - Lower latency, redundant, fault-tolerant, adaptive, etc.
  - Easily integrate sensor networks
    - Flexible message routing and filtering, sensor integration
  - Improve automation
    - Network “events” can trigger procedures
    - Automated reporting: sensors or health/status of spacecraft
- Simplified applications
  - Simplify use of distributed storage & processing
    - Data processed locally and shared efficiently
    - Software upload/installation (e.g., mods to Galileo s/w)
  - Assist failure discovery/recovery
    - Process restart or migration; application reconfiguration
  - Assist future autonomy
    - More information sources accessible for decisions
      - e.g., terrain, weather, off-board sensors
    - Simplify infrastructure for collaboration (joint planning, etc.)
      - Distributed intelligence; agents

Object Messaging Prototype (FY02)

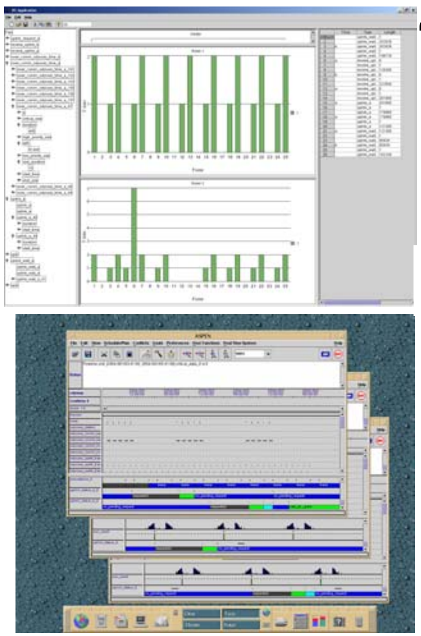
Example Scenario	Shows

Example Scenario

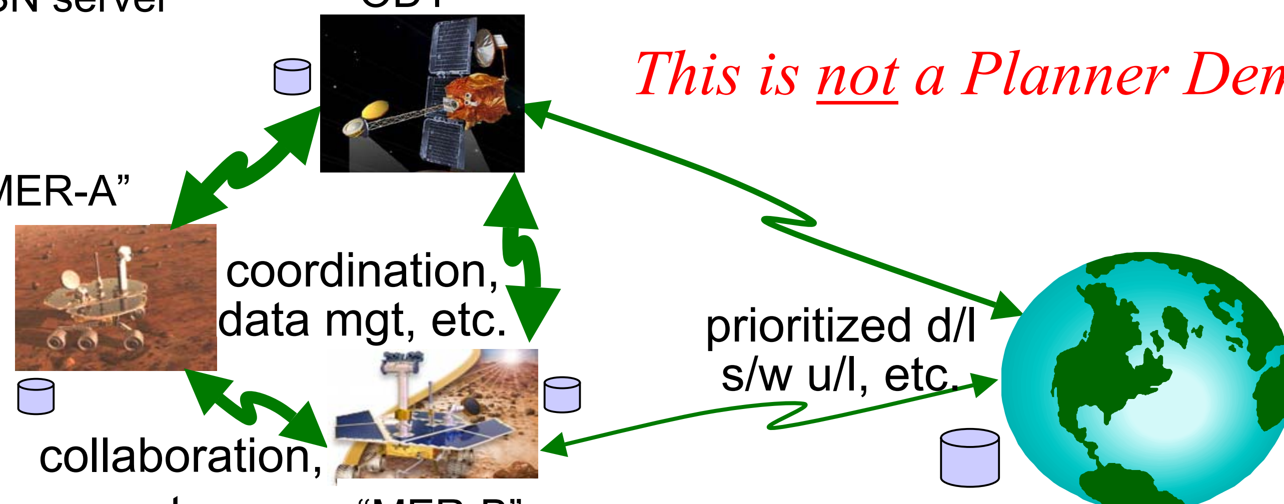
- Remote Planning Coordination:
  - e.g., MER-A/MER-B/ODY
- Plan change by one affects others
  - Time criticality (view periods)
- Negotiations reach a solution
  - Minimal use of link to Earth

Shows

- “Ad-hoc” remote comms
  - Robust MOM: buffered, async, QoS,…
  - Extensible message object model
- “Subscription” by message type
  - Simple client (Java API, C++ wrapper)
- “GUI client” displays filtered traffic
  - Can join “after the fact”

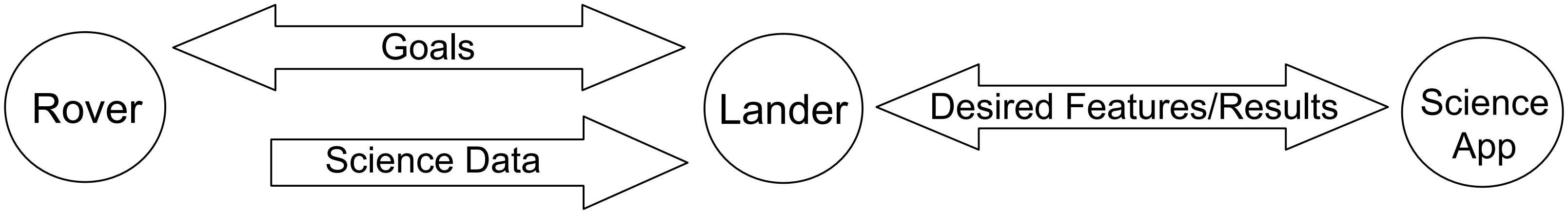


“GUI client” + SN server



This is not a Planner Demo!!

Science Application Prototype (FY03)



Object Model is again key…