

Cover Page

Note: This paper is a nominee for the Carter Award.

Paper Abstract Number: 341 (DCCS)

Title: Building Scalable Solutions to Distributed Computing Problems using Probabilistic Components.

Authors and Affiliations: Indranil Gupta¹, Ken Birman².

1. Department of Computer Science

University of Illinois, Urbana-Champaign

201 N. Goodwin Ave.

Urbana, IL, 61801.

email: indy@cs.uiuc.edu

Ph: (217) 265-5517

Fax: (217) 333-3501

2. Department of Computer Science

Cornell University

4126 Upson Hall

Ithaca, NY, 14853.

email: ken@cs.cornell.edu

Ph: (607) 255-9199

Fax: (607) 255-4428

Contact Author: Indranil Gupta

Department of Computer Science

University of Illinois, Urbana-Champaign

201 N. Goodwin Ave.

Urbana, IL, 61801.

email: indy@cs.uiuc.edu

Ph: (217) 265-5517

Fax: (217) 333-3501

Abstract: We present a composable methodology that designs a new class of reliable and scalable protocols for distributed computing problems. The methodology generates solutions for failure detection, group membership, variants of reliable multicast, leader election, data aggregation, and distributed indexing. These operations are essential in the design of distributed systems such as massive decentralized clusters, peer to peer systems and sensor networks. By using (and reusing, through composition) randomization, message redundancy, and decentralized algorithms, these new “probabilistic protocols” offer high probability reliabilities to distributed applications. The overhead imposed on participating hosts is either independent of scale, or grows very slowly with system size (often a polylogarithmic function). Deterministic reliability can then be achieved by backing up a probabilistic protocol with a recovery protocol. All the above protocols (viz., from failure detection to distributed indexing) can be generated by starting with a set of seven building block protocols, and by gluing them using three composition techniques. We discuss the preservation of liveness, scale and reliability properties under protocol composition.

General Topic Area: Distributed Systems.

Keywords: FT Communication, FT Algorithms, Probabilistic Modeling, Scalability, Design of Protocols.

Submission category: Regular Paper (DCCS).

Word count: 7500.

Declaration: This material has been cleared by the authors’ affiliations.

Note: This work was done as a part of Indranil Gupta’s PhD during his residence at Cornell University. Dr. Gupta is currently Assistant Professor of Computer Science at the University of Illinois, Urbana-Champaign.

CARTER Award Nominee: YES

Building Scalable Solutions to Distributed Computing Problems using Probabilistic Components

Indranil Gupta (Dept. of Computer Science, University of Illinois, Urbana-Champaign)

Ken Birman (Dept. of Computer Science, Cornell University, Ithaca)*

Abstract

We present a composable methodology that designs a new class of reliable and scalable protocols for distributed computing problems. The methodology generates solutions for failure detection, group membership, variants of reliable multicast, leader election, data aggregation, and distributed indexing. These operations are essential in the design of distributed systems such as massive decentralized clusters, peer to peer systems and sensor networks. By using (and reusing, through composition) randomization, message redundancy, and decentralized algorithms, these new “probabilistic protocols” offer high probability reliabilities to distributed applications. The overhead imposed on participating hosts is either independent of scale, or grows very slowly with system size (often a polylogarithmic function). Deterministic reliability can then be achieved by backing up a probabilistic protocol with a recovery protocol. All the above protocols (viz., from failure detection to distributed indexing) can be generated by starting with a set of seven building block protocols, and by gluing them using three composition techniques. We discuss the preservation of liveness, scale and reliability properties under protocol composition.

1 Introduction

Large-scale distributed applications such as those in the Grid, peer to peer file sharing, publish-subscribe, cooperative web caching, replicated databases, etc. ¹ require support from middleware. Middleware design has to employ protocols that ensure *reliability* of performance in spite of end point failures (which could either be permanent, e.g., host crashes, or transient, e.g., unresponsive applications), transient packet losses within the communication

*The authors were supported in part by DARPA/AFRL-IFGA grant F30602-99-1-0532 and in part by a MURI grant AFOSR F49620-02-1-0233, with additional support from the AFRL-IFGA Information Assurance Institute, from Microsoft Research and from the Intel Corporation.

¹Sensor network applications also form part of the list, but are excluded since we do not focus on them in this paper.

network, and continuous arrival and departure of end points. In addition, *scalability* is a requirement; the algorithms should achieve reliable operation by imposing light overheads on end points and on the network.

While many algorithms have been developed for such middleware, the *design* of these algorithms remains a challenge. A high level question is: what are the *principles that underlie the design* of distributed protocols, and that could be reused to *generate further protocols*. At first glance, this appears similar to code composition, e.g., object-oriented software. However, the above question addresses a much higher level - it is directed at the time and energy spent behind the intellectually challenging task of *design* (as opposed to the implementation) of protocols.

We have previously designed a class of *probabilistic* protocols to implement core services for large-scale distributed applications. In this paper, we are able to present a design methodology for this protocol class. The methodology is informal, and provides guidelines by which probabilistic protocols can be composed to preserve correctness, scale and reliability. We do not attempt to present a science of design, but hope this paper is perhaps a step in a larger direction.

Large-Scale Distributed Systems and Probabilistic Protocols: Both deterministic and randomized algorithms have been used to design distributed protocols. Deterministic state machine-based approaches such as centralization and two phase commit may be disadvantageous in certain settings. Two drawbacks are that they: (a) impose overheads on end points that grow linearly with group size, or (b) degrade overall group throughput due to perturbation at a single end point. See [4] for such a study in the realm of reliable multicast protocols.

Randomization is a useful technique to overcome the asymptotic limits of deterministic algorithms. The text by Raghavan and Motwani [26] contains an extensive collection of randomized algorithms. In the distributed systems community, randomized strategies were first extensively used after the FLP result [11] showed the impossibility of achieving consensus in a process group over an asynchronous network (inclusive of lossy networks). Besides eventual consensus protocols such as Lamport's Paxos approach [27], Ben-Or [2] and Rabin [28] were the first to use randomization to achieve probabilistic consensus. Chor et al [7] used randomization to construct a protocol that achieves consensus in a constant number of rounds, where a round involves all-to-all communication among the members of a group. ² Randomization is also a cornerstone in the design of scalable and reliable peer-to-peer systems. For example, distributed hash tables such as Chord, Pastry, Tapestry, CAN, etc. [10], use consistent hash functions (e.g., SHA-1) and random selection of peer pointers for routing.

Our research effort has been directed at the design, analysis, implementation, and experimentation of a class of

²Randomization has also been used to implement consensus protocols tolerant to Byzantine members - we refer the reader to reference [6] for a good summary.

probabilistic protocols that provide important services to large-scale distributed applications such as those mentioned earlier in the introduction. These services include decentralized group membership maintenance, reliable multicast, leader election, data aggregation, and distributed resource location and discovery. The description and evaluation of these protocols were previously published in references [8, 13, 14, 15, 16, 17, 18, 19, 23]. Applications built using these services include cooperative web caching [23] and reliable multicast [18].

However, the *design* of protocols is a challenging task in itself, and is the direction of this paper. This paper is a retrospective look at the technique of *designing* the above class of probabilistic protocols. Many researchers believe protocol design to be an art. This paper does not seek to define a formal framework for design. The design methodology presented is intended to serve as *guidelines* to a designer of distributed protocols. To draw an analogy, this paper's approach to protocol is more similar to design patterns used in software development [12] rather than a formal composition framework such as probabilistic I/O automata [32] (the latter of which produces a different class of protocols than in this paper).

We present a *composable* methodology for probabilistic protocol design. We are able to specify three *composition techniques* for gluing together probabilistic protocols, and analyze how these techniques preserve liveness, scale and reliability of component protocols. The methodology also delineates seven categories of *building blocks*, each of which is either a probabilistic protocol or strategy. This allows one to build hierarchies of probabilistic protocols through the application of these composition techniques. Seen a different way, a base protocol's properties can be enriched by composition with other building blocks, e.g., a failure detector and a multicast protocol can be combined into a membership protocol, a probabilistic reliable multicast protocol can be backed up with a recovery protocol to guarantee deterministically reliable multicast. For the studied set of building blocks, we find that the composition rules preserve the scale, reliability and liveness properties of the components.

Probabilistic Protocols of Interest in this paper: The probabilistic protocols we focus on [8, 13, 14, 15, 16, 17, 18, 19, 23] offer a probabilistic notion of reliability. Decentralization, randomization and message redundancy are used to achieve scale and fault-tolerance. "Scale" means that the overhead on participating members required in order to achieve a given level of probabilistic reliability grows very slowly with the total number of members. The dependence varies from being a polylogarithmic function of, to being independent of the group size. "Fault-tolerance" means that the reliability offered by the protocol is not drastically affected by continuous perturbation and failure of members, and point to point message delivery losses. Instead, the reliability degrades gracefully as the rates of these failures are increased.

Different global operations can have different notions of probabilistic reliability - in the case of an aggregation protocol [17], reliability refers to the accuracy of the global aggregate calculated by a run of the protocol, whereas in the case of a group membership system [8, 15], reliability pertains to such features as expected times taken to detect member failures, and frequency of inaccurate failure detections. In the former case, the probabilistic protocol for aggregation would calculate a highly complete estimate of the aggregate (i.e., one that includes a fraction of individual member votes that is close to 1.0). In the latter case, the the probabilistic protocol for group membership would detect member failures within a time interval that is constant on expectation (i.e., invariant with group size). A probabilistic reliable multicast protocol disseminates multicasts to a large fraction of participants (very close to 1.0). A leader election protocol [19] would elect exactly one leader in the group with very high probability.

In order to achieve this reliability, the overhead imposed on group members in order to achieve these levels of probabilistic reliability grows slowly as a function of group size. For example, the aggregation and reliable multicast protocols have a polylogarithmic dependence, while the membership protocol has a constant scale-independent cost.

Probabilistic Protocol Composition: To summarize, the main advantages of probabilistic protocols of our interest are (a) correctness (i.e., liveness) properties, (b) probabistic reliability, and (c) scalability of overhead in order to achieve the reliability in (b). As such, our study of the composition rules will be directed at the extent of preservation or inheritance of these properties in spite of composition.

There are many advantages to being able to compose such protocols. One can implement a variety of specifications for distributed tasks using appropriately chosen building blocks and rules, e.g., a membership protocol can be constructed out of a failure detector protocol and a multicast protocol. An existing protocol can be augmented to improve certain characteristics such as stress imposed on core network routers, adaptivity to fault-free scenarios, etc. Perhaps most importantly, probabilistic protocols can be backed up with a Recovery protocol to provide a more classical deterministic reliability guarantee. We expand on these through the rest of the paper.

The rest of the paper is organized as follows. Section 2 presents the model used to analyze our protocols. Section 3 discusses related work. Section 4 motivates the discussion of the methodology through a case study; Section 5 presents the methodology. Section 6 briefly describes the protocols generasted by the design methodology. We summarize in Section 7.

2 Model Used in Analysis

To make the paper self-contained, we include the network model used in protocol analyses. This model is a hybrid between the synchronous and asynchronous models used in literature.

The group consists of participants called *members*. We denote as N the size of the group, and clarify its exact meaning where needed (e.g., whether it is actual or estimated). Each member has a unique identifier. Each member maintains a membership list, also called a *view*. Members may undergo *crash-stop failures*, whereby they cease all operations after the failure. If a member rejoins the group, it does so under a different identifier.

A non-faulty member may perform the following operations. It can send unicast messages to another member, whose identifier it knows, through the communication network. A unicast message is delivered either instantaneously at the destination member, or is lost by the network with probability p_{ml} , independently and identically distributed across messages. This assumption can be relaxed to one where the time-outs used in protocols are multiples of an atomic time unit, and $(1 - p_{ml})$ is the probability of successful delivery of a message at the recipient within 1 atomic time unit of the message transmission. Also, some protocol analyses (e.g., gossip-based protocols) implicitly assume $p_{ml} = 0$; the analysis for non-zero values of p_{ml} yields the same results, albeit with the per-member overhead multiplied by a factor of $\frac{1}{1-p_{ml}}$. Each member performs operations according to a local clock. Clock rates at all members are the same. Many of the protocols presented in the paper run at each member periodically - the clock rate assumption then assures that the protocol period lengths are the same at all members. However, protocol period start times may be unsynchronized across different members. The communication bandwidth and computational power available to members is not limited. However, the protocols themselves do not require infinite bandwidth or computational power. They work in fully asynchronous settings, and minimize bandwidth and computational power usage - our previous experiments in [8, 13, 14, 15, 16, 17, 18, 19, 23] have borne this out.

Additional assumptions made while analyzing the protocols are stated where necessary.

3 Other Related Work

Epidemic-style multicast protocols such as those by Demers et al [9], Birman et al [4] spread information in a distributed group by using randomized and redundant peer-to-peer messaging. A notion of composition of probabilistic I/O automata for distributed systems has been studied in the past by Lynch [25] and Wu et al [32]. The authors defined I/O automata with probabilistic state transitions and specified how these automata can be composed. In comparison, our study of probabilistic protocol composition is geared toward one designing protocols for distributed

systems in large-scale scenarios, where the scalability and reliability offered by these protocols are primary concerns.

Stack-based communication architectures such as the Horus and Ensemble systems [31, 24] are designed to allow network protocol layers to be stacked, with the glue between layers being a standard function call interface set. This helps to provide different notions of reliability, message ordering, etc., to the end application. In comparison, our protocol composition methodology is aimed at enhancing scalability and reliability properties of distributed protocols rather than merely providing richer properties to the application.

Several toolkits are available for component-based software development and optimization, e.g., Knit. [29]. Components have been used for designing software for routers (Click system) and operating systems (e.g., Scout). Object-oriented programming languages such as C++ and Java allow component-based program development. Software components can also be glued together through higher-level scripting languages (e.g., Python, VB, Perl) [30] to filter data streams among the different components. In object request brokerage (ORB) systems (e.g., COM, CORBA, .Net), each component is an application connected to a communication network.

4 Case Study : A Protocol for Decentralized Membership Maintenance

We motivate discussion about the design methodology by presenting a case study that retrospectively looks at the design of the SWIM protocol for weakly consistent and scalable membership maintenance. We choose this protocol because it is familiar to the audience of DSN 2001 [8], and was based on the failure detector protocol presented in [15]. Although those papers described the protocol and its evaluation, the current paper focuses on design. Repetition of material is kept to a minimum, but is unavoidable in some places.

Distributed replica management, coordination, and gossip-based dissemination, require each member of the group to maintain a local list of other members in the group. We call this a membership list (also a view). In a dynamic group with members constantly joining, leaving and failing silently (crash-stop) failures, a membership maintenance protocol keeps these lists up to date - scalability and reliability (including quickness of membership change detection and dissemination) are requirements. A membership protocol has two main components: (a) a failure detector protocol, and (b) a protocol for membership update dissemination. Any implementations for the respective components can be fit into a *template*, as show in Figure 1, to generate a membership protocol. We first describe the probabilistic building blocks for each of (a) and (b), and then illustrate how the composition rule is used.

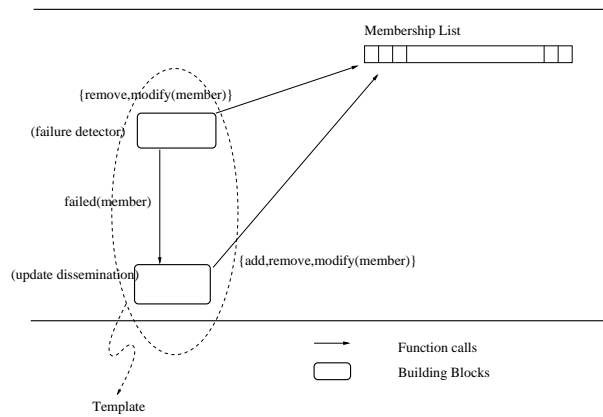


Figure 1: **Group Membership Template.** *Template-based* (also *Modular*) design of a group membership protocol is shown, with minimal function interfaces for the failure detector and update dissemination components.

4.1 Building Block: Distributed Ping Protocol

A failure detector protocol runs constantly at each non-faulty member. It is required to satisfy a liveness property called *Eventual Strong Completeness (Completeness)*, i.e., crash-failure of any group member is detected by *all* non-faulty members that knew of the member. The protocol performance is measured through its *inaccuracy* (rate of false detections of failure of non-faulty nodes), *speed* (time between a crash and its failure detection), and *scale* (overhead on non-faulty members and the network).

Chandra et al show in [5] that it is impossible to design failure detector protocol for a failure-prone network that is both complete and accurate. Consequently, middleware has typically chosen to tolerate a low false positive rate (inaccuracy) but require that completeness be maintained. Traditional heartbeat-based failure detection protocols are based on this approach.

Figure 2 gives pseudocode for the probabilistic failure detector protocol called Distributed Ping [15]. The main idea in this protocol is for each member to periodically attempt to ping one other member chosen uniformly at random, directly first, then indirectly if this does not succeed. If the target member remains unresponsive, it is marked as failed.

$View_{M_s}$ /* Membership List at member M_s */

Member M_s : Distributed Ping(T' , k)

/* T' : Protocol period */

/* k : Fanout for indirect pingging */

int $round_num=0$

Every T' time units:

$round_num := round_num + 1$

pick a node M_{target} uniformly at random from $View_{M_s}$

send a ping($M_s, M_{target}, round_num, dir$) message to M_{target}

if (not received ack($M_s, M_{target}, round_num, dir$) within $\frac{T'}{3}$ time units)

pick k members $pingreqtargs[1 \dots k]$ uniformly at random from $View_{M_s}$

for $i = 1$ **to** k

send a ping-req($M_s, M_{target}, round_num$) message to $pingreqtargs[i]$

if (not received ack($M_s, M_{target}, round_num, *$) within $\frac{2T'}{3}$ time units)

delete M_{target} from $View_{M_s}$

Anytime:

- on receiving a ping(M_t, M_s, r, dir) message
send ack(M_t, M_s, r, dir) message to M_t
- on receiving a ping-req(M_t, M_j, r) message
send ping($M_t, M_j, r, M_s, indir$) message to M_j
- on receiving a ping($M_t, M_s, r, M_i, indir$) message
send ack($M_t, M_s, r, indir$) message to M_i
- on receiving an ack($M_t, M_j, r, indir$) message
send ack($M_t, M_j, r, indir$) message to M_t

Figure 2: Distributed Ping Protocol for Failure Detection.

Member M_s :Uniform Epidemic($N, b, View_{M_s}$)

/* N : estimate of group size */

/* b : gossip fanout */

/* $View_{M_s}$: Membership List at member M_s */

on receipt of a new multicast m

for $\log N$ gossip rounds

for $i := 1$ **to** b

pick a node M_{target} uniformly at random

from $View_{M_s}$

send m to M_{target}

Figure 3: The Uniform Epidemic Multicast Protocol.

Under the model of Section 2, Distributed Ping has the following properties.

Correctness: (Eventual Strong Completeness) After a member M_j fails, for each non-faulty member M_i that knows of M_j , M_i will eventually chose M_j as a ping target, find it unresponsive, and delete M_j from its membership list.

Analysis: • (Speed) The mean time to detection of failure (at a *first* other non-faulty member) is $T' \cdot \frac{1}{1-e^{-q_f}}$, where T' is the protocol period.

• (Accuracy, Scale) Let L be the per-member overhead (messages/time unit) due to Distributed Ping. Reference [15] calculates the minimal per-member load L^* that a failure detector needs to impose in order to satisfy application-defined failure detection time \mathcal{T} , false positive probability $\mathcal{PM}(\mathcal{T})$ (likelihood of a given non-faulty process being marked as failed within a time interval \mathcal{T}), when a fraction q_f of members are non-faulty and the message receipt probability is q_{ml} . It can then be shown that $\frac{L}{L^*} = [2 + 4 \cdot \frac{\log[\frac{\mathcal{PM}(\mathcal{T})}{(q_f \cdot (1-q_{ml}^2) \cdot \frac{e^{q_f}}{e^{q_f}-1})}]}{\log(1-q_f \cdot q_{ml}^4)}] \times [\frac{e^{q_f}}{e^{q_f}-1} \cdot \frac{\log(p_{ml})}{\log(\mathcal{PM}(\mathcal{T}))}]$, a term independent of N . This makes the Distributed Ping protocol an asymptotically optimal failure detector (w.r.t. N).

Overhead-Reliability Knob: k can be varied to trade between message overhead and false positive rate. Tuning the protocol period T' trades off between message overhead and failure detection time.

4.2 Building Block: Uniform Epidemic

We now describe a second building block called Uniform Epidemic, based on the epidemic (also “gossip”) protocols in [4, 9]. Given a multicast at a sender member, a Uniform Epidemic disseminates the multicast message w.h.p. to the group, i.e., for any particular member in the group, the probability of it receiving the multicast is known and is very close to 1. The protocol Figure 3 can be shown to satisfy:

Correctness: (Eventual Dissemination) If the membership knowledge graph among non-faulty members stays connected and Figure 3 is modified so members never cease gossiping about a multicast, any member that stays non-faulty will eventually receive the multicast.

Analysis: (Reliability and Overhead) References [1, 9, 22] show that the probability of a given member receiving the multicast through the protocol in Figure 3 is $1 - \frac{1}{N^b} \cdot (1 + o(1))$. This is the probabilistic reliability achieved by the protocol.

(Latency) References [4, 20] show that the completes w.h.p. within a number of rounds that varies as $O(\log_b(N))$.

Overhead-Reliability Knob: Increasing the gossip fanout b improves reliability and latency by trading a higher message overhead per gossip round³.

³Notice that a small value of $b(\geq 2)$ suffices to guarantee that all members receive the multicast w.h.p.

```

ViewMs /* Membership List at member Ms */
Member Ms: SWIM Membership (T', k, λ, α)
/* T': Protocol period */
/* k: Fanout for indirect pingging */
/* λ: Timeout parameter for purging buffer of latest membership updates */
/* α: Number of piggybacked elements */
var LatestUpdsMs; /* Latest Membership Updates at Ms */
message mupdates
var roundnum=0;
Every T' time units:
  roundnum := roundnum + 1
  create the payload data message mupdates by selecting α entries
  uniformly at random from LatestUpdsMs
  piggyback message mupdates on all outgoing messages during this
  round
  pick a node Mtarget uniformly at random from ViewMs
  send a ping(Ms, Mtarget, roundnum, dir) message to Mtarget
  if (not received ack(Ms, Mtarget, roundnum, dir) within  $\frac{T'}{3}$  time units)
    pick k members pingreqtargs[1..k] uniformly at random from ViewMs
    for i = 1 to k
      send a ping-req(Ms, Mtarget, roundnum) message to
        pingreqtargs[i]
    if (not received ack(Ms, Mtarget, roundnum, *) within  $\frac{2 \cdot T'}{3}$  time units)
      delete Mtarget from ViewMs
      add {Mtarget, leave} to LatestUpdsMs
  purge LatestUpdsMs of all entries older than  $\lambda \cdot \log(|View_{M_s}|)$  rounds

```

Figure 4: **SWIM Group Membership Protocol.** A member joins (resp. voluntarily leaves) the group by sending a join-request (resp. leave-request) message to a member of the group.

Figure 4 (Continued)

Anytime:

- on receiving a ping(M_t, M_s, r, dir) message
 - retrieve payload $m_{updates}$
 - send ack(M_t, M_s, r, dir) message to M_t
- on receiving a ping-req(M_t, M_j, r) message
 - retrieve payload $m_{updates}$
 - send ping($M_t, M_j, r, M_s, indir$) message to M_j
- on receiving a ping($M_t, M_s, r, M_i, indir$) message
 - retrieve payload $m_{updates}$
 - send ack($M_t, M_s, r, indir$) message to M_i
- on receiving an ack($M_t, M_j, r, indir$) message
 - retrieve payload $m_{updates}$
 - send ack($M_t, M_j, r, indir$) message to M_t

Member received payload $m_{updates}$. Process each entry as follows:

- on receiving a join-request(M_t) message
 - or piggybacked { $M_t, join$ } in $m_{updates}$
 - if** (entry for M_t not present in both $View_{M_s}$ and $LatestUpds_{M_s}$)
 - add M_t to $View_{M_s}$
 - add { $M_t, join$ } to $LatestUpds_{M_s}$
- on receiving a leave-request(M_t) message
 - or piggybacked { $M_t, leave$ } in $m_{updates}$
 - if** (entry for M_t not present in both $View_{M_s}$ and $LatestUpds_{M_s}$)
 - remove M_t from $View_{M_s}$
 - add { $M_t, leave$ } to $LatestUpds_{M_s}$

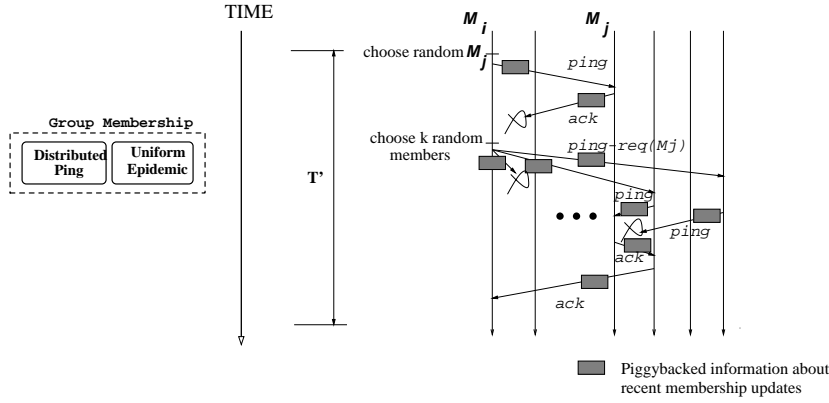


Figure 5: **SWIM Group Membership Protocol**. A scalable group membership protocol, composed from the Distributed Ping and Uniform Epidemic building blocks (box diagram on left).

4.3 Composition: SWIM Protocol for Weakly-Consistent Membership Maintenance

As mentioned earlier, a protocol for membership maintenance consists of (a) a failure detector protocol and (b) a protocol to disseminate membership updates. These can be combined modularly within a *template* as shown in Figure 1. Notice that any combination of implementations of a failure detector and a multicast protocol, each of which export the appropriate interfaces, can be fit into this template.

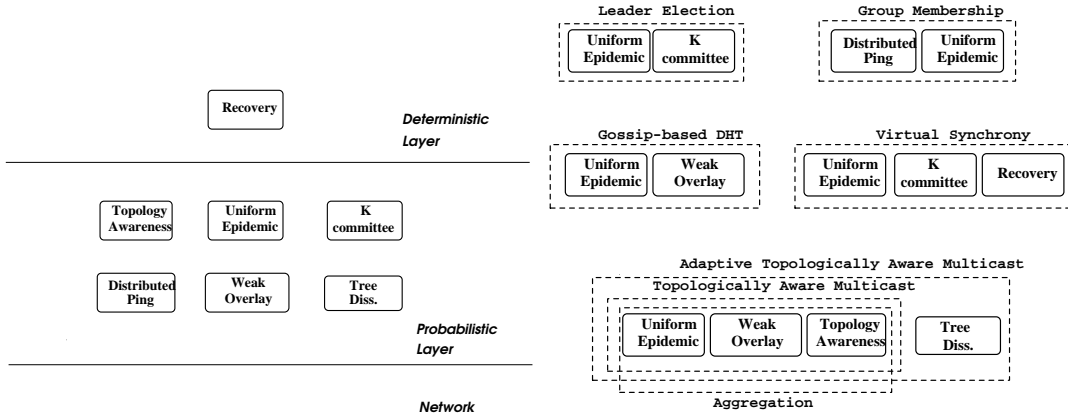
When the Distributed Ping and Uniform Epidemic protocols are fit into the failure detector and update dissemination components of Figure 1, the resultant protocol (when optimized) has the pseudocode shown in Figure 4. Figure 5 illustrates a protocol run. We call the composition rule as the *Template* or *Modular Composition* rule. The effect of this, and of the subsequent optimization, is to piggyback information about recently heard membership updates on all outgoing ping and ack messages in Distributed Ping, as well as to update membership lists based on such received information.

The SWIM protocol can be shown to satisfy [8, 15]:

Correctness: (Eventual Strong Completeness) If a member M_i fails, each other M_j that has M_i in its membership list eventually detects the failure.

(Eventual Dissemination of Updates) If the view graph among non-faulty members stays connected after the membership update first occurs in the group, and the buffer of latest updates is never purged, the update is eventually disseminated to all members that stay non-faulty.

Analysis: The composition has analytical properties that is a concatenation of the analytical properties of the Distributed Ping and the Uniform Epidemic described in Sections 4.1 and Section 4.2, i.e., w.r.t. the average fail-



(a) Seven Probabilistic Building Blocks; (b) Some Possible Compositions generated by the Methodology

Figure 6: Designing Probabilistic Protocols by Composition of Building Blocks.

ure detection time, false positive frequency, as well as the latency and reliability of dissemination of membership updates. The latter two depend on the parameters α and λ in Figure 4 - please see reference [8] for details.

Overhead-Reliability Knobs: Parameters T' , k as in the component protocols.

In summary, this first case study has shown that the first protocol composition rule we described (Template-based or Modular Composition) inherits in situ the the specified correctness, scale and reliability properties from the constituent protocols.

5 Composition Methodology

This section describes the overall protocol design methodology. It provides (the designer with) a collection of (a) building blocks - seven categories of probabilistic protocols/strategies with well-studied liveness, scalability and reliability properties - and (b) three composition techniques that generate protocols with richer semantics, by preserving scale, reliability and liveness properties of the components.

5.1 Probabilistic Building Blocks

Figure 6(a) shows the main building block types; Sections 4.1-4.2 covered two of them. We describe the remainder briefly and informally below.

Weak Overlay: A weak overlay is a scheme that each member uses to select its (partial) membership lists. In effect, a weak overlay scheme specifies a set of rules that imposes an overlay graph among the members present in the group. Examples include the Leaf Box Hierarchy [16], and peer to peer overlays Pastry and Chord.

Topology Awareness Strategies: A topology awareness scheme imbibes knowledge of round-trip time estimates or network locations (exact or approximate). It is typically used in composition with another building block, e.g., with the Leaf Box Hierarchy (as in [16]), or Uniform Epidemic (as in [21]).

Tree Dissemination: A multicast spanning tree is constructed among the nodes in a dynamic and distributed fashion [16]. A consistent map function is used; examples include the Contiguous Mapping [16], and SHA-1.

K-committee selection: This protocol selects and maintains a subgroup of members, that, w.h.p. (a) has a size within a constant factor of parameter K , and (b) has the view subgraph connected. An example is reference [19].

Recovery: Recovery protocols buffer information (e.g., recent multicasts) and supply them to requesting members that have missed receiving the information. An example is the recovery block used to implement reliable multicast and membership (virtual synchrony) in [18]. Section 6.1 expands on this protocol.

5.2 Composition Techniques

A pair of probabilistic protocols can be composed using one of the following composition techniques. These are not formally defined, but are presented as design guidelines. The guidelines also hint at which pairs of protocols could be composed.

- **Use in Protocol Template (“Template Technique” or “Modular composition”):** A “template” for a protocol can be generated from a problem specification (e.g., a group membership template in Figure 1) or from the informal specifications of two protocols one wishes to combine to achieve a protocol with collective properties. A template will typically specify the minimal interface/function calls to be exported by each component. When appropriate protocols are plugged into the template (followed by necessary optimizations), the final protocol is generated. This protocol can then be optimized appropriately. An example w.r.t. the group membership protocol was discussed in Section 4.

- **Augmentation - through Fashioning or Constraining:** A base composition C_1 could be augmented with another composition C_2 to derive a modified protocol that solves the same problem specification as C_1 . Yet, the composition imparts to the (augmented) C_1 certain additional properties.

There are two types of augmentation - fashioning and constraining. We clarify the distinction through an example. For example, a Uniform Epidemic protocol can be augmented with a component that selects epidemic targets from a membership list according to (a) a probability distribution function, e.g., based on round-trip-time estimates (*fashioning*), or (b) a set of constraints, e.g., eliminating certain types of members from ever being selected as targets (*constraining*). Figure 8 shows an augmented version of the Uniform Epidemic Protocol from Figure 3. Reference

[16] studies an augmentation of Figure 8 where the function *SelectionCriterion* is a composition of a weak overlay (Leaf Box Hierarchy) and a topology aware scheme (i.e., Contiguous Mapping). One can then show corresponding properties to those in Section 4.2 - the same **Correctness** property (Eventual Dissemination) holds still. However, the analytical properties are modified compared to the base protocol - the **Reliability** is $1 - \frac{1}{\sqrt{N}}$, and the **Latency** is sublinear. **Overhead-reliability** knobs remain the same.

Some more examples of augmentation follow. Distributed Ping (Figure 2) can be fashioned with a topology aware building block, that uses a probability distribution function, effectively preferring ping targets that are topologically close by. Uniform Epidemic can be constrained to operate within the Leaf Box Hierarchy weak overlay, thus producing the probabilistic protocol for data aggregation described in [17]. The Uniform Epidemic can be constrained within a weak overlay to produce a resource location and discovery protocol (or a distributed hash table - DHT) [14]. Similarly, ping target choices can be chosen from a partial membership list in the Distributed Ping protocol, giving a constrained Distributed Ping protocol.

The reader would notice from this discussion that for augmentations of Uniform Epidemic or Distributed Ping, constraining can indeed be seen as a special case of fashioning where certain members are assigned a probability zero of being selected as targets. The distinction is made from a designer's point of view where a constrained composition does not require the membership protocol (that will be orthogonally present in the overall middleware design) to maintain any knowledge of the zero target probability members.

Figure 6 depicts the compositions that we have investigated - Section 6 summarizes them. Incremental application of composition techniques yields a hierarchy of protocols with richer properties. Figure 7 shows the hierarchy that generates a class of epidemic-style reliable multicast protocols.

It will be evident from our description of compositions that a protocol designer writing code for distributed system middleware still has to choose the right type of building blocks and write code for them, to use appropriately chosen composition techniques, and write extra filler code besides that of the building blocks in order to complete the design of the protocol. In fact, *these requirements (especially the filler code) prevents us from being able to formally specify the protocol design methodology*. For example, the protocols for Group Membership, Topology Aware Reliable Multicast, and Adaptive Epidemic Multicast are straightforward instantiations of specific building blocks and composition techniques. The design of the remainder Leader Election, Virtually Synchronous Multicast, Data Aggregation, and Resource Location and Discovery, involve writing filler code.

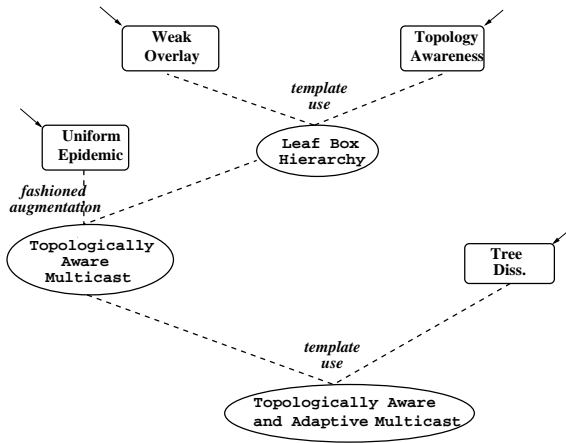


Figure 7: Example use of Composition Techniques and Building Blocks to design various flavors of Epidemic-based Multicast protocols.

Member M_s : **Uniform Epidemic**($N, b, View_{M_s}$)

/* N : estimate of group size */

/* b : gossip fanout */

/* $View_{M_s}$: Membership List at member M_s */

on receipt of a new multicast m

for $\log N$ gossip rounds

for $i := 1$ to b

pick a node M_{target}

using the SelectionCriterion($View_{M_s}$)

send m to M_{target}

SelectionCriterion($View_{M_s}$) /* an example */

Pick a member from $View_{M_s}$ with probability

proportional to $f(M_s)$

/* e.g. 1, $f(x) = \frac{1}{x^2}$ where rtt_{M_s}

is the network round trip time to M_s

e.g. 2, f is chosen as a composition of a

weak overlay and a topology aware scheme [16] */

Figure 8: Augmented Epidemic Multicast Protocol. See text for discussion. Reference [16] studies an instantiation.

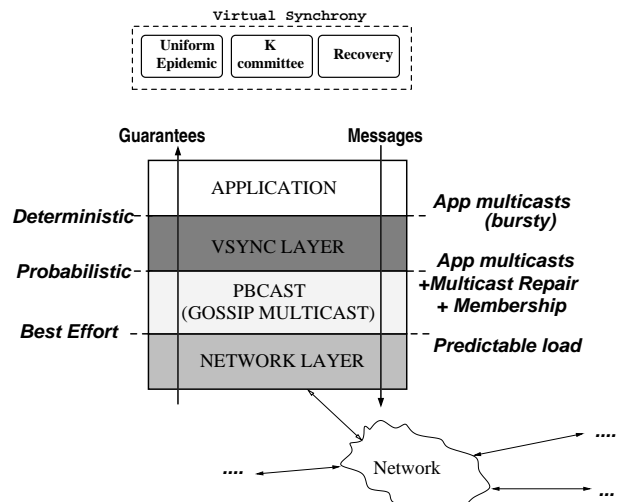


Figure 9: Converting Probabilistic Behavior Into Deterministic Reliability while Preserving Scalability: Network Stack for Probabilistic Implementation of Virtual Synchrony.

5.3 Do Compositions Inherit Properties?

We discuss our observations about the preservation of protocol properties under (and in spite of) composition. This discussion applies only to the compositions we have studied or mentioned in this paper, although general applicability is a possibility⁴. We differentiate between the *correctness* and *analytical* properties of probabilistic building blocks. Correctness properties include liveness properties such as eventual guarantees, e.g., those regarding detection of failures or eventual dissemination of a multicast when the view graph is connected. Analytical properties include performance metrics such as latency, probabilistic reliability, and per member message overhead.

Property Preservation via Composition - Correctness Properties: When two building blocks are combined using the *template* or the *augmentation* techniques, correctness properties are inherited by the composition.

Property Preservation via Composition - Analytical Properties: When two building blocks are combined using the *template* technique, analytical properties are inherited by the composition.

We present the reasoning behind these observations.

The template technique fits two components into a protocol framework (template) with specified function call interfaces defining the interaction between components. The resultant composition can then be optimized to reduce message complexity etc. (e.g., the SWIM protocol). However, the use of the protocol template implies that the resultant composition is equivalent to the constituent protocols running side-by-side in the system. As such, their original properties (both correctness and analytical) are carried over to the composition. For example, the SWIM membership protocol inherits the properties from the Distributed Ping protocol that failures are detected eventually and within an average of $\frac{1}{1-e^{-1}}$ protocol periods.

Augmentation does not affect the inheritance of correctness properties of a component C_1 as long as the composed protocol satisfies the set of preconditions specified in the original correctness property. For example, a correctness property for a uniform epidemic-based reliable multicast component says “a multicast is eventually disseminated to all members in a group if the view graph in the group stays connected”. An augmentation of the uniform epidemic with a topology aware component (e.g., where a probability distribution based on round trip time estimates is used for gossip target selection) also satisfies the above correctness property as long as the probability distribution function keeps the view graph connected.

⁴A discussion of this is beyond the scope of this paper.

Although the analytical properties of an augmented composition may be different from those of its components, we observe that the scale and reliability properties are *similar* to those of the components, in the sense that per-member overheads that vary polylogarithmically with group size suffice to achieve very high probabilistic reliability. An example is the augmented epidemic protocol discussed in Section 5.2.

6 Summary of Investigated Probabilistic Protocols

For lack of space, we summarize below the probabilistic protocols that are not discussed in detail in the current paper. All are realizable from the composable methodology described⁵.

Topology Aware Reliable Multicast: For an Internet-type hierarchical topology, members are inserted into a weak overlay structure called the Leaf Box Hierarchy using a topology aware scheme called the Contiguous Mapping [16]. Epidemic target choices are then fashioned using the positions of members in the Leaf Box hierarchy. The resulting protocol is a probabilistic multicast dissemination scheme that, compared to the Uniform Epidemic, achieves an order of magnitude reduction in the load on core network elements such as routers, links, etc. The load across network domain boundaries is a constant, as opposed to a linear variation with N for Uniform Epidemic. In turn, it suffers a small decrease in reliability and latency.

Adaptive Epidemic Multicast: A probabilistic multicast dissemination scheme (e.g., a Uniform Epidemic or a topology aware multicast) can be modularly composed with a Tree Dissemination mechanism. This yields a multicast dissemination protocol [16] that incurs low and constant overhead when failure rates (of messages and nodes) are zero or small (in comparison, Uniform Epidemic imposes an overhead that is logarithmic with N at all failure rates). The protocol automatically adapts the group overhead to increasing failure rates of members and message deliveries.

Leader Election: A leader election protocol template consists of a protocol for selection of a committee of members, for agreement among the committee members of a leader, and a multicast protocol that informs the group about the elected leader. When instantiated with the K-committee selection and Uniform Epidemic building blocks, the template generates the probabilistically correct leader election protocol of [19]. Each run of the election protocol imposes a per-member overhead that is a constant number of unicasts and multicasts. With Uniform Epidemic, the multicast overhead per run increases slowly with group size. The group overhead can be traded off for increased probability of correctness of a run of the protocol.

⁵All these protocols have been implemented as C modules, mostly layered over the Windows socket API. Experimental results from simulation and PC clusters can be obtained from the references mentioned.

Data Aggregation: This protocol aggregates individual “votes” provided by members and calculates a global aggregate, e.g., average, variance or maximum, at each of the members. Reference [17] studies a protocol that is a constraining of Uniform Epidemic within the Leaf Box Hierarchy weak overlay. A one-shot run imposes per-member message overhead and running time that grow polylogarithmically with group size. The global estimate obtained includes a fraction of votes that is close to 1.0 w.h.p. Overhead can be traded for reliability of the estimate.

Peer to peer distributed hash table (resource location and discovery system): The Kelips peer to peer routing substrate [14] uses a multicast protocol (for disseminating heartbeats) that is obtained by constraining epidemic target choices within a weak overlay (called the “affinity group scheme”) and by fashioning target choices using round-trip time estimates (topology awareness).

6.1 Composition: Obtaining Deterministic Reliability

The design methodology is powerful enough to convert probabilistic behavior into a deterministic guarantee to applications. For example, a probabilistic multicast protocol (Uniform Epidemic) can be used to design a deterministic multicast protocol (virtual synchrony) [18]. Virtual synchrony guarantees a global order on the receipt of membership changes and multicasts at all non-faulty members [3].

The composition is shown in Figure 9. The basic idea is to select a K-committee that globally orders and buffers all multicasts (and membership changes) before they go out into the group, where they are disseminated using Uniform Epidemic. A non-faulty member that misses certain multicasts can query a committee member for it, thus guaranteeing 100% reliability. Participation in the K-committee may be migratory. This is thus a modular composition of three building blocks.

Reference [18] contains an experimental evaluation of an implementation of this protocol. The protocol is able to achieve scalability by using the laziness of Uniform Epidemic, and yet guarantee reliability of multicasts. Peak throughputs obtained were 20,000 batched messages per second, at group sizes of above 100.

7 Summary

The paper has investigated a methodology for designing a suite of probabilistic protocols. These protocols use decentralization, randomization, redundancy and recovery to achieve scale and reliability. To use the methodology, a protocol designer first selects the appropriate set of building block protocols (among seven we have defined) or already-composed protocols. These are then composed using one of three techniques, and the resultant protocol is optimized. Composition techniques include fitting protocols modularly within a template, or augmenting one with

another (the latter through either fashioning or constraining). The composition techniques studied either inherit or preserve to a large degree the scalability, reliability and liveness properties from the component building blocks. Composition techniques can be used multiple times to generate a hierarchy of protocols with enriched properties. The composable methodology can be used to design solutions to several problems that are central to the design large-scale peer to peer (p2p) distributed systems in the Internet and sensor networks. These include group membership, variants of reliable multicast, data aggregation, leader election, and a resource location and discovery system.

Future Work: Our retrospective work poses a couple of questions about the design of protocols in general. What are the characteristics of protocol classes that impart to them an underlying design methodology such as the one this paper (either informal or formal)? Can one design “protocol design toolkits” that a researcher can use to explore the state space of protocols, while seated at her design table?

References

- [1] N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, 2nd edition, 1975.
- [2] M. Ben-Or. Another advantage of free choice: completely asynchronous agreement protocol. In *Proc. 2nd PODC*, pages 27–30, 1983.
- [3] K. P. Birman. *Building Secure and Reliable Network Applications*. Manning Publications and Prentice Hall, 1996.
- [4] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal Multicast. *ACM Tr. Comp. Sys.*, 17(2):41–88, May 1999.
- [5] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journ. ACM*, 43(2):225–267, March 1996.
- [6] B. Chor and C. Dwork. Randomization in byzantine agreement. *Advances in Computing Research*, 5:443–498, 1989.
- [7] B. Chor, M. Merritt, and D. B. Shmoys. Simple constant-time consensus protocols in realistic failure model. In *Proc. 4th ACM PODC*, pages 152–160, 1985.
- [8] A. Das, I. Gupta, and A. Motivala. SWIM: Scalable Weakly-consistent Infection-style process group Membership protocol. In *Proc. 2002 DSN*, pages 303–312, 2002.
- [9] A. J. Demers, D. Greene, C. Hauser, W. Irish, and J. Larson. Epidemic algorithms for replicated database maintenance. In *Proc. 6th ACM PODC*, pages 1–12, 1987.
- [10] P. Druschel, F. Kaashoek, and A. Rowstron, editors. *Proc. 1st IPTPS*, volume 2429 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.
- [11] M. J. Fischer, N. A. Lynch, and M. Patterson. Impossibility of distributed consensus with one faulty process. *Journ. ACM*, 32(2):374–382, April 1985.
- [12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns*. Addison-Wesley Pub. Co., 1st edition, 1995.
- [13] I. Gupta. *Building Scalable Solutions to Distributed Computing Problems using Probabilistic Components*. PhD Thesis, Dept. of Computer Science, Cornell University, August 2003.

- [14] I. Gupta, K. P. Birman, P. Linga, A. J. Demers, and R. van Renesse. Kelips: building an efficient and stable P2P DHT through increased memory and background overhead. In *Proc. 2nd IPTPS*, 2003.
- [15] I. Gupta, T. D. Chandra, and G. S. Goldszmidt. On scalable and efficient distributed failure detectors. In *Proc. 20th ACM PODC*, pages 170–179, 2001.
- [16] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *Proc. 21st SRDS*, 2002.
- [17] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proc. 2001 DSN*, pages 433–442, 2001.
- [18] I. Gupta, R. van Renesse, and K. Birman. Fighting fire with fire: using randomized gossip to combat stochastic scalability limits. *Journ. Quality and Reliability Engg. Intl.*, 18:165–184, May/June, 2002.
- [19] I. Gupta, R. van Renesse, and K. P. Birman. A probabilistically correct leader election protocol for large groups. In *Proc. 14th DISC, LNCS-1914*, pages 89–103, 2000.
- [20] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *Proc. 41st IEEE FOCS*, pages 565–574, 2000.
- [21] D. Kempe, J. M. Kleinberg, and A. J. Demers. Spatial gossip and resource location protocols. In *Proc. ACM STOC*, pages 163–172, 2001.
- [22] A. M. Kermarrec, L. Massoulie, and A. J. Ganesh. Reliable probabilistic communication in large-scale information dissemination systems. Tech. Rep. MMSR-TR-2000-105, Microsoft Research, Cambridge, UK, 2000.
- [23] P. Linga, I. Gupta, and K. Birman. A churn-resistant peer-to-peer web caching system. In *Proc. 1st ACM Wshop. SSRS*, October 2003.
- [24] X. Liu, C. Kreitz, R. van Renesse, J. Hickey, M. Hayden, K. P. Birman, and R. Constable. Building reliable, high-performance communication systems from components. In *Proc. 17th ACM SOSp*, 1999.
- [25] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1997.
- [26] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1st edition, 1995.
- [27] N. L. R. De Prisco, B. Lampson. Revisiting the Paxos algorithm. In *Proc. 11th WDAG*, volume 1320 of *Lecture Notes in Computer Science*, pages 111–125, 1997.
- [28] M. O. Rabin. Randomized byzantine generals. In *Proc. 24th IEEE FOCS*, pages 403–409, 1983.
- [29] A. Reed, M. Flatt, L. Stoller, J. Lepreau, and E. Eide. Knit: component composition for systems software. In *Proc. 4th OSDI*, 2000.
- [30] J. G. Schneider and O. Nierstrasz. Scripting: higher-level programming programming for component based systems. In *Tutorial, ACM SIGPLAN Conf. OOPSLA*, 1998.
- [31] R. van Renesse, S. Maffeis, and K. P. Birman. Horus: a flexible group communications system. *Comm. ACM*, 39(4):76–83, April 1996.
- [32] S.-H. Wu, S. A. Smolka, and E. W. Stark. Composition and behaviors of probabilistic I/O automata. *TCS*, 176(1-2):1–38, April 1997.