Model Checking Optimisation Based Congestion Control Models

Alessio Lomuscio[1], Ben Strulo[2], Nigel Walker[2], and Peng Wu[3]

[1] Department of Computing, Imperial College London, UK
[2] BT Research, Adastral Park, UK
[3] Department of Computer Science, University College London, UK

Abstract. Model checking has been widely applied for verification of network protocols, particularly on the sequences of interactions between protocol entities. 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 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.

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.

In the case of routing or flow control protocols network-wide properties are specially concerned, such as whether a stable routing configuration could be found, or whether link capacity could be fairly and efficiently shared between communication sources. Optimisation theory has provided a successful approach to this type of question. This approach abstracts away from the details of packet arrivals and transmissions, 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 hope to widen the scope of models of this type of network problem, and better understand the merits of the two approaches. We

take examples from the area of congestion control, which has been extensively studied in the networking literature [1]. 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 inherited from the underlying optimisation model. The source agents represent sources that control primal variables, while the resource agents represent resources that control dual variables.
- A range of options is presented for composing these agents, ranging from fully synchronous models to fully asynchronous models and various combinations of them.
- Nondeterminism can capture aspects not modelled within the optimisation framework, such as uncertain gain (a parameter within the agents) or propagation delay.

The experiments we report here used the NuSMV [2] model checker, due to its full support for CTL and LTL, as well as explicit fairness constraints. The 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 [3] and UPPAAL [4], with no better results in other slightly different settings. More details of the work are available in a technical report [5].

Related Work. Optimisation based approaches are now standard for analysing congestion control, starting with [6]. [1] proposed a stable fluid-flow framework for joint routing and rate control on which our first case study is based. [7, 8] are similar works both using a Lagrangian optimisation model, which decomposes into distributed synchronous and asynchronous algorithms for congestion control.

On the other hand, formal verification techniques were introduced into network research. [9] applied SPIN to verify the equilibrium property of a priority pricing based congestion control model. [10] presented an extended compositional network simulation environment with the capability of bounded model checking.

Our work differs from [9] in that we discretise and analyse a continuous optimisation based congestion control model and explore the issue of nondeterminism in this setting, while the model checking result in [9] is just applicable for the particular model considered. The work closest to our own is the asynchronous algorithm presented in [11], which was based on optimisation but schedules the sources and resources in a nondeterministic order.

Structure. Section 2 briefly introduces two congestion control protocols. Section 3 presents the modelling spectrum with expressiveness analysis.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. The paper is concluded in Section 6.

2 Optimisation Based Congestion Control

This section will briefly present 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 using a non-empty subset of resources. Each route connects only one source with its destination. Let $r \in s$ denote that source s can transmit along route r and $s(r)$ be the unique source s such that $r \in s$. Let x_r be the flow rate on route $r \in R$ and C_j be the capacity of resource $j \in J$. Set $A_{jr} = 1$ if $j \in r$ and $A_{jr} = 0$ otherwise. Fig. 1(b) presents the resource topology of the simple network shown in Fig. 1(a), in which each source owns two routes, each route uses only one resource, and each resource is shared by two routes .

Let $\mathtt{x}$, C and A be the corresponding vectors and incidence matrix, that is, $\mathtt{x} = (x_r, r \in R)$, $C = (C_j, j \in J)$ and $A = (A_{jr}, j \in J, r \in R)$. Let x_j denote the aggregate flow rate at resource j. A resource j is *congested* if $x_j > C_j$. A route r is *congested* if j is congested for some $j \in r$.

Then, the multi-path congestion control problem is typically specified as the following optimisation problem:

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

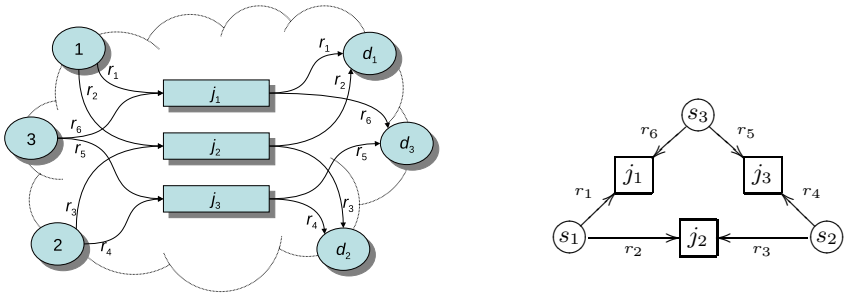
where U_s is a utility function of the total flow sent by source s over all paths available to it. The network implements a distributed algorithm to solve this problem, which is best understood by moving the constraints into the objective function to give the associated Lagrange relaxation. Assume that for each $s \in S$, U_s is increasing and strictly concave in its argument. Then, problem (1) can be solved by finding a saddle point of the following Lagrangian function:

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

where $\mathtt{y} = (y_j, j \in J)$, $\mathtt{y} \geq 0$ and $\mathtt{x} \geq 0$. This is because if $(\mathtt{x}^*, \mathtt{y}^*)$ is a saddle point of L, then $\mathtt{x}^*$ is an optimal solution to problem (1) [11].

The $\mathtt{x}$ and $\mathtt{y}$ in the Lagrangian function $L(\mathtt{x}, \mathtt{y})$ are referred to as *primal* and *dual* decision variables, respectively. Each primal (dual) variable selects its value on behalf of a source (resource) such as to maximise (minimise) the value of $L(\mathtt{x}, \mathtt{y})$, given the values of all other variables. Thus, a solution to problem (1) is where the interactions between the sources and resources reach an equilibrium [7, 8]. The dual variables have the interpretation of price, and act as a feedback signal from the network to the sources indicating congestion.

An optimisation based approach would then specify an algorithm modelled as a set of trajectories in the primal and dual decision variables. Typically the utility



(a) Network Topology (b) Resource Topology

Fig. 1. A Communication Network

functions U_s are assumed differentiable, and trajectories are specified as differential equations. The analysis would then demonstrate provable convergence of these trajectories to the optimum under certain modelling and applicability assumptions [6, 11, 1, 7, 8, 12]. The rest of this section will present two congestion control protocols based on the above multi-path setting.

2.1 Multi-path congestion/rate control

In [1] a protocol was presented that gives rise to trajectories in the primal flow rates x_r 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, U'_s is the first-order derivative of U_s 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 serving source s;
- $(z)^+_x = \min(0, z)$ if $x \leq 0$, otherwise $(z)^+_x = z$.

In [1] 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.

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, $U_s = \alpha_s \ln(x_s)$, where α_s is a utility coefficient. We choose y_j to be a linear function of the flow rates at resource j, that is,

$$y_j = \beta_j x_j \quad \text{with} \quad x_j = \sum_{j \in r} x_r \tag{4}$$

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

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

Here, κ_r can be regarded as a *gain* coefficient that defines the pace at which route r seeks its equilibrium; while $\dfrac{\beta_r}{\alpha_s}$ defines the saddle point where route r will settle.

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.

We consider a system where the sources are managing a non-empty set of constant flow rate sessions. The primal variable x_r of problem (1) can be 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 sounds appropriate for a network operator who typically treats all sessions equally.

Then 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 $r' \in s$, which is 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,

$$y_j = \frac{x_j - C_j}{x_j} \quad \text{with} \quad x_j = \sum_{j \in r} x_r \tag{6}$$

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 $r \in s$.

3 Modelling

From optimisation based congestion control models, distributed algorithms were developed, depending on whether it is the sources (primals), the resources (duals), or both controlling the evolution of a system actively, as well as the types of interleaving of their control actions. Most of these algorithms schedule the sources and/or resources synchronously along the unique continuous time scale [11, 1, 8], while an asynchronous algorithm was also presented in [11], which schedules the sources and resources in a nondeterministic order. In the rest of this section we will explore the options of these composition structures.

One standard form of these algorithms is termed primal algorithms, in which the sources actively adjust their primal variables and the resources just immediately recalculate the values of their dual variables. For such algorithms, either one, some, or all sources can act synchronously. We term these three possible systems AS, AS^*, and SS. Note that trivially the traces of AS^* includes those of both AS and SS.

The symmetric form of primal algorithms, termed dual algorithms, can also be modelled where the resources take the active parts. However, we do not present these here for lack of space.

The other form is termed primal/dual algorithms, in which both classes of agents are active. For such algorithms, we consider the asynchronous composition of these three possible systems for both primals and duals. If we compose a system where each action is that of a single agent, we obtain the fully asynchronous system termed $ASAR$. We can also compose either sources or resources or both constrained to all act synchronously giving systems $SSAR$, $ASSR$, and $SSSR$.

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

With the above consideration, the following trace inclusion relations are inferable from the standard semantics of synchronous and asynchronous composition. These relations actually also clarify in detail the difference of these modelling frameworks in representing the interactions between the sources and resources.

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. Thus changing to asynchronous agents makes uncertain the rate at which a source moves towards an equilibrium.
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 $AS \prec AS^* \prec ASSR$.
SS and AS models can be regarded as having *fast* resources, which immediately react to all source updates. So this is equivalent to a primal/dual system in which there is a synchronous resource update after every source action.
For AS^* (like SS and AS), each source update will take effect on all resources before the next one. 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.

4 Verification

With 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 find out not only whether an algorithm will reconfigure the network flow to a new optimum, but also how quickly it does so. 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.

4.1 Specifications

Herein we consider convergence of the objective function to a given value u^*, i.e., $\sum_{s\in S} U_s(\sum_{r\in s} x_r) = u^*$. 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:

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

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

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.

The Lagrangian function $L(\mathtt{x}, \mathtt{y})$ also concerns convergence of each route flow rate, which would lead to a saddle point of the function itself. So we also consider the following CTL specifications:

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

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

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 gives 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 Fig. 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 the reachable states of each model. Due to the data-oriented nature of the underlying optimisation models, it can be seen that the reachable state space grows dramatically, though as expected, from synchronous models to asynchronous models.

For the twelve models in Table 1, Specification (8) and Specification (10) were found not satisfiable, that is, each of these models was found to be unstable. However, most models show that the network does converge along some traces in the sense that there exist values for the constant u^* resulting in satisfiable Specification (9) and/or Specification (11).

The penultimate column *u^*(Stable)* reports the validity of Specification (11), and 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.

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 $\alpha_s = 36\beta_j$, $k_r = 2$ for each s, j, r. The model checking results are presented in Table 1(a).

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

(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	14-18	2	12,13
12	$ASSR$	Resource Failure	32200	4-11	5	13-18

Table 1. Model Checking Results

than a final stable state, and model checking has 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 a severe 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.

Session based rerouting and termination In the 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 [13]. 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.

The model checking results are presented in Table 1(b). Because the transition rules are designed only to shed load rather than to increase it we do not, in any of the interleaving scenarios, see any state explosion corresponding to instability.

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

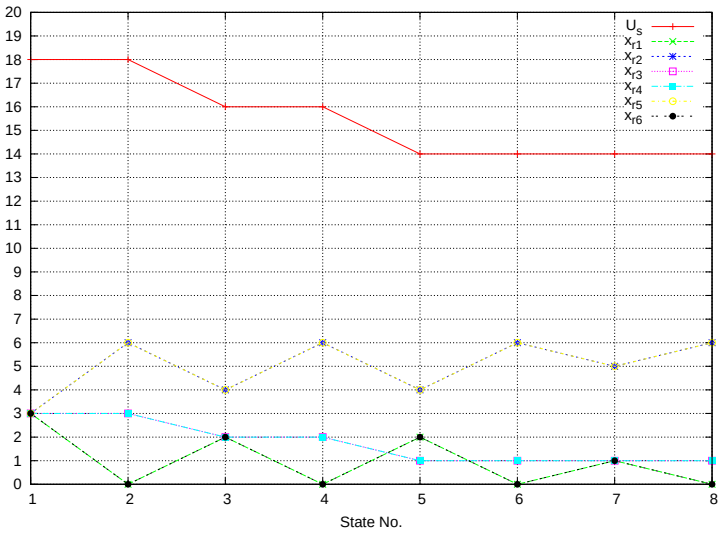


Fig. 2. Stable total load with varying route flow rates.

function converges to the shown values but at least some route flow rates $x_r, r \in s$ 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 Fig 2. This behaviour sounds possible in real systems and therefore model checking has detected a real possible issue with this simple design.

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 both 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 [14]. 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.

6 Conclusions

The paper presents a way to integrate optimisation based approaches with model checking. On 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 prov-

ing and model checking techniques. However, our experiments highlighted more subtle points concerning the interpretation and specification of the interleaving semantics.

References

1. 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
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. In: Proceedings of the 14th International Conference on Computer-Aided Verification (CAV'02), Copenhagen, Denmark (2002)
3. Holzmann, G.J.: The model checker SPIN. IEEE Transactions on Software Engineering **23**(5) (1997) 279–295
4. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Uppaal — a Tool Suite for Automatic Verification of Real–Time Systems. In: Proceedings of Workshop on Verification and Control of Hybrid Systems III. Number 1066 in Lecture Notes in Computer Science, Springer–Verlag (1995) 232–243
5. Lomuscio, A., Strulo, B., Walker, N., Wu, P.: Model checking optimisation-based congestion control models. Technical Report RN/09/02, Department of Computer Science, University College London (2009)
6. Kelly, F., Maulloo, A., Tan, D.: Rate control for communication networks: shadow prices, proportional fairness and stability. Journal of the Operational Research Society **49** (1 March 1998) 237–252(16)
7. Walker, N., Wennink, M.: Interactions in transport networks. Electronic Notes in Theoretical Computer Science **141**(5) (2005) 97–114
8. Wennink, M., Williams, P., Walker, N., Strulo, B.: Optimisation-based overload control. In: NET-COOP. (2007) 158–167
9. Yuen, C., Tjioe, W.: Modeling and verifying a price model for congestion control in computer networks using promela/spin. In: Proceedings of the 8th International SPIN Workshop on Model Checking of Software (SPIN'01), New York, NY, USA, Springer-Verlag New York, Inc. (2001) 272–287
10. Sobeih, A., Viswanathan, M., Hou, J.C.: Check and simulate: a case for incorporating model checking in network simulation. In: Proceedings of the 2nd ACM and IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE'04). (2004) 27–36
11. Low, S.H., Lapsley, D.E.: Optimization flow control, I: basic algorithm and convergence. IEEE/ACM Transactions on Networking (TON) **7**(6) (1999) 861–874
12. Strulo, B., Walker, N., Wennink, M.: Lyapunov convergence for lagrangian models of network control. In: NET-COOP. (2007) 168–177
13. Eardley, P.: Pre-congestion notification (PCN) architecture. Internet-Draft draft-ietf-pcn-architecture-08 (2008)
14. Voice, T.: Stability of multi-path dual congestion control algorithms. In: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools (VALUETOOLS '06), New York, NY, USA, ACM (2006) 56