International Journal of Automation and Computing 4 (2006) 414-424

Data Transfer Over the Internet for Real Time Applications

Cheng-Wei Dai, Shuang-Hua Yang*, Roger Knott
Department of Computer Science, Loughborough University, Loughborough, Leicestershire LE11 3TU, UK

Abstract: Efficient real time data exchange over the Internet plays a crucial role in the successful application of web-based systems. In this paper, a data transfer mechanism over the Internet is proposed for real time web based applications. The mechanism incorporates the eXtensible Markup Language (XML) and Hierarchical Data Format (HDF) to provide a flexible and efficient data format. Heterogeneous transfer data is classified into light and heavy data, which are stored using XML and HDF respectively; the HDF data format is then mapped to Java Document Object Model (JDOM) objects in XML in the Java environment. These JDOM data objects are sent across computer networks with the support of the Java Remote Method Invocation (RMI) data transfer infrastructure. Client's defined data priority levels are implemented in RMI, which guides a server to transfer data objects at different priorities. A remote monitoring system for an industrial reactor process simulator is used as a case study to illustrate the proposed data transfer mechanism.

Keywords: eXtensible Markup Language (XML), Hierarchical Data Format (HDF), Remote Method Invocation (RMI), data transfer, web based application and real time.

1 Introduction

A basic requirement in any successful application of a web-based system is efficient real time processing and data transfer over the Internet. In a significant number of real environments, real time web-based systems involve transferring and exchanging large amounts of numerical data over the Internet. The heterogeneity and limited traffic resources of the Internet considerably complicate the transfer of such bulky data; for example, if a number of clients simultaneously try to connect with the same server, or multiple data sources are accessed over the Internet via different platforms. Such situations require that a data transfer format should be acceptable to heterogeneous platforms. Additionally, large amounts of data, such as graphics, desktop videos, and images uploaded to the Internet (which may sometimes consist of gigabytes of data) are transported, while at the same time communication bandwidth available is limited as a consequence of the popularity of the Internet.

Most related research in real time data transfer has focused on the data required in high computing applications. Widener[1] used a number of message formats wrapped using the eXtensible Markup Language (XML)[2] to provide flexible run-time metadata definitions which facilitated an efficient binary communication mechanism. Clarke[3] set up a distributed interactive computing environment using the eXtensible Data Model and Format (XDMF), which incorporated network distributed global memory, Hierarchical Data Format (HDF)[4], and XML for high performance computing applications. XDMF is an active, common data hub used to pass values and metadata in a standard fashion between application modules and to provide computational engines in a modern computing environment. Nam and Sussman[5] implemented the HDF format to store National Aeronautics and Space Administration (NASA) remote sensing data within a specific schema. The ScadaOnWeb system[6] targeted a new standard together with a generic architecture for handling numerical data in addition to enabling process control, monitoring, and optimisation via the web.

While these systems have met with varying degrees of success, most of them are designed for a local environment; little work has so far been done in the Internet environment. Methodologies which prove successful in local high computing applications are not necessarily appropriate in web-based applications, as they do not consider distinctive Internet environment features, such as diverse users and platforms. For example, a web-application cannot make any assumptions about which operating system clients use, and which data format their system supports. This is in marked contrast to a typical local application, where system data load has been determined from conception. Web-based applications have a variable work load, and are subject to the limitations of Internet traffic resources. Few existing

Manuscript received June 11, 2005; revised February 3, 2006.
*Corresponding author. *E-mail address*:
S.H.Yang@lboro.ac.uk

research papers discuss how to organize, process, and transfer heterogeneous and voluminous data; indeed, transferring real time data over the Internet is one of the major challenges for successful web-based system applications.

This paper proposes an efficient mechanism for data processing and transfer in web-based applications using the Internet. The contribution of this paper is twofold. First, it presents a new data model based on XML and HDF, which provides a generic solution for the efficient handling of heavy scientific data. Heterogeneous and bulky data is divided into light and heavy data, according to the size of the data set and its physical meaning. XML and HDF are then utilized to store light and heavy data respectively. HDF based data is wrapped using XML in the Java environment. Second, light and heavy data wrapped in XML is processed and transferred as JDOM data objects with the support of Java Remote Method Invocation (RMI), rather than traditionally as numerical bytes. This reduces the size of the data set, and enables easier data collection and retrieval for client applications as a result of HDF's compact storage structure. A data priority policy is adopted in RMI to deliver data objects in sequence. A remote monitoring system for an industrial catalytic reactor is used as a case study to illustrate the proposed data transfer mechanism.

The rest of this paper is organized as follows: The real time data processing is described in Section 2, followed by a discussion of HDF wrapped using XML in Section 3. A data object transfer mechansim is presented in Section 4, and a case study to illustrate the proposed data transfer scheme is given in Section 5. Section 6 provides conclusions.

2 Real time data processing

2.1 Features of real time data in web-based applications

In web-based systems real time data needs to be exchanged between system components via the Internet. This real time data often presents the following features:

1) Timeliness: Real time data is time sensitive and has strict time restrictions. The correctness of a system depends not only on the logical correctness of computations performed but also on time factors; late data in a stream will result in a media information interruption, while very early data can cause buffer overflow.

2) Heterogeneity and complexity: Scientific data measures physical phenomena and extends to a large range of data types. Sampling scientific data can be a single binary number, a series of numbers describing physical phenomena, or text and image descriptions of physical devices. A data record may have blocks of many thousands with data corresponding to different times, positions, measuring points, and variables.

3) Server push: Real time applications can be categorized into interactive and streamed applications. Interactive applications are characterized by a two way exchange of data, examples are Internet telephony, and distributed simulations. Streamed applications, on the other hand, are essentially one-way flows of information, such as remote monitoring systems. Data exchange in both types of application is pushed by a server, which is responsible for setting up a communication channel, initiating transmission of data, and providing various data access services to remote clients.

Managing and using real time data in the Internet environment presents many challenges including:

1) Handling huge volumes of data efficiently: because of the amount of data, it is important to use efficient algorithms and mechanisms for data storage and transfer.

2) High data retrieval performance: web-based applications need to offer high quality, cost-effective data entry services suited to high volume data computing such as data extraction from a database, and accessing and mining data from distributed data sources with minimum operating time.

3) Across heterogeneous systems: some data is stored and retrieved within the same context, other is shared or distributed to many contexts. Across heterogeneous systems, operating systems are not identical, and users are heterogeneous. A storage data context may be incompatible with a user system, and need reformatting, translation, or other forms of data modification in order to create or recreate intended meaning for user understanding[5].

2.2 Light and heavy data

To simplify heterogeneous real time data processing in our study, heterogeneous data is classified into light and heavy data (the distinction is made in terms of the amount of data and its physical meaning). Therefore, light data is considered to be small quantities of scientific data, while heavy data is considered to be large amounts of data. As a rule of thumb, if data has "more than that 300" values, it is heavy data.

Light data in our study is wrapped using XML. XML is a natural candidate for light data description and binding. An XML document is composed of a hierarchy of elements, and can be nested to any level of depth. Elements are logical units of information in an XML document. Real time data is 2-dimensional, and consists of a sampling time and variable values. The description of real time data in XML is achieved us-

ing XML elements. Each data point corresponds to an element. An element has two sections which define its value and sampling time. Information between two tags indicates data content and sampling time respectively. The structure of real time data stored in XML is shown in Fig. 1. The XML tags are user-defined and application-specific. Detail is provided in the case study in Section 5.

Light data wrapped using XML semantics can be manipulated in Java. There are three popular methods to manipulate an XML document using Java: the Java Document Object Model (JDOM), the Document Object Model (DOM), and the Simple API for XML (SAX). The tree shaped DOM/JDOM model is more suitable for data storage and manipulating an XML document, the reason being that data with multiple level attributes is arranged into a single XML element using a SAX based method. Generated classes under the DOM/JDOM based method are lightweight, and data binding applications use a minimum amount of memory (and can therefore run efficiently).

Heavy data in our study is stored in HDF format using HDF datasets and groups. Data stored in HDF is organized in a hierarchical structure with two primary structures: groups and datasets. A grouping structure contains instances of zero or more groups or datasets. Metadata contains a group name and a list of group attributes. A dataset in HDF, such as a multidimensional array of numbers, has additional metadata logically associated with it, which describes attributes of the dataset such as the rank of the array and the number of elements in every dimension. Several datasets can form a group. HDF groups and links between the groups can be designed to reflect the nature and/or intended use of stored data. Real time data in an HDF dataset is shown in Fig. 2 as a 3-dimension unit, which consists of the sampling time, the position of the variable in the dataset, and the variable value. The description of variables includes the properties of the data point and an associated explanation of these properties.

3 Wrapping HDF in XML as JDOM objects

XML provides a standard way to store and structure specific application data and is omnipresent on the Web. Wrapping HDF using XML can fully incorporate all XML features and extend HDF from a local environment to the web environment, thereby making the HDF format acceptable to web-based applications.

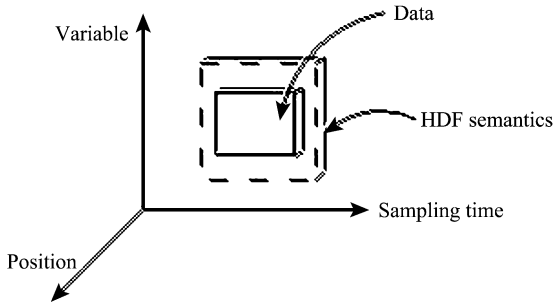

Fig. 2 HDF data storage structure

Wrapping HDF data using XML consists of two stages: structure mapping and data mapping. Structure mapping is used to ensure that relationships between data in HDF can be fully expressed in XML. Data mapping is the transference of HDF data content, attributes, and types to an XML format.

3.1 Structure mapping

XML descriptions have a tree structure with a single root and objects nested below their parents. Every JDOM document must contain one root element. Each element can contain as many children elements as needed to represent data in an XML document. JDOM provides Java specific XML functionality. With the support of JDOM, an HDF group can be mapped to a JDOM document object, and its dataset mapped to a JDOM element object. There are two cases for wrapping HDF in XML.

Case 1: HDF with a tree structure.

For a simple HDF file, which has a tree structure and is implemented as a directed graph, the HDF file can fully map to XML; in this case, the HDF group is treated as a single JDOM Document object. All objects within an HDF group are treated as elements of the structure. Attributes of groups can be represented naturally as JDOM Document attributes.

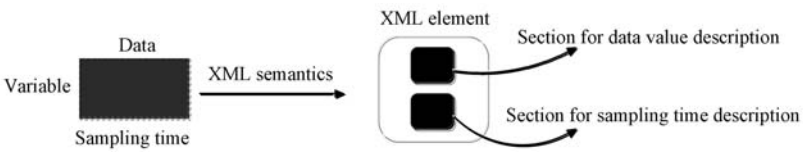

Fig. 1 XML data structure

Case 2: HDF with a complicated structure.

For complicated HDF files, which have elaborate grouping structures where some datasets are shared by a number of other groups and have more than one parent, the relationship between groups is not parallel but parental. Therefore, their structure doesn't match an XML tree structure, because there might be more than one parent for a single dataset. To map these complicated HDF structures to the structure of XML, the structure of HDF with shared datasets needs to be divided into small pieces so as to divide the overall structure into a number of tree structures. A typical example is provided in Fig. 3. Dataset 2 is shared by Group A and Group B, and Group A is a member of Group B. To simplify this complicated structure, the structure in Fig. 3(a) is decoupled, as shown in Fig. 3(b). Mapping the decoupled structure into an XML structure will lead to the XML tree structure shown in Fig. 3(c). Group A and Group B are mapped as JDOM Document A and Document B, which are at the upper level in the hierarchical structure. Datasets are mapped as JDOM elements. Due to the child-parent relationship between Group A and Group B, Group A is mapped as Element A, a child of Document B. Dataset 1 and Dataset 2 are mapped as Child 1 and Child 2. The contents of Element 1 and Element 2 in Document A are identical in JDOM to Child 1 and Child 2 in Element A respectively.

3.2 Data mapping

Data mapping between HDF and XML is achieved via JDOM elements, which have three sections: "DataValue", "DataType" and "DataAttributes". The dataset name, the size of a dataset, and other data attributes of the dataset are categorized as "Data Properties" in HDF, and stored in "DataAttributes" of a JDOM element. "DataValue" and "DataType" are identical in both HDF and JDOM. Both JDOM and HDF can be accessed in the same Java environment; therefore data in HDF can be transferred into JDOM by reading data from an HDF dataset object and writing this data into a JDOM element. These JDOM Document/Element objects are used in the same way as any other Java objects, and can be transferred over the Internet in the Java environment via the RMI infrastructure described in the following section.

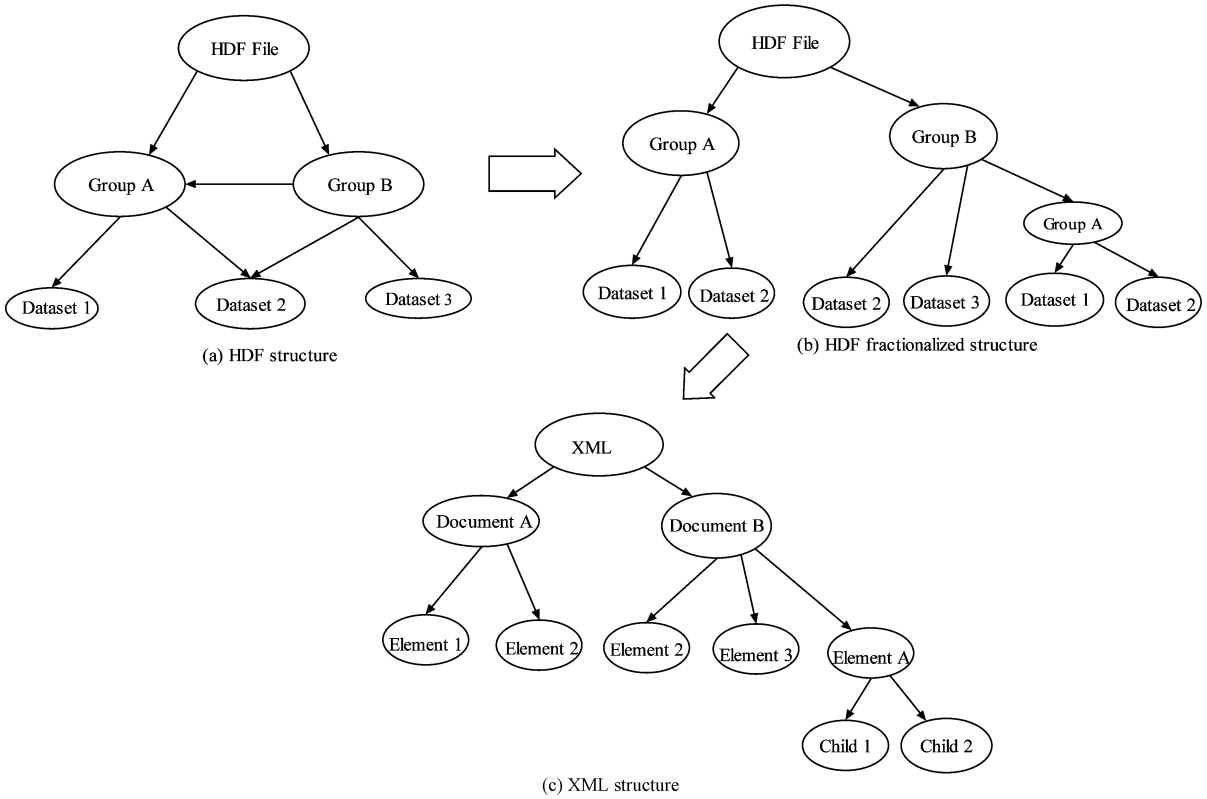

Fig. 3 Converting HDF structure to XML structure

4 JDOM data object transfer mechansim

4.1 RMI based data transfer structure

Fig. 4 depicts the structure of the data transfer mechanism proposed in this study, and it is built on the RMI infrastructure[7]. The structure is composed of four basic elements: an RMI server, an RMI Object Registry, a data processor, and remote clients. The RMI server provides back-end communication services. The Object Registry plays the role of object management and the naming service for remote objects. The RMI transfer system can pass data objects as arguments and return values using the full power of object-oriented technology in a distributed computing system. The data processor's role is to collect and process data from actual sites. According to its heterogeneous features, transfer data is first categorized into light or heavy data according to its physical meaning and requirements. Heavy data is organized into HDF format. Both categories of data are wrapped using XML, and processed as JDOM Document/Element objects in the Java environment.

It is necessary to generate a client stub and a server skeleton for RMI communication. A skeleton is a server-side entity that contains a method to dispatch calls to a remote object implementation. A stub is a proxy for a remote object at the client side, used to forward method invocations to remote objects. The stub/skeleton will be bound to the client/server through a binding action before objects begin to transfer within the RMI infrastructure. In the RMI infrastructure, Object Registry works together with the RMI server to achieve object transfer. The RMI server creates remote objects, and binds each instance with a name in the Object Registry. When a client invokes a method for a remote object, it first looks up the remote object in the Registry by the remote object name. If the remote object exists, the Registry will return a reference for the remote object to the client, which can then be used to make method invocations upon it. As long as there is a reference to the remote object, the object will be reachable from the remote clients.

The sequence of data transfer in the RMI infrastructure is summarised in the following five stages:

Stage 1: Categorize heterogeneous data into light and heavy data according to its features.

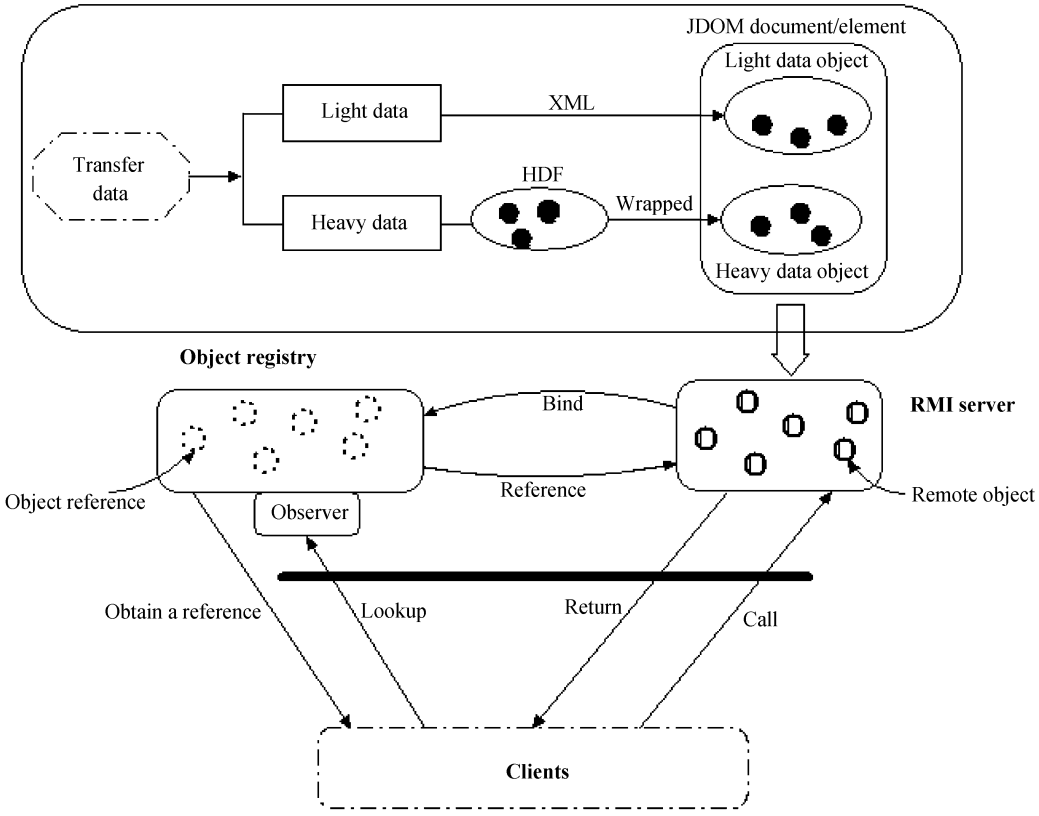

Fig. 4 Data transfer mechanism based on RMI

Stage 2: Organize heavy data into HDF format and wrap both types of data using XML.

Stage 3: Generate a stub and skeleton for remote objects.

Stage 4: Start the Object Registry and bind remote objects in the Object Registry.

Stage 5: Transfer data by employing the RMI method at every sample interval with a data object as an argument.

4.2 Data object priority

Light and heavy data is stored in XML documents, which are treated as data objects in the Java environment. These data objects describe both data values and their structure. By assigning different priorities to data objects, some data objects can be transferred faster than others so as to meet the different requirements of clients. Normally, vital data is assigned a high priority, and ordinary data given a lower priority. Data with high priority is sent out before data with a lower priority. Priority levels are maintained in the server, and updated once any new data transfer requests arrive, or a transmission is complete. Clients can terminate data transfer if necessary.

Data priority submission is implemented using the Java interface PrioritySubmission in the RMI infrastructure. By extending the interface java.rmi.Remote, the PrioritySubmission interface makes itself available from any virtual machine. The signature of the interface PrioritySubmission is as follows:

public interface PrioritySubmission extends Remote {

public void DataPriority (String [ ] *sequence) throws RemoteException;*

public void EmergencyStop (boolean [ ] *stop) throws RemoteException;*

}

The *DataPriority* () and *EmergencyStop* () methods are supported by both the server and client side. The *DataPriority* () method is used by a client to submit defined data priority to the server. The *EmergencyStop* () method is used to notify the server to suspend some data transfer. Both methods are defined as being capable of throwing a Java.rmi.RemoteException. The data object name with highest priority is placed at the start of a priority sequence, while the least important data object is placed at the end. The stop sequence is a Boolean array, and defines a logic value True/False with reference to continuing or terminating data transfer.

Data transfer using various priorities is realized via the proposed observer at the server side, as shown in Fig. 4. The proposed observer acts as an object manger. When a client submits an invocation to a remote object, there will be an order to enter the Registry to look up the remote object and obtain references at the server side. The observer is designed to make the order identical to the data priority sequence submitted by the client. In detail, the observer unbinds all remote objects in the Registry before remote object look up. When the invocation is requested from the client, the observer will rebind the remote objects step by step according to the data priority sequence. The highest priority object is rebound first, while the lowest priority object is rebound last. Only rebound remote objects can obtain a reference from the Registry to subsequently be accessed by the client; other objects which fail to obtain a reference, continue to wait until they obtain a reference from the Registry. Therefore by controlling access to a reference, remote object transfer is governed using data priority levels. In the case of a suspension request from a client, the observer will be notified to refuse rebinding of remote objects. This is requested to terminate data transfer, so that references for remote objects can no longer be obtained and data transfer is terminated. The procedure of looking up remote objects is shown in Fig. 5. Each request from a client repeatedly makes the look up appeal until a reference is obtained. The *LookUp* () method is responsible for looking up a remote object's reference. The value of the Boolean variable "hasReference" is set by the return value of the *LookUp* () method. When "hasReference" is true, it indicates that the *LookUp* () method has found the reference and the method. *GetReference* () is invocated to catch the reference and locate the remote object via the *Locate* () method.

5 Case study

In order to demonstrate the proposed data transfer mechanism, a remote monitoring system for a reactor process simulator was used as a test bed. The details of its implementation are discussed in this section.

5.1 System description

The reactor process simulator[8] shown in Fig. 6, consists of a heat exchanger E201, a reactor R201, and four-hand valves for Nitrogen inlet, liquid outlet, gas outlet, and emergency liquid outlet. The inlet temperature of the reactor is controlled by a PID controller, which manipulates the hot stream flow rate of heat exchanger E201. Fig. 7 shows the structure of the remote monitoring system, which consists of three parts: the reactor process simulator, an RMI data transfer system, and several remote clients. The local control system collects data from the reactor process and passes this to the data processor. The process data is analysed

and organized into XML and HDF data formats in the RMI data transfer system. The RMI system is linked to remote clients via the Internet. Remote clients can monitor the local reactor process, define data transfer priority levels, and submit a request to the server. Acquired data is processed as JDOM Document/Element objects, and then regularly transferred to the remote clients with a defined priority level (once any requests from clients are accepted).

5.2 Priority of data transfer

All process variables are categorized as static or dynamic data. Static data includes the parameters of the PID controller and the states of the four values (open/close and manual/automatic): nitrogen inlet, liquid outlet, gas outlet, and emergency liquid outlet. Static data is categorized as light data. This type of data is normally of limited size and amount, and changes slowly. Dynamic data includes all the process variables, i.e. reactor temperature, pressure, concentration, and flow rate. It is categorized as heavy data, and collected every sampling interval in HDF format.

The main objective of the monitoring system is to remotely monitor control performance, for example how well a controller works together with a process to track the setpoint of process output. Output variables, such as temperature and pressure in the reactor, the work state of control devices, and PID parameters, are essential for monitoring purposes. Data priority levels are set as shown in Table 1 in terms of the monitoring system specification. All light data is set with the highest priority. The small data size of the light data utilizes limited data communication bandwidth and introduces minor extra load on the communication channel. The transfer priority of flow rate is set with the lowest priority in this study.

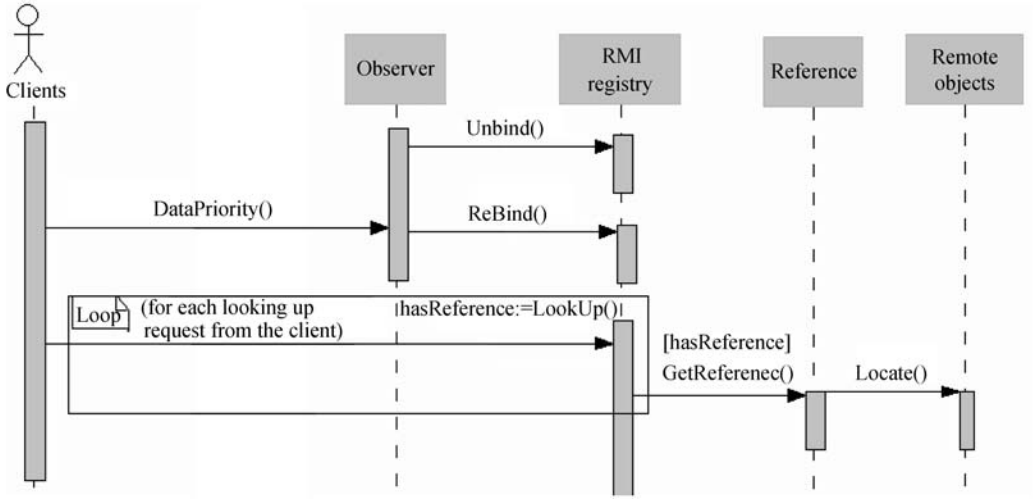

Fig. 5 Procedure for looking up remote objects

Table 1 Data priority

	Data	Transfer Priority Rank
Light Data	State of Nitrogen Inlet Valve	1
	State of Gas outlet Valve	1
	State of Liquid outlet Valve	1
	State of emergency liquid outlet Valve	1
	PID parameters	1
Heavy Data	Concentration	1
	Pressure (top)	2
	Pressure (bottom)	2
	Temperature (top)	3
	Temperature (middle)	3
	Temperature (bottom)	3
	Flow Rate	4

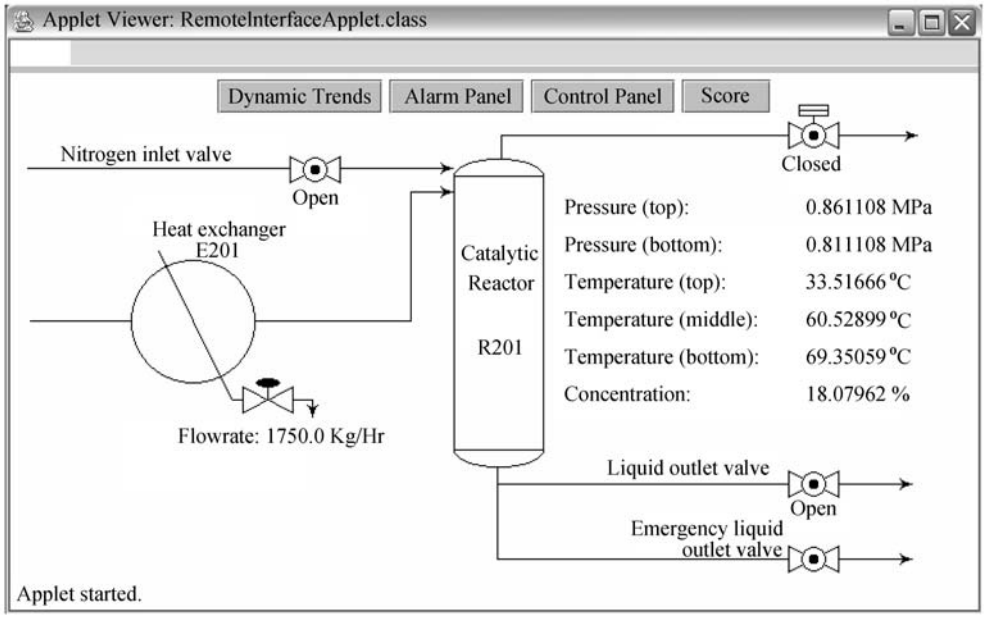

Fig. 6 Reactor process simulator

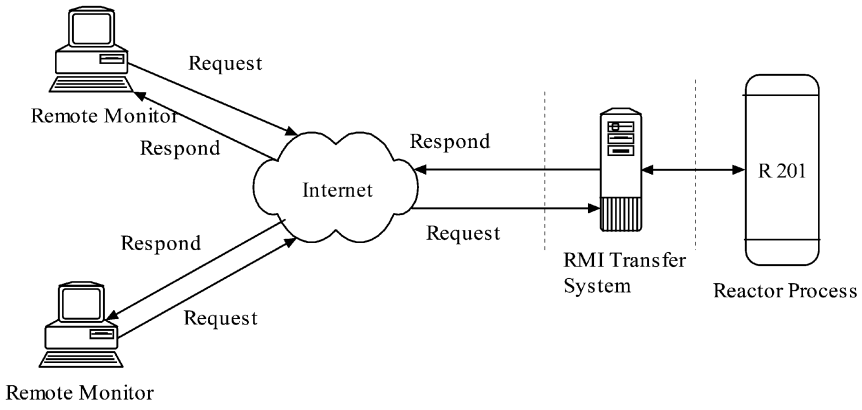

Fig. 7 Architecture of the remote monitoring system

5.3 Implementation

Two XML files were created to store and transfer light data. One contains PID parameters, while the other contains the work state and mode of the four control valves. The sampling interval of data collection is 1 second. Each data element stored in an XML file is a real time data point, which includes the value of the variable and its corresponding sampling time. Heavy data is collected in an HDF file; the HDF file has four groups which collect the pressure, temperature, concentration, and flow rate, respectively. The structure of the HDF file is shown in Fig. 8. It has a tree structure and can fully match the XML structure; each data variable is 3-dimensional, including the sampling time, the variable position in the dataset, and the variable value. The data type of the datasets is double and the data length is 300, which covers a 5 minute sampling period.

The HDF file is mapped to JDOM data objects in XML. The four groups in the HDF file are converted into four JDOM documents; the datasets are mapped to the elements of the JDOM documents; property information is stored in the element attributes. Mapping to the XML elements, "DataValue" has two XML tags: <Variable value> and <SamplingTime>. The position of data points is not used in this study. Variable values and sampling times are inserted between these two opening/closing tags. "DataType" is represented using the tag <DataType>, and "DataAttributes" using the tag <Attributes>, and saves the data dimension and size of the amount of the HDF dataset. All data is

presented as JDOM objects in the Java environment. Remote clients access the JDOM objects by invoking the *LookUp* () method. RMI security manager has been installed at both the local and remote sides, which allows a legal Java virtual machine to download stub and skeleton class files.

5.4 Simulation results and analysis

Tables 2 and 3 illustrate the wrapping of light and heavy data. Using the RMI infrastructure, the work status of the four control valves and the PID controller parameters are wrapped in XML as light data objects and sent to a remote monitoring site via the Internet. User-defined tags include <PID_P>, <PID_I>, <PID_D>, and <SamplingTime>. All PID parameters are wrapped between the opening and closing tags of these tags. Table 2 provides an XML file for PID controller parameters. Each data object, such as PID_P in Table 2, is treated as an element object in JDOM, which has two sections, the PID_P parameter value and sampling time. Heavy data is collected in the HDF format wrapped as JDOM Document/Elements. Table 3 shows the tree structure and individual dataset in an XML file for heavy data, in which a super group 'Root', its subgroup 'Group4', and its dataset 'Concentration' are nested with various XML tags. The heavy data of this dataset is transferred from an HDF file named 'HeavyData.h5'. The attribute describes the dataset as 2-dimensional, with unit '%', type 'Float', and 300 data points. Both light and heavy data are stored as JDOM Document/Elements and incorporated into the RMI transfer system. The server binds the data objects in the Object Registry so that any data objects can be looked up from a remote site. The data priority policy is applied in the RMI based data transfer mechanism. Fig. 9 shows the concentration of product and temperatures received at the remote client site, at the top, middle, and bottom of R201.

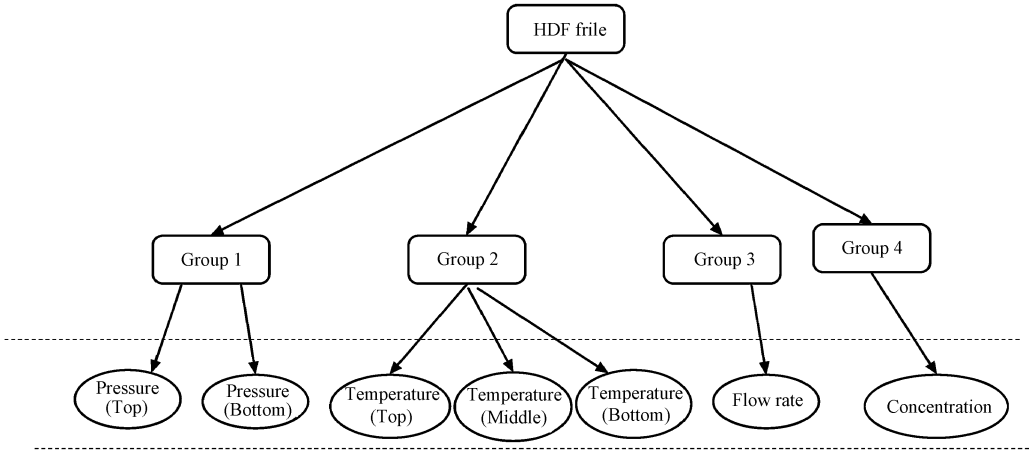

Fig. 8 HDF data structure

Table 2 XML file of light data

```
<Simulation>
PID.xml <Sector>
                <Parameters>
                        <PID_P>50</ PID_P >
                        <Sampling Time>11:34:47 12/08/2004</ Sampling Time >
                        <PID_I>5</ PID_I >
                        <Sampling Time>11:34:47 12/08/2004</ Sampling Time >
                        <PID_D>0</ PID_D >
                        <Sampling Time>11:34:47 12/08/2004</ Sampling Time >
                </Parameter>
PID.xml </Sector>
</Simulation>
```

Table 3 XML file of heavy data

```
<HDF>
HDF.xml
 <Sector>
 <RootGroup>
  <GroupName>Root</GroupName>
  <Group>
   <GroupName>Group4</GroupName>
   <DataSet>
    <DataSetName>Concentration</DataSetName>
    <HDF5File>"HeavyData.h5" </HDF5File>
    <Section>
     <Attributes>2D % 300</Attributes>
     <DataType>Float</DataType>
     <Concentration>18.10 18.08 18.06 18.02 18.00 17.99
                    . . . . . .
     </Concentration>
     <SamplingTime>19:38:57 19:38:58 19:38:59 19:39:00 19:39:01 19:39:02
                    . . . . . .
     </SamplingTime>
    </Section>
    </DataSet>
    . . .
   </Group>
   . . .
  </RootGroup>
 </Sector>
</HDF>
```

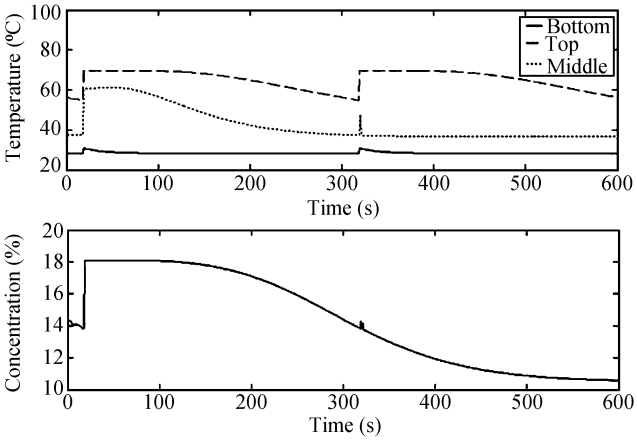

Fig. 9 Data received at the remote client side

5.5 Advantages of RMI based data transfer

Traditional data transfer methods are usually based on socket technology, which is an end-to-end communication mechanism used to establish a connection between a client and a server program. The server communicates with the client via a socket, and listens for client connection requests. Communication with a client is conducted by a data stream read from and written to the socket. To assess the proposed RMI-based data transfer scheme, a standard TCP socket based data transfer method was implemented to carry out the identical data transfer tasks described above.

Two significant advantages have been identified as follows:

1) The amount of data presented using HDF and XML is smaller than that presented as a data stream when using socket technology to transfer data. Therefore, data transfer latency using the proposed RMI based method may be shorter than that using traditional socket technology. In a comparison study, a 65 536 byte data set was collected and transferred to a remote client in 1.553 seconds. The same data set when stored in HDF format only required 22 630 bytes, because HDF supports a highly customized data filtering and scatter-gather algorithm[4]. After wrapping in XML, the data set was extended to 39 297 bytes due to the addition of XML tags; however, this is still much smaller than the original data set. The RMI based data transfer method took 1.492 seconds to complete the data transfer, which is slightly faster than the TCP socket method.

2) Data objects presented using JDOM are well structured and easy to locate, query, and retrieve. Therefore, the proposed RMI based data transfer method is more efficient during data query and retrieval than data streams in a traditional socket approach. In detail, all data in the RMI based data transfer method is wrapped using XML, and formed and transferred as JDOM data objects (which are structured as hierarchical trees and can be directly accessed by clients as local objects). To retrieve data, a client only needs to

make an invocation on objects. Under the tree structure, data retrieval in terms of data objects is simple and fast for client applications.

6 Conclusions

Web-based applications need an effective data transfer mechanism to process and transfer real time heterogeneous data. This paper proposes an RMI based data object transfer mechanism. Heterogeneous transfer data is categorised into two basic groups: light and heavy data according to physical meaning and data amount. Light data is wrapped in XML. Heavy data is wrapped in HDF initially, and then in XML. Both light and heavy data are processed as JDOM data objects in the Java environment. JDOM data objects are transferred over the Internet using the RMI mechanism. This structure is flexible; it not only allows a server to transfer data objects, but also allows clients to define and submit a data priority sequence to the server so as to guide data transfer. The main advantages of the proposed mechanism are: (a) it simplifies heavy data retrieval in client applications via remote invocation of remote data objects and a hierarchy of data objects; and (b) the size of a data set is reduced because of the compact storage structure of HDF, therefore transmission latency is reduced. A remote monitoring system for a reactor process simulator was used as a case study. The data transfer results indicate that the proposed DMI based data transfer mechanism has great potential to improve the efficiency of data transmission, and therefore meets the requirements of data transfer over the Internet in real time applications.

References

[1] P. Widener, G. Eisenhauer, K. Schwan. Open Metadata Formats: Efficient XML-Based Communication for High Performance Computing. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, pp. 315−324, 2001.

[2] T. Brag. The Real Benefit of XML. [Online], Available: http://www.fcw.com/fcw/articles/2000/0522/tec-bragg-05-22-00. asp, 2000.

[3] J. A. Clarke, J. J. Hare, J. D. Brown. Implemen- tation of a Distributed Data Model and Format for High Performance Computing Applications. [Online], Available: http://www.hpcmo.hpc.mil/Htdocs/UGC/UGC00/paper/jerry_cla-rke_paper.pdf, 2000.

[4] R. E. McGrath. HDF5 Compound Data: Technical Issues for XML, Java and Tools. [Online], Available: http://hdf.ncsa.uiuc.edu/HDF5/XML/tools/compound-data.html, 2001.

[5] B. Nam, A. Sussman. Improving Access to Multi-Dimensional Self-Describing Scientific Datasets. [Online], Available: http://csdl.computer.org/comp/proceedings/ccgrid/2003/1919/00/19190172abs.htm, 2003.

[6] ScadaOnWeb. Description of Work. [Online], Available: http://www.scadaonweb.com/publications/ScadaOn- Webdescripti-onofwork.pdf, 2001.

[7] Sun Microsystems. Java^{TM} Remote Method Invocation Specification. [Online], Available: http://java.sun.com/j2se/ 1.5/pdf/rmi-spec-1.5.0.pdf, 2004.

[8] S. H. Yang, J. L. Alty. Development of a Distributed Simulator for Control Experiments Through the Internet. *Future Generation Computer Systems*, vol. 18, no. 5, pp. 595−611, 2002.

Cheng-Wei Dai received the M. Sc and B.Sc. degrees from Wuhan University, China in 2001 and 1998, respectively. He received his Ph.D. degree in computer science from Loughborough University in 2005. He is currently working in Shandong Nuclear Power Co. Ltd., China, as an information engineer.

His research interests include remote monitoring and maintenance of control systems, and real time data transfer over the Internet.

Shuang-Hua Yang received his B.Sc. degree in instrument and automation, M.Sc. degree in process control from University of Petroleum, China in 1983 and 1986, respectively, and a Ph.D. degree in intelligent systems from Zhejiang University, China in 1991. He is a senior lecturer in Computer Science and the Director of the Networks and Control Research Group at Loughborough University, UK. He is also an overseas professor in Central China Normal University and a guest professor in Huazhong University of Science and Technology, China.

His research interests include wireless sensor networks, networked control, safety critical systems, and real time software maintenance.

Prof. Yang is a fellow of the Institute of Measurement & Control (FInstMC), a senior member of IEEE, the chairman of the East Midlands Section of InstMC, and a Chartered Engineer (CEng) in the UK. He is an associate editor of the *International Journal of Systems Science* and a member of the editorial advisory board of the *International Journal of Information and Computer Security*.

Roger Knott receive his B.Sc. degree in mathematics and Ph.D. degree in pure mathematics from the University of Warwick. He is a lecturer in the Computer Science Department at Loughborough University, UK. His work is in very much at the human end of the spectrum. He has worked in the computer industry for 40 years.

His research interests include human computer interaction and object oriented system development.