Fundamenta Informaticae 104 (2010) 1–20
IOS Press

Model Checking Optimisation Based Congestion Control Algorithms

Alessio Lomuscio
Department of Computing
Imperial College London
London, UK
a.lomuscio@imperial.ac.uk

Ben Strulo
BT Innovate
Adastral Park, UK
ben.strulo@bt.com

Nigel G. Walker
BT Innovate
Adastral Park, UK
nigel.g.walker@bt.com

Peng Wu
Department of Computer Science
University College London
London, UK
p.wu@cs.ucl.ac.uk

Abstract. Model checking has been widely applied to the verification of network protocols. Alternatively, optimisation based approaches have been proposed to reason about the large scale dynamics of networks, particularly with regard to congestion and rate control protocols such as TCP. This paper intends to provide a first bridge and explore synergies between these two approaches. We consider a series of discrete approximations to the optimisation based congestion control algorithms. Then we use branching time temporal logic to specify formally the convergence criteria for the system dynamics and present results from implementing these algorithms on a state-of-the-art model checker. We report on our experiences in using the abstraction of model checking to capture features of the continuous dynamics typical of optimisation based approaches.

Keywords: Model Checking, Distributed Optimisation, Congestion Control, Convergence

1. Introduction

Model checking has been widely applied to reason about network protocols in terms of the sequences of interactions between protocol entities. This typically allows the discovery of functional problems in network protocols, such as whether a protocol can deadlock or otherwise fail to achieve the desired outcome.

Address for correspondence: Lab of Computer Science, P.O.Box 8718, Beijing 100190, China.

In the case of routing or flow control protocols network-wide properties are studied, such as whether a stable routing configuration can be established, or whether link capacity can be fairly and efficiently shared between communication sources. Optimisation theory has provided a successful approach to this type of questions. This approach abstracts away from the details of packet arrivals and transmissions in a network, and instead considers the rate at which a source sends packets, typically measured as a positive real number. A protocol is then specified as an algorithm which defines how the rate should change in response to feedback from the network.

Our work investigates how model checking can be applied to reason about protocol behaviour at this higher level of abstraction. We seek to explore how logic, nondeterminism and discreteness, implicit in model checking, apply to network resource control problems normally modelled from the point of view of optimisation. In doing this we intend to widen the scope of models of this type of network problem, and investigate their impact on protocol behaviour. We take examples from the area of congestion control, which has been extensively studied in the networking literature [7, 8, 4]. To the best of our knowledge this is the first time congestion control has been analysed from the perspective of model checking. Our approach leads to the following notable features:

- The source and resource agents, instead of concrete protocol entities, are identified in accordance with the duality structure of the underlying optimisation model. The source agents represent communication sources that control primal variables, while the resource agents represent network resources that control dual variables.

- A range of options is presented for composing these agents to define a congestion control algorithm, ranging from fully synchronous to fully asynchronous models and various combinations of them. The expressiveness analysis of these models reveals that fully asynchronous models with both source and resource agents simulate fully synchronous models with source or resource agents only, but not vice versa.

- Nondeterminism can capture aspects not modelled within the optimisation framework, such as uncertain gain (a parameter within the agents) or propagation delay. Through model checking, we find that nondeterminism can affect the stability of congestion control algorithms by introducing different scenarios of convergence or oscillation, and also by increasing the number of states a network may undergo until stable.

The experiments we report here used the model checker NuSMV [2], due to its full support for CTL and LTL, as well as explicit fairness constraints (e.g., to ensure every agent to execute infinitely often). The model checker is able to identify *unanticipated* unstable behaviour, by returning counterexamples to the stability property. We have also tried, but not reported here, other model checkers, such as SPIN [5] and UPPAAL [1], with no better results in other slightly different settings.

Related Work. Following [7] optimisation based approaches are now standard for analysing congestion control. Kelly and Voice [8] proposed a stable fluid-flow framework for joint routing and rate control on which our first case study is based. Walker, Wennink et al. [14, 15] exploited Lagrangian optimisation models, which decompose into distributed synchronous and asynchronous algorithms for congestion control. However, it was argued in [6] that the importance of communication mechanisms has generally been overlooked in the modelling of global behaviour.

Formal verification techniques have been applied to network protocols before. For instance, Yuen and Tjioe [16] applied SPIN to verify the equilibrium property of a priority pricing based congestion control model. Sobeih, Viswanathan et al. [11] presented an extended compositional network simulation environment with the capability of bounded model checking.

Our work inherits the global perspective of optimisation based approaches and characterises stability (a network-wide property) as a temporal logic formula. This makes it different from existing literature on model checking network protocols, where local functionalities of concrete protocol entities such as border gateways and interior routers were the main concern. Specially, our work differs from [16] in that we discretise and analyse a continuous optimisation based congestion control model and explore the issue of nondeterminism in this setting. This contrasts with [16] which is just applicable to the particular model considered.

The work closest to our own is the asynchronous algorithm presented in [10], which was based on optimisation but schedules the sources and resources in the fully nondeterministic order. Jaggard, Ramachandran et al. [6] studied the impact of communication models on network convergence under the synchronous framework. The present paper further explores a range of synchronous and asynchronous compositions systematically.

Structure. The rest of the paper is organised as follows. Section 2 briefly introduces two congestion control protocols. Section 3 presents the modelling spectrum with expressiveness analysis for each composition framework. The stability property is formulated in Section 4, followed by the model checking results. Section 5 discusses the strength and the weakness of both approaches for this optimisation problem. We conclude in Section 6 with some observations.

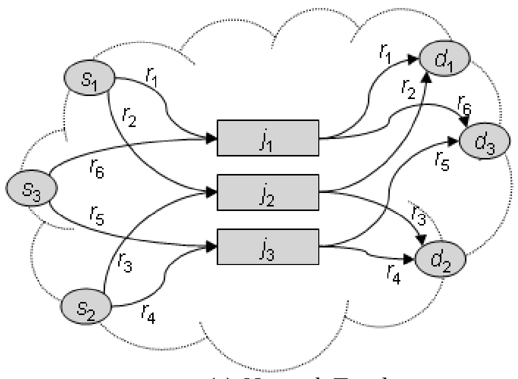
2. Optimisation Based Congestion Control

This section briefly presents an optimisation formulation of a congestion control problem. We imagine a network in which a number of sources communicate with a number of destinations. Between each source and destination a number of routes have been previously provisioned, and a source can split its traffic over these routes. Each route uses a number of links or, more generally, resources, each of which has a finite capacity constraint. We formalise this as follows.

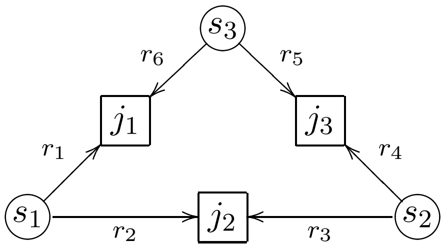
Assume a network with a set S of sources and a set J of resources. Let R be a set of routes, each identifying a non-empty subset of resources. Each route connects only one source with its pre-defined destination. Let $r \in s$ denote that source s can transmit along route r and $s(r)$ be the unique source s such that $r \in s$. For example, in the network shown in Figure 1(a), each source s_i $(1 \leq i \leq 3)$ transmits data to its destination d_i along two routes r_{2i-1} and r_{2i}. So $S = \{s_1, s_2, s_3\}$ and $R = \{r_1, \cdots, r_6\}$. Figure 1(b) presents the resource topology of the network, in which each source is configured with two routes (i.e., $r_{2i-1} \in s_i$ and $r_{2i} \in s_i$ for $1 \leq i \leq 3$), and each resource is shared by two routes (i.e., $j_1 \in r_1, j_1 \in r_6$ and $j_i \in r_{2(i-1)}, j_i \in r_{2i-1}$ for $i = 2, 3$).

Let x_r be the flow rate on route r and C_j be the capacity of resource j. It is convenient to introduce vector notations for the flows and capacity constraints. Let $\vec{x} = (x_r, r \in R)$ and $C = (C_j, j \in J)$. Define the resource matrix A such that, for $j \in J$ and $r \in R$, $A_{jr} = 1$ if $j \in r$ and $A_{jr} = 0$ otherwise.

Let x_j denote the aggregate flow rate at resource j, that is, $x_j = \sum_{\{r|j \in r\}} x_r$. A resource j is *congested* if $x_j > C_j$. A route r is *congested* if some $j \in r$ is congested. The multi-path congestion control



(a) Network Topology



(b) Resource Topology

Figure 1. A Communication Network

problem is to find an assignment of flows $\vec{x}$ which maximises the overall utility of the network such that no resource is congested.

We assume that the utility U_s experienced by each source s depends on the total flow sent over all routes available to it, and that the overall utility of the network can be expressed as a sum of utilities of all the sources. These assumptions are standard in the networking literature, and allow the multi-path congestion control problem to be specified as the following optimisation problem:

$$\max \sum_{s \in S} U_s(\sum_{r \in s} x_r) \text{ subject to } A\vec{x} \leq C, \vec{x} \geq 0 \tag{1}$$

Here, the inequalities apply component-wise on the vectors $\vec{x}$ and C. The utility functions U_s are strictly increasing and concave in their argument.

To assign flows to routes the network must implement a distributed algorithm to solve this problem. This is best understood by moving the constraints into the objective function to obtain the associated Lagrangian relaxation, a typical technique for this type of problem [7, 10, 8]. Each constraint (one for each capacitated resource) gives rise to a Lagrange multiplier y_j. Then, it is a well known result in optimisation theory that problem (1) can be solved by finding a saddle point of the following Lagrangian function:

$$L(\vec{x}, \vec{y}) = \sum_{s \in S} U_s(\sum_{r \in s} x_r) - \vec{y}(A\vec{x} - C) \tag{2}$$

where $\vec{y} = (y_j, j \in J)$ with $\vec{y} \geq 0$ and $\vec{x} \geq 0$. A saddle point $(\vec{x}^*, \vec{y}^*)$ maximises the value of L with respect $\vec{x}$ so that $L(\vec{x}, \vec{y}^*) \leq L(\vec{x}^*, \vec{y}^*)$ for any $\vec{x}$, and minimises it with respect to $\vec{y}$ sot that $L(\vec{x}^*, \vec{y}) \geq L(\vec{x}^*, \vec{y}^*)$ for any $\vec{y}$. If $(\vec{x}^*, \vec{y}^*)$ is a saddle point of L, then $\vec{x}^*$ is an optimal solution to problem (1) [10].

The variables $\vec{x}$ and $\vec{y}$ in (2) are referred to as *primal* and *dual* decision variables, respectively. The dual variables have the interpretation of price, and act as a feedback signal from the network (resources) to the sources indicating incipient congestion. In a distributed setting each source controls the flow variables x_r associated with all the routes r available to it, and the resources are assumed to control the dual variables or prices (recall there is one dual variable y_j associated with each resource j). The sources aim to maximise the value of L given the values chosen for the dual variables by the resources, and the resources aim to minimise the value of L given the flows chosen by the sources. A solution to problem (1) is where the interactions between the sources and resources reach an equilibrium [14, 15], as expressed by the saddle point conditions.

To design a distributed algorithm we must provide more details on the rules by which a source (respectively, resource) dynamically updates its values of primal (respectively, dual) variables. A common approach is to model the update rules as temporal trajectories specified by differential equations, on the assumption that the utility functions U_s are differentiable. A complete analysis would then demonstrate that these trajectories converge to the optimum solution (or saddle point) under appropriate modelling and applicability assumptions [7, 10, 8, 14, 15, 12].

The rest of this section presents two congestion control protocols based on the above multi-path setting. The first protocol (multi-path congestion/rate control) intends to capture the fluid-flow congestion control algorithm specified in [8]. Under the fluid-flow assumption the algorithm was described by differential equations and supported by a proof of stability. We apply a direct discretisation to this model so as to make it tractable for model checking. Depending on the level of granularity, this discretisation may exhibit behaviour that seems unrealistic for the system modelled in [8] (we will discuss this later in Section 4.2). But the process of discretisation leads us to consider the meaning of nondeterminism present in the discrete models, which does not exist in the differential equations.

Our second protocol (session based rerouting and termination) is inspired by this experience and considers a scenario where the assumptions behind the Kelly and Voice's model start to fail and where the assumptions of model checking become more realistic. We choose a system where flow rates are discrete-valued, corresponding to the discrete nature of model checking, and the behaviour of a source is itself naturally nondeterministic.

2.1. Multi-path congestion/rate control

First we consider the dynamics of the primal variables. As presented in [8], for each source s and route $r \in s$, the trajectory in the primal flow rates x_r is modelled as a continuous function of time t, i.e., $x_r(t)$, subject to the following differential equation:

$$\frac{d}{dt}x_r(t) = \kappa_r x_r(t)\left(1 - \frac{y_r(t)}{U'_{s(r)}(x_{s(r)}(t))}\right)^{+}_{x_r(t)} \tag{3}$$

where κ_r is a constant and

- $y_r(t) = \sum_{j \in r} y_j(t)$ is the total cost on route r; $y_j(t)$ is the cost at resource j;
- $x_s(t) = \sum_{r \in s} x_r(t)$ is the aggregate flow rate on all routes available to source s;
- U'_s is the first-order derivative of U_s;
- $(z)^+_x = \min(0, z)$ if $x \leq 0$, otherwise $(z)^+_x = z$.

In [8] propagation delay in a communication network was taken into account by defining y_r and x_s as functions of the past route flow rates. Herein, we omit this consideration and assume propagation delay to be negligible. We will come back to this point in Section 5.

For the dynamics of the dual variables controlled by the resources we choose a simpler model. We assume that the congestion price y_j depends functionally on the instantaneous total flow at resource j: $y_j = Pr_j(x_j)$. This dependency can be thought of as capturing a congestion cost, for example, due to increased packet delay. This cost will perturb the solutions of (1) slightly. However, if the congestion pricing function Pr_j is chosen appropriately it has little impact on the equilibrium configuration of the system, though has the advantage of significantly improving controllability.

Now we discretise this continuous model by making the time and state variables integer-valued under integer arithmetic, and by choosing particularly tractable instances of the generic functions in Equation (3).

When the continuous time t is abstracted into a discrete one, the flow rate function $x_r(t)$ is converted into a series of instantaneous snapshots of x_r. This also applies to $y_r(t)$ and $x_s(t)$. The relation between the current value of x_r and its next value x'_r can be defined uniformly as $x'_r = x_r + \Delta x_r$, where Δx_r is the increment of x_r in one unit time.

Following a rather common choice, we assume U_s to be a logarithmic function of the aggregate flow rate on all routes serving source s, that is, $U_s = \alpha \ln(x_s)$, where α is the utility coefficient. We choose the pricing function Pr_j to be a linear function of the flow rates at resource j, that is,

$$y_j = Pr_j(x_j) = \beta x_j \tag{4}$$

where β is the price coefficient. Then, by following the skeleton of Equation (3), Δx_r is defined as

$$\Delta x_r = \kappa_r x_r \left(1 - \frac{\beta}{\alpha} \sum_{j \in r} x_j \sum_{r' \in s(r)} x_{r'}\right)^{+}_{x_r} \tag{5}$$

Here, κ_r can be regarded as a *gain* coefficient that defines the pace at which route r seeks its equilibrium; while $\dfrac{\beta}{\alpha}$ defines how much flow will follow route r at equilibrium.

Equations (5) and (4) define how the sources and resources act, respectively. The agents in the Kelly and Voice's algorithm can be thought of as acting synchronously though in infinitesimal steps. For a discrete model composition structures that do not constrain those actions to synchrony could also be applicable, as suggested by [10]. We will explore these options in Section 3.

Once we allow this relaxation, then the sources no longer proceed with the deterministic gain implied by κ_r. Although κ_r is constant, those asynchronous models allow the sources to update their flow rates in a nondeterministic order. This then may lead the routes to equilibrate at a variable pace, which is significantly different to the deterministic behaviour specified by Equation (3).

2.2. Session based rerouting and termination

In the second protocol, we consider essentially the same underlying optimisation problem but with the context moved to a regime where a continuous real-valued model is a less reliable fit. Specially, we consider a system where the sources are managing a non-empty set of constant flow rate sessions. Two control actions are provided for a source to resolve its possible congestion status. Firstly, the source may reroute the excess sessions from the congested routes to alternative routes. Rather than considering a deterministic rerouting policy, we will let the model explore how the source may choose the new routes. Secondly, the sources may, though only *in extremis*, terminate those excess sessions. This follows the general ideas specified in the IETF Pre-Congestion Notification (PCN) Working Group, where an architecture for controlling congestion through admission control and flow termination is being defined [4]. For each route, its destination will inform its source of the proportion of congestion at the bottleneck resource (if any), while its source relies on this feedback to decide the proportion of its flow to terminate.

For the second protocol, the primal variable x_r of problem (1) is regarded as the number of sessions on route r, a more naturally discrete value. We take a linear utility function $U_s = \alpha_s x_s$, where α_s is the utility value of a single session of source s. This seems appropriate for a network operator treating all sessions equally.

The congestion control policy of the second protocol is as follows: for each congested route $r \in s$, source s will reroute a certain number of excess sessions from route r to a non-congested route $r' \in s$ (chosen nondeterministically), or terminate them if such r' does not exist. The proportions of sessions to be rerouted or terminated is based on the proportions of excess load at resources, that is, for resource j,

$$y_j = \frac{x_j - C_j}{x_j} \tag{6}$$

Then, for a congested route $r \in s$,

$$\Delta x_r = -x_r \max\{y_j \mid j \in r\} \tag{7}$$

Herein, $\max\{y_j \mid j \in r\}$ is the largest proportion of congestion at a bottleneck resource. For a non-congested route $r' \in s$, $\Delta x_{r'}$ is the aggregate number of sessions rerouted to r' from those congested routes $r \in s$.

As before these equations define how the primal and dual decision variables change over time. The possible composition structures will be explored later. However, observe this model is nondeterministic even in the synchronous case. This is because whenever there is more than one non-congested route available, source s chooses one of them nondeterministically for each congested route $r \in s$.

3. Modelling

Distributed algorithms for optimisation based congestion control differ on whether the evolution is driven by the sources (primals), the resources (duals), or both. Most of these algorithms schedule the sources and resources synchronously along the global continuous time scale [10, 8, 15]. An asynchronous algorithm was presented in [10], scheduling the sources and resources in the fully nondeterministic order. In the rest of this section we will explore the options of these composition structures.

Note that it is the data flows in a network that are concerned with these optimisation models. So, the behaviours of the sources and resources, which control and monitor the data flows, respectively, are prescribed in corresponding congestion control algorithms, but not the detailed hop-by-hop behaviours of particular protocol entities. Thus, we follow the perspective of optimisation-based approaches, that is, to encode the sources and resources as procedural agents.

Technically, we adopt a special form of symbolic transition graph with assignments (STGA) [9], termed as *symbolic assignment graph*, to model the systems above. The explicit input/output constructs in STGA is omitted, due to the fact that shared variables can be relied on for this purpose. Recall that symbolic transition graphs are a basic symbolic semantics for value-passing CCS, π-calculus, and others. Herein, the notion of symbolic assignment graph has the benefit of offering a succinct semantics to describe the systems above, which is amenable to model checking; clearly other formalisms are also possible.

We presuppose the following syntactic categories: Val is a set of values, ranged over by v; Var is a set of variables, ranged over by x; Exp is a set of data expressions over $Val \cup Var$, ranged over by e; BExp is a set of Boolean expressions ranged over by b.

An *assignment* θ has the form $\bar{x} := \bar{e}$, where $\bar{x}$ (respectively $\bar{e}$) represents a list of variables (respectively data expressions), with $\bar{x}$ and $\bar{e}$ having the same length. Assume $Assign_V$ be the set of all assignments to variables in $V \subseteq Var$.

A *valuation* ρ is a total mapping from Var to Val. Applications of ρ onto a data expression e and a Boolean expression b are denoted by $\rho(e)$ and $\rho(b)$ as usual. Especially, we write $\rho \vDash b$ if $\rho(b) = true$. The valuation $\rho\{\bar{x} \mapsto \bar{v}\}$ is same as ρ except mapping $\bar{x}$ to $\bar{v}$. Assume $Eval$ be the set of all valuations.

Definition 3.1. (Symbolic Assignment Graph)
Given a set of variables $V \subseteq Var$, a *symbolic assignment graph* (SAG) is a tuple $M_V = (Q, \mathcal{T}, q^0)$ where

- Q is a set of symbolic states;
- $\mathcal{T} \subseteq Q \times BExp \times Assign_V \times Q$ is a set of symbolic transitions, each $(q, b, \bar{x} := \bar{e}, q') \in \mathcal{T}$ denoted by $q \xrightarrow{b,\bar{x}:=\bar{e}} q'$ with $\{\bar{x}\} \subseteq V$;
- $q^0 \in Q$ is the initial symbolic state.

Informally a symbolic transition $q \xrightarrow{b,\bar{x}:=\bar{e}} q'$ denotes a possible state change of M_V from q to q', under the assumption that the guard b is evaluated to true at state q, and in doing so the values of $\bar{x}$ are changed to the ones of $\bar{e}$ evaluated at state q. The following definition makes this precise.

Definition 3.2. (Labelled Transition Systems)
Given a set of observable labels $L_V = \{\mu_W \mid W \subseteq V\}$ and an initial valuation ρ_0, the concrete semantics of the SAG M_V is a labelled transition system $[\![M_V]\!]_{\rho_0} = (P, T, p^0)$, where

- $P = \{q_\rho \mid q \in Q, \rho \in Eval\}$ is a set of states;
- $T \subseteq P \times L_V \times P$ is the minimal set of transitions given by the following operational rule:

$$\frac{q \xrightarrow{b,\bar{x}:=\bar{e}}, q'}{q_\rho \xrightarrow{\mu_{\{\bar{x}\}}} q'_{\rho\{\bar{x}\mapsto\rho(\bar{e})\}}} \quad \rho \vDash b$$

  Similarly we write $p \xrightarrow{\mu_W} p'$ for each $(p, \mu_W, p') \in T$ with $W \subseteq V$;

- $p^0 \equiv q^0_{\rho_0} \in P$ is the initial state.

To conclude we say that $\omega = \mu_{X_1} \cdots \mu_{X_i} \cdots$ is a *trace* of M_V if there exists a run (i.e., a sequence of transitions) $q^0_{\rho_0} \xrightarrow{\mu_{X_1}} q_{1\rho_1} \cdots q_{(i-1)\rho_{i-1}} \xrightarrow{\mu_{X_i}} q_{i\rho_i} \cdots$ in $[\![M_V]\!]_{\rho_0}$ for some initial valuation ρ_0.

In what follows we explore source and resource agents modelled by symbolic assignment graphs (SAGs). To define the evolution of a system we use the standard notions of synchronous and asynchronous compositions.

Definition 3.3. (Synchronous Composition)
Given two symbolic assignment graphs $M_{V_1} = (Q_1, T_1, q^0_1)$ and $M_{V_2} = (Q_2, T_2, q^0_2)$ with $V_1 \cap V_2 = \emptyset$, $M_{V_1}|M_{V_2}$ is the graph $M_{V_1 \cup V_2} = (Q_1 \times Q_2, T, (q^0_1, q^0_2))$, where T is the minimal set of transitions given by the following composition rule:

$$\frac{q_1 \xrightarrow{b_1, \bar{x}_1 := \bar{e}_1} q'_1 \qquad q_2 \xrightarrow{b_2, \bar{x}_2 := \bar{e}_2} q'_2}{(q_1, q_2) \xrightarrow{b_1 \wedge b_2, (\bar{x}_1, \bar{x}_2 := \bar{e}_1, \bar{e}_2)} (q'_1, q'_2)}$$

Definition 3.4. (Asynchronous Composition)
Given two symbolic assignment graphs $M_{V_1} = (Q_1, T_1, q^0_1)$ and $M_{V_2} = (Q_2, T_2, q^0_2)$ with $V_1 \cap V_2 = \emptyset$, $M_{V_1} \parallel M_{V_2}$ is the graph $M_{V_1 \cup V_2} = (Q_1 \times Q_2, T, (q^0_1, q^0_2))$, where T is the minimal set of transitions given by the following composition rules:

$$\frac{q_1 \xrightarrow{b_1, \bar{x}_1 := \bar{e}_1} q'_1}{(q_1, q_2) \xrightarrow{b_1, \bar{x}_1 := \bar{e}_1} (q'_1, q_2)} \qquad \frac{q_2 \xrightarrow{b_2, \bar{x}_2 := \bar{e}_2} q'_2}{(q_1, q_2) \xrightarrow{b_2, \bar{x}_2 := \bar{e}_2} (q_1, q'_2)}$$

The state defined in our optimisation models is precisely the values of the primal and dual variables: the flow rates $\{x_r \mid r \in R\}$ and congestion costs $\{y_j \mid j \in J\}$. The agents who control this state are sources and resources respectively. We take this structure over directly to our SAG models which contain source agents and resource agents.

In scheduling these agents, three forms of optimisation-based congestion control algorithms exist. One standard form is termed primal algorithms, in which the sources actively adjust their primal variables and the resources only recalculate the values of their dual variables accordingly. The symmetric form of primal algorithms is termed dual algorithms, in which it is the resources that take the active parts. Finally, the other is termed primal/dual algorithms, in which both classes of agents are active. In the rest of the section we model the cases of primal algorithms and primal/dual algorithms; the case of dual algorithms can also be modelled but we do not present it here.

3.1. Congestion control as SAGs - primal algorithms

In the case of primal algorithms the sources are active agents modelled as SAGs, while the resources are modelled by deterministic functions recalculating the congestion costs passively. Thus we are assuming that the resources respond with the up-to-date congestion information quickly on the timescale at which the sources are taking their actions. This does not allow a resource agent to apply more complex algorithms to smooth the congestion information it sends: it has to be a simple function of the current load at the resource.

Let $X_s = \{x_r \mid r \in s\}$ denote the decision variables of source s. In this framework, we model each source s as a SAG M_{X_s} updating its own variables X_s at each transition. We assume that a source updates all of its own variables simultaneously in one transition, which is counted as one source update. For the multi-path congestion/rate control model described in Section 2.1, this leads us to the following SAG: for a source s owning $k \geq 1$ routes $r_1, \ldots, r_k$:

$$M_{X_s} = (\{q\}, \{q \xrightarrow{true,\ x_{r_1}, \cdots, x_{r_k} := x_{r_1} + \Delta x_{r_1}, \cdots, x_{r_k} + \Delta x_{r_k}} q\}, q)$$

where $\Delta x_{r_i} (1 \leq i \leq k)$ is defined by equation (5).

Having defined source updates, we are left with two options to represent the interleaving of all source agents in this framework: by synchronously composition generating a system of Synchronous Sources (SS), or by asynchronous composition generating a system of Asynchronous Sources (AS):

$$SS \triangleq \big|_{s \in S} M_{X_s}$$
$$AS \triangleq \big\|_{s \in S} M_{X_s}$$

These two compositions model different scenarios. Intuitively, SS inherits the synchronous structure of dynamic equations like (3). But as a discrete model, it reflects the situation where propagation delay periods are uniform on every route.

AS constitutes the general case of naturally modelling the concurrent nature of a distributed network, where the sources are monitored in uncertain paces.

3.2. Congestion control as SAGs - primal/dual algorithms

In this subsection we model both sources and resources as SAGs and consider their compositions. Modelling resources agents explicitly can reflect the delayed reaction of the resources to source updates. Recall that resources' decision variables are the dual variables $\{y_j \mid j \in J\}$ in the Lagrangian function (2).

Similarly to the previous section for each resource $j \in J$, we associate a resource agent M_{y_j} modelled as a SAG built on y_j as its state variables. Resource agents may be composed synchronously (respectively, asynchronously), thereby generating systems of Synchronous Resources (SR) (respectively, Asynchronous Resources (AR)).

$$SR \triangleq \big|_{j \in J} M_{y_j}$$
$$AR \triangleq \big\|_{j \in J} M_{y_j}$$

Combining this analysis with the one of the previous subsection, we obtain four general modelling frameworks, in which a particular class of sources agents are asynchronously composed with a particular class of resources agents.

$$SSSR \triangleq SS \parallel SR$$
$$SSAR \triangleq SS \parallel AR$$
$$ASSR \triangleq AS \parallel SR$$
$$ASAR \triangleq AS \parallel AR$$

Moreover, we restrict $ASSR$ models with an alternative scheduling policy between sources and resources. This generates AS^* models, where source and resource updates are arranged in a turn-based mode. In each turn, only sources (or resources) get updated, followed by resources (or sources) getting updated in the next turn.

The modelling options above will in general produce different system evolutions but, as far as the underlying optimisation models are concerned, there is a certain redundancy. For instance, in a primal/dual system, consecutive updates by different sources cause the same state change as that caused by a joint synchronous update, because the sources depend only on the states of the resources, which have not yet changed during the consecutive source updates. Any interleaving of these source updates leads to the same state as the equivalent synchronous update. This redundancy also applies to consecutive resource updates. This opens the possibility of employing state reduction techniques, notably partial order reduction, when checking such a system.

3.3. Expressiveness

In this subsection the expressiveness of each modelling framework is illustrated through trace analysis. This will show that different interleaving structures of these modelling frameworks associate optimisation models with distinct senses of nondeterminism.

With the above observations, we extend the notation μ_W to represent trace fragments. Let $X = \bigcup_{s \in S} X_s$ and $Y = \{y_j \mid j \in J\}$. For some $S' = \{s_1, \cdots, s_l\} \subseteq S, l \geq 1$, let $X' = \bigcup_{s \in S'} X_s$ and $\mu_{X'}$ represent the synchronous updates on X', or any permutation of the sequence of asynchronous updates $\mu_{X_{s_1}} \cdots \mu_{X_{s_l}}$, whichever applicable. Similarly, for some $Y' = \{y_{j_1}, \cdots, y_{j_m}\} \subseteq Y, m \geq 1$, let $\mu_{Y'}$ represent the synchronous updates on Y', or any permutation of the sequence of asynchronous updates $\mu_{\{y_{j_1}\}} \cdots \mu_{\{y_{j_k}\}}$. Let $\mathcal{C}_X$ and $\mathcal{C}_Y$ be the set of compound source and resource update labels, respectively, i.e., $\mathcal{C}_X = \{\mu_{X'} \mid X' \subseteq X$ and for each $s \in S$, $X' \cap X_s = \emptyset$ or $X' \cap X_s = X_s\}$ and $\mathcal{C}_Y = \{\mu_{Y'} \mid Y' \subseteq Y\}$.

Then, the finite traces of these modelling frameworks can be expressed as regular expressions shown in Table 2(a).

Models	Traces
SS	$(\mu_X)^*$
AS	$(\{\mu_{X_s} \mid s \in S\})^*$
$SSSR$	$(\mu_X \mid \mu_Y)^*$
$SSAR$	$(\mu_X \mid \mathcal{C}_Y)^*$
$ASSR$	$(\mathcal{C}_X \mid \mu_Y)^*$
$ASAR$	$(\mathcal{C}_X \mid \mathcal{C}_Y)^*$
AS^*	$(\mathcal{C}_X \mu_Y)^*$

(a)

$$\begin{tikzcd}
 & ASAR \\
SSAR \arrow[ur, "\prec"'] & ASSR \arrow[u, "\prec"'] \\
SSSR \arrow[u, "\prec"'] \arrow[ur, "\prec"'] & AS^* \arrow[u, "\prec"'] \\
SS \arrow[u, "\prec"'] \arrow[ur, "\prec"'] & AS \arrow[u, "\prec"']
\end{tikzcd}$$

(b)

Figure 2. Modelling Spectrum

Let $traces(M)$ be the set of finite traces of agent M. We have that $M \preceq N$ if $traces(M) \subseteq$

$traces(N)$, and $M \prec N$ if $traces(M) \subset traces(N)$. The trace inclusion relations shown in Figure 2(b) can be inferred from the standard semantics of synchronous and asynchronous composition. These relations indicate the impact of synchronisation mechanisms on the interactions between the sources and the resources, and hence on the dynamic properties of congestion control, including stability.

1. $SSSR \prec SSAR \prec ASAR$ and $SSSR \prec ASSR \prec ASAR$.

   Asynchronous source agents can nondeterministically choose to ignore resource updates, while synchronous ones cannot. Hence in the case of asynchronous agents, the rate at which a source moves towards an equilibrium is uncertain.

   By including the resources explicitly as agents, these models can partially capture an uncertain propagation delay between the sources and resources, in the sense that one source (or resource) update will not take effect until the connected resources (or sources) act and thereby pass on the information.

2. $SS \prec SSSR$ and $AS \prec ASSR$.

   Since only the sources are active in SS and AS models, only source update labels appear in their traces. But if we incorporate the state change of resources' decision variable explicitly, the *full* traces for SS and AS models can be written as $(\mu_X \mu_Y)^*$ and $(\{\mu_{X_s} \mid s \in S\}\mu_Y)^*$, respectively, where μ_Y represents evaluating the resource functions. Thus, SS and AS models can be regarded as having *fast* resources, which immediately react to all source updates. This is equivalent to a primal/dual system in which there is a synchronous resource update after every source action. The difference between SS and AS is that in SS all sources act between the resource updates, while in AS only one source does.

3. $AS^* \prec ASSR$

   For AS^* (like SS and AS) each source update will take effect on all resources, that is, the resources will not miss any source update. On the contrary, $ASSR$ allows consecutive source updates, which can be interpreted as allowing the sources to update faster than the resources.

As can be seen from the above trace patterns, these modelling frameworks can capture the effect of network propagation to a certain extent. But since all agents always share the same view on the global state, these models cannot capture situations in which propagation delay leads the sources to act together but on an inconsistent view of the states of resources.

Remark. We were interested in whether nondeterminism from asynchrony would allow us to make statements about the behaviours of the congestion control protocols that are independent of propagation delay encountered by signalling mechanisms.

However, it seems that, because the state space of a system is simply the product of the state spaces of its component agents, we cannot model the *on-the-fly* message states between agents. These additional states make the system models considerably more complex but are also required if we need to model the situation when different agents can take inconsistent snapshots of the states of other agents.

A natural way to represent continuous propagation delay would be to augment the input/output (I/O) constructs of STGA with extra data facilities, such as queues. We do not pursue this here. But we observe that in the absence of fully specified queueing behaviour, the I/O constructs on their own do not help to model propagation delay since the synchronous semantics of the I/O constructs (which implements rendezvous communication) excludes any delay; while the asynchronous semantics does not preserve the correct propagation order when a sequence of outputs occurs. We approximately implemented continuous propagation delay with timed automata in UPPAAL. However, the resulting models are too complex for further investigation.

4. Verification

Given the presence of nondeterminism in the above algorithms, model checking is a natural choice for verification. Moreover, following a perturbation from an equilibrium (perhaps due to a fault), it would be useful to establish not only whether an algorithm will reconfigure the network flow to a new optimum, but also how quickly it will do. Although the objective function of problem (1) itself does not consider the convergence time, this can be investigated through model checking. In this section we report on the lessons learnt from a series of experiments we have run in this setting.

We first briefly recall the syntax and the semantics of CTL (Computation Tree Logic) [3], in which the stability property is formulated in this paper. CTL formulas are built up from atomic propositions $\mathcal{AP} = \{b, \dots\}$ using the Boolean connectives, path quantifiers "A" and "E", and temporal operators "X" (next), "F" (future), "G" (globally), "U" (until). Every occurrence of a path quantifier is immediately followed by a temporal operator. The semantics of CTL can be defined with respect to the concrete semantics of a SAG M_V. For a CTL formula φ and a state $q_{0\rho_0}$ of $[\![M_V]\!]$, let $q_{0\rho_0} \models \varphi$ denote formula φ holds at state $q_{0\rho_0}$. The relation $\models$ is defined inductively as follows (we omit the definitions for operators "X" and "U" as they were not used in our specifications):

- $q_{0\rho_0} \models b$ iff $\rho_0 \models b$;
- $q_{0\rho_0} \models AF\varphi$ iff for any run $q_{0\rho_0} \xrightarrow{\mu_{X_1}} q_{1\rho_1} \cdots$, there exists $i \geq 0$ such that $q_{i\rho_i} \models \varphi$;
- $q_{0\rho_0} \models EF\varphi$ iff there exists a run $q_{0\rho_0} \xrightarrow{\mu_{X_1}} q_{1\rho_1} \cdots$ and $i \geq 0$ such that $q_{i\rho_i} \models \varphi$;
- $q_{0\rho_0} \models AG\varphi$ iff for any run $q_{0\rho_0} \xrightarrow{\mu_{X_1}} q_{1\rho_1} \cdots$ and any $i \geq 0$, $q_{i\rho_i} \models \varphi$.
- $q_{0\rho_0} \models EG\varphi$ iff there exists a run $q_{0\rho_0} \xrightarrow{\mu_{X_1}} q_{1\rho_1} \cdots$ such that for any $i \geq 0$, $q_{i\rho_i} \models \varphi$.

4.1. Specifications

Herein we consider convergence of the objective function to a given value u^*, i.e., $\sum\limits_{s \in S} U_s(\sum\limits_{r \in s} x_r) = u^*$. To check whether there exists such a constant value to which the objective function may converge, we can repeatedly run model checking experiments exploring the range of u^* with the binary search strategy. Therefore, in our experiments we checked the following CTL specifications:

$$AFAG \quad \sum_{s \in S} U_s(\sum_{r \in s} x_r) = u^* \tag{8}$$

$$EFAG \quad \sum_{s \in S} U_s(\sum_{r \in s} x_r) = u^* \tag{9}$$

Note that even if we do not have equality in our language the formulas above can still be appropriately encoded through propositions.

Specification (8) states that the objective function always converges to some constant u^*, while Specification (9) states that it does so along at least one trace.

As well as considering the value of the objective function we can also consider the value of the flow rates when specifying stability. So we also checked the following CTL specifications:

$$AFAG \quad ((\sum_{s \in S} U_s(\sum_{r \in s} x_r) = u^*) \wedge \bigwedge_{r \in R} (x'_r = x_r)) \tag{10}$$

$$EFAG \quad ((\sum_{s \in S} U_s(\sum_{r \in s} x_r) = u^*) \wedge \bigwedge_{r \in R} (x'_r = x_r)) \tag{11}$$

Recall that x'_r is the next value of x_r.

4.2. Experimental results

The combination of the individual agent transitions described in Section 2 with the composition rules in Section 3 generates a collection of transition models that can be straightforwardly coded into NuSMV, which we ran on a Xeon Dual-Core 64-bit 2.8GHz machine with 1GB memory.

Two different initial congestion conditions were examined: *Route Overload* and *Resource Failure*. The first was designed to emulate a situation in which a network must redistribute load because one route has excess load which can be carried elsewhere. The second was to emulate a resource failure, that is, we set $C_{j_1} = 0$ so that resource j_1 cannot carry any traffic. Table 1 summarises the experimental results for the network shown in Figure 1, which has three sources and three resources. The capacity of each resource was set to 6.

The column *#Reach._st.* shows the number of reachable states of each model. It can be seen that the reachable state space grows dramatically, though as expected, from synchronous models to asynchronous models.

None of the 12 models of Table 1 satisfied Specifications (8) and (10), i.e., each of these models was found to be unstable. However, most models show that the network does converge along some trace in the sense that there exist values for the constant u^* resulting in Specification (9) or Specification (11) being satisfiable.

The penultimate column u^**(Stable)* reports the validity of Specification (11), hence Specification (9) for each model; while for this case the column *#Min_steps* presents the minimal number of control actions that the sources take to reach an equilibrium in each model. The column u^**(Vibrating)* shows the values of the constant u^* for which Specification (9) but not Specification (11) is satisfied.

The convergence results of these transition models are analysed in detail below.

Multi-path congestion/rate control. Our first batch of experiments were based on the congestion control protocol with primal transition rule (5) and dual update rule (4), with $\alpha = 36\beta$, $k_r = 2$ for each r. For the congestion condition of *route overload*, the initial configuration is set by $\vec{x} = (5, 3, 3, 3, 1, 3)$, where one route is overloaded by 2 units; while for the congestion condition of *resource failure*, the initial configuration is set with $\vec{x} = (0, 3, 3, 3, 3, 0)$ to emulate the failure at resource j_1. The model checking results are presented in Table 1(a).

Table 1. Model Checking Results

(a) Multi-Path Congestion/Rate Control

No.	Composition	Congestion	#Reach._st.	u^*(Stable)	#Min_steps	u^*(Vibrating)
1	SS	Route Overload	23	NONE		NONE
2	SS	Resource Failure	17	NONE		NONE
3	AS^*	Route Overload	82723	?		?
4	AS^*	Resource Failure	6187	$\ln(100)$	?	NONE
5	$ASSR$	Route Overload	2.06719e+06	?		?
6	$ASSR$	Resource Failure	35445	$\ln(100)$	8	NONE

(b) Session based Rerouting and Termination

No.	Composition	Congestion	#Reach._st.	u^*(Stable)	#Min_steps	u^*(Vibrating)
7	SS	Route Overload	3	NONE		NONE
8	SS	Resource Failure	7	NONE		14
9	AS^*	Route Overload	16	16	2	18
10	AS^*	Resource Failure	1418	11-12	3	13,15,16
11	$ASSR$	Route Overload	8513	12-18	2	13-18
12	$ASSR$	Resource Failure	32200	4-12	5	13-18

Because the individual transition rules are deterministic, their synchronous composition (SS) leads to a deterministic trace, and relatively few states. However, the discretisation of the continuous state space leads to a limit cycle rather than a final stable state, and model checking picked this up in the failure to satisfy Specification (8) and Specification (9).

For each AS^* and $ASSR$ model, as the interleaving constraints are relaxed, the number of accessible states increases, suggesting an instability in the system. This is not unexpected since it is now possible for one source agent to act many times before the others do: repeated actions on route r, when projected back into the optimisation framework, corresponds roughly to an increase in the gain coefficient κ_r, and increasing gain within a feedback loop typically leads to instability. Through model checking, similar limit cycles are detected in the AS^* and $ASSR$ models, which makes Specification (8) and Specification (10) unsatisfiable.

Therefore, the instability of these models is mainly caused by too much gain in each primal update. By increasing the size of the domain of route flow rates, we can set smaller gain coefficients in Equation (3). This results in finer discretisations in the sense that the step size of each primal update can be reduced. Table 2 reports the model checking results under the finer discretisation settings, where the capacity C_j of each resource j is 24 and the gain coefficient κ_r is set to 0.2 for each route r. Model k' is the correspondingly revised version of Model k in Table 1(a) under the finer discretisation setting. As before, the two types of initial congestion conditions are modelled: *Route Overload* with $\vec{x} = (20, 12, 12, 12, 4, 12)$ and *Resource Failure* with $\vec{x} = (0, 12, 12, 12, 12, 0)$. With such finer discretisations, specification (8) and (10), or their weaker forms, are satisfiable in these models for a single value or a set of values of the constant u^*. The stability results of SS and $AS^*/ASSR$ models conform to Theorem 1 and 2 in [10], respectively.

Table 2. Multi-Path Congestion/Rate Control - Finer Discretisation

No.	Composition	Congestion	#Reach._st.	$\{u^*\}$(Stable)	#Min_steps	u^*(Vibrating)
1’	SS	Route Overload	4	$\ln(10368)$	3	NONE
2’	SS	Resource Failure	4	$\ln(5400)$	3	NONE
3’	AS^*	Route Overload	11	$\ln(10368)$	3	NONE
4’	AS^*	Resource Failure	114	$\ln(5400)$	3	NONE
5’	$ASSR$	Route Overload	18	$\ln(9216)$	5	NONE
				$\ln(9600)$	4	
				$\ln(9984)$	3	
				$\ln(10368)$	2	
6’	$ASSR$	Resource Failure	225	$\ln(5400)$	6	NONE
				$\ln(5760)$	7	
				$\ln(6120)$	8	
				$\ln(6144)$	8	
				$\ln(6528)$	9	
				$\ln(6936)$	10	

Convergence in this setting mirrors the results of the underlying optimisation models in terms of differential equations. However, by means of model checking we can here quantify quite precisely the amount of control actions, hence the time, required for the network to recover from a perturbation. The above model checking results also indicate the impact of synchronisation mechanisms on the stability property of congestion control. The scenarios of convergence or oscillation under the synchronous frameworks are preserved by the asynchronous frameworks; however, variant interaction scenarios are introduced by the latter, and as a consequence, the number of states that the network may transition through before stabilising may be increased.

A composition of the agents which faithfully captures the optimisation dynamics of Equation (3) but which also allows certain nondeterminism in the interleaving of the agents is therefore not well represented in the options we have investigated. It appears that some notion of fairness is required, that is intermediate between (i) complete synchrony and (ii) forcing each agent always to act eventually. Developing a well motivated correspondence between optimisation dynamics and model checking requires both a notion of interleaving fairness and the quantisation of the state space to be taken into account together, as they can both be seen as related to the gain coefficients in the optimisation models.

Session based rerouting and termination. In light of the above observations we chose our second batch of experiments to be based on a scenario that is closer to the natural idiom of model checking. Here transition rules are not designed to correspond to smooth optimisation dynamics in any way. Instead they are designed to terminate flows in ways comparable to real systems like [4]. We use the dual update rule (6) and the primal update rule (7), while the latter features nondeterminism in which route it chooses to reallocate flow to. Again, we consider two types of initial congestion conditions: *Route Overload* with $\vec{x} = (5, 3, 3, 3, 1, 3)$ and *Resource Failure* with $\vec{x} = (3, 3, 3, 3, 3, 3)$ and $C_{j_1} = 0$.

The model checking results, presented in Table 1(b), show that the flow termination scheme does not ensure stability. Because the transition rules are designed only to shed load rather than to increase it we did not witness any state explosion corresponding to instability in any of the interleaving scenarios,

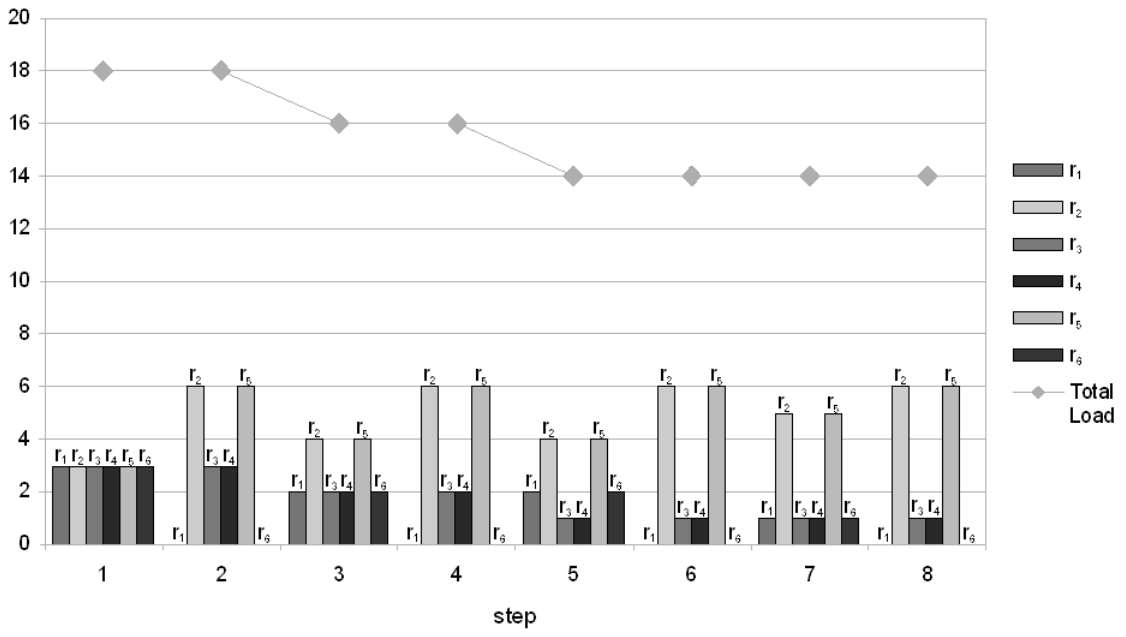


Figure 3. Stable total load with varying route flow rates

However, the last column u^**(Vibrating)* reports values of the constant u^* that are valid for Specification (9) but not Specification (11), that is, the objective function converges to the shown values but at least some route flow rates do not meet the capacity constraint and continue to change ("vibrate"). In these circumstances, the system is entering a limit cycle in which the total load offered into the network is greater than the capacity that it can carry, but no individual source knows it must terminate some flow. Instead they pass the excess flow around in the cycle. A trace showing this is presented in Figure 3. This behaviour appears possible in real systems; and model checking detected a real possible issue with this simple design.

Remark. In these two case studies, the $ASSR$ models may terminate more flows than the AS^* models do to converge. This suggests a performance degradation when the sources update faster than the resources and therefore miss a few resource updates.

Compared to the models of the multi-path congestion/rate control, the models of the session-based rerouting and termination system can lead to convergence by terminating more flows. The results also show that there exists a non-trivial lower bound on the number of excess flows to be terminated.

5. Discussion

In translating from optimisation based continuous dynamics to model checking, we identified two possibilities for the interpretation of nondeterminism. In the first, it represents the choice that each agent has *within* each action it takes. From an optimisation point of view this type of choice arises when the solution to a local problem is not unique: if two routes have equal least cost then the total flow can be split or moved arbitrarily between them. In the second interpretation, nondeterminism represents choice of *sequencing* of the actions of the agents which can be derived from the composition of the agents, and our first case study suggests that allowing too much nondeterminism in the agent composition would lead to system instability.

Another way of thinking about nondeterminism is that it accounts for loss of knowledge in moving from real systems to abstract models. We had hoped that this abstraction would allow us to make statements about the behaviour of congestion control policies that are independent of the propagation delay encountered by signalling mechanisms (indeed the resource matrix used in our models is already forgetful of details of the underlying resource connectivity). However, this was not straightforward. The difficulties arise from modelling the additional state actually present in the propagating messages. We did build models, though not reported here, in which we explicitly represented state "on-the-fly" within a signalling system (or within input queues and buffers). Our hope was that the increase in the system state space could be offset by reducing the explosion of possible evolution due to the interleaving semantics. In other words we produced larger but more deterministic models. However, our initial experiments still ran into size limitation of the model checkers that we used.

In the standard modelling idiom, the state space is the product of the state space of all the individual agents (sources and resources in our case). Crucially, this idiom abstracts away from the real state information on-the-fly. This same abstraction is also implicit in the smooth optimisation dynamics of Equation (3). In that case the agents are assumed to act sufficiently *slowly* for the abstraction to be valid. In model checking, by contrast, the agents are assumed to act instantaneously, but sufficiently *infrequently* for it to be valid. In all cases this assumption could be interpreted either as the limitation on the accuracy of the model (if analysing a system), or as constraints on implementation, or as conditions that must be policed by some other mechanism in the network (if synthesising a system). In both cases the assumption is quite brittle. In optimisation dynamics it has been shown that an arbitrarily short delay can render an otherwise stable system unstable [13]. In the model checking idiom an arbitrarily short delay could allow a sequence of transitions not captured by the delay free semantics. While the optimisation and discrete models appear at first sight to be quite different, in their modelling of delay it turns out that they share very similar types of limitation.

Hybrid model checking may also applicable for the continuous settings of optimisation. We believe it would emulate the standard optimisation based approaches to a much closer extent, though at a computational cost. However, it seems likely that it would confirm, among other things, the relationship between nondeterminism and stability discussed in the paper.

6. Conclusions

The paper presents a way to integrate optimisation based approaches with model checking. On the one hand, it associates optimisation models with nondeterminism; on the other hand, it associates the structure of optimisation models into model checking. A spectrum of modelling frameworks are presented importing different levels of nondeterminism: uncertain gain and propagation delay, and nondeterministic congestion control policies.

We believe that logic methods and model checking approaches should offer machinery that complements optimisation theory in the design and analysis of network control processes. We have investigated this proposition in the context of dynamic allocation of traffic amongst multiple routes across a network, a topic that is attracting attention within the networking research community. Our experiments showed some promise in this direction, but also some limitation. Not surprisingly we were limited to small concrete topologies by the state explosion problem. We see one way of addressing this could be to combine theorem proving and model checking techniques. However, our experiments highlighted more subtle points concerning the interpretation and specification of the interleaving semantics.

References

[1] Bengtsson, J., Larsen, K. G., Larsson, F., Pettersson, P., Yi, W.: UPPAAL - a Tool Suite for Automatic Verification of Real–Time Systems, *Proceedings of the DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems III*, number 1066 in LNCS, Springer Berlin, New Brunswick, New Jersey, United States, October 22-25 1995.

[2] Cimatti, A., Clarke, E. M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella., A.: NuSMV 2: An OpenSource Tool for Symbolic Model Checking, *Proceedings of the 14th International Conference on Computer-Aided Verification (CAV'02)*, number 2404 in LNCS, Springer Berlin, Copenhagen, Denmark, July 27-31 2002.

[3] Clarke, E. M., Emerson, E. A., Sistla, A. P.: Automatic verification of finite-state concurrent systems using temporal logic specifications, *ACM Transactions on Programming Languages and Systems (TOPLAS)*, **8**(2), 1986, 244–263, ISSN 0164-0925.

[4] Eardley, P.: Pre-Congestion Notification (PCN) Architecture, Internet-Draft draft-ietf-pcn-architecture-08, 2008.

[5] Holzmann, G. J.: The Model Checker SPIN, *IEEE Transactions on Software Engineering*, **23**(5), 1997, 279–295.

[6] Jaggard, A. D., Ramachandran, V., Wright, R. N.: The Impact of Communication Models on Routing-Algorithm Convergence, *Proceedings of the 29th International Conference on Distributed Computing Systems (ICDCS'09)*, IEEE Computer Society, Montreal, Quebec, Canada, June 22-26 2009.

[7] Kelly, F., Maulloo, A., Tan, D.: Rate control for communication networks: shadow prices, proportional fairness and stability, *Journal of the Operational Research Society*, **49**(3), 1998, 237–252.

[8] Kelly, F., Voice, T.: Stability of end-to-end algorithms for joint routing and rate control, *ACM SIGCOMM Computer Communication Review*, **35**(2), 2005, 5–12.

[9] Lin, H.: Symbolic transition graph with assignment, *Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96), LNCS 1119*, Springer-Verlag, 1996.

[10] Low, S. H., Lapsley, D. E.: Optimization flow control, I: basic algorithm and convergence, *IEEE/ACM Transactions on Networking*, **7**(6), 1999, 861–874.

[11] Sobeih, A., Viswanathan, M., Hou, J. C.: Check and simulate: a case for incorporating model checking in network simulation, *Proceedings of the 2nd ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE'04)*, IEEE, San Diego, California, United States, June 22-25 2004.

[12] Strulo, B., Walker, N., Wennink, M.: Lyapunov Convergence for Lagrangian Models of Network Control, *Proceedings of the 1st EuroFGI International Conference on Network Control and Optimization (NET-COOP'07)*, number 4465 in LNCS, Springer Berlin, Avignon, France, June 5-7 2007.

[13] Voice, T.: Stability of multi-path dual congestion control algorithms, *Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06)*, ACM, Pisa, Italy, October 11-13 2006.

[14] Walker, N., Wennink, M.: Interactions in Transport Networks, *Electronic Notes in Theoretical Computer Science*, **141**(5), 2005, 97–114.

[15] Wennink, M., Williams, P., Walker, N., Strulo, B.: Optimisation-Based Overload Control, *Proceedings of the 1st EuroFGI International Conference on Network Control and Optimization (NET-COOP'07)*, number 4465 in LNCS, Springer Berlin, Avignon, France, June 5-7 2007.

[16] Yuen, C., Tjioe, W.: Modeling and verifying a price model for congestion control in computer networks using promela/spin, *Proceedings of the 8th International SPIN Workshop on Model Checking of Software (SPIN'01)*, number 2057 in LNCS, Springer Berlin, Toronto, Ontario, Canada, May 19-20 2001.