Abstract

We propose a new approach to modelling and reasoning about dynamic networks. Dynamic networks consist of nodes and edges whose operating status may change over time (for example, the edges may be unreliable and operate intermittently). Message-passing in such networks is inherently difficult and reasoning about the behaviour of message-passing algorithms is also difficult. We develop a series of abstract models which allow us to focus on the correctness of routing methods. We model the dynamic network as a “demonic” process which runs concurrently with routing updates and message-passing. This allows us to use temporal logic and fairness constraints to reason about dynamic networks. The models are implemented as multi-threaded programs and, to validate them, we use an experimental run-time verification tool called RULER.

1 Introduction

We are increasingly reliant on highly dynamic and complex computing systems. In communication, dynamic networks are widespread these include the: Internet, peer-to-peer networks, mobile networks and wireless networks. Networks may have edges down, nodes may move, or there may be routing instability due to the changing of networks. These systems are very difficult to analyse, and their behaviour and correctness are hard to formulate and establish. To undertake formal reasoning about such systems, abstract models are essential in order to separate the general reasoning about message routing and updating of routing tables from the details of how these are implemented in particular networks. We show we can establish correctness of dynamic networks at suitable levels of abstraction.

At its simplest level, a network consists of a collection of nodes connecting to each other through edges. In a message-passing network, each node communicates by exchanging messages in an attempt to deliver messages to their destinations. In a dynamic network, nodes and/or edges may become inoperative or operative. This representation of the dynamics of a network clearly models unreliable networks. It also models mobile and wireless networks by considering edges as possible communication links and operative edges as the links established at a particular time.

The two problems of message-passing in high-level models [2, 3, 4] and self-stabilising systems [5] in dynamic networks have been widely studied [1]. Numerous models and algorithms...
have been proposed but proofs of correctness (especially liveness) have tended to need the assumption that changes in networks eventually cease, i.e., they are no longer dynamic.

Here, the correctness of dynamic networks can be established without the termination requirement. We develop the correctness of dynamic networks in terms of ensuring that: even when routing tables do not reflect the actual network connections, the routing information is correct sufficiently often; messages eventually get delivered; the network is sufficiently connected for sufficiently often; and there is no persistent livelock. The main contributions of this paper are: modelling dynamic networks using concurrent systems; factorisation of proof; and run-time verification of the implementation of dynamic network models.

We introduce a new approach to proof techniques for dynamic networks in which using ideas from concurrent systems [11] to analyse message-passing. To do so, we use Linear Temporal Logic and formulate concepts of fairness which capture network properties. In order to express dynamic networks as concurrent systems [6], we consider the dynamic changes to be the result of a “demonic” process which runs concurrently with routing updates and message-passing. By the correctness of dynamic networks, we mean that, under certain conditions, all messages will eventually be delivered. By formulating networks as concurrent systems, we can establish correctness for networks that never cease to change. By modelling at this level of abstraction, we are able to prove the properties of networks independently of the mechanisms in actual networks and therefore provide “a factorisation” of proofs of correctness for actual dynamic networks. We have implemented two abstract models as concurrent systems and then adapted a run-time verification systems RuleR [8], to analyse execution traces to test whether model instances satisfy the modal correctness for message delivery.

2 Abstract models

Consider a graph \( G = (N, E) \) where \( N \) is a set of nodes and \( E \) is a set of edges. This provides the basic connectivity of a network. To introduce dynamics (for edges) we consider the edges to be in an on or off state. Therefore we introduce edge status \( L \), which changes value whenever a disruption occurs for an edge. Routing update information is determined by \( L \).

There is a set of messages \( M \), where each message is at a node defining a function \( M \to N \).

We develop dynamic network models as concurrent systems. We introduce two models. Firstly, Model 1, as shown in Figure 1, is a two-process model with instantaneous updates, in which the routing tables are always correct. The two processes are a “demonic” Disrupter (which disrupts the connectivity of dynamic networks) and an Organizer (which attempts to deliver messages). In Model 2, we introduce a more realistic routing table update, adding a third process called Updater, as shown in Figure 2. Here, the routing tables may not be correct at any time but routing is still possible. The Disrupter process can disrupt the connection of an edge (so it becomes ‘on’ or ‘off’). The Updater runs concurrently recalculating the
routing update information to obtain actual available paths. If there is an available path, the Organizer process runs concurrently can send a message to the next node along a path.

3 Proving correctness of message-passing in dynamic networks

We use discrete-time Linear Temporal Logic (LTL) [10] to describe the properties of execution traces of this multi process systems. Some of the key properties which enable us to reason about network correctness are expressed as fairness constraints [9] in concurrent process models. We use strong fairness at the process level to express, for example, the relative frequency of network change to message motion and of routing table updates to network change.

Our aim is to formulate and prove the correctness of message-passing using the two abstract models above. We need to prove that, under certain conditions, all messages eventually reach their destination. We introduce a colouring of message according to their frequency of network change to message motion and of routing table updates to network change.

Paths exist infinitely often:

1. Need to establish the following:

   - Some in terms of fairness of process interaction, others as connectivity properties of graphs.

   Notice the formulation expressing the properties of dynamic networks as trace properties, the routing table entry saying an available path between node being any formula:

   - progressing along the route; and
   - collection. Here states. This is inspired by Gries [13] and Dijkstra [12]'s reasoning about on-the-fly garbage collection.

   ▶

   (A.)

   Trivial: by modus ponens using P1 and P2.


   Notice the formulation expressing the properties of dynamic networks as trace properties, some in terms of fairness of process interaction, others as connectivity properties of graphs.

   ▶

   (A.) messages eventually move, which means that the Organizer should access each message sufficiently often, and when it is accessed, it can be moved. Therefore, we modify P2 to P2′ to follows to include a fairness requirement.

   P2′: ∀m ∈ M. (□ (pathX2(at(m), dest(m)))) ⇒ (black(m) ∧

   (red(m) ⇒ (green(m) ∧ ∃p ∈ P(at(m), dest(m)). path(m, p))) ∧

   (∃p ∈ P(at(m), dest(m)). green(m) ∧ path(m, p) ∧ at(m) ≠ dest(m) ⇒

   (up1st_elmt(p)) ∧ path(m, tail_of_path(p)) ∧

   at(m) = next_node(p))) ∧ ((green(m) ∧ at(m) = dest(m)) ⇒ □ black(m))))
We also need:

(a) **Finiteness of paths**, which we define as:

$$\text{FP: } \forall m \in M. \ ((\text{green}(m) \land \neg \Box \text{red}(m)) \Rightarrow (\Diamond \text{black}(m)))$$

which means that if a message $m$ is green and there is no potential to become red eventually then message $m$ will eventually become black. This can only hold if the Organizer checks message $m$ infinitely often ($P2''$).

(b) No livelock. Here we define **Livelock-free** as:

$$\text{LF: } \forall m \in M. \neg \Box (\text{green}(m) \Rightarrow \Box \text{red}(m)),$$

i.e a green message becomes red (without an assigned route) only finitely often.

(B.) each message **eventually reaches** its destination.

We show that for each $m$ there is a point in the trace at which $\Box (\text{green}(m) \lor \text{black}(m))$ hence $\Diamond \text{black}(m)$.

► **Lemma 2.**

$$\forall m \in M. \ Box (\exists p \in P(at(m),\text{dest}(m)). ((P1 \land P2'' \land \text{Lemma 1} \land \text{green}(m) \land \text{path}(m,p)) \Rightarrow \Diamond \text{moved}(m,z)))$$

**Proof.** Suppose message $m$ has not reached the destination. We then follow from the proof of Lemma 1, and by modus ponens on $P1$ and $P2''$, hence we have Lemma 2. ▶

► **Lemma 3.**

$$\forall m \in M. \ Box ((FP \land LF \land \text{Lemma 1} \land \text{Lemma 2} \land P2'' \land \text{green}(m) \land \exists p \in P(at(m),\text{dest}(m)). \text{path}(m,p)) \Rightarrow \Diamond \text{black}(m))$$

**Proof.** By Lemma 1 and the definition of Livelock-free, and by modus ponens on $P1$ and $P2''$, as well as on Lemma 2 and finiteness of path definition, the message is $\Diamond \text{black}(m)$. Hence we have Lemma 3. ▶

By Lemma 1, 2, and 3, we have

► **Theorem 1.** $\forall m \in M. \ Box ((P1 \land P2 \land P2'' \land FP \land LF \land \text{red}(m)) \Rightarrow \Diamond \text{black}(m))$.

For **Model 2**, we need an additional property which expresses that the routing table is populated sufficiently often. This is formulated as follows.

$$\text{P3: } \forall n_1, n_2 \in N. \_box (\exists p \in P(n_1, n_2). \text{rt}(n_1, n_2, p)))$$

We also need to modify $P2$ to $P2''$ since paths for messages are obtained from the routing table, replacing $\text{path}(n_1, n_2)$ with $\text{rt}(n_1, n_2, p)$. We modify $P2'$ to $P2''$ to extend the fairness requirement which includes the routing tables when the message $m$ is at its position. The model also needs routing tables which are correct sufficiently often. In **Model 2**, the condition $P2''$ is replaced by $P2'''$, then finally we have a similar theorem. The proofs proceeds as for **Model 1**.
4 Experimental validation: Using run-time verification

We now consider the following question. Suppose a dynamic network is implemented as a concurrent system using multiple Java threads. When do the network properties (expressed as properties of execution traces of concurrent systems as above) hold and therefore, by the proofs above, all messages are eventually delivered?

There are several approaches. We could prove the implementation manually or we could use a verification technique such as model checking. Here we introduce a new approach based on run-time verification (RV) [7], as pictured in Figure 3. Whether or not a system satisfies the properties required for message delivery depends on interprocess interaction and the parameters involved in this. Run-time verification is particularly suitable here as it is the relationship of these parameters with the execution traces that determine the correctness of the dynamically allocated interprocess interaction. We have implemented Model 1 and Model 2, and use an experimental run-time verification tool RuleR, which has been developed by Barringer et al. [8]. RuleR is a rule-based run-time verification with dynamic rules. This is an experimental use of RV on concurrent models. Some trace properties required are properties properly of infinite traces. We show how to use RV to examine finite traces and relate this to the overall network behaviour.

Consider a result of Model 1, as Figure 4 shows, in which Disrupter process and Organizer process running concurrently (denoted as “Disrupter ∥ Organizer”, for example, “4 ∥ 1” means that the Disrupter process sleep four times longer than the Organizer process) for 60 msecs. Property P2 (i.e messages are looked at sufficiently often) is depicted as a percentage of Organizer actions within the traces (called “%AllOrg”), which is recorded by RuleR. The percentage of messages reach their destination is depicted as “%Msg”. “%PathX” is the percentage of path that exist. The result shows that: if path existence occurs infinitely often
and the messages are looked at sufficiently often, then the number of messages eventually reach their destinations are increased. This result supports the proof of Lemma 1, 2 and 3, hence the Theorem 1. As Figure 5 shows, when the traces are longer (time = 180 msecs), the relation between all conditions (fairness and properties) engenders more confidence and the messages eventually get delivered.

5 Conclusion

We have shown that, by introducing models of message-passing dynamic networks as concurrent systems, we can use standard proof techniques for concurrent systems based on temporal logic and properties such as fairness to establish correctness (i.e. the eventual delivery of all messages) of dynamic networks at an appropriate level of abstraction. Moreover, we have employed techniques recently developed in run-time verification in order to check whether implemented models of dynamic networks satisfy the required temporal properties for correct message delivery.

References