

MobiDIS - A Pervasive Architecture for Emergency Management

Massimiliano de Leoni

Fabio De Rosa

Massimo Mecella

*Univ. Roma LA SAPIENZA, Italy
Dipartimento di Informatica e Sistemistica (DIS)*

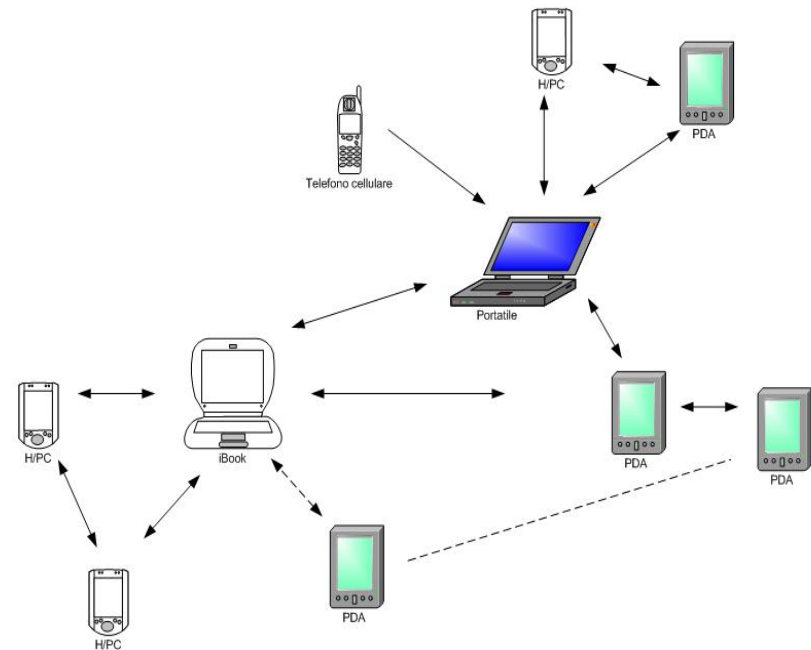
`{deleoni, derosa, mecella}@dis.uniroma1.it`

Overview

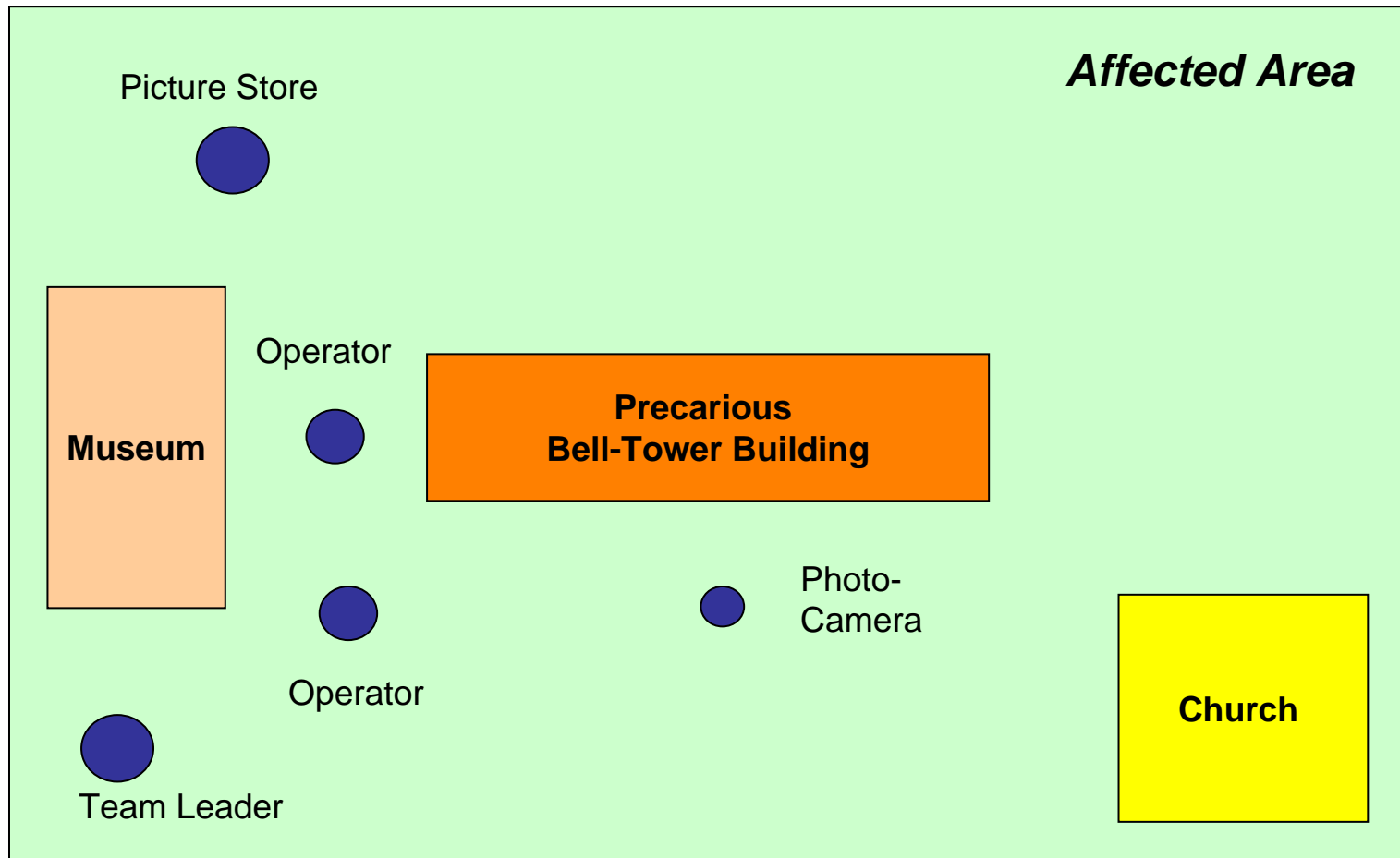
- Introduction
 - MANETs for Emergency Management
 - Process Management
- The Adaptiveness Issue
- The Architecture
- Outline of Prediction and Bridging Techniques
- The Process Restructuring Technique
- Realization and Preliminary Validation
- Conclusions and Future Work

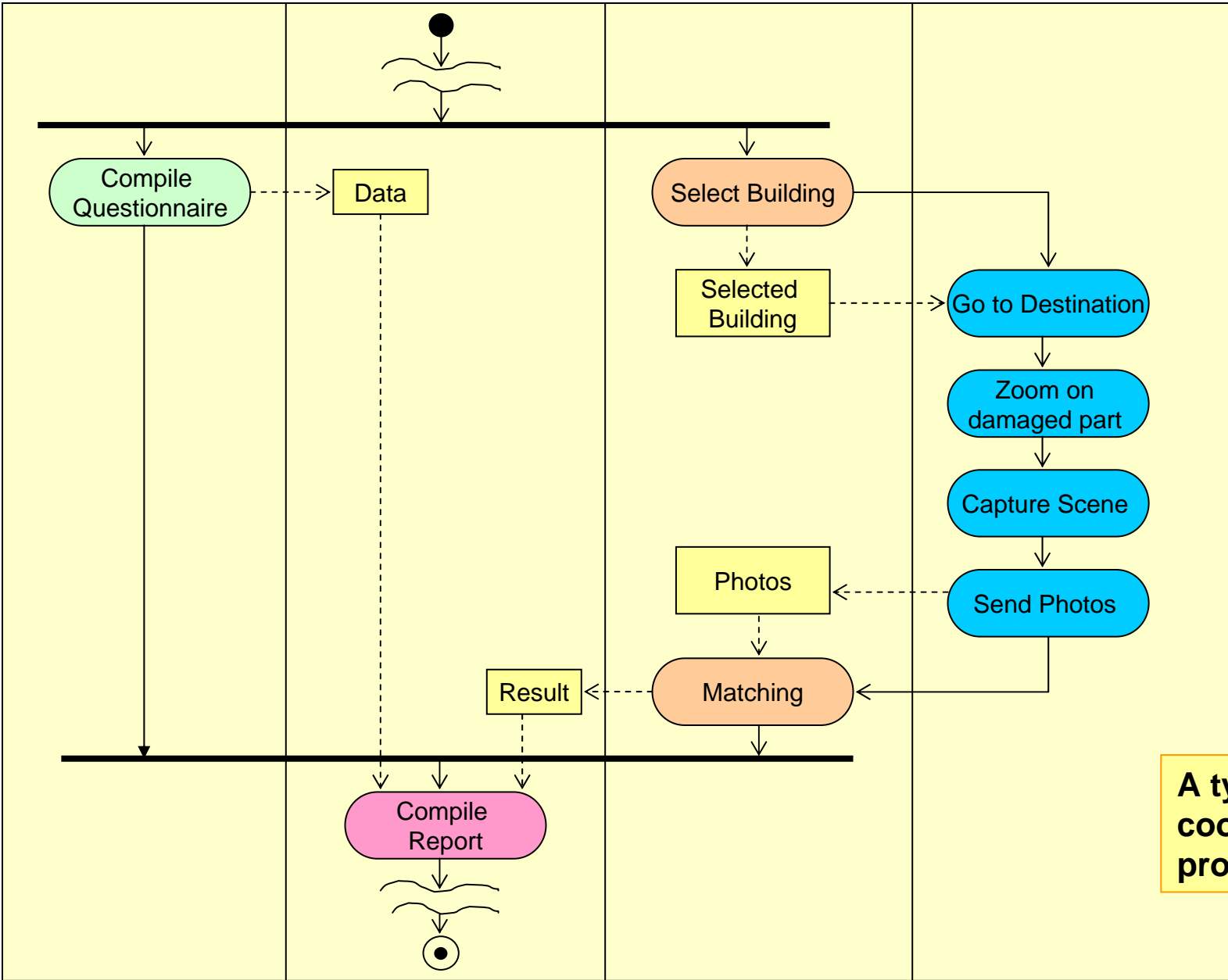
Mobile Ad-hoc NETWORKS: MANETs

- Networks of **Mobile Devices** (i.e. PDAs, laptops), where each mobile unit communicates with another ones via **wireless link**
- Devices are free to move
- **There is not an underlying infrastructure**
- Peer-to-Peer system architecture
- Very appropriate in emergency management



MANET Scenario in Emergencies





A typical cooperative process

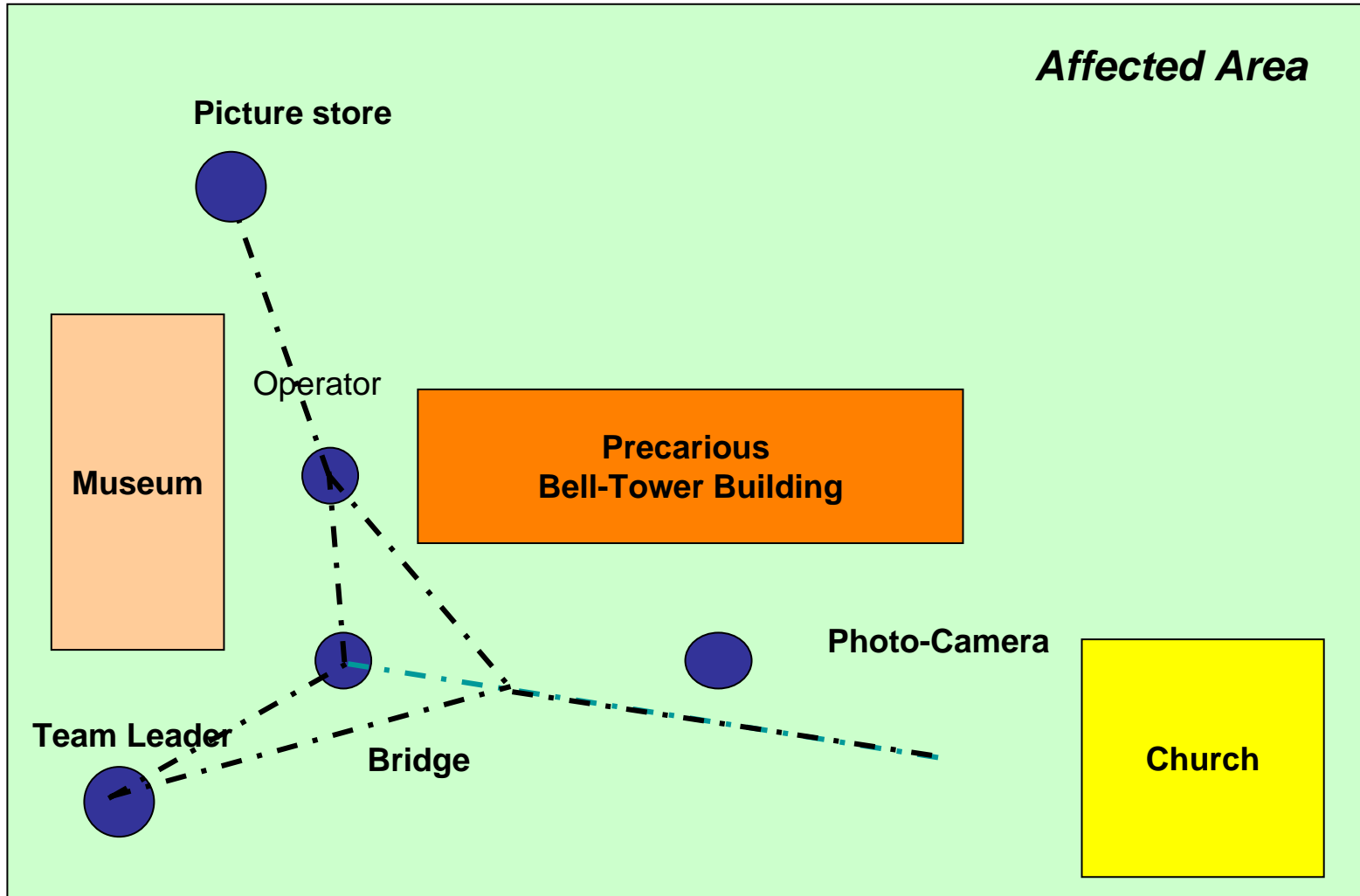
Team Member 1

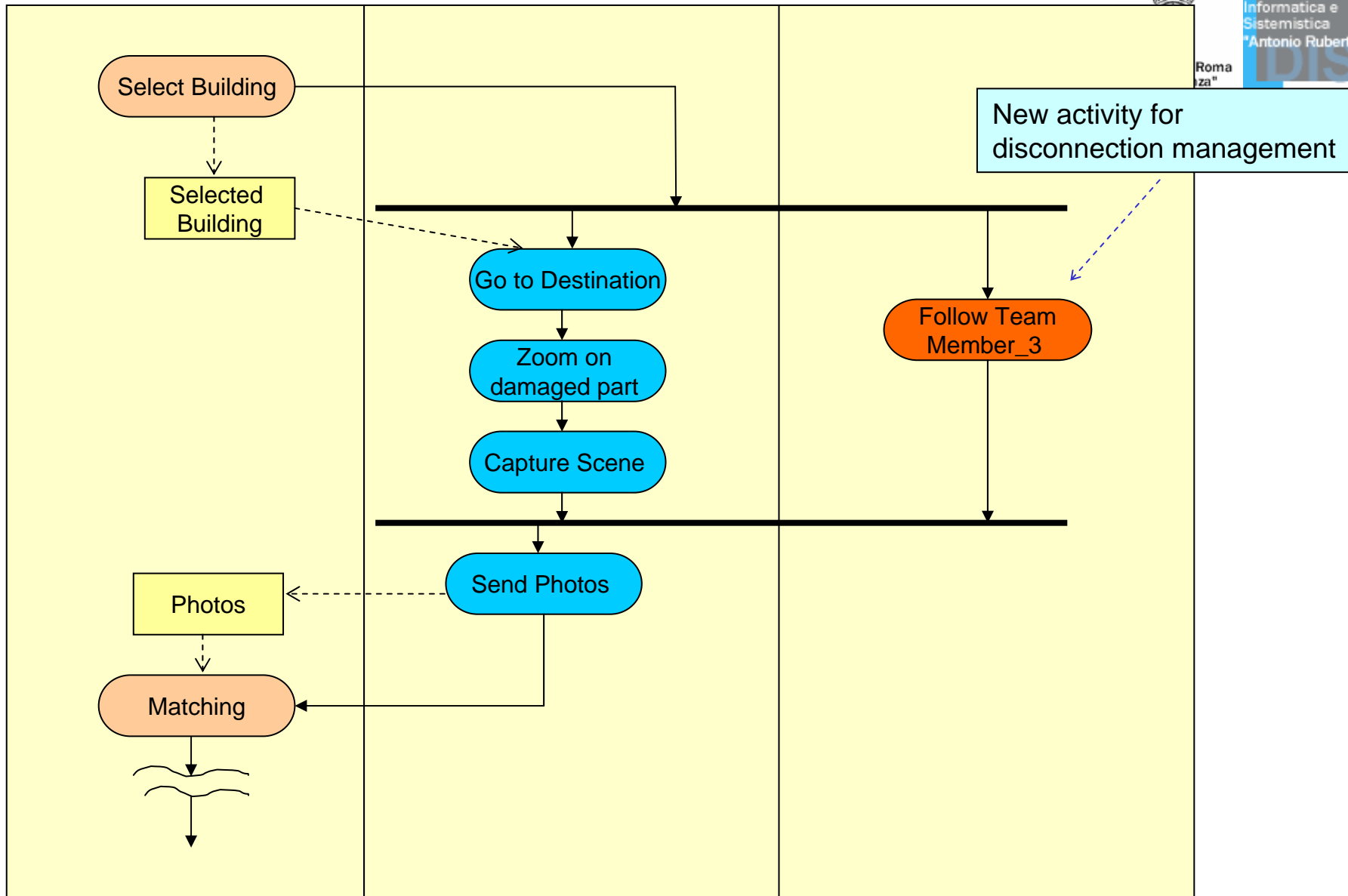
Team Leader

Team Member 2
(picture store device)

Team Member 3
(camera device)

Adaptive Process Management





Team Member 2
(picture store device)

Team Member 3 (Photo-
camera device)

Team Member 4
(bridge device)

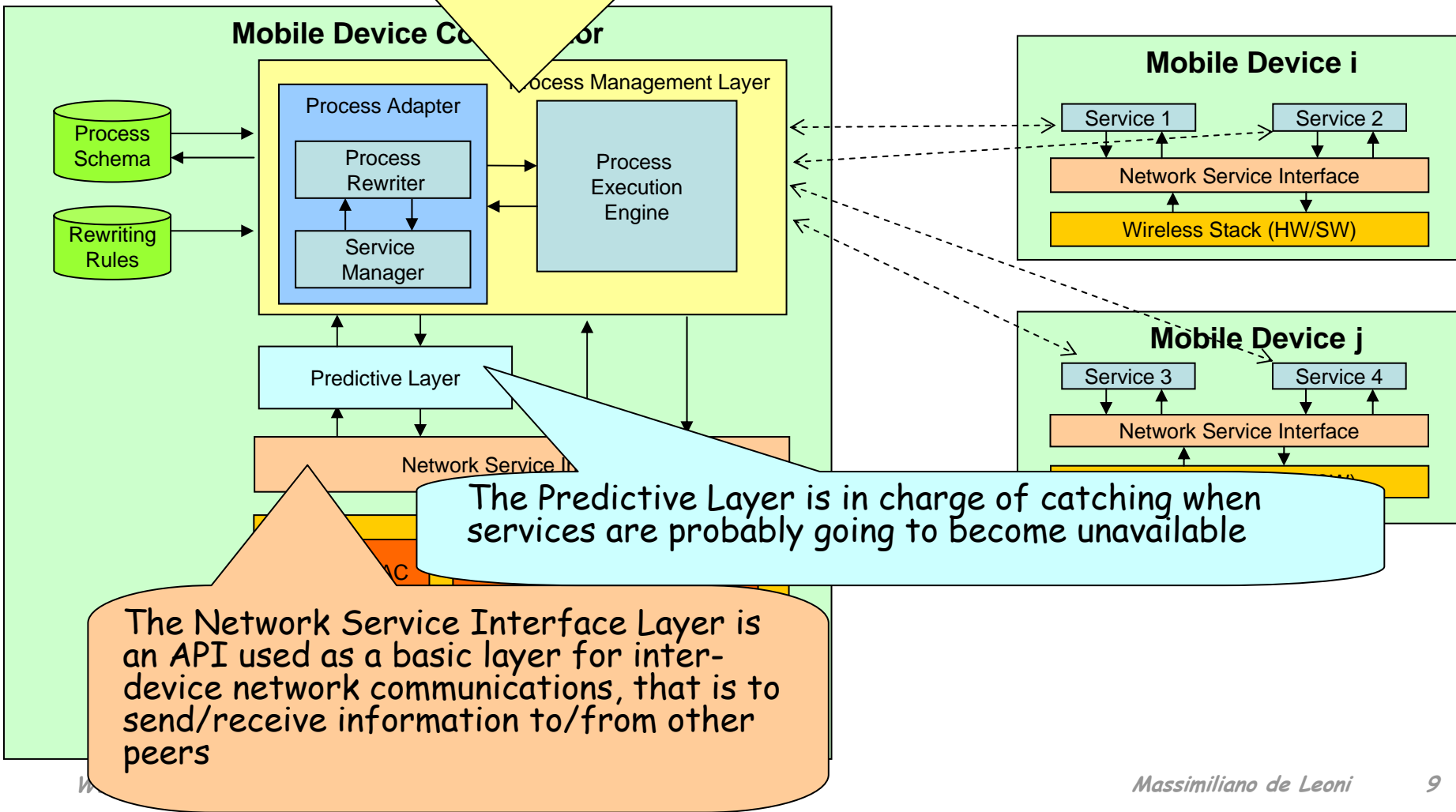
Contributions

- Definition of a **pervasive architecture** for supporting adaptive process management
- Definition of models & algorithms for
 - Managing (predicting) **service disconnection events**
 - **Service bridge choosing**
 - **Process restructuring** (assuring the correct changes)



The Process Management Layer carries out process instances and adapts them when services are going to become unavailable, by using transformation rules

DIS



The Predictive Layer is in charge of catching when services are probably going to become unavailable

The Network Service Interface Layer is an API used as a basic layer for inter-device network communications, that is to send/receive information to/from other peers

Assumptions

- Each device holds hardware that lets it know its communication distance from the surrounding devices within its radio range. Specific techniques are available to that goal, such as: TDOA, SRN, Cricket compass, GPS hardware
- At start-up all devices are connected, that is each device has a path to any other device
- A specific device, called the coordinator, centrally predicts disconnections and manages them by adapting the workflow schema and by a reassignment of process tasks among the participants

MOBIDIS Approach

- MOBIDIS approach combines local connection management with global management of both network topology and task assignment
- Local connection management consists of monitoring and checking one-hop communication between a device and its neighbours. It is realized as special services running on held-hand devices that implement techniques for estimating and calculating distances
- Global management maintains a consistent state of the network and of each peer in the network. It manages network topology and tasks each peer is in charge of, and services that peers offers as well (that is, it provides a service registry). On the basis of that information, the coordinator applies algorithms for choosing bridge nodes and/or executes reassignment of the workflow tasks

Predictive Layer in MANET

- In MANET scenarios the **device disconnections** generate **service unavailability**. The technique has to predict which devices are going to disconnect and to send “probable service unavailability” to upper layer
- Periodically, each device sends to the coordinator a message containing the distances to other devices in radio-range
- At given time t_i , in which all devices are connected, the coordinator collects all the distance information from the other devices
- The coordinator builds a probable *next connection graph*, and using the graph, the coordination layer enacts appropriate actions in the interval $[t_i, t_{i+1}]$

Prediction Technique

- Reasonable assumption
 - If two devices tend to go out of range if not controlled but are connected through the coordinator's remedial actions, this influences the next probability of going out of range
- Consider a time frame of $h > 0$ time units as the history of distances between devices i, j
 - Predicted distance between i and j at the next time unit as

$$S_{P(i,j)}^{(t+1)} = \frac{\sum_{k=1}^b \alpha_k S_{i,j}^{t-(b-k)}}{\sum_{k=1}^b \alpha_k} = \sum_{k=1}^b \left(\frac{\alpha_k}{c} \right) S_{i,j}^{t-(b-k)}$$

with $\alpha_k = k$ and

$$c = \sum_{k=1}^b \alpha_k$$



Prediction Technique

- The estimated probability of devices (i,j) still being in range at $t + 1$ is:

$$P_{P(i,j)}^{(t+1)} = \begin{cases} \frac{|S_{\text{dev}} - S_{P(i,j)}^{(t+1)}|}{S_{\text{dev}}} & S_{P(i,j)}^{(t+1)} \leq S_{\text{dev}} \\ 0 & S_{P(i,j)}^{(t+1)} > S_{\text{dev}} \end{cases}$$



The MGR Algorithm

- The Mobile Gambler's Ruin (MGR) algorithm

- $M = |E| \times |E|$

- $|E| = m$, the number of MANET mobile devices.

- M is an $m \times m$ symmetric matrix

- $m_{ij} = P_{(i,j)}^{(t+1)}$

- diagonal elements

$$P_{P(i,i)}^{(t+1)} = m_{ii} = 1$$

```
PROGRAM MGR(Comps[m])
1  numcomps ← 0
2  for i ← 0 to (m - 1)
3    do if Comps[i] = 0
4      then numcomps ← numcomps + 1
5      CCDFSG(M, i, numcomps, Comps[])
6  return Comps[]
```

```
SUB CCDFSG(M, i, numcomps, Comps[m])
1  Comps[i] ← numcomps
2  for each  $M[i, j] \geq \text{Beta}$ 
3    do if Comps[j] = 0
4      then CCDFSG(M, j, numcomps, Comps[])
5  return NIL
```

```
PROGRAM TEST_CONNECTION(i, j, Comps[m])
1  if Comps[i] = Comps[j]
2    then TEST ← true
3    else TEST ← false
4  return TEST
```

- The MGR algorithm finds the connected components graph and verifies if two devices belong to the same connected component

Experimental Results

- *Obstacle Mobility Model*
 - Provides a mechanism for modeling movement in real-world environments.
- *Simulation area*
 - $1000\mu \times 1000\mu$, where $\mu = (1/100)S_{dev}$
 - The mobility of nodes is randomly selected between 0 and 5 m/s to represent walking speeds.
 - At creating, nodes are randomly distributed but loosely connected.
 - Random obstacle
- *Simulator*
 - NS and GloMoSim
 - 5-device set and 10-device set, with 7 randomly placed obstacles
 - 2000s time frame and performed 50 experiments obtaining between 50,000 and 100,000 samples per experiment.
 - Time frame h of 5, 10, 15.

Experimental Results

TABLE 1

The predicted distances' average error, for three time frames h .

No. of devices	Network simulator	$h = 5$	Avg. error $h = 10$	$h = 15$
5	NS	2.43 μ	1.62 μ	0.64 μ
	GloMoSim	1.94 μ	1.01 μ	0.51 μ
10	NS	2.19 μ	2.04 μ	1.03 μ
	GloMoSim	2.54 μ	1.67 μ	0.76 μ

- Worst case
 - 2.5 μ (means that MGR has a precision bound of 97.5% in predicting distances between mobile devices)
- Best case
 - 0.5 μ (means that MGR has a precision bound of 99.5%)
 - In this case, coordinator must devote more space to maintain all the previous S.
- F. De Rosa, M. Mecella and A. Malizia "Disconnection Prediction in Mobile Ad Hoc Networks for Supporting Cooperative Work", *IEEE Pervasive Computing*, Vol. 4, N. 3, 2005, pp. 62-70.

Process Management Layer

- The coordination layer is a process manager that handles and executes processes
- A *process definition* (process schema) is made up of a set of *tasks*, atomic piece of work, to be carried out according to a specified *scheduling* (i.e., *routing* or *control flow*), and undertaken by *roles*, that is, services offered by network devices

Adaptive Process Management

- **Problem:** if a service is going to disconnect during the process execution, the process itself could be not terminated leading to stalling situations
- Special services running on devices are used to carry out **supporting tasks** for primary ones (e.g., the task "Follow the team member 3" is a supporting task for the primary one "Send the photos")
- The **bridging service** is used to restore an unavailable service.
- Supporting tasks are added to primary ones. Such changes cause the process restructuring

Process Model

- A directed graph is used as process model
- Each graph node represents a particular basic process construct; graphs represents well-structured processes
- Graph nodes are:
 - Tasks: elementary process
 - Parallel routing: AND-split control flow construct
 - Selective routing: XOR-split control flow construct
 - N-Join: OR-Join and AND-Join control flow constructs
 - Loops: structures' cycles, that is cycle with only one entry point to the loop and one exit point from the loop and they cannot be interleaved
- Two special tasks, named Start and End Task, resp., exist for start and end of the process

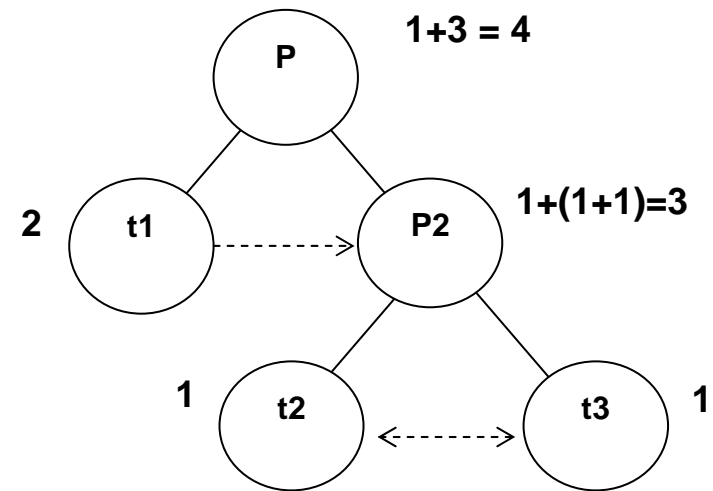
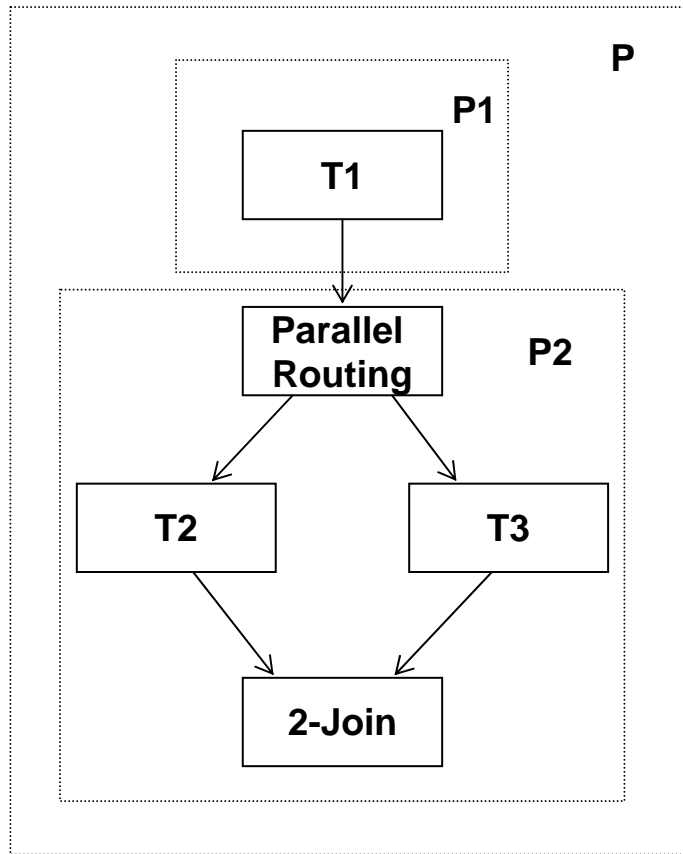
Run-time Information

- In addition to design-time information, deriving from the process structure, the adaptive coordination layer needs additional information taken at run-time
 - the set of services joined in every instant to the network
 - the set of tasks each service can perform
 - the state of every task forming the process instance

Partial Order Relationship

- **Problem:** Restructuring a process could require to keep order among tasks
- A priority algorithm is used for building a partial order relationship $(T, <)$ among process tasks
- The algorithm uses the process control flow structures and assigns weights (priorities) to each process task through a tree structure, named **process tree**, representing the process. The basic ideas are the following:
 - The priority is described by integer numbers
 - A task t has higher priority than the task t' if and only if the numeric value of t is greater than one associated to t'
 - It assigns higher priority to those tasks whose executions generate the achievement of *enabled* state for a greater number of tasks
 - The priority assignment guarantees the constraints inducted by control flow structure: **If the task t comes before the task t' in the process then the priority assigned to t is greater than one assigned to t' (i.e., we have $t' < t$)**
 - The starting process is well-structured

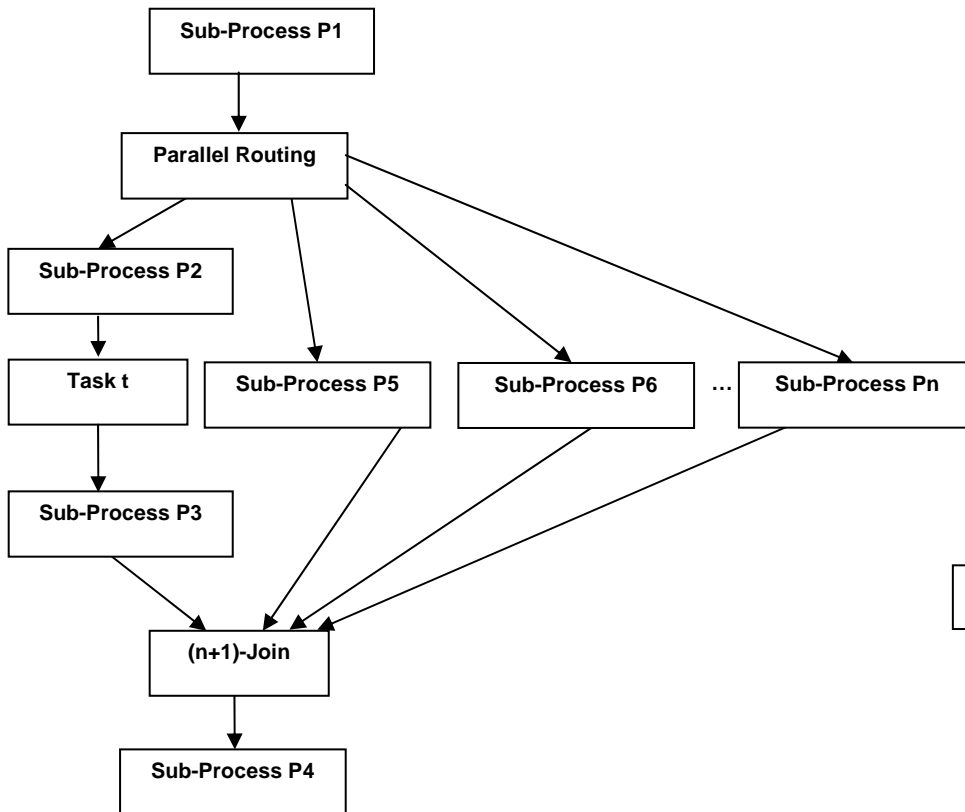
Process Tree Example



(b) Corresponding subprocess tree together with their weights

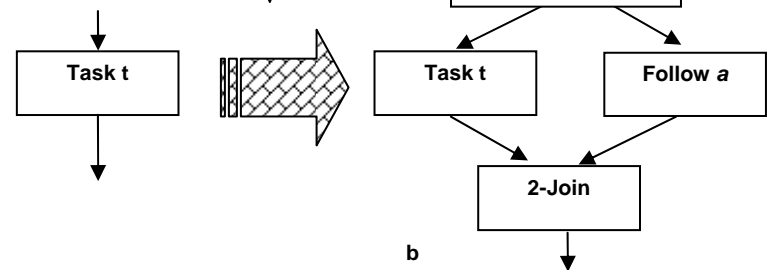
(a) Subprocess with sequence and Parallel constructs

Process Restructuring (Rule 1)



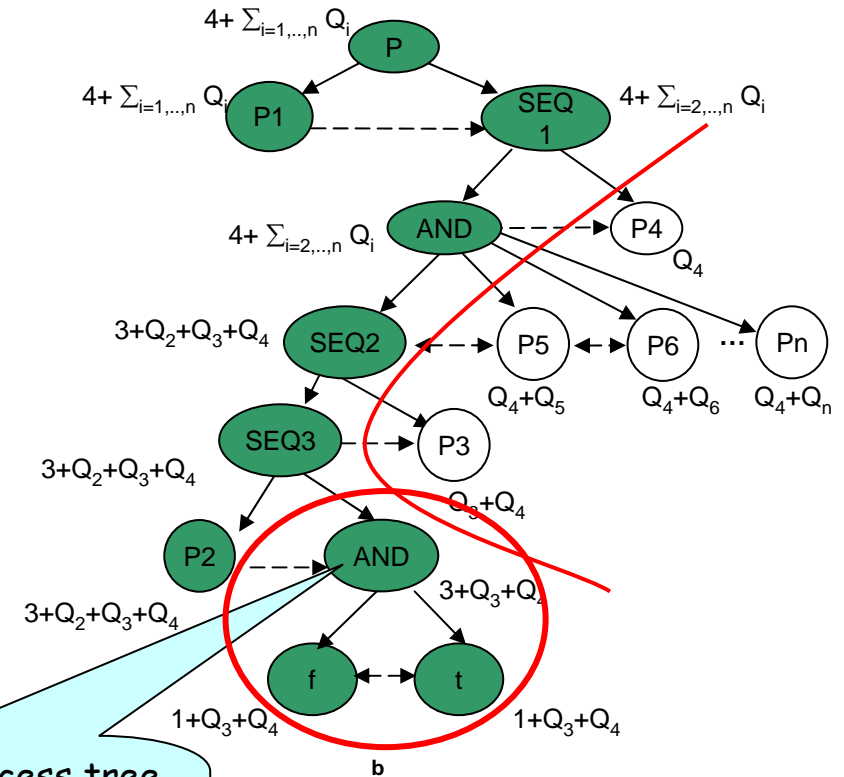
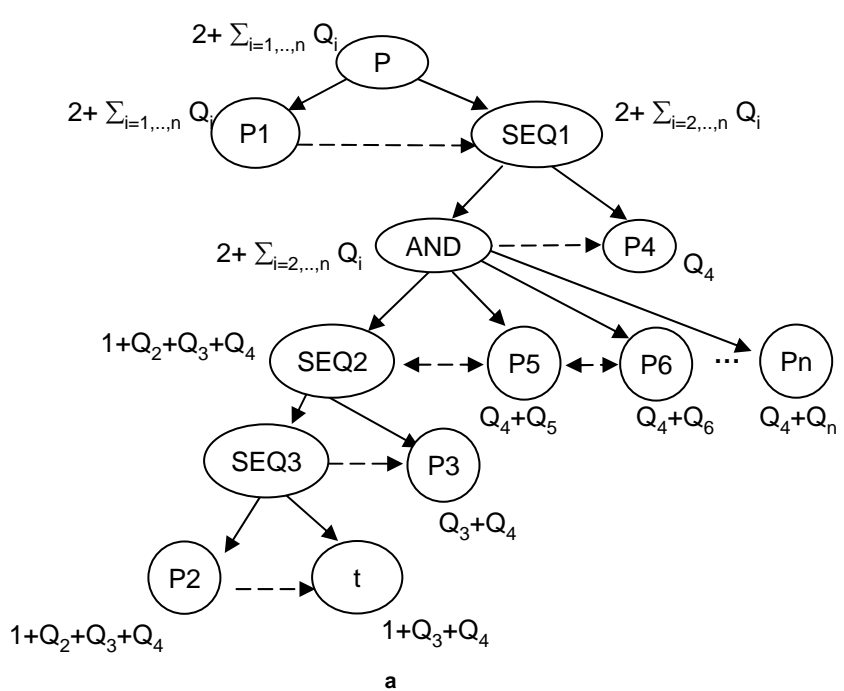
a

Case when the process management layer chooses an idle device (that is, actually it is executing no task). It is added supporting task in parallel



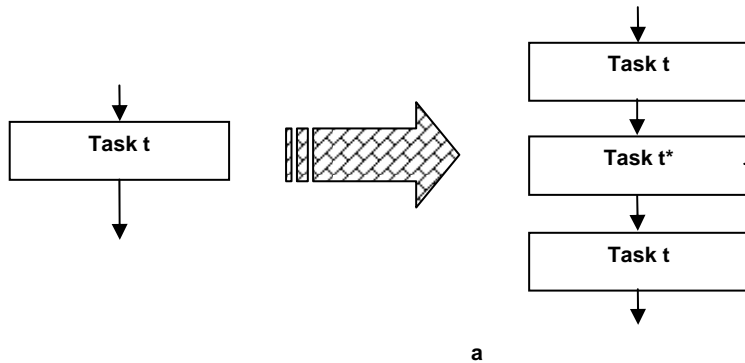
b

Changes within the Process Tree

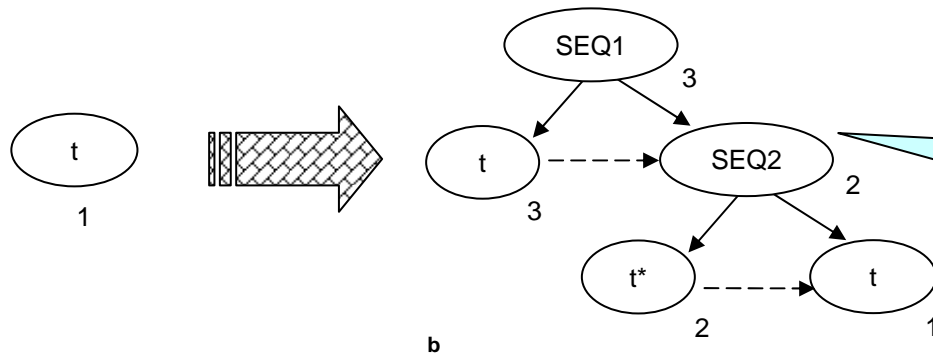


The changing in the process tree structure by applying the rule 1. Only completed task priorities are affected

Process Restructuring (Rule 2)

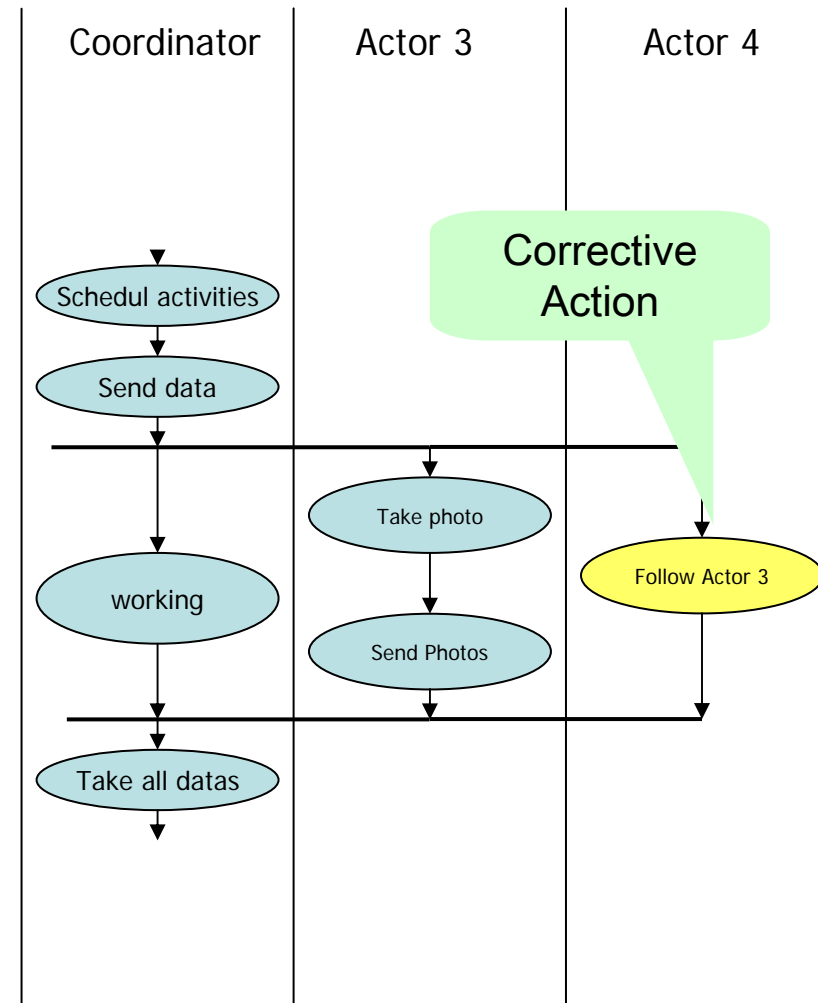
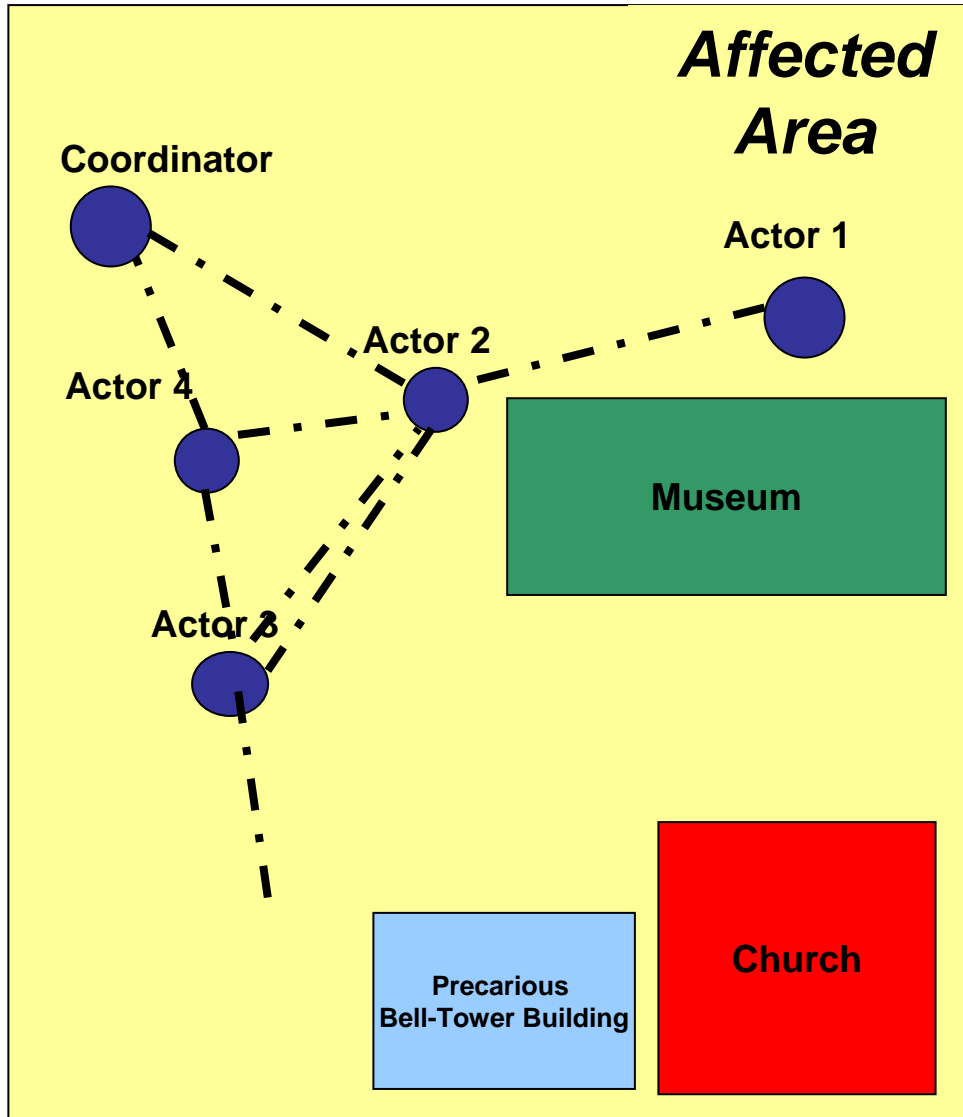


Case when the Process Management layer chooses a bridge is carrying out a task t . It introduces the corresponding recovery task t' to make the undo of t , and postpones the task t in the process



The changing in the process tree structure

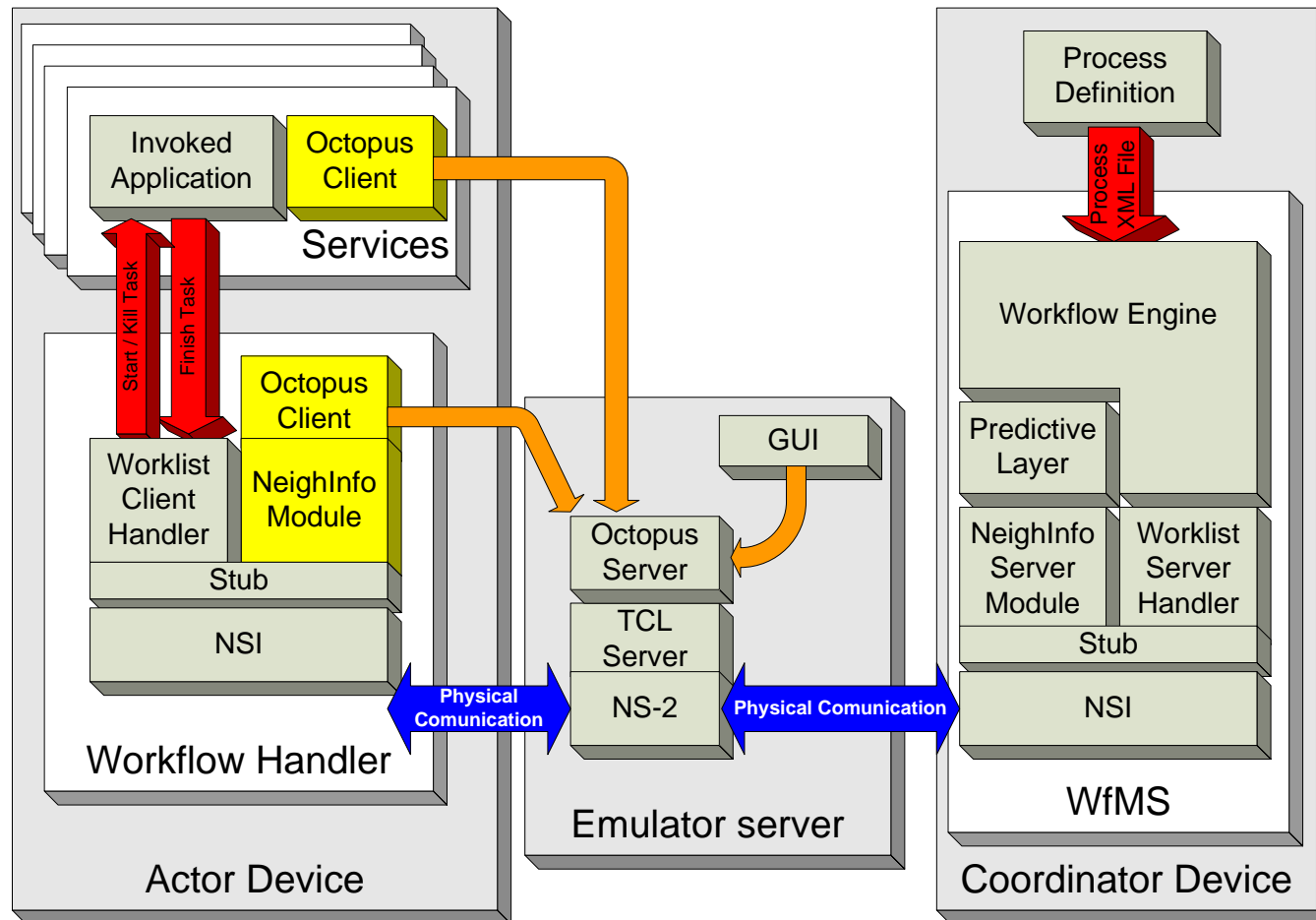
Supporting Task. The Bridging Service



Choosing the Bridge

- Based on the following criteria:
 - The nearest idle neighbour is preferred.
 - If each neighbour is carrying out a task, then the one performing the lowest priority task is chosen.
 - With same priority values, the preferred bridge is one having the smallest number of neighbours. The bridge role likely leads to movement of the node and this might cause new disconnections. By selection of a node with the lowest number of neighbours, the probability of new disconnections is minimized.
 - With same number of neighbours, it is preferred the nearest one.

Implementation and Validation Architecture

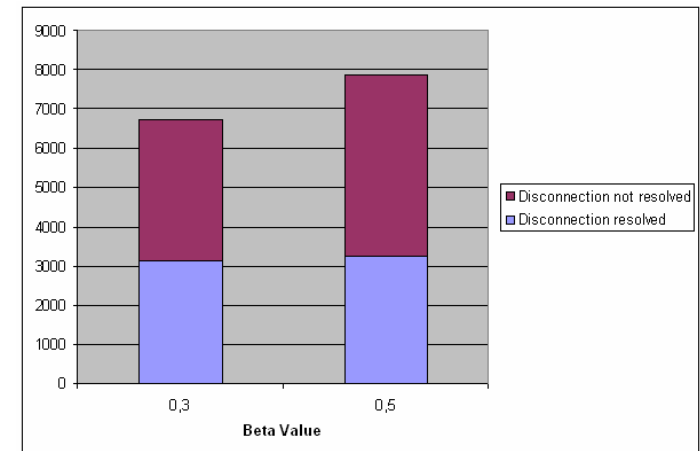


F. D'Aprano, M. de Leoni, F. De Rosa, M. Mecella: "MOBIDIS: A Pervasive Architecture for Emergency Management" (On-line appendix)

Preliminary Experimental Results / 1

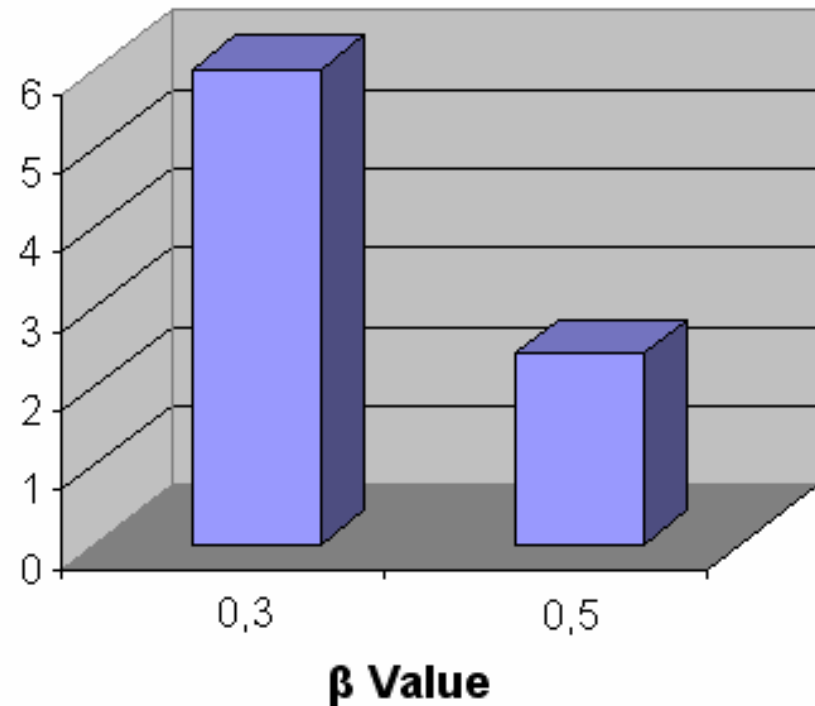
- Tuning parameters
 - *Beta*: The fraction of radio-range where possible disconnections are not signalled
 - *The Polling Time*: The shortest time between two corrective actions
- After tuning, deep experiments were done.
 - We went on in experiments with Polling time = 3 sec and Beta = {0.3,0.5}
 - Roughly the half of disconnection has been solved

Beta	0.3	0.5	0.7
Polling Time 3 sec	1%	0.09%	0.02%
Polling Time 5 sec	32%	4%	0.88%



Preliminary Experimental Results / 2

- The average number of going-to-create connected components in each run of experimentation is approximately:
 - 6 for $beta = 0.3$
 - 2 for $beta = 0.5$



Appendix & Demonstration

- Appendix available at:
<http://www.dis.uniroma1.it/~deleoni/documents/AppendixDMC2006.pdf>
- Demonstration available at:
<http://www.dis.uniroma1.it/~deleoni/documents/DMCVideo.avi>

Conclusions & Future Work

- Adaptiveness is needed for process management in MANETs.
 - Prediction of anomalies
 - Identification of corrective actions (e.g., bridging services)
 - Process restructuring
- In the context of an IST project, named WORKPAD, starting from September 2006, we plan to:
 - Deeper experimentation
 - Distributed Coordination
 - Support for (some) disconnected modus operandi