Group Therapy for Systems:
Using link attestations to manage failures

Michael J. Freedman, Ion Stoica, David Mazières, and Scott Shenker
New York University, U.C. Berkeley, and Stanford University

Abstract

Managing failures and configuring systems properly are of critical importance for robust distributed services. Unfortunately, protocols offering strong fault-tolerance guarantees are generally too costly and insensitive to performance criteria. Yet, system management in practice is often ad-hoc and ill-defined, leading to under-utilized capacity or adverse effects from poorly-behaving machines.

This paper proposes a new abstraction called *link-attestation groups* (LA-Groups) for building robust distributed systems. Developers specify application-level correctness conditions or performance requirements for nodes. Nodes vouch for each other's acceptability within small groups of nodes through digitally-signed link attestations, and then apply a link-state protocol to determine these group relationships.

By exposing such an attestation graph, LA-Groups enable the application (1) to make more informed decisions about its level of fault tolerance, security, or performance, and (2) to improve such properties by fluidly partitioning large-scale systems into small, better-suited groups. To demonstrate how LA-Groups can benefit systems, we outline designs for several applications—structured overlay routing, multicast, file sharing, and worm containment—that are robust against various failures.

1 Introduction

One of the main challenges in building Internet-scale distributed systems is handling failure. Traditional, smaller-scale systems are composed from a fixed set of potential machines within a single administrative realm. In such a setting, when nodes fail, the system can compensate by masking the fault until the nodes can be repaired. Using protocols such as BFT state-machine replication [4], systems can even tolerate malicious failures so long as the fraction of failed nodes remains sufficiently low.

Unfortunately, at the scale of the Internet, there is no fixed set of potential participant machines and no guarantee that failed nodes can eventually be repaired. Even if a system could afford the quadratic communications cost of naïvely using BFT, there is no meaningful way to reason about the fraction of faulty nodes. Moreover, for many applications, there is not even a globally correct notion of failure. For example, in the face of a network partition, it makes sense for a collaborative spam filtering or worm detection system to continue operating with the nodes in each partition considering the other set to have failed.

Rather than attempt to mask failures, we argue that Internet-scale systems should expose the failure behavior of nodes to higher-level applications. Applications in turn should structure themselves so that each node depends most critically on the nodes it considers most reliable. Toward this end, we introduce an abstraction called *link-attestation groups* (LA-Groups). The basic idea is to apply a link-state protocol to determine trust relationships within small groups of machines participating in a potentially much larger overlay network.

In the simplest case, LA-Groups can simply track network connectivity and liveness so as to allow applications to avoid problems caused by non-transitive routing. More generally, however, LA-Groups use an application-specific notion of reliability and correctness, so as to map which pairs of nodes consider each other reliable. Nodes vouch for each other's acceptability through digitally signed *link attestations*, allowing nodes to prove trust paths to other nodes.

The next section describes the LA-Groups abstraction. We then discuss the potential application of LA-Groups to three problems: non-transitive routing in structured routing overlays, grouping well connected nodes in end-system multicast, and distributed worm detection and filtering. Finally we discuss related work and conclude.

2 Group-oriented monitoring

We consider systems in which a group abstraction can be applied. Often, large-scale systems naturally decompose into small groups; other times, system designers can embed groups into seemingly flat systems to achieve desired fault-tolerance or performance properties.

Within a particular group, nodes actively monitor their neighbors and attest to each others' behavior. These *link attestations* are shared among group members, so that each node can generate a directed attestation graph describing the relationship between all known pairs of nodes

```
// Group management operations
GroupID create ()
void join (GroupID, NodeID[], cb<bool>)

// Current view of group attestations
GroupID[] groups ()
Node[] attestsOut (GroupID, NodeID)
Node[] attestsIn (GroupID, NodeID)
void attestsChange (GroupID, cb<Node>)

// Attest correctness of nodes
void startAttest (GroupID, NodeID, info)
void stopAttest (GroupID, NodeID)
```

Figure 1: The LA-Groups API

in a group. If a directed edge does not exist between nodes, either the nodes cannot directly communicate or one is behaving unacceptably according to the other's application criteria. The set of all nodes that can be reached from a node along its attestation links is called its *view*. We expect groups to be on the scale of tens of nodes.

Possessing such an attestation graph is a powerful tool for system management. While an attestation may carry application-specific information, basic graph properties can be generalized if this link information can be represented by unit-less costs. Consider how various systems use graph properties to achieve their desired goal: Quorum systems [10] ensure fault-tolerance by requiring a graph with some minimum vertex cut (the quorum threshold). Secure gossiping protocols [9] and decentralized key distribution [13, 16] require multiple vertex-disjoint paths. Structured routing overlays benefit from strongly-connected components (§3.1). Multicast systems can optimize message transmission using shortest path and max-flow algorithms (§3.2). Worm or spam filtering can use a graph to encode trust relationships (§3.3).

LA-Groups assume that nodes can use digital signatures to authenticate communication and that clocks of mutually-acceptable nodes are loosely synchronized. Other than that, LA-Groups is designed for the Internet, in which communication is asynchronous and unreliable, and nodes may crash or behave arbitrarily at any time (due to software bugs, misconfiguration, malicious intent, etc).

2.1 The LA-Groups abstraction

In this section, we describe the LA-Group interface and group membership attestation model. Figure 1 lists the proposed API for LA-Groups. A node can create a new group associated with a new globally-unique identifier (e.g., a random 160-bit value). A node attempts to asynchronously join a group by attesting *to* certain specified members and is notified by a callback upon completion. The join succeeds only if the node received an attestation *from* at least one group member.

The API provides three functions for group membership information: all groups to which a node belongs and all valid attestations from and for a node within a group. These attestations include the attestee's node ID (for example, a network address or a cryptographic hash of its public key) as well as optional application-specific $info$ about that node. This $info$ may include, for example, a node's observed throughput or uptime. (We specify that the API returns $node$, instead of an attestation, to reflect that signature and freshness verification is transparently handled by the LA-Groups layer.) A node can reconstruct its entire directed attestation graph by recursive calls to the attestations interface. Finally, a node can register to be notified whenever it loses an attestation from another group member.

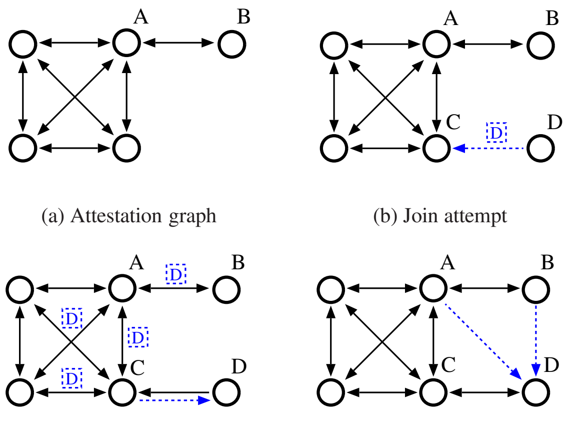
The API provides a stateful management interface: A node can start and subsequently stop attesting to another's correctness. Application writers can employ arbitrarily complex failure tests, from general liveness or performance target checking, to cryptographic data verification, and to anomaly- or voting-based detection of invalid responses. When nodes fail such tests—or pass them in the case of pending members—an application simply records this state change with the LA-Groups API.

LA-Groups can provide API support for popular properties on the attestation graphs when $info$ can be represented by a unit-less cost. Such common properties include shortest path to a node, number of vertex-disjoint paths (of length $\leq l$), max-flow min-cut of a subgraph, strongly-connected subgraphs, clustering coefficient, etc. The next section includes applications based on some of these properties.

Refreshing, disseminating, and verifying attestations occurs transparently to the application. As long as the application does not stop attesting to a group member, the LA-Groups layer will resend a fresh attestation every $< T_a$ seconds, where T_a is the period for which the attestation is valid. Given that liveness checking requires continuous communication in the first place, these attestations may be piggybacked on existing messages.

Attestations and membership transcripts. The fundamental building-block of LA-Groups is the *attestation*. Each attestation is comprised of the group identifier, the identity of the attester and attestee, any optional $info$, and an expiration time (generated as $now + T_a$). The attestation is signed by the attester's private signature key.

LA-Groups allow nodes to demonstrate attestations from a group, even to non-members, through *membership transcripts*. A membership transcript of size k is comprised of k attestations. To verify a membership tran-



(a) Attestation graph

(b) Join attempt

(c) Flood attestation

(d) Testing new node

Figure 2: Managing the attestation graph

script, a node checks whether (1) the k signatures are valid and (2) the attestations are not yet expired. This basic scheme requires k signature verifications. We describe a cryptographic optimization in Section 2.2 to decrease its length and cost to that of a single signature.

We do not require any special public-key-distribution system. LA-Groups is designed to work in fully-decentralized environments. As described, a node ID may simply be a node's network address or the cryptographic hash of its public key. Of course, LA-Groups do not preclude other key-management approaches, such as using a certificate authority or trusted out-of-band distribution.

We note that symmetric-key message authentication codes (MACs) can instead be used to authenticate communication between *directly*-connected nodes; a performance optimization that can have especial benefit for densely-connected networks. Specifically, a node need not necessarily verify the signed link attestations it receives from a neighbor if the two nodes have previously negotiated a MAC key (for instance, during the first exchange of attestations).

Views and link-state announcements. LA-Groups provide weak consistency within groups. A node's *view* is defined by the directed attestation graph rooted out from itself, call this node B. Node B is added to node A's group view as soon as A receives a valid attestation from an existing node in A's view. In other words, there exists an attestation path from A to B.

We use two techniques to ensure that attestation changes are reflected across nodes' views in a timely manner. First, the expiration time of attestations bounds the period with which attestations may be stale by T_a. Second, LA-Groups uses a link-state protocol to actively disseminate attestations among group members.

Figure 2 demonstrates LA-Groups' membership and attestation dissemination protocols. Fig. 2a shows the underlying attestation graph, where a solid arrow is drawn from a node attesting to a group member. Node A is part of a clique with four nodes, but also has a link to the stub node B. In Fig. 2b, a new node D joins by sending an attestation to an existing group member C. C begins monitoring D (as shown by the dotted arrow in Fig. 2c) and proceeds to propagate a link-state announcement to all reachable nodes in its attestation graph. We omit describing various optimizations to reduce the network overhead of this operation. Finally, when nodes hear about a new node D, they begin monitoring it (Fig. 2d) to determine whether they should generate attestations for it.

While attestations may represent complex application- or context-specific decisions, the simplest type of attestation expresses reachability and liveness, i.e., that a node continues to respond to application-level pings. We expect that such attestation graphs in many overlay networks will have very high degree and short path lengths, and thus attestation announcements will propagate quickly and result in a low incidence of routing instabilities.

2.2 Discussion

Stateful end-to-end monitoring. The LA-Groups approach allows for end-to-end monitoring that can expose system properties to applications. While some types of failures can be detected using only application-agnostic stateless monitoring (e.g., process crashes, message corruption), a large class of failure detection mechanisms require state and/or application-level knowledge. Such stateful monitoring includes anomaly detection for performance faults (e.g., disk throughput, network latency or throughput, packet flooding and denial-of-service attacks, or application delays). Or, application-specific monitoring may use a voting-based approach to determine the probable correctness of nodes' responses (e.g., for DNS resolution, name-to-public-key bindings, HTTP responses, or network-file-system operations) and then generate or withdraw attestations accordingly.

Correctness, not failure, attestations. LA-Group nodes propagate timed correctness attestations to other group members, instead of explicit failure reports. Failure reports could simply be dropped by a faulty node: the onus must be for a node to prove membership, not to prove a lack of non-membership. This is especially important when providing membership transcripts to *non-group* nodes, who do not receive link-state announcements. LA-Groups' transcripts provide such a proof without requiring any third-party interaction. In using expiration periods, LA-Groups provide a relaxed consistency model of membership, as some nodes' views may still express an attestation link up to T_a time after a node stops attesting to one of its neighbors.

Compact signature representation. We can provide a more compact representation of membership transcripts and decrease verification overhead through the use of *multi-signatures* [3]. Multi-signatures are a variation of normal signatures by which groups of nodes, each holding a unique signing key, can produce a single signed message bearing the endorsement of all group members.

Informally, nodes individually generate signatures, which are aggregated by multiplying their signatures together to a compact representation with length equal to a single signature. A node can verify the signature using the corresponding aggregated public verification key, formed by multiplying together the individual keys of each signer. A node can compute this aggregated key a single time (and amortize subgroup computations) to perform a single verification when refreshing k attestations. Multi-signatures are practical; for example, there exists a simple construction based on Gap Diffie-Helman groups [3].

LA-Groups can leverage this compact representation in two ways: First, it greatly reduces the overhead needed to transfer entire views to new nodes, *i.e.*, so it can immediately receive a snapshot of the attestation graph, as opposed to waiting until receiving individual link-state announcements. Second, it can help improve the efficiency of link-state announcements, as intermediate nodes may collect individual announcements from neighbors before retransmitting aggregated ones.

A t-robust group membership protocol. A conceptually simple use of LA-Groups is to generate a t-robust group membership protocol: If attestations represent a node's belief in another's continued correctness, then a group corresponds to k nodes that have attestations for and from at least t other group members. A static threshold of t allows the group to withstand $t-1$ faults over its lifetime and is traditionally used in quorum systems [4, 10]. A dynamic threshold, on the other hand, enables the group to enforce properties as a function of group size as it grows or shrinks, although requires consensus on group membership during view changes. An earlier form of this work restricted itself to precisely this problem; the LA-Groups approach provides a more flexible generalization.

3 Applications

This section attempts to demonstrate that the process of building and managing distributed systems can be aided by using the LA-Groups abstraction. We outline the design of several potential applications: (1) a structured routing overlay robust against non-transitive connectivity, (2) a multicast system that tolerates poorly-connected nodes, and (3) a file-sharing system resistant to mislabeled data and cooperative worm containment resistant to invalid worm signatures. These examples show three general classes of applications—flat, hierarchical, and partitioned systems—to which LA-Groups can be naturally applied.

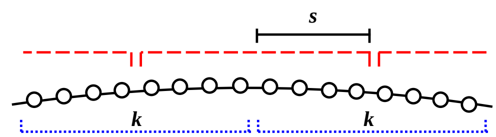


Figure 3: Structured overlay routing: Staggered LA-Groups share link-state information for robust routing.

3.1 Flat: Structured routing overlays

Problem statement. Nodes in structured routing overlays [14, 15] claim "ownership" of some sequential range of a semantic-free identifier space. Each node peers with other nodes selected from a specific ID distribution. Routing to the node controlling some target identifier proceeds by greedily visiting nodes whose IDs are progressively closer to the target.

In the Chord overlay [15], which uses a ring-like ID space of the integers modulo 2^{160}, there are two types of connections. Fingers are selected from an exponential distribution and ensure that routing is short in expectation ($O(\log n)$ hops for n-node networks). Successors immediately follow a node in the ID space and help ensure the correctness of routing.

In all existing structured routing overlays, non-transitive connectivity or inconsistent routing views can cause greedy routing to hit a local minimum and never reach the identifier's immediate successor [6].

LA-Groups for robust routing. We propose using LA-Groups' link-state information to improve the *correctness* of routing under non-transitive conditions or inconsistencies. For this application, attestations only express reachability and liveness.

Figure 3 shows an arc of the Chord ring: Nodes are arranged sequentially based on their random, unique node IDs. Each node is a member of at least two LA-Groups, each of size approximately k (say, 20). These groups can be seen as stretching clockwise and counter-clockwise with respect to a node, overlapping at $s \approx \frac{k}{2}$ nodes. They replace the traditional successor lists in Chord (or the leaf sets in Pastry [14]).

The strongly-connected graph property of an LA-Group ensures that all group nodes can robustly communicate, either directly or through some intermediate that relays packets. The two "staggered" groups ensure robust communication between groups, provided that any single pair of nodes in the overlap set can communicate. This simple LA-Group-based approach may replace the various mechanisms we describe in [6] with which deployed routing overlays handle non-transitivity.

This construction can adapt to naturally handle churn. When a node leaves the network, existing group members extend the group along the Chord ring, attempting to maintain a group size of approximately k. When several new nodes join the group, nodes near the group's edges leave it, possibly joining its immediate neighbor. To handle large-scale system growth or shrinkage—as opposed to fluctuations around a steady-state network size—a node may belong to more than two groups, albeit usually only temporarily: a new group is created by a node in the overlap when its current group is detected to be too large ($> k$ members), while groups merge when they are too small.

LA-Groups for robust storage. Structured routing overlays were proposed as a building block for distributed hash tables: Data can be stored under a key ID and subsequently retrieved. The central challenge of distributed storage applications is to ensure that data can be retrieved robustly under membership changes.

Most systems propose replicating data at a node's successors with a fixed replication factor [14, 15]. LA-Groups offers an interesting twist to this approach. Applications can dynamically tune the replication factor to achieve some desired level of fault-tolerance, based on the minimum vertex cut of the subgraph. For example, if two of a node's successors are both reachable only via a single group member due to routing problems, the system must replicate data to three nodes, not two, in order to withstand any single node failing.

3.2 Hierarchical: Multicast

Problem statement. The desired property of an end-user multicast system is straightforward: parents in a multicast tree should reliably transmit packets to their children, ideally at high throughput rates. The traditional design of such systems is for new nodes to join as leaf nodes, growing the depth of the tree. While this approach provides an intuitive and elegant design, poorly-connected intermediate nodes—those having low upstream bandwidth or high latency—can cause significant delays at all the nodes' children [2]. Tree management becomes a central and challenging problem.

LA-Groups for optimizing multicast. Each node attests to others' reachability (and hence a willingness to relay packets to them), as well as additional network information such as latency, throughput, and uptime (or mean-time-to-failure for long-lived systems [17]).

Nodes form a multicast tree with approximate degree k at each level. We define an LA-Group as a parent and its children; call this a level-i group if the parent is depth i from the root. Internal tree nodes belong to two groups; leaves belong to exactly one group. Of course, a parent cannot maintain high throughput to every child for larger values of k. Instead, a parent sends packets *directly* to some fraction of these children and appends routing information telling its children to which siblings to relay the packets. Given link-state information about the paths between siblings, the parent can directly optimize the data transmission based on network conditions (at the timescale equal to the refresh period of attestations). Finally, nodes near the root do not attest to nodes with short uptimes or poor mean-times-to-failure. Therefore, such nodes are forced to remain at tree's leaves, and the system can statistically minimize interruptions due to failures.

3.3 Partitioned: Trust networks

Problem statement. Securing applications presents special challenges when normal cryptographic techniques cannot be used to verify the correctness of data. For example, participants in file-sharing systems have complete freedom to associate any metadata with files and thus can pollute the system's searchable index with mislabeled files. In cooperative worm containment, once a firewall detects a worm, it disseminates the worm's signature to other firewalls, which in turn block the packets containing the signature [8]. Unfortunately, if a firewall distributes incorrect signatures, this can amount to a denial-of-service attack. In both cases, applying a traditional consensus protocol requires a node to wait for τ matching responses before taking action, where τ must exceed the total number of colluding nodes.

LA-Groups for trust graphs. Nodes generate an attestation graph that correlates to the trust relationships in the system. In many contexts, such trust relationships already exist. Organizations are likely more willing to engage in cooperative worm containment or spam filtering with existing partners (such as peering ISPs, collaborating research universities, business partners, and the like). Several studies of file-swapping and e-mail networks have demonstrated the natural development of interest communities [7]. Systems such as IRC, Waste, and BitTorrent explicitly join nodes into smaller communication groups—even while physical nodes may belong to multiple such groups—for precisely this reason.

Given an attestation graph from LA-Groups, a node can quantify its trust in a protocol answer using various metrics. For example, it may simply require consensus among t immediate neighbors (for $1 \leq t \ll \tau$) or instead may check that t attestation paths to a single member are vertex disjoint [13, 16]. The former provides robustness up to $t-1$ faulty nodes, while the latter claims that, historically, this member appears correct to $\geq t$ nodes.

The application must perform some type of response verification to ensure other nodes' continued compliance. In our examples, humans can check a file's validity after download, while dynamic taint analysis [12] can help verify worm signatures. If verification fails, a node stops attesting to a neighbor, limiting the duration for which incorrect metadata remains in such systems. Such verification is costly, however: LA-Groups improve system behavior by helping to achieve a low false-positive rate.

4 Related work

The distributed systems community has proposed a number of fault-tolerance protocols over the years that attempt to *mask failures*. Classic examples are virtual synchrony communication [1], replicated quorum systems [10], and replicated state machines [4]. The community also has developed the theoretical underpinnings of unreliable failure detectors [5] that enable group membership and consensus protocols. Largely, these protocols focus on abstraction generality and correctness guarantees, at the cost of high communication overhead and limited scalability.

Link-state protocols [11] are used by intra-domain routing such as OSPF in order to find all-pairs shortest paths. An LA-Group node is concerned about enumerating the strongly-connected subgraph that includes itself, not simply calculating the shortest path to all nodes.

We sketched various applications that may benefit from applying LA-Groups in Section 3. Due to space limitations, we omit the vast body of work associated with these applications. We are unaware, however, of any prior work that has proposed using application-level attestations to expose distributed system health and even embracing partitions to resist failures in large-scale systems.

5 Summary

Handling failures and organizing nodes are central challenges when building Internet-scale distributed systems. Rather than attempt to mask failures, we argue that systems should expose the process behavior of nodes to higher-level applications. Applications in turn should structure themselves into process groups that reflect nodes' observed reliability and correctness.

This paper introduces an abstraction called *link-attestation groups*. Nodes vouch for each other's acceptability through digitally-signed *link attestations*, then apply a link-state protocol to determine trust relationships in a group. By exposing such an attestation graph, the application can make more informed decisions that directly impact its fault tolerance, security, or performance: with whom to peer, from where to download a security filter, the extent with which to replicate data, and so forth. We believe that LA-Groups can help system designers directly reason about desired graph properties and, from doing so, build more robust systems.

References

[1] K. P. Birman. Replication and fault-tolerance in the ISIS system. In *SOSP*, Dec 1985.

[2] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Computer Systems*, 17 (2), 1999.

[3] A. Boldyreva. Efficient threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, Jan 2003.

[4] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI*, Feb 1999.

[5] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2), 1996.

[6] M. J. Freedman, K. Lakshminarayanan, S. Rhea, and I. Stoica. Non-transitive connectivity and DHTs. In *WORLDS*, Dec 2005.

[7] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *INFOCOM*, Mar 2001.

[8] J. Kannan, L. Subramanian, I. Stoica, and R. Katz. Analyzing cooperative containment of fast scanning worms. In *SRUTI*, Jul 2005.

[9] D. Malkhi, Y. Mansour, and M. Reiter. On diffusing updates in a byzantine environment. In *IEEE SRDS*, Oct 1999.

[10] D. Malkhi and M. Reiter. Byzantine quorum systems. *J. Distributed Computing*, 11(4), 1988.

[11] J. McQuillan, I. Richer, and E. Rosen. The new routing algorithm for the Arpanet. *IEEE Trans. Communications*, May 1980.

[12] J. Newsome and D. Song. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *NDSS*, Feb 2005.

[13] M. K. Reiter and S. G. Stubblebine. Resilient authentication using path independence. *IEEE Trans. Computers*, 47(12), 1998.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware*, Nov 2001.

[15] I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. In *IEEE/ACM Trans. on Networking*, 2002.

[16] L. Subramanian, V. Roth, I. Stoica, and S. Shenker. Listen and whisper: Security mechanisms for BGP. In *NSDI*, Mar 2004.

[17] P. Yalagandula, S. Nath, H. Yu, P. B. Gibbons, and S. Sesha. Beyond availability: Towards a deeper understanding of machine failure characteristics in large distributed systems. In *WORLDS*, Dec 2004.