Brief Announcement: Failure Detectors Encapsulate Fairness*

Scott M. Pike, Srikanth Sastry, and Jennifer L. Welch

Dept. of Computer Science and Engineering
Texas A&M University
College Station TX-77843 USA
{pike, sastry, welch}@cse.tamu.edu

Abstract. We argue that failure detectors encapsulate *fairness*. Fairness is a measure of the number of steps a process takes relative to another processes and/or messages in transit. As evidence we specify fairness-based message-passing system models that are the weakest to implement the failure detectors from [1].

Failure Detectors and Partial Synchrony. Failure detectors [1], system services that provide potentially unreliable information about process crashes in an otherwise asynchronous system, are believed to encapsulate partial synchrony [3]. That is, given a partially synchronous system model M, it is possible to replace M with an asynchronous system augmented with an appropriate failure detector $\mathcal{D}$ (that is implementable in M) such that all the problems solvable in M are also solvable in the asynchronous system augmented with $\mathcal{D}$. Hence, the pursuit to find the 'weakest' system models to implement various failure detectors.

Such pursuits have met with limited success, in part, because many such system models are specified with respect to real-time constraints on communication and computation. We argue that such 'weakest' system models should be specified as restrictions on 'fairness': a measure of the number of steps executed by a process relative to other events in the system.

Failure Detector Classes. We consider four failure detector classes from [1]:

- *Perfect failure detector:* (denoted $\mathcal{P}$) never suspects live processes and eventually suspects all crashed processes forever.
- *Strong failure detector* (denoted $\mathcal{S}$) never suspects *some* correct process and eventually suspects all crashed processes forever.
- *Eventually Perfect failure detector* (denoted $\Diamond\mathcal{P}$) eventually never suspects correct processes and suspects all crashed processes forever.
- *Eventually Strong failure detector* (denoted $\Diamond\mathcal{S}$) eventually never suspects some correct process and suspects all crashed processes forever.

* This work was supported in part by NSF grant 0964696 and Texas Higher Education Coordinating Board grant NHARP 000512-0130-2007

Fairness. We focus on fairness properties that refer to both the computation and the communication that takes place in executions. *Computational fairness* specifies the number of steps executed by processes relative to each other; *communicational fairness* specifies the number of steps executed by the recipient of a message while that message is in transit.

Computational Fairness. A process i is said to be *k-proc-distinguished* if all processes are *fair* with respect to i. Specifically, in each execution segment where a process j takes $k+1$ steps, then either (a) i takes at least one step, or (b) i is crashed. If all processes are fair with respect to i only after some unknown global stabilization time (GST), then i is said to be *eventually k-proc-distinguished*. Note that while other processes may be fair with respect to a proc-distinguished process i, process i need not be fair with respect to other processes; *i.e.*, a proc-distinguished process may take an unbounded number of steps in the duration between two consecutive steps by a non-proc-distinguished process. This is an important distinction between computational fairness and bounded relative process speeds defined in [3]. Bounded relative process speeds may be viewed as a special case where every process is (eventually) k-proc-distinguished.

Communicational Fairness. A process i is said to be *d-com-distinguished* if all other processes are *fair* with respect to messages from i. More precisely, for every message m sent by i to j, the recipient j takes no more than d steps while m is in transit and i is not crashed. Note that if i crashes while m is in transit to j, then the bound d does not apply. If all processes are fair with respect to messages from i only after some unknown GST, then i is said to be *eventually d-com-distinguished*. Like computational fairness, it is not necessary for a d-com-distinguished process i to be fair with respect to messages sent from non-com-distinguished processes to i. That is, i may take an unbounded number of steps while a message from a non-com-distinguished process is in transit to i.

Fairness-Based Partial Synchrony. We present four partially synchronous models that specify fairness properties rather than real-time behavior.

1. *All Fair* ($\mathcal{AF}$) system model is specified as follows: in every run, all processes are both k-proc-distinguished and d-com-distinguished, for known k and d.
2. *Some Fair* ($\mathcal{SF}$) system model is specified as follows: in every run, some correct process x is both k-proc-distinguished and d-com-distinguished, for known k and d.
3. *Eventually All Fair* ($\Diamond\mathcal{AF}$) system model is specified as follows: for each run, there exists a (potentially unknown) time after which the system behaves like $\mathcal{AF}$.
4. *Eventually Some Fair* ($\Diamond\mathcal{SF}$) system model is specified as follows: for each run, there exists a (potentially unknown) time after which the system behaves like $\mathcal{SF}$.

Weakest System Models. Our primary result states that $\mathcal{AF}$, $\mathcal{SF}$, $\Diamond\mathcal{AF}$, and $\Diamond\mathcal{SF}$ specify the exact amount of fairness encapsulated by $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, and $\Diamond\mathcal{S}$, respectively. In other words, $\mathcal{AF}$, $\mathcal{SF}$, $\Diamond\mathcal{AF}$, and $\Diamond\mathcal{SF}$ are the 'weakest' system models to respectively implement $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, and $\Diamond\mathcal{S}$ with any number of crashes.

We establish this result is as follows. First, we present a construction that uses a failure detector from [1] in an otherwise asynchronous system to schedule distributed applications such that each process executes its application steps fairly with respect to other processes (and messages). The fairness properties guaranteed by the scheduler depend on the available failure detector. By employing $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, or $\Diamond\mathcal{S}$, the scheduler provides fairness guarantees specified by $\mathcal{AF}$, $\mathcal{SF}$, $\Diamond\mathcal{AF}$, or $\Diamond\mathcal{SF}$, respectively. This shows that the failure detectors encapsulate at least as much fairness as is specified in the corresponding fairness-based system models. For the other direction, we present an algorithm which implements a failure detector on top of these fairness-based systems. When this algorithm is deployed in $\mathcal{AF}$, $\mathcal{SF}$, $\Diamond\mathcal{AF}$, or $\Diamond\mathcal{SF}$, it implements $\mathcal{P}$, $\mathcal{S}$, $\Diamond\mathcal{P}$, or $\Diamond\mathcal{S}$, respectively. Thus, we show that these failure detectors encapsulate no more guarantees on fairness than what is provided by the corresponding fairness-based systems.

Discussion. The notion of 'capturing the power' of a failure detector was explored in [5] for shared-memory systems. Our work, which focuses on message-passing systems, deviates from [5] in three significant ways. (a) The 'power' of a failure detector is different in shared-memory and message-passing systems. For example, in environments with arbitrary number of process crashes, the weakest failure detectors for consensus and k-set agreement are different in shared-memory and message-passing systems. Hence, 'capturing the power' of a failure detector in message-passing systems merits separate investigation. (b) Asynchronous systems under message passing are weaker than under shared memory because the quorum failure detector is necessary to implement read/write atomicity with message passing. Hence, the results from [5] need not carry over to message-passing systems. (c) We address the synchronism captured by perpetually accurate oracles (in addition to eventually accurate ones) and resolve the issue of *synchronous-systems-vs.-perfect-failure-detector* [2] whereas the results in [5] specify fairness constraints only for classes of eventually accurate oracles.

The full version of the paper can be found at [4].

References

1. Chandra, T.D., Toueg, S.: Unreliable failure detectors for reliable distributed systems. J. ACM 43(2), 225–267 (1996), http://dx.doi.org/10.1145/226643.226647
2. Charron-Bost, B., Guerraoui, R., Schiper, A.: Synchronous system and perfect failure detector: solvability and efficiency issues. In: International Conference on Dependable Systems and Networks. pp. 523–532 (2000), http://dx.doi.org/10.1109/ICDSN.2000.857585
3. Dwork, C., Lynch, N.A., Stockmeyer, L.: Consensus in the presence of partial synchrony. J. ACM 35(2), 288–323 (1988), http://dx.doi.org/10.1145/42282.42283
4. Pike, S.M., Sastry, S., Welch, J.L.: Failure detectors encapsulate fairness. Tech. Rep. 2010-7-1, Dept. of Computer Science and Engineering, Texas A&M University (2010), http://www.cse.tamu.edu/academics/tr/2010-7-1
5. Rajsbaum, S., Raynal, M., Travers, C.: The iterated restricted immediate snapshot model. In: 14th Annual International Conference on Computing and Combinatorics. pp. 487–497 (2008), http://dx.doi.org/10.1007/978-3-540-69733-6_48