A Game-Theoretic Approach to Routing under Adversarial Conditions[⋆]

James Gross[1], Frank Radmacher[2], and Wolfgang Thomas[2]

[1] Mobile Network Performance Group, RWTH Aachen University
[2] Lehrstuhl für Informatik 7, RWTH Aachen University

Abstract. We present a game-theoretic framework for modeling and solving routing problems in dynamically changing networks. The model covers the aspects of reactivity and non-termination, and it is motivated by quality-of-service provisioning in cognitive radio networks where data transmissions are interfered by primary systems. More precisely, we propose an infinite two-player game where a routing agent has to deliver network packets to their destinations while an adversary produces demands by generating packets and blocking connections. We obtain results on the status of basic problems, by showing principal limitations to solvability of routing requirements and singling out cases with algorithmic solutions.

1 Introduction

An objection to research in theoretical computer science is often the simplicity of the models under consideration in relation to much more complex situations as they arise in practice. In the present paper we attempt to bridge this gap in a specific field of networking which draws much attention currently, namely routing problems over dynamically changing networks. This is motivated by a new system concept in the domain of wireless networking referred to as *cognitive radio* networks.

To illustrate this, consider a wireless communication network (referred to as *cognitive network* in the following) which consists of a certain set of nodes. For any pair of neighboring nodes, there are several radio channels that can be used to convey data packets from one node to the next one. The network is subject to some load, i.e., at different nodes data packets are created at different times which need to be forwarded to a particular destination. Packets are forwarded by a routing scheme, which can take different information into account (channel states, network load) and is therefore reactive. In such a network, dynamic changes of the connectivity between nodes can occur due to interference. Interference occurs if some (potentially malicious) device or multiple such devices start occupying radio channels that are used by the considered cognitive network. Such an action corrupts any data conveyed on that particular channel and blocks therefore the channel as the cognitive radio network can detect interference on radio channels prior to data transmission. The main envisioned application area of such cognitive radio networks is the reuse of allocated radio channels owned by so called *primary systems*. It is well known that at any point in time the radio spectrum is heavily

⋆ Work supported by the RWTH Aachen Research Cluster UMIC ("Ultra High-Speed Mobile Information and Communication") of the German Excellence Initiative, German Research Foundation grant DFG EXC 89.

underutilized. Thus, “cognitive devices” that identify temporarily unused radio channels could solve the problem of underutilized radio spectrum (assuming that they vacate any used radio channels once a primary system starts transmitting data again).

An important question in such a network setting is if – and possibly under which conditions – a cognitive network can provide quality-of-service for transmission requests that it serves; for instance, does the system concept allow for a timely delivery of data packets, how much memory must be put into the nodes of the network to support such guarantees, how do routing schemes have to be designed etc. The key concept in this scenario is the routing of packets which is adversely affected by the way radio channels are interfered over time. As we are interested in worst-case system performance, this naturally leads to a dynamic network model with at least two decision instances: an entity performing the routing as well as an entity that causes interference. In this paper we are interested in theoretical limitations and possibilities of routing in such dynamic networks under various conditions which have implications for the technical design of a cognitive network.

To do so, we propose a game-theoretic model for routing under adversarial conditions. More precisely, we consider a structured scenario in which two adversarial agents perform actions in turn. The first agent, called demand agent, carries out actions that conflict quality-of-service provisioning in the cognitive network. As indicated, primary systems might block certain radio channels (edges) between nodes of the cognitive network, while new data packets are generated that have to be forwarded by the cognitive network to their destination. The other agent, called routing agent, sends packets from nodes to neighboring nodes, observing the fact that certain radio channels are blocked. We assume a time-slotted (discrete) mode of operation in which demand agent and routing agent do their actions in alternation. This results in an infinite system run that is also called “play”, following game-theoretic terminology. Thus, our model includes the aspect of full reactivity (between demands and routing) and of non-termination. Different quality-of-service requirements to be guaranteed by the routing agent are condensed in “winning conditions” – a play that satisfies one such winning condition is considered won by routing agent, otherwise demand agent wins. A routing algorithm that leads to satisfaction of the given requirements under all possible behaviors of the demand agent is thus a winning strategy in this infinite game. Note that a winning strategy works fully adaptively over infinite time, even under radical changes of profiles of the demand player; it is thus a stronger kind of solution than standard routing schemes.

We introduce the model in more detail in the next section. Then we show results on principal limitations of algorithmic solutions. We show that for our model in general, it is algorithmically undecidable whether, given a network and some requirements, a solution (i.e. a winning strategy for routing agent) exists (Section 4). On the other hand we then show that, assuming specific technical requirements for our network model, the existence of a solution can be decided and that in the cases where a solutions exist routing schemes can be effectively constructed (Section 5).

The issue of routing in dynamic networks has been addressed previously in the context of of online algorithms and competitive analysis by Awerbruch, Mansour and Shavit in [4]. In their work – and in a sequel of related ones [1,2,3] – the focus was on the design of online algorithms with the goal of balancing the load in the network to avoid congestion. Note that an online algorithm may be viewed as a strategy in a finite game (possibly of unbounded length), and its performance is measured relatively to a corresponding optimal offline algorithm. This view has two shortcomings. First, such a competitive analysis requires a reasonable comparison of the online algorithm with a corresponding offline algorithm. However, it is inappropriate for the analysis of problems where an online algorithm is only able to find a strict subset of the offline solutions. For example, routing problems can be analyzed via competitive analysis if the number of delivered packets (throughput) is the subject of interest. However, analyzing for which dynamic scenarios certain network properties as the delivery of all packets can be guaranteed (quality-of-service) is out of scope of the competitive analysis approach. The other shortcoming in our context is due to the fact that a network protocol or routing scheme should run without termination. As it is known from the theory of automatic verification (see e.g. [6]), several natural requirements, such as liveness and fairness conditions, can only be modeled faithfully when infinite system runs are considered rather than their approximations by finite runs of unbounded length.

Further related work on game-theoretic analysis of dynamic networks has been started in the studies of "sabotage games", which van Benthem introduced in [5]. There, a reachability problem over graphs is considered, where a "Runner" traverses a graph while a "Blocker" deletes an edges after each move. The theory of these games was developed by Löding and Rohde in [13,12,15,16] and also by others [11,8]. An enhanced non-terminating version of such games was studied in [14]. There, two players, "Constructor" and "Destructor" add resp. delete vertices/edges, and the problem of guaranteeing certain properties of the network graph (like connectivity) is addressed. However, all these approaches do not address the essential issue of simultaneous routing of many packets (which leads to a possibly infinite state space).

Let us finally mention some work on other, complementary aspects of cognitive radio networks. In [7] different solution concepts by equilibria are pursued. In [19] the construction of appropriate network architectures is addressed. Shiang and van der Schaar [17] consider a learning approach for constructing routing schemes adapted to the behavior of the network users.

2 Modeling Routing in Dynamic Networks via Games

We assume that the two agents acting in the network know the current network structure including all information about packets and blocked channels. So, in the present ideal setting we consider a game of perfect information. Moreover, the blocking of frequencies and the generation of packets are subject to certain constraints; these rules may depend on the whole network state. We allow only "deterministic" (rather than probabilistic) constraints; randomized packet generation and randomized frequency blocking [11] is not treated in the present paper.

The game and the game arena. A *dynamic network routing game* between two players, called demand agent and routing agent, is given by a tuple $\mathcal{G} = (G, C, W)$ where G is the network graph, C a list of "constraints" for moves of demand agent, and W the requirement (winning condition) to be fulfilled by the routing agent. More precisely, the *network graph* or *connectivity graph* G is a graph of the form $G = (V, E)$ with finite vertex set V and a finite set E of multi-edges (network connections, where single edges correspond to frequencies). Formally, E is partitioned into sets E_a of single edges, where $a \in \Sigma$ for some index set Σ which denotes the set of available frequencies. We write $(u, v)_a$ for the edge (u, v) in E_a. For convenience we consider in the following all edges as undirected, i.e. $(u, v)_a \in E$ implies $(v, u)_a \in E$. (However, all results in this paper hold exactly in the same way for network graphs with directed edges.) C is a list of rules, called also *constraints*, i.e. conditions imposed on edge removal and packet generation. Finally, W is the *winning condition*, formally a set of infinite plays, and containing precisely those plays that are won by routing agent. We now explain in several stages the notion of play (or admissible system run consisting of network states).

Packets and blocked frequencies. A *packet* consists of a unique identifier from $\mathbb{N}$, its destination node, and a *timestamp*, which is the number of turns since its creation. Thus a packet is a triple $(id, u, k) \in \mathbb{N} \times V \times \mathbb{N}$, indicating that it has the identifier id, the destination u, and that it was generated k turns before the current moment. We define the *packet distribution* $\lambda\colon V \to 2^{\mathbb{N} \times V \times \mathbb{N}}$ by mapping each node to the set of packets which are currently stored at this node.

Network connections (edges) can be blocked (by demand agent, more precisely by the component of primary systems) for a certain number of turns. The current status of the edges is described by a *blocked links function* $bl\colon E \to \{0, 1, \ldots, m\}$ which says that edge e is blocked for the next $bl(e)$ turns. If $bl(e) = 0$, the edge e is not blocked. Communication between two nodes via the edge e is only possible if $bl(e) = 0$.

The maximal number of turns m that can be assigned to an edge for blocking is always given by the constraints C, which are described later. We denote the set of all possible functions bl for a game $\mathcal{G}$ (i.e. E and m are fixed) by $BL_{\mathcal{G}}$ or simply by BL when the context is clear.

Network states and plays. The positions or states of a dynamic network routing game $\mathcal{G}$ are called network states. A *network state* is a triple $(0/1, \lambda_i, bl_i)$ where 0 (resp. 1) indicates that routing agent (resp. demand agent) moves next, λ_i is a packet distribution, and bl_i is a blocked links function. We denote by $Q_{\mathcal{G}}$ the set of all network states in the game $\mathcal{G}$; note that $Q_{\mathcal{G}}$ can be infinite in general, since we do not impose an a priori bound on the number of packets in the network.

The initial network state is $(1, \lambda_0, bl_0)$ with $\lambda_0(u) = \emptyset$ and $bl_0(e) = 0$ for all $u \in V$, $e \in E$, i.e. no packets are in the network, no edges are blocked, and demand agent starts. The subsequent moves are chosen by routing agent and demand agent in alternation. A *turn* is defined as two consecutive movements: the first one by demand agent and the second one by routing agent; each of the player's moves we call a *half-turn*. In the i-th half-turn where i is even, demand agent moves and the network state $(1, \lambda_i, bl_i)$ is updated according to demand agent's action to $(0, \lambda_{i+1}, bl_{i+1})$. In the subsequent half-turn (where $i + 1$ is odd), routing agent acts and generates the network state $(1, \lambda_{i+2}, bl_{i+2})$. A detailed explanation of these updates follows. A *play* is an infinite sequence of network states that is generated in this way, i.e. a sequence $\rho = \rho_0\rho_1\rho_2 \ldots \in Q_{\mathcal{G}}^{\omega}$ with $\rho_0 = (1, \lambda_0, bl_0)$.

Constraints and moves. When it is demand agent's half-turn, he generates new packets and blocks edges for a certain amount of turns. We restrict demand agent's movements that are possible in the game $\mathcal{G} = (G, C, W)$ by a list C of "constraints". An example of a constraint is the following:

$$\text{node } u \text{ has a packet } \wedge \neg \text{ edge } (u,v)_a \text{ is blocked } \longrightarrow \\ \text{block}_2((u,v)_a) \mid \text{block}_1((u,v)_a), \text{ generate}(u,v') \ .$$

This constraint says that, when there exists a packet at node u and the edge $(u,v)_a$ is not blocked, then demand agent either must block the edge $(u,v)_a$ for the next two turns or he has to block the edge $(u,v)_a$ for one turn and generate a packet at node u with destination v'. If C is a list of more than one constraint, they are processed in their given order. Generally, a *constraint* C is a list of rules of the form

$$\text{Condition} \longrightarrow \text{Behavior} \ .$$

Here, the *condition* is a Boolean combination of statements of the following form:

1. edge $(u,v)_a$ is blocked
2. node u has a packet (possibly with destination d and/or timestamp t)

A constraint is called *weak* if all of its conditions only depend on blocked edges, i.e. every condition is a Boolean combination of statements of the first form. Weak constraints reflect the natural assumption that the possibility of demands (either channel blocking or packet generation) should be restricted by information on the currently blocked channels, but not on packets in the cognitive network.

The *behavior* is a disjunction (separated by "|") of conjunctions (separated by ",") of demands, i.e. instructions of the form (1) generate(u,d) and (2) block$_m((u,v)_a)$ The first says that demand agent must generate a packet at node u with destination d. The second says that demand agent has to block the edge $(u,v)_a$ for the next m turns. Notice that also an edge $(u,v)_a$ with $bl_i((u,v)_a) \neq 0$ can be blocked again; in this case $bl((u,v)_a)$ is updated to its new value k according to the rule block$_k(u,v)$.

In order to rule out some very exotic situations, the constraints always impose a uniform bound m on the number of turns for which an edge can be blocked and on the number of packets generated per turn. The blocked links function bl is then a function from E to $\{0,1,\ldots,m\}$ (if no "block" instruction exists in the constraints, we set $m := 0$). A bound for the maximal number of packets generated per turn can be defined by the number of all "generate" instructions in the constraints.

The semantics of the constraints is defined in the natural way: The list of constraints is processed in their given order, and whenever the condition (left hand side) is true (matches the current network state), demand agent has to choose exactly one of the conjunctions of the behavior (right hand side). Then all statements of the chosen conjunction are carried out.

On a more abstract level, the constraints can be seen as a function $C\colon Q_{\mathcal{G}} \to 2^{Q_{\mathcal{G}}}$ which assigns to each network state of demand player a set of possible successor network states that are described by the given list of constraints. Since in weak constraints only conditions of form 1 are used, these constraints depend on the blocked links only. So, on this abstract level, weak constraints can be seen as a function $C\colon BL_{\mathcal{G}} \to 2^{Q_{\mathcal{G}}}$ which assigns to each blocked links function a set of possible successor network states that are described by the given list of constraints.

When it is routing agent's half-turn, she can send packets to neighboring nodes. For each node $v \in V$ and each available frequency $a \in \Sigma$, at most one packet can be transmitted from node v via frequency a. Delivered packets, i.e. packets that reach their destination in this turn, are removed from the network. For all other packets the timestamp is increased by 1. After routing agent's half-turn the value of the blocked links function bl is decremented by one for every edge (if it is not already 0).

Strategies. In this paper we only consider deterministic strategies. A *strategy for demand agent* (resp. *routing agent*) is a function, here denoted by σ (resp. τ) that describes the decisions of the agents (possibly depending on the history of the play). Formally, a strategy for demand (resp. routing) agent is a (partial) function $\sigma\colon Q_{\mathcal{G}}^{+} \to Q_{\mathcal{G}}$ (resp. $\tau\colon Q_{\mathcal{G}}^{+} \to Q_{\mathcal{G}}$) that maps each possible play prefix $\rho_1 \cdots \rho_k$ to a new network state which is permitted by the before mentioned rules.

Winning conditions. The *winning condition* W (for routing agent) describes for each play $\rho \in Q_{\mathcal{G}}^{\omega}$ whether routing agent wins ρ. We consider the following fundamental winning conditions:

- DELIVERY. Routing agent wins a play ρ if in ρ each generated packet is eventually delivered.
- DELIVERY$_\ell$. Routing agent wins a play ρ if in ρ each packet is delivered within ℓ turns after it was generated.
- BOUNDEDNESS. Routing agent wins a play ρ if in ρ the number of packets in the network is bounded, i.e. there is a k such that the number of packets is always $\leq k$.

Demand agent wins a play if it is not won by routing agent. We say that demand agent (resp. routing routing) wins a game $\mathcal{G}$ if there exists a strategy σ (resp. τ) such that he (resp. she) wins every play $\rho \in Q_{\mathcal{G}}^{\omega}$ that is played according to this strategy.

Some basic problems. From the theory of infinite games it is known that in a very abstract view of winning conditions, there are winning conditions that do not allow a winning strategy for either player. Such games are called non-determined. All winning conditions in this paper are of a somewhat concrete and simple kind (called Borel conditions) that leads to games that are *determined*; so one of the two players has a winning strategy. So we do not not address the problem of determinacy in the sequel.

A *solution of a game* consists then of

1. the decision which of the two players wins
2. and then a presentation of a winning strategy for the winner.

We will first show that the first problem is undecidable for the winning conditions BOUNDEDNESS and DELIVERY. On the other hand, for DELIVERY$_\ell$ both problems 1 and 2 will be shown to be solvable. We will then present a variant where we restrict the constraints C to be weak; these games turn out to be solvable also for the DELIVERY winning condition.

3 Toy Example

Consider the tiny network G in Figure 1 with channels over $\Sigma = \{a, b\}$. We define the dynamic network routing game $\mathcal{G} = (G, C, W)$ where demand agent's constraints C are the following. In each turn, demand agent generates at node v_1 two packets with destination v_4. Also he blocks exactly one of the a-labeled edges for one turn; so, exactly one of these edges is blocked every turn. The constraints C are weak and can be formalized as follows:

$$\begin{aligned}
&\text{true} \longrightarrow \text{generate}(v_1, v_4),\ \text{generate}(v_1, v_4)\\
&\text{true} \longrightarrow \text{block}_1(v_1, v_2)_a \mid \text{block}_1(v_1, v_4)_a \mid \text{block}_1(v_2, v_3)_a \mid \text{block}_1(v_3, v_4)_a\ .
\end{aligned}$$

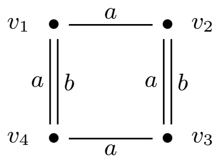

Fig. 1. A network graph of a dynamic network routing game.

First, we analyze the game $\mathcal{G}$ for the DELIVERY winning condition. Routing agent wins the game with the strategy that she sends the packet with the highest timestamp at v_1 to v_4 via the b-labeled edge. This operation is always possible since demand agent cannot block the b-labeled edge in this game. With this strategy, the packet with the highest timestamp always reaches its destination in every turn. So, routing agent wins $\mathcal{G}$ for the DELIVERY winning condition.

Next, we discuss the game with the DELIVERY$_\ell$ winning condition. Routing agent does not win with the above strategy for the DELIVERY winning condition, because by playing this strategy more and more packets have to be kept at v_1. So, routing agent has to route packets via channel a either using the edge $(v_1, v_4)_a$ or the path $v_1v_2v_3v_4$. Now, consider that demand agent blocks the edge $(v_1, v_4)_a$; so, routing agent has to send a packet via the path $v_1v_2v_3v_4$. In this case, demand agent can keep this packet at the nodes v_2 and v_3 by deleting the edge $(v_1, v_2)_a$ if the packet is at v_2 (resp. the edge $(v_3, v_4)_a$ if the packet is at v_3). Such a packet will never be delivered. So, demand agent wins $\mathcal{G}$ with the DELIVERY$_\ell$ winning condition for every ℓ.

Surprisingly, routing agent can win the game for the BOUNDEDNESS winning condition. Her strategy is the following: In every turn routing agent delivers one of the generated packets at v_1 directly via the b-labeled edge. She also delivers the other generated packet via the a-labeled edge to v_4 if this edge is not blocked; otherwise, she sends this packet to v_2. Furthermore, routing agent sends packets at node v_2 always to v_3, and she sends packets at v_3 to v_4 whenever this is possible. It is easy to see that, by playing this strategy, each of the generated packets at v_1 is sent immediately to another node, and that the number of packets at node v_2 (resp. v_3) is at most 1 (resp. 2). So, the number of packets in the network is bounded.

4 Negative Results

Our first result shows that one has to assume a certain coarseness of the constraints in order to enable an algorithmic analysis of dynamic network routing games. For general (or exotic) constraints we obtain undecidability:

Theorem 4.1. *The following problem is undecidable: Given a network routing game with the* BOUNDEDNESS *winning condition, does routing agent have a winning strategy?*

Proof. We show this result by a reduction of the boundedness problem for 2-register machines to games with the BOUNDEDNESS winning condition. The boundedness problem for 2-register machines, which are Turing-complete, is known to be undecidable.

A 2-*register machine* is a program which operations are the modification of two registers X_1, X_2; the allowed operations of these registers are the increment and decrement by 1, and the test whether a particular register is 0. Formally, a 2-register machine has the form $\mathcal{R} = I_1; I_2; \ldots; I_k$ where each I_j is one of the following instructions: $j\colon$ INC(X_i), i.e. increment register X_i by 1, $j\colon$ DEC(X_i), i.e. decrement X_i by 1 if $X_i > 0$, $j\colon$ IF $X_i = 0$ GOTO m, i.e. a conditional jump to instruction m, and $j\colon$ GOTO m, i.e. an unconditional jump to m (with $1 \leq m \leq k$). The last instruction stops the computation: $I_k = k\colon$ STOP.

We construct, given a 2-register machine $\mathcal{R} = I_1; I_2; \ldots; I_k$, a dynamic network routing game $\mathcal{G} = (G, C, \text{BOUNDEDNESS})$. The game arena $G = (V, E)$ has $|k| + 5$ vertices $V = \{v_1, \ldots, v_k, c_1, c_2, c'_1, c'_2, s\}$. Each of the vertices $v_1, \ldots, v_k$ corresponds to an instruction of the register machine. A packet starting on vertex v_1 with destination v_k will move according to the instructions of $\mathcal{R}$. The vertices c_1, c_2 represent the two counters (their values are given by the numbers of packets located at c_1, c_2), and in order to decrement a counter, the vertex s is used as destination for packets from c_1, c_2. The vertex c_1 (resp. c_2) is connected to s via the vertex c'_1 (resp. c'_2) and is used to force routing agent to decrement the number of packets in c_1 (resp. c_2). The construction only uses an edge relation over single edges; it is defined as follows:

$$\begin{aligned} E :=& \{(v_j, v_{j+1}) \mid j\colon \text{IF } X_i = 0 \text{ GOTO } j',\ \text{or } j\colon \text{INC } X_i,\ \text{or } j\colon \text{DEC } X_i\ \in \mathcal{R}\} \\ &\cup \{(v_j, v_{j'}) \mid j\colon\ \text{IF } X_i = 0 \text{ GOTO } j',\ \text{or } j\colon\ \text{GOTO } j'\ \in \mathcal{R}\} \\ &\cup \{(c_1, c'_1), (c_2, c'_2), (c'_1, s), (c'_2, s)\}\ . \end{aligned}$$

When there are no packets in the network – especially in the first turn – demand agent creates a packet with destination v_k at vertex v_1, which mimics the instruction pointer. The constraints ensure that exactly the edge between two vertices v_i and v_j is enabled when the packet is at v_i and the instruction which must be executed next in $\mathcal{R}$ corresponds to the vertex v_j. At the vertices c_1 and c_2 packets are generated according to the increment instructions, and the paths to the vertex s are enabled according to the decrement instructions. The vertices c'_1 and c'_2 allow that we can encode in the constraints a check whether routing agent really sends a packet towards s according to a decrement instruction. If the instruction pointer packet reaches its destination v_k, the game switches in a mode where all remaining packets in the network will be delivered.

With some work on the exact formulation of the constraints one can see, that with such a construction routing agent wins $\mathcal{G}$ iff the register machine $\mathcal{R}$ is bounded. □

Theorem 4.2. *The following problem is undecidable: Given a network routing game with the* DELIVERY *winning condition, does routing agent have a winning strategy?*

Proof. The argument is a one-to-one copy of the previous proof. Here we reduce the halting problem for 2-register machines to games with the DELIVERY winning condition. Namely, with the same construction as above, we see that if a 2-register machine reaches the stop instruction I_k, then all packets are delivered; so, routing agent wins. Conversely, if the 2-register machine does not reach the stop instruction I_k, at least the packet which mimics the instruction pointer does not reach its destination; so, demand agent wins. □

Note that the undecidability results above can be sharpened. They are still valid if we only consider single channel networks (using only one frequency) since only single edges are involved in the constructed network graphs.

The results can be also sharpened regarding the conditions used in the constraints. For the informally given description of the constraints, only statements of the form "edge e blocked" and "node u has a packet" are necessary. So, it is not necessary to check the destination or the timestamp of a packet in the network to obtain undecidability.

5 Positive Results

In this section we show first that dynamic network routing games with the DELIVERY$_\ell$ winning condition are solvable by a reduction to the so-called safety games. Then we show that dynamic network routing games with *weak* constraints are solvable even for the BOUNDEDNESS and the (unrestricted) DELIVERY winning condition.

5.1 Solving Games with the DELIVERY$_\ell$ Winning Condition

Before turning to dynamic network routing games with the DELIVERY$_\ell$ winning condition, we recall the fundamental notion of *safety game* from the theory of infinite games. In the case of network routing games, a *safety winning condition* for routing agent is given by a set A of "admissible network states". Routing agent wins a play $\rho = \rho_0\rho_1\dots$ if each network state ρ_i belongs to A. In other words, she has to avoid getting outside A at some point.

If the set of possible network states is finite, one can compute whether routing agent has a winning strategy (starting from the initial network state) in a safety game specified by the set A and, in this case, one can compute her winning strategy (a constructive proof can be found in [18,9]). We can now easily prove the following result:

Theorem 5.1. *Dynamic network routing games with the* DELIVERY$_\ell$ *winning condition, where $\ell \in \mathbb{N}$, are solvable (so that one can decide whether routing agent wins and in this case provide a winning strategy in terms of a suitable routing scheme).*

Proof. The idea is that the DELIVERY$_\ell$ winning condition ensures that the set of network states of the game $\mathcal{G} = (G, C, \text{DELIVERY}_\ell)$ can be assumed to be finite with a winning condition in the safety format, so that the remark above gives the desired solution.

It remains to be shown that it suffices to inspect only a finite subset of the network states. We can assume that the game is over when the timestamp of a packet exceeds ℓ

(routing agent loses in this case). So, we can assume that it is sufficient to consider packet timestamps of at most $\ell + 1$. Also, the number of packets that can be generated in one turn is bounded by the constraints, say by a constant k. So, the total number of packets in the network is at most $(\ell + 1) \cdot k$. Since each packet gets the lowest available identifier when generated, the identifiers are also bounded by $(\ell + 1) \cdot k$. So, in this case a packet distribution λ is a function from V to $2^{[(\ell+1)k] \times V \times [\ell+1]}$ where $[n] := \{0, \ldots, n\}$. The number of different functions of this form is finite. $\square$

5.2 Solving Games under Weak Constraints

We exhibit another natural scenario under which the dynamic network routing game becomes solvable, even for the conditions BOUNDEDNESS and DELIVERY.

This scenario is given by a certain format of constraints of the demand player, taking into account the division of demands into those by the primary systems and the cognitive network. A natural assumption on the constraints is that the demand actions may depend on blocked frequencies (i.e. currently active demands of the primary systems) but not on information about packets that are currently in the cognitive network. This leads to the assumption that constraints depend on the information about blocked channels only.

We already defined these *weak constraints* in Section 2; they can be seen as a function $C\colon BL \to 2^{Q_\mathcal{G}}$ which assigns to each blocked links function a set of possible successor network states. We shall show, in contrast to the results of Section 4, the solvability of games with weak constraints. The central observation will be that – assuming weak constraints – inspecting only finitely many network states is sufficient to decide DELIVERY resp. BOUNDEDNESS. As a preparation we state some auxiliary propositions.

Remark 5.2. In a dynamic network routing game with weak constraints, consider a play ρ that is won by routing agent and results from demand agent playing according to a strategy σ and routing agent playing according to a strategy τ. If demand agent changes his strategy σ to σ' by leaving out the generation of some packets, he will also lose the resulting play, i.e. demand agent cannot improve his strategy in this way.

Remark 5.3. Consider a dynamic network routing game with weak constraints, a play ρ which is currently in a network state q_i with blocked links function bl_i. Let us assume that demand agent has a strategy to reach from q_i a network state q_j with blocked links function bl_j. Then, since the constraints do not depend on the packet distribution, demand agent has a strategy from q_i to reach a network state q'_j with same blocked links function bl_j within at most $|BL| - 1$ turns (note that $|BL|$ is always ≥ 1).

For the proofs in this section, we use the following terminology. We say that a packet (id, d, k) which is currently at vertex u (in a given network state) has the *type* (u, d). So, in a network with a vertex set V the packets have at most $|V|^2$ different types.

Further, for a given game, we denote with $\varDelta_{\text{out}}$ the *maximal number of outgoing packets* that routing agent can send per node; formally, it is defined as

$$\varDelta_{\text{out}} = \max_{u \in V} \{\ |\{a \in \varSigma\}| \ :\ \text{it exists } v \in V \text{ such that } (u, v) \in E_a \ \}\ .$$

First we show solvability for the BOUNDEDNESS condition under weak constraints. For this winning condition we can give a uniform bound on the number of packets in the network (which is sufficient to achieve for routing agent).

Theorem 5.4. *Consider a network routing game $\mathcal{G}$ with weak constraints. Then, routing agent wins $\mathcal{G}$ with the* BOUNDEDNESS *winning condition iff she can guarantee in $\mathcal{G}$ that there exists at most $b := |BL| \cdot (\Delta_{out} + k)$ packets at each vertex where k is the maximal number of packets that can be generated per turn (given by the constraints).*

Proof. Clearly, if routing agent can guarantee the bound b for the number of packets at each vertex, she can guarantee the bound $|V| \cdot b$ for the total number of packets in the network; hence, she wins with the BOUNDEDNESS winning condition.

For the converse, we only sketch the proof. We assume that routing agent wins with the BOUNDEDNESS condition. Since demand agent's moves do not depend on the packet distribution, we can partition the network states $Q_{\mathcal{G}}$ in a set *Inf* for which demand agent has a strategy to visit a network state with the same blocked links function infinitely often and a set *Fin* which are network states whose blocked links function can occur at most once in a play. The sum of packets that can be generated in states in *Fin* in a play can be bounded by the constant $c := |BL| \cdot k$. For the states in *Inf* demand agent can revisit a network state with same blocked links function at least every $|BL|$ turns (see Remark 5.3). Since routing agent can send at most Δ_{out} packets to a neighboring node in each turn, routing agent can keep the number of packets at each vertex below $c + |BL| \cdot \Delta_{\text{out}}$ turns (with the c above); otherwise demand agent would have a strategy to generate an unbounded number of packets in the network (which would be a contradiction). □

With the previous theorem, we can easily reduce the game to a safety game with finite state space where routing agent has to ensure that there are at most b packets at each network node.

Corollary 5.5. *Dynamic network routing games with weak constraints and winning condition* BOUNDEDNESS *are solvable (so that one can decide whether routing agent wins and in this case provide a winning strategy in terms of a suitable routing scheme).*

Solving network routing games with DELIVERY winning condition under weak constraints requires a more involved proof. We start with some technical lemmata:

Lemma 5.6. *Given a game with weak constraints on a game arena $(\{v_1, \dots, v_k\}, E)$. Assume demand agent can reach a network state with blocked links function bl where v_l stores at least n_{lm} packets with destination v_m ($1 \leq l, m \leq k$). Let $n := \sum_{l,m} n_{lm}$. Then, demand agent can also reach such a state within $|BL|^{n+1} \cdot (\Delta_{out})^n$ turns; moreover it suffices to keep at most $|BL|^n \cdot (\Delta_{out})^{n-1}$ packets of each type for $n > 0$ (and 0 for $n = 0$), i.e. all other packets of each type may be discarded after each turn.*

Proof. We show the claim by induction over n. The case $n = 0$ is easy; demand agent has to reach a network state with blocked links function bl, which he can reach within $|BL| - 1$ turns according to Remark 5.3. For $n > 0$ demand agent has to generate at least n packets, say $P_1, \dots, P_n$, and thereafter demand agent has to reach a network state q_j with $bl_j = bl$ such that the packets $P_1, \dots, P_n$ (or equivalently n packets of the same types) remain at their vertices where there were generated. We distinguish two cases: In the first case, demand agent has a strategy to generate each packet P_i ($1 \leq i \leq n$), say in a network state q_i, without visiting a network state with blocked links function bl_i twice, and after generating all n packets in this way he reaches a network state q_j with $bl_j = bl$ where the packets $P_1, \dots, P_n$ still exist at their required vertices v_{lm}. This is the trivial case where demand agent can reach q_j within $(n+1)\cdot(|BL|-1) \leq |BL|^{n+1}\cdot(\mathit{\Delta}_{\text{out}})^n$ turns (and it suffices to keep $(\mathit{\Delta}_{\text{out}}+1)\cdot(n+1)\cdot(|BL|-1)$ packets of each type). In the second case, there exists a packet P_i (in $\{P_1,\ldots,P_n\}$) such that demand agent agent can reach the network state q_j only by revisiting a network state with blocked links function bl_i. We may assume due to our induction hypothesis that there is a strategy for demand agent to reach a network state q'_j with blocked links function bl within $x_{n-1} := |BL|^n\cdot(\mathit{\Delta}_{\text{out}})^{n-1}$ turns where at least the packets $P_1,\ldots,P_{i-1},P_{i+1},\ldots,P_n$ exist at their required vertices. Now, demand agent has a strategy to revisit a network state with blocked link function bl_i; hence, he can generate sufficiently many packets of the same type as P_i, so that at least one of these packets remains at its origin after taking the x_{n-1} turns for reaching a network state q_j with $bl_j = bl$. Since in the worst case routing agent can send at most $\mathit{\Delta}_{\text{out}}$ packets per node and turn, it is sufficient for demand agent to visit a network state with function bl_i at most $x_{n-1}\cdot\mathit{\Delta}_{\text{out}}$ times. For that demand agent needs at most $x_{n-1}\cdot\mathit{\Delta}_{\text{out}}\cdot(|BL|-1)$ turns (due to Remark 5.3). Then, from this state, demand agent needs at most x_{n-1} turns to reach a network state q_j with $bl_j = bl$ and packets $P_1,\ldots,P_n$ at their required vertices v_{lm}. Overall, demand agent can reach q_j within $x_{n-1}\cdot\mathit{\Delta}_{\text{out}}\cdot(|BL|-1)+x_{n-1} = |BL|^{n+1}\cdot(\mathit{\Delta}_{\text{out}})^n$ turns (and keeping $x_{n-1} = |BL|^n\cdot(\mathit{\Delta}_{\text{out}})^{n-1}$ packets of the same type at each vertex suffices). □

Lemma 5.7. *Given a network routing game with weak constraints on a game arena $(\{v_1,\ldots,v_k\},E)$, and given a network state q with blocked links function bl where each vertex v_l stores n_{lm} packets with destination v_m ($1 \leq l,m \leq k$). Assume that demand agent has a strategy such that from network state q routing agent cannot deliver one of the packets that are currently in the network. Then, from a network state q' with the same function bl where each vertex v_l stores only $n'_{lm} = \min\{n_{lm}, |BL|\cdot\mathit{\Delta}_{out}\}$ packets with destination v_m in the network, demand agent can also prevent the delivery of a packet.*

Proof. Towards a contradiction, we assume that demand agent has a strategy in q to prevent the delivery of at least one packet, but that routing agent has a winning strategy τ in state q'. If demand agent can prevent the delivery of a packet forever, he can do so due to reaching a network state q_- with a certain blocked links function bl_- such that from q_- onwards a particular packet, say P_-, will never be delivered. Due to Remark 5.3 demand agent has a strategy σ to reach such a network state from q within $|BL|-1$ turns. Also note it follows from Remark 5.2 that, if routing agent plays his winning strategy τ (which we assumed he has in q') in the network state q, he can guarantee that all of the n_{lm} packets of a type with $n_{lm} \leq |BL|\cdot\mathit{\Delta}_{\text{out}}$ will be delivered and that at least $|BL|\cdot\mathit{\Delta}_{\text{out}}$ of the packets with $n_{lm} > |BL|\cdot\mathit{\Delta}_{\text{out}}$ will be delivered. But since demand agent has a strategy in network state q to prevent the delivery of one of the packets, there is a type with $n_{lm} > |BL|\cdot\mathit{\Delta}_{\text{out}}$ such that one of the n_{lm} packets of this type will never be delivered if demand and routing agent play σ and τ from q. Since there are at least $|BL|\cdot\mathit{\Delta}_{\text{out}}$ many packets of this type in q' as well as in q, routing agent can deliver these packets at best in $\frac{|BL|\cdot\mathit{\Delta}_{\text{out}}}{\mathit{\Delta}_{\text{out}}} = |BL|$ turns. So, if the routing agent plays τ in q', there is still at least one of the n_{lm} packets of this type left at v_l after $|BL|-1$ turns. But according to Remark 5.3 demand agent can reach from q' a state with function bl_- within $|BL|-1$ turns; so, there is still one packet left that will never be delivered, which is a contradiction to our assumption that τ is a winning strategy in q'. □

For a game $\mathcal{G}$, we define the modified game $\mathcal{G}_{\restriction b}$ where at most b packets with the same destination are stored at each node. More precisely, for all vertices u and d, the following happen in $\mathcal{G}_{\restriction b}$ after each player's half-turn: While the number of packets at u with destination d is higher than b, the packet (id, d, t) at u with the highest id is deleted.

Theorem 5.8. *Consider a dynamic network routing game with weak constraints. Let* $b := (|BL| \cdot \varDelta_{out})^{|V|^2 \cdot |BL| \cdot \varDelta_{out}}$. *Then, routing agent wins* $\mathcal{G}$ *with the* DELIVERY *winning condition iff she wins* $\mathcal{G}_{\restriction b}$ *with the* DELIVERY *winning condition.*

Proof. Assume that routing agent wins $\mathcal{G}$. Towards a contradiction, we assume that demand agent wins $\mathcal{G}_{\restriction b}$, say with a strategy σ. We take demand agent's strategy σ for $\mathcal{G}$. Since the constraints C are independent from the packet distribution, routing agent must at least deliver all packets which would not be deleted by the additional rule in the modified game $\mathcal{G}_{\restriction b}$. Since routing agent cannot deliver all packets in $\mathcal{G}_{\restriction b}$, so in $\mathcal{G}$. Hence, demand agent wins $\mathcal{G}$ by playing σ, which is a contradiction to our assumption.

Conversely, assume that routing agent wins $\mathcal{G}_{\restriction b}$. Towards a contradiction, we assume that demand agent wins $\mathcal{G}$. Then demand agent has a strategy (for $\mathcal{G}$) to reach a network state q_- where he can guarantee that one of the packets will never be delivered. Due to Lemma 5.7 it suffices to keep at most $|BL| \cdot \varDelta_{\text{out}}$ packets of each type from the network state q_- onwards. Since the number of different types is bounded by $|V|^2$, there have to be kept at most $n = |V|^2 \cdot |BL| \cdot \varDelta_{\text{out}}$ packets in the network from q_- onwards. According to Lemma 5.6 demand agent has a strategy to reach q_- if only $|BL|^n \cdot (\varDelta_{\text{out}})^{n-1} \leq b$ (and 0 in the case $n = 0$) packets of each type are kept in the network. So, demand agent also wins $\mathcal{G}_{\restriction b}$, which contradicts our assumption. □

So, for a network routing game $\mathcal{G}$ with weak constraints, deciding the restricted game $\mathcal{G}_{\restriction b}$ with bound b on the number of packets is sufficient for deciding the unbounded game $\mathcal{G}$. Although the number of network states of the restricted game is finite, it has not the safety game format as games with the DELIVERY$_\ell$ winning condition. With the following theorem, every restricted game with a DELIVERY winning condition can be turned into a game with DELIVERY$_\ell$ winning condition.

Theorem 5.9. *Given a restricted network routing game* $\mathcal{G}_{\restriction b}$ *with weak constraints, routing agent wins* $\mathcal{G}_{\restriction b}$ *with the* DELIVERY$_\ell$ *winning condition for* $\ell = |BL|^2 \cdot |V|^2 \cdot b$ *if and only if she wins* $\mathcal{G}_{\restriction b}$ *with the* DELIVERY *winning condition.*

Proof. Winning according to the DELIVERY$_\ell$ condition directly implies winning according to the DELIVERY condition. For the converse, we assume that routing agent wins the game $\mathcal{G}_{\restriction b}$ with the DELIVERY winning condition, say with a strategy τ. Since the the number of packets at each node is bounded by $|V| \cdot b$ (because there exist at most $|V|$ different destinations), we can assume that routing agent may delay sending a packet to a neighboring node due to other packets with a higher timestamp at least $|V| \cdot b$ times. Also, we can assume that routing agent has to wait at most $|BL| - 1$ turns until she sends one of the packets of a certain type at a certain node to a neighboring node. Otherwise a network state with the same blocked frequencies function would be reached twice in the meantime (see Remark 5.3), which would imply that there are no new possibilities for routing agent to route one of the packets towards its destination. Finally, we can assume by a similar argument that routing agent sends a packet to the same node at most $|BL| \cdot |V|$ times. Otherwise a packet would visit a node twice while also a network state with the same blocked frequencies function is reached (and this would imply that there are no new possibilities for routing agent to deliver this packet in the reached network state). Altogether, we can assume that routing agent can guarantee by playing his strategy τ for the game with the DELIVERY winning condition implies that each packet is delivered within at most $|V| \cdot b \cdot (|BL| - 1) \cdot |BL| \cdot |V|$ turns (which is less than $|BL|^2 \cdot |V|^2 \cdot b$ turns). Hence, by playing τ routing agent also wins the game $\mathcal{G}_{\restriction b}$ with the DELIVERY$_\ell$ winning condition. □

Now we have all ingredients to solve games with weak constraints. First we transform the game $\mathcal{G}$ in a restricted game $\mathcal{G}_{\restriction b}$ with a bound for the maximal number of packets at a node provided by Theorem 5.8. Then, the previous theorem give us a bound ℓ such that we can solve $\mathcal{G}_{\restriction b}$ with the DELIVERY$_\ell$ winning condition using Theorem 5.1. The bounds b and ℓ are computable and the reduction to a safety game allows the construction of a winning strategy for routing agent if one exists. We obtain the following result:

Corollary 5.10. *Dynamic network routing games with weak constraints and winning condition* DELIVERY *are solvable (so that one can decide whether routing agent wins and in this case provide a winning strategy in terms of a suitable routing scheme).*

6 Conclusion and Perspective

In this paper we introduced a game-theoretic framework for routing problems in a dynamic or adversarial environment that covers the aspects of reactivity and non-termination. We showed some principal results on the solvability of this problem in terms of routing procedures.

In these results, complexity issues and questions on optimization were suppressed. Under the very liberal assumptions on constraints as considered here, reasonable complexity bounds are not conceivable, both for computing a solution (if it exists at all) and for the mere description of the resulting routing scheme.

Let us mention some variants of the game and of possible solutions that allow a more efficient treatment or a refinement of solutions regarding efficiency. More uniformity can be introduced both into the network model and the format of routing algorithms. For example, one might assume that the possibilities of the primary systems for blocking a frequency are globally the same for all edges (and not dependent on any information of the cognitive network). Similarly, one can pursue the idea that the demands by primary systems are best described in a stochastic model and using identical (but stochastic) constraints for different nodes. Also the routing algorithms can be required to be more uniform.

In current work we address also refined solutions that include aspects of optimization. Only then it is possible to compare the performance of truly reactive routing algorithms with solutions in terms of online algorithms as discussed in the introduction. Rather than requiring delivery of packets “eventually” or “with fixed time bounds” it seems more reasonable to search for solutions that simply guarantee the “best possible” time intervals for delivery under the conditions of the considered network game. In [10] we developed a method to compute optimal strategies for a natural setting.

References

1. Afek, Y., Awerbuch, B., Gafni, E., Mansour, Y., Rosén, A., Shavit, N.: Slide – the key to polynomial end-to-end communication. Journal of Algorithms 22(1), 158–186 (1997)
2. Aiello, W., Ostrovsky, R., Kushilevitz, E., Rosén, A.: Dynamic routing on networks with fixed-size buffers. In: Proceedings of SODA. pp. 771–780 (2003)
3. Awerbuch, B., Brinkmann, A., Scheideler, C.: Anycasting in adversarial systems: Routing and admission control. In: Proceedings of ICALP. Lecture Notes in Computer Science, vol. 2719, pp. 1153–1168. Springer (2003)
4. Awerbuch, B., Mansour, Y., Shavit, N.: Polynomial end-to-end communication (extended abstract). In: Proceedings of FOCS. pp. 358–363. IEEE (1989)
5. van Benthem, J.: An essay on sabotage and obstruction. In: Hutter, D., Stephan, W. (eds.) Mechanizing Mathematical Reasoning, Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 2605, pp. 268–276. Springer (2005)
6. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge, MA, USA (1999)
7. Gao, L., Wang, X.: A game approach for multi-channel allocation in multi-hop wireless networks. In: Proceedings of MobiHoc. pp. 303–312. ACM (2008)
8. Gierasimczuk, N., Kurzen, L., Velázquez-Quesada, F.R.: Learning and teaching as a game: A sabotage approach. In: Proceedings of LORI. Lecture Notes in Computer Science, vol. 5834, pp. 119–132. Springer (2009)
9. Grädel, E., Thomas, W., Wilke, T.: Automata, Logics, and Infinite Games, Lecture Notes in Computer Science, vol. 2500. Springer (2002)
10. Horn, F., Thomas, W., Wallmeier, N.: Optimal strategy synthesis in request-response games. In: Proceedings of ATVA. Lecture Notes in Computer Science, vol. 5311, pp. 361–373. Springer (2008)
11. Klein, D., Radmacher, F.G., Thomas, W.: The complexity of reachability in randomized sabotage games. In: Proceedings of FSEN. Lecture Notes in Computer Science, vol. 5961, pp. 162–177. Springer (2009)
12. Löding, C., Rohde, P.: Model checking and satisfiability for sabotage modal logic. In: Proceedings of FSTTCS. Lecture Notes in Computer Science, vol. 2914, pp. 302–313. Springer (2003)
13. Löding, C., Rohde, P.: Solving the sabotage game is PSPACE-hard. In: Proceedings of MFCS. Lecture Notes in Computer Science, vol. 2747, pp. 531–540. Springer (2003)
14. Radmacher, F.G., Thomas, W.: A game theoretic approach to the analysis of dynamic networks. In: Proceedings of VerAS. Electronic Notes in Theoretical Computer Science, vol. 200 (2), pp. 21–37. Elsevier (2008)
15. Rohde, P.: Moving in a crumbling network: The balanced case. In: Proceedings of CSL. Lecture Notes in Computer Science, vol. 3210, pp. 310–324. Springer (2004)
16. Rohde, P.: On Games and Logics over Dynamically Changing Structures. Ph.D. thesis, RWTH Aachen (2005)
17. Shiang, H.P., van der Schaar, M.: Distributed resource management in multihop cognitive radio networks for delay-sensitive transmission. IEEE Transactions on Vehicular Technology 58(2), 941–953 (2009)
18. Thomas, W.: On the synthesis of strategies in infinite games. In: Proceedings of STACS. Lecture Notes in Computer Science, vol. 900, pp. 1–13. Springer (1995)
19. Xin, C., Ma, L., Shen, C.C.: A path-centric channel assignment framework for cognitive radio wireless networks. Mobile Networks and Applications 13(5), 463–476 (2008)