A SOFTWARE ARCHITECTURE FOR CLIENT-SERVER TELEMETRY DATA ANALYSIS

Douglas M. Brockett
Nancy J. Aramaki

BBN Systems & Technologies

ABSTRACT

An increasing need among telemetry data analysts for new mechanisms for efficient access to high-speed data in distributed environments has led BBN to develop a new architecture for data analysis. The data sets of concern can be from either real-time or post-test sources. This architecture consists of an expandable suite of tools based upon a data distribution software "backbone" which allows the interchange of high volume data streams among server processes and client workstations. One benefit of this architecture is that it allows one to assemble software systems from a set of off-the-shelf, interoperable software modules. This modularity and interoperability allows these systems to be configurable and customizable, while requiring little applications programming by the system integrator.

KEY WORDS

Client-Server, Distributed Computing, Data Analysis, Data Access

INTRODUCTION

Because of an increasing need among telemetry data analysts for new mechanisms for efficient access to high-volume, real-time and post-test data, BBN has developed the Data Exchange software architecture for efficient use of networked computing resources. The Data Exchange architecture is designed to manage and support the interchange of high-volume data streams among parallel and distributed data analysis software tools, especially in client-server applications.

One benefit of the Data Exchange architecture is that it allows the assembly of software systems from a set of interoperable software modules. This modularity and interoperability allows Data Exchange-based systems to be configurable and customizable. End users can write their own modules to perform custom processing which build upon the capabilities of third party Commercial Off-The-Shelf (COTS) tools.

BBN has used the Data Exchange architecture and custom tool modules extensively over the last three years, and COTS tools based on this technology are currently in development. This paper describes the features of a set of modules which include tools for data access (real-time and post-test), data archiving, and data analysis.

Application of a COTS tool set technology to a data processing system could result in significant software cost reduction by providing mechanisms to construct a semi-custom system which maximizes the use of off-the-shelf software. Further, because the modular nature of a client-server framework encourages code reuse, development time of custom tools can often be reduced, resulting in a solution which is flexible and quickly created. Finally, since custom tools can interoperate with COTS modules, their functionality can be leveraged by the infrastructure provided by the COTS components.

THE DATA EXCHANGE ARCHITECTURE

When the Data Exchange is used both as the basis for COTS tool sets and for user developed software modules, these modules become interoperable. The Data Exchange architecture provides data-source management, data-stream fusion, data extraction, and data buffering, all in a modular, scalable framework. Because the architecture is open and modular, it is easy to add new software and/or hardware components to increase functionality.

The Data Exchange software architecture is implemented as a software library which handles the creation and use of both real-time and non-real-time data streams. Programs which use this toolbox can send and receive high-volume data to and from other processes transparently. Because the Data Exchange Application Programming Interface (API) is very high level, it relieves the programmer from having to manage interprocess communication. Also, because the Data Exchange supports both distributed and parallel computing environments, and operates under multiple operating systems, one can use low-cost, high-performance, networked workstations as part of an analysis system. This architecture has been implemented under the UNIX, Mach, pSOS, and VMS operating systems.

The parallel and/or distributed structure employed by Data Exchange-based system offers some clear advantages in real-time data processing. First, the ability to dedicate multiple processors to data stream processing can enable such a system to reduce the latency of data available to downstream modules. Second, the number of data streams coming into the system and the complexity of processing of a given stream can be expanded by adding additional processors to the system. A further benefit of the modularity and scalability of a Data Exchange based system is its ability to support multiple simultaneous users. There is no architectural limit to the number of users. To date, systems have been built which support up to 8 analyst workstations concurrently.

Data Exchange Objects

The Data Exchange distinguishes between senders (or sources) and receivers (or sinks) of data. Data Exchange sender and receiver objects are at the top of the Data Exchange object hierarchy. A sender of data might be a real-time data source (such as a 1553 bus monitoring program), a process reading back archived data, or the output of a simulation process. A receiver of data might be a data analysis program, a data archiving process, or the input to a simulation.

Each sender or receiver can support multiple sub-objects called streams. Streams are a series of time-tagged data frames. No restriction is placed on the type of data contained in the frames. For instance, frames may be PCM data frames, 1553 bus messages, or video image buffers. Sender streams can send (or publish) as much data as they wish. Receiver streams receive only the subset of data frames they require.

Receivers specify the data needed for each receiver stream via objects called selectors. Selectors specify the source type, frame type, and time period of interest for each receiver stream. Once the selectors are specified for a receiver stream, that stream receives only the data which matches the specifications in the selectors. This has the effect of filtering the data which reaches the receiver, increasing receiver efficiency, and reducing the communications load on the system. This is particularly important in networked client-server systems. As data is received into a receiving stream from any number of senders, the data is merged into a single, time-ordered series of data frames.

Fan-in and Fan-out Capabilities

Receivers can receive data from as many different senders as they wish. Similarly, senders can send data to as many receivers as they wish. This allows new functionality to be added to a system easily, since new data sources or new receivers are automatically "plug-and-play" with the existing code.

The availability of fan-in and fan-out of data allows arbitrary numbers of data senders and receivers to be added in a modular fashion. The ability of multiple clients to access the same real-time data simultaneously permits, for instance, numerous analyst workstations, each performing different analyses of data from the same sources. The ability of a single client to merge multiple sources into a single stream allows, for example, an archiving module to create a real-time archive containing data from many sources. Figure 1 shows an example of data "fanning-in" from several senders to a single receiver. Data is being filtered by selectors as it flows in to the receiving stream. Fan-out from a single source to multiple receivers works in a very similar way.

In the current implementation of the Data Exchange software library, the fan-in capability of any one receiving stream is limited to those senders which originate on a single (although potentially remote) computer. They may also originate on different CPU's of the same multiprocessor. Any single receiver module, however may have multiple receiving streams, each coming from a different computer, making this a rather small limitation.

Real Time Processing Capabilities

Data Exchange senders can either be "request-driven", or "free-running". Request driven senders wait for a receiver to ask for data, and then respond to that request, using a handshake protocol to ensure that the receiving module receives every data block which is sent. In free-running mode, senders publish data blocks as quickly as they are acquired into a circular data buffer, without waiting for control flow information from the receivers. In this way, receiving modules are assured of having access to the most recent data. Free-running senders are most appropriate for use in real time systems.

In any real-time system, if a data source is consistently sending data faster than a data sink receives it, data will eventually be lost if the test runs long enough. In the case of the free-running Data Exchange sources, since the sender module does not wait for the receiving modules, the receiving modules could fall behind real-time, and possibly lose some data blocks if their processing load is too heavy. The Data Exchange allows system designers to minimize the probability of real-time data loss by maintaining a time history buffer of a specifiable size for each data source. This is particularly effective for "bursty" data sources.

Data Exchange Network Operation

Support for client workstations is provided through high-level Remote Procedure Calls (RPCs) via the TCP/IP network protocol. The same API is used for both remote and local Data Exchange data source access. Clients wishing to receive data "subscribe" to any subset of the available data streams. Subsequent calls to receive data will return the currently available data from those sources. Clients can subscribe to multiple sources simultaneously, and data can also be "fanned-out" from a single source to multiple clients. Fan out of real-time data to clients does not use the UDP network broadcast protocol, but rather sets up multiple TCP links, one for each client. This serves two purposes. First, unlike UDP, TCP guarantees reliable, ordered data delivery to client computers. Second, having a separate TCP connection permits each client to maintain a separate position context in the time-series stream, allowing that client to process data at its own rate.

THE TOOL SET PHILOSOPHY

Members of the Data Exchange tool set are designed to be "plug and play" software modules. Each module works with the other members of the tool set, sharing data with one another using the Data Exchange protocol. This protocol, implemented in the Data Exchange library, enables software modules to send and receive time-series streams of data blocks. These blocks may be traditional telemetry frames, images, or even database records. Tool set modules can send and receive data to and from many other modules simultaneously.

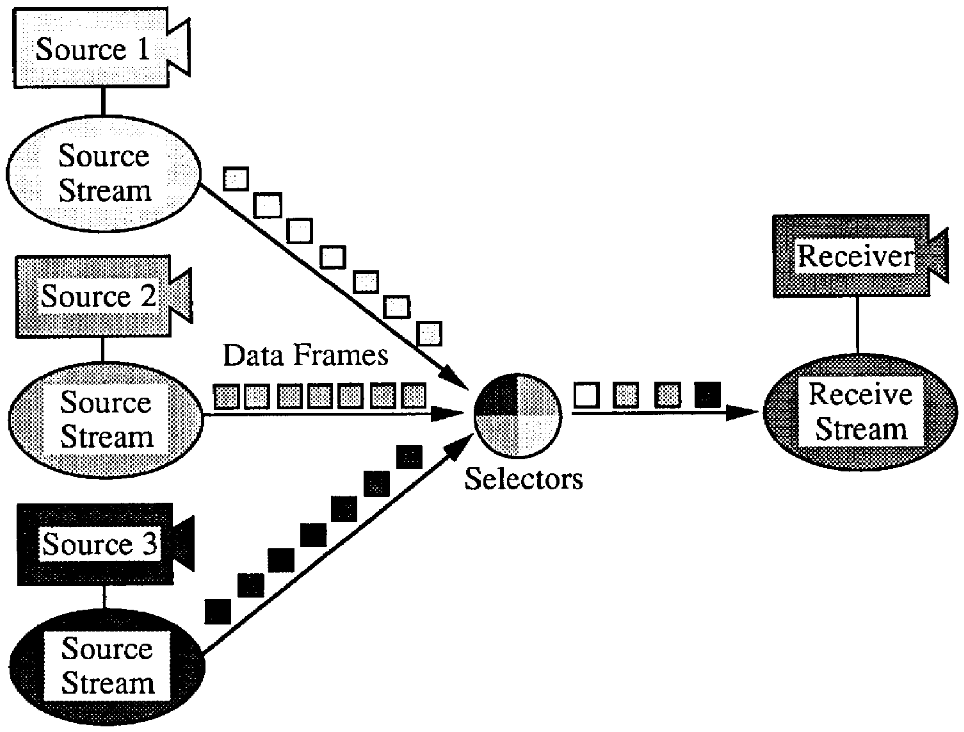

Figure 1. Fan-in and fan-out of data streams are important features of the Data Exchange architecture. Here, as data "fans-in" from several senders to one receiver, selectors filter the data frames, which are merged into a single, time-monotonic stream.

The flow of data in some modules is unidirectional. A data acquisition tool, for instance, might acquire data from a piece of hardware, and send it to other modules. This tool would be known as a Data Exchange "sender". A data archiver is known as a Data Exchange "receiver" because it is unidirectional in the opposite sense, i.e., it receives data from other modules and writes it to a storage medium. Other modules, of course, might both send and receive data; these tools are called "transceivers."

When a Data Exchange tool needs to receive data of a particular sort, it makes a request using the Data Exchange protocol for that type of data. If a sender of that type is present, the data it sends is routed to the requesting Data Exchange tool.

While it is impossible to enumerate or even imagine the set of tools which would accommodate every application, it seems clear that many applications would benefit from the ability to mix the following COTS modules with their custom modules. The modules described below and shown in Figure 2 represent a segmentation of the most important portions of many end-to-end telemetry processing systems.

Analysis and Display Modules

The ultimate goal of most telemetry processing systems is the presentation and analysis of data. This may mean real-time graphics, tabulations, function evaluation, alarm generation, et cetera. In a post-test environment, the analysis tasks often become even more demanding. The Data Exchange architecture allows developers to create powerful analysis and display software modules without having to expend enormous amounts of effort to duplicate the data access capabilities which can be provided by reusable data acquisition modules.

Sophisticated Event Detection

As the systems under test become more sophisticated and as the available telemetry bandwidth increases, many analysts find themselves overwhelmed by the amount of data that needs monitoring. More sophisticated event-detection modules are required to monitor large numbers of signals in an intelligent way, including using "waveform-parsing" based techniques. The Data Exchange architecture facilitates the creation of such tools by allowing high bandwidth access to large numbers of data streams. Further, multiple event detection modules share the same data sources while running on different networked computers, each one of which can monitor a set of signals of interest to its operator.

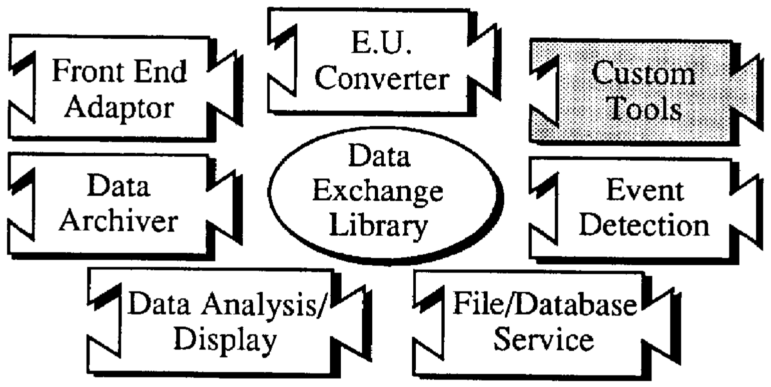

Figure 2. The Data Exchange tool set is a group of powerful, interoperable data tools.

File and Database Service

In the case of post-test operation, code to read or reformat data files is often duplicated among multiple data processing modules. By connecting a single file or database access module to the Data Exchange, the need to duplicate this code can be eliminated. Additionally, in situations where multiple modules require access to the same data, file caching and buffering performance benefits may be realized by referring all of their data access requests to single file server module.

Engineering Unit Conversion

When the data to be accessed is not available in engineering units, code must often be written to process and generate parameter values from raw data (e.g. telemetry frames). The conversion information used for this processes is usually accessible in some form of database. This tool converts raw data frames into native format engineering unit values, using the information provided by such a database. These values can then be used by other modules, including, for instance, custom display or analysis tools.

Data Archiving

Because many telemetry processing systems have specific data output requirements, a useful data archiving tool must not only produce a wide range of standard formats, but must support user-defined formats as well. A Data Exchange compliant archiver has simultaneous, uniform, high-bandwidth access to disparate data sources through a unified API. This module could produce time synchronized merged and filtered archive files in user-specified format from multiple data sources in real-time.

Front End Support

To provide access to real-time data, interface modules must be created for common data acquisition devices. Once such an interface is created, data from such a device is able to take advantage of the available downstream processing tools. This alleviates the need to provide custom processing software for each piece of data acquisition hardware. Since many acquisition devices appear similar from a software collection viewpoint, creating an interface module for any particular device is often a simple process.

Figure 3 shows an example of how a set of Data Exchange based tools might interact, and demonstrates the "fan-in" and "fan-out" capability. Some of the front end adapters publish data which is consumed by multiple sinks while some of the sinks consume data from multiple sources. Also, both real-time and post-test data sources are active.

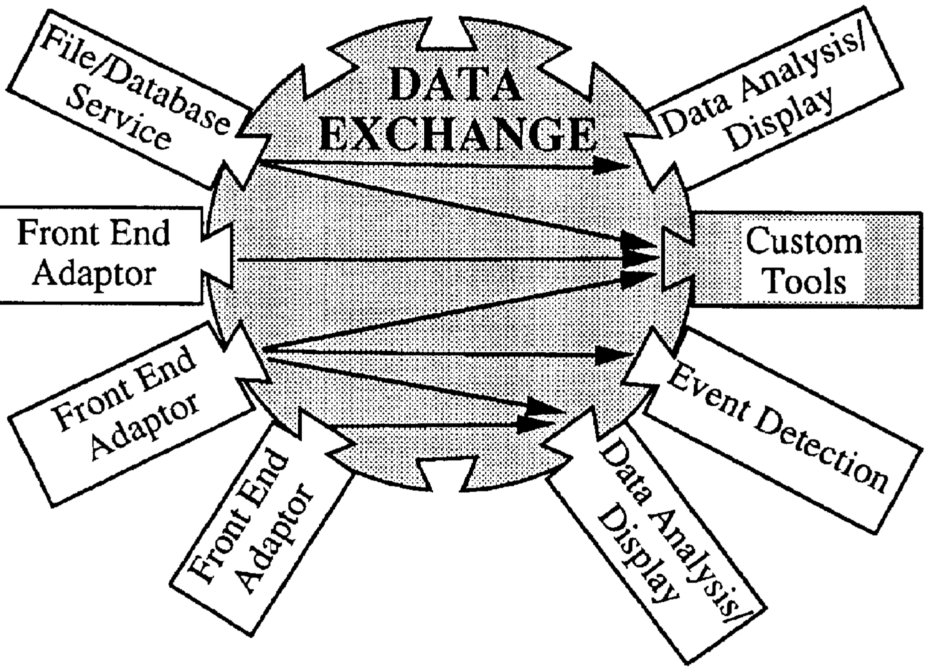

Figure 3. Information from Data Exchange senders can 'fan-out" to many receivers, and data from many senders can 'fan-in" to a single receiver.

TOOL SET APPLICATIONS

A typical data acquisition/analysis system might consist of some number of front-end telemetry processors, each with an attached workstation and peripherals. These subsystems are then connected to a single server with an associated disk pool and archiving system.

To connect the telemetry processors to the Data Exchange tool set, high performance software adapters can be created for some of the more popular hardware. One advantage realized by using the Data Exchange tool set adapters is that other processing modules do not have to be concerned with the particulars of interfacing with different pieces of equipment. Users can "mix-and-match" equipment, while maintaining a single API for any custom modules they wish to develop. It also gives users a set of software applications which operate across a range of hardware, instead of having to master a new application for each piece of hardware. In addition, these tool modules can be used to enhance each other's capabilities, resulting in a very powerful and robust system.

One Data Exchange tool which could offer the end user greatly enhanced functionality is an archive generator. Given such a tool, possible analysis and display scenarios could include running this tool on an analysis workstation in conjunction with the appropriate adapter to allow the user to write an archive of all (or a subset of) the acquired data frames to a file in real-time. This file can then be read back while the test is still going on, to provide a quick-look analysis of the test in progress. After the test, this file could be used to start a full-blown analysis immediately, instead of waiting for data to be copied from the telemetry processor's local disk. This tool could also be used on a back-end server machine to generate a file for access by other workstations or for storage by an archival system.

Given an analysis and display tool compatible with the Data Exchange software architecture, it can be used by networked workstations to analyze either real-time or post-test data made available by other tools. It could be used to perform analysis of data stored on the server, the workstation, or possibly the telemetry processors' local disks.

The post-test extraction of time series parameter values from archived raw data files is an function which could be performed by the composition of two tool set modules. A file service module could retrieve raw telemetry frames from an archive source and make them available to other modules. An Engineering Unit Converter module could then extract the desired parameter values from each these frames and send the engineering unit converted values to other Data Exchange Applications. The resulting stream of parameters could be used by other modules for custom displays, simulations, modeling or function evaluation.

Figure 4 shows a potential layout of Data Exchange tools on a target system's computing equipment. Each of the computers has an adapter to interface to the telemetry stream processors. Some machines interface with active data sources. One of the workstations is using Data Exchange tools to analyze available data, while the

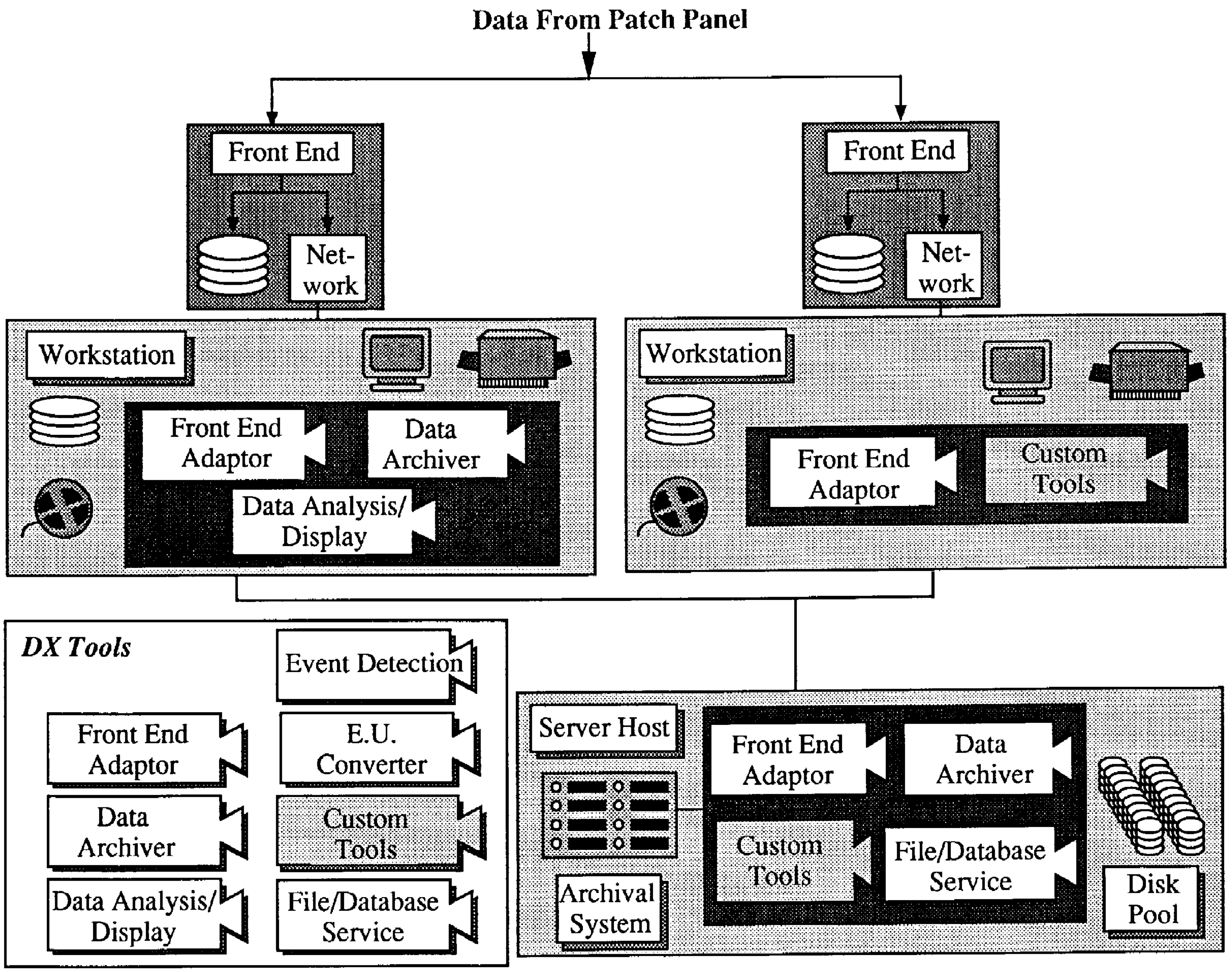

Figure 4. *A hypothetical example of how the Data Exchange tool set might be used.*

server is generating an archive of data with another Data Exchange tool. User-developed custom modules are present in another workstation. Although this example is hypothetical, it demonstrates the flexibility of a tool set based approach to client-server data analysis.

CONCLUSION

Application of BBN's Data Exchange technology, as implemented by a Data Exchange based tool set and by user-developed modules, offers a path to reduced software costs and increased functionality. Because the tools are modular and interoperable they provide a mechanism for scalable increases in capabilities with modest buy-in costs. Further, the user-extensible tool set allows customers to develop their own tools to meet unique needs. This mix of ready-to-use modules with user-developed enhancements allows the rapid and economical development of semi-custom systems.

REFERENCES

Brockett, D. M. "A Data Analysis Software Architecture for Parallel and Distributed Computation," Proceeding of the International Telemetering Conference, Volume XXVIII, October 1992.

Fidell, S., Moss, P., Fortmann, T., Sneddon, M., Milligan, S. "Distributed Processing for Real-Time Data Collection, Display, and Analysis,", Proceedings of the International Telemetering Conference, Volume XXIII, October 1987.

Lynch, T., Fortmann, T., Briscoe, H., Fidell, S., "Multiprocessor-Based Data Acquisition and Analysis," Proceedings of the International Telemetering Conference, Volume XXV, October 1989.