Developing an Approach for Analyzing and Verifying System Communication

IEEE Aerospace Conference 2009

William C. Stratton (JHU/APL)
Mikael Lindvall (FC-MD), Chris Ackermann (FC-MD),
Deane E. Sibol (JHU/APL), Sally Godfrey (GSFC)

Johns Hopkins University/Applied Physics Laboratory Space Department Ground Applications Group (JHU/APL)
Fraunhofer Center for Experimental Software Engineering Maryland (FC-MD)
Goddard Space Flight Center (GSFC)

NASA IV&V support through a Software Assurance Research Project (SARP)

Motivation

- Software systems in the aerospace domain…
  - are inherently complex,
  - operate under tight resource constraints,
  - exist in systems of systems that communicate with each other to fulfill larger tasks
- Reliable systems of systems require reliable communications, but ensuring reliable communications is difficult:
  - systems developed independently
  - ambiguities in the specification of expected communication behaviors
  - issues in communications are often subtle and can go undetected
- Communications problems can lead to waste of space link bandwidth and other precious mission resources

Organizational Approach

- NASA IV&V Software Assurance Research Program (SARP)
  - Supports development of software engineering processes and tools
  - Encourages collaboration between researchers and practitioners
- FC-MD researchers develop new processes and tools to address communications problems
- JHU/APL practitioners provide communications scenarios and test data for experimentation
- FC-MD and JHU/APL work as one team, using an iterative process…
  - Experiment with technology; apply to FC-MD testbed
  - Evaluate technology; apply it to APL's ground software systems
  - Improve technology based on feedback, results
  - Repeat
- Emerging processes and tools extend to NASA projects
  - e.g. through the SARP Research Infusion program

Technical Approach

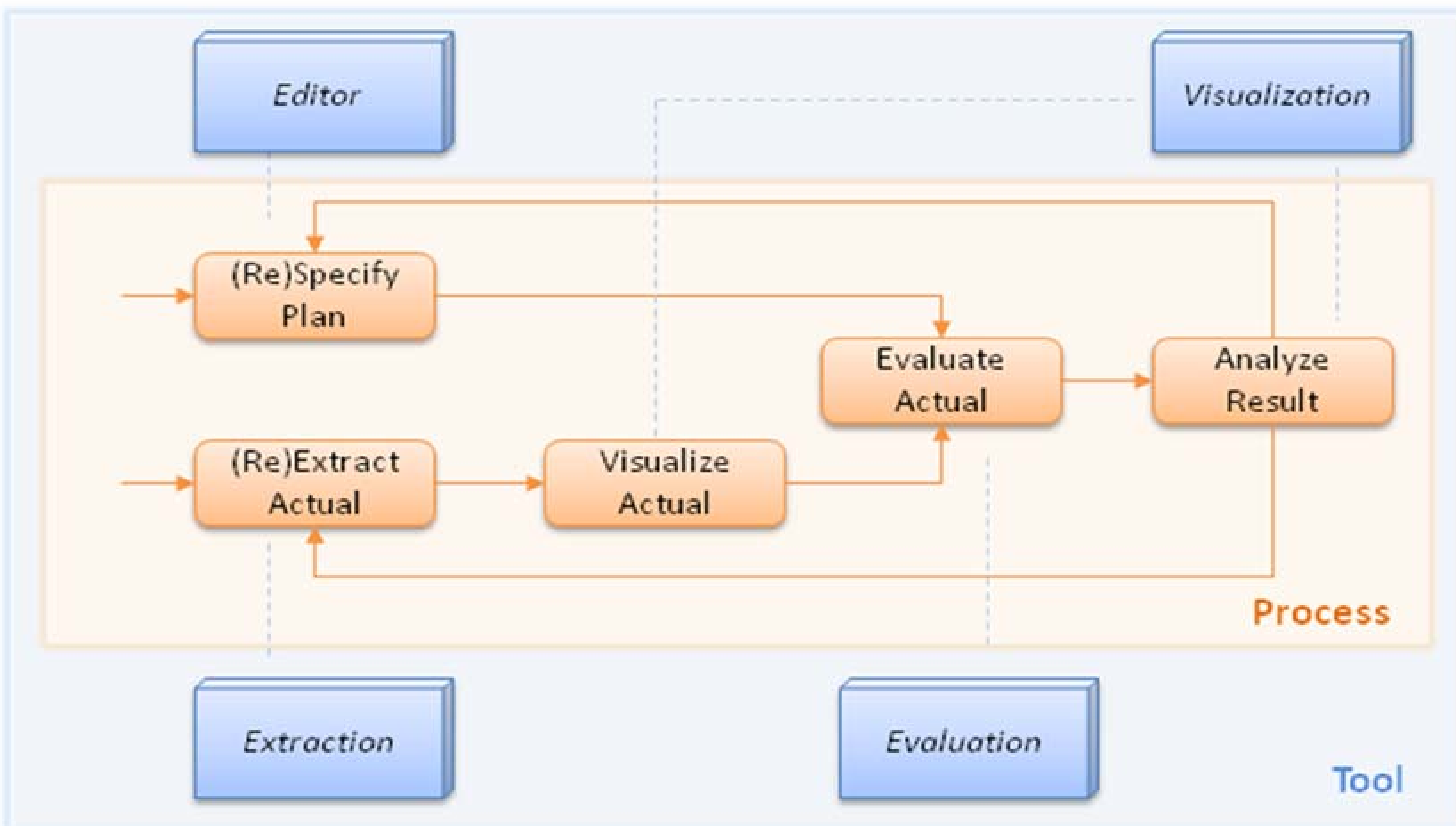
- Develop DynSAVE to detect communications problems among systems by analyzing their communication behavior:
  - Build on Fraunhofer's proven Software Architecture Visualization and Evaluation (SAVE) tool and process for static analysis of source code
  - Enhance for dynamic analysis of run-time communication behavior => Dynamic SAVE (DynSAVE)
- The DynSAVE approach consists of three steps:
  1. Monitor and record low level network traffic
  2. Convert low level traffic into meaningful application messages
  3. Visualize messages such that issues can be detected

Fraunhofer USA, Inc
Center for Experimental Software Engineering Maryland

SAVE Tool and Process

- SAVE supports static analysis:
  - software architect creates models of the planned relationships among abstract software components
  - SAVE tool parses source code and lifts the actual relationships among concrete software components
  - SAVE tool annotates the architect’s models to show deviations from the plan
  - software architect uses the SAVE tool to explore the deviations, drilling down through the annotations to the source code
  - source code and/or model are updated to eliminate the deviations
- JHU/APL and FC-MD have infused SAVE into the ground software development process:
  - used to analyze changes to legacy Common Ground software
  - incorporated into new software development for next generation of JHU/APL ground software systems beginning with Radiation Belt Storm Probes (RBSP)

SAVE
Tool and Process

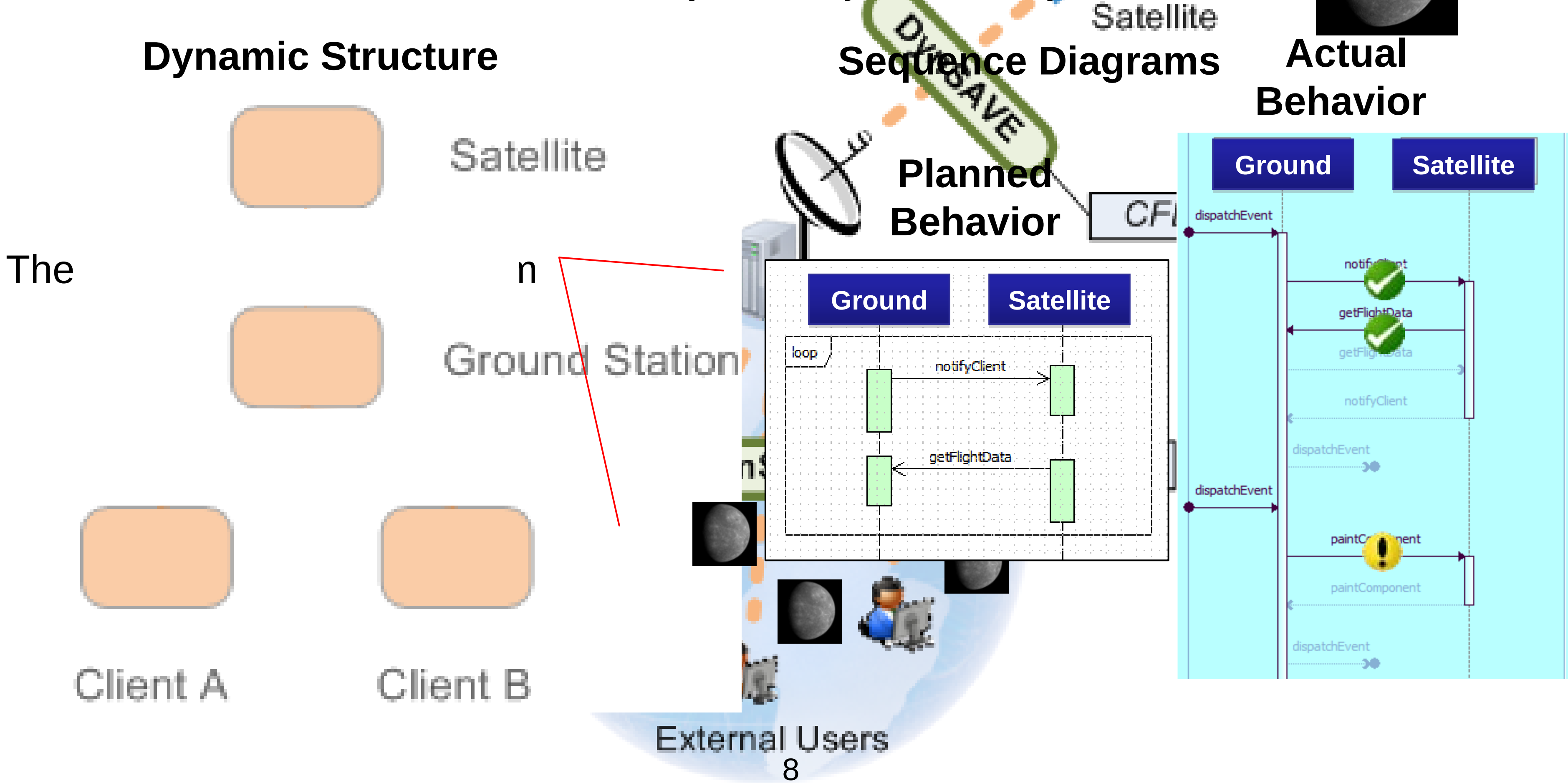


DynSAVE Tool and Process

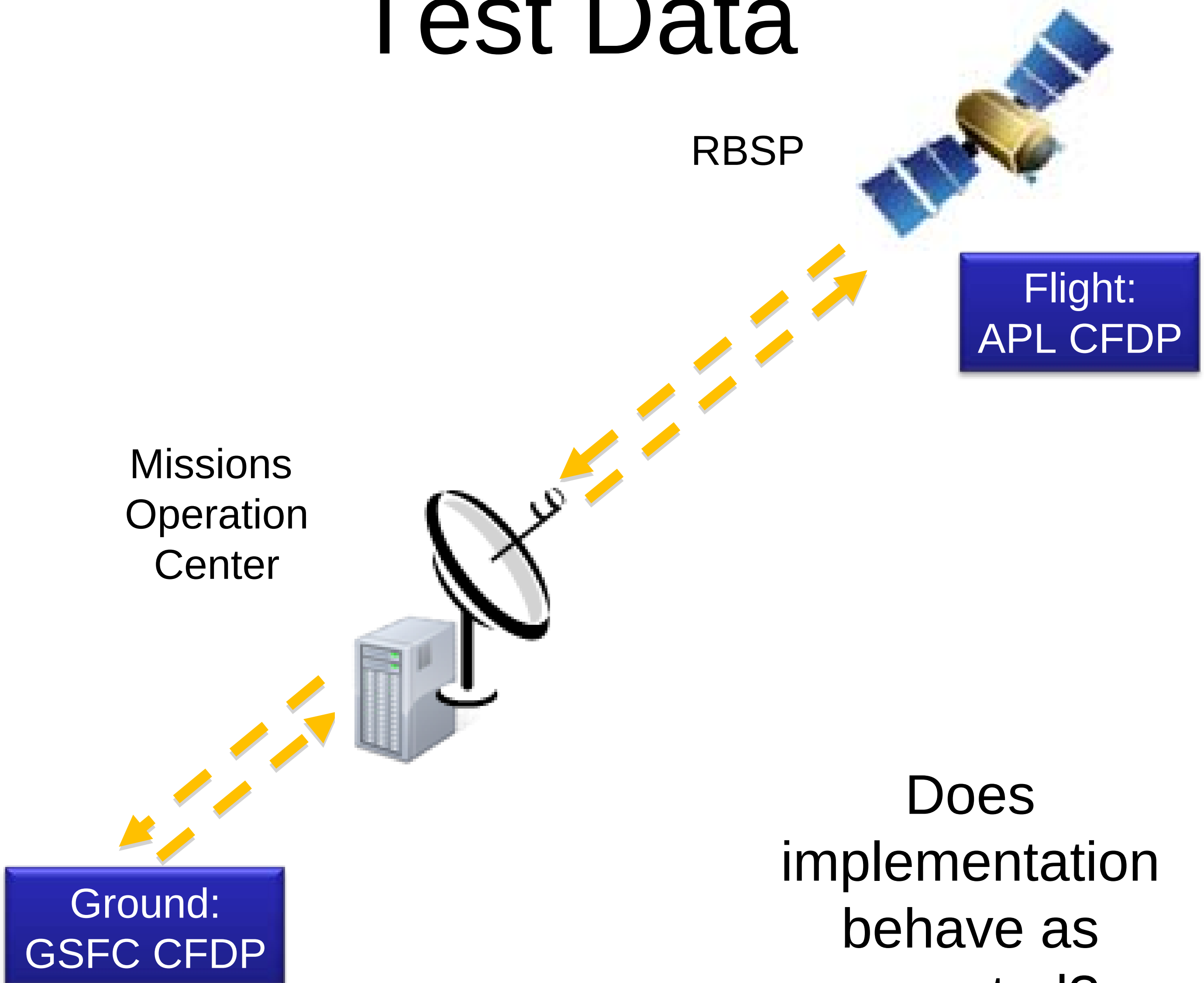
- DynSAVE extends SAVE to support dynamic analysis:
  - software architect creates models of the planned message sequences among abstract systems
  - actual messages are captured from network traces or low level communications archives
  - DynSAVE tool parses captured messages and lifts the actual message sequences among concrete systems
  - DynSAVE tool annotates the architect’s models to show deviations from the plan
  - software architect uses the DynSAVE tool to explore the deviations, drilling down through the annotations to the messages
  - systems and/or model are updated to eliminate the deviations
- JHU/APL and FC-MD have applied DynSAVE to mission data systems:
  - used to analyze legacy Common Ground software client/server communications (Aerospace 2008)
  - currently analyzing CCSDS File Delivery Protocol (CFDP) communications behaviors in RBSP and MErcury Surface, Space ENvironment, GEochemistry, and Ranging (MESSENGER)

DynSAVE Approach to CFDP

Dynamic SAVE allows for structural and behavioral
Architectural analysis of systems of systems

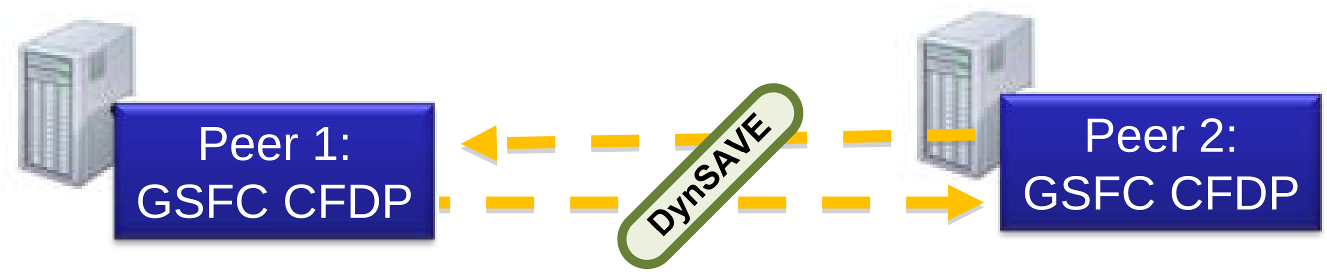


Test Data



Does
implementation
behave as
expected?

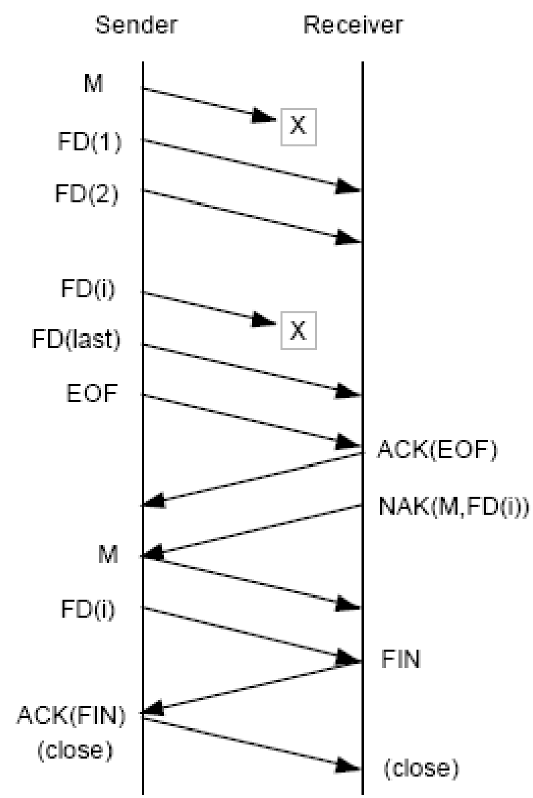
Test Data



Does implementation behave as expected?

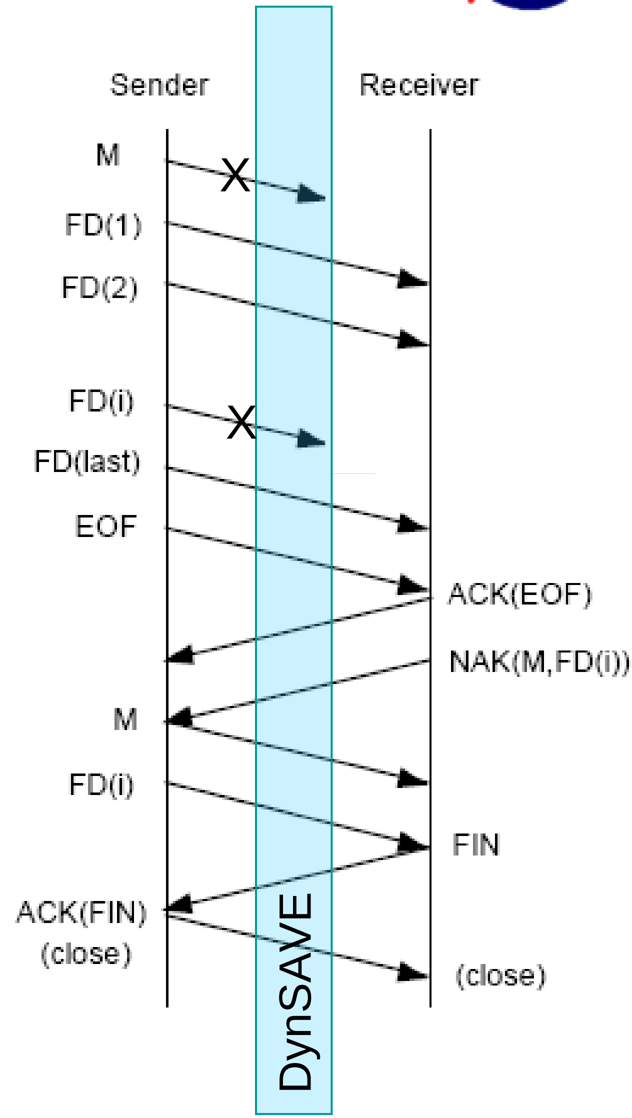
CFDP – A Mission Data System Protocol

- CFDP software provides reliable downloads of recorded on-board data
  - The implementation is distributed across flight and ground systems
  - The protocol runs on top of unreliable CCSDS command and telemetry layer
- At APL, CFDP is mostly automated, but…
  - Operators turn off CFDP uplink during critical command load sequences
  - Operators freeze and thaw timers so that pending transactions don't time out between contacts
- Improper CFDP operation can lead to unnecessary retransmissions, wasting precious downlink bandwidth

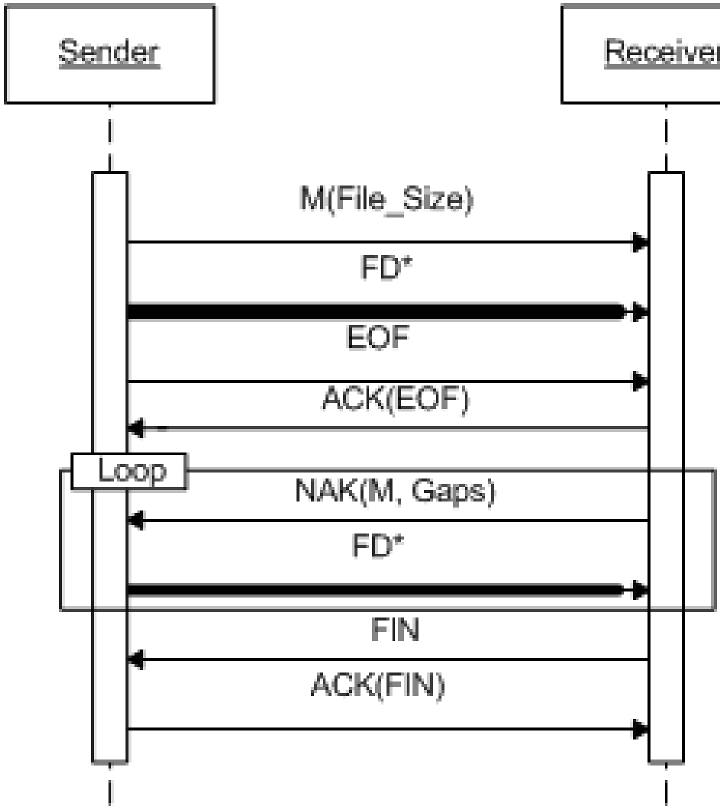


DynSAVE monitoring of CFDP

- DynSAVE monitors macro-level behaviors of the CFDP protocol without affecting flight or ground software
- DynSAVE could detect behaviors that are indicative of improper CFDP operation, for example:
  - timers were not frozen and uplink was disabled on the ground for an extended period, causing multiple retransmissions when the uplink was finally enabled again
- DynSAVE could detect behaviors that are indicative of issues in CFDP implementation, for example:
  - sender continues to send file data after the transaction has been cancelled
- These types of behaviors can go undetected (file transfers still work) but are important to detect (they can result in data loss!)



Planned CFDP Sequence



Rules:
1. Check that received FD are not NAKed *
2. Check for duplicate FDs *
3. Check that we have all FDs upon FIN *
4. Check that identical NAKs are not sent back-to-back unless timer went off

Actual CFDP Sequence

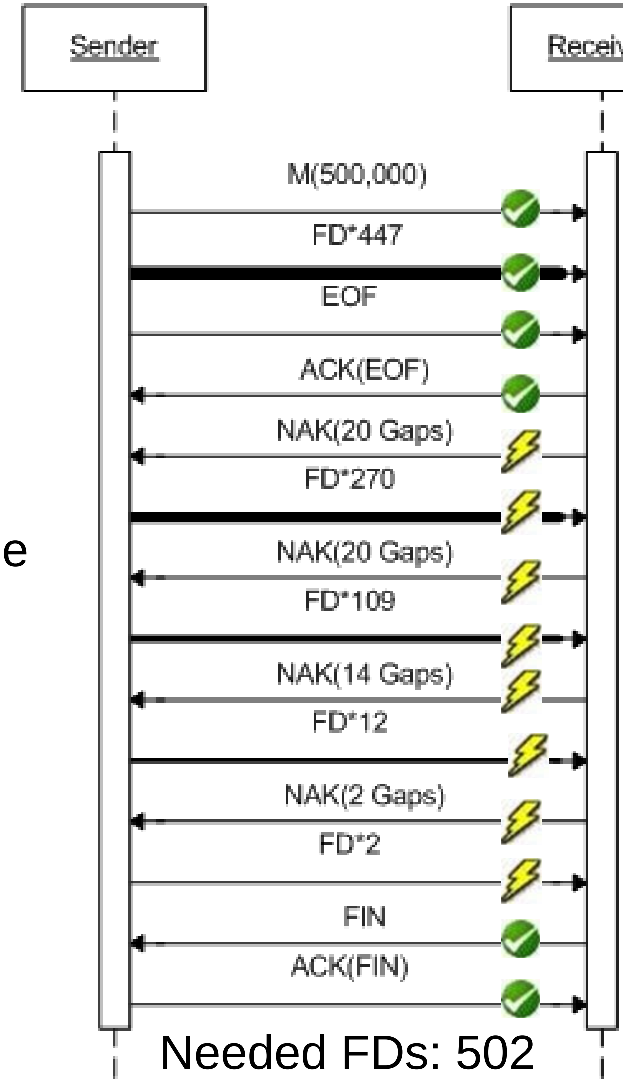
```
Metadata: 0-499999
FileData: 0-996
FileData: 997-1993
FileData: 1994-2990
FileData: 2991-3987
FileData: 3988-4984
FileData: 4985-5981
FileData: 5982-6978
FileData: 6979-7975
FileData: 7976-8972
FileData: 8973-9969
FileData: 9970-10966
FileData: 10967-11963
FileData: 11964-12960
FileData: 12961-13957
FileData: 13958-14954
FileData: 14955-15951
FileData: 15952-16948
```

```
FileData: 482548-483544
FileData: 483545-484541
FileData: 484542-485538
FileData: 485539-486535
FileData: 486536-487532
FileData: 487533-488529
FileData: 488530-489526
FileData: 489527-490523
FileData: 491521-492517
FileData: 492518-493514
FileData: 493515-494511
FileData: 494512-495508
FileData: 495509-496505
FileData: 498500-499496
FileData: 499497-499999
EOF: Condition Code=No Error
ACK(EOF): Condition Code=No Error
NAK: 19940-20937;27916-28913;36889-37886;56829-
59820;72781-73778;76769-77766;82751-85742;101694-
102691;111664-112661;115652-116649;121634-
122631;130607-131604;139580-140577;146559-
147556;153538-154535;155532-156529;170487-
171484;197406-198403;203388-204385;220337-498500
```

Mapping CFDP data

- The sniffed CFDP data is low level (packets)
- Concepts are often encoded
  - Few message names in clear text
  - Many are not: e.g. Cancel
    - If third bit in EOF control message then Cancel
- Parameters are always encoded
  - E.g. bit 4 – 16: Time stamp
- Communications are often interleaved
  - E.g. Files sent and received concurrently
- Our parser maps low level data to high level messages and values, identifies & separates interleaved communications

Actual CFDP Sequence captured in test lab

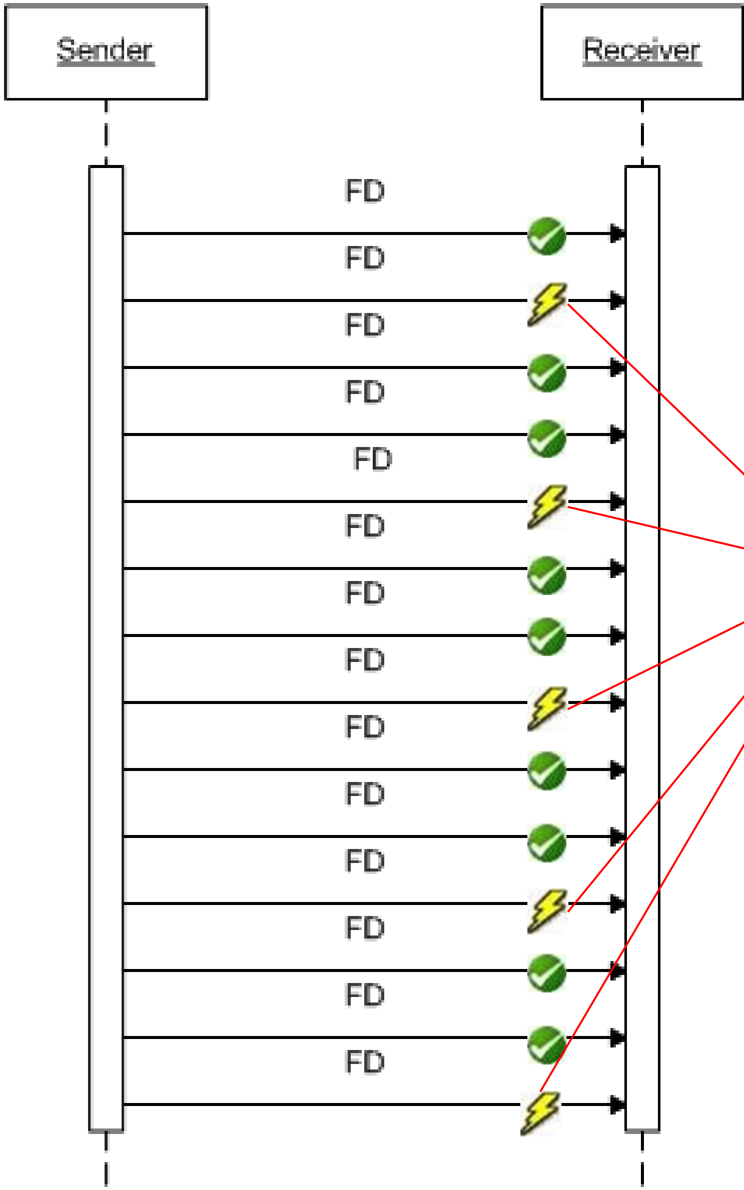


Sample Rule:
Never re-request a package that already was received

Conclusion:
Deviates from specification for certain configurations!
Decision: Use, but with different configuration

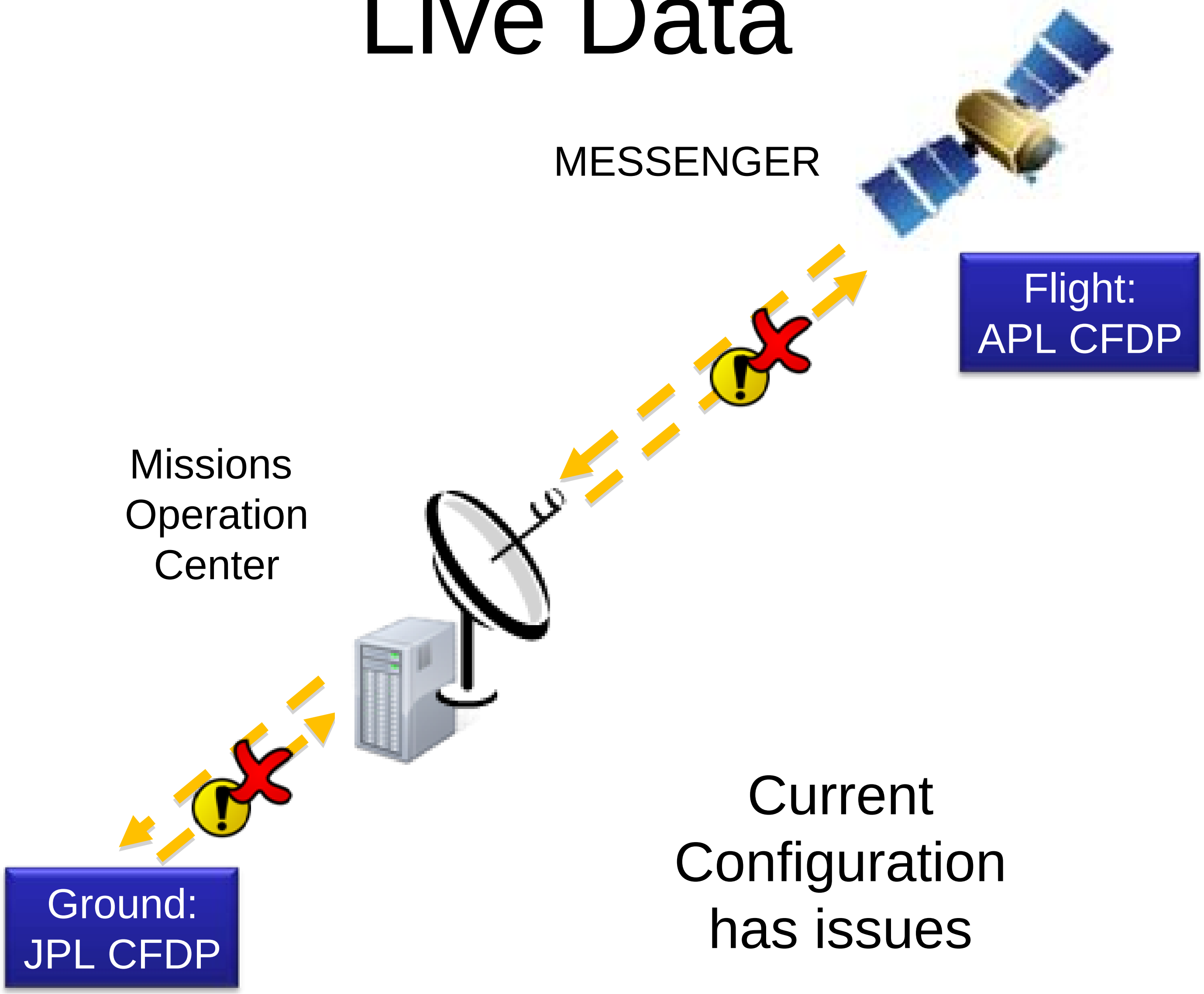
Needed FDs: 502
Sent FDs: 840
Potential Waste: ~70%? – Further analysis needed.

Zoom in on CFDP sequence



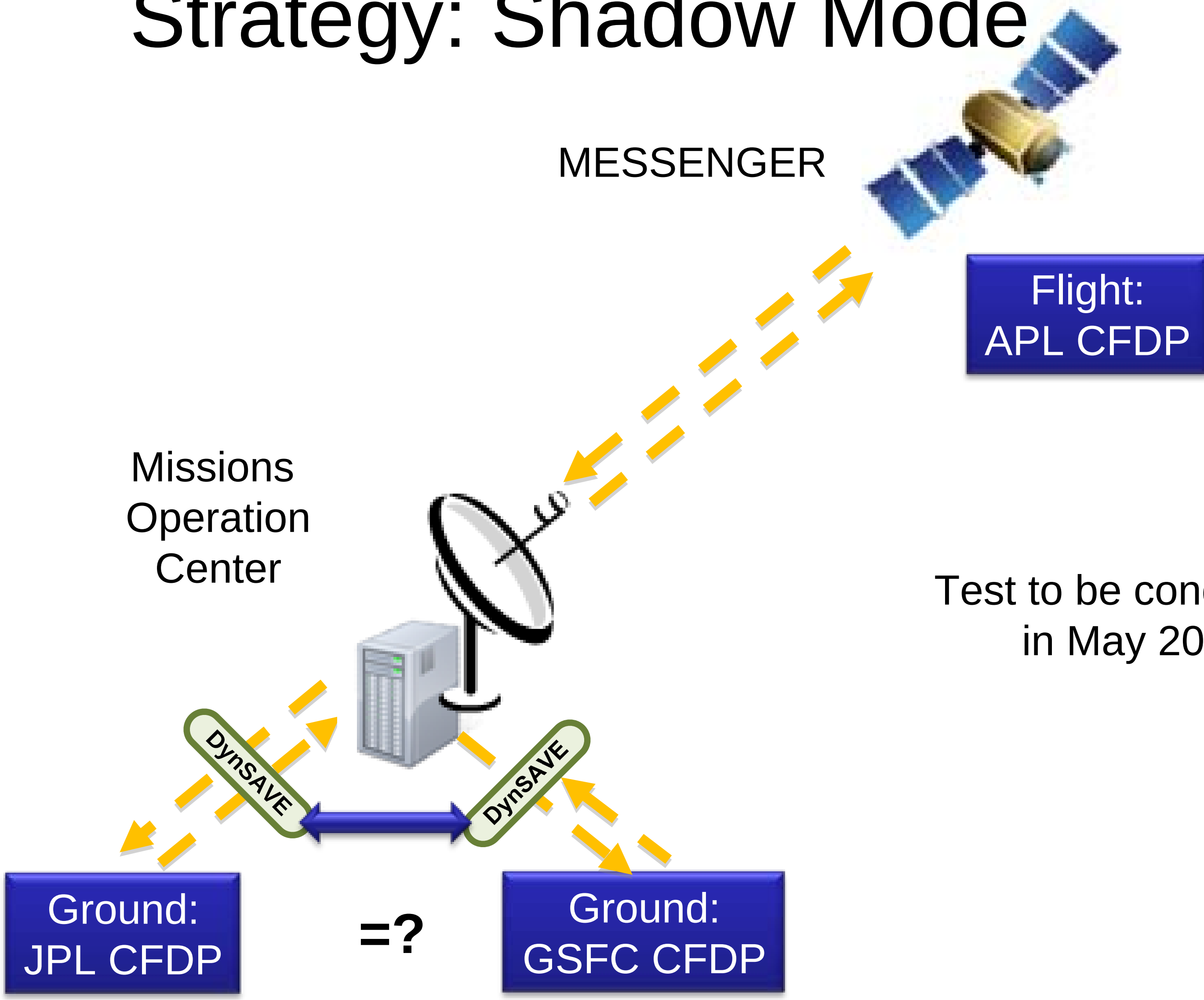
Rule 2 Violation: duplicate FD!

Live Data



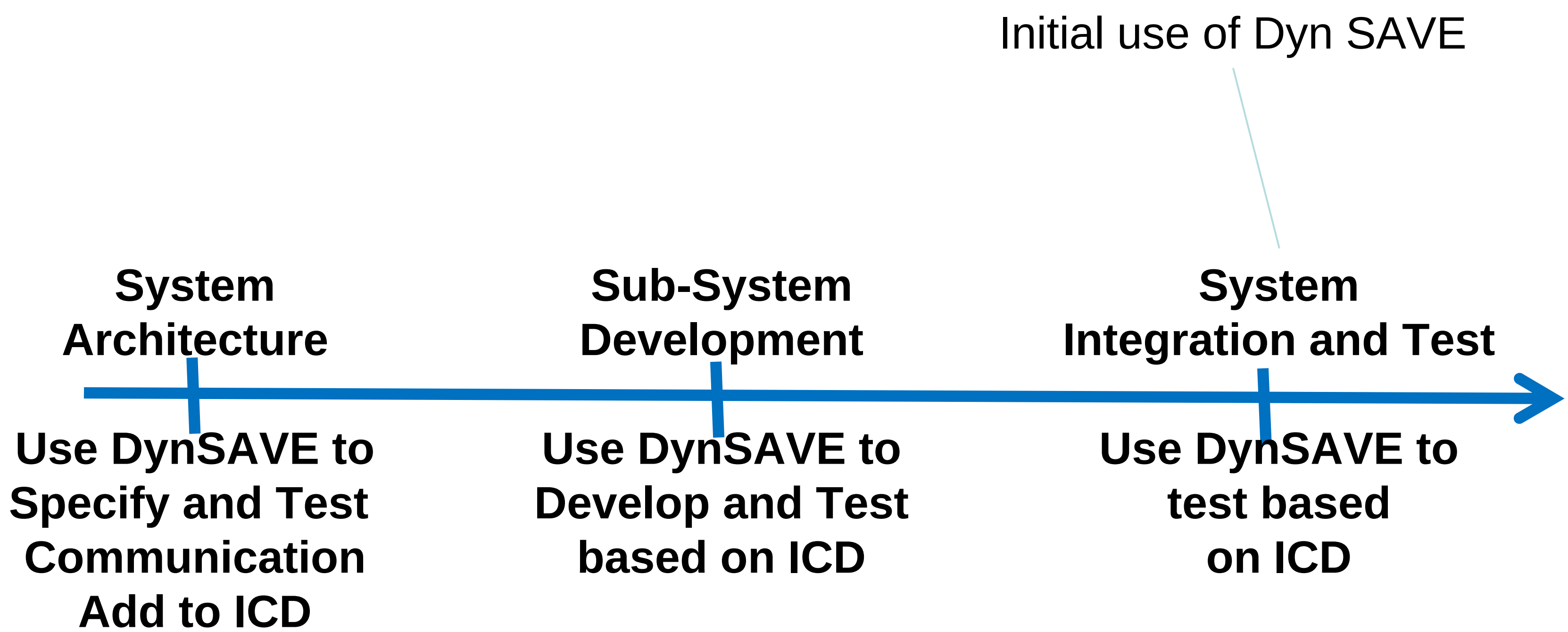
Current Configuration has issues

Strategy: Shadow Mode



Test to be conducted in May 2009

Life Cycle Support



Summary

- Analyze, Visualize, and Evaluate
  - structure and behavior using static and dynamic info of
  - individual systems as well as systems of systems
- Drive R&D by needs from JHU/APL NASA missions
  - Use open testbed for experimentation
  - Evaluate together with APL in their context
- Transfer technology when mature
- Future:
  - Add time information and constraints (current activity)
  - Add planned sequence diagrams to ICD
  - Use for analysis of Delay Tolerant Network Management