Mechanical Support for Efficient Dissemination on the CAN Overlay Network

Francesco Bongiovanni, Ludovic Henrio
INRIA Sophia Antipolis Méditérranée – CNRS – I3S – Université de Nice Sophia-Antipolis,
2004 Route des Lucioles - BP 93,
06902 Sophia Antipolis, France
firstname.lastname@inria.fr

I. Introduction

Peer-to-Peer (P2P) systems have been recognized as a key communication model to build scalable platforms for distributed applications such as file sharing, distributed storage, content delivery systems, etc. These systems often require application-level dissemination primitives which are efficient and reliable. Building such communication primitives in a reliable manner would increase the confidence regarding their behavior prior to deploying them in real settings. Using Isabelle/HOL [5], we focus our efforts around the *correct* design of an efficient communication primitive on top of the CAN overlay network [6]. In the process of this goal, we have come up with a reasoning framework that will allow us in the future to define dissemination primitives and *formally* prove their correctness properties.

II. Background and motivation

The Content Addressable Network (CAN) [6] is a structured P2P network based on a d-dimensional Cartesian coordinate space labeled $\mathcal{D}$. This space is dynamically partitioned among all peers in the system such that each node is responsible for storing data in a zone in $\mathcal{D}$; stored data consist in *(key, value)* pairs. To store the (k, v) pair, the key k is deterministically mapped onto a point i in $\mathcal{D}$ and then the value v is stored by the node responsible for the zone comprising i. The lookup for the value corresponding to a key k is achieved by applying the same deterministic function on k to map it onto i. The query processing is an iterative routing process which starts at the query originator and which traverses its *adjacent* neighbours (a peer's routing table only contains his adjacent nodes), then the neighbours' adjacent neighbours so on and so forth until it reaches the zone containing the value. as depicted by the *retrieve* operation in Figure 1.

CAN is a practical infrastructure for file sharing, data storage and can be also very effective when it comes to large scale information dissemination. As a matter of fact, network-layer multicast is still not widely adopted by most commercial ISPs [3] and this prevents the usage of practical native *one-to-many* communication primitives by today's large scale applications. This technical impediment, mainly due to costs issues and bandwidth preservation policies, was overcome with the introduction of *application-level multicast* protocols such as [2]. Such protocols can be designed on top of P2P networks such as CAN in order to take advantage of its underlying geometrical topology and the various good properties it offers such as deterministic lookup complexity, tolerance to churn,...

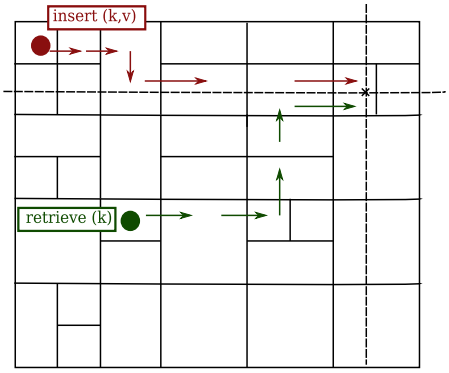

Fig. 1. Routing in CAN - example of data storage (*insert(key ,value)*) and retrieval (*retrieve(key)*).

Why Isabelle/HOL? In general, formal methods improve the reliability of proposed algorithm and the confidence one has in their properties. In our case we want to see what conditions are necessary to ensure the correctness and other properties of a broadcast algorithm over a CAN. The kind of properties we are interested in are *coverage*, i.e. all the nodes in the network correctly receive the message to broadcast, *efficiency*, i.e. a node receives the message only once and *termination*, i.e. the algorithm terminates correctly. Mechanical proofs will ensure the correctness of the studied protocols, with a much higher confidence than paper proofs which rely too often on "well known" properties or "obvious" steps that could reveal wrong or underspecified. A theorem prover enforces the precise and sound formalization of the studied protocols, and of the hypotheses ensuring their correction and properties. Proving properties on distributed algorithms could be done by specific formalisms for distributed systems, like TLA$^+$ [4], however we choose a more general theorem prover to have better support for general reasoning. Indeed, reasoning on the geometry of a CAN requires generic theorems that will be better supported by a general purpose theorem prover like Isabelle or Coq for example. Additionally, all formal methods relying on model-checking work by instantiation on a finite set of states, meaning one can only verify protocols on a small number of processes. Theorem proving on the contrary requires the help of the programmer to prove properties that are valid on an arbitrary number of processes. Consequently, the proofs performed in Isabelle/HOL are particularly adequate to study large-scale distributed systems. The expressiveness of

Isabelle's logic allows us to reason on an abstraction of the system we design, meaning that we can abstract away some properties and precise details of the CAN overlay and focus on the aspects ensuring the correctness of the dissemination algorithm properties. The benefits of using such an environment is that it gives us the confidence in the correctness of the proofs we construct. There is a strong need for enriching the existing Isabelle libraries with specific reasoning building blocks for distributed systems. CAN is a popular distributed hash table (DHT) which is used as a distributed substrate for large scale applications. Thus, our motivation is to put forward *proven* abstractions for proving correctness properties of distributed algorithms on top of CAN in order to contribute to the advancement of *correct* distributed algorithms.

III. A Mechanized Model for CAN and Broadcast Algorithms

We describe in this section some parts of our formalization of the CAN network and an intuition behind the broadcast algorithm by providing some definitions in Isabelle/HOL. Our objective is to give a rough idea of our approach, but we refer the reader to the source code available on our Web page[1] for the exhaustive formalization. The Isabelle/HOL syntax is much similar to mathematical notations; the main differences are the following: $\longrightarrow$ is the implication, :: defines the type of an expression, and $A \Rightarrow B$ is the type of functions from A to B. For manipulating structures Isabelle/HOL provides the following notations: ! accesses an element of a list, # is the list constructor it appends an element at the beginning of the list, and @ appends two lists.

A. CAN

A crucial question when formalizing a complex structure like a CAN is which level of abstraction should be used, and which notions of Isabelle/HOL should represent basic notions of CAN networks. We represent a CAN by a set of nodes, a zone for each node, and a neighboring relationship, stating whether one node is neighbour of another. More precisely, a *CAN* is a set of integers identifying the different nodes. A function Z matches each node to a *Zone*, where a zone is a *Tuple set* (a tuple is an array of integers). Note that we abstract away a few constraints of the CAN model which are not useful for us, like zones are rectangular, and we do not relate zones with the neighbouring notion as it does not reveal useful for the moment. In Isabelle, a CAN is defined as follows:

typedef *CAN* =
{(*nodes*::*nat set*, *Z* :: *nat* ⇒ *Zone*, *neighbours*:: (*nat* × *nat*) *set*) .
finite nodes ∧
finite neighbours ∧
(∀ *x y*. (*x*,*y*)∈ *neighbours* ⟶(*y*,*x*) ∈ *neighbours*)∧
(∀ *x*. (*x*,*x*)∉ *neighbours*) ∧
(∀*tup*. ∃*n*∈*nodes*. *tup* ∈ (*Z n*)) ∧
(∀ *N*∈*nodes*. ∀ *N*′∈*nodes*. *N*≠*N*′ ⟶¬ *intersects* (*Z N*) (*Z N*′))}

Additional constraints state that the set of nodes is finite, the zones cover the whole space, and are disjunct. We define three auxiliary functions *CAN_Nodes*, *CAN_Zones*, and *CAN_neighbours* returning each part of a *CAN*.

We also define a function *intersects Z Z'* that checks whether zone Z intersects zone Z': it is true if Z and Z' have at least one point (tuple) in common. In the following, we say that "a node intersects a zone Z", if the zone of the node (((*CAN_Zones C*) *node*)) intersects Z. Then we state that a zone is connected[2] if the nodes it intersects are all connected to one another (there is a path of neighbours between any two nodes intersecting the zone).

It states that if N and N' are two nodes which zones intersect Z, then there is a list of nodes (all distinct) starting at n, ending at n', only passing by nodes intersecting Z, and for which each node of the list is neighbour of the previous one. We also proved a few generic lemmas that will be useful for proving properties entailing CAN structure, however we do not detail them here.

Let us first mention an induction principle that will allow us to prove a property related to a zone by induction on the size of the zone, or more precisely by induction on the number of nodes intersecting the zone. In Isabelle, $\bigwedge$ stands for "for all" (at the meta-level), and a theorem is expressed by a term of the form ⟦*premise 1; premise 2* ⟧ ⟹ *Conclusion*. The induction theorem is thus written as follows:

theorem *induct_node_zone*:
⟦ *P* {};
⋀*n Z*. ⟦⋀ *Z*′. *card* {*node* ∈ *CAN_Nodes C*. *intersects Z*′ (*CAN_Zones C node*)}=*n* ⟹ *P Z*′;
card {*node* ∈ *CAN_Nodes C*. *intersects Z* (*CAN_Zones C node*)}=*Suc n*⟧ ⟹ *P Z* ⟧ ⟹ *P Z*

It states that, if (1) we prove a property P is true for an empty zone, and (2) we prove that if P is true for all zones of size n then it is true for all zones of size $n+1$; then the property is true for all zones.

B. Broadcast Specification

When the structure of the network is defined, we can provide a definition for messages and for the path followed by a message. A message is made of four pieces of information: an identifier for the message (which could identify uniquely its content for example), a source node, a destination node, and the zone to which it must be transmitted.

[1]The code can be found here:http://www-sop.inria.fr/oasis/personnel/Ludovic.Henrio/misc. We use the *Isabelle2009-2* version.

[2]This notion is closed to the geometrical notion of connectivity, or rather path connectivity.

The Isabelle code defining such a quadruple is very simple:

types *Message* = *nat* × *nat* × *nat* × *Zone*

We decided to rely on the notion of zone to be covered to define a broadcast algorithm, because it seems quite adapted to a CAN. Also as we are looking for an efficient algorithm, it seems quite reasonable to try to split efficiently the zone to be covered in order to avoid sending a message to the same node twice. *Message_zone*, *Message_dest*, and *Message_source* are functions accessing the three first fields. We also define an abbreviation $<m|x,y,Z>$ for defining a *Message*, this allows us to easily identify messages inside the definitions and lemmas.

We can now define a broadcast mechanism for the CAN overlay network. It is far from trivial to define an algorithm in a convincing way in Isabelle/HOL. Indeed, the basic language of Isabelle is a pure functional language similar to λ-calculus, which is not the language in which we would usually encounter broadcasting algorithms. Here we want to focus on the way a message triggers other ones, for this we concentrate our specification on the notion of consequences, and on a specification of the *set of messages* used to broadcast an original message.

In our framework *Broadcast* is a triple made of a *CAN*, a message set and initiator node constrained by several well-formedness rules as defined below:

typedef *Broadcast* =
{(*can*,*msgs*,*initiator*).
(∀ *x y m Z m′ Z′*. (<*m*|*x*,*y*,*Z*>∈*msgs* ∧ <*m′*|*x*,*y*,*Z′*>∈*msgs*) ⟶
(*m*=*m′*∧ *Z*=*Z′*)) ∧ (*initiator* ∈ *CAN_Nodes can*)∧
(∀ *m s d Z*. <*m*|*s*,*d*,*Z*>∈*msgs*
⟶ (*s*∈ *CAN_Nodes can* ∧ *d*∈ *CAN_Nodes can* ∧
CAN_neighbour can s d∧
(∃ *MsgL*. *valid_path msgs MsgL* ∧ *destination MsgL* = *s* ∧
source MsgL=*initiator*)∧
(∀ *m′ d′ Z′*. <*m′*|*d*,*d′*,*Z′*>∈*msgs*
⟶ (*intersects* (*CAN_Zones can d′*) *Z* ∧ *Z′* ⊆ *Z*))))}

The constraints expressed in the above definition state that:

- There is a single message between any 2 nodes
- The initiator is a node of the CAN
- All messages are exchanged between neighbour nodes of the CAN
- All messages must originate from a node that has been reached by a list of messages originating from the origin node: *valid_path msgs MsgL* ∧ *destination MsgL* = *s* ∧ *source MsgL*=*initiator*. Requiring the existence of such a valid path ensures that a broadcast only relies on messages transmitted from a node to its neighbour (except for the origin of course). Note that it is not sufficient to require that each message source is the destination of another message, because that would mean that loops of messages not passing by the origin would be allowed.
- Each node *d* sends only messages ($<m'|d,d',Z'>$) to nodes it has to cover, i.e. node *d'* must intersect the zone *Z* that *d* received in its message. We say that the message $<m'|d,d',Z'>$ sent by *d* is a consequence of the first one ($<m|s,d,Z>$).
- Finally, the zone of a message must always be bigger than the one of its consequences: a node can only delegate the coverage of a subset of the zone it is responsible for.

We note $<C,M,n>$ such a *Broadcast*, and define functions *BC_CAN*, *BC_msgs*, and *BC_initiator* to access its fields. We can then define a predicate checking whether a broadcast covers the whole CAN (each node of the CAN is either the initiator or the destination of a message):

definition *Coverage*:: *Broadcast*⇒*bool*
where *Coverage BC* ≡
∀*n*∈(*CAN_Nodes* (*BC_CAN BC*)).
(*n*=*BC_initiator BC* ∨ (∃ *m s Z*. <*m*|*s*,*n*,*Z*>∈*BC_msgs BC*))

From those definitions, we expect to prove completeness of some specific broadcast algorithm, but also study their optimality.

Overview of the Mechanization Process. Overall the current specification and proofs consist of 1800 lines of Isabelle code. As usual, most of the code is dedicated to proofs, but since we are in the early stages of the formalization, and as our framework requires a lot of different notions, the definitions amount for more than 10% of this code. Difficult parts of the reasoning concern the finite sets, and the difficulty to reason by recurrence on a set that is finite but not inductively defined. The CAN overlay network is a difficult setting for proofs, because the structure entails some (simple) geometrical reasoning, which is more complex than reasoning on structures that could be easily defined by induction. Indeed, Isabelle/HOL support for inductive reasoning is more valuable than for other kind of reasoning; but this "only" makes the proofs more difficult to perform, and longer.

A crucial part of our approach relies on the fact that the definition and properties are expressed in a formalism that is convincing: it must be easy for an external reader familiar with basic logics and mathematics to understand our formalism, to be convinced by our formalization of a CAN network, and of its properties. Note that we do not plan to extract code, and thus an efficient formalization is not a crucial prerequisite.

It is important for us to have a formalism for expressing the CAN broadcast that is easy to understand; that is why we presented the sketch of the specification of a broadcast. Although the specification is inductive and thus not in a classical form for a broadcast algorithm, we think it is clear enough to be convincing, and that it is easy to extract an algorithm from it. This way of expressing a broadcast algorithm is not as natural as one would expect because a form of event-based formulation of the algorithm "when a message M is received, send messages M1, M2, and M3" would be more adapted. However, such an event-like formulation is not well supported in Isabelle/HOL. In the future, we will try to provide abbreviations in Isabelle to allow a formulation closer to the reaction to message reception events. This formalization provides a set of theorems allowing one to prove the properties of communication algorithms, over CAN-like networks. The outcome of this work will thus be a set of properties for several broadcast algorithms, starting by coverage, (i.e. each

node receives the message). As we will define precisely the hypotheses on the network topology and on the algorithm, we will know exactly to which kind of networks those algorithms are applicable. For more details, we refer the readers to [1].

REFERENCES

[1] Francesco Bongiovanni and Ludovic Henrio. Mechanical Support for Efficient Dissemination on the CAN Overlay Network. Research Report RR-7599, INRIA, 2011.

[2] Y. Chu, S.G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *Selected Areas in Communications, IEEE Journal on*, 20(8):1456–1471, 2002.

[3] S. E Deering and D. R Cheriton. Multicast routing in datagram internetworks and extended lans. *ACM Transactions on Computer Systems (TOCS)*, 8(2):85–110, 1990.

[4] L. Lamport and Safari Tech Books Online (Online service). *Specifying systems: The TLA+ language and tools for hardware and software engineers*, volume 14. Addison-Wesley, 2003.

[5] T. Nipkow, M. Wenzel, and L.C. Paulson. Isabelle/HOL: a proof assistant for higher-order logic. *Lecture Notes In Computer Science; Vol. 2283*, page 205, 2002.

[6] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 161–172. ACM, 2001.