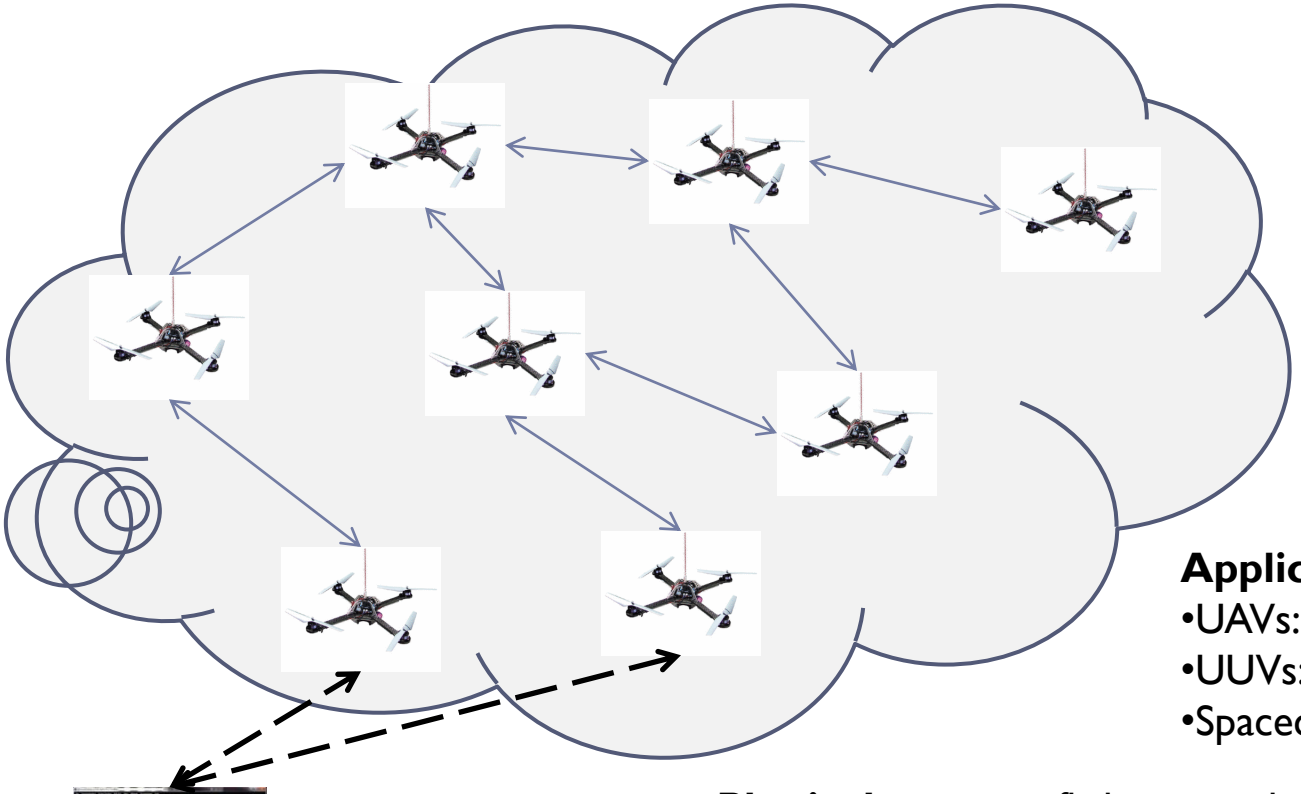
VANDERBILT UNIVERSITY
INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

Integration of Real-time and Security Properties in CPS

Gabor Karsai
in collaboration with Abhishek Dubey and Will R. Otte

A CPS Platform: Vehicle cluster



Vehicles
- Propulsion system
- NAV sensors (IMU)
- Stabilizing controller
- WLAN (Mesh)
- GND comms (OPT)

Payload:
- Compute platform
- Sensor platform

Applications
- UAVs: Wide-area survey
- UUVs: Climate data collection
- Spacecraft: DARPA F6

Ground station

Physical system: flight controls, navigation, vehicle management

Cyber system: control algorithms, networking, sensor processing and storage, (open) computing platform for running 3rd party applications

CPS *integration* challenges in V/C

VANDERBILT UNIVERSITY
INSTITUTE FOR SOFTWARE INTEGRATED SYSTEMS

- Distributed real-time platform with fluctuating network connectivity
  - Real-time and safety properties and their verification
- Dynamic architecture
  - Vehicles of the cluster, software apps running on the platform, security of information flows, location of software and hardware resources used can (be) change(d) at any time
- Secure resource sharing
  - Resources must be securely shared across *applications*: processor, communication links, memory, software services
- Secure resource sharing
  - Resources must be securely shared across *applications*: processor, communication links, memory, software services

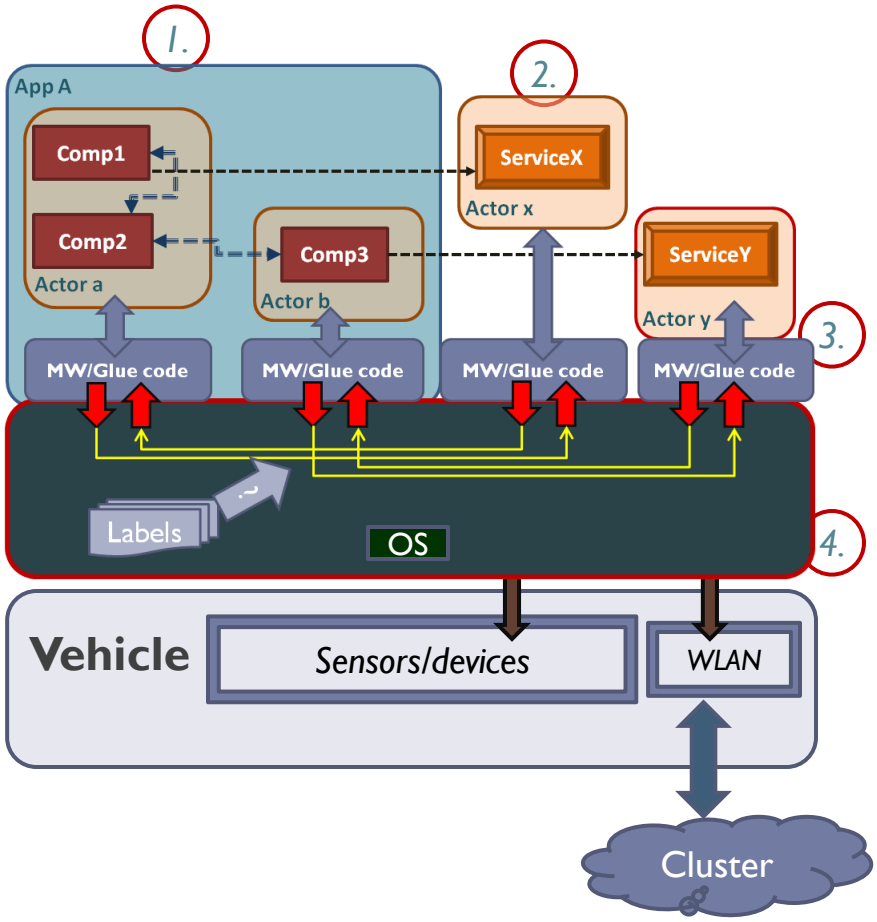
Security challenges

- Communication links are over the air
  - Solutions:
    - Jamming-resistant communications
    - IPSec to protect the network layer
- Open software platform for real-time distributed applications:

Many untrusted apps sharing the platform with high-criticality applications

  - Issues:
    - Protecting the platform
    - Isolating the applications (from each other and from the Internet)
    - Secure application management
    - Minimize interference among applications (covert channels)

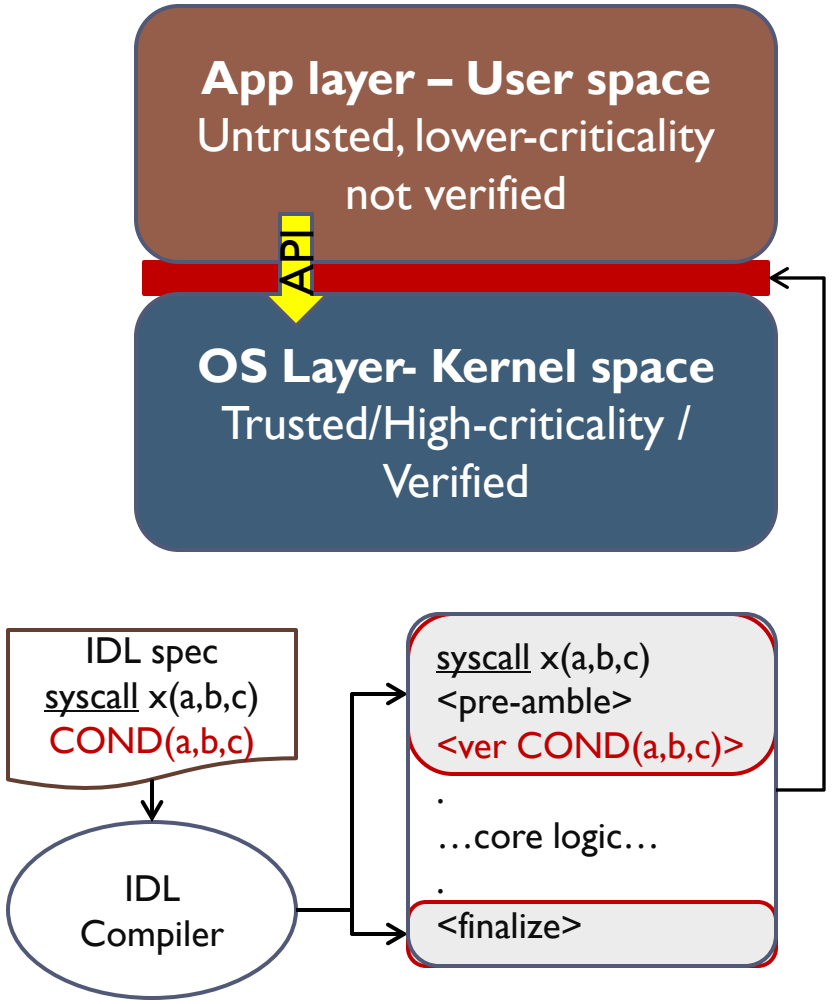
Software platform



1. *Applications* are built from software components that interact via only well-defined interaction patterns using security-labeled messages, and are allowed to use only a restricted set of low-level services provided by the operating system
2. Specialized, strictly verified and trusted *platform actors* provide system-wide high-level services (e.g. application deployment, fault management, certificate management)
3. The *middleware* libraries implement the high-level real-time communication abstractions (synchronous and asynchronous interactions) on the underlying distributed and dynamic platform
4. The *Operating System* implements all the critical low-level services for resource sharing (incl. spatial and temporal partitioning), secure information flows, communication resource management, and fault tolerance

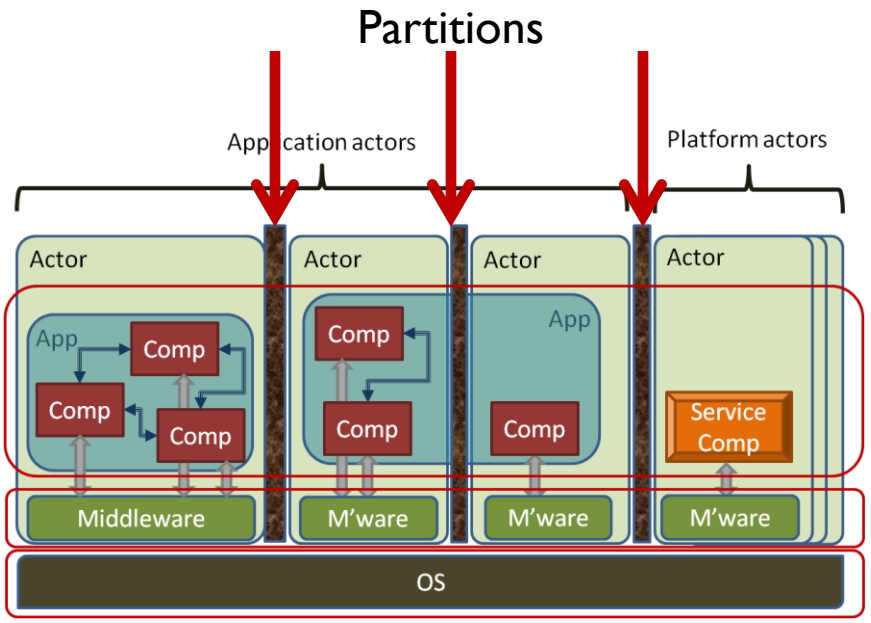
Protecting the platform

- Classic OS approach:
  - Apps make special API calls to access OS services
- Potential security issue:
  - Exploit: Specially crafted data structures passed through the OS API calls can trigger latent security defects in the OS (leading to uncontrolled execution of user code at a privileged level)
- Solution: Protect the platform APIs
  - API Definition: Formal spec in IDL with annotations
  - IDL compiler generates code for API implementation that checks data structure integrity conditions
- Benefits:
  - Simple but strong protection against OS abuse, easy to add to existing OS-s



Isolating applications

- Spatial partitioning (Memory):
  - Isolated address spaces for actors
  - Implemented by (trusted) MMU hardware, managed by (trusted) OS
  - Limited (for resource management)
- Temporal partitioning (CPU):
  - Fixed duration, periodically repeating slices of processor time
  - Implemented by (trusted) OS scheduler
- File system partitioning:
  - Each app actor has its own isolated file system
  - Implemented by (trusted) OS
- Refinement:
  - Platform (privileged and verified) actors are not subject to temporal partitioning. Such actors have protected APIs.



Real-time challenges addressed:

•Actors are guaranteed to get a bounded slice of the CPU resources: memory and time

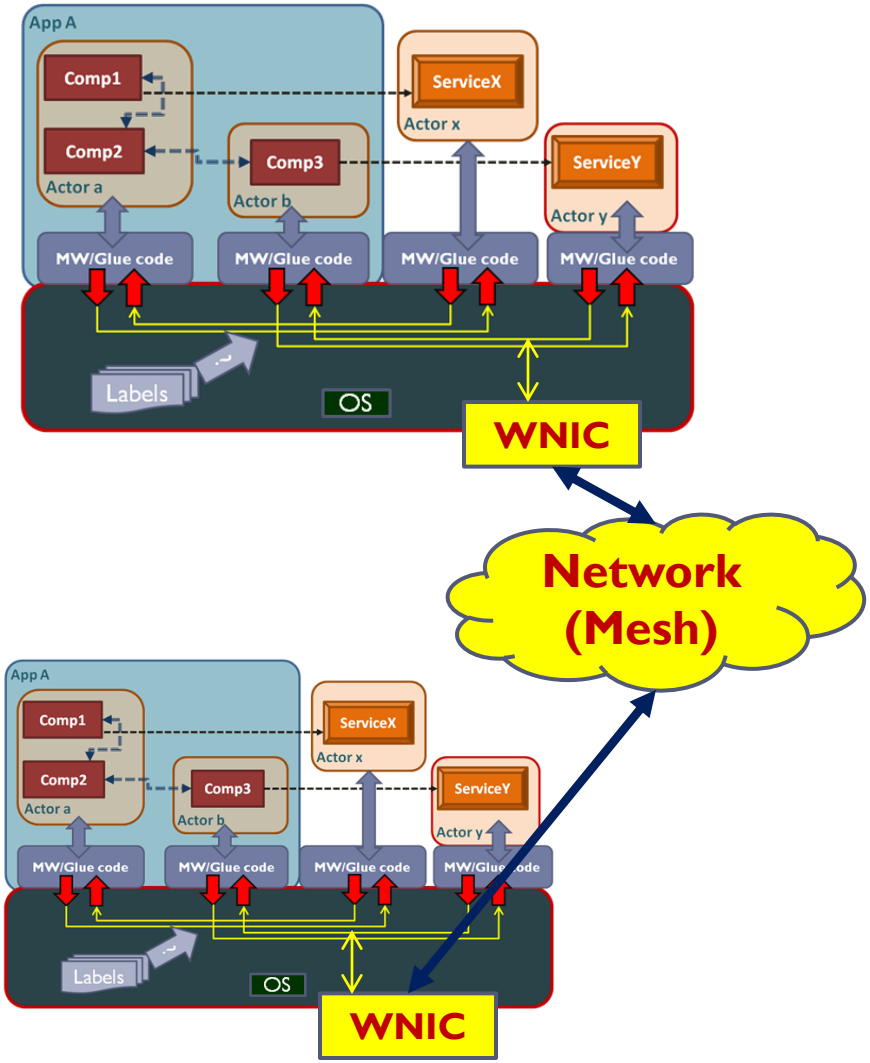
Security challenges addressed:

•Apps cannot interfere with each other through shared memory and the shared processor

•Covert channel bandwidth mitigated

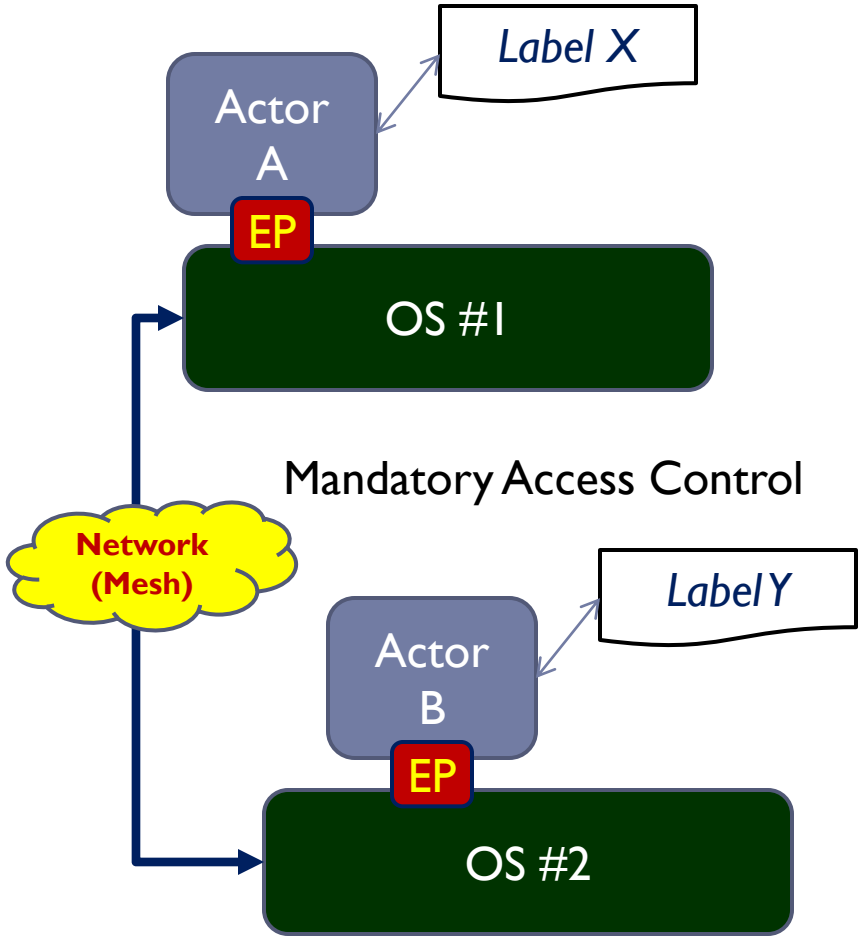
Secure information flows

- **Apps communicate via secure endpoints via flows**
- **Apps cannot create endpoints and flows, only privileged platform actors can**
- Endpoints are similar to sockets but: (1) created by the Deployment Manager (platform actor), (2) require *security labels* on all messages, (3) message transfers are time-stamped
- Flows are the logical (1-1, 1-*) connections between endpoints that represent information flows that are (1) created by the DM, (2) managed by the (trusted) OS, (3) can be mapped to various transport protocols (UDP, SCTP) running on IPSec



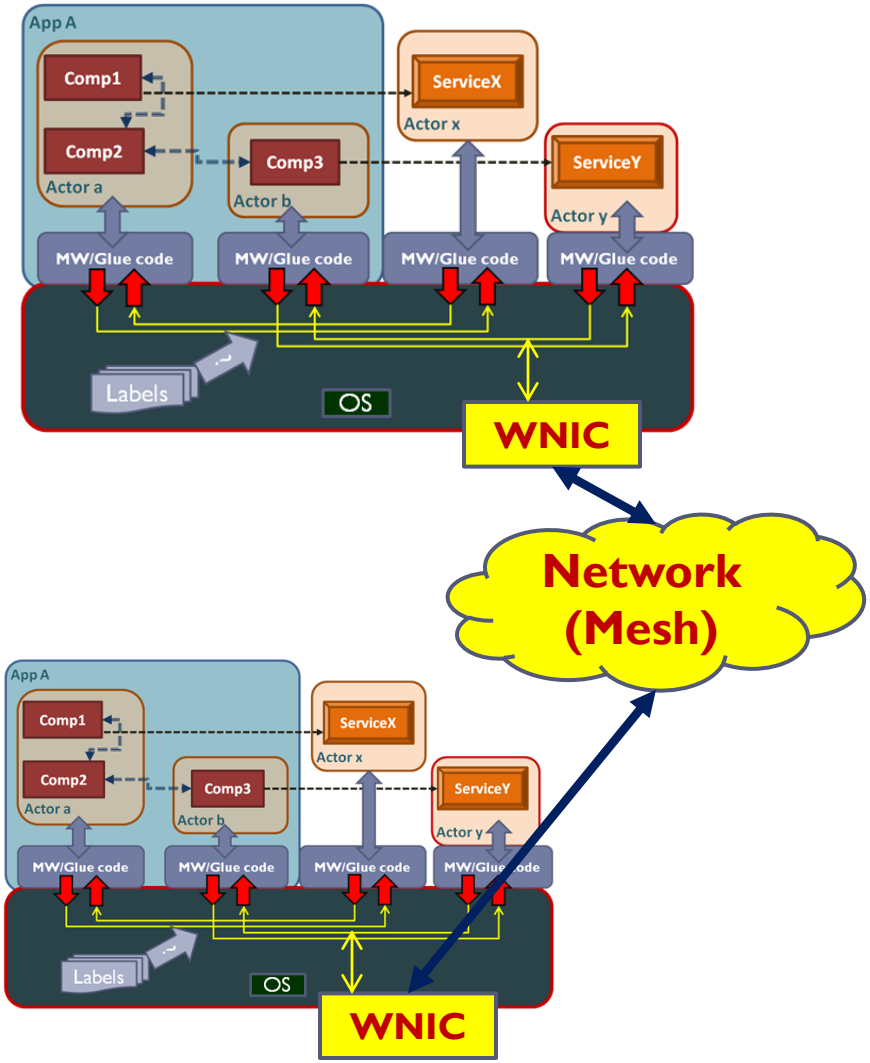
Secure information flows

- Security labels: elements of a lattice structure describing classification.
- Deployment (by DM):
  - Actor A is deployed with label X, its EP is created
  - Actor B is deployed with label Y, its EP is created
  - OS#1/2 knows A/X - B/Y and the flow between EPs
- Labeled communications
  - Actor A sends a message via its EP – must supply a label
  - OS#1 checks whether (1) A can use the label (i.e. X), (2) the labeled message can be sent a recipient with label Y. The message is sent to OS #2 only if the checks pass.
  - OS #2 checks whether (1) the labeled message can be delivered to B (with label Y). The message is delivered only if the checks pass.



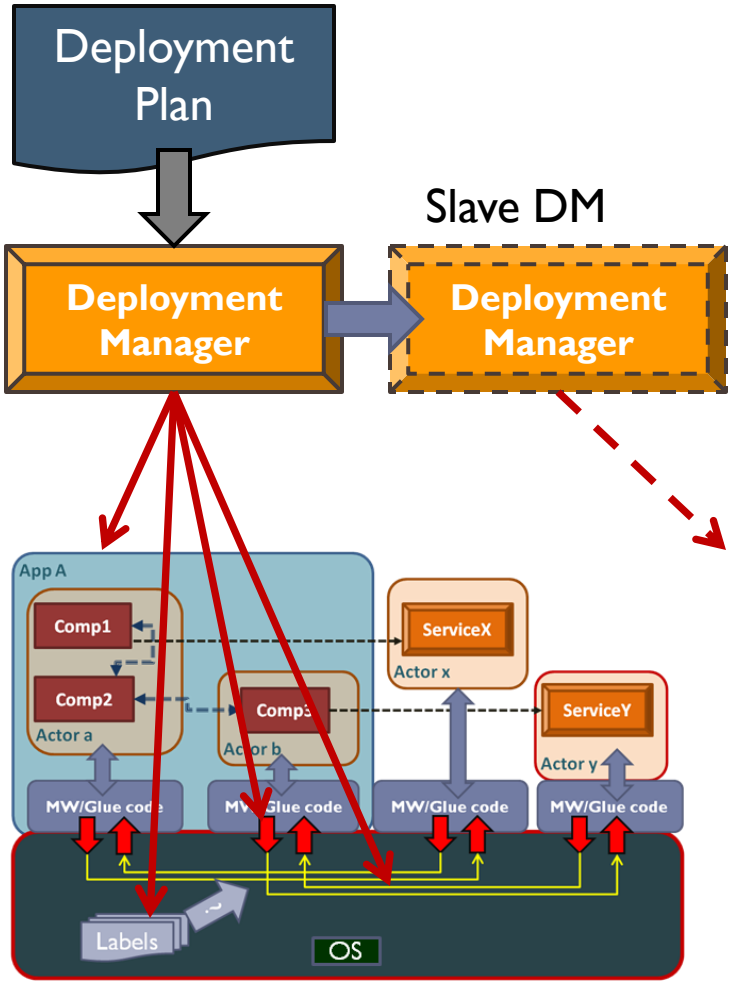
Network communications

- Implementation details
  - Network bandwidth is capped and actors are provided a budget
  - Various transport classes are available: critical, time-triggered, rate-constrained, best effort
- Real-time challenges addressed:
  - Message timestamps allow the recipients know the communication delays
  - Constant network conditions allow guaranteed real-time transfers
- Security challenges addressed:
  - Apps cannot get to the Internet directly
  - Apps cannot abuse the communication resource
  - Apps cannot perform unauthorized (un-configured) communications



Secure application management

- **Deployment Manager**: a (trusted) privileged platform actor, responsible for all app deployment and configuration activities on the platform
- **Deployment Plan**: a data structure (XML) describing the complete configuration of the system, including security labels
- **Process**: the DM parses the plan and (1) creates the (temporal) partition schedule, (2) creates endpoints and flows, (3) creates actors and assigns their labels, (3) configures application internals, (4)
- DM is distributed: every OS has one instance – a lead DM orchestrates a distributed deployment process
- **Security challenges addressed:**
- All apps are activated by a trusted entity
- All secure communication links are set up by a trusted entity



Summary

- Security concerns in distributed CPS necessitate a re-thinking of the design paradigm
  - Platform protection
  - Secure information flows
  - Trusted application management
- Security solutions must be integrated with a modern software platform that includes an OS layer and a real-time component model
- An open source implementation of the platform is being developed under the System F6 program of DARPA.