Information and Computation 239 (2014) 170–181

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

On the impact of link faults on Byzantine agreement ☆

Martin Biely [a,b,*]

[a] Embedded Computing Systems Group, Technische Universität Wien, 1040 Wien, Austria
[b] Laboratoire de Systèmes Répartis, EPFL, Station 14, 1015 Lausanne, Switzerland

ARTICLE INFO

Article history:
Received 18 January 2006
Received in revised form 3 December 2013
Available online 8 October 2014

Keywords:
Fault-tolerant distributed algorithms
Consensus
Fault models
Link faults
Byzantine faults

ABSTRACT

Agreement problems and their solutions are essential to fault-tolerant distributed computing. Over the years, different assumptions on failures have been considered, but most of these assumptions were focusing on either processes or links. In contrast, we examine a model where both links and processes can fail. In this model we devise a unified lower bound for resilience to both classes of faults. We show that the bound is tight by devising a simple retransmission scheme that allows optimally resilient algorithms to be constructed from well known algorithms by transparently adding link-fault tolerance. Our results show that when considering multiple independent failure modes, resilience bounds are not necessarily additive.

1. Introduction

One fundamental problem arising in distributed systems is that of coordination: How do distributed processes reach agreement on a common course of action? Although this question is easy to answer in the fault free case, it is the central question behind all fault-tolerant agreement problems (see [1] for a survey). One problem from this class is consensus. The consensus problem is defined by every process having some private, initial value and the need to determine an output value, which is required to be the same on all processes. More formally, an algorithm solving consensus in a system with Byzantine failures has to ensure the following properties:

Agreement: No two correct processes decide on different values.
Validity: If all correct processes share their initial value v, then the decision value must be v.
Termination: All correct processes eventually decide.

In the definition above, the set of initial values is not constrained. In the proofs of our lower bound results we argue about a variant of consensus called *binary consensus* in which only two possible values (namely 0 and 1) are considered. This does not weaken our lower bound, as any algorithm solving the problem with an arbitrary sets of inputs must be able to solve the binary variant of the problem.

While the Agreement and Termination properties do not vary, different Validity properties have been considered. Following [2], our Validity property is sometimes also referred to as *Strong Unanimity*. As pointed out in [3] most of the different validity conditions coincide in the case of binary consensus.

☆ This research has been supported by the Austrian START programme Y41-MAT and the FIT-IT project 812205 (TRAFT).
* Correspondence to: Embedded Computing Systems Group, Technische Universität Wien, 1040 Wien, Austria.
E-mail address: biely@ecs.tuwien.ac.at.

http://dx.doi.org/10.1016/j.ic.2014.10.001
0890-5401/

The contribution of this paper is a lower bound on resilience for consensus in synchronous systems in the presence of both link and process failures. In particular, we consider processes that may fail arbitrarily. That is, we consider permanent Byzantine process failures. For link failures, we consider two failure modes: Faulty links can either drop or modify messages. For our lower bound, we assume a static assignment of failures to links. For the matching upper bound, we allow dynamic link failures, i.e., the faulty links may change from one round to the next. The upper bound is shown by introducing a retransmission scheme to mask the link failures allowed by our model, rather than by introducing a (new) consensus algorithm. Using the retransmission scheme any consensus algorithms with optimal resilience to Byzantine failures becomes a consensus algorithm with optimal resilience to Byzantine failures and link failures. While we give the lower bounds for consensus only, the retransmission protocol could be used with algorithms for other problems as well.

The paper proceeds as follows: After introducing our model in Section 2, we show a series of lower bound results in Section 3. In order to develop our main lower bound (Theorem 4), we will start by presenting the impossibility of solving consensus, when, for each process, at most one incoming link drops messages, at most one (additional) incoming link can modify or drop messages, and likewise for the outgoing links. This result is based on the intuitive proof technique, called *scenario argument* by Fischer, Lynch, and Merritt [4]. We then generalize this result for larger numbers of link failures (that is larger systems) using the same technique. This contrasts with the approaches taken in [4] and [5], which use different techniques for extending the one-failure case to larger numbers of failures. (We discuss these approaches in more detail in Section 6.) Apart from making the proofs easier to understand, the advantage of our approach is that it can be used to find a unified lower bound for link and process faults, which is presented in Section 4. Up to now, lower bounds were typically either concerned with links or process failures. One might assume that bounds for combinations of link and process failures are simply obtained by adding the two bounds. It is well known that consensus requires $n > 3f$ in the presence of up to f Byzantine processes [6]. Moreover, Schmid, Weiss, and Keidar [5] have established that $n > 4\ell$ is required, where ℓ is a system parameter limiting the number of value faulty links. Thus one might expect a composite lower bound to be $n > 3f + 4\ell$. The main contribution of this paper is to show that this intuition is wrong. Rather, it is sufficient to have $n > \max\{2f + 4\ell, 3f\}$. This bound is shown to be tight by the lower bound results in Section 3. Sufficiency is established by presenting a retransmission scheme which requires $n > 2f + 4\ell$ to mask link failures (Section 5). This allows us to reuse existing consensus algorithms with optimal resilience. The resilience is not changed because link failures are masked. Here, the interesting observation is that, in the presence of Byzantine process faults, some link faults can be tolerated for free.

2. Model

Our model is based on the work of Fischer, Lynch, and Merritt [4] as well as the perception based fault model, e.g., [7]. In particular, our assumptions are based on the latter, while the modeling follows the former more closely, although the inclusion of arbitrary link faults requires some changes.

A communication graph is a directed graph G with node set $nodes(G)$ and edge set $edges(G)$. A *system* is a communication graph with an assignment of an input value and a process to each node in $nodes(G)$. Processes have identifiers and execute some algorithm. As the algorithms we consider in this paper are limited to binary consensus algorithms, the input values are taken from the set $\{0, 1\}$. Throughout the paper we will use $p_i[v]$ as a way to identify the node running a process with identifier i and input value v. In certain cases identifiers are not unique, but the combination of identifier and input value will be. If identifier and input value are clear from the context or do not matter for the discussion at hand, we will usually omit them.

In this paper we limit our attention to round based algorithms. Therefore we can define the *behavior* of a node p to be a collection of functions $\sigma_p^j(r)$, one for the identifier, j, of each out-neighbor of node p, which maps the round number, r, to the message node p sends to the node with identifier j in round r or $\perp$, if p does not send a message to that node in that round. We similarly define the *edge behavior* $\delta_e(r)$ of an edge e to map each round number r to either a message or $\perp$. Thus the behavior of both nodes and edges is defined based on the sequence of messages they emit.

As mentioned above, processes execute algorithms. An algorithm is defined by a set of states, some of which are initial states, its transition function $\tau(s, R)$, which specifies how the state evolves from round to round based on the previous state s and the set R of received messages, and a message function $\mu(s)$, which specifies what message a process running the algorithm is to broadcast based on the state s. A run of the process running on node $p_i[v]$ starts from the round 0 state, which encodes both the identifier i and the input value v. For each round $r \geq 0$, it determines the messages received based on $\delta_e(r)$ of each incoming edge e of $p_i[v]$ and then determines the round $r + 1$ state based on the round r state and these messages. The *process behavior* $s_p(r)$ of the process running on node p maps round r to the state of the process at round r. A *system behavior* is the collection of behaviors of the nodes, processes, and edges that constitute the system.

We can now define what constitutes correct and faulty behavior of nodes and edges. An edge e from node q to $p_i[v]$ exhibits correct behavior if, for each round r, $\delta_e(r) = \sigma_q^i(r)$. That is, in each round the message it delivers is what the node behavior of q specifies for processes with identifier i. If e is not correct, but $\delta_e(r) \in \{\perp, \sigma_q^i(r)\}$ for all rounds r, we say the edge exhibits message loss. In all other cases, we say the edge exhibits arbitrarily faulty behavior. Note that, in our impossibility proofs, we will consider any edge e that exhibits message loss to lose all messages, that is, for every round r, $\delta_e(r) = \perp$.