P.H.Welch and Fred Barnes

# Communicating Mobile Processes

## Abstract

This presentation develops the dynamic mechanisms introduced into occam that were reported at the last meeting of this group in Dagstuhl. The main focus is on mobile *processes*, as opposed to the mobile *channels* that were introduced last time. Language design and semantics for the mobile mechanisms of the new occam, christened occam-M for the moment, are motivated – before going on to consider safety, performance and some applications.

The formal semantics of our mobile processes (and mobile channels) do not follow the pi-calculus and its derivatives. Instead, we are working with Jim Woodcock to ensure that the ideas fit cleanly within a mobile-extended CSP framework – which means that the semantics remains compositional and denotational and that proof/development techniques, including the notion of refinement, continue to apply.

The safety and performance aspects are critical to the applications described, which require very large scale and highly dynamic networks of processes. The key safety guarantees of classical (but static) occam are preserved for systems using the mobile extensions. These include *total alias control* by compiler, which leads to zero aliasing accidents, zero race hazards, zero nil-pointer exceptions and zero garbage collection. Zero buffer overruns, of course, also remain guaranteed. Higher-level safety requirements – such as the absence of deadlocks, livelocks and process starvation – are not promised (as yet) by the language. However, a range of proven application design patterns, with widespread coverage, are on offer and do make those promises.

Performance overheads for all concurrency mechanisms are mostly unit time with the order of between 50 and 150 nanoseconds on moderate processors (e.g. an 800 MHz P3). The single exception is for waiting on multiple events – which is linear on the number of events when not much is happening, reducing to unit (100 nanosecond) time as the waiter comes under stress (i.e. when all the events start firing continuously).

Three application areas are discussed. The first is *communicating process operating system* design for both general purpose and (real-time) embedded – an outline of our *Raw Metal occam eXperience* (RMoX) project, which makes heavy use of the new dynamics, is given. Also presented are two modeling subjects: *bio-mechanisms* and *nanotechnology assemblies* (for building real artifacts and/or doing real jobs). These require vast numbers (>10 million, at least) of network elements (channels and processes), continually growing and decaying in response to both environmental and internal pressures as the modeled organisms/assemblies are born, combine, split and die. Design solutions to the problems of location (or, rather, neighbourhood) awareness are proposed – which are essential to allow mobile processes to find each other, interact and decide what to do next.

***URLs***:

| | |
|---|---|
| http://www.cs.ukc.ac.uk/projects/ofa/kroc/ | -- *official* KRoC occam |
| http://frmb.org/kroc.html | -- *latest* KRoC occam |
| http://wotug.ukc.ac.uk/ocweb/ | -- occam web server |
| http://www.cs.ukc.ac.uk/projects/ofa/jcsp/ | -- *core* JCSP |
| http://www.quickstone.com/ | -- JCSP Networking Edition (Java / J#) |
| http://www.wotug.org/ | -- user *community* |
| http://www.wotug.org/cpa2004/ | -- last CPA conference |