TALOS Computing
Architecture and Software
Development Kit
Introduction

TACTICAL ASSAULT LIGHT OPERATOR SUIT
Leverage all existing solutions possible to achieve Milestone B “Prototype”
Minimize technical risk
Keep it simple, the minimally sufficient solution is best
Invent nothing that is not required to achieve technical intent
ARCHITECTURAL DRIVERS

- Survivability (availability, redundancy, failover, and reliability)
- Interoperability
- Adaptability/Modifiability
- Extensibility/Scalability
- Modularity (at odds with integrated design and Size, Weight, and Power (SWAP) constraints)
- Security
DESIGN PHILOSOPHY

- Open
- Process at the edges
- Data reduction
- Common services
- Priorities
- Mission Modes
- Diagnostics
- Hardware Security Module (HSM) support
TERMINOLOGY

- **Node** – physical component in one location on the suit
- **Sensor node** – small, embedded, interchangeable sensor package connected to suit network, no/minimal Operating System (OS)
- **Sensor gateway** – sensor data aggregator to support common messaging
- **Compute node** – multiple Central Processing Units (CPUs) and OS’s (current plan is 2, primary and backup)
- **Service** – common computing capability shared by multiple applications/services
- **Sensor service** – common computing service to sample, digitize, and publish sensor data on suit network
- **Processing service** – software capability running on compute node(s), optionally driven by a rules engine, managing information delivery and display
- **High availability** – mechanisms to ensure fail over, load balancing, and priority/quota enforcement, relative to current mission mode
HARDWARE DIRECTION

- Overall goal: baseline final hardware as late as possible
- Heterogeneous (x86 and Advanced RISC Machines) processors
- Must run Android Apps (may be ported to minimal Linux OS)
- 2 compute nodes – primary and backup
- 4-8 multicore CPU’s per node
  - Modular, replace processing card with storage
  - Additional backup “go bag” for data logging and comms
- Small form factor PC and high core graphics processing units are current targets
- Virtualized environment, can support limited dedicated OS if required
- GigE or 10GigE switched, *wired* network
  - Potentially open to integration of wireless components also
- Designing to support significant video processing
SENSOR NODES

- Generic model for sensor integration
- Multiple detector elements per physical sensor node possible
- Micro controller with embedded OS
- Pub/sub architecture using common message system
- Plug and play registration of sensor nodes
Evaluating three approaches

1. Non-virtualized - Single OS per CPU
   - Lowest complexity
   - Requires custom high availability (HA) solution
   - Could introduce security and stability challenges

2. Traditional VMs (heavyweight)
   - Additional complexity
   - Data center HA solutions directly apply
   - “Sandboxing” increases security and stability
   - VM size still big

3. Modular OS
   - Same benefits as option 2, but also…
   - Purpose built, highly optimized VM
   - Much smaller size, increased performance, efficiency, and HA support

Goal: All common services in Modular OS

Final solution may be a mix of all 3
VIRTUALIZATION APPROACHES

**CPU**

- Modular OS
- Hypervisor

**OS**

- Service
- App
- OS
- Hypervisor

**CPU**
MESSAGING, INTERFACES, AND API

• Common messaging library key to service interaction and interoperability
  • Protocol buffers over nanomsg
  • Provided via lib_talos and the SDK
  • .proto files and nanomsg interaction semantics define the ICD
  • Example (log message):
    ```
    package talos.net;
    message LogStatement {
      required string sender = 1;
      required int32 logLevel = 2;
      required int64 time = 3;
      required string data = 4;
    }
    ```

• lib_talos provides
  • nanomsg semantics and socket set up abstraction
  • System logging abstraction
  • Service registration/authentication/discovery abstraction
  • Command channel establishment and abstraction
  • Implemented in C with minimal dependencies
  • C++ and Java libraries/bindings in development
  • Other languages expected, implemented as required
• Open source libraries forked at https://github.com/SEI-AMS
  • https://github.com/SEI-AMS/nanomsg
  • https://github.com/SEI-AMS/protobuf
  • https://github.com/SEI-AMS/protobuf-c
  • https://github.com/SEI-AMS/osv
  • https://github.com/SEI-AMS/cppnanomsg

• Build environment
  • Cmake
  • Linux
  • Compatible C/C++ Environment
APPLICATIONS

Example application types:
- System health and status reporting
- Operator health and status reporting
- Team health and status reporting
- Communications system control and interaction
- Moving map services
- Threat identification, tagging, and tracking
- Blue/grey force situational awareness monitoring
- Targeting
- Sensor management
- Terrain analysis
- 3D visual fusion (e.g. terrain overlay, route visualization)
- Intelligence gathering (e.g. audio, video, images, sensor sampling)

Evaluation of existing GOTS apps, gaps, overlaps, etc. during 2015 RPE FY16 – Heavy focus on application porting and development
CONFIGURATION CONTROL AND CONTRIBUTING

• External developers expected and welcome
• Establishing collaboration portal (access restricted and authenticated)
  – Git repository
  – Build server
  – Collaboration wiki
  – Documentation repository
• Need to identify and standardize external system interfaces
• Heavy emphasis on standardized
  – Interface definitions
    ▪ Message formats
    ▪ Interaction mechanisms
  – Application priority, criticality, and quota enforcement
    ▪ As applicable to differing mission modes
LOOKING AHEAD

• SOFIC – initial availability of SDK and lib_talos
• 2015 RPE – focus on:
  – Existing app survey
  – Service development
  – Mapping engine
  – Display alternatives
  – High Availability (HA) and load experimentation
  – Operator interface (HCI) collaboration
• Remainder of FY15
  – Initial implementation of all core services
  – Initial identification of all required applications
  – Full establishment of development collaboration capability and test labs
• FY16 – year of the application
  – Continued refinement of supporting services, infrastructure, and HA
• FY17 – year of integration and testing
  – Application refinement, integration, and initial field trials
THANK YOU

Questions and Comments…