Switching techniques in data acquisition systems for future experiments.

M. F. Letheren

CERN, Geneva, Switzerland.

Abstract

An overview of the current state of development of parallel event building techniques is given, with emphasis on future applications in the high rate experiments proposed at the Large Hadron Collider (LHC). The paper describes the main architectural options in parallel event builders, the proposed event building architectures for LHC experiments, and the use of standard networking protocols for event building and their limitations. The main issues around the potential use of *circuit switching*, *message switching* and *packet switching* techniques are examined. Results from various laboratory demonstrator systems are presented.

1 Introduction

A high energy physics experiment is usually composed of several different multi-channel detectors, each of which is equipped with its own specific modular front-end and readout electronics. Whenever an event trigger occurs, readout controllers each read the data from a local group of channels and format it into an *event fragment* in a local front-end memory. The various event fragments belonging to an event are scattered over these front-end memories, and they must be brought together[1] before the event can be processed on-line using algorithms operating on the global event data, or before the event data can be recorded to mass storage. The process of collecting together the distributed event fragments is called *event building*.

Figure 1(a) shows the architecture of a generic data acquisition system that includes a processor farm for on-line software triggering and uses a shared-medium interconnect to perform event building by moving event fragments from the *sources* (front-end buffers) into the *destinations* (members of the processor farm). All processors in the farm run identical algorithms and any given event is processed by just one processor. An event manager controls the allocation of each “new” event to a “free” processor. The shared-medium interconnect may be for example a bus (e.g. FASTBUS or VMEbus), a token-passing ring or an ethernet segment. A control protocol operates between the source and destination modules in order to provide such functions as the sequencing of the readout of event fragments, checking that all fragments are correctly received, and the detection and recovery from errors.

Before proceeding we define parameters that we will use to characterize the event builder’s performance. The *rate* at which events can be built must at least match the highest expected trigger rate. Related to event building rate is *throughput*, the quantity of event data built per unit time. The *event building latency* is the delay from the trigger until the last event fragment of the event is collected; it depends mainly on the time that fragments spend queuing in various parts of the system and on software overheads in the destination. The operating *load* is the ratio of throughput to the nominal bandwidth offered by the event builder. The operating load is a measure of the efficiency with which the hardware is used. Note that the rate is **not**

1 We will not discuss an alternative approach using the *Scalable Coherent Interface* [1] to implement a distributed, shared-memory, memory-mapped event building architecture minimizing data movement.

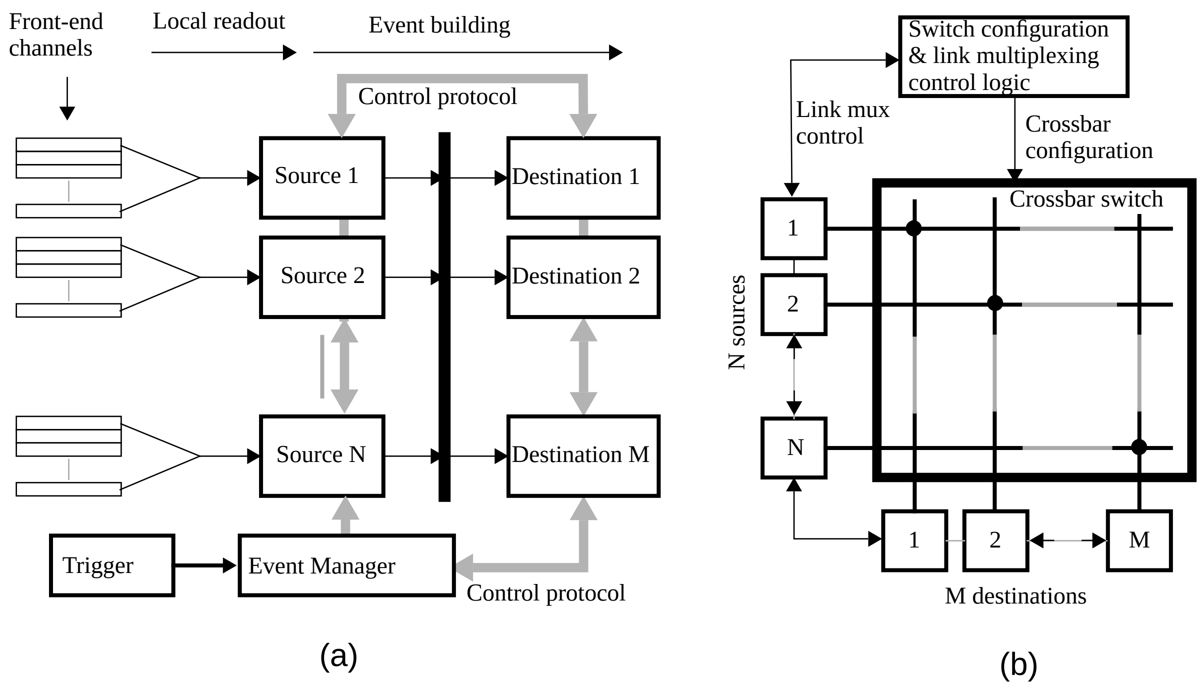


Figure 1: (a) A generic data acquisition architecture including an on-line filtering processor farm and event building based on a shared-media interconnect; (b) parallel event building using a crossbar switching fabric.

given by the reciprocal of the latency; when sufficient memory is available in the system, high rates (or equivalently, high throughput or load) can be achieved even when latencies are long (pipeline effect).

From the point of view of processing power, the architecture can be scaled to handle arbitrarily high trigger rates by adding processors. However, the shared-medium interconnect imposes a limit to the bandwidth available for moving event fragments to the destinations. Very high rate experiments require a parallel event building approach, in which events are built concurrently in the different destinations using a fabric of parallel interconnects between sources and destinations. All sources need to be able to connect to all destinations and figure 1(b) indicates how a *switching network* employing a *crossbar* architecture can be used to achieve this with one link interface per source and per destination. The switch can be electronically reconfigured to establish any desired pattern of parallel independent connections between N source and M destination modules. Until a path to the destination can be made available, each source must queue event fragments in one of M queues corresponding to the desired destination. Time multiplexing is used on the links to carry data from the different queues, while switching between different link interconnection patterns is used to route the data to the appropriate destination. A control scheme is needed to coordinate the configuration of the switch with the time multiplexing of the data on the links. In this paper we will describe switch-based event building using three different control schemes implementing *message switching, circuit switching* and *packet switching*.

The crossbar switch is said to be *non-blocking* because, from every source, one can always allocate a path to any destination which is not already connected to another source. The complexity of an N x N crossbar grows like N^2, and this limits the maximum practical size of a crossbar. Large switching networks use multi-stage topologies and have a complexity that grows like N.logN. Depending on the operating load and the traffic patterns, they suffer to a greater or lesser extent from internal blocking. Note that even with an internally non-blocking switch, *output blocking* can occur when multiple sources try to connect simultaneously to the same destination.

There are many aspects to a complete data acquisition and triggering system, but this paper limits its scope to parallel event building architectures based on switching fabrics. We first give an overview of the main architectural options in parallel event builders and then we describe the proposed event building architectures for LHC experiments, with emphasis on performance and other requirements. We then look at the use of standard networking protocols for event building and their limitations. This is followed by an overview of the issues around the potential use of *circuit switching* technologies for event building. Work on the alternative approach of using *message switching* technologies is then described and is followed by considering the application of *packet switching* and *cell switching* techniques.

2 Architecture Options for Parallel Event Building

2.1 Push versus Pull control protocols

The data flow control architecture can use either the *push* or *pull* discipline. In the push discipline, the event manager assigns a destination for the next event and broadcasts the event number and destination identifier to all sources. The sources then send out their event fragments to the assigned destination. This requires a minimal protocol overhead and has potentially the highest throughput. However, because multiple sources attempt to send their event fragments concurrently, the event fragments will arrive at the destination in an indeterminate order, requiring a scatter-gather hardware feature in the interface adapter, or a more intensive activity for buffer management and merging in the host.

One or more sources may be either “dead” or “empty” for a given event, and therefore, in the push architecture, the destination must implement some algorithm (as part of the *event building protocol*) that allows it to decide when all the event fragments for an event have been received. In addition, multiple sources compete for the same output and, depending on the switching technology and the algorithm used to assign events to destinations, the result may be reduced throughput, increased event building latency, or loss of data. In section 6.2.3 we show that these effects can be minimized by an appropriate *destination assignment algorithm,* and in section 7.2.2 we show how the *traffic shaping* technique can resolve these problems.

In the pull discipline the destination processor initiates the data transfer by requesting event fragments from each of the sources in turn. The event fragments are therefore delivered in a known, fixed sequence, and it is implicitly clear when all the event fragments have been collected. In addition, error detection and handling are relatively straight forward. The pull discipline can be used to implement intelligent, selective readout, thereby reducing the amount of data moved and allowing the use of smaller and cheaper switches. The sequential pull of fragments, or multicasting of requests to small groups of sources, avoids the congestion and blocking problems of the push architecture. The disadvantage is that the pull technique imposes a fair amount of software overhead to support the source-destination control protocols.

2.2 Event flow management algorithms

The event manager’s choice of destination for the next event may use strategies such as *round-robin, random destination, least loaded processor*, etc. Ideally the event manager should balance the load on the members of the processor farm, but, as already mentioned, the choice of the destination assignment algorithm can have a significant impact on the performance of the switching fabric.

2.3 Error detection and recovery

The control protocols for switch-based event building will be layered on the *physical* and *link layer* protocol layers[2] of the chosen link and switching technologies. These lower layers will include error detection and perhaps error correction capability.[3]

The event building protocol layers will have to handle errors passed to them from the lower layers, deciding for example how to handle a corrupted or lost event fragment. In addition they will have to monitor the validity of event fragments, and signal problems (e.g. dead sources, sources sending corrupt data, etc.) to a higher layer.

2.4 Phased event building

The generic architectures described in section 1 have been over simplified in order to introduce the basic concepts. In practice high rate experiments use multiple levels of on-line software triggering and may use *phased event building* schemes in order to reduce the required bandwidth for data movement. In phased event building, initially only a part of the event's total data are moved into the processor farm. A rapid decision is made based on this subset of the event data. For the small fraction of events that are accepted after the first phase, a second phase of event building is started in order to collect additional event data, on which more sophisticated analysis can be performed to further refine the selection of events. Multiple phases of event building can continue until the full event data is accessible to the on-line software trigger.

2.5 Switching architecture

As previously mentioned, each source (and destination) has one physical link to the switch fabric which is shared by the different logical connections to the destinations (sources). We will consider parallel event builders employing three different switching techniques distinguished by the way in which the logical connections share the physical links. Link sharing can be by synchronous or asynchronous time division multiplexing (TDM), or by sequentially opening and closing dedicated paths to the desired destination(s). Table I compares the three switching architectures, which will now be described in more detail.

2.5.1 Synchronous transfer mode and circuit switching

When synchronous TDM is used the data are said to be transported in the *Synchronous Transfer Mode* (STM). In telecommunications networks, in which developments were dominated by the requirements of voice communication, multiple channels are time multiplexed on a link using fixed-length time slots. On a regular cycle, each subscriber-to-subscriber connection (called a *circuit*) is allocated a time slot, during which it can transmit a voice sample. A global timing reference is provided by grouping time slots into a *frame* that repeats every 125 µs, and each circuit is allocated one time slot in a fixed position within the frame. When voice samples are switched in the exchange, the circuit, or equivalently the destination, of the sample is inferred from its position within the time frame.

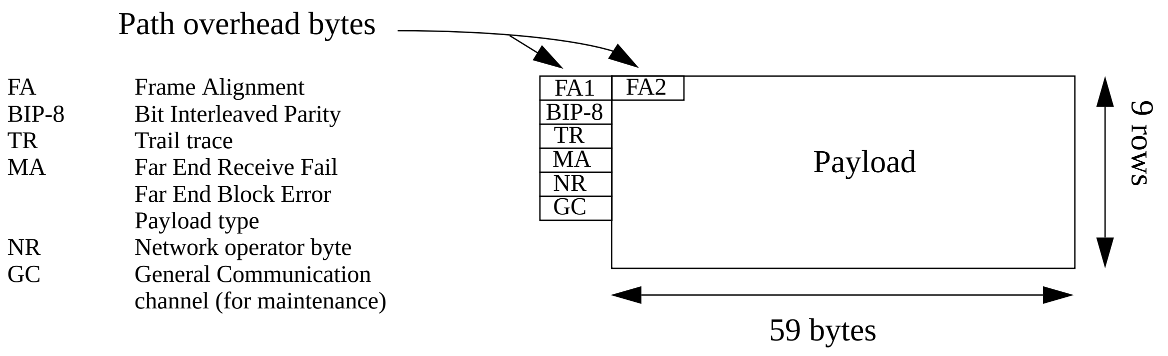
[2] We use the terminology of the ISO *Open Systems Interconnection* (OSI) reference model [2].

[3] In some standards, errors may be detected at these layers, whereas recovery may have to be implemented in the higher layers (if errors are infrequent, correcting them at a higher layer is simpler and leads to better overall system performance).

Multiplexing Scheme	Switching Technique	Application Area and Characteristics
Synchronous TDM (STM)	Circuit switching	Telephone switching technology; constant bit-rate traffic; equal bandwidth per circuit; concurrently active circuits.
Asynchronous TDM (ATM)	Packet switching	Data network switching technology; bursty traffic; bandwidth allocated per connection; concurrently active connections.
Dedicated connection	Message switching	Switching streams point-to-point; connection setup overheads; efficient for long block transfers; sequentially active connections.

Table I: Traffic multiplexing schemes and their associated switching techniques.

As an example, figure 2 shows the frame structure specified in the International Telecommunication Union’s (ITU) recommendation G.703 [3] for the transmission of circuits at the bit-rate of 34.368 Mbit/s. The frame is presented as a matrix of 59 columns (each of one byte) and 9 rows. One byte of the matrix is reserved and, together with an additional 6 bytes, forms the so-called *path overhead*, which is used for link error signalling and “operations and management” (OAM) functions. The total length of the frame is 537 bytes and it is transmitted in exactly 125 µs at the bit-rate of 34.368 Mbit/s. When a time slot of length eight bits is used, the available 530 byte payload can carry 530 circuits of 64 kbit/s each.



FA	Frame Alignment
BIP-8	Bit Interleaved Parity
TR	Trail trace
MA	Far End Receive Fail Far End Block Error Payload type
NR	Network operator byte
GC	General Communication channel (for maintenance)

Figure 2: The 125 µs G.703 frame structure used for synchronous transmission at 34.368 Mbit/s.

The STM data are switched in the exchange using the so-called *circuit switching* technique, whose principle is indicated in figure 3. The switch first synchronizes frames on incoming links, and then maps slots from the frames on each incoming link into new slots in frames on an outgoing link. In general, in the public switching networks the time slots used to carry a circuit on the incoming and outgoing links are not in the same fixed position within their respective frames. Therefore the STM switch will be built from a combination of time slot interchange (TSI) units and space switches, which together perform the reordering of slots in the time frames and the routing of slot data from input link to output link. The TSI is essentially a memory in which the time slots are organised into separate logical queues, that can be randomly accessed by the switch. The TSI and switch routing control information are shown in tabular form in the figure. This table is set up during the *signalling* protocol that establishes the circuits between the subscribers (the call dialling process).

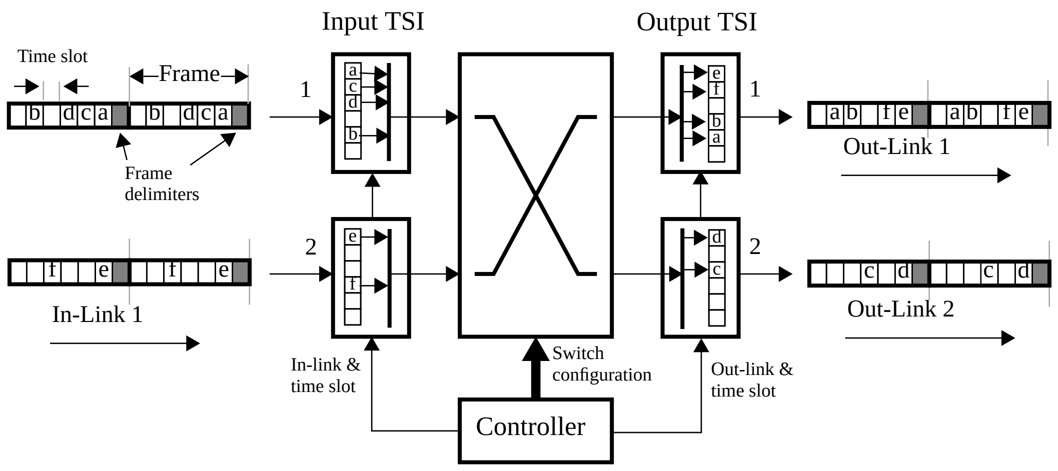


Figure 3: The principle of circuit switching. Time slots are switched between input and output time slot interchangers (TSIs) under the control of a mapping table that, for each time slot in the frame of each input link, defines (i) the switch configuration needed to route the time slot to the appropriate output link and (2) the new time slot position in the frame on the outgoing link.

Circuit	In-link	slot #	Out-link	slot #
a	1	1	1	5
c	1	2	2	3
d	1	3	2	1
b	1	5	1	4
e	2	1	1	1
f	2	4	1	2

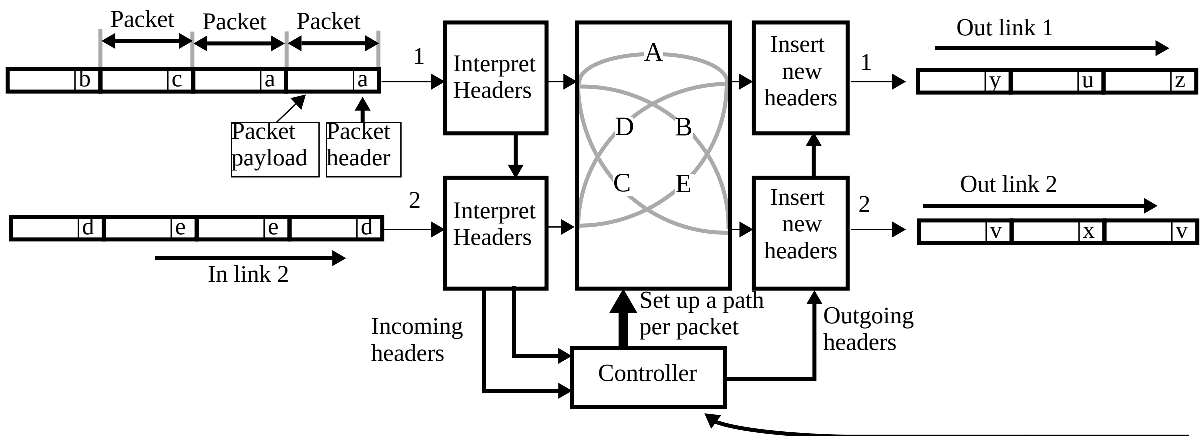
An important characteristic of the circuit switched technique is that it offers equal bandwidth to all circuits. Circuit switching is not well suited for supporting a mixture of different classes of services, such as computer network traffic, telephone traffic, compressed video, or high definition television (HDTV), each of which requires circuits with widely different bandwidths (from 64 kbit/s to 140 Mbit/s). In addition computer network and compressed video traffic are naturally bursty; circuit bandwidth is wasted when the sender is silent or not transmitting at the peak bandwidth allocated to the circuit. Such applications are better handled by:

2.5.2 Asynchronous transfer mode and packet switching

When the time division multiplexing of the link uses an asynchronous discipline, the data are said to be transported in the *Asynchronous Transfer Mode* (ATM).[4] In this mode there is no frame to provide a global time reference. Data is transmitted in chunks called *packets* and, because there is no frame, each packet must carry with it a connection label (or source and destination address) to identify the connection to which it belongs. The absence of a fixed frame and the fact that there is no longer a requirement to maintain global timing synchronization, means that packets can be of variable length.

[4] Here the acronym ATM is used in a generic sense, and is not to be confused with the specific ATM technology standardized for B-ISDN and LAN switching, which is introduced later in this section.

The switching technique associated with asynchronous TDM is known as *packet switching*, and it uses the packet's label (or destination address) to decide to which output link the packet should be routed, as shown in figure 4. The label can be thought of as identifying a *virtual connection* on which all packets with the same label travel through the system. Because subscribers can send variable length packets and can seize link bandwidth on demand, this mode offers a more flexible service for intermixing virtual connections that require different bandwidths. Because bandwidth is only used when packets are actually transmitted it can also efficiently carry bursty traffic (where the instantaneous bandwidth utilization of a virtual connection fluctuates in time). Bandwidth which is not used by a virtual connection is available for use by other virtual connections. The resources of link and switch are shared between virtual connections by statistical multiplexing. For a large number of independent connections the statistical fluctuations of individual connections average out, and the switch can run at a high average load.



conn.	In-link VCI		Out-link VCI	
A	1	a	1	u
B	1	b	2	v
C	1	c	2	x
D	2	d	1	y
E	2	e	1	z

Figure 4: The principle of packet switching. Packets are switched between input and output ports according to the *virtual connection indicator* (VCI) label in the packet header. A mapping table is used by the switch to route packets to the appropriate output and to generate a new VCI label for the outgoing packet.

The packet switching hardware uses the packet label or destination address to decide to which output port it shall send the packet. In ATM the correspondence between the label value, the output port number and the new label value of the cell on the output link is contained in tables in the switch hardware. As for the circuit switched case, these tables are initialized by a signalling protocol which establishes the virtual connections.

The flexibility of packet switching lead to its use in computer networking technologies such as *ethernet*, the *Fibre Distributed Data Interface* (FDDI), etc. More recently, this flexibility to intermix and efficiently support the differing traffic characteristics and quality of service requirements of applications, such as video and audio distribution, video conferencing, and computer data transport, has lead to it being chosen as the underlying mechanism for the Broadband Integrated Services Digital Networks (B-ISDN). The telecommunications industry has standardized a particular asynchronous transfer mode technology for the B-ISDN [4], which is based on packets with a fixed length of 53 bytes. The particular asynchronous transfer mode technology used by the B-ISDN has now subsumed the use of the acronym ATM, and from here onwards we will implicitly refer to the B-ISDN version of the asynchronous transfer mode whenever we use the acronym ATM. The 53-byte packets used for the ATM transmission and switching are called *cells* in order to differentiate them from the generic use of the term "packet" or the specific packets used at higher levels of data communications protocols. The particular technology of ATM switching is called *cell switching*.

The basic ATM standard defined by the ITU is being adopted by the computer industry for high speed networking, and the ATM Forum [5] is further developing the standards with emphasis on the computer industry's requirements.

The *Synchronous Digital Hierarchy* (SDH) [6] and the *Synchronous Optical Network* (SONET) [7] are (almost) identical framing standards that have been designed to support the multiplexing of all previously defined synchronous transmission standards in a common STM frame. However, SDH and SONET also support the asynchronous TDM transport of ATM cells by loading them into the first available position within the frame payload. The SONET / SDH standards are widely used for transport of ATM cells at the physical layer within both the telecommunications and computer industry. In addition, the ATM Forum has defined several alternative physical layer transport standards for use with ATM.

2.5.3 Dedicated point-to-point links and message switching

In contrast to the circuit switching and packet switching techniques, which use time division multiplexing to concurrently carry the traffic of multiple circuits (connections) over a link, the *message switching* approach sets up a dedicated path between source and destination for the time required to transmit a complete message (in our case an event fragment). Once the message has been transmitted the path is torn down and a new dedicated path can be set up to the next destination.

The delay required to set up a connection imposes a deadtime, during which no data are transferred. As shown in figure 5, this overhead consists of one component contributed by the routing of a connection request through the switch, and a second component contributed by the delay before the receiver signals back a connection accept. The effective throughput that can be achieved depends on the ratio of the connection set up overhead to the time required to transmit the message.

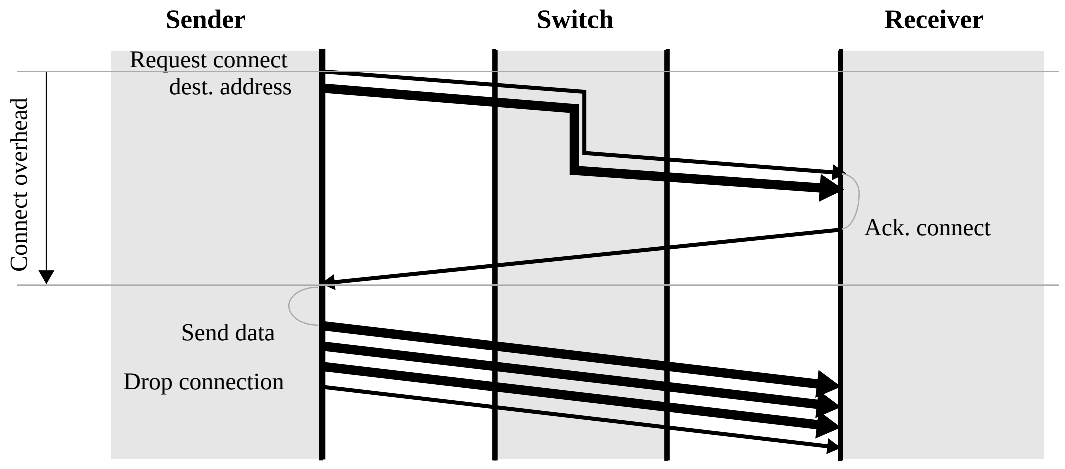


Figure 5: Message switching involves an overhead in order to set up a dedicated connection.

Examples of message switching technologies, originally developed for high speed channel connections between processors and/or peripherals, are the *High Performance Parallel Interface* (HiPPI) standard [8] and the *Fibre Channel* [9] standard (in service class 1).

2.5.4 Signalling

In telephone or typical computer networks, a *signalling protocol* must be used to establish a circuit or (virtual or dedicated) connection between source and destination before data can be transferred. Signalling sets up a contract between the two connection end points (source and destination) and the switching fabric. On the other hand, in event building the full connectivity between all sources and destinations is used, and no connection is dormant for long periods. In the circuit switched or packet switched event builder we can take advantage of this fact by defining circuits or virtual connections which are left in place throughout the data taking run. They can be initially established either by running signalling protocols, or by adopting a system-wide convention that allows off-line calculation and subsequent downloading of the mapping tables for the switches. By using such semi-permanent circuits or virtual connections we avoid overheads associated with dynamic connection set up.

By contrast, in the case of event building with message switching, the use of dedicated connections forces a signalling protocol to be executed for every message transferred, and results in a connection set up overhead as indicated in figure 5.

3 Proposed Architectures and Requirements for the LHC Experiments

We describe next the phased event building schemes proposed for the CMS and ATLAS experiments at the LHC, and use these to define event builder requirements.

3.1 Phased event building in CMS

The CMS collaboration proposes to use a single large switch and processor farm [10] to perform event building and filtering in two phases, as shown in figure 6. The first phase (which they call “virtual level-2”) operates at the full level-1 trigger rate (100 kHz) and builds the event data from a subset of the detectors (the calorimeter, preshower and muon detectors). The event fragments are stored in front-end dual port memories (DPMs) and the events are built in similar dual port memories (Switch Farm Interface - SFI) from where they are passed to a processor which then executes the second level trigger algorithm. All processors of the farm are used for

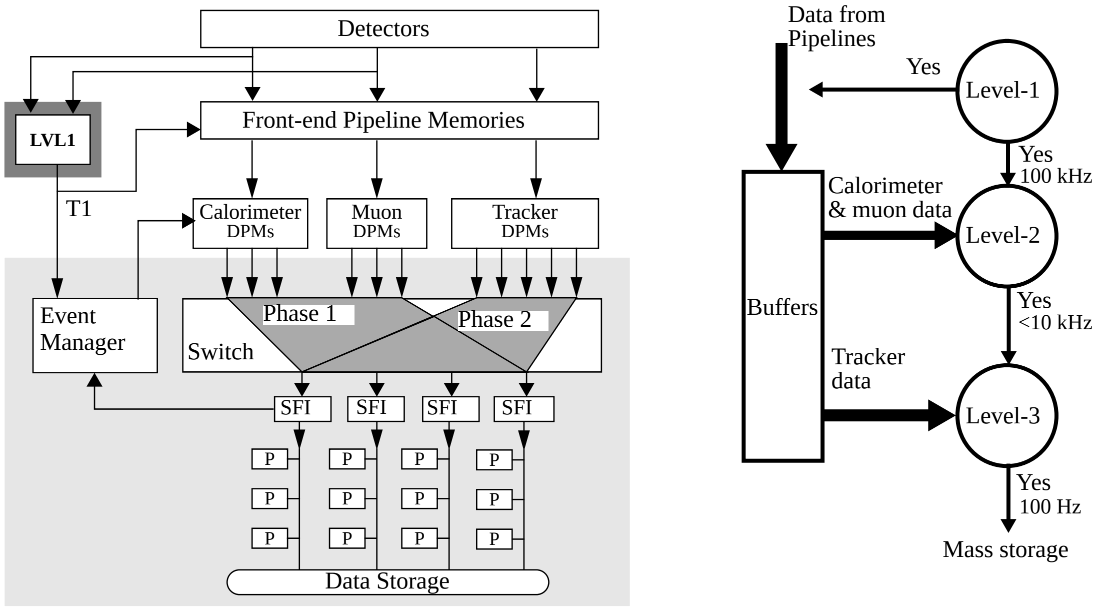


Figure 6: The phased event building scheme proposed for the CMS experiment.

the second level trigger. Whenever one of the processors finds an event that passes the second level trigger, it initiates the second phase of event building, in which the remaining data for that event (from the tracker) are assembled into the SFI. Once the entire event data is present the processor proceeds to execute the level-3 trigger algorithms.

Note that the DPMs have to emit event fragments at the full level-1 trigger rate of 100 kHz. The protocol with the event manager, the access of the appropriate event fragment queues, the adaptation of the data to the switch interface and the memory management of the DPM is a formidable challenge at these rates. Considerable effort is being expended on the development of programmable and hardwired DPM controllers [11].

3.2 Phased event building in ATLAS

Figure 7 shows the logical model of the ATLAS architecture [12] and the proposed event building phases. The level-1 trigger provides the level-2 trigger system with pointers to regions of the detector containing information that fired the level-1 trigger. The "regions of interest" (RoI) pointers change from event to event; the RoIs are dynamically selected for readout by the level-2 trigger system; the data of each RoI is processed in a different local processor (LP) in order to extract compact descriptions (<100 bytes) of the physics *features* (e.g. a shower in the calorimeter). After the local processing step, each event's features are sent to one of a farm of global processors (GP), where a level-2 trigger decision is made. The level-2 decision may not use the information from all detectors. Thus the level-2 system uses partial event building in local and global phases over the local and global networks shown in figure7. For those events that pass the level-2 trigger, there follows a full event building phase for the level-3 trigger system.

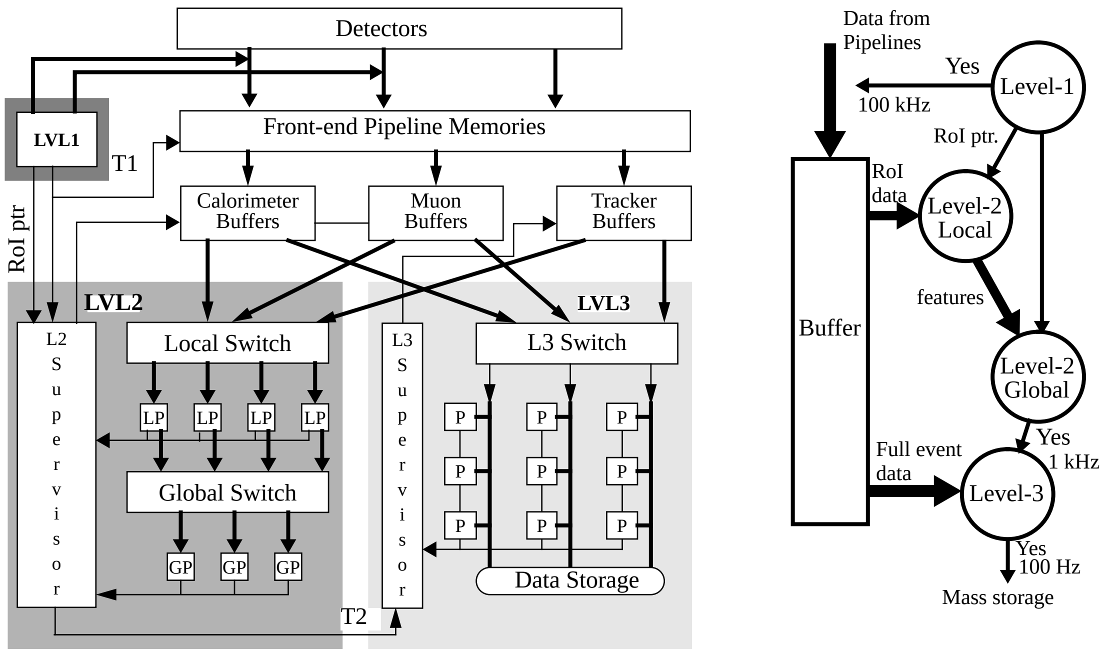


Figure 7: The phased event building scheme proposed for the ATLAS experiment

The model shown in Figure 7 is a logical model; it could be implemented in many ways, for example by using physically distinct networks and processor farms for each phase, or by handling the three logical phases with a single farm and a single large flexible switching system.

The average number of RoIs per event is expected to be about 5, and their selective readout is expected to reduce the effective level-1 trigger rate (T1) to be handled by individual front-end buffers to approximately 10 kHz (c.f. 100 kHz in CMS), which gives more scope for development of an intelligent front-end buffer [13]. In principle, the parallel processing of RoI's also reduces level-2 data collection and trigger processing latency with respect to the equivalent latencies in the CMS architecture. However, given the availability of large, affordable front-end buffer memories, this does not seem to the author to be a decisive advantage.

3.3 Requirements for event building at LHC

3.3.1 Bandwidth and latency requirements

The data acquisition systems of the LHC experiments [10, 12] are being designed to handle a maximum first level trigger rate T1 ~ 100 kHz, corresponding to operation at the nominal full LHC luminosity of 10^{34} $cm^{-2}s^{-1}$. In order to estimate the bandwidth requirements for the phased event building system, we reproduce in table II the expected event data sizes for each subdetector of the CMS experiment. The tracker and pixel detector together produce the largest contribution to event data, which is why they are not used in the virtual level-2 phase. It is (conservatively?) estimated that the virtual level-2 phase will reduce the level-1 rate T1 ~ 100 kHz to a level-2 accept rate T2 ~ 10 kHz.

Table II also shows the estimated maximum event building traffic to be handled by the CMS virtual level-2 and level-3 phases under the above assumptions. The event builder should support an aggregate throughput of ~25 GByte/s, or approximately 200 Gbit/s. In practice it is expected to be difficult to operate a large switch above 50% of its nominal aggregate bandwidth, therefore CMS have specified a total switching bandwidth of ~ 500 Gbit/s. A suitable CMS event builder could be implemented for example with a large switching fabric having 1000 input and 1000 output ports, each running at the nominal standard SONET bit-rate of 622 Mbit/s.

Sub-detector	No. channels	Occupancy (%)	Event size (kByte)	Event builder bandwidth (GByte/s)	
				L2 traffic (T1=100 kHz)	L3 traffic (T2=10 kHz)
Pixel	80 000 000	0.01	100	-	1
Inner Tracker	16 000 000	3	700	-	7
Preshower	512 000	10	100	10	-
Calorimeters	250 000	10	50	5	-
Muons	1 000 000	0.1	10	1	-
L1 trigger	10 000	100	10	1	-
TOTALS:	~ 10^8	-	~1000	17	8

Table II: Average sub-detector event size and expected event building bandwidth contributions for the CMS experiment (drawn from [10]).

The required aggregate event building bandwidth for ATLAS is somewhat smaller than for CMS, due to the selective readout of RoIs, and is estimated to be of the order 100Gbit/s.

In the CMS architecture, each of the 1000 destinations must make on average 100 level-2 decisions per second, i.e. one every 10 ms. If the processing of events is overlapped with event building, the event builder must deliver a new event at each destination every 10ms on average. The latency for building an event can be longer than 10 ms if several events are concurrently built in each destination. The upper limit of acceptable event building latency will be determined by the number of concurrently built events that can be supported by the switch interface hardware and software, and not by the available buffer memory (the planned 100 MByte capacity of the CMS DPMs is sufficient to buffer the full level-1 data stream for 1 second). A few times 10 ms is an acceptable latency for building a single event for level-2. For level-3, acceptable event building latencies are up to a few hundred ms.

3.3.2 Other requirements

There are several important “soft” requirements to be taken into consideration. The event builder must be *expandible*, and exhibit favourable scaling characteristics for throughput and latency as a function of offered load. The chosen technology should follow an *open communications standard* that will ensure plug-compatibility with equipment from different manufacturers, and the long term commercial availability of system components (interface adaptors, protocol chip sets, etc.). Ideally, the life cycle of the switch itself should match that of the LHC experiment.

In view of the scale of the LHC event builders, a *fault tolerant* switch architecture would be desirable, and good *operations and management tools* for monitoring of performance and error rates, testing and fault diagnosis will be necessary. The switching fabric should support *partitioning* into individual private networks for the independent test and development of detectors by their dedicated teams, and it should provide the ability to spy on the data streams.

A rich range of switching technologies is currently under study as potential candidates for constructing high rate event builders [e.g. 1, 14-17]. In this paper we select a few standardized, commercially supported technologies to illustrate the pros and cons of the three different switching architectures described in section 2.5.

4 Event Building using the TCP/IP Protocols

A number of experiments [18-20] are using farms of high performance UNIX workstations to perform on-line event filtering with (a possibly reduced version of) the off-line event analysis software. Front-end readout processors running a real time operating system assemble the event fragments. The events are built in the UNIX workstations using a high performance network and commercial network adaptors. The standard internet communications protocols TCP/IP, which are supported in both the UNIX operating system and the real time kernel of the front-end processor, are used to transmit event fragments. This approach minimizes the amount of special hardware and protocol software development needed.

4.1 Overview of TCP/IP

Figure 8 shows the TCP/IP protocols in terms of the seven layer OSI protocol reference model [2]. The OSI *physical* and *link* layers implement the underlying network technology-dependent functions that handle the transmission of data in blocks or packets. The *Internet Protocol* (IP) corresponds to the OSI *network* layer; it hides the underlying technology from the higher levels and guarantees interworking between networks based on different technologies. The IP layer forwards data in IP packets (length up to 64kByte) and fragments the IP packet into the technology-dependent block or packet formats used at the link layer. IP provides a connectionless service, i.e. it does not open a connection with the destination when sending the packet, and it provides no end-to-end reliability functions or flow control.

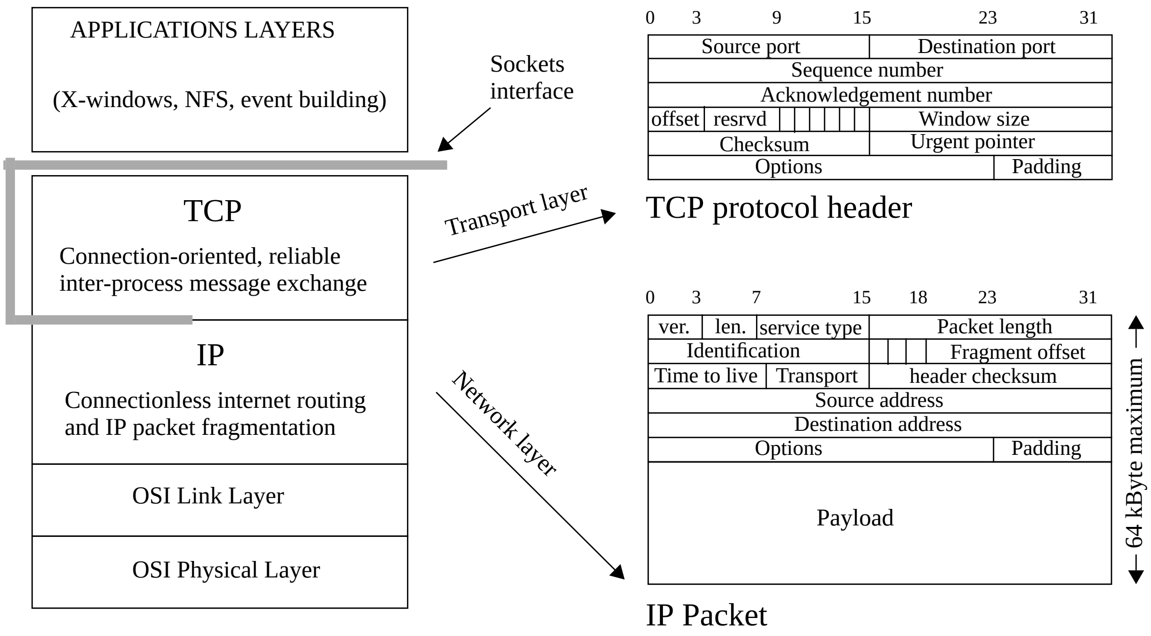


Figure 8: The TCP/IP protocol layers provide independence from underlying network technology and reliable connection-oriented data exchange

The *Transmission Control Protocol* (TCP) is layered on IP and corresponds to the OSI *transport* layer. It provides a connection-oriented service by supporting virtual connections between a source application and a destination application. The connection must be set up before messages can be exchanged. TCP ensures reliable, in-order delivery with error detection and packet re-transmission after acknowledgment timeouts. A sliding window data flow scheme increases throughput on long connections by allowing the sender to transmit ahead without waiting for acknowledgement.

4.2 Event building with TCP/IP

The UNIX operating system allows interprocess communication over TCP/IP via a de facto standard set of system calls known as the *sockets interface*. The sockets interface can be used to transfer event fragments between the front-end processors (running a UNIX-like real time operating system with support for TCP/IP sockets) and the UNIX host in which the events are to be built and filtered.

The use of TCP/IP for event building appears attractive because it ensures manufacturer and technology independence, as well as providing “for free” many important features (error checking, re-transmission etc.) that would otherwise have to be developed by the DAQ systems designer. However, TCP/IP implementations may not be optimally layered on any given technology, and the fact that they are accessed via operating system calls means that performance is probably limited by operating system overheads and the copying of data buffers between user-space and kernel-space. An additional penalty in using TCP/IP for event building is that the IP layer is largely redundant because its primary function (providing interworking in an inhomogeneous network) is not required. In addition, although TCP provides flow control on each connection, it cannot provide the flow control that is needed to manage congestion when independent TCP/IP connections concurrently send traffic to the same output port. Such congestion must be resolved by the network hardware or minimized by the system designer applying a *traffic shaping* technique, as we will describe in section 7.2.2.

4.2.1 TCP/IP performance measurements

Measurements have been made to understand whether TCP/IP-based event building could be applied to high rate data acquisition systems using packet switched network technologies such as Fibre Channel or the ATM. Table III summarizes the measurements made in [21] using a 266 Mbit/s Ancor CXT 250 Fibre Channel switch, IBM RS/6000-250 (66 MHz, 63 integer SPEC) and RS/6000-590 (66.6 MHz, 122 integer SPEC) UNIX workstations equipped with Fibre Channel adaptors from Ancor.

Transfer	TCP/IP Overhead	Message Latency	TCP/IP throughput (64 kByte message)
RS6000-250 -> switch -> RS6000-590	610 µs	1104 µs	10.3 MByte/s
RS6000-590 -> switch -> RS6000-250	365 µs	1092 µs	12.1 MByte/s
RS6000-250 -> RS6000-590	610 µs	1025 µs	10.3 MByte/s
RS6000-590 -> RS6000-250	365 µs	1025 µs	12.1 MByte/s

Table III: Measured overheads, communication latencies and throughputs using TCP/IP to pass messages between UNIX workstations over a 266 Mbit/s Fibre Channel switch [21].

In the table, *overhead* is defined as the time from an application sending a message to the time when it is ready to send a new message, not counting the time to transfer data; it was evaluated by measuring the time to perform the write of a one byte message. The reciprocal of the overhead defines an upper limit to the rate at which (zero length) event fragments can be pushed out. The *latency* was measured as half of the round trip delay for bouncing a 1 byte message back to the sender, and *throughput* was measured as the volume of data transmitted per second when using a large message of 64 kBytes.

The TCP/IP overhead was found to be dominated by software and to scale approximately with the power of the CPU used. The observed overhead limits the maximum rate of sending event fragments to a few kHz. The measured best TCP/IP throughput was limited to 12 MByte/s by available CPU power (the observed CPU utilization was of the order 90%). The observed throughput increased to 16 MByte/s when the TCP/IP protocol layers were bypassed by using a light weight “direct channel” protocol supplied by Ancor. The switch connection set up latency (hardware) only contributes 70 or 80 µs to the total latency.

Measurements of transfers made between two Digital Equipment Corporation Alpha workstations with TCP/IP running over a Digital Equipment Corporation Gigaswitch using 155 Mbit/s ATM link adaptors achieved a throughput of approximately 130 Mbit/s for large messages [22]. However, throughput dropped significantly for message lengths below 5kByte. For messages of 1 kByte, throughput was of the order 40 Mbit/s.

These measurements demonstrate that the software overheads associated with TCP/IP make it unsuitable for applications where short communications latencies or high throughput are required, or where short messages need to be sent at high frequencies. The impact of the TCP/IP overheads can be minimized by packing multiple events into each message (see for example [19]). Even so, when using 1 Gbit/s technology, current high performance workstations can only sustain throughputs corresponding to approximately 10% of the link bandwidth. Clearly this situation will improve as more powerful processors become available and/or network adaptors incorporate dedicated protocol processors.

In summary, the poor performance of TCP/IP with small packet lengths and the CPU-limited throughput obtained with larger packets make it unsuitable for high rate experiments such as ATLAS and CMS. However, TCP/IP could find application in the ALICE [23] heavy ion collider experiment, where the event rate is expected to be~50 Hz and the event size around 20 MByte. E.g. by choosing a 128 x 128 Fibre Channel switch with link speeds of 266 Mbit/s, and running TCP/IP on ~100 MIPS processors, we would perform event building with average fragment sizes of ~200 kByte. Under these conditions TCP/IP software overheads would be insignificant and one should be able to achieve approximately 40% load on the switch. In pp-collider operation the ALICE event fragments would be of size ~200 Byte and the event rate around 500 Hz, which is low enough that TCP/IP overheads would not limit throughput.

High performance event building with single events requires the elimination of operating system overheads and the layering of event building protocols directly on the network layer.

5 Parallel Event Building using Circuit Switching

The adaptation of the telephone circuit switching technique described in section 2.5.1 to parallel event building was first reported in [24]. An N x N non-blocking crossbar switch was used, and the input and output time slot interchangers shown in figure 3 were implemented by dual ported memories (DPM). As indicated in figure 9(a), each input DPM contained a logical queue for each destination, into which were placed the event fragments to be sent to that destination. Each destination DPM contained a logical queue per source, in which the event fragments coming from that source were assembled. A global control scheme defines the time slots and synchronizes the configuration of the switch with the enabling of the appropriate queues in the source, and destination modules. Figure 9(b) indicates how the controller globally orchestrates the set up and tear down of connections in successive time slots, so that no two sources simultaneously connect to the same destination, and all sources regularly connect to all destinations in a cycle (corresponding to the length of the time slot frame). Figure9(c) shows how, using a round-robin destination assignment algorithm, successive events can be built in successive output DPMs.

This simple *barrel shifter* scheme avoids output blocking and is very efficient when all event fragments have equal size and the time slot corresponds to the transmission time of one event fragment; in this case one complete event finishes per time slot. The throughput of the event builder scales linearly with the size of the crossbar switch. However, when event fragment sizes are variable, bandwidth efficiency drops because time slots are only partially filled.

AT KEK, a custom built barrel shifter event builder is being developed for possible use in a B-physics experiment [25]. High bandwidth efficiency is maintained by performing concurrent building of multiple events in each destination. As indicated in figure9(d), when a source has finished transmitting the current event fragment, it fills the remainder of the time slot by starting transmission of the next event fragment in the currently selected queue. The optimal choice of time slot size [26] was found to be equal to the average event fragment size (larger time slots increased the required queue lengths in source and destination, while smaller time slots increased the impact of overheads associated with switching between time slot states).

6 Parallel Event Builders based on Message Switching

6.1 Event building with transputers

A good example of the message switching approach is provided by event building for the

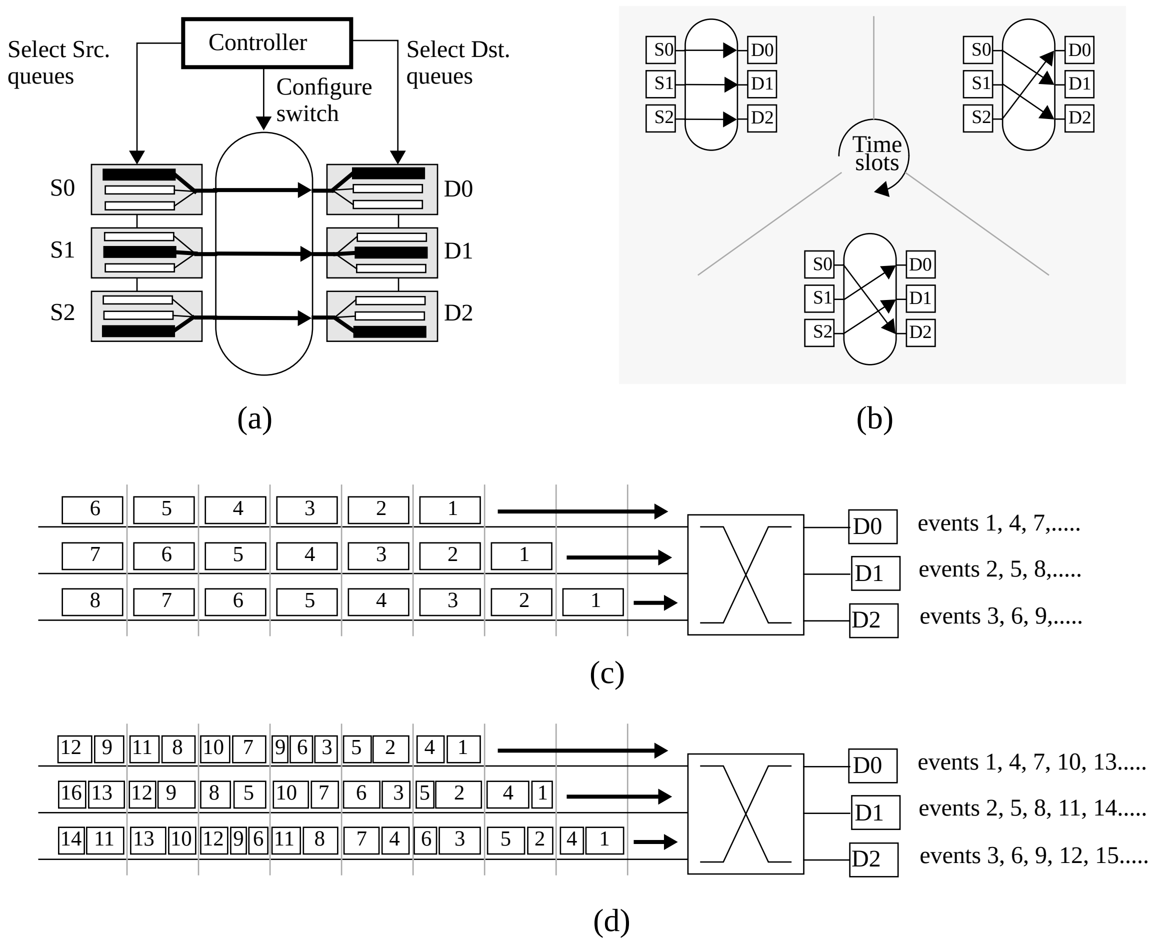


Figure 9: Event building with a simple synchronous circuit switched barrel shifter; (a) source and destination queues; (b) switch configurations in successive time slots; (c) fragment sequencing on the links for constant sized fragments, one event built per destination; (d) fragment sequencing for variable sized fragments, multiple events built per destination.

level-3 trigger farm in the Zeus experiment [27]. The Zeus event builder is constructed from T800 transputer-based modules and uses a custom made 64 x 64 crossbar switch to build events from 17 sources into 6 destinations. Each source has 3 links to the switch and each destination has 8 links to the switch. Links are based on the T800 transputer’s serial link (nominal throughput 20 Mbit/s) and the switch is built from four C004 32 x 32 crossbar switch chips. Source and destination modules were constructed using T800 transputers and triple-ported memory to hold the event fragments and the built events. The software in the transputers runs in stand-alone mode (no operating system).

A router and controller module, based on a transputer, runs software that allocates events to the least busy destinations. As soon as an event fragment becomes available in the 3-port memory of a source, the controller module is requested to set up a connection to the allocated destination via the crossbar. The event fragment is transferred using the T800 link protocol and then the connection is released. Since sources make their requests asynchronously, output blocking can occur if a source tries to connect to a destination that is already connected to another source. In that case the connection request is queued until the output becomes available. The T800 link throughput is limited to 600 kByte/s by the delay of the handshaking protocol operating over the 90 m of cable between the source and destination modules. The event builder can sustain an aggregate bandwidth of 24 MByte/s, which is sufficient to meet the Zeus design goal of handling at least 100 events/s of average size 150 kByte.

The transputer architecture offers an efficient integration of distributed multi-processing and interprocess communication (the most recent transputer technology uses packet switching) that makes it particularly suitable for building homogeneous, scalable data acquisition systems [28, 29]. The construction and evaluation, for event building applications, of a large high performance packet switch based on OMI/HIC link and switching technology [30] is being carried out at CERN [17].

6.2 Parallel message switched event building with HiPPI and Fibre Channel

A similar approach to that adopted for the Zeus event builder could be taken with commercial HiPPI or Fibre Channel (class 1) switches. Both of these technologies use message switching by including, at the head of the bit stream sent over the link, information which is intercepted by the switch and used to set up dedicated, full bandwidth connections[5].

6.2.1 HiPPI technology

HiPPI was originally developed as a point-to-point simplex (unidirectional) connection technology for high speed data transfer between mainframes and peripherals using a 32-bit wide data path (copper cable) and a 25 MHz clock, resulting in a bandwidth of 800 Mbit/s. HiPPI sender and receiver use a simple connection set up protocol based on a connection REQUEST line, asserted by the sender, and a CONNECT acknowledge line, which is asserted by the receiver if it accepts the connection request. Once the connection is established, one or more variable length *packets* (the HiPPI packet is **not** a packet in the sense of packet switching because it contains no label or address) can be sent. The HiPPI packets are broken down into *bursts* of 1024 bytes. A credit-based flow control mechanism is supported in which the receiver gives a number of "credits'" to the sender by pulsing a READY line (one credit per pulse). The sender may send one burst per credit until it has exhausted its supply of credits. The connection is broken either by the sender dropping its REQUEST line or the receiver dropping its CONNECT line. Byte parity is used on the datapath and a checkword follows each burst.

HiPPI has been extended to support message switching by augmenting the connection set up protocol with the transmission of a 4-byte I-field. The I-field is used by a switch to establish a path to the receiver, which can then issue CONNECT acknowledge. If the connection request cannot be routed to the destination, either because the switch's output port is already occupied or because of internal blocking in the switch, the sender can wait for the connection to be established, or it can drop the request (and possibly try to connect to a different receiver). HiPPI is a very simple protocol which suffers very little throughput overhead caused by header/trailer information. Typical connection set up overheads are ~ 1 µs and have little impact on effective throughput, even for short messages and fast switching rates.

[5] Often HiPPI and Fibre Channel (class 1) are referred to as circuit switched technologies because, during the lifetime of a connection, an unbroken chain of dedicated links is reserved between source and destination to transmit, at full speed, the data associated with the connection; there is no multiplexing on the links of data from other connections during the lifetime of the connection. This technique is equivalent to the early dedicated circuit switching performed in telephone exchanges. In this paper we prefer to use the term message switching in order to draw a distinction with the synchronous TDM and switching techniques used in modern telephone exchanges (as previously described in section 2.5.1).

6.2.2 Fibre Channel technology

The Fibre Channel protocol, as its name implies, was originally developed as a high performance full-duplex switched interconnect technology for communicating large blocks of data at high speed between processors and peripherals. However, the standard was later extended to support general purpose packet switched networking. The Fibre Channel standard [9] defines five protocol layers (FC-0 through FC-4).

The FC-0 layer defines the physical media, connectors, and standard bit-rates to be used on Fibre Channel links, which always consist of a pair of fibres/wires (one in each direction). The standard bit-rates are 132.8, 265.6, 531.25 and 1062.5 Mbit/s, and are supported by media ranging from shielded twisted pairs, through coax, multimode and monomode optical fibre. Fibre Channel switches available today operate at 266 Mbit/s; 1 Gbit/s switches are under development.

The FC-1 layer defines clock and data encoding in the serial bit stream, with detection of transmission errors. The method used is called 8B/10B because 8-bits of parallel data are converted into 10 bits in the serial stream.

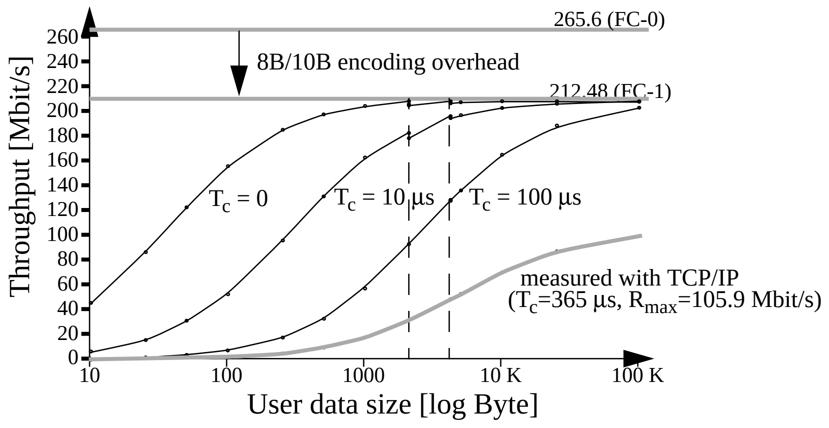


Figure 10: Effective maximum obtainable user-data throughput over a class 1 switched Fibre Channel link as a function of the length of the user data message and the connection set up and tear down overhead (T_c). Throughput drops for short messages due to the fixed 36-byte header/trailer overhead of each frame. No overheads are included for software or protocol chip set operation, except for the case of the TCP/IP curve, where measurements reported in section 4.2.1 showed that software imposes T_c = 365 µs and limits maximum throughput to 105.9 Mbit/s for long messages.

The FC-2 layer defines the transport mechanism between nodes (*N-ports*) communicating either via a point-to-point Fibre Channel link, or a switch. The basic protocol data unit is the *Frame*[6], which can carry a maximum user data payload of 2112 bytes and has a header/trailer overhead of 36 bytes. FC-2 uses a look ahead flow control mechanism where each frame is acknowledged by the receiver. At the FC-2 layer, Fibre Channel defines three basic service classes:

- *Class 1* supports message switching via dedicated, full bandwidth N-port to N-port connections. An end-to-end dedicated path has to be reserved before data can be transferred. As shown in figure 10, depending on the length of the user data,

[6] The FC-2 frame is not to be confused with the previous use of the term frame in section 2.5.1 in the context of the synchronous transport mode.

connection set up and tear down overhead can significantly impact effective throughput. Class 1 is optimized for the fast transport of large blocks of data.

- *Class 2* and *Class 3* support the packet switching approach by providing a connectionless frame switched service. The main difference between classes2 and 3 is that class 2 guarantees delivery and provides acknowledgment of delivery (or in the case of congestion in the switch, returns a *busy* status so that the sender can retry), whereas class 3 provides no acknowledgement. Unlike class 1, there is no connection set up, and therefore no connection overhead.

The FC-3 layer defines functions that involve multiple N-ports, such as striping (increasing throughput by the use of multiple parallel N-port to N-port links for one logical connection) and multicasting. We will not describe the FC-4 layer which defines mappings of certain standard upper layer channel and networking protocols onto the underlying layers.

6.2.3 HiPPI event building tests

High rate, parallel event building with the message switching approach has been demonstrated in the VME environment using the HiPPI technology [15]. The demonstrator, shown in Figure 11, consisted of a non-blocking 8 x 8 HiPPI switch, three VME-HiPPI interface modules [31] acting as sources with three more as destinations, and a RAID [32] VME master running a UNIX-like real time operating system.

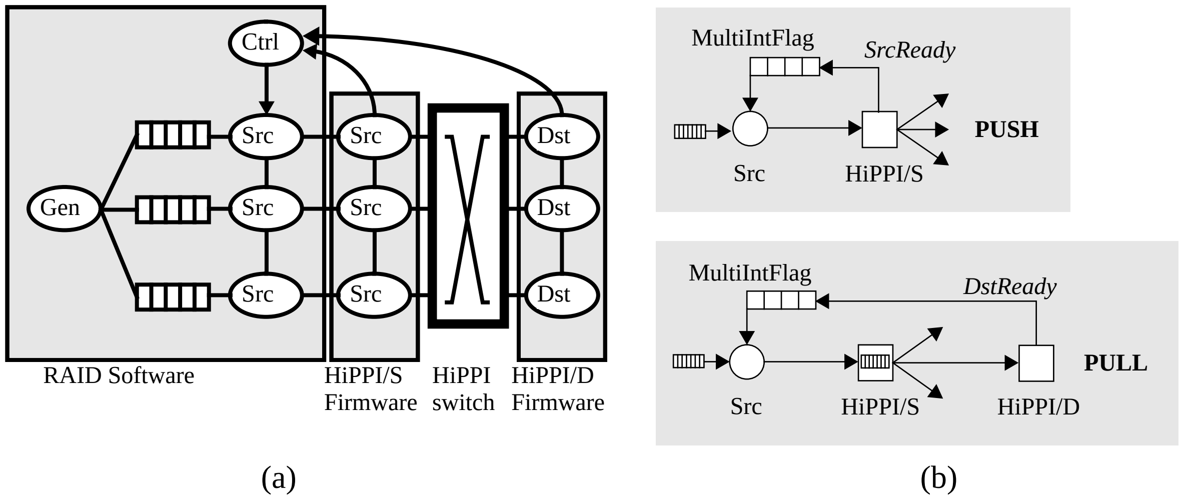


Figure 11: (a) Software and hardware components of the RD13 VME HiPPI-based event builder test bench; (b) the "push" and "pull" event building data flow scenarios.

The VME HiPPI source and destination modules include a 25 MHz RISC processor and a HiPPI link interface based on commercial chip sets. Stand-alone firmware running on the RISC implements the primitive operations needed to send and receive data over the HiPPI links. The *Src* tasks running on the RAID use the source and destination modules as HiPPI I/O servers. The *Gen* task, running on the RAID, generates event fragment sizes for the sources, and the*Ctrl* task assigns destinations and implements different event building data flow control scenarios; one "push" and two "pull". Client-server communication was via the VME bus; because of limited VMEbus bandwidth and the use of only one RAID, no event data was generated in the RAID or copied across VME into the sources; therefore the sources sent event fragments of the size generated in the RAID, but containing null data.

The minimum latency for a source to establish a connection and start data transfer was measured as ~50 µs and was dominated by the firmware overhead (the overhead for connection set up by the switch is less than 1 µs). Connection requests were queued if an output port was blocked by an already established connection. When all event fragments were of the same size and events were allocated to destinations in a round robin schedule, the aggregate throughput of the 3 x 3 event builder saturated at 120 MByte/s for large event fragments (memory speed and system bus bandwidth of the processor in the VME-HiPPI interface modules limited the sustainable HiPPI source transmit bandwidth to a maximum of 40 MByte/s). For small event fragments, the software and firmware overhead limits the load that can be applied to the switch.

Nevertheless, event handling rates of the order 10 kHz were achieved with event fragment sizes of the order 100 bytes (a typical Atlas level-2 extracted "feature" size). Taking into account that the RAID resources were shared by the client tasks of all the HiPPI sources and destinations, higher performance (20-30 kHz) should be achievable by migrating the client tasks into the RIO modules, or by using a faster processor.

When a random destination assignment algorithm was employed, or event fragment sizes were allowed to vary following different statistical distributions, connection requests had to queue in the switch for busy destinations to become free (blocking of switch output ports), and lower aggregate throughput resulted. More details can be found in [15].

The results confirm that this approach will be able to sustain the average rate of event fragment emission required in the ATLAS architecture. At the same time they demonstrate the important effect of overheads in limiting the maximum event rate that can be handled at the LHC, where average event fragment sizes are expected to be between 100 - 1000 bytes. The impact of overheads can be reduced in several ways:

- use faster processors;
- eliminate operating system and interrupt overheads by writing stand-alone software and avoiding the use of interrupts (see for example section 7.3);
- pack multiple event fragments into a "super-fragment" and build "super-events";
- use a custom hardware engine for the source and destination data flow control [11];

6.3 Discussion on event building with message switching technologies

The demonstrator results confirm that the performance required for the ATLAS architecture can be achieved with existing HiPPI adaptors and firmware running on modestly powerful processors. In principle, Fibre Channel class 1 could be used to perform event building in a very similar way. However, in comparison with the HiPPI switch used in the demonstrator, currently available Fibre Channel switches suffer from relatively large class 1 connection set up and tear down overheads (~100 µs) which, as indicated in figure 10 makes them inefficient for high rate event building with small event fragments.

If Fibre Channel switches are to be used in the CMS architecture, where event fragment switching must operate at 100 kHz, class 1 connection set up overheads in future 1 Gbit/s switches need to be reduced to around 1 µs, or better, to achieve efficient use of the nominal bandwidth. In view of the complexity of the Fibre Channel protocols and the fact that class1 is not intended for sending short messages, it remains to be seen whether Fibre Channel switch manufacturers will implement such low connection overheads. As mentioned before, a possible way around the problem is to pack multiple event fragments to form larger messages, but this adds additional processing overhead and latency that may not be acceptable for level-2 triggers.

An important question that should be answered before selecting a large message switching fabric (e.g. HiPPI or class 1 Fibre Channel) is whether connection set up handling is centralized or distributed. A centralized connection handling scheme would have to queue connection requests that arrive while it is busy handling a previous request. When the connection requests are randomized in time but the load is heavy, or when the connection requests from different ports are correlated (as for example they would be if the fabric was operated as a barrel shifter), the queuing may lead to a much increased average connection overhead, which in turn would degrade event builder throughput.

An upper limit to the event rate that can be built by an N x N switch with centralized (non-pipelined) connection request servicing is given by:

$$R < 1 /(T_c * N) \qquad (1)$$

where T_c = connection set up latency measured when the queue for connection set up requests is empty; e.g. for a switch using centralized connection handling with $T_c = 1$ µs and N = 256, we find a maximum event building rate of only 3.9 kHz.

Currently available Fibre Channel switches [33, 34] have been measured to have $T_c \sim 50 - 100$ µs, and they probably use a centralized software-driven connection set up mechanism; they are therefore likely to be far from achieving the required 100kHz performance. However, no measurements have been made on existing Fibre Channel switches to determine the maximum aggregate connect request rate that they can handle. The requirement for partitioning of the data acquisition system has lead to the proposal to use multiple, small, independent switches in ATLAS [16].

However, because very large commercial switches are not available, a large switch (e.g. for the CMS architecture) would have to be built by cascading smaller ones. A large non-blocking switch can be constructed using a multi-stage *Clos network* of smaller non-blocking switches. Figure 12(a) shows a two stage delta network which is blocking (as shown by the competing connections in bold). In the Clos network we add a third stage of switching to provide alternative paths and so make the network non-blocking as shown in figure 12(b).

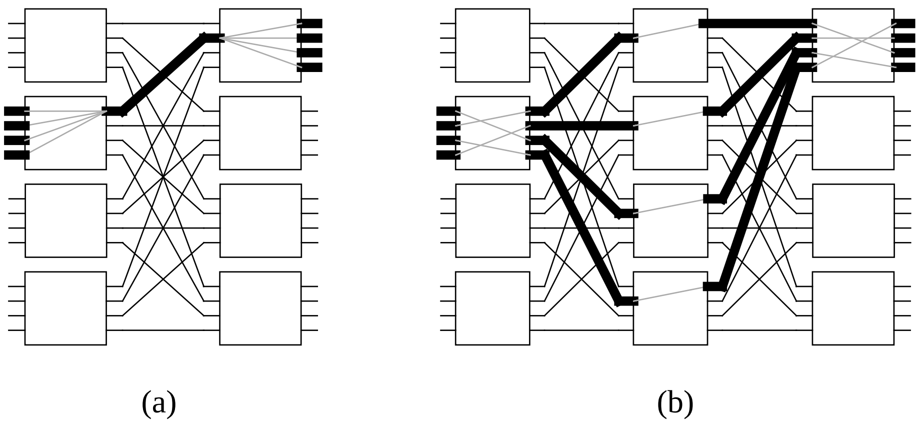


Figure 12: (a) a two-stage network suffers from blocking whenever connections compete for the same internal link; (b) the 3-stage Clos network is non-blocking.

The Clos network approach has been proposed [35] as a means of building large Fibre Channel event builders for the LHC using the existing commercial switches (which typically have up to 64 ports). With this approach the bottleneck for connection request handling is partially solved also, because individual switches do not have to handle the aggregate connection request rate of the whole event builder. On the other hand the connection set up process becomes more complex because individual switches need to communicate with each other in order to find a free path through the 3-stage network. The switch manufacturer offers connection set up software which allows this to be done with a latency which is just three times the latency of one stage. The complexity of multi-stage connection management suggests that it will continue to be implemented (mainly) in software in the future, implying that a centralized (or not fully distributed) mechanism will be used in each component switch, thereby limiting the ability to handle high trigger rates.

In short, if a message switching approach is to be used for event building, it will be important to select a technology and a switch architecture employing a very low latency, distributed connection set up mechanism. The issues around the cascading of HiPPI or Fibre Channel switches in event builders need more study.

7 Parallel Event Building using Packet Switching

As explained in sections 2.5.2 and 2.5.4, a packet switching technology allows the use of semi-permanent virtual connections (SPVCs) to provide bandwidth on demand. Figure13 shows how an event builder could be constructed in which SPVCs are defined at the start of a data taking run to fully interconnect all source and destination modules. Once created, these SPVCs would remain available until the end of the run. Since the SPVCs provide all required connectivity, the dynamic set up of connections and the associated overhead that was necessary for message switching is now eliminated. Fibre Channel class 2 or ATM cell switching are two possible technologies that could be used in this way. However, to the author’s knowledge, there is no native class 2 Fibre Channel switch presently available. We will illustrate the approach using ATM cell switching.

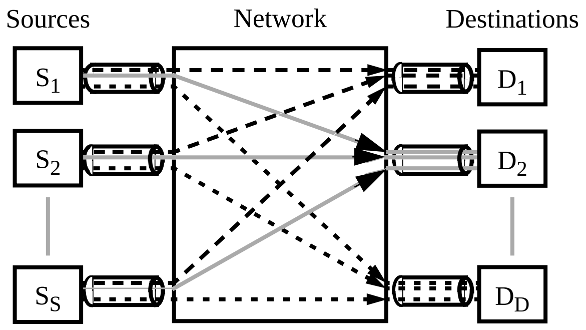


Figure 13: An event builder employing semi-permanent virtual connections to fully interconnect all sources and destinations.

7.1 The B-ISDN ATM protocols

The asynchronous transfer mode using 53-byte fixed-length cells (as introduced in section 2.5.2) has been adopted by the International Telecommunications Union (ITU) as the underlying packet switching technology to integrate all telecommunications services and specialized networks into one common, world-wide infrastructure, known as the Broadband Integrated Digital Services Network (B-ISDN). The ITU’s B-ISDN ATM standards [4] define an architecture that can scale with technology and is designed to be adaptable to efficiently meet the requirements of different applications (constant bit-rate and variable bit-rate services, with or without a real-time requirement). ATM has subsequently been actively developed by the computer industry, under the coordination of the ATM Forum [5], for high performance local area networking.

ATM is a connection-oriented, packet switched technology employing fixed-length cells made up of a 5 byte header and a 48 byte data payload. The ATM protocol defines three layers:

The *physical layer* specifies several options for the physical media, bit-rates and coding schemes. The preferred physical layer standards are SDH and SONET, as previously mentioned, which use the bit-rate hierarchy of 155.52, 622.08, 2,488.32 Mbit/s. Physical media include UTP-5 (category 5 unshielded twisted pairs for speeds up to 155 Mbit/s) and multimode and monomode fibre. Other functions of the physical layer include ATM cell header delineation and header error detection and correction.

The *ATM layer* defines the cell format and how cells are switched between links using the 8-bit *virtual path (VPI)* and 16-bit *virtual channel (VCI)* identifiers of the 5-byte cell header. The ATM layer also handles the mapping of VPI and VCI into the new values to be used on the outgoing link to identify the virtual connection to which the cell belongs. The 48 bytes of user data payload are not protected by error detection codes at this layer, but the header is protected by an 8-bit error detection and correction code (HEC). A 3-bit payload type identifier (PTI) distinguishes between cells that carry user data and cells that carry various types of control and management information used by the network itself. The *cell loss priority* bit (CLP) defines the priority to be used in dropping cells in a congested network.

The *ATM Adaptation Layer (AAL)* adapts the ATM layer to the requirements of specific classes of service. The ITU standards define 4 service classes (constant bit-rate and variable bit-rate services, with or without a realtime requirement) for different types of application. A specific adaptation layer protocol is defined for each service class. The AAL5 protocol is the most appropriate for event building applications. The AAL5 packet can be up to 64kByte in length, and is segmented into (and reassembled from) a stream of cells for the ATM layer. The packet trailer contains a length count and CRC check for error detection, but error recovery must be provided by the higher protocol layers.

The physical layer, ATM layer functions and most of the AAL5 layer functions are normally implemented in hardware. Figure 14 shows the maximum theoretical user data throughput using the ATM protocol over a semi-permanent virtual connection multiplexed on a 622 Mbit/s SDH link. The effective maximum user data throughput is 542 Mbit/s due to the combined effects of SDH framing overhead and ATM cell header overhead. The saw-tooth shaped line shows the variation of throughput due to the packetization in fixed length ATM cells. For comparison the figure also shows the throughput that is theoretically achievable with class 1 Fibre Channel for different values of the connection set up latency.

7.2 Using an ATM switching fabric for event building

Large ATM switches are constructed as multi-stage switching networks, typically built from 16 x 16 or 32 x 32 switching chips. A description of a typical multi-stage, multi-path, self-routing switching fabric can be found in [36].

7.2.1 Congestion in ATM switching fabrics

Multi-stage ATM switching fabrics route multiple virtual connections over each internal link. Cells travelling on different virtual connections therefore may contend for access to an internal

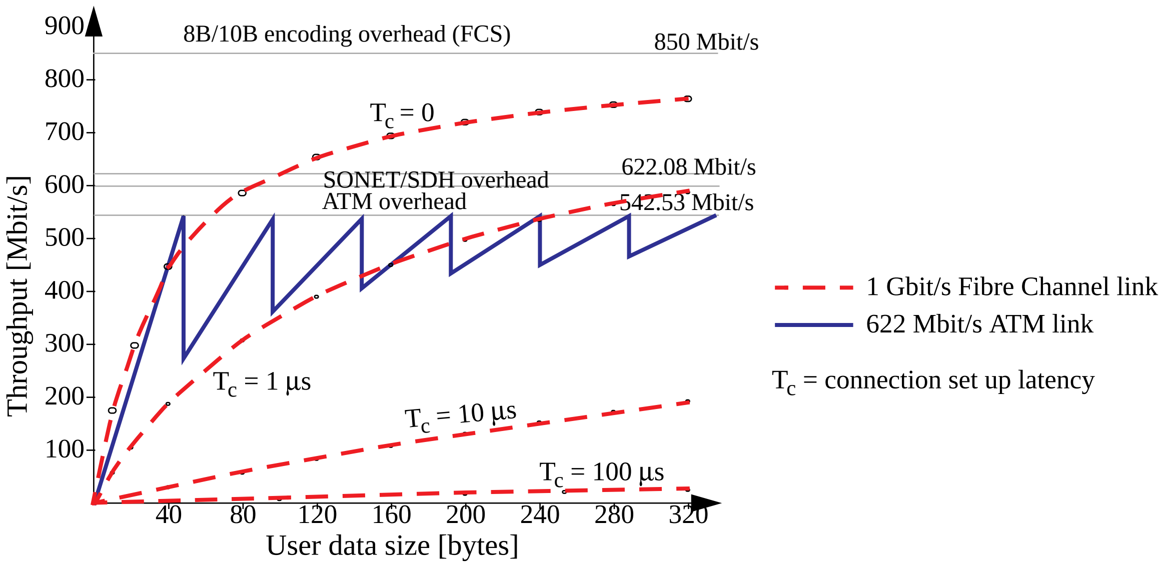


Figure 14: Maximum theoretical effective user data throughput as a function of message length for 622 Mbit/s ATM and 1062 Mbit/s Fibre Channel class 1.

link. Contention is resolved by queuing cells within each switching chip until the internal link is available. If the queue becomes full the switch is said to be congested.

There are two courses of action that can be adopted when congestion occurs. The first is to discard cells, the second is to use a hardware flow control protocol on the internal link to hold off the arrival of further cells until sufficient buffer space becomes available. The strategy of dropping cells is usually adopted in large switches designed for telecommunications infrastructure, whereas some switches intended for LAN applications use the link-level flow control strategy.

In telecommunications switches the aggregate traffic of a large number of independent subscribers is assumed to be random (i.e. cells have random destinations and arrive at random times). The internal buffers of the switching elements are dimensioned so that the probability of congestion occurring under this randomized traffic is acceptably small (typical cell loss probabilities are 10^{-10} or less).

7.2.2 Traffic shaping

The natural traffic patterns generated by a “push” architecture event builder concentrate ATM cells belonging to a given event towards the single destination assigned to that event. If sources inject the event fragments at full link speed, the internal buffers of the switching elements quickly fill up and acute congestion occurs.

In an event builder based on the telecommunications type of switch, the congestion causes severe cell loss and the event builder is unusable. In the case of a link-level flow controlled switch, the flow control prevents cell loss, but the simulation results in figure15(a) show that congestion causes the event building latency to grow non-linearly with the load on the switch [14]. When the switch is offered a load above a certain critical level, the throughput of the congested switch is insufficient to handle the offered load and queue lengths in the sources grow without limit. Better event builder performance could be achieved by minimizing congestion.

Internal congestion can be minimized (or even eliminated) by using the technique of *traffic shaping*. As shown in figure 15(b), for the flow controlled switch, traffic shaping

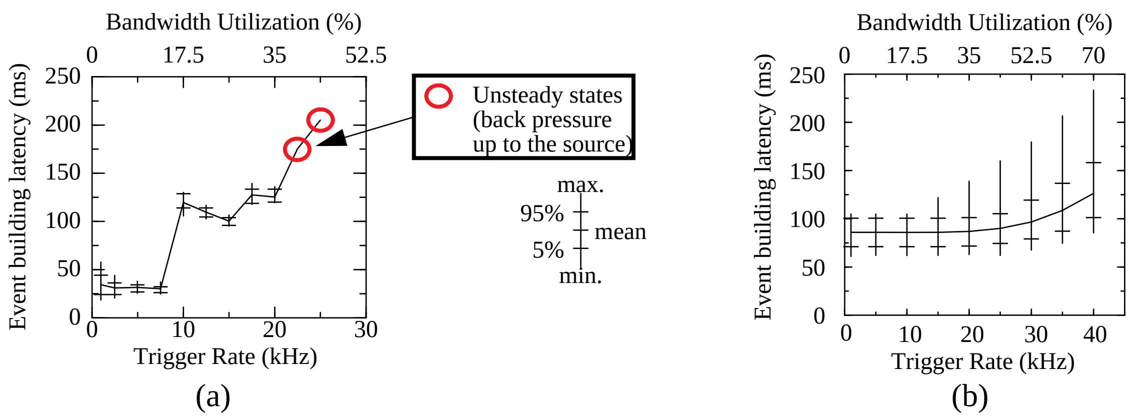


Figure 15: Event building latency as a function of trigger rate for a 1024 x 1024 event builder using link bit-rates of 640 Mbit/s with an average event size of 1 MByte; (a) for the case of a flow controlled switch without traffic shaping; (b) for the same switch with traffic shaping.

improves the shape of the event builder's latency versus load curve and allows the event builder to operate at loads of up to 70-80% typically. In the case of a telecommunications switch, the traffic shaping techniques bring the probability of losing cells down to the range for randomized traffic (10^{-10}) or even better, and in addition they give a similarly improved latency versus load profile. In fact, traffic shaping makes the performance of the event builder independent of the details of the internal switch architecture. Under these conditions the behaviour of the event builder (latency, queue lengths in sources and destinations etc.) is determined by the form of the statistical distribution of event fragment sizes.

Traffic shaping consists of modulating the traffic emitted by a source in two aspects; the first consists of controlling the *rate* at which cells are transmitted on individual virtual connections, and the second involves control of *time correlations* between traffic on different virtual connections. In an N x N event builder, rate division by a factor N must be applied to each virtual connection so that, when the individual cell streams merge towards the destination, their aggregate traffic will not exceed the available bandwidth at the output port.

Rate division is necessary, but not sufficient to avoid congestion. Due to the synchronization of the sources with the trigger, each of the sources may emit a cell to the destination at the same moment, causing "wavefronts" of cells to converge towards the destination. In a large (say 1000 x 1000) event builder, 1000 cells will arrive almost simultaneously at the last switching element before the output port. Since the switching element typically has an internal buffer of a few kByte, it is unable to absorb the wave of 1000 cells (53 kByte). Several proposals to smooth out the "wavefronts" by changing the time correlation between cell traffic at the sources have been simulated [36]. One technique adds a small random delay to the injection time of each cell, and another approach correlates emission of the cell streams from different sources to form a synchronous barrel shifter in which one cell is emitted per time slot.

Rate control on individual virtual connections is supported in all ATM interface chip sets, but features for randomizing cell injection times or organizing a set of interfaces into a cell-based barrel shifter are not generally supported. This approach therefore requires additional customized interface hardware, excluding the use of commercial interface boards. A third traffic shaping proposal (suitable for commercial interfaces) emulates a circuit switched barrel shifter [37] by using virtual connections without rate division and performing time slot switching in software (synchronized by a global interrupt).

Simulation of a “pull” strategy event builder, based a single flow controlled ATM switch used to handle both the level-2 and level-3 traffic in ATLAS has given good results [38]. Because of the local/global level-2 architecture the level-2 traffic can be handled, with acceptable latencies up to the full 100 kHz level-1 rate, without requiring the use of traffic shaping. For the level-3 traffic it was found to be sufficient to apply traffic shaping based on rate division only (which is simple to implement).

Another traffic shaping scheme chains the emission of event fragments from sources by circulating a “send event fragment” token among the sources [19]. Sources only transmit the corresponding event’s fragment when they have the token. Several tokens can be in circulation so that events can be built concurrently in different destinations.

7.3 An ATM-based event building demonstrator

Figure 16 shows a VME-based parallel event building demonstrator based on a 8-port, 155 Mbit/s ATM switch [39] of the telecommunications type (no link-level flow control). Custom-designed VME-ATM interface modules [40] are used as sources and destinations. They are based on a commercial RISC I/O (RIO) VME board [41], on which the AAL5, ATM and physical layers of the protocol are implemented with commercial chip sets [42, 43]. The event building protocol layers are implemented in software in the on-board RISC. The physical layer protocol used by the switch is the 155 Mbit/s SDH protocol over monomode fibre.

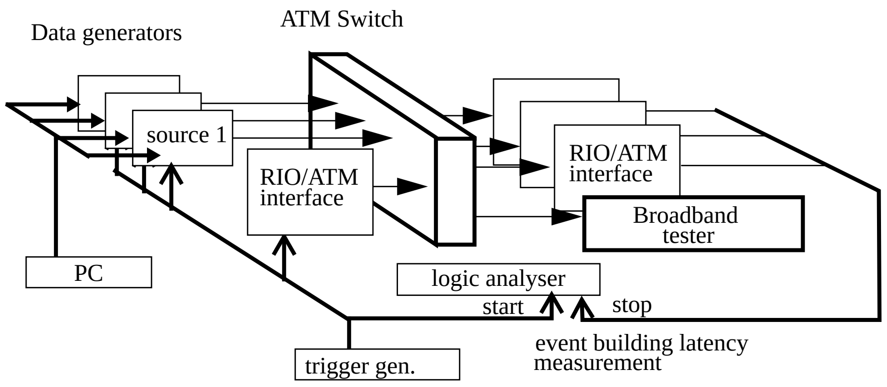


Figure 16: The ATM-based event building demonstrator.

A logic state analyser and a broadband tester [44] are used for debugging, analysis and performance measurements. In order to be able to load the switch with traffic on multiple inputs, a number of low cost ATM data generator [45] modules, controlled by a personal computer, have been constructed. The data generator is a memory in which any desired pattern of cells can be loaded and sent out via a physical layer interface. The desired cell pattern is calculated off-line and downloaded to the data generator.

7.3.1 Event building protocols

Figure 17 shows the event building protocol stacks in source and destination. The event building software is implemented in two layers; the upper *event layer* is technology independent and adapted to the underlying ATM technology by the *event fragment layer.* The software runs on the 25 MHz RISC processor and is layered directly onto the chipset implementations of the AAL5, ATM and SDH physical layers. For performance the software is implemented without operating system and uses semaphore polling in preference to interrupts (interrupts are only used for error conditions). Because the software is distributed, its performance scales to very large event builder systems.

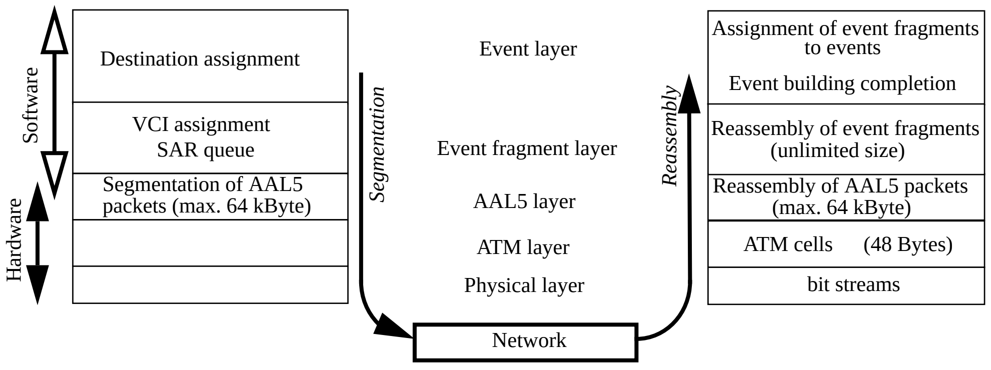

Software
Hardware
Destination assignment
VCI assignment
SAR queue
Segmentation of AAL5 packets (max. 64 kByte)
Segmentation
Event layer
Event fragment layer
AAL5 layer
ATM layer
Physical layer
Network
Reassembly
Assignment of event fragments to events
Event building completion
Reassembly of event fragments (unlimited size)
Reassembly of AAL5 packets (max. 64 kByte)
ATM cells (48 Bytes)
bit streams

Figure 17: The event building software is implemented in two layers above the hardware-supported ATM protocol stack.

In the source (destination), the event fragment layer performs fragmentation (reassembly) of event fragments into (from) AAL5 packets. In order to minimize the required receive buffer space allocation in the destinations, the length of theAAL5 packets used can be chosen close to the average event fragment size (up to the 64 kByte limit supported by the ATM standard). In the destination, the event layer reassembles events from the received fragments. A number of algorithms for detection of event completion have been proposed [46]. The event completion algorithm used to make the measurements reported in the next section was based on the timeout principle.

7.3.2 Performance measurements

Figure 18(a) shows the measured maximum output of a source when pushing out event fragments with the software described above. The dotted line is the measured output including ATM cell headers and incompletely used cell payloads; it drops for small event fragment sizes because of various hardware and software overheads [47]. The dashed line shows user data throughput, and the solid line shows the maximum frequency of emitting event fragments. The source software could support high trigger rates (~80 kHz) with small event fragments (< 2 cells), but software overheads limit the efficiency of utilization of the bandwidth (~25%) in this case.

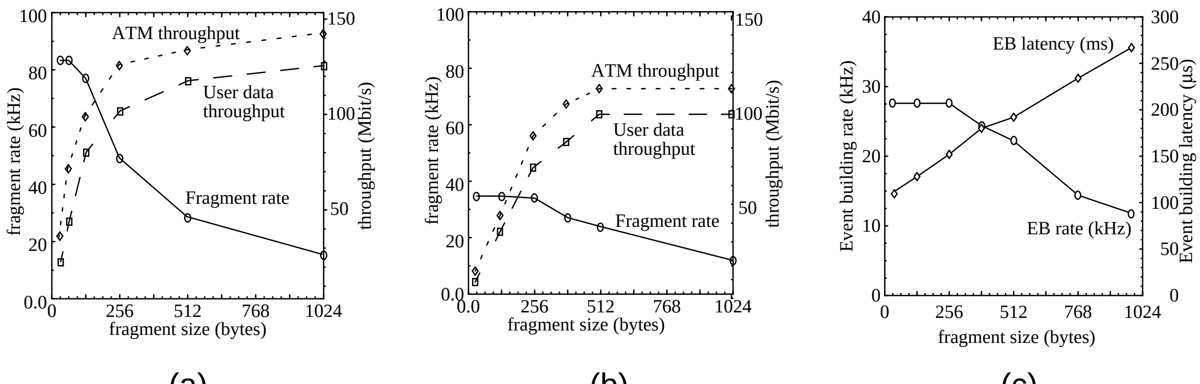


Figure 18: Performance measurements; (a) for the source pushing fragments at maximum speed; (b) for the destination receiving fragments from 8 sources and building 2 events simultaneously using the timeout algorithm; (c) measured event building latency and rate for a 2 x 2 event builder.

The corresponding measurements for the reception of event fragments, shown in figure 18(b), exhibit less throughput at larger event fragment sizes due to a bottleneck in the current design of the receiver (which can easily be removed). The more complex receiver software makes a larger impact at small event fragment sizes, limiting the maximum fragment reception rate to ~30 kHz

Figure 18(c) shows the measured event building latency and rate obtained in a 2x 2 event builder in which each source sends 1 cell alternately to each destination. For small event fragment sizes, the event building software in the destinations limits performance, while for larger fragments it is the data transfer on the links that limits the performance. If required, higher performance can be achieved by using faster processors, more destinations than sources, higher link bit-rates, or collecting data from several events into super-fragments which are then used to build super-events.

8 Conclusions

The size and required performance of the data acquisition and software triggering systems at the future LHC experiments exceed by far that of previous experiments. This paper has reviewed some of the on-going R&D into technologies and techniques that appear to be suitable for parallel event building at LHC. Architectural options, such as the choice of trigger strategies and algorithms, the use of phased event building and intelligent selective readout can reduce the required data movement bandwidth. We have seen that, in order to use the full potential of high speed link and switch technologies, careful attention has to be paid to details.

First, only a fraction of the link bandwidth is available to carry user data, the remainder being consumed by encoding schemes, and packetization overheads at each level of the protocol stack. Message switching technologies have a hardware connection set up overhead that, depending on its magnitude, may significantly impact the throughput achieved when sending short messages. The aggregate connection set up performance of such switches must be carefully evaluated for the event building application envisaged (e.g. centralized handling of connection requests would be inappropriate for implementing a synchronous barrel shifter, where all connections must be switched simultaneously). Packet switching technologies offer semi-permanent virtual connections, which avoid the connection set up problems.

In a push architecture, the concentration of event fragments from multiple sources into the destination leads to output blocking or internal congestion in the switch. Depending on the technology, the result is either that throughput is reduced and event building latency increased, or that data are discarded. These problems can be circumvented by an appropriate choice of destination assignment algorithm and by using traffic shaping. The traffic shaping technique is mandatory when using an ATM switch without flow control, but it is also beneficial with all technologies, allowing operation of large switches at high loads with still acceptable latencies.

Software overheads can easily dominate system performance and mask the advantages of high speed link and switch hardware, as we saw from the measurements made using the standard TCP/IP communications protocols in a UNIX operating system environment. The performance penalty of redundant protocol layers, unused functionality and operating system overheads can be avoided by developing optimized event building software layered directly onto the network adaptor hardware. Results from HiPPI and ATM (155 Mbit/s) event building demonstrator systems have shown that software-limited event building rates in the range 10-30 kHz can be achieved with 25 MHz processors running software partly, or completely, in stand-alone mode, and avoiding interrupts. A combination of faster processors and higher throughput links (in the case of ATM) should permit still higher event building rates to be achieved.

Acknowledgements

I am most grateful to J.-P. Dufey for suggesting numerous improvements to the paper and assistance with the preparation of some of the figures. In addition, it is a pleasure to thank the many colleagues who provided me with information and/or enlightening discussions during the preparation of this paper; D. Abbott, E. Barsotti, E. van der Bij, F. Chantemargue, S. Cittolin, M. Costa, J.-P. Dufey, T. Ferrari, G. Heyes, I. Mandjavidze, L. Mapelli, B. Martin, R. McLaren, S. Munoz, M. Nomachi, B. Segal, R. Spiwoks, A. van Praag.

References

[1] RD24 collaboration, “Application of the Scalable Coherent Interface to Data Acquisition at LHC”, *CERN / DRDC* 91-45, *DRDC* 93-20, *DRDC* 94-23 and *LHCC* 95-42.

[2] M. Santifaller, *TCP/IP and NFS; Internetworking in a UNIX Environment*, Addison-Wesley (1991).

[3] International Telecommunications Union, Geneva, Switzerland, *Recommendation G.703.*

[4] International Telecommunications Union, Geneva, Switzerland, *Recommendations I.150, I.211, I.311, I.321, I.327, I.361, I.362, I.363, I.413, I.432, I.610.*

[5] The ATM Forum, 303 Vintage Park Drive, Foster City, CA 94404, USA.
World Wide Web home page: http://www.atmforum.com/

[6] International Telecommunications Union, Geneva, Switzerland, *Recommendations G.707, G.708* and *G.709.*

[7] ANSI T1.105-1991, *Digital hierarchy - Optical interface rates and formats specifications (SONET).*

[8] ANSI X3T9.3/90-043, HiPPI Specification (1990).

[9] ANSI X3T11 committee, Fibre Channel draft proposed standard.

[10] CMS collaboration, “The Compact Muon Solenoid - Technical Proposal”,
CERN / LHCC 94-38, *LHCC / P1*, 15 December 1994.

[11] S. Cittolin (convenor), record of the LHC DAQ Workshop on FrontEnd Driver and Dual Port Memories, CERN, March 29-30, 1995.

[12] ATLAS collaboration, “ATLAS - Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN”, *CERN / LHCC* 94-43, *LHCC / P2*, 15 December 1994.

[13] J.-C. Brisson, O. Gachelin and M. Mur, “Front-end Intelligent Readout Memory”, private communication.

[14] RD31 collaboration, “NEBULAS: High performance data-driven event building architectures based on asynchronous self-routing packet switching networks”, *CERN LHCC /* 95-14.

[15] RD13 collaboration, “A scalable data taking system at a test beam for LHC”, *CERN LHCC* 95-47.

[16] W. Greiman et al., “Design and Simulation of Fibre Channel Based Event Builders”, Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[17] R. W. Dobinson et al., “Realisation of a 1000-Node High Speed Packet Switching Network”, *CERN-ECP /* 95-16 (30 August 1995).

[18] D.J. Abbott et al., “The CODA System and its Performance on the First On-line Experiments at CEBAF”, presented at the Ninth Conference on Real-time Computer Applications in Nuclear, Particle and Plasma Physics, Michigan State University, USA (May 22-25, 1995). Submitted to *IEEE Trans. Nucl. Sc.*

[19] D. C. Doughty et al., “Event Building Using an ATM Switching Network in the CLAS Detector at CEBAF”, Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[20] P. Sphicas, “CDF and D0 Data Acquisition Systems”, Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[21] F. Chantemargue et al. (RD11 collaboration), private communication; a similar set of measurements are reported in: C. Miron et al., “Fibre Channel Performances with IBM Equipment”, *RD11-EAST note* 95-05 (June 1995).

[22] C. Battista et al., “Permanent Virtual Circuits Configuration and TCP-UDP / IP Performance in a Local ATM Network”, *INFN internal report no. 1060*, submitted to *Computer Networks and ISDN Systems*. (http://www.infn.it/pub/CNAF/article.ps)

[23] N. Antoniu et al., “A Large Ion Collider Experiment at the CERN Large Hadron Collider”, *CERN LHCC /* 93-16, March, 1993.

[24] D. Black et al., “Results from a Data Acquisition System Prototype Project using a Switch-based Event Builder”, Nucl. Sc. Symposium, Santa Fe, New Mexico (USA), Conf. record, Vol. **2**, pp. 833-837 (1991).

[25] O. Sasaki et al., “A VME Barrel Shifter System for Event Reconstruction for up to 3 Gbps Signal Trains”, *IEEE Trans. Nucl. Sc.*,Vol. **40**, No. **4**, pp. 603-606 (1993).

[26] Y. Nagasaka et al., “Global Traffic Control System of a Switching Network”, presented at the Ninth Conference on Real-time Computer Applications in Nuclear, Particle and Plasma Physics, Michigan State University, USA (May 22-25, 1995). Submitted to *IEEE Trans. Nucl. Sc.*

[27] U. Behrens et al., “The Event Builder of the ZEUS Experiment”, Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[28] R. Heeley et al., “The Application of the T9000 Transputer to the CPLEAR Experiment at CERN”, *CERN / ECP* 95-8 (30 March 1995). Submitted to *Nucl. Instr. and Meth*.

[29] D. Francis et al., “Triggering and Event Building Results using the C104 Packet Routing Chip”, presented at the Conf. on Computing in High Energy Physics, Rio de Janeiro, Brazil, 18-22 September, 1995 (to appear in Conf. proceedings).

[30] E. Kristiansen et al., “Switches for Point-to-Point Links using OMI/HIC Technology”, Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[31] A. van Praag et al., “HiPPI Developments for CERN Experiments”, *IEEE Trans. Nucl. Sc.*, Vol **39**, No. **4**, pp. 880-885 (1992).

[32] Creative Electronic Systems S.A., Geneva, Switzerland, *RAID 8235 VME RISC Processor Board User's Manual,* 1992.

[33] The CXT 250 Fibre Channel fabric, Ancor Communications Inc., Minnetonka, MN 55343 (USA).

[34] The IBM 7319 model 100 Fibre Channel Switch, IBM Corp.

[35] C. Jurgens, "Fibre Channel Switches", Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[36] M. Letheren et al., "An Asynchronous Data-Driven Event Building Scheme based on ATM Switching Fabrics", *IEEE Trans. Nucl. Sc*., Vol. **41**, No. **1**, pp. 257-266 (1994).

[37] I. Mandjavidze, "A New Traffic Shaping Scheme: the True Barrel Shifter", *RD31 note* 94-03.

[38] D. Calvet et al., "A Study of Performance Issues of the ATLAS Event Selection System Based on an ATM Switching Network", presented at the Ninth Conference on Real-time Computer Applications in Nuclear, Particle and Plasma Physics, Michigan State University, USA (May 22-25, 1995). Submitted to *IEEE Trans. Nucl. Sc*.

[39] M. Henrion et al., "Technology, Distributed Control and Performance of a Multipath Self-Routing Switch", Proc. XIV Int. Switching Symposium, Yokohama, Japan, Vol.**2**, pp. 2-6 (Oct. 1992).

[40] L. Gustafsson et al., "A 155 Mbit/s VME to ATM Interface with Special Features for Event Building Applications based on ATM Switching Fabrics", Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[41] Creative Electronic Systems S.A., Geneva, Switzerland, *RIO 8260 and MIO 8261 RISC I/O Processors - User's Manual*, version 1.1 (March 1993).

[42] Transwitch Corp., Shelton, Connecticut, USA, *SARA Chip Set - Technical manual version 2.0* (Oct. 1992).

[43] PMC-Sierra Inc., *The PMC5345 Saturn User Network Interface Manual* (May 1993).

[44] Hewlett Packard, Broadband Series Test System (1994).

[45] C. Paillard, "An STS-OC3 SONET / STM-1 SDH ATM Physical Layer Implementation and Application to an ATM Data Generator", *RD31 note* 95-04 (Feb. 1995).

[46] I. Mandjavidze, "Software Protocols for Event Builder Switching Networks", Conf. record of the International Data Acquisition Conference, Fermi National Accelerator Laboratory, Batavia, Illinois, USA (October 26-28, 1994).

[47] M. Costa et al., "An ATM-based Event Builder Test System", presented at the First Workshop on Electronics for LHC Experiments, 11-15 September 1995, Lisbon, Portugal (to appear in Conf. proceedings).