Technical Reports on Mathematical and Computing Sciences: TR-C170
title: On Routing in Update Networks
author: Crasmaru Marce
affiliation:
Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
Meguro-ku Ookayama, Tokyo 152-8552, Japan (`marcel@is.titech.ac.jp`)

Abstract: An update game approach is a new model for studying networks and infinite duration processes. In this paper we analyze the routing in (bipartite) update networks. First we show how to simplify update networks. Second we prove that strategies independent of the relying graph have to use at least logarithmic space w.r.t. number of some branching vertices. Third we prove that playing randomly is an independent of the relying graph routing strategy for update networks.

1 Introduction

Any network can be represented as a finite directed graph, nodes standing for a type of a system (processors, servers, etc.), and edges standing for communication channels. When process of communication in a network is seen as a infinite duration process, proving robustness of the network is a requirement. Robustness of a network is usually related to liveness and fairness properties. While liveness means that all nodes are actively participating in the collective system, fairness means that critical information of dynamic network is distributed to all of the nodes.

A convenient mathematical model for analyzing fairness and liveness in networks was introduced in [MDBK2000], by seeing the communication process in a given network as a two player game where one player, called Survivor, does try to ensure that fairness and liveness for the given network hold and the other player, called Adversary, does not.

Definition 1.1 *An update (bipartite) game is a bipartite graph $G = \{V, E\}$, with partite sets S and A. There are two players, called Survivor and Adversary, which control vertices S and A, respectively. We stipulate that the in and the out degree of every vertex is nonzero.*

The game rules allow moves along edges of G. A play of an infinite duration game is a sequence of vertices $v_0, v_1, \ldots, v_i, \ldots$ such that the game rules are followed. Since the number of nodes is finite, some of the nodes, called persistent nodes, have to appear infinitely often in the play. A finite prefix sequence of a play is called a history. A vertex v is visited if it occurs in the play. Either Survivor or adversary may begin the play. Survivor wins a play if the persistent vertices of the play is the set V, otherwise Adversary wins. A *strategy* for Survivor is a function H from play histories $v_0, \ldots, v_i$, where $v_i \in S$, to the set A such that $(v_i, H(v_0, \ldots, v_i))$ is an edge. Adversary strategies are similarly defined.

Given H a strategy for a player, H may either win or lose the game when starting at an *initial vertex* $v_0 \in V$. A player's *winning strategy* for an initial vertex is one that wins no matter how the other player moves.

Definition 1.2 *An update game is an update network if Survivor has a winning strategy for every initial vertex.*

We will call Survivor winning strategies as routing strategies.

While deciding whether an update game is an update network is computationally easy (a $O(|V| \cdot |E|)$ algorithm was given in [MDBK2002]), designing Survivor winning strategies is still a challenge. Some simple routing strategies in update networks were analyzed first in [MDBK2000]. These strategies have the advantage of being independent of the relying digraph, but they require linear space w.r.t. number of branching Survivor vertices. A pleasant result would be to show that there exist a strategy independent of the relying digraph which requires constant space w.r.t. number of branching Survivor vertices. However, we will show that there are update networks which can not be updated by any strategy independent of the relying digraph using less than logarithmic space w.r.t. number of branching Survivor vertices.

In the real world the requirements for a routing strategy are usually severe. For example IP protocol, widely used by the Internet, assumes that routing is stateless and (almost) independent to the underlying network topology. In what follows, we show that update networks do have a very weak routing strategy which eventually updates all vertices, but requires the existence of a source of truly random bits. More precisely, in any update network, Survivor wins by playing randomly his moves.

A direct consequence of the above fact is that, unless efficiency of routing is a must, a random routing is sufficient in an update network. However, sources of truly random bits are hard to be supplied by computers and are very unlikely to be shareable over a network, pseudo-random generators being used instead. We are still researching to find the necessary conditions which a pseudo-random generator has to satisfy in order to be used instead of a truly random source of bits for our problem.

2 Simplifying Update Networks

In the following we will show that given an update game on graph $G = \{V, E\}$, it can be reduced in polynomial time to an update game $G^* = \{V^*, E^*\}$ having vertices of in and out degrees at most 2 such that G is an update network iff G^* is an update network.

First we modify $G = \{V, E\}$ by replacing any vertex $u \in A$ of out degree $|Out(u)| = |\{v_1, v_2, v_3\}| = 3$ with a new subgraph as shown in left part of Figure 1.

Let $G^1 = \{V^1 = S^1 \cup A^1, E^1\}$ be the obtained update game. Suppose that G is update network. Then exists a routing strategy H such that all vertices V appear infinitely often in a play, no matter what the Adversary is doing. Then Survivor plays in G^1 against adversary, by simulating in parallel a game on G. Let v be the current vertex in game G^1 and w be the current vertex in game G· Let H_1 be the Survivor strategy that follows.

1. if $v = w$, then play in G and G^1 the move given by H, if $v \in S$, otherwise the move of Adversary.

2. if $v \neq w$, $v \notin S$, and $w \in A - A^1$, then play a move in G^1, and do not move in G.

3. if $v \neq w$, $v \in S$, and $w \in A - A^1$, then play in G the move (w, v), and do not move in G^1.

Because by playing in G^1 following strategy H_1 we simply model a play in G following routing strategy H, the vertices $V \cap V^1$ have to appear infinitely often in the play. However, H_1 do not

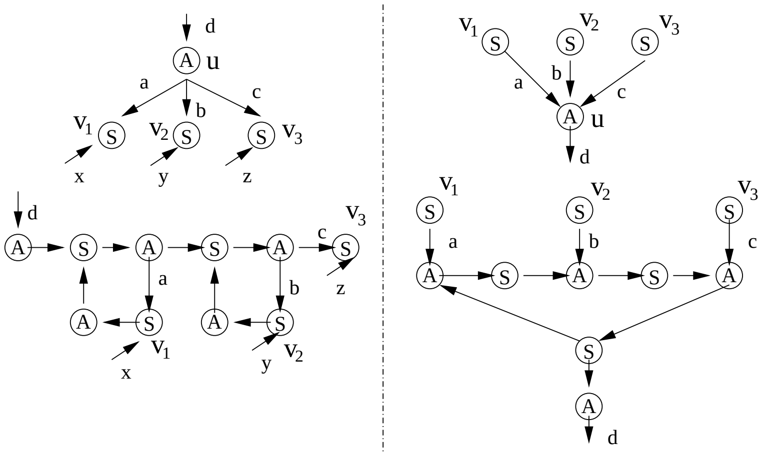

Figure 1: Reduction of an Adversary node

update all the vertices V^1. By observing that from set $V \cap V^1$ we can update all vertices V^1 in game G^1, we can easily modify H_1 to a strategy H_2 which updates all vertices V^1. We have then that G^1 is an update network.

Conversely, suppose that G^1 is an update network. Observe that substituting vertices belonging to $V^1 - V$ with the correspondingly replaced ones from $V - V^1$ in any play of G^1 creates a play in G. Then a winning strategy in G^1 can be easily modified to a winning strategy in G, following that G is an update network.

We can apply the above reduction on vertices $u \in A$ of out degree over 3 inductively, obtaining an update game such that all Adversary vertices do have an out degree at most 2.

Is it easy to check that substitution shown in the right part of Figure 1 reduces further to an update game where in degree of Adversary vertices is at most 2.

The vertices of Survivor of out and in degree over 3 can be substituted by subgraphs shown in Figure 2. It is easy to prove that the above reductions, when applied in the order given above, creates an update game $G^* = \{V^*, E^*\}$ which is an update network iff the initial game $G = \{V, E\}$ is. Moreover, all vertices do have an in and an out degree of at most 2.

Lemma 2.1 *Any update game on graph $G = \{V, E\}$, can be reduced in polynomial time to an update game on a graph $G^* = \{V^*, E^*\}$ such that the following properties hold.*

1. *$|In(v)|, |Out(v)| \leq 2$ for all $v \in V^*$;*
2. *G is update network iff G^* is update network.*

3 A Space Lower Bound For Independent Routing Strategies

By Lemma 2.1, we can focus only on update games played on graphs having vertices of out degree at most 2.

Let M be a Log-space like Deterministic Turing Machine having the following properties. M has two read-only input tapes and a working tape. As input it takes the encoding of an update network G and an initial vertex v on first read only tape. On the second read only tape

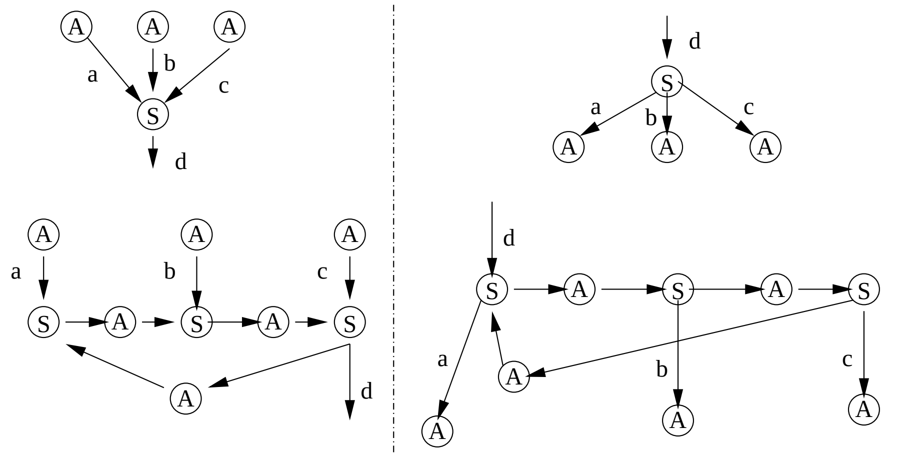

Figure 2: Reduction of a Survivor node

lies an advice string, i.e., an infinite binary sequence (a_i), $i > 0$. M performs an infinite walk on G beginning from v, by providing the moves for Survivor vertices of out degree 2 and using a_i as the ith move on Adversary vertices of out-degree 2. During its computation M uses at most $\lfloor \log(|S_G|) \rfloor$ bites of its working tape, where S_G is the set of branching Survivor nodes. Moreover, we impose on M the condition that read/write head to be leftmost positioned and the inner state of M to be the same when M starts to computes a move for Survivor vertices of out degree 2. That is, the only passed information between moves lies on the working tape.

Suppose that for every update network G, every vertex v of G, and every advice string (a_i), all nodes of G are visited an unbounded number of times on walk performed by M on $G, v, (a_i)$.

Consider the family of bi-partite digraphs $G_d = \{S_d \cup A_d, E_d\}$, when $d > 0$, as shown in Figure 3. One can easily check that G_d is an update network, that is, Survivor can visit any vertex infinitely often no matter what Adversary is doing. Let S_{G_d} be the set of branching Survivor vertices, that is, $S_{G_d} = \{v_1^1, v_1^2, v_2^2, \ldots, v_1^d, \ldots, v_{2^{d-1}}^d\}$. By a simple counting one can see that S_{G_d} is of cardinality $2^d - 1$.

One can observe that for every digraph $G_d, d > 0$ and every starting vertex v of G_d, Adversary has only forced moves. Thus the computation of M on input $G_d, v, (a_i)$ is invariant with respect to the advice string (a_i). Thus we will omit (a_i) in what follows.

Consider the walk performed by M starting from input G_d, v. Because v_1^1 appears infinitely often in this walk, we can define $C(d, v, v_1^1)$ as set of all strings appearing infinitely often on the working tape of M, in all configurations of the above walk in which M is to provide a move for vertex v_1^1.

One can observe that are exactly 2^d different paths starting from v_1^1 and finishing to vertices of type a_{i0}, a_{i1}, when $1 \leq i \leq 2^{d-1}$. M being deterministic, from a configurations such that M is to provide a move for vertex v_1^1, exactly one of the above paths is followed.

From $\lfloor \log(|S_{G_d}|) \rfloor = d - 1$, it follows that $|C(d, v, v_1^1)| \leq 2^{d-1}$. We have then $|C(d, v, v_1^1)| \leq 2^{d-1} < 2^d$. We conclude that there is at least one path starting from v_1^1 and finishing to a certain vertex a_{ij}, when $1 \leq i \leq 2^{d-1}, j \in \{0, 1\}$, such that a_{ij}, is not visited infinitely often on walk performed by M starting from input G_d, v. This contradiction proves the following lemma.

Lemma 3.1 *There is no strategy independent of the relying digraph using less than logarithmic space w.r.t. number of branching Survivor vertices.*

Proof:

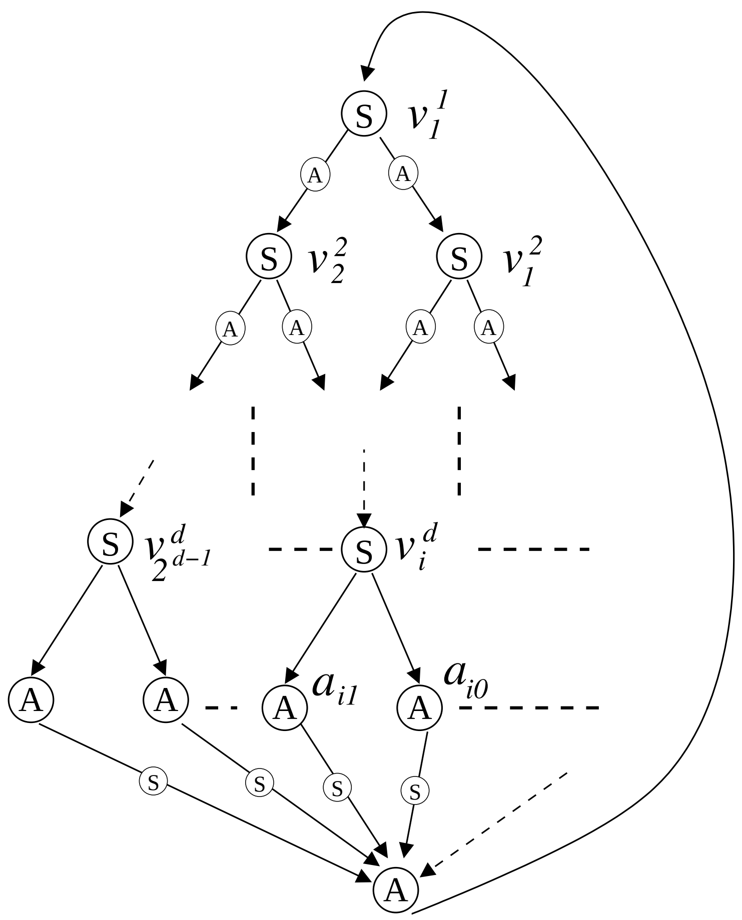

Figure 3: The Family of Update Networks G_d, where $d > 0$

4 The Cycling Neighbor Strategy

In [MDBK2002] two simple deterministic update strategies were given. We will remember here only one of them, the *cyclic neighbor strategy.* This strategy needs only to follow simple local rules without any knowledge of the topology of the network, and its formal definition follows. Each Survivor node s keeps an index i_s between 0 and $d_s = |Out(s)| - 1$, initially set to 1. Survivor's next move from node s is to its i_s-th out-neighbor and i_s is incremented modulo d_s. In [MDBK2002] was proved the following theorem.

Theorem 4.1 *The cycling neighbor strategy is a winning strategy for any update network.*

In the following we use h^{CNS} as a shorthand for the cycling neighbor strategy.

Using Lemma 2.1, we can safely suppose that all update games are played on graphs having vertices of out degree at most 2. Let $G = \{V = S \cup A, E\}$ be an update game and let S_2^G be the set $S_2^G = \{u \in S \mid Out(u) = 2\}$. By the definition of h^{CNS}, there are exactly $2^{|S_2^G|}$ different cycling neighbor strategies for game G, which can be identified as follows. Let σ be a map $\sigma : S \longrightarrow A$. We say that σ is a *signature* for game G iff $(u, \sigma(u)) \in E$ for all $u \in S$. It is clear that are exactly $2^{|S_2^G|}$ different signatures for game G. Given a signature σ_1, and an initial vertex $s_1 \in S$, the cycling neighbor strategy of signature σ_1 and denoted by $h(\sigma_1)$ generates the following play,

$$(s_1, \sigma_1) \stackrel{h(\sigma_1)}{\rightarrow} (a_1 = \sigma_1(s_1), \sigma_2) \stackrel{g}{\rightarrow} (s_2, \sigma_2) \stackrel{h(\sigma_1)}{\rightarrow} \ldots (s_j, \sigma_j) \stackrel{h(\sigma_1)}{\rightarrow} (a_j = \sigma_j(s_j), \sigma_{j+1}) \stackrel{g}{\rightarrow} (s_{j+1}, \sigma_{j+1}) \stackrel{h(\sigma_1)}{\rightarrow} \ldots,$$

where g is an Adversary strategy, and σ_j are signatures verifying

$$\sigma_{j+1}(u) = \begin{cases} \sigma(u) & \text{if } Out(s_j) = 1 \text{ or } u \in S - \{s_j\} \\ v & \text{where } v \in Out(s_j) - \{a_j\} \text{ otherwise,} \end{cases}$$

when $j > 0$.

Lemma 4.2 *Let $G = (V = S \cup A, E)$ be an update network, let σ be a signature, and let $h(\sigma)$ be the cycling neighbor strategy of signature σ. Let $(s_i, \sigma_i), \ldots, (s_{i+k}, \sigma_{i+k})$ be a sequence of a play G generated by the strategy $h(\sigma)$, i.e.,*

$$\ldots (s_i, \sigma_i) \stackrel{h(\sigma)}{\rightarrow} (a_i = \sigma_i(s_i), \sigma_{i+1}) \stackrel{g}{\rightarrow} (s_{i+1}, \sigma_{i+1}) \stackrel{h(\sigma)}{\rightarrow} \ldots (s_j, \sigma_j) \stackrel{h(\sigma)}{\rightarrow} \ldots \stackrel{g}{\rightarrow} (s_{i+k}, \sigma_{i+k}),$$

where $k > 1$ and g is an Adversary strategy. If $s_i = s_{i+k}$, and $\sigma_i = \sigma_{i+k}$, then $V \subseteq \{s_i, \ldots, s_{i+k}\} \cup \{a_i, \cdots, a_{i+k-1}\}$.

Proof: Suppose that there exists $v^* \in V$ such that $v^* \notin \{s_i, \ldots, s_{i+k}\} \cup \{a_i, \cdots, a_{i+k-1}\}$, for a particular Adversary strategy g. Then Adversary has the chance of avoiding node v^*, when Survivor plays accordingly to strategy $h(\sigma)$. More precisely, the Adversary may first play till (s_i, σ_i) appears in the play, then he can cycle forever, as follows,

$$\ldots (s_i, \sigma_i) \stackrel{h(\sigma)}{\rightarrow} \ldots (s_j, \sigma_j) \stackrel{h(\sigma)}{\rightarrow} \ldots \stackrel{g}{\rightarrow} (s_{i+k}, \sigma_{i+k}) = (s_i, \sigma_i) \stackrel{h(\sigma)}{\rightarrow} \ldots (s_j, \sigma_j) \stackrel{h(\sigma)}{\rightarrow} \ldots \stackrel{g}{\rightarrow} (s_i, \sigma_i) \ldots,$$

assuming that Survivor is consistent to its strategy $h(\sigma)$. Then there is a strategy for Adversary such that v^* is not updated infinitely often, contradicting Theorem 3.1.

5 Playing Randomly Is an Update Strategy

In what follows, we prove that the Survivor's strategy given bellow is an update strategy for update networks.

1. In the current vertex s, randomly and independently pick up an outgoing edge e with probability $1/|Out(s)|$, and play accordingly to e.

Let us denote by h^{rand} the above strategy.

First we show that for any given update game $G = \{S \cup A, E\}$, and any cycling neighbor strategy $h(\sigma)$, there is a constant K_G such that after K_G moves, all vertices are visited, for every starting position, no matter how the Adversary plays.

Lemma 5.1 *Given an update game $G = \{S \cup A, E\}$, and a cycling neighbor strategy $h(\sigma)$, there exists an integer K_G such that for all strategies g of the Adversary, the play:*

$$(v_1, \sigma_1) \stackrel{h(\sigma)}{\to} (v_2, \sigma_2) \stackrel{g}{\to} \ldots (v_{K_G}, \sigma_{K_G})$$

verifies that $V \subseteq \{v_1, \ldots, v_{K_G}\}$.

Proof: Let $|S| = m$, $|S_2^G| = n$, and $|A| = a$. By taking $K_G = (m-n+a)\cdot 2^{n+1}+1$, there have to exist two vertices $v_i, v_{i+k} \in S$ in the play of first K_G moves such that $(v_i, \sigma_i) = (v_{i+k}, \sigma_{i+k})$. Proof follows from Lemma 3.2.

Theorem 5.2 *The h^{rand} strategy is a winning strategy for any update network.*

Proof: Let g be any adversary strategy and $G = (V, E)$ an update network. Consider then the play generated by h^{rand} and g partitioned in blocks of length K_G.

$$v_1^1 \stackrel{h^{rand}}{\to} v_2^1 \stackrel{g}{\to} \ldots v_{K_G}^1 \stackrel{g}{\to} v_1^2 \stackrel{h^{rand}}{\to} v_2^2 \stackrel{g}{\to} \ldots v_{K_G}^2 \ldots$$

In the above play there is at least one vertex, say $v^* \in S$, appearing infinitely often such that

$$\ldots v^* = v_1^{i_1} \stackrel{h^{rand}}{\to} v_2^{i_1} \stackrel{g}{\to} \ldots v_{K_G}^{i_1} \stackrel{g}{\to} \ldots v^* = v_1^{i_2} \stackrel{h^{rand}}{\to} \ldots v_{K_G}^{i_2} \stackrel{g}{\to} \ldots v^* = v_1^{i_j} \stackrel{h^{rand}}{\to} \ldots v_{K_G}^{i_j} \stackrel{g}{\to} \ldots$$

S being finite.

Let E_j be the following event in the probability space defined by h^{rand}:

$$E_j = \exists \sigma_j \text{ a signature s.t. the play } v^* = v_1^{i_j} \stackrel{h(\sigma_j)}{\to} v_2^{i_j} \stackrel{g}{\to} \ldots \stackrel{h(\sigma_j)}{\to} v_{K_G}^{i_j} \text{ is a valid play in } G,$$

where $j \geq 1$. Then for all j, k, $0 < j < k$, we have $\Pr(E_j) = \Pr(E_k) = p \geq 2^{|S_2^G|}/2^{K_G}$. Moreover, E_j and E_k are mutually independent. Therefore the sequence E_1, $E_2, \ldots$, is equivalent to a sequence of Bernoulli trials of event E_1.

Let $S(k)$ be defined as the number of successful events E_j, where $0 < j \leq k$. From the above observation and by low of large numbers, we have that $lim_{k \to \infty} S(k)/k = p$. Because $p > 0$, it follows that there is an unbounded number of of successful events $E_{j_1}, E_{j_2}, \ldots, E_{j_q}, \ldots$, therefore infinitely often we will be in the hypothesis of Lemma 4.1. We can conclude that V appears infinitely often in the play generated by h^{rand} and any Adversary strategy g. This finishes the proof.

References

[MDBK2002] M.J. Dineer and B. Khoussainov, Update games and Update Networks, *Journal of discrete Algorithms*, 2(1), 55–68, (2002).

[MDBK2000] M.J. Dineer and B. Khoussainov, Update Networks and Their Routing Strategies, WG2000, *Lecture Notes in Computer Science,* 1928, (2000).