System of System Integration Technology and Experimentation (SoSITE)
The Future of Interoperability

John Shaw
Program Manager, DARPA/STO
703-526-2852
john.shaw@darpa.mil

15 September 2016

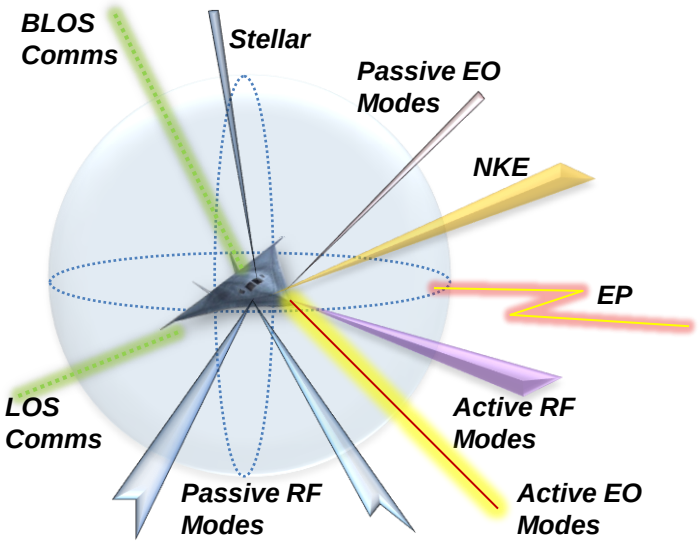
Abstract # 18869

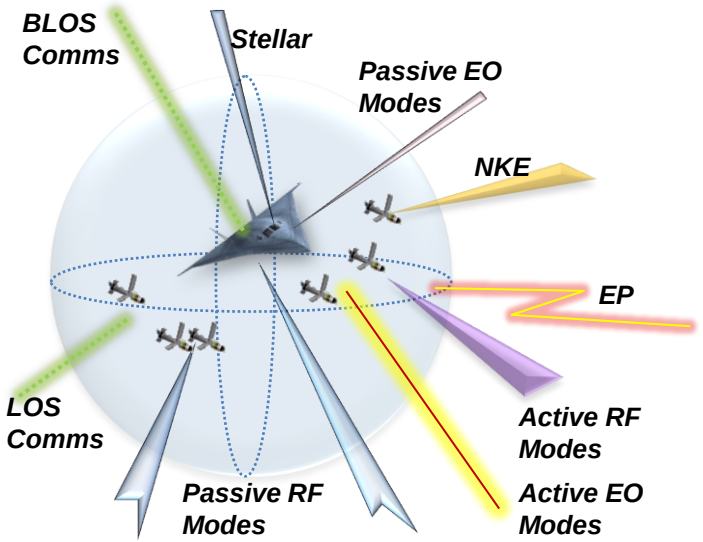
We Need to Rethink how to Maintain Air Dominance

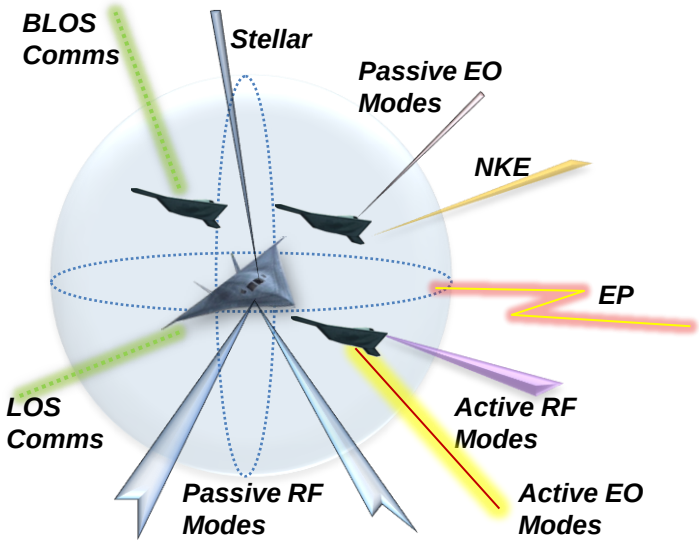
- Manned platforms alone cannot defeat networked proliferated threats
  - Survivability is THE dominant cost driver for manned combat aircraft
  - Imposes an impediment to upgrading in stride with pace of technology
- Mix of high- and low-end platforms can avoid obsolescence better than high-end platforms alone
  - What does interoperability need to be in order to stay apace of technology?

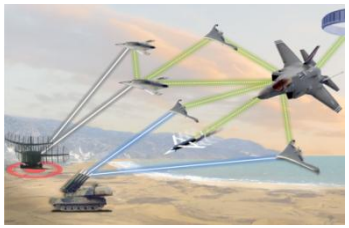
Platform	1st Priority	2nd Priority	Reason
Manned	Survivability	Lethality	Human Life and $$$$$ Vehicle
Attritable UAV	Lethality	Survivability	Mission Needs, then $$$ Vehicle
Expendable UAV	Lethality	Cost	Mission needs and $ Vehicle

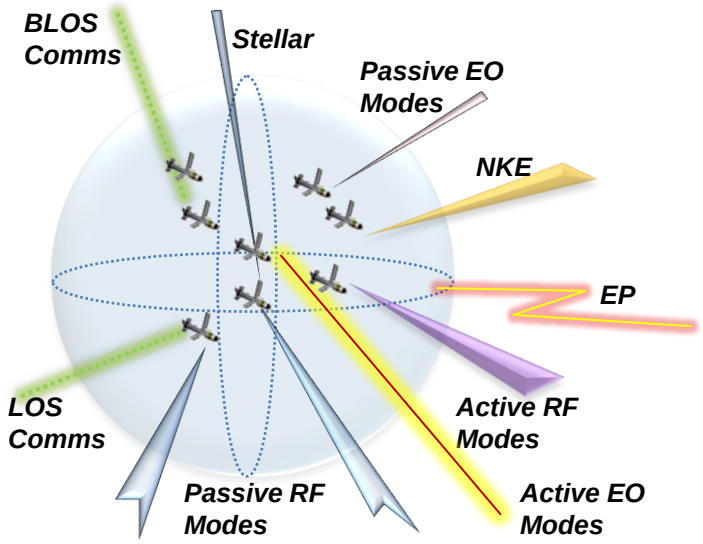
Opportunity: Composable Architectures to Avoid Obsolescence











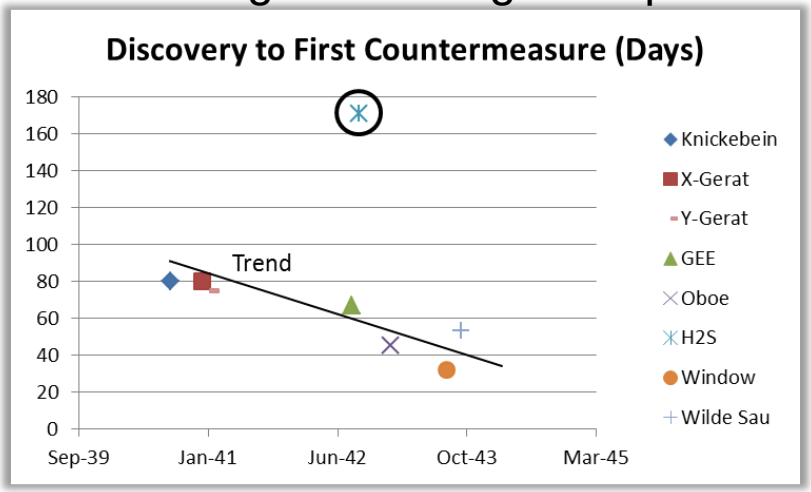
Courtesy of Dr. Joshua Bernstein, Northrop Grumman Electronic Systems

PED: Processing, Exploitation, Dissemination

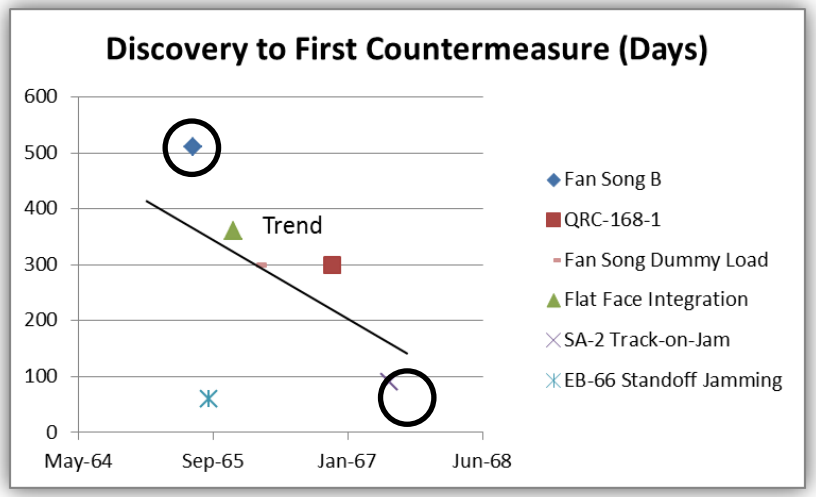
Systems Have Short Effective Lifetimes Once Adversaries Discover Their Countermeasures

- Time to develop a system usually longer than the time to develop its countermeasure
  - Response time quickens as Combat Darwinism kicks in
- The best is the enemy of the good
  - We may be able to approach the best with informed assemblies of the good
  - What principles for composability might we adopt to achieve this?

WW II Night Bombing Competition



Rolling Thunder Competition



What Do I mean by Composable Systems of Systems and Why Should We Care?

Global Interoperability without Global Consensus

Not necessary for parties to agree globally on capabilities their systems will provide or how they will interact

Compose then Optimize

Ability to optimize the systems and their inter-relationships to fulfill your need after you have selected those systems

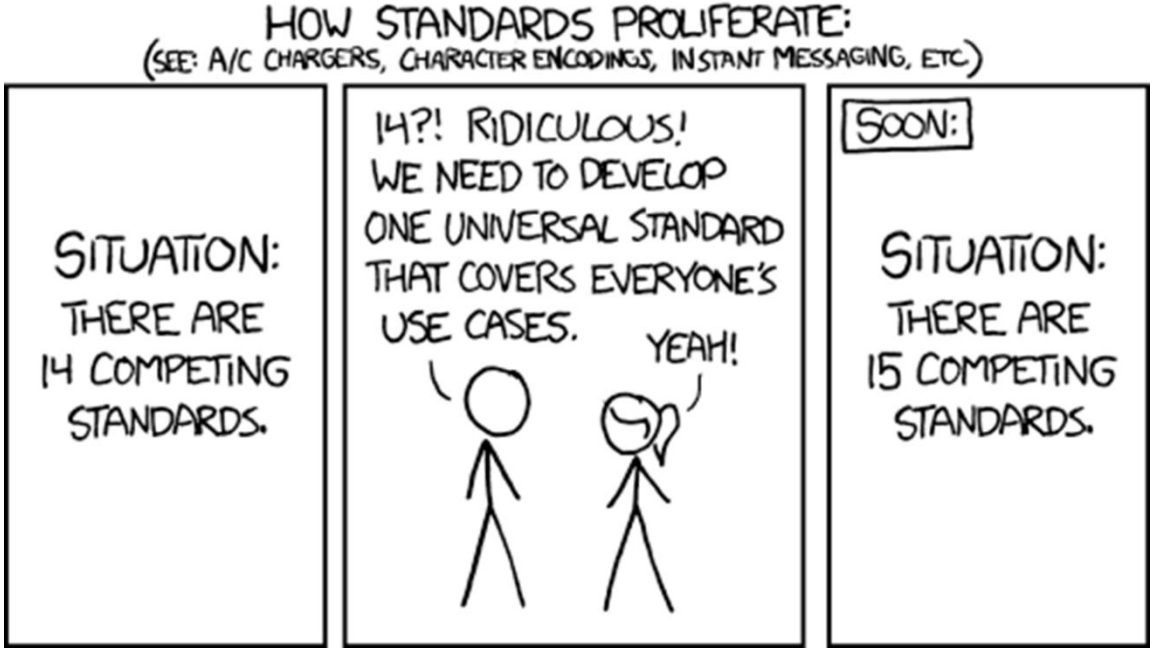
Float and Flow

Don't settle on the capabilities you need, the systems you will use, and the roles they will play until you must fulfill that need

We should be able to create capabilities when we need them from whatever we have in the Nation's S&T portfolio

Open Architectures Helpful, but Rethink Them

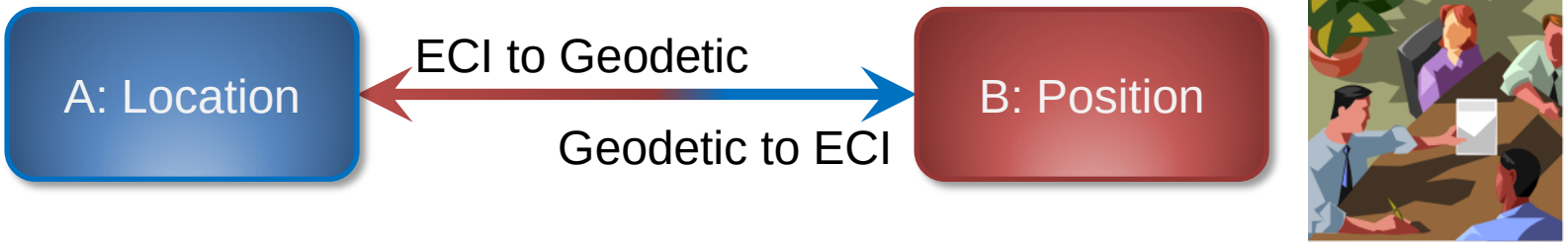
- Today's Open Architectures all require global consensus on message interfaces or data models (e.g., (e.g., Link-16, Link-11, DAML)
  - Consensus has to work for all: company-specific variations discouraged
  - Consensus takes time, creating barriers to new technologies
- How do we preserve OSAs when technological change outpaces them?



Courtesy of XKCD

Can Local Consensus Replace Global Consensus? **Baby Steps**

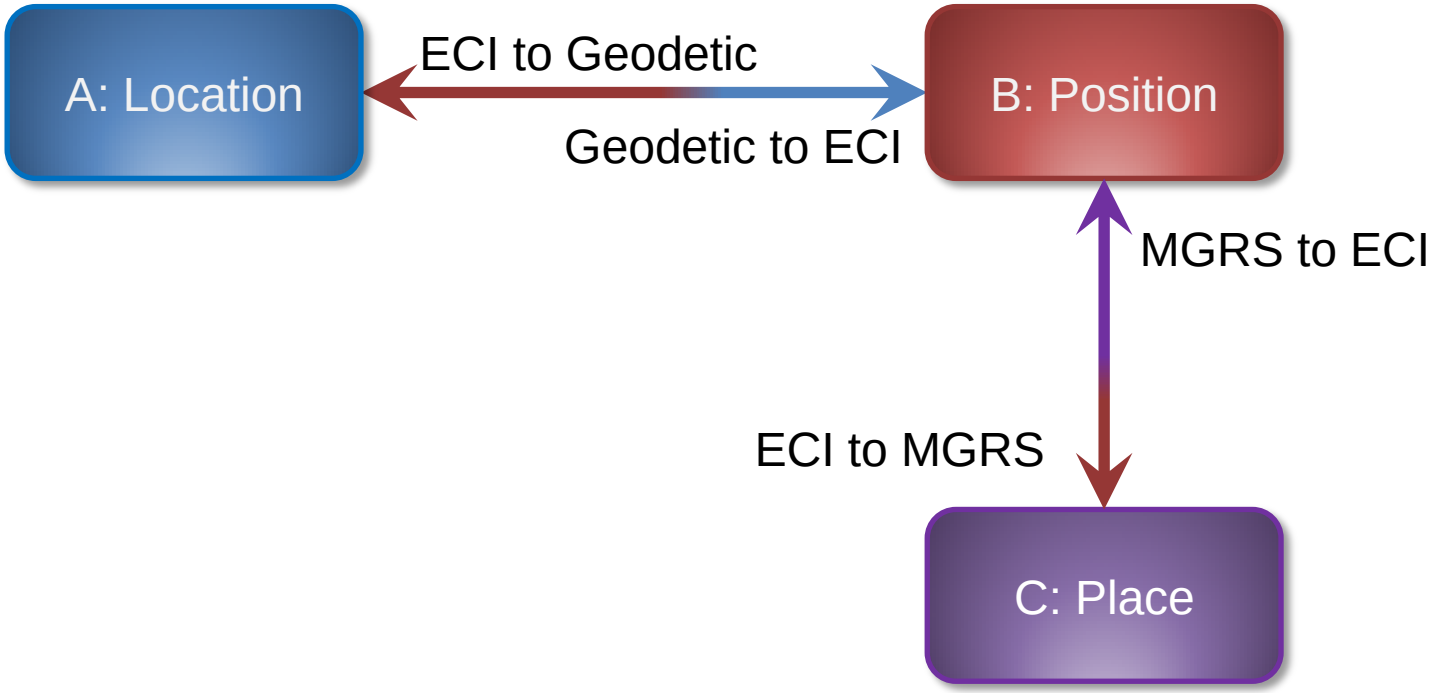
- Organization A uses Geodetic coordinates in its systems to report *Location*
- Organization B uses Earth-Centered-Inertial to report *Position*
- Organization C uses the Military Grid Reference System to report *Place*
- A and B agree on precise rules to translate to and from Geodetic and ECI



- B and C develop precise rules to translate to and from ECI and MGRS

Local Consensus **CAN** Replace Global Consensus

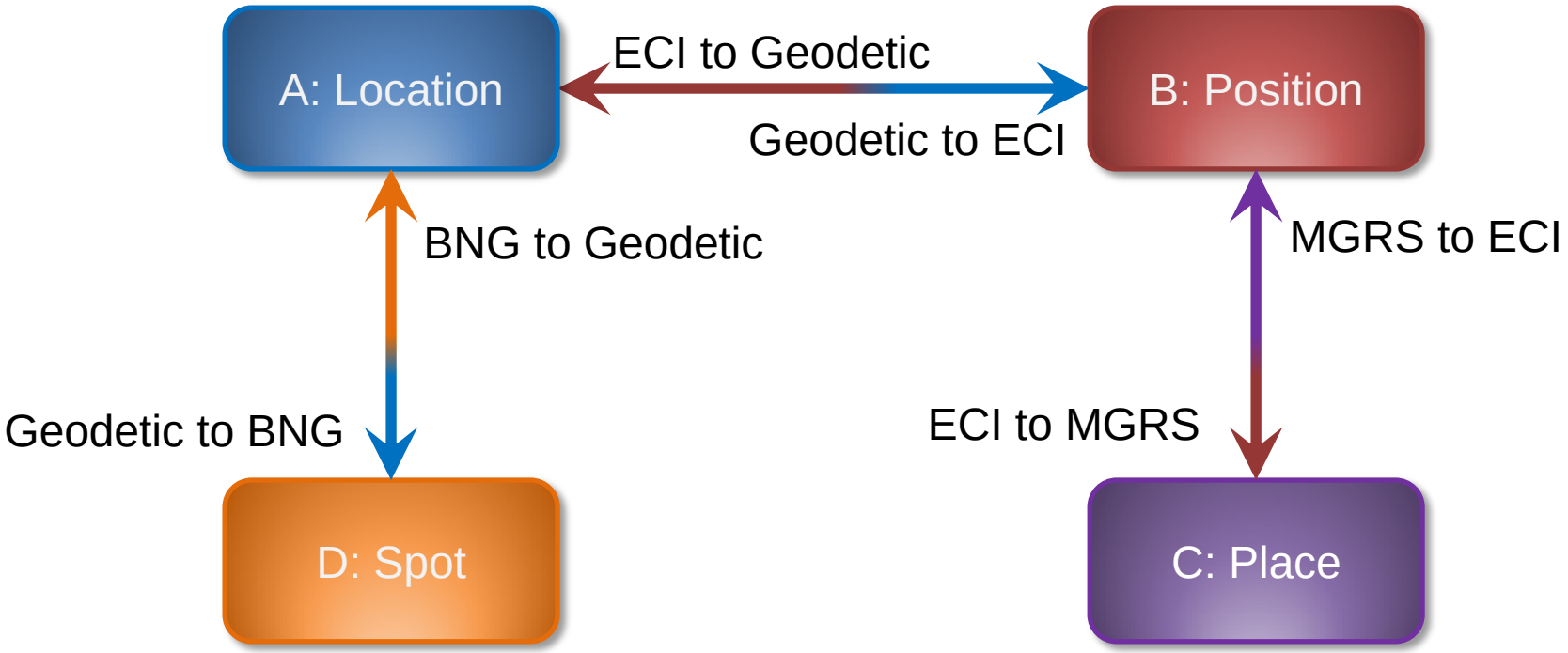
- Use the pair-wise agreements to achieve interoperability between A & C



- A ***Consensus Network!***

Extending a Consensus Network is Not Hard

- Organization D, a British company, uses British National Grid for *Spot*
- D simply agrees on translations with ***one*** of the three, e.g. Organization A, to attain interoperability with ***all*** of the three:



A Few Observations

- Software will easily assemble translation steps
  - An expression such as D.Spot == C.Place can instruct software to the sequence of translations from one to the other and back
- We don’t need to write code to assemble interoperable interfaces
  - Code generators will do this
- We don’t need to select translation sequences until we need to connect specific systems
  - Code generators will locate and assemble the translation steps for us
  - *Float and Flow!*
- Chaining translations doesn’t imply we get inefficient interfaces
  - Compliers will eliminate unnecessary or redundant computations when they compile the auto-generated code into machine binaries
  - *Compose then Optimize!*

Achieving Extendibility

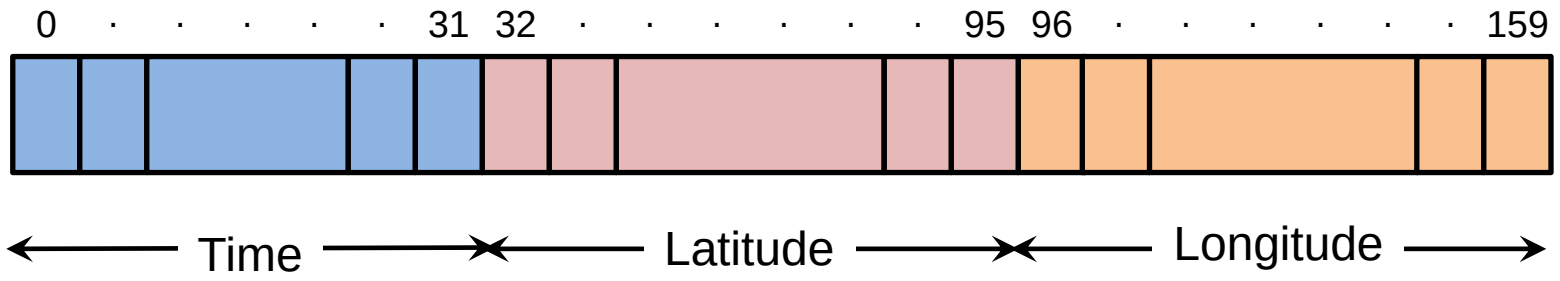
- Systems designed to be upgraded frequently often use extendible datatype messages
- Example: conveying Time, Latitude, and Longitude within a Position message using Xtendible Markup Language (XML):

```
<Message>
    <Name>Posjition</Name>
    <FieldType>
        < Name>Time</Name>
        <FieldCode>4 byte</fieldCode>
    </FieldType>
    <FieldType>
        < Name>Latitude</Name>
        <FieldCode>8 Byte</fieldCode>
    </FieldType>
    <FieldType>
        < Name>Longitude</Name>
        <FieldCode>8 byte</fieldCode>
    </FieldType>
</Message>
```

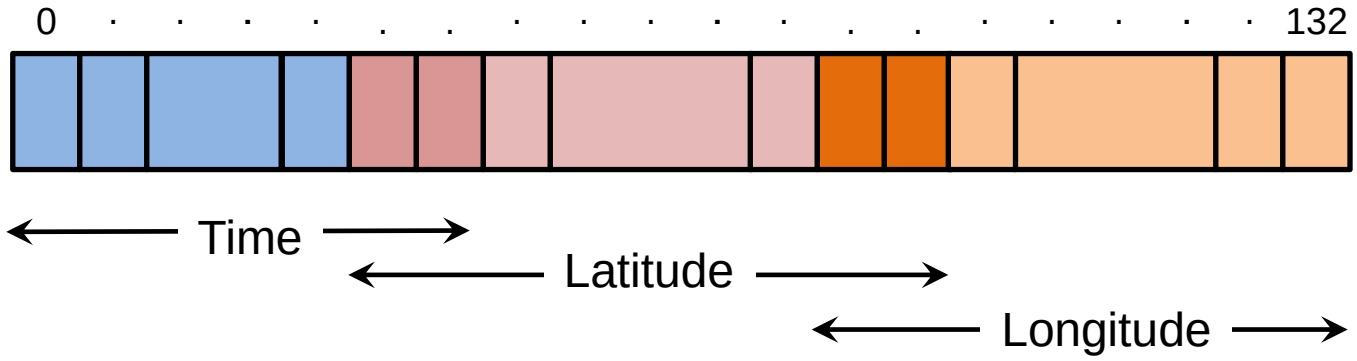
- The message provides all the information needed to understand what it is conveying, but it is verbose!

Achieving Efficiency

- Systems designed to run in real-time use bit-packed messages to achieve high throughput, low latency
- Conveying the same 20 bytes for Time, Latitude, and Longitude in a 160-bit message…..

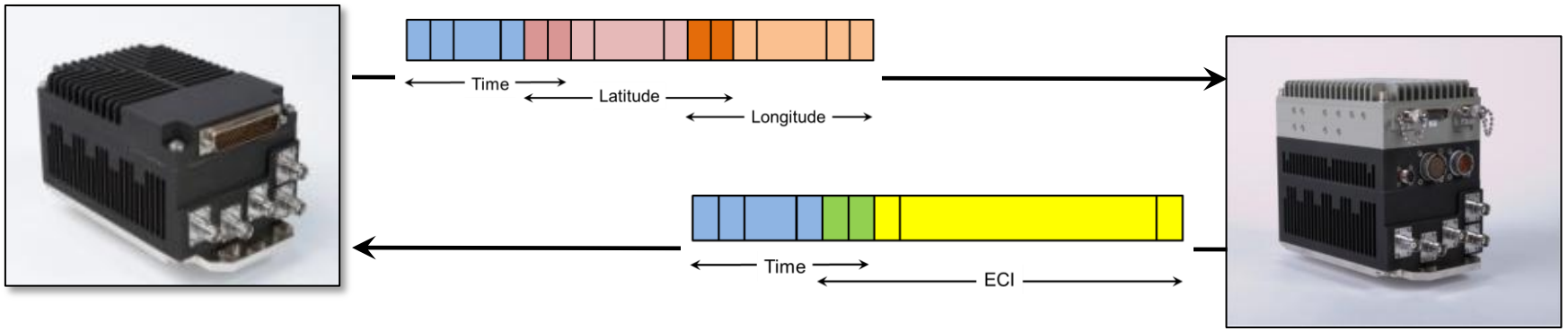


- ….and conveying the same information with even fewer bits using compression



Extending Consensus Networks to Bit-Packed Messages

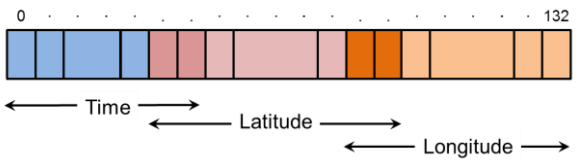
- Interoperability between systems using ***Different*** bit-packed schemes without either developer knowing the representation used by the other!



Can We Achieve Extendibility & Efficiency?

Extendibility

```
<Message>
    <Name>Posjition</Name>
    <FieldType>
        < Name>Time</Name>
        <FieldCode>4 byte</fieldCode>
    </FieldType>
    <FieldType>
        < Name>Latitude</Name>
        <FieldCode>8 Byte</fieldCode>
    </FieldType>
    <FieldType>
        < Name>Longitude</Name>
        <FieldCode>8 byte</fieldCode>
    </FieldType>
</Message>
```



Efficiency

Do I have an Existence Proof? Yes!!

- SoSITE has delivered a prototype tool to the Air Force, Navy, SEI
- Locates the transforms needed to achieve interoperability between messages that aren't designed to a common standard
- Auto-generate the interface code (Java, C++) from those transforms
- Compiles the interface code into compact, highly-efficient executable binaries
- Navy is using on the CNO's Interoperability Demo for Task Force Netted Navy
- Software Engineering Institute is evaluating for new missile defense architectures

Prototype Training Sessions at MITLL

www.darpa.mil

Experiment To Test Efficiency Hypothesis

- Subsystems (All Subsystems Are C++ Cores and Use Packed Interfaces)
  - R: Radar Source SS, uses R.PackedMessage with Time and Dets
  - T1: Tracker Destination SS, uses R.PackedMessage (no transform needed)
  - T2: Tracker Destination SS, uses T2.PackedMessage (only changes Time units)
  - T3: Tracker Destination SS, uses T3.PackedMessage (switches Lat and Lon in Dets)
  - T4: Tracker Destination SS, uses T4.PackedMessage (change Time and Dets units)
- Each Interface Has both a Packed and Unpacked FTG Node with (Painful) Transforms between them
- We Consider Two Pathways through the FTG
  - PUUP: $\text{Packed}_{\text{Source}} \rightarrow \text{Unpacked}_{\text{Source}} \rightarrow \text{Unpacked}_{\text{Destination}} \rightarrow \text{Packed}_{\text{Destination}}$
  - PP: $\text{Packed}_{\text{Source}} \rightarrow \text{Packed}_{\text{Destination}}$
- Hypotheses that We Are Looking to Test:
  - Can we Compile the PUUPs to an acceptable level of performance?
  - How does the performance of the Optimized PUUPs compare to the hand-generated PP transforms?

Comparison Run-Time Performance

[Packed → Unpacked → Unpacked → Packed] vs. [Packed → Packed]

APOGEE RESEARCH

Connection	PUUP vs PP	Java HCALS		C++ HCALs	
		Speed Mbps	Latency ms	Speed Mbps	Latency ms
R1 -> T1 (No Transform)	PUUP	3000±35	1.1±0.1	2889±52	0.7±0.1
R1 -> T1 (No Transform)	PP	3005±18	1.0±0.1	2897±38	0.7±0.1
R1 -> T2 (Only Change Time)	PUUP	1972±18	1.1±0.1	2891±38	0.7±0.1
R1 -> T2 (Only Change Time)	PP	1967±22	1.2±0.1	2889±53	0.7±0.1
R1 -> T3 (Switch Order Lat, Lon)	PUUP	1100±9	1.5±0.1	1035±32	1.1±0.1
R1 -> T3 (Switch Order Lat, Lon)	PP	1058±9	1.6±0.1	1042±25	1.2±0.1
R1 -> T4 (Change All Fields)	PUUP	685±5	2.0±0.1	963±23	1.2±0.1
R1 -> T4 (Change All Fields)	PP	755±7	1.9±0.1	898±21	1.3±0.05