Compositional gossip: a conceptual architecture for designing gossip-based applications

Étienne Rivière
IRISA / Rennes 1 Univ.
France
eriviere@irisa.fr

Roberto Baldoni
Roma Univ. "La Sapienza"
Italy
baldoni@dis.uniroma1.it

Harry Li
Univ. of Texas at Austin
USA
harry@cs.utexas.edu

José Pereira
Univ. of Minho
Portugal
jop@di.uminho.pt

ABSTRACT

Most proposed gossip-based systems use an ad-hoc design. We observe a low degree of reutilization among this proposals. We present how this limits both the systematic development of gossip-based applications and the number of applications that can benefit from gossip-based construction. We posit that these reinvent-the-wheel approaches poses a significant barrier to the spread and usability of gossip protocols.

This paper advocates a conceptual design framework based upon aggregating basic and predefined building blocks (B^2). We show how to compose building blocks within our framework to construct more complex blocks to be used in gossip-based applications. The concept is further depicted with two gossip-based applications described using our building blocks.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; D.2.13 [**Software Engineering**]: Reusable Software

General Terms

Design, Algorithms, Standardization

Keywords

Gossip-based computing, Distributed Applications, Design framework

1. INTRODUCTION

The impact of global-scale applications such as file sharing (with BitTorrent, Overnet, ...) or the service platform underlying broadly used shopping sites such as Amazon, have led to a surge in research in large scale distributed and decentralized applications. Such applications demand a combination of fault-tolerance, decentralized control, and adaptability in dynamic settings that challenges established distributed programming frameworks. Most strikingly, something as basic as knowing the list of participant nodes is unachievable in this setting, thereby rendering many existing techniques obsolete.

In this context, gossip-based protocols have gained attention. Gossip protocols, sometimes referred as epidemic or probabilistic protocols, are deceptively simple. They are simple in that each process takes actions based upon partial local information that is repeatedly exchanged with a small set of peers. And they are deceptive in that those local steps lead to desirable global behavior such as eventually consistent states despite process and link failures.

Starting from the early proposal from Demers *et al.* for databases maintenance [10], many distributed systems researchers and developers recognized the advantages that gossip protocols offer to large-scale systems. As a result, further instances of successful gossip-based solutions exist in multiple key application areas, among which we can cite multicast [30], membership management [1, 35], overlay construction [34, 13], eventual consistency [4] or publish and subscribe systems [3, 33]. Despite some work identifying several abstractions that promote separation of concerns for gossip protocols and application design, and therefore make a step towards the reuse of code among multiple protocols [31, 20, 2, 17], there is no systematic approach to decompose applications into abstract entities capturing basic gossip-based operations.

This paper moves in that direction by presenting a conceptual framework whose aim is to highlight the gossip-based building blocks (B^2) of a distributed software architecture [1]. Such a conceptual framework would allow us to map novel applications to existing solutions. This will turn out in a major advantage for practitioners designing gossip based applications by reusing gossip based B^2s. In addition, it would help researchers to focus on well defined, self-contained problems, and better compare their competing proposals.

Section 2 presents a brief overview of gossip. Section 3 proposes the generic abstract pattern of a gossip component. Section 4 focuses on identifying gossip B^2s. Finally, in Section 5, we apply the framework to decompose two different publish/subscribe applications [3, 33], conveying the ability of our model to describe complex composed systems.

[1]It is important to note that such applications are intrinsically peer-to-peer, and we will not distinguish between computing entities based on roles, although proposed implementations can obviously take into account node heterogeneity in their design rationales.

2. GOSSIP-BASED PROTOCOLS

In a typical gossip protocol, each node maintains a (partial) view of the nodes participating in the protocol. Periodically, each node exchanges messages with a randomly selected peer from its view. In some gossip protocols, the initiating node *pushes* information to its peer. In others, the initiating node *pulls* information from the peer.

This simple communication pattern provides a robust way to disseminate information in large-scale distributed systems. Demers *et al.* [10] pioneered gossip in distributed systems by using an epidemic approach to maintain consistency across databases on Xerox's Clearinghouse servers. Birman et al. [5] later used gossip to build a probabilistically reliable multicast primitive. More recently, some have even leveraged gossip to create the structure of logical networks and overlays. This task is made easier by basic gossip-based protocols that maintain up-to-date membership information.

We now would like to present our vision of gossip and what are the —current— best uses it has, from the application designer point of view. We invite the reader not to take it as a fixed consensus, but rather as a position based on the current state of the art, a position that is prone to evolve with new advances in research. Using gossip to solve distributed systems problems usually trades off safety guarantees for liveness and/or performance. We therefore should be careful both that probabilistic guarantees are adequate for an application's requirements and that the target environment satisfies the probabilistic assumptions.

Gossip is appropriate in non time-critical systems, where the following properties are required or desired for the application: adaptability and self-organization, scalability, robustness and simplicity of design & implementation. Most distributed applications on the Internet present these requirements. Notably, content distribution networks and their affiliated mechanisms are candidates to a gossip-based implementation ; as are communication systems, at the middleware layer (event notification mechanisms, resource discovery). Other potential application we envision as primary target for a gossip-based implementation include monitoring and fault detection algorithms, and communication platforms and tools (such as RSS feeds update dissemination, discovery and maintenance of syndication systems, public keys dissemination, and more). One should note that gossip proves also useful in applications with limited scalability. Specifically, gossip has been used in detecting failures for implementing view synchronous group communication[29], garbage collection[14], and even for consensus[27]. The key advantage in these proposals is reducing the number of messages each process sends and receives from quadratic (all-to-all) to linear (gossip).

Gossip is inappropriate when a failure to meet a guarantee can lead to a catastrophic failure. As a rule of thumb, probabilistic safety guarantees are adequate when the output of the gossip protocol has a limited lifetime and consequences. In contrast, the well-known use of group membership for selecting a primary for passive replication is not compatible with probabilistic safety guarantees, as even a transient failure to identify a unique primary could lead to irreversible inconsistency. Even when gossip seems to be a good fit for a problem, there is an additional danger. Analytical models and simulations of gossip protocols make a number of assumptions that are a poor fit with real world scenarios, especially in wide area networks. In particular, assuming that links and processes fail independently is at odds with real Internet topologies. Furthermore, assuming that all processes fail benignly in a system composed of thousands of machines leaves an obvious security hole in many gossip protocols. Challenging traditional assumptions in gossip protocols and designing secure solutions is an active research area.

3. GENERIC BUILDING BLOCKS (B^2S)

In a gossip-based protocol, each node possesses a small local view of the network state: other nodes, shared data or values, ... A gossip-based protocol relies on local views at each node being eventually consistent, according to some desired property. A protocol's objectives are to: (i) have local views converge towards this properties *a.s.a.p.* and (ii) to cope with dynamism by keeping these views as coherent *i.r.t.* the whole system as possible, at any time. These two properties are important also if this protocol is the basis for other protocol functionalities.

We can define the use and construction of such views as the network component sampling service. This service provides access to a set of samples from the global network; samples can be the set of latest data items sent in the network (thus providing a notification service); if it is a uniformly random sample of nodes, the service is a peer sampling service. The peer sampling service is a sound basis to provide more complex protocols [2, 33]. We present this service as one of the blocks in Section 4.1.

Gossip-based protocols present common general functionalities: (i) selecting peers with which to exchange information, (ii) determining which set of information to share between nodes and (iii) updating the new local view.

We observe two key sets here. First, we will denote SEL (**sel**ect) the set of nodes (IP addresses) from which a peer to gossip with may be chosen. Second, we will denote EXC (**exc**hange) as the set of information (network component samples, that is, nodes, data, etc.; depending on the protocol) that is used as the local view and that is exchanged between peers.

We will in the following consider these two sets as the main inputs the application designer has to link to any gossip-based protocol provided as an application B^2. A gossip-based protocol has the goal to provide a local view to the node that rely on some global desired property. This view is the **out**put OUT of the block, and if it contains nodes it can be used as a entry for the SEL and EXC sets of another block. This provides the possibility to pile up gossip-based protocols to provide more complex protocols and properties to the application. Figure 1 depicts the generic principle of a B^2 in our framework, with its input and output. Note that only the SEL block itself is mandatory for any block, the presence and need for other depends on the protocol being implemented. We present in the following section some fundamental blocks and examples of blocks composition.

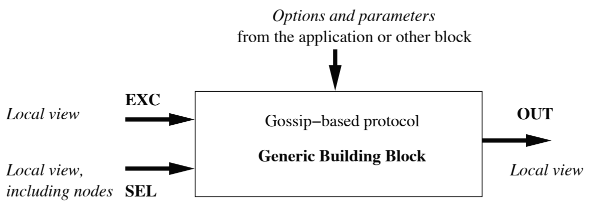

Figure 1: A generic B^2.

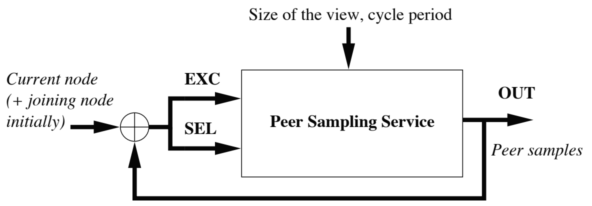

Figure 2: The peer sampling service B^2.

4. BASIC B^{2}S

We present basic B^2 of existing protocols that are promising foundations of more complex gossip-based applications. We later show how to integrate these conceptual blocks to create more complex services.

4.1 Fundamental services

Peer sampling service. The peer sampling service [17] is responsible for keeping the network consistent. It provides each node running the service with a random sample of participating nodes. It is also in charge of discovering nodes that have joined and nodes that have left, eventually integrating such information into local views. A goal of such a service is to provide a set OUT of nodes samples that shares properties with random graphs [31, 6]. Each node should have the same probability of being in another node's view as any other node. With high probability, random graphs remain connected. It has been shown that without a peer sampling service, most upper level protocols never converge to the desired state, as some nodes may never be discovered and present in some other node view. Set OUT eventually contains only alive nodes drawn at random, and no failed nodes. A newly joined node needs to appear in some other nodes' OUT sets rapidly.

Figure 2 presents the peer sampling B^2. This service requires only the node address as an input, and uses a loop-back approach from OUT to SEL to select random nodes to gossip with. Note that one needs to initialize the view of the node to some joining node in the network. Therefore, the input of the peer sampling B^2 contains the joining node in the network to be able to bootstrap the node's local view. The implementations of the peer sampling service are out of the scope of this paper, and we refer the interested reader to [17, 31]. One of the approaches is to shuffle the sets of peers for the two new OUT sets of nodes communicating during a gossip cycle, and has shown to be efficient yet very simple.

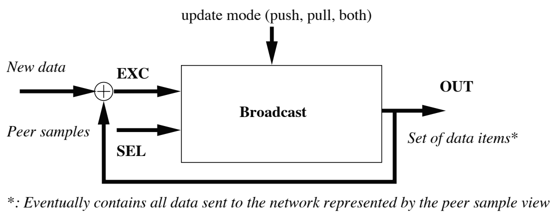
update mode (push, pull, both)
New data
EXC
Broadcast
OUT
Peer samples
SEL
Set of data items*
*: Eventually contains all data sent to the network represented by the peer sample view

Figure 3: A broadcasting service B^2.

Broadcast. Broadcasting was the first problem for which gossip-based solution were analyzed by researchers. The seminal paper by Demers *et al.* [10] presents a gossip-based protocol to disseminate database consistency information, and show the inherent potential of such an approach. The broadcasting service selects nodes to gossip with in a node sample view and exchange its set of data items with it. The exchange is either unidirectional or bi-directional (push or pull approach, versus push-pull approach). Eventually, all nodes get new data items in $O(\log N)$ steps of retransmission, provided the underlying node sample is uniformly drawn from the set of nodes to which the data should be disseminated [2]. The reliability of broadcast protocols based upon gossip-based dissemination follows a bimodal distribution [5, 7, 30, 15, 12, 11]. Broadcast protocols have been proposed in different settings, such as replication maintenance [10], ad-hoc networks [16] and eventual consistency [4].

Figure 3 represents the broadcast service. SEL is a set of peer samples from the set of nodes to which data has to be disseminated, and EXC the data to disseminate (including new data from the node). The actual implementation of the broadcast protocol is left to the programmer, and a library of blocks with diverse behaviors and possibilities will be available to tackle all potential needs for gossip-based dissemination.

4.2 Group composition

The following protocols share the common following goal. Their output (OUT set) is similar to the output of the Peer Sampling service, in that they present a set of nodes as a view for the application or other B^{2}s. However, these nodes are selected only among the nodes that satisfy some properties, or predicates. We distinguish between membership management, where groups of nodes are formed based on their advertised memberships, network slicing where a set of nodes is constructed based on the size of the set relatively to the whole network size (or sub-network, as group composition blocks can be piled up), node provisioning where a set of nodes of a desired size emerges and finally interest proximity-based composition where nodes with similar (expressed) semantics are grouped together.

Figure 4 presents the general framework for group composition protocol B^{2}s. This generic figure can be instantiated either by the application designer, or using one of the specific instantiation we present in the following. It is necessary to express either the node semantics (that can be as simple as

[2]That is the reason why using the peer sampling service makes sense here.

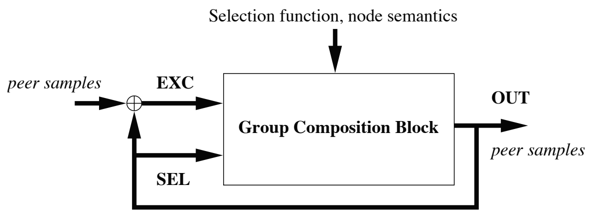

Figure 4: Generic group composition B^2.

a number or as complex as IR information about text documents the node holds), and a selection function. The selection function tells, given two sets of nodes (the current set of nodes and the one received from a gossip partner) which subset is to be kept as the new local view of the network, exported to the application or other B^2s. In the following, we present some instantiations of group composition B^2s.

Membership management. The first group composition protocol is membership management. The node semantic is simply a set of belonging to groups, expressed by the node (communities, interest in update about a data item, ...). The selection function then keeps nodes that belong to the same memberships according to the node expressed membership. The utility of the peer sampling service as an input is obvious: without the peer sampling service, a member group would be on its own, without any link to the rest of the network (which is desirable) but it would be the responsibility of newly joined nodes to find one of the member, which is not desirable. The peer discovery feature of the peer sampling service is here a sound basis, as it is for all the following group composition protocols.

Network slicing. Network slicing [18] is the operation by which group of nodes of relative size are formed. For example, an application may want to split its activities in three main activities, and devote a third of the network to each of these activities. First, one needs to use a broadcast protocol to disseminate to the group of nodes to split the information that this set has to be split in k parts.

The protocol does not need any node semantics information, as the separation is independent of the node status or data. The case where information about the node results in this node belonging to one or the other of the sets is dealt by node provisioning protocols.

Each node's protocol draws a random number between 0 and 1, and the goal is basically to have every node select peers from the fraction of this space $[0 : 1]$ split in k parts they do belong to. Eventually, on each node, the sample of nodes OUT belong to the fraction where the node belongs to. Changing k is possible by simply broadcasting a control message to all nodes onto the split network.

Node provisioning. Node provisioning [18] is quite similar to network slicing, in that the number of nodes one can ask is a function of the size of the network to be split. However, this also permits to express criteria on the node semantics itself. For instance, if the node semantics represent mean uptime, one can ask the network to be split in two parts: the 10% of nodes with highest uptime (that are more likely to be stable in the future) and on the other part the 90% other. Obviously, since there exists churn and changes in node semantics, this partition is not as stable as with network slicing. The ability of gossip-based protocols to gracefully deal with dynamicity of the network composition is a key point to support node provisioning in an efficient and elegant way.

Interest proximity. Interest proximity is the more generic group composition method. It does not necessarily cluster the network in separate groups, but rather constructs a graph of peers. The peers constructed for a node are exported to other blocks or to the application in the OUT set. These peers are those who present a proximity with the node, potentially expressed as a function over an evolving node semantic. The semantics of the node can be of any kind, and since it is used only by the selection function, genericity mechanisms permits to use the interest proximity block without any modifications. Potential instantiations of this blocks are:

- If the semantic of a node corresponds to a network coordinate [9, 23], and the selection function selects nodes at smallest distance, one can build a dissemination network that is more efficient with respect to network stress than broadcasting on top of a generic, peer sampling based network would be.
- If the semantic of a node corresponds to its set of shared data items, the selection function will select nodes with shared interest, therefore building an unstructured overlay driven by the application semantics. In peer-to-peer file sharing systems, it has been proven that flooding requests on top of such a semantic network gives more results in fewer hops [32]
- One can also use networking information from the application itself or across applications. For instance, node semantics can be the actual user's instant messaging contacts list, and one can decide on a selection function that maximizes the size of the intersection of two peers' lists (exchanged semantics), and uses it to improve search or distributed trust & reputation mechanisms. It has been shown that using the social network as presented by the application leads to better efficiency and recall for search mechanisms [35] or a better substrate for efficient broadcast [26]. The main reason is that such social networks present small-world characteristics [36].

More interest proximity based B^2s can be defined, one of such is characterized in Section 5, as part of the second B^2 of the Sub2Sub system instantiation.

4.3 Distributed computation

Another interesting family of gossip-based protocols are devoted to the computation of global properties, made available locally to all nodes, based on the exchange of nodes'

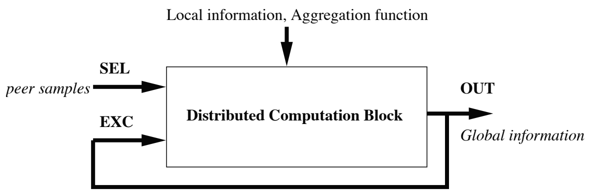
Local information, Aggregation function
SEL
peer samples
EXC
Distributed Computation Block
OUT
Global information

Figure 5: Generic distributed computation B^2.

local or observed properties. These properties can concern node load, availability, or application-specific information. Sensor networks are a perfect target application for this kind of protocols, as they often aim at computing global or digest information based on local sensor results. We present in a first paragraph aggregation related protocols, and in the second various examples of other distributed computation B^2s.

Figure 5 presents the generic block used to derive distributed computation blocks. It uses some local information as an additional input, and an aggregation function which is used to decide a global value based on local information exchanged within peers. As usual, SEL requires a set of peers from a peer sample interface, and OUT and EXC depend on the computation being performed.

Aggregation. Aggregation permits to calculate digest values (such as mean, average, maximum or minimum) of global properties represented as scalar types. For example, the reader can think of a group composition B^2 that aggregate sensor nodes, based on theirs geographical proximity. By using an aggregation B^2 over this group composition B^2, the network is able to straightforwardly calculate digest values for a given area, or for a given radius around a specific point (which can be the base mechanisms for triggering a specific action, if one of the digest values reaches a threshold). Gossip-based protocols to provide scalable and resilient aggregation exists [19, 21]. They will benefit from being released as B^2s for application designer.

Other examples. Other distributed computation protocols can be derived, either based on a modification of aggregation protocols, or based on schemes from different disciplines [8]. Our example problem is the counting of peers. In most unstructured networks, the number of peers is unknown and dynamic, but this number could however be useful for many reasons, such as tuning the protocol behavior (adaptive routing, guard delays adjustments, ...). However, computing the size of a large scale system is far from being trivial, and even if dedicated algorithmic solutions were proposed [24, 22], peer counting can be implemented using aggregation protocols. The idea is as follows [19]: all peers but one have an initial value of 0, while the initiating peer uses value 1. Using the aggregation protocol to compute the mean of all peers values, every peer will eventually be able to compute an estimation of the total number of peers in the network (as 1 over the local value exported by the aggregation B^2). Note that one may needs to use several initiators and aggregation concurrently to ensure that a sound value is available at each node in presence of nodes failures.

4.4 From randomness to structured overlays

The last example of basic B^2 family we present is devoted to the addition of structure to a initially unstructured network, such as the networks represented by peer sampling interface (either from the peer sampling service at the network level, or from any group composition block). Basically, a structured overlay imposes neighborhood of a node based on this node description, name or key (*e.g.* a hash of its name). It takes as input a set of nodes belonging to the sub-network for which the structured overlay needs to be built, and produces as an output a routing service with more guarantees or functionalities than what constrained flooding on an unstructured overlay provides. This service is most of the time, for basic protocols, a Key-Based Routing service. This provides, for any key (designing either an object, or a node) the ability to reach that key holder or representative in a limited number of redirections and in a deterministic (and often greedy) fashion.

Using a gossip-based approach to build structured overlays is relevant as it permits to alleviate the need for dealing with dynamicity, node joins and leaves explicitly [20]. Instead, the inherent self-organizing capabilities of gossip-based overlay construction protocols permits to express the required neighborhood at each peer in a deterministic manner and let the underlying peer sampling mechanism and group composition protocols deal with the dynamicity, peer deletion & addition or transient routing failures.

Figure 6 presents a generic instance of such a logical topology construction B^2. It takes as an input a set of peers from the composed group among which it will bootstrap a structured overlay. Most protocols of this kind require the node (or an object) being identified by an identifier (either a key, a name, or a more complex representation). One can create a new instance of a logical topology construction protocol by providing the generic instance with functions [34] to: (i) select the peer in SEL for local state exchange ; (ii) select the set of information EXC that is exchanged between gossip partners and finally (iii) select the new local state based on current local state and obtained information. The output OUT is depending on the overlay created: it will most probably be an API with functions for routing (unicast), queries mechanisms, communication systems, etc. It can also output the set of peers at each node as a peer sample service to other blocks.

We present two examples of such overlay bootstrapping blocks. The first one constructs a Distributed Hash Table (DHT) by linking nodes in a ring. The second constructs a structure that eventually resembles a set of skip-lists, by linking application objects and permitting range queries over their description attribute.

Bootstrapping a DHT. The first example is the bootstrapping of the Chord DHT [28], presented by Montresor *et al.* in [25]. Using a proximity-oriented gossip-based overlay construction protocol and the appropriate proximity measure, they are able to bootstrap the Chord DHT in a logarithmic number of cycles, even if the network is a random network (*i.e.* built by a peer sampling service) at the beginning. The idea is to assign nodes of interests (neighbors in the Chord structure) with the maximum utility and nodes with less

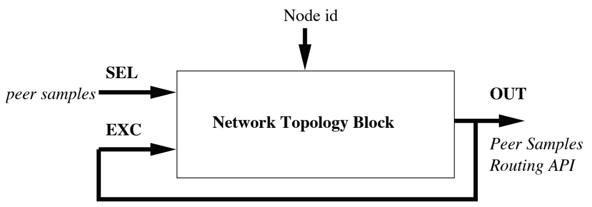

Figure 6: Generic network logical topology construction B^2.

interest a small utility. The local view evolve towards the set of peers that cover $\frac{1}{2^k}$ nodes (finger nodes), these nodes having the highest proximity. The protocol is inherently resilient to churn, the deletion and arrival of peers being manages at the underlying peer sampling service block.

Bootstrapping a set of Skip-Lists. The second example is the GosSkip overlay [13]. GosSkip links elements that can be application objects or physical nodes, in a structure that, while initially chaotic, eventually resembles a set of redundant Skip Lists. GosSkip is constructed by keeping nodes that are $1, 2, \ldots, 2^k$ hops away w.h.p., at each node. Since no hashing is used, nor a fixed size of naming space, such a structure inherently permits the support of range queries (and nearest-neighbor queries). This is something Chord can't do (although Chord can be used without hashing mechanisms to be able to handle such queries ; at the cost of an explicit load balancing algorithm and more nodes' pointers reconfiguration).

5. EXAMPLES OF APPLICATIONS

In this section we present some examples of how our proposed framework and B^{2}s will be used in practice. We present a simple group composition and redesign (in part) two software architectures, using the B^{2}s introduced in the previous sections. Note that composition is feasible only between blocks that present sets of the compatibles elements (node identifiers, data items, etc.) as an output or input ; this is something that can be entirely statically checked by a composition framework tool.

5.1 Multicast systems

A multicast system's goal is to disseminate data among a defined set of peers. Most protocols follow an ad-hoc mechanism to build the group while disseminating the information. This is both expensive and complex to adapt in different settings.

Within our framework, one can implement multicast as a composed B^2, including a group composition B^2 that permits to group peers to which the message has to be send together, and a broadcast B^2, that will disseminate a data element over the peers in the aforementioned group. It is important to note that the choice of each B^2 is independent and provides a great flexibility. For instance, simple multicast based on group membership can be implemented using one of the many B^2 for broadcasting, and a dissemination system that aims at sending data to a fraction of nodes based on their proximity of affinities will use a interest proximity group composition B^2, and a constrained flooding broadcasting B^2.

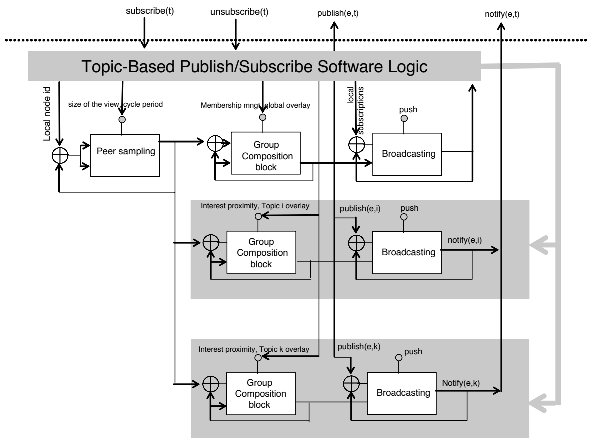

Figure 8: Architecture of TERA a topic based publish subscribe system [3].

5.2 Publish/subscribe systems

Publish/subscribe is a distributed communication paradigm, where participants to the system can act both as producers (publishers) and consumers (subscribers) of information. Publishers inject information in the system in the form of events, while subscribers declare their interest in receiving some of the published events, by issuing subscriptions. Once an event is published, for each subscription whose conditions are satisfied by the event, the corresponding subscriber must be notified.

In the following we re-design through the conceptual architecture the scalable topic based publish/subscribe architecture, TERA, proposed in [3] and the content-based publish/subscribe system, Sub-2-Sub [33].

First example: TERA [3]. TERA is a topic-based publish/subscribe system that achieves scalable event diffusion by implementing traffic confinement, that is, events have a high probability to reach *only* interested subscribers. To this aim, subscribers with common interests (e.g. subscribers to the same topic) are clustered together, using a membership group composition B^2. In this way, once the event belonging to a certain topic reaches one member of the cluster of that topic, its diffusion can be limited to that cluster. The authors used gossiping for keeping connected both a global overlay including all the nodes (using a peer sampling B^2) and a specific overlay for each single topic clustering all nodes that subscribed for that topic (using a membership group composition B^2. A simplified conceptual architecture of TERA using B^{2}s defined in the previous section is shown in Figure 8.

The peer-sampling B^2 provides basic connectivity to avoid network partitions, to discover new nodes and to remove failed/left nodes. Output OUT of this service is a parallel input to a set of group composition blocks SEL sets, one for membership management of the global overlay and one per topic the node has subscribed to. If the node subscribes to (respectively un-subscribes from) a topic, a composed block formed by a group composition block and a broadcast one is dynamically added to (resp. removed from) the system.

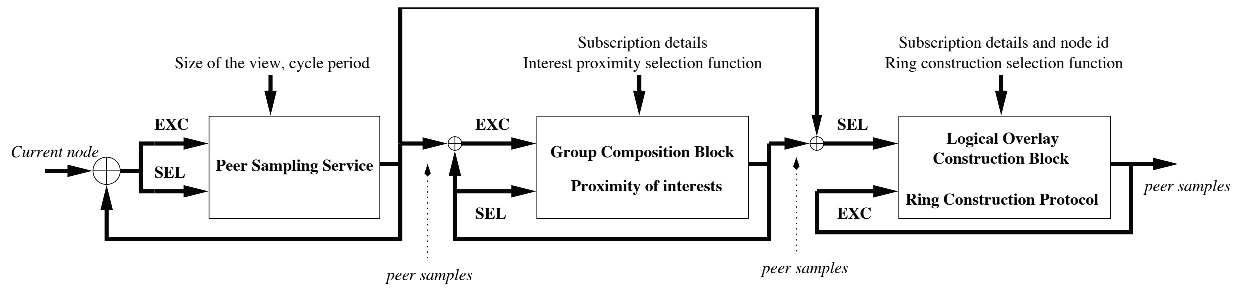

Figure 7: Sub-2-Sub composition of blocks for content-based pub/sub overlay construction [33].

Information about subscriptions currently handled by a node are diffused regularly by the node itself on the global overlay through the broadcast B^2. This dissemination has multiple objectives: firstly to let nodes be aware of possible new topics instantiated in the system, secondly, to merge multiple distinct groups handling the same topic (due to concurrent creation of the same topic on different nodes) and to help the routing of an event to reach the cluster related to the topic the event belongs to. This routing is done through a random walk on the general overlay. The output of that broadcast B^2 is an input for the TERA software logic that is responsible for realizing from an operational viewpoint previous objectives.

Second example: Sub-2-sub [33]. Sub-2-Subs is a content-based publish/subscribe system. It is a structured overlay built using only gossip-based protocols, that permits efficient and collaborative support to content-based publish and subscribe. Subscribers express their interest with a subscription formed of predicates over a set of attributes. The space of possible events is a multidimensional naming space (a hyper rectangle of volume = 1). Predicates are expressed over this naming space. A set of predicates form a hyper-rectangle in the space of possible events. The objective of the overlay construction mechanisms, implemented with our B^2s and presented in Figure 7, is to link subscriptions that are "near" together (that is, subscriptions for which common events may occur), so that subscribers with similar interests get clustered together. This is the job of the second block represented in the middle. This is a group composition block instance, that links near subscription together, according to a proximity function provided by the application designer. It uses the output of the peer sampling service (nodes drawn at random from the entire network), which permits the discovery of new peers, resilience to dynamism and network partitions. The union of the output of the peer sampling service and the group composition service is further given to the logical topology construction block, whose goal is to form efficient dissemination structures for every possible event that can occur. To this end, a random id is assigned to each subscription and the protocol creates ring-like structure, that is used along with the proximity links for efficient dissemination. Details of the protocol for dissemination are omitted and only the overlay construction blocks are shown for space reasons.

An interesting remark is that blocks are pretty independent from each other. One can decide to replace the logical topology construction block by a GosSkip-like approach, for instance, or to alternatively use a flooding mechanism on top of the group composition block to spread events in a probabilistic way. The proposed framework is the ideal setting to test such different solutions in an unified and simple way.

6. CONCLUSION AND PERSPECTIVES

Raising the level of abstraction when designing a large software architecture has been one of the key factors for the success of modeling technologies such as UML. Reasoning on models allows to mask details, bring out the big picture, and focus on different aspects of the architecture. This reflects the position we took in this paper by advocating the need of modeling gossip-based applications through a conceptual architecture. This should help to deal with the complexity of embedding gossip mechanisms inside an application avoiding thus an ad-hoc design that limits code reuse. On the other hand, some work can be done to factorize the costs of several co-located Gossip protocols (for instance by sharing communication partners, or using other protocols' local views, etc.). More, the use of a conceptual architecture helps a reader to localize and to better understand the use of gossiping inside a complex software architecture and, additionally, to compare two different architecture with respect to their usage of gossiping. Finally, our proposal is a common platform for researchers to compare and test new protocols, by inserting them in larger software architectures in a simple way.

7. REFERENCES

[1] A. Allavena, A. Demers, and J. E. Hopcroft. Correctness of a gossip based membership protocol. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles Of Distributed Computing*, pages 292–301, New York, NY, USA, 2005. ACM Press.

[2] O. Babaoglu, M. Jelasity, A.-M. Kermarrec, A. Montresor, and M. van Steen. Managing clouds: a case for a fresh look at large unreliable dynamic networks. *SIGOPS Operating Systems Review*, 40(3):9–13, 2006.

[3] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. T. Piergiovanni. Tera: Topic-based event routing for peer-to-peer architectures. In *Proceedings of the Inaugural International Conference on Distributed Event-Based Systems*, Toronto, Canada, 2007.

[4] R. Baldoni, R. Guerraoui, R. Levi, V. Quema, and S. T. Piergiovanni. Unconscious eventual consistency with gossips. In *Proceedings of the Eight International Symposium on Stabilization, Safety and Security of Distributed Systems (SSS 06)*, Dallas, Texas, Nov. 2006. LNCS.

[5] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[6] F. Bonnet, F. Tronel, and S. Voulgaris. Brief Announcement: Performance Analysis of Cyclon, an Inexpensive Membership Management for Unstructured

P2P Overlays. In *DISC 2006*, pages 560–562, Stockholm, Sweden, Sep. 2006.

[7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530, 2006.

[8] P. Costa, V. Gramoli, M. Jelasity, G. P. Jesi, E. L. Merrer, A. Montresor, and L. Querzoni. Exploring the interdisciplinary connections of gossip-based systems. *SIGOPS Operating Systems Review*, 2007.

[9] R. Cox, F. Dabek, F. Kaashoek, J. Li, and R. Morris. Practical, distributed network coordinates. In *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, Massachusetts, November 2003. ACM SIGCOMM.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles Of Distributed Computing*, pages 1–12, New York, NY, USA, 1987. ACM Press.

[11] P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.

[12] P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic information dissemination in distributed systems. In *Computer*, volume 37, pages 60–67. IEEE, May 2004.

[13] R. Guerraoui, S. B. Handurukande, K. Huguenin, A.-M. Kermarrec, F. L. Fessant, and É. Rivière. Gosskip, an efficient, fault-tolerant and self organizing overlay using gossip-based construction and skip-lists principles. In *P2P '06: Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing*, pages 12–22, Cambridge, UK, Sep. 2006. IEEE Computer Society.

[14] K. Guo. *Scalable Message Stability Detection Protocols*. PhD thesis, Cornell University, Computer Science Department, May 1998.

[15] I. Gupta, A.-M. Kermarrec, and A. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *Proceedings of the 21st Symposium on Reliable Distributed Systems (SRDS'02)*, page 180, Oct. 2002.

[16] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, 2006.

[17] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*, pages 79–98, New York, NY, USA, 2004. Springer-Verlag New York, Inc.

[18] M. Jelasity and A.-M. Kermarrec. Ordered slicing of very large-scale overlay networks. In *Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing (P2P 2006)*, Cambridge, UK, 2006.

[19] M. Jelasity, A. Montresor, and O. Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3):219–252, 2005.

[20] M. Jelasity, A. Montresor, and O. Babaoglu. The bootstrapping service. In *Proceedings of International ICDCS Workshop on Dynamic Distributed Systems (ICDCS-IWDDS'06)*, Lisboa, Portugal, July 2006. IEEE Computer Society.

[21] S. Kashyap, S. Deb, K. V. M. Naidu, R. Rastogi, and A. Srinivasan. Efficient gossip-based aggregate computation. In *PODS '06: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 308–317, New York, NY, USA, 2006. ACM Press.

[22] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers. Decentralized schemes for size estimation in large and dynamic groups. In *NCA '05: Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, pages 41–48, Washington, DC, USA, 2005. IEEE Computer Society.

[23] J. Ledlie, P. Pietzuch, and M. Seltzer. Stable and accurate network coordinates. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, page 74, Washington, DC, USA, 2006. IEEE Computer Society.

[24] L. Massoulié, E. L. Merrer, A.-M. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles Of Distributed Computing*, pages 123–132, New York, NY, USA, 2006. ACM Press.

[25] A. Montresor, M. Jelasity, and O. Babaoglu. Chord on demand. In *P2P '05: Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P'05)*, pages 87–94, Washington, DC, USA, 2005. IEEE Computer Society.

[26] J. A. Patel, I. Gupta, and N. Contractor. Jetstream: Achieving predictable gossip dissemination by leveraging social network principles. In *NCA '06: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications*, pages 32–39, Washington, DC, USA, 2006. IEEE Computer Society.

[27] J. Pereira and R. Oliveira. The mutable consensus protocol. In *23rd IEEE Symposium on Reliable Distributed Systems*, pages 218–227, Florianpolis, Brazil, Oct. 2004. IEEE, IEEE Computer Society.

[28] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, pages 149–160, San Diego, California, Aug. 2001.

[29] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. Technical Report TR98-1687, Cornell university, 28, 1998.

[30] W. Vogels, R. van Renesse, and K. Birman. The power of epidemics: robust communication for large-scale distributed systems. *SIGCOMM Computer Communication Review*, 33(1):131–135, 2003.

[31] S. Voulgaris. *Epidemic-Based Self-Organization in Peer-to-Peer Systems*. PhD thesis, Vrije Universiteit, Amsterdam, November 2006.

[32] S. Voulgaris, A.-M. Kermarrec, L. Massoulié, and M. van Steen. Exploiting semantic proximity in peer-to-peer content searching. In *10th International Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004)*, China, May 2004.

[33] S. Voulgaris, É. Rivière, A.-M. Kermarrec, and M. van Steen. Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. In *IPTPS'06: the fifth International Workshop on Peer-to-Peer Systems*, Santa Barbara, USA, Feb. 2006.

[34] S. Voulgaris and M. van Steen. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In A. K. Marcus Brunner, editor, *Self-Managing Distributed Systems: 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2003*, volume 2867, pages 41–54. Springer-Verlag GmbH, 2003.

[35] S. Voulgaris and M. van Steen. Epidemic-style management of semantic overlays for content-based searching. In P. D. M. José C. Cunha, editor, *: Euro-Par 2005 Parallel Processing: 11th International Euro-Par Conference*, volume 3648, page 1143, sep 2005.

[36] T. Wong, R. Katz, and S. McCanne. An evaluation of preference clustering in large-scale multicast applications. In *IEEE Infocom '00: The Conference on Computer Communications, Volume 2: 19th Annual Joint Conference of the IEEE Computer and Communications Societies (IC '00)*, 2000.