http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (24): 8081-8087, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.8081.8087

Real-time Exchange of Mass Data in the Internet of Things

Hao Xu, Xuemiao Xu, Yuqi Fan and Yulong Guo
School of Computer Science and Engineering, South China University of Technology,
Guangzhou, Guangdong 510641, China

Abstract: In this study, we design a novel real-time mass data exchange system for IOT. Compared to the existing methods, our method can achieve real-time data transmission while guaranteeing the high stability and accuracy. It is designed with the following four unique improvements. First, by employing asynchronous socket to allow connection from mass TCP clients, system resource is significantly saved and efficiency is ensured. Secondly, the concept of generic protocol is proposed to solve the problem of parsing protocols of various formats, enabling our solution to be self-adaptable and well-extensible. Furthermore, we also propose a novel three-level data storage model to handle concurrent access and modification to the cache space by multiple threads. Lastly, to improve performance, we propose to reduce the number of database queries by utilizing the mapping relationship between the terminal IDs and the caches of the query and forwarding threads.

Key words: Internet of things, mass data, real-time exchange, multi-thread, data storage model

INTRODUCTION

The Internet of Things (IOT) (Ashton, 2009) is the connection of mass sensor devices that communicate through the Internet and has the capability of identifying, tracking down, monitoring and managing. The IOT's technical architecture mainly consists of three parts: the perception layer, the network layer and the application layer. However, data communication between the network and perception layer involves plenty of terminal devices and application servers, which leads to several challenges. First, the continuous data flow from the large amount of terminal devices may easily reach the data load bottleneck for an application server. Secondly, thousands of applications may require the data from the terminal devices to be distributed to different servers deployed at different domains. Moreover, the correspondence between the terminal devices and servers is updated dynamically. A common solution to these problems is to set up a server group (2002, 1999) and appoint independent servers to receive the data uploaded by different terminals. This solution could relieve the burden on individual servers to some extent. However, unfortunately, since it requires every terminal device to be re-configured of its own binding server IP, while the number of terminal devices is very large and they are geographically dispersed, this makes this solution barely feasible in practical.

To avoid this problem, we propose the usage of data exchange technology to achieve high quality communication between the terminal devices and the server group. Figure 1 takes the traffic monitoring platform as an example and shows our data exchange system which automatically distributes the satellite positioning data uploaded by every on-board terminal to its corresponding application server. However, our exchange system is faced with two difficulties, the mass data and real-time exchange. The mass GPS data from on-board terminals are likely to cause congestion in the system and disable the data exchange of being real-time.

Data exchange is an active research topic due to the increased need for exchange of data in various applications. Researchers attempted to apply the data exchange technology to practical applications (Miller *et al*., 2001) and also discussed some theoretically foundational problems in data exchange (Kolaitis, 2005).

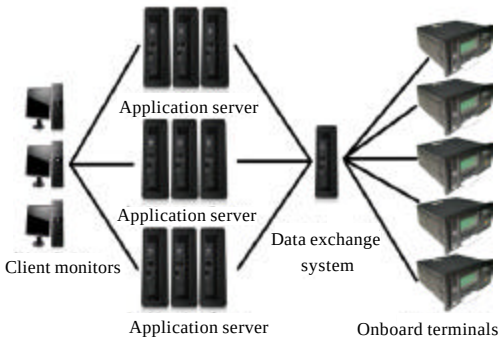

Fig. 1: Multi-application server array based on the data exchange system

Corresponding Author: Xuemiao Xu, School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong, 510641, China

As for practical applications, a system Clio for data exchange was built (Miller *et al*., 2001; Popa *et al*., 2002) and partly incorporated into the IBM's db2 product;s He *et al*. (2005) have proposed a secure data exchange system that can block away viruses and attackers and can isolate the internal and external databases; Zhang *et al*. (2000) have proposed an Internet-based STEP data exchange framework for virtual enterprises which are based on various computer systems. Amer-Yahia and Kotidis (2004) proposed an efficient XML data exchange method and applied it to business applications. As for the study of theoretical foundations, it was started with the influential papers by Fagin *et al*. (2003). They laid the theoretical foundation of exchange of relational data (Fagin *et al*., 2003) and studied various issues in data exchange such as schema mapping composition (Fagin *et al*., 2004) and query rewriting (Arenas *et al*., 2004; Yu and Popa, 2004). Arenas and Libkin (2005) looked into the basic properties of XML data exchange and investigated the consistency problem and determined its exact complexity. Kolaitis *et al*. (2006) explored the complexity of data exchange. Libkin (2006) discussed the relationship between data exchange and incomplete information. Nevertheless, a method for mass data exchange is yet to be proposed. And our goal is to bring such a method to realise real-time exchange of mass data between the network layer and perception layer in IOT.

In this study we look into the major factors that affect the efficiency of data exchange between the network layer and perception layer in IOT. The factors are as follows: mass terminal devices sharing the same server's IP and port; parsing protocols in different specifications; mutual exclusion in cache of multi-thread query and forwarding modules; the storage of destination information which is used to forward the datagrams to the next application server. In this study we propose the corresponding solutions. First by socket's non-blocking listening, mass TCP clients are granted access; then we design a well-extensible datagram parsing module; we use a novel three-level data storage model to solve the problem of multi-thread concurrent access to cache; additionally we put forward the technique of utilizing the mapping relationship between terminal IDs and the cache space of query and forwarding threads and reduce the number of database queries to improve performance. The final results of the experiments show that our system is highly efficient and stable.

PROPOSED METHODS

In this section, based on the objectives of our system(the single port, high concurrency, high configurability and high performance), we will thoroughly discuss about the four major factors that have influence on the efficiency of our data exchange system and propose the solutions accordingly.

Single IP and port for the access of mass terminal devices: Our solution requires a single port for exchanging data with on-board GPS terminals and other systems. The on-board GPS terminal may use TCP or UDP. For the terminals that use UDP we can bind the specific IP and port with a single socket and start a thread to listen for incoming data. If the on-board terminals use TCP, the listening for all the connections demands a huge amount of threads, which would exceed the system's capacity.

We propose that one thread is enough for mass TCP terminals' access through listening to a socket. Each time a connection successfully builds up, it is wrapped into a TCP object (CTcpClient) and we can use the beginReceive() method in c# to listen for the incoming TCP datagrams. Since this method is non-blocking, when no datagram is arriving, the TCP connection does not consume CPU cycles. Until a datagram arrives the TCP connection object is called up to receive data and delivers data to protocol parsing module. Therefore, we can use only a single IP and port for connection of mass terminal devices. Figure 2 demonstrates the sequence diagram of our proposed method of utilizing socket communication module to process the data received from an uplink with on-board terminals and a downlink with application servers.

Parsing protocols of various formats: The parsing module needs to parse out IDs of remote terminals in order to query in the database and obtain forwarding IPs. But the satellite positioning data from different terminals present dissimilarity since they employ various protocols. The manufacturers often customise the protocols under demand of customers. If we design protocol parsing modules for all the manufacturers and write classes for all types of protocols, the system would be built into a mess. We put forward the concept of generic protocol so that the system can be self-adaptable and well-extensible. By extracting a common structure from various kinds of protocols, all of them are within the generic protocol.

The system calls the appropriate parsing methods according to the configuration profile in the procedure illustrated in Fig. 3. The parsing thread gets the datagrams from the task buffer queue one at a time and extracts the protocols from the configuration XML file in order. If a protocol is valid by default (namely the device ID is in certain word length), then we apply the following steps sequentially: protocol check (matching the protocol's

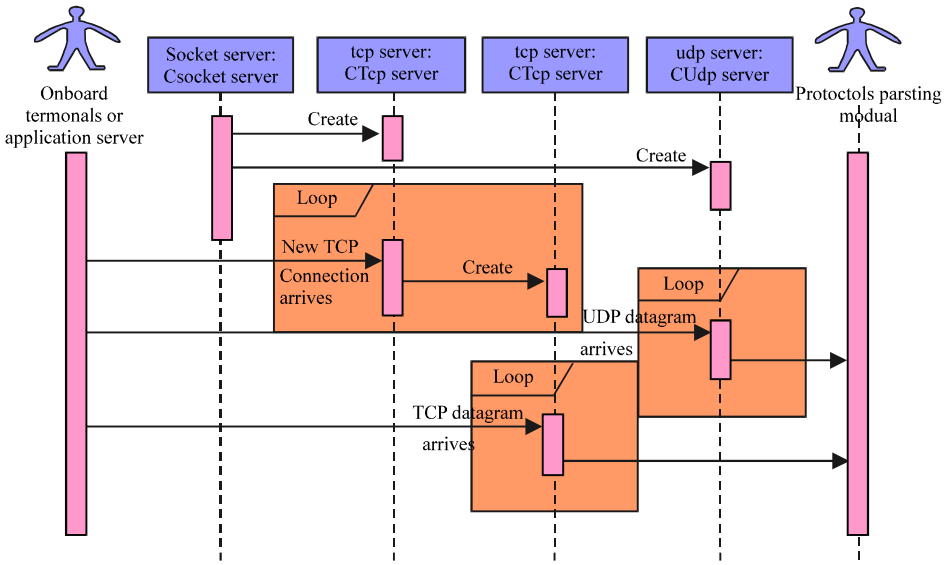

Fig. 2: Sequence diagram of the socket communication module

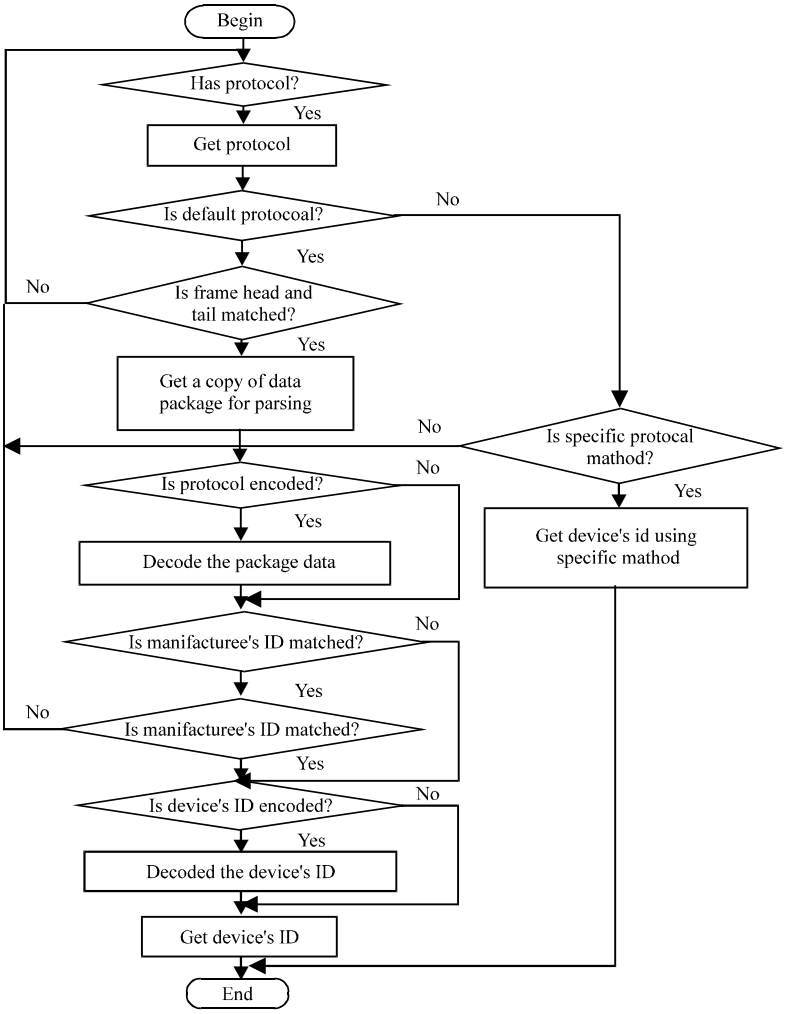

Fig. 3: Flow chart of parsing the datagram for the device's ID

frame head and tail), data decoding (decoding the data between the frame head and tail), matching manufacturer number, decoding the terminal ID and finally retrieving the ID of the terminal device. Otherwise, it decodes the terminal ID in a custom method.

Mutual exclusion in cache of multi-thread querying: By studying similar system frameworks (Boasson, 1995; Perry and Wolf, 1992), we discovered that the system bottlenecks appear as follows: First, frequent operations of querying and modifying to the database; Second, frequent lock operations to the data cache. Because mass concurrent queries to the database and access to the same cache will interrupt other threads and waste CPU cycles, we redesign the cache strategies to increase performance.

We are aware of the fact that the second problem is introduced to solve the first one. Therefore we have two approaches to avoid such dilemma. First, the query and forwarding module has N instance objects of query and forwarding units and every instance has its own operating thread and datagram buffer queue which is prepared for forwarding. Multiple threads can use the database operation module at the same time to get the forwarding IPs of the datagrams and the queries to the database can use a read-write lock so that one writing thread and another won't be in mutual exclusion. Secondly, each query and forwarding unit has its own cache hash table with the terminal Ids as the keys to buffer the forwarding IPs queried from a local database. Since multiple buffers exist, we expect less mutual exclusion in multi-threaded access to the same data-cache of hash table.

Storage of terminal forwarding destinations: The system needs to store the terminals' forwarding IPs before powering off and must reduce the access to the database because of the mass data. Therefore, we cannot use a simple cache-database two-level storage structure for accessing.

Designing the cache read-write strategy has always been a major difficulty in multi-threaded programming. Our task is the distribution of mass data based on the information stored in the database; therefore we propose a novel three-level data storage model to cope with the frequent access to the cache. As demonstrated in Fig. 4 the first level is the database, where would store the forwarding information when the system powers off. The second level is the virtual DataTable cache object in the database operating module. When this module is initialised, it automatically generates the DataTable objects corresponding to the tables in the database. The third level is the private caches controlled by the query and forwarding threads in the query and forwarding module. Each thread first searches in its own cache for the forwarding IP for a terminal ID or else it queries in the DataTable and updates its cache with the absent information. Since the terminal IDs match the query and forwarding threads as stated in Section 3.3, when the next datagram from the same terminal arrives, the thread only needs to search in its own cache for the forwarding IP and does not need to access the database operating module. The program also supports updating the cache with forwarding IPs of multiple terminal IDs via a provided user interface. Therefore, each operation directly updates the DataTable objects and every 60 sec the database update thread

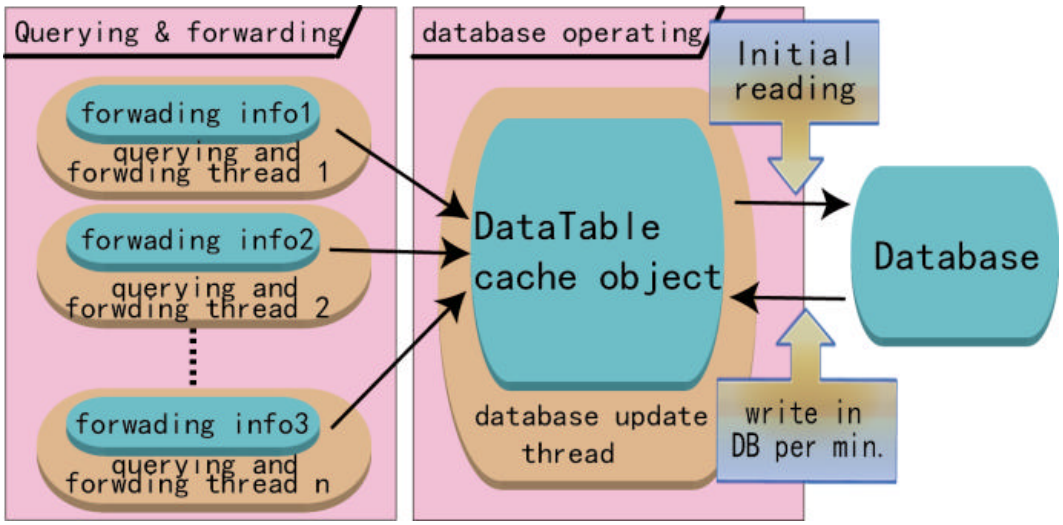

Fig. 4: Three-level data storage model

automatically adds new terms to the database according to the changes in the DataTable so that the database is up-to-date.

IMPLEMENTATION AND EVALUATION

Implementation: This section details how our data exchange system was implemented. As shown in Fig. 5, the system consists of 5 parts including the datagram class, the socket communication module, the multi-thread parsing module, the multi-thread query and forwarding module and the database operating module.

The datagram class(CPackage) is to package the following data members: the original data bytes received from socket, the communication protocol type (TCP/UDP), the source IP of the datagram, the sending direction of the datagram (IN/OUT) according to the source IP and the terminal ID extracted by the protocol parsing module. The socket communication module (PSocketServer) is to receives datagrams and provides listening service of non-blocking TCP and simple UDP connections. The multi-thread protocol parsing module (PParseService) is to manage multiple parsing threads. It receives datagram objects from the socket communication module and extracts their terminal ID, according to the ID it then sends the object to one of the threads in the query and forwarding module.The multi-thread query and forwarding module (PProcessService) is to manage multiple query and forwarding threads. The database operating module (PDatabaseService) is to manage a database updating thread and maintains a virtual data table object (Datatable) for caching.

Evaluation: To test and verify our solution, we established the system on an ordinary PC to run for a period of time. The testing environment is as follows:

- **Hardware:** Intel Core i5-2410 2.30GHz CPU, 4GB RAM, 500GB/5400rpm HDD
- Network: Internet
- **Software environment:** Microsoft Windows 7 Professional, Microsoft. Net Framework 4.0, MySQL 5.0
- **Connected vehicles:** 20000 in total, among which 5000 are actual running vehicles and 15000 are simulated ones. Each vehicle is expected to update the positioning data to server every 8 sec
- **Application servers (simulated):** 10 PCs is used to deploy the data receiving program. Each of them gathers and records statistics of received data. The data exchange system forwards the data from the 20000 vehicles to the 10 servers simultaneously
- **System configuration:** The data flow in this experiment is large. The data exchange system employed 10 protocol parsing threads and 15 query and forwarding threads in total

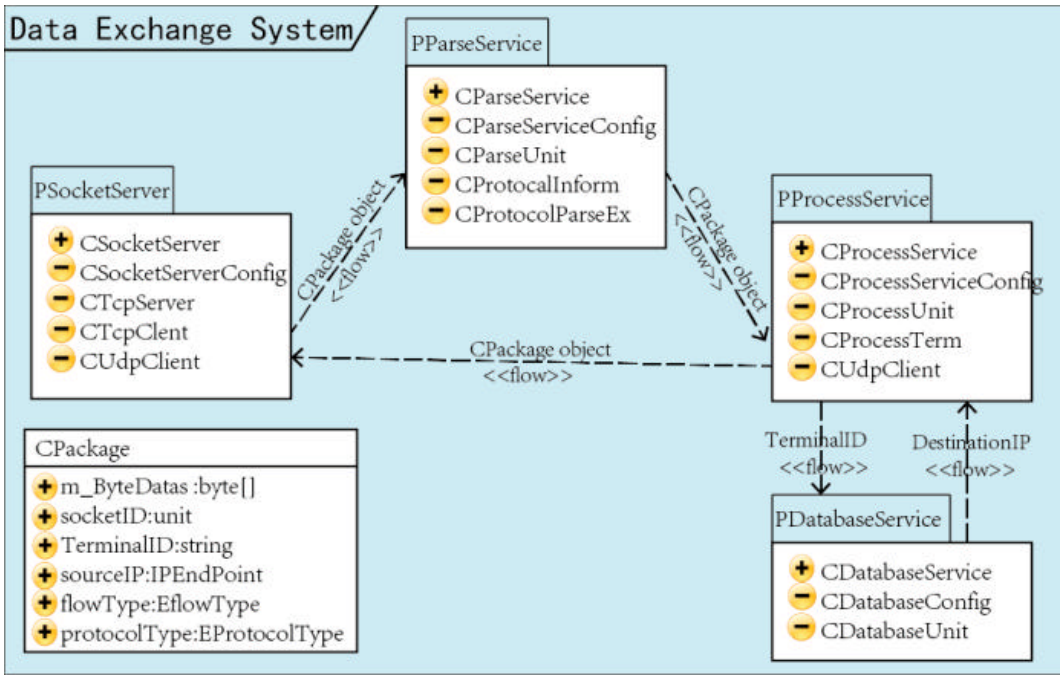

Fig. 5: Implementation framework of the data exchange system

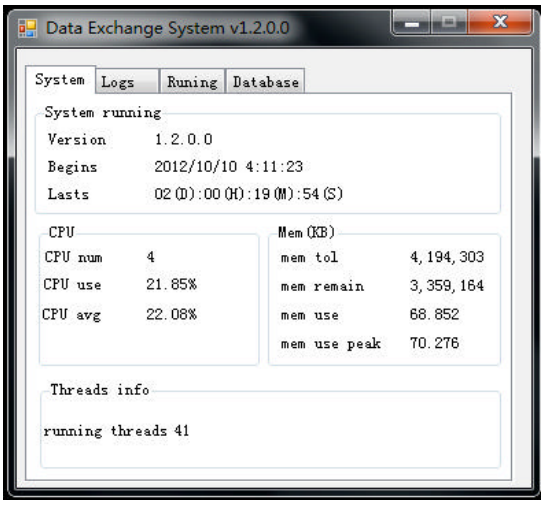

Fig. 6: Situation of the running system

Table 1: Data source

Data source	Running vehicles	Simulators
Quantities	5000	15000
Data upload interval (sec)	8	8
Total running time (h)	48	48
Actual running time (h)	About 16	48
Actual message count (Piece)	36000000	324000000

Table 2: Forwarding rate of exchange system

Receiving count	345896120 pieces
Sending count	345896120 pieces
Success rate	100%

For our real-time data exchange system, we mainly focus on three aspects: the stability of the system, the success rate of the forwarding system and the success rate of receiving at the server end.

Stability: After 48 h of continuous running, the system is in good status. As shown in Fig. 6, the system maintained an ideal resource consumption level after long time running.

Success rate of the forwarding system: In Table 1 is the data source, namely the theoretical data quantity sent by the on-board devices. But the vehicles don't run all the time, so the actual data quantity is lower than this. Table 2 shows the statistics of the data actually received and forwarded by the exchange system. It can be seen that the success rate in our exchange system is 100%.

Success rate of receiving at the server end: There may be package loss due to the exchange system employing UDP protocol for forwarding. Therefore in Table 3 we state the statistics of the success rate at the server end. It can be seen that the average success rate of receiving among the ten simulated application servers is 99.97%. This success rate is usually enough for general applications.

Table 3: Success rate at the server end

App. servers	Expected data count (pieces)	Received data count (Pieces)	Success rate (%)
Server 1	345896120	345896120	100.00
Server 2	345896120	345662720	99.93
Server 3	235698717	235604437	99.96
Server 4	267856900	267722972	99.95
Server 5	300123476	300123476	100.00
Server 6	142398756	142270597	99.91
Server 7	200564390	200564390	100.00
Server 8	286954305	286954305	100.00
Server 9	138769001	137797618	99.93
Server 10	325698001	325698001	100.00

CONCLUSIONS

Based on the example of the traffic monitoring platform, by designing and implementing a real-time exchange system of mass satellite positioning data, this study solves the problem of data load bottlenecks in the application servers caused by the mass data transmission between the network layer and the perception layer in IOT and the problem of real-time exchange of datagrams between the mass terminal devices and the various application servers deployed at different domains. This platform is based on the systems such as GPS, GIS and GPRS wireless communication platform and can realise dynamic monitoring and information managing of all the target vehicles in the range of the GPRS network. This study focuses on the real-time exchange system of mass satellite positioning data, introducing the technical aspects in designing and realizing the key modules such as the socket communication module and the multi-thread parsing and querying module. Additionally, we propose in this study a well-extensible datagram protocol parsing module and a novel three-level data storage model to solve the problem of multi-thread access to the cache and the database. Our solution is well supported by our experiments.

ACKNOWLEDGMENT

This study was supported by NNSF of China (Grant No. 61103120); the Fundamental Research Funds for the Central Universities of South China University of Technology (Grant No. 2013ZM0086); Guangzhou Novo Program of Science and Technology (Grant No.0501-330);Funds for Key Areas of Guangdong Province and Hong Kong (Grant No. 2011A011305004); Research Fund for the Doctoral Program of Higher Education (Grant No.20110172120026).

REFERENCES

Arenas, M., P. Barcelo, R. Fagin and L. Libkin, 2004. Locally consistent transformations and query answering in data exchange. Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, June 2004, Paris, France, pp: 229-240.

Arenas, M. and L. Libkin, 2005. XML data exchange: Consistency and query answering. Proceedings of the 24th ACM Symposium on Principles of Database Systems, June 2005, Baltimore, USA., pp: 13-24.

Ashton, K., 2009. That internet of things' thing. RFID J., 22: 97-114.

Boasson, M., 1995. The artistry of software architecture. IEEE Software, 12: 13-16.

Fagin, R., P.G. Kolaitis, R.J. Miller and L. Popa, 2003. Data exchange: Semantics and query answering. Proceedings of the 9th International Conference on Database Theory, January 8-10, 2003, Siena, Italy, pp: 207-224.

Fagin, R., P.G. Kolaitis, L. Popa and W.C. Tan, 2004. Composing schema mappings: Second order dependencies to the rescue. Proceedings of the 23rd ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 14-16, 2004, Paris, France, pp: 83-94.

He, B., D. Zheng and Z. Huang, 2005. A secured data exchange system. Microcomput. Appl., 21: 32-35.

Kolaitis, P.G., J. Panttaja and W.C. Tan, 2006. The complexity of data exchange. Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, June 27- 29, 2006, ACM New York, pp: 30-39.

Libkin, L., 2006. Data exchange and incomplete information. Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, June 27-29, 2006, Chicago, IL., USA., pp: 60-69.

Miller, R.J., M.A. Hernandez, L.M. Haas, L.L. Yan, C.H. Ho, R. Fagin and L. Popa, 2001. The clio project: Managing heterogeneity. SIgMOD Record, 30: 78-83.

Perry, D.E. and A.L. Wolf, 1992. Foundations for the study of software architecture. ACM SIG-SOFT Software Eng. Notes, 17: 40-52.

Kolaitis, P.G., 2005. Schema mappings, data exchange and metadata management. Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, June 13-16, 2005, Baltimore, Maryland, USA., pp: 61-75.

Popa, L., Y. Velegrakis, M.A. Hernandez, R.J. Miller and R. Fagin, 2002. Translating web data. Proceedings of the 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China, pp: 598-609.

Yu, C. and L. Popa, 2004. Constraint-based XML query rewriting for data integration. Proceedings of the International Conference on Management of Data and Symposium on Principles Database and Systems, June 13-18, 2004, Paris, France, pp: 371-382.

Zhang, Y.P., C.C. Zhang and H.P. Wang, 2000. An Internet based STEP data exchange framework for virtual enterprises. Comput. Ind., 41: 51-63.