Operating systems support for the Desk Area Network (DANOS) GR/J11164
Final Report

Ian Leslie - University of Cambridge, UK.
Derek McAuley - University of Glasgow, UK.

July 24, 1996

Introduction

The Desk Area Network (DAN) is a workstation based on the use of an ATM switch fabric as the primary device interconnect. The machine was designed and implemented to be a multitasking workstation capable of supporting applications engaged in multimedia processing, capture and presentation in real time – so called second generation multimedia applications.

As such the interaction between the underlying hardware resources and the applications requires an operating system capable of:

- allocating the resources in response to application demand in the face of Quality of Service (QoS) constraints;
- providing access to the devices with high performance without compromising intertask protection;
- enable all components (devices, services, and applications) to effectively avoid or deal with overload;

These were the primary goals of the research undertaken.

The work was carried out in parallel with the Pegasus project which developed a multimedia operating system, known as Nemesis, for conventional workstation architectures.

Results

The output from the DANOS work is of three types:

1. influence on Nemesis
2. investigation of scheduling interconnect
3. novel I/O architectures and devices

It may seem slightly odd to list the influence on another Laboratory project (or set of projects) in advance of the direct, externally visible outputs of the project. However, Nemesis has a strong probability of having a major impact on operating system design in the future, and the DANOS work has contributed fundamentally to this.

Synchronisation architecture in Nemesis

A key influence on Nemesis has been the use of event counts and sequencers as the fundamental concurrency primitive. Event counts and sequencers are well known to be appropriate in multiprocessor situations, such as the DAN, but we know of no other system where these are used as the "native" primitives. The key observation is that they can be used without requiring a coherent or even shared memory implementation. This property means they provide a natural building block for a single bulk communication interface which can be used irrespective of whether the communication is between tasks, processors, devices or complete systems [3].

Furthermore, their use in Nemesis has led to a number of observations. Event counts and sequencers are very effective and efficient for decomposing device drivers into interrupt handlers of a few instructions (turn off the interrupt, increment an event count) and device handler "processes" running in "user space". This allows the software concerned with processing the IO transaction to be under the control of the operating system scheduler and to restrict where necessary access (primarily) to processor resource should the QoS demanded by other operating system tasks demand it.

Finally, this is achieved without loss of generality – a number of other concurrency primitives (semaphores, POSIX threads and SRC threads[1]) have been built over event counts and sequencers. This exercise has been much simpler than building over any other concurrency primitives that we have used before.

Interconnect Scheduling

Two types of investigation of interconnect scheduling were performed. The first was to support constant bandwidth streams across the ATM fabric by deterministic cell scheduling. This took advantage of the simple priority scheme within the fabric. Some traffic was determined to be constant rate and allocated slots within the schedule, very much as in the DEC Gigaswitch [2]. These slots were sent across the fabric at higher priority than non constant rate cells. If a slot allocated for constant rate did not have a customer cell, a non constant rate cell could be sent at a lower priority. The schedule for the constant rate traffic is alway contention free; non constant rate traffic can be disrupted either by higher priority constant rate or by equal priority non constant rate.

This is a clear example of separating the control path (the setting up of the schedule) from the data path (the use of the schedule), with the control path being executed only when changes in channel conditions were required.

The general principles were proven by this work, but the low level synchronisation required was extremely processor intensive. The processing elements had not been designed with this operation in mind, and the work showed that some low level support for this type of operation would have been beneficial.

The second investigation was concerned with scheduling at a time scale more appropriate to the scheduler in the operating system (milliseconds) rather than at the time scale of ATM cells. This produced a system in which capacity across the fabric was given away on a coarser time scale. (There are particular advantages to this scheme when two sources wish to send to the same destination since it avoids interleaving communication.) Because of the timescales, it was straightforward to integrate this scheme into the scheduling system of the operating system.

[1] the threads library designed and implemented at Digital Equipment Corporations Systems Research Laboratory
[2] See DEC Systems Research Center Research Report No 99.

IO architecture

It is in the area of I/O architectures that the project has made the greatest contribution. In the desk area network I/O devices are more autonomous than in traditional systems – indeed this trend can be seen in many new devices for PCs where latency constraints require a device to handle time critical actions without reliance on availability of the main processor resource.

In principle DAN devices can communicate directly with the network without the knowledge of processors in the system. This facility is provided by the concept of a channel through the DAN which provides a mechanism for controlling the use of devices. A channel maps very naturally onto an ATM virtual circuit and its associated resource reservation. Channels can be set up "out-of-band" by the operating system, and parametrised to ensure that they have sufficient resources for their communications needs over the DAN interconnect. Once a channel has been set up applications can communicate directly to devices without the intervention of the operating system, once a channel to a device is established.

These observations, coupled with the scheduling of the interconnection resource described above, leads to system in which I/O streams can have quality of service guarantees associated with them. These guarantees are closely backed by real resource. This includes both the communication bandwidth to the I/O device and the resources within the I/O device (which can again include a component of bandwidth).

This has led to a concept of *User Safe Device* in which devices have a small additional protection facility to ensure that a user is conforming with the restrictions established when the channel was set up. This can be viewed as taking the communication network notion of *policing* and applying it in a general sense inside the device. In communication networks, a policing function ensures that a source does not violate its contract (usually simply a resource consumption contract) by actively inspecting the information sent by the source. Here the policing function includes both resource control (bandwidth, buffer space) and security reasons, such as access control. For example the restriction might be as simple as deciding whether the user has access to a pixel they are attempting to write in a frame buffer.

Such a solution has a cost; however, as with the addition of virtual address translation hardware for virtual memory and addressing, the added benefits in the way of protection, performance and accountability for QoS support make the cost worth paying – not least as our experience from the examples developed (frame buffer, ATM network interface and disc) the additional cost was small.

This work has fed back into the Nemesis development where the concept of device, even in a traditional workstation, follows that of devices in the DAN. The functionality of policing (but no other) is provided by the device driver handling the data path from application to device.

A window system for the DAN frame store was developed as a final year undergraduate project. (The student came top of the final year.) This was a client rendering system in which the application (client) runs the bulk of the window system and communicates directly with the frame store. The API seen by the application programmer is a large subset of the Xlib API.

Resource model

The user safe device is an implementation strategy for a policing mechanism to ensure that high performance direct device access can be achieved with protection and in the face of QoS constraints.

What is policed? A key result of the work in this area has been the identification of appropriate divisions in operating system services into in-band and out-of-band components where the division if performed with the goal of making the in-band components simple low level resources which can be provided with a straight forward policing mechanism. Deciding on the particular split is dependent on the particular device characteristics.

For example, in the case of the frame buffer, two low level resources are identified for in-band policing: pixel access rights and frame buffer "bandwidth". Both are easily policed with modest additional hardware and device driver overhead, whereas the respective out-of-band components can be quite complex: in the first instance the policy is based on Window Manager preferences and user input; in the second by the quality of service manager which must also interact processor scheduler appropriately.

Similar decompositions has been applied to interconnect bandwidth, network interface, video/audio capture device (AVA-200) and disc.

Output

Papers and Internal Reports

The Nemesis operating system (in actual fact a precursor to Nemesis) implementation on the Desk Area Network is described in [3].

Some early results of considerations of the organisation of devices in the DAN are described in [2].

Early work on the scheduling of the interconnection in the DAN is described in [5]. This was followed up by a more detailed study and implementation which is described in [4].

The concepts of device organisation are fully detailed in two PhD dissertations, one complete [1] and one nearing completion [6]. A final year undergraduate project made use of this device organisation to construct a client rendering window system on the DAN [7].

Results of the DANOS work will be incorporated in a publication which will present the entire Nemesis system, including the I/O architecture. This publication is in the planning stages.

Exploitation

The work on resource sharing for the AVA-200 has been transferred to Nemesys Research Ltd for commercial exploitation.

Other exploitation is through the Nemesis OS, but the DAN version of Nemesis is also closely related to the ARM version of Nemesis. Direct exploitation of this work is under negotiation.

Exploitation of other aspects of the work related to device architecture is under consideration.

Follow on

Work continues on the operating system work in the EU funded Pegasus-II; DAN architecture and OS implications are already funded by Sun Microsystems in the "DeskNet" project; more effective policing and feedback mechanisms for controlling cache and interconnect bandwidth will form the basis of an upcoming research proposal.

Conclusion

The DANOS project had a major impact on the Nemesis operating system design. The approach based on semi-autonomous protected devices with an appropriate split between in-band and out-of-band functionality has demonstrated that complex QoS policies derived from applications demands can be enforced with comparatively simple extensions to the hardware and device architectures. There remain fundamental questions as to where functionality should be provided; the DAN and traditional workstation architectures provide suitable platforms for making many comparisons.

The use of the DAN with it's inherent latencies has results in an architecture that is extensible to the domain of networked devices; this has had the unforeseen result that the work is finding application in the development of appropriate software architectures for "Set Top Box" and "Network Computers" systems.

Publications

[1] Barham, P. *Devices in a Multi-Service Operating System.* PhD thesis, University of Cambridge Computer Laboratory, July 1996.

[2] Barham, P., Hayter, M., McAuley, D., and Pratt, I. Devices on the Desk Area Network. *IEEE Journal on Selected Areas In Communications 13*, 4 (May 1995).

[3] Black, R. Explicit Network Scheduling. Tech. Rep. 361, University of Cambridge Computer Laboratory, December 1994. Ph.D. Dissertation.

[4] Donnelly, A. Scheduling the ATM fabric. Final Year Project, May 1996.

[5] Khan, N. Cell scheduling to support CBR traffic. in ATM Document Collection (Green Book), October 1995.

[6] Pratt, I. *User Safe Devices.* PhD thesis, University of Cambridge Computer Laboratory, Sept(?) 1996.

[7] Stratford, N. A window system for the DAN frame store. Final Year Project, May 1996.