Telegraphos: High-Speed Communication Architecture for Parallel & Distributed Systems

Two key ingredients are needed to make high performance parallel and distributed computer systems: high-speed computation, and high-speed communication. The application of RISC principles to modern processor architecture has yielded high-speed computation engines. Our work is inspired by those same principles, in developing building blocks for high-speed communication – switches, and network-processor interfaces.

"Telegraphos" – from the Greek words τηλέ (remote) and γράφω (write) – is an R&D project to design architectures and build prototypes for such components. Telegraphos is funded by ESPRIT project 6253 "SHIPS." Our research focuses on congestion-tolerant networks, and on remote-write based network-processor interfaces. Our prototypes connect workstations within an office space, turning them into a parallel computer. Telegraphos I, our first prototype, uses low-end technology (FPGA) as a means for rapid prototyping, and runs at 133 Mbps/link. Systems using ASIC and full-custom CMOS, at about 400 and 1600 Mbps/link, as well as ATM switches for gigabit LAN's, are next on our schedule. Our remote-write based interfaces drastically cut down the software overhead for communication, making Telegraphos into something more than a workstation farm: a true parallel computer.

Point-to-Point Links with Back-Pressure
High-speed networks cannot afford multiple-access media any more – turn-around delays and arbitration complexity are excessive. Telegraphos uses switches interconnecting multiple point-to-point links, thus providing high throughput and increased parallelism. Preventive flow-control – ticket (window) based back-pressure – is used at the link level, so that packets never have to be dropped and retransmitted. This economizes on network bandwidth, and reduces the hardware complexity. Storage is less expensive than communication in modern IC technology, so enough buffer space is provided for peak throughput not to be compromised.

Congestion-Tolerance
Parallel programs may generate excessive and uncontrolled traffic loads at times; the network must behave robustly when that happens. Switch buffers that are shared among packets going in different directions degrade the network throughput when the load exceeds a certain limit. Telegraphos uses dedicated buffers per virtual path in each switch, and implements flow-control at that granularity, thus achieving good and stable throughput at increased network load.

Remote-Write: Efficient Communication Primitive
The central operation of the Telegraphos processor-network interface is the remote write: when the virtual address of a store instruction executed by the processor translates into a physical address in the Telegraphos device I/O space, the Telegraphos network interface sends it to the destination node (determined by the physical page number) in a network packet. The Telegraphos interface at that node performs the (local) memory write. In this way, the existing memory protection (address translation) mechanism is also used to implement message protection: a process *S* can only send messages to the mailboxes of those processes *R* for which the OS has given *S* an entry in its translation table; similarly, a process can only receive (read) the messages that were sent into pages that it has access to. As a consequence, message exchange can be done strictly *at user-level,* without need for any time-consuming system call. Additionally, this scheme excludes receive buffer overflows, and provides automatic message re-assembly from the multiple packets into which the message was broken, again simplifying the hardware.

Communication and Sharing in Parallel Systems
Besides low-overhead message passing, we are also working on low-cost support for shared memory. Telegraphos I includes hardware support for *eager sharing:* writes into a page are automatically reflected, in hardware, into all copies of the page that may exist on multiple nodes. Also, besides remote writes, we provide remote atomic operations and blocking and non-blocking remote reads. Thus, for each page, the compiler, the run-time environment, or the OS can choose local replication or remote accessing; in case of replication, a choice of coherence protocol is provided between write-invalidate and write-update. These choices – static or dynamic – are assisted by hardware counters, counting the number of local and the number of remote accesses per sharable page.

For further information, please contact
Dr. M. Katevenis, Dr. C. Nikolaou:
E-mail: {katevenis, nikolaou}@ics.forth.gr
Tel. +30 (81) 39 16 00, Fax +30 (81) 39 16 01
Institute of Computer Science,
Foundation for Research & Technology – Hellas
P.O.Box 1385, Heraklio, Crete, GR-711-10 Greece.