(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 December 2003 (31.12.2003)

PCT

(10) International Publication Number
WO 2004/001615 A1

(51) International Patent Classification⁷: G06F 13/10

(21) International Application Number: PCT/SE2002/001225

(22) International Filing Date: 19 June 2002 (19.06.2002)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant (*for all designated States except US*): **TELEFONAKTIEBOLAGET LM ERICSSON** [SE/SE]; S-126 25 Stockholm (SE).

(72) Inventor; and
(75) Inventor/Applicant (*for US only*): **ANDJELIC, Mario** [SE/SE]; KransbindarväGEN 41, S-126 36 Hägersten (HR).

(74) Agents: **HEDMAN, Anders Aros Patent AB** et al.; Box 1544, S-751 45 Uppsala (SE).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

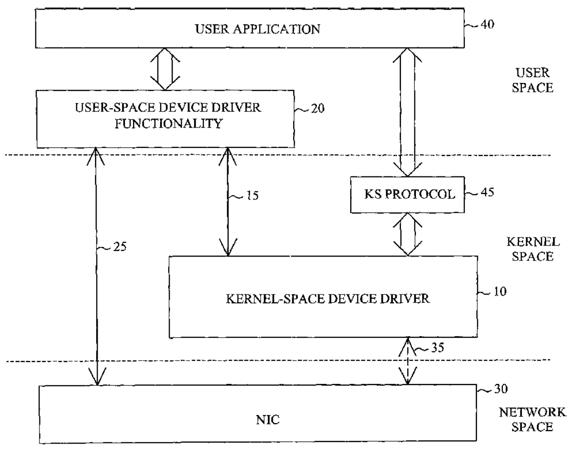
Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2004/001615 A1

(54) Title: A NETWORK DEVICE DRIVER ARCHITECTURE



(57) Abstract: The invention proposes a network device driver architecture with functionality distributed between kernel space and user space. The overall network device driver comprises a kernel-space device driver (10) and user-space device driver functionality (20). The kernel-space device driver (10) is adapted for enabling access to the user- space device driver functionality (20) via a kernel-space-user-space interface (15). The user-space device driver functionality (20) is adapted for enabling direct access between user space and the NIC (30) via a user-space-NIC interface (25), and also adapted for interconnecting the kernel-space-user-space interface (15) and the user- space-NIC interface (25) to provide integrated kernel-space access and user-space access to the NIC (30). The user-space device driver functionality (20) provides direct, zero-copy user-space access to the NIC, whereas information to be transferred between kernel space and the NIC will be "tunneled" through user space by combined use of the kernel-space device driver (10), the user-space device driver functionality (20) and the two associated interfaces (15,25).

A NETWORK DEVICE DRIVER ARCHITECTURE

TECHNICAL FIELD OF THE INVENTION

The present invention generally relates to a network device driver architecture for efficient and flexible access to a network interface controller (NIC).

BACKGROUND OF THE INVENTION

Computer software can generally be divided into two types, operating system software and application software. The operating system (OS) can be viewed as a resource manager that makes the computer's resources such as processors, memory, input/output (I/O) devices and communication devices available to the users. It also provides the base functionality upon which application software can be written and executed. Important operating system functions include sharing hardware among users, preventing users from interfering with each other, resource scheduling, organizing data for secure and rapid access, and supporting I/O functions and network communications.

The central part of the OS is commonly referred to as the kernel. The kernel is normally only a portion of the code of what is commonly thought of as the entire OS, but it is one of the most intensively used portions of the code. The kernel defines the so-called user-space, in which the application software runs, and provides services to user applications, including memory management, allocating processing resources, and responding to system calls from user applications or processes. Other important kernel functions include interrupt handling, process management and synchronization, as well as I/O management including network communications.

Since many different hardware devices can be connected to the computer system, some of the I/O functionality is typically implemented as common functionality that is device independent. Device related functionality is then allocated within so-called device drivers. This means that a user application that needs to access a particular hardware device, such as a network communication device, makes a system call to the OS, which in turn invokes the device driver associated with the hardware device.

A Network Interface Controller (NIC) is a hardware device that is commonly connected to computer systems for providing network communication capabilities, such as Ethernet or ATM communication. NIC controllers usually implement lower-level protocols, such as layer 1 (PHY) and layer 2 (MAC, LLC) protocols, whereas higher level protocols (e.g. the TCP/IP protocol suite) traditionally are allocated in the OS, running in kernel mode. Moreover, clusters, for example, usually have proprietary protocols running on top of Ethernet because TCP/IP (Transport Communication Protocol/Internet Protocol) is not very well suited for cluster computing in System Area Networks (SANs). These proprietary protocols are generally also running in kernel mode.

However, centralized in-kernel protocol processing prevents user applications from realizing the potential raw performance offered by the underlying high-speed networks. The performance problem is mainly caused by message copying between user space and kernel space, polluted cache, interrupts and non-optimized code. The intensive message copying creates a large overhead, especially for short messages, and constitutes the main reason for high processor load and low throughput of network subsystems with standard operating systems.

This problem has become more pronounced with the advent of high-performance network communication technologies such as Gigabit Ethernet, ATM and Infiniband. The main challenge in putting such high-performance communication technologies into use lies primarily in building systems that can efficiently interface these network media and sustain high bandwidth all the way between two network communicating applications.

This has lead the computer industry to develop network device drivers that support NIC access directly from user space, avoiding message copying between user space and kernel space. The most commonly known example of this type of user-space network access architecture is the Virtual Interface Architecture (VIA) developed by Intel Corporation, Microsoft Corporation and Compaq Computer Corporation. The Virtual Interface Architecture (VIA) is an industry standard for System Area Networks that supports direct, zero-copy user-space access to the NIC. The VIA Architecture was designed to eliminate message copying, per-message interrupts and other kernel overhead that have made traditional networked applications become performance bottlenecks in the past. As described, e.g. in the specification *Intel Virtual Interface (VI) Architecture Developer's Guide*, September 9, 1998 and the International Patent Application WO 00/41358, the VIA Architecture avoids intermediate data copies and by-passes the operating system kernel to achieve low latency, high bandwidth communication. The VIA model includes a VI consumer and a VI provider. The VI consumer typically includes a user application and an operating systems communication facility and a VI user agent. The VI provider typically includes the combination of a VI NIC and a VI kernel agent. The Virtual Interface (VI) is a direct interface between a VI NIC and a user application or process. The VI allows the NIC to directly access the user application's memory for data transfer operations between the application and the network. The VI generally comprises a send queue and a receive queue, each of which can be mapped directly to user address space, thus giving direct user-space access to the network level and by-passing the operating system kernel.

The technical report *DART - A Low Overhead ATM Network Interface Chip*, TR-96-18, July 1996 discloses an ATM NIC designed for high bandwidth, low overhead communication, by providing direct, protected application access to/from the network.

The main drawback of the VIA architecture (and similar architectures) is that it requires special VIA-enabled NIC controllers, and can not run on off-the-shelf NIC controllers such as ordinary Ethernet NIC controllers. Since a lot of functionality for network communication rely on kernel-level protocols such as TCP/IP, both a VIA-enabled NIC and an ordinary Ethernet (TCP/IP) NIC are required with the VIA architecture. The VIA architecture is thus not optimized for implementation into existing systems, but generally requires hardware re-design of existing systems, adding an extra NIC and/or NIC port to the system. Re-designing a circuit board, including design, testing, product handling, maintenance, spare parts, etc. may easily lead to extra costs in the order of millions of dollars.

SUMMARY OF THE INVENTION

The present invention overcomes these and other drawbacks of the prior art arrangements.

It is a general object of the present invention to provide efficient and flexible access to a network interface controller (NIC), eliminating the CPU as the bottleneck in the communication chain.

It is also an object of the invention to provide an improved and cost-optimized network device driver architecture. In particular, it is beneficial if the network device driver architecture is suitable for implementation and integration into existing systems.

Yet another object of the invention is to provide a robust and flexible network device driver that is not NIC dependent and works with any off-the-shelf NIC hardware.
These and other objects are met by the invention as defined by the accompanying patent claims.

The general idea of invention is to provide an efficient, flexible and cost-effective network device driver architecture by means of integrated kernel-space access and user-space access to the NIC, preferably over the same NIC port. This is accomplished by enabling direct user-space access to the NIC, in similarity to user-space network

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.