



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team NETQUEST

Declarative Network Programming

Paris - Rocquencourt

Theme : Distributed Systems and Services

Activity
R *eport*

2010

Table of contents

1. Team	1
2. Overall Objectives	1
3. Scientific Foundations	2
3.1. Introduction	2
3.2. Logic and query languages	3
3.3. Distributed computation and algorithms	3
3.4. Protocols	4
3.5. Verification	4
4. Application Domains	5
4.1. Spatial information	5
4.2. Rapid prototyping of network protocols	5
4.3. Pervasive adaptation	5
4.4. Heterogeneous networks	5
5. Software	6
5.1. Overview	6
5.2. Netquest system	6
5.3. Netlog development environment	7
5.4. Simulation & Visualization platform, Netqvis	7
6. New Results	8
6.1. Logic and query languages	8
6.2. Distributed algorithms	9
6.3. Protocols	9
6.4. Verification	9
7. Contracts and Grants with Industry	9
8. Other Grants and Activities	9
8.1. Actions in France	9
8.2. Actions in China	10
8.3. Actions in Europe	10
9. Dissemination	10
9.1. Service	10
9.2. PhD Jurys	10
9.3. Internships	10
9.4. Short-term visitors	10
10. Bibliography	10

NETQUEST is an Inria international project-team, located in the Sino-French IT Lab, LIAMA in Beijing, China, and attached to the Rocquencourt Unit.

1. Team

Research Scientist

Stéphane Grumbach [Team leader, Senior researcher, INRIA]

PhD Students

Fang Wang [PhD, ISCAS, 2005-2010]

Wenwu Qu [PhD, ISCAS, 2005-2010]

Kun Suo [PhD, CASIA, 2008-2011]

Ahmad Ahmand-Kassem [PhD, INSA, 2009-2012]

Fuda Ma [PhD, INSA, 2010-2014]

Post-Doctoral Fellow

Xiang Zhou [Post doc, CASIA, 2009-2011]

Others

Xin Qi [Master, CASIA, 2007-2010]

Lin Tang [Assistant, LIAMA]

2. Overall Objectives

2.1. Overall Objectives

Networks of independent entities, cooperating to handle global tasks, constitute a fascinating class of systems. They are widely found in nature, with cells exchanging information with their neighbors, or neurons through synapse connections, as well as in social organizations. Such organizations are making their ways in microelectronic systems and might in the near future become ubiquitous. They are made possible by the development of ever smaller and cheaper electronic devices with increased memory capacity and computational power, together with the standardization efforts for both wireless communication and data exchange format. The Internet of things, in which potentially all objects, whether virtual or real, will become addressable and smart, attracts now a considerable attention.

One of the main barriers today to the development of such networks is the **lack of programming abstraction**. Smart devices are usually dedicated systems based on ad hoc models, which are not generic enough to support the needs of future applications (flexibility, scalability, ease to maintain, etc.). The deployment of a sensor network for instance is a tedious task which requires an expertise in the underlying OS and hardware. Applications implemented today on top of TinyOS for instance impose to deal with low-level issues such as memory.

The objectives of the Netquest project are to develop solutions that allow to program networks in a declarative manner, by specifying the intended functionalities without having to deal with system aspects, much as in database systems. The separation of a **logical level**, accessible to users and applications, from the **physical layers** constitutes the basic principle of Database Management Systems. It is at the origin of their technological and commercial success. This fundamental contribution of Codd in the design of the relational model of data, has lead to the development of universal high level query languages, that all vendors recognize, as well as to query processing techniques that optimize the declarative queries into (close to) optimal execution plans.

Two levels of abstraction can be distinguished in distributed systems: (i) the global level, where the network can be programmed as a whole, and (ii) the local level, where programs specify the nodes behavior. At the global level, a node can for instance fire a query asking for a route satisfying some properties, without any algorithmic specification. The network will rely on the distributed query engine on each node to evaluate that query, thus resulting in a distributed algorithm. At the local level of abstraction on the other hand, the behavior of the nodes can be specified. Routing protocols for instance are programmed at the local level by specifying the behavior of nodes as the exchange of messages used to achieve a task.

The global level of abstraction relies on a very simple idea: **model the network as a database**. The network is thus essentially hidden and perceived by each node as a database, with which it interacts through declarative query languages. The communication between devices thus consists of queries and data. Each node should be equipped with a **distributed query engine**, evaluating all queries whether posed by the node itself or received from other nodes. We consider classical query languages and their expressive power and complexity in this distributed framework, as well as new languages, with primitives relevant for distributed applications, such as aggregation or non-deterministic constructs.

For the local level, we consider rule-based languages, à la Datalog, which are well adapted to express the behavior of the nodes, as actions to perform under some events. Classical rule-based languages can be extended with communication primitives, as well as other primitives required by distributed applications. This approach known as **declarative networking** has already shown promising for the conciseness of the code. It blurs the traditional distinction between communication and application layers. Both are handled in a uniform fashion.

The approach we pursue is data centric, the languages we consider for both levels of abstraction, express transformations over the data which are stored on the nodes, whether they relate to applications or to the network itself. These data can be seen globally in an abstract way by ignoring their physical location in the network, or on the contrary by taking into account their distribution. For most of our languages, we considered an implementation based on SQL and embedded DBMS on the nodes. This data centric approach allows to define data structures (e.g. trees, tables, etc.) for which it is easy to state and verify global properties (e.g. loopfree).

Our ambition is to show that recursive rule based languages help to develop programs which are:

- easy to write in a concise way;
- efficient to execute, that is they can be compiled into efficient distributed algorithms, which can adapt to dynamic environment;
- verifiable thanks to a clear semantics;
- portable over heterogeneous devices and networks.

The objective of the project is to solve the **theoretical and practical problems** raised by the programming abstraction. We concentrate on the following four research directions:

- Establish **theoretical foundations** for network query languages: design, distributed complexity, and expressive power.
- Implement **distributed query engines** to execute queries of the network query languages, with distributed optimization.
- Develop formal models to allow the **verification** of the declarative protocols, using proof assistants.
- Validate the declarative approach through **real network problems**, such as networking protocols, and distributed network applications (sensor, vehicles, M2M, etc.).

3. Scientific Foundations

3.1. Introduction

The scientific foundations combine techniques from different fields. First, the field of databases, both its theoretical background, with the theory of query languages, as well as system issues, such as query processing. Distributed computing, algorithmic complexity and impossibility results relate to the meta-theorem we are targeting. Graph theory, and the developments in graph relabeling are interesting for high-level abstraction. Formal models and verification methods for network protocols are used for declarative protocols. Application fields on networks, such as networking protocols, wireless networks, sensor or vehicular networks constitute fundamental validation areas.

3.2. Logic and query languages

Logical formalisms, such as first-order logic (FO), fixed-point logic (FP), and monadic second-order logic (MSO) for instance, allow to express problems in a declarative way. Instead of describing how to compute problems step by step, only the desired results of the computation are specified by logical expressions. The use of declarative query languages based on logical formalisms for data management was largely exploited by Codd in the 1970's [32] for the relational model which offers a separation between the logical and the physical levels. Since then, the investigation of the theoretical foundations of query languages has been a strong focus of the database theory community.

Two important measures characterize query languages: their expressive power and their complexity. Given a query language, deciding which problem can be expressed in this language characterizes its expressive power. How complex it is to compute the queries in a given query language, characterizes its computational complexity.

On general finite structures, the expressive power and sequential computational complexity of classic logics have been intensively studied [35]. Recently similar questions have been investigated on various restricted classes of graphs [39] [43]. It has been shown that MSO can be evaluated in linear-time over bounded tree-width graphs [33], and FO can be evaluated in linear-time over bounded degree graphs [60] and planar graphs [37], and FO definable optimization problems on classes of graphs with excluded minors can be approximated in polynomial time to any given approximation ratio [34].

Although classical query languages have been intensively studied in the context of sequential computation, their distributed computation on graphs has attracted only little attention. We consider the expressive power of classical query languages for describing distributed data structures, such as spanning trees, routing tables, and dominating sets, etc., and study their distributed complexity. We also propose to introduce new primitives into classical query languages to design proper logical formalisms for multihop networking (global level abstraction), while achieving a nice balance between expressive power and distributed computational complexity.

For the local level of abstraction, recursive rule languages (variants of Datalog) have been used to describe communication protocols [46] [48], thus reviving the recursive languages developed in the 80's for deductive databases [26] [27] [62], well-suited to define routes in networks. Query languages allow the expression of protocols, one or two orders of magnitude simpler than classical imperative programming languages. We continue this trend to demonstrate the potential of declarative rule languages for the local abstraction level, clarifying their semantics in asynchronous distributed computation, investigating further their expressive power and the complexity of their distributed evaluation. The definition of a rule language which admits efficient distributed execution while offering enough expressive power is still an active topic of research [40].

3.3. Distributed computation and algorithms

Peer-to-peer networks, wireless ad hoc and sensor networks, or even the Internet form loosely coupled distributed systems, in which there is no centralized control and nodes communicate with each other by exchanging messages. We adopt the classical *message passing model* [25] [50], which formalizes networks with independent entities communicating with one another.

In the *message passing model* of distributed computation, the network is modeled as an undirected graph $G = (V, E)$, in which each vertex $v \in V$ represents a node (host, device, processor, ...) of the network. Two nodes $u, v \in V$ are adjacent if and only if there is a bidirectional communication channel between them. Moreover, due to degrees of synchrony, there are two extreme message passing models widely investigated: the fully synchronous one and the totally asynchronous one.

In large-scale networked systems, every node can only directly communicate with a limited number of neighboring nodes, and no node is typically able to collect global information from the entire network, because gathering information from the whole network topology is either too resource consuming or simply impossible due to mobility, dynamism, or churn. In spite of these inherent limitations to *local information*, networks are expected to finish some kinds of *global tasks*, e.g. spanning tree construction, coloring, etc. The expressive power of various forms of *local computation*, i.e. the question to what degree local information is sufficient to solve global tasks [54], has attracted a lot of attention and is of particular interest in the context of the problems definable by logical formalisms in a distributed environment.

In wireless ad hoc and sensor networks, nodes are typically severely constrained, with limited CPU, memory, energy etc. As a consequence, protocols running on these devices should run as fast as possible and require as little communication as possible. In this context, local distributed algorithms [23] [44] [45] [52] [61] are of special interest. A distributed algorithm is local, if its running time and performance guarantees are independent of the number of nodes in the network. We consider generalization of these definitions to communications bounded but not necessarily local, resulting in the class of frugal computations.

3.4. Protocols

Networking is made possible by fundamental network protocols. They provide basic services (e.g. routing) through constructing and maintaining distributed data structures, such as spanning tree, shortest path, dominating set, etc. Many networking protocols have been developed, including various routing protocols, such as DSDV (Destination-Sequenced Distance Vector Routing) [56], OLSR (Optimized Link-State Routing) [42], AODV (Ad hoc On-demand Distance Vector Routing) [57], and VRR (Virtual Ring Routing) [29], as well as self-configuration and self-organization protocols, such as ASCNET [30] or FISCO [49].

Although network protocols are crucial for networking, their design is a complex and error prone task, when constraints (e.g. mobility, energy efficiency, etc.) have to be taken into account. One of the main challenges originates from the inherent complexity of developing correct program codes for network protocols.

Since protocols usually describe the behavior of nodes under events, such as messages received, they can be easily written in rule-based languages. Our objective is to show that the declarative specification of particular network protocols, can lead to efficient behavior. Thus implementing protocols in declarative rule languages can reduce the inherent complexity of programming for network protocols as well as other distributed application.

3.5. Verification

Having correct and robust protocols is fundamental for critical distributed systems. Ensuring the desirable properties of protocols is a very difficult problem. Neither simulations nor testbed implementations can ensure the quality required for network protocols. As an alternative to these methods, some researchers have successfully investigated the use of formal verification as a mean to guarantee the quality of protocols [28] [36] [51] [59].

Formal verification is a technique that assures whether a system enjoys a given property, based on a formal model of the system under evaluation. There are roughly two approaches to formal verification. The first approach is model checking [31], which consists of an exhaustive exploration of all states and transitions in the formal model of the system. A lot of model checking tools have been developed, such as SPIN [41], UPPAAL [20], PRISM [19], etc. The second approach is logical inference. It relies on a formal mathematical model for reasoning about the system, usually using theorem proving software such as the HOL [18], or the Coq [17]. This is usually only partially automated and is driven by the user's understanding of the system to validate.

We have pursued the second approach, by using the Coq proof assistant for protocols expressed in rule languages. This requires the modeling in Coq of the distributed systems, the machine evaluating the rule programs, as well as the theory of interest for the class of protocols considered.

4. Application Domains

4.1. Spatial information

Applications of ubiquitous networks are emerging in many areas such as intelligent transportation, games, social networking, sensor networks, ambient intelligence, etc. We have considered widely spatial information systems in the past. Their interaction with networks is of great interest to support queries relating to the ambient space and positioning issues. Distributed in spatial environment of different scale (e.g. building, landscape) sensor networks constitute a promising application to validate the Netquest approach.

4.2. Rapid prototyping of network protocols

The Netquest approach provides a (global-level or local-level) programming abstraction that allows network protocol designers to program their protocols in a declarative way. The Netquest system is responsible for transforming these protocols into low-level code(queries and messages) and executing them. The computation of protocols in Netquest can be monitored using the network simulator WSNNet as well as a simple network emulator, and visualized, by a visualization tool developed in the group, showing the network activity as well as the evolution of the databases in the nodes. Compared to the implementation of protocols in imperative programming languages, the declarative specification can be two order of magnitude shorter [47]. More generally, Netquest offers an environment which simplifies the design of protocols by relying on the DBMS for fundamental aspects such as transactions.

4.3. Pervasive adaptation

In the absence of sufficient or sufficiently accurate knowledge, adaptive methods have been developed for query execution, that allow to alternate query processing with query execution phases. This was introduced long ago in system R [24] for situations where the statistics would be misleading or incomplete. This is now a topic of increasing interest with the development of applications running over data distributed over networks. Adaptation is the key challenge for ubiquitous networks. More generally, the capacity to self-assemble, grow, repair, organize, evolve over long period of time while maintaining essential functionalities is of fundamental importance for networks of cooperating objects. The combination of networking and application layers, jointly processed by distributed query engines, offers a huge potential for pervasive adaptation, because the query engine can adapt the queries to the network (adaptive evaluation) and the network to the queries (Quality of service and content based routing).

4.4. Heterogeneous networks

Heterogeneous networks, in which the node architecture, operating system, data format, etc. might vary significantly, pose additional challenges to network management and applications. Netquest offers a high level abstraction which allows to specify an application or a protocol independently of the underlying architecture. The Netquest system can run on any type of devices assuming nodes are not too constrained and are equipped with a (local) DBMS. The Netquest approach will be used to test network management and data-centric applications in heterogeneous networks.

5. Software

5.1. Overview

The Netlog language is a rule based language for network programming. To support the development of this language, we have implemented the following components:

- the Netquest system, a virtual machine to evaluate Netlog programs;
- the Netlog compiler, integrated in a Netlog programming environment, which optimizes programs, and compiles them as SQL queries; and
- the simulation & visualization platform, Netqvis, used to monitor the execution of Netlog programs over small networks.

5.2. Netquest system

Participants: Stéphane Grumbach, Xin Qi, Kun Suo.

The Netquest system is the runtime environment of Netlog. The system relies on an embedded DBMS, e.g. SQLite, MySQL, which handles the storage of all the local data as well as the evaluation of SQL queries, corresponding to Netlog programs. The use of an embedded DBMS offers several advantages: (i) it simplifies the code of the Netquest system by making use of standard functionalities offered by the DBMS, such as persistence, transaction management, etc.; (ii) it supports data intensive applications; and (iii) it makes the system portable over heterogeneous architectures, as soon as they support embedded DBMSs.

The architecture of the Netquest system (Figure 1) consists of a Router, a Local Database, a Timer Manager, and a Deductive Engine. The Router is the interface of the Netquest system, with the communication facility of the node, and so the network. This module handles the reception, emission, and forwarding of messages. The Local Database handles all data local of the node, such as application data, network data, as well as Netlog programs coded as SQL queries. The Timer Manager manages all the timers in the system, which allows Netlog programs to have periodic behavior. The Deductive Engine handles the evaluation of Netlog programs, by computing their local fixpoint in a bottom-up way.

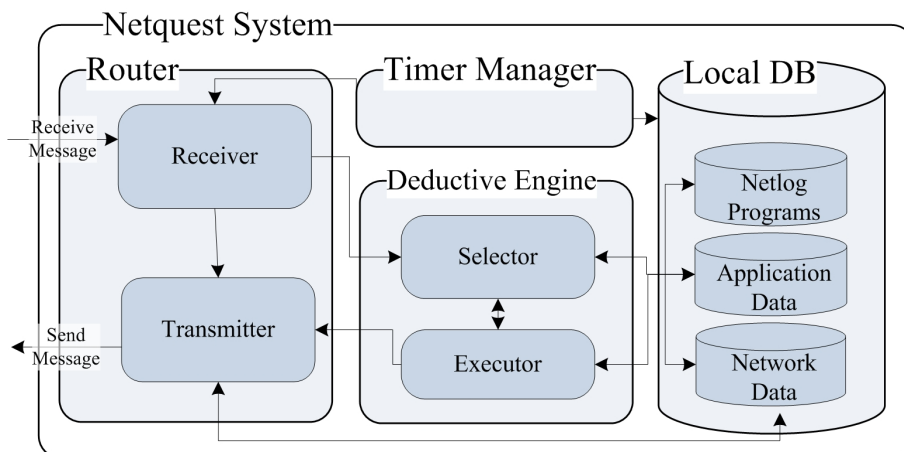


Figure 1. The architecture of Netquest system

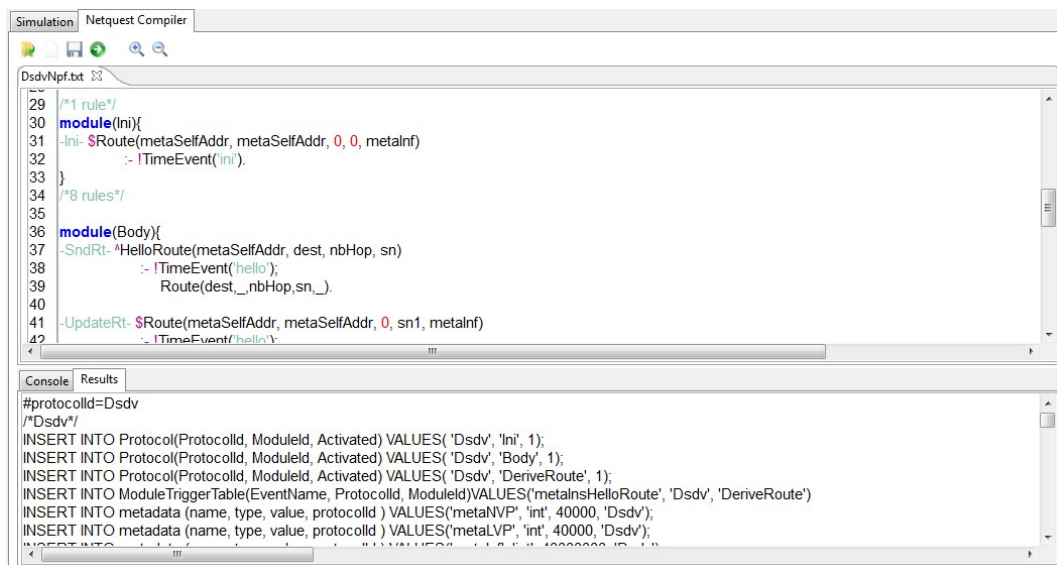
The Netquest system has been ported on two network simulators, WSNNet [21], and Netqvis, as well as a testbed of iMote devices [7].

5.3. Netlog development environment

Participants: Stéphane Grumbach, Xin Qi, Kun Suo.

The Netlog compiler transforms netlog programs into SQL queries which can be executed by the Netquest system. The Netlog IDE is composed of an editor, a parser, a code generator, and a packager. The editor allows the users to easily compose or edit a Netlog program with syntax high lighting and basic syntax check, as shown in Figure 2. The parser is generated by a parser generator for the Java programming language, JavaCC, which is similar to Yacc and generates a parser from a formal grammar written in EBNF notation. The code generator accepts the output of the parser, a syntax tree, as input, from which it generates SQL queries according to the semantics of Netlog programs.

Each rule in a Netlog program is translated into one or more SQL queries. Each SQL query carries one operation, such as push, store or delete, from the rule, when the condition specified in this rule body is satisfied. All the SQL queries translated from one Netlog rule are treated as an atom, either all of them are evaluated or none of them are. The packager packages the SQL queries produced by the code generator into a SQL script which can be executed by the embedded DBMS to store the SQL queries into their corresponding tables.



```

Simulation Netquest Compiler
DsdvNpf.txt
29 /*1 rule*/
30 module(Ini){
31   -ini- $Route(metaSelfAddr, metaSelfAddr, 0, 0, metaInf)
32     :- !TimeEvent('ini').
33 }
34 /*8 rules*/
35
36 module(Body){
37   -SndRt- ^HelloRoute(metaSelfAddr, dest, nbHop, sn)
38     :- !TimeEvent('hello'),
39       Route(dest, nbHop, sn, _).
40
41   -UpdateRt- $Route(metaSelfAddr, metaSelfAddr, 0, sn1, metaInf)
42     :- !TimeEvent('hello').
43 }
Console Results
#protocolId=Dsdv
/*Dsdv*/
INSERT INTO Protocol(ProtocolId, ModuleId, Activated) VALUES( 'Dsdv', 'Ini', 1);
INSERT INTO Protocol(ProtocolId, ModuleId, Activated) VALUES( 'Dsdv', 'Body', 1);
INSERT INTO Protocol(ProtocolId, ModuleId, Activated) VALUES( 'Dsdv', 'DeriveRoute', 1);
INSERT INTO ModuleTriggerTable(EventName, ProtocolId, ModuleId)VALUES('metaInHelloRoute', 'Dsdv', 'DeriveRoute')
INSERT INTO metadata (name, type, value, protocolId ) VALUES('metaNVP', 'int', 40000, 'Dsdv');
INSERT INTO metadata (name, type, value, protocolId ) VALUES('metaLVP', 'int', 40000, 'Dsdv');

```

Figure 2. The editor of Netlog development environment

Since different DBMS supports different SQL dialects, the Netlog compiler can generate code customized for different DBMS. Currently, MySQL and SQLServer are supported.

5.4. Simulation & Visualization platform, Netqvis

Participants: Stéphane Grumbach, Xin Qi, Kun Suo.

Netqvis is a simple network simulator allowing users to simulate, monitor, and visualize the behavior of Netlog programs on small networks. During the simulation, the topology and the activity of the simulated network are displayed on the GUI of Netqvis. Besides the network information, both the content of the local database of each node and the performance information of the running Netlog program can be visualized on the GUI.

All these tools allow the users to rapidly check the behavior and the performance of Netlog programs. Netqvis allows in particular to color specific parts of the network, such as all the edges of a route stored in the local DBMS on one node, as shown in Fig 3. Besides generating random topology and loading a topology from a file, Netqvis allows the user to modify the topology, the number of nodes, their position as well as their communication ranges, directly on the GUI when the simulation is running.

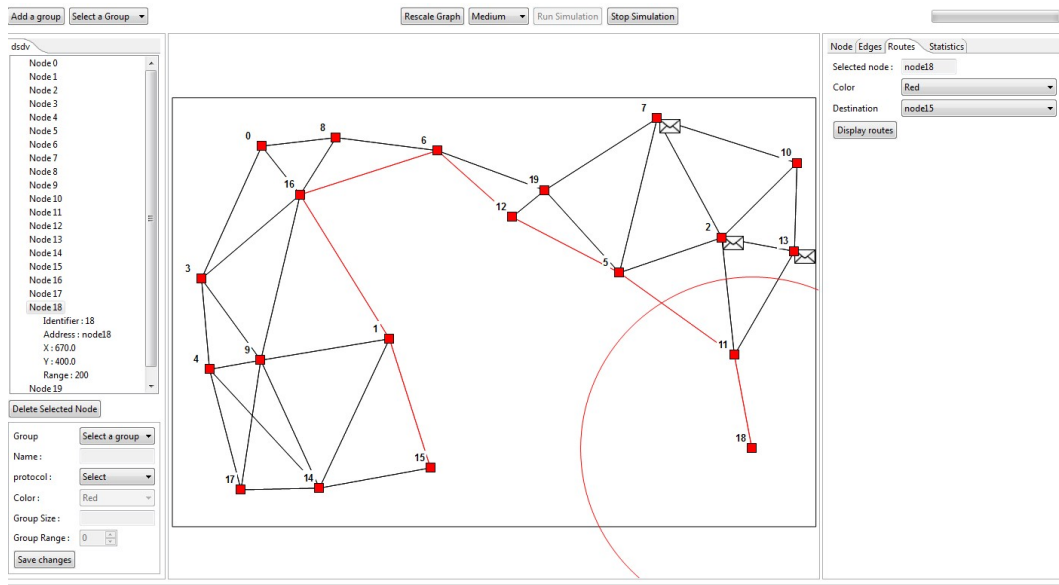


Figure 3. the GUI of Netqvis

Two versions of this platform have been implemented: in C# for Windows; and in Java, OS independent.

6. New Results

6.1. Logic and query languages

Participants: Ahmad Ahmand-Kassem, Stéphane Grumbach, Fang Wang.

For the global abstraction, we have obtained results on the distributed complexity of first-order (FO), Fixpoint (FP) and monadic second order (MSO) logic on various classes of graphs [15]:

- We showed that first-order properties can be frugally evaluated, that is, with only a bounded number of messages, of size logarithmic in the number of nodes, sent over each link, over bounded degree networks as well as planar networks. Moreover, we show that the result carries over for the extension of first-order logic with unary counting. These results relate the locality of the logic, in the sense of Gaifman [38] to the complexity of the distributed computation in terms of the number of messages handled on each node, which can be shown to be constant, a property weaker but which resembles the locality of distributed computations.
- In [9], we considered monadic second order logic, and showed that MSO can be evaluated in distributed linear time with only a constant number of messages sent over each link for planar networks with bounded diameter, as well as for networks with bounded degree and bounded tree-length. The distributed algorithms rely on the translation of MSO sentences into finite automata

over trees, and on nontrivial transformations of linear time sequential algorithms for the tree decomposition of bounded tree-width graphs.

For the local level of abstraction, expressing the nodes behavior, we have designed a rule-based language, Netlog [8][14], which extends Datalog [58] with aggregation and non-deterministic constructs and communication primitives. We defined a sound distributed fixpoint semantics for this language, which takes explicitly into account the in-node behavior as well as the communication between nodes. This language has been shown to be suitable to express a large collection of classical networking protocols. We have adapted the semi-naive bottom-up algorithm [22] for Datalog to evaluate Netlog and implemented it in the Netquest system.

6.2. Distributed algorithms

Participant: Stéphane Grumbach.

We considered *motion planning on directed graphs*, a problem related to data motion in a network with constrained node capacity and unidirectional links, which is an abstraction of the structure of wireless sensor networks. We proposed algorithms for solving the feasibility of motion planning on acyclic and strongly connected directed graphs [5] and general graphs [16] thus extending results on undirected graphs by Papadimitriou et al [53].

6.3. Protocols

Participants: Stéphane Grumbach, Wenwu Qu, Kun Suo.

We have developed a library of protocols written in Netlog [10][11] such as routing protocols for ad hoc networks, DSDV (Destination Sequence Distance Vector) [56], AODV(Ad-hoc On-demand Distance Vector) [55], OLSR (Optimized Link-State Routing) [42], and VRR (Virtual Ring Routing) [29]. Some of these protocols were experimented in the framework of the WSNNet and Netqvis platforms, or in the iMote testbed [7]. In [6], we proposed a neighbor discovery protocol which adapts dynamically the refreshment period of entries in neighborhood tables. In [12], we proposed ripple routing to reduce the communication cost.

6.4. Verification

Participant: Stéphane Grumbach.

Netlog is a language designed to describe distributed programs. It has a precise semantics, provides a high-level of abstraction thanks to its datalog flavor and benefits from an efficient implementation. This makes it a very interesting target language for proofs of distributed programs. In [13], with the Coq proof assistant, we formalized the distributed computation model based on message passing with either synchronous or asynchronous behaviors; built the translation of Netlog programs; modeled the embedded machine evaluating Netlog programs, and thus established a framework to formally verify properties of distributed programs in Netlog. To test the framework, we proved the correctness of a concrete distributed program for constructing spanning trees over connected graphs.

7. Contracts and Grants with Industry

7.1. Government Contracts

- ANR Ubiquet, Ubiquitous Quest: declarative approach for integrated network and data management in wireless multi-hop networks, with Grenoble Institute of Technology (Christine Collet, Christophe Bobineau), and INRIA CITI Laboratory in Lyon (Stéphane Ubéda, Fabrice Valois), 2009 – 2012

8. Other Grants and Activities

8.1. Actions in France

- NETQUEST has started a collaboration with Jean-François Monin of FORMES team and Pierre Casteran at INRIA-Bordeaux-Sud-Ouest on models for distributed algorithms and verification.
- NETQUEST has a cooperation with Stéphane Ubéda and Fabrice Valois, on declarative networking, in the framework of a Sino-French PRA project for the period 2007-2008, and now supported by an ANR project, Ubiquest, for the period Sep 2009 - Aug 2012. The work focuses on various networking protocols, such as flooding, self-configuration, self-organization, routing, medium access, in the context of multihop networks, by using declarative modeling. Ahmad Ahmad-Kassem is under the joint supervision of Stéphane Ubéda and Stéphane Grumbach.
- NETQUEST has a collaboration with Christine Collet and Christophe Bobineau from the Laboratory LSR/IMAG in Grenoble on the development of query optimization techniques in the context of networks, in the framework of the ANR project, Ubiquest, for the period Sep 2009 - Aug 2012.

8.2. Actions in China

- NETQUEST has close links with the Institute of Software of the Chinese Academy of Sciences, ISCAS. The professor Huimin Lin, academician, is the supervisor of the two students Fang Wang and Wenwu Qu. Zhilin Wu is now researcher at ISCAS.
- NETQUEST collaborates with the LIAMA project FORMES and Jiaotong University in Shanghai on the verification of distributed algorithms expressed in Neglog.

8.3. Actions in Europe

- NETQUEST has a collaboration with Michel Ferreira from Porto University on declarative protocols for vehicular networks.

9. Dissemination

9.1. Service

- Stéphane Grumbach is a PC member of APWeb'10, DASFAA'10, DEXA'10, IDEAS'10, MDM'10.

9.2. PhD Jurys

- Stéphane Grumbach co-supervised with professor Huimin LIN the thesis of Wenwu QU, who defended in May 2010.
- Stéphane Grumbach co-supervised with professor Huimin LIN the thesis of Fang Wang, who defended in Nov 2010.

9.3. Internships

- Stéphane Grumbach supervised the following internships:
Shruti Agrawal, IIT Kanpur, on engine optimization, May 2010 - July 2010;
Eric Bellemon, INSA Lyon, on compiler, April 2010 - July 2010;
Mathieu Declercq, INSA Lyon, on visualization tool, April 2010 - July 2010.

9.4. Short-term visitors

- Christophe Bobineau, from LSR/IMAG in Grenoble, visited NETQUEST in Jan 2010.

10. Bibliography

Major publications by the team in recent years

- [1] S. GRUMBACH, P. RIGAUX, L. SEGOUFIN. *Handling Interpolated Data.*, in "Comput. J.", 2003, vol. 46, p. 664-679.

- [2] S. GRUMBACH, L. TINININI. *On the content of materialized aggregate views.*, in "J. Comput. Syst. Sci.", 2003, vol. 66, p. 133-168.
- [3] S. GRUMBACH, Z. WU. *Logical locality entails frugal distributed computation over graphs*, in "WG 2009, the 35th International Workshop on Graph-Theoretic Concepts in Computer Science", Montpellier, June 2009.
- [4] Z. WU, S. GRUMBACH. *Feasibility of Motion Planning on Directed Graphs*, in "Theory and Applications of Models of Computation, 6th Annual Conference, TAMC 2009", Changsha, China, May 18-22 2009, p. 430-439.

Publications of the year

Articles in International Peer-Reviewed Journal

- [5] Z. WU, S. GRUMBACH. *Feasibility of motion planning on acyclic and strongly connected directed graphs*, in "Discrete Appl. Math.", 2010, vol. 158, p. 1017-1028.

International Peer-Reviewed Conference/Proceedings

- [6] A. AHMAD-KASSEM, N. MITTON. *Adapting Dynamically Neighbourhood Table Entry Lifetime in Wireless Sensor Networks*, in "The 2010 International Conference on Wireless Communications & Signal Processing (WCSP 2010)", Suzhou, P.R. China, 2010.
- [7] M. BAUDERON, S. GRUMBACH, D. GU, X. QI, W. QU, K. SUO, Y. ZHANG. *Programming iMote Networks Made Easy*, in "The Fourth International Conference on Sensor Technologies and Applications", IEEE Computer Society, 2010, p. 539-544.
- [8] S. GRUMBACH, F. WANG. *Netlog, a Rule-based Language for Distributed Programming*, in "PADL'10, Twelfth International Symposium on Practical Aspects of Declarative Languages", Madrid, Spain, January 2010.
- [9] S. GRUMBACH, Z. WU. *Distributed tree decomposition of graphs and applications to verification*, in "APDCM2011: 13th Workshop on Advances in Parallel and Distributed Computational Models", 2010, p. 1-8.
- [10] K. SUO, W. QU, A. B. IRIONDO. *Declarative Programming of Network Protocols*, in "ICCT 2010: International Conference on Communication Technology", Nanjing, China, 2010.
- [11] K. SUO. *Declarative Programming of Ad Hoc Routing Protocols*, in "ICFIT 2010: International Conference on Future Internet Technology", Changsha, China, 2010.
- [12] J. ZHANG, S. GRUMBACH, D. YANG, A. M. ANAYA. *Ripple routing: an On-demand Routing Protocol for In-network Query Processing on Wireless Sensor Networks*, in "ICMA2010: IEEE International Conference on Mechatronics and Automation", 2010, in ICMA 2010: International Conference on Mechatronics and Automation, Xi'an, China.

Other Publications

- [13] Y. DENG, S. GRUMBACH, J.-F. MONIN. *Towards Verifying Declarative Netlog Protocols with Coq*, 2010, inria-00506093.

- [14] S. GRUMBACH, F. WANG. *Netlog, a Rule-Based Language for Distributed Programming*, 2010, journal version of the PADL10 article, submitted to publication.
- [15] S. GRUMBACH, Z. WU. *On the Distributed Computation of Logical Properties of Graphs*, 2010, submitted for publication.
- [16] Z. WU, S. GRUMBACH. *Feasibility of Motion Planning on Directed Graphs?*, 2010, Submitted for publication.

References in notes

- [17] *The Coq Proof Assistant*, <http://coq.inria.fr>.
- [18] M. J. C. GORDON, T. F. MELHAM (editors). *Introduction to HOL: a theorem proving environment for higher order logic*, Cambridge University Press, New York, NY, USA, 1993.
- [19] *PRISM - Probabilistic Model Checker*, <http://www.prismmodelchecker.org>.
- [20] *Uppaal Model Checker*, http://en.wikipedia.org/wiki/Uppaal_Model_Checker.
- [21] *WSNet*, <http://wsnet.gforge.inria.fr>.
- [22] S. ABITEBOUL, R. HULL, V. VIANU. *Foundations of Databases.*, Addison-Wesley, 1995.
- [23] D. ANGLUIN. *Local and Global Properties in Networks of Processors (Extended Abstract)*, in "STOC", 1980, p. 82-93.
- [24] M. M. ASTRAHAN, M. W. BLASGEN, D. D. CHAMBERLIN, K. P. ESWARAN, J. GRAY, P. P. GRIFFITHS, W. FRANK III. KING, R. A. LORIE, P. R. MCJONES, J. W. MEHL, G. R. PUTZOLU, I. L. TRAIGER, B. W. WADE, V. WATSON. *System R: Relational Approach to Database Management*, in "ACM Trans. Database Syst.", 1976, vol. 1, p. 97-137.
- [25] H. ATTIYA, J. WELCH. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*, Wiley-Interscience, 2004.
- [26] F. BANCILHON. *Naive evaluation of recursively defined relations*, in "On knowledge base management systems: integrating artificial intelligence and database technologies", 1986, p. 165-178.
- [27] F. BANCILHON, D. MAIER, Y. SAGIV, J. D. ULLMAN. *Magic sets and other strange ways to implement logic programs (extended abstract)*, in "PODS '86: Proceedings of the fifth ACM SIGACT-SIGMOD symposium on Principles of database systems", New York, NY, USA, ACM, 1986, p. 1-15.
- [28] K. BHARGAVAN, D. OBRADOVIC, C. A. GUNTER. *Formal verification of standards for distance vector routing protocols*, in "J. ACM", 2002, vol. 49, p. 538-576.
- [29] M. CAESAR, M. CASTRO, E. B. NIGHTINGALE, G. O'SHEA, A. ROWSTRON. *Virtual ring routing: network routing inspired by DHTs*, in "SIGCOMM Comput. Commun. Rev.", 2006, vol. 36, p. 351-362.

-
- [30] A. CERPA, D. ESTRIN. *ASCENT: Adaptive Self-Configuring sSensor Network Topologies*, in "SIGCOMM Comput. Commun. Rev.", 2002, vol. 32, p. 62–62.
- [31] E. M. CLARKE, O. GRUMBERG, D. A. PELED. *Model Checking*, The MIT Press, Cambridge, MA, USA, 1999.
- [32] E. F. CODD. *A Relational Model of Data for Large Shared Data Banks*, in "Commun. ACM", 1970, vol. 13, p. 377-387.
- [33] B. COURCELLE. *Graph Rewriting: An Algebraic and Logic Approach*, in "Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)", Elsevier and MIT Press, 1990, p. 193-242.
- [34] A. DAWAR, M. GROHE, S. KREUTZER, N. SCHWEIKARDT. *Approximation Schemes for First-Order Definable Optimisation Problems*, in "LICS", 2006, p. 411-420.
- [35] H. EBBINGHAUS, J. FLUM. *Finite model theory*, Springer-Verlag, Berlin, 1999.
- [36] A. P. FELTY, D. J. HOWE, F. A. STOMP. *Protocol Verification in Nuprl*, in "In Tenth International Conference on Computer Aided Verification", Springer-Verlag, 1998, p. 428–439.
- [37] M. FRICK, M. GROHE. *Deciding first-order properties of locally tree-decomposable structures*, in "J. ACM", 2001, vol. 48, p. 1184-1206.
- [38] H. GAIFMAN. *On local and non-local properties*, in "Proceedings of the Herbrand Symposium, Logic Colloquium '81", North Holland, 1982.
- [39] M. GROHE. *Logic, graphs and algorithms*, in "Logic and Automata: History and Perspectives", J. FLUM, E. GRÄDEL, T. WILKE (editors), Texts in Logic and Games, Amsterdam University Press, 2008, vol. 2.
- [40] J. M. HELLERSTEIN. *The declarative imperative: experiences and conjectures in distributed logic*, in "SIGMOD Record", 2010, vol. 39, p. 5-19.
- [41] G. J. HOLZMANN. *The Model Checker SPIN*, in "IEEE Trans. Software Eng.", 1997, vol. 23, p. 279-295.
- [42] P. JACQUET, P. MUHLETHALER, T. CLAUSEN, A. LAOUITI, A. QAYYUM, L. VIENNOT. *Optimized link state routing protocol for ad hoc networks*, in "Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International", 2001, p. 62–68.
- [43] S. KREUTZER. *Algorithmic Meta-Theorems*, in "CoRR", 2009, vol. abs/0902.3616.
- [44] F. KUHN, T. MOSCIBRODA, R. WATTENHOFER. *What cannot be computed locally*, in "PODC", 2004, p. 300-309.
- [45] N. LINIAL. *Locality in Distributed Graph Algorithms.*, in "SIAM J. Comput.", 1992, vol. 21, p. 193-201.
- [46] B. T. LOO, T. CONDIE, M. N. GAROFALAKIS, D. E. GAY, J. M. HELLERSTEIN, P. MANIATIS, R. RAMAKRISHNAN, T. ROSCOE, I. STOICA. *Declarative networking: language, execution and optimization.*,

in "Proceedings of the ACM SIGMOD International Conference on Management of Data", Chicago, Illinois, USA, June 27-29 2006.

- [47] B. T. LOO, T. CONDIE, J. M. HELLERSTEIN, P. MANIATIS, T. ROSCOE, I. STOICA. *Implementing declarative overlays*, in "SIGOPS Oper. Syst. Rev.", 2005, vol. 39, p. 75–90.
- [48] B. T. LOO, J. M. HELLERSTEIN, I. STOICA, R. RAMAKRISHNAN. *Declarative routing: extensible routing with declarative queries*, in "SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications", New York, NY, USA, ACM, 2005, p. 289–300.
- [49] J.-L. LU, F. VALOIS, D. BARTHEL, M. DOHLER. *FISCO: A Fully Integrated Scheme of Self-Configuration and Self-Organization for WSN.*, in "IEEE/WCNC", 2007.
- [50] N. A. LYNCH. *Distributed Algorithms*, Morgan Kaufmann, 1996.
- [51] M. MUSUVATHI, D. R. ENGLER. *Model Checking Large Network Protocol Implementations*, in "In Proceedings of the First Symposium on Networked Systems Design and Implementation", 2004, p. 155–168.
- [52] M. NAOR, L. J. STOCKMEYER. *What Can be Computed Locally?*, in "SIAM J. Comput.", 1995, vol. 24, p. 1259-1277.
- [53] C. H. PAPADIMITRIOU, P. RAGHAVAN, M. SUDAN, H. TAMAKI. *Motion planning on a graph*, in "FOCS'94", 1994, p. 511-520.
- [54] D. PELEG. *Distributed computing: a locality-sensitive approach*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
- [55] C. E. PERKINS, E. M. BELDING-ROYER. *Ad-hoc On-Demand Distance Vector Routing*, in "WMCSA", 1999, p. 90-100.
- [56] C. E. PERKINS, P. BHAGWAT. *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*, in "ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications", 1994, p. 234–244.
- [57] C. E. PERKINS. *Ad-hoc on-demand distance vector routing*, in "In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications", 1999, p. 90–100.
- [58] R. RAMAKRISHNAN, J. D. ULLMAN. *A survey of research on deductive database systems*, in "Journal of Logic Programming", 1993, vol. 23, p. 125–149.
- [59] J. RUSHBY. *Specification, Proof Checking, and Model Checking for Protocols and Distributed Systems with PVS*, in "Tutorial, FORTE X/PSTV XVII'97", 1997.
- [60] D. SEESE. *Linear time computable problems and logical descriptions*, in "Electr. Notes Theor. Comput. Sci.", 1995, vol. 2.
- [61] J. SUOMELA. *Survey of local algorithms*, 2009.

- [62] L. VIEILLE. *Recursive Axioms in Deductive Databases: The Query/Subquery Approach*, in "Expert Database Conf.", 1986, p. 253-267.