Forecasting Unstable Policy Enforcement

Javier Baliosian and Ann Devitt,
Network Management Research Centre, Ericsson Ireland
Athlone, Ireland
Email: {javier.baliosian,ann.devitt}@ericsson.com

***Abstract*— Policy-based network management (PBNM) is a promising but not yet delivering discipline aimed at automating network management decisions based on expert knowledge and strategic business objectives. One of the issues which is almost not being addressed in PBNM is the stability of the managed system as the result of the dynamic interaction between the “natural” network behaviour with the autonomous decision making. Yet this issue is central to the design of a self-management networking system comprised of autonomous entities making decisions driven by policies with often unknown consequences. Decisions made by one entity may change the context and configuration of other autonomous entities which may in turn react changing the context and configuration of the first entity triggering an unbounded chain of re-configuration actions. It is possible to model obligation policies and their constraints with finite state transducers (FST). It is also possible to learn patterns of recurrent behaviour using Bayesian networks (BN), a structurally similar kind of graph. The method presented in this paper analytically composes both finite state machines to derive predictions of the consequences of enforcing a given policy improving system stability.**

I. Introduction

The current explosion in size, complexity and heterogeneity of networks which has been driven by recent advances in wired and wireless networking technology is set to continue into the future with networks growing at an exponential rate. At the same time, network operators are struggling in a highly competitive market where they must keep their running costs to a minimum. The conflict between the ever-increasing demands of running large, complex and heterogeneous networks and the ever-decreasing OPEX (operational expenditure) budgets of network operators is driving intensive research in the area of autonomous networks. A promising development in this domain is policy-based network management (PBNM) which aims to automate network management decision-making, imposing strategic business and technical objectives on all network self-configuration activity.

A core issue for any autonomous system, however, is stability and this has not been the focus of research to date. Given its freedom to instigate changes in the network, an autonomous management system may provoke cascades of changes or configuration flipflops by zealous application of operator directives in the form of policies. In order to mature, like any autonomous being, the system must learn to observe and deal with the consequences of its actions. In effect, to be truly autonomous and not reliant on human intervention for its development, it must learn from experience. This entails observing its own activity, determining what are the effects of its actions and acting on this knowledge to modify future actions. The research presented in this paper aims to address this requirement by combining state of the art policy-based management and machine learning techniques. The outcome is a policy engine which can modify its decisions at run-time with reference to patterns of behaviour learnt automatically over long timescales. Section II presents the architecture of an adaptive policy-driven autonomous node. Section II-A describes the policy model for the self-ware node. Section II-B outlines the machine learning component designed to learn patterns of recurrent behaviour. The stability of the system is controlled by a composition of these two structures which is described in section III and illustrated by an example in section IV. Section V outlines how the output of the machine learning component can be further exploited to provide a probabilistic assessment of the risk of particular instabilities in the system. Section VI concludes with a discussion of how this forecasting method will be realised and evaluated.

II. Self-ware Node Architecture

The method presented in this paper must be considered in the context of a network management system, such as that outlined in [1], where the network elements have been granted a degree of autonomy to manage themselves in a fully or partially distributed environment. In this context, network nodes are empowered to perform configuration and maintenance tasks. The architecture of a single node in such an autonomous system is depicted in Figure 1. It relies on the interaction of three different kinds of knowledge with their respective approaches for knowledge capture and manipulation in order to direct and constrain the behaviour of the network elements. These layers of knowledge map to the following three components: **Network Model** (NM), **Behaviour Model** (BM) and **Policy Model** (PM). The Network Model comprises relatively static prior knowledge about network configuration constraints and procedures modelled as an ontology where the model instance is updated in real-time. The Behaviour Model represents and learns patterns of network behaviour either as it is realised by management operations or the control plane. The Policy Model (PM) models and enforces expert knowledge and preferences about network configuration and strategic objectives. The architecture assumes an event-based system where events are messages about occurrences, such as alarms or performance parameter increment events, or action requests from one node to itself or another node. Events are distributed by the Event Bus component in Figure 1, which is

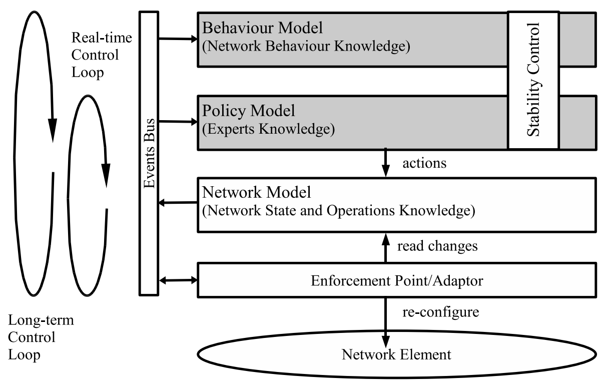
Real-time Control Loop
Long-term Control Loop
Events Bus
Behaviour Model (Network Behaviour Knowledge)
Policy Model (Experts Knowledge)
Stability Control
actions
Network Model (Network State and Operations Knowledge)
read changes
Enforcement Point/Adaptor
re-configure
Network Element

Fig. 1. Context Architecture

the local realisation of a semi-distributed event management functionality, set out in [1].

This architecture, described in more detail in [2], was the platform used to develop the present composition method but it may be considered in the context of any other architecture which includes behaviour model and policy model components. The following sections outline the functionality of these two components and how their underlying graphical representations may be composed to predict consequences of configuration changes. This functionality addresses the key issue of stability of self-configuring systems by monitoring the effects of configuration actions over long timescales and adapting policy decisions on the basis of observed effects.

A. Policy Model

This architecture follows the event-condition-action paradigm where the event may be an alarm or a service request, the condition is evaluated on the event parameters and the network state encoded in the distributed Network Model, and the action is the desired response to that event/condition, as defined by the operator. The Policy Model in this architecture includes a classical Policy Decision Point (PDP) [3], responsible for listening for events coming from the Events Bus and evaluating the conditions and the local policies in order to decide whether a policy condition has been met and therefore the attendant reconfiguration action(s) must be performed. Policies are modelled using a special kind of finite state machine, presented in [4], called *Finite State Transducers extended with Tautness Functions and Identities* (TFFST). These machines are graphs with two labels on each edge, one expressing an input symbol and another specifying an output symbol. A simple *if-then* rule like *"**if** $|jitter| > 20ms$ **then** re-route Video-class connection;"* may be expressed as a transducer that receives information about jitter for one connection and, depending on this value, produces a certain re-routing action. This is illustrated in Figure 2. The TFFST has only one edge, its input label represents the type of event the rule is expecting (i.e., jitter) and the condition that this must fulfil (i.e., the jitter value) and the output label the output action that must be performed. This evaluation model is oriented to the resolution of policy

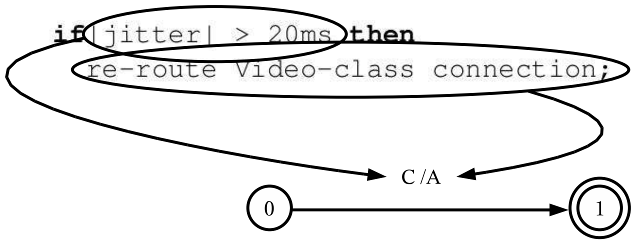

Fig. 2. Correspondence between if-then constituents and TFFST elements.

conflicts and is intended to show good policy evaluation performance. Its morphology is relevant to the analytic procedure presented in section III.

B. Behaviour Model

In any self-managed network, such as that considered here, the network devices take decisions and execute actions in order to fulfil a service and/or user requirements. These actions or behaviour can affect the network as a whole. Thus, it becomes critical to observe the behaviour of network devices and the network as a whole in order to evaluate the effects of their autonomous activities, such as configuration changes. The behaviour of a network device is represented by the events (actions and notifications) on that network device. The Behaviour Model component effectively serves to log and summarise over individual events to give a general view of the activity of a network device.[1]

More precisely, the internal representation of the Behaviour Model is a Dynamic Bayesian Network (DBN) which is state-of-art technology used to monitor different types of behaviour with a temporal dimension, for example power consumption of machines [5] or fault propagation in industrial processes [6]. Bayesian Networks (BN) [7] provide a means of monitoring behaviour by specifying the dependencies and independencies that hold between aspects of a system. BNs consists of a directed acyclic graphical structure (DAG), where the nodes represent variables from an application domain, in this case, events in a network, and the arcs represent the influential relationships between them. Furthermore there is an associated conditional probability distribution over these variables which encodes the probability that the variables assume their different values, for example present vs. absent given the values of other variables in the BN. Figure 3 shows an example Bayesian Network for the network events "jitter" and "rerouteVideo" from the TFFST in figure 2, their possible values and a probablity distribution for each value of the variables. In the Behaviour Model described here, the probability distribution is learnt incrementally on-line for each network device from the event activity of that network device. Time in Bayesian Networks is implicitly represented by the arcs of the model which denote

[1]In the overall network architecture, the local node behaviour models are aggregated to more global "group of nodes" behaviour models. In the semi-distributed environment described in [1], nodes are organised in clusters and individual nodes summarise their local model to be aggregated in the global model of the clusterhead node.

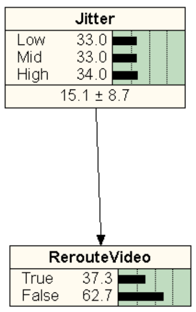

Fig. 3. Bayesian Network representation of jitter TFFST

a causal relationship. Dynamic Bayesian Networks [8] are a generalisation of Bayesian Networks that explicitly model changes in the model over time with additional temporal arcs. The system exploits this temporal dimension to ensure the sequential nature of events represented in the Behaviour Model.

III. Forecasting Instability

In the last section we have introduced the two entities needed for our forecasting method, the Behaviour Model encoding long-term statistical knowledge about the observed network behaviour and the Policy Model containing the experts knowledge that drives the adaptation of the network to the changing context towards some strategic objectives. Their morphological similarity and the fact that both models are composed of the same set of events and actions have induced us to combine their information analytically. This section describes the general method for this combination of the two models and section IV presents the TFFST and DBN for an example flip-flop prediction scenario and illustrates how the DBN and TFFST models are composed to enforce system stability.

A. Bayesian Network to Finite State Transducer Conversion

In order to analytically combine the knowledge in the Behaviour Model with the knowledge in the Policy Model, they must be translated to a common representation, in this case, the TFFST representation on which the policies are already modelled. The nodes of the Behaviour Model DAG represent an individual random variable, the range of values that variable can take and the probability of the variable assuming that value given the values of its parent node(s). For our purposes, those random variables are different event types in the system and the values this event can take (parameter ranges, presence/absence, etc). The graph expresses the probability of a child node assuming a particular value of a variable given the parent node variable assumes a given value, e.g., the probability that the reRouteVideo variable in figure 3 has value $true$ when the jitter variable has value $high$. As described in section II-A, in the TFFST policy model, each arc represents the desired consequence (the output label) of a given event (the input label), according to an expert preferences. In order to convert the Behaviour Model DAG to the Policy Model TFFST, the DAG must be compiled out to the representation of transitions between the various values or states of its random variables. For example, the DBN in figure 3 is compiled out into the 3 FSTs in figure 4, ignoring the $reRoute = false$ value. Figure 5 illustrates the general case where the DBN (shown on the left) with variable values $\{a, b, c\}$ is translated into an TFFST (shown on the right).[2] To translate a DBN into a weighted transducer [9], a TFFST is created in which each event of the DBN is the consequence of its parent events in the DBN graph and the weight of that specific path of the TFFST is the corresponding conditional probability in the DBN.

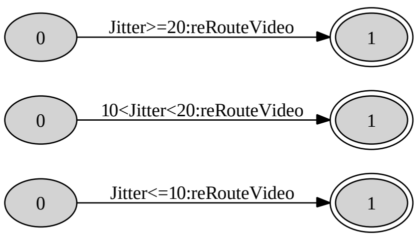

Fig. 4. Converted TFFST from BN

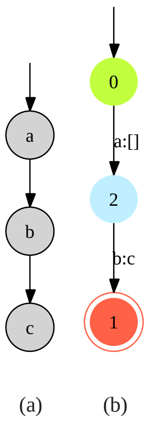

Fig. 5. The DBN on the left is translated into the TFFST on the right.

B. Composition of Finite State Transducers

Transducers composition is a precisely defined operation between FSTs. The meaning of composition here is the same as for any other binary relations:

$$R_1 \circ R_2 = \{(x; z) \mid (x; y) \in R_1; (y; z) \in R_2\}$$

This can be seen as a chain of processing events: the output events from the first transducer are taken as input to the second one. However the whole process is expressed and carried

[2]The special symbol "[]", also represented as an ϵ, means that no output is produced by the transition. See [4] for details on TFFSTs.

out by a single FST, the one resulting from the composition operation. For example in Figure 4 the composition of the first and second FSTs produces the third FST, as detailed in [4].

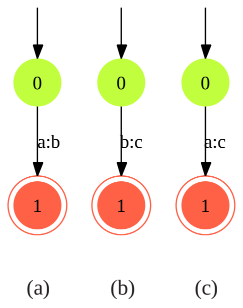

Fig. 6. Basic Composition Example.

C. Forecasting

The purpose of the analytic composition of the behaviour DBN and policy TFFST models is to derive predictions of the consequences of enforcing a given policy, in particular to detect flip-flop configuration changes. The steps of the analytic flip-flop forecast process proceed as follows:

1) Translate Bayesian Network of the BM into a TFFST.
2) Compute the union of BM and PM TFFSTs.
3) Compose the TFFST from the previous step with itself.
4) Detect repetitions of events in the input and the output of every possible path.
5) If there is no repetition detected in step 4, return to step 3, otherwise stop the process.

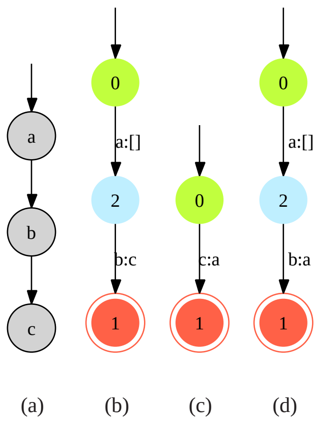

Fig. 7. Analytic Flip-Flop Discovery.

As a high-level example, Figure 7(a) represents the discovered pattern of the action *a* followed by the action *b* and the event *c*. As in section III-A, Figure 7(b) represents the same pattern as a TFFST. The TFFST in Figure 7(c) represents the rule "if c then a" and Figure 7(d) is the composition of the transducers in figures 7(b) and 7(c). In the composition, the action *a* appears on both sides of the transducer, in the input and in the output. This means that performing action *a* eventually (or with a high probability) will cause the execution of the same action *a* again, a flip-flop behaviour that may be prevented by ignoring the rule modelled by Figure 7(c). A detailed and specific example is presented in section IV.

When to Stop the Iteration: In the Behaviour Model, the probability of the occurrence of an event is computed inside a given sampling time window, each temporal edge in the BM DBN corresponds to that period of time. Therefore, the shortest path length of the DBN graph times the sampling window corresponds to the shortest period of time represented by the Behaviour Model (T_{min}) and the longest path length times the sampling window is the longest period of time modelled (T_{max}). In this way, we can associate each loop in the iteration of steps 3 to 5 with a period of time between T_{min} and T_{max}. The aim of this method is to identify repetitive behaviour that will produce a continuous device reconfiguration over short–medium timescales. If the flip-flop occurs over a medium–long timeframe (i.e. days or weeks), this may be interpreted as a routine system adaptation. Thus, the iteration must finish when the accumulated time reaches a predefined threshold or safe duration.

IV. An Example: a Dynamic Standby Link

In a 3G network, an RNC node is connected to each of its RBS nodes by a primary link and a secondary standby link. Traditionally the standby link is a second fibre on which a standby link is configured when the network is rolled out. This scenario represents a hypothetical network management function in a 3G telecommunications network using IP transport to configure standby links dynamically according to network demands rather than statically at network roll-out time. This function is potentially very useful as standby links reserve specific resources in the nodes they connect and also in any node that cross-connects the link. In this scenario, network nodes which are experiencing high traffic can free up resources by dropping one or many standby links which they cross-connect. Dynamic reconfiguration of standby links would allow nodes to free up these reserved resources at need for revenue-producing traffic.

Figure 8 illustrates this scenario for four network nodes. The standby link between the RNC and RBS B is cross-connected through Router A. This standby link therefore has reserved resources in the RNC, RBS B and also Router A. In this example, Router A can drop the standby link between RNC and RBS B if it is experiencing high traffic levels and requires the reserved resources to meet user demand. The network device which governs the standby link configuration, the RNC, will then try to reconfigure a new standby link. This dynamism to configure and drop links at will can result in a cycle where the RNC configures a standby link through Router A who then drops the link when it is overloaded and then the RNC tries to reconfigure the link through Router C who may then drop the standby link again if it is overloaded. If the traffic load on both Router A and C are not correlated this might not be a problem. The resulting flip-flop may occur over a large or medium timescale, constituting a dynamic adaptation rather than a disrupting inconvenience. However, if the loads of Router A and Router C are correlated, for example if both are in the vicinity of a stadium and on Saturdays the zone receives much more people than on working days, the flip-flop described may occur at a very fast rate, becoming a problem for the devices being re-configured continuously. Thus, the desired dynamism could result in an unbounded set of changes where no standby link is stable for any substantial duration as the area itself is inherently overloaded suggesting that some longer term solution must be found.

It is this kind of instability that this method is designed to foresee and hence avoid. The sections below outline the policies, configuration actions and behaviour model for this scenario and describe how the composition of the policy model and behaviour model can identify the cycle of reconfiguration actions and thereby stop the cycle.

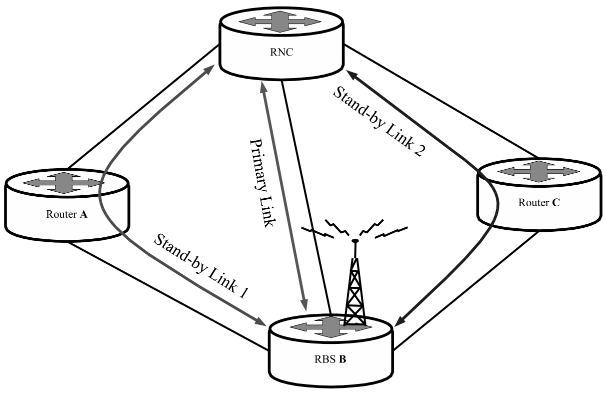

Fig. 8. Standby link scenario network schema.

A. Messages

In this example, the workflow of the system is driven by messages that are interchanged or broadcast between the managed nodes. These messages report occurrences or action requests made by one node to another. The relevant messages for this example are:

- **reqA** and **reqC**: a standby link acceptance request message sent from RNC to Router A and Router C respectively.;
- **dropA** and **dropC**: a message from Router A or Router C sent to the RNC requesting to drop and re-route the standby link through them;
- **oLoad**: the set of events informing about a node overload;
- **oLoadA**: an event informing that specifically Router A is overloaded;
- **oLoadC**: an event informing that specifically Router C is overloaded;
- **sat**: a time event stating that it is Saturday.

B. Policies

The system presented in this simple example is governed by only one policy that governs what the nodes do when they are overloaded in order to gain resources. In this case, the preferred way to gain resources is to drop hosted standby links because they are reserving resources but not using them. The TFFST model for this policy is depicted in Figure 9.

Policy 1: `If the node N is overloaded then ask RNC to drop and reroute standby links which are currently routed through N.`

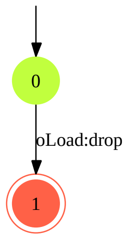

Fig. 9. Policy 1 as a TFFST.

C. Belief States

Figures 10, 11, 12 and 13 show subparts of the DBN model and the FST translations of the learnt high probability transitions or belief states relevant to this scenario. For simplicity, in this example we are disregarding the probability of the correlation between events. This is equivalent to assuming that two successive events in the DBN are correlated with a probability of 1 ($p = 1$). Section V discuss the impact of considering the probabilities in the DBN.

Belief State 1: On Saturdays and after it was requested to host a standby link, Router C may be overloaded for some periods of time (Fig. 10, DBN left, FST right).[3]

Belief State 2: On Saturdays and after it was requested to host a standby link, Router A may be overloaded for some periods of time (Fig. 11, DBN left, FST right).

Belief State 3: After Router C asks RNC to reroute standby links, RNC requests Router A to host a standby link (Fig. 12, DBN left, FST right).

Belief State 4: After Router A asks RNC to reroute standby links, RNC requests Router C to host a standby link (Fig. 13, DBN left, FST right).

D. Union of Belief States and Policies

To combine policies and observations, we can assume that a policy is also a correlation between the event triggering it and the action enforced with a probability of 1. The union (see Figure 14) of all the observations plus the policies is a FST that models all the known correlation between events

[3]The first column of nodes in the DBN is in a first time slice and the second column in a second time slice. The representation can be rolled out over multiple time slices.

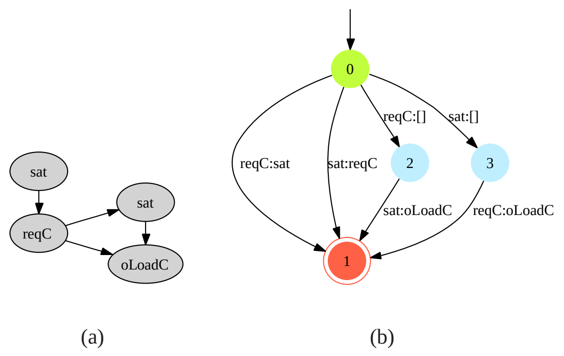

Fig. 10. Belief State 1.

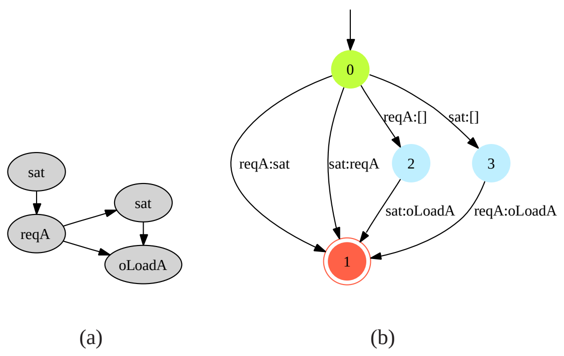

Fig. 11. Belief State 2.

and actions, those that are explicitly enforced by policies and those that are just observed because they are consequence of matters out of the management system's control, for example the "natural" behaviour of the network dictated by the control plane.

E. Composition Iteration

If we iterate composing the FST above with itself we obtain a view of what happens when the output of the system feeds back into the system again as its input. The first composition produces the FST in Figure 15. This figure shows two possible paths with repetitions in the input and the output ($(0) \longrightarrow reqA : reqA \longrightarrow (1)$, $(0) \longrightarrow reqC : reqC \longrightarrow (1)$). This repetition should be read as: due to the combination of policies and behaviour/context which is out of the management systems control, requesting the Router A to host a standby link will cause, after a chain of events and actions, the replication of the same request. By tracing back the pair of edges which generated the repetitive one, it is possible to deduce that this state of affairs occurs on Saturdays and because the enforcement of Policy 1.

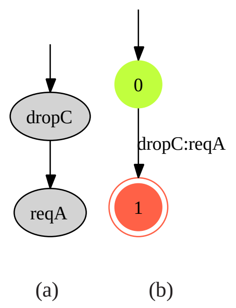

Fig. 12. Belief State 3.

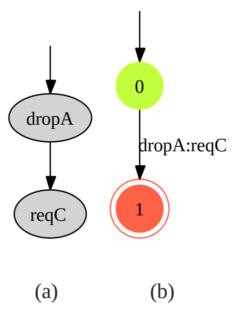

Fig. 13. Belief State 4.

V. Incorporating Probabilities

The example set out above presents the simple case where only deterministic behaviour (transitions where $p = 1$) is considered in the TFFST composition process. However, the probabilistic information encoded in the DBNs should be exploited rather than ignored. In the DBN-to-FST conversion process, each arc connecting $node_i$ and $node_j$ in the DBN is compiled out into multiple arcs of the FST which represent the transition from each value of $node_i$ to each value of $node_j$. The probabilities of these transitions are what is encoded in the conditional probability table of $node_j$. Therefore, each arc in the DBN FST has an associated probability of occurrence. The probabilities for individual arcs (learnt probabilities of DBN FST arcs and deterministic ($p = 1$) policy FST arcs) can be combined as part of the composition process. For paths which constitute flipflops or cycles of activity, the probability associated with any policies in this path can be interpreted as an evaluation of the risk associated with this policy. The probability may then be used by the system or a human operator to evaluate whether the risk presented by enforcing a given policy is worth taking.

VI. Conclusion and Future Work

The stability of distributed policy-based and self-management communication systems is an open issue which as yet has not been tackled on a large scale. Yet, a

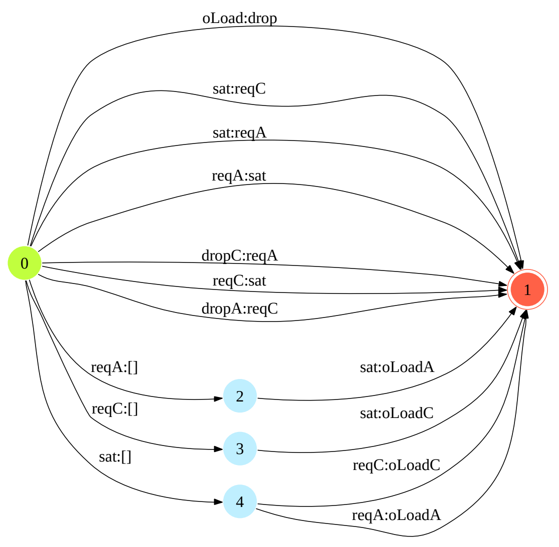

Fig. 14. A TFFST Union of Belief States and Policies.

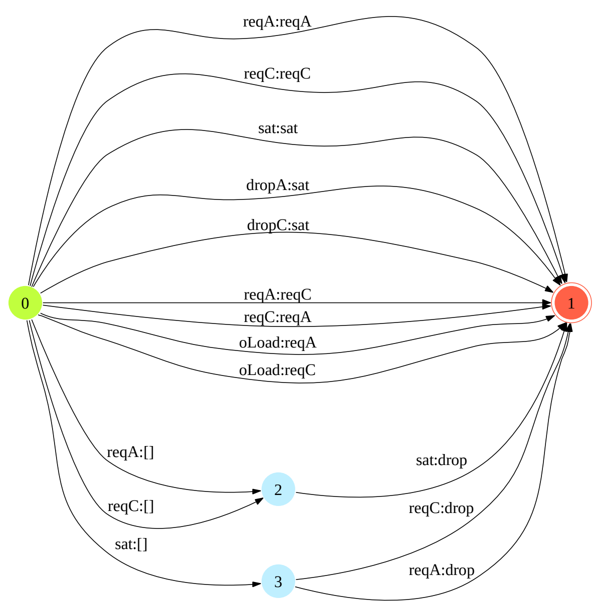

Fig. 15. First Composition of the union TFFST with Itself.

commercially viable self-management system must ensure such stability in order to realise the reduction in OPEX costs which a self-management system is intended to provide. This paper presents an analytic and implementable method to identify and hence reduce unstable behaviour in such systems and is a basic step towards a more general theory of stability in distributed policy-based systems. The method described here could be used both as an offline tool to assist the creation and update of policies using system wide data and on the network devices at runtime, using more accurate and up-to-date behavioural knowledge to avoid unexpected flip-flops or unbounded sequences (cascades) of reconfiguration actions. This analytic method is currently being prototyped as part of a distributed self-management system for 3G telecommunications nodes. The policy evaluation and enforcement functionality described here will be evaluated against a traditional policy engine baseline for processing speed and complexity and also relative to an overall system goal to maximise network resource usage.

References

[1] J. Baliosian, F. Sailhan, A. Devitt, and A.-M. Bosneag, "Automating knowledge and distribution for self-managed networks," in *Proceedings of Workshop on Distributed Autonomous Network Management Systems (DANMS 2006)*, Dublin, Ireland, 16 June 2006, to appear.

[2] J. Baliosian, H. Oliver, A. Devitt, F. Sailhan, E. Salamanca, B. Danev, and G. Parr, "Self-configuration for radio access networks," in *Proceedings of 7th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2006)*, London, Canada, June 5-7 2006, to appear.

[3] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, "Terminology for policy-based management," *RFC*, vol. 3198, Nov. 2001.

[4] J. Baliosian and J. Serrat, " Finite State Transducers for Policy Evaluation and Conflict Resolution ," in *Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, June 2004, pp. 250–259.

[5] C. Harris and V. Cahill, "Power management for stationary machines in a pervasive computing environment," in *Proceedings of the Hawaii International Conference on System Sciences*, 2005.

[6] G. Arroyo-Figueroa and L. Sucar, "A temporal bayesian network for diagnosis and prediction," in *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*. San Francisco, CA: Morgan Kaufmann Publishers, 1999, pp. 13–20.

[7] D. Heckerman, "Bayesian networks for knowledge discovery," in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI Press / The MIT Press, 1996, pp. 273–305.

[8] A. E. Nicholson, "Monitoring discrete environments using dynamic belief networks," Ph.D. dissertation, Department of Engineering, Oxford, 1992.

[9] M. Mohri, F. Pereira, and M. Riley, "Weighted automata in text and speech processing," in *Extended Finite State Models of Language: Proceedings of the ECAI'96 Workshop*, A. Kornai, Ed., 1996, pp. 46–50. [Online]. Available: citeseer.ist.psu.edu/mohri96weighted.html