On the Design of Distributed Protocols from Differential Equations *

Indranil Gupta
Dept. of Computer Science
University of Illinois (Urbana-Champaign)
indy@cs.uiuc.edu

ABSTRACT

We propose a framework to translate certain subclasses of differential equation systems into distributed protocols that are practical. The synthesized protocols are state machines containing probabilistic transitions and actions, and they show equivalent stochastic behavior to that in the original equations. The protocols are probabilistically scalable and reliable, and are derived from two subclasses of equations with polynomial terms. We prove the equivalence of protocols to the source equations. Rewriting techniques to bring equations into the appropriate mappable form are also described. In order to illustrate the usefulness of the approach, we present the design and study of scalable and probabilistically reliable protocols for migratory replication and majority selection. These two protocols are derived from natural analogies represented as differential equations - endemics and the Lotka-Volterra model of competition respectively. Well-known epidemic protocols are also shown to be an output of the framework. We present mathematical analysis of the protocols, and experimental results from our implementations. We also discuss limitations of our approach. We believe the design framework could be effectively used in transforming, in a very systematic manner, well-known natural phenomena into protocols for distributed systems.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Distributed Systems*

General Terms

Algorithms, Design, Reliability

Keywords

Science of Protocol Design, Distributed Protocols, Scalability, Reliability, Endemic Protocols, Probabilistic protocols.

1. INTRODUCTION

Much attention has been paid to studying the scalability and reliability of protocols for peer to peer systems, the Grid, etc. However, the *science of design* of efficient and reliable protocols for large-scale distributed systems remains a difficult problem. In this paper, we describe a framework to translate sets of differential equations (called a "system of differential equations", or abbreviated as "equation system") into a protocol for distributed systems. The techniques generate a state machine where states are derived from variables in the original equations, and actions and transitions are derived from the terms in the original equations. The source equation systems are required to be in a certain form - we describe this through a taxonomy, and also present equation rewriting techniques to enable conversion of other equation systems to the appropriate mappable form.

In order to illustrate practical utility of the presented methodology, we present two case studies where natural processes represented as differential equations are used to design practicable protocols for a dynamic and migratory replication scheme and for probabilistic majority selection. The two protocols are respectively called endemic replication and the Lotka-Volterra (LV) protocol.

Previously, differential equations have been used to study and analyze algorithmic solutions to problems such as independent vertex sets and degree distributions [24], 3-SAT [1], randomized load balancing [18], etc. The analyzed solutions are usually randomized algorithms. However, little of this work has addressed the converse direction, i.e., a design methodology to translate differential equations into distributed protocols. Such a methodology can be invaluable because scientists and engineers express their ideas and results by using the language of differential equations. These ideas can then be systematically converted into distributed protocols that consequently have well-known behavior (by virtue of the original equations).

Our protocols can be seen as probabilistic I/O automata [25], however the cited work gives a broad specification of protocols, rather than a framework for building and analyzing protocols. Several randomized protocols have been proposed to circumvent the FLP impossibility of consensus result [10], starting from Rabin's work in [19]. Distributed computing with infinitely many processes, and the relation to finite groups, was studied in [15, 17, 18]. Strogatz's textbook [23] gives a thorough coverage for the analysis of nonlinear dynamic systems.

The protocols generated by our framework have the common characteristics that they offer probabilistic reliability, utilize low and scalable bandwidth at each process, and have

*Please see Errata list on page 11 of this paper.

PODC'04, July 25–28, 2004, St. Johns, Newfoundland, Canada.

low convergence times. Most importantly, the distributed protocol inherits the stochastic behavior of the original differential equation system, e.g., the existence of stable equilibrium points in the original equations implicitly map into self-stabilizing behavior in the protocol. The mathematical analysis of these protocols borrows techniques from the study of non-linear dynamics. Although not new, our particular style using *phase portraits* and simple perturbation analysis is appropriate for our goals, and has not been used before to study distributed protocols. The protocols are intended for asynchronous network settings, however our analysis makes certain simplifying assumptions.

In summary, the contributions of this paper are:
(i) A novel framework to translate differential equations, from two subclasses, into equivalent distributed protocols;
(ii) A natural taxonomy of differential equations, and equation rewriting techniques that facilitate translation;
(iii) Use of this framework to design new scalable protocols for majority selection and responsibility migration; and
(iv) Use of analytical techniques from the study of non-linear dynamics in analyzing distributed protocols.

System Model: We assume a closed group G of N processes, communicating over an asynchronous network. The closed group assumption means that there are no joins by new processes; simulations show that our protocols work in open groups. Each process may suffer from crash-stop or crash-recovery failures. The communication medium is unreliable, and can drop messages or connections. For simplicity, our analysis assumes that clock drifts are bounded; however our final results hold for the *average* clock speed in the group, allowing the protocols to run among processes with unsynchronized clocks. Each process also knows about the maximal group membership, i.e., the other $N-1$ processes [1]. Our protocol analysis assumes that the group size N is large enough so that variables that denote the fraction of processes in a given state of a finite state machine can be assumed to be continuous. This also enables us to assume that variation of these variables is continuous in time (as opposed to discrete).

A Motivating Example - Epidemics: Epidemic spread of diseases and rumors can be represented through differential equations [3]. In a closed group (i.e., no participant joins) of N participants, let the fraction of infected and susceptible processes be y and x respectively; $x = 1 - y$. Then:

$$\begin{aligned} \frac{dx}{dt} = \dot{x} &= -xy \\ \frac{dy}{dt} = \dot{y} &= xy \end{aligned} \tag{0}$$

A *pull* epidemic protocol can be designed from these equations. This is done by creating a state machine with two states - x (susceptible) and y (infected). The actions in the state machine involve susceptible processes periodically sampling the group for infectives (protocol period fixed at all processes), giving exactly the canonical epidemic protocol, variants of which are used in systems such as Clearinghouse [9]. For unfamiliar readers, the canonical epidemic pull algorithm works by having each process p that has not received the multicast (i.e., is susceptible) periodically contact one other process selected uniformly at random from the group. If the remote process has received the multicast (i.e., is infected), it sends the multicast to p, which then turns infected. The analysis predicts that as $t \to \infty$: $x(t) \to 0, y(t) \to N$, and it takes $O(log(N))$ rounds to reach $x \simeq O(1)$.

We remind the reader that epidemics are only a motivating example, and are *not* the focus of this paper; epidemics were analyzed by Demers et al. in [9].

Section 2 gives a taxonomy of differential equation systems, and Sections 3, 6, present mapping techniques. Sections 4 and 5 show how the framework can be applied, and present experimental results. Section 7 presents equation rewriting techniques, and Section 8 concludes.

2. A TAXONOMY OF DIFFERENTIAL EQUATION SYSTEMS

In order to clearly classify the differential equation classes that can be converted into distributed protocols, we first describe a taxonomy of differential equation systems. We first consider systems of equations that have a single differential per equation, and are of order and degree 1. Section 7 discusses translations for equations of higher order and degree.

Let X denote a set of m independent variables. Let $\bar{X}$ be a finite-sized vector of these variables. We are concerned only with systems of differential equations in the form $\dot{\bar{X}} = \bar{f}(\bar{X})$. Here, $\bar{f}$ refers to an m-sized vector of functions (each in m variables), and $\dot{\bar{X}}$ refers to a vector of the m variables in $\bar{X}$, each differentiated once with respect to time, i.e., $\dot{\bar{X}} = \frac{d}{dt}(\bar{X})$. We also denote the equation for variable $x \in X$ as $f_x(\bar{X})$. Each of these equations is thus of order 1 (highest derivative) and degree 1 (power of highest derivative).

We define two properties of equation systems:

Complete Equation Systems: An equation system $\dot{\bar{X}} = \bar{f}(\bar{X})$ is said to be *complete* if and only if $\Sigma^{x \in X} \dot{x} = \Sigma^{x \in X} f_x(\bar{X}) = 0$. In other words, the right hand sides all sum to zero. Without loss of generality, we will henceforth assume $\Sigma^{x \in X} x = 1$.

Completely Partitionable Equation Systems: An equation system $\dot{\bar{X}} = \bar{f}(\bar{X})$ is said to be *completely partitionable* if and only if (i) it is complete, and (ii) all terms occurring in $\bar{f}(\bar{X})$ can be grouped into pairs so that each pair sums to zero. For example, equation system (0) is completely partitionable.

Based on the nature of $\bar{f}$, we can define the following two subclasses of equation systems:

Polynomial: For each equation $\dot{x} = f_x(\bar{X})$ in the system, $f_x(\bar{X})$ can be written as a sum of polynomial terms. Each term T is of the form $\pm c_T \Pi^{y \in X} y^{i_{y,f_x,T}}$, where all $i_{y,f_x,T}$'s are non-negative integers. c_T is a positive constant specific to the term.

Restricted Polynomial: An equation system $\dot{\bar{X}} = \bar{f}(\bar{X})$ is restricted polynomial if it is polynomial, and for each $f_x(X)$, it is true that each negative term $-T = -c_T \Pi^{y \in X} y^{i_{y,f_x,T}}$ that occurs in it ($c_T > 0$) has $i_{x,f_x,T} >= 1$.

3. MAPPING AN EQUATION SYSTEM THAT IS POLYNOMIAL AND COMPLETELY PARTITIONABLE

The key idea in translating a given different equation system that is completely partitionable, into a distributed pro-

[1] Well-known results can be used to reduce this size to logarithmic in group size.

tocol, involves creating a state machine that contains (a) one state per basic variable in the original equation system, and (b) *actions* mapped onto these states. We denote the corresponding state for a given variable x simply as "state x".

Behaviorally, a given variable x is mirrored in the protocol as the fraction of processes in the system that are in state x. Each term in the equation system is mapped to a set of actions that ensure there is a corresponding rate of outflow (or inflow) of processes in the distributed group of processes.

3.1 Generating Actions from Terms

Below, we describe two techniques to convert terms into actions. All actions are executed periodically, once at the beginning of every protocol period. The protocol period duration is fixed at all processes. Protocol periods start at arbitrary times at different processes. Although we assume all clock drifts are negligible, our analysis holds for average period across the group. The protocols do not require either global clocks, or global synchronization, or agreement.

Flipping: A term of the type $-c.x$ (where c is a constant) occurring on the right hand side (r.h.s.) of $\dot{x} = f_x(\bar{X})$ is mapped to a flipping action. A process in state x *periodically* (i.e., once every protocol period) and *locally* tosses ("flips") a biased coin that has heads probability $p.c.$ p is a normalizing constant chosen throughout the system so that $p \leq 1, pc \leq 1$. The process transitions out of state x (and into the corresponding state y where $\dot{y} = f_y(\bar{X})$ contains the corresponding $+c.x$ term, and $y \neq x$) only when the coin turns up heads.

One-Time-Sampling: A term $-T$ of the type $-T = -c.x^{i_{x,f_x,T}}.\Pi^{y \in X-\{x\}} y^{i_{y,f_x,T}}$ occurring on the r.h.s. of $\dot{x} = f_x(\bar{X})$ with $i_{x,f_x,T} >= 1$ is mapped into a periodic action (executed once every protocol period) as follows. A process in state x periodically samples $(i_{x,f_x,T} - 1 + \Sigma^{y \in X-\{x\}}(i_{y,f_x,T}))$ other processes uniformly at random from across the group. In addition, the process also flips *locally* a biased coin that has heads probability $p.c$, where p is the same normalizing constant as above chosen so $pc \leq 1$. Let the variables $y \in X$ be ordered lexicographically. Then the process makes a transition out of state x (and into the corresponding state with the $+T$ term) if and only if (a) each of the first $i_{x,f_x,T} - 1$ target choices happen to be in state x; and (b) for each $j, 1 \leq j \leq \Sigma^{y \in X-\{x\}} i_{y,f_x,T}$, the j^{th} process sampled is in the same state as the j^{th} variable in $\Pi^{y \in X-\{x\}} y^{i_{y,f_x,T}}$ (when ordered lexicographically); and (c) the flipped local coin falls heads [2].

Theorem 1: Flipping and One-time Sampling are sufficient in mapping equation systems that are restricted polynomial and completely partitionable, into distributed protocols that have equivalent behavior in infinite sized groups.
Proof: See [12].

Message Complexity: The message complexity of the protocol is bounded by the size of the original equations. The number of sampling messages sent out by a process in state x, per protocol period, equals the sum of the number of occurrences of all variables in negative terms in $f_x(\bar{X})$, less the number of negative terms in $f_x(\bar{X})$.

The Effect of Failures: Message delivery losses and process failures modify the equation that is modeled by the protocol. If fractions $x \in X$ are fractions of alive processes in respective states, each original term T for a one-time-sampling action takes on a multiplicative factor of $(\frac{1}{1-f})^{|T|-1}$, where f is the group-wide failure rate per connection attempt, and $|T|$ is the total number of variable occurrences in term T. In order to faithfully model the original equations for a system with a known f, it is enough to increase the heads probability of the flipped coin for one-time-sampling terms by a multiplicative factor of $(\frac{1}{1-f})^{|T|-1}$. The normalizing constant p may need to be reduced so that in all the terms, the biased coins' heads probability is smaller than 1.0.

4. TRANSLATION CASE STUDIES

We explore two case studies to demonstrate application of the design framework. This section describes protocols for probabilistic responsibility migration and probabilistic majority selection.

4.1 Case Study I: Responsibility Migration

We define a responsibility migration problem that aims at selecting a subgroup of processes from the group and migrating membership of this subgroup. The subgroup members could be used to share responsibility for a task, e.g., storing replicas of a given file, buffering multicasts received by the group, running consensus, etc. In this paper, we discuss the responsibility migration problem in the context of partial replication of files, and specifically, for a persistent distributed file system. The problem can be formally stated as follows:

Distributed Responsibility Migration Problem:

At any point of time in a group G of processes, each process is either a *responsible* process, or not. A non-responsible process can turn responsible only after contacting one that is responsible. Further,

`Safety:` The number of responsible processes in the system never becomes zero.

`Liveness:` A process that is currently responsible will eventually become non-responsible.

`Fairness (optional):` Over a long time of running, each process in the system bears responsibility for an equal fraction of time.

For a persistent distributed file system application (e.g., a concept similar to the eternity storage service introduced by Anderson in [2]), where each object is a file, each file has a responsibility migration protocol running on its behalf. At any time, the responsible processes for a file are the only ones storing replicas of the file. `Safety` ensures that a file, once inserted, never disappears from the system. `Liveness` ensures that a responsible process deletes the file eventually[3].

Partial data replication has been studied for many years, e.g., in databases [9, 11], email and file systems [14, 21, 22], and more recently, in peer to peer systems [6]. Gray et al. [11] argue against the dangers of large-scale replication and warn that maintaining consistency among a large number of replicas might counter the goal of scalability. Replication has two tasks: (a) replica location (placement), and (b) replica management. Replica location decides, for a given

[2]The idea behind this condition is of course similar to that behind the well-known *Law of Mass Action*. Flipping is in fact a special case of One-time-sampling; we differentiate the two for purposes of clarity.

[3]This does not preclude the process from becoming responsible again at some later point of time.

object, "how many" and "where" replicas of the object are located. Replica management deals with how replicas are accessed and updated in a consistent manner, e.g, active and passive replica management are well known techniques [7].

In this paper, we focus only on replica location - these strategies can be combined with appropriate replica management strategies, although such issues are the subject of a future article. Most existing solutions to replica location [20, 21] locate replicas *statically* and *reactively*, i.e., the subset of hosts selected to hold replicas of a given object does not change unless a special event happens, e.g., one of the hosts could crash. This has the following three disadvantages: (1) they can be expensive in systems containing millions of hosts, where a large fraction have short lifetimes (O(several minutes)) and rejoin multiple times (6.4 times/day as reported in the Overnet system [4]) [4]; (2) From a security stand-point, static and reactive strategies allow an attacker to easily locate and attack all the individual replicas of an object. A malicious host could track the current replicas for a given object, and then subject each of them simultaneously to a kind of directed attack (e.g., a DOS attack on each host, sustained until the host crashed, would suffice) in such a way that all copies of the object are destroyed; (3) Static and reactive strategies attempt to satisfy safety but neither liveness nor fairness.

Dynamic and migratory replication strategies can avoid the above drawbacks, and provide other interesting properties. These strategies proactively move replicas of the object among different hosts in the system. Thus, an attacker finds it difficult to locate all the replica hosts, and even if the attacker does locate them, it has only a short window before the subgroup membership changes. Further, the availability of the object is not affected by either short host availability periods or massive failures in the system. Contrary to intuition, endemic protocols can offer scalable and reliable behavior w.r.t. liveness, safety, and fairness, while incurring only a constant message overhead at each process in the group. This paper focuses on the scalability and performance of migratory replication strategies and not on a security analysis.

Theorem 2 *(Impossibility of achieving* `Safety`*)*: No responsibility migration protocol can achieve `Safety`. This is because there exists a run where at some instant of time, all responsible processes crash simultaneously.

4.1.1 A Simple Solution, and its Drawback

Consider a replica migration protocol where a process storing an object replica hands it off to another process after a while and immediately deletes the object. A crash-stop failure of the former process before the transfer effectively destroys an object replica. Over time, the number of replicas of a given object will then go down to zero (unless there is a periodic refresh).

4.1.2 Endemic Protocol

We present an *endemic protocol* for probabilistic responsibility migration, designed from differential equations for endemic infectious diseases in a closed human population.

[4]One could restrict replicas to be stored only on hosts with at least a threshold availability [5], but this does not address the issue (2).

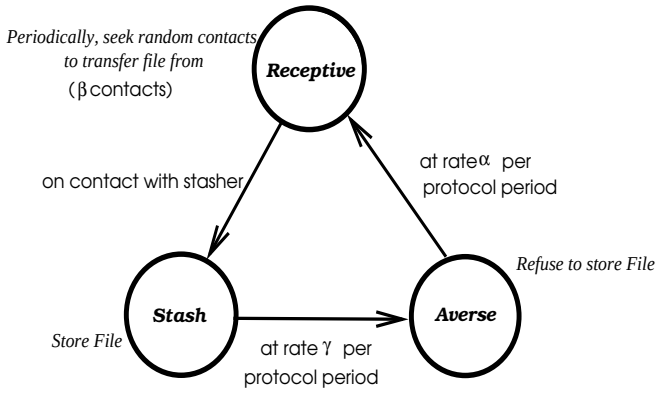

Figure 1: *An Endemic Protocol.*

This natural analogy is appropriate because the desired behavior of the protocol bears resemblance to the persistent survival of folklores in human society, and common cold in human populations, both for centuries and in a manner that resists the death of individuals. Let x, y, z be the fractions of susceptible, infected, and immune individuals, respectively. We consider a new variant of endemic equations:

$$\begin{aligned} \dot{x} &= -\beta xy + \alpha z \\ \dot{y} &= \beta xy - \gamma y \\ \dot{z} &= \gamma y - \alpha z \end{aligned} \tag{1}$$

Here, β, γ, α respectively stand for the rates of infection, recovery, and susceptibility. α and γ each lie in the interval $(0, 1]$. We assume that β is a small even integer, and $\beta > \gamma$.

This equation system is restricted polynomial and completely partitionable. Using the framework of Section 3, the state machine derived has three states - x (susceptible or *receptive*), y (infected or *stash*), z (immune or *averse*). *A process is responsible if and only if it is in the stash state.* The protocol, depicted in Figure 1, is derived using the rules in Section 3.

The protocol uses a parameter $b(= \beta)$. State actions are executed periodically, i.e., once every protocol period. (i) (γy term) A process p in the stash state periodically tosses a coin with a biased heads probability γ - if the coin falls heads, process p changes its state to averse. This transition is accompanied by a deletion of the object replica at p. (ii) (αz term) A process p in the averse state periodically tosses a coin with heads probability α, and changes state to receptive if this coin falls heads up. (iii) (βxy term) A process p in the receptive state periodically chooses b targets uniformly at random, and if any of these targets is in the stash state, process p changes its state to stash (after an object transfer).

In order to make the protocol more efficient, we add a fourth action and modify $b = \beta/2$. (iv) (βxy term) A process p in the stash state periodically chooses b processes uniformly at random; any of the selected target processes that is receptive immediately transitions to the stash state (after an object transfer). This does not change the differential equations modeled. β is the contact rate and is $= N(1 - (1 - \frac{b}{N})^2) \simeq 2b$. One can also account for aborted file transfers with an extra multiplicative factor for b.

The third averse state ensures that there is a time interval after a process deletes a replica when it will not store the file again. As we will see in Section 5, this helps the protocol

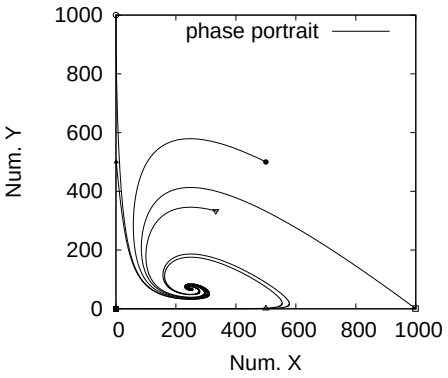

Figure 2: *Phase Portrait of an Endemic Protocol - Stable Spiral: The plot shows the phase portrait obtained by simultaneously plotting the numbers of susceptibles and infectives* $(X, Y) = (Nx, Ny)$ *over time, in a system with* $N = 1000$ *processes. This plot shows the system started at these initial points -* (X, Y, Z)*=blank square (999,1,0), dark square (0,1,999), blank circle (0,1000,0), dark circle (500,500,0), blank triangle (500,1,499), dark triangle (1,500,499), and blank inverted triangle (333,333,334). For the parameter setting* $N = 1000, \alpha = 0.01, \beta = 4, \gamma = 1.0$*, the non-trivial equilibrium point above is a stable spiral.*

perform even when some processes are chronically averse.

4.1.3 Endemic Protocol Analysis

We analyze the endemic protocol[5], with the goal of demonstrating how the generated protocols are to be studied. First, we note that the endemic protocol of Figure 1 satisfies `Liveness` if $\gamma > 0$. Since the protocol is symmetric across processes, it also satisfies `Fairness`. The probabilistic safety of the protocol is discussed at the end of this section.

Equilibrium in the endemic equations occur when $(\dot{x}, \dot{y}, \dot{z}) = 0$. Substituting in the above equations gives us two equilibria: $(x_\infty, y_\infty, z_\infty) = (N, 0, 0)$
$(x_\infty, y_\infty, z_\infty) = (\gamma/\beta, \frac{N-\gamma/\beta}{1+\gamma/\alpha}, \frac{N-\gamma/\beta}{1+\alpha/\gamma})$ (2)

The first equilibrium results in all copies of the object disappearing; all processes eventually turn receptive. The second equilibrium is more desirable, since it guarantees `Safety`. We study stability and convergence properties of these equilibrium points. Since an explicit solution is difficult, we resort to an indirect analysis.

Are the Equilibria Self-Correcting? We study this through a perturbation analysis. Let us assume a small deviation from the equilibrium point. Let us start the system in the state:

$$(x_0, y_0, z_0) = (x_\infty(1+u), y_\infty(1+v), z_\infty(1+w))$$

where $0 \leq u, v, w \ll 1$. Substituting this into equations (1) and simplifying, we get:

$$\dot{u} = \frac{\beta N - \gamma}{1+\gamma/\alpha}(w - u - v) \quad (3a)$$
$$\dot{v} = \gamma u \quad (3b)$$
$$\dot{w} = \alpha(v - w) \quad (3c)$$

where the uv term is ignored (as $u, v \ll 1$), and where we have assumed that $y_\infty = \frac{N-\gamma/\beta}{1+\gamma/\alpha} \neq 0$. Eliminating w and v from equations (3(a-c)), and introducing a new variable t, we get in matrix form:

$$\dot{T} = A \cdot T, \quad (4)$$

where $\sigma = \frac{\beta N - \gamma}{1+\gamma/\alpha}$ and

$$T = \begin{bmatrix} t \\ u \end{bmatrix}, \qquad \mathrm{A} = \begin{bmatrix} -(\sigma+\alpha) & -\sigma(\gamma+\alpha) \\ 1 & 0 \end{bmatrix}$$

Lemma 1: The only equilibrium point in equation system (4) is $(t, u) = (0, 0)$.
Proof: $\dot{u} = 0 \Rightarrow t = 0$, and then $\dot{t} = 0 \Rightarrow u = 0$.

Corollary: The equation system (3(a,b,c)) has only one equilibrium point $(u, v, w) = (0, 0, 0)$. This is easy to see by substitution of the equilibrium value $u = 0$ into equations (3(b,c)).

Theorem 3 *(All paths lead to the Second Equilibrium)***:** For the system of differential equations (1), the equilibrium point $(x_\infty, y_\infty, z_\infty) = (\gamma/\beta, \frac{N-\gamma/\beta}{1+\gamma/\alpha}, \frac{N-\gamma/\beta}{1+\alpha/\gamma})$ is *always stable*, if both $\alpha, \gamma > 0$ and $N > \frac{\gamma}{\beta}$.
Proof: The stability of the equilibrium point $(u, v, w) = (0, 0, 0)$ depends on the trace and determinant of matrix A. If the trace is negative and the determinant is positive, the equilibrium point is stable. However, if the trace and determinant were both positive, the equilibrium would be unstable. Finally, if the determinant is negative, the equilibrium is a saddle point (i.e., some trajectories in the vicinity converge to it, and the rest diverge) [23].

We can calculate:

τ = trace(A) = sum of leading diagonal elements in A $= -(\sigma + \alpha)$, and

$$\Delta = \det(A) = \sigma(\gamma + \alpha) \quad (5)$$

By our choice of parameters, $\sigma > 0$ and $\alpha, \gamma > 0$. Therefore, we *always have that* $\tau < 0, \Delta > 0$. This means that the solution of $u = 0$ for equations (4) is *always stable*.

From equations (3(b,c)), Lemma 1 and the corollary, we can say that any perturbations ((u, v, w) in the values of (x, y, z)) around the second non-trivial equilibrium point will die out, and the system is self-stabilizing around the second equilibrium point. □

Corollary: If $N < \frac{\gamma}{\beta}$ and $\alpha, \gamma > 0$, then given that $x, y, z > 0$, the system of equations (1) has the (stable) equilibrium point $(N, 0, 0)$.
Corollary *(First Equilibrium is a Saddle Point)***:** If $\alpha, \gamma, \sigma > 0$ and $N > \frac{\gamma}{\beta}$, the first equilibrium point $(x, y, z) = (N, 0, 0)$ is a *saddle point*. This means that it is partly stable; as long as y remains 0, the system converges back to the equilibrium. However, inclusion of even a single stasher will drive the system towards the second, more stable equilibrium with a larger, non-zero number of replicas for the object.

Convergence Complexity (Definition): The convergence complexity for an equilibrium point P of a protocol state machine with m states is an $m-$sized vector of functions in variable t that describe the variation of fractions of processes in the m respective states, if the system is started in some neighborhood of point P.

For endemics, the nature of the trajectories around the

[5]To the best of our knowledge, equation (1) has not been analyzed elsewhere in the literature.

second stable equilibrium point, as well as the time taken by the protocol for convergence, both depend on the eigenvalues and eigenvectors of the matrix A. The eigenvalues of A are $\lambda_1 = \frac{\tau+\sqrt{\tau^2-4\Delta}}{2}$, and $\lambda_2 = \frac{\tau-\sqrt{\tau^2-4\Delta}}{2}$.

From equations (5) in the last section, we can calculate and simplify:

$\tau^2 - 4\Delta = (\frac{\beta N-\gamma}{1+\gamma/\alpha} - \alpha)^2 - 4\frac{\beta N-\gamma}{1+\gamma/\alpha}\gamma$

Three cases arise:

1. $\tau^2 - 4\Delta < 0$ (eigenvalues distinct and complex): The variation of the displacement u in the number of susceptibles x, as a function of time, can be calculated as:

$u = u_0 e^{-\frac{t(\sigma+\alpha)}{2}} cos(t\sqrt{\sigma\gamma - \frac{(\sigma-\alpha)^2}{4}})$

where u_0 is the initial value of u. Notice that with time, u decreases exponentially fast to 0. The cosine term causes a (damped) oscillation in the value of u. This leads to a *stable spiral*, which means that the convergence takes the form of a damped oscillation.

2. $\tau^2 - 4\Delta > 0$ (eigenvalues distinct and real): The variation of u as a function of time is given by

$u = \frac{\dot{u_0}-\lambda_2 u_0}{\lambda_1-\lambda_2} e^{t\lambda_1} + \frac{\dot{u_0}-\lambda_1 u_0}{\lambda_2-\lambda_1} e^{t\lambda_2}$

where u_0 and $\dot{u_0}$ are the initial values of u and $\dot{u}$ respectively.

3. $\tau^2 - 4\Delta = 0$ (eigenvalues equal and real): The variation of u as a function of time can be calculated as $u = u_0 e^{-t\frac{\sigma+\alpha}{2}}$, where u_0 is the initial value of u.

Thus, the system converges exponentially quickly from the neighborhood of the second equilibrium point. Figure 2 illustrates a *phase portrait*, the simultaneous variation of three variables x, y, z from several initial points. The equilibrium point in the plot is a stable spiral.

Probabilistic `Safety` - Longevity of Object Replicas: In any computer system, there is always a non-zero probability of all replicas of a object disappearing completely. We present a back of the envelope calculation of the likelihood of this happening in an endemic protocol that is at equilibrium. Each of the y_∞ stashers creates new stashers at a rate $\beta x_\infty 1 = \gamma$. Each stasher is also turning averse at the same rate γ, thus it is equally likely to die before creating any new stashers. The likelihood that none of the y_∞ stashers create any new replicas is $= (\frac{1}{2})^{y_\infty}$.

If protocol parameters α, γ, b are chosen so that $y_\infty = c.log_2 N$, the probability of all stashers dying before creating new ones is $\frac{1}{N^c}$. If a protocol period is 6 minutes long, $N = 1024$ and 50 replicas gives us an expected object longevity of 1.28×10^{10} years. With $N = 2^{20}$ and 100 replicas, we get an object lifetime of 1.45×10^{25} years.

4.2 Case Study II: Majority Selection Problem

Implementations of distributed systems often need to *select* between two choices (e.g., two differing types of file replicas with the same filename) based on the *majority* of voting processes in the system, e.g., in a distributed digital library application such as LOCKSS [16], for distributed replica management, etc. We state the problem as:

Majority Selection: *In a distributed group of N processes, each process initially proposes either 0 or 1. The Majority Selection protocol ensures that all processes agree on which of the two values (0 or 1) has a majority of non-faulty proposers.*

This problem is related to the Consensus problem [10], where each of N processes initially proposes a value (0 or 1), but eventually sets its output variable exactly once, and to the same value as other non-faulty processes. Ref. [10] shows that consensus is impossible to solve in an asynchronous system.

Observation: Majority selection is impossible to solve in an asynchronous system, since a solution could be used to implement consensus.

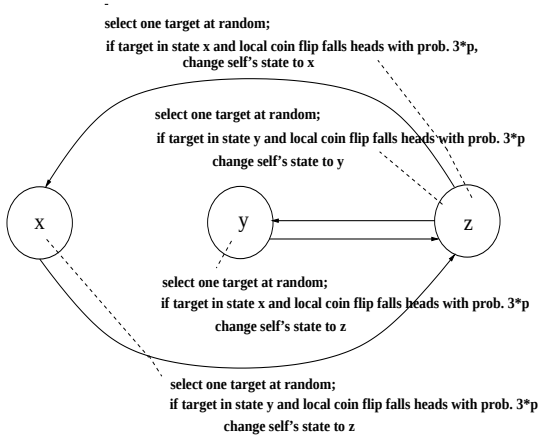

Figure 3: *The LV protocol for Probabilistic Majority Selection.*

Below, we give a specification for probabilistic majority selection. The probabilistic majority selection runs forever, and it maintains a *running* decision variable with possible values 0 or 1 or b (undecided).

Probabilistic Majority Selection: *In a distributed group of N processes, each process initially proposes either 0 or 1. The Majority Selection protocol ensures that decision variables at all non-crashed processes eventually agree, and w.h.p. this is the same as the initial majority value.*

Probabilistic majority selection is useful for applications where the decision value is allowed to be set multiple times, e.g., LOCKSS [16][6].

4.2.1 The LV Protocol

The *Principle of Competitive Exclusion* says that "Two species competing for the same limited resource typically cannot coexist" [23]. The classic *Lotka-Volterra (LV) model* presents a mathematical modeling of the above phenomenon. In the model, a variable x denotes the number of rabbits, and variable y denotes the number of sheep, in a given ecosystem. The LV model encapsulates the competition in a system of differential equations. We use the following new equations because they are appropriate to our purpose[7]:

$$\begin{aligned} \dot{x} &= 3.x.(1 - x - 2y) \\ \dot{y} &= 3.y.(1 - y - 2x) \end{aligned} \tag{6}$$

To make this equation system complete, we add a new variable $z = 1 - x - y$, and the equation

$$\dot{z} = -\dot{x} - \dot{y}$$

[6]However, consensus cannot be achieved using this as it cannot be known *when* to finalize decision variable.

[7]These have not been presented elsewhere in literature.

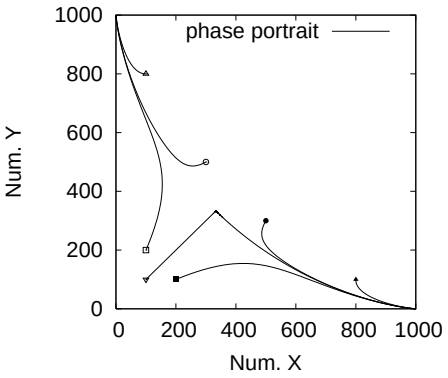

Figure 4: *Phase Portrait of the LV Protocol: The plot shows the phase portrait obtained by simultaneously plotting $(X, Y) = (Nx, Ny)$ over time. This plot shows a system of $N = 1000$ processes started at these initial points - (X, Y, Z)=blank square (100,200,700), dark square (200,100,700), blank circle (300,500,200), dark circle (500,300,200), blank triangle (100,800,100), dark triangle (800,100,100), and blank inverted triangle (100,100,800). All initial points with $x < y$ converge to (0,1000,0), and all initial points with $x > y$ converge to (1000,0,0). Initial points with $x = y$ move towards (333.3,333.3,333.3), but may then move arbitrarily towards one of the two stable points.*

A glance at equation system (6) shows a $+3x$ term on the r.h.s. of $\dot{x}$, indicating that the equations may not be partitionable. However, let us rewrite these equations as:

$$\begin{aligned} \dot{x} &= +3xz - 3xy \\ \dot{y} &= +3yz - 3xy \\ \dot{z} &= -3xz - 3yz + 3xy + 3xy \end{aligned} \tag{7}$$

These equations are both restricted polynomial, and completely partitionable. We can now apply Flipping and One-time-Sampling to generate the state machine shown in Figure 3. This uses a normalizing paramter p. We call this state machine as the "LV protocol".

4.2.2 Analysis of the LV Protocol

The exact source equations used for the LV protocol are not analyzed anywhere else in the literature.

Assuming the fractions of processes in different states to be x, y, z respectively, the equilibrium points for equation (6) (equivalent to equation (7)) are $(x, y) = (0, 0)$, $(0, 1)$, $(1, 0)$, and $(1/3, 1/3)$. The proof of the following theorem appears in [12].

Theorem 4 *(Correctness of LV protocol)***:** $(x, y) = (0, 1)$ and $(1, 0)$ are stable points, while $(0, 0)$ is an unstable point and $(1/3, 1/3)$ is a saddle point. Further:

1. If the system starts from any initial point that is to the right of $x = y$, i.e., has $x_0 > y_0 \geq 0, x_0 + y_0 \leq 1$, it will eventually converge towards $(1, 0)$.
2. If the system starts from any initial point that is to the left of $x = y$, i.e., has $x_0 < y_0 \geq 0, x_0 + y_0 \leq 1$, it will eventually converge towards $(0, 1)$.
3. If the system starts from any initial point that lies on $x = y$, i.e., has $x_0 = y_0 \geq 0, x_0 + y_0 \leq 1$, it will eventually converge towards $(1/3, 1/3)$.

Thus, an infinite-sized system eventually self-stabilizes. □

Figure 4 depicts the phase portrait of the system for several initial operating points.

In a finite-sized group of processes, the $x = y = 1/3$ point is unsustainable, because randomization will eventually push the system into either $x < y$ or $x > y$. Section 5 studies LV protocol in finite (large) sized group.

Self-Stabilization, Open Groups: Since the LV protocol runs forever, it is also *self-stabilizing*. Even in an open group, it proactively continues to converge back to an equilibrium point in spite of dynamic changes (e.g., new processes) that may perturb the operation point.

Convergence Complexity: For the LV state machine of Figure 3, we calculated the convergence complexity of stable point $(0, 1)$ as $(x(t), y(t)) = (u_0 e^{-3t}, 1 - (6u_0 t + v_0)e^{-3t})$, in the neighborhood $0 \leq u, v \ll 1$. A symmetric result holds for $(1, 0)$. The LV protocol thus has an exponential convergence complexity, implying that from an operating point in the vicinity of a stable point (e.g., $(0, 1)$), it takes $O(log(N))$ protocol periods to reach a global state where $O(1)$ processes are in the minority.

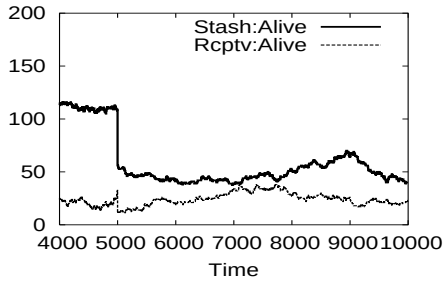

Figure 5: *Endemic Protocol - Massive Failures: In a setting with $N = 100,000$, $b = 2, \alpha = 10^{-6}, \gamma = 10^{-3}$, the number of stashers and replicas in a 100, 0000 host system is very small. Massive failure of 50% of the hosts at time t=5000 causes the system to stabilize quickly. However, we do notice that the stabilization is more sluggish than in settings with higher values of α, γ.*

5. EXPERIMENTAL STUDIES

We now present initial results from C implementations of an endemic protocol (designed for the application of a distributed file system for persistent storage of files), and an LV protocol. The protocols were tested in a simulated environment, with multiple instances running synchronously over a simulated network, all on a single machine (1.7 Ghz Intel Celeron CPU, 256 MB RAM, WinXP Pro). We are able to report numbers in 100,000-host groups. The Mersenne Twister pseudorandom generator is used for random number generation. In all the plots, the "Time" variable on the horizontal axis is normalized in protocol periods (6 minute intervals).

5.1 Endemic Replication - Experiments

This is an implementation of the protocol from Section 4.1. The protocol period is fixed at 6 minutes at each host. Values of b, γ, α vary for different experiments. All numbers are for a single file only.

Overhead, Fault-tolerance and Self-Stabilization: A 100,000 host system initially at equilibrium is subject to failure of a random 50% of the hosts. $b = 2, \alpha = 10^{-6}, \gamma = 10^{-3}$ are used. After the failure at time $t = 5000$, the number of

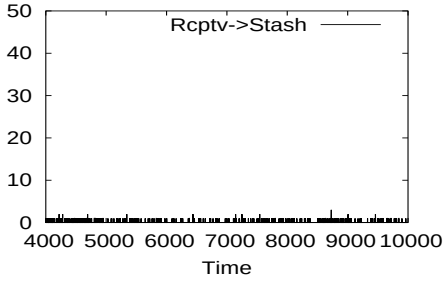

Figure 6: *Endemic Protocol - File Flux Rate = number of file transfers per protocol period. Same experiment as in Figure 5. Occurrence of a massive number of failures at time $t = 5000$ does not affect file flux rate drastically.*

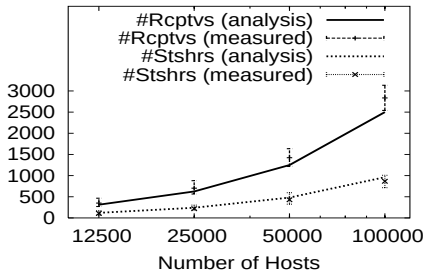

Figure 7: *Endemic Protocol - Accuracy of Continuous Time Analysis: The experimental results from the simulation match well with those predicted by the mathematical analysis. With $b = 2, \gamma = 0.1, \alpha = 0.001$, the above plot shows the median number (over a time interval 2000 periods long) and the analytically expected numbers of both receptives and stashers. The minimum and maximum measured values over this interval are also shown.*

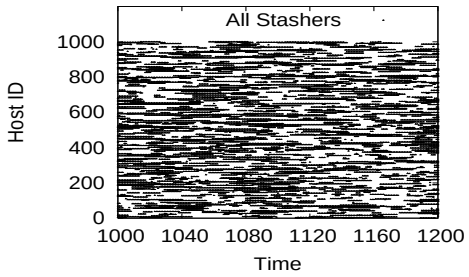

Figure 8: *Endemic Protocol - Replica Untraceability and Load Balancing: For a population of $N = 1000$ hosts and $b = 2, \gamma = 0.1, \alpha = 0.001$, the plot shows which hosts are stashers at the end of every protocol period. The absence of significant horizontal lines indicates good load balancing (no hosts store a replica for very long). The absence of any correlations w.r.t. time or hostid in the figure shows the difficulty faced by an attacker who seeks to destroy all replicas of the file.*

stashers (Figure 5) and the number of averse (not shown) each drop by a factor of about two. The number of receptives does not change since after the failure, 50% of the contacts initiated by any alive host are directed at a crashed host, and are hence fruitless (this reduces the effective value of b by 2, thus doubling the original equilibrium fraction $x_\infty = \frac{\gamma}{\beta}$). Figure 6 also shows that the overhead stays low, there is no wild variation in the number of stashers in spite of the massive failure, and that this number converges to the equilibrium value rather quickly.

The oscillation in the numbers of stashers and receptives, occurring due to random choice of contacts in the endemic protocol, eventually die out. The oscillations are more sluggish after the massive failure as fewer contacts are fruitful.

Notice that the behavior at times $t > 2500$ is also characteristic of a *heterogeneous* setting, where half the hosts are chronically averse to storing the file or even perhaps to running the protocol.

Reality Check: In a 100,000-host system, each host would be storing a given file for 0.1% of the time (since number of stashers $\simeq 100$), effectively once every 100,000 hundred hours, or 4166 hours. The file would be stored for an average duration of 100 hours (a little over four days) each time (expected time for a stasher to turn averse is $\frac{1}{\gamma} = 1000$ periods). If one assumes an average file size of 88.2 KB as in [20], a 6 minute protocol period implies a bandwidth utilization of 3.92×10^{-3} bps per file per host.

Analysis vs. Real Behavior: Our analysis in Section 4.1.3 assumed an infinitely large group. Figure 7 compares this with the results for large but finite groups. The two tally very closely, thus verifying that the considered group sizes are large enough for the analysis to apply.

Untraceability of Replicas: Figure 8 shows the distribution of stashers over time in a population of $N = 1000$ hosts, with protocol parameters $b = 2, \gamma = 0.1, \alpha = 0.001$. The stable number of stashers is 88.63. The distribution of the stashers (dark dots) does not appear to have correlations either in time or across the host id. Unless the attacker knows about the location of all the replicas of a file at a given point of time *and* destroys all these copies before any new stashers are created (with the current parameters, one stasher is created every 40.6 seconds), the file will survive inside the system. The plot also illustrates load balancing.

Trace-based simulations - Effect of Host Churn: Real protocol deployments are also required to tolerate dynamic stresses such as host churn, i.e., rapid arrival and departure. We model the worst effect of host failure - a host loses all stored file replicas when it fails or departs. When it rejoins the system, it is in the receptive state towards all files but does not participate in any startup file transfers. Figures 9 and 10 show the behavior of endemic replication in a 2000 host system under host churn, injected in the form of availability traces taken from the Overnet system [4]. The original availability traces were taken once every hour; for our experiments, these were spread out over each hour. The protocol period was set to 6 minutes, and the values of $\alpha = 0.005, \gamma = 0.1, b = 32$ used. Hourly churn rates were 10% to 25% of the system size. The endemic protocol is seen to maintain a stable number of stashers and low file flux rate, and is thus churn-resistant.

5.2 LV Protocol - Experiments

We present excerpts from simulation results of our LV protocol implementation of Section 4.2 - further numbers

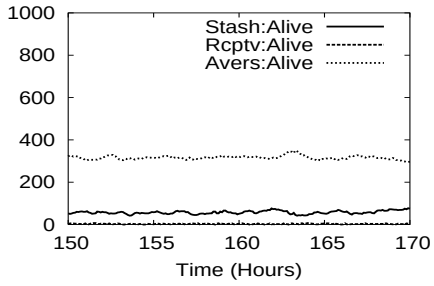

Figure 9: *Endemic Protocol - Effect of Host Churn A: Churn traces were injected hourly into the system with the protocol period set to 6 minutes at each host.* $N = 2000, b = 32, \gamma = 0.1, \alpha = 0.005$. *This plot shows that the numbers of stashers, averse and receptives remain stable in the system, and the the number of stashers stays low.*

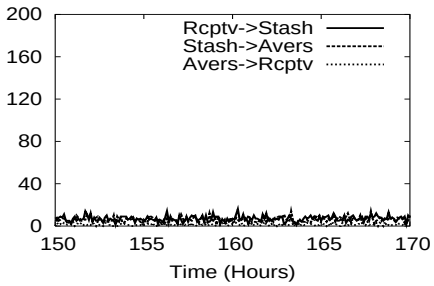

Figure 10: *Endemic Protocol - Effect of Host Churn B: For the experiment in Figure 9, this plot shows the number of state transitions, per protocol period, across hosts.*

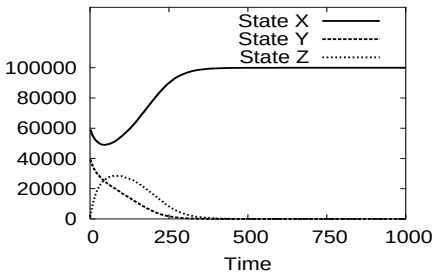

Figure 11: *LV Protocol - Variation of Populations: In a 100,000 process group, starting with 60,000 processes in state x and 40,000 processes in state y.*

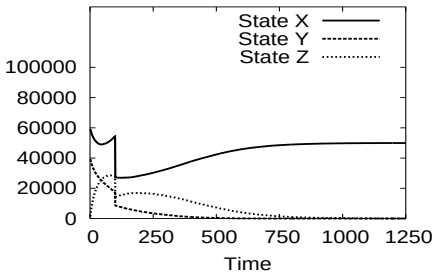

Figure 12: *LV Protocol - Effect of Massive Failures: For the same initial conditions as in Figure 11, half the processes (selected at random) are caused to fail at time* $t = 100$. *Convergence still occurs, although a little late (at time* $t = 862$*).*

are in [13]. The normalizing parameter $p = 0.01$.

Convergence: Figure 11 shows that a 100,000 process group started with 60,000 processes in state x and 40,000 processes in state y converges quickly towards having everyone in the initial majority state x. The convergence time is less than 500 rounds; with a protocol period of (say) 1 s, this means convergence within 8 minutes.

Effect of Massive Failures: For the same initial conditions as in Figure 11, half the processes, selected at random, are caused to crash at time $t = 100$. Figure 12 shows that the system converges nevertheless. The convergence time is delayed, at $t = 862$.

6. MAPPING EQUATION SYSTEMS THAT ARE POLYNOMIAL

Flipping and one-time-sampling may not suffice to translate equation systems that are polynomials but not restricted polynomials, even if they are completely partitionable. For example, an equation $\dot{x} = f_x(\bar{X})$ might contain a $+c$ term, or a term $-\Pi^{y \in X} y^{i_{y,f_x,T}}$ term with $i_{x,f_x,T} = 0$.

We describe how to map equations that are both polynomial and completely partitionable.

Tokenizing: In a completely partitionable and polynomial equation system, consider a term $-T = -c_T.\Pi^{y \in X} y^{i_{y,f_x,T}}$ that occurs on the r.h.s. of the equation for variable $\dot{x}$. Further, $i_{x,f_x,T} = 0$. Otherwise, flipping or one-time-sampling can be used. A *Tokenizing* action can be combined with Flipping or One-time-sampling to map this term into an action. This is done as follows - choose a variable $w \in X$ such that $i_{w,f_x,T} \neq 0$. If no such variable exists, term T is a constant c - in that case, rewrite $+T$ as $+c.(\Sigma^{v \in X} v)$ and $-T$ as $-c.(\Sigma^{v \in X} v)$, and repeat these steps.

Now, create an appropriate flipping or one-time-sampling action for processes in state w. This action is created in a similar manner as in Section 3. However, modify the action as follows: when all conditions are satisfied by the flipping or one-time-sampling action (coin turns up heads, or three conditions are true, respectively), then the process in question does not transition, but instead creates a *token* specifying the action. It immediately passes the token on to another process that it knows is in state x. On receipt of the token, the process in state x transitions to the state of the variable that has the corresponding $+T$ term. If no processes in the system are in the state x, the token is dropped.

Theorem 5: Flipping, One-time-sampling and Tokenizing are sufficient in mapping, into distributed protocols, equation systems that are polynomial.

Proof: Similar to Theorem 1.

Limitations of Tokenizing: After a token is generated at a process p, the ability to pass this on to another process that is in the needed target state for the token, requires continuous maintenance of knowledge (at p) of which states other processes are in (or at least one process for each possible state). This could be achieved by using a scalable membership protocol such as SWIM [8], yet may require optimization especially if transitions are occurring very rapidly.

An alternative is to associate each token with an integer-valued time-to-live (TTL) and pass it along a random walk among the processes until a process in the target state is encountered. A finite TTL has a probability of expiring before an appropriate target is met, while an infinite TTL may cause state transitions to occur too late. In either case, the behavior of the protocol may be different from the orig-

inal equation system. However, the new behavior can still be analyzed by modifying the original equation system with multiplicative terms in tokenized actions that account for the likelihood of the generated token being effective.

7. REWRITING EQUATIONS

We describe techniques that translate an equation system into a mappable form, i.e., completely partitionable, and either polynomial or restricted polynomial.

Rewriting an equation into a Complete form: Any equation system $\dot{\bar{X}} = f(\bar{X})$ can be rewritten into an equivalent equation system that is complete, by (i) introducing a new variable $z \notin X$ so that $z = 1 - \Sigma^{x \in X} x$, and (ii) considering instead the modified equation system, for variables in $X' = X \cup \{z\}$, consisting of the union of the original equation system $\dot{\bar{X}} = f(\bar{X})$ and the equation $\dot{z} = \Sigma^{x \in X}(-f_x(\bar{X}))$.
E.g., LV equation rewriting in Section 4.2.

Normalizing: A completely partitionable system of differential equations of the form $\dot{\bar{X}} = f(\bar{X})$ may not satisfy $\Sigma^{x \in X} x = 1$. Each variable may need to be normalized by *multiplying* it with a constant quantity $N = \Sigma^{x \in X} x$. N is a constant since the sum of fractions of processes in different states does not change in a complete system, but it may appear as a constant in the modified equations.
E.g., The epidemic equation system (0) in Section 1 were derived from the original system $\dot{x} = -\frac{1}{N}xy, \dot{y} = \frac{1}{N}xy$, where x and y are the *numbers* of susceptibles and infectives. The normalizing constant is $N = x + y$, the (fixed) group size.

Mapping Differential equations of higher Orders: *Some* equation systems of higher order (all derivatives w.r.t. t only) can be rewritten, e.g., an equation in a single variable, that has arbitrary order $k \geq 1$ (highest derivative of a variable) and degree 1 (power of highest derivative), can be rewritten as an equivalent equation system by introducing new variables for higher order terms. If the original equation has m variables, the equation system of equations may contain up to mk extra variables.
E.g., $\ddot{x} + \dot{x} = x$ can be rewritten as the completely partitionable system: $\dot{x} = u; \dot{u} = x - u; \dot{z} = -x$.

Finally, *some* equations of arbitrary degree and order can be rewritten. For example, $\ddot{x}^2 + 5\ddot{x} + \dot{x} - 3 = 0$ can be solved for $\ddot{x}$, and then the above techniques applied.

8. CONCLUSION

We have proposed a systematic framework that translates systems of differential equations into practical distributed protocols. The equation systems initially considered are of the form $\dot{\bar{X}} = f(\bar{X})$, with polynomial terms on the right hand side. The protocols are equivalent to the equations, in the sense that their behavior in an infinite-sized system is the same as that of the differential equations. We have also given techniques that can be used to rewrite differential equations of higher order and degree into a form that allows translation. We have shown the usefulness of this design framework by using it to design probabilistic protocols for endemic migratory replication and majority selection. We believe that the framework can be very useful in the design of distributed protocols, because many fields of science represent ideas and results through differential equations.

Other Questions: (1) What is a comprehensive set of rewriting techniques? (2) [12] presents mapping techniques for arbitrary non-polynomial equation systems - are these sufficient? (3) Can one formalize the relation between protocol performance at infinite group size and finite group size, as in [15, 18]? (5) Is complete=completely partitionable?

Acknowledgements: We thank Steve Bond for useful discussions.

9. REFERENCES

[1] D. Achlioptas, "Lower bounds for random 3-SAT via differential equations", *TCS*, 265, 2001, 159-185.

[2] R. J. Anderson, "The Eternity Service", *Proc. 1st Intl. Conf. Th. and Appl. of Cryptology*, Prague, 1996.

[3] N. T. J. Bailey, *Epidemic Theory of Infectious Diseases and its Applications*, Hafner Press, Second Edition, 1975.

[4] R. Bhagwan, S. Savage, G.M. Voelker, "Understanding availability", *Proc. IPTPS*, Feb 2003, 135-140.

[5] R. Bhagwan et al., "Total Recall: system support for automated availability management", *Proc. Usenix NSDI*, 2004.

[6] E. Cohen, S. Shenker, "Replication strategies in unstructured peer-to-peer networks", *Proc. ACM SIGCOMM*, 2002, 177-190.

[7] G. Colouris, J. Dollimore, T. Kindberg, *Distributed Systems: Concepts and Design*, Addison-Wesley, Third Edition, 2001.

[8] A. Das, I. Gupta, A. Motivala, "SWIM: Scalable Weakly-consistent Infection-style process group Membership protocol", *Proc. IEEE DSN*, 2002, 303-312.

[9] A. Demers et al., "Epidemic algorithms for replicated database maintenance", *Proc. ACM PODC*, 1987, 1-12.

[10] M. J. Fischer, N. A. Lynch, M. Patterson, "Impossibility of distributed consensus with one faulty process", *JACM*, 32:2, Apr 1985, 374-382.

[11] J. Gray et al., "The dangers of replication and a solution", *Proc. ACM SIGMOD*, 1996, 173-182.

[12] I. Gupta, "On the Design of Distributed Protocols from Differnetial Equoations, NCSTRL T. R. UIUCDCS-R-2004-2440, May 2004.

[13] I. Gupta, "A migratory approach to dynamic replication in large-scale distributed systems", NCSTRL T.R. UIUCDCS-R-2004-2441, May 2004.

[14] J. Kubiatowicz et al., "Oceanstore: an architecture for globalscale persistent store". *Proc. ACM ASPLOS*, Cambridge, MA, Nov 2000.

[15] T.G. Kurtz, *Approximation of population processes*, Regional Conference Series in Applied Mathematics, SIAM, 1981, ISBN 0-89871-169-X.

[16] P. Maniatis et al., "Preserving peer replicas by rate-limited sampled voting", *Proc. ACM SOSP*, Oct 2003.

[17] M. Merritt, G. Taubenfeld, "Computing with infinitely many processes", *Proc. DISC*, Springer LNCS 1914, 2000, 164-178.

[18] M. Mitzenmacher, "The power of two choices in randomized load balancing", *IEEE TPDS*, 12:10, Oct 2001, 1094-1104.

[19] M.O. Rabin, "Randomized Byzantine generals", *Proc. FOCS*, Nov 1983, 403-409.

[20] A. Rowstron and P. Druschel, "Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility", *Proc. ACM SOSP*, Oct 2001.

[21] Y. Saito et al., "Taming aggressive replication in the Pangaea file system", *ACM SIGOPS Oper. Sys. Rev.*, 2001, 15-30.

[22] M. D. Schroeder, A. D. Birrell, R. M. Needham, "Experience with Grapevine: the growth of a distributed system", *ACM TOCS*, 2:1, Feb 1984, 3-23.

[23] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*, Perseus Book Group, First Edition, 2001.

[24] N.C. Wormald, "Differential equations for random processes and random graphs", Annals of Appl. Prob., 5:4, 1995, 1217-1235.

[25] S.-H. Wu et al., "Composition and behaviors of probabilistic I/O automata", *TCS*, 176:1-2, Apr 1997, 1-38.

Errata. The above version of the paper is identical to the one presented in Proceedings of the 23^{rd} Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), 2004. The following corrections apply to the ACM PODC version of the paper.

- The protocol in Figure 1 is a variant of that obtained through the methodology of Section 3.
- Section 4.1.3 changes notation: x, y, z become numbers of processes in different states. And $\beta = 2b/N$ (rather than $2b$). Figure 2 reflects the earlier notation of Section 4.1.2,
- *Theorem 5* should read as *Flipping, One-time-sampling and Tokenizing are sufficient in mapping, into distributed protocols, equation systems that are both polynomial and completely partitionable.*