TOWARDS TRUSTWORTHY INTERWORKING OF AUTONOMOUS AGENTS

Pierre Peloso, Laurent Ciavaglia, Samir Ghamri-Doudane – Alcatel Lucent Bell Labs
Kostas Tsakgaris, Panagiotis Demestichas – University of Piraeus Reserach Center
Mikhail Smirnov, Fraunhofer
Zwi Altman, Orange Labs

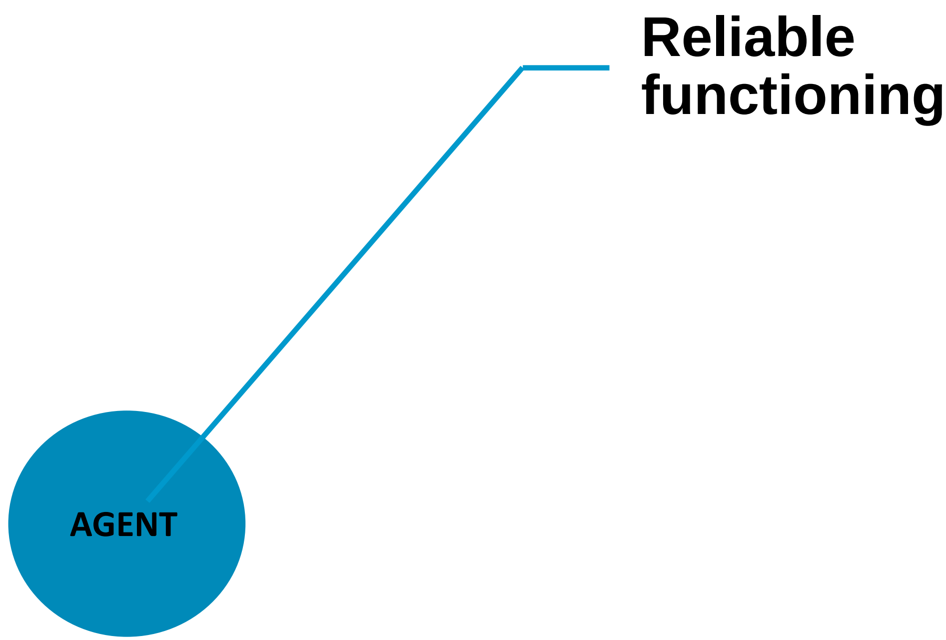
Autonomous agents: an arbitrary definition*

- When an agent keeps **its own decision-making process** it is referred to as autonomous.
  Such agents are able to **i) interact with their environments** and other agents beyond concurrent state-determined interaction with this environment and other agents, and **ii) autonomously adapt their decision** (and thus adapt and modify their behavior) on future action to changing environment, changing conditions (i.e., using sequence of k observations of k past system states, the next action does not solely depend on the previous state but also depends on the information stored in the agent's memory that can itself serve as input to some learning process).
- Learning agent: when such agents can additionally perform reasoning, e.g., learn beliefs about the environment, other agents, or the utility of performing some actions, they are referred to as learning agents.
- It is important however to remember that each agent is resource-constrained: both the memory and the processing capacity of each agent are finite, implying limited capabilities in terms of sensing, computation, and communication.

*Credits to D. Papadimitriou

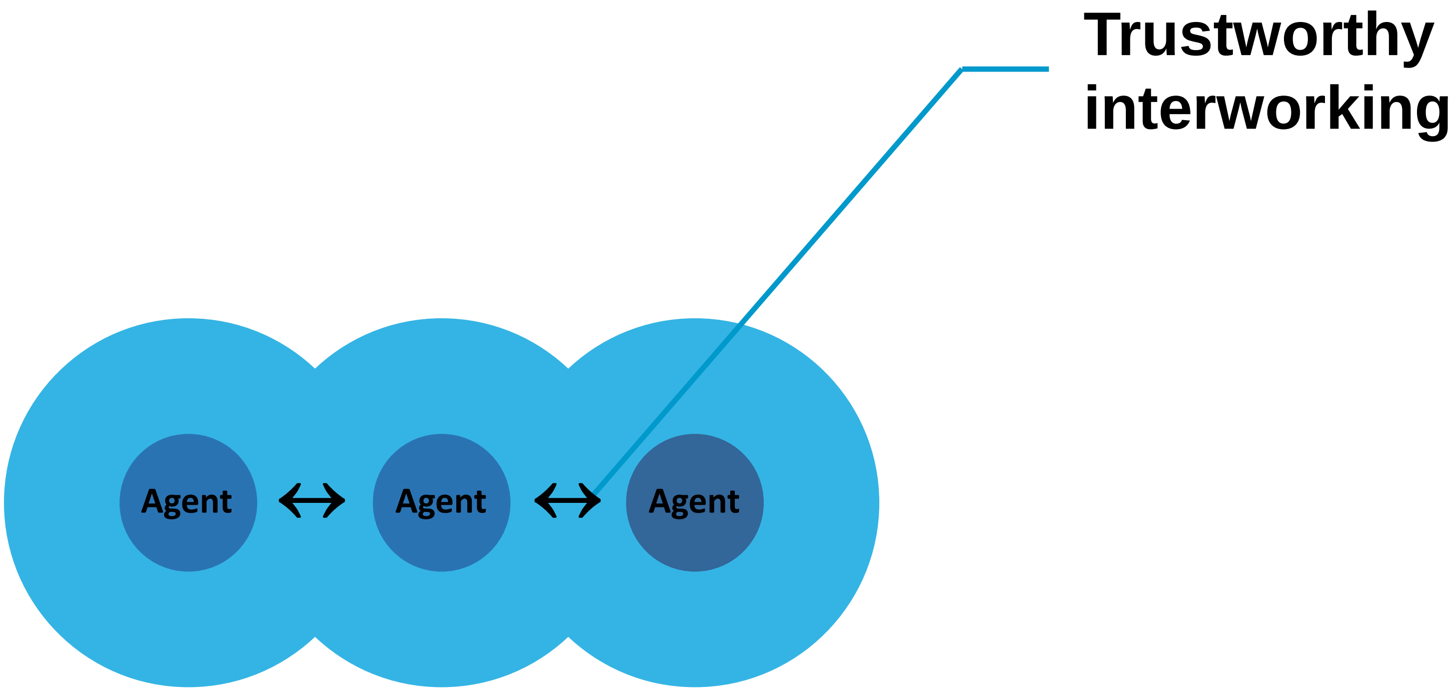
HETEROGENOUS DISTRIBUTED SYSTEMS
ANALOGY

#1 Assessing agent’s reliable functioning



Characterize-Test-Verify-Certify the performance and conformance of the function/system wrt. “referential” (e.g. canonical implementation, behavioral model, requirements, specifications, benchmark)

#2 Assessing agents trustworthy interworking



Coordination

Conflict maps: pre-defined/a priori ; dynamic/run-time

Schemes: static/fixed/dynamic ; centralized/distributed

EXAMPLE CONTEXT
RADIO ACCESS NETWORKS USING SELF ORGANIZING NETWORKS

CHALLENGE
RAPIDLY VARYING TRAFFIC

Users move around, download, hence the traffic is moving,
The user requirements depend on applications

SOLUTION
DYNAMICALLY ADAPTING NETWORK

Using SON functions to achieve the adaptation
These are autonomous functions optimizing
some of the network parameters

INDUCED ISSUES
CHAOS IN NETWORK DUE TO CONFLICTS

Caused by competition between
Autonomic behaviors

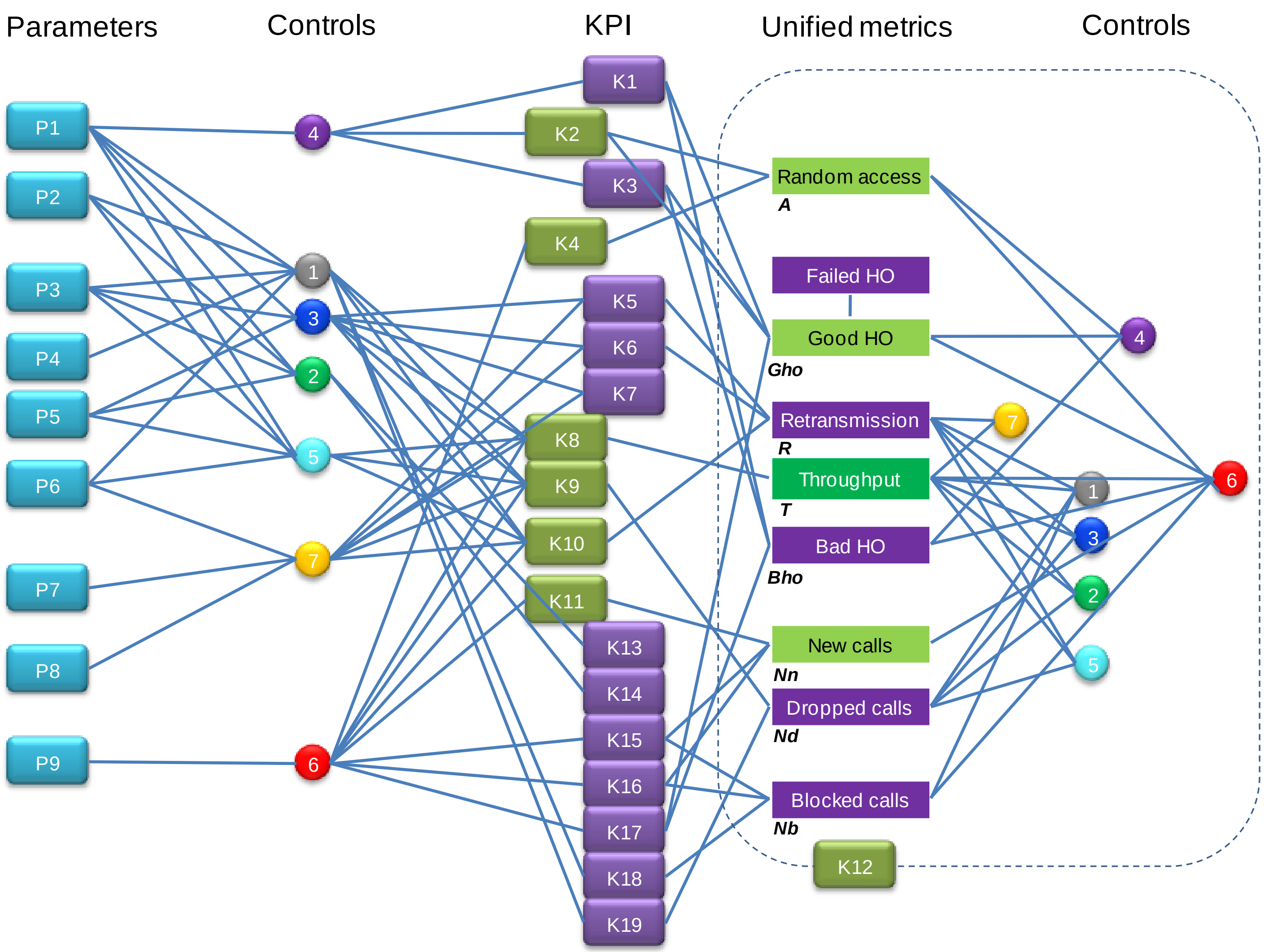
CHAOS?

TRUSTWORTHY COORDINATION
SPLITTING THE PROBLEM IN SMALLER CHALLENGES

- **Challenge 1:**
  - Identifying conflicts
    - Prior to deployment
    - Once deployed over a network
      - ✓ Before running
      - ✓ Once running
    - Challenge: Avoiding false positive though not missing true positive

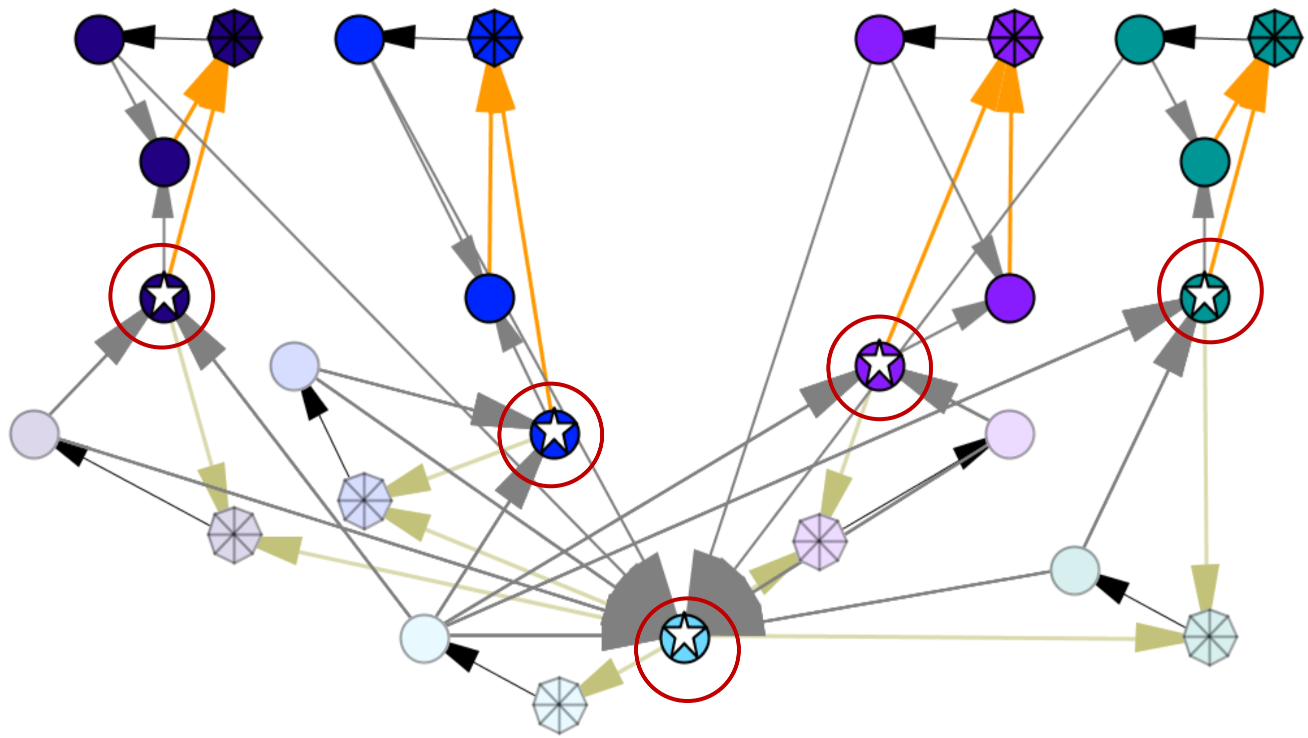
IDENTIFYING CONFLICTS
PRIOR TO DEPLOYMENT

Example of a static conflict map



IDENTIFYING CONFLICTS
AFTER DEPLOYMENT – BEFORE RUNNING

- **Using self-description from Autonomic Functions to**
  - Make inventory of metrics monitored, of actions performed and how they are computed
- **Build graphs showing control loops** (use knowledge for metrics influences)
- **Identify conflicting control loops**



Next 2 slides

THE KNOWLEDGE IS NEEDED TO IDENTIFY CONFLICTS

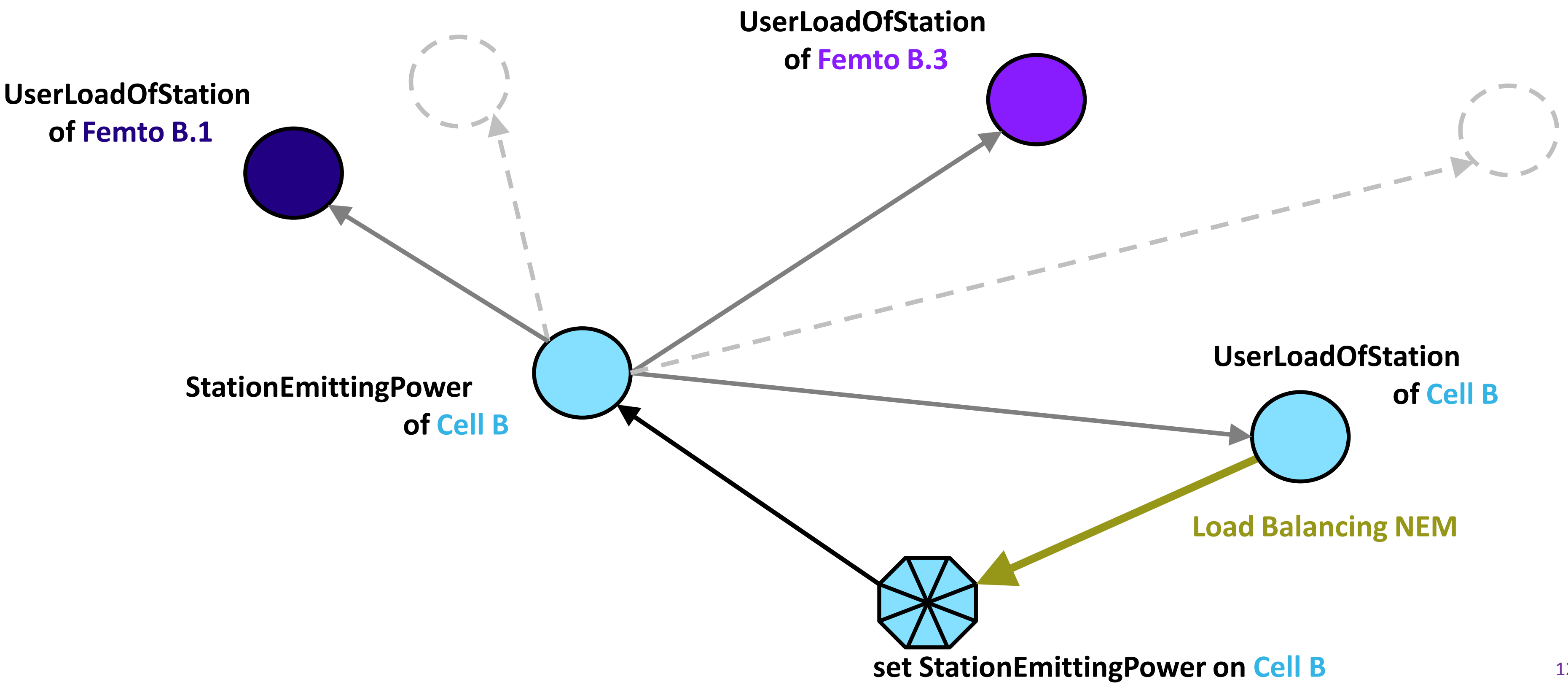
Knowledge-based Conflict Identification
Demystifying the conflict graph

$\text{StationEmittingPower}_{CellB} = \text{LoadBalancing}(\text{UserLoadOfStation}_{CellB},\ldots)$

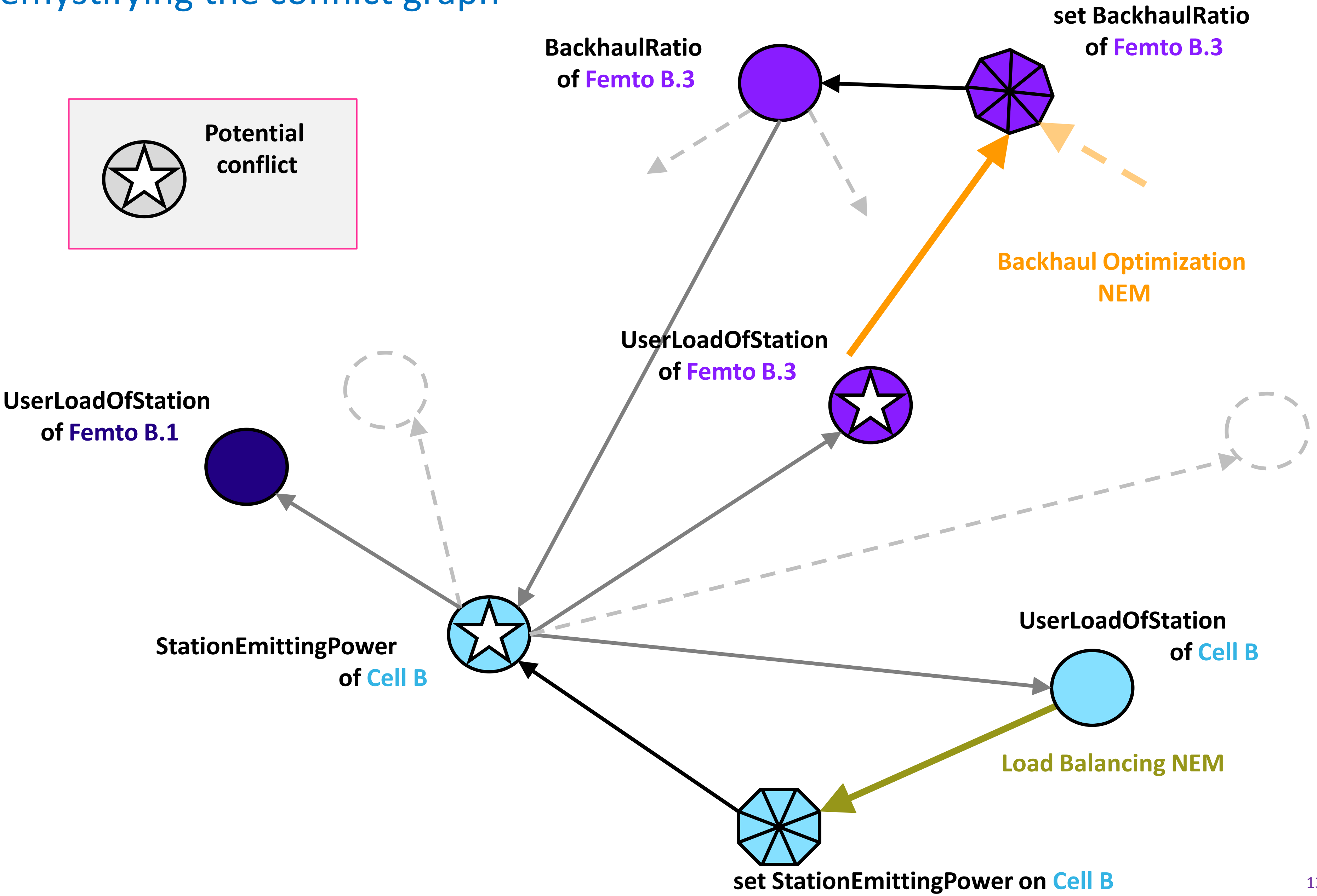
[... in order to perform action] *[takes as input...]*

$\text{UserLoadOfStation}_{FemtoB.3} = F(\text{StationEmittingPower}_{CellB,\ FemtoB.3\ldots\ldots},\ldots)$

[has impact on...]



Knowledge-based Conflict Identification
Demystifying the conflict graph



IDENTIFYING CONFLICTS
AFTER DEPLOYMENT – WHILE RUNNING

- **Either:**
  - Being able to find new dependencies between metric
  - Being able to find that autonomic functions are conflicting

- **The idea**
  - Use inference on monitored values to determine these dependencies

IDENTIFYING CONFLICTS CHALLENGES
SUMMARY

Prior to Deployment	**|**	**Requires all autonomic functions are known a priori. Does not take into account network.**
Before run	**|**	**Requires an exhaustive knowledge to determine metric dependencies while making discrimination.**
During run	**|**	**Waits for conflicts to appear before addressing them. Can it scale?**

- **Challenge 2:**
  - Applying coordination mechanisms
    - Choosing between available mechanisms
    - Affecting conflicts to mechanisms
    - Setting configuration parameters to such mechanism

Listing various coordination mechanisms

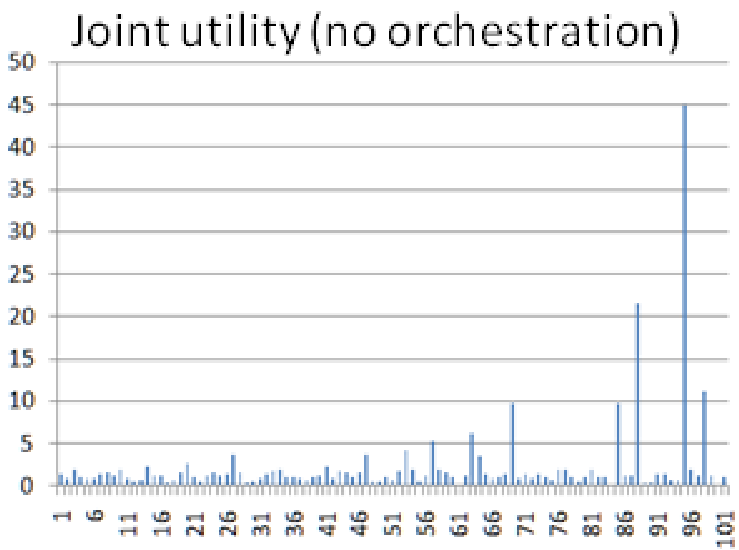
- Self-orchestration of multiple agents
- Hierarchical optimization
  - Time separation
- Centralized multi-objective optimization
- Control theory approaches

Self-Orchestration of multiple Agents

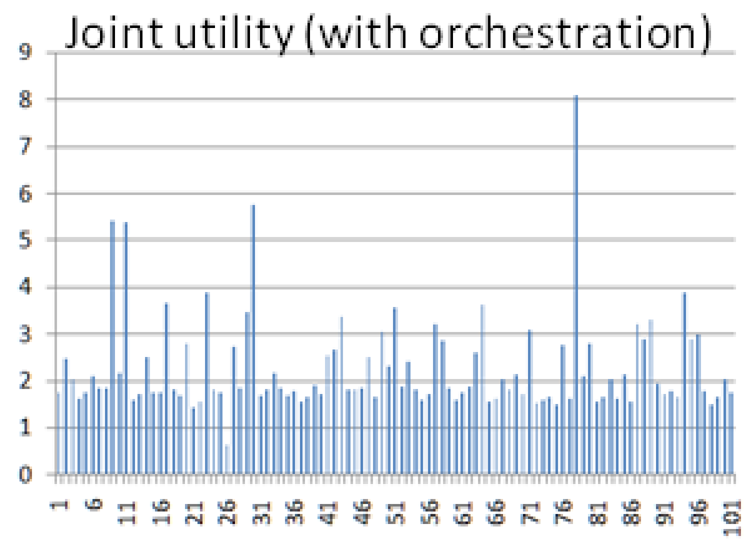
Principle:

- each NEM provides an estimated utility for the next time slot
- an orchestrator gives the token to the NEM with the highest utility

Examples:



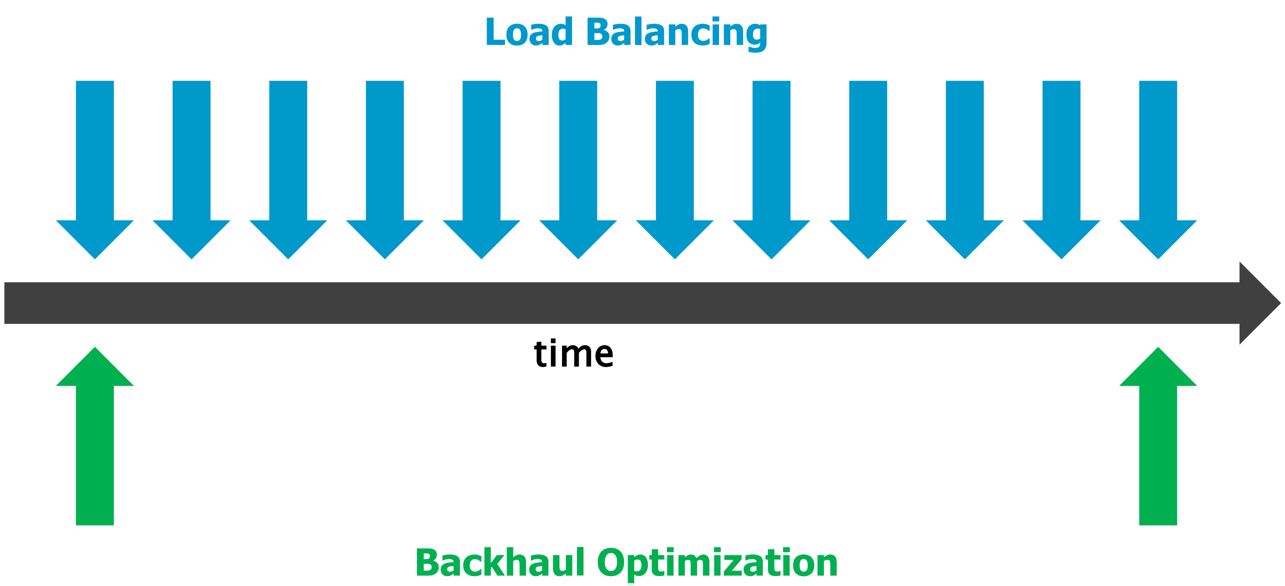
(a)

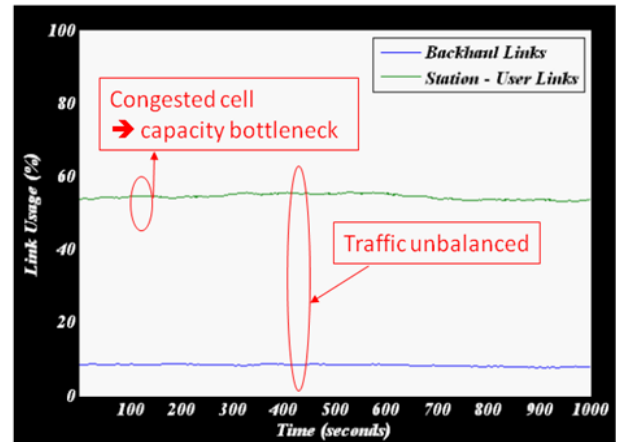


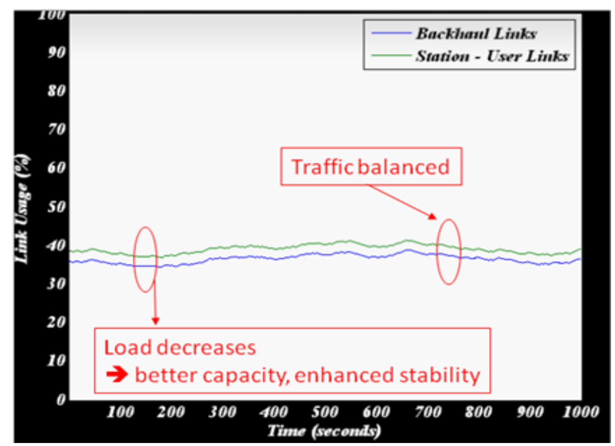
(b)

Hierarchical optimization (a family of algorithms)

- **Principle:**
  - One control loop as precedence over the other
- **One example – Time separation**
  - Principle: Each Agent is executing its control loop at different paces



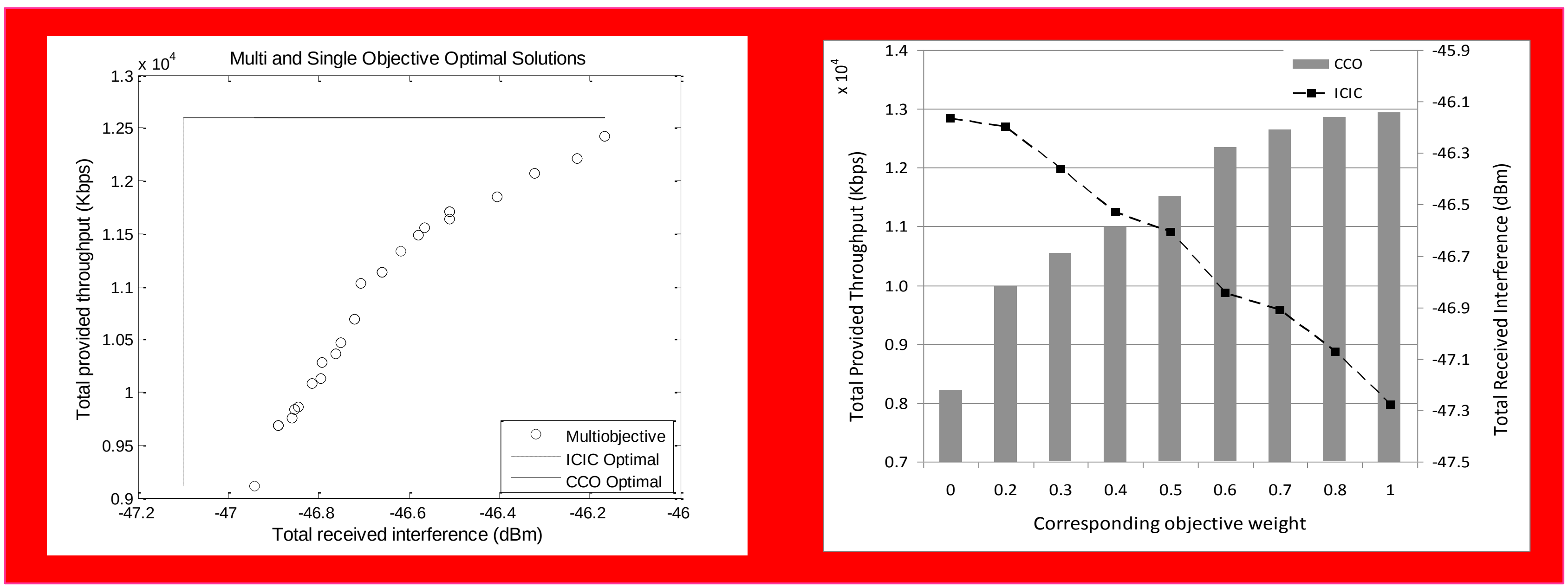




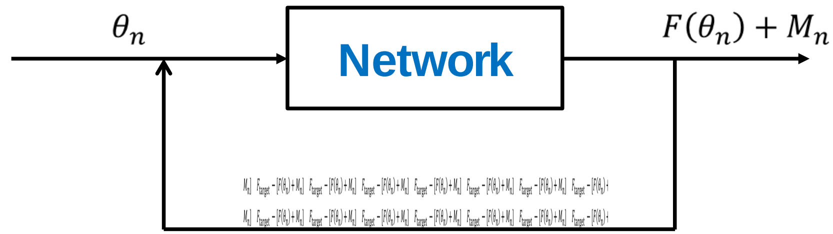
Centralized multi-objective optimization

- **Principle**
  - aggregation of the weighted utilities of each agent
  - ➔ Pareto optimal solutions are sought

$$U_{total} = \omega_{ICIC} U_{ICIC} + \omega_{CCO} + U_{cco}$$

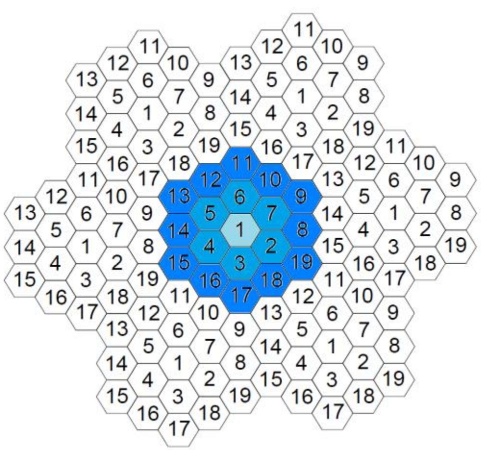


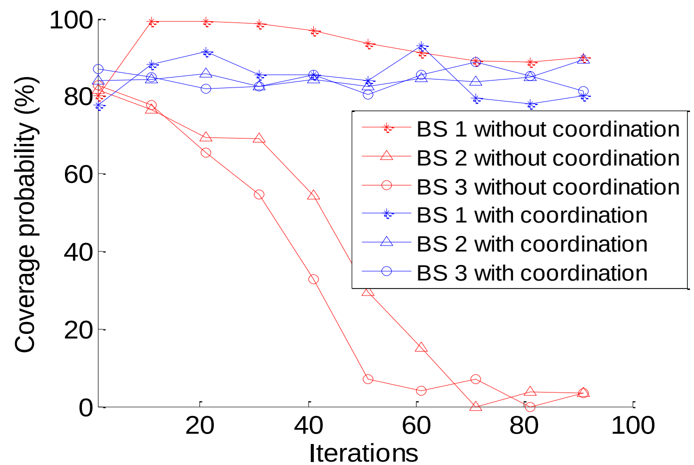
Use of control theory

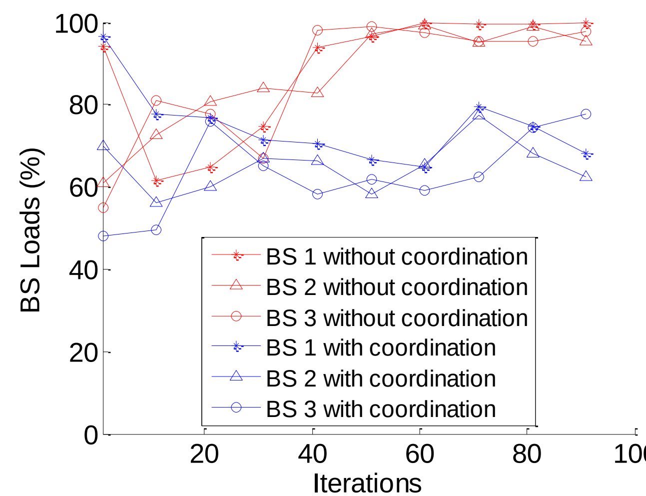


Principle: Weighting each loop

$$\frac{\partial}{\partial t}\theta(t) = F(\theta(t))$$	control loop
$$\theta_{n+1} = \theta_n + \epsilon(F(\theta_n) + M_n)$$	discretized control loop = SON
	a set of control loops
	coordinated control loops





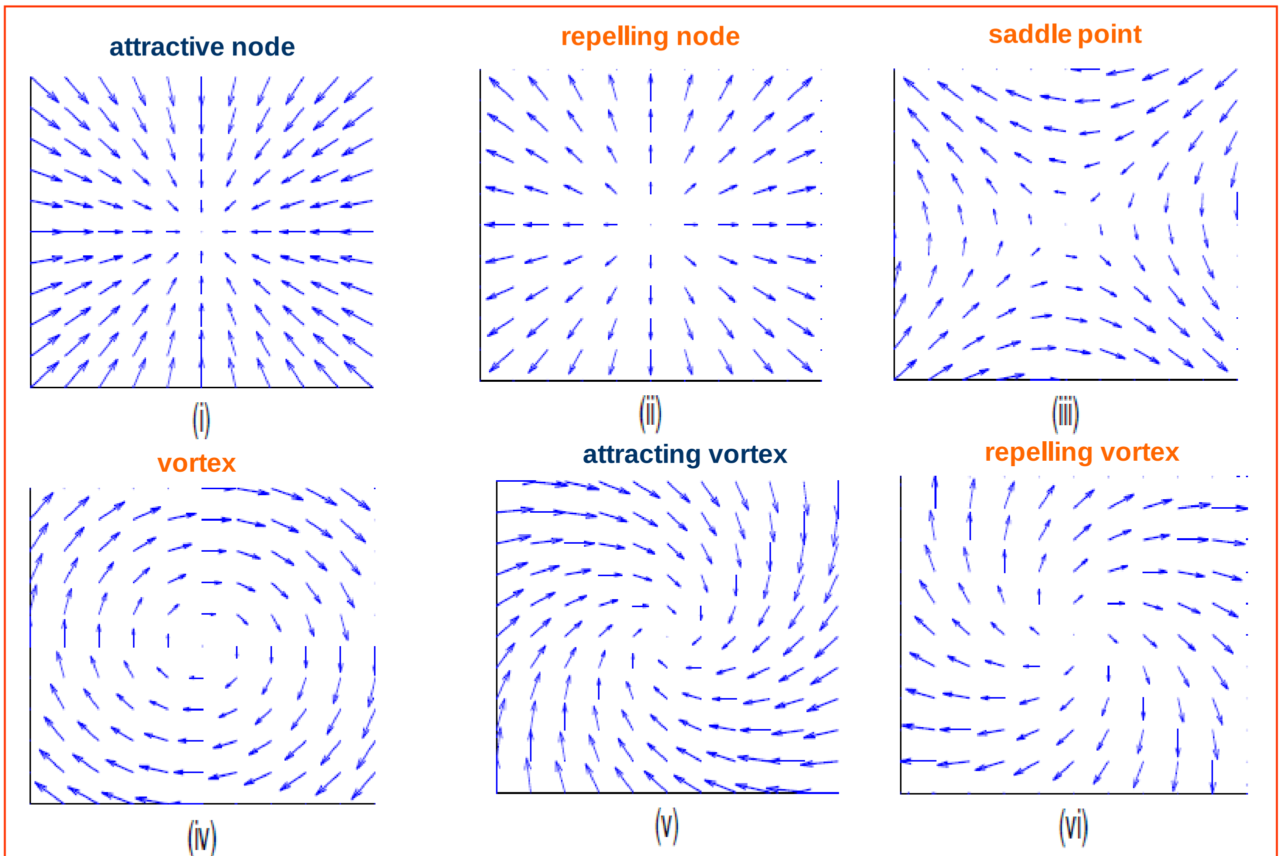


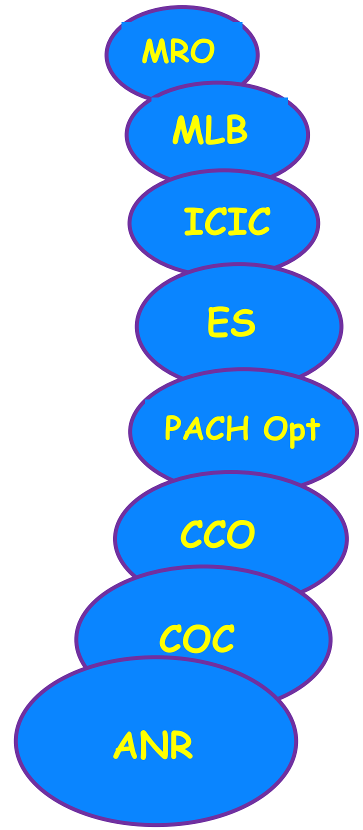
APPENDIX

Introduction (1/2): Why coordination

- **SON / NEMs are control loops**
  - Control theory tells us that when putting together 2 independent control loops, one can get different behaviors (solutions)

$$\dot{x} = Ax \text{ with } x \in \mathbb{R}^2 \text{ and } A \in \mathbb{R}^{2\times 2}$$





Introduction (2/2): SON Coordination in 3GPP

- **3GPP/SA5 specifies high-level solution for avoiding conflicts between SONs**
  - The SON coordination is a logical function, that can be implemented as a separate entity or as part of a SON function
  - Coordination is achieved using specific policies that can be of the form of
    - weights / priorities / specific actions
  - Policies are defined for a set of use cases, e.g.
    - **COC-ES** (Cell Outage Compensation, Energy Saving)
    - **COC-CCO-ES** (CCO – Coverage Capacity Optimization)
    - **HOO-LBO** (Handover Optimization – Load Balancing Optimization)
      - ✓ "Policy may assign higher priority for HOO function than LBO function or higher weight for target of HOO function than targets of LBO function when resolving MRO issues …"

3GPP TS 28.628, "Self-organizing networks (SON) policy network resource model (NRM) integration reference point (IRP) information service (IS)" June 2013