RTMonitor: Real-Time Data Monitoring Using Mobile Agent Technologies

Kam-Yiu Lam[1], Alan Kwan[1] and Krithi Ramamritham[2]

Department of Computer Science[1]
City University of Hong Kong
83 Tat Chee Avenue, Kowloon, Hong Kong

Email: cskylam@cityu.edu.hk

Department of Computer Science and Engineering[2]
Indian Institute of Technology Bombay
Mumbai, India 400076
Email: krithi@iitb.ac.in

Abstract

RTMonitor is a real-time data management system for traffic navigation applications. In our system, mobile vehicles initiate time-constrained navigation requests and RTMonitor calculates and communicates the best paths for the clients based on the road network and real-time traffic data. The correctness of the suggested routes highly depends on how well the system can maintain temporal consistency of the traffic data. To minimize the overheads of maintaining the real-time data, RTMonitor adopts a cooperative and distributed approach using mobile agents which can greatly reduce the amount of communications and improves the scalability of the system. To minimize the space and message overheads, we have designed a two-level traffic graph scheme to organize the real-time traffic data to support navigation requests. In the framework, the agents use an Adaptive PUSH OR PULL (APoP) scheme to maintain the temporal consistency of the traffic data. Our experiments using synthetic traffic data show that RTMonitor can provide efficient support to serve navigation requests in a timely fashion. Although several agents may be needed to serve a request, the size of each agent is very small (only a few kilobytes) and the resulting communication and processing overheads for data monitoring can be maintained within a reasonable level.

Keywords: real-time data, mobile agents, data monitoring, mobile computing

**Proceedings of the 28th VLDB Conference,
Hong Kong, China, 2002**

1. Motivation

Owing to advances in mobile communication technologies and devices, many new data-intensive applications are emerging, e.g., mobile stock trading systems and real-time navigation systems. Many of these new applications need to manage a large amount of real-time data items, which are used to record the real-time status of the entities in the external environment. Each access may be associated with a soft-deadline on its completion time and it is important to meet the deadline. Requests may be submitted as continuous queries [LPT99] and exist in the system until their deadlines have expired. For example, in a real-time traffic navigation system, a mobile client may generate a navigation request for the best path to its destination from its current position and the best path will have to be continuously tracked until a client reaches his/her destination. RTMonitor, is a real-time navigation system designed to support navigation requests efficiently (i.e., meeting the deadlines and providing correct results). To improve the scalability of the system, a distributed database system is used to manage real-time traffic data. The costs for monitoring the real-time data items for execution of complex queries should not be very high since their values are highly dynamic and the execution overheads of a complex query can be expensive [LTP99]. To ensure the correctness of the results, the data items used by the queries must be to be temporally consistent. Otherwise, incorrect results may be generated. However, maintaining data temporal consistency is a difficult problem in such a distributed mobile environment [DKPR01]. To support the navigation function efficiently, in the design of RTMonitor, we have adopted a cooperative and distributed approach using mobile agents to adaptively monitor real-time traffic data based on the urgency of the requests and the system workload. In the framework, the agents use an Adaptive PUSH OR PULL (APoP) scheme to maintain the temporal consistency of the traffic data for best path calculation. In formulating the *APoP* scheme, methods are designed based on the urgency of the requests, the system status and the dynamic properties of the traffic data to minimize data monitoring overheads.

2. Mobile Agents and Real-Time Data Monitoring

Mobile agents are software agents that can move from one host to another. Usually, a creating process initiates and dispatches the mobile agent to a target host where the agent executes in the environment provided by the target host. A mobile agent may communicate and cooperate with other mobile agents locally or remotely for achieving its function. It has been shown that the mobile agent framework is an efficient way for searching information, especially with complex conditions in a distributed environment [DNM99, PSP00]. Our RTMonitor system shows how this framework is also suitable for real-time data monitoring applications.

In real-time data monitoring, complex conditions may be defined over time-critical data items, which are distributed over multiple sources/servers. For example, in a navigation system, the traffic conditions for road segments are maintained by a distributed set of servers and each server maintains a directed graph corresponding to the road segments in a designated part of the system and the traffic conditions. The length of an edge in the graph indicates the time normally required to travel from the start point to the end point of a road segment. Due to the dynamics of the traffic conditions, the length of an edge varies with time. (Re)Calculating the best path could be very expensive when we want to have a close monitoring of the best path to satisfy the navigation request from a client such that whenever there is any change in the best path, the client is informed immediately. This is important when clients have specified deadlines on their arrival times. Using mobile agents to solve the data monitoring problems in such applications has many advantages:

- The distributed and autonomous properties of mobile agents can minimize the communication overheads and the amount of data to be transmitted amongst the servers for calculating and monitoring the best path for a client.
- It makes the system more adaptive to the changing situations of the environment.
- Since a mobile agent can execute asynchronously and autonomously, a fixed connection is not necessary and it can operate even under in disconnected environments.

3. Main Design Issues

RTMonitor assumes that the whole service area is divided into regions with each server responsible for managing the traffic data items and servicing the clients within a region. We adopt a cooperative and distributed approach for managing the real-time traffic data in which a two-level traffic graph method is used for organizing the traffic data and a mixed agent framework is designed to monitor the traffic graphs using an adaptive PoP scheme.

3.1 Two-Level Traffic Graph

Instead of building a complete global traffic graph, we use a distributed graph approach to organize the traffic data. Each region server maintains a local traffic graph based on the road connections and traffic data of its region. It also maintains a global virtual traffic graph for the whole system as shown in Figure 1. A local traffic graph connects to its neighbouring local traffic graphs through its external nodes. There are two types of external nodes: entry-nodes and exit-nodes. Through an entry-node, a path is provided from a neighbouring local graph into the local graph. Through an exit-node, a path is provided to go into a neighbouring local graph. The external nodes of all the local graphs form the global virtual graph for the whole system. The external nodes are pre-defined based on the connections of the roads in the whole system. Virtual graphs are constructed using periodic traffic updates from remote servers. An important advantage of using the two-level traffic graph scheme is that it can significantly reduce the data volume for building the global graphs and the searching overheads and the size of the graphs is greatly reduced.

3.2 Mix Agent Framework

In our proposed mixed agent framework, the system consists of two types of agents, stationary and mobile agents. Stationary agents are mainly used for managing the traffic graphs such that the temporal consistency of the traffic data of the graphs can be maintained in accordance with the client requirements. The mobile agents are used mainly for serving the navigation requests, and they will move along with the clients. At each regional server, there is a LocalServerAgent (a stationary agent) that represents the regional server for creating other agents at run-time. The GraphAgent (another stationary agent) sitting on top of the LocalServerAgent maintains the local graph of its region and a virtual graph. The GraphAgent generates GraphMonitors (mobile agents), which are dispatched and parked at remote regions to collect traffic data from remote GraphAgents for updating the virtual graph. QueryAgents (mobile agents), which are initiated by mobile clients and move along with the clients, generate requests to GraphAgent for getting the best path information on behalf of the clients.

3.3 Using PUSH/PULL for Data Monitoring

It is assumed that there are traffic sensors to capture the latest traffic data for road segments. A traffic update will be sent to the server of its region sporadically whenever its new value is significantly different from the previous value. In calculating the best path, it is important to use the latest traffic data (temporally consistent data). However, frequent re-computations of the shortest path will induce a heavy workload at the servers.

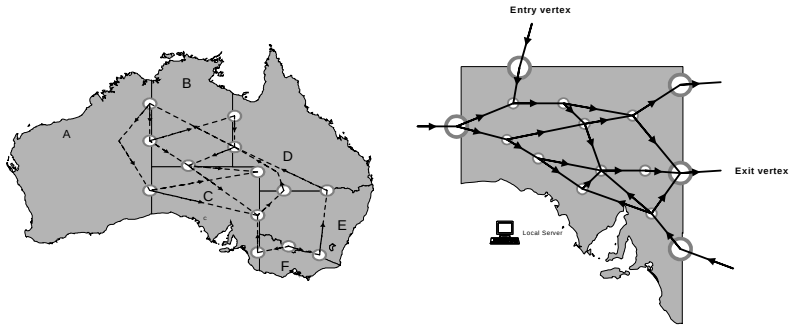

Figure 1. The global virtual graph (left) and a local traffic graph of region C (right)

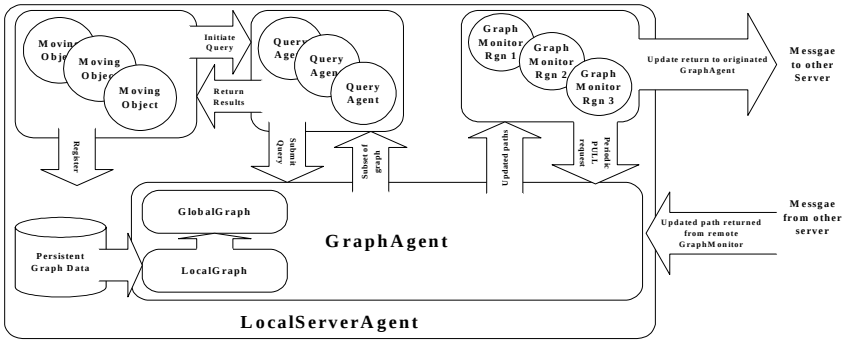

Figure 2. Agent Model of RTMonitor

To minimize the re-computation workload without significantly affecting the correctness of the navigation results, we have formulated an adaptive PUSH or PULL (APoP) scheme for data monitoring in which the monitoring period is defined based on the urgency of the request, the system status and the dynamic properties of the traffic data.

When the LocalServerAgent at a region receives a navigation request, it generates a QueryAgent for the client and at the same time, it generates a request to its GraphAgent to calculate the best path for the request using the shortest path algorithm, i.e., Dijkstra's algorithm, on its local graph and the virtual graph. If the destination of the request is in a remote region, it will send requests to GraphMonitors at the remote regions to collect the updated virtual path information connecting the originating region with other remote regions. The best path computed for the client consists of three parts: a path in the local graph of the originating region, a virtual path to the destination region, which is the region containing the destination of the request, and the local path from an entry point to the destination in the destination region. In addition, the GraphAgent also calculates the estimated arrival time at the destination and the slack time of the request, which is equal to the arrival deadline of the request minus the estimated arrival time. After serving the request for the first time, the GraphAgent will monitor the best path by: (1) monitoring its local graph, (2) sending requests to its MonitorAgents which are at the remote regions in the virtual path of the client, to monitor virtual path at the remote regions, and (3) sending a request to its MonitorAgent at the destination region to monitor the local graph at the destination region. If the initiating client enters another region, the monitoring job will be taken up by the GraphAgent of the new region and its MonitorAgents.

Initially, the system uses PUSH to monitor the best paths for the requests. The GraphAgent at a region may be serving multiple navigation requests. An important parameter in data monitoring is what should be the period for a GraphAgent to re-calculate its local graph and then regenerate new virtual path information. The new virtual graph information will be passed to the MonitorAgents at its region, and then the MonitorAgents will pass the information to their initiating GraphAgents for building new virtual graphs at their regions. The GraphAgent at a region determines the period based on the slack times of the current set of requests which it is serving, the workload at its server, the dynamic properties of its traffic data. In principle, if the slack time is smaller and the values of the edges change rapidly, a smaller re-calculation period will be used so as to provide a closer monitoring of the local graph. When the workload at the server is heavy, some less critical and urgent requests will be switched to PULL mode. In PULL mode, the QueryAgent of the client will generate path calculation request to the GraphAgent of its current region periodically and the GraphAgent will pull virtual path information from its MonitorAgents. The next time to pull from the client is defined again based on the slack time and the dynamic properties of the traffic data.

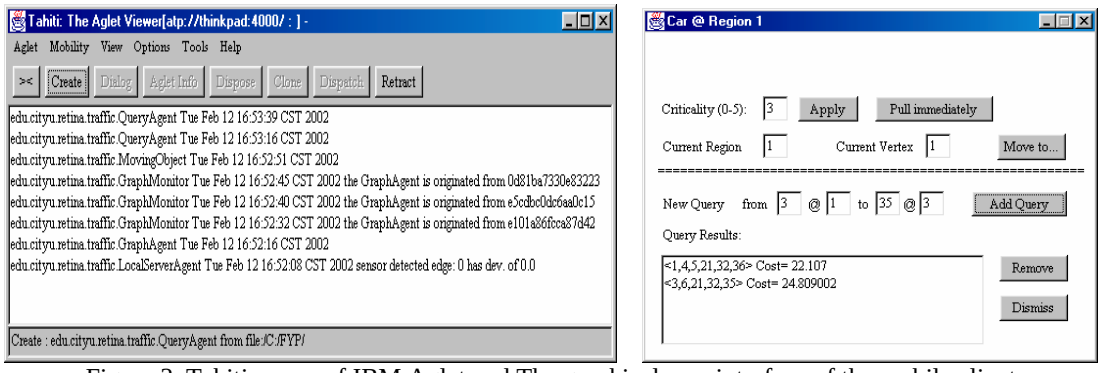

Figure 3. Tahiti server of IBM Aglet and The graphical user interface of the mobile client

In real-time data monitoring, it is important to minimize the set of data to be monitored. So, in our design, the monitoring of the best path will initially concentrate on the regions on the best path if the slack time of the request is greater than zero, i.e., it is estimated that the client can arrive before its arrival deadline. However, if the slack time is zero or very small, the GraphAgent, which is serving the request, may also send requests to its MonitorAgents, which are not on the virtual path, with the purpose of closely monitoring the best path. The actual number of regions to be monitored depends on the urgency of the request and the existing workload at each server. The same principle is used for serving a request in PULL mode.

4. Implementation

RTMonitor is implemented in Java using IBM Aglet .We use Tahiti as the servers for creation and execution of the agents. As IBM Aglet and the implementation is Java based, it can be running on any operating system that supports standard Java Virtual Machine (JVM) such as MS Windows NT and UNIX. A database is used to maintain the pre-defined connections of the roads and this information will be used for building the local graph and virtual graph at each server at system startup. In our implementation, we choose MS Windows NT and MSSQL for their simplicity of setup and availability. We have developed both thin clients, i.e., handheld PC running the Win CE, and ordinary clients, in Windows 98. To simulate the real-time traffic data, we have defined a process to generate traffic data randomly for each server.

5. Demonstration

In the demonstration, we set up a simulation environment with three servers using Tahiti where each server is responsible for maintaining a region of traffic data. At system startup, the servers build the traffic graphs based on the definitions of the road connections in the database and the initial traffic data from the traffic generation process defined for their regions. Client processes reside both in WinCE and Windows 98 environments. Several client processes are assigned to each region. Through a graphic user interaction a client specifies the criticality, current location, and other query input parameters. A client follows the path suggested by the system. In order to demonstrate the effectiveness of the designed techniques, we have included procedures in RTMonitor to collect performance statistics in run-time such as the number of messages transferred between Aglets, and computation cost of path searching, for analysis purposes. Furthermore, an event log is implemented to log the activities in the system while it is in operation. Figure 3 shows the screen captured from the execution of the Tahiti server when the Aglets are running, and the screen for inputting the parameters for a client.

References

[DKPR01] P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, “Adaptive Push-Pull: Dissemination of Dynamic Web Data”, in Proceedings of 10th International World Wide Web Conference , Hong Kong, May 2001.

[DNM99] P. Dasgupta, N. Narasimhan, L.E. Moser, P.M. Melliar-Smith, “Magnet: Mobile Agents for Networked Electronic Trading”, IEEE Transactions on Knowledge and Data Engineering, vol. 11, no. 4, 1999.

[LPT99] Ling Liu, Calton Pu, Wei Tang. “Continual Queries for Internet Scale Event-Driven Information Delivery”, in Special issue on Web Technologies, IEEE Transactions on Knowledge and Data Engineering, vol.11, no.4, pp610-628, 1999.

[PSP00] S. Papastavrou, G. Samaras and E. Pitoura, “Mobile Agents for World Wide Web Distributed Database Access”, IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 5, pp 802-820, 2000.

Acknowledgement: The work described in this paper was partially supported by a grant from the Research Grants Council of Hong Kong SAR, China [Project No. CityU 1078/00E].