Model Checking MANETs with Arbitrary Mobility

Fatemeh Ghassemi, Saeide Ahmadi, Wan Fokkink, Ali Movaghar

Model Checking MANETs with Arbitrary Mobility

Fatemeh Ghassemi[1], Saeide Ahmadi[1], Wan Fokkink[2], and Ali Movaghar[3]

[1] University of Tehran, Iran
[2] VU University Amsterdam, The Netherlands
[3] Sharif University of Technology, Tehran, Iran

Abstract. Modeling arbitrary connectivity changes of mobile ad hoc networks (MANETs) makes application of automated formal verification challenging. We introduced constrained labeled transition systems (CLTSs) as a semantic model to represent mobility. To model check MANET protocol with respect to the underlying topology and connectivity changes, we here introduce a branching-time temporal logic interpreted over CLTSs. The temporal operators, from Action Computation Tree Logic with an unless operator, are parameterized by multi-hop constraints over topologies, to express conditions on successful scenarios of a MANET protocol. We moreover provide a bisimilarity relation with the same distinguishing power for CLTSs as our logical framework.

1 Introduction

In mobile ad hoc networks (MANETs), nodes communicate along multi-hop paths using wireless transceivers. Wireless communication is restricted; only nodes located in the range of a transmitter receive data. Due to e.g. noise in the environment, interferences, and temporary communication link errors, wireless communication is unreliable, which together with mobility of nodes complicates the design of MANET protocols. Formal methods provide valuable tools to design, evaluate and verify such protocols.

We introduced *Restricted Broadcast Process Theory* (*RBPT*) [9] to specify and verify MANETs, taking into account mobility. *RBPT* specifies a MANET by composing nodes using a restricted local broadcast operator. A strong point of *RBPT* is that the underlying topology is not specified in the syntax, which would make it hard to set up the initial topology for each scenario in a verification. In similar approaches, the mobility is modeled as arbitrary manipulation of the underlying topology (given as part of the semantic state), which may make the model infinite and insusceptible to automated verification techniques. Instead in the semantic model of *RBPT*, a *constraint labeled transition system* (*CLTS*) [10], transitions are enriched with so-called network constraints, to restrict the possible topologies. This symbolic representation of network topologies in the semantics is more compact and allows automated verification techniques to investigate families of properties on a unified model.

Properties in MANETs tend to be weaker than in wired networks, due to the topology-dependent behavior of communication, and consequently the need for multi-hop communication between nodes. For instance, an important property in routing or information dissemination protocols is *packet delivery*: If there exists an end-to-end route between two nodes A and B for a long enough period of time, then packets sent by A

will be received by B [7]. To reason about properties that require such topology conditions, we introduce a temporal logic CACTL based on ACTLW [16], which consists of Action CTL [3] with an until operator. Our approach supports flexibility in verifying topology-dependent behavior (without changing the model), and restricting the generality of mobility as opposed to existing approaches. CACTL is interpreted over CLTSs. Path operators are parameterized with multi-hop constraints over the underlying topologies. We present a model checking algorithm for CACTL; a model checker for CACTL is being implemented, using the rewrite logic Maude. This provides a framework, supporting both equational reasoning [9] and model checking of MANET protocols, to verify topology-dependent properties like "existence of a route".

We moreover introduce a novel notion of branching network bisimilarity, based on branching bisimilarity [24], that induces the same identification of CLTSs as CACTL. This relation is finer than the one introduced in [10], due to reliability of communication: A receiving node is not equivalent to a deadlocked node anymore, since in parallel with a sending node, an unsuccessful communication cannot be matched to a communication with no enabled receiver (which is the case in the lossy framework).

2 Related Work

MANET protocols have been studied either using existing formalisms such as SPIN [5, 26, 1] and UPPAAL [26, 27, 15, 8], or introducing specific frameworks mainly with an algebraic approach [20, 21, 18, 13, 17, 23, 14, 7]. Important modeling challenges in MANETs are *local broadcast*, *underlying topology* and *mobility*. The modeling approach using existing formalisms can be summarized as follows: The underlying topology is modeled by a two-dimensional array of Booleans, mobility by explicit manipulation of this matrix, and local broadcast by unicasting to all nodes with whom the sending node is presently connected, using the connectivity matrix. The verification approach tends to be based on model checking techniques restricted to a pre-specified mobility scenario. Lack of support for compositional modeling and arbitrary topology changes has motivated new approaches with a primitive for local broadcast and support of arbitrary mobility. These approaches are CBS#, bKlaim, CWS, CMAN, CMN, ω-calculus, *RBPT*, CSDT, and AWN [20, 21, 18, 13, 17, 23, 9, 14, 7]. The common point among them (except *RBPT*) is implicit manipulation of the underlying topology in the semantics to model arbitrary connectivity changes and mobility. The analysis techniques supported by these frameworks, except bKlaim and AWN, are based on a behavioral congruence relation. In [10] we provided an axiomatization to derive that a specification of a MANET protocol is observably equal to a specification of its desired external behavior. Equational reasoning (applied at the syntactic or the semantic level) requires either abstraction from the actual specification of the MANET protocol, or knowledge about the overall behavior of the MANET beforehand. The model checking approach is useful to investigate specific properties of MANET protocols with less effort and knowledge. The mix of broadcast behavior and mobility leads to state-space explosion, hampering the application of automated verification techniques like model checking. In bKlaim [21], the semantic model is abstracted to a finite labeled transition system such that the mobility information is preserved; a variant of ACTL is

introduced to determine which properties hold if movement of nodes is restricted. To this aim, ACTL operators are parameterized by a set of possible network configurations (topology). However, topology-dependent behavior cannot be checked. AWN [7] verifies topology-dependent behavior properties using CTL [2], by treating a transition label carrying (dis)connectivity information as a predicate of its succeeding state [7] and defining predicates over the topology as part of the syntax. This approach can be extended to algebras, e.g. CMAN and ω-calculus, with (dis)connectivity information on transition labels. However, this approach needs auxiliary strategies to extract predicates from the states and transitions, to restrict connectivity changes during model checking and thus limit the state space. These challenges are tackled with the help of the model checker UPPAAL, by transforming AWN specifications to automata and exploiting an auxiliary automaton which statically restricts connectivity changes [8], similar to [15].

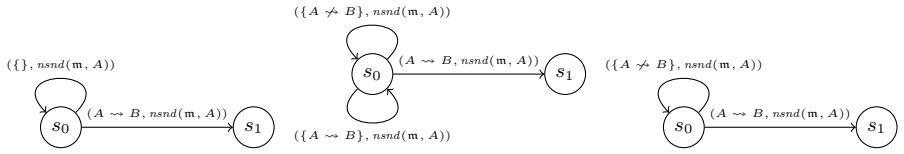
3 Background

Communication in wireless networks tends to be based on local broadcast: Only nodes that are located in the transmission area of a sender can receive. A node B is *directly connected to* a node A, if B is located within the transmission range of A. This asymmetric connectivity relation between nodes introduces a *topology* concept. A topology is a function $\gamma : \mathit{Loc} \to I\!P(\mathit{Loc})$ where Loc denotes a finite set of (hardware) addresses A, B, C. We extend Loc with the unknown address $?$ to model open communications, which is helpful in giving semantics to MANETs in a compositional way.

*Constrained labeled transition system*s (CLTSs) [10] provide a semantic model for the operational behavior of MANETs. A transition label is a pair of an action and a *network constraint*, restricting the range of possible underlying topologies. A network constraint $\mathcal{C}$ is a set of connectivity pairs $\rightsquigarrow$: $\mathit{Loc} \times \mathit{Loc}$, where only the first address can be $?$. In this setting, non-existence of connectivity information between two addresses in a network constraint can imply three consequences; we do not have any information about the link (this is helpful when the link has no effect on the evolution of a network), the link was disconnected, or the link exists, but due to unreliable communication, the communication was unsuccessful. To distinguish these cases from each other, we extend the network constraints of CLTSs with a set of disconnectivity pairs $\not\rightsquigarrow$: $\mathit{Loc} \times \mathit{Loc}$; while $B \rightsquigarrow A$ denotes that A is connected to B directly and consequently A can receive data sent by B, $B \not\rightsquigarrow A$ denotes that A is not connected to B directly and consequently cannot receive any message from B. In this setting, non-existence of connectivity information between two addresses in a network constraint means a lack of information. We write $\{B \rightsquigarrow A, C \ - \ B \not\rightsquigarrow D, E\}$ instead of $\{B \rightsquigarrow A, B \rightsquigarrow C, B \not\rightsquigarrow D, B \not\rightsquigarrow E\}$.

A network constraint $\mathcal{C}$ is said to be *well-formed* if $\forall \ell \rightsquigarrow \ell' \in \mathcal{C}\ (\ell' \neq ? \wedge \ell \not\rightsquigarrow \ell' \notin \mathcal{C})$ and $\forall \ell \not\rightsquigarrow \ell' \in \mathcal{C}\ (\ell' \neq ? \wedge \ell \rightsquigarrow \ell' \notin \mathcal{C})$. Let $\mathbb{C}$ denote the set of well-formed network constraints that can be defined over network addresses in Loc. Each network constraint $\mathcal{C}$ represents the set of network topologies that satisfy the (dis)connectivity pairs in $\mathcal{C}$, i.e., $\{\gamma \mid \mathcal{C} \subseteq \mathcal{C}_\Gamma(\gamma)\}$, where $\mathcal{C}_\Gamma(\gamma)$ extracts all one-hop (dis)connectivity information from γ. So the empty network constraint $\{\}$ denotes all possible topologies over Loc. Let Act_τ be the set of actions (including the silent action τ), ranged over by η.

A *CLTS* is of the form by $\langle S, \Lambda, \rightarrow, s_0 \rangle$, with S a set of states, $\Lambda \subseteq \mathbb{C} \times Act_\tau$, $\rightarrow \subseteq S \times \Lambda \times S$ a transition relation, and $s_0 \in S$ the initial state. A transition $(s, (\mathcal{C}, \eta), s') \in \rightarrow$, denoted by $s \xrightarrow{(\mathcal{C},\eta)} s'$, expresses that a MANET protocol in state s with an underlying topology $\gamma \in \mathcal{C}$ can perform action η to evolve to state s'. Extending network constraints with disconnectivity pairs enables us to define different behaviors for the communication primitive (see Fig. 1). Furthermore, it allows us to reason about existence of a communication path between nodes, as will be explained in Section 4.1.



(a) Lossy communication (b) Disconnectivity as well as lossy communication (c) Reliable communication

Fig. 1. Modeling different communication behaviors: s_0 represents a state in which A broadcasts its data, while s_1 represents a state in which data has been transferred from A to B.

4 Constrained Action Computation Tree Logic

Properties of MANETs tend to be weaker than of wired networks, due to topology-dependent behavior of communication, and consequently the requirement of existence of a multi-hop communication path between nodes. CLTSs provide a suitable platform to verify topology-dependent properties, using the (dis)connectivity information encoded into the transition labels: While transitions are traversed to investigate a behavioral property, (dis)connectivity information is collected to verify the topology conditions on which the behavior depends. To this aim, we introduce a temporal logic based on Action CTL (ACTL) [3] which includes the *until* and *next* operators from CTL [2], parameterized with a set of actions. Recently a more expressive variant of ACTL called ACTLW [16] was introduced, in which the *next* is replaced by an *unless* operator.

4.1 Concepts

Since the behavior of MANET protocols depends on the underlying topology of the network, many properties depend on constraints on this topology. For example, to examine whether a routing protocol can find a route from node A to node B, the existence of a multi-hop path from A to B is a pre-condition. Viewing a network topology as a directed graph, the simplest form of constraint consists of the (non-)existence of multi-hop relations between nodes.

As explained in Section 3, states in a CLTS do not hold information about the underlying topology. E.g., from the transition sequence $t_0 \xrightarrow{(\{A \rightsquigarrow B\}, \eta_1)} t_1 \xrightarrow{(\{B \rightsquigarrow C\}, \eta_2)} t_2$ we can infer that at the moment we reach t_1, B was connected to A, and at the moment we reach t_2, C was connected to B. So we can conclude that to reach t_2 via this path,

two links must exist (not essentially at the same time). That is, a multi-hop communication link from A to C, denoted by $A \dashrightarrow C$, must exist to reach t_2. In general, to examine a property pre-conditioned by a multi-hop constraint over the topology, we look for a path in the CLTS along which the multi-hop relations are inferred.

Let $T = \langle S, \Lambda, \rightarrow, s_0 \rangle$ be a CLTS. A *path* σ of T is a sequence of transitions $t_0(\mathcal{C}_0, \eta_0)t_1(\mathcal{C}_1, \eta_1)t_2 \ldots$ where $\forall i \geq 0\,((t_{i-1}(\mathcal{C}_i, \eta_i)t_i) \in \rightarrow)$. A path is said to be *maximal* if it either is infinite or ends in a deadlock state.

When a multi-hop relation is the pre-condition of a property, we are stating a set of single-hop links (leading to a multi-hop connection) required for a set of communications. Inversely, we can infer from the network constraints of such communications over a path that the multi-hop connection exists. To this aim, we determine multi-hop connections by collecting single-hop constraints along a path in a forward fashion using a set of computations over network constraints. The operator $\oplus : \mathbb{C} \times \mathbb{C} \rightarrow \mathbb{C}$ allows to gather information along a path in a CLTS. It merges connectivity information, where the second argument overwrites conflicting information of the first argument:

$$\mathcal{C}_1 \oplus \mathcal{C}_2 = \mathcal{C}_2 \cup \{p \mid \neg p \notin \mathcal{C}_2 \wedge p \in \mathcal{C}_1\}$$

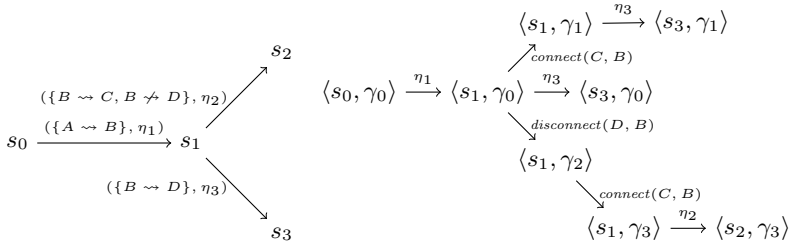
where $\neg(\ell \rightsquigarrow \ell') = \ell \not\rightsquigarrow \ell'$ and $\neg(\ell \not\rightsquigarrow \ell') = \ell \rightsquigarrow \ell'$. This operator is left-associative and non-commutative; $\{\}$ is its identity element. Let $\oplus_{k=i}^{j}\mathcal{C}_k$ denote $(\ldots((\mathcal{C}_{k=i} \oplus \mathcal{C}_{k=i+1}) \oplus \mathcal{C}_{k=i+2})\ldots \oplus \mathcal{C}_{k=j})$. We say $\xi \in \mathbb{C}$ *conforms* to $\mathcal{C}$ if ξ does not include (dis)connectivity information that contradicts $\mathcal{C}$, which can be formally tested by $\mathcal{C} \subseteq \mathcal{C} \oplus \xi$. Extending CLTSs with disconnectivity pairs allows to correctly update topology information gathered along a path when the communication behavior distinguishes lossy from disconnectivity by providing precise information in the labels (see Fig. 1(b) and 1(c)) in the case of unsuccessful communication. For instance, updating the connectivity information $\{A \rightsquigarrow B, C - B \rightsquigarrow C\}$ with $A \not\rightsquigarrow B$ results in $\{A \rightsquigarrow C, B \rightsquigarrow C, A \not\rightsquigarrow B\}$.

A path $t_0(\mathcal{C}_0, \eta_0)t_1(\mathcal{C}_1, \eta_1)t_2 \ldots$ is called *$\mathcal{C}$-path* if $\mathcal{C}_i$ conforms to $\mathcal{C}$ for all $i \geq 0$.

4.2 CACTL Syntax

To provide a logic to verify topology-dependent properties of MANET protocols, our modal path operator is parameterized with multi-hop constraints over the topology. This parameter specifies the pre-condition required for inspecting the property; if the pre-condition never holds, the property does not need to hold. Moreover, to verify properties of MANETs with regard to different mobility scenarios, the satisfaction relation is parameterized with single-hop constraints. This parameter expresses the (non-)existence of communication links and also restricts node mobility; nodes can only move in such a way that the specified links do not change. This is achieved by only traversing transitions that conform to the specified links. For instance, consider the CLTS in Fig. 2(a). We examine properties under the network constraint $\{B \not\rightsquigarrow C\}$, meaning that C is never in the transmission range of B. To this aim we should traverse transitions with network constraints $\mathcal{C}$ such that $\{B \not\rightsquigarrow C\} \subseteq \{B \not\rightsquigarrow C\} \oplus \mathcal{C}$ (like $s_0 \rightarrow s_1 \rightarrow s_3$ in the CLTS in Fig. 2(a)). This can be explained by partially unfolding the CLTS, as depicted in Fig. 2(b), with an initial topology γ_0 where $B \in \gamma_0(A)$, $D \in \gamma_0(B)$ and $C \notin \gamma_0(B)$. Three possible mobility scenarios of state $\langle s_1, \gamma_0 \rangle$ are shown: One moves C in the transmission range of B, while another moves D out and C in the transmission range of B.

According to the mobility restriction, the resulting topologies γ_1 and γ_2 do not satisfy $\{B \rightsquigarrow C\}$. Therefore only the middle scenario to $\langle s_3, \gamma_0 \rangle$ is possible.



(a) The given CLTS (b) Partial unfolded behavior of CLTS in 2(a)

Fig. 2. Restricted mobility, achieved through restricted transition traversal.

Our logic borrows its temporal operators from ACTLW: $\mathbf{AU}$, $\mathbf{EU}$, $\mathbf{AW}$, and $\mathbf{EW}$. Let $\eta \in Act_\tau$, $\ell, \ell' \in Loc$, $\mathcal{C} \in \mathbb{C}$. *Action formula* χ, *topology formula* μ, *state formula* ϕ (also called *CACTL formula*), and *path formula* ψ are defined by the grammars:

$$\begin{array}{rcl} \chi & ::= & \mathit{true} \mid \eta \mid \neg\chi \mid \chi \wedge \chi' \\ \mu & ::= & \mathit{true} \mid \ell \dashrightarrow \ell' \mid \neg\mu \mid \mu \wedge \mu' \\ \phi & ::= & \mathit{true} \mid \neg\phi \mid \phi \wedge \phi' \mid \mathbf{E}\psi \mid \mathbf{A}\psi \\ \psi & ::= & \phi \ {}_{\{\chi\}}\mathbf{U}^{\,\mu}{}_{\{\chi'\}}\phi' \mid \phi \ {}_{\{\chi\}}\mathbf{W}^{\,\mu}{}_{\{\chi'\}}\phi' \end{array}$$

While action and state formulae are the same as in ACTLW, path formulae carry a condition over topologies. Intuitively, a path formula $\phi \ {}_{\{\chi\}}\mathbf{U}^{\,\mu}{}_{\{\chi'\}}\phi'$ specifies a path along which states satisfying property ϕ perform actions from χ, until the accumulated (dis)connectivity information along this path satisfies the topology formula μ, and a state satisfying property ϕ' is reached after an action from χ'. An infinite path that never stabilizes to a situation where μ is always satisfied, still satisfies this path formula if all its states satisfy ϕ and all its transitions are from χ. But if a path does stabilize to such a situation, then eventually a transition from χ' must lead to a state where ϕ' holds. Typically, μ could define that there is a path from node A to node B, and ϕ' could define that some information broadcast by A reaches B.

The path formula based on the unless (weak until) operator $\phi \ {}_{\{\chi\}}\mathbf{W}^{\,\mu}{}_{\{\chi'\}}\phi'$ specifies a path along which states satisfying property ϕ perform actions from χ at least as long as either μ is never satisfied or no state satisfying ϕ' is visited by an actions from χ'. We note that $\mathbf{EW}$ cannot readily be defined in terms of $\mathbf{AU}$ as opposed to CTL, due to actions of χ and χ' that should be visited to reach states satisfying ϕ and ϕ'.

4.3 CACTL Semantics

Let $\eta' \in Act_\tau$, $\zeta \in \mathbb{C}$, and $\langle S, \Lambda, \rightarrow, s_0 \rangle$ a CLTS. Satisfaction under network constraint ζ of action formula χ by $\eta \in Act_\tau$ (written $\eta \models_\zeta \chi$), topology formula μ by network constraint $\mathcal{C}$ (written $\mathcal{C} \models_\zeta \mu$), state formula ϕ by state t (written $t \models_\zeta \phi$), or path formula ψ by maximal path σ (written $\sigma \models_\zeta \psi$), is inductively defined below. Let

σ_i^s, $\sigma_i^{\mathcal{C}}$ and σ_i^{η} denote the i-th state, network constraint and action on path σ. With $\oplus_{k=1}^{\infty}\sigma_k^{\mathcal{C}} \not\models_\zeta \mu$ we mean that $\oplus_{k=1}^{m}\sigma_k^{\mathcal{C}} \models_\zeta \neg\mu$ for infinitely many $m \geq 1$.

$\eta \models_\zeta true$	always
$\eta \models_\zeta \eta'$	iff $\eta = \eta'$
$\eta \models_\zeta \neg\chi$	iff $\eta \not\models_\zeta \chi$
$\eta \models_\zeta \chi \wedge \chi'$	iff $\eta \models_\zeta \chi \wedge \eta \models_\zeta \chi'$
$\mathcal{C} \models_\zeta true$	always
$\mathcal{C} \models_\zeta \ell \dashrightarrow \ell'$	iff there are $\ell_0, \ldots, \ell_n \in Loc$ with $\ell_0 = \ell$, $\ell_n = \ell'$, and $\ell_i \rightsquigarrow \ell_{i+1} \in \zeta \oplus \mathcal{C}$ for all $i = 0, \ldots, n-1$
$\mathcal{C} \models_\zeta \neg\mu$	iff $\mathcal{C} \not\models_\zeta \mu$
$\mathcal{C} \models_\zeta \mu \wedge \mu'$	iff $\mathcal{C} \models_\zeta \mu \wedge \mathcal{C} \models_\zeta \mu'$
$t \models_\zeta true$	always
$t \models_\zeta \neg\phi$	iff $t \not\models_\zeta \phi$
$t \models_\zeta \phi \wedge \phi'$	iff $t \models_\zeta \phi \wedge t \models_\zeta \phi'$
$t \models_\zeta \mathbf{E}\psi$	iff there exists a maximal ζ-path σ such that $t = \sigma_0^s \wedge \sigma \models_\zeta \psi$
$t \models_\zeta \mathbf{A}\psi$	iff for all maximal ζ-paths σ, $t = \sigma_0^s \Rightarrow \sigma \models_\zeta \psi$
$\sigma \models_\zeta \phi\ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'$	iff $\sigma_0^s \models_\zeta \phi$, and either $\oplus_{k=1}^{\infty}\sigma_k^{\mathcal{C}} \not\models_\zeta \mu$, $\forall j \geq 1\,(\sigma_j^s \models_\zeta \phi \wedge (\sigma_j^{\eta} \models_\zeta \chi \wedge \zeta \subseteq \zeta \oplus \sigma_j^{\mathcal{C}}) \vee (\sigma_j^{\eta} = \tau \wedge \sigma_j^{\mathcal{C}} = \{\}))$ or there exists an $i \geq 1$ such that $\sigma_i^s \models_{\zeta \oplus (\oplus_{k=1}^{i}\sigma_k^{\mathcal{C}})} \phi'$, $\sigma_i^{\eta} \models_\zeta \chi'$, $\zeta \subseteq \zeta \oplus \sigma_i^{\mathcal{C}}$, $\oplus_{k=1}^{i}\sigma_k^{\mathcal{C}} \models_\zeta \mu$, and $\forall 1 \leq j < i\,(\sigma_j^s \models_\zeta \phi \wedge ((\sigma_j^{\eta} \models_\zeta \chi \wedge \zeta \subseteq \zeta \oplus \sigma_j^{\mathcal{C}}) \vee (\sigma_j^{\eta} = \tau \wedge \sigma_j^{\mathcal{C}} = \{\})))$
$\sigma \models_\zeta \phi\ {}_{\{\chi\}}\mathbf{W}^{\ \mu}{}_{\{\chi'\}}\phi'$	iff $\sigma \models_\zeta \phi\ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'$, or $\sigma_0^s \models_\zeta \phi$ and $\forall j \geq 1\,(\sigma_j^s \models_\zeta \phi \wedge ((\sigma_j^{\eta} \models_\zeta \chi \wedge \zeta \subseteq \zeta \oplus \sigma_j^{\mathcal{C}}) \vee (\sigma_j^{\eta} = \tau \wedge \sigma_j^{\mathcal{C}} = \{\})))$

We use $\models$ to denote $\models_{\{\}}$ and ${}_{\{\chi\}}\mathbf{U}_{\{\chi'\}}$ to denote ${}_{\{\chi\}}\mathbf{U}^{\ true}{}_{\{\chi'\}}$. We remark that the semantics of the until operator is somewhat different from CTL: $\mathbf{E}(\phi_{\{\chi\}}\mathbf{U}_{\{\chi'\}}\phi')$ in ACTLW (similar to ACTL) requires to perform at least one action from $\{\chi'\}$ to reach a state satisfying ϕ', while $\mathbf{E}(\phi\mathbf{U}\phi')$ in CTL is satisfied if the first state satisfies ϕ'. Furthermore, our semantics explicitly distinguishes silent actions τ (over all possible topologies) and visible actions (similar to ACTL, in contrast to ACTLW).

For example, the state formula $\mathbf{A}(true\ {}_{\{\tau \vee init\}}\mathbf{U}^{\ A \dashrightarrow B}{}_{\{get\}}true)$ indicates that if there exists a path from A to B, the action $init$ is followed by action get, after some communication (specified by τ). E.g., the path

$$\mathcal{M}_0 \xrightarrow{(\{\},init)} \mathcal{M}_1 \xrightarrow{(\{A \rightsquigarrow C\ -\ A \not\rightsquigarrow B, D\},\tau)} \mathcal{M}_2 \xrightarrow{(\{C \rightsquigarrow D\ -\ C \not\rightsquigarrow A, B\},\tau)} \mathcal{M}_3 \xrightarrow{(\{D \rightsquigarrow B\ -\ D \not\rightsquigarrow C\},\tau)} \mathcal{M}_4 \xrightarrow{(\{\},get)} \mathcal{M}_5 \rightarrow \ldots$$

satisfies $(true\ {}_{\{\tau \vee init\}}\mathbf{U}^{\ A \dashrightarrow B}{}_{\{get\}}true)$ with no restriction on mobility of nodes ($\zeta = \{\}$), since get is observed after passing transitions with labels of $init$ and τ, and the accumulated network constraints over these transitions, $\{\} \oplus \{A \rightsquigarrow C\ -\ A \not\rightsquigarrow B, D\} \oplus C \rightsquigarrow D\ -\ C \not\rightsquigarrow A, B\} \oplus \{D \rightsquigarrow B\ -\ D \not\rightsquigarrow C\} \oplus \{\}$ induces that $A \dashrightarrow B$ via C and D.

4.4 CACTL Model Checking

We adapt the CTL model checking algorithm (see [6]) to CACTL. Model checking a CACTL formula φ under $\zeta \in \mathbb{C}$ starts with smallest sub-formulae and works outwards toward φ. The model checking will operate by adding to each state a set of labels of the form $\langle \phi, \Omega, O \rangle$, where ϕ is a subformula of φ, Ω a set of (dis)connectivity pairs, not necessarily well-formed, and O a set of *topology obligations*. The set Ω maintains the history of links experienced during exploration of φ, and is helpful in the verification of state formulae based on (weak) until operators. A topology obligation is a pair of a topology formula μ and a network constraint $\mathcal{C}$. A topology obligation $\langle \mu, \mathcal{C} \rangle$ is said to be satisfied when $\mathcal{C} \models_\zeta \mu$. Initially all states are labeled by $\langle true, \emptyset, \emptyset \rangle$. A state satisfies φ iff at the end it includes a label $\langle \varphi, \Omega, O \rangle$ with the topology obligations in O all satisfied. The pseudo code of the model checking algorithm's backbone is given in Fig. 3.

Procedure $ModelCheck(\varphi, \zeta)$
Initially set the labels of all states to $\{\langle true, \emptyset, \emptyset \rangle\}$;
repeat
  let ϕ be the next innermost formula of φ;
  switch ϕ **do**
    case $\mathbf{E}(\phi\ {}_{\{\chi\}}\mathbf{U}\ {}^{\mu}{}_{\{\chi'\}}\phi')$
      $ModelCheck(\phi, \zeta)$;
      $ModelCheck(\phi', \zeta)$;
      $CheckEU(\phi, \chi, \phi', \chi', \mu, \zeta)$;
    case $\phi \wedge \phi'$
      $ModelCheck(\phi, \zeta)$;
      $ModelCheck(\phi', \zeta)$;
      $CheckAnd(\phi, \phi', \zeta)$;
    ...
  endsw
until $\phi = \varphi$;
if $\exists \langle \varphi, \Omega, O \rangle \in label(s_0) \wedge \forall \langle \mu, \mathcal{C} \rangle \in O \cdot \mathcal{C} \models_\zeta \mu$ **then return** $true$;
else return $false$;

Fig. 3. Model checking algorithm

We explain the idea of procedure *CheckEU* for the $\mathbf{EU}$ operator; other CACTL operators can be dealt with in a similar way. The pseudo code of this procedure is given in Appendix A. For simplicity we assume that CLTSs are deadlock-free. We extend the application of $\oplus$ to topology obligations: $\mathcal{C} \oplus O = \{\langle \mu, \mathcal{C} \oplus \mathcal{C}' \rangle \mid \langle \mu, \mathcal{C}' \rangle \in O\}$. Two possible cases should be examined. In the first case, state s satisfies formula $\mathbf{E}(\phi\ {}_{\{\chi\}}\mathbf{U}\ {}^{\mu}{}_{\{\chi'\}}\phi')$ if there exists a path from s consisting of states satisfying property ϕ under ζ and actions from χ, until a state satisfying ϕ' under $\zeta \oplus \xi$ is reached after an action from χ' and ξ induces μ, where ξ is the accumulated (dis)connectivity information along this path. To check this case, we move backward starting from the states where ϕ' holds under ζ, first over a transition with an action from χ', and then over transitions with an action from χ, passing over states where ϕ holds under ζ. We record the status of links encountered during backward exploration of executions (note that these links conform to ζ). To ensure conformability of the links recorded for ϕ' to $\zeta \oplus \xi$, we incrementally

check conformability of these links to the partial of ξ being formed in the backward exploration. Since the yet unknown ξ should induce μ, we initially include topology obligation $\langle \mu, \{\} \rangle$ in the state label; its network constraint is incrementally updated while moving backward. Furthermore, we record the topology obligation generated during exploration of ϕ and ϕ'. To ensure ϕ' holds under $\zeta \oplus \xi$, we incrementally update its recorded topology obligation while moving backward.

Let Ω and Ω' contain the links that occurred over executions during exploration of ϕ and ϕ', and O and O' the topology obligations generated during exploration of ϕ and ϕ' (under ζ), respectively. Since first ϕ and only after that ϕ' needs to hold, these sets can be kept separate. The sets Ω and Ω' may contain conflicting conditions, and even if Ω' conforms to the partial of ξ, $\Omega \cup \Omega'$ may not. To check conformability of Ω' to and update O' with partial ξ, we postpone mixing these sets until the end, and exploit a senary labeling $\langle \mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'), \Omega, \Omega', \mathcal{C}, O, O' \rangle$. Let $\mathcal{C}$ be the accumulated value of network constraints over the traversed execution path. By moving backward over a $(\mathcal{C}, \eta)$-transition (where $\mathcal{C}$ conforms to ζ and η satisfies χ') from the state labeled with $\langle \phi', \Omega', O' \rangle$ to the state labeled with $\langle \phi, \Omega, O \rangle$, we add the label $\langle \mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'), \Omega \cup \mathcal{C}, \Omega', \mathcal{C}, O, \{\langle \mu, \mathcal{C} \rangle\} \cup \mathcal{C} \oplus O' \rangle$ to states labeled with $\langle \phi, \omega, O \rangle$, if Ω' conforms to $\mathcal{C}$; it should be noted that $\mathcal{C}$ is added to Ω (and Ω conforms to ζ), O' is updated with $\mathcal{C}$, and the obligation $\{\langle \mu, \mathcal{C} \rangle\}$ ($\mathcal{C} \oplus \{\langle \mu, \{\} \rangle\}$) is generated during model checking of $\mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi')$. At the end of model checking, the senary labels $\langle \mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'), \Omega, \Omega', \mathcal{C}, O, O' \rangle$ are replaced by $\langle \mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'), \Omega \cup \Omega', O \cup O' \rangle$.

Next we continue moving backward from the states with labels of the form $\langle \mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'), \Omega, \Omega', \mathcal{C}', O, O' \rangle$ over $(\{\}, \tau)$ or $(\mathcal{C}, \eta)$-transitions of which their action satisfies χ and their network constraint conforms to ζ, to reach states labeled by $\langle \phi, \Omega'', O'' \rangle$ for some Ω'' and O''. We add the label $\langle \mathbf{E}(\phi \ {}_{\{\chi\}}\mathbf{U}^{\ \mu}{}_{\{\chi'\}}\phi'), \Omega \cup \Omega'' \cup \mathcal{C}, \Omega', \mathcal{C} \oplus \mathcal{C}', O \cup O'', \mathcal{C} \oplus O' \rangle$ to these states, if Ω' conforms to $\mathcal{C} \oplus \mathcal{C}'$. We continue moving backward until no new label is added to the states.

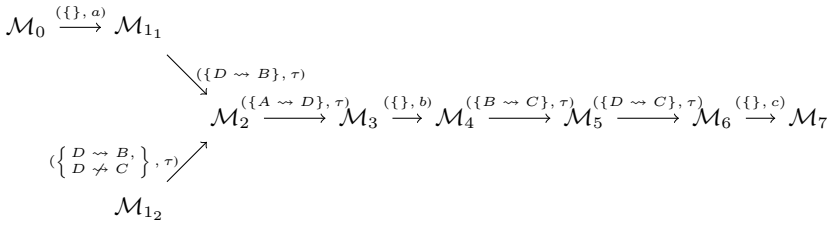


Fig. 4. The CLTS to be checked for $\mathbf{E}(true \ {}_{\{a \vee \tau\}}\mathbf{U}^{\ A \dashrightarrow B}{}_{\{b\}}\mathbf{E}(true \ {}_{\{a \vee \tau\}}\mathbf{U}^{\ A \dashrightarrow C}{}_{\{c\}} true))$.

As an example, we verify $\mathbf{E}(true \ {}_{\{a \vee \tau\}}\mathbf{U}^{\ A \dashrightarrow B}{}_{\{b\}}\mathbf{E}(true \ {}_{\{a \vee \tau\}}\mathbf{U}^{\ A \dashrightarrow C}{}_{\{c\}} true))$ under $\{\}$ over the CLTS given in Fig. 4. States are initially labeled by $\langle true, \emptyset, \emptyset \rangle$. Table 1 includes the labels given to the states in each step; first we label states $\mathcal{M}_7$ to $\mathcal{M}_4$ for the inner until operator, and then we label states $\mathcal{M}_4$ to $\mathcal{M}_0$ for the outer until operator. State $\mathcal{M}_{1_2}$ is only labeled by $\langle true, \emptyset, \emptyset \rangle$ and cannot be labeled further, be-

cause the set of links encountered during exploration of the inner until formula, i.e. $\{B \rightsquigarrow C, D \rightsquigarrow C\}$, does not conform to $\{D \rightsquigarrow B, D \not\rightsquigarrow C\}$. State $\mathcal{M}_0$ includes the label $\langle \varphi_2, \{D \rightsquigarrow B, A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}, \{\langle A \dashrightarrow C, \{D \rightsquigarrow B, A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{D \rightsquigarrow B, A \rightsquigarrow D\}\rangle\}\rangle$, so it satisfies φ_2 (under $\{\}$), since $\{D \rightsquigarrow B, A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\} \models A \dashrightarrow C$ and $\{D \rightsquigarrow B, A \rightsquigarrow D\} \models A \dashrightarrow B$. The topology formula $A \dashrightarrow C$ in the inner until formula is satisfied after the network constraints $A \rightsquigarrow D$ and $D \rightsquigarrow B$ update their corresponding topology obligation while moving backward to model check the outer formula.

Table 1. Labels of states in Fig. 4 while checking formula $\varphi_2 \equiv \mathbf{E}(\phi\ {}_{\{a\vee\tau\}}\mathbf{U}^{\ A\dashrightarrow B}_{\{b\}}\varphi_1)$, where $\varphi_1 \equiv \mathbf{E}(\phi\ {}_{\{\chi\}}\mathbf{U}^{\ A\dashrightarrow C}_{\{c\}}\phi)$ and $\phi \equiv \mathit{true}$.

Steps	Actions
1	$\langle \mathit{true}, \emptyset, \emptyset\rangle$ are added to $\mathcal{M}_{0,1_1,1_2,2-7}$
2	$L_1 \equiv \langle \varphi_1, \emptyset, \emptyset, \{\}, \emptyset, \{\langle A \rightsquigarrow C, \{\}\rangle\}\rangle$ is added to $\mathcal{M}_6$
3	$L_2 \equiv \langle \varphi_1, \{D \rightsquigarrow C\}, \emptyset, \{D \rightsquigarrow C\}, \emptyset, \{\langle A \dashrightarrow C, \{D \rightsquigarrow C\}\rangle\}\rangle$ is added to $\mathcal{M}_5$
4	$L_3 \equiv \langle \varphi_1, \{B \rightsquigarrow C, D \rightsquigarrow C\}, \emptyset, \{D \rightsquigarrow C, B \rightsquigarrow C\}, \emptyset, \{\langle A \dashrightarrow C, \{D \rightsquigarrow C, B \rightsquigarrow C\}\rangle\}\rangle$ is added to $\mathcal{M}_4$
5	L_1 is replaced by $\langle \varphi_1, \emptyset, \{\langle A \dashrightarrow C, \{\}\rangle\}\rangle$ in $\mathcal{M}_6$
6	L_2 is replaced by $\langle \varphi_1, \{D \rightsquigarrow C\}, \{\langle A \dashrightarrow C, \{D \rightsquigarrow C\}\rangle\}\rangle$ in $\mathcal{M}_5$
7	L_3 is replaced by $\langle \varphi_1, \{B \rightsquigarrow C, D \rightsquigarrow C\}, \{\langle A \dashrightarrow C, \{B \rightsquigarrow C, D \rightsquigarrow C\}\rangle\}\rangle$ in $\mathcal{M}_4$
8	$L_4 \equiv \langle \varphi_2, \emptyset, \{B \rightsquigarrow C, D \rightsquigarrow C\}, \{\}, \emptyset, \{\langle A \dashrightarrow C, \{B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{\}\rangle\}\rangle$ is added to $\mathcal{M}_3$
9	$L_5 \equiv \langle \varphi_2, \{A \rightsquigarrow D\}, \{B \rightsquigarrow C, D \rightsquigarrow C\}, \{A \rightsquigarrow D\}, \emptyset, \{\langle A \dashrightarrow C, \{A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{A \rightsquigarrow D\}\rangle\}\rangle$ is added to $\mathcal{M}_2$
10	$L_6 \equiv \langle \varphi_2, \{D \rightsquigarrow B, A \rightsquigarrow D\}, \{B \rightsquigarrow C, D \rightsquigarrow C\}, \{D \rightsquigarrow B, A \rightsquigarrow D\}, \emptyset, \{\langle A \dashrightarrow C, \{D \rightsquigarrow B, A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{D \rightsquigarrow B, A \rightsquigarrow D\}\rangle\}\rangle$ is added to $\mathcal{M}_{1_1}$ and $\mathcal{M}_0$
11	L_4 is replaced by $\langle \varphi_2, \{B \rightsquigarrow C, D \rightsquigarrow C\}, \{\langle A \dashrightarrow C, \{B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{\}\rangle\}\rangle$ in $\mathcal{M}_3$
12	L_5 is replaced by $\langle \varphi_2, \{A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}, \{\langle A \dashrightarrow C, \{A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{A \rightsquigarrow D\}\rangle\}\rangle$ in $\mathcal{M}_2$
13	L_6 is replaced by $\langle \varphi_2, \{D \rightsquigarrow B, A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}, \{\langle A \dashrightarrow C, \{D \rightsquigarrow B, A \rightsquigarrow D, B \rightsquigarrow C, D \rightsquigarrow C\}\rangle, \langle A \dashrightarrow B, \{D \rightsquigarrow B, A \rightsquigarrow D\}\rangle\}\rangle$ in $\mathcal{M}_1$ and $\mathcal{M}_0$

In the second case, s satisfies $\mathbf{E}(\phi\ {}_{\{\chi\}}\mathbf{U}^{\ \mu}_{\{\chi'\}}\phi')$ if there exists a path from s along which the states satisfy ϕ, the actions are from χ, and the accumulated (dis)connectivity information never induces μ permanently (see Fig. 5 for two simple examples). To check the occurrence of this case, we decompose the CLTS into non-trivial strongly connected components (SCCs), meaning that they contain at least one edge.

We first restrict to states that include a label $\langle \phi, \Omega, O\rangle$ for some Ω and O, and to $(\{\}, \tau)$ and $(\mathcal{C}, \eta)$-transitions where η satisfies χ and $\mathcal{C}$ conforms to ζ. Next, we partition the new CLTS into SCCs using the algorithm explained in [2]. We initially move backward in an SCC over $(\mathcal{C}, \eta)$-transitions, and for any $\langle \phi, \Omega_1, O_1\rangle \in \mathit{label}(s)$ and $\langle \phi, \Omega_2, O_2\rangle \in \mathit{label}(t)$, where s and t are origin and destination of transition, we add the label $\langle \mathbf{E}(\phi\ {}_{\{\chi\}}\mathbf{U}^{\ \mu}_{\{\chi'\}}\phi'), \Omega_1 \cup \Omega_2 \cup \mathcal{C}, \emptyset, \mathcal{C}, O_1 \cup O_2, \{\langle \neg\mu, \mathcal{C}\rangle\}\rangle$ to s. The obligation $\langle \neg\mu, \mathcal{C}\rangle$ indicates that the accumulated value of network constraints does not induce μ. Then, similar to first case, we move backward out of the SCC until no new label is added to the states, and at the end we replace senary labels by triples.

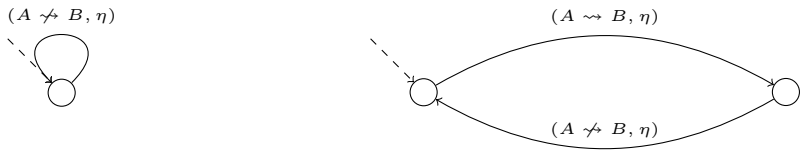


Fig. 5. Two examples of infinite execution paths for which the accumulated network constraints never induce $A \dashrightarrow B$ permanently.

5 Protocol Analysis with CACTL

To illustrate the expressiveness of CACTL in the analysis of MANETs, we specify properties for two important classes of protocols, namely routing and leader election.

The most fundamental error in routing protocol operations is failure to route correctly. The correct operation of MANET routing protocols is defined as follows [26]: *If from some point in time on there exists a path between two nodes, then the protocol must be able to find some path between the nodes. Furthermore, when a path has been found, and for the time it stays valid, it must be possible to send packets along the path from the source node to the destination node*. To verify the first part of property, let $init(src)$ and $found(src)$ indicate initialization and completion respectively of the route discovery protocol in a node with address src for a specific address dst. The property "whenever there exists a path from src to dst and from dst to src, each $init(src)$ is proceeded by its corresponding $found(src)$", for $scr \in \{A, C\}$ and $dst = B$, is specified by the CACTL formula

$$\mathbf{A}(true\ {}_{\{\tau \vee init(A) \vee init(C)\}}\mathbf{U}^{A \dashrightarrow B \wedge B \dashrightarrow A}_{\{found(A)\}} true) \wedge \mathbf{A}(true\ {}_{\{\tau \vee init(C) \vee init(C)\}}\mathbf{U}^{C \dashrightarrow B \wedge B \dashrightarrow C}_{\{found(C)\}} true)$$

where τ abstracts away from communications between nodes. By model checking the CLTS model of a MANET in which the nodes deploy a routing protocol, we can verify this property with respect to arbitrary topology changes. This property was examined in [7] for the AODV protocol using CTL.

To verify the second part of the property, let $insert(src)$ and $delivery(src)$ indicate submission and arrival of a data over the route found beforehand at src and dst respectively. The property "whenever there exists paths from src to dst and from dst to src, when a route is found from src to dst; and while this path is valid each $insert$ is followed by $deliver$", for $src = C$ and $dst = B$, is specified by the CACTL formula

$$\mathbf{A}(true\ {}_{\{init(C) \vee init(A) \vee \tau\}}\mathbf{U}^{C \dashrightarrow B \wedge B \dashrightarrow C}_{\{found(C)\}}(\mathbf{A}(true\ {}_{\{insert \vee \tau\}}\mathbf{U}^{true}_{\{deliver\}} true)))$$

The outer until formula looks for all maximal paths in which $found$ is performed after $init$ while the accumulated network constraints ξ induce $C \dashrightarrow B$ and $B \dashrightarrow C$ (or this topology formula is never induced), and by $found$ reach a state of which all maximal ξ-paths satisfy the inner until path formula. The network constraints over a ξ-path do not violate the single-hop (dis)connectivity pairs in ξ. It can be said that (dis)connectivity pairs are frozen, and consequently the route from src to dst is still valid.

Classical leader election algorithms aim at electing a unique leader from a fixed set of nodes. In the context of MANETs such protocols should consider arbitrary topology

changes, and aim at finding a unique leader which is the most-valued node within a connected component [25]. Let $leader(id, lid)$ indicate that a node with address id has found its leader with address lid, and let node A be the most-valued node. We can investigate correctness of such a leader election algorithm with the CACTL formula

$$\mathbf{A}(true \ {}_{Act_\Lambda}\mathbf{U}^{\substack{A \dashrightarrow B \wedge A \dashrightarrow C \wedge \\ B \dashrightarrow A \wedge B \dashrightarrow C \wedge \\ C \dashrightarrow A \wedge C \dashrightarrow B}}_{\{leader(A,A),leader(B,A),leader(C,A)\}} true)$$

It expresses that in any connected component containing nodes A, B and C, eventually A is chosen as the leader. Being in the same connected component is indicated by the existence of multi-hop paths among them. We can also investigate scenarios in which two disconnected components merge together with the help of a CACTL formula like

$$\mathbf{A}(true \ {}_{Act_\Lambda}\mathbf{U}^{\substack{A \dashrightarrow B \wedge B \dashrightarrow A \wedge \\ C \dashrightarrow D \wedge D \dashrightarrow C \wedge \\ \neg A \dashrightarrow C \wedge \neg A \dashrightarrow D \wedge \\ \neg B \dashrightarrow C \wedge \neg B \dashrightarrow D}}_{\{leader(A,A),leader(B,A),leader(D,C)\}} (\\ \mathbf{A}(true \ {}_{Act_\Lambda}\mathbf{U}^{C \dashrightarrow B \wedge B \dashrightarrow C}_{\{leader(A,A),leader(B,A),leader(D,A),leader(C,A)\}} true))$$

meaning that nodes A, B and nodes C, D belong to the same component with leader A and C respectively; if these two components get connected via B and C, then they will eventually converge to the same leader, i.e. A.

6 Branching Network Bisimilarity

We define a novel notion of branching network bisimilarity that induces the same identification of CLTSs as our logical framework.

Definition 1. *Let $\langle S, Λ, \rightarrow, s_0\rangle$ be a CLTS. States $r, s \in S$ are logically equivalent, denoted by $r \sim_L s$, iff $\forall \zeta \in \mathbb{C}\ \forall \varphi \in CACTL\ (r \models_\zeta \varphi \Leftrightarrow s \models_\zeta \varphi)$.*

Intuitively, equivalent states in a CLTS exhibit the same behavior for any topology. This behavior includes communication and internal actions. Communication actions carry a message and the address of the sender, which can be abstracted into the unknown address ?. Two equivalent states must match on every internal action, receive action, and send action with a known address. A send action with unknown address can be mimicked by a send action with either a known or unknown address. Let $\Longrightarrow$ denote the reflexive-transitive closure of τ-transitions, over all possible topologies.

Definition 2. *A binary relation $\mathcal{R}$ on states in a CLTS is a* branching network simulation *if $t_1 \mathcal{R} t_2$ and $t_1 \xrightarrow{(\mathcal{C},\eta)} t_1'$ implies that:*

- *either $(\mathcal{C}, \eta)$ is $(\{\}, \tau)$, and $t_1' \mathcal{R} t_2$; or*
- *there are t_2' and t_2'' such that $t_2 \Longrightarrow t_2' \xrightarrow{(\mathcal{C},\eta)} t_2''$, where $t_1 \mathcal{R} t_2'$ and $t_1' \mathcal{R} t_2''$; or*
- *$\eta \equiv nsnd(\mathfrak{m}, ?)$, and there are t_2', t_2'' and ℓ such that $t_2 \Longrightarrow t_2' \xrightarrow{(\mathcal{C}[\ell/?], nsnd(\mathfrak{m},\ell))} t_2''$, where $t_1 \mathcal{R} t_2'$ and $t_1' \mathcal{R} t_2''$.*

$\mathcal{R}$ is a branching network bisimulation *if $\mathcal{R}$ and $\mathcal{R}^{-1}$ are branching network simulations. Two terms t_1 and t_2 are branching network bisimilar, denoted by $t_1 \simeq_b t_2$, if $t_1 \mathcal{R} t_2$ for some branching network bisimulation relation $\mathcal{R}$.*

Theorem 1. *$\simeq_b$ is an equivalence relation.*

This theorem can be proved in a similar fashion as for branching computed network bisimilarity in [9]. As said, branching network bisimilarity and the equivalence relation induced by CACTL coincide. This can be proved for CLTSs with so-called bounded-nondeterminism following the approach of [4]. The result can be lifted to general CLTSs in the same vein as [19], by resorting to infinitary logics (see [11] for the proof).

Theorem 2. *Let $\langle S, \Lambda, \rightarrow, s_0 \rangle$ be a CLTS. For any $r, s \in S$, $r \simeq_b s$ iff $r \sim_L s$.*

7 Conclusion and Future Work

We introduced the branching-time temporal logic CACTL, interpreted over CLTSs, to reason about topology-dependent behavior of MANET protocols. We can investigate scenarios like *after a route found* and *after two disconnected components merged* with the help of multi-hop constraints over topologies, which are specified as a part of path operators in our logic. Advantages of our approach are flexibility in verifying topology-dependent behavior (without changing the model), and restricting the generality of mobility. By nesting until operators, a specific path can be found with the help of topology constraints (without a need to specify how a topology constraint should be inferred), and then fixed for further exploration. The (dis)connectivity information in CLTS transitions makes it possible to restrict the generality of mobility as desired. By contrast, in approaches like [7], the inferences leading to the establishment of topology constraints should be embedded in the specification. Existing approaches to model mobility either are insusceptible to model checking [13, 23, 7], or require separate modeling of mobility [8]. The logic in [21] does not support verification of topology-dependent behavior.

A model checker for CACTL is being implemented, using the rewrite logic Maude. We also intend to verify real-world MANET protocols.

References

1. K. Bhargavan, D. Obradovid, and C. A. Gunter. Formal verification of standards for distance vector routing protocols. *Journal of the ACM*, 49(4):538–576, 2002.
2. E. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, LNCS 131, pp. 52–71. Springer, 1981.
3. R. De Nicola and F. Vaandrager. Action versus state based logics for transition systems. In *Semantics of Systems of Concurrent Processes*, LNCS 469, pp. 407–419. Springer, 1990.
4. R. De Nicola and F. Vaandrager. Three logics for branching bisimulation. *Journal of the ACM*, 42(2):458–487, 1995.
5. R. De Renesse and A.H. Aghvami. Formal verification of ad-hoc routing protocols using SPIN model checker. In *MELECON*, pages 1177–1182. IEEE, 2004.
6. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 2001.
7. A. Fehnker, R. van Glabbeek, P. Höfner, A. McIver, M. Portmann, and W.L. Tan. A process algebra for wireless mesh networks. In *ESOP*, LNCS 7211, pp. 295–315. Springer, 2012.
8. A. Fehnker, R. van Glabbeek, P. Höfner, A. McIver, M. Portmann, and W.L. Tan. Automated analysis of AODV using Uppaal. In *TACAS*, LNCS 7214, pp. 173–187. Springer, 2012.

9. F. Ghassemi, W. Fokkink, and A. Movaghar. Equational reasoning on mobile ad hoc networks. *Fundamenta Informaticae*, 103:1–41, 2010.
10. F. Ghassemi, W. Fokkink, and A. Movaghar. Verification of mobile ad hoc networks: An algebraic approach. *Theoretical Computer Science*, 412(28):3262–3282, 2011.
11. F. Ghassemi, S. Ahmadi, W. Fokkink, and A. Movaghar. Model Checking MANETs with Arbitrary Mobility. In pre-proceedings of *FSEN'13* , pp. 207–223, 2013.
12. J.C. Godskesen. Observables for mobile and wireless broadcasting systems. In *COORDINATION*, LNCS 6116, pp. 1–15. Springer, 2010.
13. J.C. Godskesen. A calculus for mobile ad hoc networks. In *COORDINATION*, LNCS 4467, pp. 132–150. Springer, 2007.
14. D. Kouzapas and A. Philippou. A process calculus for dynamic networks. In *FORTE*, LNCS 6722, pp. 213–227. Springer, 2011.
15. A. McIver and A. Fehnker. Formal Techniques for Analysis of Wireless Network. In *ISoLA*, LNCS 6722, pp. 263–270. IEEE, 2006.
16. R. Meolic, T. Kapus, and Z. Brezocnik. ACTLW - An action-based computation tree logic with unless operator. *Information Sciences*, 178(6):1542–1557, 2008.
17. M. Merro. An observational theory for mobile ad hoc networks. In *MFPS XXIII*, ENTCS 173, pp. 275–293. Elsevier, 2007.
18. N. Mezzetti and D. Sangiorgi. Towards a calculus for wireless systems. In *MFPS XXII*, ENTCS 158, pp. 331–353. Elsevier, 2006.
19. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
20. S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theoretical Computer Science*, 367(1):203–227, 2006.
21. S. Nanz, F. Nielson, and H. Nielson. Static analysis of topology-dependent broadcast networks. *Information and Computation*, 208(2):117–139, 2010.
22. C.E. Perkins and E.M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *WMCSA*, pp. 90–100. IEEE, 1999.
23. A. Singh, C.R. Ramakrishnan, and S.A. Smolka. A process calculus for mobile ad hoc networks. In *COORDINATION*, LNCS 5052, pp. 296–314. Springer, 2008.
24. R. van Glabbeek and W.P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM*, 43(3):555–600, 1996.
25. S. Vasudevan, J. Kurose, and D. Towsley. Design and analysis of a leader election algorithm for mobile ad hoc networks. In *ICNP*, pages 350–360. IEEE Computer Society, 2004.
26. O. Wibling, J. Parrow, and A. Pears. Automatized verification of ad hoc routing protocols. In *FORTE*, LNCS 3235, pp. 343–358. Springer, 2004.
27. O. Wibling, J. Parrow, and A. Pears. Ad hoc routing protocol verification through broadcast abstraction. In *FORTE*, LNCS 3731, pp. 128–142. Springer, 2005.

A Pseudo Code of Procedure *CheckEU*

Procedure $CheckEU\ (\phi_1, \chi_1, \phi_2, \chi_2, \mu, \zeta)$
$T' := \{s \mid \langle \phi_2, -, - \rangle \in label(s)\}$; `//--the first case--`
$T := \emptyset$ and let $\varphi \equiv \mathbf{E}(\phi_1 \ {}_{\{\chi_1\}}\mathbf{U}\ {}^{\mu}{}_{\{\chi_2\}}\phi_2)$;
forall the t *such that* $(t, (\mathcal{C}, \eta), s) \in \rightarrow$ *and* $\eta \models \chi_2$ *and* $\zeta \subseteq \zeta \oplus \mathcal{C}$ **do**
  forall the $\langle \phi_1, \Omega_1, O_1 \rangle \in label(t)$ **do**
    forall the $\langle \phi_2, \Omega_2, O_2 \rangle \in label(s)$ *such that* $\mathcal{C} \subseteq \mathcal{C} \oplus \Omega_2$ **do**

$label(t) := label(t) \cup \{\langle \varphi, \mathcal{C} \cup \Omega_1, \Omega_2, \mathcal{C}, O_1, \{\langle \mu, \mathcal{C}\rangle\} \cup \mathcal{C} \oplus O_2\rangle\}$;
$T := T \cup \{t\}$;
while $T \neq \emptyset$ **do**
choose $s \in T$ and $T := T \setminus \{s\}$;
forall the $(t, (\mathcal{C}, \eta), s) \in \rightarrow$ *and* $((\eta \models \chi_1 \wedge \zeta \subseteq \zeta \oplus \mathcal{C}) \vee (\mathcal{C} = \{\} \wedge \eta = ?))$ **do**
forall the $\langle \phi_1, \Omega, O\rangle \in label(t)$ **do**
forall the $\langle \varphi, \Omega_1, \Omega_2, \mathcal{C}', O_1, O_2\rangle \in label(s)$ *s.t.* $\mathcal{C} \oplus \mathcal{C}' \subseteq \mathcal{C} \oplus \mathcal{C}' \oplus \Omega_2$ **do**
$newLabel = label(t) \cup \{\langle \varphi, \mathcal{C} \cup \Omega \cup \Omega_1, \Omega_2, \mathcal{C} \oplus \mathcal{C}', O \cup O_1, \mathcal{C} \oplus O_2\rangle\}$;
if $newLabel \setminus label(t) \neq \emptyset$ **then**
$label(t) := newLabel$;
$T := T \cup \{t\}$;
endif
endwhile
$T := \{s \mid \langle \varphi, -, -, -, -, -\rangle \in label(s)\}$;
forall the $s \in T$ **do**
forall the $\langle \varphi, \Omega_1, \Omega_2, \mathcal{C}, O_1, O_2\rangle \in label(s)$ **do**
$label(s) := label(s) \setminus \{\langle \varphi, \Omega_1, \Omega_2, \mathcal{C}, O_1, O_2\rangle\}$;
$label(s) := label(s) \cup \{\langle \varphi, \Omega_1 \cup \Omega_2, O_1 \cup O_2\rangle\}$;
$S' = \{s \mid \langle \phi_1, -, -\rangle \in label(s)\}$; //--the second case--
$\rightarrow' = \{(t, (\mathcal{C}, \eta), s) \mid (t, (\mathcal{C}, \eta), s) \in \rightarrow \wedge((\eta \models \chi_1 \wedge \zeta \subseteq \zeta \oplus \mathcal{C}) \vee (\mathcal{C} = \{\} \wedge \eta =$
$?)) \wedge s, t \in S'\}$;
$SCC := \{SC \mid SC$ is a non-trivial SCC of CLTS$\langle S', \Lambda, \rightarrow', s_0\rangle\}$;
$T' := \bigcup_{SC \in SCC} \{s \mid s \in SC\}$;
forall the $s \in T'$ *do* **do** /* initializing states on SCCs */
forall the $(t, (\mathcal{C}, \eta), s) \in \rightarrow'$ *and* $\eta \models \chi_1 \wedge \zeta \subseteq \zeta \oplus \mathcal{C}$ **do**
forall the $\langle \phi_1, \Omega_1, O_1\rangle \in label(t)$ **do**
forall the $\langle \phi_1, \Omega_2, O_2\rangle \in label(s)$ **do**
$label(t) := label(t) \cup \{\langle \varphi, \mathcal{C} \cup \Omega_1 \cup \Omega_2, \emptyset, \mathcal{C}, O_1 \cup O_2, \{\langle \neg\mu, \mathcal{C}\rangle\}\rangle\}$;
T := T';
while $T \neq \emptyset$ **do** /* finding the accumulated network constraints in each SCC and next moving out of SCCs */
choose $s \in T$;
$T := T \setminus \{s\}$;
forall the $t \in S'$ *such that* $(t, (\mathcal{C}, \eta), s) \in \rightarrow'$ **do**
forall the $\langle \phi_1, \Omega, O\rangle \in label(t)$ **do**
forall the $\langle \varphi, \Omega_1, \Omega_2, \mathcal{C}', O_1, O_2\rangle \in label(s)$ **do**
$newLabel = label(t) \cup \{\langle \varphi, \mathcal{C} \cup \Omega \cup \Omega_1, \Omega_2, \mathcal{C}\mathcal{C}', O \cup O_1, \mathcal{C} \oplus O_2\rangle\}$;
if $newLabel \setminus label(t)$ **then**
$label(t) := newLabel$;
$T := T \cup \{t\}$;
endif
endwhile
$T := \{s \mid \langle \varphi, -, -, -, -\rangle \in label(s)\}$;
forall the $s \in T$ **do**
forall the $\langle \varphi, \Omega_1, \Omega_2, \mathcal{C}, O_1, O_2\rangle \in label(s)$ **do**
$label(s) := label(s) \setminus \{\langle \varphi, \Omega_1, \Omega_2, \mathcal{C}, O_1, O_2\rangle\}$;
$label(s) := label(s) \cup \{\langle \varphi, \Omega_1 \cup \Omega_2, \mathcal{C}, O_1 \cup O_2\rangle\}$;