In “Gossiping for Communication-Efficient Broadcast”, the authors apply gossiping - propagating a message by sending it to a few random parties who in turn do the same, until the message is delivered - as the communication model for efficient communication protocols under dishonest majority, for Single-Sender Broadcast (BC) as well as for Parallel Broadcast (PBC).

Bulletin PKI (public key infrastructure) - no trusted setup, all parties register their public keys to a public bulletin board before the start of the protocol and no assumption is made on how the parties generate their keys

Trusted PKI (public key infrastructure) - trusted setup is required, a trusted party generates all keys honestly and distributed them to the parties prior to the protocol execution

Single-Sender Broadcast - BulletinBC

The protocol replaces Dolev-Strong’s $\mathrm{SEND}-\mathrm{ALL}(x)$ instruction with a $\mathrm{SEND}-\mathrm{RANDOM}(x,m)$ instruction. This instruction is implemented by sending the message x to party $i, i \in [n]$, with a probability m/n for some fixed $m = \Theta(\kappa)$, where κ is the security parameter and represents the size of a digital signature. In order for the message to propagate the protocol has to run for an additional $O(\log n)$.

A procedure called AddRandomEdges simulates the propagation of messages from honest nodes to the rest of the network. Over a graph G whose n vertices V are partitioned into three arbitrary disjointed sets and has no edges initially, the procedure add the edge (v,u) to the graph with probability m/n for every pair of nodes $v \in S \subseteq V-(S_2 \cup S_3)$, $u \in V$. S represents the set of parties that send a message q at a specific round r and S_2 represents the set of parties that have not received q in previous rounds. The edge from $v \in S$ to $u \in V$ represents that party v sends message q to party v in round r. In order to know how many parties in S_2 received q for the first time in round r, we need to study the degree of nodes in S_2. The number of nodes in S_2 that acquire an edge in G is at least twice the number of nodes in S, which intuitively shows that messages propagate very quickly.

The protocol BulletinBC from the view of an honest party p takes as an input an initial bit b in the case of the designated sender, no input else, and returns a final bit b'. In contrast to the Dolev-Strong protocol, the number of rounds is $t+O(\log n)$ instead of t ($t<n$ is the number of corrupted parties), instead of sending to all parties this protocol sends to each party randomly with probability m/n, and for all rounds $r \geq t+1$ the protocol doesn’t require $r+1$ signatures to add to the extracted set but just $t+1$. The article provides proofs for t-consistency (if the sender is honest and holds an input v, all honest parties output v) and t-validity (is the sends holds an input v, all honest parties output the same value v') of BulletinBC.

The $\mathcal{M}-Convergence$ Problem

Under an adaptive adversary that can inject messages, there is a fixed message set $\mathcal{M}$, and all initial honest parties $p \in \mathcal{H}$ begin with a set of messages $M_p \subseteq \mathcal{M}$ and a constraint set of messages $\mathcal{C}_p \subseteq \mathcal{M}$. In the end, all remaining honest parties q should output a set S_q that is a superset of $\cup_p M_p - \cup_p \mathcal{C}_p$. Using the $\mathcal{M}$-ConvergenceRandom protocol outlines in the article, every honest party runs $O(\log n)$ rounds and in every round sends every message in its local set to a randomly-chosen subset of parties, not to all of them, which results in communication complexity of $\tilde{O}\left(n|\mathcal{M}|+n^2\right)$ (in a naive solution where all honest parties send their sets M_p and $\mathcal{C}_p$ to all other n parties, the complexity is $O\left(n^2|\mathcal{M}|\right)$,since M_p and $\mathcal{C}_p$ are subsets of $\mathcal{M}$).

In round 1 of the protocol, honest party p, for every message $x \in M_p - \mathcal{C}_p$, picks $i \in [n]$ as a recipient with probability m/n. Then p constructs lists $\mathcal{L}_j, j=1,\ldots,n$, containing messages $x \in M_p - \mathcal{C}_p$ that were assigned to the recipient j (each list is padded to at most $2m\lceil|\mathcal{M}|/n\rceil$ to prevent overflow). The lists are encrypted so that an adaptive adversary doesn’t gain an advantage, and $\mathcal{L}_j$ is sent out to party j for $j=1,\ldots n$. In rounds $i=2,\ldots,\lceil\log\epsilon \cdot n\rceil$ every honest party p adds to its constraint set $\mathcal{C}_p$ all messages sent in the previous round $i-1$, collects all lists $\mathcal{L}_p$ from round $i-1$ to a new set M_p and performs the same tasks as in round 1 - random assigning of elements in $M_p - \mathcal{C}_p$, compiling into lists and sending them. The article provides proofs that with overwhelming probability the gossiping protocol delivers every element contained in the sets of the initially-honest parties in a communication complexity of $\tilde{O}\left(n|\mathcal{M}|+n^2\right)$. The protocol also achieves adaptive security (against adaptive adversaries who can corrupt up to t parties, each at every point during the execution of the protocol, learning its internal state which consists of any long term secret keys and short term values that haven’t been deleted), since the adversary can’t tell which are the true recipients of the message because each list $\mathcal{L}_j$ is padded to the same size and encrypted, therefore the adversary can’t corrupt the specific set of recipients that receive the message from the honest sender. The different messages that are sent by a sender in a particular round of the protocol “cover” for one another and hide to whom a particular message is being sent, since all the recipients of a single sender obtain the same amount of encrypted information, and the exact path of the message is untraceable.

The article provides descriptions to procedure AddRandomEdgesAdaptive and ideal functionality $\mathcal{F}_{\text{prop}}$ that enables a party i to send a set of messages M to, on average, m/n randomly selected parties without leaking which those parties are to the adversary which are used in the $\mathcal{M}$-ConvergenceRandom Protocol that solves the $\mathcal{M}$-Convergence problem. An extension called the $\mathcal{M}$-DistinctConverge Protocol is also described, where messages are considered equivalent based on a parameter k.The $\mathcal{M}-Convergence$ Problem

Parallel Broadcast - BulletinPBC

In PBC, all n parties simultaneously act as a sender and wish to consistently distribute their message. The proposed protocol in the article views this situation as an instance of the $\mathcal{M}$-Converge problem, instead of naively running multiple instances of the BC protocol. In the Dolev-Strong protocol, for all rounds $r < n$, signatures from r parties are observed before a bit is accepted by an honest party, which then adds its signature and sends the message using $\mathrm{SEND} - \mathrm{ALL}$. Combining n of these protocols for PBC results in communication complexity of $O\left(n^4 \cdot \kappa\right)$. The main communication overhead in this protocol comes from all parties sending lists of $O\left(n\right)$ signatures from the same designated sender, when it's possible all parties hold the same list of r signatures in some round r that made them accept a message for that round - there will be a large amount of redundancy when all honest parties send all of these signatures to all parties. In the article this is mitigated by using the $\mathcal{M}$-Convergence problem, where input sets consist of the signatures in the parties' lists and the constraint sets ensure that signatures which were already sent will not be repeated. $\mathcal{M}$-ConvergenceRandom s used to propagate messages for all slots simultaneously regardless of what slot they belong to.

In the protocol, each party maintains internally its state for every signature, represented by triplets (b, s, j), where $b \in \{0, 1\}$ and $s, j \in [n]$, and each triplet defines a specific signature. The propagation of a specific signature is independent of whether the bit was added to the local set in the given round, but rather, if the signature was previously unobserved by a party i, it will propagate the signature twice. If the new signature was observed during the sub-round of some $\mathcal{M}$-ConvergenceRandom, the parties will continue to propagate it only during the next sub-round and will initiate the next round's $\mathcal{M}$-ConvergenceRandom with the signature in the input set but not in the constraint set. The article goes on to provide proof of consistency, validity and the communication complexity of $O\left(n^3 \cdot \kappa\right)$ resulting from the protocol.

Parallel Broadcast - TrustedPBC

This protocol depends on the protocol by Chan et al. that solved broadcast in the trusted PKI model with a communication complexity of $O\left(n^2 \cdot \kappa^2\right)$. That protocol is essentially a Dolev-Strong protocol run among a random committee of κ parties. In round $r < \kappa$, whenever an honest committee party p has observed r signatures on a bit b for the first time, p adds its signature and sends a message of $r + 1$ signatures to all parties using $\mathrm{SEND} - \mathrm{ALL}$, and when an honest party p observes r signatures on a bit b for the first time p forwards just the r signatures to all parties using $\mathrm{SEND} - \mathrm{ALL}$ (and doesn't add a signature). If used naively in the parallel case, the protocol's communication complexity is $O\left(n^3 \cdot \kappa^2\right)$. The article presents the TrustedPBC protocol which abstracts each step of the protocol as an instance of the $\mathcal{M}$-Converge problem which can reduce the complexity to $O\left(n^2 \cdot \kappa^4\right)$

This protocol uses trusted setup in order to reduce the communication complexity of PBC. It uses the $\mathcal{M}$-DistinctConverge Protocol mentioned in The $\mathcal{M} - Convergence$ Problem section. In round $r \leq R$, an honest party p first collects the set $\mathcal{V}$of valid r-batches on messages not in the local set. All parties run $\mathcal{M}_r$-DistinctConverge with the initial constraint set $\mathcal{C}_p$ empty, and the protocol assures that all parties eventually see the other parties' input. Since there is some overlap between input messages it uses less communication. An honest party p then checks which valid r-batches in their local set has not been voted on, and checks if its a member of the committee using the functionality $\mathcal{F}_{\text{mine}}$ and if so the party will add its own signature.