Agreement in epidemic data aggregation

Conference or Workshop Item

Accepted Version


It is advisable to refer to the publisher’s version if you intend to cite from the work. See Guidance on citing.

Published version at: https://doi.org/10.1109/ICPADS.2017.00099

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

CentAUR
Central Archive at the University of Reading
Reading’s research outputs online
Agreement in Epidemic Data Aggregation

Mosab M. Ayiad  Giuseppe Di Fatta
Department of Computer Science, University of Reading
Whiteknights, Reading, Berkshire, RG6 6AY, UK

Abstract
Computing and spreading global information in large-scale distributed systems pose significant challenges when scalability, parallelism, resilience and consistency are demanded. Epidemic protocols are a robust and scalable computing and communication paradigm that can be effectively used for information dissemination and data aggregation in a fully decentralised context where each network node requires the local computation of a global synopsis function. Theoretical analysis of epidemic protocols for synchronous and static network models provide guarantees on the convergence to a global target and on the consistency among the network nodes. However, practical applications in real-world networks may require the explicit detection of both local convergence and global agreement (consensus).

This work introduces the Epidemic Consensus Protocol (ECP) for the determination of consensus on the convergence of a decentralised data aggregation task. ECP adopts a heuristic method to locally detect convergence of the aggregation task and stochastic phase transitions to detect global agreement and reach consensus.

The performance of ECP has been investigated by means of simulations and compared to a tree-based Three-Phase Commit protocol (3PC). Although, as expected, ECP exhibits total communication costs greater than the optimal tree-based protocol, it is shown to have better performance and scalability properties; ECP can achieve faster convergence to consensus for large system sizes and inherits the intrinsic decentralisation, fault-tolerance and robustness properties of epidemic protocols.

Keywords: Epidemic protocols, Gossip-based protocols, Distributed consensus, Decentralised algorithms, Large-scale distributed computing.

1 Introduction
The Internet as a ubiquitous infrastructure and the widespread use of mobile and wireless devices have laid the foundation for the emergence of innovative large-scale network applications and computing paradigms. Complex network applications may involve the real-time aggregation of large-scale distributed data, thus raising challenges and demands for the application scalability, resilience and consistency. In particular, the computation of global aggregation functions over a large set of distributed values is a non-trivial problem due to the decentralised nature and continuous variability of the environment [1].

Distributed data aggregation is essential for a broad range of network services such as calculating system size, resource capacity and average uptime [2]. In a distributed large-scale scenario, a centralised aggregation mechanism cannot be adopted as it would limit the system scalability and would be subject to a single points of failure and to bottlenecks [3]. Thereby an effective solution has to utilise decentralised and fault-tolerance computational paradigms. Moreover, practical applications like failure detection [4], distributed data mining [5], global attribute computation in Wireless Sensor Networks (WSN) [3], coordination among vehicles [6] and reaching unanimity in transactions order in the P2P electronic cash system (Bitcoin) [7], may require the explicit detection of (1) the local convergence to a target value and (2) the global agreement among participating nodes. A typical example is the local approximation of a global data aggregation function, where a node may need to achieve three different levels of information, the true target value with some approximation, awareness of local convergence to the target and certainty of a global agreement on the target.

Achieving global agreement is a fundamental problem in large-scale distributed systems (Consensus Problem) and is critical to many distributed applications. The problem studied in this work is the extension of the consensus problem to distributed data aggregation. Nodes in the system have to (1) compute a local approximation of a global data aggregation function and (2) achieve consensus on the approximation target value. The challenge is to achieve global agreement among all participants from locally computed data with a full decentralisation and fault-tolerance.

Epidemic (a.k.a. Gossip-based) protocols are known for their applicability, scalability and fault-tolerance properties. Epidemic protocols enable fully decentralised solutions for various distributed problems, such as

*email: {m.m.ayiad@pgr., g.difatta@reading.ac.uk}
information dissemination and data aggregation [1]. In epidemic data aggregation, computation and communication are uniformly distributed among nodes thanks to a randomised communication strategy. Moreover, the random pairwise communication approach provides stochastic robustness and guarantees that the nodes in the system ultimately converge to a common state in logarithmic time w.r.t. the system size [8].

This paper proposes a novel decentralised solution that integrates distributed data aggregation and distributed consensus using the epidemic paradigm. The innovative contribution is that the proposed solution is not only achieving local convergence on the data aggregation target but also explicit global agreement among all nodes using a decentralised decision mechanism. An Epidemic Consensus Protocol (ECP) is introduced in the proposed solution. ECP achieves consensus on the data aggregation using transition across phases, heuristic methods and epidemic computation. The validation and performance of ECP are examined by means of simulations.

This paper also presents a comparative analysis on the performance and the communication overhead of ECP and a tree-based 3PC protocol. The 3PC protocol provides a baseline performance with optimal communication overhead, though it is not fault-tolerant and is generally affected by single points of failure and potentially large communication latency. ECP achieves consensus faster than the tree-based 3PC protocol and presents better scalability and decentralisation properties in large-scale networks.

The following section briefly reviews distributed data aggregation and consensus in structured networks. Section 3 describes the concept of the epidemic consensus solution. The protocol ECP is described in section 4. The comparative analysis and experimental results are detailed in section 5. In section 6 some related works are listed and conclusions are drawn in section 7.

2 Distributed Consensus and Distributed Data Aggregation

Two of the fundamental support services in large-scale distributed applications are distributed consensus and distributed data aggregation. The former attains the agreement among participants and the latter provides participants with a global information.

The consensus is a collective decision-making process to reach agreement among participants on a common target [9]. The consensus process is usually formulated by the way in which is used to reach the agreement. In a classic formulation, the distributed process of consensus is an iterative procedure that each participant follows to achieve a global agreement on an output which is in a common interest of all participants [10]. In a distributed system of n nodes, each node i proposes an initial value xi; and eventually, all nodes decide on some target value ˆx that is among the set of proposed values [11].

Another formulation of consensus utilises the distributed averaging approach, e.g. the Coordination Problem in WSN and multi-agent systems [12]. In this case, the consensus target is the average of initial values \( \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \). Each node i performs a pair data exchange with a neighbour node j and computes \( \hat{x}_i = \frac{1}{2}(x_i + x_j) \). After a number of iterations, \( \hat{x}_i \) at each node i will converge to \( \bar{x} \). The consensus is achieved when the convergence to \( \bar{x} \) holds in all nodes [13]. However, consensus based on distributed averaging is shown to be feasible under certain conditions, for instance, reliable and static networks. In large asynchronous networks, the local approximation \( \hat{x}_i \) may not converge to \( \bar{x} \) at all nodes at the same time, thus failing to provide certainty of the global agreement.

Distributed data aggregation protocols generally provide global information about a system by computing a synopsis function \( f \in \{ \text{sum, average, random, sample, quantiles, etc.} \} \) over distributed data values [8]. Let us consider a system with n nodes where each node i has a set of neighbours and holds a numeric value \( x_i \) which describes a property in node i or in its environment. The aggregation protocol at node i exchanges \( x_i \) with neighbours to compute some global function \( f \). At the reception of a message from node j, the node i performs \( \hat{x}_i = f(x_i, x_j) \) and updates its local value \( x_i = \hat{x}_i \). After a sufficient number of data exchanges, all nodes in a system will converge to the same output \( \hat{x} \) which is the aggregation target.

The optimum performance of distributed data aggregation is obtained when the computation of a function \( f \) is distributed over a structured topology such as trees [14]. There are two mechanisms to perform data aggregation over trees, BROADCAST-CONVERGECAST and CONVERGECAST only. In BROADCAST-CONVERGECAST, a root node disseminates a request for the aggregation over a tree and collects results. In a CONVERGECAST tree, data computation starts from the deepest nodes and results are sent towards the root node, e.g. the Collection Tree Protocol (CTP) [15] provides a faster gathering of results at the root node. Each non-root node i in CTP computes \( \hat{x}_i = f(x_i, \hat{x}_1, \ldots, \hat{x}_{m_i}) \) where \( m_i \) is the number of child nodes at the node i and \( \hat{x}_1, \ldots, \hat{x}_{m_i} \) are the computation results at child nodes. Thereafter, node i sends \( \hat{x}_i \) to its parent node. After a sufficient number of steps, the root node receives \( \hat{x}_1, \ldots, \hat{x}_{m_i} \) from child nodes and calculates the final target value \( \hat{x} \).

In CTP and similar schemes, the target \( \hat{x} \) is eventually known to the root node only. In the consensus problem, \( \hat{x} \) is in the interest of all participants [11]. For a consensus to be achieved on \( \hat{x} \), a root node needs to disseminate \( \hat{x} \) back to the tree [15]. The consensus is then achieved because the \( \hat{x} \) become known to all nodes. However, in unreliable networks, the root is no more certain about the reception of \( \hat{x} \) at non-root nodes. Thus,
a root node needs to collect acknowledgements from all non-root nodes to attain the certainty on the reception of $\hat{x}$.

The Commitment Protocols such as 3PC protocol can achieve consensus in the presence of asynchrony and failure [16, 17]. In a tree that using 3PC protocol to achieve consensus on the output of $f$, the protocol begins in (Broadcast phase) in which a coordinator broadcasts a “compute” message to a tree and collects acknowledgements. Each non-root node $i$ computes $\hat{x}_i = f(x_i, \hat{x}_{i1}, \ldots, \hat{x}_{in})$ and sends $\hat{x}_i$ in the acknowledgement to its parent node and so on towards the coordinator. Subsequently, in the (Agreement phase) and in the absence of faulty nodes, the coordinator computes the final $\hat{x}$ and disseminates it back to the tree. Since $\hat{x}$ is disseminated to all nodes, the same global knowledge is then known to each node and so each node can acknowledge its readiness to commit. After the collection of acceptance acknowledgements from all non-root nodes in the Agreement phase, the coordinator attains the certainty on the reception of $\hat{x}$ and so it sends commit message to the tree and starting the (Commit phase). At the reception of commit message, non-root nodes can commit as the certainty of the global agreement is attained at each node. The 3PC protocol encloses certainty steps to achieve consensus in unreliable networks but it still susceptible to the single points of failure problem, for instance, the failure of coordinator in the Commit phase.

Unlike using the convergence in distributed averaging to achieve the consensus which is merely possible in reliable and static networks, and in the contrast to fast computation of $\hat{x}$ in CTP over a presumed tree, the tree-based 3PC protocol can achieve the consensus in unreliable networks with optimal overhead. On another hand, static trees are not ideal in the real environment due to node failure problem and network dynamics [2]. Dynamic trees are introduced to cope with some problems in the real environment. However, dynamic trees require additional overhead to establish a tree for every change in the underlying network [15]. Generally, the robustness of tree-based schemes typically relies on the consistency of the network and dynamic trees require additional effort for tree construction and maintenance.

This paper presents a decentralised solution that overcomes limitations in tree-based schemes. The new solution achieves consensus without establishing or maintaining any particular network structure. Moreover, the solution uses local information to detect global agreement on the outcome of data aggregation. Consider a distributed application of $n$ nodes, each node holds a local value $x_i$ and the application wants to compute a global target value $\hat{x}$ and also wants to achieve consensus on the target. Thereby, each node $i$ locally computes an approximation $\hat{x}_i$ by performing a global aggregation function $f$ and uses $\hat{x}_i$ to detect convergence to $\hat{x}$. Thereafter, each node passes across phases based on stochastic measurements on the local information to reach an explicit global agreement. More description of the epidemic consensus solution is given in the following section.

3 Consensus in Epidemic Data Aggregation

In large-scale distributed systems, the detection of a global agreement requires addressing potential issues such as randomisation, asynchrony and variability of approximation during the convergence of epidemic aggregation. Each node uses local information to decide upon convergence and eventually agreement. The reliance on the local information in an epidemic algorithm can lead to an incorrect approximation of the target value or to early false detection of convergence. Additionally, some nodes may reach convergence before other nodes; and naturally nodes have no local knowledge on the convergence state of other nodes. From that, attaining certainty of a global agreement at each node requires global awareness of the convergence state among nodes. The global awareness should be obtained from locally available information and using epidemic computation.

Similar to 3PC that achieves consensus in three consecutive phases, the proposed solution achieves consensus by accomplishing a number of successive phases. In particular, the protocol ECP comprises four consecutive phases. The first phase in ECP is dedicated to data aggregation. In the first phase, each node $i$ computes a global aggregation function $f$ and periodically monitors the local approximation output $\hat{x}_i$ to detect the convergence. The protocol in node $i$ makes the transition to the successive phase when the approximation $\hat{x}_i$ converges to a target value that holds for a certain period of time.

After the completion of the first phase, each node needs to obtain awareness on the convergence state of other nodes to ensure the certainty of the convergence. Therefore, a second phase is established intending to estimate the number of nodes achieved convergence in the first phase using the global aggregation function count. The target value of the count function in this phase is the initial system size $n$. In this phase, The convergence to $n$ at each node $i$ indicates local awareness of other nodes. Due to asynchrony, nodes will not converge to $n$ exactly at the same time. Therefore, to reach the explicit global agreement, a third phase is needed to count the number of nodes which have converged to $n$ in phase two. At the completion of phase three, each node has a global knowledge about every node else in terms of local convergence and awareness of convergence in other nodes, and thus the certainty of a global agreement is attained. The prior mechanism provides the required consistency for the global implementation of an action or a decision. The next Section details the inner stochastic transitions of computation across phases of ECP.
4 The Epidemic Consensus Protocol (ECP)

Algorithm 1: Epidemic Consensus Protocol (ECP)

1 Require: \( \varepsilon_1 \) and \( \varepsilon_2 \): error thresholds; \( \Upsilon \): consecutive cycles threshold; \( l \): length of the history queue \( \mathcal{H} \); getSize() : size estimation service in SSEP; getRandomPeer() : peer sampling service in NCP;

2 Initialisation: at each node \( i \) set: \( \text{vd} = x_i \), \( \text{wd} = 1 \), \( \text{vc} = 0 \), \( \text{va} = 0 \), \( w = 0 \);

\( \text{phase} = \text{AGGREGATION}; \mathcal{H} \leftarrow \{\} \) where \( |\mathcal{H}| = l \), leader = \( i \);

3 At each cycle at node \( i \):

\( j = \text{getRandomPeer()} \) // contact random peer

\( \text{vd} = \frac{\text{vd}}{2}, \text{wd} = \frac{\text{wd}}{2}, \text{vc} = \frac{\text{vc}}{2}, \text{va} = \frac{\text{va}}{2}, w = \frac{w}{2} \)

4 switch phase do // make assessment and phase transitions

5 case AGGREGATION do

6 if \( C_v(\mathcal{H}) \leq \varepsilon_1 \) for at least \( \Upsilon \) cycles then \( \text{// detect convergence in Aggregation phase} \)

7 \( \text{vc} = \text{vc} + 1 \)

8 if leader has not changed for at least \( \Upsilon \) cycles then \( \text{// detect leader node} \)

9 if leader == \( i \) then \( w = 1 \)

10 case CONVERGENCE do // obtain awareness of convergence

11 if \( \frac{\text{getSize}(\mathcal{H})}{\text{getSize}()} \leq \varepsilon_2 \) for at least \( \Upsilon \) cycles then \( \text{// detect convergence to system size} \)

12 \( \text{va} = \text{va} + 1 \)

13 if leader == \( i \) then \( w = 1 \)

14 case AGREEMENT do // obtain explicit global agreement

15 if \( \frac{\text{getSize}(\mathcal{H})}{\text{getSize}()} \leq \varepsilon_2 \) for at least \( \Upsilon \) cycles then \( \text{// detect agreement} \)

16 \( \text{phase} = \text{COMMIT} \) // take some application-specific decision or action

21 At event "received \( m \) message from \( j \)" at node \( i \):

22 if \( \text{m.reply} \) then // reply to incoming message

23 \( \text{vd} = \frac{\text{vd}}{2}, \text{wd} = \frac{\text{wd}}{2}, \text{vc} = \frac{\text{vc}}{2}, \text{va} = \frac{\text{va}}{2}, w = \frac{w}{2} \)

24 send(\( j, \langle \text{leader}, \text{vd}, \text{wd}, \text{vc}, \text{va}, w \rangle, \text{reply} = \text{false} \) )

25 \( \mathcal{H} \leftarrow \mathcal{H} \cup \{ \langle \text{vd}, \text{wd}, \text{vc}, \text{va}, w \rangle \} \) // enqueue estimates to \( \mathcal{H} \) where \( |\mathcal{H}| = l \)

26 \( \text{vd} = \text{vd} + \text{m.vd}, \text{wd} = \text{wd} + \text{m.wd}, \text{vc} = \text{vc} + \text{m.vc}, \text{va} = \text{va} + \text{m.va}, w = w + \text{m.w} \)

27 leader \( \leftarrow \max(\text{leader}, \text{m.leader}) \) // elect leader node

ECP consists of four subsequent phases: Aggregation, Convergence, Agreement and Commit. It maintains a tuple containing three aggregation pairs \( \langle \text{vd}, \text{wd} \rangle \) for data aggregation, \( \langle \text{vc}, \text{va} \rangle \) and \( \langle \text{vc}, \text{va} \rangle \) for Convergence and Agreement phases respectively. The tuple also contains \text{leader} for the computation of the election. In order for ECP to function, two other protocols are involved in the solution, System Size Estimation Protocol (SSEP) [18] and Node Cache Protocol (NCP) [19]. SSEP is an independent epidemic protocol that provides a robust estimate of system size and exports the size using the service getSize(). The protocol NCP is a simple dynamic topology management protocol that exports a peer sampling service through getRandomPeer().

In the Aggregation phase, ECP computes the global average for the distributed set of initial data values in the system. Each node \( i \) holds a numeric aggregation pair \( \langle \text{vd}_i, \text{wd}_i \rangle \) [19] where \( \text{vd}_i \) is set to local value \( x_i \) and the weight is set to \( \text{wd}_i = 1 \). Node \( i \) halves the values of its pair \( \langle \frac{\text{vd}_i}{2}, \frac{\text{wd}_i}{2} \rangle \) and sends the results to a random peer. At the reception of a message from node \( j \), node \( i \) halves its pair values, replies to \( j \) and updates its local pair values \( \langle \text{vd}_i + \text{vd}_j, \text{wd}_i + \text{wd}_j \rangle \). And thus, the initial pair values \( \langle \text{vd}_i, \text{wd}_i \rangle \) at each node \( i \) are divided and evenly distributed to the entire system. After a number of cycles, data values eventually converge to \( \frac{1}{n} \sum_{i=1}^{n} \text{vd}_i \) and weight values converge to \( \frac{1}{n} \sum_{i=1}^{n} \text{wd}_i = 1 \). An approximation of the global average can be estimated at each node \( i \) by \( \frac{\text{vd}_i}{\text{wd}_i} \) at any cycle.

In the Convergence phase, nodes attain awareness of convergence of other nodes. This is accomplished by computing the global count of nodes converged to the global average in the Aggregation phase. The global count function requires each node \( i \) to hold a pair \( \langle \text{vc}_i, \text{va}_i \rangle \) [19] such that \( \text{vc}_i = 1 \) at all nodes, \( \text{va}_i = 1 \) at a single node \( i \) and \( \text{va}_i = 0 \) where \( 0 < i < n \) and \( i \neq i \). After a sufficient number of cycles, \( \text{va}_i \) will distribute equally in the system and each node will converge to a target fraction \( \frac{1}{n} \) where \( n \) is the initial system size. The number
of nodes can be estimated locally by \( \frac{v_c}{w_i} \) at every node \( i \), whereas \( \frac{1}{n} \sum_{i=1}^{n} v_c = 1 \) and \( \frac{1}{n} \sum_{i=1}^{n} w_i = \frac{1}{n} \).

The Agreement phase adopts the same mechanism in the Convergence phase to attain the certainty of global agreement. The pair \( \langle v_a, w_i \rangle \) is used to calculate the global count in this phase. The same setting of \( w_i \) is used for the sake of optimisation as described later in this section. Finally, ECP phases end in the Commit phase representing the reach to the explicit global agreement and the right moment to take a system-wide decision or a global action.

ECP detects convergence by computing the relative error value of the local approximation result at each cycle using dedicated heuristic methods [20]. For this purpose, two error thresholds \( \varepsilon_1 \) and \( \varepsilon_2 \) are defined to enable distinct accuracy measurements in different phases. The determination of the threshold value is an application requirement that trades-off convergence speed against approximation accuracy. The error threshold \( \varepsilon_1 \) is used for accuracy in the Aggregation phase and \( \varepsilon_2 \) is used in the subsequent phases. The convergence in the Aggregation phase is detected using moving average method. Thus, each node maintains a fixed length history queue \( H \) and stores local estimate and a peer’s estimate after each received message. At each cycle, each node examines the Coefficient of Variance \( \bar{C}_v = \frac{H}{\bar{H}} \) for all elements in the queue \( H \) where \( \bar{H} \) is the standard deviation and \( \bar{H}_x \) is the average. The local convergence is detected when \( \bar{C}_v \) decreases below \( \varepsilon_1 \) for a number of consecutive cycles \( T \). The consecutive cycles threshold \( T \) is used to avoid precocious convergence detection. On another hand, the threshold error \( \varepsilon_2 \) is used for the transition in the Convergence and the Agreement phases. The transition is made when the percentage of absolute error between the count approximation and the size provided by SSEP falls below the error threshold for \( \Upsilon \) cycles.

The ECP carries out an epidemic election to appoint a single node as a leader [19]. The weight value \( w_i \) in the leader node \( i \) will be set to \( w_l = 1 \) as required by the count aggregation function. Leader election assumes that each node has a unique global identifier which can be the IP address and the local port. The leader is the node with the highest identifier. The convergence to a leader node is achieved when the local leader estimate holds for \( \Upsilon \) cycles. Leader election is embedded into Aggregation phase and starts immediately when the system starts allowing early propagation of \( w_l \) and enabling faster convergence in the subsequent phases.

The protocol ECP is illustrated in Algorithm 1, at each cycle, the protocol halves the aggregation pairs and sends the pairs to a random peer in lines 4-6. In line 9, the protocol detects the convergence in the Aggregation phase. Lines 10-11 in the protocol makes the transition to the Convergence phase. In line 12, the convergence to a leader node is detected and line 13 sets the value of \( w \) at the leader node. In lines 15 and 19, the criterion decides upon the transition to Agreement and Commit phases. Lines 16-17 trigger the transition to the Agreement phase and line 20 triggers the transition to Commit phase. At the reception of a message, ECP halves the aggregation pairs in lines 22-23 and responds in line 24. In line 25, the protocol stores estimates of the Aggregation phase in the history queue \( H \). In line 26, the protocol updates the local pairs and computes a new leader in line 27.

5 Simulations and Experimental Results

5.1 Distributed System Model

The model of the distributed system consists of large number of nodes \( n \). Nodes are connected to the Internet and communicate using the uniform gossiping paradigm [21]. A connected physical topology and a reliable transport protocol are assumed. On another hand, a partial synchrony setting is adopted in the model influenced by the impossibility result [22] that refers to the unattainable detection of consensus in an asynchronous distributed system of unreliable processes. The setting identifies the minimal properties of distributed systems that are needed to solve the consensus problem [10]. Upper bounds of relative process speed and communication latency are defined but unknown to nodes. Each node starts the randomised communication at the beginning of each cycle. Cycles are time intervals of fixed length and the length is equivalent to the average Round-Trip-Time \( (RTT) \) of the Internet [23, 24]. The start of the first cycle at each node is uniformly distributed within a bound offset \( T_{start} \) equivalent to maximum drifts among internal clocks in network nodes. After \( T_{start} \), protocol cycles within a node do not overlap, however, the subsequent cycle of different nodes may overlap. The value of \( T_{start} \) is adjusted to 100 milliseconds.

Communication Latency is generated using the Gaussian distribution where the mean of send/receive delays is \( T_{delay} = 200 \) milliseconds and standard deviation \( \sigma = 75 \) milliseconds such that 50% of messages are delivered in \( T_{delay} \), 95% are delivered within \( T_{delay} \pm 2\sigma \) milliseconds and the rest are eventually delivered but after long delays. The minimum message delay is 50 milliseconds. The cycle length \( T_{cycle} \) is set to the maximum \( RTT \) value and thus \( T_{cycle} = 2 \times T_{delay} \). The prior setting is a reasonable trade-off between maximum latency needed for most messages to be delivered and minimum process speed. The setting forms a practical implementation of a large-scale network of simultaneous processes and asynchronous communication where the presence of message interleaving [25], late arrival messages and out-of-cycle messages [26] are present at all times.
5.2 Simulations Model and Configurations

Simulations are carried out using PeerSim [27], a Java-based discrete-event P2P simulation tool. PeerSim is flexible, configurable and scalable. The simulation model is fully Event-based and uses the event-driven engine in PeerSim. Two events are defined in the model: (i) The Activate Event is scheduled within $T_{start}$. The event then occurs at every $T_{cycle}$ and stops after a predefined number of cycles. At this event, a node contacts a random peer and makes assessment and phase transitions. (ii) The Message Receive Event occurs when a node receives a message from a peer. At this event, the incoming message is processed. The local aggregation pairs are updated and a leader is computed.

The simulation model includes four protocols, ECP, PTP+ and two tree-based 3PC protocols. The protocol PTP+ is inspired by Phase Transition Protocol (PTP) in [18]. PTP+ is adopted earlier in the formulation of this solution to achieve consensus in distributed data aggregation. PTP+ is used in the comparative simulations for the evaluation of ECP.

The protocol 3PC is simulated over a static binary tree [28]. Nodes identifiers are assumed globally unique and incremental. The node with identifier 0 is the coordinator and each node $i$ has two child nodes $2i + 1$ and $2i + 2$. The binary spanning tree is constructed by a dedicated initialiser prior to simulations. 3PC is implemented to achieve the global agreement on the outcome of the distributed averaging over the same data distribution that is used in ECP and PTP+. Two versions of 3PC protocol are used in the experimental simulations, a classic 3PC and a modified version of 3PC motivated by the CTP [15] namely 3PC-Convergecast or (3PC-C). In 3PC-C, the step of broadcasting compute message is omitted and the protocol starts in a Convergecast step from the depth of the tree towards the coordinator. The subsequent phases of 3PC-C proceeds as same as in classic 3PC. This optimisation improves the total time required for 3PC-C to achieve consensus and thus challenge the performance competition with ECP.

The performance, convergence and communication overhead of ECP, PTP+, 3PC and 3PC-C are periodically monitored using dedicated observation modules. All protocols stop when the global agreement is achieved and the simulation terminates when the total number of cycles is reached. Different random seeds are used in each simulation to enforce randomisation and each experiment is repeated for tens of times to validate the setting and ensure results. The protocols are initialised by a Peak data distribution where $v_{di} = n$ in a single node $i$ and $vd_0 = 0$ where $0 < i \leq n$ and $i \neq i$. Local parameters in ECP are tuned carefully to certain values depending on the experimental results of the internal performance of the protocol in Section 5.3. The error thresholds $\varepsilon_1$ and $\varepsilon_2$ are set to $\varepsilon_1 = \varepsilon_2 = 0.01$. The consecutive cycles threshold is set to $\Upsilon = 5$. The length of the history queue $H$ is set to $l = 10$. NCP is configured to maintain a random k-regular overlay with $k = 10$.

5.3 Results and Discussion

The execution of ECP involves local detection of convergence and the transition across phases. Phases transition towards the explicit global agreement in the ECP is illustrated in Figure 1. Figure 1a shows the percentage of nodes in each phase over time. The figure also illustrates the smooth transition from a phase to the successive. Figure 1b shows the average of estimates in each phase. It is clear that estimates in each phase converge to the same approximation value at all nodes. In the Aggregation phase, local estimates converge to 1 which is the correct average of spreading $vd_i = n$ over $n$ nodes. Estimates in Convergence and Agreement phases converge to $n$ as expected. In Figure 1c, the variance of estimates over all nodes is tending towards very small value indicating the reduction in estimation error and the reach of convergence among nodes in each phase. Results in Figure 1 validates the ability of ECP to locally detect convergence, makes the transition in phases and attain the certainty of the explicit global agreement on the outcome of the global averaging.

The internal performance of the ECP is examined by varying one of the associated parameters in each experiment. The error thresholds $\varepsilon_1$ and $\varepsilon_2$ are used in different phases, the effect of each one can be recognised by monitoring the corresponding phase, and hence both parameters are set to the same value. Figure 2 shows linear rising in the completion times of each phase when error thresholds are set for a higher accuracy. The values of $\varepsilon_1$ and $\varepsilon_2$ can be tuned to trade-off between accuracy and speed. For instance, a small error threshold can be used in Aggregation phase whilst using bigger one in Convergence and Agreement phases to speed up convergence. On another hand, the use of higher values of $\Upsilon$ significantly slows the detection of convergence in each phase. Thereby, $\Upsilon$ is set to 5 which allows feasible convergence speed for large network sizes up to one million nodes. Also, a small delay in the completion time of Aggregation phase is noticed when the length of history queue $H$ increases as shown in Figure 2b, and thus the use of minimum reasonable length is preferable for faster convergence and less execution load.

A summary on the comparative experiments among ECP, PTP+, 3PC and 3PC-C protocols is illustrated in Figure 3a. The phases transition and completion times of each protocol are illustrated in Figure 3b. The comparison involves the performance of protocols over various system sizes. Figure 3a shows the modest linear increase in the completion times in ECP. Also, it shows faster completion times in comparison to PTP+ reflecting the use of tuned parameters and the early leader election mechanism. The completion times of 3PC
and 3PC-C protocols significantly rise w.r.t the system size. The increase in depth of the tree increases the convergence time in 3PC and 3PC-C protocols. The ECP shows equivalent performance in small sizes but performs much better in large sizes. ECP outperforms tree-based 3PC and 3PC-C protocols without the need to establish any specific network structures.

The communication overhead in ECP, 3PC and 3PC-C protocols are illustrated in Figure 4. Figure 4.a shows the average number of messages at each cycle in ECP. Each node sends two messages in average at each cycle, one to contact a random peer and one to reply to an incoming message. Figure 4.a illustrates the distribution of communication load among all nodes in ECP. Figure 4.b illustrates the cumulative number of messages sent in each phase. ECP produces higher overhead in each phase due to the continuous communication natural of epidemic protocols.

The communication overhead in tree-based 3PC and 3PC-C protocols are shown in Figures 4.c-4.f. In general, nodes in 3PC and 3PC-C protocols send two compute messages to child nodes in Broadcast step and two acknowledgement messages are sent back to the parent node in CONVERGECAST step in each phase. Broadcast messages and CONVERGECAST messages occur at different cycles and encounter different latency, hence two messages are sent in average per-node at each cycle. As the structure of the tree expands towards the depth, the number of messages increases in the Broadcast step and decreases in the CONVERGECAST step until nodes reach the global agreement. At the most depth of the tree, the number of sent messages is \( \frac{1}{2}n + 1 \) and thus in the corresponding cycle, around \( \frac{1}{4} \) message is sent in average per-node. Figures 4.c and Figure 4.e illustrate this effect in 3PC and 3PC-C protocols. Figures 4.d and Figure 4.f, shows the cumulative number of messages in each phase which is noticeably less than ECP protocol.

In summary, ECP achieves consensus faster than tree-based protocols in large system sizes. Tree-based protocols have optimal total communication overhead whilst ECP has higher overall overhead. The overhead in ECP is distributed over system nodes and hence the per-node overhead is perfect.

6 Related work

The protocol PTP is proposed by [18] to achieve consensus on the convergence of information dissemination over large number of nodes. A simple application scenario (IDA) is used to demonstrate the key idea of the solution. PTP achieves consensus in IDA for multiple items without a global uniqueness of items identifiers and without centralised coordination or prior knowledge of system size. The work in [4] introduces three failure detection and consensus algorithms using randomised pinging and the timeout mechanism. In particular, the third algorithm is a three-phase distributed failure detection and consensus algorithm that provides consistency guarantees. The algorithm achieves consensus on failed processes among correct processes using the epidemic aggregation. In [20], a number of heuristic methods to locally detect convergence in epidemic protocols are investigated. For instance, the standard deviation and root-mean-square error of a fixed number of buffered estimates are used as a criterion in the detection formulas.

The work in [29] studied global data aggregation in spanning trees and gossip-based schemes. The study concludes that gossip-based approaches are adaptive and efficient for dynamic topologies while spanning trees are preferable in stable topologies. In [15], an evaluation of performance, communication overhead and accuracy in gossip-based and tree-based approaches are conducted. Randomised gossip, broadcast gossip and CTP protocols are compared for the aggregation of distributed averaging over WSN. The evaluation shows that broadcast gossip is more efficient and requires no prior setting up whilst CTP performance is restrained due to the need of tree maintenance.

The Consensus Problem is described in [10, 11]. Studies on the consensus in asynchronous distributed systems are provided in [9, 16]. In [13], a corrective consensus algorithm to achieve consensus using distributed averaging in WSN is presented. The algorithm is proven to converge to a very close approximation to the correct average. The work in [6] introduces a review on distributed consensus strategies, assumptions and research issues.

The work in [7] provides a wide analysis of Bitcoin challenges. For instance, the double spending problem which requires a decentralised global agreement on the order of cash transactions, and hence ECP could be a choice solution.

7 Conclusions

This paper has presented an innovative epidemic-style approach for the consensus problem in large-scale distributed aggregation. The proposed approach provides a novel practical solution to real-world applications which need to achieve both local convergence and global agreement in a decentralised, scalable, fault-tolerant and consistent way. The solution not only achieves local convergence on the data aggregation target but also attains explicit global agreement among all nodes using a decentralised decision mechanism.
Figure 1: Convergence and phases transition in ECP, $n = 10^4$, $\epsilon_1 = \epsilon_2 = 0.01$, $\Upsilon = 5$, $l = 10$

Figure 2: Phases completion times in ECP when varying simulations parameters, $n = 10^4$

The paper has introduced the Epidemic Consensus Protocol (ECP) for the determination of consensus on the convergence of a distributed data aggregation task. ECP adopts heuristic methods for the local detection of convergence and makes phase transitions to achieve the explicit global agreement. A comparative study of the performance and the communication overhead between ECP and the tree-based 3PC protocol has been presented. Experimental results have shown that ECP reaches the global agreement faster than 3PC in large systems. Although the overall communication load associated with the performance of ECP is noticeably high, the total overhead is distributed among the nodes in the system and hence the overhead at each node is typical. The protocol ECP is fault-tolerant, scalable and requires no structured networks; whilst more effort is needed to establish and maintain tree structures for 3PC and trees are susceptible to propagation delay and single points of failure.

Future research may amend the epidemic consensus approach to attain consistency and reliability in dynamic networks. An adaptive approach with a continuous detection of network churn could be a solution. Other future work may apply the epidemic consensus approach to other agreement problems, e.g. coordination in multi-vehicle networks, decision-making in service oriented IoT and the "proof-of-work" system in Bitcoin.

8 Acknowledgements

The author Mosab M. Ayiad is supported for his PhD project by the Islamic Development Bank-Merit Scholarship Programme for High Technology (MSP), 2015-(600029531).

References

Figure 3: Phases transition and completion times in ECP, PTP+, 3PC and 3PC-C protocols for various system sizes, $\varepsilon_1 = \varepsilon_2 = 0.01$, $\Upsilon = 5$, $l = 10$


Figure 4: Communication overhead in ECP, 3PC and 3PC-C, \( n = 10^4, \varepsilon_1 = \varepsilon_2 = 0.01, T = 5, t = 10 \)