# An Efficient Long-Lived Adaptive Collect Algorithm*

Burkhard Englert

California State University Long Beach, Dept. of Comp. Engr. & Comp. Science,
Long Beach, CA 90840
englert@cecs.csulb.edu

**Abstract.** We present a new long-lived, efficient, adaptive collect algorithm. Namely, our algorithm adapts to $\mathcal{K}$-contention - it has the property that if during an operation the interval contention $k$ exceeds a predetermined constant $\mathcal{K}$ the step complexity is $O(N)$. If, it falls below $\mathcal{K}$, the processors executions will eventually have adaptive step complexity of $O(k^3)$. Moreover, for $\mathcal{K}$ such that $\mathcal{K}^3 \leq N$ our algorithm requires only $O(N^2)$ shared memory registers.

## 1 Summary

To solve many well known problems such as atomic snapshot or renaming, it is essential that processors are able to gather information about each other. A simple way in which communication can be established is through the use of an array of Single-Writer Multi-Reader (SWMR) registers where each processor has a unique array entry assigned to it. Only a fixed processor is allowed to write to each array location while all processors can read all array entries. To update information about itself a processor writes into its array entry and to collect information about the other processors it reads all entries in an arbitrary order. Such a collect algorithm with step complexity $O(N)$, however, where $N$ is the total number of processors in the system, is possibly inefficient if only few of the $N$ processors are actually participating. This motivated researchers to look for *adaptive* algorithms whose step complexity only depends on the number of the concurrently participating processors.

Motivated by Lamport's MX algorithm [5], many adaptive algorithms have since been designed. Recently, a number of different adaptive collect algorithms were presented [1,2,3,4]. The algorithm by Attiya, Fouren and Gafni [3] for example, has an asymptotically optimal $O(k)$ step complexity, but it is a *one-shot* algorithm and the memory consumption is exponential in $N$. More recently Attiya, Kuhn, Wattenhofer and Wattenhofer [4] presented a new randomized adaptive collect algorithm with asymptotically optimal step complexity and polynomial memory overhead. For any constant $\gamma > 1$ they also presented a new deterministic collect algorithm with $O(k^2/((\gamma - 1)\log n))$ step complexity

* An extended abstract of this paper appears in the Proceedings of the DPNA 2005 (IEEE-ICPADS 2005) workshop.

P. Fraigniaud (Ed.): DISC 2005, LNCS 3724, pp. 516–518, 2005.

and $O(n^{\gamma+1})/(\gamma-1)\log n)$ memory complexity. However, their algorithms are *one-shot*, not long-lived and hence adapt only to total contention with respect to shared memory operations. On the other hand the collect algorithm by Afek, Stupp and Touitou [1] is long-lived and adapts to the point contention (and hence to interval contention) $k$. It is designed for low contention, has step complexity $O(k^3)$ and uses $O(N^3)$ shared memory registers. As a result, if the interval contention $k$ during the execution of a collect is high such that $k^3 >> N$, where $N$ is the number of processors in the system, their algorithm [1] is less efficient than a "straightforward" collect algorithm where processors read all $N$ SWMR registers in any order. Moreover, since such a "naive" algorithm only requires $O(N)$ shared memory cells but the long-lived adaptive collect algorithm by Afek, Stupp and Touitou [1] requires $O(N^3)$ shared memory registers, a significant memory overhead is encountered. In this sense it is very desirable to have an algorithm that itself can "adapt" to low or high (interval) contention.

We call such an algorithm *adaptive to $\mathcal{K}$-contention*. The constant $\mathcal{K}$ can be determined in advance. For example, if, as in [1], the adaptive step complexity is $O(k^3)$, we can let $\mathcal{K}$ be the largest integer such that $\mathcal{K}^3 < N$. Our new algorithm provides a mechanism to switch back and forth between the two approaches solely based on the interval contention a processor encounters.

Our paper makes the following contributions:

- We present a new long-lived adaptive collect algorithm. Previous algorithms [1] were designed for low contention and their performance suffers if the contention encountered exceeds $\mathcal{K}$ such that $\mathcal{K}^3 > N$. Our algorithm is *efficient* since it guarantees that a processor never needs to perform more than $O(N)$ steps in a *collect* operation. Moreover, our algorithm guarantees that if the interval contention $k$ of two successive operations by a processor $p$ is less than $\mathcal{K}$ and the contention of all other operations by other processors that occur or are active between these two operations is also less than $\mathcal{K}$, then the step complexity of the second operation by $p$ is $O(k^3)$.
- Our algorithm introduces a mechanism that allows processors to switch back and forth from operation to operation between reading the registers of all other processors and an adaptive execution.
- Our algorithm requires only $O(N^2)$ shared memory registers (instead of $O(N^3)$ [1]) thereby reducing the memory complexity overhead encountered by long-lived adaptive collect algorithms.

**Acknowledgement.** We are grateful to Eli Gafni for helpful discussions and comments.

## References

1. Y. Afek, G. Stupp and D. Touitou. Long-lived adaptive collect with applications. *Proc. of the 40th Ann. Symp. on Foundations of Computer Science*: 262-272, October 1999.

2. H. Attiya and A. Fouren. Algorithms adaptive to point contention. In *J. ACM*, 50(4): 444-468, July 2003.
3. H. Attiya, A. Fouren and E. Gafni. An adaptive collect algorithm with applications. *Distributed Computing*, 15(2): 87-96, 2002.
4. H. Attiya, F. Kuhn, M. Wattenhofer and R. Wattenhofer. Efficient Adaptive Collect using Randomization. *Proc. 18th Annual Conference on Distributed Computing (DISC)*, 2004.
5. L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1): 1-11. February 1987.