



Building Dependable Cyber-Physical Systems using Timed Actors

Marjan Sirjani

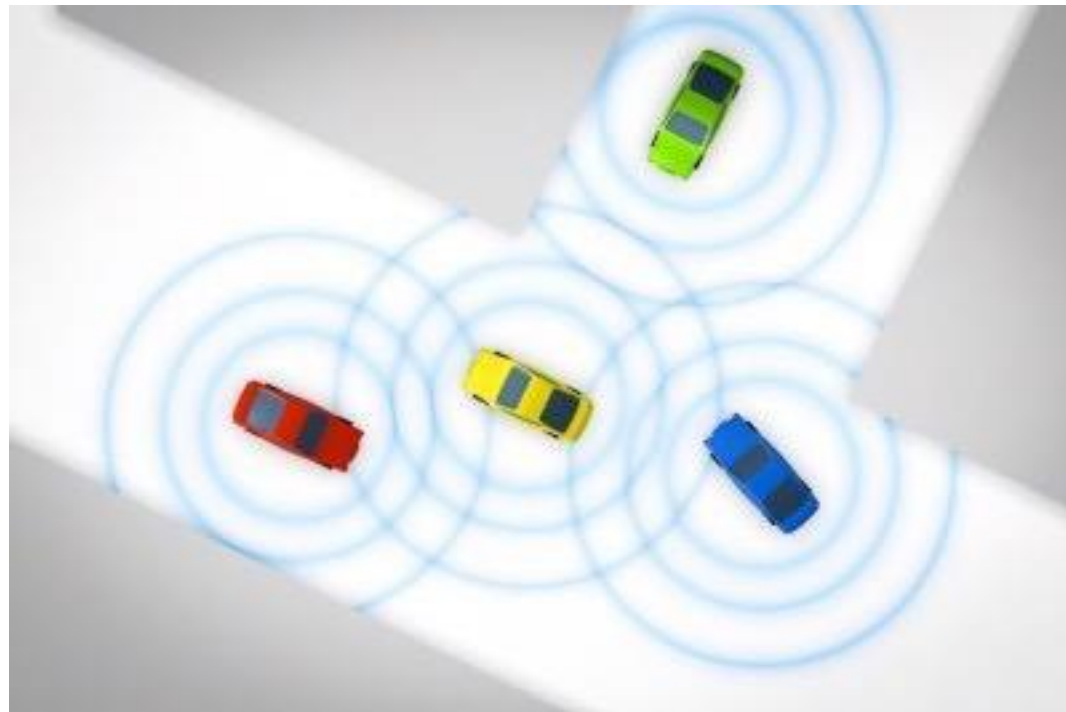
Mälardalen University, Sweden

Reykjavik University, Iceland

Networks and Systems Summer School

August 15, 2018

Google Wants Driverless Cars, but Do We? (NYT Dec. 2016)



**Will vehicle-to-vehicle communication
in new cars save lives?** (readwrite.com
2016/12/15)

SIGNS OF THE TIMES

Traffic signs could soon be beamed directly on to your car's dashboard



- **TRADITIONAL** road signs and overhead gantries could be ditched in favour of new technology that is able beam traffic information directly into cars.



The Sun, a news UK Company
By Dan Elsom
30th July 2018

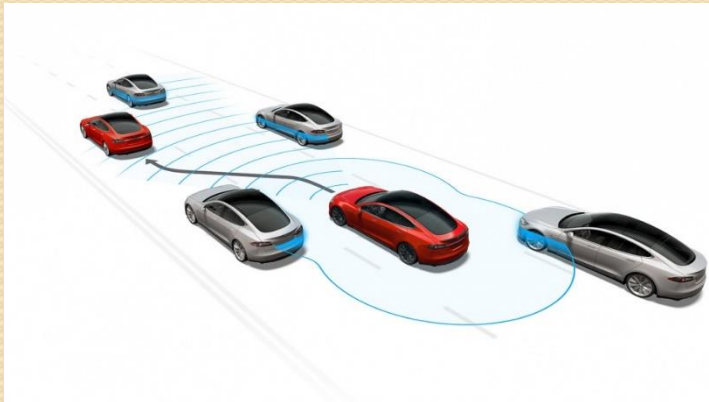
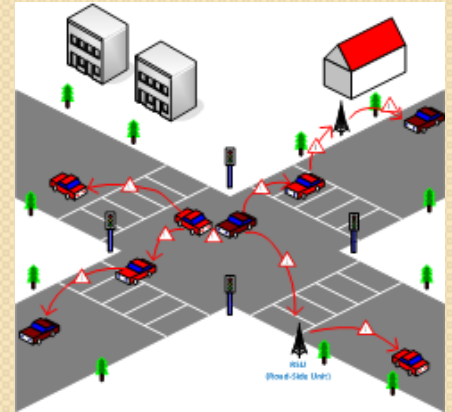
<https://www.thesun.co.uk/motors/6894994/speed-limits-and-traffic-news-could-soon-be-beamed-straight-onto-your-cars-dashboard-to-replace-motorway-signs/>

VANETs and V2V Communication



Platooning operations
such as platoon
formation, breakup
and string stability in
the platoon

Intersection coordination

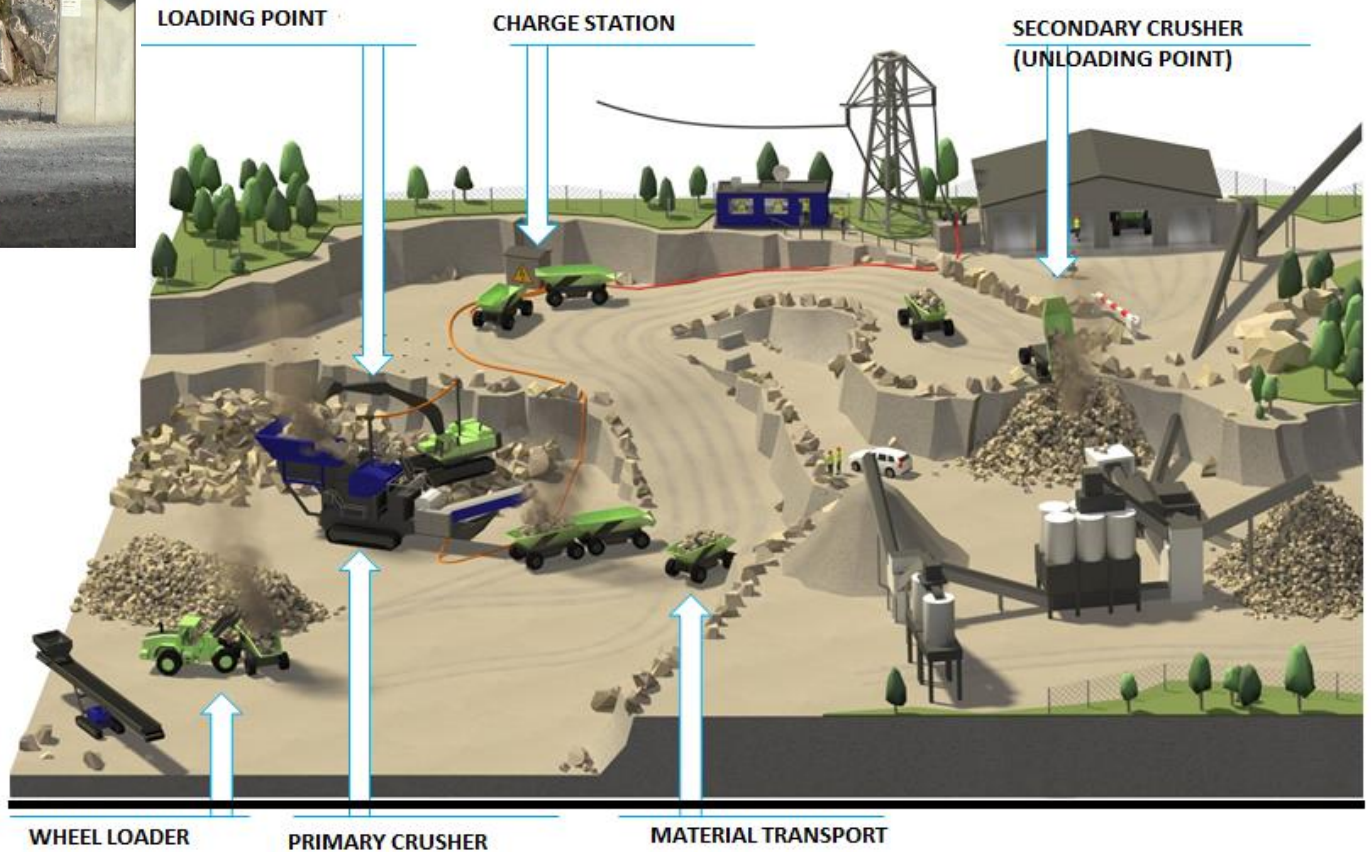


Lane change assistance

Cooperative collision
avoidance by using
warning message
dissemination



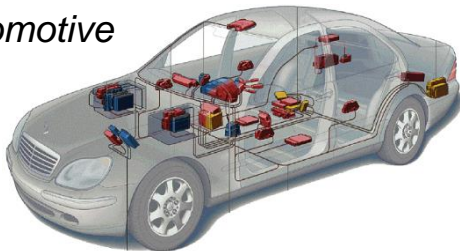
Volvo CE Electric site



Cyber-Physical Systems

- Integrations of computation, networking, and physical processes.
- Embedded computers and networks monitor and control the physical processes

Automotive



Cyber-Physical Systems

- Different Dimensions ... Different Features ...
- Sensors
- Network
- Clouds
- Connecting the Cyber and the Physical worlds
- ...

- Security
- Privacy
- Safety
- Reliability
- ...



Learning

Clouds

Security

Control Software

Sensors

Real-Time

Embedded
Systems

Models &
Methodologies

Safety

Wireless
Networks

Dependable Systems

- **Dependability**
 - **Availability, reliability, safety ...**
 - **Correct behavior**
- **Thorough Analysis**
 - **Testing, simulation, correct by construction, formal methods, ...**
- **We need Analyzability**

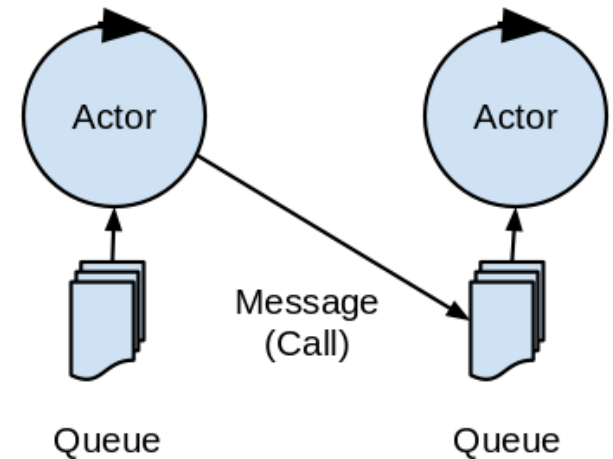
Models, Methodologies, and Tools

- Our Focus in modeling:
 - Network: **asynchronous and event-based**
 - Dealing with **time and uncertainty**: timed probabilistic model
 - Adaptive: handling **change**
 - Cyber and Physical: mapping and the interface between **discrete and continuous**
- Our Focus in Analysis:
 - Safety and Performance
(not on security or privacy, yet)

Will Show you

- The actor-based language, **Rebeca**, provides a **usable** and **analyzable** model for distributed, concurrent, event-based asynchronous systems.
- To build dependable Cyber-Physical Systems

Actors (Hewitt, Agha)

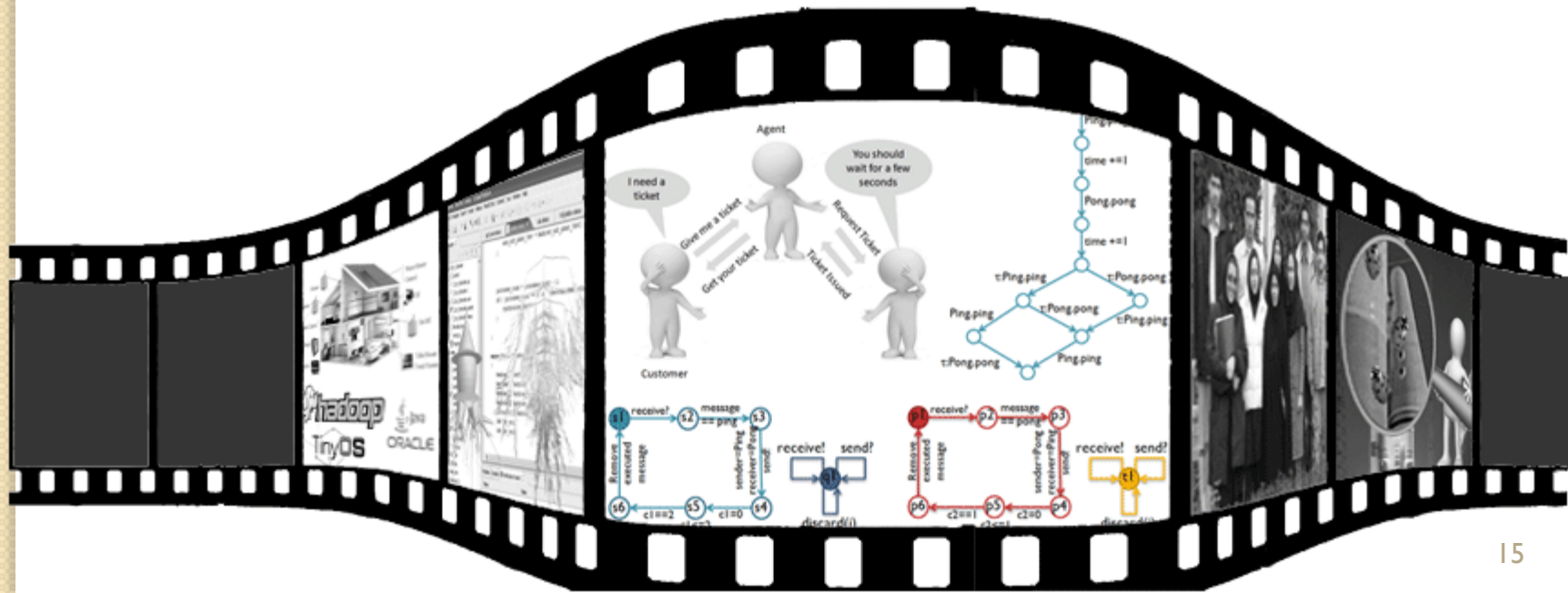


- A reference model for concurrent computation
- Consisting of concurrent, distributed active objects
- Proposed by [Hewitt](#) as an agent-based language (MIT, 1971)
- Developed by [Agha](#) as a concurrent object-based language (UIUC, since 1984)
- Formalized by [Talcott](#) (with Agha, Mason and Smith): Towards a Theory of Actor Computation (SRI, 1992)

Why actors?

- Friendly:
 - Faithful: smaller semantic gap, constructs for asynchronous modeling
 - Usable: OO is familiar for practitioners
- Analyzable: formal basis
 - Loosely coupled actors help in developing more efficient analysis techniques
 - Decomposition, Compositional verification, Abstraction
- Proved to be a Good Choice!
 - LinkedIn, Twitter, Facebook Chat, Halo Game Engine, etc., all are written in Actor languages
 - Microsoft making actor languages like P and Orleans

Probabilistic Timed Rebeca



Rebeca: The Modeling Language

- **Rebeca: Reactive object language** (Sirjani, Movaghar, 2001)

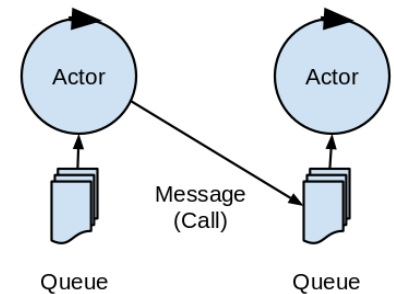
- Based on actor model
- Concurrent reactive objects (OO)
- Java like syntax

- **Communication:**

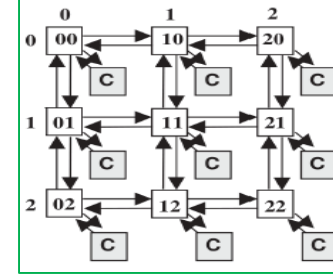
- Asynchronous message passing: non-blocking send
- Unbounded message queue for each rebec
- No explicit receive

- **Computation:**

- Take a message from top of the queue and execute it
- Event-driven



ASPIN: Rebeca abstract model



reactiveclass Router{

knownrebcs

{Router[4] neighbor, Core myCore}

statevars{int[4] buffer;}

Router (myId-row, myId-col) {

}

msgsrv reqSend() {

neighbor[x].giveAck() after(3); ...

}

msgsrv getAck() {

// receive ack from the receiver

// get ready for receiving the next packet

...

}

msgsrv giveAck (...) {

//if the message is for my core use it

myCore.forMyCore()

//send ack to the sender

sender.getAck() after(3);

// if not route it to the receiver ...

} }

reactiveclass Core{

knownrebcs {Router myRouter}

statevars{ ...}

Core (...) {

....

msgsrv forMyCore() {

// get the Packet and use it

...

}

main(){

Router r00(r02,r10,r01,r20)(0,0);

Router r01(r00,r11,r02,r21)(0,1);

...

Core c00(r00)

Core c01(r01)

...

}

Actor
type and
its
message
servers

Constructor

A
message
server

Asynchronous
message sending

Instances of
different actors

Parameters

Known rebecs



Rebeca is used for Model-Driven Development

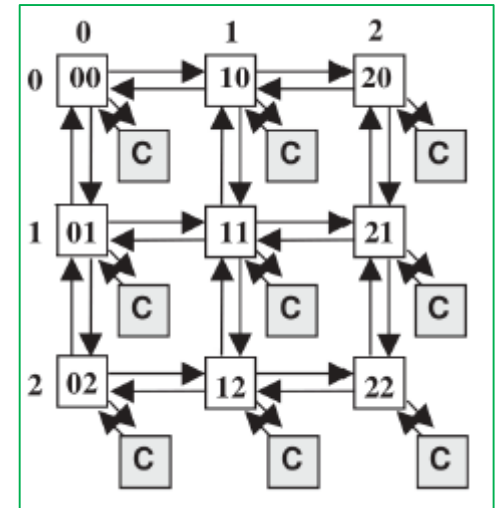
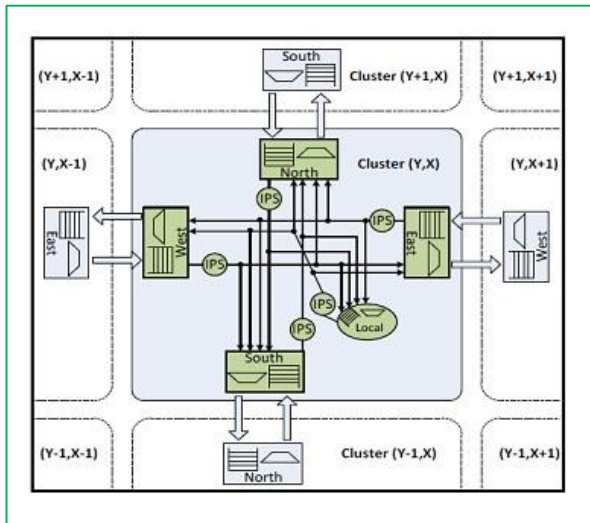
- In the early stages of the design flow and throughout the design process
 - Explore the design space
 - Analyze safety and performance
- Help designers make better architectural decisions

Timed Rebeca

- An extension of Rebeca for real time systems modeling
 - Computation time (**delay**)
 - Message delivery time (**after**)
 - Periods of occurrence of events (**after**)
- Message expiration (**deadline**)

Network on Chip: an example

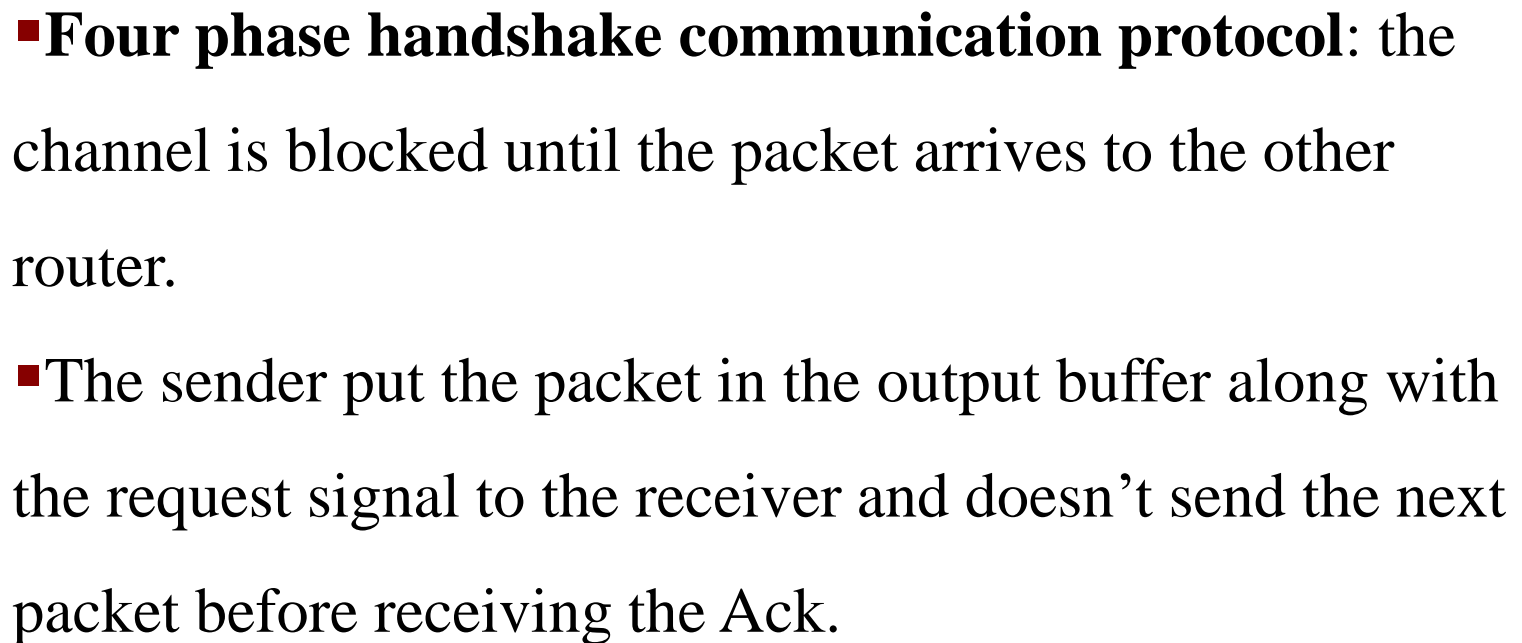
NoC is a communication paradigm on a chip, typically between cores in a system on a chip (SoC).

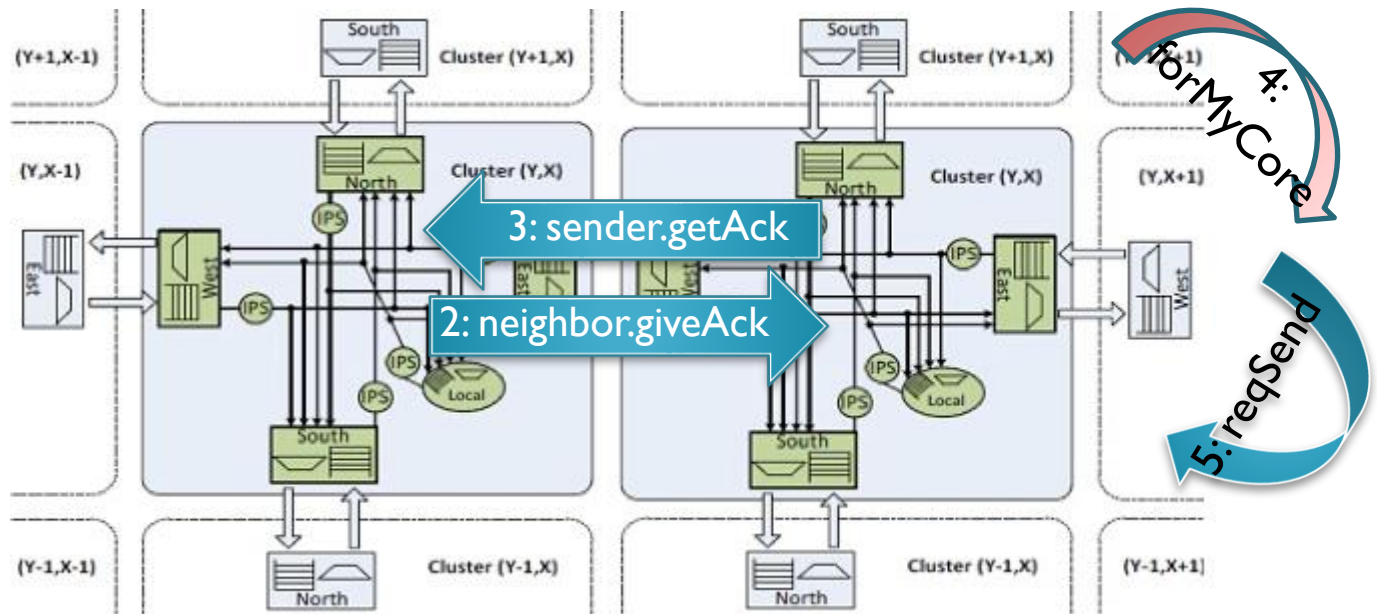


- Explore the design space
 - Evaluate routing algorithms
 - Select best place for memory
 - Choose buffer sizes
 - ...

ASPIN: Two-dimensional mesh GALS NoC







reqSend:
 //Route the Packet
 neighbor.giveAck;

getAck:
 //send the Packet
 //set the flag of your port to free

giveAck:
 //if I am the final Receiver
 //then Consume the Packet
 sender.getAck;
 myCore.forMyCore;

//else if my buffer is not full
 //get the Packet
 sender.getAck
 //and route it ahead
 self.reqSend;

else (my buffer is full) wait

ASPIN: Rebeca abstract model

```

reactiveclass Router{
  knownrebecs {Router[4] neighbor}
  statevars{int[4] buffer;}
  Router ( ... ) {
    ....
  }
  msgsrv reqSend() {
    delay(2);
    neighbor[x].giveAck() after(3) deadline(6);
    ...
  }

```

Time progress
because of
computation delay

Deadline for the
receiver

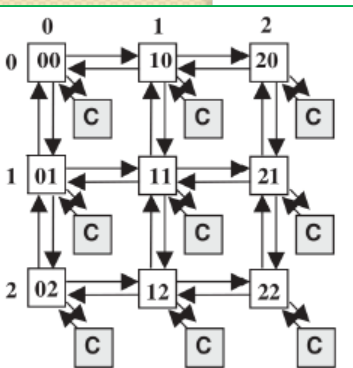
Communication
delay

```

msgsrv giveAck (...) {
  //if the message is for my core use it
  myCore.forMyCore()
  //send ack to the sender
  sender.getAck() after(3);
  // if not and buffer not full then route it to the receiver ...
  // if buffer full then busy-wait until buffer empty
  else self.giveAck() after(10),
}
...

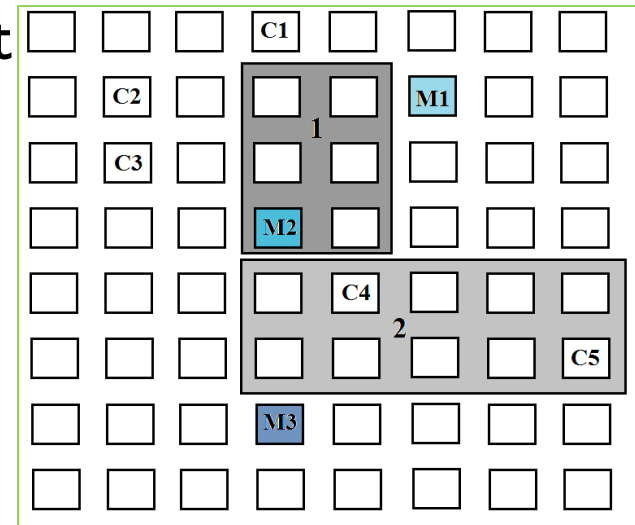
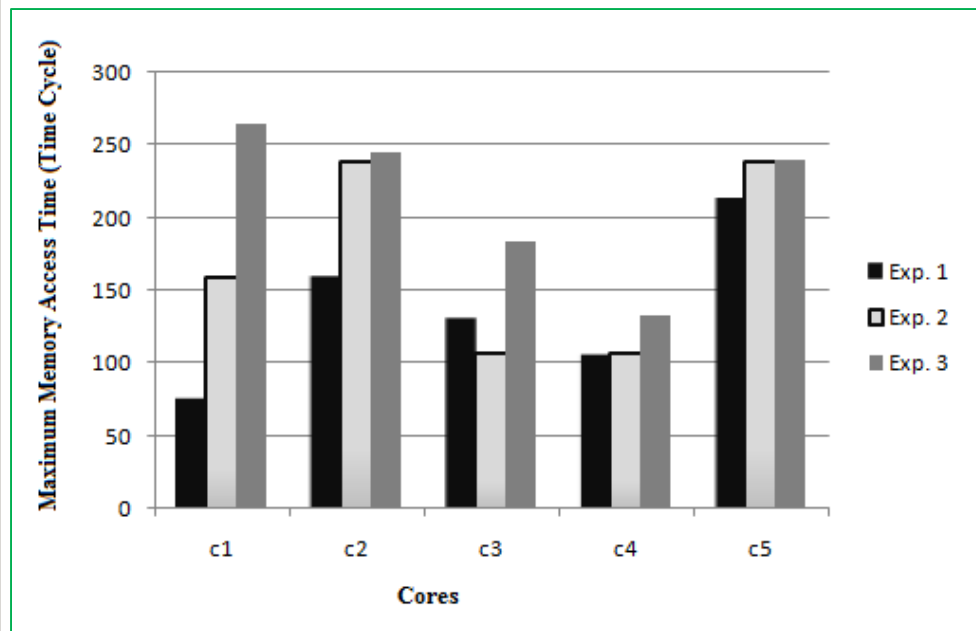
```

periodic tasks



Evaluation of different memory locations for ASPIN 8×8

- Consider 5 cores and their access time to the memory
- 3 choices for memory placement
- 40 packets are injected
- High congestion in area 1 and 2



- Unlike our expectation M1 is a better choice than M2
- The packet injection is based on an application (note that cores have different roles)

Modeling NoC in TRebeca

ASPIN Component
Router + Core

Model in Rebeca
Rebec

Keep the constructs and features that affect the properties of interest and check the following:

1. Possible Deadlock
2. Successful sending and receiving of packets
3. Estimating the maximum end-to-end packet latency

Channel
Communication protocol

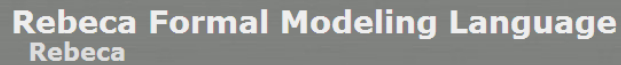
Destination address & ID
Model checking: 3 seconds HSPICE: 24 hours

Much less details.

Showed the same trend.

Tool Support

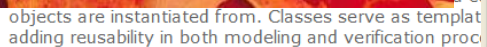
- Model checking toolset
 - Automatic deadlock and queue size analysis
 - Assertion Check
 - Schedulability Analysis
 - CTL and LTL model checking
- Reduction techniques
 - Atomic execution of a method
 - Partial order and symmetry reductions



Rebeca (Reactive Objects Language) is an actor-based language with a formal foundation, designed in an effort to bridge the gap between formal verification approaches and real applications. It can be considered as a reference model for concurrent computation, based on an operational interpretation of the actor model. It is also a platform for developing object-based concurrent systems in practice.

The screenshot displays the Eclipse IDE interface with three callouts identifying key components:

- Project editor:** Points to the left-hand Project Explorer, which shows a hierarchical tree of project files and folders, including 'arb', 'arb_prom', and 'arb_property'.
- Model and Property editor:** Points to the central editor window, which displays the source code of a Verilog-like model. The code includes declarations for signals like 'karbiter_clk', 'karbiter_clk_Post', 'karbiter_rst', and 'karbiter_rst_Post', and a process block 'msgsrv Proc'.
- Model checking result view:** Points to the bottom-right pane, which is currently empty, showing the 'Problems', 'Console', and 'Verification Result' tabs.



<http://www.rebeca-lang.org/>

- **Ten years of Analyzing Actors: Rebeca Experience (Sirjani, Jaghour)**
Invited paper at Carolyn Talcott Festschrift, 70th birthday, LNCS 7000, **2011**
- **On Time Actors (Sirjani, Khamespanah)**, Invited paper at Frank de Boer Festschrift, 60th birthday, LNCS 9660, **2016**


Sample Projects

- NoC Design
 - Routing and Dispatching Analysis
- Network Protocols
 - Deadlock and loop-freedom
- Wireless Sensors Application
 - Task Scheduling
- ROS Programs
 - Safety and Timing analysis
- Track-based Traffic Systems
 - Adaptive (re)scheduling and (re)routing

Mobile Adhoc Networks: MANETs

(UT, Fatemeh Ghassemi)

- Checking the routing protocol:
 - Ad-hoc On-demand Distance Vector Version 2 (AODVv2)
 - Used by mobile routers in wireless, multi-hop networks
 - Arbitrary **changes** of the underlying **topology**
 - Provide a way of communication between two indirectly-connected nodes
 - An algorithm for each node to continuously maintain the information required to properly route traffic.

- 
- **Key Safety Property:**
 - There are no routing loops in any reachable global state.
 - A routing loop is formed when the next-hop entries in routing tables are connected in a cyclic manner, e.g., node A has next-hop B; node B has next-hop C; and node C has next-hop A.

- The AODVv2-version I I
 - Goal → improving the performance
 - Consequence → jeopardizing the loop freedom property
- The AODVv2-version I 3 and version I 6
 - Bugs were found
 - Loop is formed

Schedulability Analysis of Distributed Real-Time Sensor Network Applications (collaboration with OSL, UIUC, Gul Agha)

Smart Structures

"... one highly **intelligent bridge** knows what to do when trouble arises: send [the engineers] an e-mail."

The New York Times



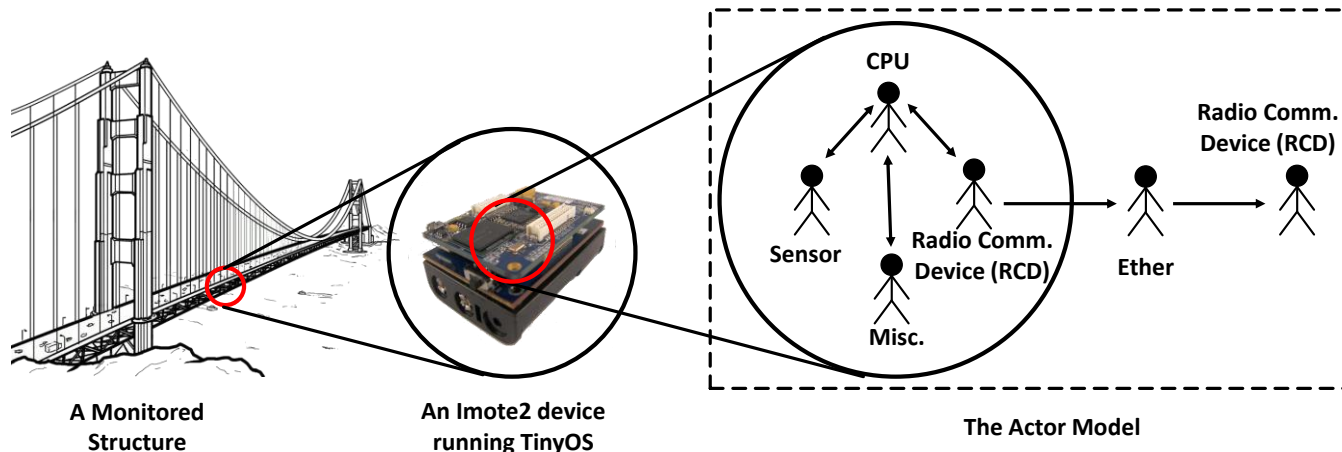
- **Structural Health Monitoring (SHM)**
- Make infrastructure safer while reducing maintenance costs
- Using Networks of Sensors



The Problem:

Finding the best configuration

- Modeling the interactions between
 - the CPU, sensor and radio within each node,
 - as well as interactions among the nodes.
 - alongside tasks belonging to other applications, middleware services, and operating system components.



Schedulability Analysis

- All the periodic tasks (sample acquisition, data processing, and radio packet transmission) are completed before the next iteration starts.
- This defines the deadline for each task.
- Configuration: possible sampling rate in case of different communication protocols, number of nodes, sensor internal task delays, and radio packet size



On Analysis:

Alternative Analysis Techniques

- For schedulability analysis:
 - Simulation
 - Analytical Approach
 - Model Checking
- Smaller state space than Timed Automata when it comes to asynchrony
- Less abstract than analytical approaches

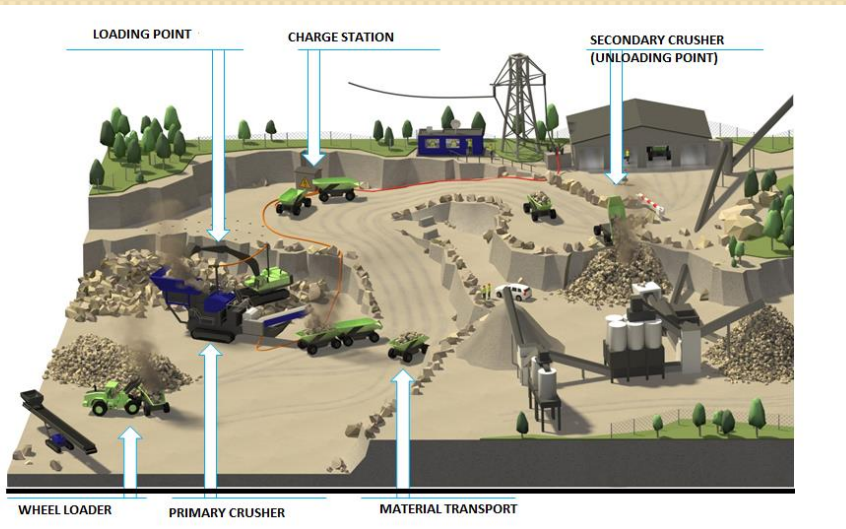
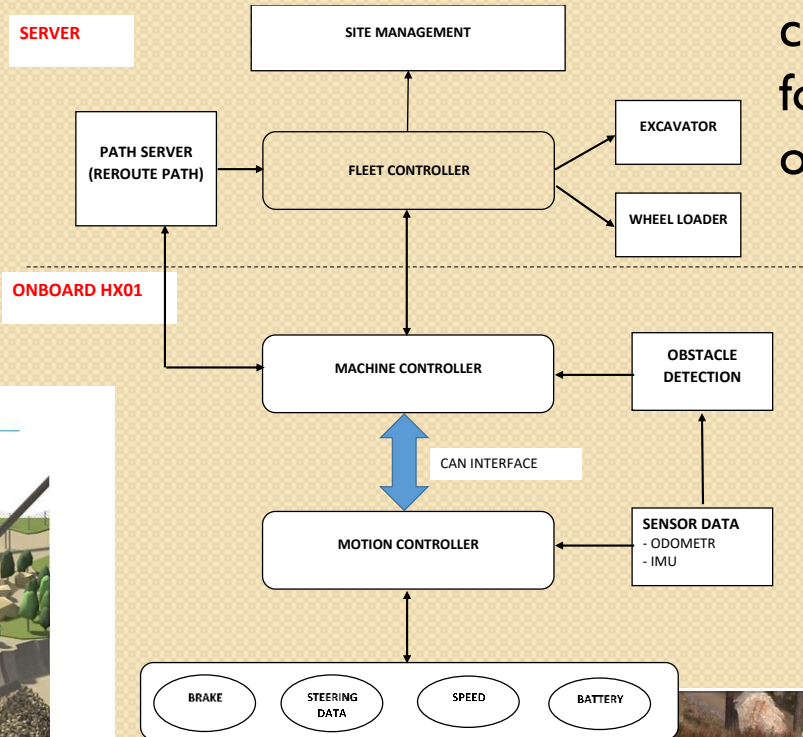
Automated Machines: ROS-based codes

(Volvo-CE, Stephan Baumgart and Torbjörn Martinsson)

- Derive the components and their computation and communication for Automated Hauler
- Check the schedulability and queue lengths by changing different parameters like sensing periods

Automated Hauler Platform HX01

✓ ROS based control system for autonomous operation



Safety Properties to be Checked

- At any point, 'Machine Control' node has a 'correct' knowledge of current position of 'wheel loader'.
- Fleet control node receives all the three required mandatory inputs (in order to perform error free fleet coordination).
- Application of Server emergency brake signal leads to stopping of the vehicle movement.
- Non- Availability of Sensor data 'can lead' to temporary vehicle pausing state.

SmartHub Project

(Unicam Smart MObility Lab, Andrea Polini and Francesco De Angelis)

- People moving in a city, and mobility services are distributed in Smart Hubs
- Smart Hubs are Local **container** of one or more smart mobility services
- Offering new traveling opportunities to the surrounding urban population.



As new mobility opportunities are introduced, commuters are expected to change their daily commuting patterns.

352A King's Road, London SW3 5UU, United Kingdom

Big Ben, London SW1A 0AA, United Kingdom

via A3212 19 min
12 min without traffic
3.3 miles
This route has tolls.
DETAILS

via King's Rd/A3217 and A302 23 min
13 min without traffic
2.7 miles

352A King's Road, London SW3 5UU, United Kingdom

Big Ben, London SW1A 0AA, United Kingdom

5:12 PM–5:40 PM 28 min
49 345 > Circle District
5:13 PM from Beaufort Street Kings Road (Stop QU)
4 min every 7 min
DETAILS

5:13 PM–5:40 PM 27 min
11 22 > Circle District

5:13 PM–5:46 PM 33 min
22 > Jubilee

5:14 PM–5:56 PM 42 min
11

352A King's Road, London SW3 5UU, United Kingdom

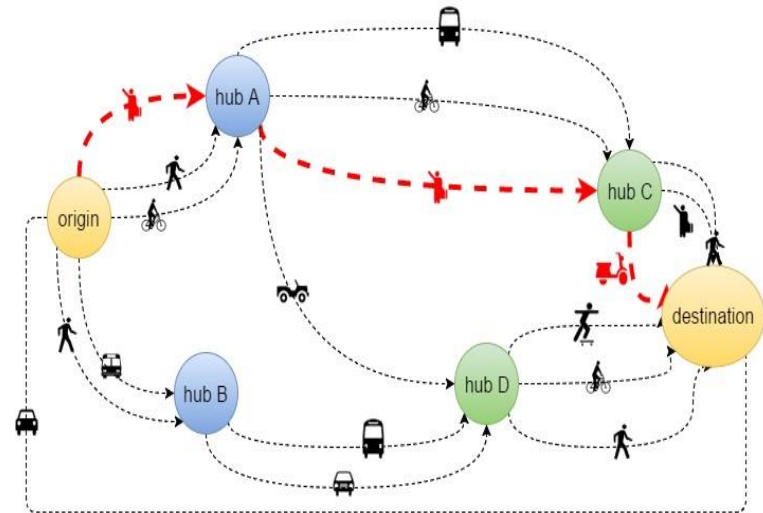
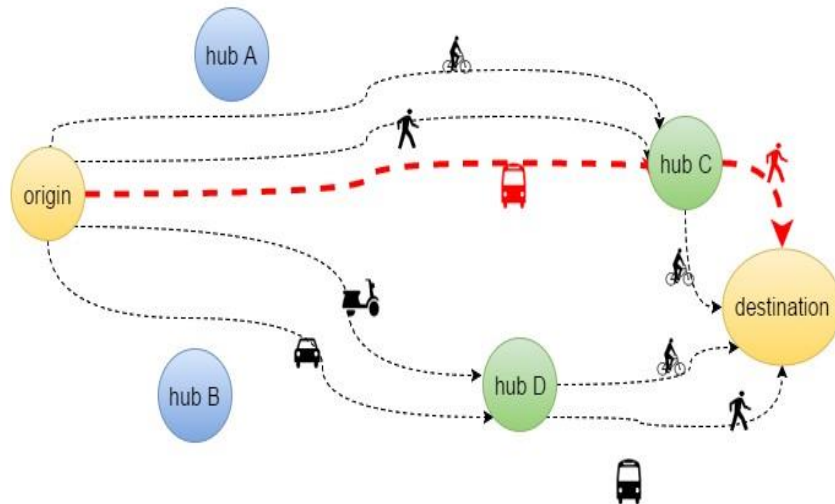
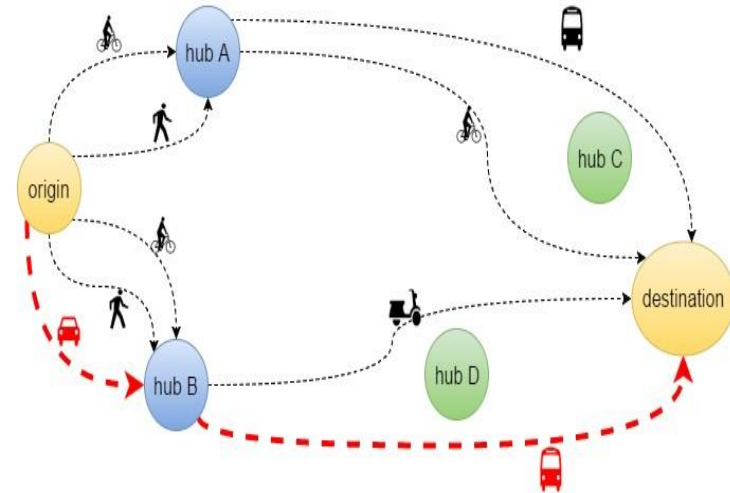
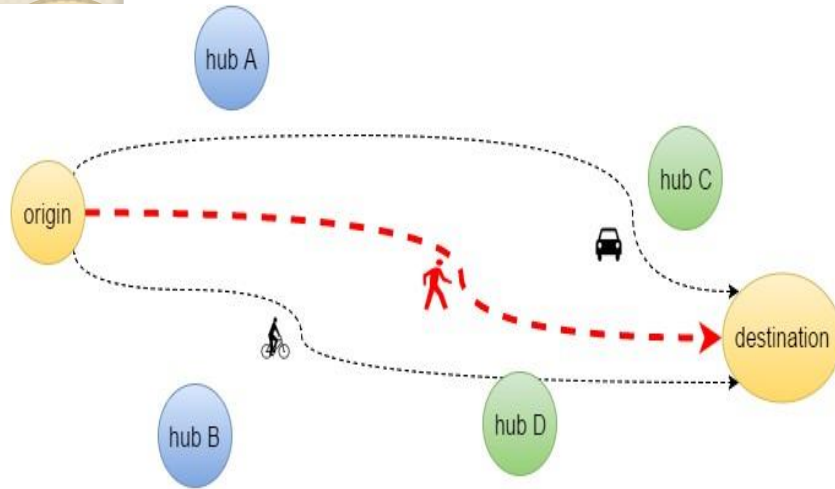
Big Ben, London SW1A 0AA, United Kingdom


via King's Rd/A3217 and A302 55 min
2.6 miles
DETAILS

via King's Rd/A3217 55 min
2.6 miles

via A3212 and A302 1 h 2 min
3.0 miles

With SmartHub, the commuter has a bunch of new different ways to organize his daily commuting patterns.

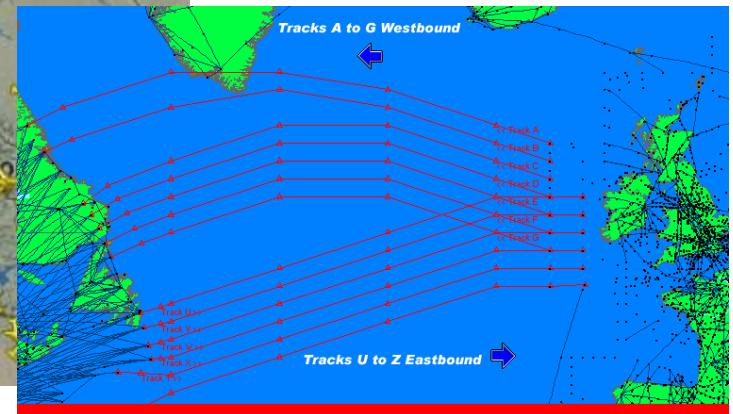
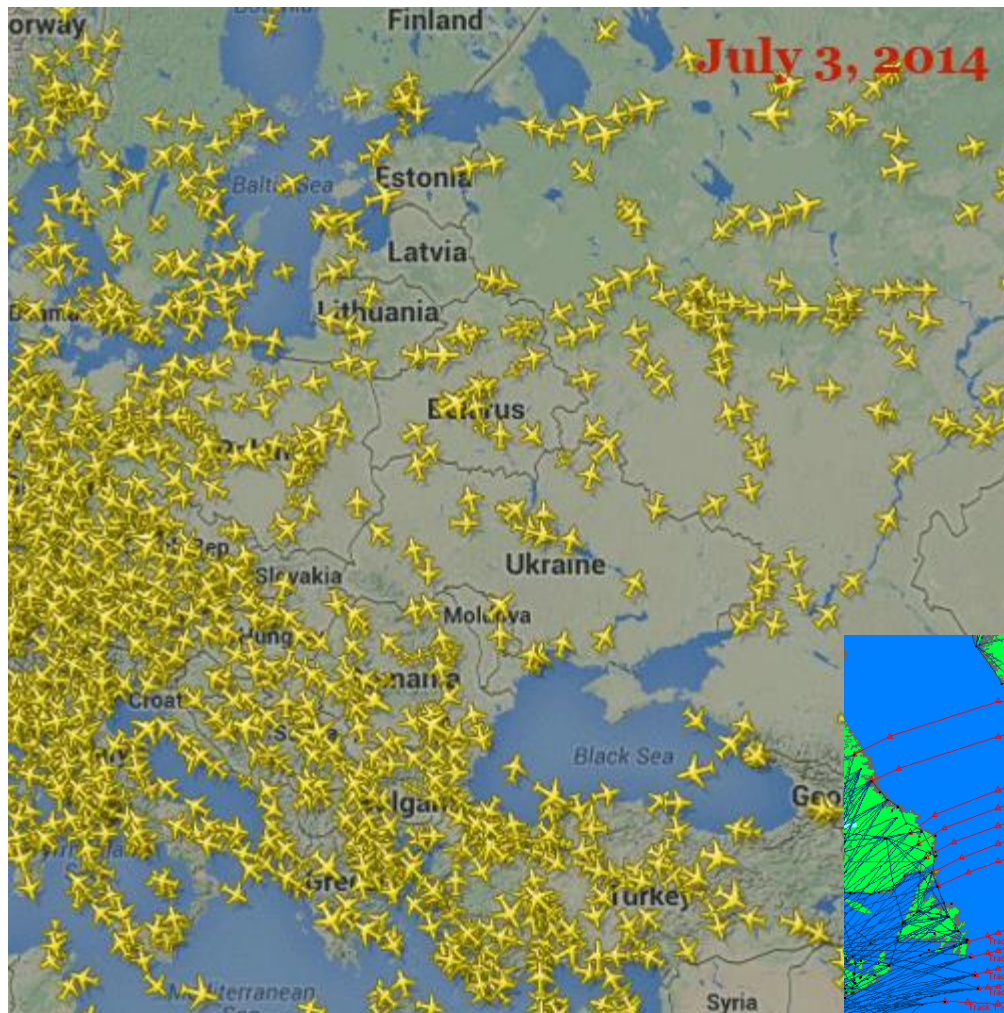


- 
- **Minimize:**
 - number of service disruptions
 - number of mobility resources in smarthubs
 - cost of mobility for commuters
 - travel time for commuters
 - travel distance for commuters

An example: mobility of Ascoli

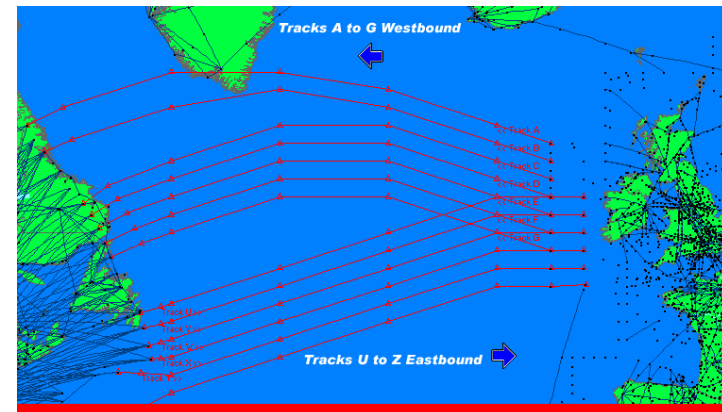
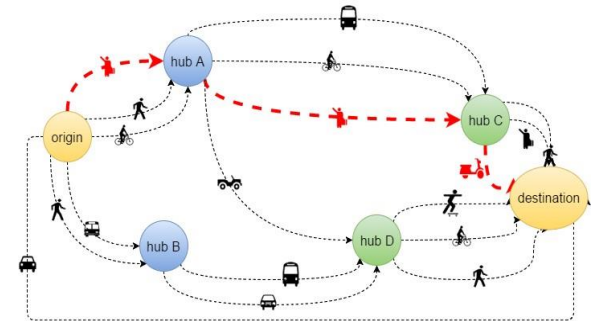
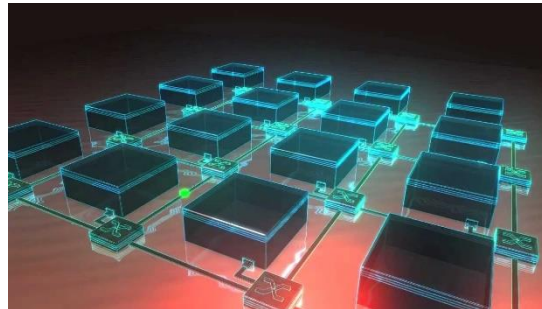
- 1317 commuters as messages
- 9 Smart-hubs as rebecs
- Start from an expensive configuration:
 - 75 vehicles for each smart-hub, i.e. 675 for the whole scenario
- After getting the results we were able to drastically decrease the number of vehicles for each smart-hub and configure them in a more rational way.


ATC



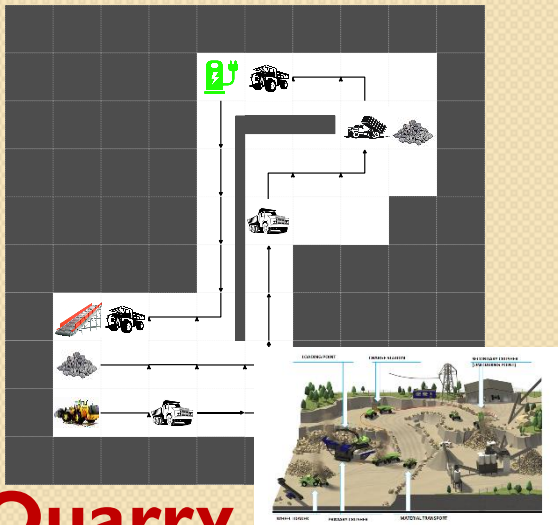
Track-based Traffic Control

- Common Pattern: Traffic is guided through pre-specified tracks

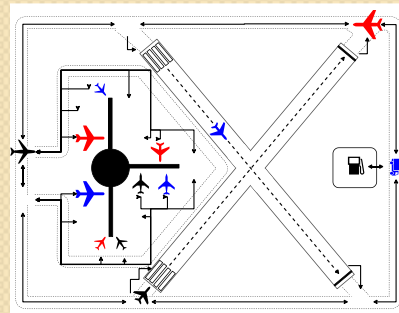


- 
- Flow Management Abstract View
 - Eulerian versus Lagrangian models
 - Track as Actor versus Object as actor
 - Adaptation
 - Model@runtime
 - Maginfier

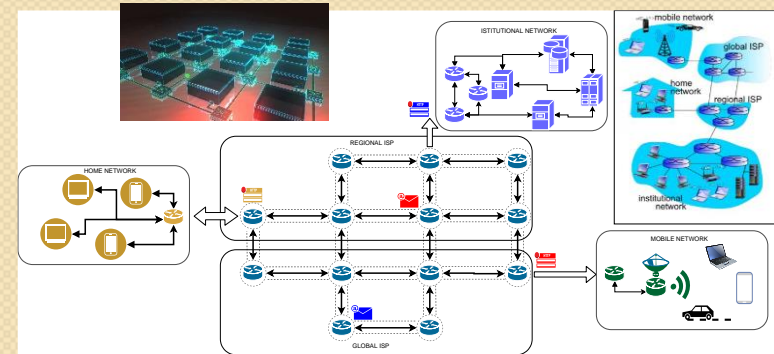
Flow Management: An Abstract View



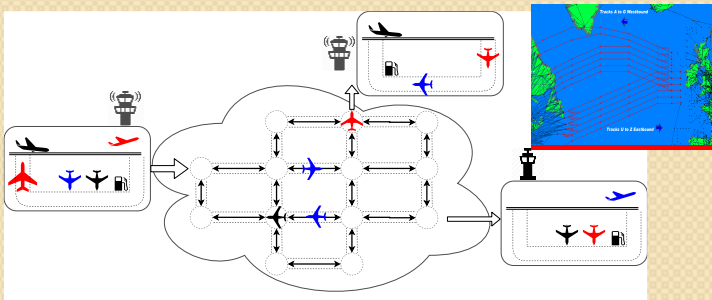
Quarry



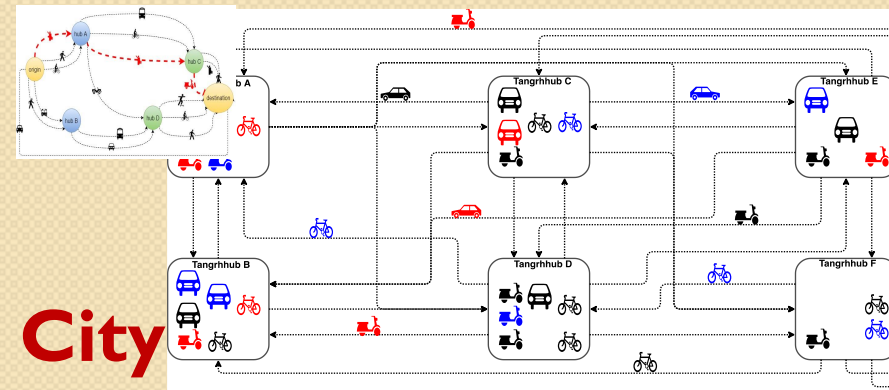
Airports



Networks

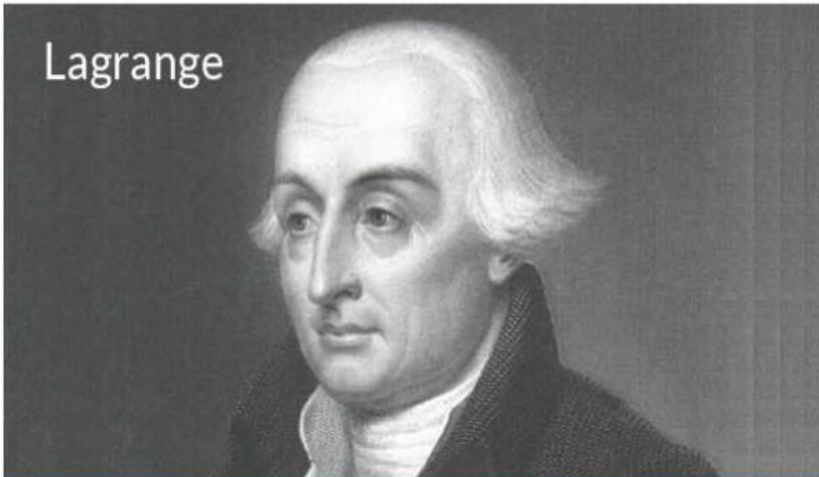


Air Space

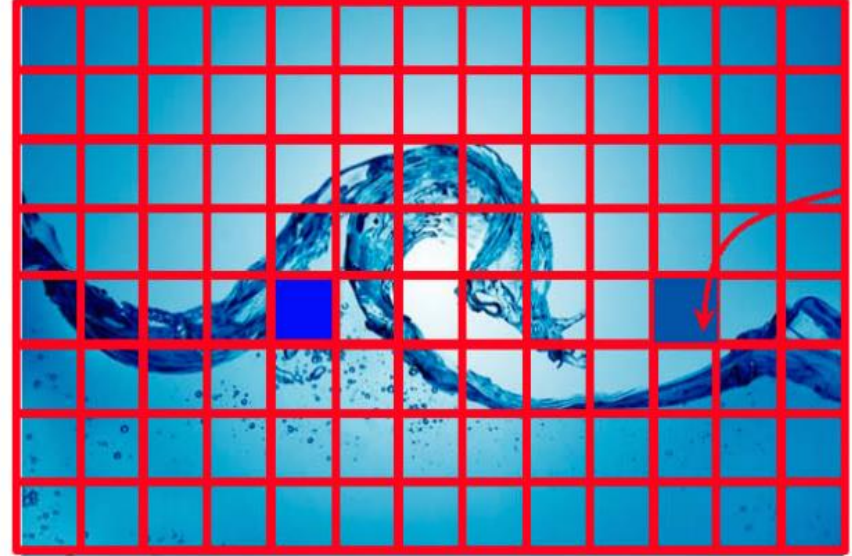


City

Lagrange



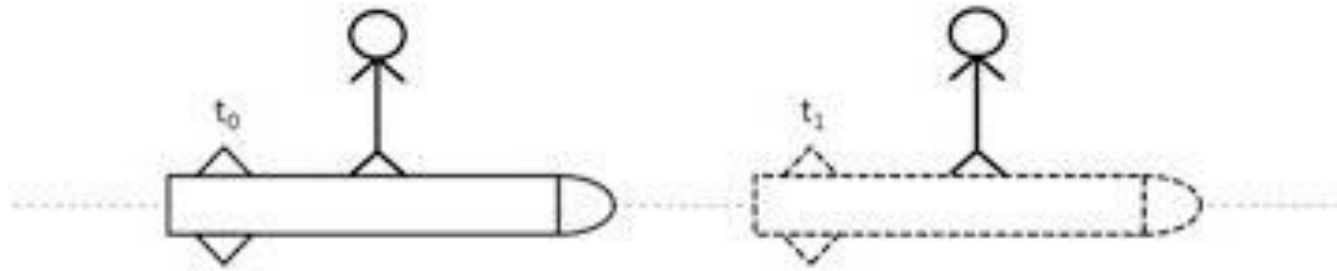
Euler



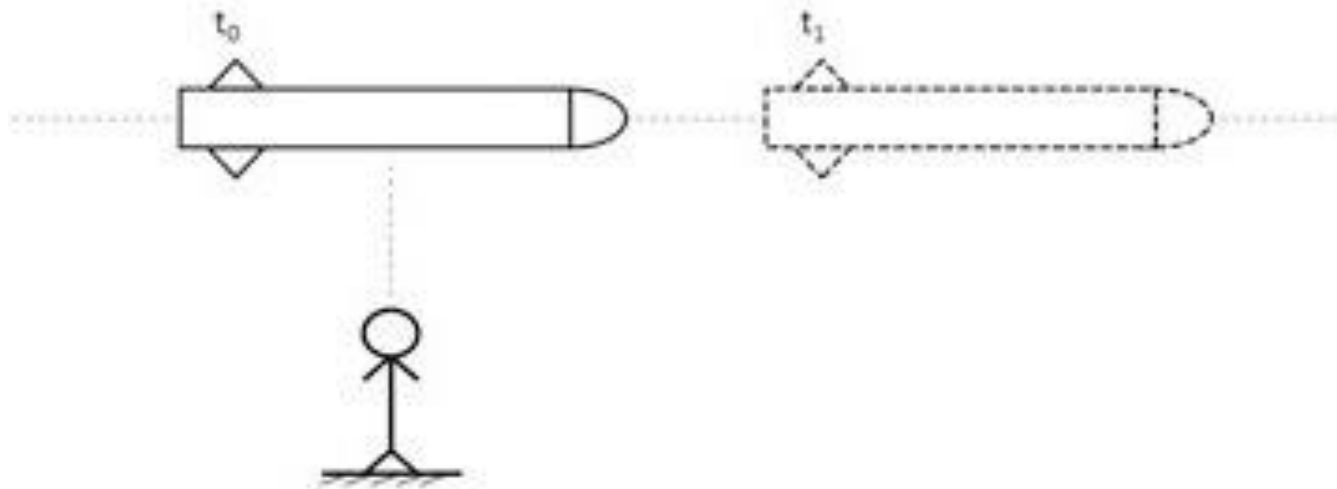
May need a copyright

Flow Analysis: Eulerian and Lagrangian

Lagrangian - An observer fixed to the missile

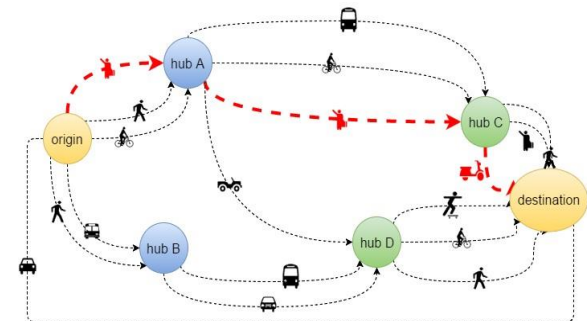
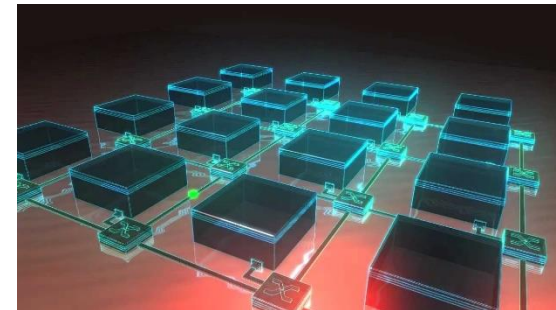
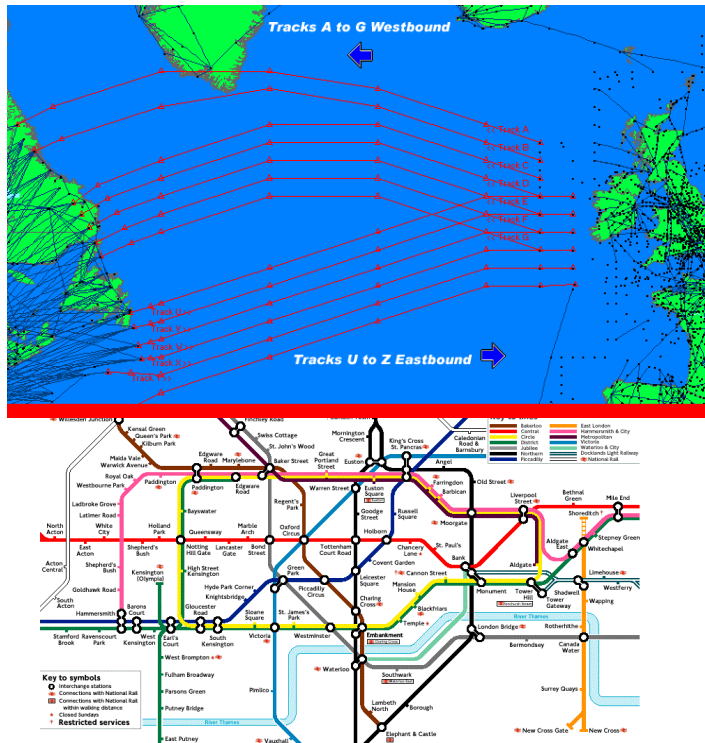


Eulerian - An observer on the ground to follow

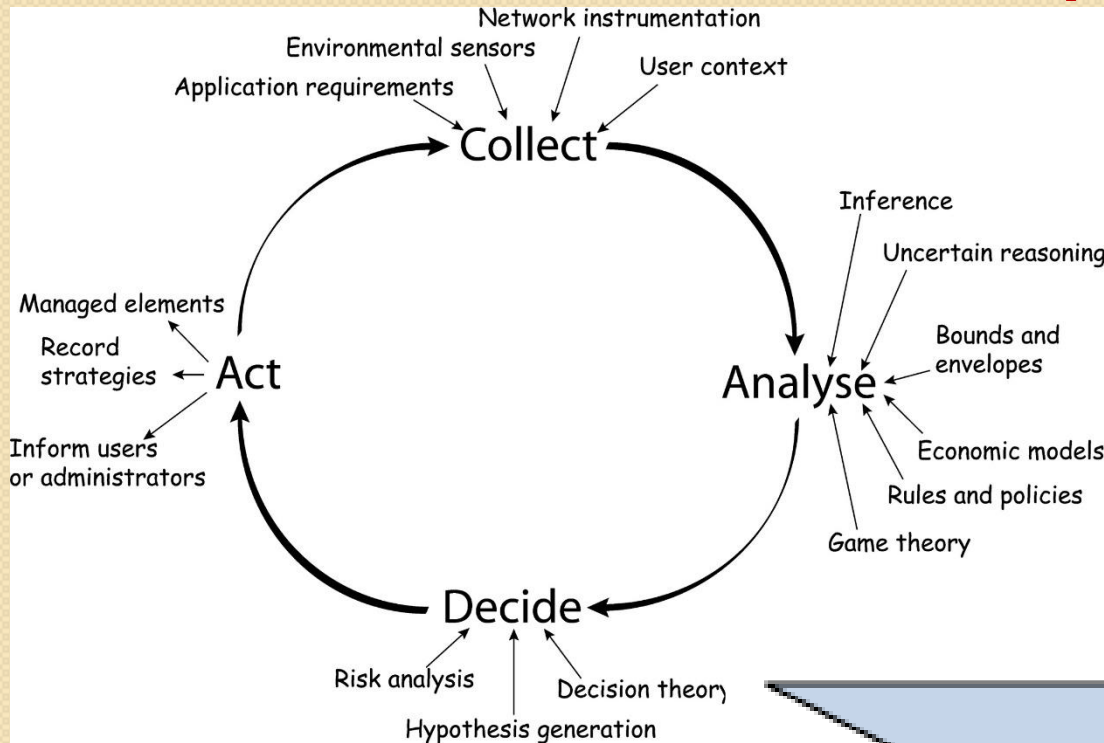


Track-based Traffic Control

- Globally Distributed – Locally Centralized
- Totally Distributed

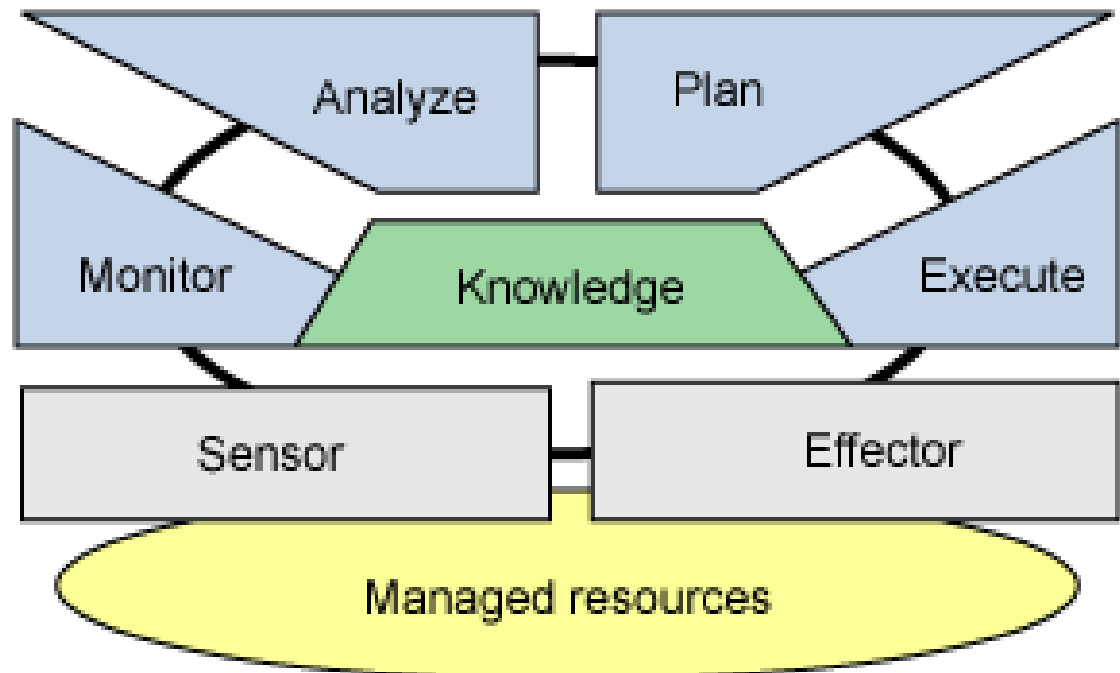


Autonomy and dealing with change



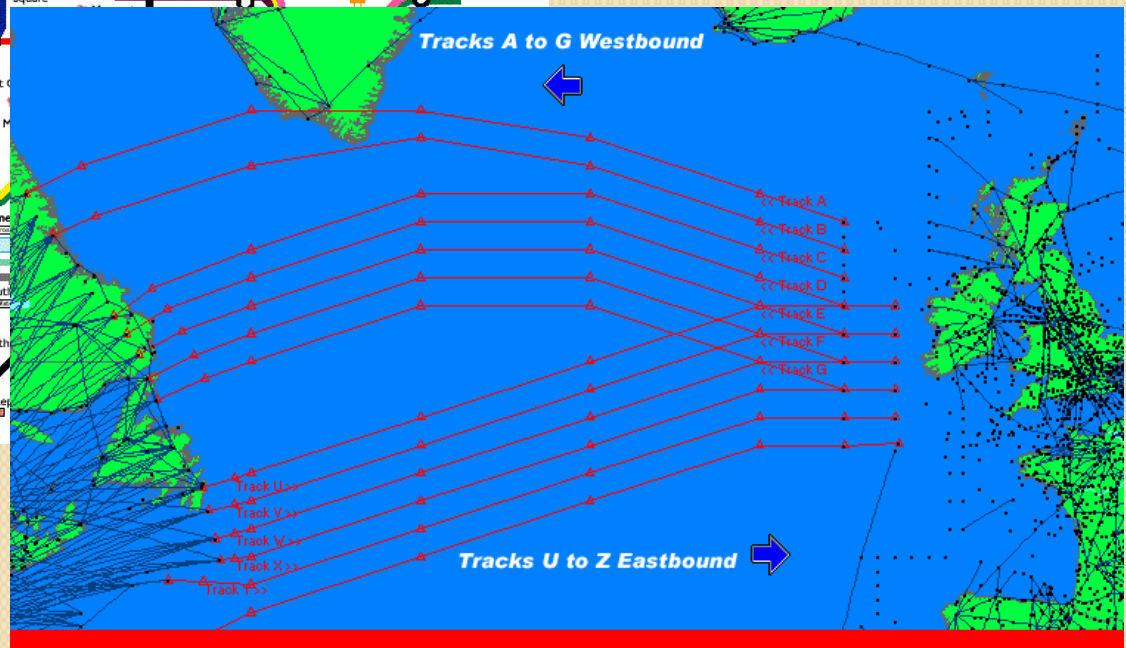
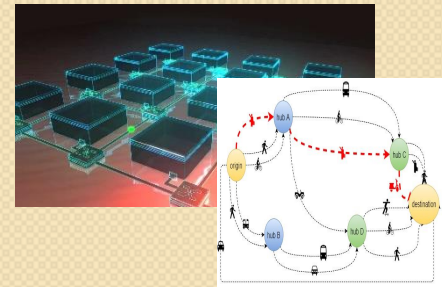
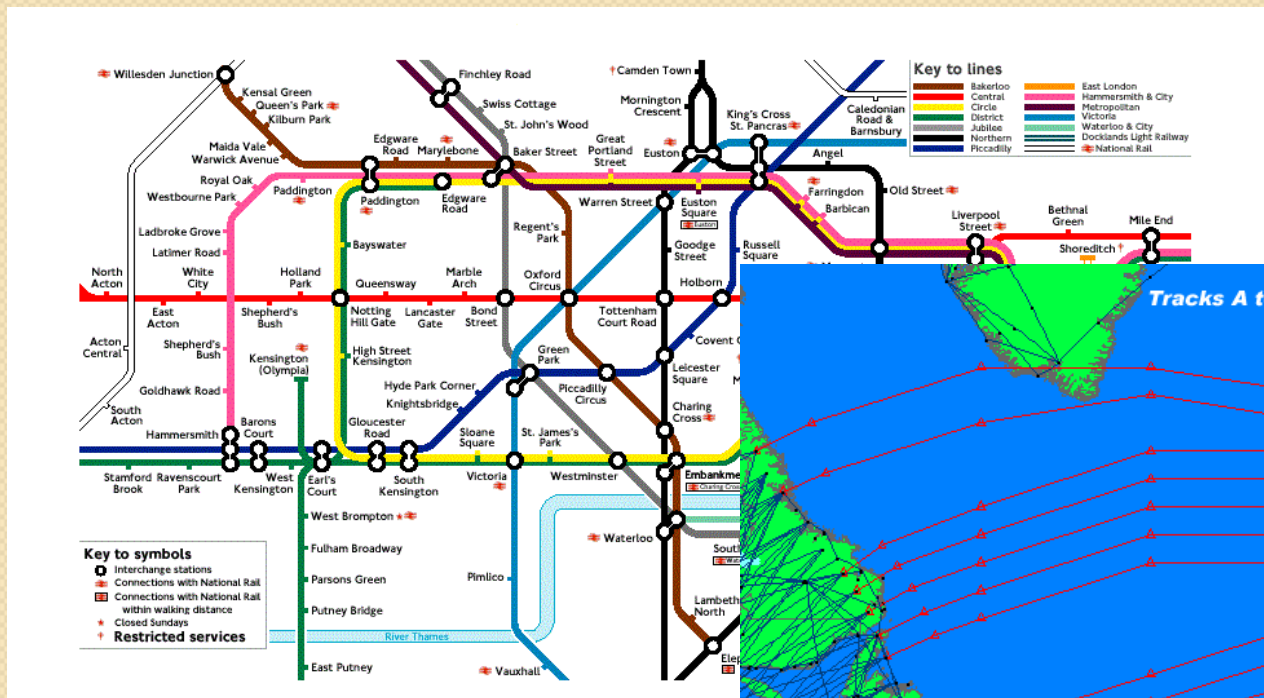
IBM MAPE-K

Typical Autonomic Control Loop



Common Pattern: Track-based Traffic Control

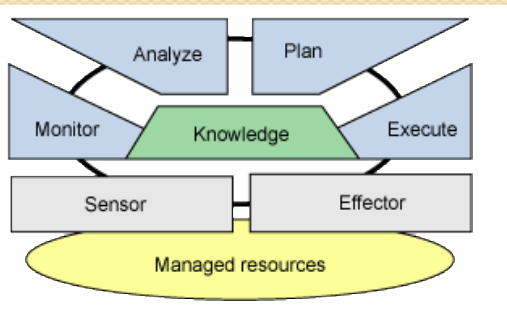
- Traffic is guided through pre-specified tracks, **and is coordinated**



Dependable Self-Adaptive Actors

(Ptolemy Group, UC Berkeley, Edward Lee)

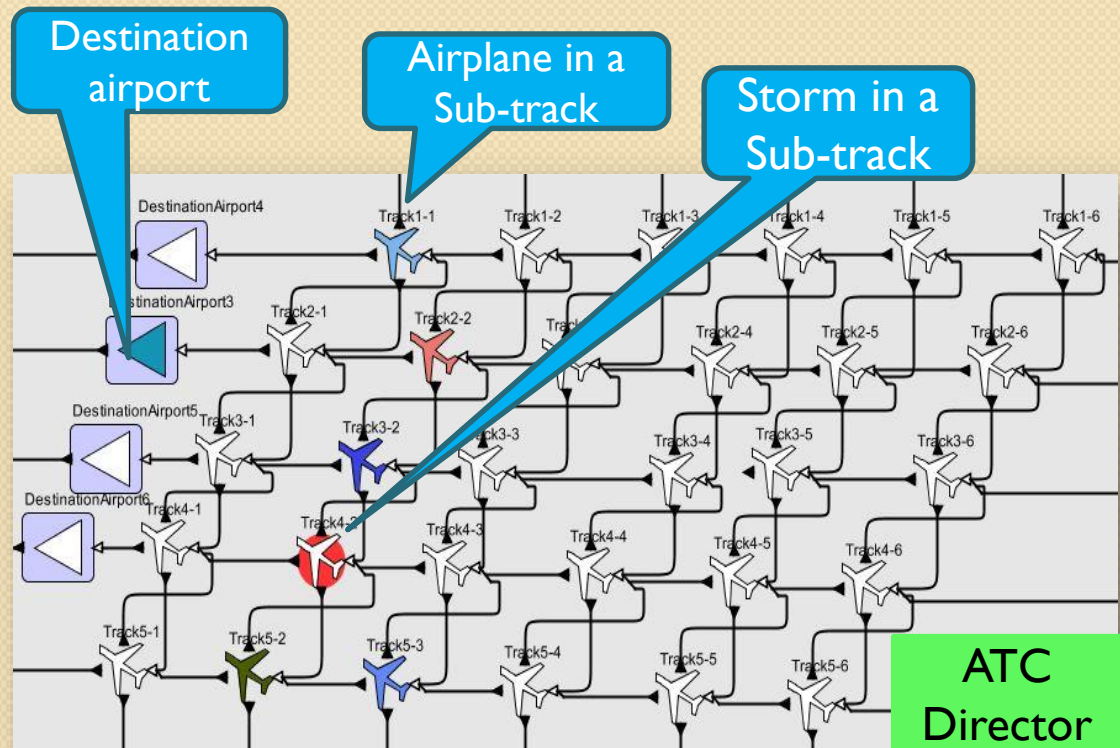
- Coordinated Actors in Ptolemy
- Model Change and Handle Rerouting
- Use `model@runtime`



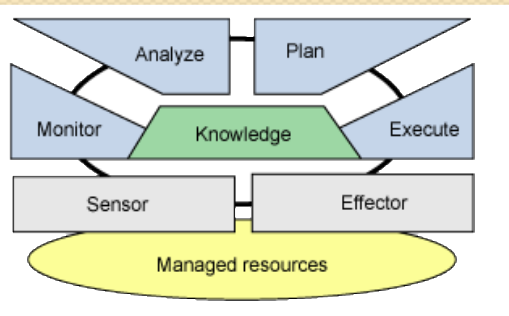
Challenges

- Designing **policies** for managing change through **rescheduling and rerouting** the traffic
- Assuring safety and acceptable performance while managing change
- Manage the timing issues and the uncertainty involved in the system

Model@runtime



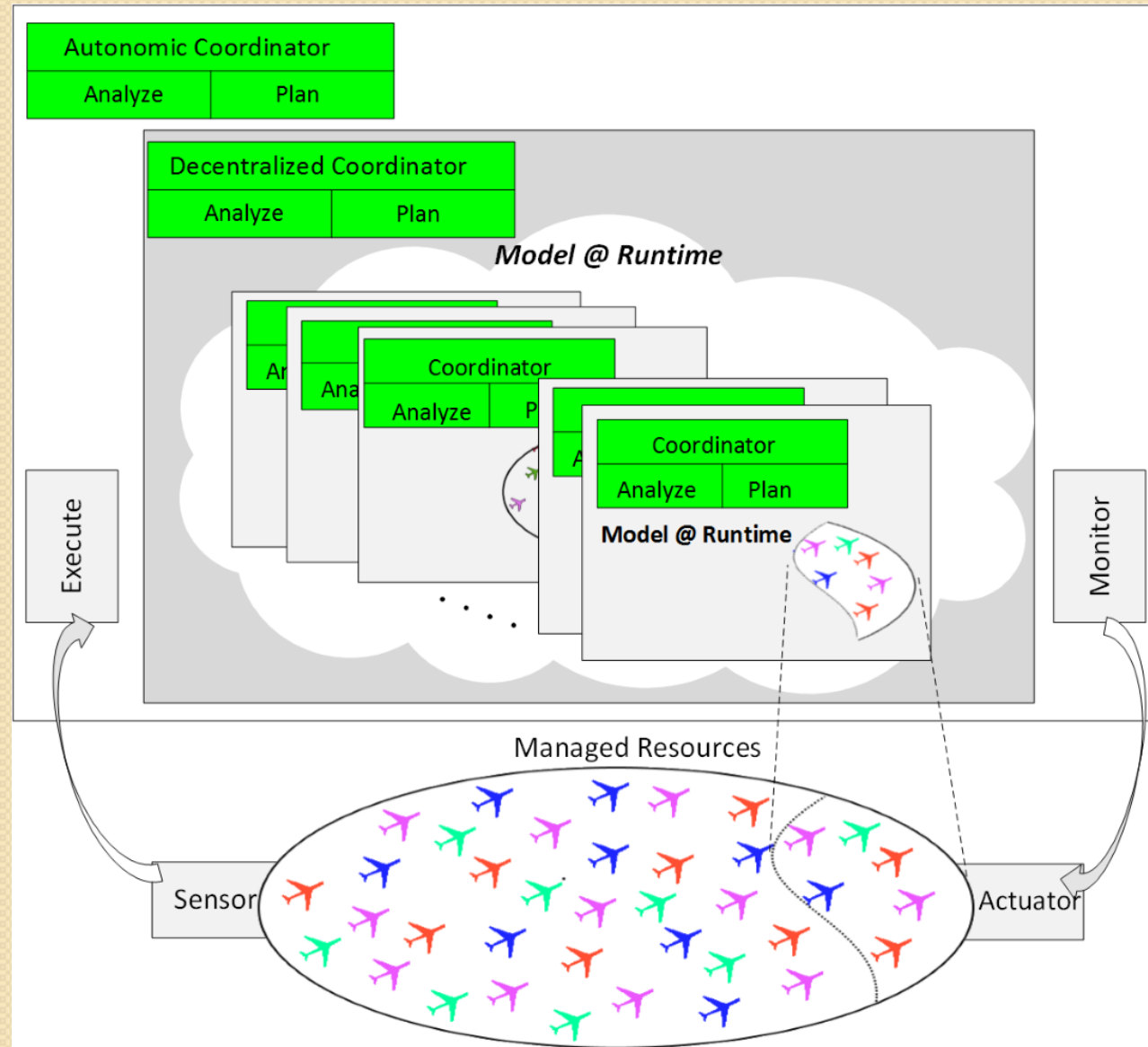
North Atlantic Organized Track System



Architecture

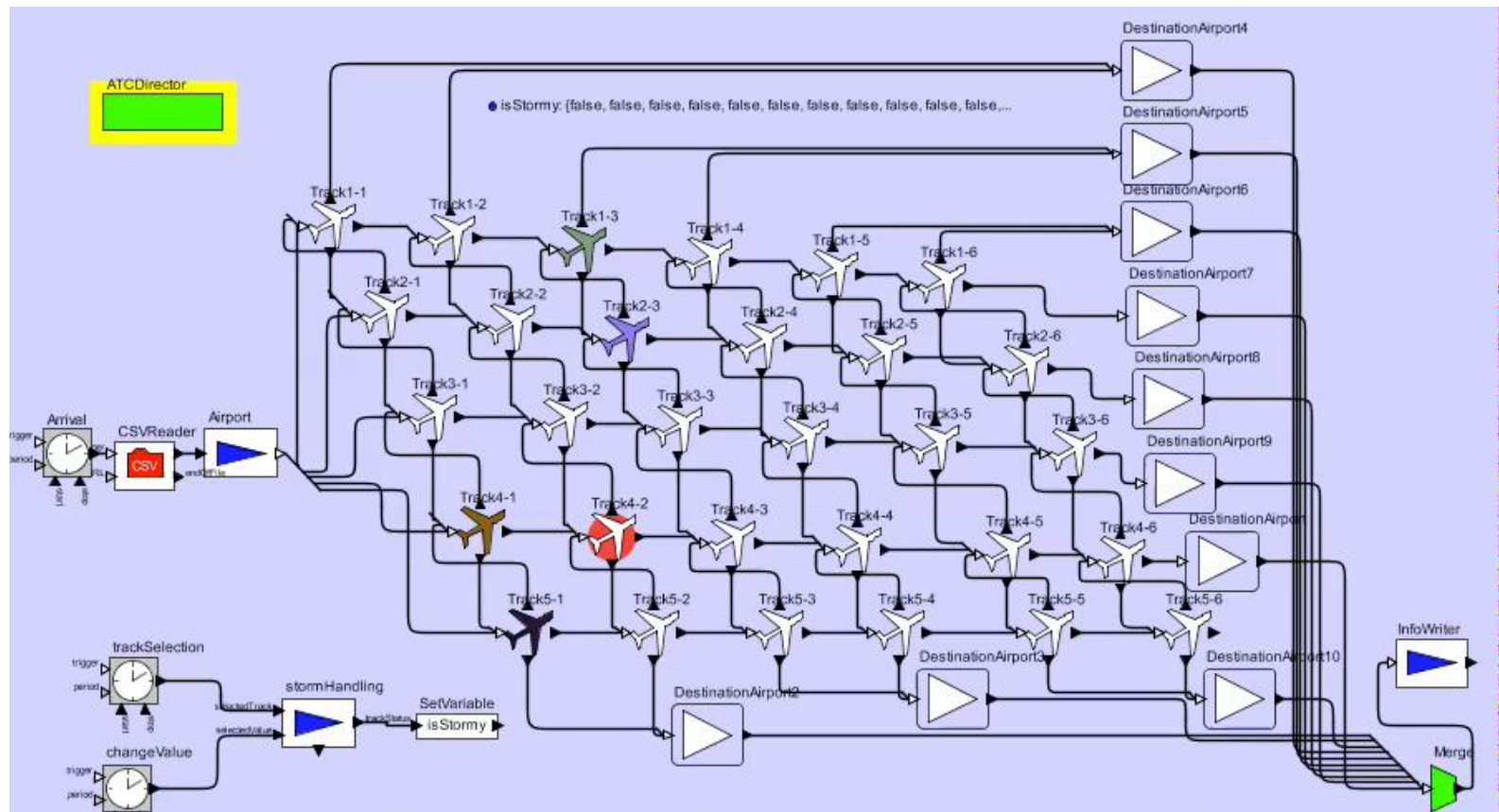
Challenges:

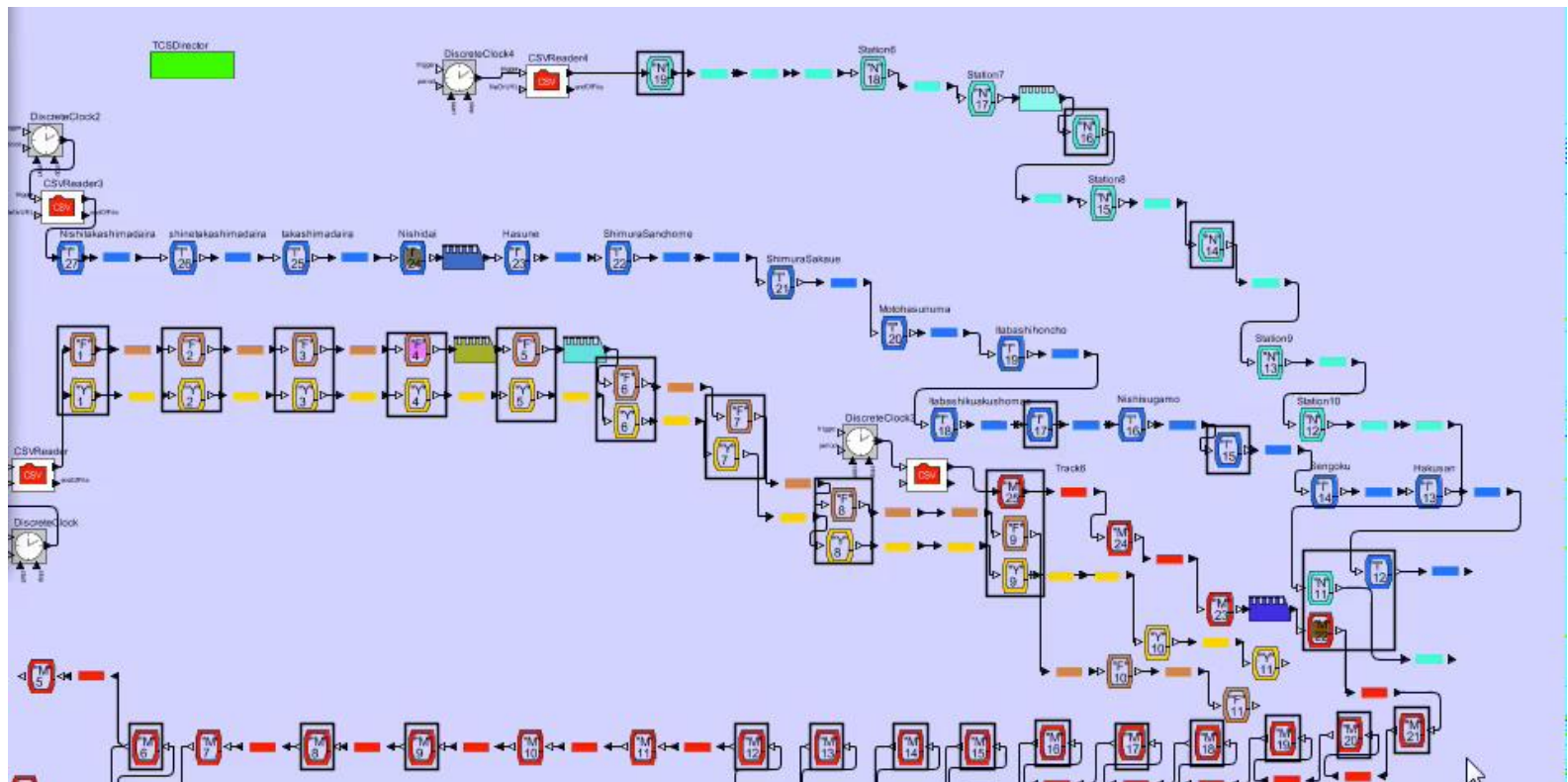
- Designing the coordination pattern that captures the **feedback loop** and the **interface** between the physical system and the control component: Autonomic Coordinator
- Building a structure to use the **centralized coordinators** as well as the **decentralized control**



Prediction ...

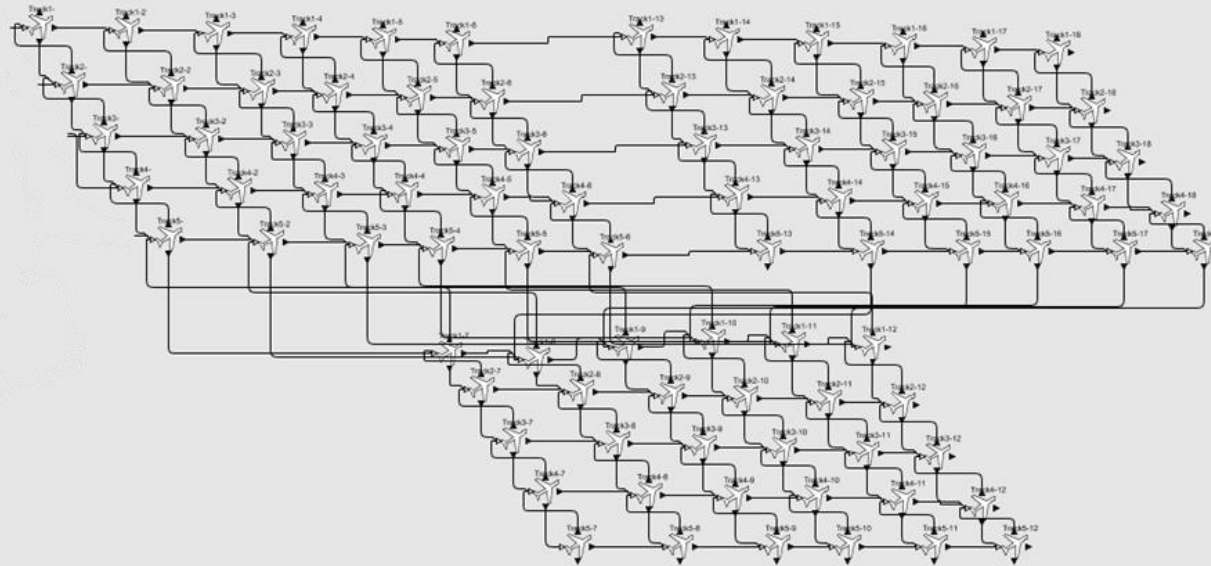
- Model predictive control
- Analyzer/Planner
 - Freeze the model as is (check-pointing)
 - Fork ahead to predict (clone the time)
 - Or clone the model and go ahead ...
- Monitor/Execute
 - Connect to the sensors and actuators





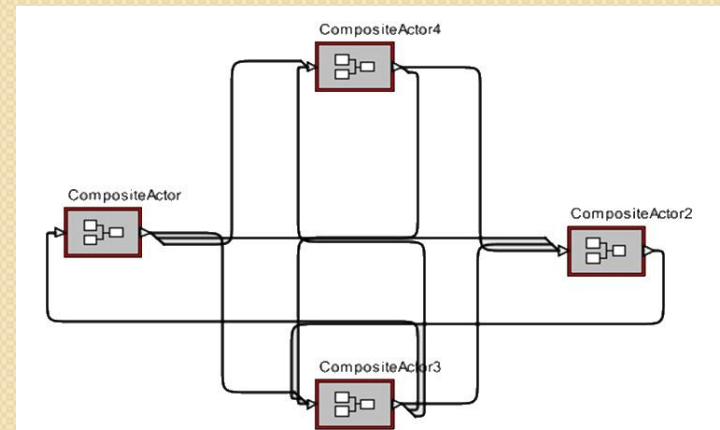
Decentralized Director ... allow non-determinism

When we have a centralized controller (within one control area), then we have no non-determinism caused by concurrency.

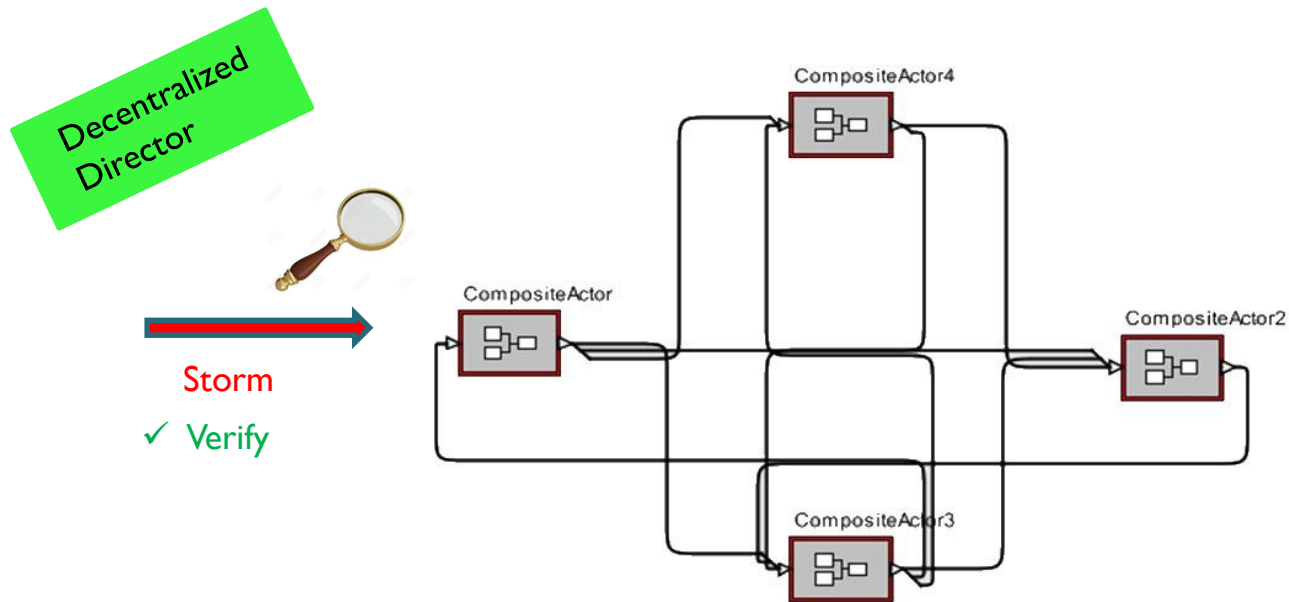


At runtime, there is no randomly generated storm. So, nondeterminism does not exist.

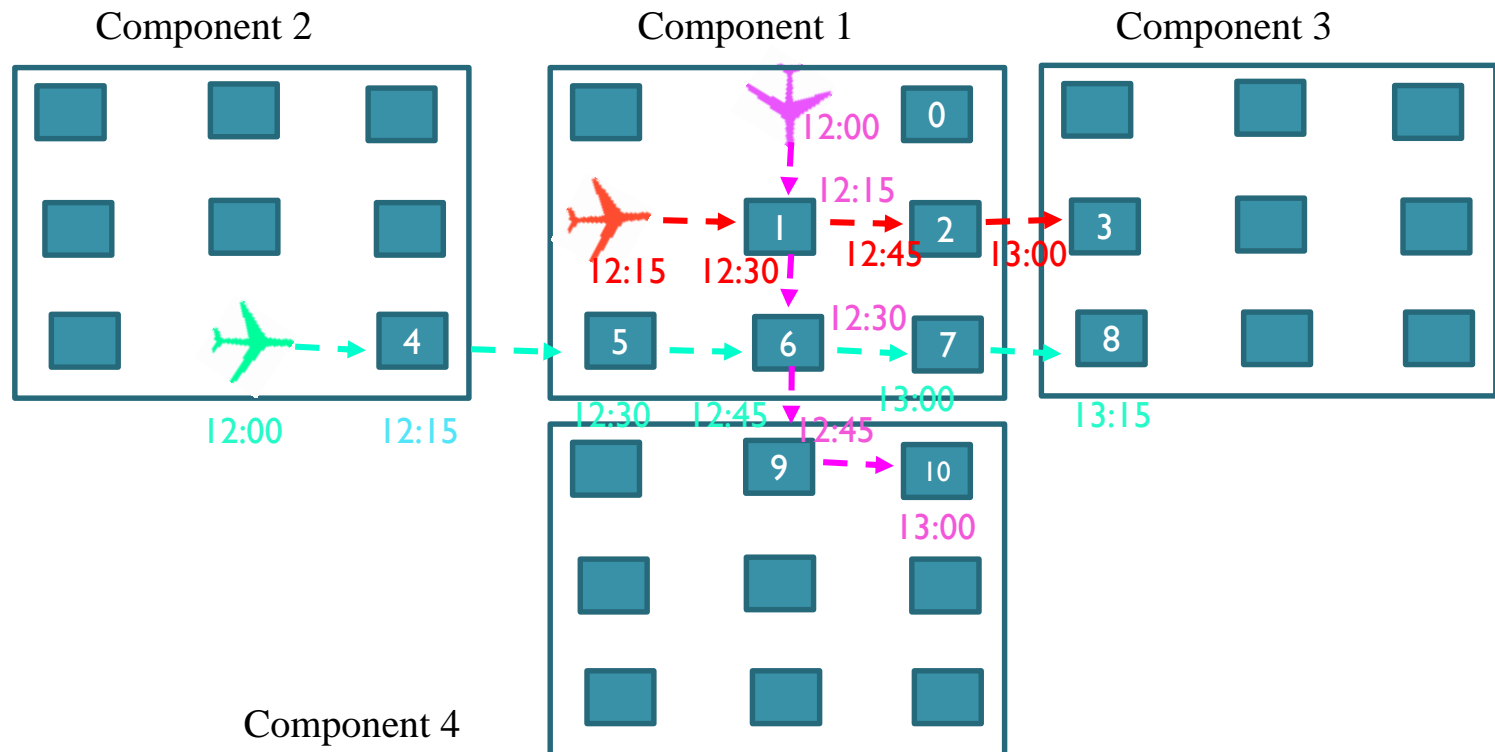
When we have two control areas, we are back in a “real” distributed world with nondeterminism.



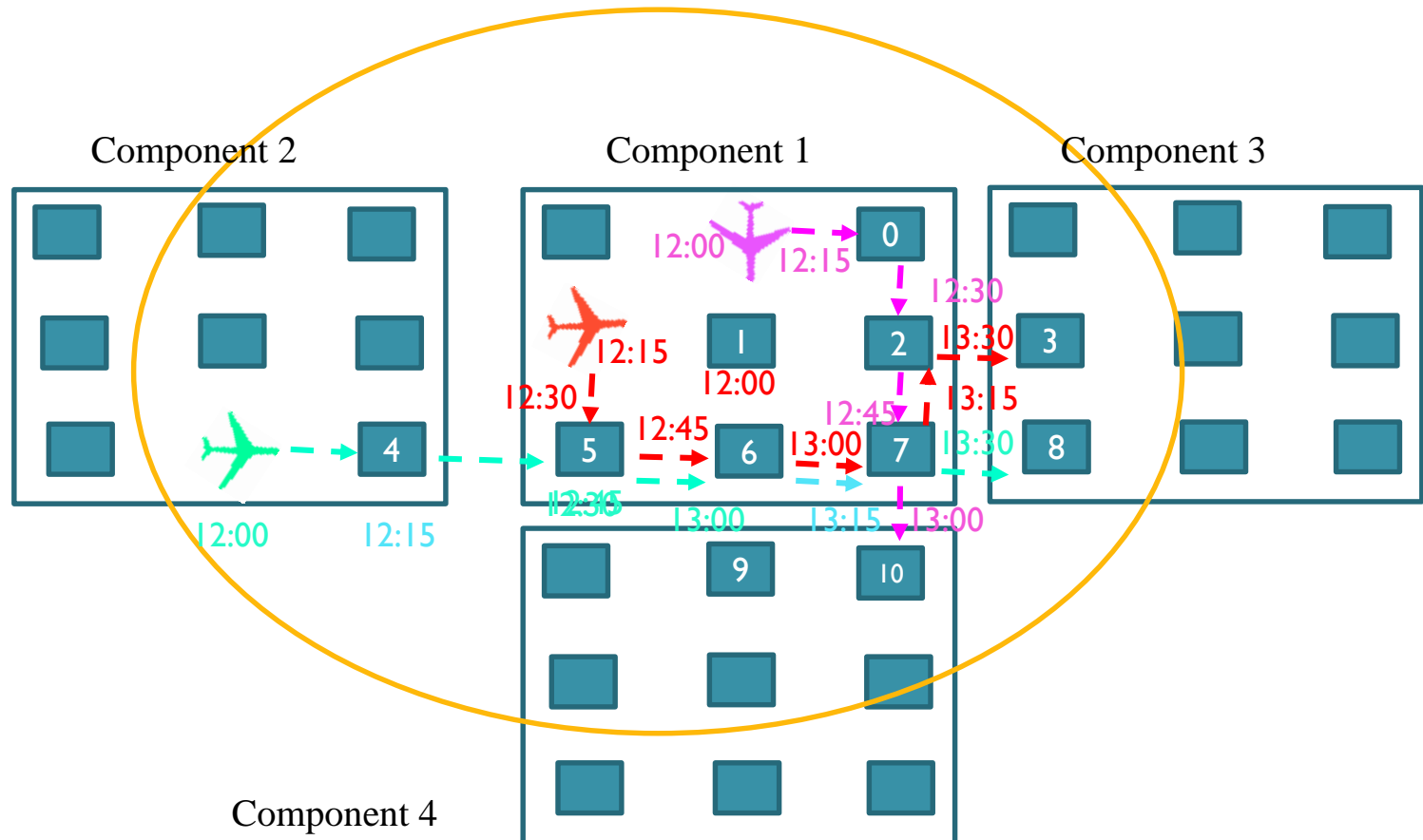
Analysis using Magnifier: Zoom-in – Zoom-out



Re-routing Example: Propagation and Safety Zone



We have a storm ...



References

- M. Sirjani, A. Movaghar, and M.R. Mousavi, **Compositional Verification of an Actor-Based Model for Reactive Systems**, in Proceedings of Workshop on Automated Verification of Critical Systems (AVoCS'01), Oxford University, April **2001**.
- M. Sirjani, M. M. Jaghoori, **Ten Years of Analyzing Actors: Rebeca Experience**, LNCS 7000, pp. 20-56, **2011**.

Invited paper at **Carolyn Talcott Festschrift, 70th birthday**

- M. Sirjani, A. Movaghar, A. Shali, F.S. de Boer, **Modeling and Verification of Reactive Systems using Rebeca**, Fundamenta Informaticae, Volume 63, Number 4, ISSN 0169-2968, pp. 385-410, 2004.
- M. M. Jaghoori, M. Sirjani, M. R. Mousavi, E. Khamespanah, A. Movaghar, **Symmetry and Partial Order Reduction Techniques in Model Checking Rebeca**, Acta Informatica, Volume 47, Issue 1, pp. 33-66, 2009.
- N. Razavi, R. Behjati, H. Sabouri, E. Khamespanah, A. Shali, M. Sirjani, **Sysfier: Actor-based Formal Verification of SystemC**, ACM Transactions on Embedded Computing Systems, Vol. 10, No. 2, Article 19, **2010**.

Timed Rebeca

- M. J. Izadi, [Analyzing Timed Rebeca using UPPAAL](#) , 2009, MSc Thesis in Farsi
- L. Aceto, M. Cimini, A. Ingolfsson, A. H. Reynisson, S. H. Sigurdarson, M. Sirjani, Modelling and [Simulation of Asynchronous Real-Time Systems using Timed Rebeca](#), ENTCS, Proceedings of FOCLASA 2011
- E. Khamespanah, Z. Sabahi Kaviani, R. Khosravi, M. Sirjani, M. J. Izadi, [Timed-Rebeca Schedulability and Deadlock-Freedom Analysis Using Floating-Time Transition System](#), ACM AGERE! 2012
- Z. Sharifi, M. Mosaffa, S. Mohammadi, M. Sirjani, [Functional and Performance Analysis of Network-on-Chips Using Actor-based Modeling and Formal Verification](#), AVoCS 2013
- Z. Sharifi, S. Mohammadi, M. Sirjani, [Comparison of NoC Routing Algorithms Using Formal Methods](#), PDPTA 2013
- E. Khamespanah , M. Sirjani, M. Viswanathan, R. Khosravi, [Bounded Floating-Time Transition System: Significant Reduction for Analysing Actors](#), FACS 2015