Temporal Property Testing in Dynamic Networks: Application to Software-Defined Vehicular Networks

Yessin M Neggaz, Soufian Toufga

Temporal Property Testing in Dynamic Networks: Application to Software-Defined Vehicular Networks

Yessin M. Neggaz
LAAS-CNRS, Université de Toulouse
INSA
Toulouse, France
myneggaz@laas.fr

Soufian Toufga
LAAS-CNRS, Université de Toulouse
CNRS
Toulouse, France
soufian.toufga@laas.fr

***Abstract*—With the recent advances in the area of wireless communication networks, a new kind of networks emerges where entities make contact over time with one another, which makes topology highly dynamic *e.g UAV* networks, wireless sensor networks, vehicular networks *etc.* A challenge in this context is to study dynamics patterns and temporal properties and decide if the evolution of the topology satisfies requirements for given protocol, algorithm, task *etc.* and adapt at different levels. In this paper we focus on testing temporal properties in dynamic networks, then we consider *SDVN – Software-Defined vehicular networks* as a case study where we propose the integration of a new component in SDVN architecture based on a framework that allows dynamic networks properties analysis and adaptation to a given dynamic context. We demonstrated how network control functions can take benefit from the provided information.**

***Index Terms*—Dynamic networks, Software-Defined Vehicular Networks, SDN.**

I Introduction

One of the major evolutions in the area of computer science, we note the emergence of dynamic networks. These networks consist of entities making contact over time with one another, which makes them different from static networks where the topology remains unchanged. Recently, the community has explored contexts where the dynamic is considered as a property of the network, rather then exception and several works have highlighted the importance of studying and defining mobility patterns and characterizing dynamic properties in a dynamic context. In a static context, the stability allows one to have all parameters to preview the execution on a given network. A major challenge in dynamic networks is difficulties to detect mobility patterns and decide whether the evolution of the topology satisfies requirements. From this point of view we distinguish two types of dynamic networks: Controlled dynamic networks where contacts and topological changes can be directed in a way such that they adapt to the execution of an algorithm; non-controlled networks where the evolution of the network topology is completely or partially independent from the execution and unpredictable without any analysis. The latter category represents a large part of the practical contexts, as vehicular networks, telephone networks, social networks *etc.* where the movements of communicating and computing units depend on the movements of the underlying mobile objects. The types of network dynamics are different, some of them are connected all the time [16], others are not but give a temporal connectivity (connectivity over time) [13], others are stable, periodic, *etc.* All of these contexts can be represented as temporal properties and dynamic graph classes [5]. In [4] authors propose an analytical approach that aims to characterize how appropriate a given algorithm is to a given dynamic context and define based on this relevant temporal properties. Given a dynamic network, an interesting question to ask is what temporal property it has and how it evolves over time. Thus, being able to learn about the evolution of the network state is useful for determining which problem can be successfully solved. Furthermore, it can be useful to adapt at different levels or to control the network to satisfy a needed property. Different strategies were proposed in the literature for testing temporal properties in dynamic networks. In [6] authors present a generic framework for computing maximum and minimum parameters, testing properties and show its application by solving some relevant problems. In this work we are interested in testing temporal properties and computing parameters in an emerging issue in the field of dynamic communication networks: *SDVN - Software-Defined Vehicular Networks*. We present an adaptation of the existing frameworks to a more general case and show its application by proposing a new component in the SDVN architecture allowing network control functions to learn about the network topology dynamics and adapt to its evolution.

The paper is organized as follows: In section *II* we present dynamic network model and some basic definitions on dynamic graphs. Section *III* discusses the characterization of temporal properties and their impact on algorithms and application suitability. In Section *IV* we present some existing strategies for testing properties and computing parameters, then a framework that we will apply in Section *V* to SDVN context where we highlight the benefit of our proposition.

II Model

In this section we present the dynamic networks model and some basic notions and concepts that will be used in this paper.

II-A Dynamic Graph Model

In a dynamic context, networks can be represented in many ways, it depends on the definition of the temporal domain in which the system evolves. Many models exist and are widely used in the literature, like *Time-varying graphs* [5], *temporal networks* [14] and *link streams* [19]. It is often suitable to represent the evolution of the network as a sequence of global

graphs representing the general state of the network during a time interval. In what follows, we will use *evolving graphs* [9] where the evolution of the graph is modelled by an indexed sequence of graphs $\{G_i = (V, E_i)\}$, each graph G_i being associated to a time interval $[t_i, t_{i+1})$ during which it is available. Figure 1 shows an example. Formally:

Definition 1 (Evolving Graph)**.** *Let $G = (V, E)$ be a graph, an* evolving graph *$\mathcal{G}$ is represented by a triplet $\mathcal{G} = (G, \mathcal{S}_G, \mathcal{S}_\mathbb{T})$ where $\mathcal{S}_G$ is an indexed sequence $\{G_i = (V, E_i)\}$ of G subgraphs and $\mathcal{S}_\mathbb{T} \subseteq \mathbb{T} \subseteq \mathbb{N}$ or $\mathbb{R}$ is the associated sequence of dates $t_1, t_2, ...$ where $\forall e \in E_i$, e is present in the time interval $[t_i, t_{i+1})$ (only the graph G_i is present in this interval).*

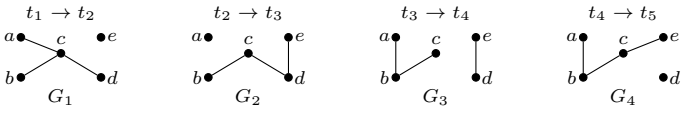

Fig. 1. Example representation of a dynamic network as an evolving graph.

In a more general case edges could be directed to represent non-symmetrical links. The state of the links and nodes could be presented by labels on the edges *e.g* QoS, capacity, load *etc.*

II-B Temporal Properties

In this work, we are interested in temporal and topological properties, in other words, characterizing and testing properties on the evolution of the topology but not only on the state of the network at a time t. For instance, an important property in dynamic networks is the connectivity between nodes. At no time, in Figure 1 in the graph $\mathcal{G}$ a is connected to e (in a classical sense). But by allowing the combination of edges from different graphs, e can be reached from a by means of a temporal path, e.g: $((a, c) \in E_1, (c, e) \in E_4)$. This temporal aspect of a path gives the notion of *journey* (temporal path).

Other properties can be defined, for example on the stability of a (same) path $e.g$ $((b, c), (c, d))$ in $\{G_i : 1 \geq i \geq 2\}$, the existence of a path between tow nodes during a given duration, the recurrence of a link (periodic, time-bounded, recurrent ...), properties on a set of links, on a part of the network *etc.* A natural question is "what gives sense to a given property and makes it interesting to consider?". An efficient approach is to analyze algorithms, protocols, tasks ... and to look at the assumptions made on the dynamic graph.

III Characterizing temporal properties

In [4] authors propose a framework that allows to examine what impact a property has on the execution of an algorithm and that aims to characterize how appropriate a given algorithm is to a given dynamic context. The workflow is presented in Figure 2. Algorithms are analyzed to characterize conditions that define temporal properties.

Their general methodology consists in considering a problem, then characterizing the conditions for the success of its algorithm execution in terms of network dynamics. They model this in two predicates: *objective* and *condition*. The

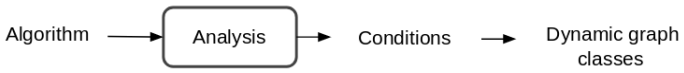

Fig. 2. Characterizing temporal properties [4].

objective $\mathcal{O}_\mathcal{A}$ defines the success of the execution of an algorithm $\mathcal{A}$. For example, for a propagation algorithm $\mathcal{A}$, which consists in transmitting an information I from a node u to all the other nodes of the graph, the objective $\mathcal{O}_\mathcal{A}$ is $\forall v \in V, v$ has the information I. *Property: There exists a journey from a node u to all other nodes of the graph,* is proved to be a necessary condition for $\mathcal{A}$.

A hierarchy of general classes of dynamic graphs, corresponding to temporal properties, was defined based on this process [4] *e.g Temporal Connectivity* (for all pairs of nodes $u, v \in V$, there exist a journey), *Constant Connectivity* ($\forall G_i \in \mathcal{G}, G_i$ is connected), *T-interval Connectivity* (every sequence of length T in the dynamic graph $\mathcal{G}$ shares a common connected spanning subgraph), *Time-bounded Reappearance of Edges*, *Periodic Reappearance of Edges etc.* The defined classes are generic and define properties on the entire graph. One could be interested in properties on a specific set of nodes in the graph or a specific path, link *etc.* Specific properties can be defined based on what exists or by considering new network problems and algorithms. This method can be generalized (Figure 3) to allow the characterization of specific temporal and topological requirements based on a practical evaluation of protocols, tasks, algorithms beside the existing analytical approach.

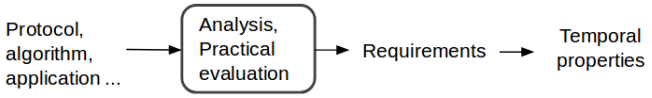

Fig. 3. Characterizing requirements.

In an online setting, where only a recent history of the network state is considered, it is more interesting to study the temporal property evolution instead of defining a property on the whole graph $i.e$ temporal properties on windows of the network. For example, it gives more information to compute the *Temporal diameter*: the smallest duration in which every node can join all other nodes using a journey at any instant t, than merely test whether the dynamic network is *temporally connected*. This information about the evolution of the property is called *Temporal parameter*:

Definition 2. *p_i is a parameter of $\mathcal{G}$ with respect to a given property P, at time i, iff $\{G_i, G_{i+1}, ..., G_{i+p_i-1}\}$ verifies the temporal property P, that is, starting from time i, the property P is true for a duration p_i.*

Let's take for example *Temporal connectivity*, if every node can reach all the other nodes, using a journey, at time i in a duration of 5 but not 4 then $p_i > 4$ is a parameter and 4 defines the *temporal diameter* of the graph at time i $i.e$ minimization of the parameter p_i. Different temporal parameters can be defined based on existing properties, and related information can be extracted. For instance, the smallest duration in which

a set of links/paths reappear at least once, the largest duration in which a path is stable *etc.* Testing temporal properties and computing parameters allows one to collect information about the network based on which could decide which task, protocol or algorithm could be used and when, or adapt its behavior.

IV Testing Temporal Properties and Computing Parameters

Dynamic graphs instances can be obtained from network traces generated from the collection of real-world networks or network topology estimation. Then testing temporal properties and computing parameters can give an indication about the suitability of a given algorithm in a mobility context. This allows one to adapt the execution of the algorithm, the behavior of a protocol or an architecture at a given level to the network dynamics, and conversely control the network topology evolution to guarantee needed properties. The workflow presented in Figure 4 shows a general representation for an automatic testing framework that produces information about the dynamics of the network with respect to a given temporal property.

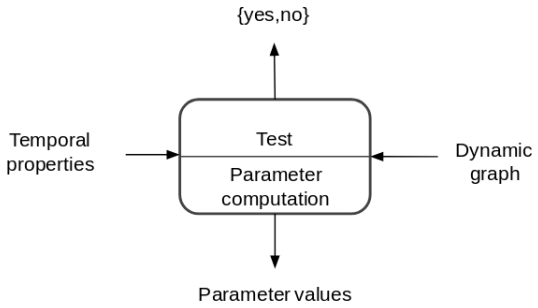

Fig. 4. Automatic classification of dynamic networks and parameter computation.

A key issue is that of understanding how far such a framework could be automated. Several works contributed to the automation of the core operations of *Property testing* and *Parameter computation*, by proposing strategies for the classification of dynamic graphs for specific classes like *Temporal Connectivity* [3] and *T-interval connectivity* [7].

In [6] authors present a generic framework for computing maximum and minimum parameters and testing properties in dynamic networks and show its application by solving minimization and maximization problems like computing the temporal diameter, the round trip temporal diameter of a given dynamic graph, the reappearance-bound of its edges and computing T for which a given $\mathcal{G}$ is T-interval connected.

The proposed algorithm computes a sequence of high-level structures (graphs) $\{G_{(i,j)}\}$ with $j \geq i$. Each structure $G_{(i,j)}$ corresponds to a sub-sequence $\{G_i, G_i + 1, ..., G_j\}$ of the original dynamic graph and represents information about the considered temporal property *e.g transitive closures of journeys* $\{G_{(1,3)}, G_{(2,4)}, ..., G_{(6,8)}\}$ in Figure 6. From this sequence, by testing the presence of an edge (u, v) in $G_{(i,i+2)}$, one can directly test if a node v can be reached from a node u using a journey, in a duration of 3 starting at time i. This high-level sequence is computed using a *composition operation* defined by the user for the considered property or parameter *e.g transitive closure concatenation* as operation to compute the sequence of *transitive closure of journeys*.

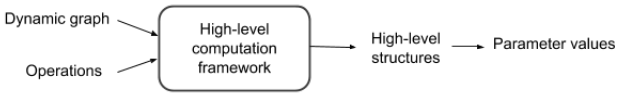

Fig. 5. Simplified presentation of the framework.

So far, the generic framework is designed to solve minimization and maximization problems (*i.e* finding the smallest/largest duration for which a given temporal property is verified at any time). The framework can be directly extended to compute the needed high-level structures and extract the desired information about the network based on what information the user needs at a given time or depending on external events. Figure 5 shows a general presentation of this high-level computation framework. Depending on a specific temporal property, a composition operation is used to compute the high-level sequence from the dynamic graph given as input (see [6] for more details about the composition operation). Then temporal parameters (w.r.t the considered property) can be computed directly from the resulting high-level structures. This hides the complexity of the network dynamics and allows one to focus only on the extraction of interesting information about the evolution of the temporal properties of the network and then adapt to changes at this level. For instance, if the application tolerates a maximum latency l using a store and forward adhoc communication, then the system/protocol must adapt if the temporal diameter (the considered temporal parameter in this case) exceeds l, for example, by using an other communication mode (*e.g* infrastructure). If the system can be completely or partially controlled, then the topology can be directed in a way to guaranty the needed requirement. More complex computation can be done using the framework to extract more precise information on the network.

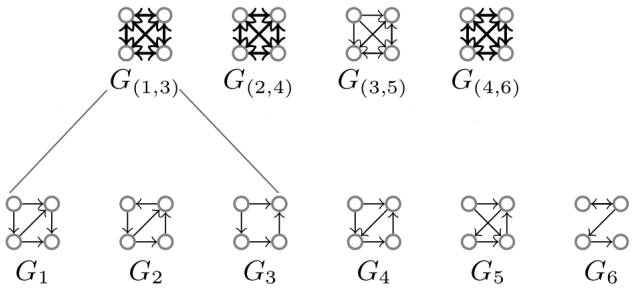

Fig. 6. Example of high-level structures (transitive closure of journeys) to test temporal connectivity.

V Case Study

Intelligent Transport System (ITS) brings the idea of providing the vehicles and the transport infrastructure with communication capabilities, in an effort to improve their safety, reliability, efficiency and quality [1]. To that end, a large variety of companies including car makers along with automotive industry suppliers, telecom operators, and research

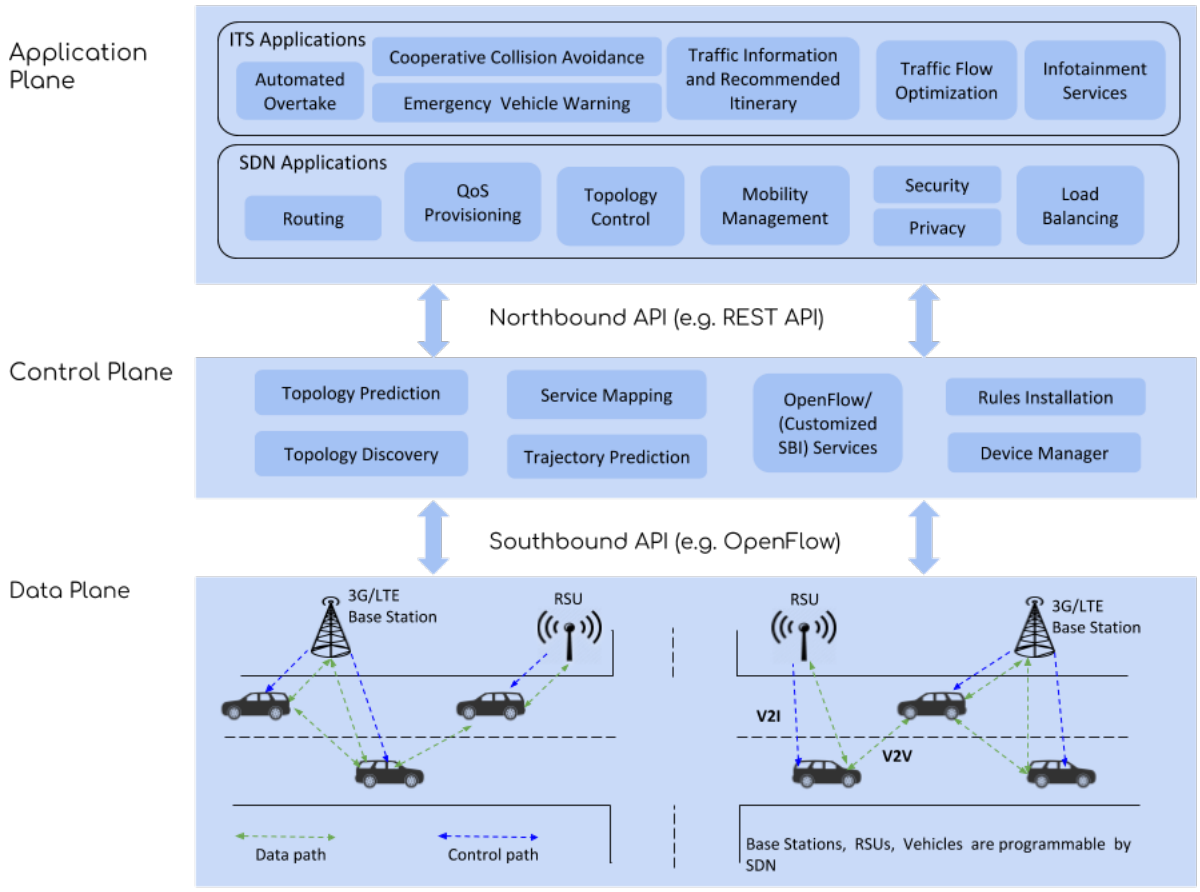

Fig. 7. Software-Defined Vehicular Networks Architecture.

bodies are gathered to propose different ITS services, such as, Cooperative Collision Avoidance, Automated Overtake, Traffic information and recommended itinerary, and Vulnerable Road User Discovery [2]. Therefore, the next generation of vehicles will be equipped with one or more interfaces enabling it to communicate with the surrounding vehicles, as well as, the infrastructure, and organized in networks, known as *Vehicular networks*. In the last years, vehicular networks are attracting more attention, recent research activities [10], [17], [15] are investigating the possibility of applying Software Defined Network (SDN) paradigm to vehicular networks to bring the flexibility and programmability to such dynamic network, with the aim of efficiently supporting the emerging ITS services. The SDN paradigm advocates the idea of taking control plane functions out of network forwarding devices and relocating them on remote external computing machines called SDN controllers. By doing so, the network intelligence becomes logically centralized and the network nodes only forward packets according to the rules installed by the SDN controller [2].

V-A SDVN – software-defined Vehicular networks

Applying SDN to vehicular networks consists on the separation of data plane and control plane. The main objective of data plane is the forwarding of data , while the control decisions are taken by the control plane, for example, to decide from which interface each packet will be forwarded. As shown on Figure 7 and described on [11], the data plane is composed of various Forwarding Elements as Base Stations (3G/4G), Road Side Units (RSU), Vehicles, all under the control of SDN controller. Each node sends periodically to the SDN controller information about its status (connectivity, position, speed, direction ...) through the Southbound interface. These information are used in the control plan to build a global view of the network, which is exposed via the Northbound interface to the network control applications (routing, mobility management, ...) in order to generate the control decisions to be implemented by the data plane, In addition, the Business Applications (ITS services) can specify their requirements and the expected behavior of the network through the northbound interface.

The network control functions constitute the intelligence of the network, these functions benefit from a personalized and simplified representation of the underlying network, this representation is constructed and maintained by the topology discovery service using the information sent by the forwarding elements. for example a routing function needs the graph of the topology of the network and the capacities of the various links in order to compute the routing path. However, with the high mobility of vehicles, keeping this view up to date represents a challenging task, the authors in [11] propose the idea of using a topology prediction service based on the potential trajectory of

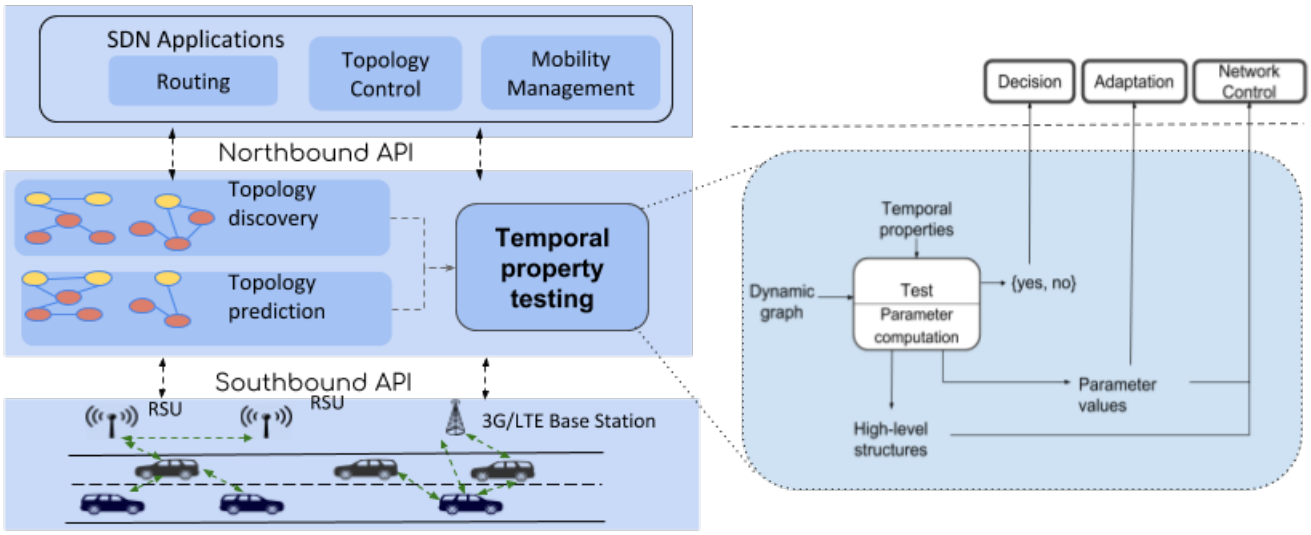

Fig. 8. Temporal property testing in SDVN.

a vehicle in order to expose an estimated view of the network.

V-B *Temporal property testing in SDVN*

In SDVN architecture, the SDN controller, which represents the control plane, constitutes the intelligence of the network, it executes various network control functions (known as SDN applications) allowing the control and management of the network in a centralized way. These network functions benefit from the information provided by the basic services of SDN, such as topology discovery service. In order to enrich the information provided to the network control functions, we propose to study the evolution of the graph topology using the framework presented in the previous sections. Multiple temporal properties and parameters can be computed using a historical graph sequence (provided by the topology discovery service), or an estimated one (provided by the topology prediction service).

Figure 8 shows the integration of the presented framework in the SDVN architecture. The defined component (*Temporal property testing*) operates in the control plane in interaction with the topology discovery and prediction services, which provide the needed information about the network topology evolution as a dynamic graph. Outputs are computed based on the desired property specified by the SDN application and then used directly to (1) make decision about the control policies (*e.g* switch from an adhoc strategy to a centralized one if the required property is not verified), (2) adapt the network control functions behavior to the network, and (3) control the network to satisfy the desired requirements.

One of the most important network control functions in such architecture is *Routing*. The main objective of this service consists in computing optimal routes to send information from a source node to one or many destination nodes. In general, the routing protocols use several metrics like hop numbers (shortest path), link quality (reliable, fastest path) *etc.* In a dynamic environment, communication link disconnections occur frequently. So, routing based only on classical metrics may not be efficient in such a context. Testing temporal properties and computing parameters in dynamic networks could give relevant information which can be considered in the optimal path computation. An important temporal parameter is the stability of paths over time [18]. By giving the right operation, the routing function can use the output of the introduced component (*Temporal property testing*) to test the stability of paths in a horizon of h (defined by the application) from the sequence given by the *Topology prediction* service *i.e* starting from a time t test if a path is shared by a sequence $\{G_i, G_{i+1}, ..., G_{i+d-1}\}$ with $d \leq h$.

To compute this parameter one should be able to directly test the stability of a path from the corresponding high-level structure. An option could be to compute $G_{(i,i+h-1)}$ containing all edges that appear in G_i labeled with their stability: every edge e in $G_{(i,i+h-1)}$ is labeled with the largest $k \leq h$ such that $e \in \cap\{G_i, G_{i+1}, ..., G_{i+k-1}\}$. This can be computed thanks to the *Union* composition operation that computes $\cup\{G_i, G_{i+1}, ..., G_{i+h-1}\}$ with all appearance dates of edges and then keeps on $G_{(i,i+h-1)}$ only edges belonging to G_i with labels corresponding to *first_disappearance_date*$-i$ (see [6] for more details about composition operations in this framework). To test if a path is stable in the network for a duration d starting from a time i, it is sufficient to test the existence of the path in $G_{(i,i+h-1)}$ with labels $\geq d$ on all the edges. This can be done in a linear time in the number of edges. Figure 9 shows an example.

To have more information about the presence of links, all appearance dates could be kept on edges in $\{G_{(i,i+h-1)}\}$. So, one can determine when missing links are needed to be established to ensure the stability of a specific path, then, request the *Topology control* function with this precise information in order to satisfy the general property (path stability) when possible. In this case (SDVN), thanks to the programmability offered by SDN, the topology and links can be controlled by adjusting the range of communication devices for example.

Other important SDN applications could benefit from the

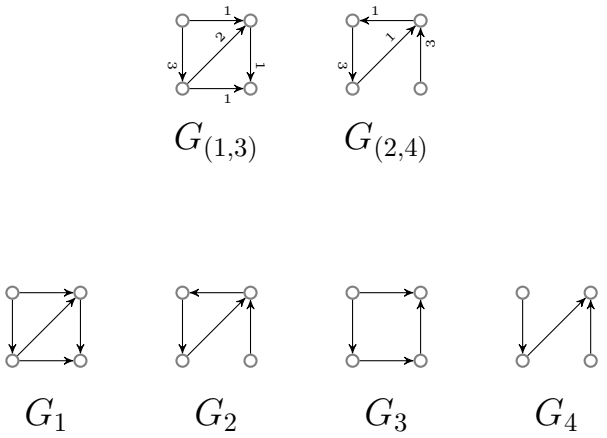

Fig. 9. Example of high-level structure for testing path stability with $h = 3$.

Temporal property testing module in the same manner. Table I exposes some network control functions and related properties. Besides *Path stability*, *Routing control function* can take advantage from other properties like *Temporal connectivity* (presented in section II-B) and *Link recurrence* (periodic and time-bounded links reappearance), which enable different routing strategies (*e.g.* store-and-forward, foremost broadcast *etc.*). It also can be interesting for *Load balancing control function* to use information about the connectivity between a set of vehicles on one hand, and their connectivity with the infrastructure (*e.g.* Fog data centers), on the other hand: For instance, depending on the nature of transmitted data, a vehicle can use a distant data center (instead of directly connected one) using a *stable path* (*e.g.* for a continuous traffic, *Instant connectivity* otherwise) or a *Journey* (*Temporal connectivity*) if transmitted data tolerate a certain latency.

Temporal Properties	Network Control Function			
	Routing	Mobility Mgmt	Load balancing	Topology control
Path stability	√	√	√	√
Instant connectivity	√		√	√
Temporal connectivity	√		√	√
Link recurrence	√			√

TABLE I
EXAMPLE OF TEMPORAL PROPERTIES FOR NETWORK CONTROL FUNCTIONS.

Furthermore, an other utilization of the computed output could be the analysis of the computed parameters flow and its variation using analytical tools for the extraction of correlation relations with other parameters or other information on the network (*e.g.* using Pearson's correlation coefficient [8]), in order to find relevant information and meaning. We can note that it is easier to predict high-level structures or parameter values based on the output computed before a given instant than predict the topology of the dynamic network itself. This could lead to predictive strategies using for example Long Short-Term Memory (LSTM) Based Recurrent Neural Network Architectures [12] for sequence prediction.

VI Conclusion

In this paper, we presented an application of a general framework for testing temporal properties in *Software-defined Vehicular Networks*. We integrated a new component in the existing architecture allowing the network control functions to take benefit from the computed network evolution information, in order to efficiently control the network in such a highly dynamic context.

As a perspective, one direction could be to define new requirements and properties for other fundamental network control functions. Also, it could be interesting to extend the application of the framework to consider more properties. Finally, this work could profit from an experimental evaluation which would show its applicability.

References

[1] Its definition, etsi. www.etsi.org/images/files/ETSITechnologyLeaflets/IntelligentTransportSystems.pdf. Accessed: 2018-03-10.

[2] 5g automotive vision. Technical report, 5G-PPP, The 5G Infrastructure Public Private Partnership, White paper, 2015.

[3] M. Barjon, A. Casteigts, S. Chaumette, C. Johnen, and Y. M. Neggaz. Testing temporal connectivity in sparse dynamic graphs. In *2nd Int. Conference on Research challenges for future RPAS/UAV systems*, France, 2014.

[4] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *Proc. of SIROCCO*, pages 126–140, Piran, Slovenia, 2009. Springer. (Full version in *CoRR, abs/1102.5529*).

[5] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[6] A. Casteigts, R. Klasing, Y. Neggaz, and J. G. Peters. A generic framework for computing parameters of sequence-based dynamic graphs. In *24th International Colloquium on Structural Information and Communication Complexity SIROCCO*, 2017.

[7] A. Casteigts, R. Klasing, M. Y. Neggaz, and J. G. Peters. Efficiently testing t-interval connectivity in dynamic graphs. In . *Conference on Algorithms and Complexity CIAC*, Lecture Notes in Computer Science, Paris, France, 2015.

[8] B. E. Cooper. Correlation and function fitting. *Statistics for Experimentalists*, pages 753–758, May 2014.

[9] A. Ferreira. On models and algorithms for dynamic communication networks: The case for evolving graphs. In *Proc. of 4e rencontres francophones sur les Aspects Algorithmiques des Télécommunications (ALGOTEL)*, 2002.

[10] X. Ge, Z. Li, and S. Li. 5g software defined vehicular networks. *IEEE Communications Magazine*, 55(7):87–93, 2017.

[11] Z. He, J. Cao, and X. Liu. Sdvn: enabling rapid network innovation for heterogeneous vehicular communication. *IEEE Network*, 30(4):10–15, July 2016.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. of SIGCOMM*, pages 145–158, 2004.

[14] D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, STOC '00, pages 504–513, New York, NY, USA, 2000. ACM.

[15] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira. Towards software-defined vanet: Architecture and services. In *2014 13th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, pages 103–110, June 2014.

[16] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proc. of DIALM-POMC*, pages 104–110, Cologne, Germany, 2005. ACM.

[17] S. TOUFGA, P. Owezarski, S. Abdellatif, and T. Villemur. An SDN hybrid architecture for vehicular networks: Application to Intelligent Transport System. In *9th European Congress on Embedded Real Time Software And Systems (ERTS)*, page 8p., Toulouse, France, Jan. 2018.

[18] D. K. N. Venkatramana, S. B. Srikantaiah, and J. Moodabidri. Scgrp: Sdn-enabled connectivity-aware geographical routing protocol of vanets for urban environment. *IET Networks*, 6(5):102–111, 2017.

[19] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609 Part 1:245–252, Jan. 2016.