The DZERO Level 3 Data Acquisition System

R. Angstadt, G. Brooijmans, D. Chapin, M. Clements, D. Cutts, A. Haas, R. Hauser, M. Johnson, A. Kulyavtsev, S. E. K. Mattingly, M. Mulders, P. Padley, D. Petravick, R. Rechenmacher, S. Snyder, and G. Watts

***Abstract*—The DZERO experiment began RunII datataking operation at Fermilab in spring 2001. The physics program of the experiment requires the Level 3 data acquisition (DAQ) system system to handle average event sizes of 250 kilobytes at a rate of 1 kHz. The system routes and transfers event fragments of approximately 1–20 kilobytes from 63 VME crate sources to any of approximately 100 processing nodes. It is built upon a Cisco 6509 Ethernet switch, standard PCs, and commodity VME single board computers (SBCs). The system has been in full operation since spring 2002.**

***Index Terms*—Computer networks, data acquisition, monitoring, physics, triggering.**

I. Introduction

The DZERO experiment [1] is one of two experiments studying high energy proton-antiproton collisions at the Fermi National Accelerator Laboratory's Tevatron collider. Collisions occur at the interaction point inside the DZERO detector with a maximum rate of 2.5 MHz. On average, only 50 of the most interesting events per second are written to tape storage for later analysis. The reduction in rate is realized by a three-level trigger system, which evolved from the original RunI DZERO trigger and data aquistion sytem [2]. Systems similar in general design are in operation by the ZEUS [3] and CDF [4], [5] experiments. The first level (L1) trigger, built from custom high speed electronics, demands simple criteria to be satisfied from a combination of detector subsystems and has an output rate of about 5 kHz. The second level (L2) trigger, built from a combination of custom electronics and generic processors, combines information from detector subsystems and applies more complicated criteria to reduce the rate to 1 kHz. The third level (L3) trigger, a commodity processor farm, combines and reconstructs the full data for each event and selects approximately 50 Hz to be written to tape storage.

The Level 3 data acquisition system (L3DAQ) transports fully digitized data from detector subsystems to L3 trigger filter processes running on the L3 trigger farm. After a L2 trigger accept, up to 63 VME crates must be read out, each containing 1 to 20 kB of data distributed among VME modules. The total average event size, summed over all readout crates, is approximately 250 kB under normal physics running conditions. Under special running conditions (calibration), individual crate sizes can be as large as 256 kB and the total event size can be as large as 2 MB. A single board computer (SBC) in each crate reads out the VME modules and sends the event fragment over an Ethernet network to a L3 trigger farm node, specified by routing instructions received from the Routing Master (RM) process as shown in Fig. 1. The RM chooses a farm node for each event based on the L2 triggers that fired, the number of available buffers in each farm node, and the run partition information. Event fragments are built into complete events on the farm nodes and processed by L3 trigger processes. Events which pass L3 trigger criteria are sent via a separate network for eventual storage to tape.

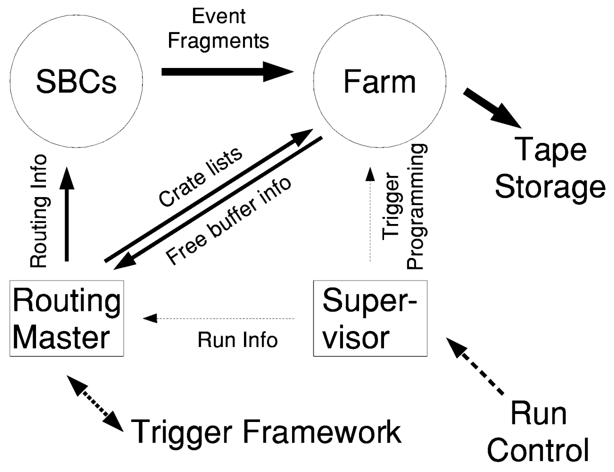

Fig. 1. Communication and data flow within the L3DAQ system.

II. Hardware Components and Interfaces

The L3DAQ system is built from commodity hardware components: single board VME computers, Ethernet switches, and PCs.

A. SBCs

The SBCs that are used in the system are VMIC [6] model number 7750. The VMIC-7750 is a 6U format VME board with a Tundra UniverseII [7] PCI-to-VME interface capable of DMA transfers from VME to physical memory. It contains a single 933-MHz Intel Pentium III processor, 128 MB flash disk, 128 MB of RAM, and two 100-Mb Ethernet interfaces.

The VMIC-7750 has a single PCI Mezzanine (PMC) slot, which has been populated with a third-party 64-channel TTL digital I/O (DIO) board: model PMCDIO64 from BVM [8]. The

Manuscript received June 9, 2003; revised January 19, 2004. This work was supported by the Department of Energy and National Science Foundation.

R. Angstadt, G. Brooijmans, M. Johnson, A. Kulyavtsev, M. Mulders, D. Petravick, and R. Rechenmacher are with Fermi National Accelerator Laboratory, Batavia, IL 60510 USA.

D. Chapin, M. Clements, D. Cutts, and S. E. K. Mattingly are with Brown University, Providence, RI 02912 USA (e-mail: chapin@hep.brown.edu).

A. Haas and G. Watts are with the University of Washington, Seattle, WA 98105 USA.

R. Hauser is with Michigan State University, Easst Lansing, MI 48824 USA.

P. Padley is with Rice University, Houstaon, TX 77251 USA.

S. Snyder is with Brookhaven National Laboratory, Upton, NY 11973 USA.

Digital Object Identifier 10.1109/TNS.2004.828785

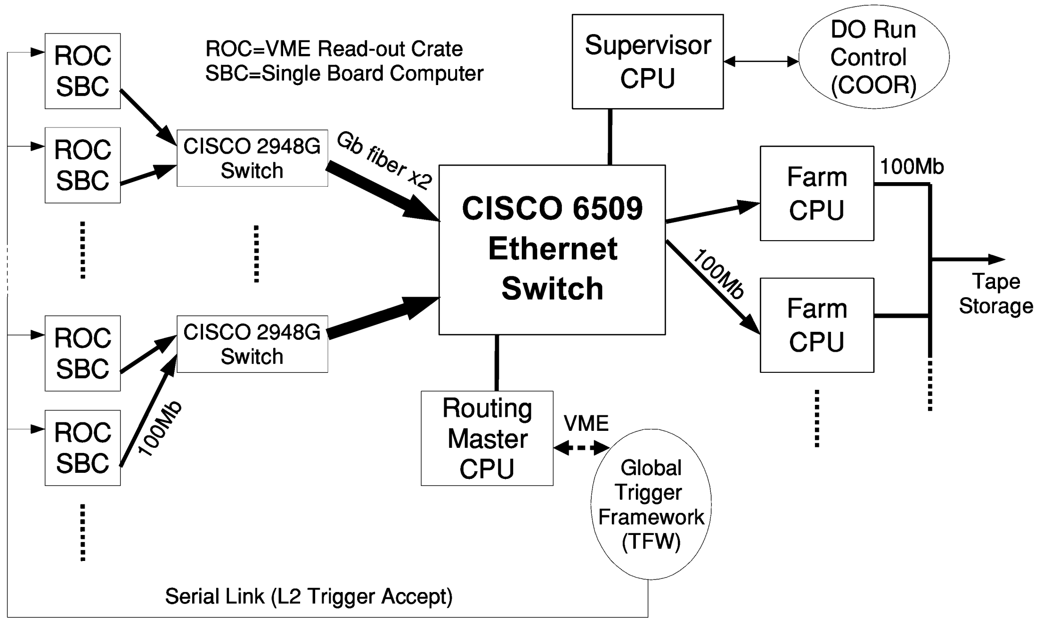

Fig. 2. The physical network configuration of the L3DAQ system.

DIO board is capable of generating PCI interrupts, triggered by a change-of-state on any of its inputs.

A custom extender board has been built to support the 6U VMIC-7750 in the 9U VME crate format common to DZERO readout crates. The SBC fits in the top-rear section of the resulting 9U card such that the SBCs VME connectors make direct contact with VME backplane: standard VME signals are not propagated on the extender board. The extender board propagates Ethernet, VGA, serial, and keyboard connector signals to the front panel of the extender. It also propagates two control signals from the DIO board to the VME user (J3) backplane and 24 outputs from the DIO board to several LED packs on the front panel of the extender.

B. Farm Nodes

The Level 3 Trigger processing nodes (farm nodes) are commodity, dual-processor, rack-mount PCs with dual 100 Mb Ethernet interfaces. There are currently 82 farm nodes: 48 with dual 1-GHz Pentium III processors and 34 with dual AMD Athlon 2000 processors. One Ethernet interface on each farm node connects to the main L3DAQ Ethernet switch described in Section II-C. The second interface connects to a separate network used to transport accepted event data to eventual tape storage.

C. Network Components

The network is built around a single CISCO [9] 6509 Ethernet switch, as shown in Fig. 2. The switch contains two 48-port modules, each with 100-Mb Ethernet ports and 1 MB of output buffer memory per port. It also contains one 16-port module with Gb fiber connections. The farm nodes are connected directly to the switch via the 100 Mb ports.

Groups of up to 20 SBC Ethernet ports are first connected to Cisco 2948G switches. Four of the five 2948G switches are connected to the 6509 switch over dual Gb fiber links that are aggregated using Cisco's EtherChannel(tm) technology. The fifth switch is lightly populated and requires only a single Gb link. A single VLAN is in use for all switches.

D. Trigger Framework Interface

The DZERO global trigger framework (TFW), part of the DZERO trigger system, makes the final L1 and L2 trigger decisions. There is a one-to-one association between Level 1 and Level 2 triggers. Up to 128 individual L1/L2 triggers can be defined within the TFW system. Per event L2 trigger accept/reject signals, along with a 16-bit event identifier (L3 transfer number), are sent by the TFW to readout controllers (Section II-E) in the VME readout crates via dedicated serial links.

The TFW also transports the 128 individual trigger decisions and the L3 transfer number to the L3DAQ system for each event accepted by the L2 trigger. A hardware FIFO containing a 128-bit mask of trigger decisions and the 16 bit L3 transfer number is made available through a group of VME boards for this purpose. The FIFO depth is adjustable and currently set to 100. A 128-bit disable register is also made available to the L3DAQ system through these VME boards. For any bit set in the disable register, the TFW will forcibly reject subsequent L1 triggers corresponding to that bit.

E. VME Readout Crate Interface

Each VME readout crate provides a common interface with the L3DAQ SBC. Data to be readout are made available via VME. Additionally, a subsystem-specific controller card in each crate receives the L3 transfer number from the TFW over a dedicated serial link and makes it available via VME. The controller card informs the SBC that data are ready to be read out by asserting a single line (SLVRDY) on the VME user backplane.

Upon completion of the VME data transfer, the SBC informs the controller card by asserting another line (DONE). The controller card and SBC then complete this handshake sequence by deasserting SLVRDY and DONE in order, respectively.

III. Software Component Operation

The L3DAQ is primarily a software based system. The farm nodes and SBCs are commodity processor platforms and run the Linux 2.4.X operating system. A custom Linux kernel driver written in C controls the VME readout and event fragment buffering in the SBCs. The processes that control event data flow (the Routing Master, the Event Builder on the farm nodes, and the Event Sender on the SBCs) are written in C++ and make significant use of the open, multiplatform ACE [10] networking and utility library. TCP/IP sockets form the communication links between all L3DAQ software components.

The robust features of the TCP protocol greatly facilitated software development and the design of the system. The TCP implementation in the Linux kernel handles data flow control (Section IV) and packet-loss recovery.

A. Supervisor

The primary interface used by shift personnel to control data taking of the DZERO experiment is the main run control software (COOR). It is the responsibility of COOR to respond to human-requested run transitions and disseminate run configuration information to the subcomponents of the DZERO data acquisition system, including the L3DAQ. The DZERO data acquisition system is designed to handle multiple simultaneous *runs*, with events possibly shared by more than one run. This feature is typically used to partition the detector, in order to allow concurrent detector testing and calibration during nonphysics running.

For each run, COOR defines one or more *routing groups* for the L3DAQ system, where a routing group is an exclusive set of L2 trigger bits associated with a set of readout crates and a set of farm nodes. Typically there is a one-to-one mapping of runs to routing groups. As discussed in Section III-B, when any associated L2 trigger fires for a particular routing group, the associated readout crates are read out and the event data are sent to one of the nodes in the associated set of farm nodes.

The Supervisor process is the interface between the L3DAQ system and COOR. When a new run is configured, COOR sends to the Supervisor via a TCP/IP connection the initial details for each routing group (a list of L2 trigger bits and a list of readout crates) associated with the run and the Level 3 trigger configuration for the run. The Supervisor allocates a set of farm nodes for the run and sends the L3 trigger configuration to the Event Builder process (see Section III-E) on those farm nodes. The Supervisor then sends the list of L2 trigger bits, list of readout crates, and list of farm nodes for each routing group to the Routing Master process.

If the Routing Master or a farm node restarts during an active run, the presence of a new connection causes the Supervisor to automatically refresh its configuration. However, any in-progress events are lost. The Supervisor can also reassign farm nodes from one active run to another.

B. Routing Master (RM)

The RM process runs on an SBC located in a VME crate that contains the Trigger Framework Interface VME boards, described in Section II-D. The Routing Master has two purposes. The first is to direct the sending of event fragments from the SBCs to the farm nodes. The second is to inform the Trigger Framework to stop issuing trigger accepts when the L3 trigger farm nodes are unable to process events immediately.

The RM polls the Trigger Framework Interface FIFO every 10 ms over VME and retrieves the L2 trigger bitmask and L3 transfer number for an accepted event. For each routing group (Section III-A) that overlaps with at least one of the passed L2 triggers, the RM chooses a farm node to which the event fragments should be sent. The decision is based on the the number of free buffers in each farm node in the set of farm nodes assigned to the routing group. The RM maintains an internal table of the number of free buffers on each farm node. Each time an event is routed to a farm node the corresponding entry in the table is decremented by one. The table entries are updated periodically when the RM receives messages from the farm nodes indicating the number of free buffers available. A farm node is chosen in a round-robin fashion from amongst the set of farm nodes with the most free buffers.

A set of readout crates that must be read out for each routing group is generated by forming the union of all the known trigger lists with the L2 triggers that passed for the event. The RM creates a Route Tag for each SBC in the expected set of readout crates. The Route Tag contains the L3 transfer number for the event and a set of farm node indices representing the farm nodes to which the SBC should send its event fragment. When all routing groups have been considered for an event, the Route Tags are added, based on their destination SBC, to routing bundles, which group together multiple Route Tags into a single message. This reduces the number of Ethernet packets that the RM must send out. A routing bundle containing a nonzero number of Route Tags is sent to the appropriate SBC either after 250 ms or when it contains some maximum number of Route Tags (currently 10). The 250 ms timeout is in place to reduce the unnecessary latency that would be created by the bundling when running at event rates of <40 Hz, and also to ensure that partially full bundles are sent with modest latency if triggers are stopped due to a run pause or stop or a backup elsewhere in the system. A small amount of randomization is also added to ensure that routing bundles for all SBCs are not sent after the same event. After each event, each partially full routing bundle has a 0.1% chance of being sent. Quite quickly, the number of Route Blocks in each routing bundle is distributed flatly between zero and the routing bundle maximum. Allowing the SBCs to send their bundled data to the farm nodes at independent times smooths the flow of data in the network switches, thus requiring a smaller instantaneous switch backplane data rate, as well as reducing the chances of buffers in the data chain filling up.

The RM also sends information to each chosen farm node for each event: the L3 transfer number, the crate list, and the list of passed L2 triggers. The L3 transfer number and crate list are used by the Event Builder process on the farm node to determine when all event fragments have been received. The trigger list is used for accounting purposes.

If the number of free buffers available among the farm nodes in a run reaches some minimal value (currently 16), the RM disables all the L1 triggers associated with that run by setting the appropriate bits in the disable register of the Trigger Framework Interface. When additional free buffers are available, the RM reenables the appropriate triggers. The threshold value of 16 takes into account the possibility that a finite number of L2 accepts may be issued by the TFW even after the L1 triggers have been disabled.

C. SBC Driver Module (DM)

The SBC DM is a Linux kernel module that services interrupt requests from the DIO board and UniverseII chip, initiates VME transfers, manages a pool of event fragment buffers, and provides an interface to the buffered event fragments for the user-level Event Sender process. For VME access the Driver Module uses routines in a a vendor-supplied kernel module for the Universe II chip. The DM makes use of the features of the bigphysarea Linux kernel patch to allocate large contiguous blocks of kernel-space memory.

Upon initialization, the DM maps a small, 2-kB section of memory to a prespecified VME address, to which the readout crate controller writes a VME address that contains the L3 transfer number, a list of the VME addresses (DATAPOINTERS) that must be read out for each event, and a list of VME addresses (WCPOINTERS) that contain the number of bytes that must be read out from each DATAPOINTER.

An interrupt service routine (DIO ISR) in the Driver Module is called when the DIO card generates a PCI interrupt, triggered by a change on the SLVRDY line on the VME user backplane (see Section II-E). If the SLVRDY line has been asserted, the DM first reads the L3 transfer number address and reads the WCPOINTER addresses to obtain the number of bytes that must be read out of each of the VME modules. The DM then initiates a DMA transfer of the readout crate data from the specified DATAPOINTER addresses into one of several (currently 50) buffers in kernel-space memory. Another interrupt service routine in the DM is called when the Universe II completes the data transfer. This routine asserts the DONE line via the DIO card. The DONE line is deasserted by the DIO ISR when the state of the SLVRDY line changes to deasserted. If no free buffers are available, the DM module will wait for a buffer to become free before starting the VME data transfer.

D. Event Sender

The Event Sender is the user-level process that runs on the readout crate SBCs. It receives routing information in the form of Route Tags (in routing bundles) from the Routing Master, and it attempts to match, by L3 transfer number, Route Tags with event fragments held by the SBC Driver Module. Both the Route Tags and the event fragments are held in queues. The L3 transfer number of the head Route Tag is compared with that of the head event fragment. In the case of a match, the event fragment is sent to the farm node(s) specified in the Route Tag. In the case of a L3 transfer number mismatch, either the Route Tag or the event fragment is discarded depending on which number was behind. By discarding fragments or Route Tags, the Event Sender resynchronizes itself when routing or event data is occasionally missing or duplicated.

The send mechanism implemented by the Event Sender uses pointers to the event fragment data in kernel-space to avoid kernel-to-user-level copying of the event fragment. Consequently, event fragment data are copied only once during the readout cycle, when the data are transferred from the Driver Module buffer to TCP socket buffers during the send call.

In order to simplify the programming logic and to reduce overhead in the Event Sender, a single thread matches Route Tags to event fragments and makes all send calls.

E. Event Builder

A single Event Builder process (EVB) runs on each farm node. Its primary purpose is to receive event fragments from the SBCs and concatenate them into complete events for processing by one of the two L3 Trigger filter processes also running on the farm node. As a secondary task, the EVB passes L3 Trigger programming information that it receives from the Supervisor process on to the L3 Trigger filter processes.

As event fragments are received, asynchronously, from Event Sender processes (Section III-D) on the SBCs, each fragment is appended to the contents of an appropriate shared memory buffer, keyed by L3 transfer number. When building the event, the EVB verifies that the event will fit within the size of a shared memory block, which is currently set to two megabytes, large enough to hold any one subcomponent's full calibration data. A basic check for duplicate fragments is also made. All component-specific data integrity checks are performed by the L3 Trigger filter process.

An event is marked as complete when it contains events fragments from all expected readout crates, as specified by the crate list sent to the EVB by the Routing Master for that event. Partial events which are still not complete after one second are discarded and the shared memory buffer is marked as free. (Partial events are typically the result of errors in the readout electronics upstream of the SBCs.)

L3 Trigger filter processes are signaled when complete events are available for processing. When a L3 Trigger filter process signals to the EVB that it has completely processed a particular event, the EVB marks the associated shared memory buffer as free.

Whenever a shared memory buffer is marked as free, the EVB sends a message to the Routing Master indicating that another buffer is available, unless too many free buffers are already being advertised (see Section IV-D).

IV. Flow Control and Buffering

Efficient operation at an event rate of 1 kHz is obtained by maximizing the available Ethernet bandwidth out of the SBCs, avoiding packet loss in the main switch by careful setting of the TCP protocol parameters on the farm nodes, and reducing the effects of latencies in the system by buffering data and communications.

A. SBC Dual Ethernet Operation

Sixteen readout crates in the system have possible event fragment sizes above 12 kilobytes, easily saturating a single 100-Mb Ethernet interface at an event rate of 1 kHz. Therefore, on these crates both SBC Ethernet interfaces are utilized. All farm nodes make two TCP/IP socket connections to the Event Sender in this case, one to each of the IP addresses of the interfaces. By default, socket connections are not associated with a specific hardware interface, so the the Event Sender uses the BINDTODEVICE socket option available under Linux to force the association. The Event Sender then alternates sending event fragments between the two socket connections.

Both interfaces are on the same IP subnet, which simplifies the switch configuration. However, Ethernet Address Resolution Protocol (ARP) [11] requests from farm nodes, which query all devices on the subnet in order to map IP addresses to Ethernet hardware addresses, appear at both Ethernet interfaces on an SBC, and the default Linux kernel ARP response logic always responds with the Ethernet hardware address of the first Ethernet interface in the route table for the associated subnet. Thus, the ARP responses from both interfaces contain the identical hardware address. The farm nodes, receiving the same hardware address, send messages from both socket connections to the same interface on the SBC. To achieve the desired ARP response, the standard Linux kernel was modified to respond to ARP requests with the hardware address of the interface that actually received the request [12].

B. TCP Receive Window Settings

Each 100 Mb output port on the Cisco 6509 switch has a finite-size FIFO buffer of approximately 1 MB. If this buffer becomes full, any further packets destined for the farm node serviced by this port will be dropped by the switch.

To avoid filling this output buffer, the TCP receive window size on the farm nodes is set such that the total amount of data in transit from all Event Senders to a single farm node is always less than the size of the corresponding output port buffer of the switch. Under the TCP protocol, for each socket connection the sender restricts the amount of data in flight to no more than the receiver's receive window size. The farm nodes' TCP receive window has been set to 10 kB and there are 80 data connections (the RM, 47 single Ethernet readout crates, and 16 dual Ethernet readout crates) to each farm node. Therefore the amount of data in transit to a single farm node is no more than 800 kB—less than the 1 MB output port buffer size.

C. Driver Module Buffering

The SBC Driver Module places event fragments in one of 50 statically allocated buffers. At least ten of these buffers are necessary to absorb the data resulting from the forced latency in the Routing Master: as described in Section III-B, the Routing Master bundles up to ten Route Tags into a single message before sending the message to the target SBC. The additional buffers account for fluctuations in the trigger rate.

D. Event Sender Buffering

As discussed in Section III-D, a single thread in the Event Sender makes all send calls. As a consequence, if a single send call to a farm node blocks, the Event Sender will be unable to send further events to any farm nodes. A single call will block when the TCP send buffer for that connection on the SBC fills up, most likely because the farm node associated with that connection is nonresponsive.

To avoid filling up any single TCP send buffer on the SBC, the TCP send buffer size is set to an integer multiple of the maximum possible event fragment size of 256 kB. The maximum number of event fragments in flight to any farm node at any moment is then limited to this integer multiple value (currently three) by means of a free buffer advertisement scheme discussed below. The value of three is chosen to keep the product of the send buffer size and the total number of farm node connections on an SBC smaller than the total available memory. Currently, the the per-socket send window size is set to 768 kB (three times 256 kB).

As discussed in Section III-B the Routing Master will instruct SBCs to route event fragments to a particular farm node only if that farm node has free buffers available. The Event Builder on a farm node, as discussed in Section III-E, sends periodic messages to the Routing Master informing the RM of the number of free buffers. Though the EVB has a maximum of 20 buffers (see Section IV-C), it reports at most three to the Routing Master. Therefore, an SBC will never be instructed by the RM to send more than three event fragments at any moment to a particular farm node.

E. Event Builder Buffering

The Event Builder stores complete events in 20 shared memory buffers. At least three buffers are required to implement the free buffer advertisement scheme described in Section IV-D. The remaining buffers absorb event pile-up due to the variable processing latency of the L3 Trigger filter processes.

Currently, the Event Builder initially stores received event fragments in a queue of depth 80 (enough for one complete event) before concatenating fragments into complete events. However, this buffering is not necessary. The free buffer advertisement scheme described in Section IV-D guarantees the availability of free shared memory buffers in the EVB. And the free buffer advertisement scheme and the TCP receive window settings (described in Section IV-B) guarantee that output buffers in the SBC and main switch switch, respectively, will not overflow regardless of the amount of time that the Event Builder requires to process an event fragment. A planned upgrade to the EVB logic will remove the event fragment queue and receive event fragments directly into the shared memory buffers.

V. Monitoring

All software components in the system (about 150 total) provide detailed statistics and status information to the Monitor Server, a software process that runs on a dedicated PC (similar to the farm node PCs). Software components connect to the Monitor Server (MS) as *clients* and respond to MS initiated queries. The MS itself responds to queries from *displays* such as web-page scripts and dedicated graphical-based programs. The displays query can request any set of information from any set of software components in the system. The MS makes only the necessary client queries and collects responses asynchronously.

The Monitor Server is capable of supporting a large number of displays making simultaneous queries, without burdening the clients. All client responses are cached in the Monitor Server and provided to display queries such that no client is queried more than once per second.

The data format of the queries and responses is XML. Human-readable, text-based XML formatting is flexible enough to handle the varied types of information in the system and can be parsed by all major programming languages on almost any computer platform.

VI. Performance

The L3DAQ system has been in continuous operation since May 2002. During a typical run, the DZERO L2 trigger accept rate is approximately 500 Hz, and the average event size is nearly 200 kB, dependent upon Tevatron performance and conditions. The system has performed reliably at these nominal parameters, incurring no deadtime in the DZERO data acquisition chain. CPU consumption on the SBCs is typically less than a few percent. The Routing Master and Event Builder processes consume less than 10% of the total CPU resources.

The CISCO 6509 is carefully monitored. Peak utilization of the gigabyte fiber module is approximately 15% during normal running conditions. Packet loss associated with the 100 Mb output modules is insignificant.

A limited test has been performed to evaluate the dual Ethernet operation of the SBC at a 1-kHz event rate using dummy data generated in the SBC. A data rate of approximately 22 MB/s has been achieved, close the theoretical maximum of 25 MB/s for two 100-Mb interfaces.

VII. Conclusion

The DZERO Level 3 DAQ system has been built from commercially available hardware: single board VME computers, Ethernet switches, and PCs. The software components rely upon high level programming languages; the Linux operating system; widely used, open libraries; and standard networking protocols. The system has performed reliably since commissioning in May 2002 and satisfies the current and future needs of the DZERO experiment.

Acknowledgment

The authors thank the staff at Fermilab and acknowledge support from the Department of Energy and the National Science Foundation.

References

[1] The D0 Collaboration, "The upgraded D0 detector," *Nucl. Instrum. Methods*, to be published.

[2] S. Abachi *et al.*, "The D0 Detector," *Nucl. Instrum. Methods A 338*, p. 185, 1994. D0 Collaboration.

[3] R. Carlin, W. H. Smith, K. Tokushuku, and L. W. Wiggers, "The trigger of zeus, a flexible system for a high bunch crossing rate collider," *Nucl. Instrum. Methods A 379*, p. 542, 1996. ZEUS Collaboration.

[4] B. L. Winer, "Status of the CDF II online trigger," *Int. J. Mod. Phys. A 16S1C*, p. 1169, 2001.

[5] K. Anikeev *et al.*, "Event builder and level 3 trigger at the CDF experiment," *Comput. Phys. Commun. 140*, p. 110, 2001.

[6] VMIME-7750 [Online]. Available: http://www.vmic.com/

[7] Universe II Manual [Online]. Available: http://www.tundra.com/

[8] PMCDIO64 [Online]. Available: http://www.bvmltd.co.uk/

[9] Cisco Catalyst 6509 Switch [Online]. Available: http://www.cisco.com/

[10] D. C. Schmidt and S. D. Huston, *C++ Network Programming, Vol. 1: Mastering Complexity with ACE and Patterns*. Reading, MA: Addison-Wesley, 2001.

[11] Ethernet Address Resolution Protocol [Online]. Available: http://www.rfc-editor.org/

[12] One ARP Patch [Online]. Available: ftp://linux-rep.fnal.gov/pub/drivers/