FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

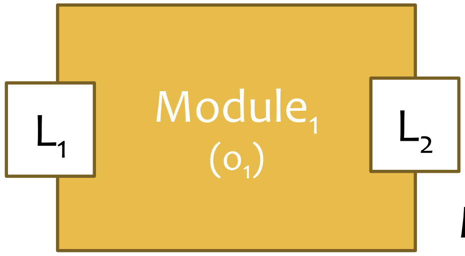
Applying Decentralized Information Flow Labels for System-level Synthesis

Istvan Madari
Janos Sztipanovits
ISIS-Vanderbilt

Berkeley UNIVERSITY OF CALIFORNIA
Massachusetts Institute of Technology
NSF
MICHIGAN
VANDERBILT UNIVERSITY

Background
Apply Decentralized Label Model

* Apply decentralized label model (DLM) on component-based software systems
  * Assign security policies to input/output ports of components
    * DLM elements: Labels, Policies, Principles (readers, owners)



$L_1 = \{o_1: r_1, r_2; o_2: r_2, r_3\}$

$L_2 = \{o_1: r_1, r_2, r_3; \cancel{o_2: r_2, r_3}\}$

  * Detect possible information leaks (policy violations) in development time (constraint checking)
  * Goal: Avoid declassification

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Background
Deployment

* Ensure that when we deploy these logical component interaction models to distributed hardware nodes, the information flow restrictions are not invalidated.
* The deployment model consists of processing nodes, channels and their properties, as well as mappings from the component model's components to nodes and links to channels.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Background
Detect Policy Violations in FORMULA

- FORMULA (Formal Modeling Using Logic Programming and Analysis)
  - Modern formal specification language targeting model-based development (MBD).
  - It is based on algebraic data types (ADTs) and strongly-typed constraint logic programming (CLP), which support concise specifications of abstractions and model transformations
  - Model finding (based on Microsoft Z3 SMT solver)
- Detect policy violations using constraint checking
  - No owners removed and no new readers added
- Check deployment mapping using constraint checking
  - E.g.: Prevent mapping a link between components executing on the same node to a channel

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

SMT Solver-based Propagation of Security Labels

- Context
  - Given a dataflow model with set of principals, modules and their connections through ports
  - Security labels on ports are unknown (or just partially known)
  - Goal: find a valid label set based on the existing principals and module connections.
- The symbolic domain problem is encoded as an SMT problem and then an SMT solver is used to find a solution.
- Finding a satisfiable label set: *propagation*
  - Reuse existing DLM constraints
  - Optimization rules (exclude irrelevant solutions)
  - Type specific rules

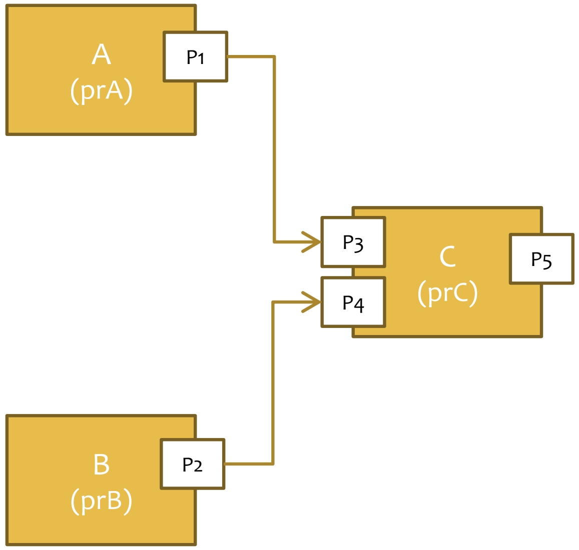
Components Domain with Constraints

```
domain Components extends LabelSecurity
{
  primitive Principal ::= (id:String).
  primitive Policy ::= (lbl:Label, owner:Principal, reader:Principal).
  primitive Label ::= (id:Integer).
  primitive Port ::= (id:String, lbl:Label).
  primitive Component ::= (id:String, owner:Principal).
  primitive ComponentInputPort ::= (comp:Component, port:Port).
  primitive ComponentOutputPort ::= (comp:Component, port:Port).
  primitive Link ::= (src:Port, dst:Port).

  RemovedOwners ::= (port:Port, owner:Principal).
  RemovedOwners(dstPort, owner) :-
     Link(srcPort, dstPort)
   , srcPort is Port(_,srcLbl)
   , dstPort is Port(_,dstLbl)
   , srcPol is Policy(srcLbl, owner, _)
   , no Policy(dstLbl, owner, _)
   .
  OwnerRemoved :=
     RemovedOwners(port,removedOwner)
   , ComponentPort(comp, port)
   , comp is Component(_, compOwner)
   , no ActsForTR(compOwner, removedOwner)
   .

  Propagation := !OwnerRemoved.
}
```

Propagation of Security Labels by FORMULA SMT Solver

```
partial model C of Components
{
  prA is Principal("A")
  prB is Principal("B")
  prC is Principal("C")

  c1 is Component("A", prA)
  c2 is Component("B", prB)
  c3 is Component("C", prC)

  p1 is Port("A out", _)
  p2 is Port("B out", _)
  p3 is Port("C in 1", _)
  p4 is Port("C in 2", _)
  p5 is Port("C out 1", _)

  ComponentInputPort(c3, p3)
  ComponentInputPort(c3, p4)
  ComponentOutputPort(c3, p5)

  ComponentOutputPort(c1, p1)
  ComponentOutputPort(c2, p2)

  Link(p1, p3)
  Link(p2, p4)
  Link(p3, p5)
  Link(p4, p5)
}
```

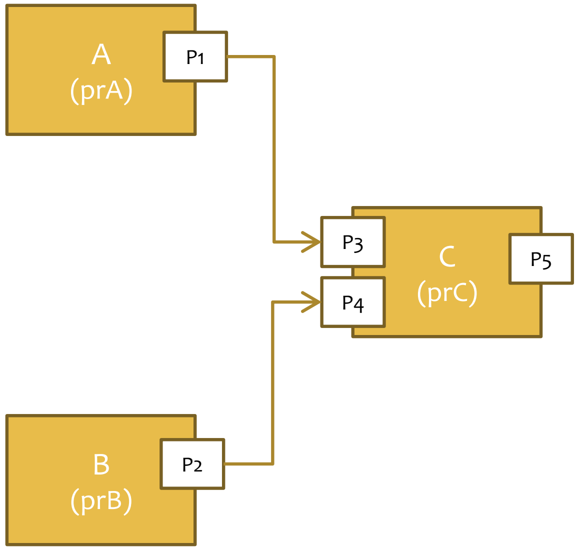
Ports with unknown security labels



FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Solution

```
partial model C of Components
{
  port_0x8 is Port("A out", label_0x5)
  port_0x9 is Port("B out", label_0x6)
  port_0xA is Port("C in 1", label_0x5)
  port_0xB is Port("C in 2", label_0x6)
  port_0xC is Port("C out 1", label_0x7)

  Policy(label_0x5, prA, prC)
  Policy(label_0x6, prB, prC)
  Policy(label_0x7, prA, prC)
  Policy(label_0x7, prB, prC)
  Policy(label_0x7, prC, prC)

  label_0x5 is Label(0)
  label_0x6 is Label(1)
  label_0x7 is Label(2)

  c1 is Component("A", prA)
  c2 is Component("B", prB)
  c3 is Component("C", prC)
  prA is Principal("A")
  prB is Principal("B")
  prC is Principal("C")
  ComponentInputPort(c3, port_0xA)
  ComponentInputPort(c3, port_0xB)
  ComponentOutputPort(c1, port_0x8)
  ComponentOutputPort(c2, port_0x9)
  ComponentOutputPort(c3, port_0xC)
  Link(port_0x8, port_0xA)
  Link(port_0x9, port_0xB)
  Link(port_0xA, port_0xC)
  Link(port_0xB, port_0xC)
}
```



Component Model
Refined Component Types

- New component types added
  - Application
    - CodeType: Managed/Unmanaged
      - Tampering
    - Isolation level: Sandbox, AppContainer
      - Restricted operating system environment
    - RunningAs: Kernel, System, Local Service, Network Service
  - WebApplication
  - Sensor
  - Storage

```
domain Components extends LabelSecurity
{
 [Closed(owner)]
 primitive Component ::= (id:String, owner:Principal).

 [Closed(comp)]
 primitive Application ::= (comp:Component, cpg:ComponentPropertyGroup).

 CodeTypeEnum ::= {MANAGED, UNMANAGED}.
 RunningAsEnum ::= {KERNEL, LOCALSERVICE, NETWORKSERVICE}.

 primitive ComponentPropertyGroup ::= (Integer).
 primitive CodeType ::= (ComponentPropertyGroup, CodeTypeEnum).
 primitive RunningAs ::= (ComponentPropertyGroup, RunningAsEnum).
 …
}

transform Analyze
 from in1::Deployment
 to out1::Threats
{
 out1.ThreatInfo_P2("Process memory tampered", comp) :-
    node is in1.Node(_, COMPUTATION_NODE, npg)
   , no in1.ComputationNodeSeparationKernel(npg, true)
   , in1.NodeReaderConflict(node)
   , in1.CompMap(comp, node)
   , in1.Application(comp, cpg)
   , in1.CodeType(cpg, UNMANAGED)
  .
}
```

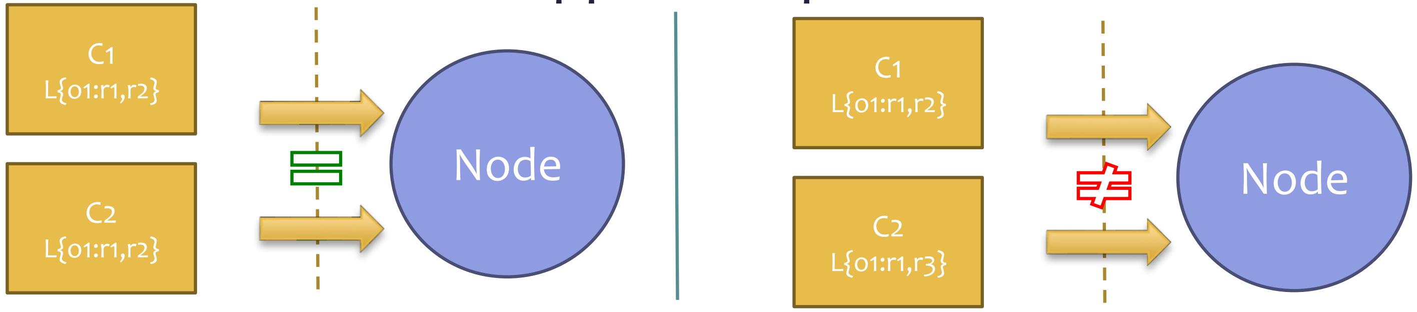
FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Deployment
New Type Specific Nodes

- Platform model extended
  - New node types and properties
    - Computation node - Separation Kernel
      - Type of security kernel used to simulate a distributed environment
      - Protection of all resources (including CPU, memory and devices) from unauthorized access
    - Sensor
    - Storage (File system, Database, Device)
      - Encryption
  - New channel types and properties
    - Protection (Protected/Unprotected)
    - Physical Network (Wired, Wireless)
    - Protocols
      - Context dependent

Mapping Different Components to the Same Computation Node

- Ensure that the readers of the mapped components are not conflicting
  - The reader lists of the mapped components are the same



- Resolve conflicts by node properties if possible
  - If the readers of the mapped components are in conflict on the given platform, then set the Separation Kernel property to true.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Deployment Configuration by FORMULA SMT Solver

- Context
  - Given a dataflow model with set of principals, modules and their connections through ports
  - Given component mapping
  - Security labels on ports and node properties are unknown
  - Goal: find a valid label set AND node property settings based on the existing principals, module connections and component mapping.
- Goals
  - Finding a satisfiable label set: *propagation* (with FORMULA SMT solver)
  - Set the node properties in order to avoid reader conflicts

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Deployment Domain with Constraints

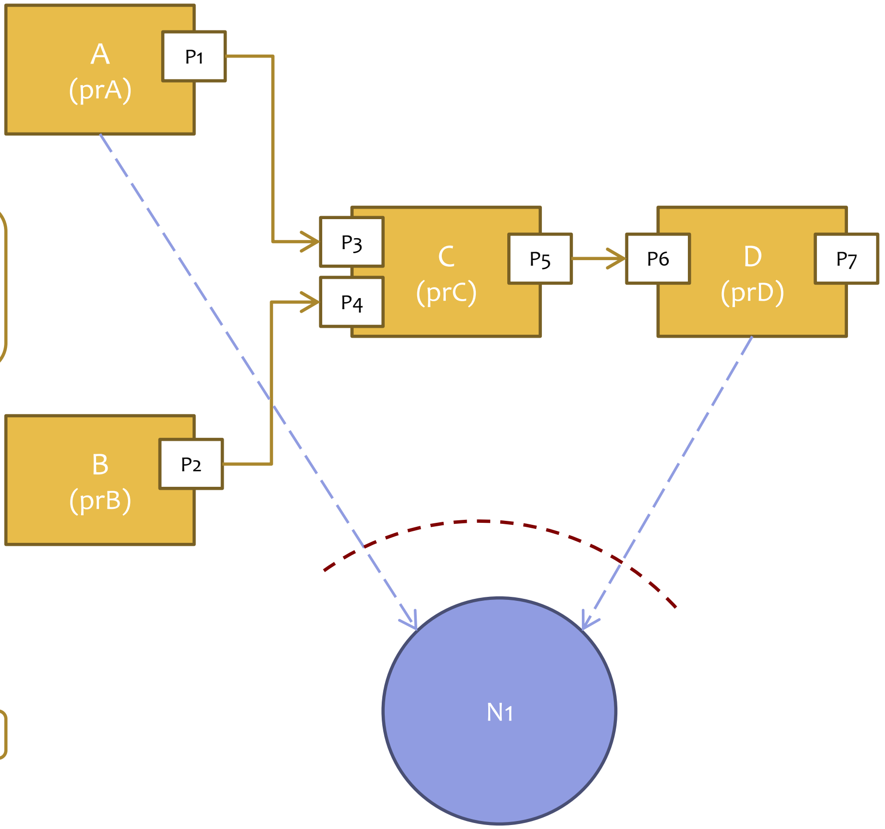
```
domain Deployment extends Components, NodePropertySpec
{

  NodeTypes ::= {COMPUTATION_NODE, SENSOR, STORAGE, NULL}.

  primitive Node ::= (id:String, nt:NodeTypes, npgp:NodePropertyGroup).
  primitive CompMap ::= (comp:Component, n:Node).
  primitive ComputationNodeSeparationKernel ::= (npg:NodePropertyGroup, val:Boolean).
  primitive NodePropertyGroup ::= (Integer).

  MergedComponentReaders ::= (Node, Principal).
  MergedComponentReaders(n, reader) :-
    n is Node(_, COMPUTATION_NODE, _),
    CompMap(c,n),
    ComponentReaders(c, reader).

  NodeReaderConflict ::= (Node).
  NodeReaderConflict(n) :-
    n is Node(_, COMPUTATION_NODE, _),
    CompMap(c, n),
    MergedComponentReaders(n, r),
    no ComponentReaders(c, r).

  GoodSepKernalMapping :=
    n is Node(_, COMPUTATION_NODE, npg),
    NodeReaderConflict(n),
    ComputationNodeSeparationKernel(npg, true)
    ;
    n is Node(_, COMPUTATION_NODE, npg),
    no NodeReaderConflict(n),
    ComputationNodeSeparationKernel(npg, false). }
```

Deployment - Find Configuration

```
[Search(Label, Policy, NodePropertyGroup, ComputationNodeSeparationKernel)]
partial model C of Deployment
{
 ActsFor(prC, prA)
 ActsFor(prC, prB)

 prA is Principal("A")
 prB is Principal("B")
 prC is Principal("C")
 prD is Principal("D")

 c1 is Component("A", prA)
 c2 is Component("B", prB)
 c3 is Component("C", prC)
 c4 is Component("D", prD)

 p1 is Port("A out", _)
 p2 is Port("B out", _)
 p3 is Port("C in 1", _)
 p4 is Port("C in 2", _)
 p5 is Port("C out 1", _)
 p6 is Port("D in 1",_)
 p7 is Port("D out 1",_)

 ComponentInputPort(c3, p3)
 ComponentInputPort(c3, p4)
 ComponentOutputPort(c3, p5)

 ComponentOutputPort(c1, p1)
 ComponentOutputPort(c2, p2)

 ComponentInputPort(c4, p6)
 ComponentOutputPort(c4, p7)

 Link(p1, p3)
 Link(p2, p4)
 Link(p3, p5)
 Link(p4, p5)
 Link(p6, p7)
 Link(p5, p6)

 n1 is Node("n1", COMPUTATION_NODE, _)

 CompMap(c1, n1)
 CompMap(c4, n1)
}
```

Ports with unknown security labels

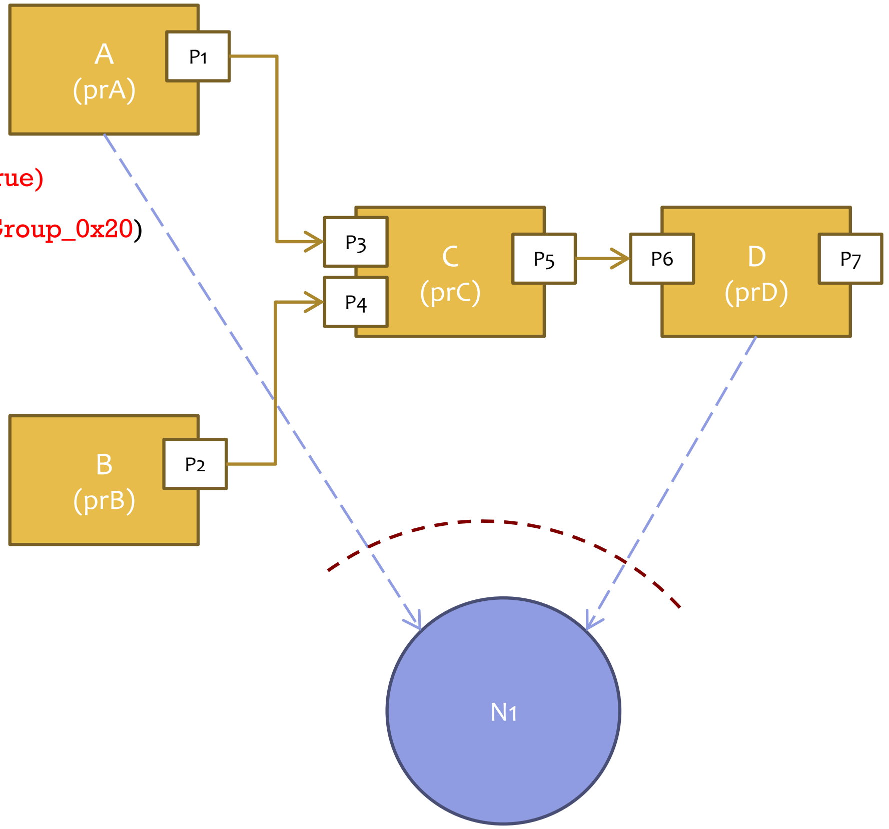
Unknown properties



FORCES FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

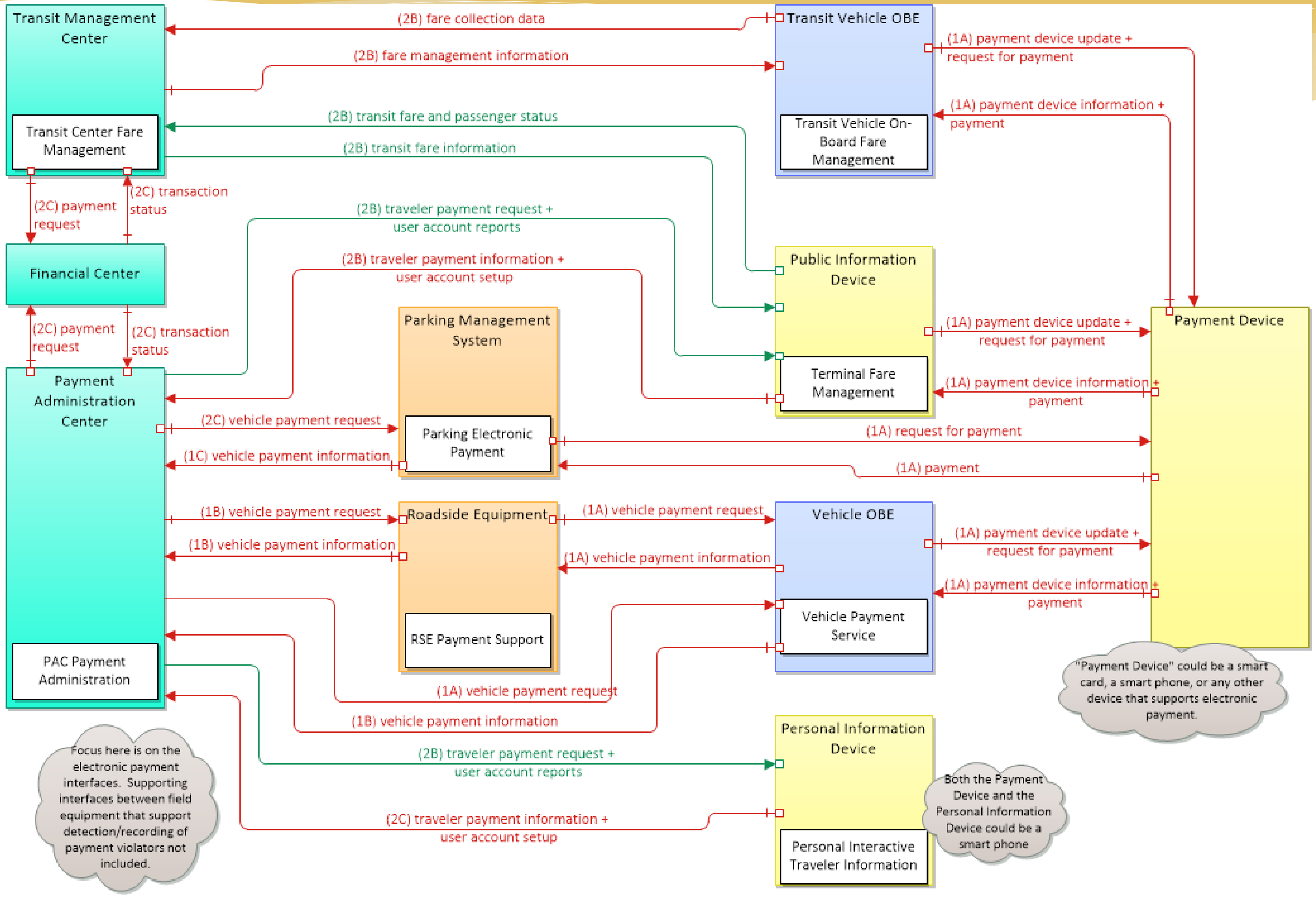
Solution

```
[Search(Label, Policy, NodePropertyGroup,
ComputationNodeSeparationKernel)]
partial model C of Deployment
{
 Policy(label_0x5, prA, prC)
 Policy(label_0x6, prB, prC)
 Policy(label_0x7, prC, prD)
 Policy(label_0x8, prC, prD)
 Policy(label_0x8, prD, prD)


 ComputationNodeSeparationKernel(nodePropertyGroup_0x20, true)
 nodePropertyGroup_0x20 is NodePropertyGroup(3)
 node_0x21 is Node("n1", COMPUTATION_NODE, nodePropertyGroup_0x20)

 prA is Principal("A")
 prB is Principal("B")
 prC is Principal("C")
 prD is Principal("D")


 ActsFor(prC, prA)
 ActsFor(prC, prB)

 c1 is Component("A", prA)
 c4 is Component("D", prD)

 port_0xB is Port("A out", label_0x5)
 port_0xC is Port("B out", label_0x6)
 port_0xD is Port("C in 1", label_0x5)
 port_0xE is Port("C in 2", label_0x6)
 port_0xF is Port("C out 1", label_0x7)
 port_0x10 is Port("D in 1", label_0x7)
 port_0x11 is Port("D out 1", label_0x8)
 CompMap(c1, node_0x21)
 CompMap(c4, node_0x21)
}
```



FORCES
FOUNDATIONS OF RESILIENT
CYBER-PHYSICAL SYSTEMS

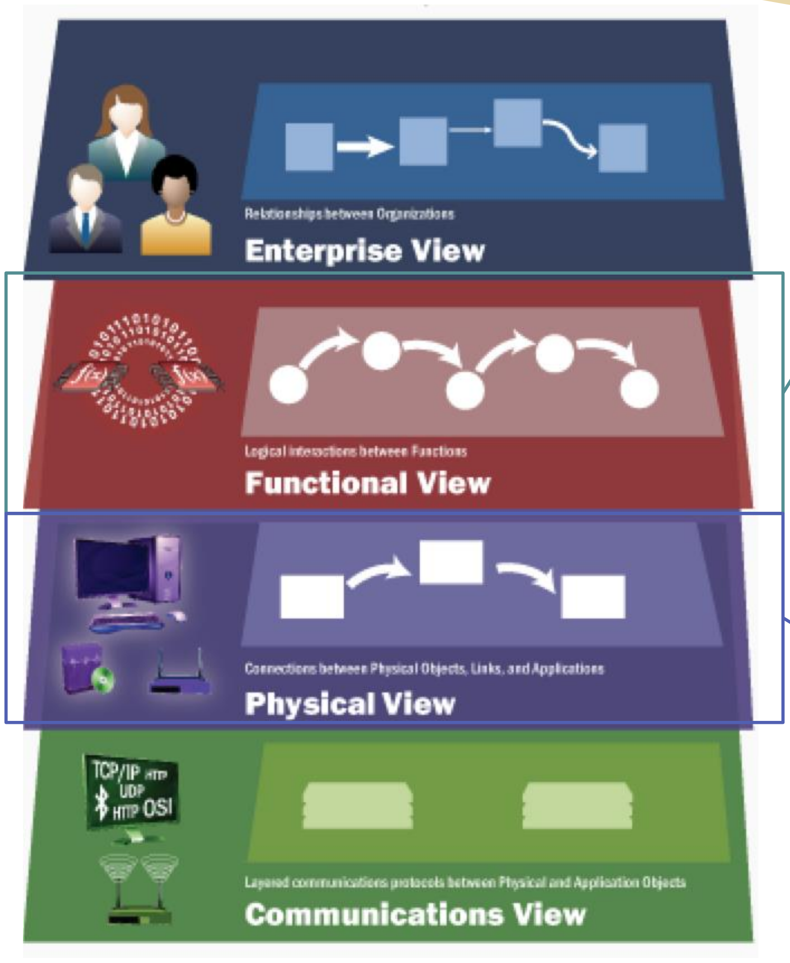
Current and Future Work (1)



Transit Management Center
Transit Center Fare Management
Transit Vehicle OBE
Transit Vehicle On-Board Fare Management
(2B) fare collection data
(2B) fare management information
(2B) transit fare and passenger status
(2B) transit fare information
(1A) payment device update + request for payment
(1A) payment device information + payment
(2C) payment request
(2C) transaction status
Financial Center
(2B) traveler payment request + user account reports
(2B) traveler payment information + user account setup
Public Information Device
Terminal Fare Management
Payment Device
Parking Management System
Parking Electronic Payment
Payment Administration Center
PAC Payment Administration
(2C) vehicle payment request
(1C) vehicle payment information
(1A) request for payment
(1A) payment
Roadside Equipment
RSE Payment Support
(1B) vehicle payment request
(1B) vehicle payment information
(1A) vehicle payment request
(1A) vehicle payment information
Vehicle OBE
Vehicle Payment Service
Personal Information Device
Personal Interactive Traveler Information
(2C) traveler payment information + user account setup
"Payment Device" could be a smart card, a smart phone, or any other device that supports electronic payment.
Both the Payment Device and the Personal Information Device could be a smart phone
Focus here is on the electronic payment interfaces. Supporting interfaces between field equipment that support detection/recording of payment violators not included.

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS

Current and Future Work (2)


Relationships between Organizations
Enterprise View
Logical Interactions between Functions
Functional View
Connections between Physical Objects, Links, and Applications
Physical View
TCP/IP HTTP UDP HTTP OSI
Layered communications protocols between Physical and Application Objects
Communications View

- **Functional View**
  - Set of processes
  - Data flows that move between processes
  - **FORMULA extends this data flow with security labels**

Processes are associated to physical objects (nodes)

- **Physical View**
  - Physical objects
  - Information flows
  - **This is the deployment model, FORMULA extends it with properties (e.g.: separation kernel or security protocol on channel)**

FORCES
FOUNDATIONS OF RESILIENT CYBER-PHYSICAL SYSTEMS