Appendix A

Basic Concepts of fault tolerance

A fault tolerant system can continue with its work in the presence of a fault with the purpose of supporting the operation the system. A fault is a defect or flaw that occurs within some hardware or software component. Understanding a technical area as complex as fault tolerance distributed computing requires identifying its fundamental concepts and naming them unambiguously.

On information distributed systems three stages are distinguished on the evolution from a correct system state in which the service provided corresponds with the specification to the opposite state. The stages are fault, error and failure; they correspond by its order to the damage layer on the system.

A.1. Fault, error and failure

A system is said to fail when it cannot meet its promises, which is when it does not provide a correct service. Correct service is delivered when the service implements the system function described by its specification. A system may fail either because it does not comply with the specification, or because the specification did not adequately describe its function. A failure is a transition from correct service to incorrect service. A transition from incorrect service to correct service is service restoration. An error is a part of the system state that may cause a subsequent failure; a failure occurs when an error reaches the service interface and alters the service. A fault is the adjudged or hypothesized cause of an error. A fault is active when it produces an error; otherwise it is dormant [ALR2001].

According to an analogous definition on [DQP2002], the term fault is utilized to designate a defect, by example, when a memory cell returns always the value 0 without caring the value that it has been registered there. By consequence, an error is defined as the fault manifestation over the systems state. An error is a part of the system state susceptible to cause a failure. A system incurs on a failure when its behavior does not correspond with the service specification. By the definition a fault presents to be the cause of an error on the system state, and this put the system on a failure.

A failure is exposed to the exterior when its cause is a fault. It is important to consider that when a failure appears, surely exists a fault on the system, nevertheless the simple existence of a fault does not lead unavoidably to the appearance of a failure. Figure A1 illustrates the concepts fault, error and failure.

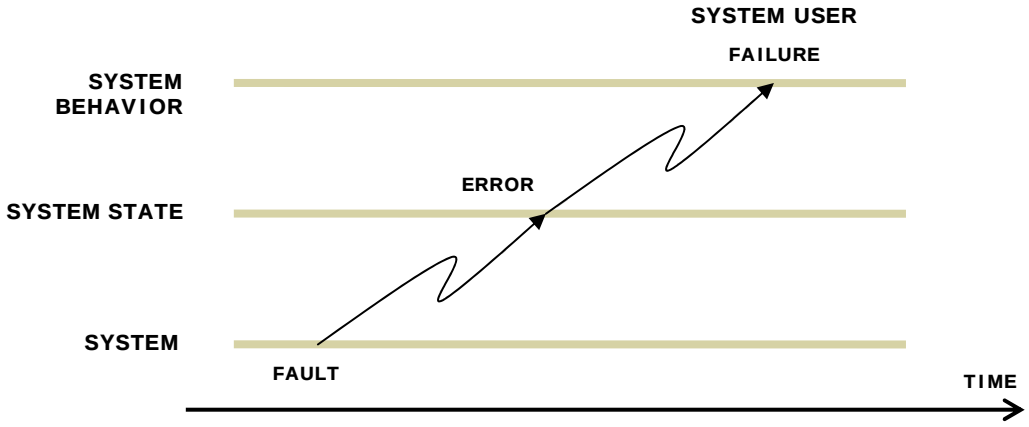

Figure A1 Fault, error and failure

A system component on a failure state is able to affect other(s) component(s) that have a dependency relation with it. It means that the failure on a sub-system constitutes a fault for the global system. Figure A2 illustrates a system conformed by many sub-systems, where the service provided by every sub-system is represented by the semi-circle in bold, and the separation border between sub-systems is represented by the semi-circle in points.

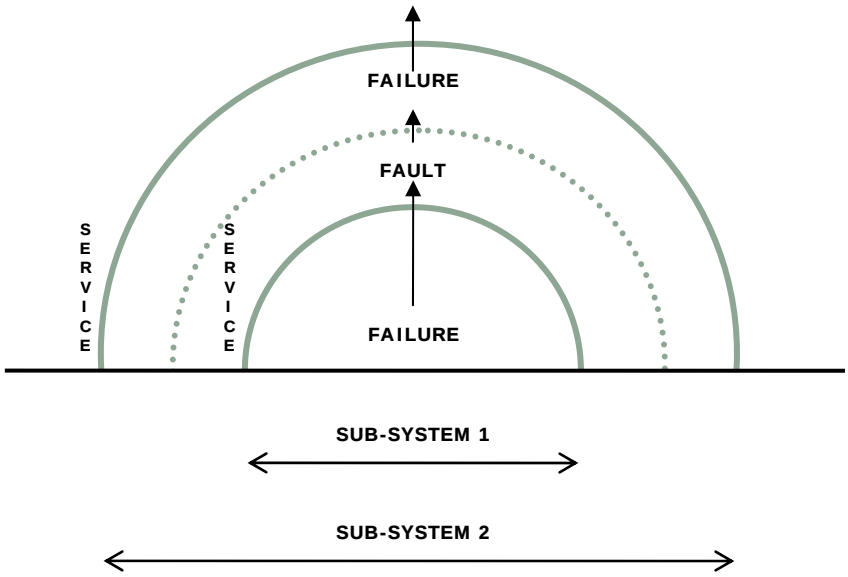

Figure A2 Failure propagation

A system with faults possibly can continue offering its service, that's mean that it is not on failure. A system with this characteristic is denominated fault tolerant. An analog explanation about the pathology of failures can be found in [ALR2001].

A.2. Fault classification

According to the behavior of the failure subsystems, a set of faults (fault class [GÄR1999]) could be identified [CRI91]: *omission, timing, response, crash* and *Byzantine.*

- Omission fault: It occurs when a system omits to respond to an input. Several things might go wrong. In the case of a receive omission fault, the system perhaps never got the request in the first place. Also, a receiver omission fault will generally not affect the current state of the system, as the system is unaware of any message sent to it.

Likewise, a send omission fault happens when the system has done its work, but somehow fails in sending a response. Note that, in contrast to a receive omission fault, the server may now be in a state reflecting that it has jus completed a service for a client. As a consequence, if the sending of its response fails, the system may need to be prepared that the client will reissue its previous request.

Other types of omission faults not related to communication may be caused by software itself such as infinite loops or improper memory management by which the system is said to "hang".

- Timing fault: It occurs when the system's response is functionally correct but untimely, i.e. the response occurs outside the real-time interval specified. Timing failures thus can be either early timing failures or late timing failures (performance failures).

- Response fault: It occurs when the system responds incorrectly: either the value of its output is incorrect (value failure) or the state transition that takes place is incorrect (state transition failure).

- Crash fault: If, after a first omission to produce output, a system omits to produce output to subsequent inputs until its restart, the system is said to suffer a crash fault. Depending on the system state at restart, one can distinguish between several kinds of crash fault behaviors:

    - Amnesia-crash: It occurs when the system restarts in a predefined initial state that does not depend on the inputs seen before the crash.

    - Partial-amnesia-crash: It occurs when, at restart, some part of the state is the same as before the crash while the rest of the state is reset to a predefined initial state.

    - Pause-crash: It occurs when a system restarts in the state it had before the crash.

    - Halting-crash: It occurs when a crashed system never restarts.

    Note that while crashes of stateless systems, pause-crashes and halting crash behaviors are subsets of omission faults behaviors; in general, partial and total amnesia crash behaviors are not a subset of omission fault behaviors.

- Arbitrary fault: It is the most serious fault class, also known as Byzantine faults [TV2002]. When arbitrary faults occur, clients should be prepared for the worst. In particular, it may happen that a system is producing output it should never have produced, but which cannot be detected as being incorrect.

    If you are wondering like me about why to use the term "Byzantine", the answer is that it refers to the Byzantine Empire, a time (330-1453) and place (the Balkans and modern Turkey) in which endless conspiracies, intrigue, and untruthfulness were alleged to be common in ruling circles.

Byzantine faults are closely related to crash faults. The definition of crash faults as presented above is the most benign way for a system to halt. They are also referred to as fail-stop faults. In effect, a fail-stop system will simply stop producing output in such a way that its halting can be detected by other systems (sub-systems).

A.3. Inclusion relation

In the presented classification there is an inclusion relation between the fault types [VPMG2004]. Figure A3 illustrates the inclusion relation.

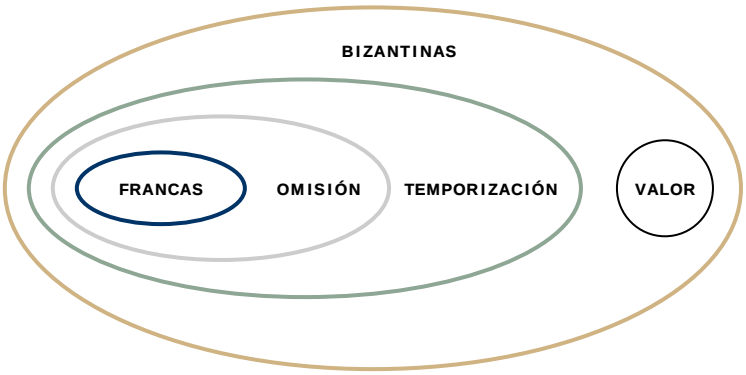

Figure A3 Fault type inclusion relation

For example, in the previous figure, a crash fault is considered as an omission fault when a system has a crash fault and it cannot respond to incoming messages. By consequence, it incurs in an omission fault too.

A.4. Fault tolerance forms

With respect to the perception degree that a user (human or other system) can have given a fault occurrence, four perception layers are defined [DQ P2002]:

- Nothing: The fault is ever perceived.
- Delaying: A degraded performance of the system is perceived, but it is able to recover a consistent state and recover is normal execution.
- Signaling: The fault is signal. The fault occurrence is indicated and the system execution stops.
- Chaos: The system behavior is deferent from its specification.

According with the influence that a fault has over a system properties, four cases are defined [GÄR1999] [DQP2002].

- Masking: The system properties are preserved in presence of a fault. There exists a real fault tolerance capacity.
- Unmasking: The system properties are altered at the time when the fault occurs, but when the occurrence finish, the system is able to recover a normal state that satisfies the properties again. There exists a real fault tolerance capacity.
- Signaling: Some properties of the system are altered at the time when the fault occurs and the system is capable to recognize this. The system signalizes the faults.
- Nothing: There no actions in front of the fault presence. The system properties are altered and there is not a fault tolerance capacity.

All this cases are named fault tolerance forms. Every form has different impact over the recovery strategy implementation cost. The most robust form is masking. The correspondence between the fault tolerance form and the fault tolerance perception layers is masking – nothing, unmasking – delaying, signaling – signaling, nothing – chaos.