ChatGPT

Plan for standards and protocols

1 Design philosophy and high-level standards

- **Modular Open Systems Approach (MOSA)** – Design systems as a collection of self-contained modules with clearly defined, stable interfaces. The DoD's MOSA guidance emphasises employing modular design, designating key interfaces, using open standards and verifying conformance [1]. Open interfaces allow subsystems to be independently upgraded and replaced, avoiding vendor lock-in [1].
- **Open architecture frameworks** – Leverage open frameworks to accelerate integration and maximise reuse. Anduril's Lattice OS was built from the ground up as an open architecture that integrates data from Anduril and third-party sensors and platforms; its open design follows standards such as Open Mission Systems (OMS) / Universal Command and Control Interface (UCI), the Future Airborne Capability Environment (FACE) and broader MOSA principles [2]. Using these frameworks as inspiration ensures that custom solutions remain interoperable with existing defence ecosystems.
- **Standard selection** – Identify which open standards apply to each domain. Under the MOSA umbrella the DoD highlights several open standards including SOSA (Sensor Open Systems Architecture), FACE, VICTORY, MORA, HOST and CMOSS. Each addresses a different domain (see table below). Multiple standards can coexist because they cover different layers (e.g., SOSA for sensor modules; VICTORY for in-vehicle networks).

Domain	Applicable open standards	Notes
Sensor & mission computing	**SOSA (Sensor Open Systems Architecture)**	Focuses on open interfaces for sensors (communications, EO/IR, EW, radar, SIGINT) and supporting equipment. Promotes modular cards and defined pinouts so that sensors can be swapped without system redesign.
Avionics & airborne systems	**FACE (Future Airborne Capability Environment)**	Provides an open avionics environment that encourages portable software across aircraft. Uses profiles with defined APIs for portable flight software.
Ground vehicles & C4ISR/EW	**VICTORY** – defines an Ethernet/IP based in-vehicle data bus and messaging interfaces [3]; **MORA (Modular Open RF Architecture)** and **CMOSS** for RF payloads and integrated chassis	VICTORY provides system-level and component-level interfaces for C4ISR/EW subsystems. CMOSS defines plug-in cards that can be inserted into common chassis to add Position/Navigation/Timing, radios or mission applications.
Hardware modules & plug-in cards	**HOST (Hardware Open Systems Technology)**, **CMOSS**	Provide mechanical/electrical specifications for modules; align with SOSA and VICTORY backplanes.

Domain	Applicable open standards	Notes
Software integration & distributed systems	**DDS (Data Distribution Service)**, **Open Mission Systems (OMS)/ UCI**	DDS is a data-centric publish-subscribe standard used as the data transport layer in SOSA and FACE [4]. OMS/UCI provides common interfaces for command and control messaging.

2 Data transport and messaging protocols

The system should support more than one data exchange protocol, selecting the appropriate tool for each requirement (latency, throughput, security).

2.1 Real-time distributed data

- **DDS (OMG Data Distribution Service)** – A high-performance publish-subscribe standard that defines both an API and a Real-Time Publish-Subscribe (RTPS) wire protocol. DDS is widely used in mission-critical systems and underpins the SOSA and FACE architectures [4]. It allows different vendors' implementations to interoperate and supports fine-grained Quality-of-Service tuning.
- **DDS-XRCE and DDS-RPC** – Extensions for resource-constrained devices and remote procedure calls [5]. Use these for microcontroller-based sensors or when request/response semantics are preferred.

2.2 Message queuing & IoT

- **MQTT/AMQP/CoAP** – Lightweight broker-based protocols. They are simpler than DDS and well-suited for Internet-connected devices but may not meet defence real-time requirements. Use them for non-time-critical telemetry or remote management.
- **OPC UA** – Industry standard for industrial sensors; integrates with DDS via defined bindings [6]. Consider this where equipment already uses OPC UA.

2.3 Legacy & tactical data links

- **Link 16 / Variable Message Format (VMF) / Cursor on Target (CoT)** – Common tactical datalinks supported by Anduril's Lattice OS [2]. Include gateways to translate internal messages to/from these formats for external systems.
- **STANAG 4586 (UAV C2)**, **MAVLink**, **UAVCAN** – Use these when integrating UAVs that require standardized ground control interfaces or autopilot messaging.
- **NMEA 0183/2000** for GPS and marine sensors; **AIS** for maritime tracking.

2.4 Protocol translation and middleware

Systems rarely speak the same protocol. A middleware layer or gateway can translate between DDS, MQTT, legacy protocols and tactical links. The MOSA white paper notes that protocol gateways and Enterprise Service Bus (ESB) patterns allow systems with different data models to interoperate [7]. Anduril's Lattice uses microservices and containerized services; a similar approach can host protocol adapters.

3 Network and physical interfaces

- **Ethernet/IP** – Default transport for high-bandwidth sensors. The VICTORY standard defines an Ethernet-based in-vehicle data bus and fully specifies messaging interfaces, including parameter semantics and encoding [3]. Use Cat6/7 or fibre for long runs; support Power over Ethernet when sensors require power.
- **Time-Sensitive Networking (TSN)** – For deterministic Ethernet; beneficial for real-time video or radar sensors.
- **CAN/CAN FD, RS-485/422, I2C, SPI, UART** – Use on embedded boards and low-level sensors. CAN is widely used in automotive and ground vehicles; RS-485 is robust for long cables; I2C/SPI for on-board chips.
- **MIPI CSI/DSI, LVDS, USB 3.1** – Interfaces for high-resolution cameras and LiDAR modules. Choose according to sensor vendor and bandwidth requirement.
- **Wireless links** –
- For high-capacity mobile nodes: 4G/5G NR or Wi-Fi 6E; integrate encryption and VPN.
- **MANET/mesh networking** – Anduril's Lattice Mesh uses resilient mobile ad-hoc networks with frequency agility and end-to-end encryption [8]. Use similar mesh radios (Silvus, Doodle Labs) for DIL (degraded/intermittent/low-bandwidth) environments.
- **Low-power sensors** – LoRaWAN, Zigbee, BLE or UWB depending on range and power.

4 Security and encryption

- **End-to-end encryption** – Adopt strong cryptography such as AES-256 with FIPS-140-3 compliant modules for data in motion [9]. Use TLS 1.3 for TCP-based protocols; for UDP-based DDS, enable DTLS or secure DDS profiles.
- **Authentication & key management** – Implement PKI or pre-shared keys; ensure that each sensor/module has a unique identity. Mesh networks should support dynamic key rollover and device revocation [10].
- **Least-privilege & data classification** – Follow zero-trust principles; segregate networks; use role-based access controls.

5 Software integration and APIs

- **Open APIs** – Lattice OS exposes gRPC and HTTP/RESTful APIs for third-party integration [11]. Use similar API styles for your system. gRPC offers type-safety and performance, while REST is widely supported. Provide language bindings (C++, Python, Java, Go, Rust) to encourage adoption [12].
- **Open data models** – Define canonical data models for sensor observations, tracks, tasks and metadata. Lattice's open data models allow developers to create and interpret C2 tasking messages [13]. Adopt or adapt existing models like OGC SensorThings or OASIS Common Alerting Protocol for events. Use JSON Schema or protobuf to specify.
- **Microservices & containerization** – Follow Anduril's microservices architecture that enables rapid capability insertion [14]. Package services as Docker/OCI containers for edge deployments [15]. Use orchestration (K3s or Kubernetes) on larger platforms to manage lifecycle.
- **Edge computing** – Place processing close to sensors (e.g., Jetson, x86) to reduce latency. Lattice OS supports edge processing on low-power systems and high-performance servers [16]. Ensure that models can update over-the-air with secure channels.
- **Programming languages and frameworks** – Adopt languages with strong concurrency and memory safety (Rust or Go) for new modules. Use existing frameworks (ROS 2 uses DDS, enabling robotics integration; RTI Connext for DDS) where appropriate.

6 Sensor-specific considerations

1. **Electro-optical/infrared (EO/IR) cameras** – High-bandwidth video; use MIPI CSI or USB 3 for short runs and GigE Vision or CoaXPress for longer runs. Use H.265 for encoding; transport over RTP/RTSP or DDS. Provide metadata (timestamp, geolocation) with KLV or MISB standards.
2. **Radar/LiDAR** – Often delivered via Ethernet with proprietary packets; encapsulate in DDS topics. Use time synchronization (PTP/1588) for multiple sensors.
3. **Acoustic/Seismic sensors** – Low data rate; integrate via RS-485, I2C or wireless LoRa; publish via MQTT or DDS.
4. **RF sensing and EW** – Follow MORA/CMOSS for modular cards; define baseband and digital IF interfaces; use VICTORY data bus for platform integration.
5. **GPS/PNT** – Expose NMEA or PTP; share position/time across network via VICTORY PNT services [17].

7 Putting it together

1. **Adopt MOSA principles** – Design the system as modular services with clearly defined interfaces; document interface specifications and select open standards that match the domain. Where multiple standards apply (e.g., VICTORY for vehicle integration and SOSA for sensor modules), adopt both and encapsulate translation layers.
2. **Choose data transport per use-case** – Use DDS as the core real-time backbone for high-bandwidth sensing and control; supplement with MQTT/AMQP for non-real-time telemetry. Provide gateways to tactical data links and legacy protocols.
3. **Build on Ethernet and mesh** – Standardise on Ethernet/IP for wired connections and adopt TSN for determinism; implement resilient MANET/mesh radios for mobile nodes with AES-256 encryption [18]. Support multi-band operation and dynamic waveform selection to cope with contested spectra [19].
4. **Expose open APIs and data models** – Publish sensor data and command/task messages via gRPC/RESTful APIs; use open schemas; provide sample client libraries; ensure backward compatibility.
5. **Implement security by design** – Encrypt all communications; authenticate each module; adopt zero-trust; manage keys centrally.
6. **Validate interoperability** – Develop conformance tests and simulation environments to ensure modules conform to chosen standards. Use test harnesses to validate DDS QoS, VICTORY interface compliance, etc. Perform regular plug-and-play demonstrations.

Potential pitfall / alternative view

While open standards and MOSA principles promise interoperability and rapid upgrades, they can introduce complexity. Ensuring that multiple standards (SOSA, VICTORY, MORA, etc.) interoperate seamlessly requires significant integration effort and rigorous conformance testing [7]. In some cases a simpler, vertically integrated solution using a single vendor's protocol stack may deliver faster time-to-field and lower integration risk. The choice should be guided by the project's lifecycle, threat environment and available resources.

---

[1] [4] [5] [6] [7] FACE & SOSA White Paper
https://www.opengroup.org/sites/default/files/
FACE_TIM_2025_Andre_Connecting%20Open%20Standards%20using%20MOSA_v4.pdf

[2] [8] [9] [10] [11] [12] [13] [14] [15] [16] [18] [19] Anduril Industries Product Cheatsheet: Autonomous Defense Systems & AI
https://cheatsheets.davidveksler.com/anduril-products.html

[3] [17] 200101-DSPJ.pdf
https://www.dsp.dla.mil/Portals/26/Documents/Publications/Journal/200101-DSPJ.pdf