Far East J. Math. Sci. (FJMS) **22**(1) (2006), 43-67
This paper is available online at http://www.pphmj.com

A DISCRETE DYNAMICAL SYSTEMS FRAMEWORK FOR PACKET-FLOW ON NETWORKS

A. Å. HANSSON and C. M. REIDYS

(Received December 10, 2005)

Submitted by K. K. Azad

Abstract

In this paper we introduce a certain class of network-routing protocols over arbitrary simple, undirected graphs. These protocols transmit data-packets between adjacent vertices and will be formally specified as local maps of a sequential dynamical system (SDS). An SDS consists of (a) a finite (labeled) graph Y with vertex set $\{v_1, ..., v_n\}$, where each v_i has a finite state, $x_{v_i} \in K$, (b) a vertex labeled multi-set of L_Y-local functions $(F_{v_i, Y})_{w_i}$, where L_Y is a mapping from Y-vertices into subgraphs of Y and (c) a word w, i.e., a multi-set $(w_1, ..., w_k)$, where w_i is a Y-vertex. The local function $F_{w_i, Y}$ updates the state of vertex w_i as a function of the vertices contained in the subgraph $\mathsf{L}_Y(w_i)$ and simultaneously updates all states of the $\mathsf{L}_Y(w_i)$-vertices other than w_i. The SDS is then the product of the local maps: $[\mathfrak{F}_Y, w] = \prod_{i=1}^{k} F_{w_i} : K^n \to K^n$.

We will specify generic, parameterized data transmission protocols as $\mathsf{L}_Y(v)$-local maps, where $\mathsf{L}_Y(v)$ is a subgraph of Y. The protocols will facilitate the transmission of unlabeled packets from a fixed source S to a fixed destination D and route using the distance to D, the actual load of the neighbors and their total capacities as base parameters. The state of a vertex will be the vector of queue-sizes of their respective next-hop-

2000 Mathematics Subject Classification: 05C25, 37F99, 37N99.

Keywords and phrases: sequential dynamical system, asynchronous cellular automation, packet-switched networks, routing protocols.

destinations. We will present several instances of the system dynamics produced by these protocols and utilize the SDS-framework in order to study which update schedules induce equal or equivalent systems. Finally we investigate further system symmetries that arise from automorphisms of the base graph itself using group actions and allow to determine additional equivalences.

1. Introduction

Communication theory is the legacy of seminal work by Wiener, Kotelnikov, and Shannon. It assumes circuit-switching, in which a dedicated channel is allocated for transmission between two (or more) parties. This classical theory of communication can be contrasted to *packet-switching*, where the data flow is divided into packets prior to transmission. Each packet is transmitted individually and is routed via multiple switching nodes, possibly along multiple paths, before it reaches its destination. Intuitively, this type of communication is more robust, but it is unfortunately not well understood today.

The scope of this paper is to cast packet-switching problems in the particular mathematical framework of *sequential dynamical systems* (SDS). Such a formulation has the potential to shed new light on networking in general, and routing at the networking layer in particular.

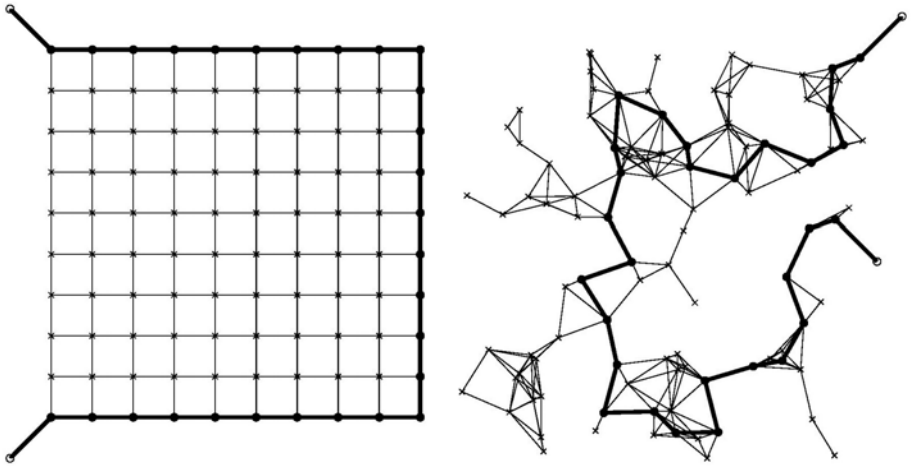
A standard metric for routing decisions is the hop-count, which is useful in estimating the shortest path to the destination. In ad hoc (or self-organized) networks with mobile radio transceivers, this metric is not reliable at all times, and several routing protocols have been proposed to alleviate this problem [3, 7]. However, hop-count is not sufficient to guarantee effective forwarding decisions, and the IETF has suggested that routing protocols should include additional metrics to become *congestion-aware*. Some examples of these new metrics are *maximum link bandwidth*, *packet loss ratio*, and *link propagation delay*. A limited number of congestion-aware routing protocols have been proposed in the literature [2, 4, 5, 12, 13, 15]. The common approach is to adopt an end-to-end perspective in the spirit of TCP in that the perceived congestion is a function of the states of *all* nodes along a path. In contrast, we study a class of *locally* load-sensing protocols, and we are only aware of one paper

that elaborates on similar ideas [14]. To our knowledge, this is a first attempt to rigorously formalize these ideas.

In this paper, we simplify the analysis by assuming that all links are static and perfectly reliable (or error-free) with zero delay. The network is modeled as a simple, undirected graph, Y, in which we fix a *source* and *destination* vertex S and D, respectively, and we study the flow of (discrete) data-packets from S to D. We assume that packets can only be transmitted from one vertex to another if they are adjacent in Y, i.e., only Y-local transmission is possible. In particular, we investigate how our protocols perform over grid graphs and random geometric graphs, exemplified in Figure 1.

We will be interested in describing the dynamics of the queue sizes, i.e., the number of packets located at the respective vertices. More precisely, for fixed vertex v we will consider all queue-sizes of the packets at v whose next hop-destination is the v-neighbor v_i, individually. That is, for fixed vertex we consider a multi-set (or vector) of queue-sizes. For our purposes we will deal with *unlabeled packets*, i.e., our packets do not contain routing information in their headers and cannot be addressed individually. This ansatz implies that our routing procedure has to be encoded in the vertex itself, which is possible, since we assume that all packets have one (or more) *fixed* destination(s). Explicitly we will assume that some large number of packets are injected into the network via the source vertex and that the destination has enough capacity to receive all data-packets. The source successively loads the network and after some finite number of steps the system reaches an orbit in phase space. We will study the time evolution of the system, in particular the time evolution of the total number of packets located at the vertices, i.e., *total load* and various other quantities, like for instance the *throughput*, i.e., the rate at which packets arrive at the destination.

Our protocols will be defined within the framework of sequential dynamical systems (SDS), as introduced in [6, 11]. This framework will suggest a certain perspective on our investigations, like for instance to study system dependence on the particular choice of the update schedule.



(a) (b)

Figure 1. (a) Let Y_0 be the 10×10 grid graph in which all nodes on the path p_0 (bold edges and vertices) have queue capacity 1000, while all other nodes have queue capacity 10. The source and the sink are connected to the lower left and upper left corners, respectively.

(b) Let Y_1 be a random geometric graph over 100 nodes with diameter is 22. p_1 is a distance-30 path (bold edges and vertices) from the source to the sink, along which all nodes have queue capacity 1000. The remaining nodes all have queues that are restricted to buffer at most 10 packets.

We will define SDS in detail in the next section. Intuitively an SDS is a dynamical system generated by the composition of local, vertex-labeled functions. Each of these local functions updates the state of vertex k and the states of the vertices contained in the Y-subgraph $\mathsf{L}_Y(k)$ (which contains at least the vertex k itself) as a function of the states of all $\mathsf{L}_Y(k)$-vertices. Related models, like for example, asynchronously updated Boolean networks or asynchronous cellular automata (CA) exclusively update the state of a single vertex based on the states of its respective neighborhood. In the context of packet transmission protocols it is evident that we need to update more than one vertex at the time,

since the transmission of a packet *simultaneously* changes the queue-size of the sending and receiving vertex. Brooks et al. [1] have provided experimental evidence that CA can model UDP and TCP traffic. Although their work inspired us, our focus is different in that we aim at analyzing a novel class of congestion-aware routing protocols.

The paper is structured as follows: In Section 2 we introduce the framework of L-local sequential dynamical systems. In Section 3 we then specify our systems and define our data-transmission protocols as local maps of SDS and proceed in Section 4 by presenting several paradigms of the system dynamics induced by these protocols. Finally, we show in Section 5 two instances of how SDS concepts can be used in order to identify equivalence classes of dynamical systems. One instance is based on a combinatorial structure w.r.t. the update schedule of SDS and the other arises from a natural group action of $\mathsf{Aut}_D(Y)$, the subgroup of Y-automorphisms, which fix the destination vertex D.

2. Sequential Dynamical Systems

Let Y be a loop-free, labeled, simple, undirected graph with *vertex set* $\mathrm{v}[Y] = \{v_1, ..., v_n\}$ and *edge set* $\mathrm{e}[Y]$. A *word* over Y is a finite tuple $(w_1, w_2, ..., w_k)$, where $w_i \in \mathrm{v}[Y]$, for $i = 1, ..., k$. We set

$$\mathsf{B}_{1,Y}(v_i) = \{v_j \in \mathrm{v}[Y] \mid v_j = v_i \vee \{v_j, v_i\} \in \mathrm{e}[Y]\}. \tag{2.1}$$

Each vertex v_i has an associated state $x_{v_i} \in K$, where K is some finite domain and we call the n-tuple

$$(x_{v_1}, ..., x_{v_n})$$

the *system state*. Let $d(v_i)$ be the vertex-degree of v_i in Y. Let further

$$\mathsf{L} : \mathrm{v}[Y] \to \{X \mid X \text{ is a subgraph of } Y\}, \quad v_i \mapsto \mathsf{L}(v_i) \tag{2.2}$$

and let $\lambda(v_i)$ denote the cardinality of the vertex set of the subgraph $\mathsf{L}(v_i)$. We set

$$f_{v_i} : K^{\lambda(v_i)} \to K^{\lambda(v_i)}. \tag{2.3}$$

For each vertex $v_i \in \mathrm{v}[Y]$ we consider the sequence

$$s_Y(v_i) = (x_{v_{j_1}}, ..., x_{v_{j_s}}, x_{v_{j_{s+1}}} = x_{v_i}, x_{v_{j_{s+2}}}, ..., x_{v_{j_r}}), \tag{2.4}$$

where $j_t < j_{t+1}$ and $v_{j_h} \in \mathsf{L}(v_i)$.

We then introduce the map

$$\mathsf{proj}[v_i] : K^n \to K^{\lambda(v_i)},$$

$$(x_{v_1}, ..., x_{v_n}) \mapsto (x_{v_{j_1}}, ..., x_{v_{j_s}}, x_{v_i}, x_{v_{j_{s+2}}}, ..., x_{v_{j_r}}), \tag{2.5}$$

which projects into the states of $\mathsf{L}(v_i)$-vertices. We now have the compositum

$$f_{v_i} \circ \mathsf{proj}[v_i](x) : K^n \to K^{\lambda(v_i)}$$

and write

$$f_{v_i} \circ \mathsf{proj}[v_i](x) = f_{v_i}((x_{v_{j_1}}, ..., x_{v_{j_s}}, x_{v_i}, x_{v_{j_{s+2}}}, ..., x_{v_{j_r}}))$$

$$= (y_{v_{j_1}}, ..., y_{v_{j_s}}, y_{v_i}, y_{v_{j_{s+2}}}, ..., y_{v_{j_r}}).$$

Via $f_{v_i} \circ \mathsf{proj}[v_i]$ we now define the L-local map of v_i, $F_{v_i,Y} : K^n \to K^n$:

$$F_{v_i,Y}(x) = (y_{v_1}, ..., y_{v_n}), \tag{2.6}$$

where

$$y_{v_h} = \begin{cases} (f_{v_i} \circ \mathsf{proj}[v_i](x))_{v_h}, & \text{for } v_h \in \mathsf{L}(v_i) \\ x_{v_h}, & \text{otherwise.} \end{cases}$$

In the following we will assume that

$$\mathsf{L}(v_i) = \mathsf{Star}_Y(v_i), \tag{2.7}$$

where $\mathsf{Star}_Y(v_i)$ is the Y-subgraph with vertex set $\mathrm{v}[\mathsf{Star}_Y(v_i)] = \mathsf{B}_{1,Y}(v_i)$ and edge set $\mathrm{e}[\mathsf{Star}_Y(v_i)] = \{\{v_i, v_j\} | \{v_i, v_j\} \in \mathrm{e}[Y]\}$.

Example. Let

$$Y = \begin{array}{ccc} v_1 & \text{———} & v_2 \\ | & & | \\ v_4 & \text{———} & v_3 \end{array}.$$

Then we have

$$\mathsf{Star}_Y(v_1) = v_4 \text{ ——— } v_1 \text{ ——— } v_2$$

$$\mathsf{Star}_Y(v_2) = v_1 \text{ ——— } v_2 \text{ ——— } v_4$$

$$\mathsf{Star}_Y(v_3) = v_2 \text{ ——— } v_3 \text{ ——— } v_4$$

$$\mathsf{Star}_Y(v_4) = v_3 \text{ ——— } v_4 \text{ ——— } v_1.$$

In case of $\mathsf{L}(v_i) = \mathsf{Star}_Y(v_i)$, the $\mathsf{L}(v_i)$-local functions are of the form $f_{v_i} : K^{\delta(v_i)+1} \to K^{\delta(v_i)+1}$ and we have the $\mathsf{Star}_Y(v_i)$-local maps

$$F_{v_i, Y}(x) = (y_{v_1}, ..., y_{v_n}), \tag{2.8}$$

where

$$y_{v_h} = \begin{cases} (f_{v_i} \circ \mathsf{proj}[v_i](x))_{v_h}, & \text{for } v_h \in \mathsf{B}_{1,Y}(v_i) \\ x_{v_h}, & \text{otherwise.} \end{cases}$$

Now we are prepared to define an L-local SDS over a word [9, 10]:

Definition 1. Let $w = (w_1, ..., w_k)$ be a word, $\mathsf{L} : \mathrm{v}[Y] \to \{X < Y\}$ and $[\mathfrak{F}_Y, \ ]$ be the mapping

$$[\mathfrak{F}_Y, \ ] : W \to \mathsf{Map}(K^n, K^n), \quad [\mathfrak{F}_Y, w] = \prod_{i=1}^{k} F_{w_i, Y}. \tag{2.9}$$

We call $[\mathfrak{F}_Y, w]$ the L-*local sequential dynamical system* (SDS) (over Y and w). In case of $\mathsf{L}(v_i) = \mathsf{Star}_Y(v_i)$ we simply call $[\mathfrak{F}_Y, w]$ an SDS. The *phase space* of $[\mathfrak{F}_Y, w]$ is the digraph $G[\mathfrak{F}_Y, w]$ with vertex set $x = (x_{v_i})_i$ and directed edges $(x, [\mathfrak{F}_Y, w](x))$.

We immediately observe that $G[\mathfrak{F}_Y, w]$ is a *unicyclic digraph* and we call $G[\mathfrak{F}_Y, w]$-vertices contained in $G[\mathfrak{F}_Y, w]$-cycles the *periodic points* of $[\mathfrak{F}_Y, w]$. We denote the set of periodic points of an SDS $[\mathfrak{F}_Y, w]$ by $\mathsf{Per}[\mathfrak{F}_Y, w]$. We call two SDS $[\mathfrak{F}_Y, w]$ and $[\mathfrak{F}'_Y, w']$ *equivalent* and write $[\mathfrak{F}_Y, w] \sim [\mathfrak{F}'_Y, w']$ if and only if there exists a *digraph-isomorphism*

$$\varphi : G[\mathfrak{F}_Y, w] \to G[\mathfrak{F}'_Y, w'], \tag{2.10}$$

i.e., we have the commutative diagram

$$\begin{array}{ccc} K^n & \xrightarrow{[\mathfrak{F}_Y, w]} & K^n \\ \downarrow{\scriptstyle \varphi} & & \uparrow{\scriptstyle \varphi^{-1}} \\ K^n & \xrightarrow{[\mathfrak{F}'_Y, w']} & K^n \end{array}.$$

3. SDS-specification of Transmission Protocols

In this section we will utilize the SDS framework described in Section 2 under the assumption $\mathsf{L}(v_i) = \mathsf{Star}_Y(v_i)$ to introduce a data transmission protocol as a local map of an SDS. The protocols will be able to route dynamically, based on purely Y-local information and have static global knowledge in form of the path-lengths in Y. That is, in addition to *a priori* information about the distance to the destination vertex D the transmission protocols are *locally load-sensing*.

3.1. Vertex and system states

Let

$q_{k,i}$ = the number of packets at vertex k with next location i,

$m_1(k)$ = the maximal queue-size for packets located at vertex k,

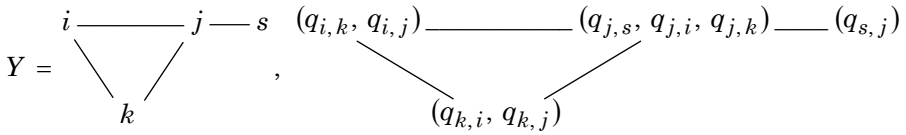
$m_2((k, i))$ = the maximal number of packets that can be transmitted via an edge from k to i.

Without loss of generality we will assume that $m_1(k)$ is the cardinality of a finite field, K, i.e., we have $m_1(k) = p^m$, where p is the

characteristic of K. The state of a *vertex* k is the tuple

$$(q_{k,i})_{i\in \mathsf{S}_1(k)}, \quad \text{where } q_{k,i} \in K. \tag{3.1}$$

That is, the state of a vertex is the collection of queues of packets that are subject to be forwarded to its nearest neighbors at the next update step. For instance we have for



Accordingly, the state of the *system* is then tuple

$$(q_{i,k}, q_{k,i})_{\{i,k\}\in \mathrm{e}[Y]} \tag{3.2}$$

since each edge $\{i, k\}$ gives rise to the two queues $q_{i,k}$ and $q_{k,i}$.

3.2. α-maps

By construction, we cannot track individual packet identities. Once "sorted" into the corresponding queues any two data-packets are virtually identical. One consequence of this ansatz is that we need to create some mechanism by which forwarded packets are distributed into the corresponding queues of the receiving vertex. We will achieve this by introducing the multi-set of vertex-indexed mappings, $(\alpha_k)_{k\in \mathrm{v}[Y]}$.

Let $\mathsf{S}_1(k) = \{i_1, \ldots, i_{\delta(k)}\}$, $t = (t_{i_1}, \ldots, t_{i_{\delta(k)}})$ be the state of vertex k (i.e., the vector of queue-sizes) and $q = (q_{i_j})_{1\le j\le \delta(k)}$ the vector of total queue-sizes of the k-neighbors. Then an α-map at k, α_k, is a mapping of the form

$$\alpha_k : K \times K^{\delta(k)} \times K^{\delta(k)} \to K^{\delta(k)},$$

$$(s, (t_{i_1}, \ldots, t_{i_{\delta(k)}}), (q_{i_1}, \ldots, q_{i_{\delta(k)}})) \mapsto (h_{i_1}, \ldots, h_{i_{\delta(k)}}) \tag{3.3}$$

with the property $\sum_{j=1}^{\delta(k)} h_{i_j} \le s + \sum_{j=1}^{\delta(k)} t_{i_j}$. We may interpret an α-map associated to a vertex k as to facilitate the *distribution* of packets sent to

k into their respective queues. The map will factor in two types of information: *a priori global information* (in form of the distance to the respective destination vertex) and *dynamic local information* (in form of the actual queue-sizes of the immediate neighbors, i.e., the vector $q = (q_{i_j})_j$).

Suppose that m packets are being forwarded to k. In the following we will define two α-maps. Obviously, k can receive at most

$$\widetilde{m} = \min\left\{m,\ m_1(k) - \sum_{j \in \mathsf{S}_{1,Y}(k)} q_{k,j}\right\} \tag{3.4}$$

packets. We introduce the function

$$H_{k,f,g}(x,\ y,\ z) = f(x)^{a_k} \cdot g(y)^{b_k} \cdot z^{c_k}, \tag{3.5}$$

where f, g are monotone functions $f, g : [0, 1] \to [0, 1]$ such that $f(0) = g(0) = 1$ and $f(1) = g(1) = 0$ and a_k, b_k, c_k are positive constants. Explicitly we will set for our experiments in Section 4:

$$f(x) = \cos(\beta \cdot \pi/2 \cdot x), \quad g(y) = \cos(\pi/2 \cdot y) \ \text{ with } \ 0 < \beta < 1. \tag{3.6}$$

We denote the *relative distance* to the destination D and the *relative load* of vertex i_s by

$$d^*_Y(i_s,\ D) = d_Y(i_s,\ D)/\mathrm{diam}(Y), \ \text{ and } \ q^*_Y(i_s) = \left(\sum_{j \in \mathsf{S}_1(i_s)} q_{i_s,j}\right) \Big/ m_1(i_s),$$

respectively and

$$w(i_s) = H_{k,Y}(d^*_Y(i_s,\ D),\ q^*_Y(i_s),\ (m_1(i_s) \cdot m_2((k,\ i_s)))). \tag{3.7}$$

Finally we normalize $w^*(i_s) = w(i_s) \Big/ \left(\sum_{j \in \mathsf{S}_1(k)} w(j)\right)$.

The mapping α_k: We set

$$y_{i_r} = \lfloor \widetilde{m} \cdot w^*(i_r) \rfloor \ \text{ and } \ \Delta = \widetilde{m} - \sum_{i_j \in \mathsf{S}_1(k)} y_{i_j}. \tag{3.8}$$

Without loss of generality, we may assume $y_{i_1} \geq y_{i_j}$, $j \in \mathsf{S}_1(k)$ and define

$$s_{i_r} = \begin{cases} y_{i_r}, & \text{for } r \neq 1 \\ y_{i_r} + \Delta, & \text{for } r = 1. \end{cases} \tag{3.9}$$

Accordingly we have (note that $t_{i_r} + s_{i_r} = q_{k,i_r}$)

$$\alpha_k(m, t, q) = \alpha_k(\tilde{m}, t, q) = (s_{i_1} + t_{i_1}, ..., s_{i_{\delta(k)}} + t_{i_{\delta(k)}}),$$

and

$$\sum_{i_s \in \mathsf{S}_{1,Y}(k)} s_{i_s} = \tilde{m}, \tag{3.10}$$

where $q = (q_{i_j})_j$ with $q_{i_j} = \sum_{g \in \mathsf{S}_1(i_j)} q_{i_j,g}$ is the vector of total queue-sizes of the k-neighbors.

The mapping α'_k: Let now

$$y_{i_r} = \left\lfloor \left(\tilde{m} + \sum_j t_{i_j} \right) \cdot w^*(i_r) \right\rfloor \text{ and } \Delta = \left(\tilde{m} + \sum_j t_{i_j} \right) - \sum_{i_j \in \mathsf{S}_1(k)} y_{i_j}. \tag{3.11}$$

Again, we may, without loss of generality, assume that y_{i_1} is maximal and define

$$s_{i_r} = \begin{cases} y_{i_r}, & \text{for } r \neq 1 \\ y_{i_r} + \Delta, & \text{for } r = 1. \end{cases} \tag{3.12}$$

Accordingly we have

$$\alpha'_k(m, t, q) = \alpha'_k\left(\tilde{m} + \sum_j t_{i_j}, 0, q \right) = (s_{i_1}, ..., s_{i_{\delta(k)}}). \tag{3.13}$$

3.3. Transmission protocols as local maps

We introduced local maps in Section 2. Since we have $\mathsf{L}(v_i) = \mathsf{Star}_Y(v_i)$ (eq. (2.7)) a local map is a mapping

$$F_{k,Y} : K^{2|\mathrm{e}[Y]|} \to K^{2|\mathrm{e}[Y]|}$$

with the property

$$F_{k,Y}(x) = (y_1, \ldots, y_m), \text{ where } y_h = \begin{cases} (f_k \circ \mathrm{proj}[k](x))_h, & \text{for } h \in \mathsf{B}_{1,Y}(k) \\ x_h, & \text{otherwise.} \end{cases}$$

For $i \in \mathsf{S}_1(k)$ we now consider

$$\xi_i = \xi_i((q_{h,s}, q_{s,h})_{\{h,s\}\in \mathrm{e}[Y]}) = \min\left\{ m_2((k, i)), q_{k,i}, m_1(i) - \sum_j q_{i,j} \right\}, \quad (3.14)$$

i.e., the number of packets that k can actually send to its neighbor i. For $i \in \mathsf{S}_1(k)$ and $a \in \mathsf{S}_1(i)$, we set

$$\widetilde{q}_a = \begin{cases} \sum_{i\in \mathsf{S}_1(k)} q_{k,i} - \sum_i \xi_i & \text{for } a = k \\ \sum_{x\in \mathsf{S}_1(a)} q_{a,x} + \xi_a & \text{for } a \in \mathsf{S}_1(i) \cap \mathsf{S}_1(k) \\ \sum_{x\in \mathsf{S}_1(a)} q_{a,x} & \text{for } a \in \mathsf{S}_1(i) \setminus \mathsf{S}_1(k). \end{cases} \quad (3.15)$$

We now define the α-map-induced local map as follows:

$$(F_{k,Y}(q_{h,s}))_{(h_0,s_0)}$$

$$= \begin{cases} q_{k,i} - \xi_i & \text{for } h_0 = k,\ s_0 = i \in \mathsf{S}_1(k) \\ q_{i,j} + \alpha_i(\xi_i, t, (\widetilde{q}_a)_{a\in \mathsf{S}_1(i)})_j & \text{for } h_0 = i \in \mathsf{S}_1(k),\ s_0 = j \in \mathsf{S}_1(i). \end{cases} \quad (3.16)$$

In other words, vertex k forwards *in parallel* data-packets to all its neighbors. These then apply their corresponding α-mappings, taking into account the forwarded packets sent from k.

One first observation w.r.t. SDS induced by $F_{k,Y}$ is that they can *deadlock*, i.e., produce a 0-packet throughput. In Figure 2 and Table 1 we present an example of a system deadlock.

Table 1. Analysis of the deadlock illustrated in Figure 2

	F-protocol											*F'* -protocol										
After	queue *y*			*z*		*x*			*w*			*y*			*z*		*x*			*w*		
scheduling	*v*	*z*	*x*	*y*	*x*	*y*	*z*	*w*	*x*	·	·	*v*	*z*	*x*	*y*	*x*	*y*	*z*	*w*	*x*	·	·
node *x*	0	10	0	0	0	8	0	0	0	0	0	0	10	0	0	0	8	0	0	0	0	0
y	0	0	0	10	0	8	0	0	0	0	0	0	0	0	10	0	4	0	4	0	0	0
z	0	10	0	0	0	8	0	0	0	0	0	0	10	0	0	0	4	0	4	0	0	0
v	0	10	0	0	0	8	0	0	0	0	0	0	10	0	0	0	4	0	4	0	0	0
w	0	10	0	0	0	8	0	0	0	0	0	0	10	0	0	0	4	0	4	0	0	0

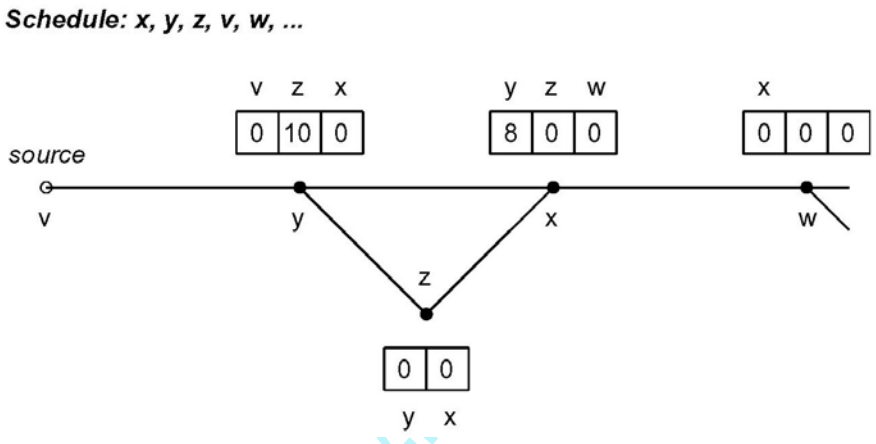


Figure 2. Exemplification of system deadlock. All queues have a capacity of 10 packets (with the exception of the source, which by definition is full at all time instants). Node x and y have distances $d \geq 21$ and $d+1$ to the sink, respectively. By assumption we schedule x first. (x) : since y is full it cannot forward any packets. (y) : the 10 packets in y that are destined to z are all distributed to sub-queue y of node z (after rounding) since x is full to 80% of its capacity. (z) : z returns all 10 packets to node y, and now all packets are being sent back to sub-queue z of node y (the packets are "bouncing" between node y and z). (v) : source v cannot send any packets since node y is full. (w) : there are no packets to send.

We next define a local map which automatically avoids the build-up of queues which cannot be processed. This local map is induced by the mapping α' (eq. (3.11)):

$$(F'_{k,Y}(q_{h,s}))_{(h_0,s_0)}$$

$$= \begin{cases} q_{k,i} - \xi_i, & \text{for } h_0 = k, s_0 = i \in \mathsf{S}_1(k) \\ \alpha_i\left(\xi_i + \sum_{j \in \mathsf{S}_1(i)} q_{i,j}, 0, (\tilde{q}_a)_{a \in \mathsf{S}_1(i)}\right)_j, & \text{for } h_0 = i \in \mathsf{S}_1(k), s_0 = j \in \mathsf{S}_1(i). \end{cases} \tag{3.17}$$

Table 1 shows the local map F' resolves the deadlock of the SDS induced by F.

4. Packet Routing Dynamics: Two Examples

In this section we present two examples of the system dynamics induced by the network-protocol associated with the local map F. Rather than providing an in-depth analysis, which will be subject of a subsequent paper, the purpose of this section is to show that the F-protocols have interesting and sometimes counterintuitive features.

By definition, the system dynamics is produced by iterating the mapping

$$[\mathsf{F}_Y, \sigma] = \prod_{i=1}^{n} F_{\sigma(k),Y} : K^{2|\mathrm{e}[Y]|} \to K^{2|\mathrm{e}[Y]|}, \tag{4.1}$$

where σ is some permutation of the Y-vertices. We furthermore set

$$H_{k,f,g}(x, y, z) = \cos(\beta \cdot \pi/2 \cdot x)^{a_k} \cdot \cos(\pi/2 \cdot y)^{b_k} \cdot z^{c_k}. \tag{4.2}$$

In our first experiment we consider the graph Y_0 of Figure 1. By assumption the vertices on the path p_0 have buffers with queue-size 1000 and all others have buffer size 10. In Figure 3 we run two protocols. The first protocol, P1, has the parameter combination $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$ and the second, P2, has parameters $a_k = 1$, $b_k = 0$, $c_k = 0$ and $\beta = 0.8$. The simulation shows that the performance of P1 strongly depends on the choice of schedule, i.e., the permutation σ

according to which we execute the local data transmission. Furthermore Figure 3 indicates that a rationally designed schedule, in this case the one that updates the vertices of p_0 from source to sink, is not the best choice in terms of data throughput. Finally, protocol P2 which can be interpreted as some version of a *shortest path protocol* does perform relatively poorly as it basically locks up the unique shortest path connecting source and sink. In Figure 4 we display the *activities* of the above two protocols, i.e., how many packets are being transmitted during one system iteration (excluding the source destination pair). Here we observe that the design and the best performing random schedule have almost identical activities while another random schedules activity decays after initially peaking. It appears that after the initial loading of the system that particular schedule "freezes." As expected P2 never shows any high activity. In Figure 5 we monitor the *load* of the system, i.e., the total number of packets in the network.

In our second experiment we consider the graph Y_1 of Figure 1. We proceed in analogy to our first experiment and display, *throughput* (Figure 6), *activity* (Figure 7), and *load* (Figure 8) for the two protocols P1 and P2. Here P1 has parameters $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$; and P2 is again the shortest path protocol with parameters $a_k = 1$, $b_k = 0$, $c_k = 0$ and $\beta = 0.8$. The key observation from Figure 6 is how poorly the designed schedule performs. In analogy to the situation for Y_0 the designed schedule updates the nodes on the path p_1 from source to sink. After an initial loading phase its throughput decays almost to the level of poorly performing random schedules. It is accordingly not able to maintain a high throughput. We can conclude from Figure 7 that all three protocols exhibit similar activities. Despite the significant performance differences all protocols transmit packets and do not "freeze." Only P2 has a distinctively lower activity. Finally, Figure 8 shows that random schedule 1 maintains its system load while the two others decay after what seems to be an initial loading phase in which both protocols exceed the load of random schedule 1. Despite the fact that P2 exhibits very low activity, its load eventually exceeds the load of P1 with the designed schedule.

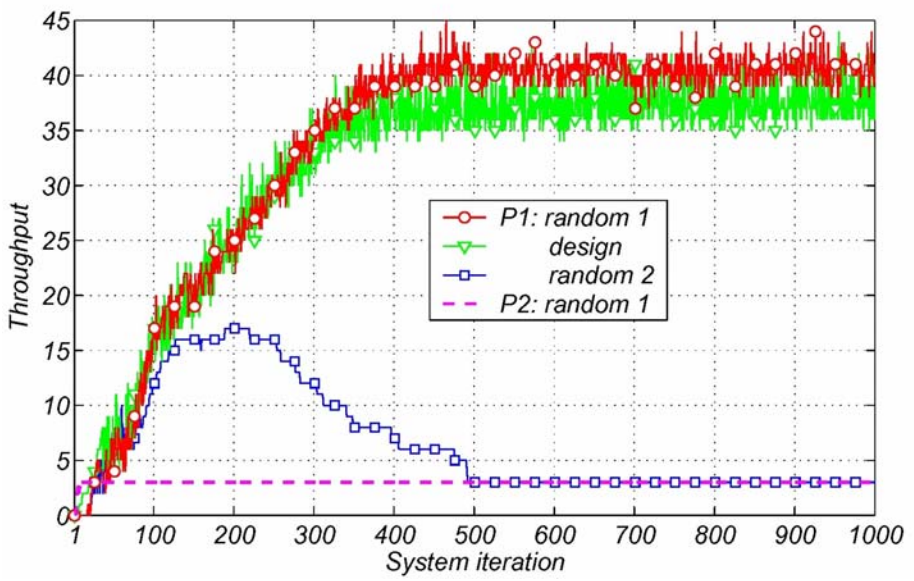


Figure 3. Throughput of Protocol 1 (P1) for three different update schedules, over Y_0 of Figure 1. P1 has the parameter combination $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$. For reference, we have also plotted P2 with parameters $a_k = 1$, $b_k = 0$, $c_k = 0$, $\beta = 0.8$.

5. Equivalence

In the following we show how to use the SDS-framework in order to classify and categorize the system dynamics based on specific knowledge of the underlying base graph. We will show how combinatorial properties of the underlying base graph can be used in order to compute *a priori* combinatorial *equivalence classes* of dynamical systems.

Our first equivalence result is a consequence of the fact that an SDS is a composition product of local maps. We will start by observing that any two local maps, indexed by vertices which have a sufficient Y-distance commute. This will give rise to combinatorial equivalence classes of the corresponding update schedules.

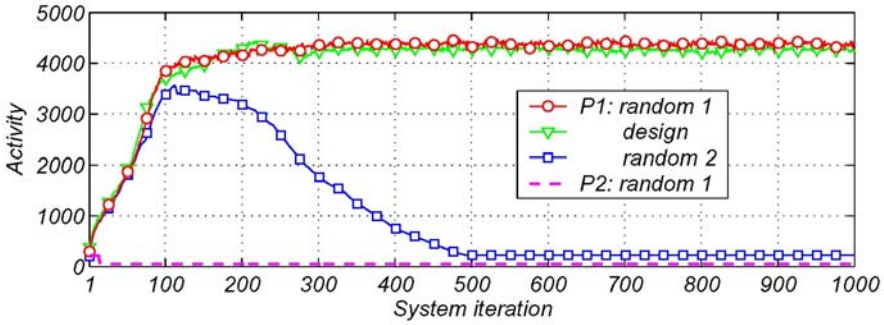


Figure 4. Activity of P1 for three different update schedules, all employed over Y_0. P1 uses the parameter combination $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$. Activity is defined as the number of sent packets (aggregated over the whole network, excluding the source/sink pair). For reference, we have also plotted P2 with parameters $a_k = 1$, $b_k = 0$, $c_k = 0$, $\beta = 0.8$.

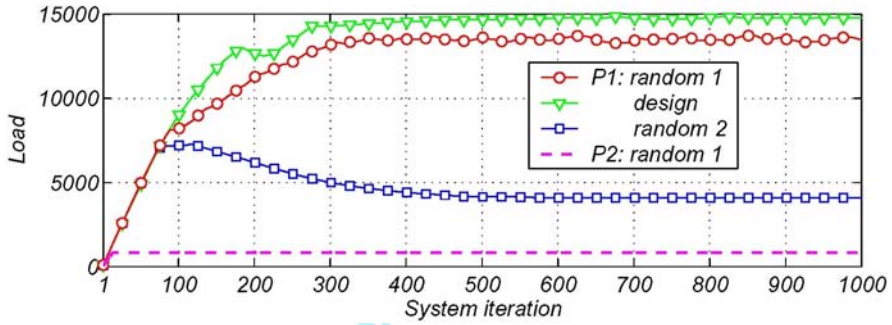


Figure 5. Load of P1 for three different update schedules, over Y_0. P1 uses the parameter combination $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$. Load is defined as the number of buffered packets (aggregated over the whole network, excluding the source/sink pair). For reference, we also present P2 with parameters $a_k = 1$, $b_k = 0$, $c_k = 0$, $\beta = 0.8$.

Lemma 1. *Let* $k, o \in \mathrm{v}[Y]$ *with the property* $d_Y(k, o) > 4$. *Then we have*

$$F_{k,Y} \circ F_{o,Y} = F_{o,Y} \circ F_{k,Y}.$$

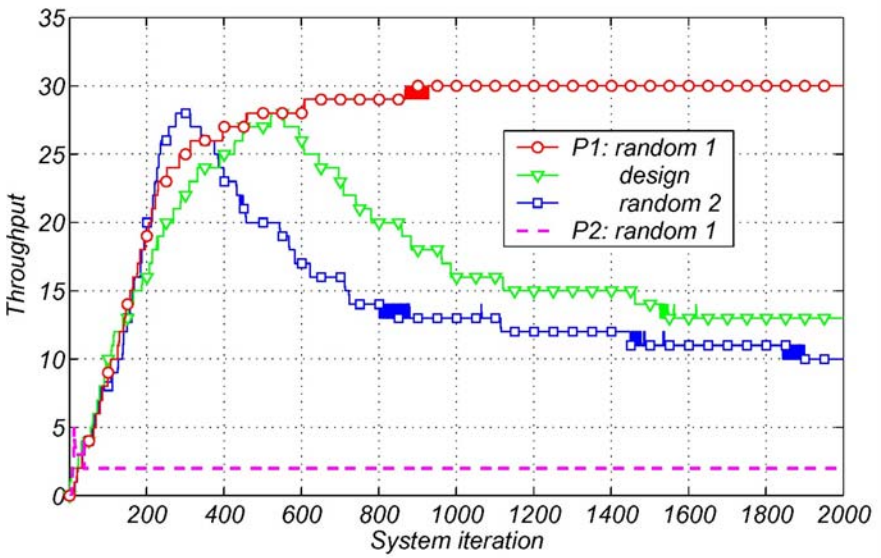
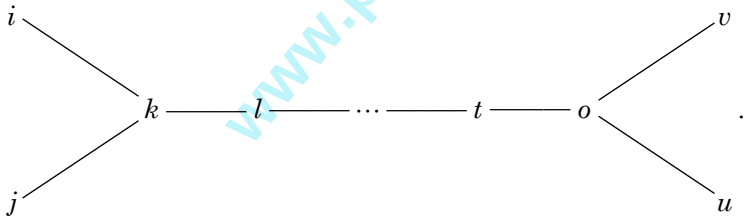


Figure 6. Throughput of Protocol 1 (P1) for three different update schedules, over the random geometric graph Y_1 of Figure 1. P1 uses the parameter combination $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$. For reference, we have also plotted Protocol 2 (P2), for which $a_k = 1$, $b_k = 0$, $c_k = 0$, $\beta = 0.8$, that is, P2 corresponds to a shortest path protocol.

Proof. We have the following situation



By construction the update function $F_{s,Y}$ potentially alters the states of all vertices contained in $\mathsf{B}_{1,Y}(s)$ and there exists no k-neighbor which is adjacent to some o-neighbor. Therefore we obtain

$$F_{k,Y} \circ F_{o,Y} = F_{o,Y} \circ F_{k,Y}$$

and the lemma follows.

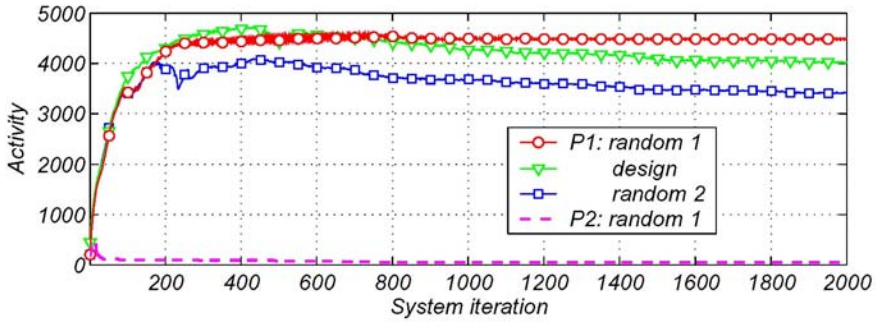


Figure 7. Activity of P1 for three different update schedules, over Y_1. P1 has parameters $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$. We have also plotted P2 with $a_k = 1$, $b_k = 0$, $c_k = 0$, $\beta = 0.8$.

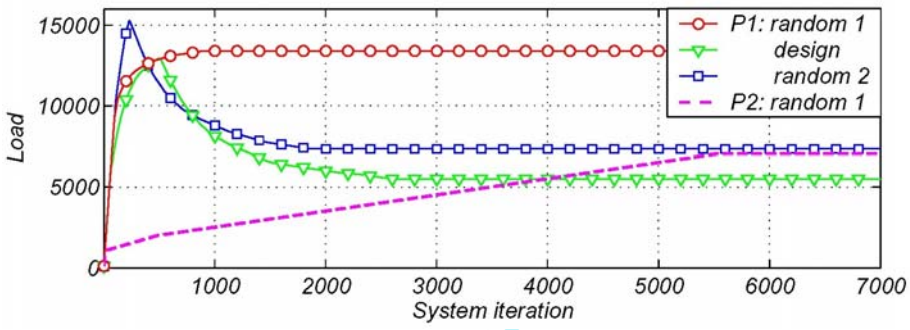


Figure 8. Load of P1 for three different update schedules, over Y_1. P1 has parameters $a_k = 1$, $b_k = 5$, $c_k = 1$, and $\beta = 0.8$. We have also plotted P2 with $a_k = 1$, $b_k = 0$, $c_k = 0$, $\beta = 0.8$.

Let $(i_1, ..., i_m)$ be a sequence of Y vertices. We will investigate in the following the class of sequences which will induce one and the same SDS. According to the above lemma we may transpose the local functions of two vertices i_k, i_{k+1} that are successive in $(i_1, ..., i_m)$ if $d_Y(i_k, i_{k+1}) > 4$. In view of this we introduce for fixed simple, undirected loopfree graph Y the graph Y' as follows:

Definition 2. For given simple, undirected graph Y, let Y' be the graph

$$v[Y'] = v[Y]$$

$$\mathrm{e}[Y'] = \{\{i,\ j\} \mid d_Y(i,\ j) \leq 4\}.$$

We immediately observe that, if Y is a simple, undirected loop-free graph, then Y' is also simple, undirected and loop-free. Let $\vec{i} = (i_1, ..., i_m)$ and $\vec{j} = (j_1, ..., j_m)$.

Definition 3. Let Y be a simple loopfree undirected graph. $\mathcal{W}_m = \mathcal{W}_m(Y)$ is the graph with vertex and edge set

$$\mathrm{v}[\mathcal{W}_m] = \{(i_1, ..., i_m) \mid i_h \in \mathrm{v}[Y]\}$$

$$\mathrm{e}[\mathcal{W}_m] = \{\{\vec{i},\ \vec{j}\} \mid \exists k;\ \{i_k,\ i_{k+1}\} \notin \mathrm{e}[Y],\ i_k = j_{k+1},$$

$$i_{k+1} = j_k,\ i_h = j_h,\ h \neq k,\ k+1\}.$$

We call two vertices $\vec{i},\ \vec{j}$ *equivalent* and write (with explicit reference to the graph Y)

$$\vec{i} = (i_1, ..., i_m) \sim_Y \vec{j} = (j_1, ..., j_m) \tag{5.1}$$

if and only if they belong to the same $\mathcal{W}_m$ -component.

In the particular case of the update schedules that are *permutations*, i.e., words of length n without repetitions let $\mathcal{U}(Y)$ be the induced subgraph of $\mathcal{W}_n$, i.e., two different permutations ($\mathcal{U}(Y)$-vertices) $(i_1, ..., i_n)$, $(h_1, ..., h_n)$ are adjacent iff (a) $i_\ell = h_\ell,\ \ell \neq k,\ k+1$ and (b) $\{i_k, i_{k+1}\} \notin \mathrm{e}[Y]$.

The following Theorem is proved in [8]. To keep the paper self-contained we provide a full proof which requires some familiarity with *acyclic orientations*, the *directed graphs* induced by them and graph *independence sets*.

Theorem 1 [8]**.** *Let X be a simple, undirected, loop-free graph. Then we have the bijection*

$$f_X : [\mathsf{S}_n/\sim_X] \to \mathsf{Acyc}(X). \tag{5.2}$$

Proof. We first claim that f_X is well defined. To prove this it suffices (using induction on the length of a $\mathcal{U}(X)$-path) to consider two

permutations π, π' that are adjacent in $\mathcal{U}(X)$. By definition the latter differ only by the transposition of two consecutive coordinates, $\{i_r, i_s\}$ that are not extremities of a X-edge, whence

$$f_X([\pi]) = f_X([\pi']).$$

Therefore we have a well-defined mapping

$$f_X : [\mathsf{S}_n/\sim_X] \to \mathsf{Acyc}(X)$$

and it remains to show that f_X is bijective. Let $\mathfrak{O}_X$ be an acyclic orientation of X. Then $\mathfrak{O}_X$ can be represented as a tuple of X-independence sets $(I_1, ..., I_k)$, where I_1 is the set of all $\mathfrak{O}_X$-sources and I_h is the set of X-vertices that have the maximum distance h in an $\mathfrak{O}_X$-directed path from a vertex contained in I_1. Suppose $I_h = \{i_h^{(1)}, ..., i_h^{(\ell_h)}\}$, such that $i_h^{(1)} < \cdots < i_h^{(\ell_h)}$, $h = 1, ..., k$. It is straightforward to verify that

$$g_X : \mathrm{Acyc}(X) \to [\mathsf{S}_n/\sim_X] \quad g_X(\mathfrak{O}_X) = [(i_1^{(1)}, ..., i_1^{(\ell_1)}, ..., i_k^{(1)}, ..., i_k^{(\ell_k)})]_X \quad (5.3)$$

is a well-defined mapping with the properties

$$g(X,) \circ f(X,) = \mathrm{id}, \quad f(X,) \circ g(X,) = \mathrm{id},$$

whence Theorem 1.

Our next result, Theorem 2, deals with a different type of phase space relation. Here we have not identical SDS anymore but two SDS with *isomorphic* phase space digraphs. The key feature for the validity of eq. (5.4) below is the fact that our local functions $F_{k,Y}$ update the states of the $\mathsf{S}_{1,Y}(k)$-vertices *in parallel*. This fact allows for the action of graph automorphisms which will typically permute $\mathsf{S}_{1,Y}(k)$-vertices. Not all automorphisms will be considered, though: an additional constraint arises from the fact that the α mappings depend on the path to the fixed destination, D. Accordingly, we restrict ourselves to the subgroup $\mathsf{Aut}_D(Y)$, consisting of all elements that *fix* D.

Theorem 2. *The automorphism group* $\mathsf{Aut}(Y)$ *acts naturally on the states* $(q_{k,i})_{k,i}$ *via*

$$\forall \gamma \in \mathsf{Aut}(Y); \quad \gamma(q_{k,i}) = (q_{\gamma^{-1}(k),\, \gamma^{-1}(i)}). \tag{5.4}$$

Let $\mathsf{Aut}_D(Y)$ *be the subgroup of* $\mathsf{Aut}(Y)$ *consisting of all elements that fix the vertex D. Then we have*

$$\forall \gamma \in \mathsf{Aut}_D(Y); \quad \gamma \circ F_{k,Y} \circ \gamma^{-1} = F_{\gamma(k),Y}, \quad \gamma \circ F'_{k,Y} \circ \gamma^{-1} = F'_{\gamma(k),Y} \tag{5.5}$$

and accordingly

$$\forall \gamma \in \mathsf{Aut}_D(Y), \sigma \in \mathsf{S}_n; \quad [\mathfrak{F}_Y, \sigma] \sim [\mathfrak{F}_Y, \gamma \circ \sigma] \text{ and } [\mathfrak{F}'_Y, \sigma] \sim [\mathfrak{F}'_Y, \gamma \circ \sigma]. \tag{5.6}$$

Proof. In order to compute the RHS of eq. (5.5) we set

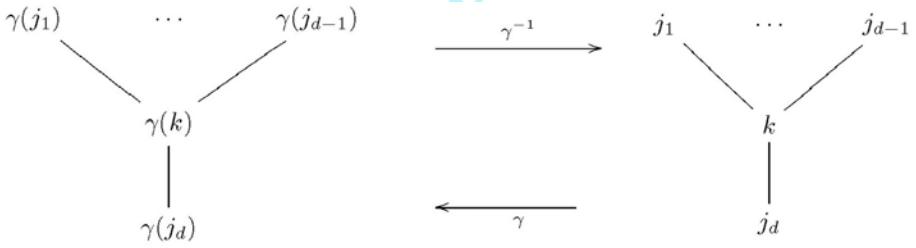
$$\vec{q}_{\gamma(j_1)} = (q_{\gamma(j_1),\gamma(h)})_h$$

$$\vec{q}_{\gamma(j_{d-1})} = (q_{\gamma(j_{d-1}),\gamma(h')})_{h'}$$

$$\vec{q}_{\gamma(j_d)} = (q_{\gamma(j_d),\gamma(h'')})_{h''}$$

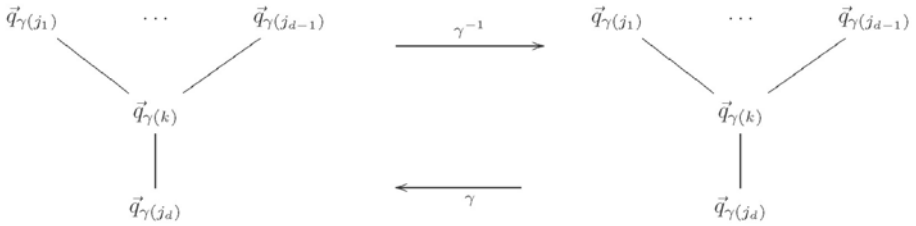
$$\vec{q}_{\gamma(k)} = (q_{\gamma(k),\gamma(j_1)}, \ldots, q_{\gamma(k),\gamma(j_{d-1})}, q_{\gamma(k),\gamma(j_d)}).$$

Since $\gamma \in \mathsf{Aut}(Y)$, we have the following situation



(5.7)

Now $\gamma^{-1}(q_{k,i}) = (q_{\gamma^{-1}(k),\gamma^1(i)})$ implies that the state values of the vertices in $\mathsf{Star}_Y(\gamma(k))$ are mapped into the state values of the vertices contained in $\mathsf{Star}_Y(k)$, as follows:



(5.8)

According to eqs. (3.16) and (3.17) the local mappings $F_{k,Y}$ and $F'_{k,Y}$ are given by

$$(F_{k,Y}(q_{h,s}))_{(h_0,s_0)}$$

$$= \begin{cases} q_{k,i} - \xi_i & \text{for } h_0 = k,\ s_0 = i \in \mathsf{S}_1(k) \\ q_{i,j} + \alpha_i(\xi_i,\ t,\ (\tilde{q}_a)_{a\in\mathsf{S}_1(i)})_j & \text{for } h_0 = i \in \mathsf{S}_1(k),\ s_0 = j \in \mathsf{S}_1(i). \end{cases}$$

$$(F'_{k,Y}(q_{h,s}))_{(h_0,s_0)}$$

$$= \begin{cases} q_{k,i} - \xi_i & \text{for } h_0 = k,\ s_0 = i \in \mathsf{S}_1(k) \\ \alpha_i\left(\xi_i + \sum_{j\in\mathsf{S}_1(i)} q_{i,j},\ 0,\ (\tilde{q}_a)_{a\in\mathsf{S}_1(i)}\right)_j & \text{for } h_0 = i \in \mathsf{S}_1(k),\ s_0 = j \in \mathsf{S}_1(i). \end{cases}$$

By construction the mapping α_s depends on (a) the states of the vertices contained in $\mathsf{S}_1(s)$ (eq. 2.7) and (b) the length of paths from s to the destination D. By assumption, $\gamma \in \mathsf{Aut}_D(Y)$ fixes the vertex D and hence there exists a one-to-one correspondence between the paths from s to D and the paths from $\gamma(s)$ to $\gamma(D) = D$. Furthermore we observe

$$a \in \mathsf{S}_1(k) \cap \mathsf{S}_1(s) \Leftrightarrow \gamma(a) \in \mathsf{S}_1(\gamma(k)) \cap \mathsf{S}_1(\gamma(s)).$$

In view of eq. (5.8) we can conclude

$$\forall \gamma \in \mathsf{Aut}_D(Y);\ s \in \{j_1, \ldots, j_d\};\quad F_{\gamma(k),Y} = \gamma \circ F_{k,Y} \circ \gamma^{-1},$$

$$F'_{\gamma(k),Y} = \gamma \circ F'_{k,Y} \circ \gamma^{-1},$$

whence Theorem 2.

Acknowledgements

The authors want to thank the group CCS-5 for stimulating discussions. In particular we want to thank Stephan Eidenbenz and Venkatesh Ramaswamy for their help and feedback.

References

[1] R. R. Brooks, C. Griffin and T. A. Payne, A cellular automata model can quickly approximate UDP and TCP network traffic, Complexity 9 (2004), 32-40.

[2] H. Hassanein and A. Zhou, Routing with load balancing in wireless ad hoc networks, Proc. 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2001, pp. 89-96.

[3] D. B. Johnson and D. A. Maltz, Dynamic source routing in ad hoc wireless networks, Mobile Computing 5 (1996), 153-181.

[4] A. V. S. Kumar, M. V. Marathe, S. Parthasarathy and A. Srinivasan, Algorithmic aspects of capacity in wireless networks, Proc. ACM Sigmetrics, 2005, pp. 133-144.

[5] S.-J. Lee and M. Gerla, Dynamic load-aware routing in ad hoc networks, Proc. IEEE International Conference on Communications, 2005, pp. 3206-3210.

[6] H. S. Mortveit and C. M. Reidys, Discrete, sequential dynamical systems, Discrete Math. 226 (2001), 281-295.

[7] C. E. Perkins and E. M. Royer, Ad hoc on-demand distance vector routing, Proc. of 2nd IEEE Workshop on Mobile Computing Systems and Applications, 1999, pp. 90-100.

[8] C. M. Reidys, Acyclic orientations of random graphs, Adv. Appl. Math. 21 (1998), 181-192.

[9] C. M. Reidys, Sequential dynamical systems over words, Annals of Combinatorics (2004), in press.

[10] C. M. Reidys, Combinatorics of sequential dynamical systems, Discrete Math. (2005), submitted.

[11] C. M. Reidys, On certain morphisms of sequential dynamical systems, Discrete Math. 296 (2005), 245-257.

[12] J. Song, V. Wong and V. Leung, Load-aware on-demand routing protocol for mobile ad hoc networks, Proc. IEEE Vehicular Technology Conference, 2003, pp. 1753-1757.

[13] C.-K. Toh, Associativity-based routing for ad-hoc mobile networks, Wireless Personal Communications, 1997, pp. 103-139.

[14] M. Transier, H. Fuessler, M. Mauve, J. Widmer and W. Effelsberg, Dynamic load balancing for position-based routing, CoNEXT'05, October 24-27, 2005.

[15] K. Wu and J. Harms, Load-sensitive routing for mobile ad hoc networks, Proc. IEEE International Conference on Computer Communications and Networks, 2001, pp. 540-546.

Los Alamos National Laboratory
CCS-5, 87545 New Mexico, U. S. A.
e-mail: duck@santafe.edu