HAL
open science

Brief Announcement: Waiting in Dynamic Networks

Arnaud Casteigts, Paola Flocchini, Emmanuel Godard, Nicola Santoro, Masafumi Yamashita

To cite this version:

Arnaud Casteigts, Paola Flocchini, Emmanuel Godard, Nicola Santoro, Masafumi Yamashita. Brief Announcement: Waiting in Dynamic Networks. 31st ACM Symposium on Principles of Distributed Computing (PODC), Jul 2013, Portugal. pp.99-100. hal-00854266

HAL Id: hal-00854266

https://hal.science/hal-00854266v1

Submitted on 26 Aug 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Brief Announcement: Waiting in Dynamic Networks*

Arnaud Casteigts
University of Ottawa
casteig@eecs.uottawa.ca

Paola Flocchini
University ot Ottawa
flocchin@eecs.uottawa.ca

Emmanuel Godard
Université Aix-Marseille
egodard@cmi.univ-mrs.fr

Nicola Santoro
Carleton University
santoro@scs.carleton.ca

Masafumi Yamashita
Kyushu University
mak@csce.kyushu-u.ac.jp

We consider infrastructure-less highly dynamic networks, where connectivity does not necessarily hold, and the network may actually be disconnected at every time instant. These networks are naturally modeled as *time-varying* graphs. Clearly the task of designing protocols for these networks is less difficult if the environment allows *waiting* (i.e., it provides the nodes with store-carry-forward-like mechanisms such as local buffering) than if waiting is not feasible. We provide a quantitative corroboration of this fact in terms of the *expressivity* of the corresponding time-varying graph; that is in terms of the language generated by the feasible journeys in the graph. We prove that the set of languages $\mathcal{L}_{nowait}$ when no waiting is allowed contains all computable languages. On the other end, we prove that $\mathcal{L}_{wait}$ is just the family of *regular* languages. This gap is a measure of the computational power of waiting. We also study *bounded waiting*; that is when waiting is allowed at a node only for at most d time units. We prove the negative result that $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$.

Keywords

Time-varying graphs, dynamic networks, buffering, expressivity of TVGs

Categories and Subject Descriptors

F.1.1 [**Models of Computation**]: Networks of machines; F.4.3 [**Formal Languages**]: Classes defined by automata; D.4.4 [**Communications Management**]: Buffering

1. INTRODUCTION

In infrastructure-less highly dynamic networks, computing and performing even basic tasks (such as routing and broadcasting) is a very challenging activity due to the fact that connectivity does not necessarily hold, and the network may actually be disconnected at every time instant. Such highly dynamic systems do exist, are actually widespread, and becoming more ubiquitous; the most obvious class is that of wireless ad hoc mobile networks.

From a formal point of view, the highly dynamic features of these networks and their temporal nature are captured by the model of *time-varying* (or *evolving*) graphs, where edges between nodes exist only at some times, a priori unknown to the algorithm designer. Formally [1], a *time-varying graph* $\mathcal{G}$ is a quintuple $\mathcal{G} = (V, E, \mathcal{T}, \rho, \zeta)$, where V is a finite set of entities or *nodes*; $E \subseteq V \times V \times \Sigma$ is a finite set of relations between these entities (*edges*), possibly labeled by symbols in an alphabet Σ. The system is studied over a given time span $\mathcal{T} \subseteq \mathbb{T}$ called *lifetime*, where $\mathbb{T}$ is the temporal domain (typically, $\mathbb{N}$ or $\mathbb{R}^+$ for discrete and continuous-time systems, respectively); $\rho : E \times \mathcal{T} \to \{0, 1\}$ is the *presence* function, which indicates whether a given edge is available at a given time; $\zeta : E \times \mathcal{T} \to \mathbb{T}$, is the *latency* function, which indicates the time it takes to cross a given edge if starting at a given date (the latency of an edge could vary in time).

A crucial aspect of dynamic networks is that a path from a node to another might still exist over time, even though at no time the path exists in its entirety. It is this fact that renders routing, broadcasting, and thus computing possible in spite of the otherwise unsurmountable difficulties imposed by the nature of those networks. Hence, the notion of "path over time", called *journey*, is a fundamental concepts and plays a central role in the definition of almost all concepts related to connectivity in time-varying graphs. Informally a journey is a walk $<e_1, e_2, ..., e_k>$ and a sequence of time instants $<t_1, t_2, ..., t_k>$ where edge e_i exists at time t_i and the latency $\zeta(e_i, t_i)$ of this edge at that time is such that $t_{i+1} \geq t_i + \zeta(e_i, t_i)$.

While the concept of journey captures the notion of "path over time" so crucial in dynamical systems, it does not yet capture additional limitations that the environment can impose during a computation. More specifically, there are systems that provide the entities with store-carry-forward-like mechanisms (e.g., local buffering); thus an entity wanting to communicate with a specific other entity at time t_0, can *wait* until the opportunity of communication presents itself. There are however environments where such a provision is not available and thus waiting is not allowed. In time-varying graphs, this distinction is the one between a *direct* journey (where $\forall i, t_{i+1} = t_i + \zeta(e_i, t_i)$), and an *indirect* journey (where $\exists i, t_{i+1} > t_i + \zeta(e_i, t_i)$).

Clearly the task of designing protocols for these networks is less difficult if the environment allows waiting than if waiting is not feasible. No quantitative corroborations of this fact exist; in fact, up to now, there are no answers to these questions: *"if waiting is allowed, how much easier is to solve problems?"*, *"what is the computational power of waiting?"*

A first difficulty in addressing these important questions is that most of the terms are qualitative, and currently there are no measures that allow to quantify even the main concepts.

We consider these qualitative questions, and examine the complexity of the environment in terms of the *expressivity* of the corresponding time-varying graph; that is in terms of the language gen-

*The results announced in this paper can be found in: A. Casteigts, P. Flocchini, E. Godard, N. Santoro, and M. Yamashita. "Expressivity of time-varying graphs and the power of waiting in dynamic networks", *CoRR*, abs/1205.1975, 2012.

Copyright is held by the author/owner(s).
PODC'12, July 16–18, 2012, Madeira, Portugal.
ACM 978-1-4503-1450-3/12/07.

erated by the feasible journeys in the graph. We establish results showing the (surprisingly dramatic) difference that the possibility of waiting creates.

We find that the set of languages $\mathcal{L}_{nowait}$ when no waiting is allowed contains all computable languages. On the other end, using algebraic properties of quasi-orders, we prove that $\mathcal{L}_{wait}$ is just the family of *regular* languages. In other words, when waiting is no longer forbidden, the power of the accepting automaton (difficulty of the environment) drops drastically from being as powerful as a Turing machine, to becoming that of a Finite-State machine. This (perhaps surprisingly large) gap is a measure of the computational power of waiting. We also study *bounded waiting*; that is when waiting is allowed at a node only for at most d time units. We prove the negative result that $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$; that is, the expressivity decreases only if the waiting is finite but unpredictable (i.e., under the control of the protocol designer and not of the environment).

2. OVERVIEW OF THE RESULTS

Given a dynamic network modeled as a time-varying graph $\mathcal{G}$, a journey in $\mathcal{G}$ can be viewed as a word on the alphabet of the edge labels; in this light, the class of feasible journeys defines the language $L_f(\mathcal{G})$ expressed by $\mathcal{G}$, where $f \in \{wait, nowait\}$ indicates whether or not indirect journeys are considered feasible by the environment. Hence, a time-varying graph $\mathcal{G}$ whose edges are labeled over Σ, can be viewed as a TVG-automaton $\mathcal{A}(\mathcal{G}) = (\Sigma, S, I, \mathcal{E}, F)$ where Σ is the input *alphabet*; $S = V$ is the set of *states*; $I \subseteq S$ is the set of *initial states*; $F \subseteq S$ is the set of *accepting states*; and $\mathcal{E} \subseteq S \times \mathcal{T} \times \Sigma \times S \times \mathcal{T}$ is the set of *transitions* such that $(s, t, a, s', t') \in \mathcal{E}$ iff $\exists e = (s, s', a) \in E : \rho(e, t) = 1, \zeta(e, t) = t' - t$.

Figure 1 shows an example of a deterministic TVG-automaton $\mathcal{A}(\mathcal{G})$ that recognizes the context-free language $a^n b^n$ for $n \geq 1$ (using only direct journeys). The presence and latency of the edges of $\mathcal{G}$ are specified in Table 1, where p and q are two distinct prime numbers greater than 1, v_0 is the initial state, v_2 is the accepting state, and reading starts at time $t = 1$.

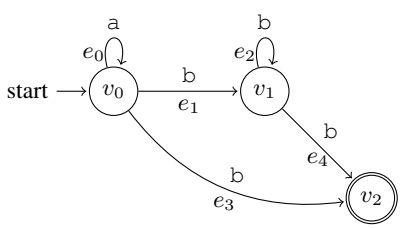

Figure 1: A TVG-automaton $\mathcal{A}(\mathcal{G})$ such that $L_{nowait}(\mathcal{G}) = \{a^n b^n : n \geq 1\}$.

e		Presence $\rho(e,t) = 1$ iff		Latency $\zeta(e,t) =$
e_0		always true		$(p-1)t$
e_1		$t > p$		$(q-1)t$
e_2		$t \neq p^i q^{i-1}, i > 1$		$(q-1)t$
e_3		$t = p$		any
e_4		$t = p^i q^{i-1}, i > 1$		any

Table 1: Presence and latency functions for the edges of $\mathcal{G}$.

We focus on the sets of languages $\mathcal{L}_{nowait} = \{L_{nowait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$ and $\mathcal{L}_{wait} = \{L_{wait}(\mathcal{G}) : \mathcal{G} \in \mathcal{U}\}$, where $\mathcal{U}$ is the set of all time-varying graphs; that is, we look at the languages expressed when waiting is, or is not allowed. For each of these two sets, the complexity of recognizing any language in the set (that is, the computational power needed by the accepting automaton) defines the level of difficulty of the environment.

We first study the expressivity of time-varying graphs when waiting is not allowed, that is the only feasible journeys are direct ones. We prove that the set $\mathcal{L}_{nowait}$ contains all computable languages.

THEOREM 2.1. *For any computable language L, there exists a time-varying graph $\mathcal{G}$ such that $L = L_{nowait}(\mathcal{G})$*

The proof is constructive.

We next examine the expressivity of time-varying graphs if indirect journeys are allowed, that is entities have the choice to wait for future opportunities of interaction rather than seizing only those that are directly available. In striking contrast with the non-waiting case, we show that the languages $\mathcal{L}_{wait}$ recognized by TVG-automata is precisely the set of *regular* languages.

THEOREM 2.2. *$\mathcal{L}_{wait}$ is the set of* regular *languages.*

We introduce a quasi-order on words based upon the possibility of inclusion for corresponding journeys. The fact that it is a well quasi-order cannot be derived by a simple application of classical results of the domain (eg. from [3]) but can be proved using a specific technique. The proof is algebraic and relies on a theorem by Harju and Ilie (Theorem 4.16 in [2]) that enables to characterize regularity from the closure of the sets for a well quasi-order.

In other words, we prove that, when waiting is no longer forbidden, the power of the accepting automaton (i.e., the difficulty of the environment, the power of the adversary), drops drastically from being at least as powerful as a Turing machine, to becoming that of a Finite-State Machine. This (perhaps surprisingly large) gap is a measure of the computational power of waiting.

To better understand the power of waiting, we then turn our attention to *bounded waiting*; that is when indirect journeys are considered feasible if the pause between consecutive edges in the journeys have a bounded duration $d > 0$. In other words, at each step of the journey, waiting is allowed only for at most d time units. We examine the set $\mathcal{L}_{wait[d]}$ of the languages expressed by time-varying graphs in this case and prove the negative result that the complexity of the environment is not affected by allowing waiting for a limited amount of time.

THEOREM 2.3. *For any fixed $d \geq 0$, $\mathcal{L}_{wait[d]} = \mathcal{L}_{nowait}$,.*

The basic idea of the proof is to reuse the same technique as for the nowait case, but with a dilatation of time, i.e., given the bound d, the edge schedule is time-expanded by a factor d (and thus no new choice of transition is created compared to the no-waiting case). As a result, the power of the adversary is decreased only if it has no control over the length of waiting, i.e., if the waiting is unpredictable.

3. REFERENCES

[1] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In *Proc. 10th Int. Conf. on Ad Hoc Networks and Wireless (ADHOC-NOW)*, pages 346–359, 2011.

[2] T. Harju and L. Ilie. On quasi orders of words and the confluence property. *Theoretical Computer Science*, 200(1-2):205–224, 1998.

[3] G. Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, s3-2:326–336, 1952.