Dynamicity Aware Graph Relabeling Systems and the Constraint Based Synchronization: a Unifying Approach to Deal With Dynamic Networks

Arnaud Casteigts and Serge Chaumette

LaBRI, Université Bordeaux 1,
351 Cours de la Libération, F-33405 Talence, France.
{arnaud.casteigts,serge.chaumette}@labri.fr
http://www.labri.fr/

Abstract. Many research projects are being done in the domain of wireless mobile and/or ad-hoc networks to provide tools, algorithms and applications that make it possible to handle or use their dynamic characteristics. The purpose of most of these projects is to solve a specific problem, in a specific context. Most often, the models and formalisms that are used are also specific and the results are therefore difficult both to understand and to use in another context. We believe that what is needed is a general model offering a very high level of abstraction, in order to define and characterize what is feasable or not feasable in a dynamic network, depending upon some of its characteristics. In this paper we define such a model and its associated formalism. They are adapted to the study of dynamic networks and to the modeling of algorithms in a dynamic context at a high level of abstraction. The proposed model (Dynamicity Aware Graph Relabeling Systems) derives from what has been achieved in the area of local computations, and that produced useful results in the context of static networks. Our contribution comprises a model and the associated formalism, plus an original synchronization mode between nodes that allows to seemlessly adapt an algorithm to different mobility contexts. All the concepts are illustrated and discussed through the example of a document propagation algorithm, with a resume feature.

The work presented in this paper is carried out at LaBRI (Laboratoire Bordelais de Recherche en Informatique) and more precisely in the SOD (Distributed Systems and Objects) team. It is partly achieved within the framework of the Sarah (Asynchronous services for mobile ad-hoc networks) project supported by the ANR (Agence Nationale de la Recherche).

1 Introduction

The research activity in the area of wireless and pervasive networks keeps on growing. The emergence of numerous broadband wireless standards such as 802.11/16, bluetooth or 3G/UMTS that offer many new possibilities enforces the trend.

While many research projects focus on routing layers [9], or on a specific problem in a given network, we study these different kinds of networks from a theoretical point of view in terms of what basically makes them similar or different, and what can be done (or not) in each of them. This requires a well adapted model, which makes it possible to take into account the `dynamicity` of the network, and the `locality` of the communications. To understand this last point, let us examine more precisely one single device in such a network. The only thing this device can basically be aware of is its own state, and the state of its direct neighbors. The only thing it can be sure of is that, `at a given instant`, it can communicate with these neighbors, and that this communication link can have disappeared at the next instant, because one of the devices has moved or has been turned off.

Another property of the desired model (in addition to locality and dynamicity) is to be at a high level of abstraction. It should be possible to ignore physical implementation details when designing an algorithm.

Our first contribution is the **DA-GRS model** (Dynamicity Aware Graph Relabeling Systems). It has been inspired by the area of local computations, in which most of the desired properties

described above were considered, even though it addresses distributed algorithms in the context of static networks.

Our second contribution is what we call the **constraint based synchronization**. While local computations traditionally synchronize the nodes by using a rendezvous algorithm[8], we propose in this paper a new approach that allows to seamlessly adapt an algorithm to different contexts.

The rest of this paper is organized as follows. In section 2 we present local computations and the GRS formalism. In section 3 we present our contributions. We first explain the DA-GRS model, the grounds of which have been introduced in [4]. We then describe the rendez-vous synchronization algorithm used by local computations and the new synchronization mode that we propose, stressing the fact that it is well adapted to DA-GRS. In section 4 we illustrate the model and the synchronization mode by means of an example that consists in propagating a document within a dynamic ad-hoc network. This propagation supports a resume feature, so it can be used for the diffusion of a large, for instance multimedia, file. We eventualy conclude, present futur work and some other possible applications in section 5.

2 Related Work

The model that we propose in this paper (the DA-GRS model), is an extension of local computations to dynamic networks. This section briefly presents the area of local computations, and the precise model we derive from. The formalism that is used is that of the well known Graph Relabeling Systems.

Model. Local computations have been introduced as a technique to express and evaluate distributed algorithms on a network of communicating processors. The physical network is seen as a graph, the edges of which represent the communication links, and the vertices (or nodes) represent the computing resources. Local computation models are very "powerful" theoretical models in the sense that a problem that has no solution with local computations has no solution with any other model (the opposite being false). This property is due to the fact that the communication model (message passing, mail boxes, shared memory, etc..) is abstracted. A basic computation step is defined by a couple (pre-condition, post-condition) that modifies the states of the involved nodes and/or edges according to their previous states, and without considerations about the way they actually do it.

Local computation models do not require any assumption about the network such as the existence of unique identities for nodes, topology knowledge, etc. On the contrary it is often used to characterize which assumptions have to be done in order to solve a given problem. Local computations also provide a way to express distributed algorithms in an elegant way, and help to prove their correctness.

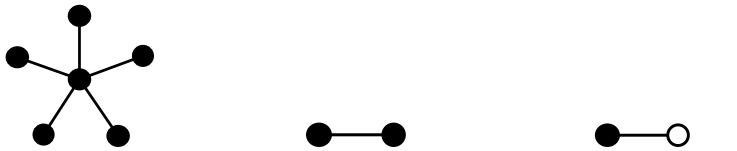
Regarding the properties that we stated to be important in the introduction, the most interesting one for dynamic networks is **locality**, that matches the reality of the network. Local computations only consider communications between nodes that are direct neighbors.

Many local computation models exist, from the most powerful - where the evolution of a node in one step depends on the states of its neighboring ball of radius 1, and can impact on the state of the whole ball (figure 1(a)) - to the weakest - where the evolution of a node in one step depends on its state and on the state of one of its neighbors, and impacts only its own state (figure 1(c)). We have chosen to use the model illustrated on figure 1(b), which we believe is the most realistic in a dynamic context, because of the difficulty to synchronize more than two nodes that can move at any time.

The reader is further referred to [2], [6] and [10] for the grounding work on local computation, respectively by Angluin and by Yamashita and Kameda. A classification of local computation models can be found in [5].

Formalism. Graph Relabeling Systems (GRS) are one of the formalisms that can be used to represent local computations. With this formalism, the states of vertices and edges are represented by labels, and a computation step is represented by a relabeling rule.

A basic and intuitive example, as described in [7], is the modelization of a spanning tree algorithm. In this example, each node has its label in $\{I, N\}$ that represents the fact that it is

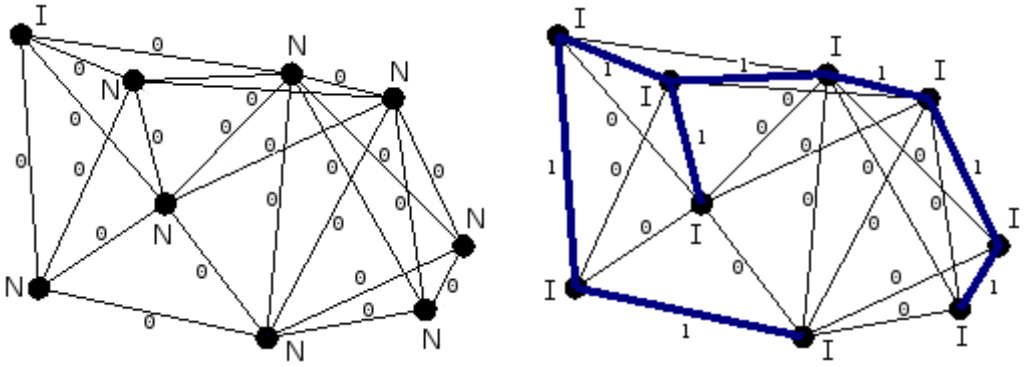


(a) Most powerful (b) Model used by DA-GRS (c) Less powerful

Fig. 1. Local computation models

Integrated in the tree or *Not*. An edge belongs to the tree if it is labeled 1. Initially, all edges are labeled 0 and all nodes are labeled N except one, the root, which is labeled I. The tree is then built by repeatedly appling the following simple rule:

Such an execution will stop when no rule is applicable anymore. Figure 2 shows an execution example of this algorithm in an arbitrary network.



(a) Initial state (b) A possible final state

Fig. 2. Spanning tree algorithm (multiple applications of rule R1)

Note that the tree is computed in a finite number of relabeling steps (depending on the diameter of the graph), and all the computation steps are done locally.

3 The DA-GRS Model and its Synchronization Mode

The local computation model presented in the previous section deals with static graphs, where $V(G)$ and $E(G)$ do not change during computation. In order to use a similar approach in a dynamic context, this model has to be adapted. This is what we have done by defining the DA-GRS model. In addition, while a rendez-vous is used to synchronize two nodes before they work together in a static network, we propose another synchronization mode that we call *constraint based synchronization* that enables to adapt an algorithm to different target networks, without modifying the algorithm itself.

3.1 The DA-GRS Model

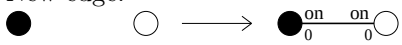
This section presents the main adaptations that have been done based on local computations and GRS. First of all, the graphs that are now delt with are dynamic. In order to be able to target

any kind of network, this dynamicity has no restriction, i.e. $V(G)$ and $E(G)$ can both change at any time. From a strictly local point of view, all the topological changes are seen by nodes as appearance/disappearance of neighbors and links. These are the events that have to be handled. The DA-GRS model therefore adds a special label on each edge, that represents the availability of the communication link. Now, an edge has two labels on both sides (on both sides because it represents the fact that information about edges are stored in the nodes themselves). These two labels are: *value* and *activity*. The *value* label is the usual state of the edge, while the *activity* label represents the fact that the link is available or has been broken (on/off). When a communication link between two nodes is broken, two cases are considered:

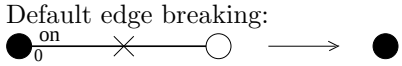
- if the label "*value*" of the edge is still to its default value noted 0, it means that the algorithm does not bother about this edge. The edge thus simply disappears.
- if the label "*value*" label of the edge has been set to something different from its default value (here noted 1), then the *activity* label takes the value off, what will allow a node to apply a specific rule to react to the loss of a communication link.

Here are the corresponding internal actions, as seen by the black node:

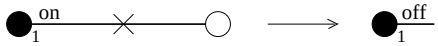
- New edge:



- Default edge breaking:



- Special edge breaking:



Note that these operations are not part of any algorithm. They are intrinsic reactions of the model to topology changes and make it possible for an algorithm to handle dynamicity.

3.2 The Rendezvous Algorithm

The rendezvous algorithm used in a static context, ensures a fair distribution of the computation steps between the nodes. In this synchronization algorithm, each node iterates through the following actions:

- *select one of my neighbors* $c(v)$ *at random*
- *send* 1 *to* $c(v)$
- *send* 0 *to the other neighbors*
- *receive a message from each of my neighbors*
- *there is a rendezvous with* $c(v)$
  if I receive 1 *from it.*

When there is a rendezvous between two nodes, they try to apply a rule if possible. More information about the rendezvous algorithm can be found in [8].

3.3 The Constraint Based Synchronization

Intuitive Idea. In many cases, the management of dynamicity must be adapted to the context (pedestrians, vehicles on the road, researchers in a conference room, sensors with weak mobility, etc.) and to the type of application that is being considered (propagation of information, construction of a global structure for topology control, music sharing, etc). From a purely local point of view, this adaptation can be carried out by giving a higher priority to the interactions satisfying certain criteria, which we call *constraints*. We propose to manage these constraints at the synchronization level, replacing the rendezvous algorithm presented above in order to relieve the algorithm from these considerations.

Architecture. The resulting global model is shown figure 3.

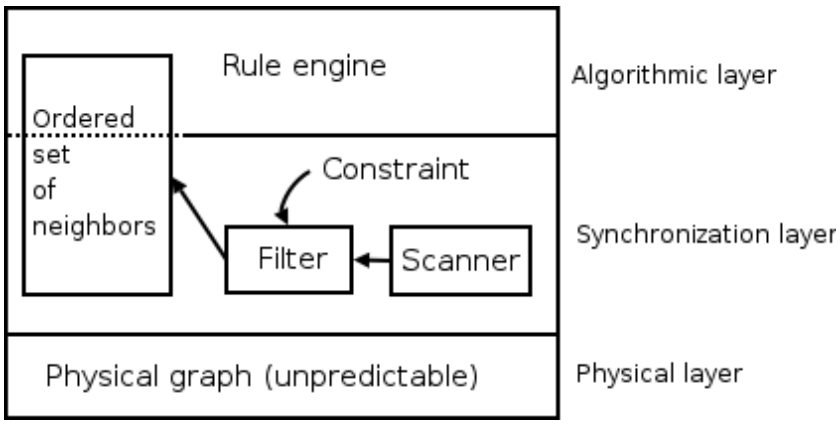


Fig. 3. The multi-layer model

Explanation. At regular intervals, the network *scanner* of each node builds a set of reachable neighbors that are detected. This set is then ordered by the *filter*, according to a given constraint (see below). Once the set of neighbors has been ordered, the top level layer, i.e. the DA-GRS rule engine, uses it as a replacement for the rendezvous algorithm. Instead of choosing a neighbor at random, the rule engine tries to apply a rule with each neighbor in the order given by the constructed set until success. This mechanism ensures that the communication links that match the constraint are favored.

Discussion About the Constraint. The pieces of information that a node can have about a communication link with a neighbor can be of different natures. This could be for example the time since it is available, the strength or the variation of the (radio) signal. These pieces of information, possibly combined, make it possible to give priority to a communication that would for instance be *stable*, *weak*, *near-to-break*, *with a good bandwidth*, etc...

4 Illustration

The example presented in this section illustrates the formalism, the model and the use of the constraint based synchronization. We first present the algorithm and give explanation about DA-GRS. We then discuss it and show how it can be adapted to different target networks without any modification.

4.1 The DA-GRS Propagation Algorithm

The example that we use is an algorithm the aim of which is to propagate a possibly large document in a dynamic network. To receive the document, a node selects a source in its neighbourhood and tries to keep on working with this source. If this is not possible (because of topological changes for instance) it will go on working (resume) with any other node that has some of the parts of the document that it misses, and so on. Note that all parts of the document are thus received in an ordered manner.

States of the Nodes. Each node has two labels representing its state ($main$ and $offset$). The possible values of the state $main$ are the following:

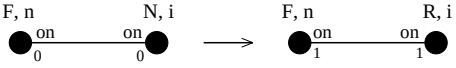
- F: The node has the whole document, and does not try to get it anymore.
- N: The node does not have the whole document, and it wants to connect to another node to get the parts that it misses.
- R: The node is currently receiving parts of the document from a connected neighbor.

The value of the state $offset$ is the number of the last part of the document that the node has received. The document is split into n pieces.

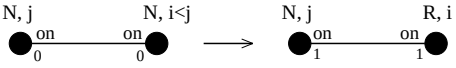
States of the Edges. Two connected nodes have their common edge locally labeled 1. The other edges are labeled 0.

The algorithm.

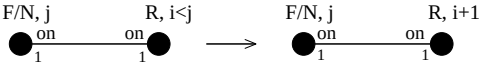
When a node labeled F meets a node labeled N, a connection start between them:

R1: 

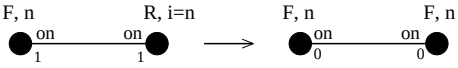
When a node labeled N meets another node labeled N, and their $offset$ are different, a connection starts between them:

R2: 

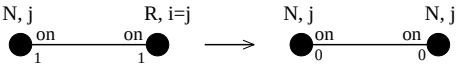
The following rule models the transfer of a block between two nodes. (F/N means that the rule applies the same way for nodes labeled F or N):

R3: 

During the connection, if the node labeled R has received the last block of the document then the transfer is finished, it becomes labeled F and the connection is closed:

R4: 

During the connection, if the node labeled R has received the last block the node labeled N can give, then the connection is closed, and it returns in the waiting state:

R5: 

If the link between two connected nodes breaks, then the receiving node returns to the waiting state:

R6: 

and the sending node cleans its state by removing the edge:

R7: 

Note that the algorithm is able to react to topological changes thanks to the dynamicity management that we have introduced in the DA-GRS model.

4.2 Adaptation to Different Target Networks

As explained above, we use a constraint to order the neighbors of a node. The chosen constraint impacts on which neighbor nodes will preferably work together. This enables to take into account some characteristics of the context in which the algorithm is executed. For the example of document propagation presented above, we can distinguish several contexts to which we would like to adapt the algorithm.

First context: *little mobility and large messages.* If the nodes are hardly mobile and the data to transfer are large (e.g. people in a room who exchange multimedia files), then we would like to give priority to the most stable communication links. The chosen constraint is then the *Minimum signal variation*.

Second context: *strong mobility and short messages.* If the nodes are very mobile and the messages are short (e.g. cars on a road that propagate information relative to an accident), it may be worth giving priority to the communication links that are the more volatile, in order to inform a maximum number of nodes (cars that are going away). The chosen constraint is then the *Maximum signal variation*.

Third context: *big covering and short messages.* If the aim of the propagation is to inform the farthest node or to cover the largest area in the smallest amount of time, the constraint would be the *Less powerful signal*.

Fourth context: *big bandwidth.* If the application needs the biggest available bandwidth, the favored communications would be those which offer the strongest signal. The constraint is then the *Most powerful signal*.

An algorithm can thus be studied and possibly simulated in various contexts, without being modified.

5 Conclusion

In this paper, we have presented our contribution to the study of wireless, dynamic and mobile network algorithms. We have introduced the DA-GRS model and the constraint based rendezvous. The DA-GRS model, allows to work on general issues of dynamic networks, while keeping control of dynamicity. Its main advantages are its ability to take into account the locality of the communication, the dynamicity of the topology in an innate way, and to provide an abstraction of the communication layer.

The example that we have shown has been designed to illustrate the model and the fact that an algorithm could be quite easily described with this formalism. We have focused on how the *constraint based synchronization* mechanism makes it possible to take into account different contexts of dynamicity, each using a set of constraints to impact the behavior of the algorithm. We have shown that the adaptation from one context to another is straightforward and is done without any modification of the algorithm itself. Concerning the practical validation of the model, we have developed a free software simulator (under the terms of the GPL) that allows to run algorithms designed with the DA-GRS formalism. This simulator can be found at [1].

The research on the DA-GRS model are quite recent, and it has opened many potential directions. First of all, we plan to use it to classify dynamic networks, in terms of what kinds of problems can be solved, in which network classes, and with which asumptions. The problems we plan to study in a first step are classical theoretical problems, such as election, naming or enumeration. From a more practical point of view, we are also studying the model driven capabilities of the DA-GRS [3] in terms of code and interface generations from rules. This will be used along with an embedded rule engine.

References

1. SimuDAGRS, a Dynamic Network simulator for algorithms modeled by DA-GRS. available at http://www.labri.fr/~casteigt/simulator.html, 2005.
2. D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on theory of computing*, pages 82–93, 1980.
3. A. Casteigts. Model driven capabilities of the da-grs model. In *Proceedings of the 1st IEEE workshop on self-adaptability and self-management of context-aware systems (SELF'06)*.
4. A. Casteigts and S. Chaumette. Dynamicity Aware Graph Relabeling Systems - a model to describe MANet algorithms. In *Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2005.
5. J. Chalopin, Y. Metivier, and W. Zielonka. Election, naming and cellular edge local computations. In *second international conference on graph transformation*, pages 242–256. Lectures Notes in Computer Science, 2004.
6. T. Kameda and M. Yamashita. Characterizing the solvable cases. In *Computing on anonymous networks*, pages 69–89. IEEE Transactions on parallel and distributed systems, 7, 1996.
7. I. Litovsky, Y. Metivier, and E. Sopena. Graph relabelling systems and distributed algorithms. In World Scientific Publishing, editor, *Handbook of graph grammars and computing by graph transformation,*, volume Vol. III, Eds. H. Ehrig, H.J. Kreowski, U. Montanari and G. Rozenberg,., pages 1–56, 1999.
8. Y. Metivier, N. Saheb, and A. Zemmari. Analysis of a randomized rendezvous algorithm. *Inf. Comput.*, 184(1):109–128, 2003.
9. E.M. Royer and C.K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. In *Personnal Communications, IEEE*, pages 46–55, 1999.
10. T. Kameda and M. Yamashita. Decision and membership problems. In *Computing on anonymous networks*, pages 90–96. IEEE Transactions on parallel and distributed systems, 7, 1996.