End-System Multicast in Decentralized Distributed Systems

Graydon Hoare and David Mazières

IETF104

Wednesday, March 27, 2018

SCP update

`draft-mazieres-dinrg-scp` **seems to be stabilizing**

Also new simplified description of the protocol available

Big open issue: how to disseminate messages?

- Without multicast spec, no hope of interoperability
- Yet unclear existing solutions are secure enough

Recall SCP's security goals

- Allow secure payments, secure software/firmware/certificate transparency logs, secure IP prefix delegation, etc.

Today's talk: the secure multicast problem

- Only questions, no solutions

Example: Bitcoin overlay network

Typical node: 8 outgoing, 117 incoming TCP connections

- Initially get peers from DNS seeds (e.g., `seed.bitcoinstats.com`)
- Get more in received ADDR commands (up to 1000 peers+timestamps)

Each node divides peers into *tried* and *new* tables

Tried table contains known good peers with timestamps

- IPs hashed into 64 buckets of 64 entries each
- Any IPv4 /16 can only hash to one of four buckets
- Eviction from bucket: pick 4 random nodes, send oldest to new table

New table is newly learned or demoted nodes (might not work)

- 256 buckets of 64 nodes each
- Also group by IPv4 /16 that sent us ADDR command

Connection dropped? Connect to random node

- Pick between tried and new with probability depending on number of good outgoing connections, ratio of new/tried sizes

Eclipse attacks [Heilman15]

Use botnet to own IP addresses in many groups

Connect to victim node 117 times

Send ADDR command with all your own IP addresses

- Effectively a Sybil attack on overlay network

By controlling what victim miners learn, can:

- Engineer block races (so miners waste time on orphan blocks)
- Facilitate selfish mining attacks

Worse: PoW is unsafe in asynchronous model [Pass'16]

- Eclipse attack can impose arbitrary delays
- Split mining power among disconnected groups
- Allows double-spend by attackers with no hashing power!

What about asynchronous protocols?

Isn't SCP an *asynchronous* protocol?

- Yes, unlike PoW, it is safe in asynchronous setting, but…
- DoS can induce safety failures in layers above consensus

Example: higher-layer blockchain protocols

- Escrow, payment channels, etc., often involve timeouts (e.g., if no one contests your action for 24 hours, can take funds)
- So blocking disputes lets you steal money

Availability of higher-layer systems can affect safety

- E.g., what if you can't update log for new certificates, or to revoke vulnerable versions of software?

Overview of P2P multicast approaches

Medium-sized end-system multicast—e.g., Narada'02

- Form mesh where all nodes have complete member list
- Run routing protocol over mesh to form multicast trees
- Scales to 1000s(?) of nodes, no security story

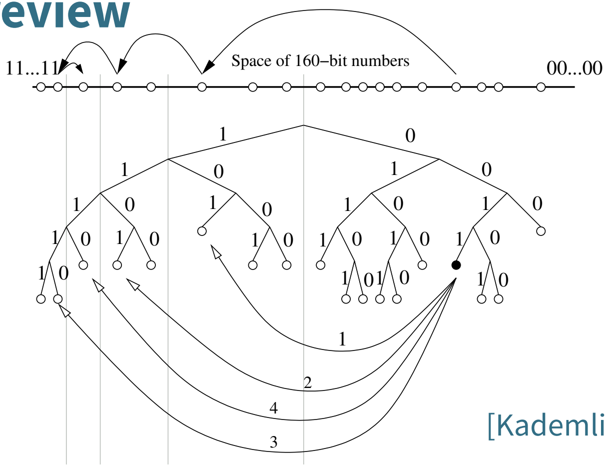
Structured tree- or DHT (distributed hash table)-based multicast

- Potentially scales to *millions* of nodes
- Efficient (don't receive many redundant messages)
- Brittle in the face of failure (particularly Byzantine)

Unstructured gossip-based protocols

- More robust to failure, but more wasteful (redundant) messages sent
- Scalability depends on partial vs. full view
  full view consumes bandwidth to disseminate membership list
- Vulernable to Byzantine failures, particularly with partial view

DHT review

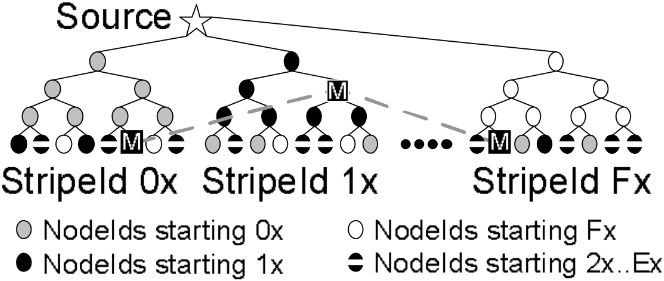


[Kademlia’02]

DHTs provide scalable key-value store [Chord,Pastry,Kademlia]

- Each node has a random (e.g., 160-bit) ID
- Store (key, value) pair on nodes with IDs closest to key
- With n nodes in system, each node knows $O(\log n)$ (or sometimes $O(1)$) other nodes, can find key after querying $O(\log n)$ servers

DHT-based multicast



[Splitstream'03]

Scribe'02: DHT multicast based on reverse path forwarding

- Multicast source is node with ID closest to multicast group
- Route a join request to source (up to $O(\log n)$ hops)
- At each hop node forwards traffic to children
- Interior nodes bear all the forwarding burden

Splitstream'03: Split data into multiple streams

- In forest, each node internal for one stream, leaf for others
- Maybe combine with forward error correction for robustness

DHT security [Survey'11]

Sybil attacks – attacker joins multiple times

- Admission control based on CAs, IP prefixes, network characteristics, join path, quotient cut in social graph, proof-of-work, incentives
- Or SCP shows open systems can circumvent Sybil assumptions

Eclipse attacks (routing table poinsoning)

- Separate optimized & secure/fallback routing table
- Leverage network characteristics
- In-degree/out-degree analysis
- Re-organization regions on joins (invalidating targeted joins)

Routing and storage attacks

- These matter when data is sharded, not fully replicated
- Use iterative routing, independent paths, fiddle with identifier allocation, CAs, sending requests to whole swarms, fiddle with routing topology
- Maybe attacks less relevant to today's applications?

Gossip-based multicast

Basic idea: send each new message to *k* random peers

- Will quickly propagate to whole system
- But, maintaining list of peers a scalability bottleneck

Scale with "Partial View" protocols, e.g., [HyParView'07]

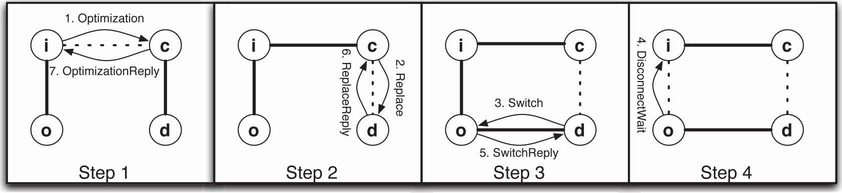
- HyParView maintains separate *active* & *passive* peer sets
- Active peers are symmetric and form the gossip network
- Passive peers can replace any failed active peers

HyParView maintains passive sets through shuffle protocol

- Forward some active & passive peers through random walk
- After TTL hops, recipient of shuffle message sends back some peers
- Add newly learned peers to passive set; on overflow drop ones you sent or drop random ones

Maintains active set by join or repair messages

Biased peer selection



[X-BOT]

Random peer selection can be bad for locality

- Forward message half way around the world to reach your neighbor

Can bias peer selection to favor network locality

X-BOT optimization strategy:

- Node *i* has *o* in active set, but thinks *c* from passive set closer
- Arrange to pair off *o* with one of *c*'s active nodes *d*
- Still need some fraction of peers random for correctness

Q: Could we bias peer selection for security?

Hybrid structured/gossip approaches

Plumtree'07 uses partial views segregated into two groups

- Eager push peers: immediately send all new messages to these
- Lazy push peers: lazily send just message IDs, possibly in batches, and have peers request any missing messages

Form a broadcast tree as follows:

- Initialize with random eager push peers, and no lazy ones
- When you first hear a message, add node to eager push peers
- When you hear a duplicate, move peer to lazy push peers

Repair broken tree when you get message ID but not message

- First wait for a timeout period
- Then request message from first node to send message ID
- Also upgrade that sender to eager push peer

Q: Can we securely disseminate message IDs to repair trees?

Secure gossip & dissemination

Tolerating malicious gossip [Minsky'03]

- Very BFT-like model tolerates at most t failures
- Assumes $k > t$ honest nodes start with same ground truth
- Idea: path verification (w/o signatures): require $t + 1$ disjoint paths
- Combine w. sampling schemes to increase efficiency
- Q: Can to irregular, open-membership systems?

Monitoring w. multiple membership rings [Fireflies'06]

- Classify nodes as correct, crashed, and malicious
- Assume malicious is at most fraction p of non-crashed
- Nodes accuse other nodes they believe have crashed
- Falsely accused node can mask some bad membership rings
- Q: Does admission control requirement rule out approach?

Secure overlay networks

Lightweight Intrusion-Tolerant Overlay Network [LITON'06]

- Use mobile ad-hoc like routing protocols for unicast in overlay
- All packets source routed
- Failure detection based on loss/RTT inferred from ACKs
- Quarantine end-to-end bad routes, links forwarding corrupt messages
- Uses flooding, probably vulnerable to underlay attacks
- Q: Can it be adapted to broadcast without amplification attacks?

Ack. compression [Nicolosi'04]

Idea: verify recipients got message without knowing recipients

- Phase 1: Peers join multicast tree, get secure join receipt
- Phase 2: If all peers get message, source can verify
- Otherwise, can get list of missing peers and their IP addresses, and verify only those peers missed messages

Leverages signatures based on Gap Diffie-Hellman groups

- Key property: if σ_1 is signature of m under key y_1 and σ_2 is signature of same m under y_2, then $\sigma_1\sigma_2$ is signature of m under y_1y_2

Works well on structured multicast trees

- Combine public keys on join, signed receipts on ack

Q: Can this be adapted to gossip networks?

Set reconciliation

Many decentralized systems involve multiple senders

- E.g., blockchain protocols with many nodes submitting transactions
- With many transactions, even set of transaction IDs could be big
- Hard to keep traffic asymptotically small when scaling gossip

Hash trees or error correcting codes [Minsky'02] could help

- Not efficient enough

Invertible Bloom filters (IBFs) make practical [Eppstein'11]

- Can subtract one IBF from another to get difference
- Use log many IBFs to estimate size of set difference
- Lets you reconcile sets with network traffic proportional only to set difference size

Quorum slices [SCP]

SCP forms quorums in open system based on *quorum slices*

- Invalidates Sybil attack assumption of no physical knowledge of peers

Each node v picks a set of *quorum slices* $\mathbf{Q}(v)$

- v only trusts quorums that are a superset of some $q \in \mathbf{Q}(v)$
- Include organizations you don't want to be forked from in every slices

Definition (Quorum)

A quorum $U \subseteq \mathbf{V}$ is a set of nodes that contains at least one slice of each of its members: $\forall v \in U, \exists q \in \mathbf{Q}(v)$ such that $q \subseteq U$

Q: Can you subscribe just to senders you transitively care about?

Q: Leverage trust expressed in quorum slices? e.g., weight:

Definition (Slice weight)

$\text{weight}(u, v) \in [0, 1]$ is the fraction of node u's quorum slices containing node v.

Conclusions

Decentralized protocols need message dissemination

- Efficient and scalable to many nodes
- Self-organizing with no central authority
- Byzantine fault-tolerant

Currently a big trade-off between efficiency and security

We have building blocks for efficiency and scalability

- DHT techniques, peer sampling, biased peer selection, lazy push, set reconciliation, …

We have building blocks for security

- Path verification, monitoring/accusation, ack compression, quorum slices, …

But no good solution we can even think of standardizing yet