Promise theory – a model of autonomous objects for pervasive computing and swarms

Mark Burgess and Siri Fagernes
Oslo University College, Norway

Abstract— The theory of promises describes policy governed services, in a framework of completely autonomous agents, which assist one another by voluntary cooperation alone. We propose this as a framework for analysing realistic models of modern networking, and as a formal model for swarm intelligence.

Keywords— **Pervasive computing, control theory, promise theory, swarm intelligence.**

I. Introduction

Traditionally IT managers and companies have attempted to control and govern computer systems using an authoratative, totalitarian approach, by commanding the component nodes in a system to behave according to a common plan from a central location. This model is becoming increasingly unrealistic as computing devices mutate into personal objects that users own, and which interact with their surroundings primarily according to their owner's wishes. In spite of the popularity of peer to peer services and personal consumer electronics, models of computer management are still struggling to break away from this idea of centralized control.

The legacy approach to modelling management behaviour is typified by control theory[1]. This is a form of analysis that stems from electrical engineering, where one imagines a network of components that drive one another in a predictable way. Downstream components of a control network are driven by their upstream counterparts, and the entire network is driven by a number of sources. Typically one is interested in regulating the value at the output, given some external disturbances. This approach is optimistic in its assumption that the components will behave as they are instructed, and we propose that this belief is unfounded in a scenario where personal *autonomy* is prevalent.

Promise theory is a new theory about what can happen in a network of entirely autonomous components[2], [3]. Rather than adopting the conventional belief that "only that which is programmed happens", it takes the opposite viewpoint: "only that which is promised can be predicted". It therefore approaches management from the viewpoint of uncertainty with realism rather than faith in compliance.

In promise theory, one assumes a *service viewpoint* of interactions between computing devices. Each node offers to play a part in a collaborative network by promising to constrain its behaviour in various ways. A typical use of promise theory is to examine the steady state behaviour of a number of autonomous agents (components). Promise theory does not necessarily fully determine the behaviour of its components, it only represents constraints within a set of defined behaviours. It is not limited to linear promises.

A principal difference between promise theory and control theory is that promise theory assumes the autonomy of its components, whereas control theory assumes the servility of its components: a promise network is not a driven network; rather it is one that operates (or not) by voluntary cooperation. Its components cannot be forced to perform their function.

Promise theory is thus better suited to model situations in which there are possibly irrational decision makers in a network, or where there are no explicit channels of control between the components (such as in swarm intelligence). A working representation in promise theory makes all assumptions about promised behaviour explicit and assumes no more than that which is promised. It turns the attention away from desired or presumed effect of the whole, towards the roles of the individual components in the network. It is therefore more closely related to reliability theory[4]. It will probably take years for computer scientists to accept that computers are probabilistic

II. Basic promise theory and policy based management

Promise theory can be a represenation of policy, from the viewpoint of autonomous agents—i.e. agents with 'freewill' codified into some kind of policy. Its strength lies, amongst other things, in making normally hidden assumptions explicit, so as to avoid conflicts of policy. Several approaches have been used in the past to resolve such conflicts[5], [6], [7], [8], [9], [10], [11]. Other formalisms that might be conisdered for describing policy include process algebras[12] such as the π-calculus[13] or Petri nets[14] or process algebras with probabilistic extensions[15]. However, these formalisms are more like control theory— they are about understanding events in fully specified systems, with sychronization and resource contraints. They do not capture the behaviour of a system given an imperfect knowledge or specification, such as that one usually faces in a management scenario. There is thus additional uncertainty to be considered.

In ref. [16], policy is identified as a specification of the average configuration of the system over persistent times. An important aspect of this definition is that it allows for error tolerances, necessitated by randomly occurring events that corrupt policy in the system management loop. There is a probabilistic or stochastic element to system behaviour and hence policy can only be an average property in general. We do not require a full definition of host policy here

from ref. [16]. It suffices to define the following.

Definition 1 (Agent policy) The policy of an individual computing device (agent or component) is a representative specification of the desired average configuration of a host by appropriate constraints[16].

Promise theory takes a service viewpoint of policy and uses a graphical language to compose system properties and analyse them. The promise theory manifesto is to begin with detailed, *typed* promises, from which the algebra of cooperation can be understood, and then gradually to eliminate the types through transformations which elucidate large scale structure and reliability.

A key feature of promises is that they make a clean separation between what is being constrained and to whom a constraint applies. This is another area in which confusions can arise in policy theory. The form of a promise is:

$$n_1 \xrightarrow{\pi} n_2 \tag{1}$$

where n_1 is the node making a promise, n_2 is the recipient of the promise and π is the typed promise body, which describes the interpretation and limitation of the promise. A promise body is a constraint on behaviour, asserted from one agent to another. We shall associate agents with components in order to discuss control theory. Promises fall into approximately four notable categories:

Promise type	Notation	Interpretation
Basic	$n_1 \xrightarrow{\pi} n_2$	Provide service/flow
Cooperative	$n_1 \xrightarrow{C(\pi)} n_2$	Imitate/Follow
Use	$n_1 \xrightarrow{U(\pi)} n_2$	Use/Accept from
Conditional	$n_1 \xrightarrow{\pi_1/\pi_2} n_2$	Promise 'queue': π_1 if π_2

Basic service promises form any number of types, e.g. 'provide web service in less than 5 milliseconds' or 'tell you anything about X'. Some other examples include:

- $X \xrightarrow{q \leq q_0} Y$: agent X promises to never exceed the limit $q \leq q_0$.
- $X \xrightarrow{q=q_0} X$: agent X promises to satisfy $q = q_0$.
- $X \xrightarrow{\ell \subseteq L} Y$: X promises to keep ℓ to a sub-language of the language L.
- $X \xrightarrow{S} Y$: agent X offers service S to Y.
- $X \xrightarrow{R} Y$: agent X promises to relay R to Y.
- $X \xrightarrow{\neg R} Y$: agent X promises to never relay R to Y.
- $X \xrightarrow{S,t} Y$: agent X promises to respond with service S to Y within t seconds.

They are somewhat analogous to service level agreements.

Cooperative promises are used in order to build concensus amongst individual agents with respect to a basic promise type. If π is a promise type, then $C(\pi)$ is a corresponding promise type between agents to collaborate or imitate one another's behaviour with respect to π.

A promise is said to be *broken* if an agent makes two different promises of the same type at the same time (this is different from a promise that has expired or been changed).

III. A simple comparison

Let us compare promise theory with control theory, so begin with a simple example. Consider how one would model a basic feedback loop consisting of a number of components.

A basic control theory network consists of a number of components (called controllers) connected together in a functional view: some observable value is injected into a network and a corresponding value is extracted elsewhere. The ingress of the observable is at the behest of an external agent; the egress occurs deterministically in the manner of a flow.

Control theory uses a scalar value with a compatible engineering dimension throughout (like an electric current or voltage). Promise theory, on the other hand, involves promises of several different types, based possibly on possibly structured information models. A single promise can be a declaration of intent to keep a general tuple of data within a given range of values. Promises are thus *typed* and their attributes can belong to discrete languages[17].

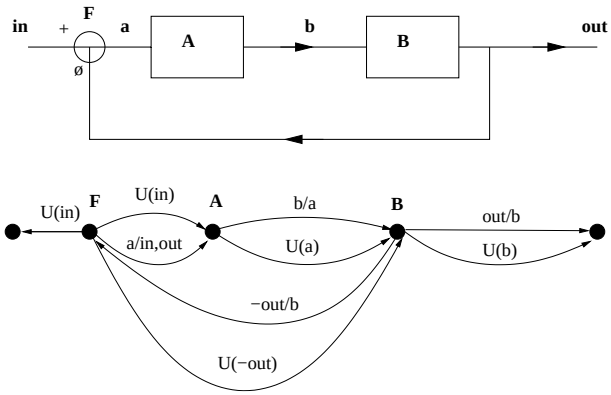

Fig. 1. Corresponding diagrams of control theory (above) and promise theory (below).

In the upper diagram, one represents the following deterministic relations, in real time parameterization:

$$\begin{aligned} A: \text{out}(t) &= \int B(\tau) b(t-\tau) d\tau \\ B: \quad b(t) &= \int A(\tau) a(t-\tau) d\tau \\ F: \quad a(t) &= \text{in}(t) - \text{out}(t). \end{aligned} \tag{2}$$

In the promise diagram, the basic service promises are:

- $F \xrightarrow{\pi_a} A$ where $\pi_a : a(t) = \text{in}(t) - \text{out}(t)$ if out is promised.
- $A \xrightarrow{\pi_b} A$ where $\pi_b : b(t) = \int A(\tau) a(t-\tau) d\tau$ if a is promised
- $B \xrightarrow{\pi_{\text{out}}} A$ where $\pi_{\text{out}} : \text{out}(t) = \int B(\tau) b(t-\tau) d\tau$, if b is promised.

There are several things to note about this rough translation. First of all, a promise looks very much like a service and (at this stage), a promise asserted is assumed to be a promise kept. However, the autonomy of nodes makes this diagram much more complicated than the control diagram, since there is no a priori obligation for the components to work together. Rather we must ensure this by explicit means.

The promises above are conditional promises: i.e. they can only be made if the agents or components making them are in receipt of a dependent promise. The so-called 'use' promises are required because we have turned the control scenario around from being mandatory compliance of components to voluntary cooperation. No autonomous component is actually obliged to play a role in this network, and each must assure its compliance to dependent interests by pledging to use the services promised to them in order to fulfill their own promise. This reflects a basic result in the theory:

Property 1 (Forward Dependency promise) A dependency promise requires the recipient of the service to receive assurance that the dependent service will be used.

$$n_1 \stackrel{\pi_d}{\rightarrow} n_2 \otimes \left(n_2 \stackrel{U(\pi_d)}{\rightarrow} n_3 \oplus n_2 \stackrel{\pi/\pi_d}{\rightarrow} n_3 \right) \;\Rightarrow\; n_2 \stackrel{\pi}{\rightarrow} n_3, \qquad (3)$$

where $\oplus$ represents the parallel composition of promises. We use the following shorthand for this:

$$n_1 \stackrel{\pi_d}{\rightarrow} n_2 \otimes n_2 \stackrel{\pi/\pi_d}{\not\rightarrow} n_3 \;\Rightarrow\; n_2 \stackrel{\pi}{\rightarrow} n_3\,. \qquad (4)$$

Example 1: For example, see fig. 2. Consider a webserver that depends on a database.

$$a \stackrel{\text{db}}{\rightarrow} b \otimes b \stackrel{\text{use-db}}{\rightarrow} c \oplus b \stackrel{\text{www/db}}{\rightarrow} c \;\Rightarrow\; b \stackrel{\text{www}}{\rightarrow} c\,. \qquad (5)$$

The server b makes a conditional promise to the client c which is verified by the agreement about database support from a, and thereby is turned into a real promise.

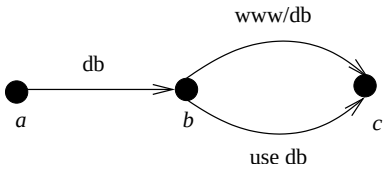

Fig. 2. Turning a conditional dependency into a real promise. The necessary structure is shown in graphical form. Here c receives the assurance that its supplier b will use a dependent service from somewhere, when offering a conditional promise.

This additional wiring might seem like an unnecessary burden, but in fact it is a liberating clarity. Many inconsistencies occur in policy systems from the assumption of a compliance that is not given. By making compliance explicit through declaration, we avoid hidden assumptions.

We note only briefly that other attempts to model compliance in systems of autonomous agents are quite unlike ours in that they generally assume the existence of an outside authority. See for instance the Law Governed Interaction work[18].

IV. Game theory: feedback

Game theory is about rational feedback relationships, in which agents have a selfish optimization interest. One of promise theory's useful properties is its close relationship with discrete bargaining games[19]. Control theory,

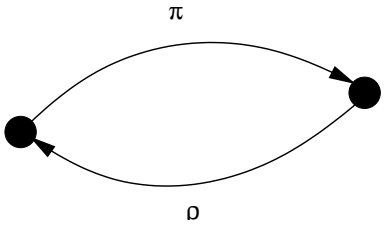

Fig. 3. A game is a pair of back-to-back promises of comparable value that reinforce one another, i.e. increase the probability of promises being kept.

on the other hand, can be related to differential optimization games (see for instance ref. [20]). Promise theory takes from game theory the view that reasonable solutions to the the receipt of feedback are those which form stable equilibria. It seems reasonable to seek stabilty in the management of a system. Certainly in evolutionary terms stability (i.e. survivability) is the criterion for success. This will also lead us to evolutionary cooperation of swarms below.

What is the value of a promise to its receiver? This is not immediately obvious. A promise issues a *constraint* on a variable, asserting that it will inhabit a subset of some hidden values; it has no direct 'value' in any form of scalar currency, as does a circuit in control theory. There is no voltage or current to measure *a priori*. Autonomous agents have, *a priori*, no common concept of universal currency. Each agent is free to decide on some valuation of a promise as it sees fit. The value of a promise could be a measured level of service quality, or any function of the service quality. Alternatively it could be a heuristic such as an arbitrary scale of 'business relationship value' or other subjective valuation. Such subjective valuations are the basis of economic markets, so we wish to allow for the general case. One can define two *types* of value in promise networks[21]:

Definition 2 (Received Value) Any function of the probability of keeping a promise, measured over the duration for which the promise is constant, in any currency/units. e.g. average quality of service over the time of a service level agreement. Only the recipient of a promise perceives this value.

This value is related to the perceived reliability of a promise sender, as measured by the receiver. It exists for a single promise, or for multiple promises.

A second kind of value can emerge through paired interactions. Once we allow agents to interact, the notion of value can becomes an interplay between peers:

Definition 3 (Bargaining Value) The value of mutually traded promises, in which the behaviour of a sender depends on its trading relationship with the receiver. This is the value of the game between the agents.

Trading of promises will turn out to be important to stability and reliability—i.e. whether agents choose to keep their promises. Clearly the exchange of value motivates the reliability where this depends on an agent's power to decide.

The value to Internet Service Providers in border peering promises is known to be based more on fostering good relations and business acumen than on technical necessity[22], [23]. One should therefore take seriously a multi-currency view of promise interactions.

A form of bargaining game theory is the iterative interaction[24], [25]. Iterative feedback systems are simple two-person continuum games, i.e. they are approximately differential games[20]. An example of this, applied to service level agreements, was presented in ref. [24].

Consider the first of our models (see fig 1); the two interdependent promises are:

$$\begin{aligned} n_1 &\stackrel{q=q_1(t)}{\longrightarrow} n_2 \\ n_2 &\stackrel{q=q_2(t)}{\longrightarrow} n_1 \end{aligned} \tag{6}$$

where,

$$\begin{aligned} q_1(t+1) &= \alpha_2 q_2(t) + \beta_2\, q_2(t) q_2(t-1) \\ q_2(t+1) &= \alpha_1 q_1(t) + \beta_1\, q_1(t) q_1(t-1) \end{aligned} \tag{7}$$

giving us two coupled equations for the time-development of the service level promises, and hence the average development of policy for the two peers.

The mutual agreements have two terms; these can be given a game-theoretical interpretation:

$$q_1(t+1) = \underbrace{\alpha_2 q_2(t)}_{\text{tit-for-tat}} + \underbrace{\beta_2\, q_2(t) q_2(t-1)}_{\text{tit-for-2tat}} \tag{8}$$

The tit-for-tat term says that we should try to reciprocate the service level that our peer gave us on the previous round. The quadratic term is like $q_1(t)$AND $q_1(t-1)$ says that we should add on a goodwill term depending on whether the peer has been faithful for the previous two rounds, i.e. we are rewarding its generosity.

This form of the game interaction emphasizes the connection with the theory of cooperation[26], [27].

V. Reliability

Promise theory starts by assuming that a promise given is always kept. Once this idealized view has been worked out, one can then look at transformations of the promise graphs, which swap the typed constraint labels for the *probabilities* of keeping the promises, i.e, reliability.

In control theory, stochastic effects in a system can be partly modelled by the use of 'disturbances', which are essentially noise modulation functions. However, there is no direct notion of how an unreliable component will impinge on a working system, nor of finding out where likely points of failure will occur.

Any typed graph can be represented as a set of adjacency matrices for each individual type. If we combine these individual graphs into a single matrix, however, by mapping each promise into the common currency of its reliability, we can find the component reliability network:

The value of the currency can be defined as any linear combination of functions which take a label and map it into the reals. Then the common currency graph has an adjacency matrix of the general form:

$$\begin{aligned} A_{ij} &= \alpha_1 f_1(n_i \stackrel{\pi_1}{\rightarrow} n_j) + \alpha_2 f_2(n_i \stackrel{\pi_2}{\rightarrow} n_j) + \dots \\ &= \alpha_1 f_1(A^1_{ij}) + \alpha_2 f_2(A^2_{ij}) + \dots \end{aligned} \tag{9}$$

where the promises range over whatever subclass or subset of promises we wish to consider. An obvious candidate for reliabilty studies is to make the common currency of the graph "probability that a promise is kept". In this case, the graph has the features of an epidemic process[**?**], [**?**], in which the function along each link if the probability if infection given that the original node is infected. One can then see how efficiently policy 'infects' the agents.

The purpose of a reduced graph is to create a convenient representation for applying graph spectral analysis to the relationships. Spectral (eigenvector) methods are very powerful ways of analysing the structure of relationships without requiring a complex semantic model.

VI. BGP: Autonomous Systems

An area of system management that control theory has difficulty addressing is that of the behaviour of a network which uses the Border Gateway Protocol (BGP)[28]. BGP is perhaps the only autonomous peer system in widespread use today, and it is frequently the lament of network analysts and designers. Rarely does a year pass without a conference paper trying to understand its consequences.

Why can't we understand BGP? It is our belief that the problem is the difficulty of predicting the behaviour of the autonomous systems within it. Since no party can force the autonomous areas within the global Internet to behave according to their rules, nor can anyone force the autonomous parties to repair broken policies, BGP breaks the mould of deterministic algorithms and hence breaks Computer Science's traditional arsenel of techniques for analysis. It is voluntary cooperation which leads analysts to fear BGP. Promise theory is perfect for this task, however and it is worth revisiting this old problem in some detail to discover the limits of BGP. This remains to be done. It seems clear already at the outset that, in the absence of any standards of practice or guidelines for BGP administration, it is relatively easy to make inconsistent promises, which therefore fail to produce functional, collaborative systems.

BGP's main tasks for an 'Autonomous System' (AS) are to advertise networks within its borders.and distribute proposed routing paths to neighbours. It involves implicit trust, and therefore intrinsic uncertainty. These tasks are

performed as services, with each AS having the power of veto. In BGP, filter lists and distribution lists are access control filters which set policy about who shall receive route advertisements or traffic. Incoming access control is clearly implemented by either having or not having either a 'use promise' or a conditional relay promise. Outgoing data transfer is implemented or not by having a data transfer promise.

BGP *communities* are labels that attach to traffic which suggests actions to be carried out by the remote agent. If all the customer nodes promise to follow the conventions laid out by their service provider agent, then they form a *appointed role* in the language of promise theory. Hence a BGP community is a role in promise theory. There is no corresponding notion in control theory, hence it seems that promises are uniquely suited to modelling this problem.

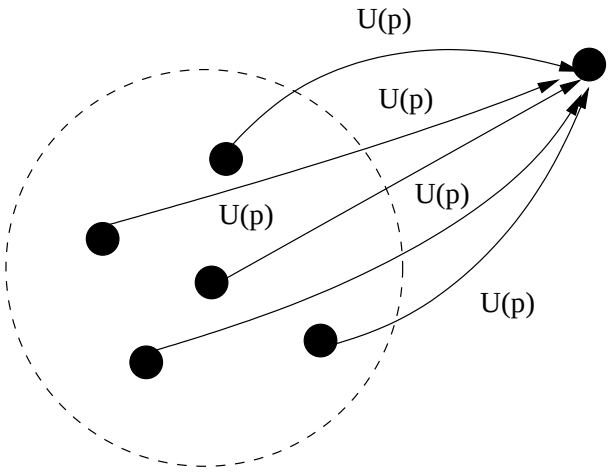

Fig. 4. An appointed role by mutual allegiance to use instructions from a single leader. (An instruction service promise from the leader to each client is not shown.) This view models the idea of a BGP community.

VII. Swarm intelligence

The study of swarm intelligence was made popular by the work of the Santa Fe Institute[29]. It has been used primarily to motivate algorithms that are based on the observed behaviour of collaborating insect workforces, such as ants and termites. Other forms of swarm behaviour include fish shoals and flocks of birds that navigate together.

Although the behaviour of certain insects has been emulated with some success in areas such as routing algorithms etc, one is missing a convincing model that explains swarm behaviour, with predictive power.

From an evolutionary perspective one would expect game theory to provide some of the answers[30]. If there is sufficient payoff or utility in a cooperative agreement, then it will survive, i.e. promises will be kept reliably. But a swarm of insects or a flock of birds seems to make no obvious promises within its collective, at least not in the sense of the agents providing a service to one another. So how shall one understand swarming from a promise perspective?

In fact, in our view, promise theory makes this very clear. Ants and insects often leave pheremone trails and other signals that guide the behaviour of their collaborators, i.e. they provide a service, and their collaborators promise to pay attention to this signal, which is passed through an environmental intermediary channel. Fish and birds, on the other hand, do not seem to do this at first glance. In fact, however, they make an implicit promise to be detectable. An agent that conceals its presence cannot form a swarm, but an agent that is visible can. Other agents can then promise to follow an agent if they see one. Thus a swarming agent *promises to be visible.* This results in an exchange of promises that canbe mapped onto a bargaining game.

When an agent can see many neighbouring agents and promises to follow each of them, there could be a conflict. We do not have time to discuss the resolution of this issue here. What is clear however is that there is an implicit agreement taking place even in a mutually navigating herd or swarm. An autonomous agent can choose to follow any other, or not, at its option: that is its autonomous right.

A swarm could be defined in many ways — as a spatial relationship in Euclidean space, as a trend or 'norm' in a more abstract space, or as a combination of these. In order to understand the phenomenon we must abstract its essence. We choose therefore to define a swarm as a collection of agents that promise to behave harmoniously, i.e. adjust their policies so that they behave similarly, if they receive a message from neighbours informing them of what they are about to do. For example, consider a simple case of two promises between an agent a and an agent b which exhibit behaviours X and Y:

- $a \rightarrow b$: do X if Y observed
- $a \leftarrow b$: show Y

and we assume that X is something that moves the agent closer to policy Y. If we now symmetrize the promises of a and b, then both agents have common policies that tend to make them imitate one another. This is a perfectly general observation that can be adapted to describe a variety of swarms. One can also include intermediaries that enable communication via environmental channels, or third parties.

This simple observation also allows us to model other scenarios as swarms, which we might not immediately associate with the concept. See for example, the pervasive computing scenario in ref. [31] (see fig. 5).

This diagram shows cooperative relationships forming bargaining games that keep pairs of agents (customers/sellers etc) togther in a shopping mall simulation. Without these game incentives, they would drift apart and have no reason to maintain this formation. According to the loose defintion above, this is a swarm, with local substructures that are also swarms—i.e. swarms within swarms. Promise theory therefore elucidates obvious properties of swarming behaviour in a natural way. For this reason, we believe it will be important in modelling not only computer networks but biological networks.

VIII. Conclusions

In control theory, nothing that is not stated happens. The boundedness of the output of a control network is a

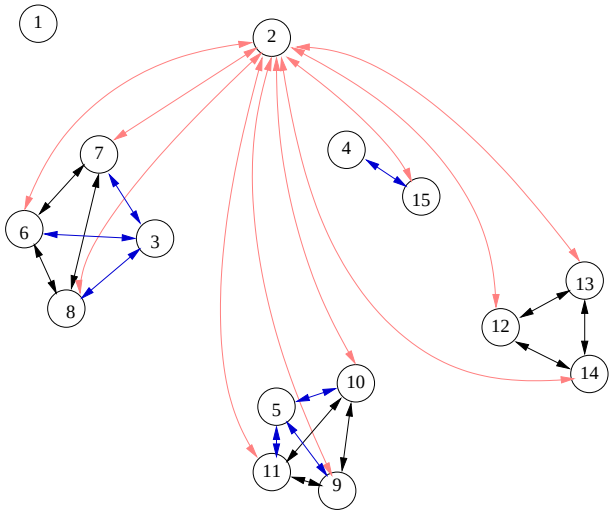

Fig. 5. The graph of various bargaining games within a shopping-mall network.

result of the properties of the components and the nature of the input. In promise theory, anything can happen at each agent node, and only the promises that are explicitly stated can bound or limit the possibly behaviour at some part of the promise network.

The connections in a promise network are not transport connections, but logical connections. If several components collaborate in altering a flow of data, transformations of the data occur both in the nodes between them. A transport connectivity is implicit and need not be stated. In control theory, the connections are transport links. Transformations take place in the components here also.

Promise theory seeks consistency of constraints by considering the logical algebra of the constraint composition. There is no automatic transitivity of promises: transitivity can only be arranged by explicit conditional transfer and 'use' promises. It is possible to model the scenarios of control theory using promise theory, by reformulating the graphs but the reverse is not true. However, if one knows that an authoratative control is present, there would be no particular advantage to using the promise theory framework: one would, at best, gain an understanding of the roles individual components play in network reliability.

In summary, we believe that promise theory has a promising future that is rich in potential applications.

This paper was supported in part by the EC IST-EMANICS Network of Excellence (#26854).

References

[1] J.L. Hellerstein, Y. Diao, S. Parekh, and D.M. Tilbury, *Feedback Control of Computing Systems*, IEEE Press/Wiley Interscience, 2004.

[2] Mark Burgess, "An approach to understanding policy based on autonomy and voluntary cooperation," in *IFIP/IEEE 16th international workshop on distributed systems operations and management (DSOM), in LNCS 3775*, pp. 97–108.

[3] M. Burgess and S. Fagernes, "Pervasive computing management: A model of network policy with local autonomy," *IEEE Transactions on Networking*, p. (submitted).

[4] A. Høyland and M. Rausand, *System Reliability Theory: Models and Statistical Methods*, J. Wiley & Sons, New York, 1994.

[5] E. Lupu and M. Sloman, "Conflict analysis for management policies," in *Proceedings of the Vth International Symposium on Integrated Network Management IM'97*. May 1997, pp. 1–14, Chapman & Hall.

[6] R. Ortalo, "A flexible method for information system security policy specifications," *Lecture Notes on Computer Science*, vol. 1485, pp. 67–85, 1998.

[7] J. Glasgow, G. MacEwan, and P. Panagaden, "A logic for reasoning about security," *ACM Transactions on Computer Systems*, vol. 10, pp. 226–264, 1992.

[8] H. Prakken and M. Sergot, "Dyadic deontic logic and contrary-to-duty obligations," in *Defeasible Deontic logic: Essays in Nonmonotonic Normative Reasoning*, vol. 263 of *Synthese library*. Kluwer Academic Publisher, 1997.

[9] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *Proceedings of the 5th IEEE Workshop on Policies for Distributed Systems and Networks*, 2004.

[10] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo, "Using event calculus to formalise policy specification and analysis," in *Proceedings of the 4th IEEE Workshop on Policies for Distributed Systems and Networks*, 2003.

[11] A.L. Lafuente and U. Montanari, "Quantitative mu-calculus and ctl defined over constraint semirings," *Electronic Notes on Theoretical Computing Systems QAPL*, pp. 1–30, 2005.

[12] J. Bergstra, A. Ponse, and S.M. Smolka, Eds., *The Handbook of Process Algebra*, Elsevier, 2001.

[13] J. Parrow, *An Introduction to the π-Calculus, in The Handbook of Process Algebra*, p. 479, Elsevier, Amsterdam, 2001.

[14] R. David and H. Alla, "Petri nets for modelling of dynamic systems — a survey," *Automatica*, vol. 30, pp. 175–202, 1994.

[15] B. Jonsson, W. Yi, and K.G. Larsen, *Probabilistic Extensions of Process Algebras, in The Handbook of Process Algebra*, p. 685, Elsevier, Amsterdam, 2001.

[16] M. Burgess, "On the theory of system administration," *Science of Computer Programming*, vol. 49, pp. 1, 2003.

[17] H. Lewis and C. Papadimitriou, *Elements of the Theory of Computation, Second edition*, Prentice Hall, New York, 1997.

[18] N. Minksy and P-P. Pal, "Law-governed interaction: A coordination and control mechanism for heterogeneous distributed systems," *ACM Transactions on Software Engineering and Methodology*, vol. 9, pp. 273–305, 2000.

[19] J.F. Nash, *Essays on Game Theory*, Edward Elgar, Cheltenham, 1996.

[20] R. Isaacs, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, J.Wiley and Sons (1965), Dover (1999), Toronto, 1965.

[21] M. Burgess and S. Fagernes, "Pervasive computing management: Policy through voluntary cooperation," *IEEE Transactions on Networking*, p. (submitted).

[22] W.B. Norton, "The art of peering: The peering playbook," Tech. Rep., Equinix.com, 2001.

[23] W.B. Norton, "Internet service providers and peering," Tech. Rep., Equinix.com, 2001.

[24] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes, "Summary of the stability of adaptive service level agreements," in *Proceedings of the 6th IEEE Workshop on Policies for Distributed Systems and Networks*. 2005, pp. 111–114, IEEE Press.

[25] K. Begnum, M. Burgess, T.M. Jonassen, and S. Fagernes, "On the stability of adaptive service level agreements," *eTransactions on Network and System Management*, vol. (in press), 2006.

[26] R. Axelrod, *The Evolution of Co-operation*, Penguin Books, 1990 (1984).

[27] R. Axelrod, *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*, Princeton Studies in Complexity, Princeton, 1997.

[28] T.G. Griffin and G. Wilfong, "On the correctness of ibgp configuration," *ACM SIGCOMM'02)*, 2002.

[29] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Oxford, 1999.

[30] J. Maynard-Smith, *Evolution and the Theory of Games*, Cambridge University Press, Cambridge, 1981.

[31] M. Burgess and S. Fagernes, "Pervasive computing management: Applied promise theory," *IEEE Transactions on Networking*, p. (submitted).