Karlsruhe Institute of Technology (KIT)
Institute of Telematics

TELEMATICS TECHNICAL REPORTS

Analysis of the Random Selection Function in Gossip-based Random Number Agreements

Sebastian Mies
Institute of Telematics, Karlsruhe Institute of Technology (KIT), Germany
mies@kit.edu

July, 5th 2011

TM-2011-3

ISSN 1613-849X

http://doc.tm.uka.de/tr/

Institute of Telematics
Karlsruhe Institute of Technology (KIT)
Zirkel 2, 76131 Karlsruhe, Germany

Analysis of the Random Selection Function in Gossip-based Random Number Agreements

Sebastian Mies
Institute of Telematics, Karlsruhe Institute of Technology (KIT), P.O. Box 6980, D–76049 Karlsruhe, Germany
Email: mies@kit.edu

I. Gossip-based Consensus

Gossip-based consensus algorithms [1] assume a andom network graph and use a gossiping protocol to achieve a consensus between devices. Furthermore, we know that gossiping protocols are highly scalable, robust, and well-understood in theory [2]. The main idea of these protocols is that each period T a node exchanges information (gossip) with its randomly selected neighbors. Thus, each device has a constant "fan-out", (i. e., number of messages sent), and constant "fan-in", (i. e., number of messages received), if the number of neighbors k is constant. Algorithm 1 describes the the random number agreement. When a node starts, it chooses a random number $p \in \mathbb{N}$ and sends this random number to its neighbors and waits for other nodes to send their proposals. Respectively, X receives a sequence of (p_i) proposals from its neighbors after waiting a period T. A node X derives one proposal from the received proposals $p := D((p_i)_{i=1..k})$ using the selection function D. The next iteration of the algorithm sends p to the neighbors. Hence, the function D decides on convergence and properties of the random agreement. The sequence of proposals (p_i) may contain many duplicates (n when the algorithm converged to be exact). To deal with duplicates, let $(u_i) := (u_1, \ldots, u_n)$ be the sequence of pairwise different proposals, ordered descending by $o(u) \in \mathbb{N}_0, u \in U$ that denotes the number of neighbors proposing the same u. Hence, $o(u_1) > \ldots > o(u_n)$. In this technical report the random selection function is analysed:

$$D_{\mathrm{R}}((p_i)_{i=1..k}) := p_x, x \in \{1, \ldots, k\} \text{ chosen at random}$$

```
   // Initialization: choose a random number proposal
1  p ← getRandom();
2  (p_i) ← ();
3  running ← true;
   // Main loop: exchange proposals
4  while running do
       // send proposal to neigbors
5      forall neighbors x do send(x, p);
       // wait and receive propoals from other devices
6      (p_i) ← receive_and_wait();
       // calculate new proposal
7      p ← D((p_i)_{i=1...k});
8  end
9  return;
```

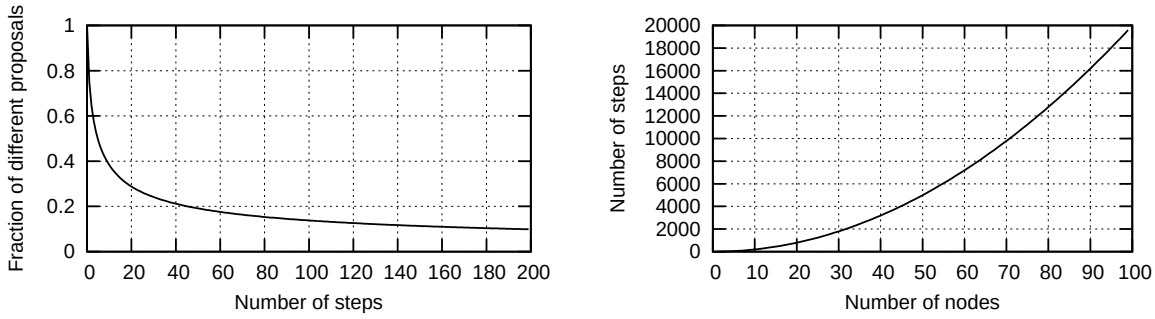
Algorithm 1: A generic random number agreement

II. Analysis of the random selection function

To determine the convergence speed we first show how the function's impact on cycle or path graphs, before generalizing this to arbitrary graphs. Finally, we derive a recurrence relation that describes the reduction of different proposals.

Lemma II.1 (D_{R} on rings and paths)**.** *Let $G := (V, E)$ be a cycle or path of length $|V| \geq 3$, and amount of unique proposals $|V|$. Then, the random agreement algorithm reduces the number of different proposals by $\frac{1}{4} \cdot |V|$ on average in the first, synchronous step using D_R.*

Proof: (Sketch) Let $G := (V, E)$ be a cycle of length $n := |V|$, then each vertice $v \in V$ of G has exactly two neigbors, and therefore receives two proposals it can choose from. Hence, the graph comprises 2^n combinations of possible proposal choices. Analogous, there are 2^{n-2} combinations for a path, since the first, and the last element of the path have no choice to choose the proposal from the only neighbor. These combinations can be interpreted as bit-strings, e. g., 01, which means, if $V := \{A, B\}$, $E := \{\{A, B\}\}$, A chooses the proposal from B and vice versa. The interesting question is, how many combinations lead to a loss of a different proposal (i. e., dupliction of one proposal). One can see that there are exactly two bit-patterns, 001 and 011, that lead to proposal duplication. For our proof we need to determine the number of occurences of these patterns in all combinations. For this purpose, we divide our problem into two areas: first, we consider the occurences in the core of the graph, which has 2^l combinations. Second, we consider occurences in the border of the graph, i.e., the occurences introduced in the cycle by the warp, and path ends. First consider a core of $l = 3$, therefore, 8 combinations need to be considered: $000, 001, 010, 011, 100, 101, 110, 111$. The bit-patterns occured 2 times, hence, $\frac{1}{4}$ of the combinations lead to proposal duplication. This can be generalized, because every additional bit introduces 2^{l-2} occurences of the bit-patterns (note, that this includes overlapping bit-patterns, which leads to multiple loss in one bit-string). Hence, in a bit-string of length $l \geq 3$ the bit-patterns occur $(l-2) \cdot 2^{l-2}$ times. In the next step, we consider the border of the bit-string of a cycle. Without loss of generality, we choose a bit-string with $m = 3$ and count the number of patterns that are wraped in the cycle: $000, 01|0 \rightarrow 001, 001, 1|00 \rightarrow 001, 1|01 \rightarrow 011, 11|0 \rightarrow 01, 111$ (the horizonal bar denotes the split



(a) Fraction of different proposals

(b) Convergence speed

Fig. 1: Fraction of different proposals and convergence speed using the recurrence relation of D_{R}

point of the warp and the value on the right hand side of the arrow denotes the recognized bit-pattern). We count 4 out of 8 combinations, which is half of all combinations. Extending this to a larger number of m, we yield 2^{m-1} combinations that lead to duplication of proposals. Analogous, we yield 2^{m-2} combinations, for paths. Combination of border and core, yields $d_c := (n-2)\cdot 2^{n-2} + 2^{n-1}$ proposal duplications out of $a_c := 2^n$ combinations for cycles, and $d_l := (n-4)\cdot 2^{n-4} + 2^{n-2}$ proposal duplications out of $a_l := 2^{n-2}$ for paths, of length $n \geq 3$. Hence, we yield, for cycles, and paths, that $\frac{d_c}{a_c} = \frac{d_l}{a_l} = \frac{1}{4}\cdot n$, different proposals are lost. ∎

We can extend this lemma to proof, that the same behaviour applies to random graphs, using the following theorem:

Theorem II.2 (D_{R} on random graphs)**.** *Let G := (V,E) be a random graph, and number of unique proposals $|V|$. Then, the consensus algorithm reduces the number of unique proposals* at least *to $\frac{1}{4}\cdot|V|$ on average in the first step using D_R w.h.p.*

Proof: (Sketch) Each vertice $v \in V$ chooses exactly one proposal from its neighbors $|\Gamma(u)|$. Therefore, all choices made, can only lead to either cycles or paths. Hence, Lemma II.1 applies to each of these rings, leading to the loss of $\frac{1}{4}\cdot|V|$ different proposals. ∎

To this point, we gained insights on how the function can indeed reduce the number of different proposals in the system in the first iteration of protocol. We can use these insights to derive a recurrence relations for the next steps of the protocol:

Corollary II.3 (D_{R} convergence)**.** *The upper bound of the number of different proposals n in an arbitrary graph $G := (G,V)$ follows the recurrence relation, $n_0 = |V|$, and, $n_{i+1} := n_i - \frac{1}{4}\cdot\frac{n_i^3}{|V|^2}$, for each step i.*

Proof: (Sketch) We know from theorem II.2, that the number of unique proposals is reduced by $\frac{1}{4}\cdot n$ in an random graph. This is true, iff no duplicate proposals exists. To extend this, we consider pairs of different proposals. The probability that we find such a pair is given by $(\frac{n}{|V|})^2$. This assures that each pair duplicates a proposal. Futhermore, the pairs may result in the same duplicates with probability $1-\frac{n}{|V|}$, hence, we need to assure that the resulting duplicates are unique, and yield $(\frac{n}{|V|})^3$. Therefore the expected number of different proposal loss is $\frac{1}{4}\cdot\frac{n^3}{|V|^2}$ in each step and leads to the recurrence relation of the corrolary. ∎

This recurrence relation can be used to approximate D_{R} convergence time.

This recurrence Since the recurrence is not solveable by using a known methodology (e. g., master-method), we do not try to find a closed-form. We rather use the recurrence to gain more insights on the D_{R}. The first insight is, that D_{R} does indeed reduce the number of unique proposals on average. Figure 1(a) shows the fraction of different proposals as function of algorithm steps. We find that convergence to a single proposal value x needs many steps and gets improbable as the number of vertices increases. However, in the first steps, the slope is very steep, highly reducing the number of different proposals. Figure 1(b) shows the number of steps needed to converge to a single different proposal as a function of the amount of nodes.

III. Conclusion

In this technical report the convergence bounds of the random selection function D_R have been shown.As expected D_R does not scale with the number of nodes. However, from the recurrence relation one can derive that it in fact highly reduces the number of distinct proposals in the first steps. This makes the function a candidate for use with relative majority votes.

References

[1] S. P. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *INFOCOM*. IEEE, 2005, pp. 1653–1664.

[2] B. Pittel, "On spreading a rumor," *SIAM J. Appl. Math.*, vol. 47, no. 1, pp. 213–223, 1987.