Specification of Synchronous Network Flooding in Temporal Logic

Ra'ed Bani Abdelrahman[1], Rafat Alshorman[2], Walter Hussak[3], and Amitabh Trehan[3]
[1]SoftwareEngineering Department, Ajloun National University, Jordan
[2]Department of Computer Science, Yarmouk University, Jordan
[3]Computer Science Department, Loughborough University, United Kindom

Abstract: *In distributed network algorithms, network flooding is considered one of the simplest and most fundamental algorithms. This research specifies the basic synchronous memory-less network flooding algorithm where nodes on the network don't have memory, for any fixed size of network, in Linear Temporal Logic. The specification can be customized to any single network topology or class of topologies. A specification of the termination problem is formulated and used to compare different topologies for earlier termination. This paper gives a worked example of one topology resulting in earlier termination than another, for which we perform a formal verification using the model checker NuSMV.*

Keywords: *Network flooding, linear temporal logic, model checking.*

Received December 17, 2018; accepted June 11, 2019
https://doi.org/10.34028/iajit/17/6/5

1. Introduction

Distributed systems can be specified as a composition of the specifications of constituent components using well-studied process-calculi approaches such as Communicating Sequential Processes (CSP) [9], CCS [18], the π-calculus [19], the Ambient Calculus [2] and I/O automata [17]. These methods are useful when components have significant internal actions/state that affect external actions but need to be abstracted away to prove properties of external behaviour. Our interest in this paper is the network flooding algorithm where individual components, in this case physical nodes in the network, have minimal internal state. The global properties to be proved derive their complexity from the topology of the network graph. It may be difficult to achieve a desired global topology as some kind of composition of components may necessitate an extra proof to show that the topology is indeed achieved.

This research chooses a more direct logic-based approach specifying the overall system-algorithm and network topology-as a set of temporal logic constraints in order to prove required properties. This has the added benefit that a different topology for the network can be specified easily by changing a single constraint, rather than many components, in order to achieve the same effect. This paper is structured as follows. Section 2 describes the synchronous flooding algorithm. Section 3 defines the temporal logic and operators used in the specification. The specification of the network flooding is in section 4 along with the proof obligation for the basic property of termination. This is applied to comparing termination in different network topologies in section 5. A worked example is described in section 6, as well as its proof in NuSMV. The conclusions are in section 7.

2. The Synchronous Flooding Algorithm

Distributed networks routing algorithms deal with directing and redirecting messages between the different network routers and end points [3]. This research refers to routers and endpoints as nodes. The router's job is to send the message to one of its neighbours that has a connection to in order to deliver the message to its destination [3]. A fundamental algorithm which can also be also be used for routing is the flooding algorithm [1]. Flooding forms the basis of many important distributed processes, for example, construction of BFS trees which are used example in one of the author's work on distributed Leader Election [14]. The flooding algorithm is an algorithm which utilizes every path in the network [23]. In flooding, a message is sent from one node to all of its neighbours.

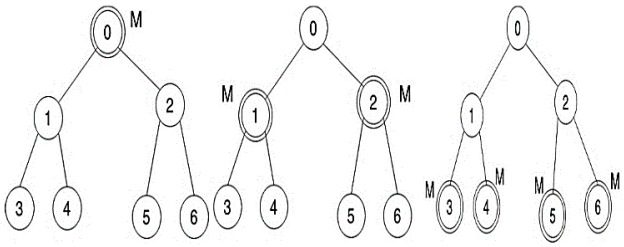
A neighbour which receives a message, will forward the message to all of its neighbours except the one(s) from which it originally received the message [1, 23, 25, 27]. Synchronous distributed algorithms assume a 'global clock' where actions happen in clock ticks or rounds. This means that the network has bounded link delays and lockstep synchronization with pulses of the global clock. In the message synchronization property, a message sent from node v to neighbour u at pulse p of v must be delivered to u before pulse p+1 of u [23].

In the first round, a message sent from the initial node to its neighbours as shown in Figure 1-b). In the second round, a neighbour which received this message will forward the message to all of its neighbours except the ones which it received it from.

Eventually, all nodes in the network will receive a message in a certain round.

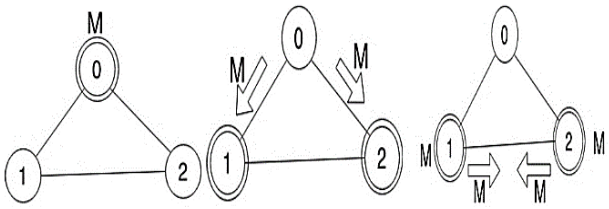
This paper investigates 'memoryless' flooding i.e., a node does not explicitly remember if it has previously taken part in the process or who it interacted with before. This may happen, for example, if the node does not have enough memory to store past history or there are multiple flooding operations going on which it does not want to, or cannot, distinguish. It does, however, know which node(s) sent it the message in the present round and forwards copies of the message to all the neighbours apart from the one(s) it received from.

Notice that if any round, a node receives the message from all its neighbours, the node does not need to do anything. If at some point, no node forwards the message, we say, flooding has terminated. It is hard to know if the flooding process will ever terminate especially in complicated topologies with cycles. Figure 1-a and Figure 1-c demonstrates the synchronous flooding algorithm in a network of four nodes. Nodes which hold a message "M" are double-circled. Figure 2-a and Figure 2-e demonstrates another example of the synchronous flooding algorithm in a network of three nodes. Nodes which hold a message "M" are double-circled.

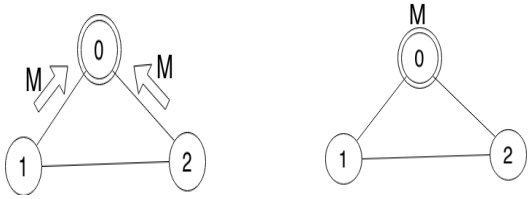


a) Round 0. b) Round 1. c) Round 2.

Figure 1. Flooding example 1.



a) Round 0. b) Round 1. c) Round 2.



d) Round 3. e) Round 4.

Figure 2. Flooding example 2.

3. Linear Temporal Logic

This chapter and the next will use standard Linear Temporal Logic (LTL) with the temporal operators defined below as it has approved to be a successful approach in networking applications as in [12].

3.1. Syntax of LTL

The alphabet of LTL consists of a set of propositional symbols P_i,i=0,1,2,… (this paper will use different capital letters to P in different contexts), Booleans $\neg,\wedge,\top,\bot$, and temporal operators X, Y, F, G. Formulae in LTL are those generated by: $\phi ::= P_i \left|\neg\phi\right|\phi_1 \wedge \phi_2 \left|X\,\phi\right|Y\,\phi\left|F\phi\right|G\phi$ The Boolean connectives ∨, ⇒ and ⇔ will be defined in terms of ¬ and ∧ in the usual way.

3.2. Semantics of LTL

LTL is interpreted over a sequence of temporal states (which this research sometimes refers to as 'points in time', even though they may not represent real time) $s_0,\ldots,s_a,\ldots(a\in\mathbb{N})$ An interpretation for LTL, I, (S_a), at a given state S_a assigns truth values $P_i^{Is(a)}$ to the propositional symbols P_i. A structure M is a sequence of interpretations I, (S_0),…, I, (S_a)… for the sequence of states. The semantics of a LTL formula ø is given by a truth relationship $M,s_a \vDash \phi$ which means that ø holds at state S_a in the structure M. The relation ⊨is defined inductively as follows:

$$M,s_a \vDash P_i \text{ iff } p_i^{I(s_a)} = \top$$
$$M,s_a \vDash \neg\phi \text{ iff } M,s_a \nvDash \phi$$
$$M,s_a \vDash \phi_1 \wedge \phi_2 \text{ iff } M,s_a \vDash \phi_1 \text{ and } M,s_a \vDash \phi_2$$
$$M,s_a \vDash \mathbf{X}\phi \text{ iff } M,s_{a+1} \vDash \phi$$
$$M,s_a \vDash \mathbf{Y}\phi \text{ iff } M,s_{a-1} \vDash \phi \quad (a>0)$$
$$M,s_a \vDash \mathbf{F}\phi \text{ iff there exists } k \geq a \text{ such that } M,s_k \vDash \phi$$
$$M,s_a \vDash \mathbf{G}\phi \text{ iff, for all } k \geq a,\ M,s_k \vDash \phi$$

Intuitively, the temporal operator X reads as "in the next state", Y reads as "in the previous state", F reads as "in some future state", and G reads as "in all future states". A structure M is a model of a LTL formula ø iff $M,s_0 \vDash \phi$. In general, a given LTL formula ø has many models. The behaviours of network flooding in different contexts (e.g., in different network topologies) is a set of models. This research specifies such network flooding in temporal logic by giving a LTL formula ø whose models correspond exactly to these behaviours of network flooding. We can then use this ø to construct further LTL formulae (called 'proof obligations') that will assert properties of ø such as when the behaviour of flooding leads to termination.

4. Specification of The Synchronous Flooding Algorithm

In the specification here, temporal logic states will correspond to rounds in the progression of network flooding. Successive rounds change the state of the network, for example the nodes in the network that are

receiving messages in a particular round. Let N be the set of nodes in the size of network under consideration. The following subsections give the propositions and constraints on them that define the behaviour of rounds.

4.1. Edge Propositions

The set of graph edge propositions is given by:

$$\{E_{\{g,h\}} \mid g,h \in N, g \neq h\}$$

Intuitively, $E_{\{g,h\}}$ is true iff there is an edge between the distinct nodes g and h in the graph. Notice that this research have used a set $\{g,h\}$ as a subscript in $E_{\{g,h\}}$. This is to indicate that $E_{\{g,h\}}$ is the same proposition as $E_{\{g,h\}}$, i.e., edges in N are undirected, and to say that there is an edge from g to h is the same as saying that there is an edge from h to g. We can specify whether or not two nodes g and h have an edge between them by specifying if $E_{\{g,h\}}$ is True or False. In this way a specific graph topology can be defined one edge at a time. Secondly, we can give general Boolean constraints on the edge propositions. The set of solutions of the constraints is a set of combinations of edge propositions being true corresponding to a set of graph topologies for N. As a third possibility, we may choose not to specify any Boolean constraints on edge propositions if we want to prove some property of network flooding for all graph topologies on N. However, the use of edge propositions does require a basic temporal constraint on them, that is that the Boolean value of an edge variable is time-independent. Nodes g and h have an edge between them either always or never, as edges represent physical connections that do not change with time. This temporal constraint is given by:

$$\phi_e \equiv \bigwedge_{\substack{g,h \in N, \\ g \neq h}} (\mathbf{G}E_{\{g,h\}} \vee \mathbf{G}\neg E_{\{g,h\}})$$

4.2. Send-Message Propositions

Messages may be sent between nodes g and h in both directions. So, we have send propositions

$$\{S_{g,h} \mid g,h \in N, g \neq h\}$$

Where $S_{g,h}$ is true in a particular round iff node g sends a message to node h in that round. As the sending of messages between nodes is directional, $S_{g,h}$ and $S_{g,h}$ are different propositions which may differ on their respective truth values in each round. Also, the sending of messages is time-dependent so the truth value of a particular send will vary over time. The basic constraint on send propositions relates to the edge propositions, as messages can only be sent from node g to node h along an edge from g to h, and so $E_{\{g,h\}}$has to be *True*. The constraint is:

$$\phi_s \equiv \bigwedge_{\substack{g,h \in N, \\ g \neq h}} \mathbf{G}(S_{g,h} \Rightarrow E_{\{g,h\}})$$

This states that, at any given point in time, if a message is sent from node g to node h, i.e., $S_{g,h}$ is *True*, then there must be an edge between g and h at that point in time, ie., $E_{\{g,h\}}$ is *True*. This constraint and, indeed, edge propositions are only needed when a class of graph topologies for N is being considered where edges may be present in some topologies in the class and absent in others. If a fixed graph topology is under consideration, we do not need edge propositions as we can restrict, once and for all, the set of send propositions to pairs of nodes between which we know we have edges.

4.3. Message-Received Propositions

This research has a set of propositions M_g for the nodes $g \in N$

$$\{M_g \mid g \in N\}$$

Such that, in any given round, M_g is *True* iff node g receives a message. In our model of the flooding algorithm, after the initial round, node g holds a message iff a message is received by g in that round, i.e., some neighbour node h sends a message to g in that round-see the first conjunct in ϕ_m below. However, node h will only send a message to g if g did not send a message to h in the previous round-see the second conjunct in ϕ_m below.

$$\phi_m \equiv (\mathbf{XG} \bigwedge_{g \in N} (M_g \Leftrightarrow \bigvee_{\substack{h \in N, \\ h \neq g}} S_{h,g})) \wedge$$
$$(\mathbf{XG} \bigwedge_{\substack{g,h \in N, \\ g \neq h}} (S_{g,h} \Leftrightarrow \mathbf{Y}(M_g \wedge \neg S_{h,g})))$$

4.4. Initial Conditions

The initial temporal state corresponds to the initial round when some initial node holds a message which is then sent to all its neighbours in the next round, thus triggering network flooding. Therefore, M_g will be true for exactly one $i_0 \in N$ and, as our send-message propositions are true in the round that the corresponding message is received, no send-message proposition is true in the initial round. These two conditions are captured in the two outer-level conjuncts below:

$$\phi_i \equiv (M_{i_0} \wedge \bigwedge_{g \in N, g \neq i_0} \neg M_g) \wedge (\bigwedge_{g,h \in N} \neg S_{g,h})$$

To vary the initial node, one can use the following variable version:

$$\phi_{i_v} \equiv (\bigvee_{i \in M} M_i \wedge \bigwedge_{g \in N, g \neq i} \neg M_g) \wedge (\bigwedge_{g,h \in N} \neg S_{g,h})$$

4.5. Topological Constraints

Section 4.1stated that edges of *N* can be defined in one of three ways:

1. Define a single topology for *N* explicitly by listing the edges.
2. Define a class of topologies for *N* implicitly by defining constraints on edges in *N*.
3. Allow for all topologies on *N*.

Case (3) means that there are no constraints. This research gives a worked example of case (1) later in the paper. Here, the research considers case (2) and show how common classes of network topologies, that are of interest in network flooding, can be defined by Boolean constraints on the propositions

$$E_{\{g,h\}}(g,h \in N).$$

4.5.1. Regular Graphs

Suppose that *N* has *n* nodes

$$N = \{g_1,\ldots,g_n\}$$

A regular graph with nodes *N* has degree *m*, where $1 \leq m \leq n-1$, i.e., every node $g \in N$ has *m* neighbours. The class of all regular topologies on *N* is specified by the following condition on the edge propositions below. This research denotes the cardinality of a set *H* by $|H|$.

$$\phi_{top} \equiv \bigvee_{1\leq m<n-1} \bigwedge_{g\in N} \bigvee_{\substack{H\subseteq N-\{g\},\\ |H|=m}} (\bigwedge_{h\in H} E_{\{g,h\}} \wedge \bigwedge_{h\notin H} \neg E_{\{g,h\}})$$

For the set of nodes $N = \{1,2,3,4\}$, ϕ_{top} instantiates to:

$$\begin{aligned}
&((E_{\{1,2\}} \wedge \neg E_{\{1,3\}} \wedge \neg E_{\{1,4\}}) \vee (E_{\{1,3\}} \wedge \neg E_{\{1,2\}} \wedge \neg E_{\{1,4\}})\\
&\vee (E_{\{1,4\}} \wedge \neg E_{\{1,2\}} \wedge \neg E_{\{1,3\}})) \wedge ((E_{\{2,1\}} \wedge \neg E_{\{2,3\}} \wedge \neg E_{\{2,4\}})\\
&\vee (E_{\{2,3\}} \wedge \neg E_{\{2,1\}} \wedge \neg E_{\{2,4\}}) \vee (E_{\{2,4\}} \wedge \neg E_{\{2,1\}} \wedge \neg E_{\{2,3\}}))\\
&\wedge ((E_{\{3,1\}} \wedge \neg E_{\{3,2\}} \wedge \neg E_{\{3,4\}}) \vee (E_{\{3,2\}} \wedge \neg E_{\{3,1\}} \wedge \neg E_{\{3,4\}})\\
&\vee (E_{\{3,4\}} \wedge \neg E_{\{3,1\}} \wedge \neg E_{\{3,2\}})) \wedge ((E_{\{4,1\}} \wedge \neg E_{\{4,2\}} \wedge \neg E_{\{4,3\}})\\
&\vee (E_{\{4,2\}} \wedge \neg E_{\{4,1\}} \wedge \neg E_{\{4,3\}}) \vee (E_{\{4,3\}} \wedge \neg E_{\{4,1\}} \wedge \neg E_{\{4,2\}})))\\
&\vee (((E_{\{1,2\}} \wedge E_{\{1,3\}} \wedge \neg E_{\{1,4\}}) \vee (E_{\{1,3\}} \wedge \neg E_{\{1,2\}} \wedge E_{\{1,4\}})\\
&\vee (E_{\{1,4\}} \wedge E_{\{1,2\}} \wedge \neg E_{\{1,3\}})) \wedge ((E_{\{2,1\}} \wedge E_{\{2,3\}} \wedge \neg E_{\{2,4\}})\\
&\vee (E_{\{2,3\}} \wedge \neg E_{\{2,1\}} \wedge E_{\{2,4\}}) \vee (E_{\{2,4\}} \wedge E_{\{2,1\}} \wedge \neg E_{\{2,3\}}))\\
&\wedge ((E_{\{3,1\}} \wedge E_{\{3,2\}} \wedge \neg E_{\{3,4\}}) \vee (E_{\{3,2\}} \wedge \neg E_{\{3,1\}} \wedge E_{\{3,4\}})\\
&\vee (E_{\{3,4\}} \wedge E_{\{3,1\}} \wedge \neg E_{\{3,2\}})) \wedge ((E_{\{4,1\}} \wedge E_{\{4,2\}} \wedge \neg E_{\{4,3\}})\\
&\vee (E_{\{4,2\}} \wedge \neg E_{\{4,1\}} \wedge E_{\{4,3\}}) \vee (E_{\{4,3\}} \wedge E_{\{4,1\}} \wedge \neg E_{\{4,2\}})))
\end{aligned}$$

4.5.2. Expander Graphs

Expanders are a very important class of graphs (having the property of being simultaneously sparse and well connected) that have applications in various areas of computer science and mathematics-in the design and analysis of communication networks, cryptography, error correcting codes, pseudo randomness, complexity, coding theory, metric embeddings etc (for details, see this well-knownsurvey [10]. For example, in the context of distributed computer networks, they have been used for building censorship resistant networks [5, 6], fault tolerant networks [24], efficient (Byzantine) agreement and leader election algorithms [4, 13, 15, 26] and analysing information spreading etc. [8]. Thus, even efficient construction (in static or dynamic fault-tolerant settings) of expander networks is an important line of research [7, 16, 20, 21, 22].

Intuitively, an 'expander' graph *N* is one where every subset $S \subseteq N$ of vertices expands 'quickly'. How quickly it expands is determined by an 'expansion parameter'. A graph *N* has expansion parameter ϵ if, for every subset $S \subseteq N$ with $|S| \leq |N|/2$, the set of edges connecting nodes in *S* with nodes not in *S* is greater than or equal to $\epsilon|S|$. To constrain the network *N* to topologies with expansion parameter, use the following Boolean constraint on propositions:

$$\begin{aligned}
\phi_{top} \equiv &\bigwedge_{\substack{S\subseteq N,\\ |S|\leq|N|/2}} \bigvee_{\substack{T\subseteq N\{\times\}N,\\ |T|\geq\epsilon|S|}} \bigwedge_{\{g,h\}\in T} (E_{\{g,h\}} \wedge (\{g,h\} \cap S \neq \emptyset)\\
&\wedge (\{g,h\} \not\subseteq S))
\end{aligned}$$

Here, the set $N\{\times\}N$ is the set of ordered pairs of nodes (g, h) in the cartesian product $N\times N$ viewed as two-element sets $\{g, h\}$ (so that $\{g, h\} = \{h, g\}$, whereas (g, h) 6= (h, g)). Also, $\{g, h\} \cup S \neq \emptyset$ and $\{g, h\} \not\subseteq S$ are evaluated to True or False accordingly in each respective conjunct. The constraint essentially states that corresponding to every subset of nodes *S*, with $|S| \leq |N|/2$, there is a set of edges *T*, where $|T| \geq \epsilon|S|$, each of which connects a node in *S* with a node not in *S*.

4.6. Termination

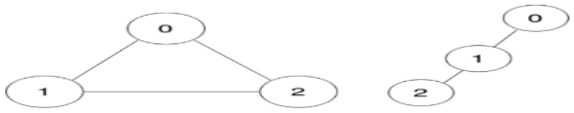
The required property that first comes to mind in network flooding is termination. Termination occurs if, in some round, no node in the system receives a message. In our temporal model, this means that no message-received proposition m_g will be true. So, if network flooding is modelled by ϕ_e,ϕ_s,ϕ_m,ϕ_i and ϕ_{top} as above, then the proof obligation for termination is:

$$\phi_e \wedge \phi_s \wedge \phi_m \wedge \phi_i \wedge \phi_{top} \Rightarrow \mathbf{F} \bigwedge_{g\in N} \neg m_g$$

5. Applications

This research uses its specification of flooding to compare the time it takes for the flooding algorithm to terminate in different topologies. Whilst standard LTL is not designed for resolving timing issues, we can determine which network topology takes fewer rounds to terminate by superimposing the temporal behaviour of the network in one topology over the behaviour in the other topology. So, the temporal model has two

cases of network flooding, on the same set of nodes but with different connections, proceeding together in rounds in lock-step fashion, and with two messages-one for each topological case-circulating in the network. We can illustrate this model with a simple example. Suppose that N={0, 1, 2} and the two topologies are as in the following figure:



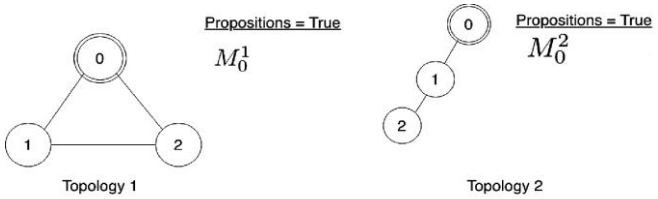
a) Topology 1. b) Topology 2.

Figure 3. Flooding on two topologies.

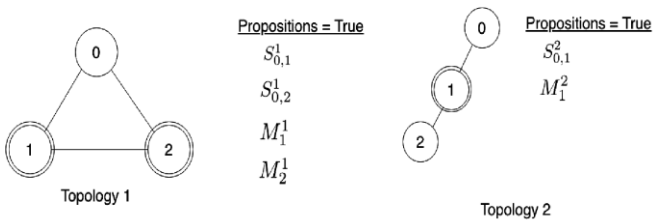
Assume 0 is the initial node in both cases. We illustrate the progression of the rounds in terms of the send-message propositions $S_{g,h}$ and message-received propositions M_g. Distinguishing these propositions for the two topologies, we have the following propositions:

$$Topology1\colon S^1_{0,1}, S^1_{0,2}, S^1_{1,2}, M^1_0, M^1_1, M^1_2$$
$$Topology2\colon S^2_{0,1}, S^2_{1,2}, M^2_0, M^2_1, M^2_2$$

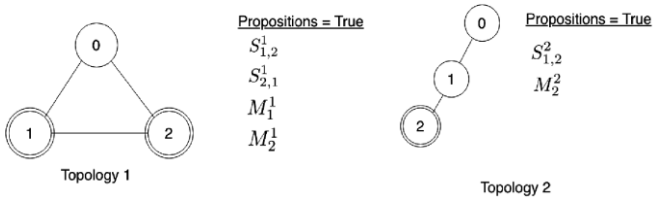
The propositions that are true in successive rounds in the two models are shown in Figures 4-a to 4-d below. Nodes which hold a message are circled.



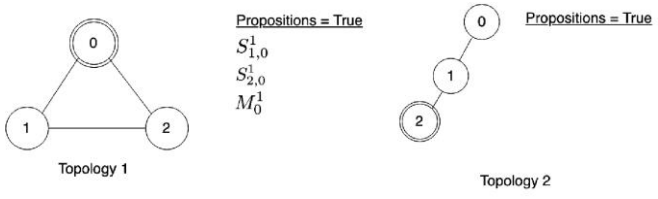
a) Round 0.



b) Round 1.



c) Round 2.



d) Round 3.

Figure 4. Flooding rounds on two topologies.

Notice that in round 3 M^1_0 is True, whereas none of M^2_0, M^2_1, or M^2_2 is True. So, Topology2 terminates before Topology1 as there is a round where no node holds a message in Topology2, whereas a node still holds a message in Topology1. We give the formal proof that has to be carried out in the general case next.

Given two topologies Topology1 and Topology2 on a set of nodes N, the proof obligation that Topology1 terminates before Topology2 is:

$$(\phi^1_e \wedge \phi^1_s \wedge \phi^1_m \wedge \phi^1_i \wedge \phi^1_{top}) \wedge (\phi^2_e \wedge \phi^2_s \wedge \phi^2_m \wedge \phi^2_i \wedge \phi^2_{top}) \Rightarrow \mathbf{F}((\bigwedge_{g\in N} \neg M^1_g) \wedge (\bigvee_{g\in N} M^2_g)) \quad (1)$$

Here, ϕ^1_e, ϕ^1_s and ϕ^1_m relabel the propositional variables from ϕ_e, ϕ_s and ϕ_m of subsections 4.1, 4.2, 4.3 respectively, adding a superscript 1, and ϕ^2_e, ϕ^2_s and ϕ^2_m do the same but with a superscript 2. The formulae ϕ^1_i and ϕ^2_i also possibly differ in their respective initial nodes, and ϕ^1_{top} and ϕ^2_{top} according as to the topologies that they define. In (1) we check for validity. So, if ϕ^1_{top} and ϕ^2_{top} each define a range of topologies, (1) is True (valid) iff all topologies of ϕ^1_{top} terminate before all topologies of ϕ^2_{top}. If (1) returns False, then some topology of ϕ^2_{top} terminates before some topology of ϕ^1_{top}. We could then proceed to test if all topologies of ϕ^2_{top} terminate before all those of ϕ^1_{top}, by checking the validity of:

$$(\phi^2_e \wedge \phi^2_s \wedge \phi^2_m \wedge \phi^2_i \wedge \phi^2_{top}) \wedge (\phi^1_e \wedge \phi^1_s \wedge \phi^1_m \wedge \phi^1_i \wedge \phi^1_{top}) \Rightarrow \mathbf{F}((\bigwedge_{g\in N} \neg M^2_g) \wedge (\bigvee_{g\in N} M^1_g)) \quad (2)$$

It is possible that (2) would also return False, in which case some topologies of ϕ^1_{top} would terminate before some topologies of ϕ^2_{top} and also some topologies of ϕ^2_{top} would terminate before some topologies of ϕ^1_{top}. Apart from varying topologies on the network N, we could also vary the initial node. This can be achieved by replacing initial conditions ϕ_i that have a fixed initial node, by initial conditions ϕ_{i_v} that vary the initial node, in the proof obligation. Thus,

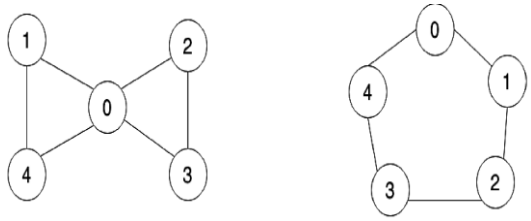
$$(\phi^1_e \wedge \phi^1_s \wedge \phi^1_m \wedge \phi^1_{i_v} \wedge \phi^1_{top}) \wedge (\phi^2_e \wedge \phi^2_s \wedge \phi^2_m \wedge \phi^2_{i_v} \wedge \phi^2_{top}) \Rightarrow \mathbf{F}((\bigwedge_{g\in N} \neg M^1_g) \wedge (\bigvee_{g\in N} M^2_g)) \quad (3)$$

is valid iff, for all topologies ϕ^1_{top} starting from any initial node, flooding terminates before flooding terminates in any topology ϕ^2_{top} with any initial node.

6. Worked Example

Here, we compare termination for two topologies on a network of five nodes by the use of formal proofs. The two network topologies for this example are depicted

in Figure 5.



a) Topology 1. b) Topology 2.

Figure 5. Two topologies of five nodes.

Firstly, we will test to see if one topology terminates before the other, with both having initial node 0. As mentioned in 4.2 we may optimize the number of propositions used, by only having send-message propositions for the edges that are present in each of the respective topologies, as we are comparing two fixed topologies. This restriction of send-message variables for each topology also defines the topology and no additional variable edge propositions $E_{i,j}$ are required. Thus, we have the following propositions for the two topologies in Figure 5 above:

$$\begin{aligned} Topology1: \quad & S^1_{0,1}, S^1_{1,0}, S^1_{0,2}, S^1_{2,0}, S^1_{0,3}, S^1_{3,0}, \\ & S^1_{0,4}, S^1_{4,0}, S^1_{1,4}, S^1_{4,1}, S^1_{2,3}, S^1_{3,2}, \\ & M^1_0, M^1_1, M^1_2, M^1_3, M^1_4 \\ Topology2: \quad & S^2_{0,1}, S^2_{1,0}, S^2_{1,2}, S^2_{2,1}, S^2_{2,3}, S^2_{3,2}, S^2_{3,4}, \\ & S^2_{4,3}, S^2_{0,4}, S^2_{4,0}, M^2_0, M^2_2, M^2_2, M^2_3, M^2_4 \end{aligned}$$

As there are no edge propositions, we ignore $ø_e$, $ø_s$ and $ø_{top}$ and only consider $ø_i$ and $ø_m$ for each topology. Instantiating the definitions of $ø_i$ and $ø_m$ of subsections 4.4 and 4.3 respectively, yields:

$$\begin{aligned} \phi^1_i \quad \equiv \quad & M^1_0 \wedge \neg M^1_1 \wedge \neg M^1_2 \wedge \neg M^1_3 \wedge \neg M^1_4 \wedge \\ & \neg(S^1_{0,1} \vee S^1_{1,0} \vee S^1_{0,2} \vee S^1_{2,0} \vee S^1_{0,3} \vee S^1_{3,0} \vee \\ & S^1_{0,4} \vee S^1_{4,0} \vee S^1_{1,4} \vee S^1_{4,1} \vee S^1_{2,3} \vee S^1_{3,2}) \end{aligned}$$

$$\begin{aligned} \phi^1_m \quad \equiv \quad & (\mathrm{XG}(\quad (M^1_0 \Leftrightarrow S^1_{1,0} \vee S^1_{2,0} \vee S^1_{3,0} \vee S^1_{4,0}) \wedge \\ & (M^1_1 \Leftrightarrow S^1_{0,1} \vee S^1_{4,1}) \wedge \\ & (M^1_2 \Leftrightarrow S^1_{0,2} \vee S^1_{3,2}) \wedge \\ & (M^1_3 \Leftrightarrow S^1_{0,3} \vee S^1_{2,3}) \wedge \\ & (M^1_4 \Leftrightarrow S^1_{0,4} \vee S^1_{1,4})) \ ) \ \wedge \\ & (\mathrm{XG}(\quad (S^1_{0,1} \Leftrightarrow \mathrm{Y}(M^1_0 \wedge \neg S^1_{1,0})) \wedge \\ & (S^1_{1,0} \Leftrightarrow \mathrm{Y}(M^1_1 \wedge \neg S^1_{0,1})) \wedge \\ & (S^1_{2,0} \Leftrightarrow \mathrm{Y}(M^1_2 \wedge \neg S^1_{0,2})) \wedge \\ & (S^1_{0,2} \Leftrightarrow \mathrm{Y}(M^1_0 \wedge \neg S^1_{2,0})) \wedge \\ & (S^1_{0,3} \Leftrightarrow \mathrm{Y}(M^1_0 \wedge \neg S^1_{3,0})) \wedge \\ & (S^1_{3,0} \Leftrightarrow \mathrm{Y}(M^1_3 \wedge \neg S^1_{0,3})) \wedge \\ & (S^1_{0,4} \Leftrightarrow \mathrm{Y}(M^1_0 \wedge \neg S^1_{4,0})) \wedge \\ & (S^1_{4,0} \Leftrightarrow \mathrm{Y}(M^1_4 \wedge \neg S^1_{0,4})) \wedge \\ & (S^1_{1,4} \Leftrightarrow \mathrm{Y}(M^1_1 \wedge \neg S^1_{4,1})) \wedge \\ & (S^1_{4,1} \Leftrightarrow \mathrm{Y}(M^1_4 \wedge \neg S^1_{1,4})) \wedge \\ & (S^1_{2,3} \Leftrightarrow \mathrm{Y}(M^1_2 \wedge \neg S^1_{3,2})) \wedge \\ & (S^1_{3,2} \Leftrightarrow \mathrm{Y}(M^1_3 \wedge \neg S^1_{2,3}))) \ ) \end{aligned}$$

$$\begin{aligned} \phi^2_i \quad \equiv \quad & M^2_0 \wedge \neg M^2_1 \wedge \neg M^2_2 \wedge \neg M^2_3 \wedge \neg M^2_4 \wedge \\ & \neg(S^2_{0,1} \vee S^2_{1,0} \vee S^2_{1,2} \vee S^2_{2,1} \vee \\ & S^2_{2,3} \vee S^2_{3,2} \vee S^2_{3,4} \vee S^2_{4,3} \vee S^2_{0,4} \vee S^2_{4,0}) \end{aligned}$$

$$\begin{aligned} \phi^2_m \quad \equiv \quad & (\mathrm{XG}(\quad (M^2_0 \Leftrightarrow S^2_{1,0} \vee S^2_{4,0}) \wedge \\ & (M^2_1 \Leftrightarrow S^2_{0,1} \vee S^2_{2,1}) \wedge \\ & (M^2_2 \Leftrightarrow S^2_{1,2} \vee S^2_{3,2}) \wedge \\ & (M^2_3 \Leftrightarrow S^2_{2,3} \vee S^2_{4,3}) \wedge \\ & (M^2_4 \Leftrightarrow S^2_{3,4} \vee S^2_{0,4})) \ ) \ \wedge \\ & (\mathrm{XG}(\quad (S^2_{0,1} \Leftrightarrow \mathrm{Y}(M^2_0 \wedge \neg S^2_{1,0})) \wedge \\ & (S^2_{1,0} \Leftrightarrow \mathrm{Y}(M^2_1 \wedge \neg S^2_{0,1})) \wedge \\ & (S^2_{1,2} \Leftrightarrow \mathrm{Y}(M^2_1 \wedge \neg S^2_{2,1})) \wedge \\ & (S^2_{2,1} \Leftrightarrow \mathrm{Y}(M^2_2 \wedge \neg S^2_{1,2})) \wedge \\ & (S^2_{2,3} \Leftrightarrow \mathrm{Y}(M^2_2 \wedge \neg S^2_{3,2})) \wedge \\ & (S^2_{3,2} \Leftrightarrow \mathrm{Y}(M^2_3 \wedge \neg S^2_{2,3})) \wedge \\ & (S^2_{3,4} \Leftrightarrow \mathrm{Y}(M^2_3 \wedge \neg S^2_{4,3})) \wedge \\ & (S^2_{4,3} \Leftrightarrow \mathrm{Y}(M^2_4 \wedge \neg S^2_{3,4})) \wedge \\ & (S^2_{4,0} \Leftrightarrow \mathrm{Y}(M^2_4 \wedge \neg S^2_{0,4})) \wedge \\ & (S^2_{0,4} \Leftrightarrow \mathrm{Y}(M^2_0 \wedge \neg S^2_{4,0}))) \ ) \end{aligned}$$

To prove that Topology1 terminates before Topology2 with initial node 0 for both, we need to prove (by (2) of section 5 above, ignoring $ø_e$, $ø_s$, and $ø_{top}$):

$$\begin{aligned} & (\phi^1_m \wedge \phi^1_i) \wedge (\phi^2_m \wedge \phi^2_i) \Rightarrow \\ & \mathbf{F}((\neg M^1_0 \wedge \neg M^1_1 \wedge \neg M^1_2 \wedge \neg M^1_3 \wedge \neg M^1_4) \wedge \\ & (M^2_0 \vee M^2_1 \vee M^2_2 \vee M^2_3 \vee M^2_4)) \end{aligned}$$

This proof has been carried out using NuSMV and does return true, showing that Topology1 terminates before Topology2 with initial node 0 for both. As a check of our specification, we have also used NuSMV to prove that the following expression, which states that Topology2 terminates before Topology1 with initial node 0 for both,

$$\begin{aligned} & (\phi^2_m \wedge \phi^2_i) \wedge (\phi^1_m \wedge \phi^1_i) \\ & \Rightarrow \mathbf{F}((\neg M^2_0 \wedge \neg M^2_1 \wedge \neg M^2_2 \wedge \neg M^2_3 \wedge \neg M^2_4) \wedge \\ & (M^1_0 \vee M^1_1 \vee M^1_2 \vee M^1_3 \vee M^1_4)) \end{aligned}$$

Is False. Indeed, NuSMV does return False. As Topology1 has been proved to terminate before Topology2 with initial nodes 0, we consider the possibility of Topology1 terminating before Topology2 whichever initial node is chosen for each. By the discussion in subsection 4.4, this means replacing ϕ^1_i and ϕ^2_i by the following $\phi^1_{i_v}$ and $f^2_{i_v}$ respectively:

$$\begin{aligned} \phi^1_{i_v} \equiv & (M^1_0 \wedge \neg M^1_1 \wedge \neg M^1_2 \wedge \neg M^1_3 \wedge \neg M^1_4 \vee \\ & \neg M^1_0 \wedge M^1_1 \wedge \neg M^1_2 \wedge \neg M^1_3 \wedge \neg M^1_4 \\ & \vee \neg M^1_0 \wedge \neg M^1_1 \wedge M^1_2 \wedge \neg M^1_3 \wedge \neg M^1_4 \vee \\ & \neg M^1_0 \wedge \neg M^1_1 \neg \wedge \neg M^1_2 \wedge M^1_3 \wedge \neg M^1_4 \vee \\ & \neg M^1_0 \wedge \neg M^1_1 \neg \wedge \neg M^1_2 \wedge \neg M^1_3 \wedge M^1_4) \end{aligned}$$

$$\phi^2_{i_v} \equiv (M^{\,2}_{\,0} \wedge \neg M^{\,2}_{\,1} \wedge \neg M^{\,2}_{\,2} \wedge \neg M^{\,2}_{\,3} \wedge \neg M^{\,2}_{\,4} \vee$$
$$\neg M^{\,2}_{\,0} \wedge M^{\,2}_{\,1} \wedge \neg M^{\,2}_{\,2} \wedge \neg M^{\,2}_{\,3} \wedge \neg M^{\,2}_{\,4} \vee$$
$$\neg M^{\,2}_{\,0} \wedge \neg M^{\,2}_{\,1} \wedge M^{\,2}_{\,2} \wedge \neg M^{\,2}_{\,3} \wedge \neg M^{\,2}_{\,4} \vee$$
$$\neg M^{\,2}_{\,0} \wedge \neg M^{\,2}_{\,1} \neg \wedge \neg M^{\,2}_{\,2} \wedge M^{\,2}_{\,3} \wedge \neg M^{\,2}_{\,4} \vee$$
$$\neg M^{\,2}_{\,0} \wedge \neg M^{\,2}_{\,1} \neg \wedge \neg M^{\,2}_{\,2} \wedge \neg M^{\,2}_{\,3} \wedge M^{\,2}_{\,4})$$

So, the proof obligation for Topology1 always terminating before Topology2, for all initial nodes, is to check validity of:

$$(\phi^1_m \wedge \phi^1_{i_v}) \wedge (\phi^2_m \wedge \phi^2_{i_v}) \Rightarrow$$

$$\mathbf{F}((\neg M^{\,1}_{\,0} \wedge \neg M^{\,1}_{\,1} \wedge \neg M^{\,1}_{\,2} \wedge \neg M^{\,1}_{\,3} \wedge \neg M^{\,1}_{\,4}) \wedge$$
$$(M^{\,2}_{\,0} \vee M^{\,2}_{\,1} \vee M^{\,2}_{\,2} \vee M^{\,2}_{\,3} \vee M^{\,2}_{\,4}))$$

Where we substitute the $\phi^1_{i_v}$ and $\phi^2_{i_v}$ given above. The result of executing the proof in NuSMV is true, i.e., Topology1 terminates before Topology2 wherever we start in each.

7. Conclusions

We have provided a specification of network flooding in propositional linear temporal logic suitable for proving termination properties. The specification can cater for any class of graph topologies of a given size of network. It does not cater for networks of arbitrary size. A temporal-logic specification of flooding for networks of arbitrary size would need to use first-order temporal logic. Although, first-order temporal logic can specify problems of unlimited size-for example the specification of a transactional system over an unbounded number of data items in [11]-there are practical and theoretical obstacles to formal verification in such logics. Even with the specifications here, and the use of one of the most powerful model checkers available, NuSMV, proofs will only be possible in practice for fairly small sizes of network. Nevertheless, experimentation with network topologies on a small scale can provide insight into the design of networks on a larger scale. The intended use of the approach here is to facilitate such design of network hardware and software by experimentation with different topologies and also different code/algorithms at nodes. The flooding problem gives an example of a very basic algorithm at a network node - on receipt of a message a node sends on the message to all neighbours from which it did not receive the message. In the same way as network topologies can easily be changed by changing the topological constraints, so also can the code/algorithm at nodes be changed by supplying new, possibly more sophisticated, message processing constraints which can then be verified.

References

[1] Attiya H. and Welch J., *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*, John Wiley and Sons, 2004.

[2] Cardelli L. and Gordon A., "Mobile Ambients," *Theoretical Computer Science*, vol. 240, no. 1, pp. 177-213, 2000.

[3] Comer D. and Droms R., *Computer Networks and Internets*, Prentice-Hall, 2003.

[4] Dwork C., Peleg D., Pippenger N., and Upfal E., "Fault Tolerance in Networks of Bounded Degree," *Society for Industrial and Applied Mathematics*, vol. 17, no. 5, pp. 975-988, 1988.

[5] Fiat A. and Saia J., "Censorship resistant peer-to-Peer Content Addressable Networks," *in Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithm*, USA, pp. 94-103, 2002.

[6] Fiat A. and Saia J., "Censorship Resistant Peer-To-Peer Networks," *Theory of Computing*, vol. 3, no. 1, pp.1-23, 2007.

[7] Gkantsidis C., Mihail M., and Saberi A., "Random Walks in Peer-To-Peer Networks: Algorithms and Evaluation," *Performance Evaluation*, vol. 63, no. 3, pp. 241-263, 2006.

[8] Hillel K. and Shachnai H., "Partial Information Spreading with Application to Distributed Maximum Coverage," *in Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, New York, pp. 161-170, 2010.

[9] Hoare C., *Communicating Sequential Processes*, Prentice Hall, 1985.

[10] Hoory S., Linial N., and Wigderson A., "Expander Graphs and their Applications," *Bulletin of the AMS*, vol. 43, no. 04, pp. 439-562, 2006.

[11] Hussak W., "The Serializability Problem for A Temporal Logic of Transaction Queries," *Journal of Applied Non-Classical Logics*, vol. 18, no. 1, pp. 67-78, 2008.

[12] Khan S. and Waheed A., "Modeling and Formal Verification of IMPP," *The International Arab Journal of Information Technology*, vol. 2, no. 3, pp. 192-198, 2005.

[13] King V., Saia J., Sanwalani V., and Vee E., "Towards Secure And Scalable Computation in Peer-To-Peer Networks," *in Proceedings of 47th Annual IEEE Symposium on Foundations of Computer Science*, Berkeley, pp.87-98, 2006.

[14] Kutten S., Pandurangan G., Peleg D., Robinson P., and Trehan A., "On The Complexity of Universal Leader Election," *Journal of the ACM*, vol. 62, no. 1, pp. 1-27, 2015.

[15] Kutten S., Pandurangan G., Peleg D., Robinson P., and Trehan A., "Sublinear Bounds for Randomized Leader Election," *Theoretical Computer Science*, vol. 561, pp. 134-143, 2015.

[16] Law C. and Siu K., "Distributed Construction of Random Expander Networks," *in Proceedings of 20nd Annual Joint Conference of the IEEE*

Computer and Communications Societies, San Francisco, pp. 2133-2143, 2003.

[17] Lynch N. and Tuttle M., "Hierarchical Correctness Proofs for Distributed Algorithms," *in Proceedings of the 6th Annual ACM Symposium on Principles of Distributed computing*, New York, pp. 137-151, 1987.

[18] Milner R., *Communication and Concurrency International Series in Computer Science*, Prentice Hall Englewood Cliffs, 1989.

[19] Milner R., *Communicating and Mobile Systems: the Pi Calculus*, Cambridge University Press, 1999.

[20] Pandurangan G., Raghavan P., and Upfal E., "Building Low-Diameter P2P Networks," *in Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, USA, pp. 492-499, 2001.

[21] Pandurangan G., Robinson P., and Trehan A., "DEX: Self-Healing Expanders," *Distributed Computing*, vol. 29, no. 3, pp. 163-185, 2016.

[22] Pandurangan G. and Trehan A., "Xheal: A Localized Self-Healing Algorithm Using Expanders," *Distributed Computing*, vol. 27, no 1, pp. 39-54, 2014.

[23] Peleg D., "Distributed Computing," *SIAM Monographs on Discrete Mathematics and Applications*, vol. 5, 2000.

[24] Pippenger N. and Lin G., "Fault-Tolerant Circuit-Switching Networks," *SIAM Journal on Discrete Mathematics*, vol. 7, no. 1, pp. 108-118, 1994.

[25] Raynal M., *Distributed Algorithms for Message-Passing Systems*, Springer, 2013.

[26] Upfal E., "Tolerating a Linear Number of Faults in Networks of Bounded Degree," *Information and Computation*, vol. 115, no. 2, pp. 312-320, 1994.

[27] Wan J., Yuan D., and Xu X., "A Review of Routing Protocols In Wireless Sensor Networks," *in Proceedings of 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Dalian, pp. 1-4, 2008.

Ra'ed Bani Abdelrahman is an Assistant Professor of Software Engineering at Ajloun National University, Jordan. He has a Ph.D from Loughborough University, UK and his MSc. and BSc. from Yarmouk University, Jordan. His interest is in modelling safety and liveness properties of critical systems. His other interests are in the area of concurrent systems, network algorithms and protocols.

Rafat Alshorman is an associate professor in the department of computer science at Yarmouk University/Jordan. He completed his Ph.D. at Loughborough University/UK and his undergraduate studies at Yarmouk University/Jordan. His research interests lie in the area of algorithms and mathematical models, ranging from theory to implementation, with a focus on checking the correctness conditions of concurrent and reactive systems. In recent years, he has focused on theoretical computer science such as Graph theory and Numerical analysis. Dr. Alshorman research interests are: 1. Mathematical methods in computer science, 2. Temporal logics, 3. Concurrent systems, 4. machine learning 5. Numerical analysis.

Walter Hussak is an Honorary Fellow in the Department of Computer Science at Loughborough University, UK. Who has retired from lecturing but continues with research and project supervision. He has an MSc in Systems Design and a PhD in Mathematics. As such, he has broad interests in theoretical computer science and mathematics. Most of his work has been in the theory and applications of temporal logics, concurrent systems and graph theory.

Amitabh Trehan is a Lecturer (i.e., Assistant Professor) of Computer Science at Loughborough University, UK. His research interests center around designing provably efficient algorithms and modelling and reasoning about multi-agent dynamic scenarios using tools such as graph theory and game theory. In particular, he has a large body of work on distributed algorithms - in static networks (e.g. Leader Election and flooding) and on dynamic scenarios, in particular, developing self-healing algorithms.