**Gossip protocol**

| | | | |
|---|---|---|---|
| **笔记本:** | 技术研究 | | |
| **创建时间:** | 2018/9/25 15:57 | **更新时间:** | 2018/9/25 16:09 |
| **标签:** | P2P, 共识机制, 通信协议 | | |

# Gossip protocol

A **gossip protocol**[1] is a procedure or process of computer–computer communication that is based on the way social networks disseminate information or how epidemics spread. It is a communication protocol. Modern distributed systems often use gossip protocols to solve problems that might be difficult to solve in other ways, either because the underlying network has an inconvenient structure, is extremely large, or because gossip solutions are the most efficient ones available.

The term **epidemic protocol** is sometimes used as a synonym for a gossip protocol, because gossip spreads information in a manner similar to the spread of a virus in a biological community.

## Gossip communication

The concept of *gossip communication* can be illustrated by the analogy of office workers spreading rumors. Let's say each hour the office workers congregate around the water cooler. Each employee pairs off with another, chosen at random, and shares the latest gossip. At the start of the day, Alice starts a new rumor: she comments to Bob that she believes that Charlie dyes his mustache. At the next meeting, Bob tells Dave, while Alice repeats the idea to Eve. After each water cooler rendezvous, the number of individuals who have heard the rumor roughly doubles (though this doesn't account for gossiping twice to the same person; perhaps Alice tries to tell the story to Frank, only to find that Frank already heard it from Dave). Computer systems typically implement this type of protocol with a form of random "peer selection": with a given frequency, each machine picks another machine at random and shares any hot rumors.

The power of gossip lies in the robust spread of information. Even if Dave had trouble understanding Bob, he will probably run into someone else soon and can learn the news that way.

Expressing these ideas in more technical terms, a gossip protocol is one that satisfies the following conditions:

- The core of the protocol involves periodic, pairwise, inter-process interactions.
- The information exchanged during these interactions is of bounded size.
- When agents interact, the state of at least one agent changes to reflect the state of the other.
- Reliable communication is not assumed.
- The frequency of the interactions is low compared to typical message latencies so that the protocol costs are negligible.
- There is some form of randomness in the peer selection. Peers might be selected from the full set of nodes or from a smaller set of neighbors.
- Due to the replication there is an implicit redundancy of the delivered information.

## Gossip protocol types

It is useful to distinguish three prevailing styles of gossip protocol:[2]

- **Dissemination protocols** (or rumor-mongering protocols). These use gossip to spread information; they basically work by flooding agents in the network, but in a manner that produces bounded worst-case loads:
    1. *Event dissemination protocols* use gossip to carry out multicasts. They report events, but the gossip occurs periodically and events don’ t actually trigger the gossip. One concern here is the potentially high latency from when the event occurs until it is delivered.
    2. *Background data dissemination protocols* continuously gossip about information associated with the participating nodes. Typically, propagation latency isn’ t a concern, perhaps because the information in question changes slowly or there is no significant penalty for acting upon slightly stale data.

- **Anti-entropy protocols** for repairing replicated data, which operate by comparing replicas and reconciling differences.
- **Protocols that compute aggregates**. These compute a network-wide aggregate by sampling information at the nodes in the network and combining the values to arrive at a system-wide value – the largest value for some measurement nodes are making, smallest, etc. The key requirement is that the aggregate must be computable by fixed-size pairwise information exchanges; these typically terminate after a number of rounds of information exchange logarithmic in the system size, by which time an all-to-all information flow pattern will have been established. As a side effect of aggregation, it is possible to solve other kinds of problems using gossip; for example, there are gossip protocols that can arrange the nodes in a gossip overlay into a list sorted by node-id (or some other attribute) in logarithmic time using aggregation-style exchanges of information. Similarly, there are gossip algorithms that arrange nodes into a tree and compute aggregates such as "sum" or "count" by gossiping in a pattern biased to match the tree structure.

Many protocols that predate the earliest use of the term "gossip" fall within this rather inclusive definition. For example, Internet routing protocols often use gossip-like information exchanges. A gossip substrate can be used to implement a standard routed network: nodes "gossip" about traditional point-to-point messages, effectively pushing traffic through the gossip layer. Bandwidth permitting, this implies that a gossip system can potentially support any classic protocol or implement any classical distributed service. However, such a broadly inclusive interpretation is rarely intended. More typically gossip protocols are those that specifically run in a regular, periodic, relatively lazy, symmetric and decentralized manner; the high degree of symmetry among nodes is particularly characteristic. Thus, while one could run a 2-phase commit protocol over a gossip substrate, doing so would be at odds with the spirit, if not the wording, of the definition.

Frequently, the most useful gossip protocols turn out to be those with exponentially rapid convergence towards a state that "emerges" with probability 1.0. A classic distributed computing problem, for example, involves building a tree whose inner nodes are the nodes in a network and whose edges represent links between computers (for routing, as a dissemination overlay, etc.). Not all tree-building protocols are gossip protocols (for example, spanning tree constructions in which a leader initiates a flood), but gossip offers a decentralized solution that is useful in many situations.

The term *convergently consistent* is sometimes used to describe protocols that achieve exponentially rapid spread of information. For this purpose, a protocol must propagate any new information to all nodes that will be affected by the information within time logarithmic in the size of the system (the "mixing time" must be logarithmic in system size).

## Examples

Suppose that we want to find the object that most closely matches some search pattern, within a network of unknown size, but where the computers are linked to one another and where each machine is running a small *agent* program that implements a gossip protocol.

- To start the search, a user would ask the local agent to begin to gossip about the search string. (We're assuming that agents either start with a known list of peers, or retrieve this information from some kind of a shared store.)
- Periodically, at some rate (let's say ten times per second, for simplicity), each agent picks some other agent at random, and gossips with it. Search strings known to A will now also be known to B, and vice versa. In the next "round" of gossip A and B will pick additional random peers, maybe C and D. This round-by-round doubling phenomenon makes the protocol very robust, even if some messages get lost, or some of the selected peers are the same or already know about the search string.
- On receipt of a search string for the first time, each agent checks its local machine for matching documents.
- Agents also gossip about the best match, to date. Thus, if A gossips with B, after the interaction, A will know of the best matches known to B, and vice versa. Best matches will "spread" through the network.

If the messages might get large (for example, if many searches are active all at the same time), a size limit should be introduced. Also, searches should "age out" of the network.

It follows that within logarithmic time in the size of the network (the number of agents), any new search string will have reached all agents. Within an additional delay of the same approximate length, every agent will learn where the best match can be found. In particular, the agent that started the search will have found the best match.

For example, in a network with 25,000 machines, we can find the best match after about 30 rounds of gossip: 15 to spread the search string and 15 more to discover the best match. A gossip exchange could occur as often as once every tenth of a second without imposing undue load, hence this form of network search could search a big data center in about 3 seconds.

In this scenario, searches might automatically age out of the network after, say, 10 seconds. By then, the initiator knows the answer and there is no point in further gossip about that search.

Gossip protocols have also been used for achieving and maintaining distributed database consistency or with other types of data in consistent states, counting the number of nodes in a network of unknown size, spreading news robustly, organizing nodes according to some structuring policy, building so-called overlay networks, computing aggregates, sorting the nodes in a network, electing leaders, etc.

## Epidemic algorithms

Gossip protocols can be used to propagate information in a manner rather similar to the way that a viral infection spreads in a biological population. Indeed, the mathematics of epidemics are often used to model the mathematics of gossip communication. The term *epidemic algorithm* is sometimes employed when describing a software system in which this kind of gossip-based information propagation is employed.

## Biased gossip

Above, a purely random peer-selection scheme for gossip was described: when agent A decides to run a gossip round, it picks some peer B uniformly and at random within the network as a whole (or launches a message on a random walk that will terminate at a random agent). More commonly, gossip algorithms are designed so that agents interact mostly with nearby agents, and only sometimes with agents that are far away (in terms of network delay). These *biased* gossip protocols need to ensure a sufficient degree of connectivity to avoid the risk of complete disconnection of one side of a network from the other, but if care is taken, can be faster and more efficient than protocols that are purely random. Moreover, as a purely practical question, it is much easier to maintain lists of peers in ways that might be somewhat biased.

## Code examples

There are three known libraries that implement a gossip algorithm to discover nodes in a peer-to-peer network:

- Apache Gossip communicates using UDP written in Java, has support for arbitrary data and CRDT types.
- gossip-python utilizes the TCP stack and it is possible to share data via the constructed network as well.
- Smudge is written in Go and uses UDP to exchange status information; it also allows broadcasts of arbitrary data across the constructed network.

## See also

- Gossip protocols are just one class among many classes of networking protocols. See also virtual synchrony, distributed state machines, Paxos algorithm, database transactions. Each class contains tens or even hundreds of protocols, differing in their details and performance properties but similar at the level of the guarantees offered to users.
- Some gossip protocols replace the random peer selection mechanism with a more deterministic scheme. For example, in the NeighbourCast algorithm, instead of talking to random nodes, information is spread by talking only to neighbouring nodes. There are a number of algorithms that use similar ideas. A key requirement when designing such protocols is that the neighbor set trace out an expander graph.
- Routing
- Tribler, BitTorrent peer to peer client using gossip protocol.

## References

1. **Jump up ^** Demers, Alan; Greene, Dan; Hauser, Carl; Irish, Wes; Larson, John; Shenker, Scott; Sturgis, Howard; Swinehart, Dan; Terry, Doug (1987-01-01). "Epidemic Algorithms for Replicated Database Maintenance". *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing*. PODC '87. New York, NY, USA: ACM: 1–12. doi:10.1145/41840.41841. ISBN 089791239X.
2. **Jump up ^** Jelasity, Márk (2011-01-01). Serugendo, Giovanna Di Marzo; Gleizes, Marie-Pierre; Karageorgos, Anthony, eds. *Gossip*. Natural Computing Series. Springer Berlin Heidelberg. pp. 139–162. doi:10.1007/978-3-642-17348-6_7. ISBN 9783642173479.

Here are some additional references to recent work from the gossip community. The paper by Demers is considered by most researchers to be the first to have really recognized the power of these protocols and to propose a formal treatment of gossip.

- Correctness of a Gossip-based Membership Protocol. André Allavena, Alan Demers and John Hopcroft. Proc. 24th ACM Symposium on Principles of Distributed Computing (PODC 2005).
- Bimodal Multicast. Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu and Yaron Minsky. ACM Transactions on Computer Systems, Vol. 17, No. 2, pp 41–88, May, 1999.
- Lightweight probabilistic broadcast. Patrick Eugster, Rachid Guerraoui, S. B. Handurukande, Petr Kouznetsov, Anne-Marie Kermarrec. ACM Transactions on Computer Systems (TOCS) 21:4, Nov 2003.
- Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead. Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, Robbert van Renesse. Proc. 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)
- Systematic Design of P2P Technologies for Distributed Systems. Indranil Gupta, Global Data Management, eds: R. Baldoni, G. Cortese, F. Davide and A. Melpignano, 2006.
- HyParView: a Membership Protocol for Reliable Gossip-based Broadcast. João Leitão, José Pereira, Luís Rodrigues. Proc. 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)
- Efficient and Adaptive Epidemic-Style Protocols for Reliable and Scalable Multicast. Indranil Gupta, Ayalvadi J. Ganesh, Anne-Marie Kermarrec. IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 7, pp. 593–605, July, 2006.
- T-Man: Gossip-based fast overlay topology construction. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Computer Networks, 53(13):2321–2339, 2009.
- Epidemic Broadcast Trees. João Leitão, José Pereira, Luís Rodrigues. Proc. 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'07).
- Gossip-based aggregation in large dynamic networks. Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. ACM Transactions on Computer Systems, 23(3):219–252, August 2005.
- Ordered slicing of very large overlay networks. Márk Jelasity and Anne-Marie Kermarrec. IEEE P2P, 2006.
- Proximity-aware superpeer overlay topologies. Gian Paolo Jesi, Alberto Montresor, and Ozalp Babaoglu. IEEE Transactions on Network and Service Management, 4(2):74–83, September 2007.
- X-BOT: A Protocol for Resilient Optimization of Unstructured Overlays. João Leitão, João Marques, José Pereira, Luís Rodrigues. Proc. 28th IEEE International Symposium on Reliable Distributed Systems (SRDS'09).
- Spatial gossip and resource location protocols. David Kempe, Jon Kleinberg, Alan Demers. Journal of the ACM (JACM) 51: 6 (Nov 2004).
- Gossip-Based Computation of Aggregate Information. David Kempe, Alin Dobra, Johannes Gehrke. Proc. 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS). 2003.
- Active and Passive Techniques for Group Size Estimation in Large-Scale and Dynamic Distributed Systems. Dionysios Kostoulas, Dimitrios Psaltoulis, Indranil Gupta, Ken Birman, Al Demers. Elsevier Journal of Systems and Software, 2007.
- Build One, Get One Free: Leveraging the Coexistence of Multiple P2P Overlay Networks. Balasubramaneyam Maniymaran, Marin Bertier and Anne-Marie Kermarrec. Proc. ICDCS, June 2007.
- Peer counting and sampling in overlay networks: random walk methods. Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, Ayalvadi Ganesh. Proc. 25th ACM PODC. Denver, 2006.
- Chord on Demand. Alberto Montresor, Márk Jelasity, and Ozalp Babaoglu. Proc. 5th Conference on Peer-to-Peer Computing (P2P), Konstanz, Germany, August 2005.
- Introduction to Expander Graphs. Michael Nielsen. https://pdfs.semanticscholar.org/4c8a/e0bc0dca940264b7ed21fa58f826937f7b12.pdf. Technical report, June 2005.
- Building low-diameter P2P networks. G. Pandurangan, P. Raghavan, Eli Upfal. In Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS), 2001.

- Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. Robbert van Renesse, Kenneth Birman and Werner Vogels. ACM Transactions on Computer Systems (TOCS) 21:2, May 2003.
- Exploiting Semantic Proximity in Peer-to-peer Content Searching. S. Voulgaris, A.-M. Kermarrec, L. Massoulie, M. van Steen. Proc. 10th Int'l Workshop on Future Trends in Distributed Computing Systems (FTDCS 2004), Suzhou, China, May 2004.

- Reputation aggregation in peer-to-peer network using differential gossip algorithm. R. Gupta, Y. N. Singh. CoRR, vol. abs/1210.4301, 2012.

Although this textbook is old, many gossip researchers cite it as an authoritative source for information about the mathematical modelling of gossip and epidemic protocols:

- The Mathematical Theory of Epidemics. N.J.T. Bailey, 1957. Griffen Press.