Multiple Entity Types Wireless Broadcast Database System

Agustinus Borgy Waluyo[1] Feng Zhu[2] David Taniar[2]
Bala Srinivasan[2] Wenny Rahayu[3]

[1]*Institute for Infocomm Research (I^2R), Singapore*

[2]*Clayton School of Information Technology, Monash University, Australia*
{David.Taniar, Bala.Srinivasan@infotech.monash.edu.au}

[3]*Department of Computer Science and Computer Engineering, La Trobe University, Australia*
{W.Rahayu@latrobe.edu.au}

Abstract

In a wireless environment, data broadcast paradigm has been recognized as an effective and scalable mechanism to disseminate frequently requested information to a large number of clients. This paper presents the development of a multiple entity types wireless broadcast database system. The broadcast system is designed to serve transitive queries or queries that access related data items belonging to different entity types. The broadcast data are retrieved from a central database server. We apply three different data broadcast schemes including: (i), filtering technique for mobile device to perform transitive queries and display the desired data from incoming multiple entity types broadcast data items; (ii), the index broadcasting scheme designed to predict the arrival of the desired data to appropriately execute power conserving mode; (iii), incorporate multiple channel environments to allow mobile device to tune into multiple broadcast channels to receive the desired data. The proposed model uses a share indices price context to demonstrate the effective use of these data broadcast schemes.

1. Introduction

Broadcasting and receiving data in a wireless environment is a new dimension in data communication and processing [11]. With the limited bandwidth and power supply in mobile devices, data broadcasting has become the method of choice for disseminating information to a large user population [12]. This is due to communication asymmetry and scalability. From the power consumption viewpoint, communication asymmetry indicates that it is far more costly to send than to receive data; scalability demonstrates that it is independent of the number of users is providing a service for.

Broadcast mechanism refers to push or disseminate information or data instances to a number of clients through one or more broadcast channels and allows a mobile client to capture and select data items that s/he are interested in [3]. Access to data is sequential and the behavior of the broadcast channel is unidirectional which means the server disseminates a set of data periodically to a multiple number of users. With this mechanism, the request in not known *a priori*. The main challenge in periodic broadcast is to minimize query response time and tuning time of retrieving database items [2, 10]. In some cases, the response time is equal to the tuning time. As the data size increases over time, the required number of data items to be broadcast also increases accordingly. This situation may cause some mobile clients to wait for a substantial amount of time before receiving a desired data item. Consequently, the advantages of periodic broadcast strategy will be diminished. Multi-channeling method can be used to maintain a low query response time, whilst indexing method is used to minimize clients tuning time so that energy efficiency can be preserved.

Most data broadcasting mechanisms focuses on broadcasting flat files for the selective retrieval of single data record [5]. However, since most-real world applications databases involving multiple entity types, transitive queries accessing related data items belonging to different entities are very common. For example; in a relational database (RDB), join operations are required to relate data items from different relations. In this paper, the term entity type relates to a generic collection of data items with the same logical structure. Since this paper concerns with relational database system, the term entity types and tables are used interchangeably.

In this paper, we *present* multiple entity types data broadcast information system to serve transitive queries in mobile ad-hoc environment. We apply indexing and multi-channeling schemes to assist users performing query operation efficiently. The broadcasted data are retrieved from central database server. In order to demonstrate the effective uses of the proposed system, we use share price indices context. This work is an extension of our previous work, which focused on single entity data broadcasting [13].

21st International Conference on Advanced Networking and Applications(AINA'07)
0-7695-2846-5/07 $20.00 © 2007 IEEE

2. Related Work

Although many researchers like Barbara [1], Imielinski, Viswanathan and Badrinath [6] have argued the effectiveness of data broadcast strategy when involving a large number of users, to the best of our knowledge there is yet existing works that realize multiple entity types wireless broadcast database system as what we present in this paper.

Some location aware systems have been developed including the Mobile Shadow [4], which concerns with building prototype applications for proactive cell-based location aware services with mobile code. Kostakos et al [9] introduced a conceptual framework for designing as well as analysing pervasive systems in urban environment. The framework concerns with an integrated facet of urban design, which not only considers the architectural space but also the interaction spaces and use the information to support the movements, behaviors and interaction of people. Kjeldskov and Paay [8] presented a Just-for-Us project that develops mobile web services to facilitate new forms of interaction by adapting content to the user's physical and social context. The prototype system has been tested in Federation Square, Melbourne, Australia.

The majority of research on wireless data broadcasting centered on theoretical or mathematical models to address the issues affecting mobile computing. Furthermore, the research conducted also revolves around simulation models to highlight the benefits of various techniques or architectures. As such, there is a distinct lack of research pertaining to physical models that represent the techniques or architectures presented.

3. Architecture

The proposed system is designed to broadcast database items belonging to different entities via wireless network. The mobile device is able to receive the data, and then filter out the data before passing it to the user. There are three different services supported by the server application; (*i*) data broadcasting scheme, (*ii*) index broadcasting scheme, and (*iii*) multi-channeling.

3.1 Data Broadcasting Scheme

The server broadcasts data through per-determined channels. The client listens to those channels and retrieves the desired data. Figure 1 illustrates the broadcast data items belong to three entity types.

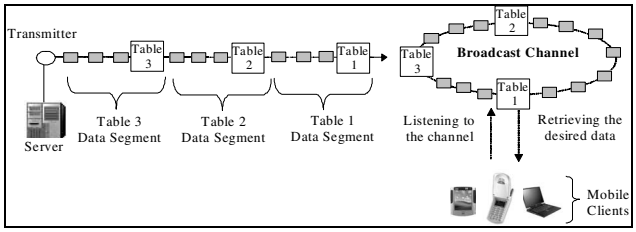


Figure 1. Data broadcast model

3.2 Index Broadcasting Scheme

Index broadcasting scheme, one of most important part of the application, is applied by using index trees. Index trees enable client applications to predict the broadcasting time of data items. The client application will automatically turn to power saving mode for sometimes upon receiving a relevant index key, and turn on the screen back once the desired data arrives. Figure 2 illustrate the indexing scheme.

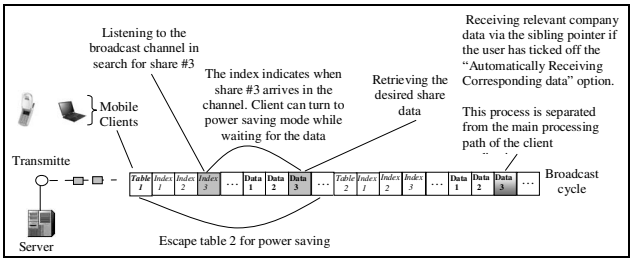


Figure 2. Index broadcasting model

3.3 Multi-channeling scheme

Multi-channeling relates to multiple logical channels in the wireless network. In this scheme, the index directory and the data segment can be broadcast into a separate channel. Client first tunes into the index channel to obtain the right index that relates to the relevant data item. The index indicates the time when the data will arrive in the data channel. Figure 3 depicts this model.

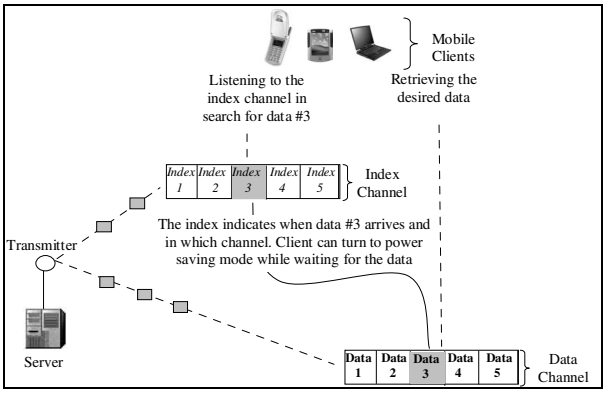


Figure 3. Data and Index Broadcasting in Multi-channel model

4. Implementation

The server application adopts two network protocols, TCP/IP and UDP, to perform different functionalities. TCP/IP protocol is used to transfer basic information to a client, so the client can initialize the running environment for itself before it is able to receive any further information. The protocol is also used to notify the client of the current broadcast type just before the client starts to receive data, ensuring the client to process data properly.

4.1 Technology

There are three components in the proposed model; they are the data source, a server application and a

client application. A Microsoft Access® database is used as the data source to contain share and company data which are accessed by the server application.

Two notebooks used in the application, one acts as a server and another acts as a Pocket PC-based Personal Digital Assistant (PDA). The operating system on both laptops is Microsoft® Windows XP Professional Edition. Both laptops have a wireless card in place. The server broadcasts data to the simulated PDA via a wireless network which is constructed with the standard wireless Ethernet networking technology 802.11b, while the client application receives services from the server.

Both server application and client application are programmed using Microsoft® Visual Basic® 6.0 Enterprise Edition. The server application retrieves data from the data source via ADO (ActiveX Data Objects), a built-in ActiveX object in VB 6.0 which enables connectivity with any sort of data source. The server application broadcasts data over a wireless network using UDP/IP [7]. The client application receives and filters out the data from the server. The client application is able to launch a TCP/IP connection with the server to obtain necessary data for building and controlling its running environment. The server application and the client application communicate with each other via Microsoft® Winsock controls.

The data source comprises of four entities which are linked with each other. The entity *tblIndexTree* and *tblCompanyTree*, contain index tree architecture for share prices and company information respectively. The index tree is used to determine when relevant data is broadcast. The other two entities, *tblSharePrices* and *tblCompanyDetails*, contain details about shares and companies. Figure 4 illustrates the relationships and structures of the entities.

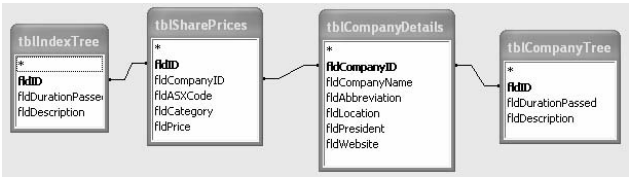


Figure 4. Entities of the proposed broadcast model

4.2 Server Application

There are two main features in the server application. One is using TCP/IP protocol, and another is multiple-entity data broadcasting. A TCP/IP connection is necessary at the very beginning stage for the client to make sure that the server is running there. More importantly, during the first TCP/IP connection, the server will send a set of data to the client, whereby the client is able to dynamically establish its running environment, so it is crucial to ensure the client to receive the dataset.

4.2.1 Initialization

Before the server application is able to broadcast data through the network, it has to perform initializations to establish its running environment. When the Initial button is pressed, two things will be executed. First, the server application initializes two socket controls (one is for TCP/IP connections with clients and another is for UDP data broadcast). Subsequently, it connects to the database and retrieves a set of data which will be stored into a global array. Once a client sends a request "INIT", the server will return the dataset to the client.

4.2.2 Environment Setting

Two setting options are available in the server application. One is related to multiple-channel broadcast, and another is related to broadcast period. Figure 5 depicts the setting interface.

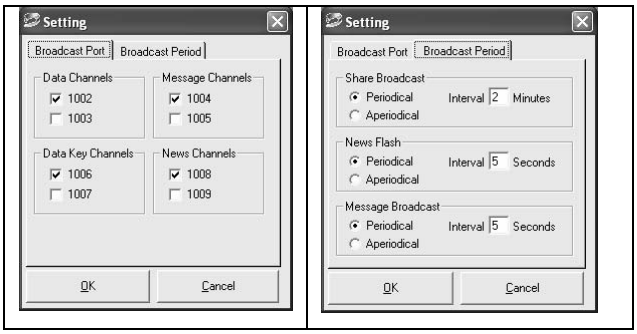


Figure 5. Setting Interface

4.2.3 Multiple Entities Data Broadcast

In this paper, broadcast data is derived from two related entities, one is share entity and another is company entity. Figure 6 illustrates the situation when the two data entities are broadcasted.

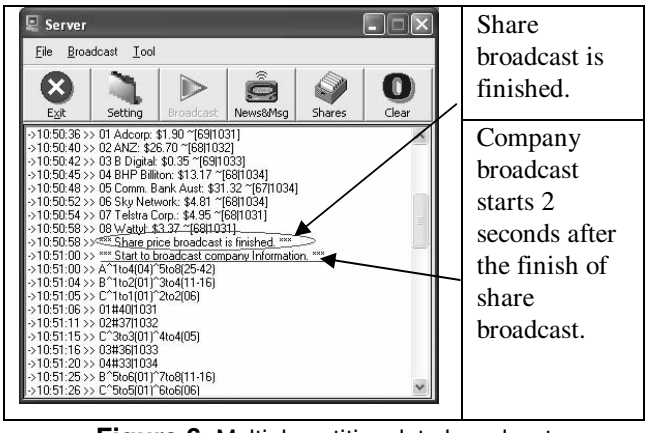


Figure 6. Multiple entities data broadcast

An entire broadcast starts with share broadcast. First of all, the server application retrieves share structure keys and the durations from entity *tblIndexTree*, It then creates three Timer arrays to address the broadcast of share structure keys, the broadcast of index keys and the broadcast of share prices respectively. Each st[illegible]

key and the corresponding duration will be assigned to one of the elements of the Timer array which is responsible for the broadcast of structure key broadcast as well as the timing of the broadcast.

The broadcast of index keys is associated with the broadcast of structure keys. When a Timer event of the structure key Timer array occurs, it will load an element of the index key Timer array into memory. The loaded Timer will be responsible for the broadcast of an index key which is linked to the structure key. For example, a structure key, "C^1to1(01)^2to2(06)", is broadcast in a structure key Timer event where an index key Timer is loaded for broadcasting the index key "01#39|1026". Another index key Timer is also loaded for broadcasting the index key "02#37|1010".

The broadcast of share prices is associated with the broadcast of index keys. An element of the share price Timer array is loaded, which is used to control the broadcast timing and the broadcast of a specific share prices. For example, in an index key Timer event, an index key, "01#39|1026", is broadcast. After that, the share ID "01" extracted from the index key will be used to retrieve share price from entity *tblSharePrices*. The retrieved share price will be store in an element of a structure array. An element of the share price Timer array is loaded to broadcast the share price.

The index of each array element is the corresponding share ID. For example, an element of the index key array, "*tmrBroadcastSingularIndex (5)*", is responsible for the broadcast of the share "05". The broadcast process of company data is exactly the same as share broadcast.

The user is able to browse all the share records in the database. Figure 7 shows what the share detail window looks like. The share detail window contains all details about a share, such as share code, company name, price, and so on. It also shows some other important information regarding the broadcast, such as port number and duration.

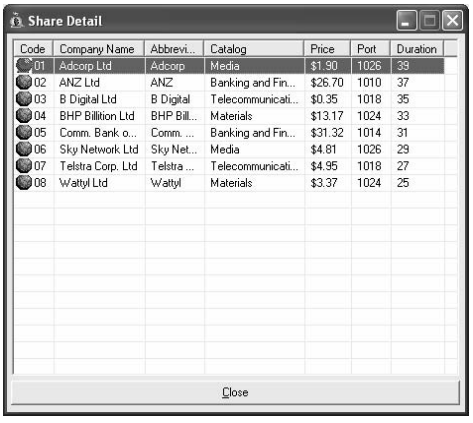


Figure 7. Browse Shares

4.3. Client Application

The client application performs a set of interrelated functions including sever connection, environment setting, data filtering, share and company selection. It has to obtain necessary data from the server in order to establish its own running environment.

4.3.1 Server Connection

The client application has to be successfully connected to the server via TCP/IP protocols before it is able to receive UDP data. Once the client application successfully connects to the server, it will get a set of data from the server to initialise its own running environment. The set of data contains the numbers of shares and companies, the share and company IDs, as well as the categories and cities. The TCP/IP connection will be disconnected by the server as soon as the client application receives the dataset. Figure 8 depicts the server connection interface.

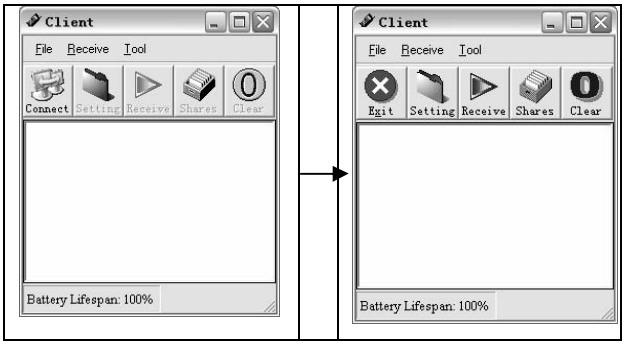


Figure 8. Server Connection

4.3.2 Environment Setting

- Power saving

There are two power saving options available in the client application. One is partial disconnection and another is complete disconnection. When partial disconnection option is ticked off, the screen light will be turned off if no mouse or keyboard events occur in 5 seconds. It will not be turned on until the desired data items arrive or a mouse or keyboard event occur. When complete disconnection option is ticked off, the device will be turned off instead of just the light. Consequently, battery power will be saved dramatically. Figure 9 illustrates the interface of power saving functionality.

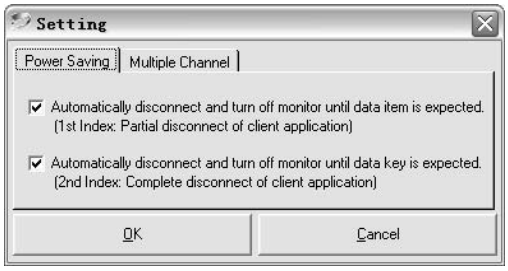


Figure 9. Power Saving Setting

- Multi-channeling

The client application is able to receive data from more than one channel. Figure 10 shows the interface of multi-channeling functionality.

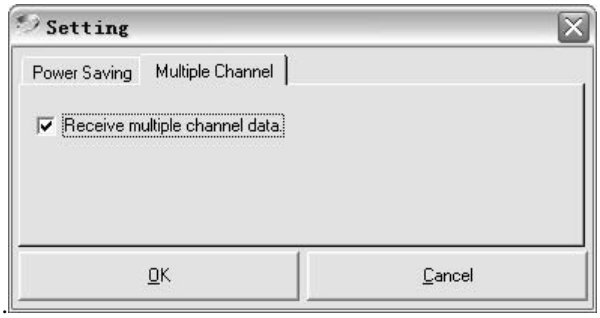
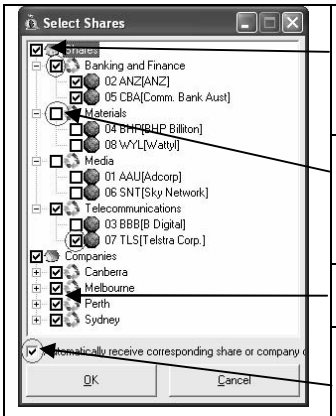


Figure 10. Multi-channeling Setting

4.3.3 Share Selection

Once the client application has initialized its running environment successfully, share selection interface is ready to use. The share selection interface adopts VB tree controls to provide flexible setting options. Figure 11 illustrates the interface of the share selection functionality. The interface provides several possible selection combinations to meet the end user's needs. It enables the end user to select a specific share or/and an industry, or all shares. The *automatically receive corresponding data* option provides additional facility to enable the user to receive corresponding share or company data.



Select a category, which means all share data belonging to this industry is desired.

Deselect a category, which indicates the client application should filter all Material shares.

Select a specific share.

Receive corresponding share or company data in terms of the setting.

Figure 11. Share Selection

4.3.4 Data Filtering

When the client application receives data from the server, the first thing it needs to do is to identify the data type. There are a number of data types involved in the application, such as structure keys, index keys, real data, news and messages, control information, and so on. Every piece of data from the server has a one-character header, which represents a specific type of data. Table 1 outlines the main data types used in the client application.

A significant feature in this data receiving module is that the client application will automatically turn to power saving mode when the requested data is in the next entity to arrive and turn back on when the entity is broadcasted. Figure 12 illustrates this feature.

Table 1. Data Types

Sample Data	Header Key	Data Type
B^5to6(01)^7to8(11-16)	B	Structure Key
01#39|1026	0 ... 9	Index Key or Data Item
News Flash	N	News
Server Message	S	Standard Message
$ #1	$	Broadcast Type

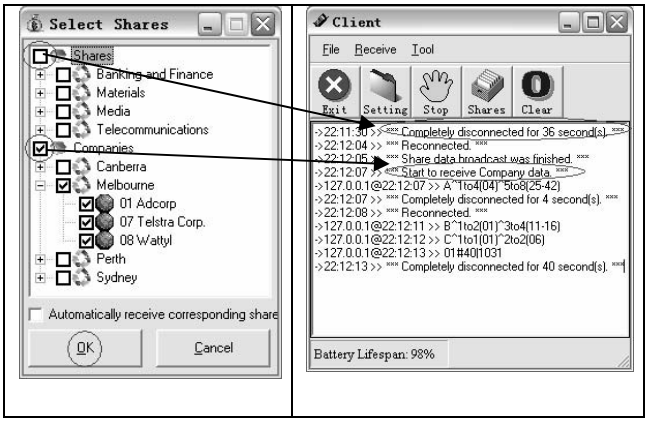


Figure 12. Data Filtering

When a user only selects one type data, for example, company data, the client application will figure out whether or not it should be completely disconnected, turn off the monitor, and escape the current broadcast duration. Once OK button is pressed, a TCP/IP connection will take place between the client application and the server application.

5. Case Studies

The client application is designed to receive desired database items in many different ways. There are three case study scenarios among them which are considered to be representative.

5.1 Case I – Receive Share Data Only

If a user is interested in share data only, the client application is able to receive just the share data that the user expects. Suppose the user is interested in all media shares, all he/she needs to do is to press the **Shares** button on the main window as what shown in Figure 16. Subsequently, the user needs to select the media category and deselect the checkbox on the share selection window. Once the **OK** button is pressed, the setting will immediately reflect to the client application.

In this case, we assume that the end user uses both partially disconnected and complete disconnected power saving mode, which is illustrated in Figure 13 In response to the above setting, the client applicati

process data filter functionality and power saving functionality together to minimize power consumption. Figure 14 provides a visual illustration on how the client application responds to the power saving setting.

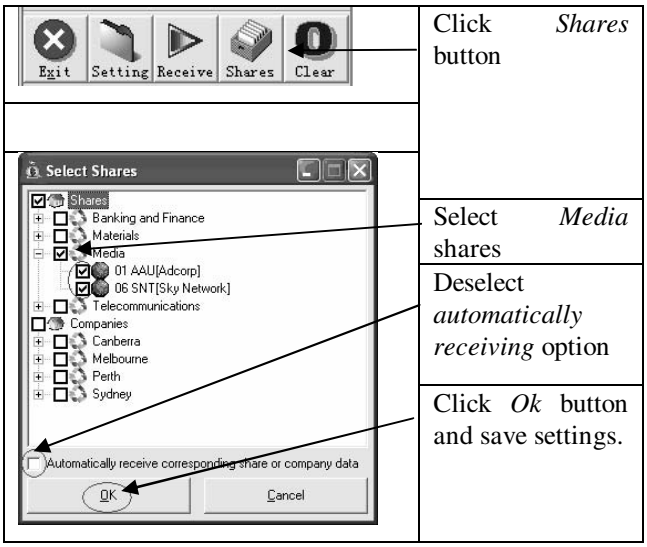


Figure 13. Share Selection Window – Case 1

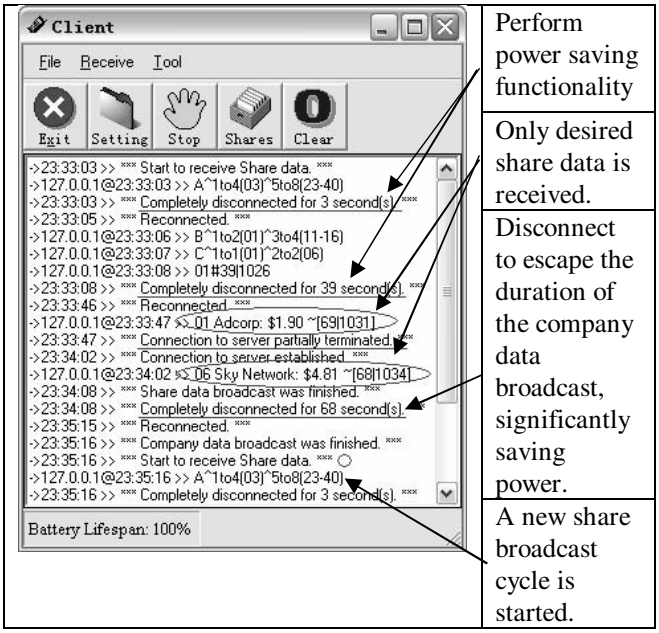


Figure 14. Data Filtering – Case 1

This process can be described as follows:

- Firstly, it connects to the server and asks for the current broadcast segment. In this case, the current broadcast segment is share data, so it starts to receive share data immediately; otherwise, it will completely disconnect for certain seconds until a new broadcast cycle starts.
- When the first structure key, "A^1to4(03)^5to8(23-40)", arrives, the client application figures out that one of the selected shares, Adcorp(01), is in the slot of "A^1to4(03)". Therefore it completely disconnects for 3 seconds, and turns off the monitor to perform power saving functionality.
- It reconnects after 3 seconds, followed by the second structure key, "B^1to2(01)^3to4(11-16)", arriving. It remains connected since the next structure key will arrive in a second.
- Then the third structure key, "C^1to1(01)^2to2(06)", arrives. Since the desired index key will come in a second, the client application remains connected.
- When the index key, "01#39|1026", arrives, the application will disconnect for 39 seconds and turn off the monitor again to save battery power.
- After 39 seconds, it reconnects again, and the expected share data, "01 Adcorp: $1.90 ~[69|1031]", gets to the screen. Then it partially disconnects for a random time until the next expected data arrives, which further performs power saving functionality.
- When the second share data, "06 Sky Network: $4.81 ~[68|1034]", arrives, the client application automatically reconnects.
- After the completion of the share broadcast, it is complete disconnected for a number of seconds to skip the broadcast duration of company data until a new broadcast cycle starts. Thus, it significantly saves battery power.

Based on the above scenario, the power saving functionality is performed very well. During the 133 second broadcast cycle, the client application turns off the monitor for about 115 seconds meanwhile the desired data is processed properly. Figure 15 shows the tuning time performance with and without indexing scheme.

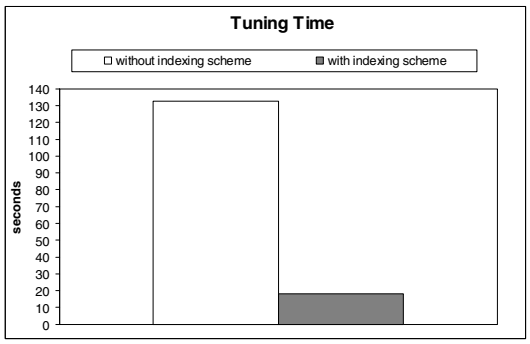


Figure 15. Tuning Time

5.2 Case II – Receive Company Data Only

The process of receiving company data is similar to the process above. Suppose a user has selected a company, and Figure 16 illustrates how the selection interface looks like. We assume that the user uses the same power saving mode as above. Figure 17 illustrates how the application client responds to the selection setting. The mechanism can be described as follows:

- Firstly, the client application detects that the current broadcast type is share and the user does not select any share data, so it immediately

disconnects for 18 seconds for the purpose of power saving.

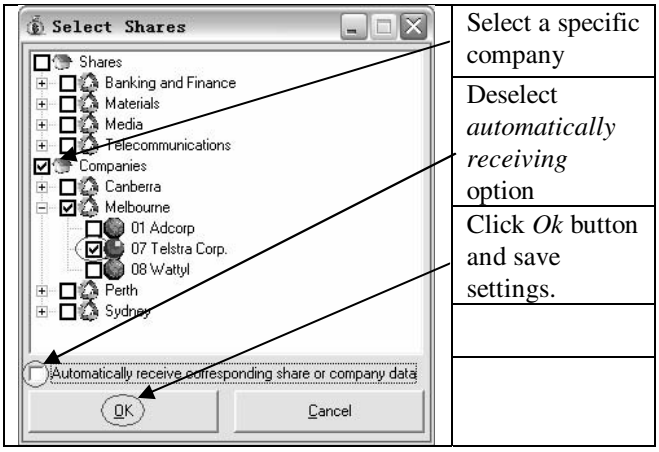


Figure 16. Company Selection Window – Case 2

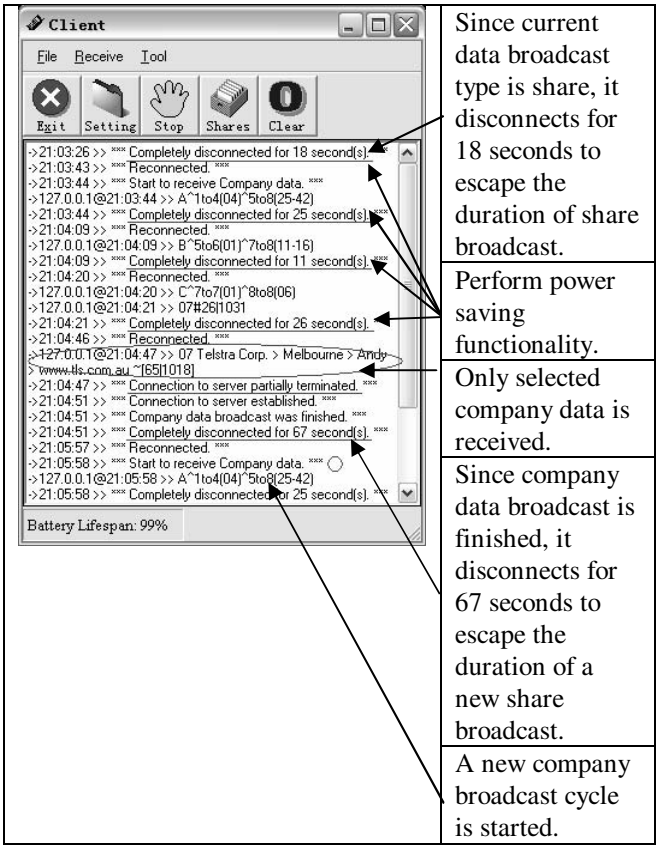


Figure 17. Data Filtering – Case 2

- After 18 seconds, it reconnects, and starts to receive company data.
- First company structure key, "A^1to4(04)^5to8(25-42)", arrives. The client application works out that the desired company is in the slot of "^5to8(25-42)", so it disconnects for 25 seconds to perform power saving functionality.
- It reconnects after 25 seconds, and receives the second structure key "B^5to6(01)^7to8(11-16)" which indicates that the third structure will come through in 11 seconds. In response to this, the client application disconnects for 11 seconds.
- It reconnects again and receives the third structure key "C^7to7(01)^8to8(06)". It remains connected since the index key will arrive in a second.
- The index key "07#26|1031", arrives. The client application figures out that the desired company data will arrive in 26 seconds, so it disconnects for that long.
- It reconnects again, and receives the desired data "07 Telstra Corp. > Melbourne > Andy > www.tls.com.au ~[65|1018]".
- Upon the successful receiving of the company data, the client application partially disconnects for uncertain time until the next data arrives, which could result in significant power saving.
- It receives a control message from the server, which implies that the current data broadcast is finished and the share broadcast is ready. Since the user does not select any share, so it disconnects until a new company data broadcast cycle starts.

In this scenario, the client application receives the desired company data with little power consumption. It will automatically disconnect from the channel if the current broadcast segment is not what is expected, as a result, battery power is saved significantly.

5.3 Case III – Receive Share Data and Corresponding Company Data

The setting of receiving share data and the corresponding company data is similar to scenario 1. The difference is that the *automatically receiving* option is ticked off in this case. Figure 18 shows the selection interface setting in this case. The process of receiving share data and the corresponding company data is also similar to scenario 1. However, more complex issues need to be considered. Figure 19 illustrates how the application client responds to this case.

The process of receiving data in this scenario is similar to the scenario 1 before the desired share data "08 Wattyl: $3.37 ~[68|1031]" arrives. So the discussion will start from the arrival of the desired share data.

- When the share data "08 Wattyl: $3.37 ~[68|1031]" arrives, the client application works out that the corresponding company data is expected. So it extracts the sibling pointer "37 ~[68|1031]", and passes it to a relevant processing module.
- The relevant processing module then extracts the duration "68" and the port number "1031". The duration, which indicates how soon the corresponding company data will come through, is used to initialise a special Timer. When the timer event is triggered, the port number is used to initialise a new UDP winsock control that is responsible for receiving the relevant company data.
- Once the timer event is triggered (which means that the relevant company data will come throu

couple of seconds), the UDP winsock control is activated and waiting for the arrival of the expected company data on the port. The UDP winsock control will automatically disconnects as soon as it receives the company data.

- The remaining process is the same as scenario 1.

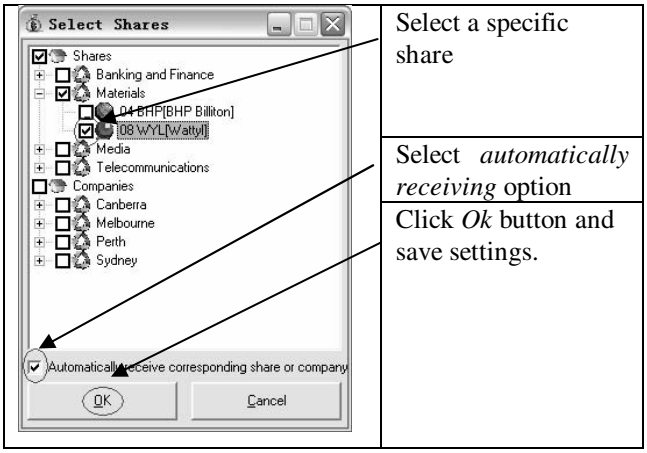


Figure 18. Share Selection Window – Case 3

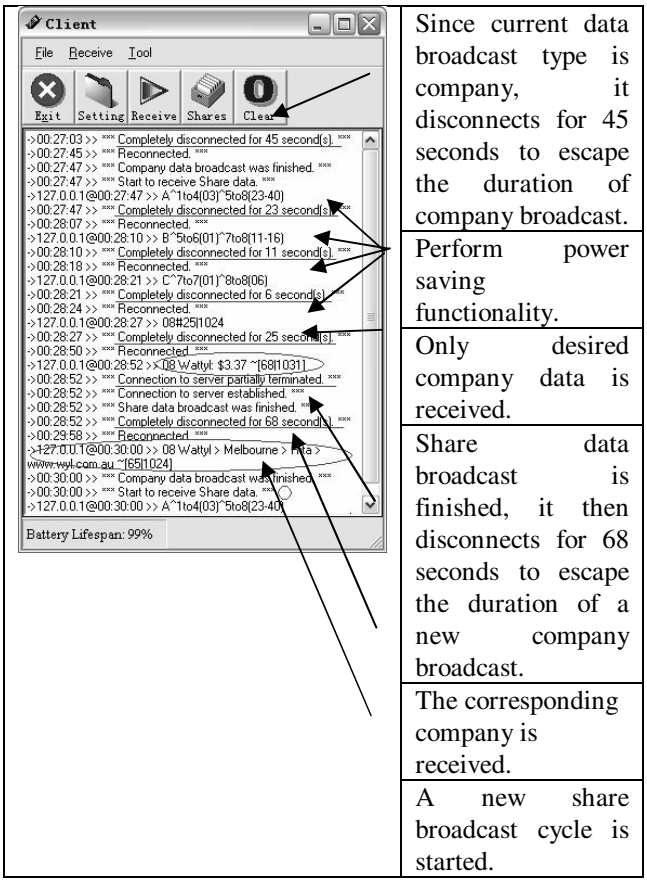


Figure 19. Data Filtering - Case 3

The mechanism of receiving the corresponding company data is much like a thread. It is self-regulated. It does not impact the rest part of the application and receives the company data in a very efficient way in terms of power saving. In practice, it only takes a couple of seconds for the UDP winsock control from waiting to actually receive the relevant company.

Another case might be to receive company data and the corresponding share data, which can be done exactly the same as this scenario.

6. Conclusion

Power conservation in mobile devices is a critical problem in mobile computing environment. Data broadcast strategies aim to reduce access time and tuning time. Although various data broadcast strategies emerge over recent years, research into practical and physical models has been lacking. This paper presents an effective and efficient way to broadcast multiple-entity type database items in a wireless multiple-channel environment.

7. References

1. Barbara, D.,"Mobile Computing and Databases-A Survey", *IEEE Transactions on Knowledge and Data Engineering*, **11**(1): 108-117, 1999.
2. Chung, Y.D. and Kim, M.H., "Effective Data Placement for Wireless Broadcast", *Distributed and Parallel Databases*, **9**(2): 133–150, 2001.
3. Franklin, M. and Zdonik, S., "Data in Your Face: Push Technology in Perspective", *In Proc. of the ACM SIGMOD,* pp. 516-519, 1998.
4. Fischmeister, S., Menkhaus, G. & Pree, W., "Context-awareness and Adaptivity Through Mobile Shadows", *Technical Report TR-C047*, Software Research Lab, University of Salzburg, Austria, 2002.
5. Imielinski, T., Viswanathan, S. and Badrinath, B. R., "Energy Efficient Indexing on Air", In *Proceedings of the ACM Sigmod Conference***,** pp.25-36, 1994.
6. Imielinski, T., Viswanathan, S. and Badrinath, B. R., "Data on Air: Organisation and Access", *IEEE Transactions on Knowledge and Data Engineering,* **9**(3): 353-371, 1997.
7. Jones, A. and Ohlund, J.,"Network Programming for Microsoft Windows", *Microsoft Press*, Redmond, Washington, U.S.A, 2002.
8. Kjeldskov, J. and Paay, J., "Public Pervasive Computing: Making the Invisible Visible", *IEEE Computer Computer Magazine*, **39**(9):60-65, 2006.
9. Kostakos, V., O'Neill, E., and Penn, A., "Designing Urban Pervasive Systems", *IEEE Computer Computer Magazine*, **39**(9):52-59, 2006.
10. Liberatore, V., "Multicast Scheduling for List Requests". In *Proceedings of IEEE INFOCOM Conference*, pp.1129-1137, 2002.
11. Myers, B.A. and Beigl, M., "Handheld Computing", *IEEE Computer Magazine*, **36**(9):27-29, 2003.
12. Su, C.J., Tassiulas, L. and Tsotras, V.J., "Broadcast scheduling for information distribution", *Wireless Networks*, Vol. 5, pp. 137-147, 1999.
13. Waluyo, A.B., Goh, G., Srinivasan, B., and Taniar, D., "On-Building Data Broadcast System in a Wireless Environment, *International Journal of Business Data Communications and Networking*, **1**(4),14-36, 20(