Distributed Protocols on General Hybrid Adversary Structures

Klaus Kursawe
Katholieke Universiteit Leuven
Email: klaus.kursawe@esat.kuleuven.ac.be

13th July 2004

Abstract

Most of the literature of fault tolerant computing applies a threshold point of view: for a given number n of parties, there is a constant t such that up to t failures can be tolerated.

In the real world, there are dependencies between failures; if a program crashes because of an incompatibility with the operating system, other parties with the same operating system are more likely to crash as well, if it fails because of a corrupted system administrator, all parties within this administrative domain can be corrupted.

To deal with these real world structures, *adversary structures* have been introduced. Instead of defining thresholds, all sets of parties that may fail together are defined, leading to much mode flexibility.

This paper defines the necessary structures and tools to adapt protocols to this more realistic model, and demonstrates them by transferring a reliable broadcast protocol into a setting where both crash– and malicious failures occur, and where the failure tolerance is defined by an adversary structure.

1 Introduction

So far, the larger part of fault-tolerance literature takes a rather homogenous view of the world; all parties are considered equal, and all failures happen independently. In the original fault tolerance settings such as aircraft control systems, this assumption is justifiable; each component has its own, more or less independent, failure probability which determines the number of redundant systems.

While one can argue if failures of modern computing systems are still independence, introducing malicious failures and active attacks makes it impossible to justify the assumption of fault independence. In fact, the first papers on Byzantine fault tolerance where never meant to consider malicious faults in the first place for exactly this reason.

To reflect the real world, it is crucial to consider dependencies between failures. Even without malicious attackers, various factors influence the systems deployed today. Parties at the same physical location may fail due to a power failure, and systems with the same operating-system may be victims of the same software bug.

In this paper, we go beyond the traditional t– out-of-n failure model. Our goal is to represent structures that exist in the real world — corruptions do not happen independently, and some failures are more likely to occur than others. The approach taken here is rather general and has been tested on several protocols for different problems. We demonstrate the transition on a relatively simple protocol, the reliable broadcast as defined in [3, 1]. Essentially, we consider two ways to adapt the protocols:

Hybrid Failures. Instead of only accepting Byzantine failures, we want to tolerate a mixture of Byzantine and crash failures. This allows the protocol to tolerate more overall failures. More precisely, if c is the number of crash failures, and b is the number of Byzantine failures, then the necessary and sufficient condition is $2c + 3b < n$.

Adversary Structures. Instead of tolerating a fixed number of failures, *adversary structures* define sets of parties that may fail simultaneously. This does not increase the number of tolerable failures, but adds a degree of flexibility which is needed to model real world structures. There are relative direct correspondences between the threshold model and adversary structures, which allow for a generalization of existing protocols.

2 Definitions

Instead of specifying the number of corruptions tolerated, an *adversary structure* $\mathcal{Z}$ explicitly specifies all sets of parties (the *coalitions*) which may be corrupted. Numeric conditions (for example, that the number of corrupt parties is less than a third of the total number of parties) are then replaced by conditions on the allowed coalitions (e.g., that no union of three of those sets is the full set of parties).

This does not allow us to tolerate more corruptions. Rather, we can shift the balance, i.e., we can tolerate much more than a third of corruptions (if the right parties are corrupted), but it the same time much less (if the wrong parties are corrupted). Initially, they consider active and passive failures separately, whereas the corresponding adversary structures are as follows:

Definition 2.1. Call $\mathcal{P}$ the set of all parties involved in the protocol, i.e., $\mathcal{P} = \{P_1, \ldots, P_n\}$.

We now define all sets $c \subset \mathcal{P}$ of parties that we allow to be corrupted. An adversary structure $\mathcal{Z}$ is a monotone set of subsets of $\mathcal{P}$. It satisfies the predicates

$Q^2(\mathcal{P}, \mathcal{Z})$, if no two coalitions $A, B \in \mathcal{Z}$ satisfy $A \cup B = \mathcal{P}$.

$Q^3(\mathcal{P}, \mathcal{Z})$, if no three coalitions $A, B, C \in \mathcal{Z}$ satisfy $A \cup B \cup C = \mathcal{P}$.

We can show that Q^2 is a necessary condition to solve agreement in the crash failure model, while Q^3 is necessary in the Byzantine model.

If one puts more restrictions on the number of Byzantine failures, additional crash failures can be tolerated, in a way that the total number of tolerable failures increases.

A similar approach has been taken in [4], where the two types of failure are *active Byzantine* and *passive Byzantine.*

The coalitions defined by the adversary structure $\mathcal{Z}$ now consist of pairs $(\mathcal{B}, \mathcal{C})$, where $\mathcal{B}$ is the set of malicious parties and $\mathcal{C}$ is the set of crashing parties. We do not consider crash-recovery failures at this point, as those failures have no effect on the required thresholds.

Definition 2.2. Let $\mathcal{P}$ be the set of all parties and $\mathcal{Z}$ an adversary structure on $\mathcal{P}$ like in Definition 2.1

We say that the adversary structure $\mathcal{Z} = \{\mathcal{B}, \mathcal{C}\}$ satisfies the predicate $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$, if no combination of two coalitions in $\mathcal{Z}$ and the Byzantine parties in one coalition in $\mathcal{Z}$ covers $\mathcal{P}$. More formally,

$$Q^{(3,2)}(\mathcal{P}, \mathcal{Z}) \Leftrightarrow \forall (B_1, C_1), (B_2, C_2), (B_3, C_3) \in \mathcal{Z} : B_1 \cup B_2 \cup B_3 \cup C_1 \cup C_2 \neq \mathcal{P};$$

In particular, if no crash failures are allowed, then $\mathcal{Z}$ satisfies $Q^3(\mathcal{P}, \mathcal{Z})$; if no Byzantine failures are allowed, then $\mathcal{Z}$ satisfies $Q^2(\mathcal{P}, \mathcal{Z})$. Thus, both $Q^3(\mathcal{P}, \mathcal{Z})$ and $Q^2(\mathcal{P}, \mathcal{Z})$ are special cases of $Q^{(3,2)}(\mathcal{P}, \mathcal{Z})$.

We start by defining sets that correspond to the different thresholds usually needed in the threshold model. The protocols mainly investigated belong to the SINTRA protocol suite [6]; other asynchronous protocols can be converted into the new model as well, though some may require sets with different properties. Other protocols, such as Byzantine Quorum Systems [7], require different thresholds - while this should not pose a significant problem, some more work is required to generalize the transformation that far.

Definition 2.3. Let $\mathcal{Z}$ define an adversary structure that satisfies $\mathcal{Q}^{(3,2)}(\mathcal{P}, \mathcal{Z})$. A set $\mathcal{A} \subset \mathcal{P}$ is called a

– *full set*, if there exists a class $(C, B) \in \mathcal{Z}$, such that $\mathcal{A} \supseteq \mathcal{P} \setminus (B \cup C)$.

– *big set*, if there exists two classes (C_1, B_1) and $(C_2, B_2) \in \mathcal{Z}$, such that $\mathcal{A} \supseteq \mathcal{P} \setminus B_1 \cup C_1 \cup B_2$.

– *small set*, if there is no class $(C, B) \in \mathcal{Z}$ such that $B \supseteq \mathcal{A}$.

More intuitively, a $\mathcal{A}$ is a full set if all parties in $\mathcal{P} \setminus \mathcal{A}$ might be faulty; after receiving messages from a full set $\mathcal{A}$ of parties, a party can not expect to receive more messages. This corresponds to the threshold $n - t$ in our previous notation. Given a full set of parties, some of which are Byzantine, a *big set* is the set that remains when the Byzantine parties are taken away. This corresponds to the threshold $n - 2t$ in our conventional notation. Finally, a small set of parties is a set that contains at least one non-Byzantine party. This corresponds to the threshold of $t + 1$.

After defining the sets, we need some elementary properties. These properties originate in the properties of the original thresholds that where used to prove the protocols in the literature.

Lemma 2.1. *The following properties hold:*

P1. *Any big set contains at least one non-Byzantine party.*

P2. *Two full sets always have at least one non-Byzantine party in common.*

P3. *Each full set of parties contains at least a big set of non-Byzantine parties.*

P4. *If a full set of parties performs a action X, and no non-Byzantine party performs both actions X and Y, then no full set of parties performs action Y.*

P5. *Every big set is a small set.*

P6. *Each pair of a full set and a big set of parties intersects.*

3 Example: Efficient Reliable Broadcast

Reliable broadcast provides a way for a party to send a message to all other parties. It requires that all honest parties deliver the same set of messages and that this set includes all messages broadcast by honest parties. However, it makes no assumptions if the sender of a message is corrupted and does not guarantee anything about the order in which messages are delivered. The definition and the original protocol can be found in [3], with some adaptions from [1].

```
Protocol HQ3-RBC
  input: message $m$, round number $r$.
  upon initialization
        $\mathcal{S}_m \leftarrow \emptyset$
        $\mathcal{R}_m \leftarrow \emptyset$
  upon $(ID.j, \text{r-broadcast}, m)$
        send $(ID.j, \text{r-send}, m)$ to all parties
  upon receiving message $(ID, j, \text{r-send}, m)$ from $P_l$ for the first time
        if $j = l$,
            send $(ID.j, \text{r-echo}, m)$ to all parties.
  upon receiving message $(ID, j, \text{r-echo}, m)$ from $P_l$ for the first time
        $\mathcal{S}_m \leftarrow \mathcal{S}_1 \cup \{m\}$
        if $\mathcal{S}_1$ is a full set for the first time, and $\mathcal{R}_m$ is not a small set then
            send $(ID.j, \text{r-echo}, m)$ to all parties.
  upon receiving message $(ID, j, \text{r-ready}, m)$ from $P_l$ for the first time
        $\mathcal{R}_m \leftarrow \mathcal{R}_1 \cup \{m\}$
        if $\mathcal{R}_m$ is a small set for the first time, and $\mathcal{S}_m$ is not a full set then
            send $(ID, j, \text{r-ready}, m)$ to all parties.
        else if $\mathcal{R}_m$ is a full set,
            output $(ID.j, out, \text{r-deliver}, m)$
```

Figure 1: Protocol HQ3-RBC

4 Attribute Based Structures

The definitions presented so far allow for a great amount of flexibility, but that comes with a price. As the number of possible sets increases exponentially with the number of parties, defining all possible sets may be unmanageable if n grows sufficiently large.

Thus, for practical purposes, it may be necessary to artificially reduce the flexibility offered by the scheme to allow for easier management of sets. One idea to reach this goal is to use *attribute based* structures [2].

In this, every party has several attributes, such as the geographical position, the operating system, or the owner. Instead of allowing $t-$ out of n parties to fail, one now allows a number of attribute values to fail, e.g., two out seven geographical areas and one out of four operating systems.

5 Conclusions

In this paper we presented an approach for designing fault tolerant applications, where it is possible to tolerate different kinds of failures and consider correlations between failures using adversary structures. It is relatively easy to transform threshold based protocols into the new setting, as there are direct equivalences between the two models.

While adversary structures provide a lot of flexibility, they may be hard to manage in practice; an implementation has to be aware of the entire adversary structure. If secret sharing, or cryptography based on shared secrets is involved, this might get difficult, as it may be required to have an independent sharing for each and every set of honest parties that is allowed to reconstruct the secret.

One way to make adversary structures more manageable is to restrict the flexibility by using attribute based structures [2, 6, 5]. In this model, there are some number of attributes that characterize a party. In practice, those attributes represent dependabilities, i.e., parties with the same attribute are more likely to fail together than parties with different attributes. For example, the operating system or the geographical location of a party would be suitable attributes. This does not change anything on the protocols described here, as the changes are more on an implementational level. To fully utilize the attributes, however, it is necessary to carefully redo the cryptographic proofs in the new model.

Our model allows to represent failure dependencies in the most general way, but it does not take into account failure probabilities; there is no way of modeling a statement like “If P_i fails, then the probability of P_j failing increases by 10%”. It is possible to use such probabilities to compute the optimal adversary structure, but it is not clear how a protocol could profit from knowing about such probabilities.

References

[1] M. Bakes and C. Cachin. Reliable broadcast in a computational hybrid model with byzantine faults, crashes, and recoveries. Technical Report RZ 3466, IBM Research, November 2002.

[2] C. Cachin. Distributing trust on the Internet. In *Proc. Intl. Conference on Dependable Systems and Networks (DSN-2001)*, pages 183–192, June 2001.

[3] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup. Secure and efficient asynchronous broadcast protocols. Research Report RZ 3317, IBM Research, 2001. Also available from Cryptology ePrint Archive, Report 2001/006, `http://eprint.iacr.org/`.

[4] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In K. Y. Lam, E. Okamoto, and C. Xing, editors, *Advances in Cryptology - ASIACRYPT '99*, volume 1716 of *Lecture Notes in Computer Science*, pages 232–246. Springer-Verlag, 1999.

[5] F. Junqueira, K. Marzullo, and G. Volker. Coping with dependent process failures. Technical Report TR CS2002-0723, University of California San Diego, 2002.

[6] K. Kursawe. *Distributed Trust.* PhD thesis, University of Saarbrücken, March 2002.

[7] D. Malkhi and M. Reiter. Unreliable intrusion detection in distributed computation. In *CSFW97*, 1997.