21

Wireless Communication Protocols for Distributed Computing Environments

Romano Fantacci, Daniele Tarchi and Andrea Tassi
University of Florence
Italy

1. Introduction

The distributed computing is an approach relying on the presence of multiple devices that can interact among them in order to perform a pervasive and parallel computing. This chapter deals with the communication protocol aiming to be used in a distributed computing scenario; in particular the considered computing infrastructure is composed by elements (nodes) able to consider specific application requests for the implementation of a service in a distributed manner according to the pervasive grid computing principle (Priol & Vanneschi, 2008; Vanneschi & Veraldi, 2007).

In the classical grid computing paradigm, the processing nodes are high performance computers or multicore workstations, usually organized in clusters and interconnected through broadband wired communication networks with small delay (e.g., fiber optic, DSL lines). The pervasive grid computing paradigm overcomes these limitations allowing the development of distributed applications that can perform parallel computations using heterogeneous devices interconnected by different types of communication technologies. In this way, we can resort to a computing environment composed by fixed or mobile devices (e.g., smartphones, PDAs, laptops) interconnected through broadband wireless or wired networks where the devices are able to take part to a grid computing process. Suitable techniques for the pervasive grid computing should be able to discover and organize heterogeneous resources, to allow scaling an application according to the computing power, and to guarantee specific QoS profiles (Darby III & Tzeng, 2010; Roy & Das, 2009).

In particular, aim of this chapter is to present the most important challenges for the communication point of view when forming a distributed network for performing parallel and distributed computing. The focus will be mainly on the resource discovery and computation scheduling on wireless not infrastructured networks by considering their capabilities in terms of reliability and adaptation when facing with heterogeneous computing requests.

2. Related works

A particular interest in the literature is towards the interactions between high performance and distributed computing schemes. The main paradigms that allow a wide-area computing in a distributed fashion are represented by the grid (Parashar & Lee, 2005) and the cloud computing (Foster et al., 2008); in the last years these paradigms, originally disjointed, aim to be more overlapped considering the remarkable research efforts regarding communication standards and computing devices. In particular, if we also take into account hardware and

software capabilities of consumer mobile devices (like PDAs or mobile phones) we can realize that they no longer can be considered simply as *service consumers* through which to request a service to an elaboration center, but now they are able to take part to a distributed elaboration process (e.g., the distributed problem solving, etc.).
The pervasive grid and cloud computing paradigms are placed in a scenario characterized by fixed and mobile nodes characterized by different computing power and interconnected by several wired and wireless links relying on different communication technologies with heterogeneous rate and delay profiles.

2.1 Strategies for pervasive grid computing

The pervasive grid paradigm is strictly related to the computational grid concept (Foster & Kesselman, 1999); in this vision a computing architecture is composed by a central computing center made of clusters of fixed nodes that provide a set of services on the outside. An user can exploit these services through a pervasive infrastructure allowing a completely transparent access to the end-user at the computing center. The pervasive infrastructure can be composed by heterogeneous networks, devices with different computing power and equipped by different softwares.
The pervasive grid computing represents a significant innovation because in this case the computing resources are "pervasive" (Parashar & Pierson, 2010); for this reason not only a cluster of workstations can take part to a distributed and parallel computing process, but also a mobile device can be used as a computing node.
The most challenging problems related to a computing infrastructure composed by mobile nodes, interconnected by not reliable communication networks (e.g., P2P networks, etc.) are defined in (Batista & da Fonseca, 2010; Darby III & Tzeng, 2010; Ranjan et al., 2008). The key aspects of a distributed application are: the context-awareness, the self-adaptiveness, the QoS-awareness.
Ref. (Hingne et al., 2003; McKnight et al., 2004; Vanneschi & Veraldi, 2007) present a set of distributed computing models that try to address the context-awareness problem; it is important to note that this aspect is strictly related to the self-adaptiveness problem because if a change in the computing resource set is detected (e.g., a CPU is overloaded or a PDA battery is exhausted) the distributed application should react to this change in order to preserve, e.g., the integrity of a result in a distributed problem solving process. This problem has been addressed in two ways:

- relying on a middleware software layer shared among the computing nodes that globally reconfigures the distributed application (e.g., switching at run-time to different implementation or configuration of a service (Coronato & De Pietro, 2008; Coulson et al., 2005; Emmerich, 2000)) in a way fully invisible to the application itself;
- expressing the self-adaptiveness directly in the distributed application demanding to it the self-configuration problem (Aldinucci et al., 2008; Fantacci et al., 2009; Huebscher & McCann, 2008; Vanneschi & Veraldi, 2007). In particular (Fantacci et al., 2009) describes a pervasive gird application as a set of components interconnected by logical communication channels; a component provides a pool of functionalities and can be deployed in different versions characterized, e.g., by different memory footprints, CPU usages, etc. In this model the self-adaptiveness problem is addressed by switching at run-time between the versions of the same component deployed in the computing nodes.

The QoS problem related to the grid paradigm has been addressed in the literature jointly with the resource discovery problem and the optimal job allocation: a job should be completed according to the SLA between the user and the organization holding the computing

infrastructure. In order to make it possible a job must be mapped in a set of nodes with enough computing power; they can be identified only through an efficient resource discovery technique. In (Al-ali et al., 2003; Sundararaj et al., 2004) all these aspects have been analyzed for networks at "Internet scale"; in particular, the first one proposes an extension of the classical Globus model (Morohoshi & Huang, 2005) adopting a DiffServ (Park, 2006) approach and the second one addresses these problems building a virtual overlay network interconnecting all computing resources in order to route better the network traffic and making the grid infrastructure able to react to bandwidth fluctuations.
In the literature the QoS problems relative to the pervasive grid paradigm have been addressed in the following ways:

- the resource discovery and monitoring is demanded to the middleware layer or to a software service that makes the distributed application aware of the available computing resources (Noble, 2000; Schmidt & Parashar, 2004). The common drawbacks of these techniques are that all the information discovered must be stored in centralized or distributed indexes that can be built only through ad-hoc messages contributing to the network congestion;
- there are other strategies specifically designed for pervasive grid environments, characterized by fixed and mobile devices interconnected through heterogeneous wireless networks where the resource discovery and network routing capabilities are covered by the same service; this happens in (Li et al., 2005) and (Fantacci et al., 2010) where an enhanced version of two routing protocol optimized for mobile and ad-hoc network (MANET) with integrating resource discovery capabilities is described.

Finally, in the literature other aspects has been addressed as the fault tolerance (Agbaria & Sanders, 2005; Bertolli, Vanneschi, Ciciani & Quaglia, 2010; Oliner et al., 2005) or the security (Saadi et al., 2005; The Globus Security Team, 2005) of a pervasive grid application.

2.2 Strategies for cloud computing

Foster et al. in (Foster et al., 2008) define the cloud computing paradigm as a large-scale computing paradigm in which a pool of resources are delivered on demand to external customers over the Internet. In the distributed system scenario a cloud computing system represent a sort of evolution of the classical grid paradigm, like the pervasive grid.
A cloud computing system can provide services at three levels (Zhang et al., 2010):

- Software as a Service (SaaS) making users able to share applications running on a distributed infrastructure;
- Platform as a Service (PaaS) allowing users to access to an integrated environment in order to develop and deploy parallel application;
- Infrastructure as a Service (IaaS) allowing users to share hardware resources (like computing power, storage, etc.).

For these reasons, from the application context-awareness, self-adaptiveness and QoS-awareness point of view, the cloud computing paradigm shares with the pervasive grid one the same problems and solutions (Foster et al., 2008).

3. Implementation model of a parallel and distributed decision support system

In order to introduce the communication protocols definition we introduce now the considered pervasive grid approach proposed in (Aldinucci et al., 2008) and successively adopted in (*CoreGRID Network of Excellence*), where a distributed application can be modeled

as a graph: the vertices are the logical components and the edges are the communication channels. The logical components of the application can be different, performing each one a specific set of operations on the input data. The communication between logical components is achieved through data streams, considered as a set of elements transmitted serially by a component to another one (Bertolli, Buono, Mencagli & Vanneschi, 2010).
A logical component of a distributed application relying on such parallel computing model can be considered as a process running on a device; each node can (virtually) executes all the components forming the distributed application. One of the most important characteristics is that the whole application is able to dynamically reconfigure itself in order to match specific QoS constraints or to react to the context changes (e.g., changes in network topology, or in the computing node load). The adaptivity and the context awareness of the application can be explicitly programmed in every logical component. The two main types of adaptivity are:

- *performance adaptivity* - realized when the parallelism degree of a component changes;
- *functional adaptivity* - realized whenever a different version of the same resolution algorithm (e.g., a reduced memory footprint, etc.) is adopted.

A general purpose model for a parallel computing application should be based on the following set of entities:

- *the data source* - the source generating data; each measure is called *point*. The whole source area can be divided in parts, called *cells*, where the data source is placed;
- *the processing nodes* - in each cell a pervasive computing application relies on a distributed computed infrastructure (DCI) composed by a variable number of mobile users, having a device equipped with a wireless network adapter used to communicate with the data aggregators of the cell and with the other users;
- *the data aggregators* - in each DCI one or more devices are involved, gathering the points produced by the data source operating in the cell where the aggregator is; a user device can also play the role of data aggregator.

A parallel application can be developed according to different models: the two most common paradigms are task farm and data parallel. These two models differ in the type of logical components involved in the model and in the way the logical components cooperate to solve a problem. Other more complex structures can be easily seen as a combination of the two considered models.

3.1 Task farm paradigm

This paradigm, represented in Fig.1, relies on the replication of the same logical components used to perform the same operations on different input data.
A task farm application is composed by the following types of logical components:

- *the emitter (E)* receives an input stream formed by elements of the same type, each of which is transmitted to a worker;
- *the workers (W)* receive a different input stream element but process it with the same function *F*;
- *the collector (C)* gathers the outputs of the workers and forward them on the output channel with the form of an output stream.

In certain cases the emitter role can be played by the data aggregator node; however, we will consider in the following that the two functions are decoupled for a more general case. The main contribution to the computing power comes from the mobile nodes, considering that

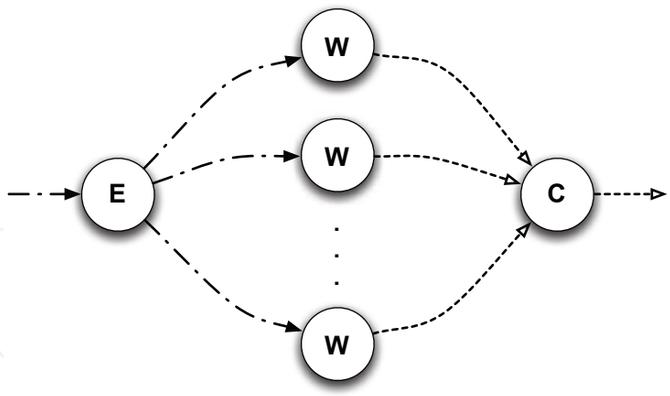


Fig. 1. Task farm model.

each node can execute one or more workers in parallel (e.g., a collector process can run on a single node or in multiple network nodes at the same time).
A task farm application, in a steady state, achieves a five stage pipeline system, where the first stage is composed by the emitter, the second by the communication link between the emitter and the worker node, the third by the workers, the fourth by the communication link between the worker and the collector node and the last by the collector. For this reason the average service time of the application (T_{farm}) can be expressed as:

$$T_{\text{farm}} = \max \left\{ T_E, T_{l_1}, \frac{T_W}{N_{farm}}, T_{l_2}, T_C \right\} \tag{1}$$

where T_E is the execution time at the source node (i.e., the time needed to distribute the tasks to be executed at the mobile nodes), T_{l_1} is the communication latency of a stream originating from the source and directed to the worker nodes, T_W is the execution time of W, T_{l_2} is the communication delay between a worker node a the collector, and T_C is the time needed at the collector for re-assemble the output of the workers, and N_{farm} represents the number of workers involved in the parallel computation. Being the third right member of (1) usually the biggest term, we have:

$$T_{farm} = \frac{T_W}{N}. \tag{2}$$

Let T_A the average time elapsed between the consecutive emission of two points (i.e., the average arrival time). We define the optimal parallelism degree ($\hat{N}_{farm}$) as the value that allows to have the average service time of the application equal to the average arrivals time. Hence, we have:

$$\hat{N}_{farm} = \left\lceil \frac{T_W}{T_A} \right\rceil \tag{3}$$

3.2 Data parallel paradigm

In the data parallel paradigm, represented in Fig. 2, the working processes involved in the computation are combined in order to solve one task at a time. As before, we have three types of modules:

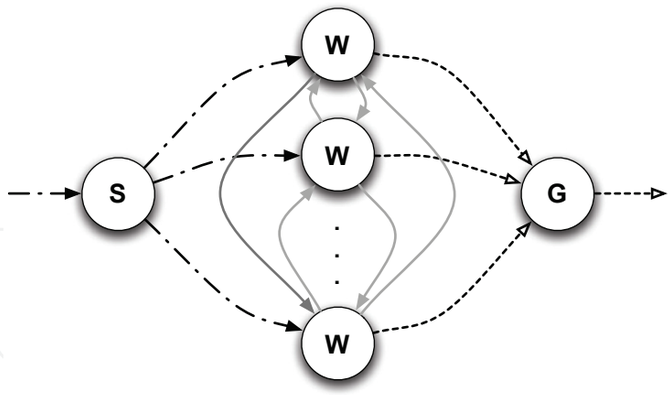


Fig. 2. Data parallel model.

- *the scatter (S)* receives the input stream, and compiles the *input state* (**M**) of the task. **M** is split in a number of parts (called sub-states) identical to the number of workers involved in the computation; each sub-state is transmitted to a worker;
- *the workers (W)* perform the sub-computations initialized with the sub-states received. If we consider **M** as a square matrix of $n \times n$ elements, formed by only a subset of columns of the **M** matrix and that all the sub-states have the same size, the computation can be modeled as an iteratively application of a function $H(\cdot)$, $\hat{S}$ times on each element of the sub-state;
- *the gather (G)* collects and reorders the outputs of the workers. G also builds the solution to be sent on the output line.

It should be noted that a worker involved in the computation can process the sub-state regardless of the other ones (the data parallel application is called *map*) or have some functional dependencies with other workers (in this case the application is called *stencil*). Moreover, we have a functional dependence when a worker needs to know some partial results from one or more workers in order to perform an iteration of $H(\cdot)$ on an element.
Likewise the task farm paradigm, the gather process can run on one or more nodes at the same time, and each node of the local computing cloud can execute in parallel one or more working processes. The node, where the scatter process is executed, is selected time by time through an optimal criteria (see Section 4). Finally, we assume that each worker transmits its output to the data aggregator node that also acts as the gather.
A map application, in a steady state, as in the task farm paradigm, realizes a five stages pipeline paradigm, where the first stage is composed by the scatter, the second by the communication link between the emitter and the worker node, the third by the workers, the fourth by the communication link between the worker and the collector node and the last by the gather. The average service time (T_{map}) can be expressed as:

$$T_{map} = \max\left\{T_S, T_{l_1}, T_W, T_{l_2}, T_G\right\} \tag{4}$$

where T_S and T_G are, respectively, the average service time of S and G, T_{l_1} and T_{l_2} are, respectively, the maximum communication latencies encountered by a communication between the scatter or the gather node and each worker involved in a computation. Also in this type of parallelism, usually, T_W is greater than the other terms. Hence, we have:

$$T_{map} = T_W. \tag{5}$$

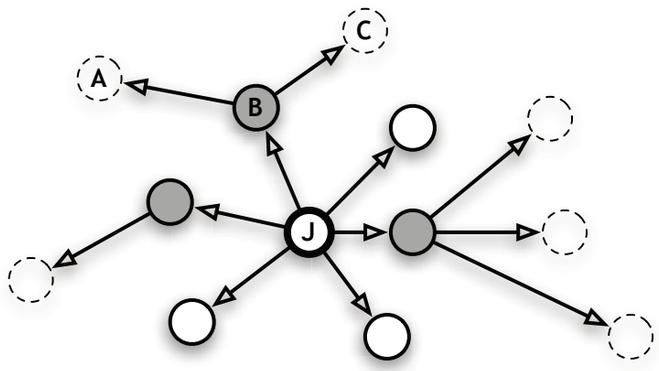


Fig. 3. A flat topology network based on the multi-hop communication paradigm.

Finally, T_W can be expressed as:

$$T_W = \hat{S} \cdot T_H \cdot \frac{n^2}{N_{map}} \tag{6}$$

where T_H is the average time needed by a computing device to apply the H function just one time on a single element coming from the input state, N_{map} is the number of workers involved in the computation and the ratio between the total number of elements of $\mathbf{M}$, n and N_{map} gives the sub-state size (in terms of number of elements).
With this type of parallelism we define the optimal number of workers ($\hat{N}_{map}$) as the amount of computing processes by which it is possible to have an average service time of the whole application equal to the average arrival time, and hence avoiding the saturation of the input queue of the parallel application. In order to have $T_A = T_{map}$ or equivalently, as assumed above, $T_A = T_W$, the workers number, according to (6), results to be:

$$\bar{N}_{map} = \left\lceil \frac{\hat{S} \cdot n^2 \cdot T_H}{T_A} \right\rceil \tag{7}$$

4. Resource discovery and routing

When facing with a heterogeneous network the first challenge to be considered is the discovering of the resources in the network in order to have a more detailed model of the active devices. Herein we assume to be in a flat topology network where each node has the same role. This model allow us to consider the most general case, where no hierarchy is considered within the network.
In such networks (Fig. 3), if the node **A** needs to communicate with **C** (and they are not in the radio connectivity range), it is possible through the relay node **B** according to multi-hop approach.
In order to perform the best selection among the DCI we have to identify the best nodes to be allocated for the execution of a specific distributed application. On the other hand a flat topology network needs to be based on a smart routing algorithm able to find the best route between different nodes; this is even more important in the case mobility becomes an issue: in this case we will refer to the so-called MANET (mobile and ad-hoc networks). The most convenient approach seems to be an exploitation of the routing algorithm in order to fulfill the resource discovery task.

There are several basic routing protocols for ad-hoc networks (Badis & Al Agha, 2005; Clausen & Jacquet, 2003; Haas & Pearlman, 2001; Perkins & Royer, 1999); they can be classified into three families:

- *proactive family* - the routing strategy relies on a periodic update of the routing information stored in each node in order to make a node able to communicate with the rest of the network, the routing table of a node is composed by each possible route. The most popular in this class is the OLSR algorithm (Clausen & Jacquet, 2003) and its QoS-aware extension called QOLSR (Badis & Al Agha, 2005), whose aim is to consider the QoS requirements of the different traffic flows for a better routing decision;
- *reactive family* - differently from the previous routing family, a route from two network nodes is discovered and used in the source and in the intermediate devices of the path only when a source node needs to communicate with the destination node. A well known reactive routing protocol is the AODV (Ad-hoc On-demand Distance Vector (Perkins & Royer, 1999));
- *hybrid family* - these routing algorithms are composed by two parts: the first operate in a proactive way, and the second in a reactive way. The ZRP (Zone Routing Protocol) (Haas & Pearlman, 2001) is one of the most important protocol belonging to this class.

Herein the aim is to identify a routing protocol able to allow a pervasive grid computation in flat topology network. It is well known that reactive protocols require less network traffic than the proactive ones, but the QOLSR principle (Badis & Al Agha, 2005), despite a higher network control traffic with respect to reactive alternatives, can provide the following characteristics:

- the reactive protocols have a not predictable setup time for setting up a route and update the routing tables for the nodes involved in the communication;
- the end-to-end QoS of a path between the source and the destination nodes has to be explicitly monitored in each intermediate node;
- the routing messages cannot be used to periodically broadcast information or as probe messages in order to estimate some QoS indexes.

4.1 An enhanced routing scheme for pervasive grid computing

As highlighted before in the scenario we are considering we have to face with the problem of discovery the resources within the network by exploiting a routing algorithm. The EOLSR protocol (Fantacci et al., 2010), relying on the QOLSR protocol, adds the following characteristics in order to fulfill the requirements of the considered scenario:

- it operates in a distributed way and does not require any supervisor node;
- it does not require a reliable transmission of routing messages;
- it performs an hop-by-hop routing approach, and each network node, belonging to the path connecting the source with the destination device, chooses the next destination to send the traffic;
- it is able to find always the optimal path between every pair of network nodes.

As for the QOLSR case, with the EOLSR protocol each node has to perform the following operations:

1. detecting, through the received HELLO messages, its 1-hop neighbors;
2. performing the MPR (multipoint relay) selection and updating its topology table thought the topology control (TC) messages;

3. computing the routing table.

A node J (Fig. 3) broadcasts the HELLO messages to its 1-hop neighbors with the source address and the list of its 1-hop neighbors. For each element E of the advertised list, the HELLO message carries on also additional informations, as the rate and delay of the links interconnecting J to E. As usual, the HELLO messages are never relayed by the nodes.
The set of 1-hop neighbors of a generic node J is called "neighbor set of J" (N_J^1) and every network device can compute it relying only on the received HELLO messages. The node J, through the advertised list carried on each HELLO message, can also build its 2-hops neighbors set (N_J^2).
According to the QOLSR algorithm (Badis & Al Agha, 2005), every node has to compute its MPR set. The MPR set of the node J (MPR_J) is a subset of N_J^1 and it is composed by those network nodes through which is possible to communicate with every element of the N_J^2 set relying only on two hops paths.
The generic node A, member of the MPR_B, is called "MPR of a node B"; A is informed of its new state through the HELLO messages transmitted by B itself; A keeps a list of nodes (called "MPR selector list", indicated as MPR_{adv}) that have chosen it as MPR node.
All the nodes with a not empty MPR selector list must transmit periodically a Traffic Control (TC) message embedding its MPR selector list; each element of that list stores the IP address of the node that has A in its MPR set, the QoS indexes of the communication link and the context information relative to the that node itself. Differently from the HELLO messages, the TC messages must be received by each node, but they are not broadcast across the network: a TC message of A is processed by only the members of the MPR_A, as well as, A will process only message received by its MPR set. It can be proven that in this way all the network devices receive the TC messages transmitted by a node under the assumptions of an ideal channel and medium access.
Through the received TC messages, each node can build a topology table consisting of entries formed by the IP address of an element coming from the MPR selector list carried by a TC message (called *destination address*), the source address of the TC message itself (called last-hop address), the rate and the delay of the communication link between the last-hop address, and the destination address and the context information relative to the last-hop node.
Aim of the EOLSR protocol is to integrate the resource discovery functionality in the routing procedure. Toward this end, the set of QoS indexes present in the HELLO message has been extended (Fantacci et al., 2010) in order to include the following context information: the remaining battery power, the CPU architecture type (e.g., i386, XScale, MIPS, etc.), the amount of CPU and allocated memory (this one normalized respect to its maximum value) of each 1-hop neighbor advertised by the HELLO message. It is important to note that, even if in the EOLSR the HELLO header size is increased of 32 bits, we have to stress that the size of each element of the advertised list remains of the same dimension.
In order to carry the context aware information it is needed also to extend the TC messages; in particular to make each node able to build an exact snapshot of the network and computing resources, the extended TC messages should be transmitted not only by the node with a not empty MPR selector list but by all network nodes. If the TC messages were transmitted only by the nodes with a non empty MPR selector list, as QOLSR requires for the regular TC messages, only their extended neighbor lists would be propagated all over the network resulting in a partial perception of the real state of the communication and processing capabilities of the network.
The extension of the TC messages does not increase the message header size or the dimensions of the elements of the advertised MPR selector list size whereas the classical QOLSR

dimensions are increased by the extended neighbor list carried within the message; an entry of that list has the same dimension of an element of the MPR selector list (i.e., 12 bytes) and has a number of elements equal to $|E_J^n| - |\text{MPR}_{adv}|$.

When a node has completed the filling (or updating process) of the neighbor and topology tables, it computes (or periodically updates) its own routing table, differently from the QOLSR protocol; the classical entry (destination address, next-hop address and, the path length) has been extended (Fantacci et al., 2010) in order to carry also:

- the rate (R) and the delay (d) of the path (not required by the QOLSR protocol) respectively defined as
$$R(r) = \min\{R(A,B), R(B,C), \ldots, R(E,F)\} \tag{8}$$
and
$$d(r) = \sum_{i=1}^{N-1} d(i, i+1), \tag{9}$$
where r is a path through $n-1$ hops from the source A and the destination node F along the $B, C, \ldots, E$ devices.
- the remaining battery life of the path r defined as:
$$P(r) = \min\{P(B), P(C), \ldots, P(F)\} \tag{10}$$
where $P(I)$ is the remaining battery life of the node I normalized respect to its maximum value;
- the processor type, the amount of CPU occupied and memory allocated (this one normalized respect to its maximum value) in the destination node.

The extended version of the routing table can be built (or updated) in the following way:

1. all the entries are removed from the table;
2. each element of the neighbor set is put into a routing table entry, by setting:
   - the destination and the next-hop IP equal to the address of the 1-hop neighbors;
   - the information on the network and computing power of the destination node as the same values of those in the neighbor table;
   - the path length to one;
3. for $j = 1$, until at least one is updated, an iteration is performed:
   - for each element (TC_{elem}) of the topology table, with its destination address not matching the destination address of any route (RT_{elem}) and its last-hop address corresponding to a destination IP reachable through a route already present in the routing table (with a path length equal to j), the following entry is added: the destination address is the same of that entry, the next-hop address is set equal to the next-hop address, the path length is set equal to $j+1$, the rate and the remaining battery life of the path are set to the minimum value respectively stored in TC_{elem} e RT_{elem}, the delay equal to the sum of that indexes reported in the same two entries, the indexes describing the computing power of the destination node are the same of those reported in the topology table entry considered;
   - $j = j + 1$;
4. all the not considered entries of the topology table can be erased.

In order to simplify the task scheduling to the nodes, a *cluster table* needs to be defined. In a network of N nodes a cluster table has $N-1$ entries with the following fields:

- *the cluster center (CC)* - it is a data structure storing the IP address of the a certain node; it stores the cluster table, the rate, the delay and the remaining battery power of the path *p* and its computing information (the processor type, the amount of CPU and memory occupied);
- *the list of the cluster members* - given a certain cluster, a network node is one of its cluster member if it is reachable through a path, composed by a number of hops less or equal to *cd* (called "cluster depth"). Each element of that list stores the computing information of the considered cluster member (such as the processor type, the amount of CPU and memory occupied) and the QoS indexes (rate, delay and remaining battery power) related to the path between CC the cluster member itself.

More details regarding the building process of a neighbor, a topology or a cluster table can be found in (Clausen & Jacquet, 2003) and (Fantacci et al., 2010)

5. Lower layers reconfigurability

In the wireless network scenario we are considering, also the lower layers became of great importance. In that sense it will be of particular interest those techniques that foresee the link selection on a reliability base: we will refer to those algorithms that aim to optimize the resource discovery phase by considering the link status among the nodes.
In that sense, we will refer to a scenario where multiple wireless communication technologies co-exist allowing the choice of different paths with different link layer technologies among the nodes. The scenario refers to the wireless networks, often overlapped among them, in presence of multi-interface terminals that can connect to different technologies. Main characteristic of these techniques is to allow to the different technologies to supplement among them and not to compete for the bandwidth.
Typical example is the 3G technology that has a broad coverage, medium bandwidth and not low access cost, and the IEEE 802.11x technologies that are broadband, low cost, but with a low coverage area. This heterogeneity can be an advantage for the mobile devices, thanks to the exploitation of multi-interface solutions, by selecting the best interface for creating a map of the available resources. For this reason the resource discovery techniques defined in the previous can help to map and estimate as best as possible the actual network status, and proceed to its reconfiguration.
At link layer, adaptive techniques for the multiple access management are developed by considering ad-hoc or not infrastructured networks where the QoS respect for certain streams needs to be satisfied; the aim is the definition of a set of adaptive schemes, that can be selected each time in function of the actual network status. In this scenario, also considering the physical layer status, radio resource allocation techniques needs to be considered. In that context, we will refer to the opportunistic scheduling techniques, where the amount of data to be sent depends not only from the priority but also from the wireless channel status.
At physical layer, it is possible to foresee the use of adaptive schemes for the physical parameters for the management of the radio resources. Almost all modern communication systems allow the adaptation of the modulation, coding, and transmission power schemes, and in some cases, also the timing and frequency division. In this scenario, also those schemes that will use multiple antennas for both beamforming and diversity schemes needs to be considered.

6. Computation scheduling

After resource discovery has been performed the computing source needs to schedule the computation to the nodes following the chosen parallelization scheme.

6.1 Mapping for a task farm application

When a new task needs to be scheduled by the emitter, it needs to choose the best network node where mapping the computation. For each network node i we can define a variable and fixed cost (Fantacci et al., 2010), respectively, C_i and B_i, as:

$$C_i = \Delta\widehat{MEM}_i + \Gamma\widehat{CPU}_i \tag{11}$$

$$B_i = \alpha\check{R}_i + \beta d_i + \gamma P_i + \theta MEM_i + \psi CPU_i \tag{12}$$

The first one is a linear combination of the amount of occupied memory ($\widehat{MEM}_i$) and CPU ($\widehat{CPU}_i$) that would be allocated if the computation would be mapped onto the node i, i.e., the "cost to be payed" whenever a computation is mapped on that node; the same computation can allocate a different amount of memory or cause a different CPU load accordingly to the architecture type of the elaboration device where the working process is executed. For this reason it is called variable cost. B_i is a linear combination of:

- the rate margin $\check{R}_i$ (defined as the difference between the maximum applicable rate value and R_i itself), the delay (d_i) and the amount of consumed battery power (P_i) concerning the path between the node E, where the emitter process is executed, and the network node i
- the amount of allocated memory (MEM_i), and the occupied CPU (CPU_i) in the node i at a certain time instant of the network kept by the node E (thanks to its own routing table).

All the QoS and the context information indexes appearing in (11) and (12) are normalized respect to their maximum values. The emitter has to map a computation on the network device i with the minimum *effective cost* K_i, where $K_i = C_i + B_i$. This mapping problem can be expressed as:

$$\text{(TF)} \quad \textit{minimize} \quad \sum_{i \in V} M_i K_i \tag{13}$$

$$\textit{subject to} \quad \sum_{i \in V} M_i = 1 \tag{14}$$

where the set V is the routing table hold by the node where the emitter process is executed, the vector M is the variable of the optimization problem and, M_i (i.e., the i-th component of M with $i \in \{1, 2, \ldots, |V|\}$) is the number of computations that will be mapped performed by the node i. By the constraint (14) we will map only one computation at a time as required by the task farm paradigm.

The emitter can solve the optimization problem by performing an exhaustive search in the admissible solution set; for this reason the developed solution is always the optimal mapping regardless the network topology and the distribution of the computing resources in the network nodes. Note that this is not a too computationally expensive approach because a routing table is composed by a number of entries equal to the number of nodes participating to the network, whose value is usually not so high.

Procedure 1 Sub-optimal scheduling scheme for the data parallel paradigm.

$S \leftarrow \hat{S}_M$
while $S \geq \hat{S}_m$ **do**
  $W \leftarrow f(S)$
  if $W \geq W_{CT_i}$ **then**
    $B \leftarrow sort(B)$
    $j \leftarrow 1$
    while $W > 0$ **do**
      if $maxWorker(B_i) \leq W$ **then**
        $M_{B_j} \leftarrow \widehat{W}_{B_j}$
      else
        $M_{B_j} \leftarrow W$
      end if
      $W \leftarrow W - M_{B_j}$
      $j \leftarrow j + 1$
    end while
    return M
  else
    $S \leftarrow S - 1$
  end if
end while
return *the sub-computation can't be mapped*

6.2 Mapping for a data parallel application

The mapping process for a data parallel application should be done in two steps: first of all we choose the optimal cluster of network devices, and then we select the intra-cluster mapping. As described above, in a data parallel application, a set of working processes is globally involved in the solution of one and only one task at time.

Each sub-computation performed by a worker could have a particular stencil relation with other workers; for these reasons all the sub-computations related to a task should be mapped in a set of workers running in a group of network devices interconnected by links with a short delay and high rate (according to the QoS constraints of the computation). The uniform mapping of the sub-computations onto the network devices members of the optimal cluster is not always an optimum solution because they should be mapped preferably on the most powerful or lowest loaded nodes. The cost of the cluster I composted by z network nodes can be defined as:

$$CC_I = \sum_{j=1}^{z} B_j + F(\widehat{W} - W_I) \tag{15}$$

where B_j is the fixed cost of the node j (with $j \in I$), $\widehat{W}$ is the maximum number of sub-computations where a task can be divided, and W_I is the number of the working processes to be executed on the network nodes belonging to the considered cluster. We assume in what follows the optimal cluster, as the one having the lowest cost. The optimal cluster can be selected by performing an exhaustive search in the cluster table of the node where the task dispatcher process is executed. Note that this is a feasible approach because the cluster entries are no more than the routing entries.

The mapping process of the sub-computations, related to nodes belonging to I, is performed by the task dispatcher process itself and can be expressed in terms of the following optimization problem:

$$\text{(DP)} \quad \textit{minimize} \quad J \cdot \left[\sum_{i \in I} (M_i C_i + B_i) \right] + K \left(\hat{S}_M - \hat{S} \right) \tag{16}$$

$$\textit{subject to} \quad \hat{S}_m \leq \hat{S} \leq \hat{S}_M$$

$$0 \leq M_i \leq \widehat{W}_i, \quad \forall i \in I \tag{17}$$

$$\sum_{i \in I} M_i = W \tag{18}$$

where $\hat{S}_M$ and $\hat{S}_m$ are respectively the maximum and the minimum number of iterations that can be performed by the H function on an element of the input state, $\hat{S}$ are the iterations actually performed, $\widehat{W}_i$ is the maximum number of workers that can be executed in parallel on the node i (where $i \in I$) according to the architecture type of the node itself, W is the number of sub-computations where the task has been divided, J and K are two not negative weights. The variables of that optimization problem are the components of the vector M (M_i with $i \in [1, \ldots, |I|]$) and $\hat{S}$; these variables are all integer and not negative.
The minimization performed in (16) results in a minimization of the mapping cost and in a maximization of the number of iterations performed on each element; the optimization problem is not only bi-objective but also not linear: the number of workers W is function of S (e.g., for a MAP it is given by (7)), C_i is function of W then the fixed cost is function of S. In this case the solution of the mapping problems can not be found through an exhaustive search in the admissible solutions space. This heuristic (reported in Procedure 1 and summarized as follow) can be used to get a sub-optimal solution (Fantacci et al., 2010):

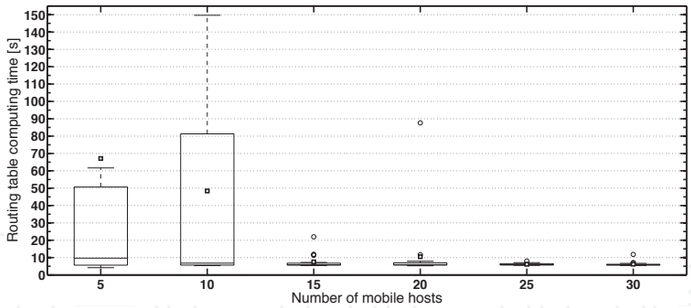
1. compute the fixed cost of all the network devices belonging to the optimal cluster;
2. map in each device a number of sub-computations equal to the number of working processes that can be executed in parallel in the node itself or equal to the remaining sub-computations, starting from the node with a smaller fixed cost;
3. assign the scatter role to the node with the minimum fixed cost.

7. Performance results

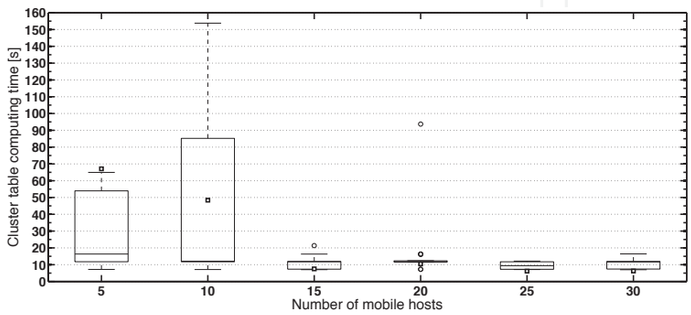
In order to have a performance estimation of the distributed computing application in the wireless environment in this section we summarize some numerical results. The following simplified scenario has been considered:

Coefficients	Policy A	Policy B	Policy C
α	2	2	2
β	6	6	6
γ	0	4	4
Δ, θ	0	1500	1500
Γ, ψ	0	1	0
F	100	100	100

Table 1. The weights used to define the policies A, B and, C.



(a) RT_{fill} of the fixed node



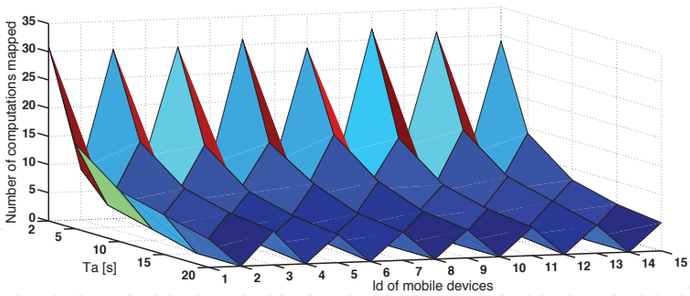
(b) CT_{fill} of the fixed node

Fig. 4. Average time needed to fill-up the routing and the cluster table of the fixed node.

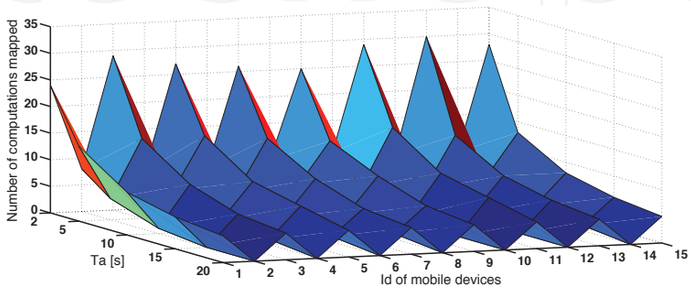
- a fixed node placed in the center of a square of area 0.25 km^2 or 1 km^2;
- a variable number ($5 \div 30$) of mobile nodes randomly placed in the playground, moving according to the random waypoint model (RWP (Bettstetter et al., 2003)), considering a pedestrian model with a speed uniformly distributed within [3 km/h, 5 km/h] and the possibility for each node to remain stationary for a time interval uniformly distributed between 3 s and 30 s;
- communication links using the IEEE 802.11g technology with a radio data rate of 54 Mbit/s.

One of the main performance indicator is the routing tables (RT_{fill}) and the cluster tables (CT_{fill}) filling time, beginning from an empty structure, stored in the node that computes the mapping solution. It can be shown that this values represent the worst case for the updating process because the time interval between two consecutive updates will never be greater than the time required to compute (or refresh) all the items of the routing or cluster table.
Note that it is not possible to identify the globally optimal values for RT_{fill} and CT_{fill} because they depend on the particular application to be implemented according to the pervasive grid computing paradigm. However, for the case of interest here, our analysis has shown that in a low mobility scenario the transmission of the HELLOs at least every 2 s ($T_{\mathrm{HELLO}} = 2$) and the TCs every 5 s ($T_{\mathrm{TC}} = 5$) is the optimal solution. In particular, in Fig.4(a) and 4(b), the RT_{fill} and CT_{fill} are drawn as box-plot[1]. Looking at these results it can be noted that the average values

[1] The top of the rectangle represents the twenty-fifth percentile of the observations, the bottom is the seventy percentile, the horizontal line into the boxes represents the medium value, the whiskers originating from the rectangles connects the minimum and the maximum value not considered as outliers, the circles are the outliers and the little squares represents the mean values of the observations.

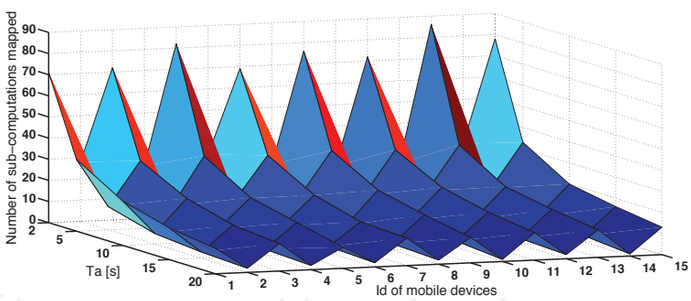


(a) Policy B

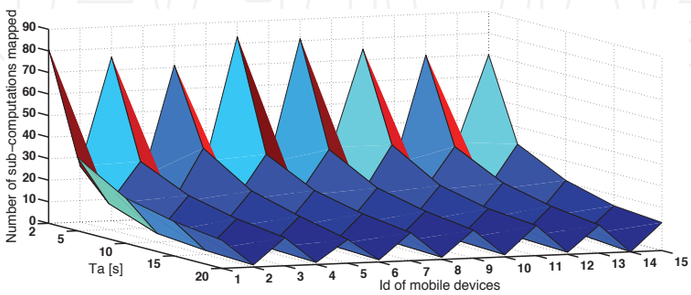


(b) Policy C

Fig. 5. Number of computations mapped on each node relying on a network with nodes with different battery capacity.



(a) Policy B



(b) Policy C

Fig. 6. Number of sub-computations mapped on each node relying on a network with nodes having different battery life.

for RT_{fill} and CT_{fill} are always between 5 s and 10 s with networks composed by 15 or more nodes. These values can be reduced or increased changing the maximum transmission period for the HELLOs and TCs In particular, under equal hypothesis, except for the TC messages transmitted every 2 s, it can be noted that the RT_{fill} and CT_{fill} are both less or equal to 10 s.
The other parameter to be taken into account is referred to the scheduling and resource allocation. In particular, we will consider a task farm and a data parallel application characterized by:

- an input and an output state of 1 MB;
- the emission of a new point every 5 s or 10 s.

By properly choosing the weights (see Tab. 1) introduced in (11), (12) and (15), it is possible to compare the results by considering three different policies:

- *Policy A* - the computations or the sub-computations are mapped using only the rate and the delay indexes;
- *Policy B* - the mapping is performed using all the QoS indexes and the context information;
- *Policy C* - it is the same of the policy B while the amount of CPU occupied or that will be occupied in a node i is ignored.

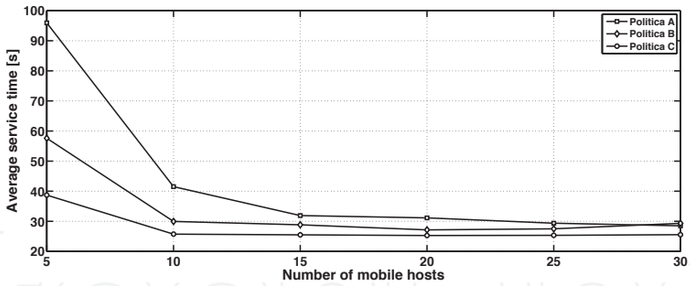
The performance results are expressed in terms of number of computations that can be mapped on each node. We have considered that the nodes are equipped with batteries having a different battery life; in particular, the node with odd id had batteries with an higher battery life than that related to the even ones.
In Figs.5 and6, the performance of the proposed approach is reported in terms of computations (or sub-computations) number mapped on each mobile node for the policies B and C. As for the previous cases, we can see that these policies correctly map more computations on nodes characterized by a greater remaining battery life.
Other two important performance metrics are the average service time and outage probability. The first parameter is the average time needed to finish a task from the emission of a point until than the whole state has not completely received by the node where is running the gather (or the collector) process; the second one can be defined as:

$$\hat{O}_{TF} = 100 - \frac{N_{comp} \cdot 100}{N_{mapped}} \tag{19}$$

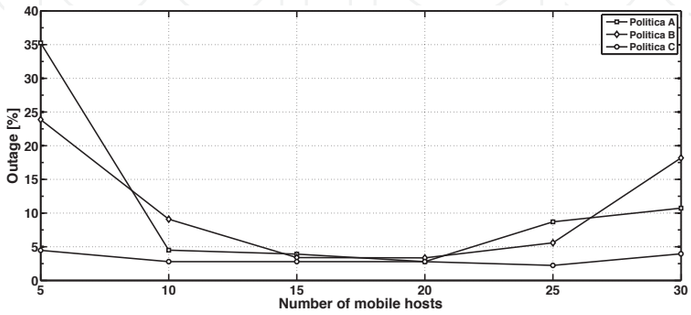
for a task farm application, where N_{comp} is the number of output states successfully received by the collector process and N_{mapped} is the number of computations mapped on each working processes in the time interval considered. Likewise, this parameter results to be:

$$\hat{O}_{DP} = 100 - \frac{N_{comp} \cdot 100}{N_{arrived}} \tag{20}$$

for a data parallel application, where $N_{arrived}$ is the number of output states successfully or partially recovered by the gather process in the time interval considered and, in this case, N_{comp} is related to the gather process.
In Fig. 7, the average service time and the outage probability are shown by varying the number of mobile devices, randomly placed in a square of 1 km^2, for the cases of T_a equal to 5 s (Fantacci et al., 2010) and with tasks requiring a computing time T_c equal to 22.65 s. Moreover Figs. 8(a) and 8(b) show, respectively, the average number of pending computations, mapped in a reference working node, that are waiting to be processed and the

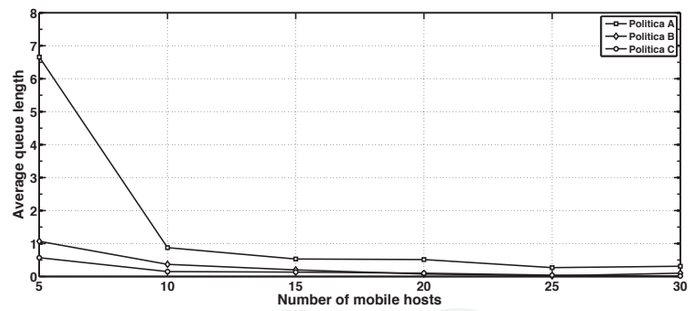


(a) Average service time

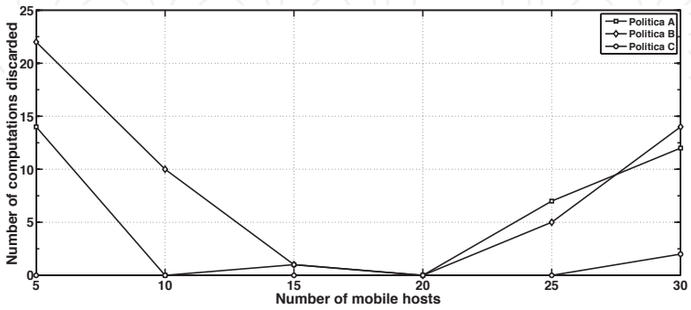


(b) Outage probability

Fig. 7. Computing performances of a task farm application.

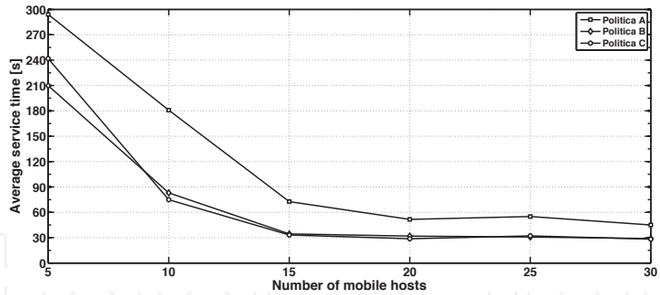


(a) Average input queue length

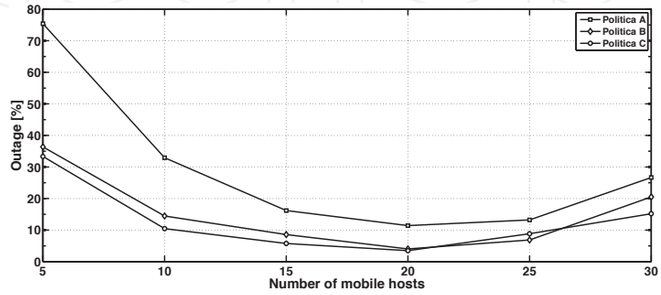


(b) Number of computations discarded

Fig. 8. Average queue length and number of computations discarded for a task farm application.



(a) Average service time



(b) Outage probability

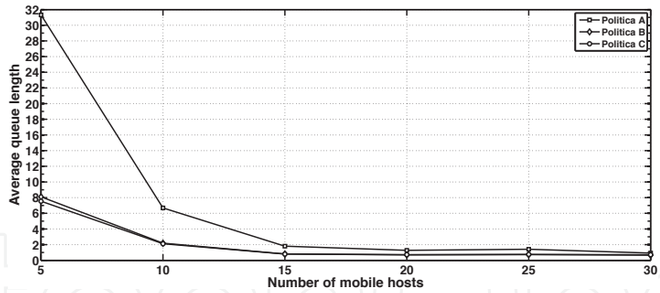
Fig. 9. Computing performances of a data parallel application

number of computations completed but discarded by the working node itself (for the three policies). From Fig. 7, it is possible to note that the policies B and C globally outperforms the policy A. Moreover, it is important to note that:

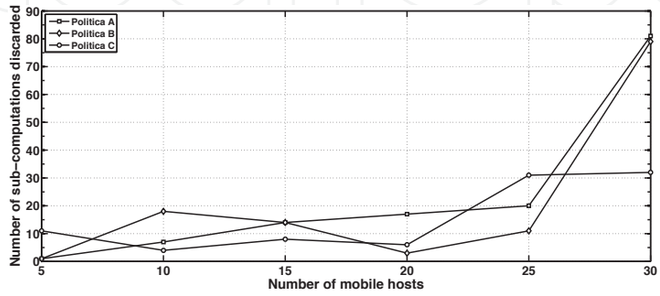
- the outage events in a network composed up to 15 mobile nodes are mainly caused by a non-homogeneous mapping and small number of computing resources present in the network (resulting on a increment of the time spent in the input queue of the device, Fig. 8(a)). The outage events are also caused by the cancellation events of tasks that occurs when the output of a sub-computation can not fully be transferred to the collector process due to the output state size and the small spatial density of nodes (as shown in Fig. 8(b));
- in networks composed by 20 or more nodes, as depicted in Fig. 8(b), the outage events are mainly caused by the cancellation events caused by the network interferences that characterize medium/large networks;

In Fig. 9, the computing performance is reported considering a data parallel application characterized by clusters of three network nodes (with one working process running on each one) and using sub-computations 15 s long. We can see that with this form of parallelism the policies B and C outperforms A while B and C are characterized mainly by the same performance.

As for a task farm application, Figs. 10(a) and 10(b) depict, respectively, the average number of pending sub-computations and the number of the discarded sub-computations (for the three policies). In this case the outage events are caused by the non homogeneous mapping in networks composed up to 15 nodes, otherwise, by the cancellation events due to the network interferences.



(a) Average input queue length



(b) Number of sub-computations discarded

Fig. 10. Average queue length and number of sub-computations discarded for a data parallel application.

8. Conclusion

Distributed computing systems are gaining an even more attention in the world due to their ability in processing great amounts of data. Their importance is even more increased in the recent years due to the introduction of wireless communications protocol able to connect even mobile terminals with broadband connections. Moreover, for the consumer electronics sphere there has been the introduction of small devices with high computations capabilities. This allowed the introduction of the pervasive grid concept aiming to exploit several different devices connected with heterogeneous communication links in order to realize a whole processing system.

In this chapter we have focused our attention on the most important aspects of distributed computing in wireless scenarios. First of all we have to face with the problem of discovering the resources in terms of device and communication link capabilities. This can be realized by exploiting routing algorithms that need to be used within such scenario due to the flat topology of a distributed network. Moreover also lower layer behavior became of importance due to their effect in the communication performance. Finally the scheduling phase is described aiming to find the best nodes in the sense of minimize certain cost functions. The performance results allow to see the importance of a good resource discovery and scheduling algorithm in the distributed computing problems when facing with the wireless environment.

9. References

Agbaria, A. & Sanders, W. H. (2005). Application-driven coordination-free distributed checkpointing, *Proc. of ICDCS 2005*, Columbus, OH, USA, pp. 177–186.

Al-ali, R., Hafid, A., Rana, O. F. & Walker, D. W. (2003). On QoS adaptation in service-oriented grids, *Proc. of MGC2003*, Rio de Janeiro, Brazil.

Aldinucci, M., Campa, S., Danelutto, M., Vanneschi, M., Kilpatrick, P., Dazzi, P., Laforenza, D. & Tonellotto, N. (2008). Behavioural skeletons in GCM: Autonomic management of grid components, *Proc. of PDP 2008*, Toulouse, France, pp. 54–63.

Badis, H. & Al Agha, K. (2005). QoS routing for ad hoc wireless networks using OLSR, *European Transactions on Telecommunications* 16(5): 427–442.

Batista, D. & da Fonseca, N. (2010). A survey of self-adaptive grids, *IEEE Transactions on Communications* 48(7): 94–100.

Bertolli, C., Buono, D., Mencagli, G. & Vanneschi, M. (2010). Expressing adaptivity and context awareness in the ASSISTANT programming model, *Autonomic Computing and Communications Systems*, Vol. 23 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg, pp. 32–47.

Bertolli, C., Vanneschi, M., Ciciani, B. & Quaglia, F. (2010). Enabling replication in the ASSISTANT programming model, *Proc. of IWCMC '10*, Caen, France, pp. 509–513.

Bettstetter, C., Resta, G. & Santi, P. (2003). The node distribution of the random waypoint mobility model for wireless ad hoc networks, *IEEE Transactions on Mobile Computing* 2(3): 257–269.

Clausen, T. & Jacquet, P. (2003). Optimized link state routing protocol (OLSR), RFC3626. URL: *http://www.ietf.org/rfc/rfc3626.txt*

CoreGRID Network of Excellence (n.d.). URL: *http://www.coregrid.net*

Coronato, A. & De Pietro, G. (2008). MiPeG: A middleware infrastructure for pervasive grids, *Future Generation Computer Systems* 24(1): 17 – 29.

Coulson, G., Grace, P., Blair, G., Duce, D., Cooper, C. & Sagar, M. (2005). A middleware approach for pervasive grid environments, *Proc. of UK-UbiNet/ UK e-Science Programme Workshop on Ubiquitous Computing and e-Research*, Edinburgh, Scotland.

Darby III, P. J. D. & Tzeng, N.-F. (2010). Decentralized QoS-aware checkpointing arrangement in mobile grid computing, *IEEE Transactions on Mobile Computing* 9(8): 1173–1186.

Emmerich, W. (2000). Software engineering and middleware: a roadmap, *Proc. of ICSE '00*, Limerick, Ireland, pp. 117–129.

Fantacci, R., Tarchi, D. & Tassi, A. (2010). A novel routing algorithm for mobile pervasive computing, *Proc. of IEEE Globecom 2010*, Miami, FL, USA.

Fantacci, R., Vanneschi, M., Bertolli, C., Mencagli, G. & Tarchi, D. (2009). Next generation grids and wireless communication networks: towards a novel integrated approach, *Wireless Communications and Mobile Computing* 9(4): 445–467.

Foster, I. & Kesselman, C. (1999). *The Globus toolkit*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Foster, I., Zhao, Y., Raicu, I. & Lu, S. (2008). Cloud computing and grid computing 360-degree compared, *Proc. of GCE '08*, Austin, TX, USA, pp. 1 –10.

Haas, Z. J. & Pearlman, M. R. (2001). ZRP: a hybrid framework for routing in ad hoc networks, *Ad hoc networking*, Addison-Wesley, Boston, MA, USA, pp. 221–253.

Hingne, V., Joshi, A., Finin, T., Kargupta, H. & Houstis, E. (2003). Towards a pervasive grid, *Proc. of IPDPS'03*, Nice, France.

Huebscher, M. C. & McCann, J. A. (2008). A survey of autonomic computing—degrees, models, and applications, *ACM Comput. Surv.* 40(3): 1–28.

Li, Z., Sun, L. & Ifeachor, E. C. (2005). Challenges of mobile ad-hoc grids and their applications in e-healthcare, *Proc. of CIMED 2005*, Lisbon, Portugal.

McKnight, L. W., Howison, J. & Bradner, S. (2004). Guest editors' introduction: Wireless grids–distributed resource sharing by mobile, nomadic, and fixed devices, *IEEE Internet Computing* 8(4): 24– 31.

Morohoshi, H. & Huang, R. (2005). A user-friendly platform for developing grid services over globus toolkit 3, *Proc of ICPADS'05*, Fukuoka, Japan, pp. 668 – 674 Vol. 1.

Noble, B. (2000). System support for mobile, adaptive applications, *IEEE Personal Communications Magazine* 7(1): 44 –49.

Oliner, A. J., Sahoo, R. K., Moreira, J. E. & Gupta, M. (2005). Performance implications of periodic checkpointing on large-scale cluster systems, *Proc. of IPDCS 2005*.

Parashar, M. & Lee, C. A. (2005). Scanning the issue: Special isssue on grid-computing, *Proceedings of the IEEE* 93(3): 479–484.

Parashar, M. & Pierson, J.-M. (2010). Pervasive grids: Challenges and opportunities, *in* K.-C. Li, C.-H. Hsu, L. T. Yang & J. Dongarra (eds), *Handbook of Research on Scalable Computing Technologies*, IGI Global, chapter 2, pp. 14–30.

Park, S. (2006). DiffServ quality of service support for multimedia applications in broadband access networks, *Proc. of ICHIT '06*, Vol. 2, Cheju Island, Korea, pp. 513 –518.

Perkins, C. E. & Royer, E. M. (1999). Ad-hoc on-demand distance vector routing, *Proc. of IEEE WMCSA'99*, New Orleans, LA, USA, pp. 90–100.

Priol, T. & Vanneschi, M. (eds) (2008). *From Grids To Service and Pervasive Computing*, Springer, New York, NY, USA.

Ranjan, R., Harwood, A. & Buyya, R. (2008). Peer-to-peer-based resource discovery in global grids: a tutorial, *IEEE Communications Surveys and Tutorials* 10(2): 6–33.

Roy, N. & Das, S. K. (2009). Enhancing availability of grid computational services to ubiquitous computing applications, *IEEE Transactions on Parallel and Distributed Systems* 20(7): 953–967.

Saadi, R., Pierson, J. M. & Brunie, L. (2005). APC: access pass certificate distrust certification model for large access in pervasive environment, *Proc. of ICPS '05*, pp. 361 – 370.

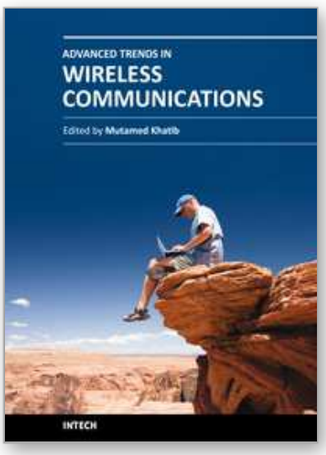
Schmidt, C. & Parashar, M. (2004). A peer-to-peer approach to web service discovery, *World Wide Web* 7(2): 211–229.

Sundararaj, A. I., Gupta, A. & Dinda, P. A. (2004). Dynamic topology adaptation of virtual networks of virtual machines, *Proc. of LCR'04*, Houston, TX, USA.

The Globus Security Team (2005). Globus toolkit version 4 grid security infrastructure: A standards perspective, *Technical report*, The Globus Alliance.
URL: *http://www.globus.org/toolkit/docs/4.0/security/GT4-GSI-Overview.pdf*

Vanneschi, M. & Veraldi, L. (2007). Dynamicity in distributed applications: issues, problems and the ASSIST approach, *Parallel Computing* 33(12): 822–845.

Zhang, S., Zhang, S., Chen, X. & Huo, X. (2010). Cloud computing research and development trend, *Proc. of ICFN'10*, Sanya, Hainan, China, pp. 93–97.



Advanced Trends in Wireless Communications

Edited by Dr. Mutamed Khatib

ISBN 978-953-307-183-1
Hard cover, 520 pages
Publisher InTech
Published online 17, February, 2011
Published in print edition February, 2011

Physical limitations on wireless communication channels impose huge challenges to reliable communication. Bandwidth limitations, propagation loss, noise and interference make the wireless channel a narrow pipe that does not readily accommodate rapid flow of data. Thus, researches aim to design systems that are suitable to operate in such channels, in order to have high performance quality of service. Also, the mobility of the communication systems requires further investigations to reduce the complexity and the power consumption of the receiver. This book aims to provide highlights of the current research in the field of wireless communications. The subjects discussed are very valuable to communication researchers rather than researchers in the wireless related areas. The book chapters cover a wide range of wireless communication topics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Romano Fantacci, Daniele Tarchi and Andrea Tassi (2011). Wireless Communication Protocols for Distributed Computing Environments, Advanced Trends in Wireless Communications, Dr. Mutamed Khatib (Ed.), ISBN: 978-953-307-183-1, InTech, Available from: http://www.intechopen.com/books/advanced-trends-in-wireless-communications/wireless-communication-protocols-for-distributed-computing-environments

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen