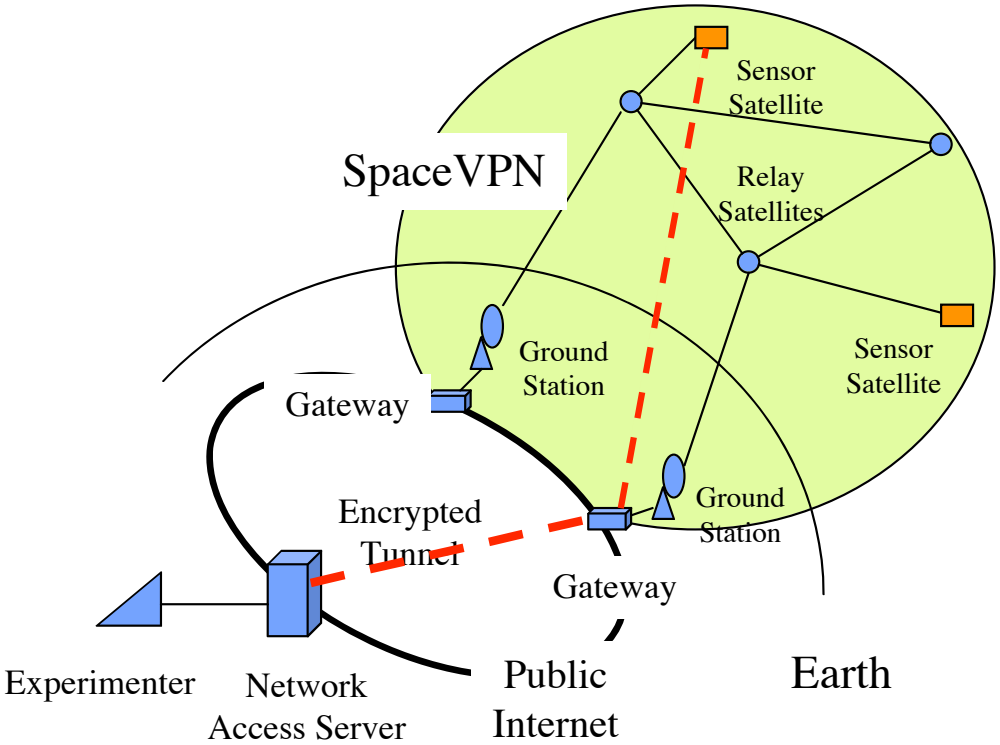
BBN TECHNOLOGIES

Multi-Spacecraft Network for Ground-Space VPNs: Architecture, Radio Specifications and Testbed



ESTC 2006

June 28, 2006

Dr. Marcos A. Bergamo*, Principal Investigator

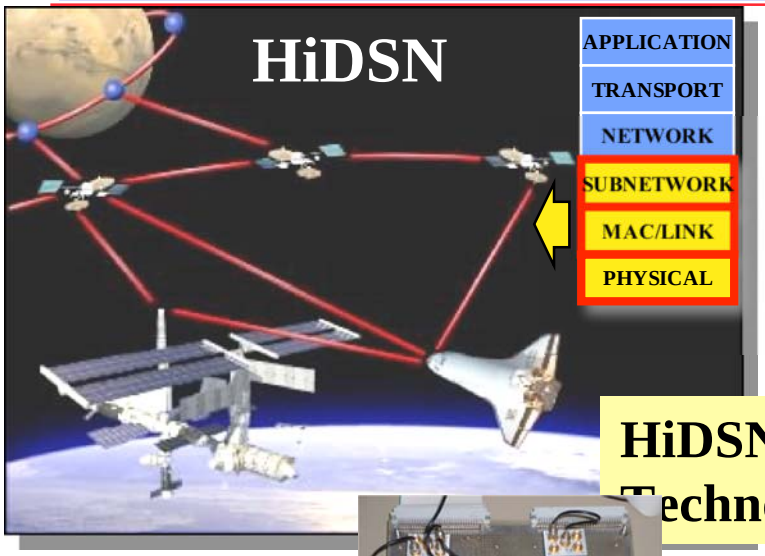
Dr. David Lapsley, Network Scientist

* US Space Technology Hall of Fame (1997)
IEEE Judith Resnik (2005)

NASA ESTO
Earth Science Technology Office

SpaceVPN Project At a Glance

This presentation

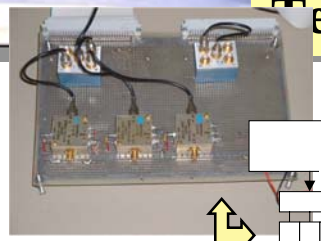


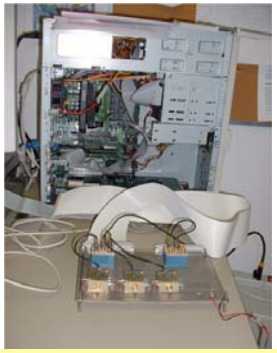
Project Tasks

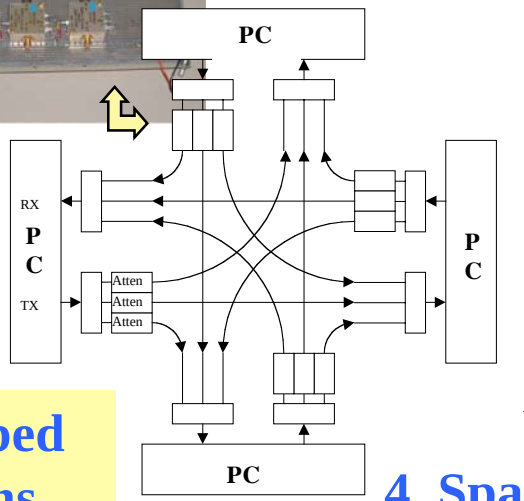
1. SpaceVPN Architecture
2. Laboratory Testbed
3. Reference Specifications
4. End-to-End Secure SpaceVPN Demo

Year 2 Focus

HiDSN Architecture, Technologies and Protocols

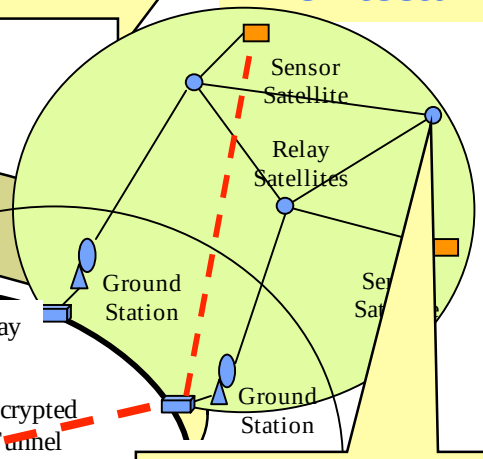




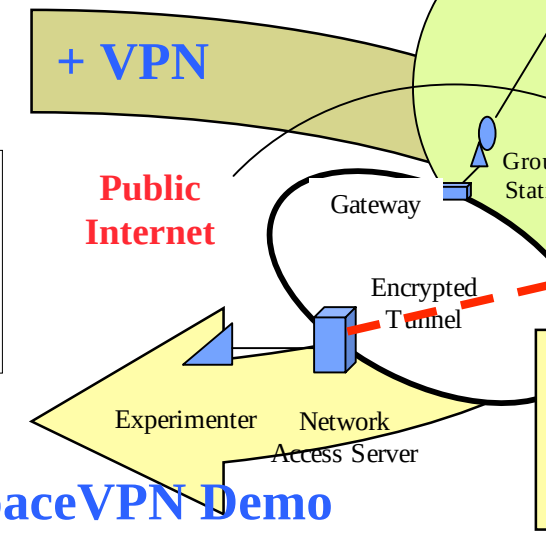


2. Laboratory Testbed with Dynamic Beams, Mobility Emulator

1. SpaceVPN Architecture



+ VPN

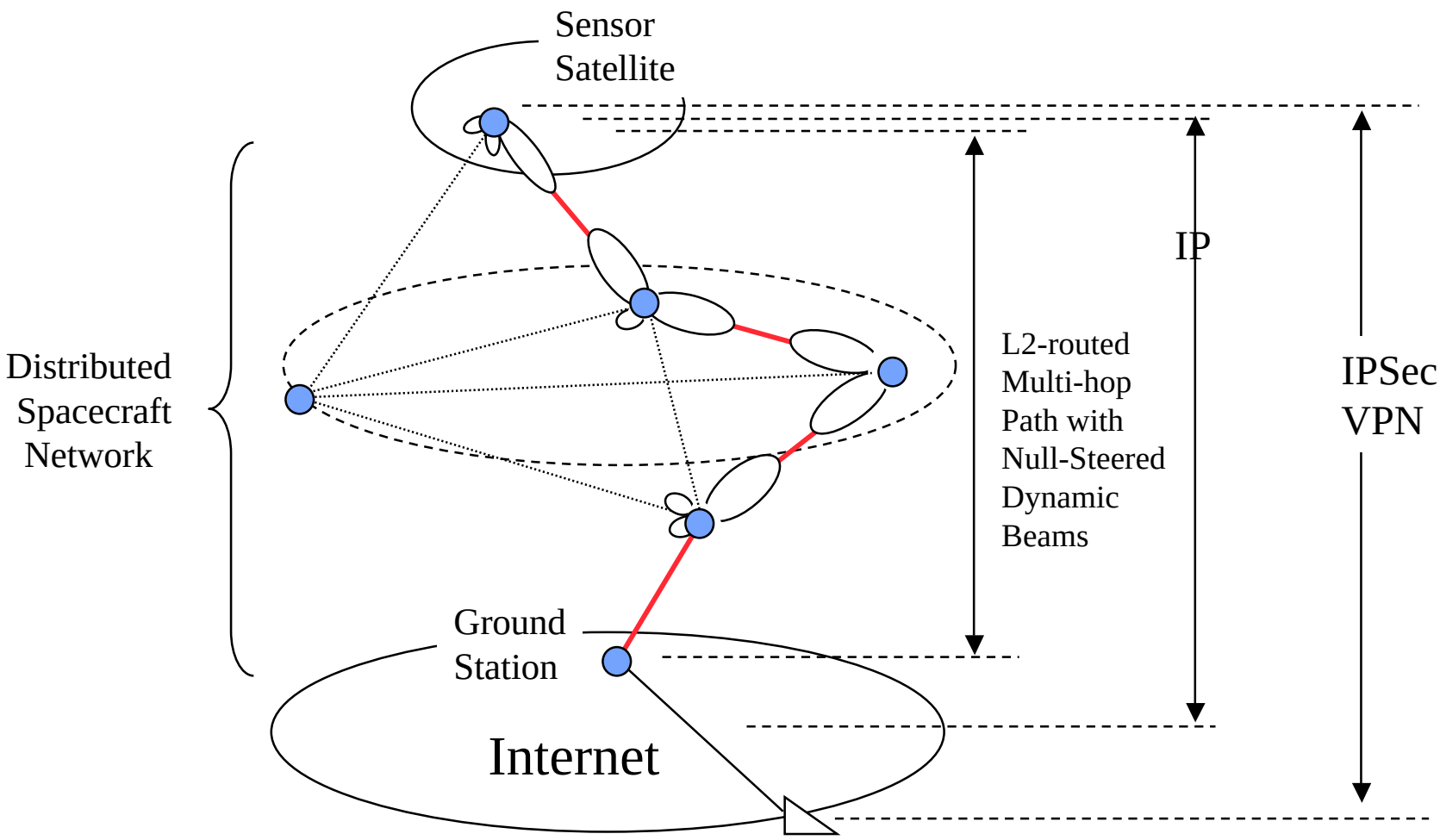


4. SpaceVPN Demo

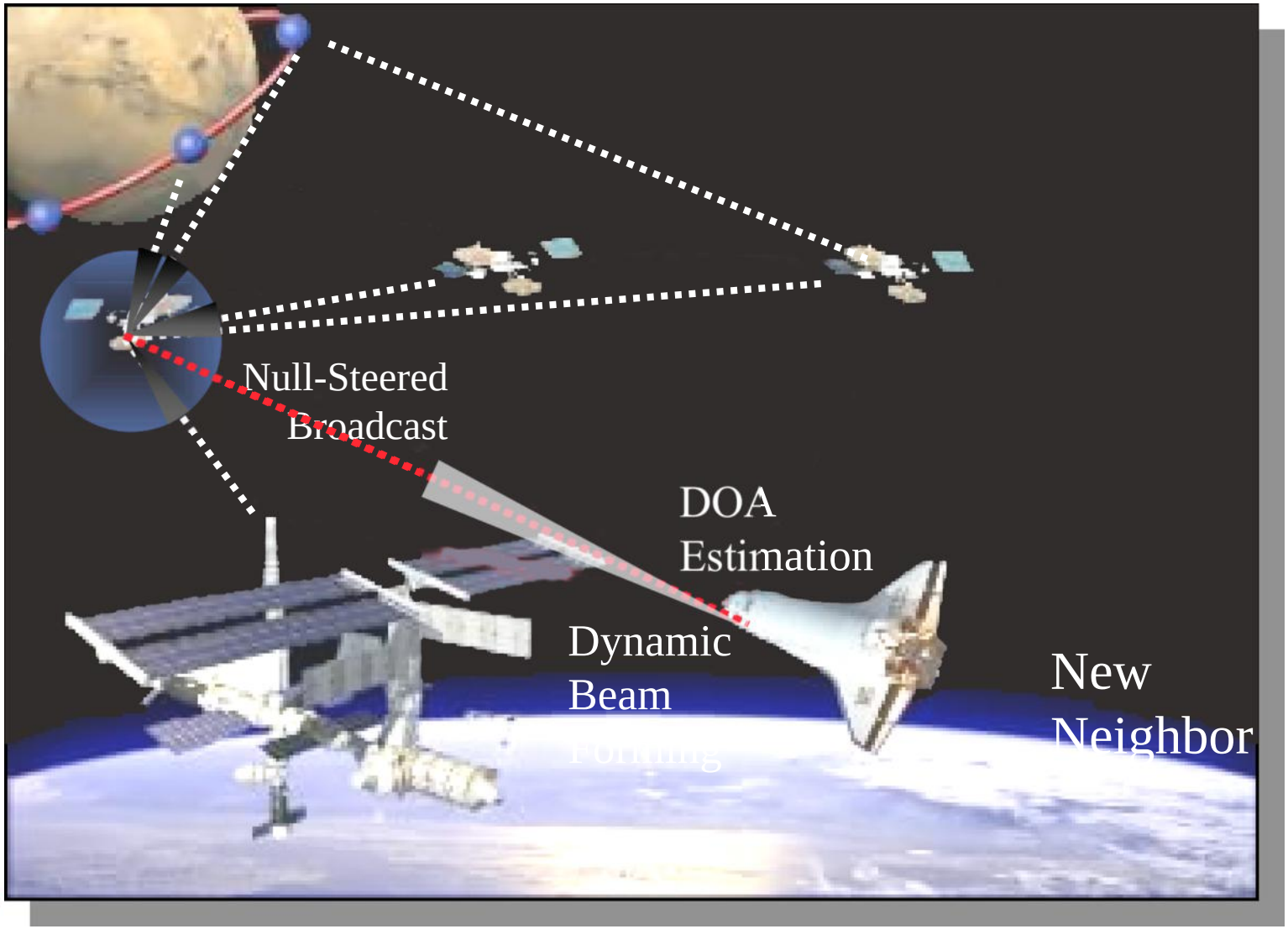
3. Reference Payload Specifications

Space VPN Architecture

Typical Ground-Space Access

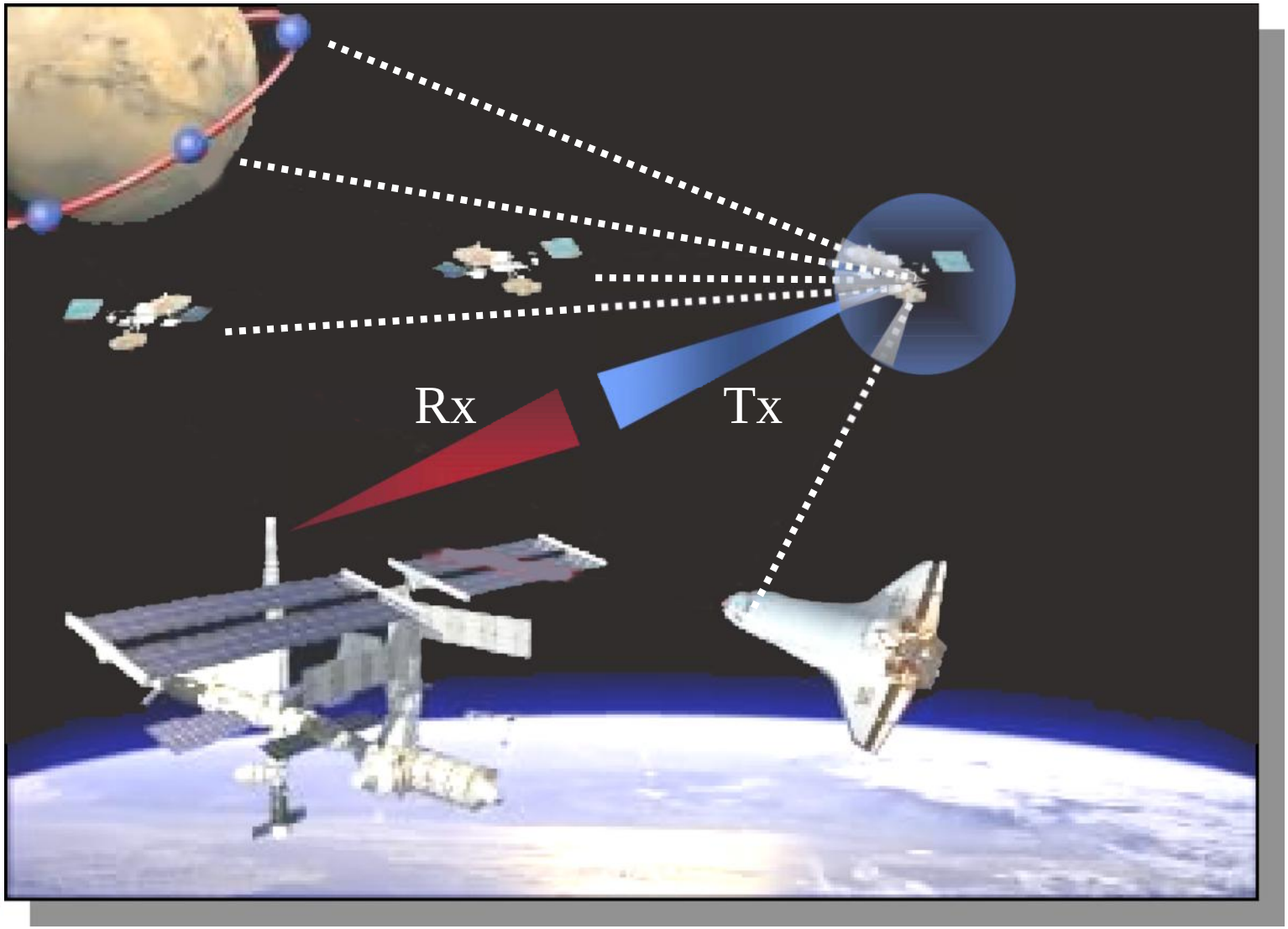


Step 1: Establish Links Using Dynamic/Position-Based Neighbor Discovery



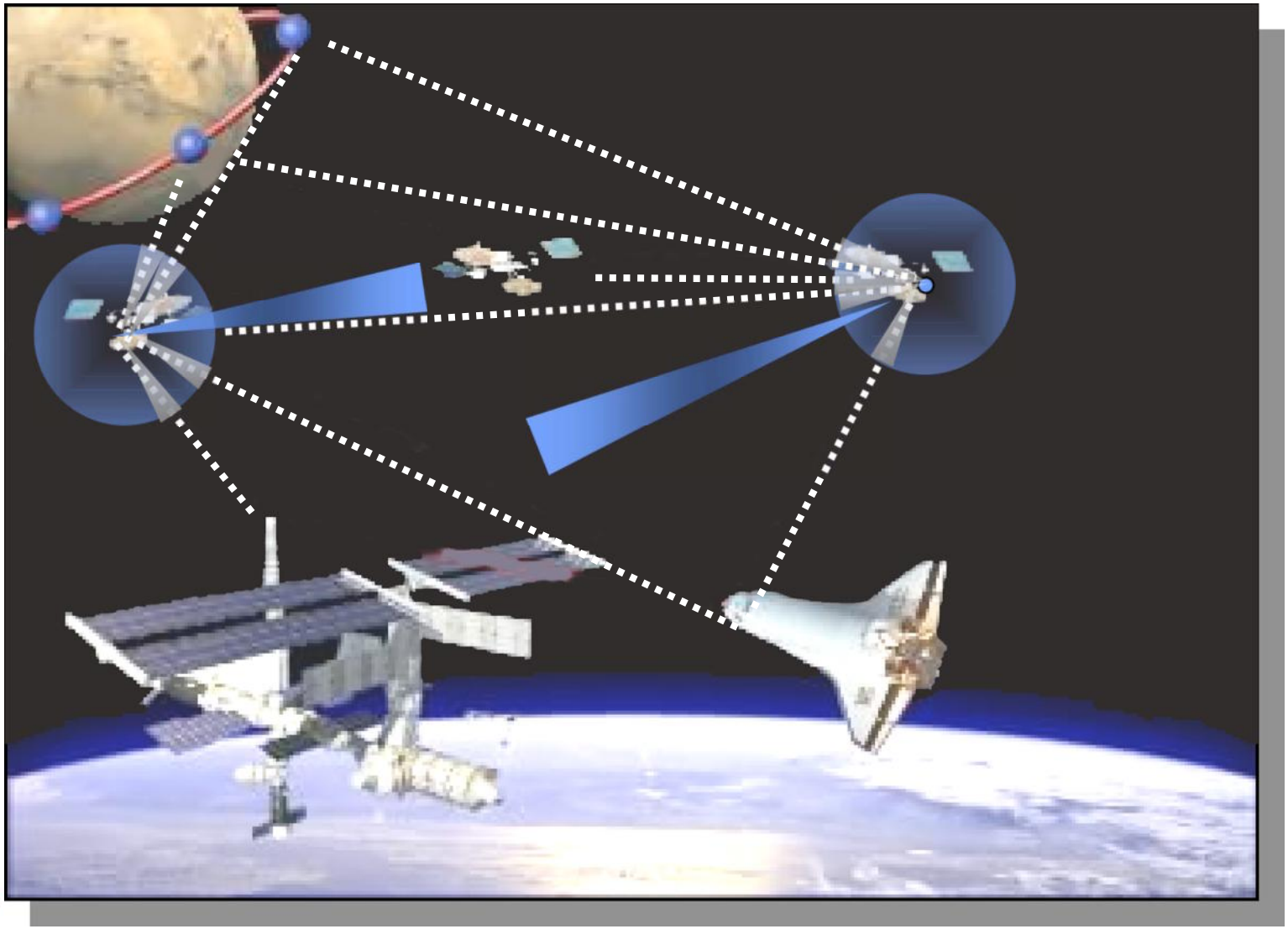
BBN TECHNOLOGIES

Step 2: Transmit at a high-data-rates using high-gain null-steered beams without causing interference to neighbor nodes



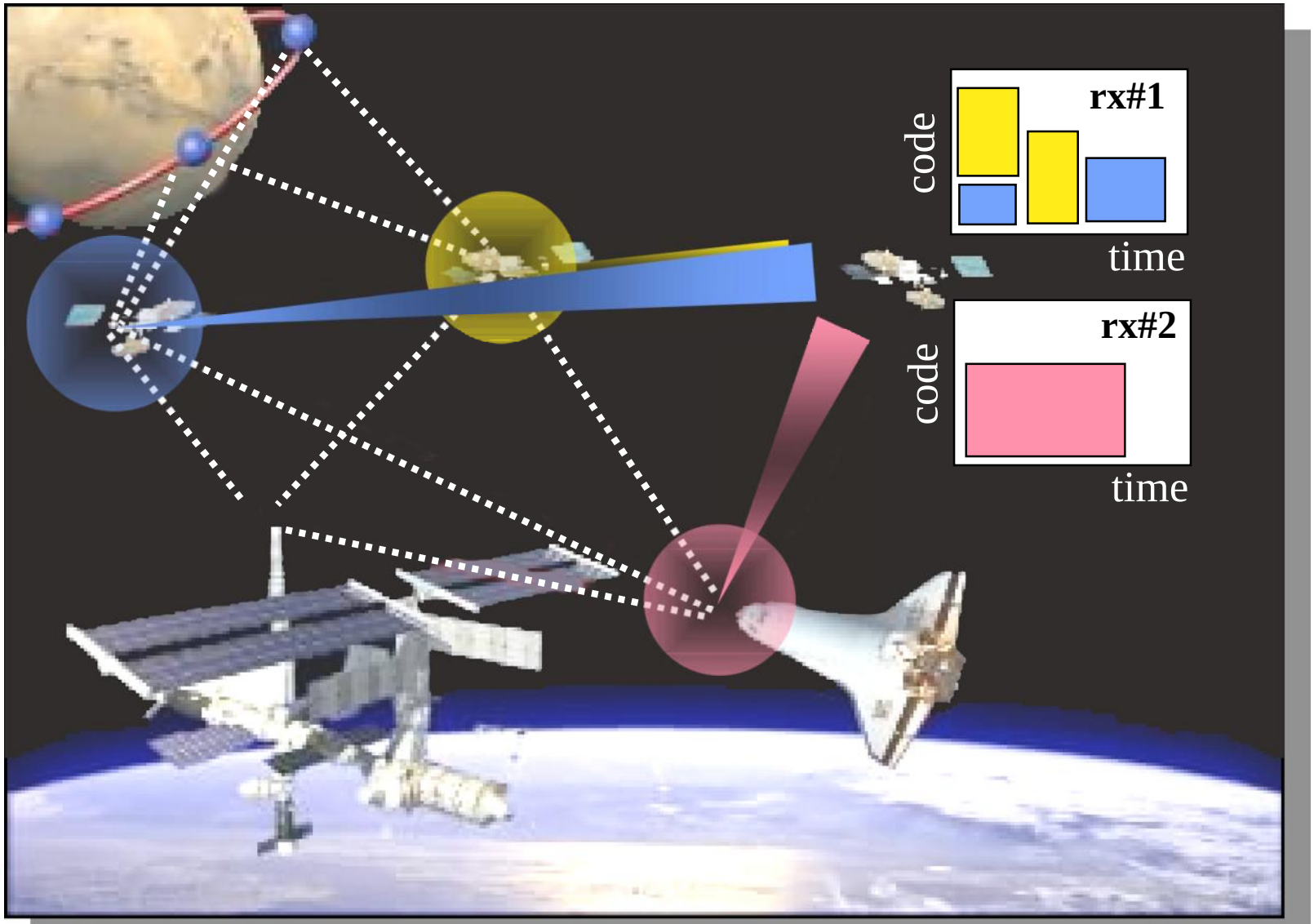
BBN TECHNOLOGIES

Step 3: Achieve multiple times over reuse of the spectrum with spatial multiplexing



ESTO
Earth Science Technology Office

Step 4: Adapt to relative movement by integrating time, code and spatial multiplexing to close the links as needed, avoid interference, meet QoS requirements and maximize throughput



PHY- Layer

Key Physical-Layer Technologies

Variable-Rate TCeMA Encoding

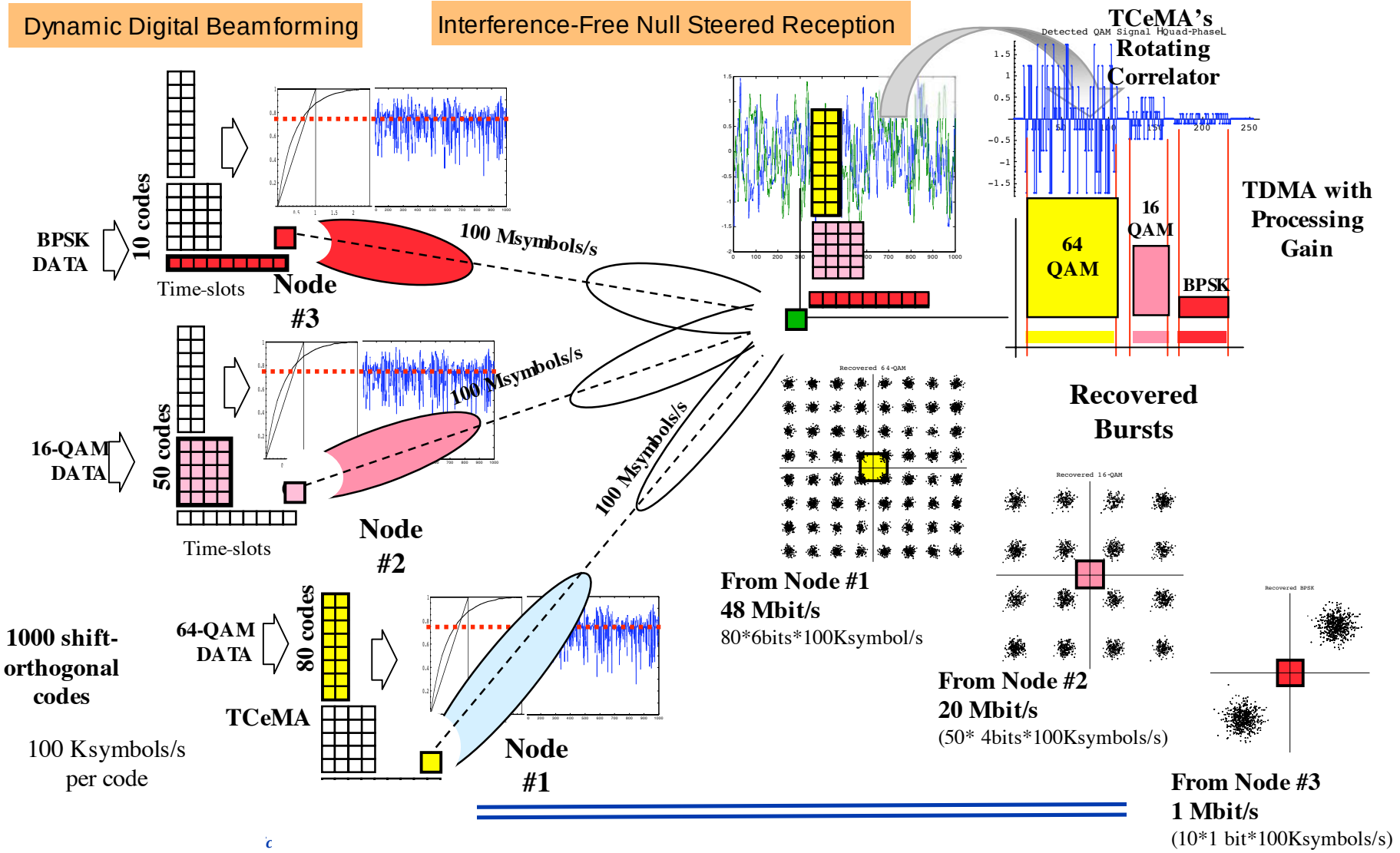
M-Ary QAM Modulation

Constant-Envelope Split-Phase Shift Keying Modulation

Interference-Free Orthogonal Code Decoding

Dynamic Digital Beamforming

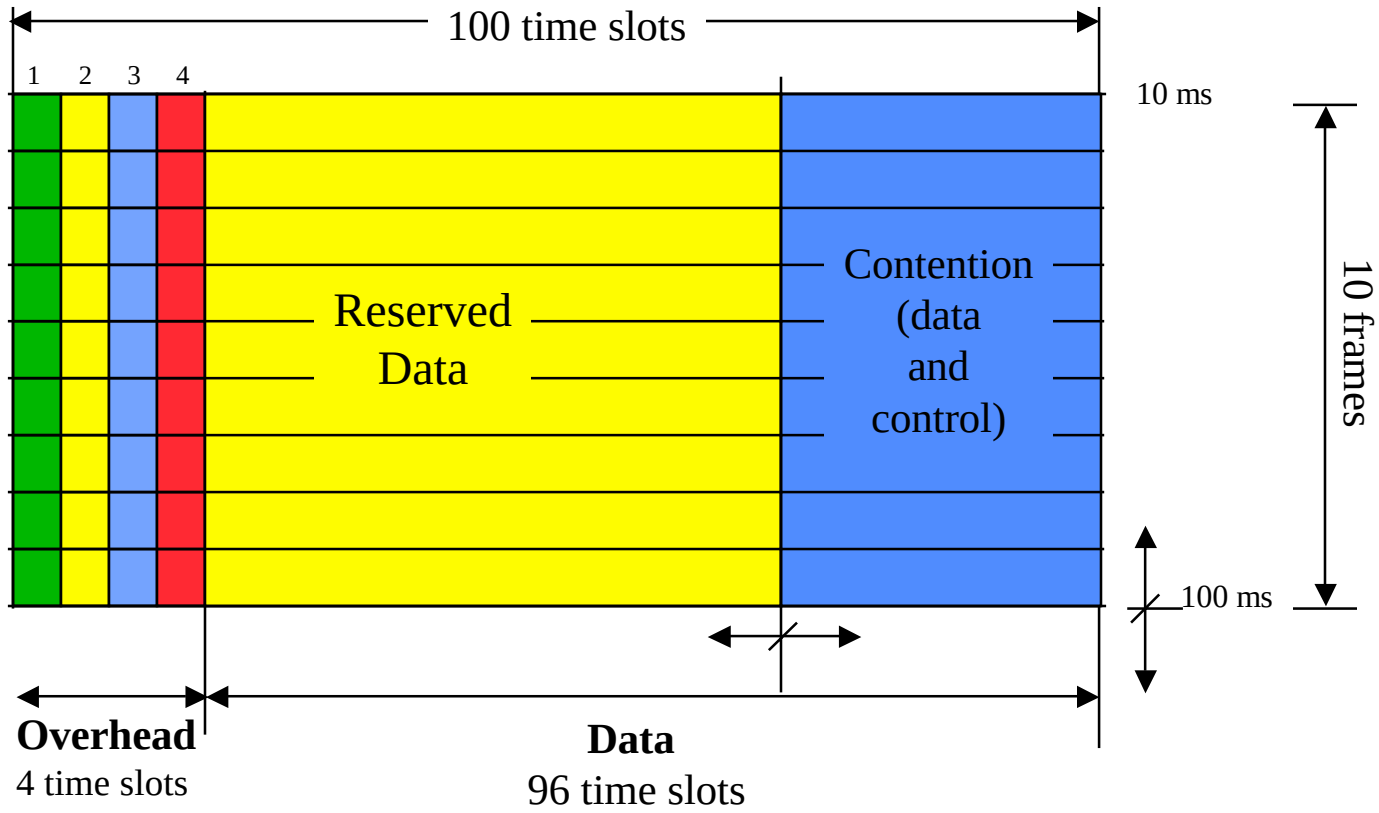
Interference-Free Null Steered Reception



System Frame

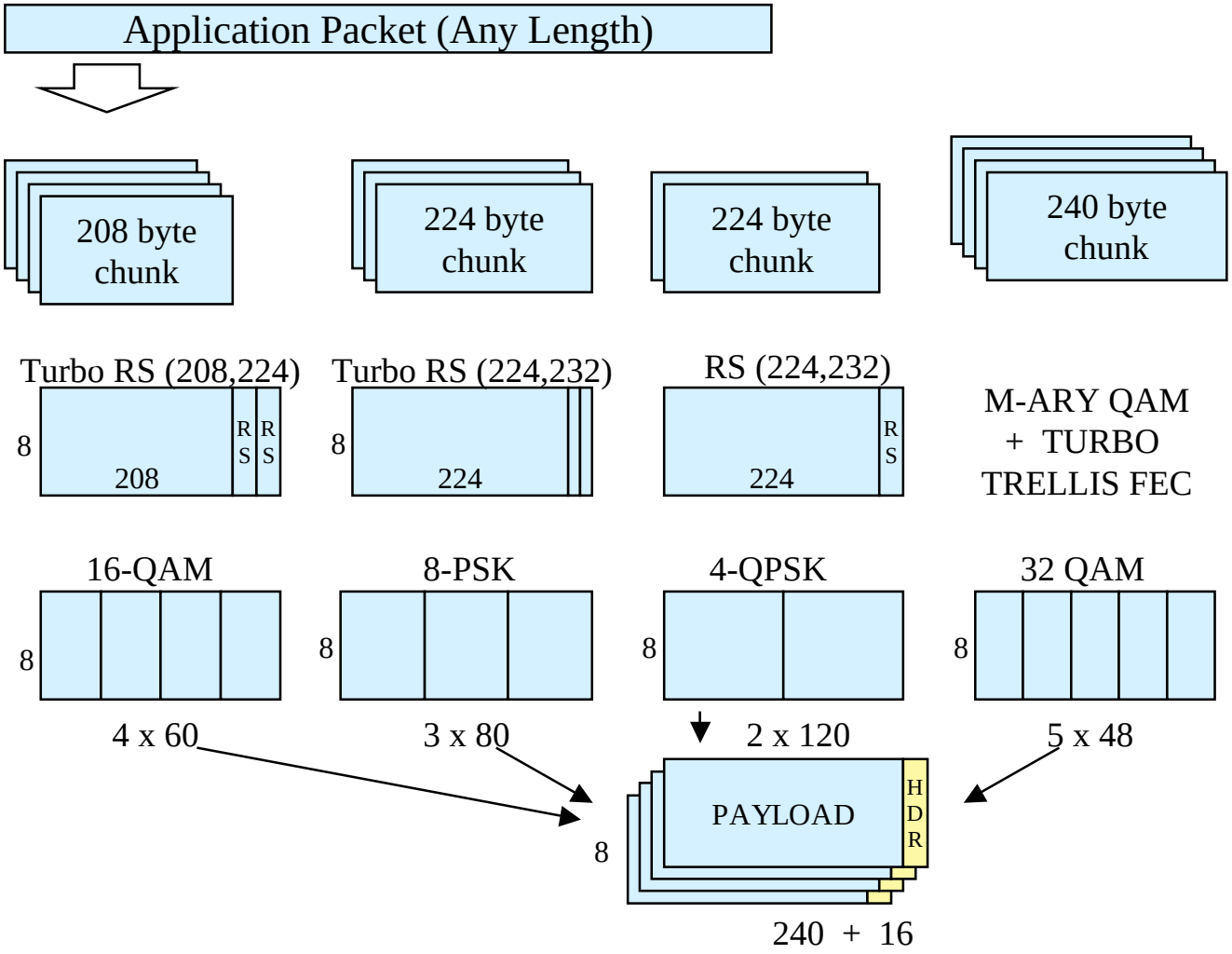
HiDSN Frames are Receiver-directed

HiDSN Frame capacity is used mostly for data (96%) and includes a small overhead area (4%) for neighbor discovery, node synchronization and burst transmission control

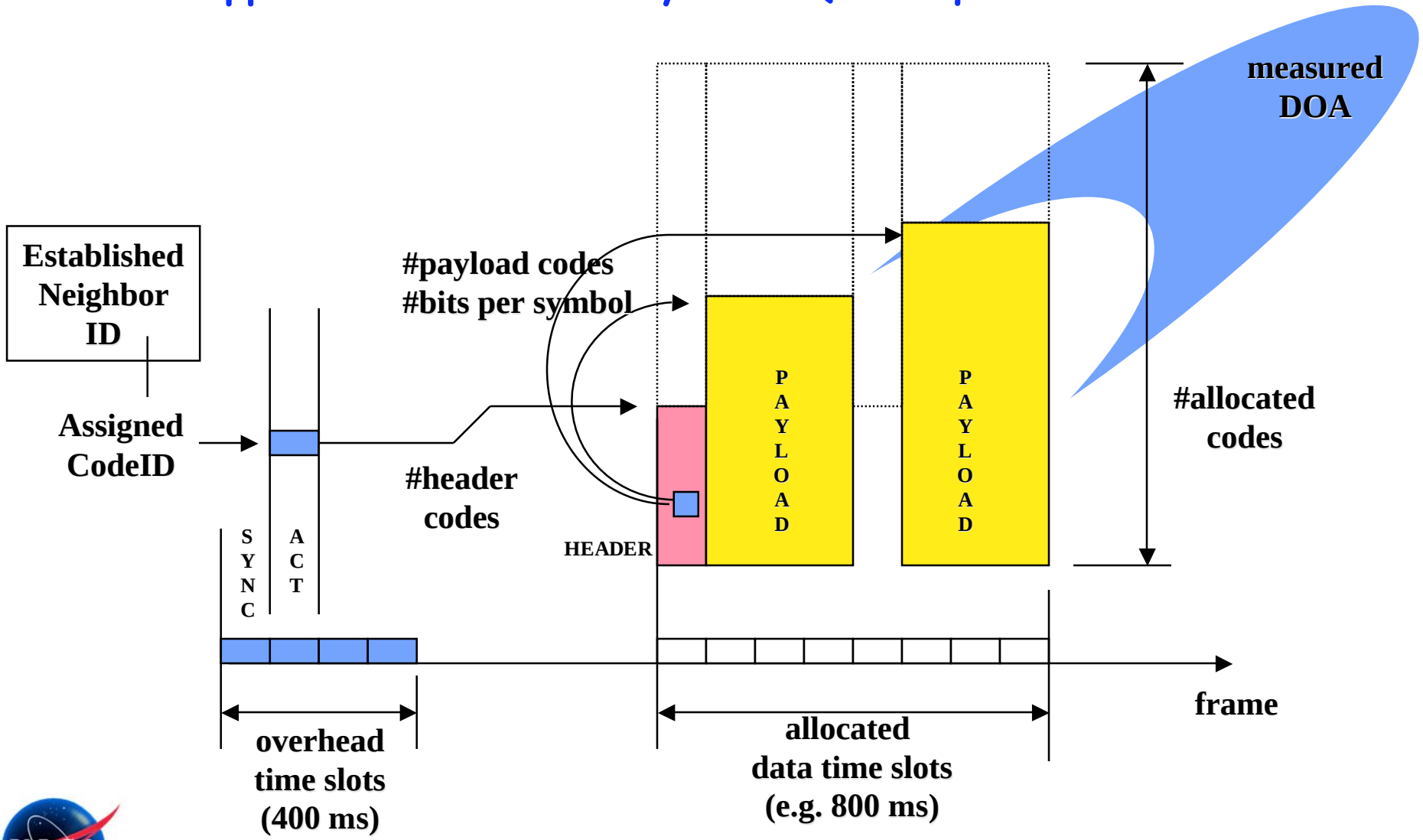


1. **SYN**Chronization
2. **ACT**ivity
3. **CONT**ention (look ahead)
4. **DISC**overy

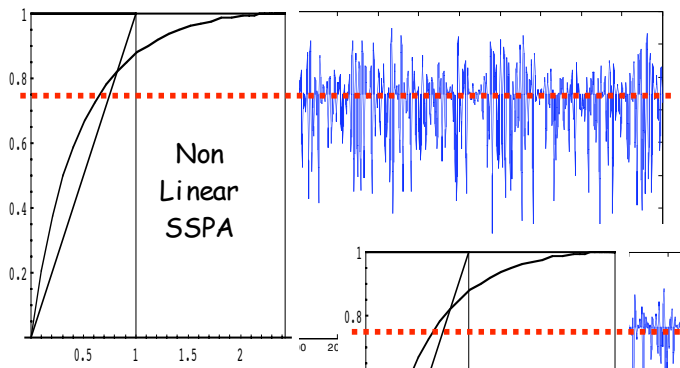
Packets segmentation, FEC-encoding, QAM-Modulation and Containerization



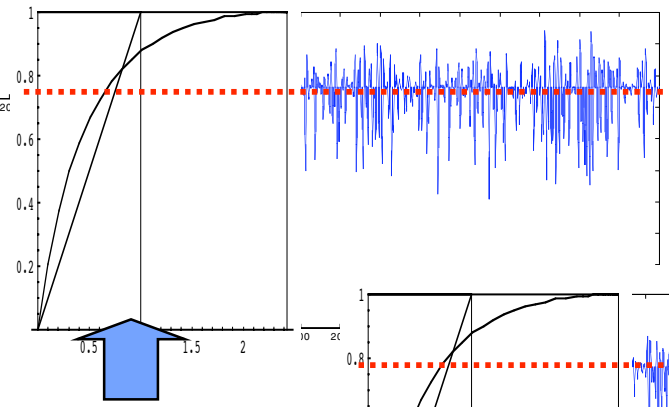
Transmission of packets with variable bit rates (without re-allocations) to satisfy mixed-application traffic delay/BER QoS requirements



Split Phase Shift Keying (SPSK) overlay modulation enables transmission of null-steered beams with antenna array SSPAs operating at near saturation

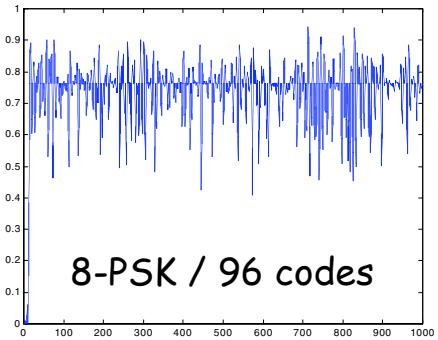


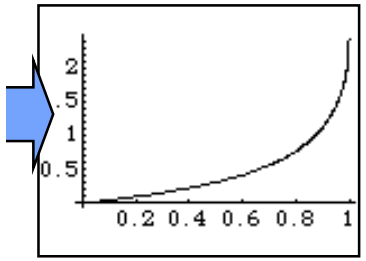
8-PSK / 24 codes/SPSK



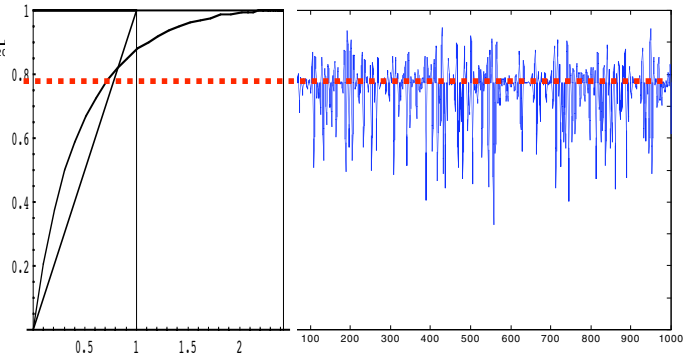
8-PSK / 96 codes/SPSK

Multi-code TCeMA-encoded SPSK-Modulated Filtered Signal



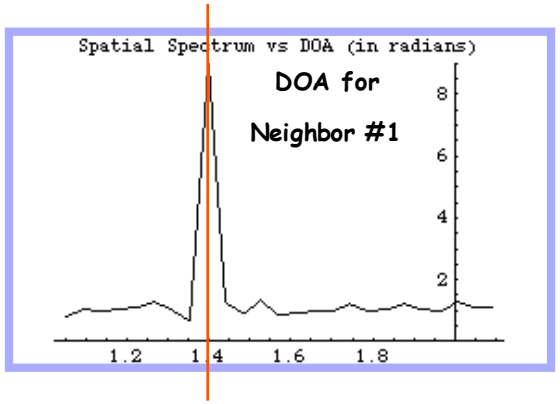


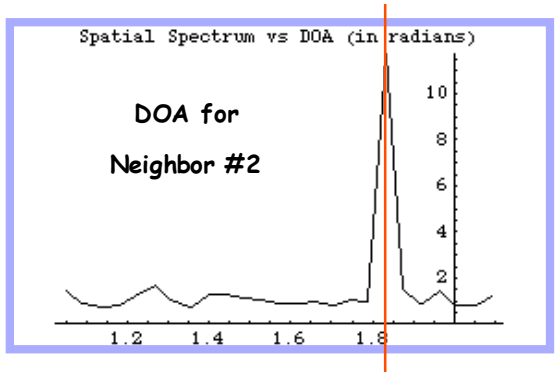
AM/AM Equalizer



Transmitted Signals

Systematic use of null-steering enables ad hoc space comms without interference

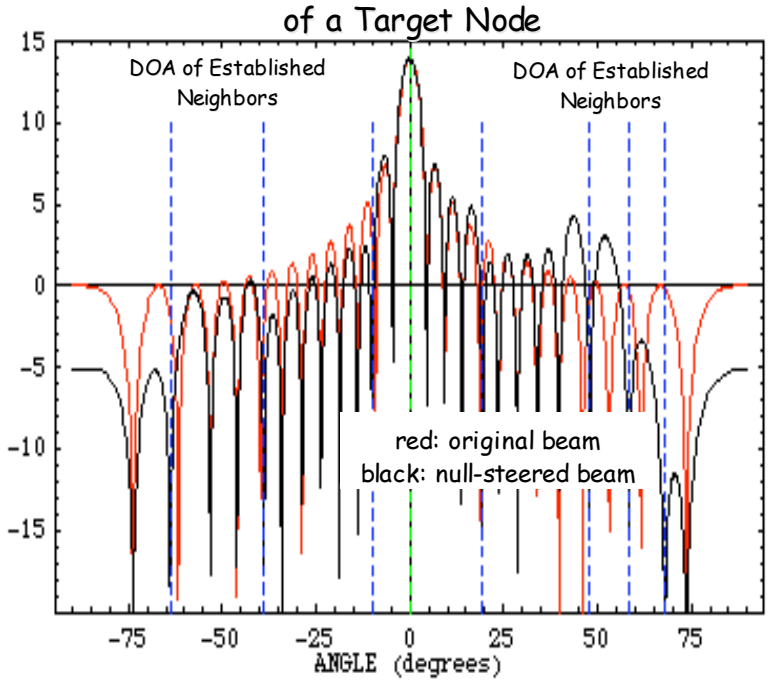




DOA estimated with MUSIC Algorithm
Applied ot the overhead bursts

Angular directions are measured & updated periodically

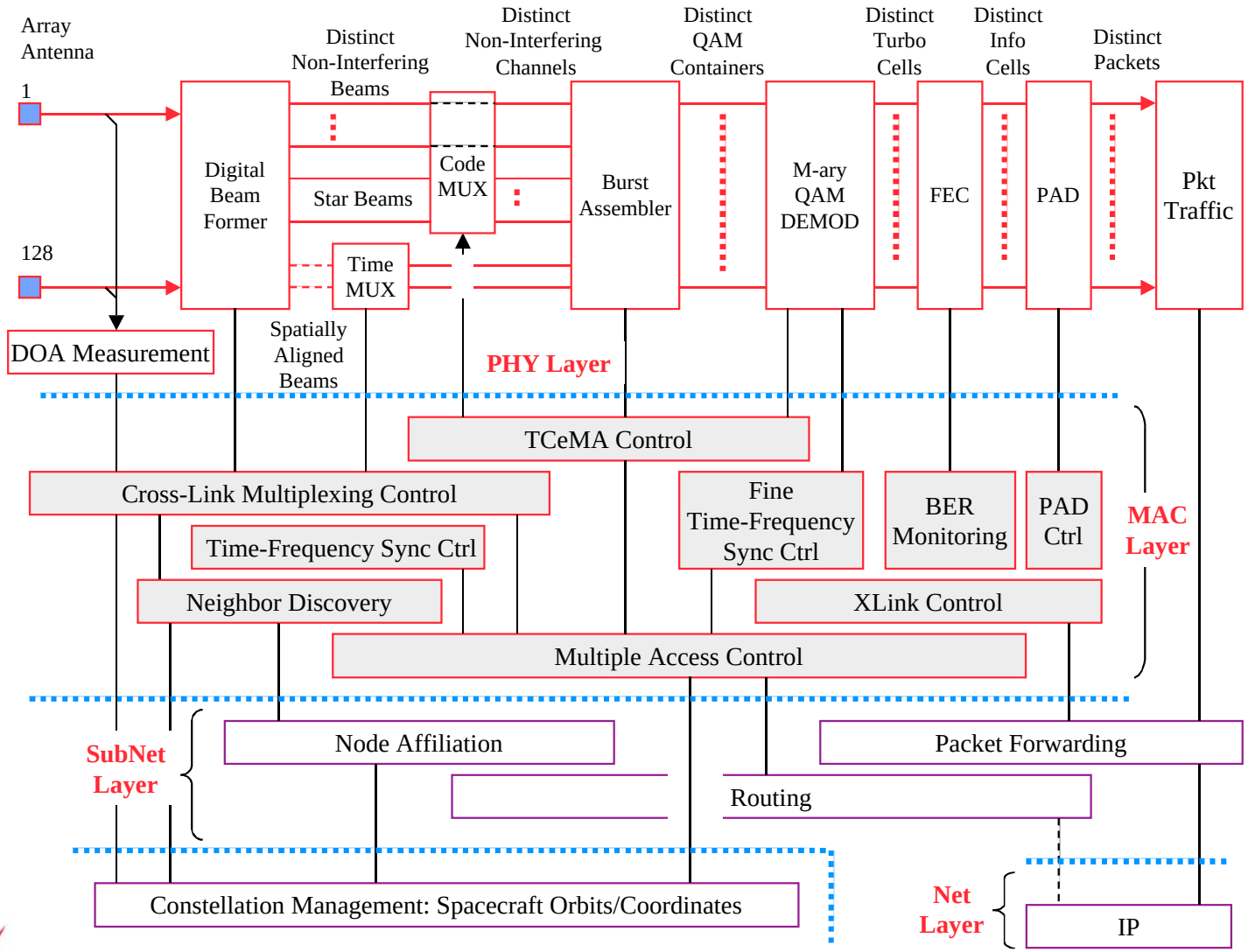
High-gain beam
in the direction
of a Target Node



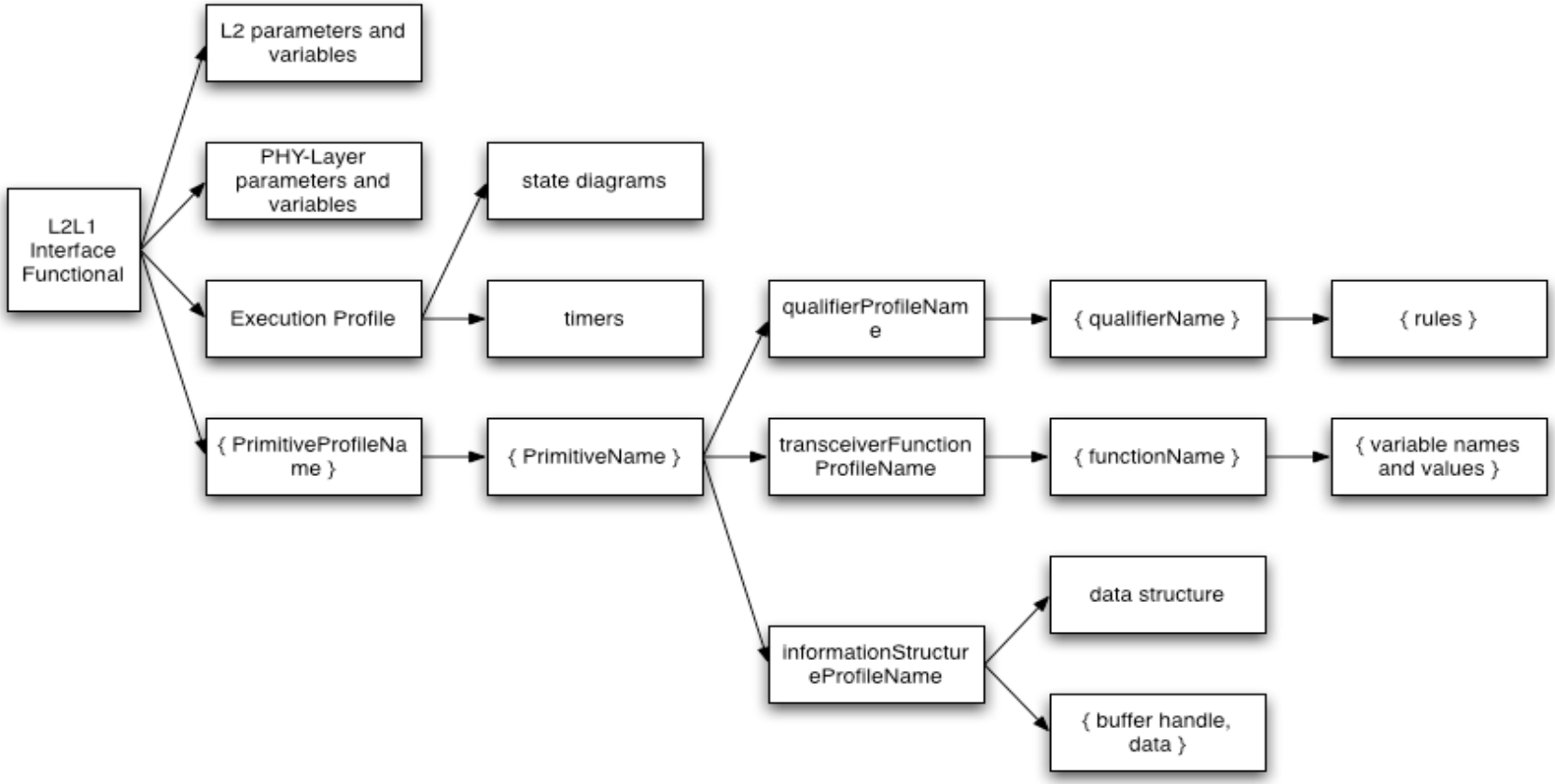
Null-Steered Beam Generated Using Novel Closest Zero Replacement Algorithm

MAC-Layer

MAC-Layer Components and Interfaces

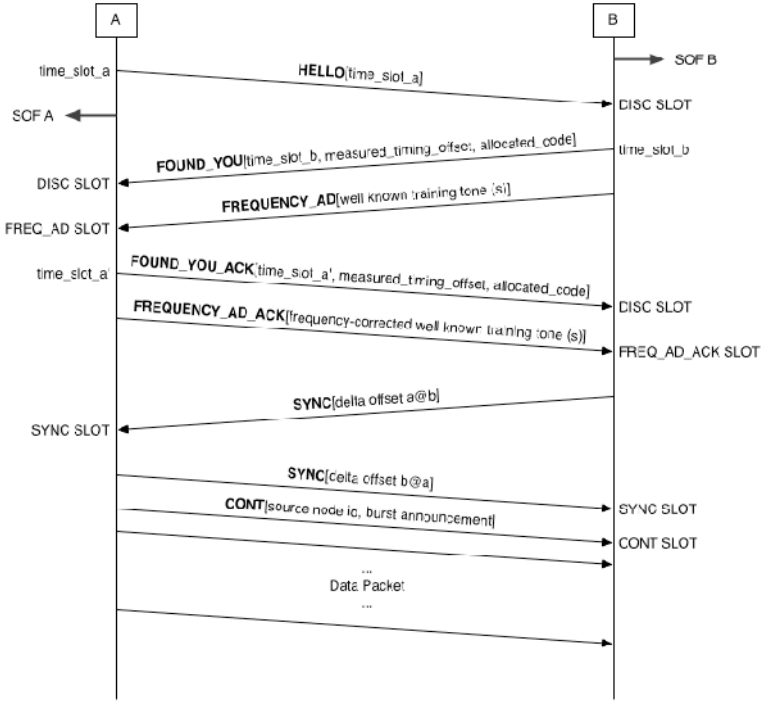


Interface Functionals and Profiles provide concrete basis for complex function implementation/MAC-Layer Configurability



MAC Example #1: Functional and Associated Commands

Functional @ MAC layer
(e.g., Neighbor Discovery)

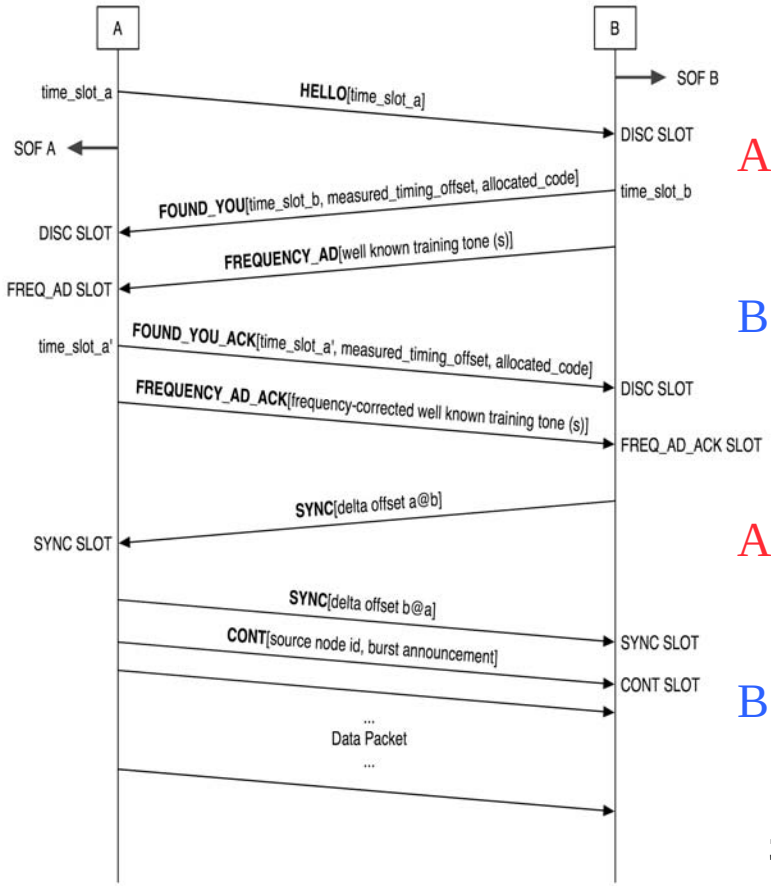


Comamnds used by Functional @ PHY layer

- tran_cmd_xmt_overhead_burst (self, kwargs):
- tran_cmd_xmt_discovery_burst (self, kwargs):
- tran_cmd_xmt_l2pdu (self, kwargs):
- tran_cmd_rcv_overhead_burst (self, kwargs):
- tran_cmd_rcv_discovery_burst (self, kwargs):
- tran_cmd_rcv_l2pdu (self, kwargs):
- tran_cmd_stop_transmission (self, kwargs):
- tran_cmd_stop_reception (self, kwargs):
- tran_cmd_capture_rcv_samples (self, kwargs):
- tran_cmd_sync_ref_frame_epoch (self, kwargs):
- tran_cmd_sync_ref_frame_number (self, kwargs=None):
- tran_cmd_sync_ref_4x_sample_clock (self, kwargs):
- tran_cmd_reset (self, **kwargs):
- tran_cmd_reset_xmit_module (self, kwargs):
- tran_cmd_xmt_active (self, kwargs):
- tran_cmd_xmt_inactive (self, kwargs):
- tran_cmd_rcv_active (self, kwargs):
- tran_cmd_rcv_inactive (self, kwargs):
- tran_cmd_proc_execute (self, kwargs):
- tran_sig_ref_frame_interrupt (self, frame_number, result_queue, test=False):
- tran_int_ref_frame_interrupt (self, test=False):

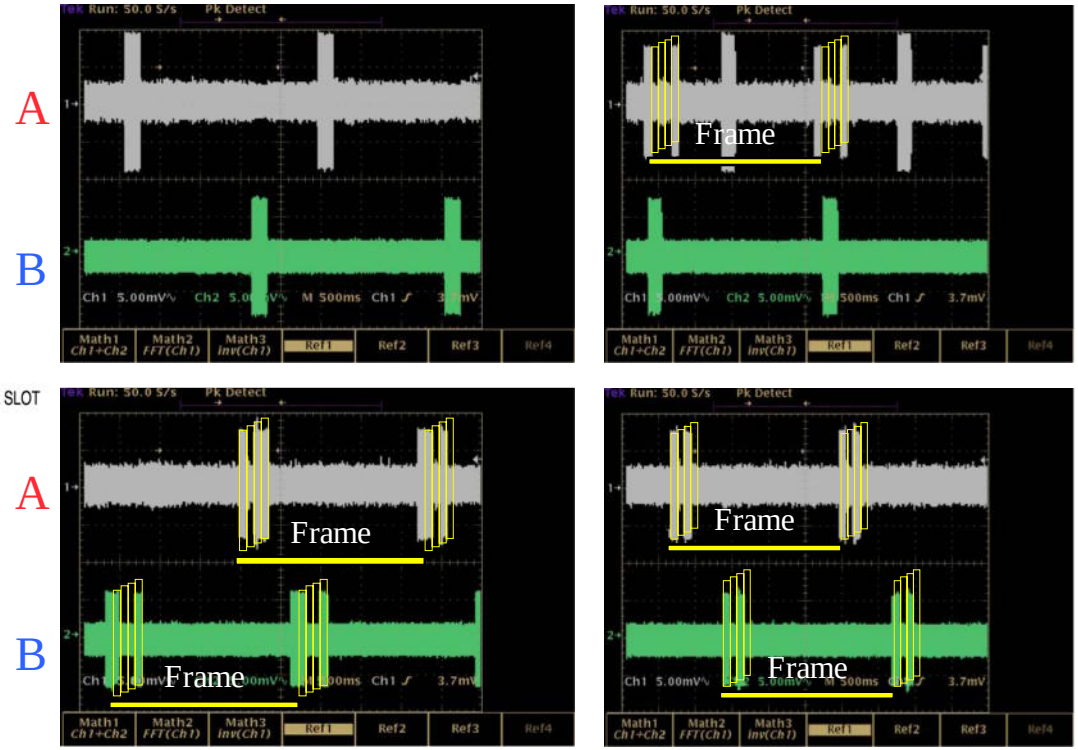
MAC Example #2: Network Self-Formation with Inter-Node Time and Frequency Synchronization

Neighbor Discovery



1. (A) HELLO, HELLO
(B) HELLO, HELLO

2. (A) FOUND_YOU, FREQ_AD
(B) HELLO, HELLO



3. (A) SYNC, _ , CONT, FOUND_YOU
FREQ_AD, SYNC, _ ,CONT
(B) FREQ_AD, FOUND_YOU_ACK

4. (A) SYNC, _ , CONT
(B) SYNC, _ , CONT

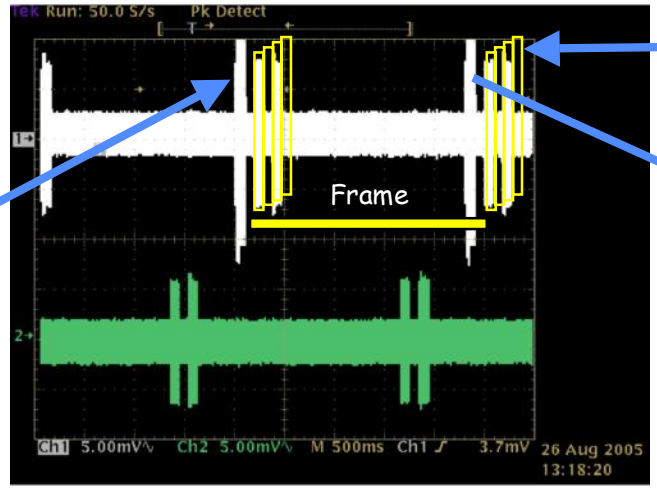
MAC Example #3: Function-based "profiles" that "goes along" with commands/responses

Example Profile: L2PDU Burst Reception

	Values/Range
Information Profile	
L2PDU Detected Sequenc e	
(optional) L2PDUrcvdHeader_numberRcvd4XSamples	*TBD confParam*
(optional) L2PDUrcvdHeader_raw4XSamples	*{ TBD # of I/Q values*
(optional) L2PDUrcvdHeader_tcemaDecoded4XSamples	*{ TBD # of I/Q values*
(optional) L2PDUrcvdHeader_closest4XSampleIndex	*{0,1,2,3}*
(optional) L2PDUrcvdHeader_sampleNumberInWhichUWd e t e c t e d	
(optional) L2PDUrcvdHeader_interpolatedBPSKSymbols	*{128 I/Q values}*
L 2 PDUrcvdHeader_detectedBits	*{a sequence of 16*8 bits*
(optional) L2PDUrcvdPayload_numberOf4XSamples	
(optional) L2PDUrcvdPayload_rcvdRaw4XSamples	*{sequence of I/Q values*
(optional) L2PDUrcvdPayload_rcvdDecoded4XSamples	*{sequence of I/Q values*
(optional) L2PDUrcvdPayload_closestSampleIndex	*{0,1,2,3}*
(optional) L2PDUrcvdPayload_estimatedFractionalSampleTimeOffset	*0 ≤ (.) <1*
(optional) L2PDUrcvdPayload_rcvdInterpolatedMaryQAMSymbols	*{sequence of I/Q values*
(optional) L2PDUrcvdPayload_detectedMaryQAMSymbols	*{sequence of I/Q values*
(optional) L2PDUrcvdPayload_bitsPerSymbol	*1 ≤ (.) ≤ 8*
L 2 PDUrcvdPaylod_detectedBits	*{a sequence bits*
L2PDU QoS Performance Characterization	
L2PDUrcvdPayload_measuredAfterTCeMASignalPlusNoisePowerDB	
L2PDUrcvdPayload_measuredAfterTCeMANoiseOnlyPowe r D B	
FEC Profile	
(optional) L2PDUrcvdPayload_measuredNumbeBitsCorrected	
(optional) L2PDUrcvdPayload_calculatedAfterFec BER	
Modulation Profile	
L 2 PDUrcvdPayload_bitsPerSymbol	*1 ≤ (.) ≤ 8*
L2PDUPayload_modulation	*{ 4 5 ° B PSK,QPSK,8PSK,16QAM,32QAM,64QAM,128QAM,246QAM}*

	Values/Range
TCeMA Encoding Profile	*configuration parameter*
F r a me Scheduling Profile	
L2PDUburst_multiframeNumberInWhichReceived	
L2PDUburst_frameNumberInWhichRecieved	
Time Slot Scheduling Profile	
L2DUburstStart_firstTimeSlotNumber	*value in burst heade*
L2PDUburstLength_inTimeSlots	*value in burst heade*
Code Slot Scheduling Profile	
L2PDUburstCodeStripe_firstCodeNumber	*value in burst heade*
(optional) L2PDUheaderCodeStripe_numberOfCodes	*from ACT/CONT bursı*
L2PDUpayloadCodeStripe_numberOfCodes	*value in burst he*
Array Selection/Configuration/Orientation Profile	
a n t e n n a A r r a y _ s e l ectedNumber	*k ≤*
(optional) antennaArray_numberOfElementsSelected	*≤ 51:*
(optional) upConverter_selectedNumber	*(k)*
Beam Forming Profile	*N/A*
Null Forming Profile	*N/A*
RF Power/Equalization Profile	N/A
Frequency Profile	configuration parameter
L 2 PDUrcvdPayload_estimatedCarrierFrequencyOffs e t	*TBD rang*
Transmit/Receive Time Profile)	
L 2 PDUrcvdHeader_sampleNumberInWhichUWdetected	*TBD range*
L 2 PDUrcvdPayload_closestSampleIndex	*{0,1,2,3}*
L2PDUrcvdPayload_estimatedFractionalSampleTimeOffset	0 ≤ (.) <1

MAC Example #4: Transmission of IP Packets Between Established Neighbors

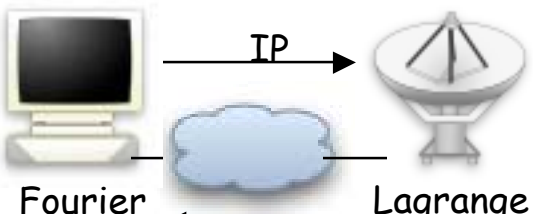


Sync + Contention Bursts

IP Data Bursts

IP Data Bursts with Time/code/spatial multiplexing

Experimenter



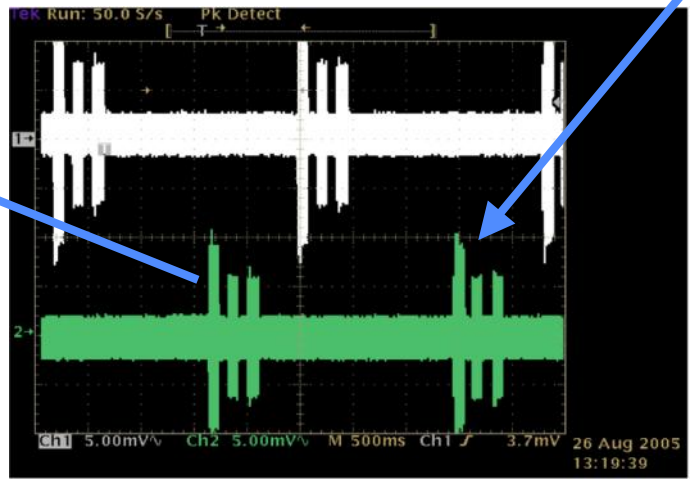
Fourier (10.0.2.1)

Lagrange (10.0.0.1)

Maxwell (10.0.0.2) → Kepler (10.0.3.1)

Sensor Satellite

On-Board Instrument



IP Data Bursts

```
root@fourier:[root@fourier root]#  ping -I 10.0.2.1 10.0.3.1
PING 10.0.3.1 (10.0.3.1) from 10.0.2.1 : 56(84) bytes of data.
64 bytes from 10.0.3.1: icmp_seq=1 ttl=62 time=59023 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=62 time=60197 ms
64 bytes from 10.0.3.1: icmp_seq=3 ttl=62 time=61359 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=62 time=62660 ms
64 bytes from 10.0.3.1: icmp_seq=5 ttl=62 time=62686 ms
64 bytes from 10.0.3.1: icmp_seq=6 ttl=62 time=65272 ms
64 bytes from 10.0.3.1: icmp_seq=7 ttl=62 time=66536 ms
```

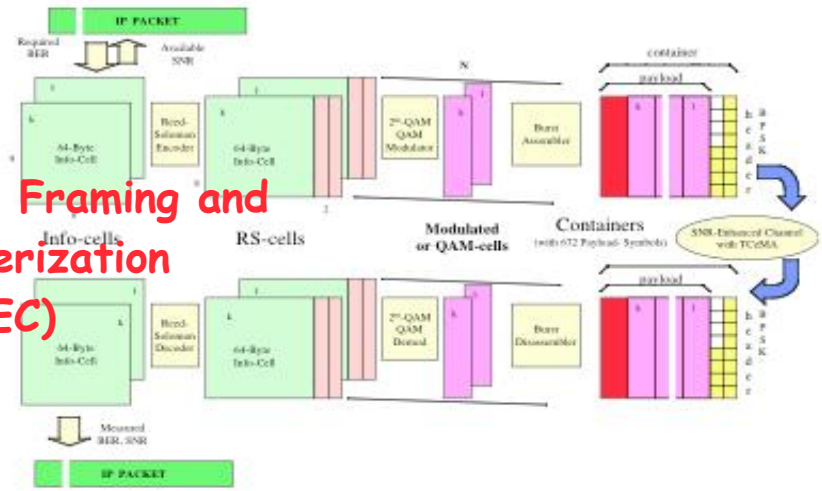
BBN TECHNOLOGIES

SpaceVPN Software Defined Radio

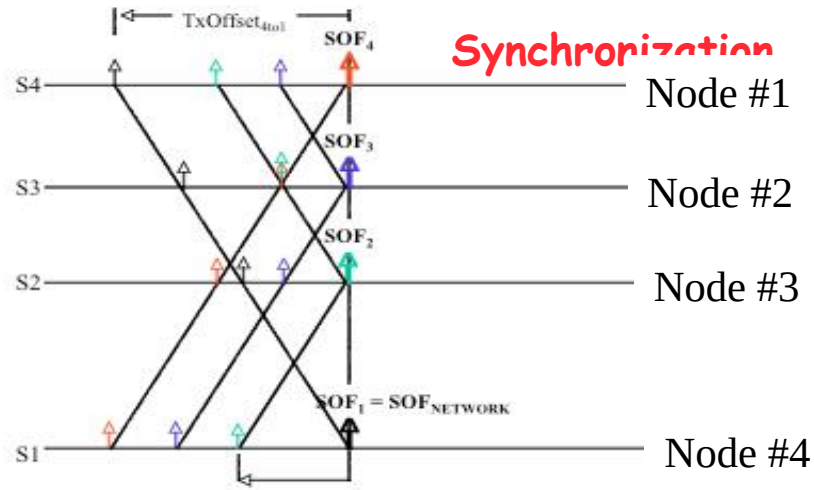
ESTO
Earth Science Technology Office

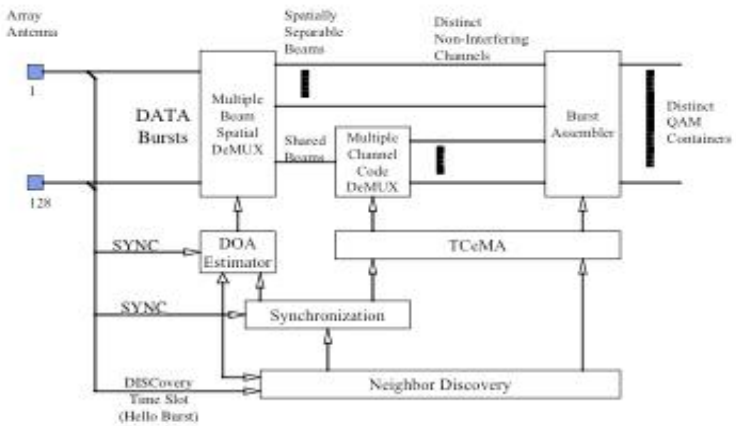
Software Requirements from SpaceVPN Architecture

Flexible Framing and Containerization (with FEC)

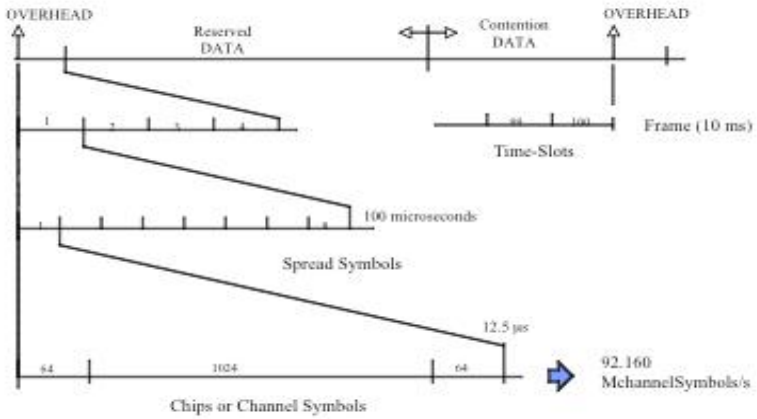


Multi-Node Synchronization





Complex Module Composability



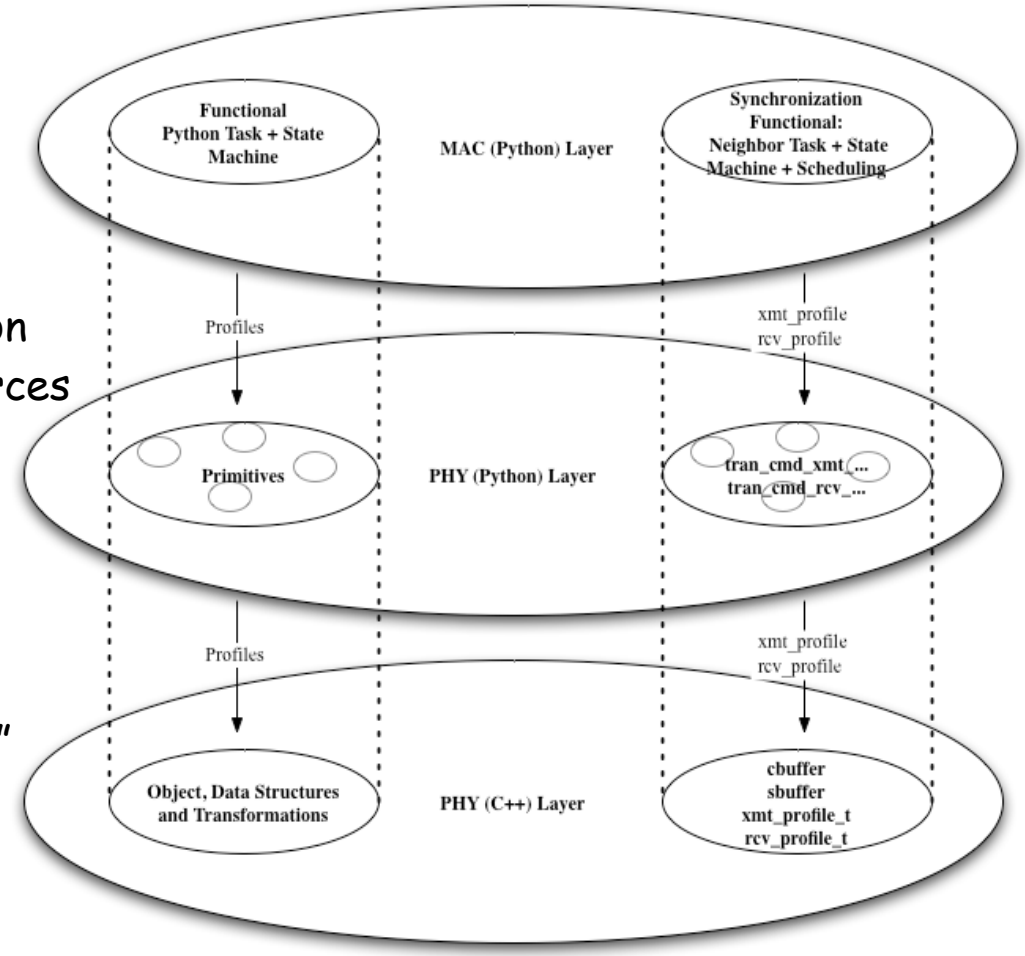
Flexible Rate Encoding w/ TCeMA

Implementation Approach #1: GNU-enhanced software-radio that matches specs based on functionals, primitives and profiles

Light-weight
GNU-based
Implementation
Fosters/Enforces
Modularity
Testability

&

"Brain/Muscle"
partitioning



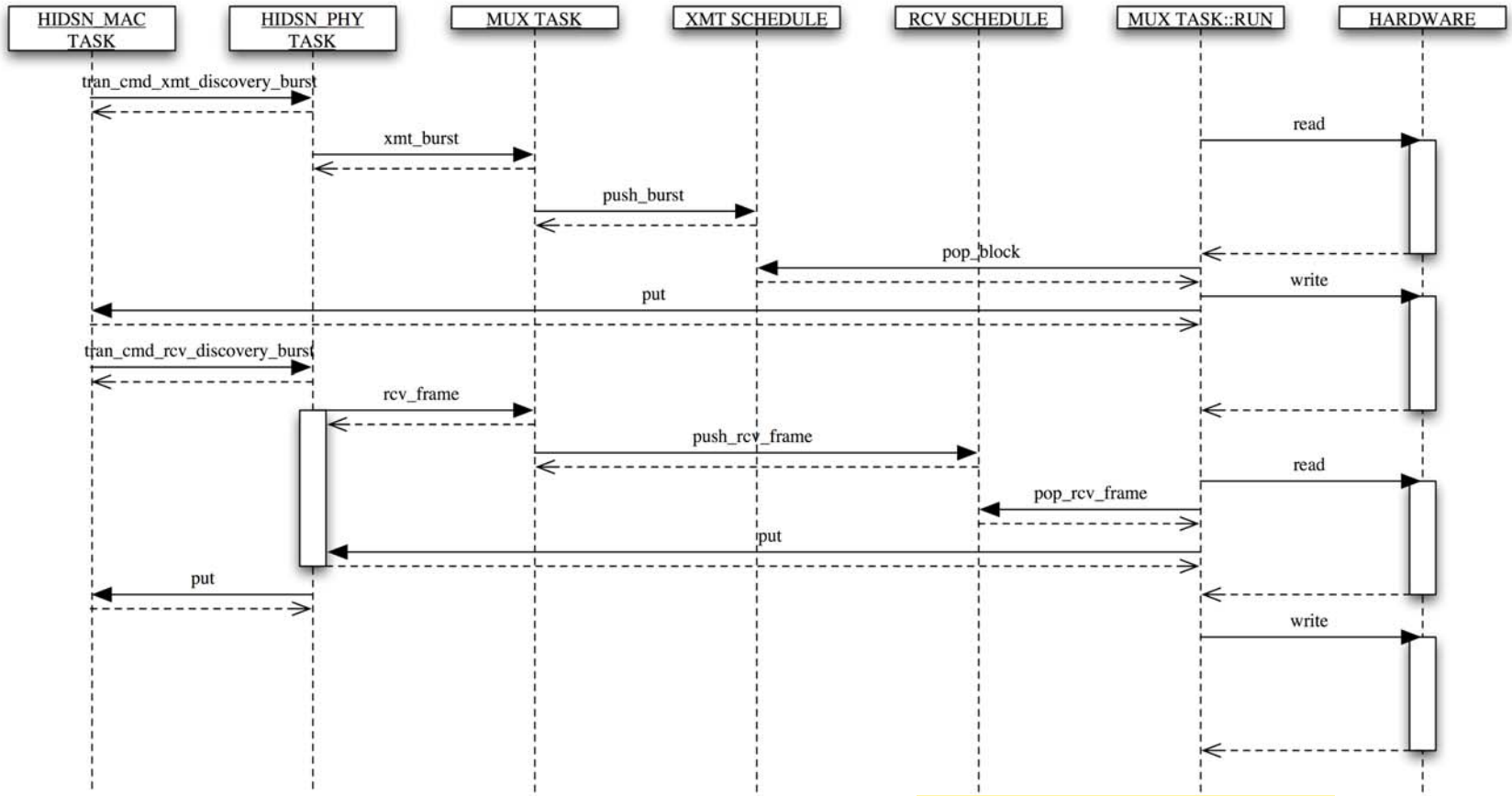
Near-perfect
mapping between
Functionals<=>Tasks
Primitives <=> msgs
Profiles <=>
{functions
& variables}

fosters/enforces
Traceability to specs

Implementation Approach #2: GNU-enhanced SDR system with integrated support for radio, MAC and multi-hop routing functions

"Standard" GNU (Radio only)	SpaceVPN GNU+ (Radio + MAC + multi-hop routing)
Only at PHY layer	At PHY and MAC layers
No State Machines	State machines incorporated in a configurable manner (see below)
Static processing chain configuration	Dynamic processing chain configuration
Designed for signal processing	Designed for discrete event driven systems
Buffer limitation	No buffer limitation Pass pointers instead of buffers
No visibility b/w modules	High visibility and exchange of information between modules
No support for frame structure	Support for frame structure
No support for asynchronous events	Support for flexible, acknowledgeable asynchronous event notification
No distinction between data and control traffic	Parallel data and control streams Each data stream has an associated control stream
Messy buffer management	Clean Buffer Management
	Configurable MAC layer Based on HiDSN L1L2 functionals
	Integrated with packet forwarding (soon with multi-hop position-based routing)

Implementation Approach #3: Asynchronous Request-Response Model Enables Synchronized/Frame-Based Operations and Provides PHY-Layer Visibility to MAC/Higher-Layers

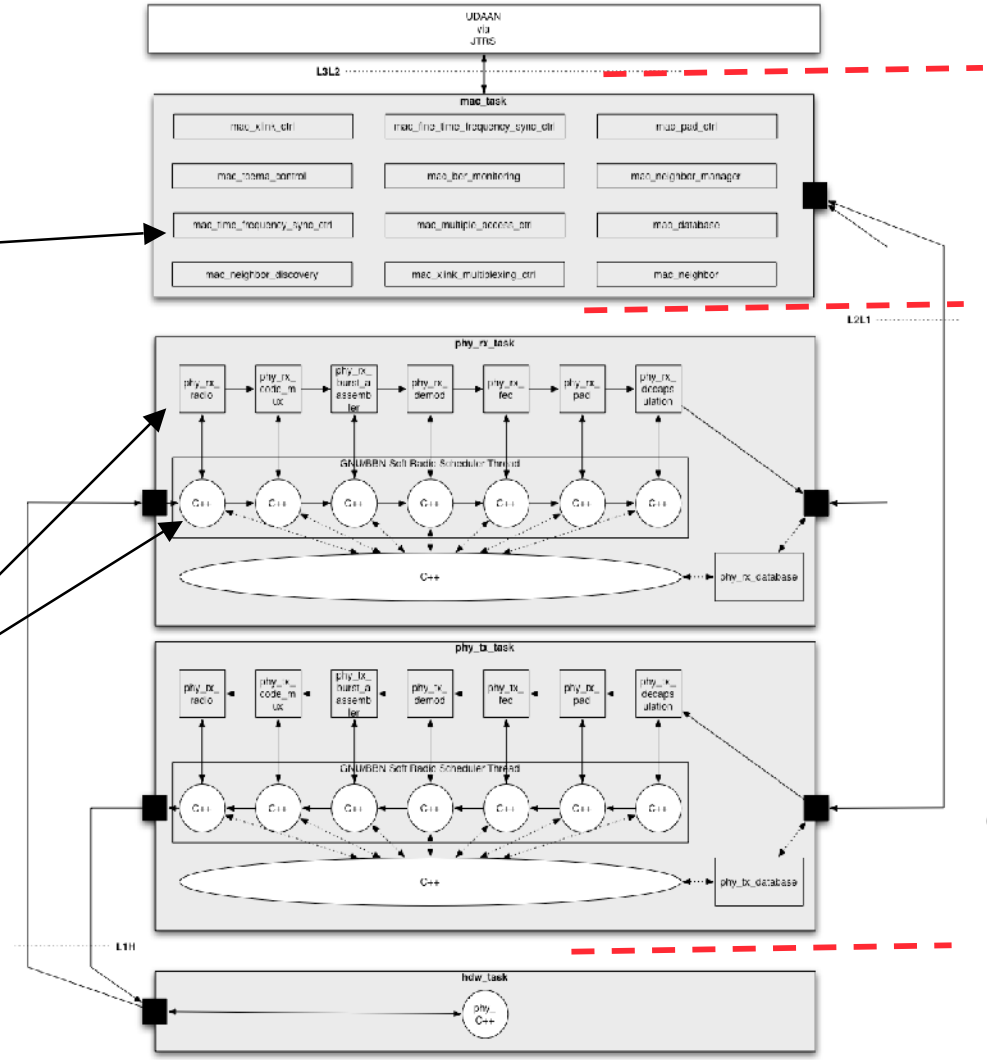


Asynchronous ⟷ Frame-based synchronous

Implementation Approach #4: Composable modules with flexible division of functionality between Python (flexible testability) and C++ (greater performance)

MAC Layer with **configurable and modular** blocks or "**functionals**" that combine state machines, timers with PHY layer blocks and primitives.

Configurable, **modular** and **composable** GNU-like Python/C++ blocks.

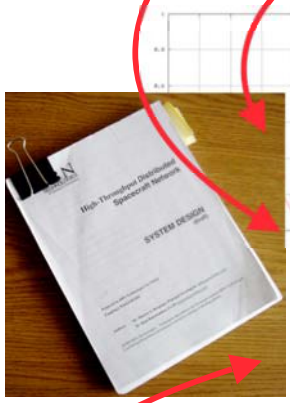


Clean interface between MAC and upper layers

Clean PHY-MAC interface for **GNU** interoperability

Clean PHY-Hardware Interface for **hardware** Interoperability

Implementation Approach #5: Test cycle that integrates Mathematica-based specification/simulation, SDR-channel emulation and lab verifications



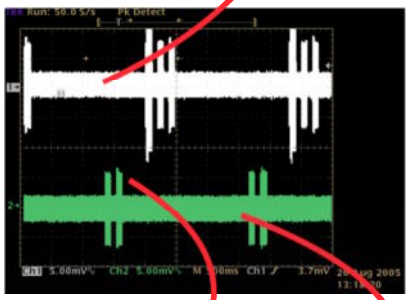
Live Mathematica-based Algorithm, Waveform, and Functional/Interface Specifications and Simulation

GNU-SDR Prototype Implementation

GNU-SDR Baseband/RF Channel Emulation

GNU-SDR Network Topology, and Mobility Emulation

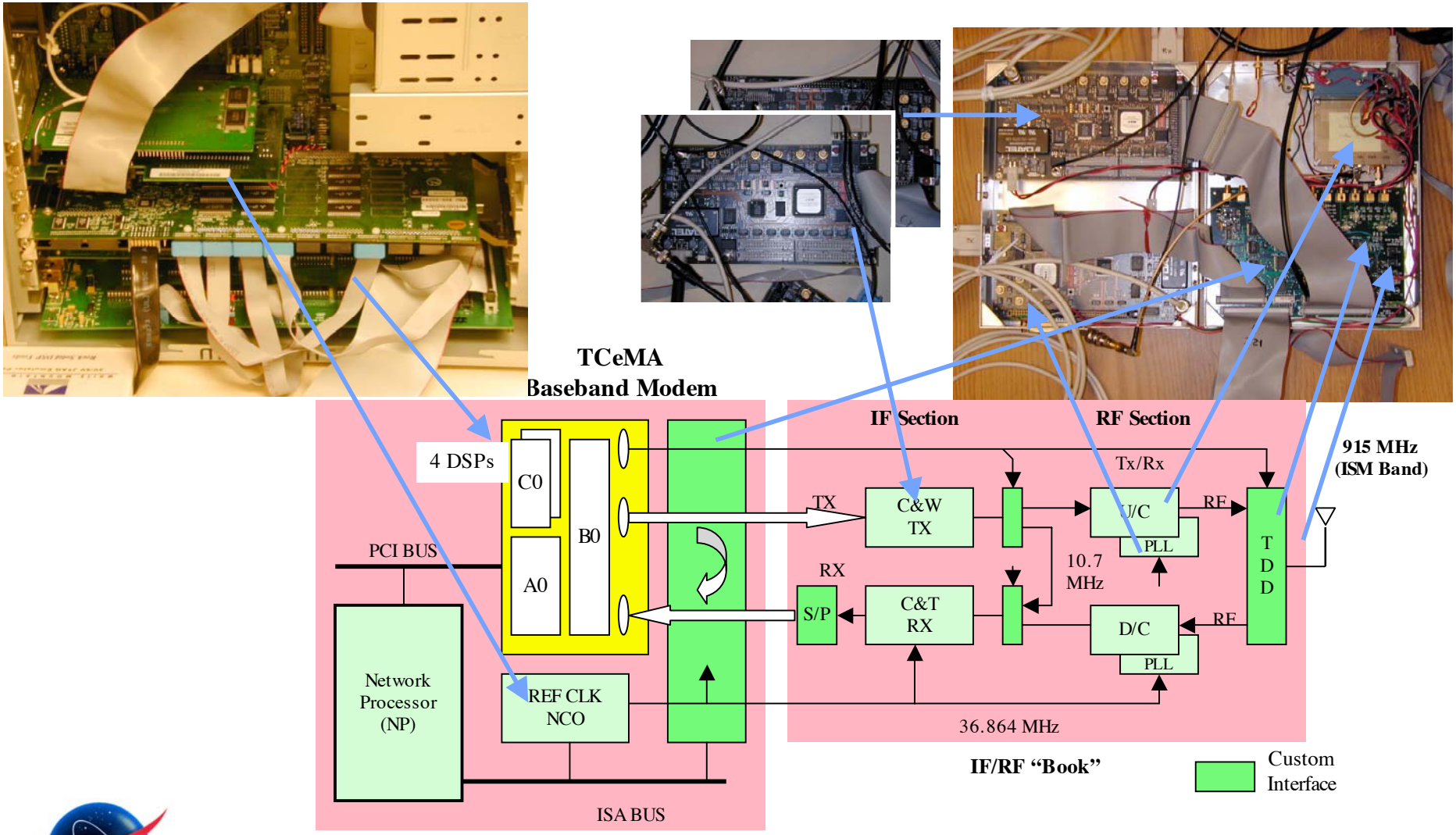
Lab verification



Results

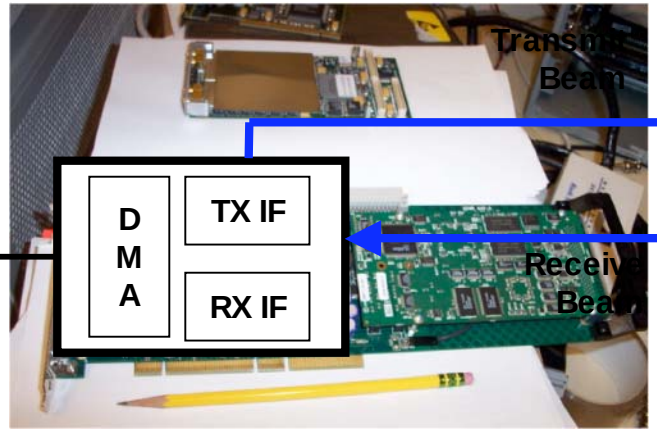
Lab Testbed

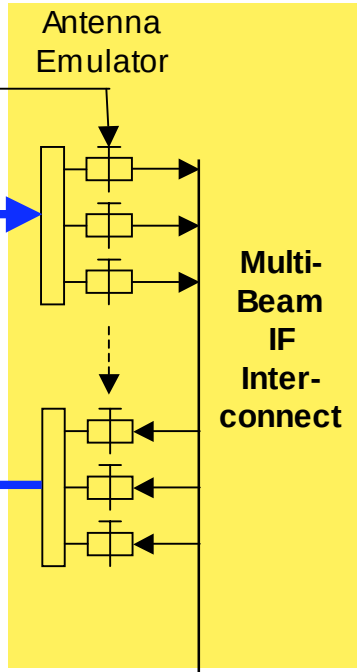
Testbed Evolution: SDR Generation 1



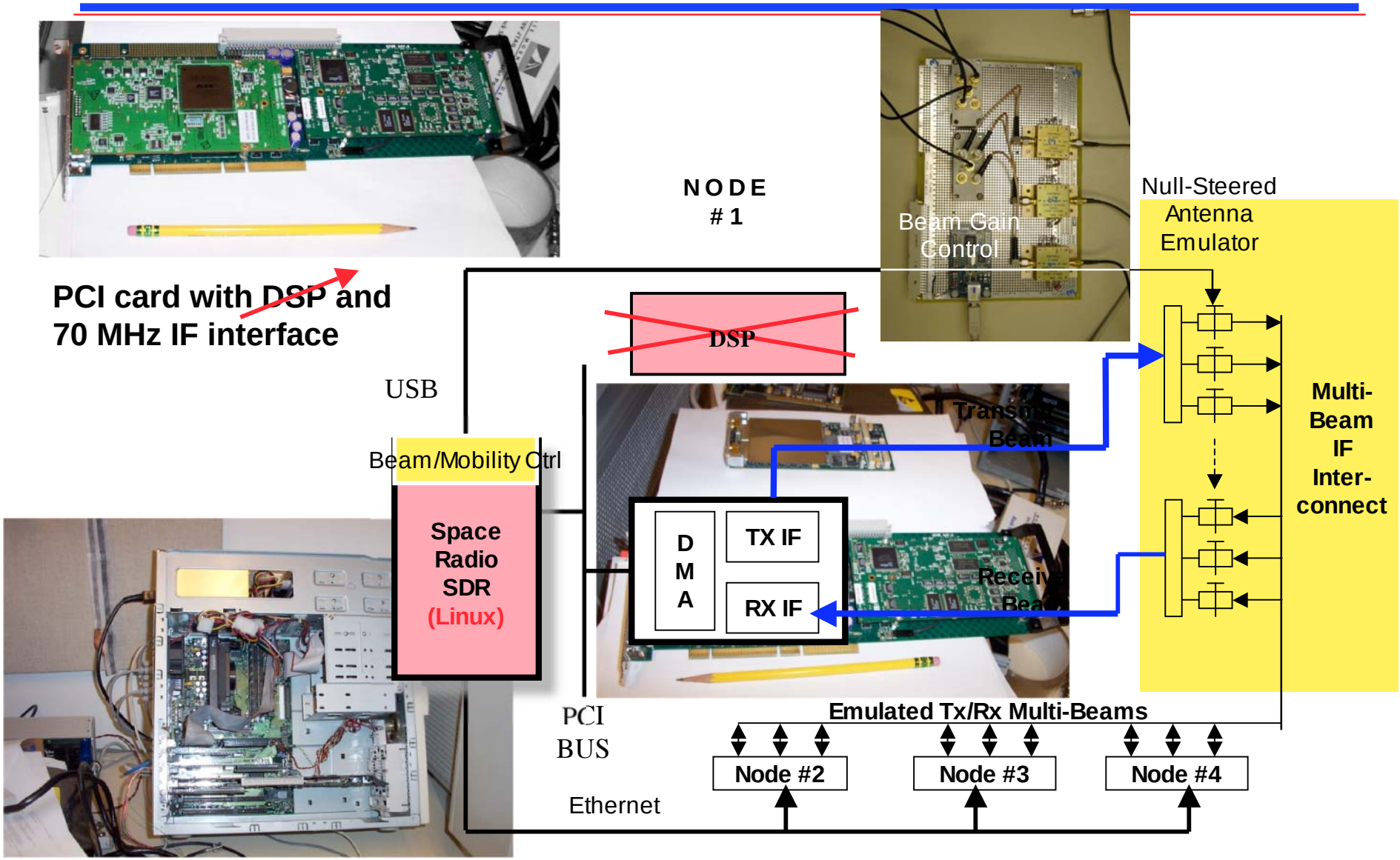
Testbed Evolution: SDR Generation 2

PCI card with DSP and 70 MHz IF interface



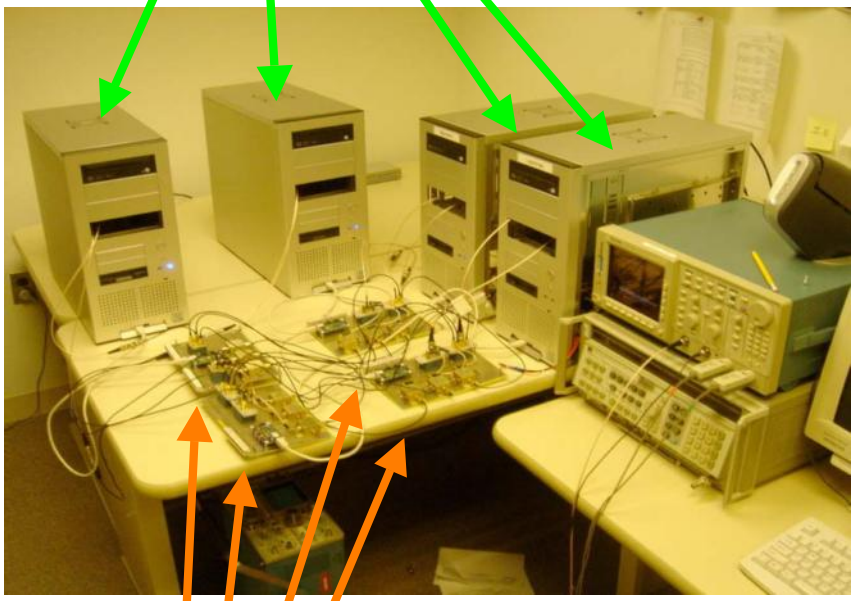


Testbed Evolution: SDR Generation 3



SpaceVPN Testbed

RF Interface equipped Linux PCs



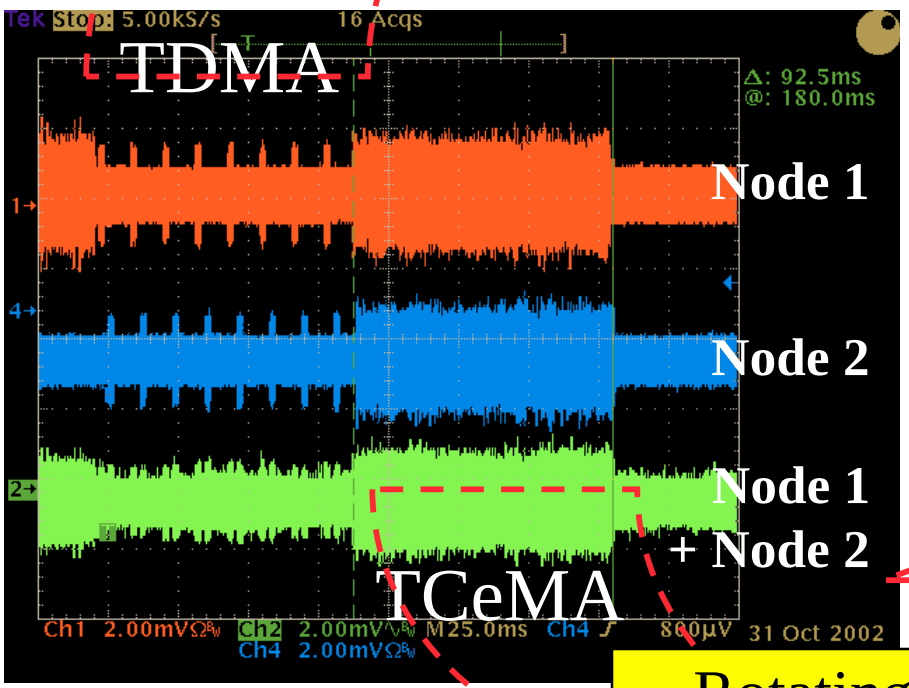
Directional Beam Emulation Boards

- **Testbed:**
  - **4 Pentium IV Linux PCs**
  - **Equipped with RF interface boards**
  - **Radios connected in a mesh via** Dynamic Beam Emulation **boards that form topologies by controlling transmit-side attenuation**
  - Mobility Manager **also controls transmit-side attenuation to emulate satellite mobility**
  - **Space VPN Soft Radio installed**
- **Results:**
  - **Mobility Emulation**
  - **Directional beam emulation**
  - **Neighbor Discovery**
  - **Dynamic Network Formation**
  - **Time and Frequency Synchonization**
  - **Link characterization**
  - **Variable Link Rates**
  - **BPSK and M-ary QAM modulation**
  - **IP packet exchange**

Test Results

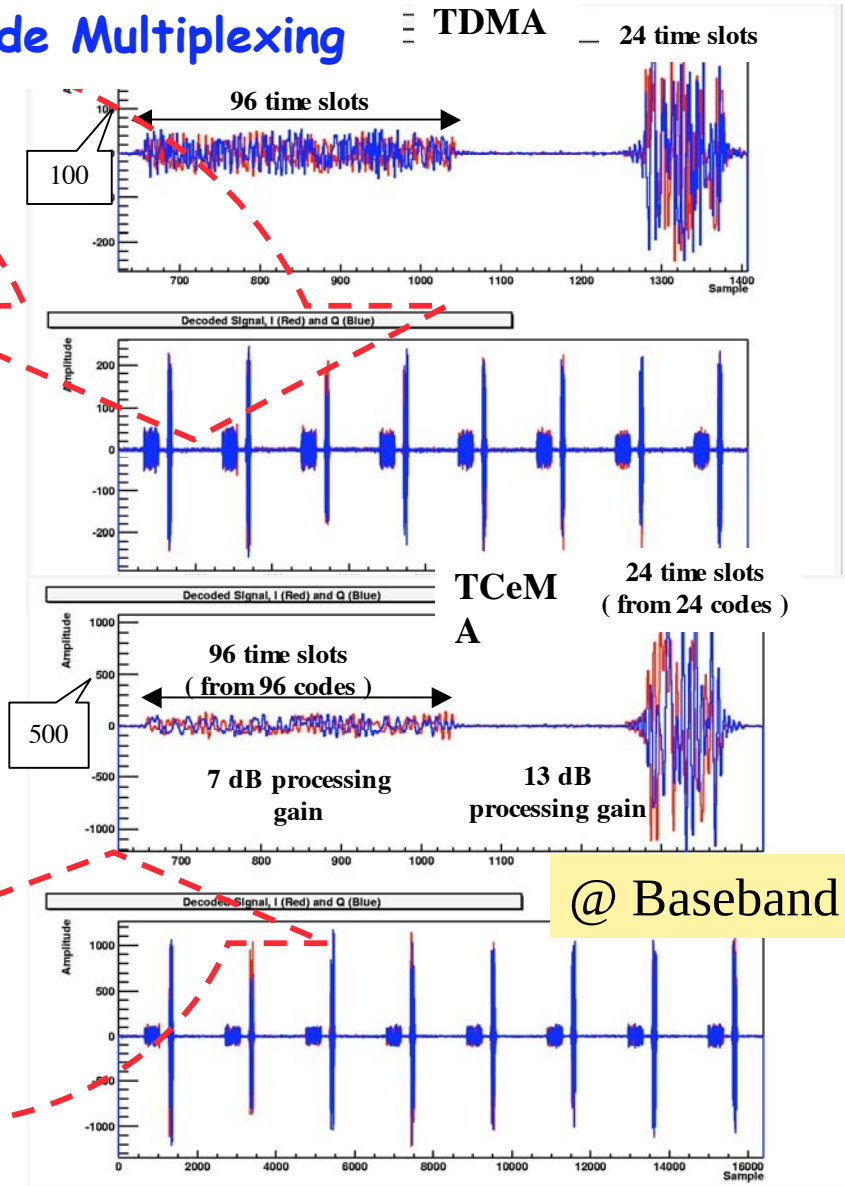
BBN TECHNOLOGIES

Results Set#1:
- Time-Code Duality
- Variable Rates with Multi-Code Multiplexing



@ 70 MHz

Rotating Correlator

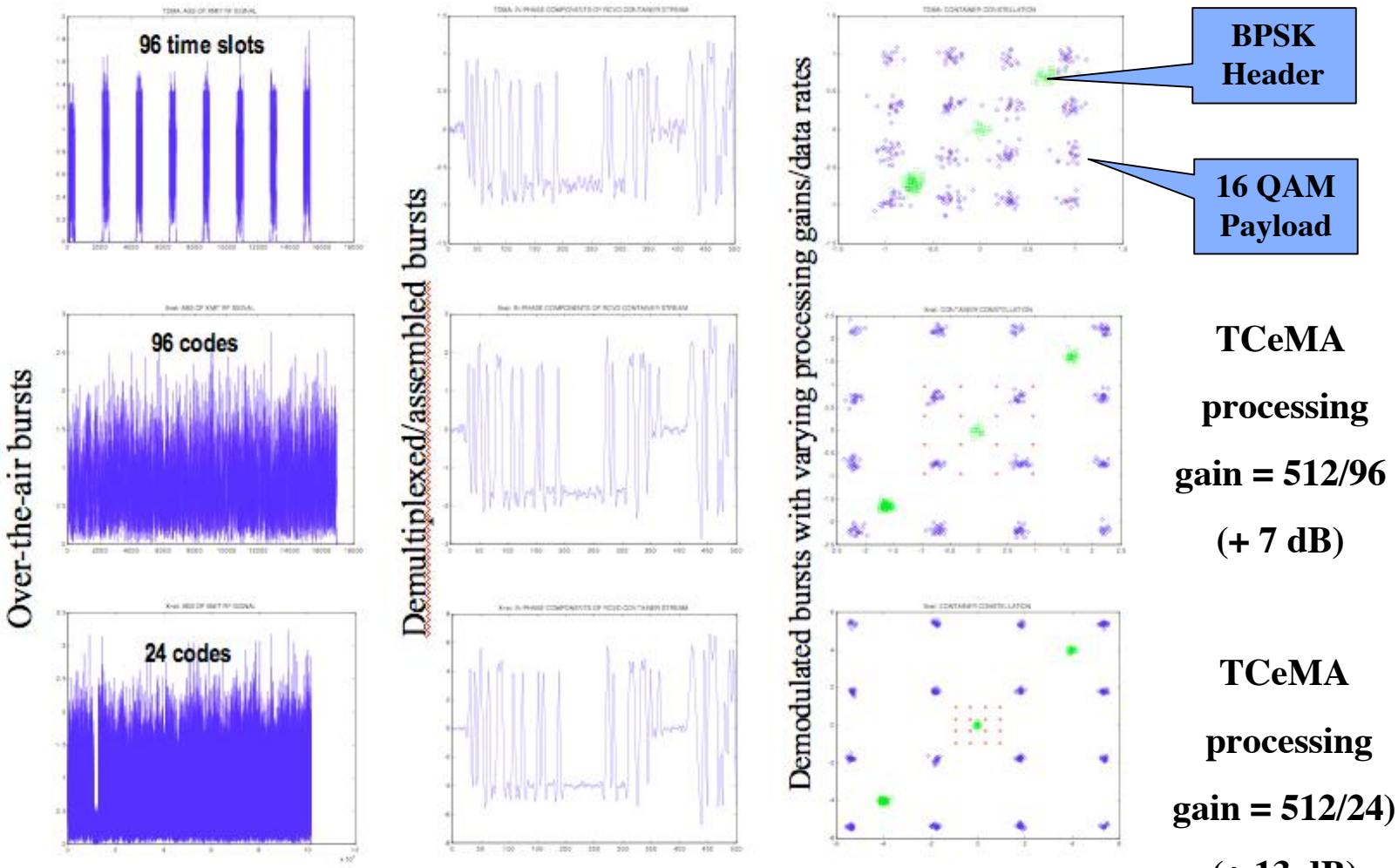


NASA ESTO
Earth Science Technology Office

BBN TECHNOLOGIES

Results Set #2: Support for variable bit rates

- Processing gain as needed to close links at different distances
- Variable symbol rate (three orders of magnitude with multi-codes)
- Support for multi-bit modulation (BPSK to 64-QAM Modulation)

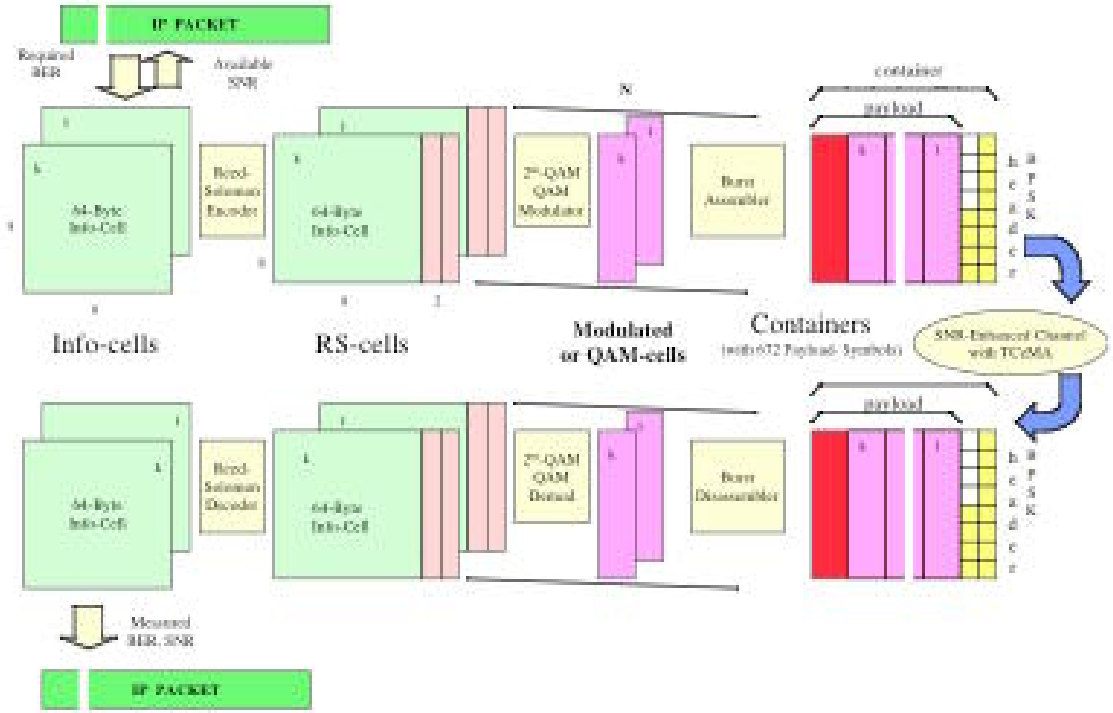


TCeMA processing gain = 512/96 (+ 7 dB)

TCeMA processing gain = 512/24) (+ 13 dB)

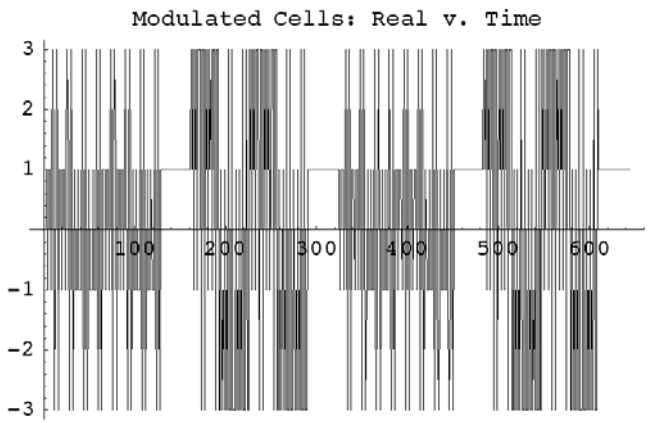
BBN TECHNOLOGIES

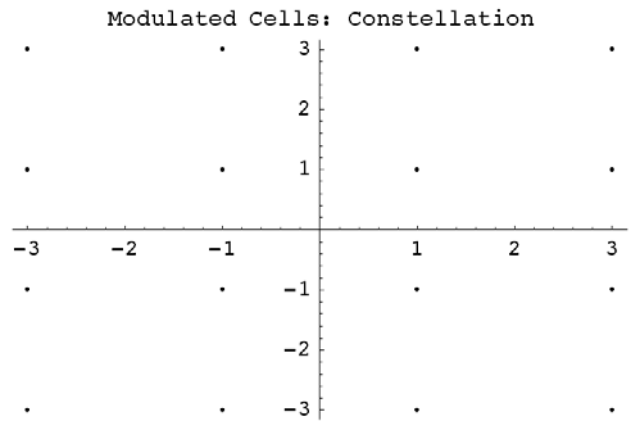
Results Set #3: Containerization, Modulation, Spreading and Interpolation

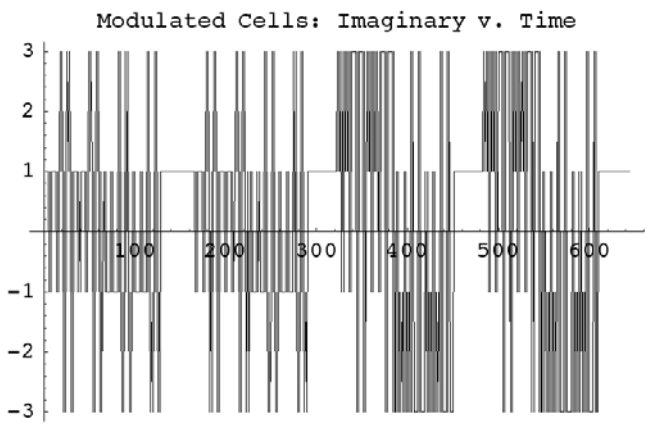


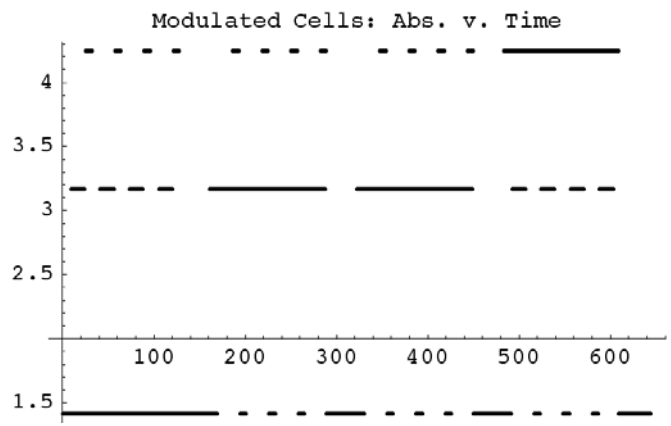
NASA ESTO
Earth Science Technology Office

Containerization: Modulated Cells

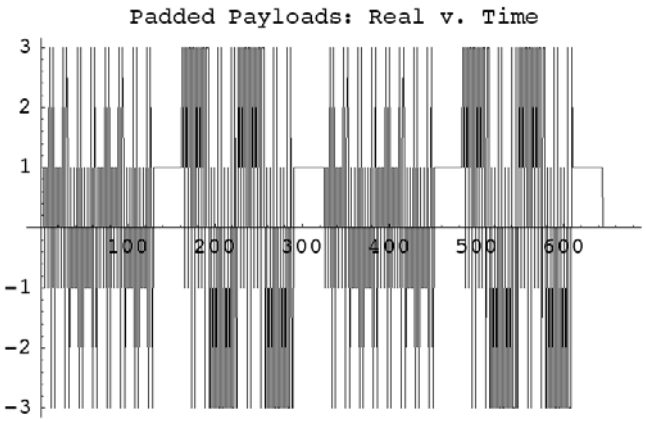


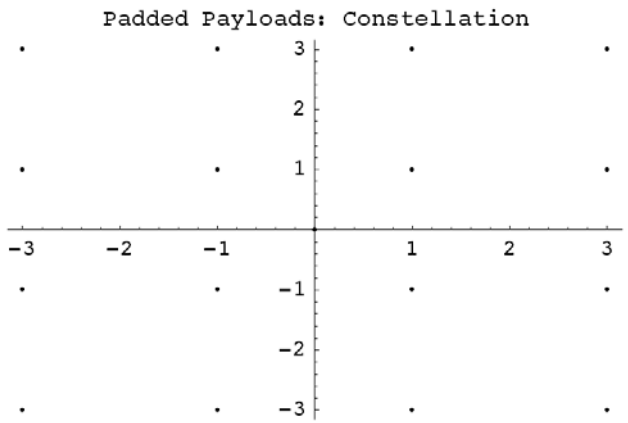


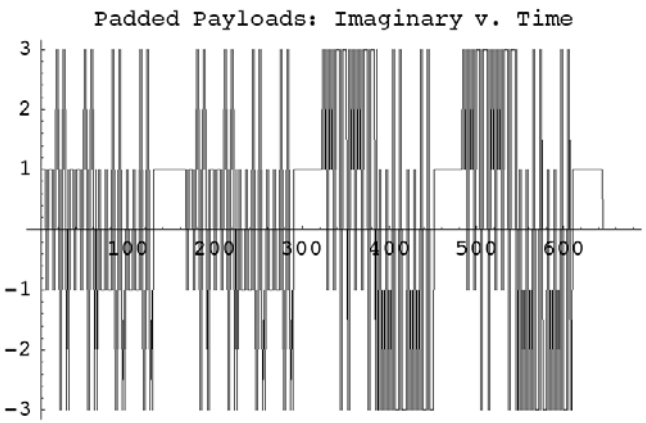


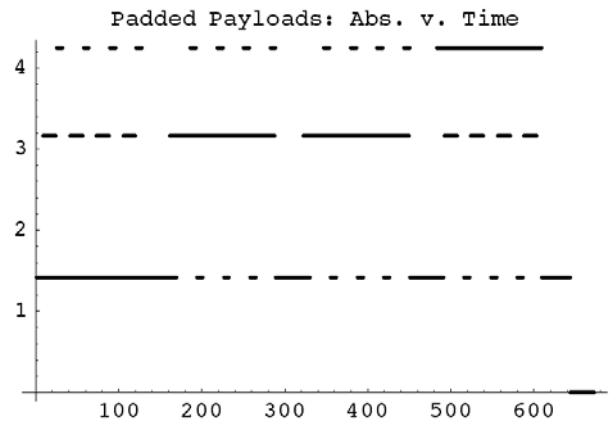


Containerization: Padded Payloads

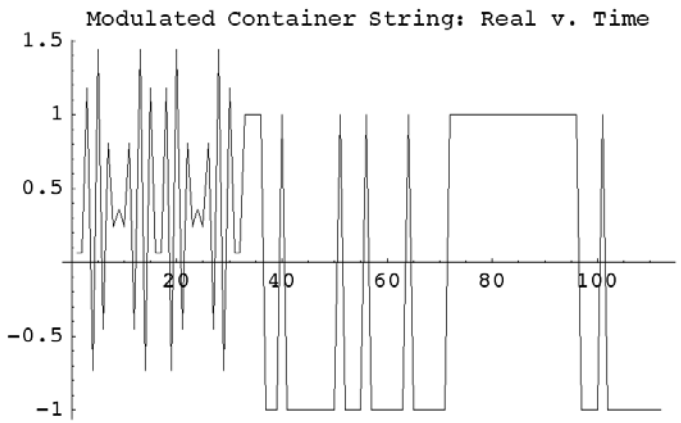


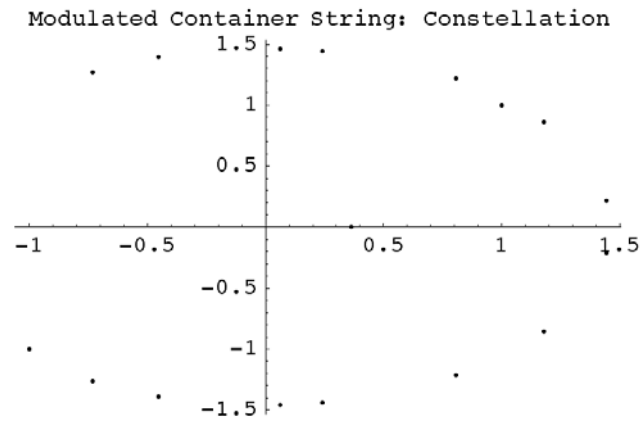


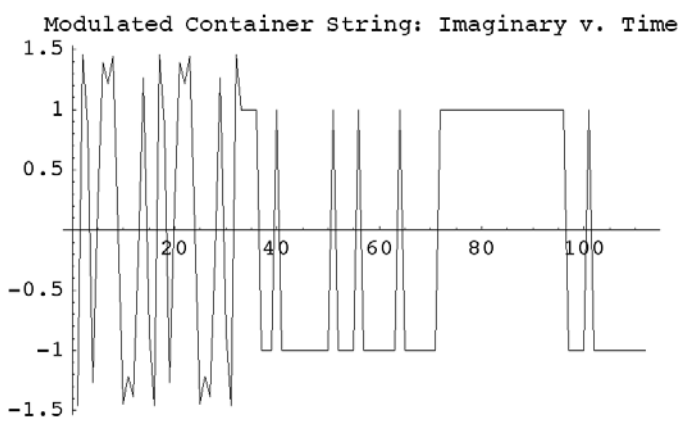


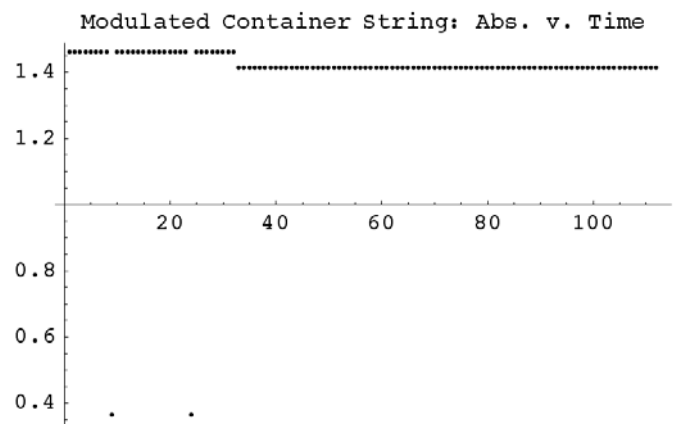


Containerization: Modulated Container Headers

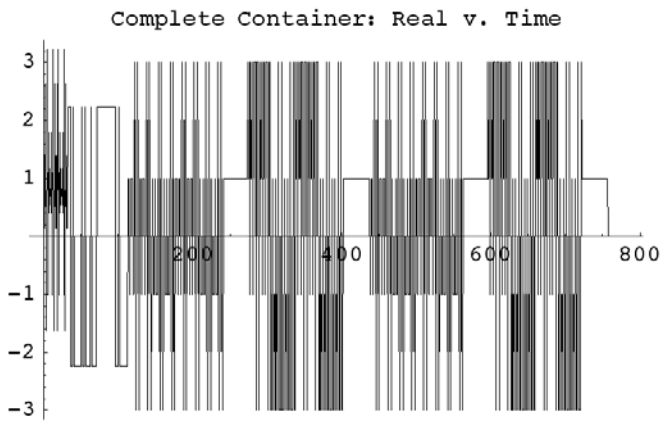


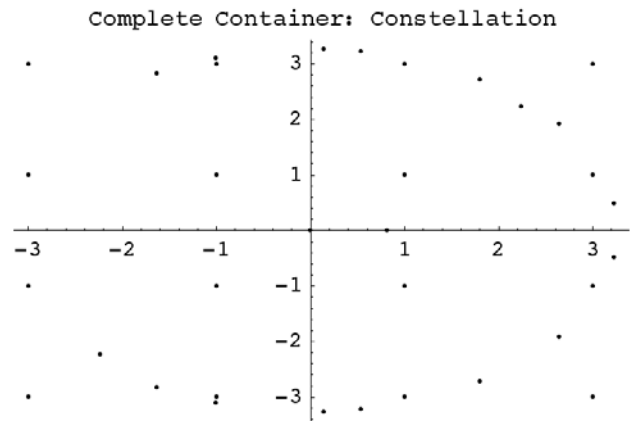


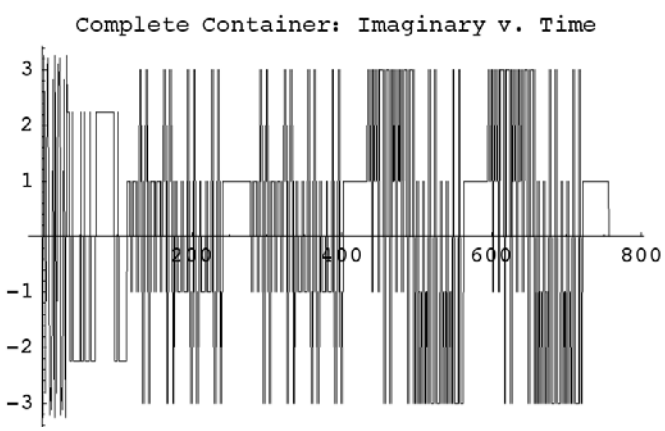


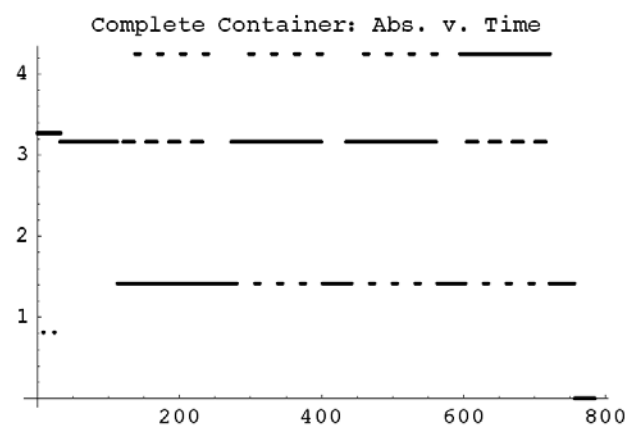


Containerization: Complete Containers

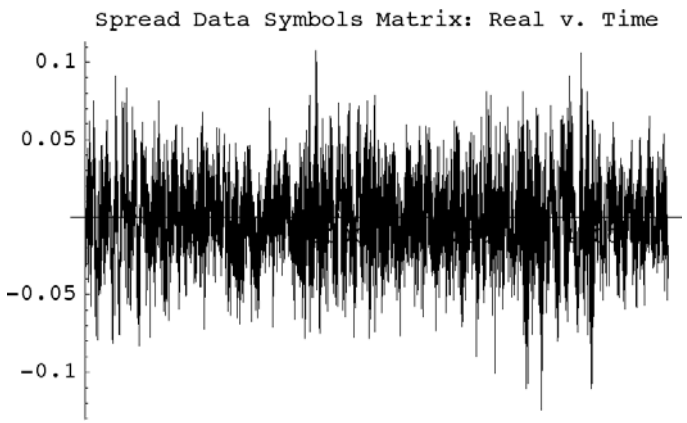


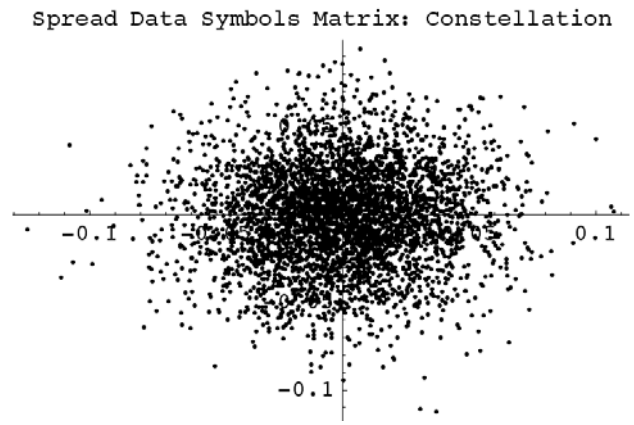


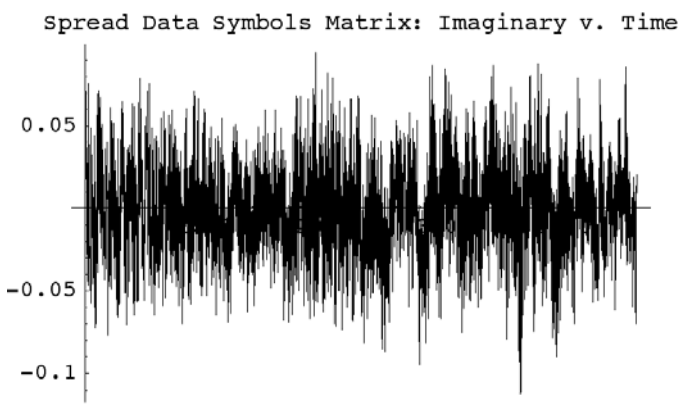


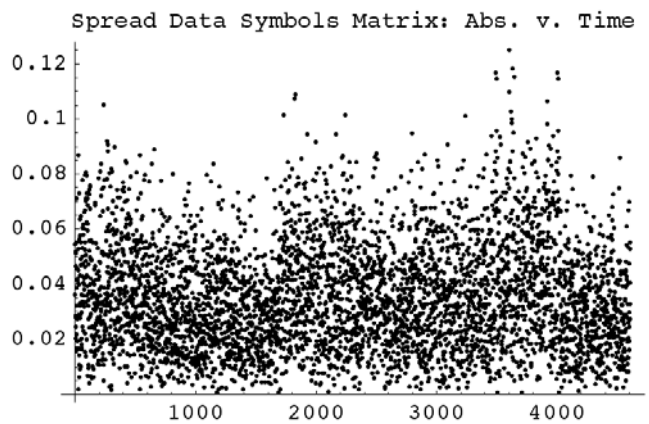


Containerization: Spread Containers



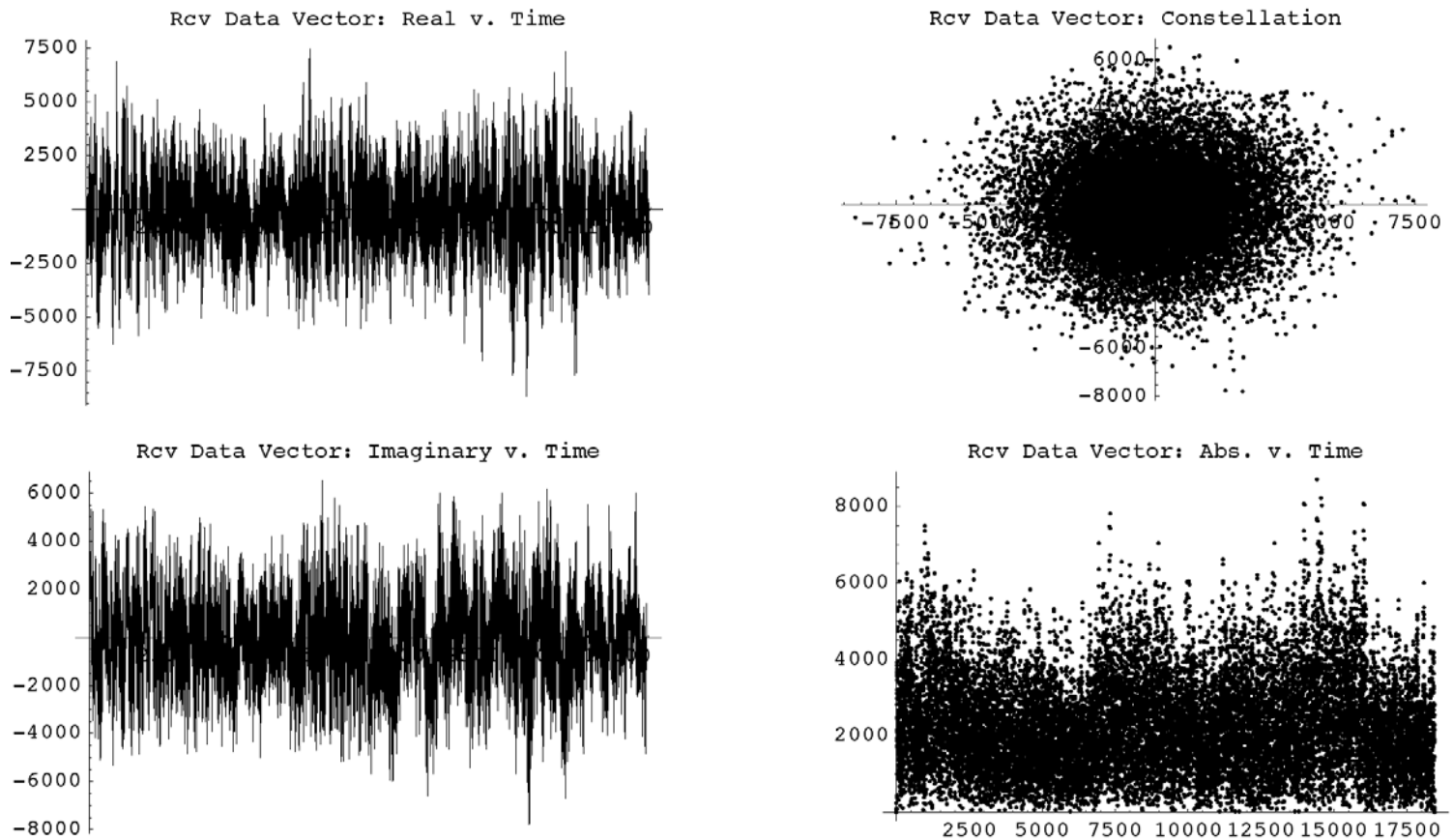




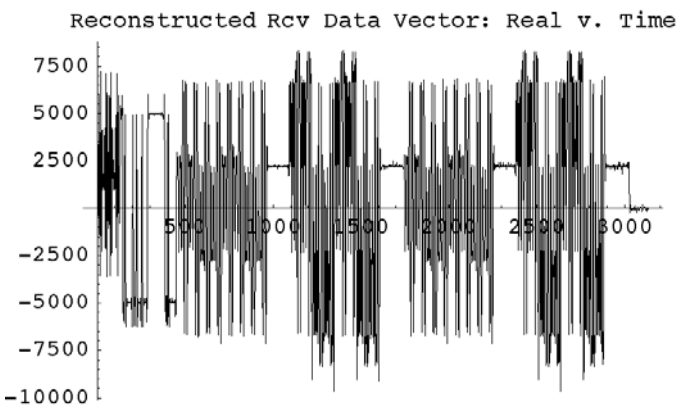


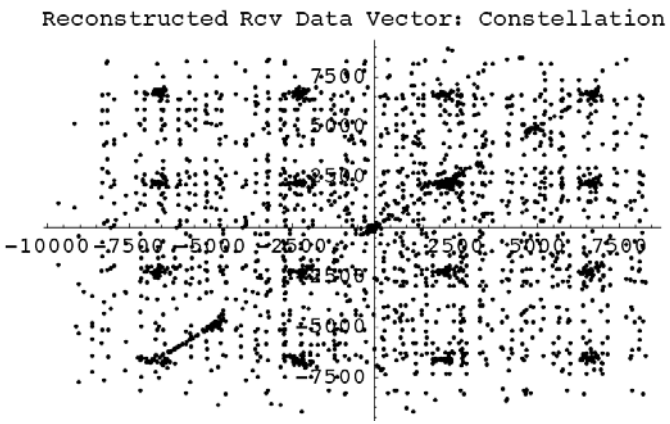
ESTO
Earth Science Technology Office

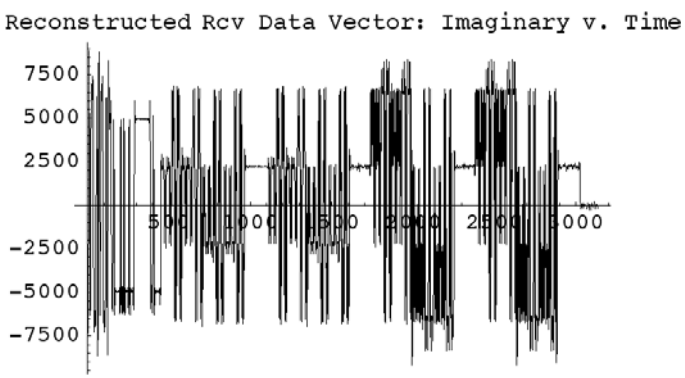
Containerization: Received Data

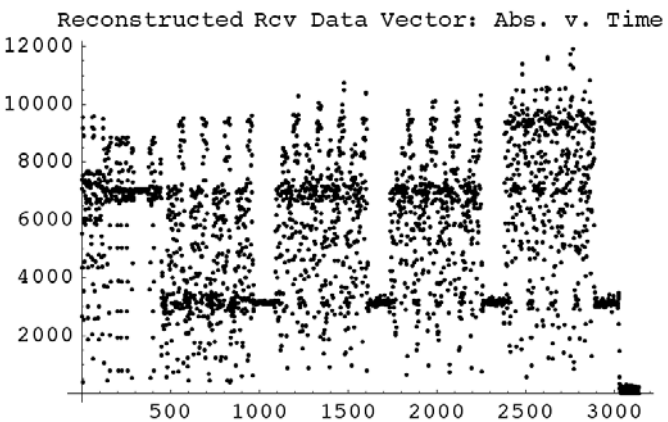


Containerization: Reconstructed Container

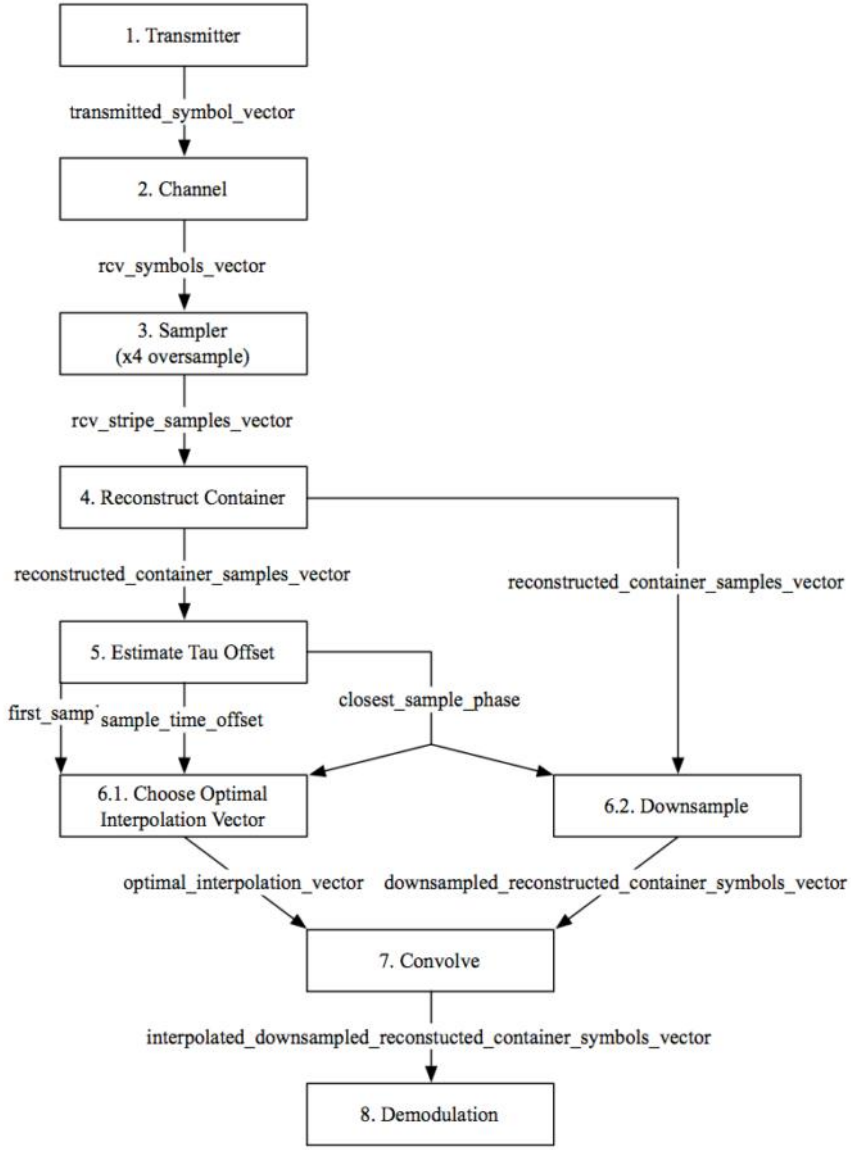




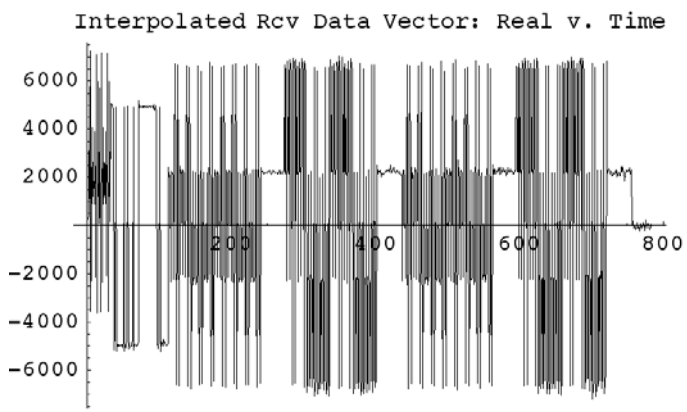


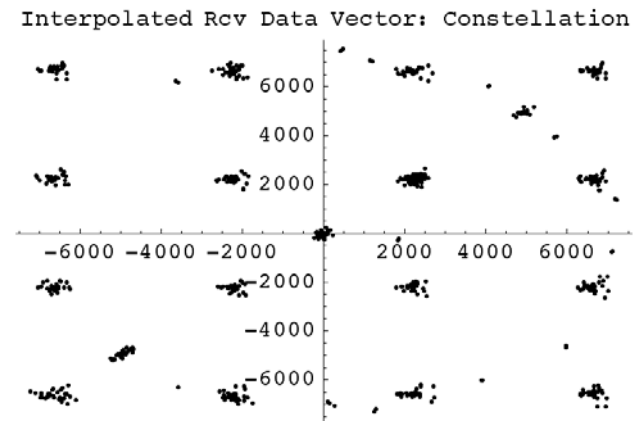


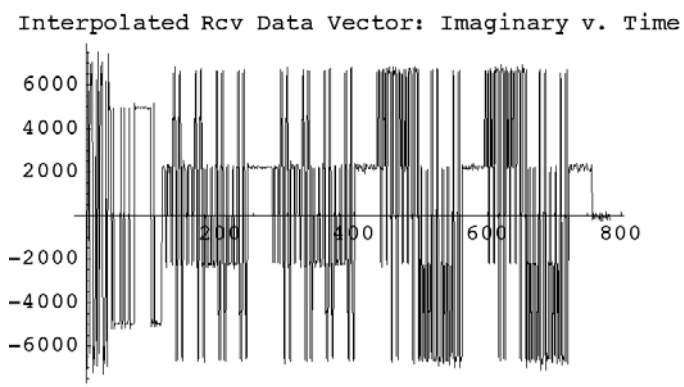
Burst-by-Burst Synchronization and Interpolation

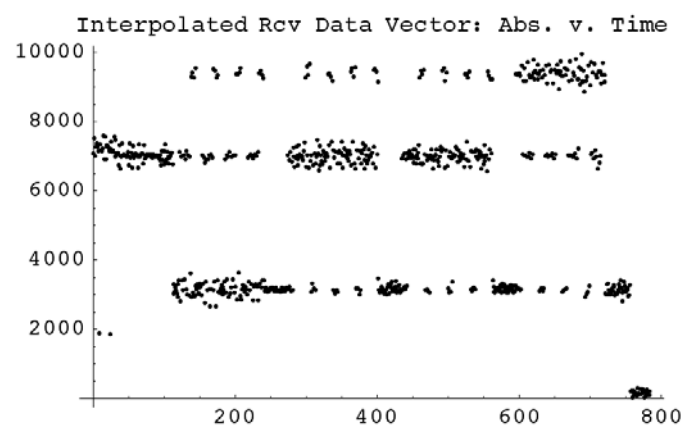


Containerization: Interpolated Container

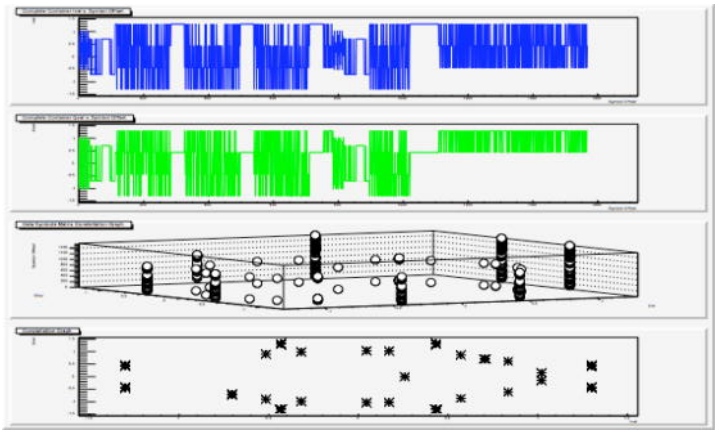


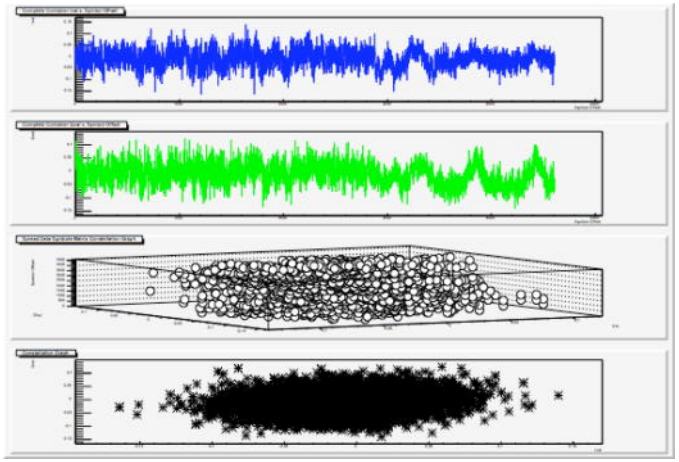






Containerization Summmary





I-vals

Q-vals

3D constellation

2D constellation

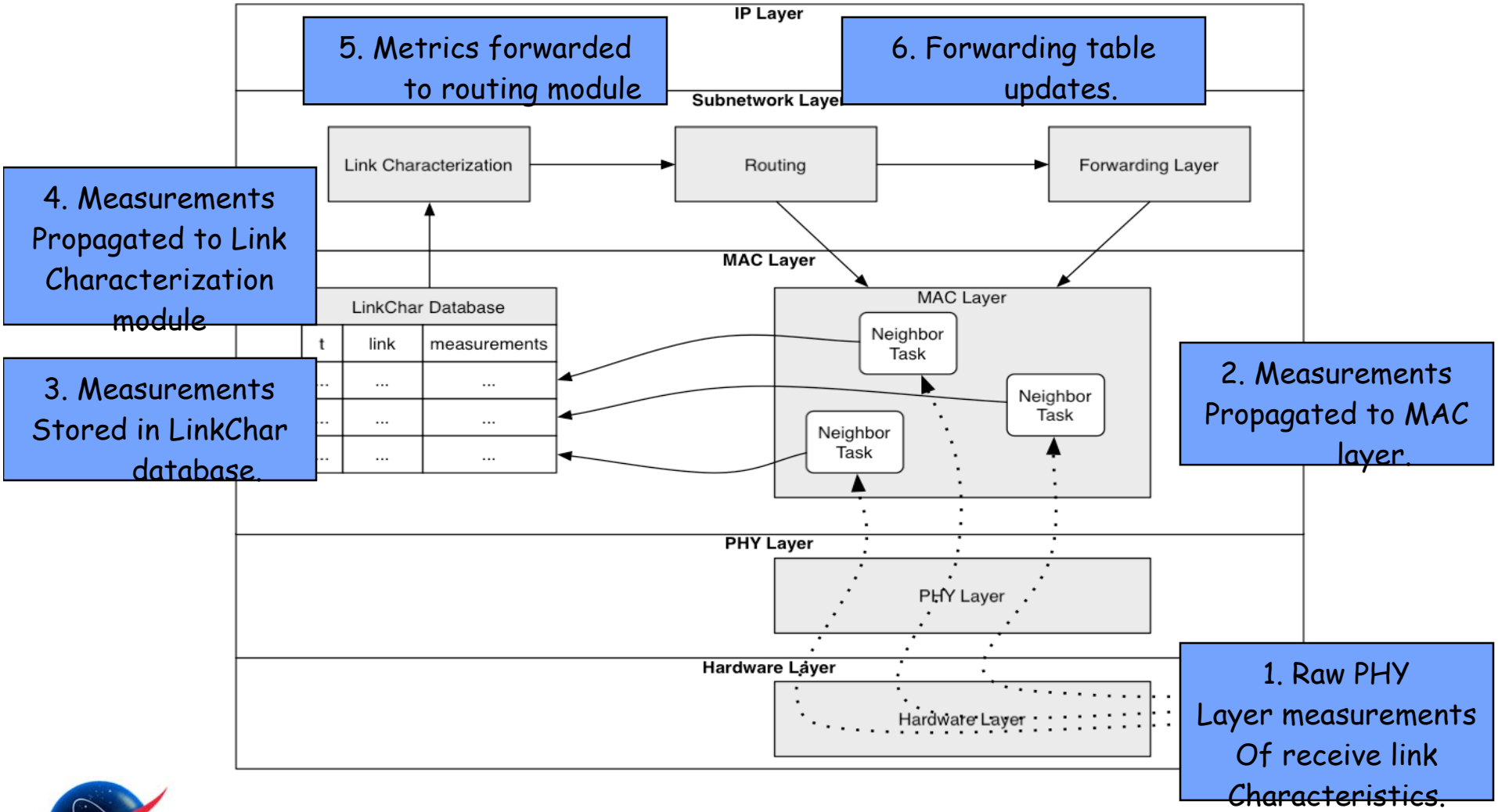
Time Domain

After Spreading

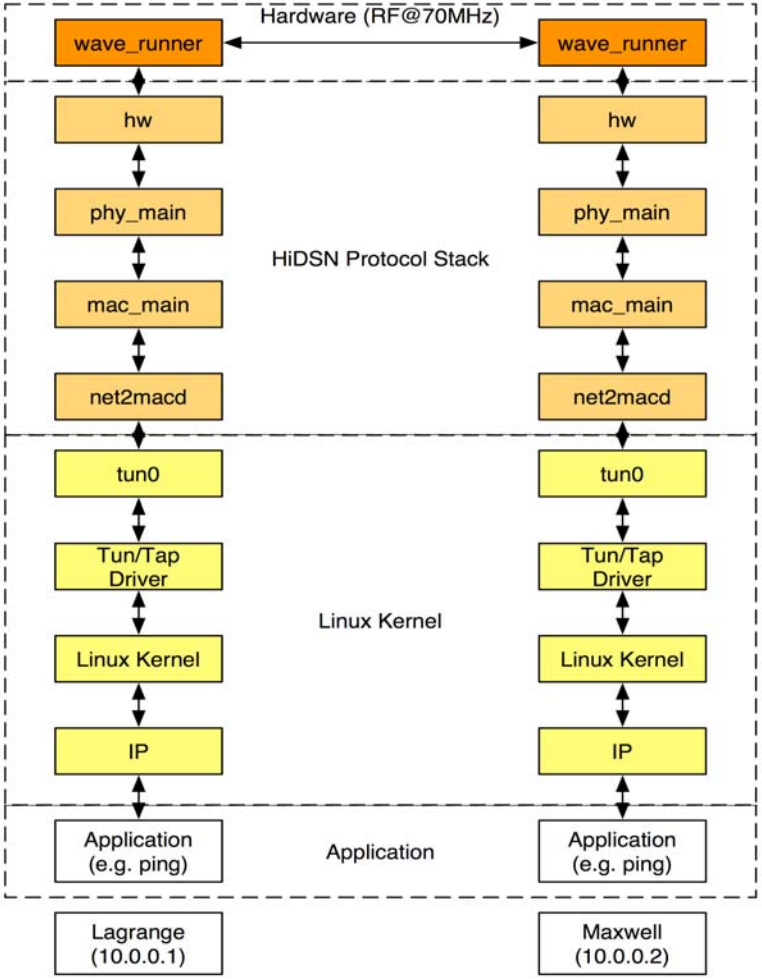
- Space VPN containerization provides highly flexible and variable rate through the use of multi-level M-ary QAM modulation schemes.
- Container format supports the changing of the level of modulation and hence rate on a container by container basis.
- Sample container waveforms are shown above.
- Time domain version of modulated and containerized information shown on left
- Same container after spreading shown on the right.

Results Set #4: IP Integration and Support

Support for IP

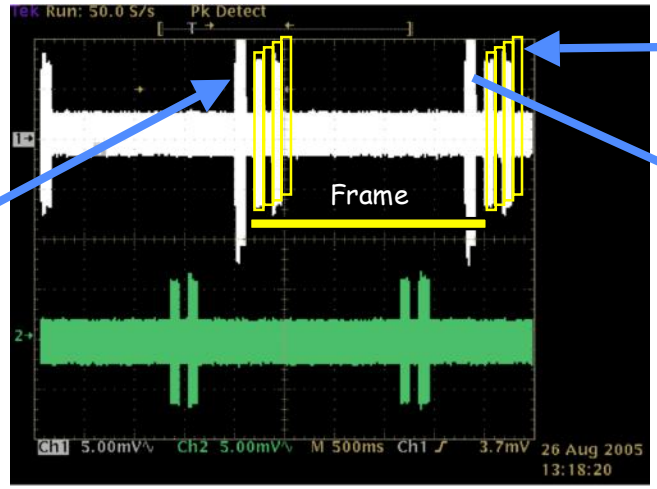


IP Stack Integration



- On transmit, IP packets from applications such as ping are forwarded to the Linux IP stack.
- IP packets are intercepted by the Linux kernel and forwarded to the HiDSN protocol stack via the Tun/Tap driver and "net2macd" daemon
- Packets then traverse the HiDSN protocol stack and are transmitted via RF hardware
- On the receive side, the reverse process is used.

IP Data Transmission

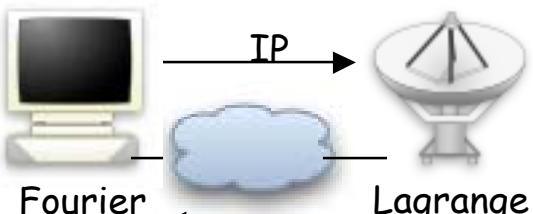


Sync + Contention Bursts

IP Data Bursts

IP Data Bursts with Time/code/spatial multiplexing

Experimenter



Fourier (10.0.2.1)

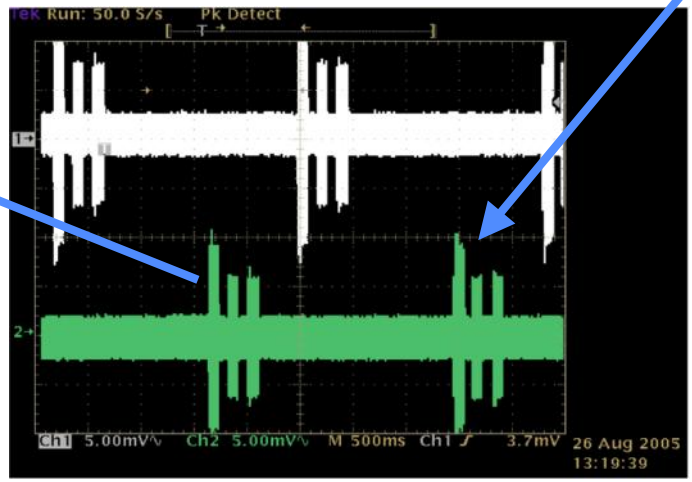
Lagrange (10.0.0.1)

Maxwell (10.0.0.2) → Kepler (10.0.3.1)

Sensor Satellite

On-Board Instrument

IP Data Bursts



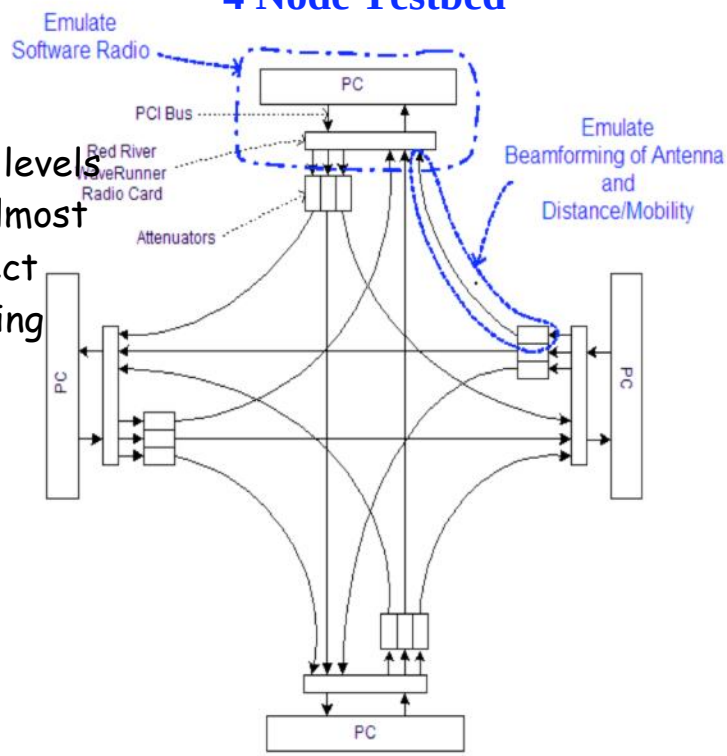

```
root@fourier:[root@fourier root]#  ping -I 10.0.2.1 10.0.3.1
PING 10.0.3.1 (10.0.3.1) from 10.0.2.1 : 56(84) bytes of data.
64 bytes from 10.0.3.1: icmp_seq=1 ttl=62 time=59023 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=62 time=60197 ms
64 bytes from 10.0.3.1: icmp_seq=3 ttl=62 time=61359 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=62 time=62660 ms
64 bytes from 10.0.3.1: icmp_seq=5 ttl=62 time=62686 ms
64 bytes from 10.0.3.1: icmp_seq=6 ttl=62 time=65272 ms
64 bytes from 10.0.3.1: icmp_seq=7 ttl=62 time=66536 ms
```

Results Set #6

Beam Forming and Mobility Emulation

Beam Forming

4 Node Testbed

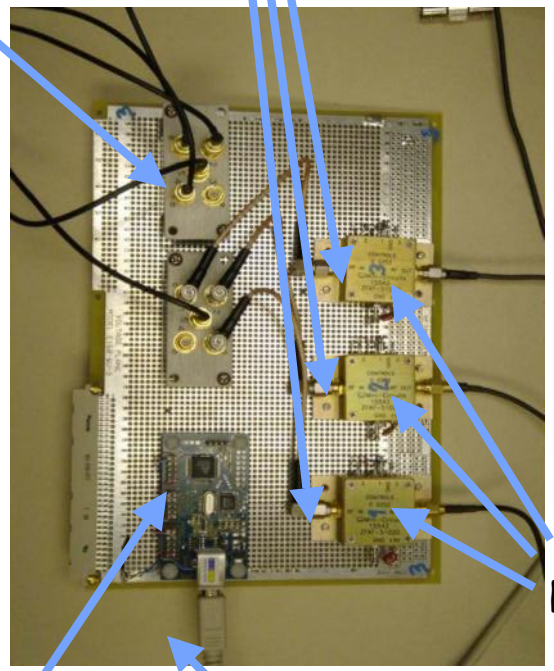


Discrete levels due to almost perfect sampling

- 4 RF Interface Boards
- 3 Neighbors (maximum)
- 12 Unidirectional Links

TTL Controlled Attenuators: 4 - 35 dB

RF In



RF Out

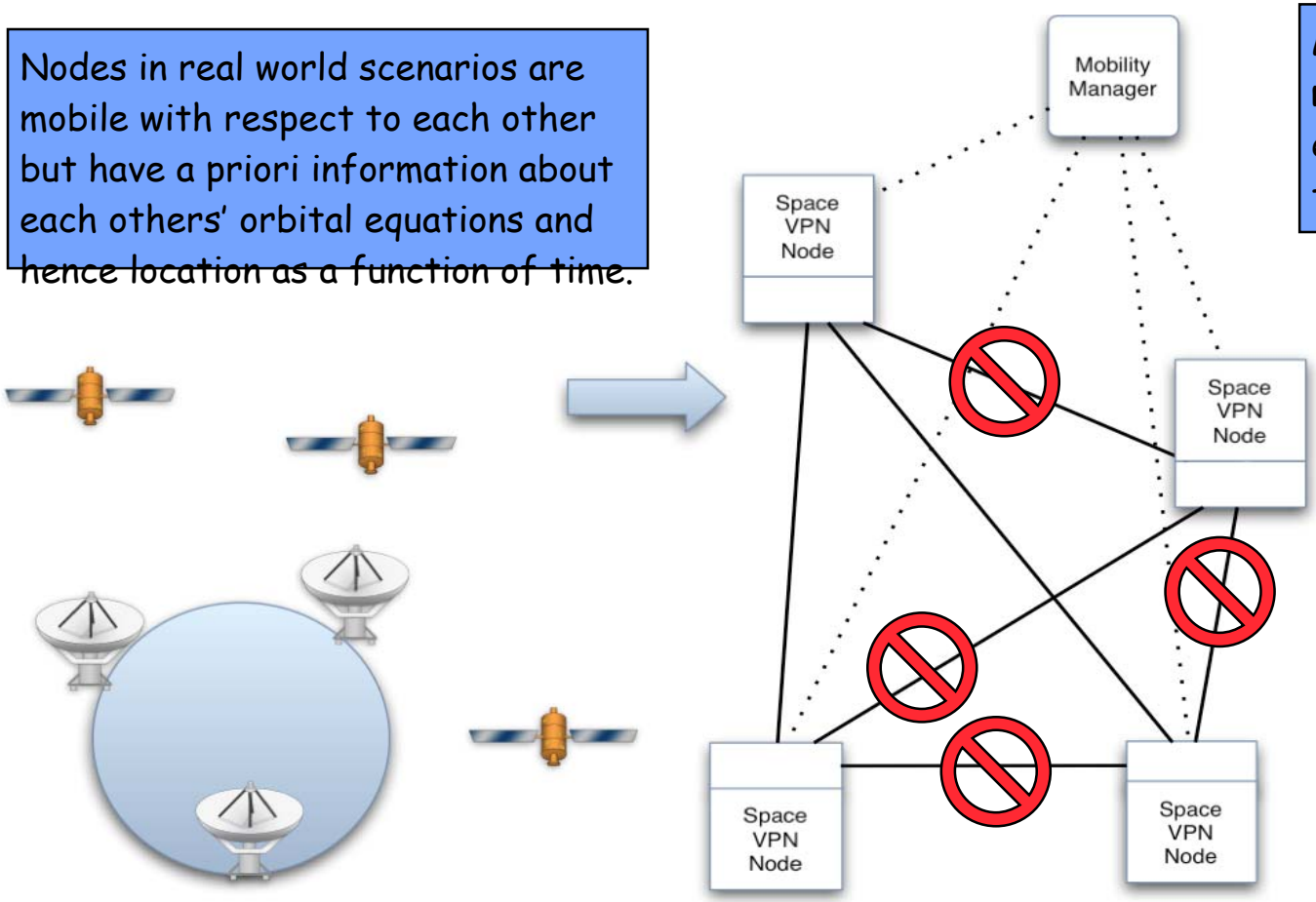
Array Antenna Emulator @ 70 MHz

USB to Computer

USB I/O 24 TTL Controller

Mobility Manager

Nodes in real world scenarios are mobile with respect to each other but have a priori information about each others' orbital equations and hence location as a function of time.



Mobility manager emulates this function.

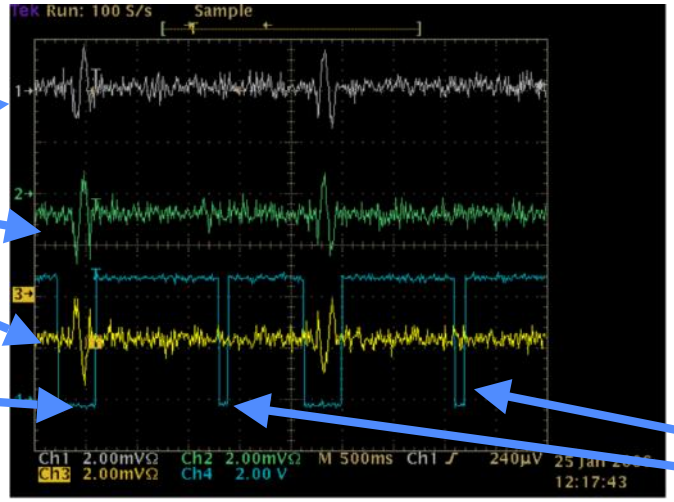
Mobility manager controls attenuation of all transmission links to create dynamically changing topologies.

Beam Forming/Mobility Emulation Results

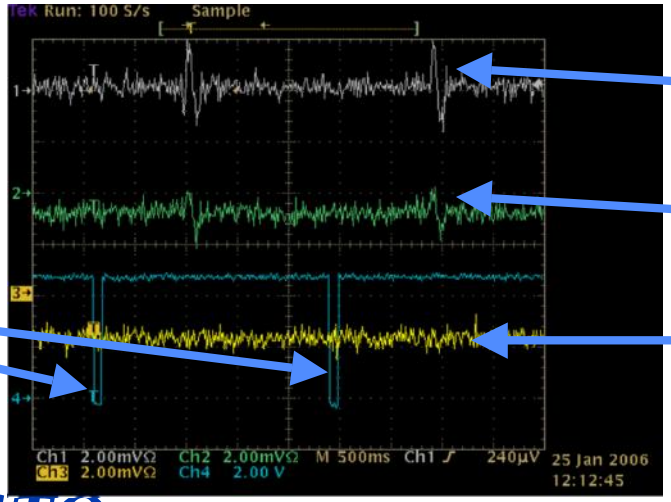
Burst by burst beam emulation.

Transmit beams

TTL Control Pulses
TTL Low: 4 dB attenuation
TTL High: 35 dB attenuation



Start of Frame Indication



4 dB Attenuation

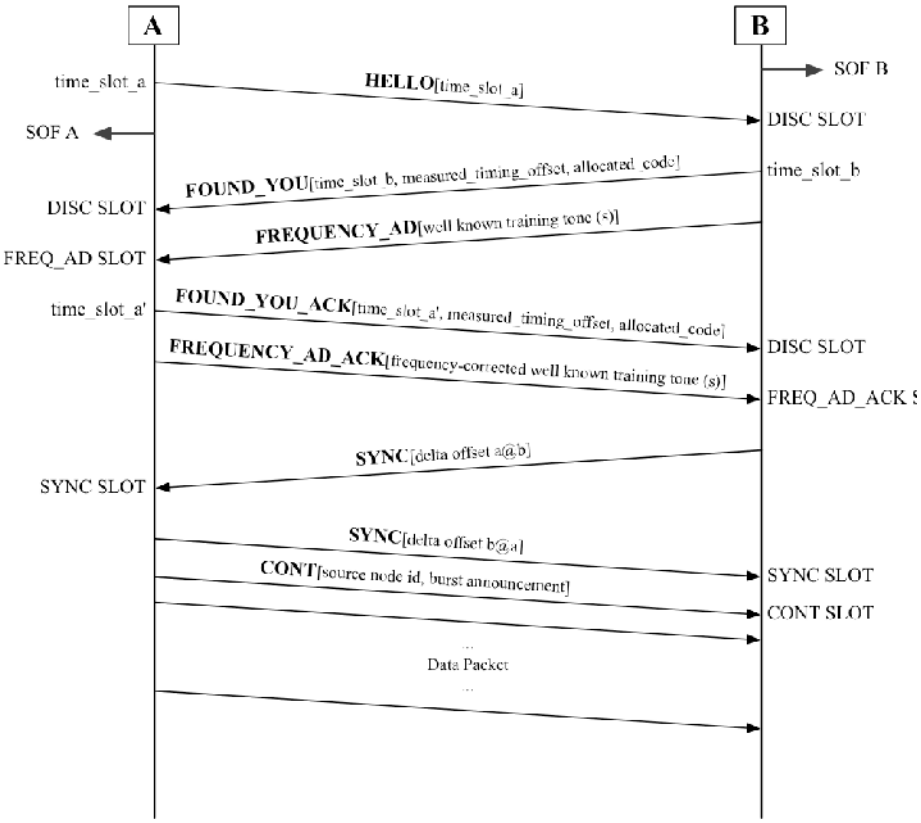
9 dB Attenuation

35 dB Attenuation

Start of Frame Indication

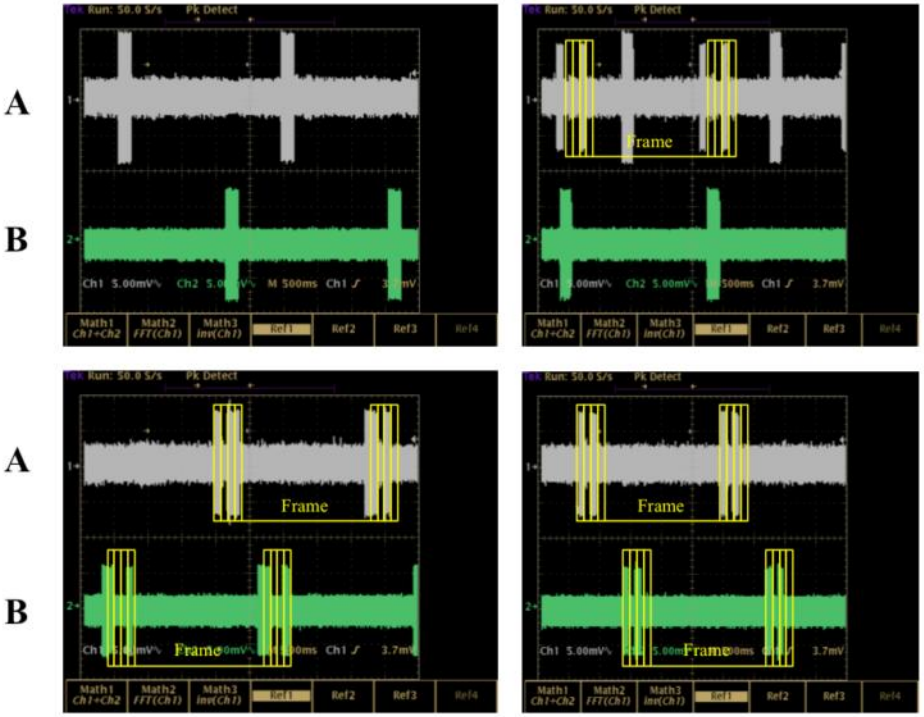
Variable Link Attenuation

Result Set #7: Network Self-Formation



1. **(A)** HELLO, HELLO
(B) HELLO, HELLO

2. **(A)** FREQ_AD, _, _,FOUND_YOU, __, HELLO, FREQ_AD, _, _, FOUND_YOU, __, HELLO
(B) HELLO, HELLO



3. **(A)** SYNC, _, CONT, FOUND_YOU, FREQ_AD, SYNC, _, CONT
(B) FREQ_AD, SYNC, _, _, FOUND_YOU_ACK, __, FREQ_AD, SYNC, _, _,FOUND_YOU_ACK

4. **(A)** SYNC, _, CONT
(B) SYNC, _, CONT

Summary: Testbed Achievements and Results

- **Major Achievements**
  - Emulated dynamic, real-time beams under software control
  - Space VPN GNU+ Software Radio Model

- **Specific results:**
  - IP/SpaceVPN MAC Integration
  - Support for simultaneous, variable rate bursts from different neighbors
  - End to end IP demonstration
  - Integration of link characterization into MAC protocol stack
  - Burst by burst dynamic beam emulation
  - Topology control with mobility manager