Actors for Analysis

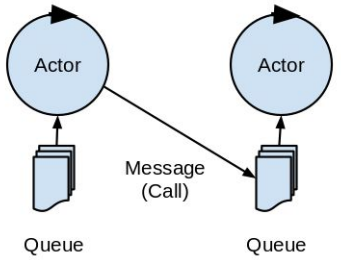
Marjan Sirjani
Professor in Software Engineering
MDH, IDT, Sweden
Cyber-Physical Systems Analysis Group

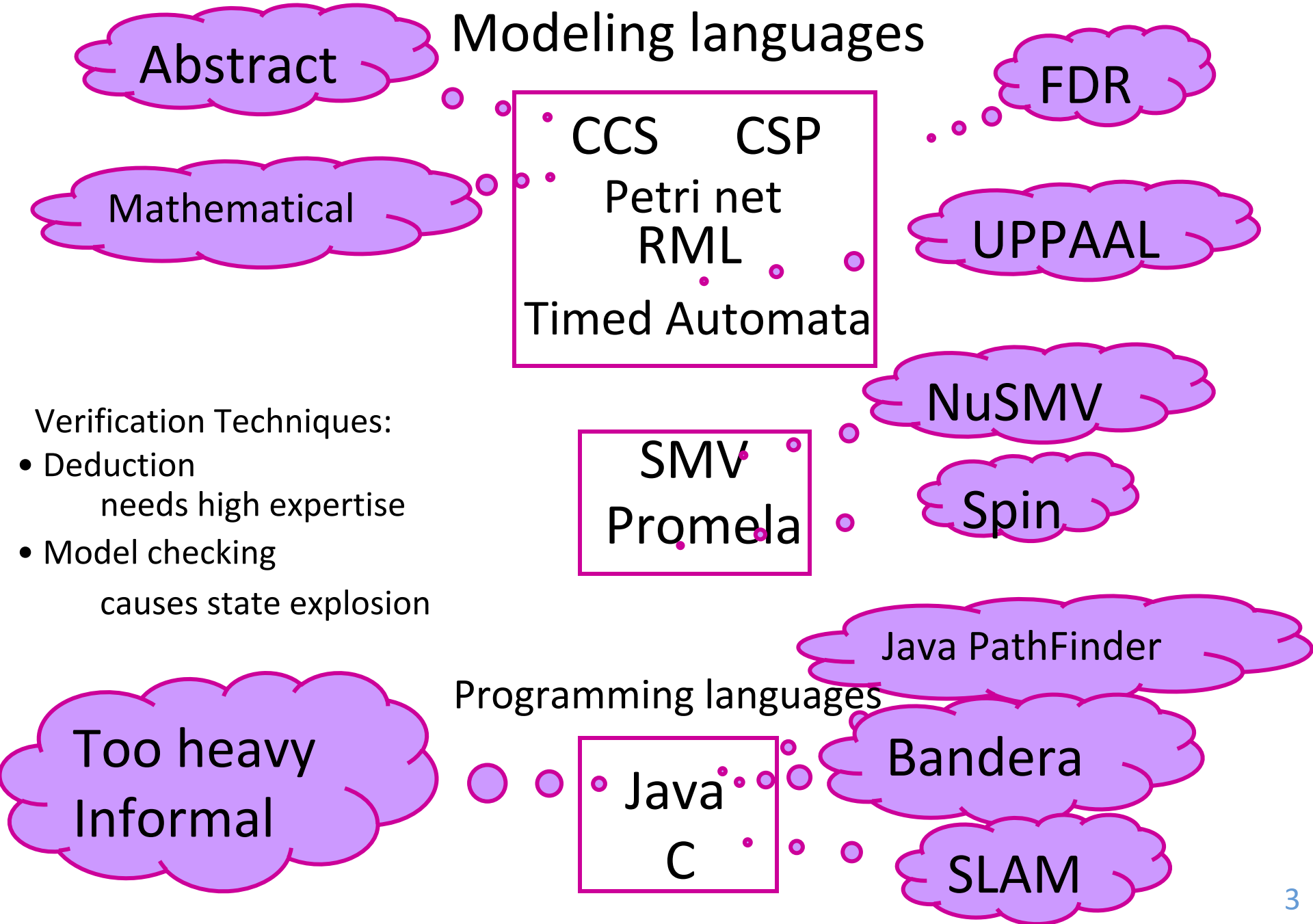
Shonan Village Center, Japan
May 27, 2019

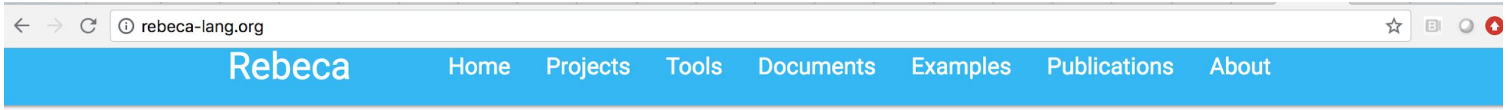
Rebeca: Reactive object language

(Sirjani, Movaghar, **2001**)

- **Designed Rebeca for model checking distributed systems**
  - Based on Hewitt actors, and Gul's papers
  - Java like syntax
- Communication:
  - Asynchronous message passing: non-blocking send
  - No explicit receive
- Computation:
  - Event-driven
  - Non-preemptive methods



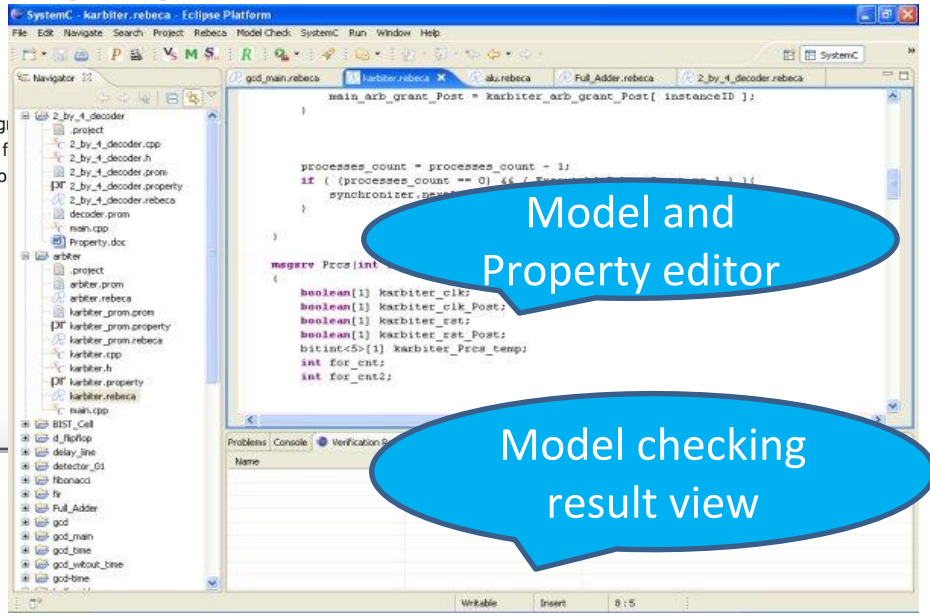

Modeling languages
Abstract
Mathematical
CCS
CSP
Petri net
RML
Timed Automata
FDR
UPPAAL
NuSMV
Spin
SMV
Promela
Verification Techniques:
• Deduction
needs high expertise
• Model checking
causes state explosion
Too heavy
Informal
Programming languages
Java
C
Java PathFinder
Bandera
SLAM



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

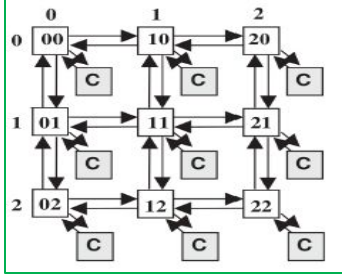
Rebeca Modeling Language

Actor-based Language with Formal Foundation



- **Ten years of Analyzing Actors: Rebeca Experience (Sirjani, Jaghouri), Invited paper, Carolyn Talcott Festschrift: 70th birthday, LNCS 7000, 2011**
- **On Time Actors (Sirjani, Khamespanah), Invited paper, Theory and Practice of Formal Methods, Frank de Boer Festschrift, LNCS 9660, 2016**
- **Power is Overrated, Go for Friendliness! Expressiveness, Faithfulness and Usability in Modeling - The Actor Experience (Sirjani), Invited Paper, Principles of Modeling, Edward Lee Festschrift, LNCS 10760, 2017.**

ASPIN: Rebeca abstract model




```
reactiveclass Router{
  knownrebecs
    {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 …
} }
```

Actor type and its message servers

Constructor

A message server

Asynchronous message sending

```
reactiveclass Core{
  knownrebecs {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)
    …
}
```

Instances of different actors

Known rebecs

Parameters

Analysis:
Starting from Model Checking

Use Actor characteristics for state space reduction

- Compositional Verification
- Symmetry Reduction

- Add time and probability
  - Floating Time Transition System

- Used for different types of analysis in different domains …

Cyber-Physical Systems: Scientific Foundations and Challenges (Insup Lee et al)

- CPS Composition
- Robustness, Safety, and Security of CPS
- Control and Hybrid Systems
- Computational Abstractions
- Architecture
- Real-Time Embedded Systems Abstractions
- Sensor and Mobile Networks
- Model-based Development of CPS
- Verification, Validation, and Certification of CPS

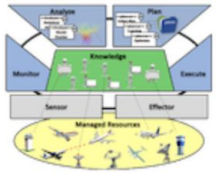
Models, Methodologies, and Tools

- 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
  - Performance
  - Security from a Safety Point of View

Sample Projects

- NoC Design
  - Routing and Dispatching Analysis, Explore the design space
- 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

Projects



SEADA

In SEADA (Self-Adaptive Actors) we will use Ptolemy to represent the architecture, and extensions of Rebeca for modeling and verification. Our models@runtime will be coded in an extension of Probabilistic Timed Rebeca, and supporting tools for customized run-time formal verification

RoboRebeca

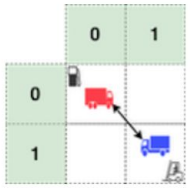
RoboRebeca is a framework which provides facilities for developing safe/correct source codes for robotic applications. In RoboRebeca, models are developed using Rebeca family language and automatically transformed into ROS compatible source codes. This framework is

HybridRebeca

Hybrid Rebeca, is an extension of actor-based language Rebeca, to support modeling of cyber-physical systems. In this extension, physical actors are introduced as new computational entities to encapsulate the physical behaviors. Learn more

Tangramob

Tangramob offers an Agent-Based



AdaptiveFlow

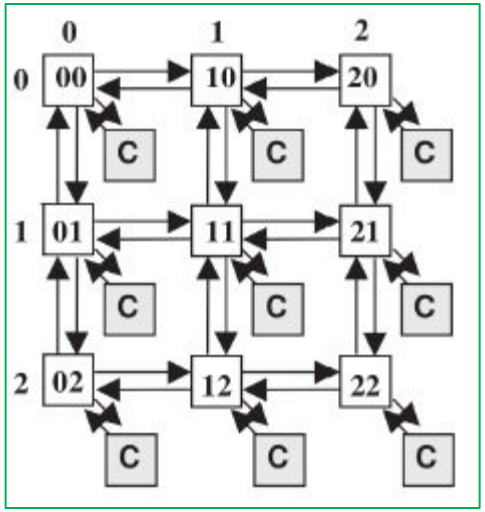
AdaptiveFlow is an actor-based eulerian

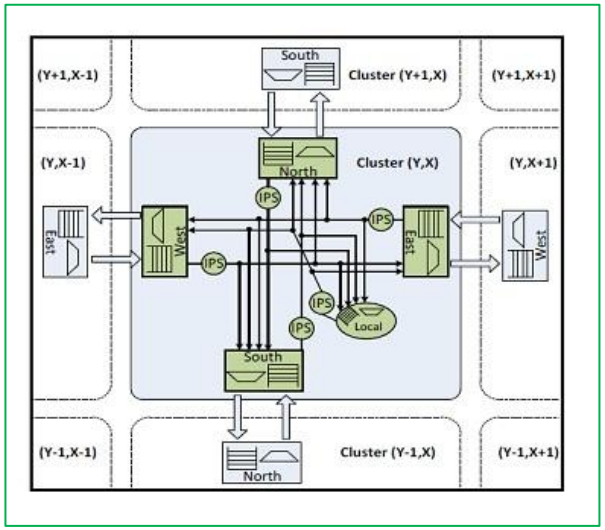
wRebeca

wRebeca is an actor-based modeling

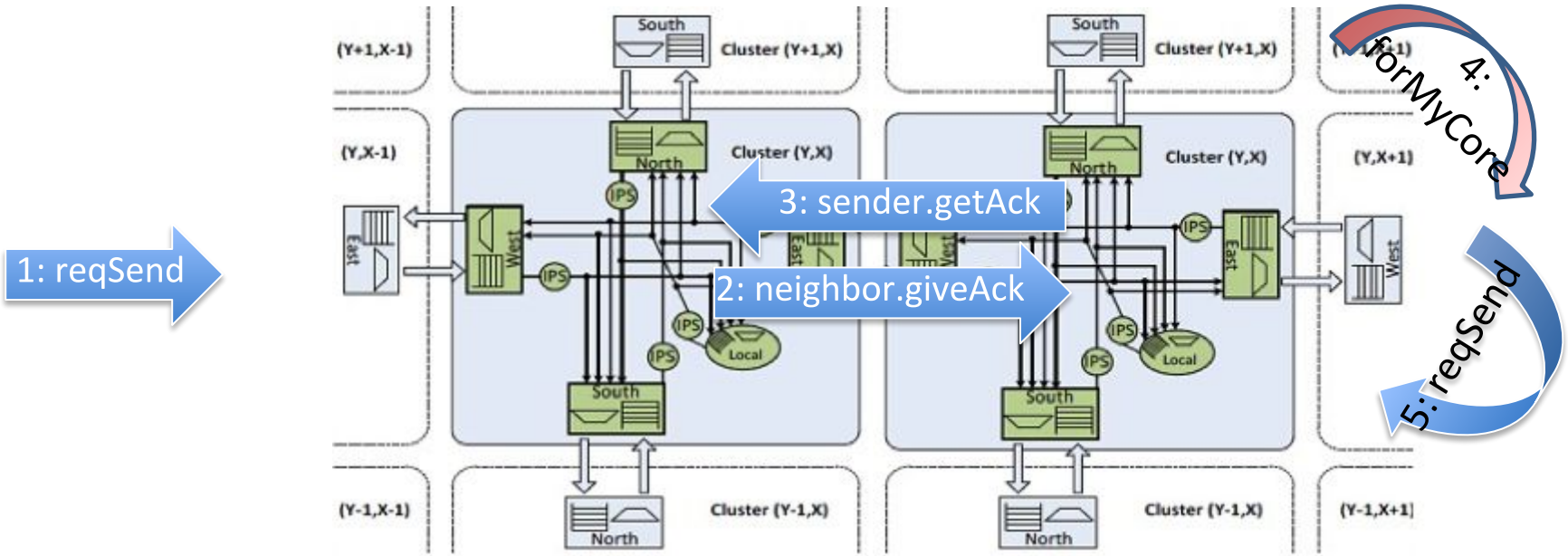
Network on Chip
(UT, Siamak Mohammadi)

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
  - …



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;

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.

- The AODVv2-version11
  - Goal improving the performance
  - Consequence jeopardizing the loop freedom property

- The AODVv2-version13 and version 16
  - Bugs were found
  - Loop is formed

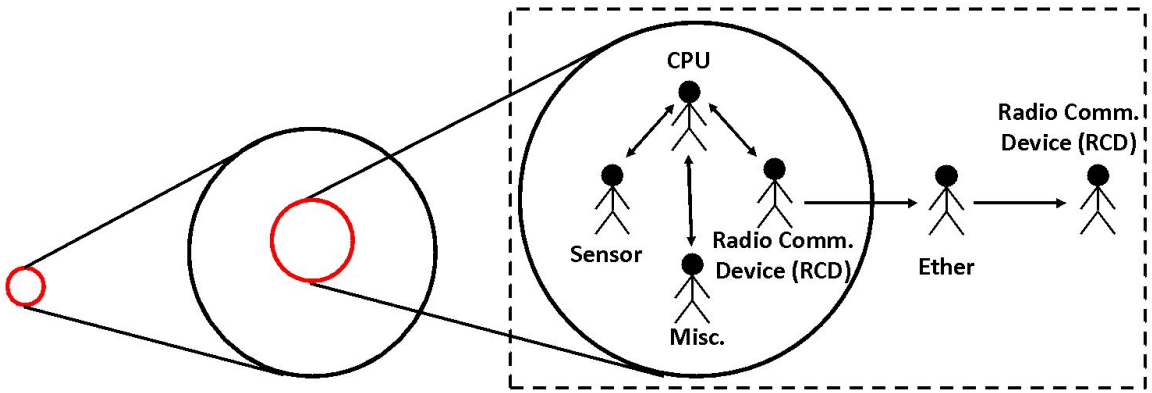
Schedulability Analysis of Distributed Real-Time Sensor Network Applications

(collaboration with OSL, UIUC, Gul Agha, and Ehsan Khamespanah, UT)

Smart Structures

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.

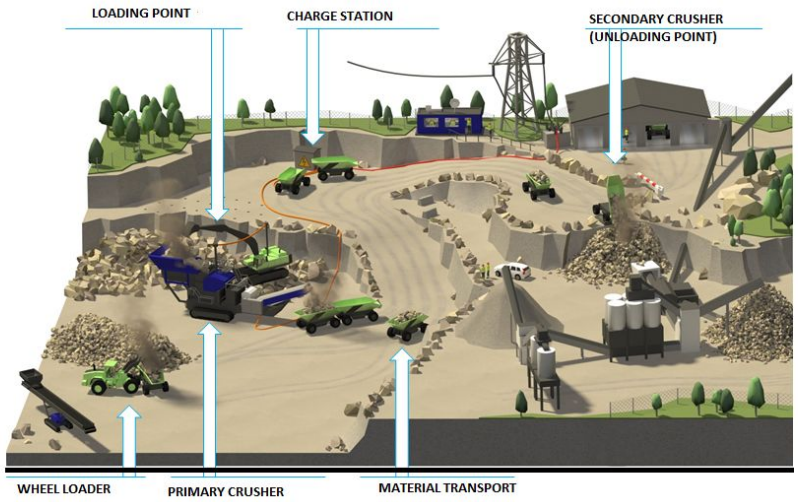


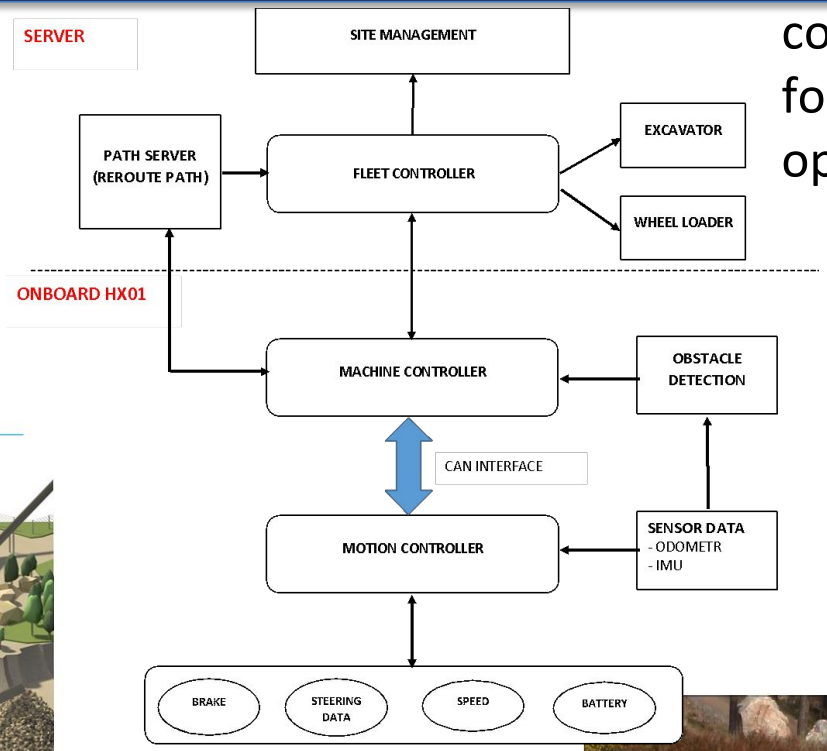
Automated Machines: ROS-based codes

(Volvo-CE, Stephan Baumgart and Torbjörn Martinsson; Sharif and UT, Movaghar, Kargahi)

✔ ROS based control system for autonomous operation

Derive the components and their computation and communication for Automated Hauler





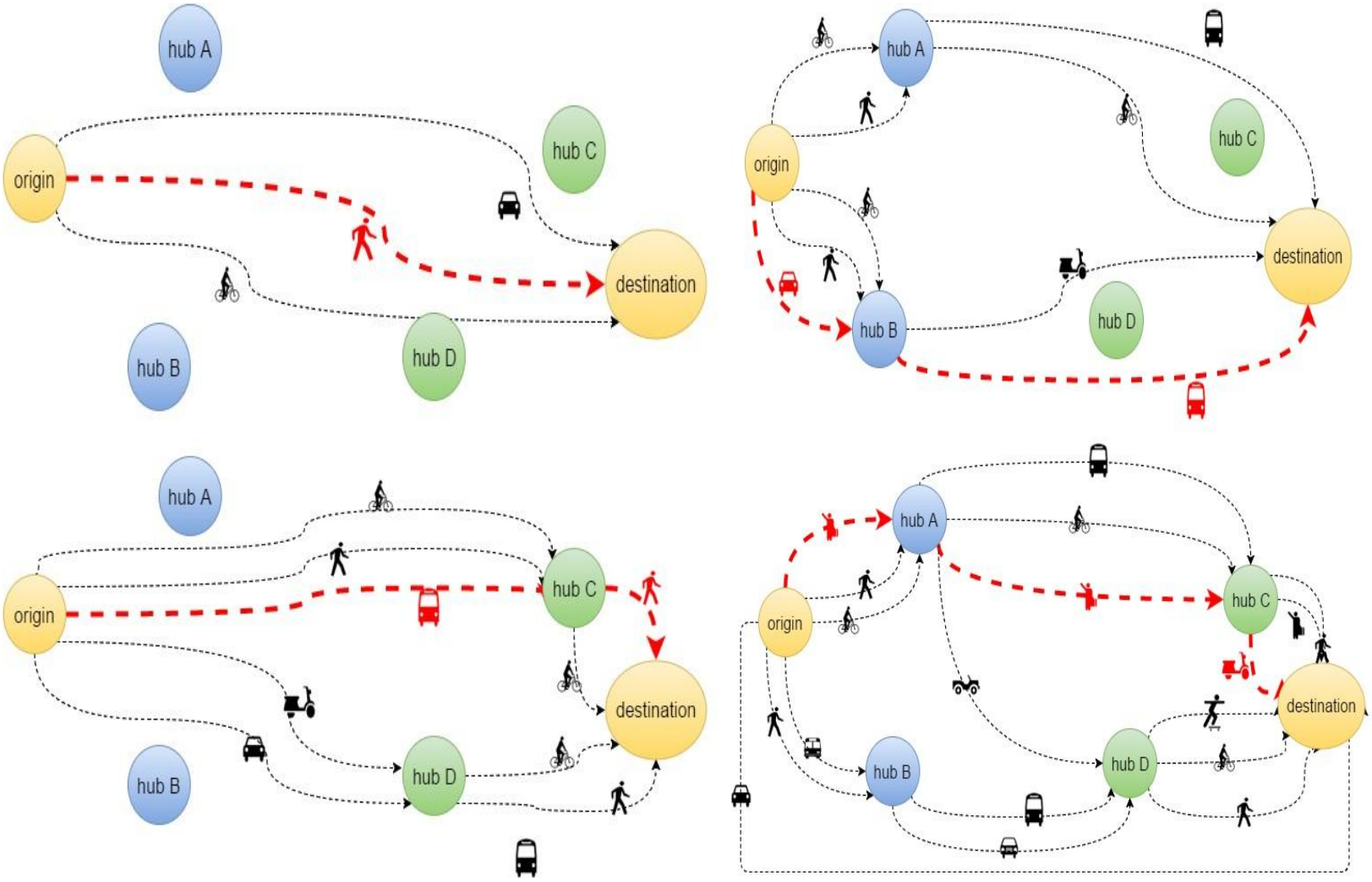
Check the schedulability and queue lengths by changing different parameters like sensing periods

SmartHub Project

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

- Smart Hubs are Local **container** of one or more smart mobility services in cities
- Offering new traveling opportunities to the surrounding urban population.

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

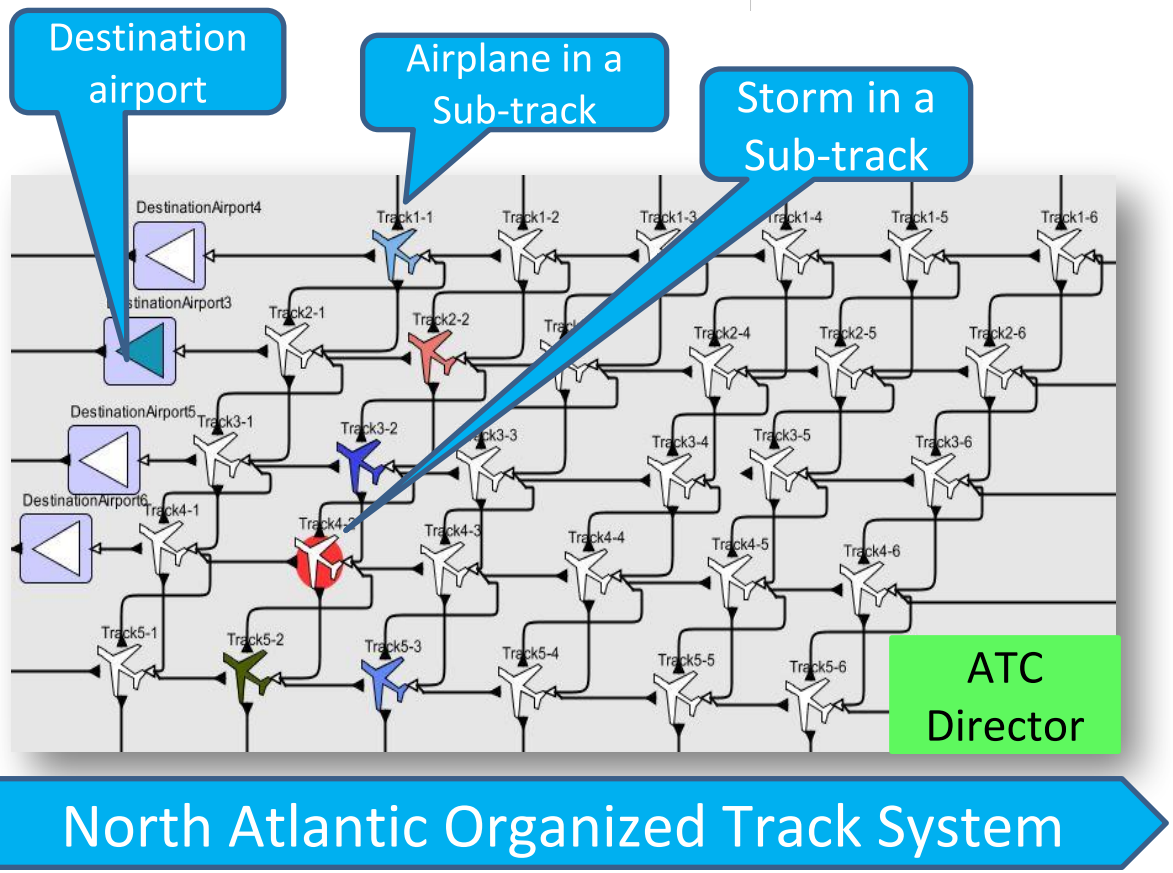


- Minimize:
  - number of service disruptions
  - number of mobility resources in smarthubs
  - cost of mobility for commuters
  - travel time for commuters
  - travel distance for commuters

Dependable Self-Adaptive Actors

(collaboration with UC Berkeley, Edward Lee and Sharif, Ali Movaghar)

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



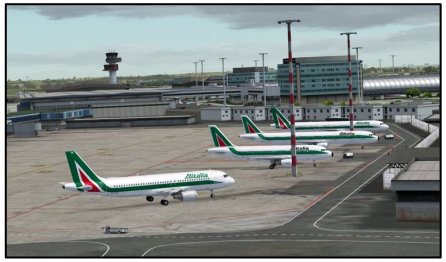
Flow Management of Track-based Applications

Warehouse Management System

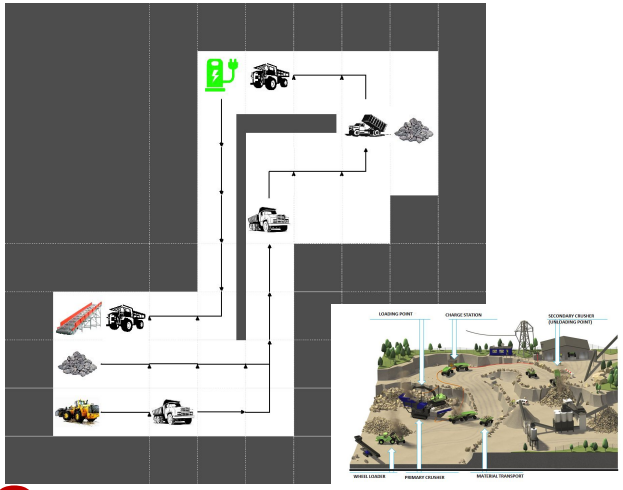
Public Transportation System

Transporting Shuttles in a Close Environment

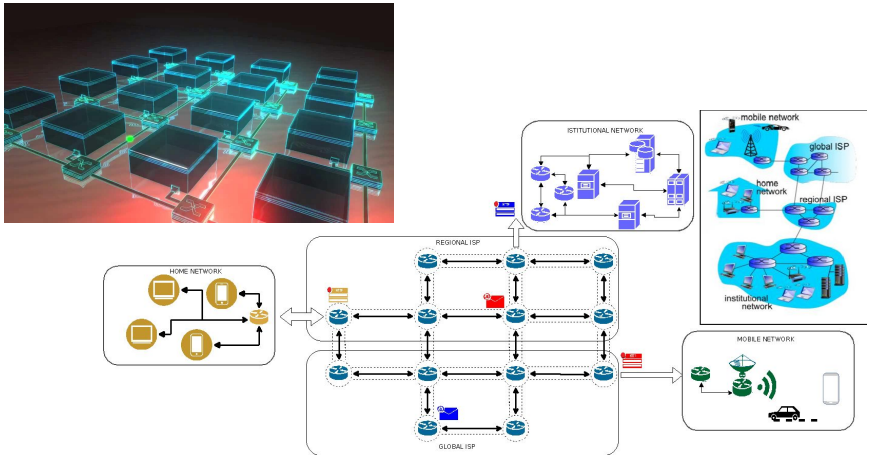
Airport and Airspace Systems



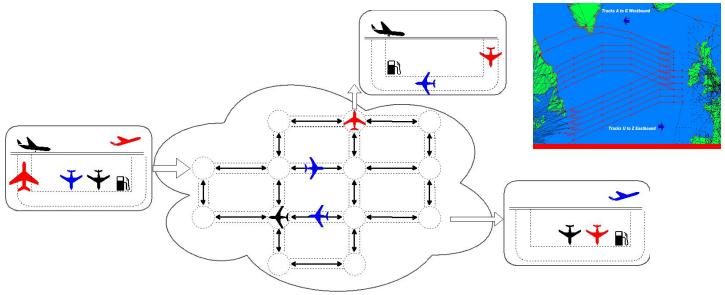
Flow Management: An Abstract View



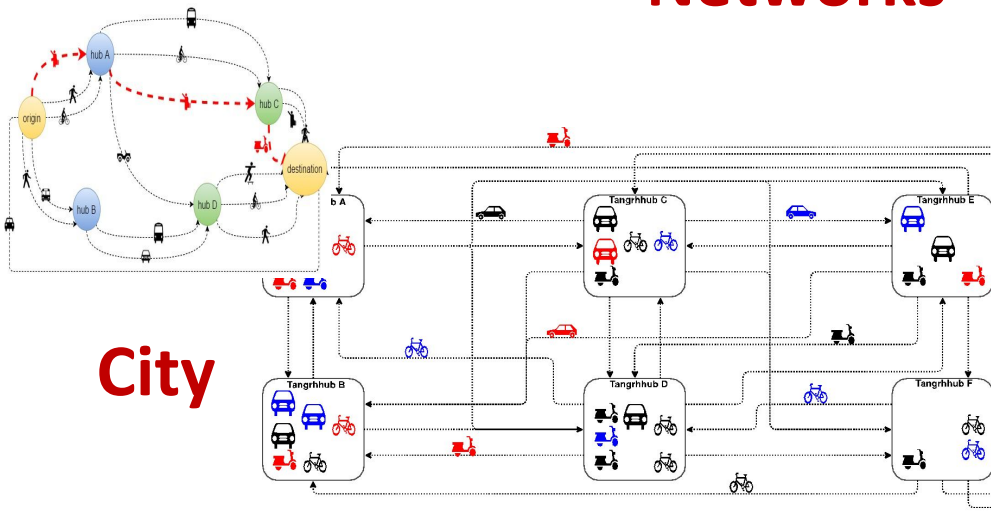
Quarry



Networks



Air Space



City

Similar Pattern: Flow of objects on tracks

Topology

- Sources
- Destinations
- Intermediate Destinations
  - Charging stations
  - Bus stations
  - Hubs

Configuration, design variables and constraints

- Capacity
  - Bandwidth
- Speed
- Latency / Time
- Cost

Goals

- Minimum Time
- Minimum Fuel
- Maximum Throughput
- …

Analysis

- Safety
- Optimization and Performance Analysis

- Self-Adaptation

Questions

- The “model” is really friendly, how can we continue using it for analysis in different domains?

- More in the domain of this meeting: what we need in designing the language?
  - For example why do we need Future?
  - https://ptolemy.berkeley.edu/publications/papers/19/LohstrohEtAl_Reactors_DAC_2019.pdf

References

- For publications, see

http://rebeca-lang.org/publications

- For projects, see

http://rebeca-lang.org/projects