A High Performance Scalable Data Collection System for Moving Objects

Azedine Boulmakoul
Computer Science Department
Faculty of Sciences and Technology (FSTM), University Hassan II, Mohammedia, Morocco

Lamia Karim
Computer Science Department
Faculty of Sciences and Technology (FSTM), University Hassan II, Mohammedia, Morocco

Ahmed Lbath
Computer Science Department
University of Grenoble 1
UJF/UPMF/Grenoble INP, CNRS, LIG UMR 5217, Grenoble, France

ABSTRACT
To help Locations Based Services systems to meet market demands, a performance and scalable collection framework was provided to gather different kinds of geographical data (from GPS devices, RFID, Data base transactions, etc.) based on the unified moving object trajectories' Meta-model. Basically, the collection framework offers components to collect spatio-temporal data from GPS enabling devices using .Net sockets, and benefits from executing all code in .NET Common Language Runtime to increase the system performance compared to unmanaged codes that decrease performance because of additional requirement security checks. In order to test the scalability of the proposed collection system, a vehicle tracking simulator was developed to generate and simulate tremendous spatio-temporal data of different moving objects. The main goal of this manuscript is to illustrate the importance of the collection framework and analyze the worst-case testing. Following this strategy, monitoring how scalable the proposed data collection framework is possible when dealing with very large and simultaneous moving objects trajectories datasets in real time.

Keywords
Trajectory data modeling, Trajectory data collection framework, moving object database, space time path, spatial data engineering, trajectory meta-model.

1. INTRODUCTION
Emerging and promising information added to dissemination technologies, like wireless sensor network, are becoming increasingly important regarding location based services. Furthermore, "80% of All Information is Geospatially Referenced" [1] and multitude of spatio-temporal positions data captured means are available with low cost and high quality, like mobile devices, IP-cameras, and electronic payment terminal. Therefore, GIS applications and services growth and needs unified model to deal and explore captured data, e.g. a system for destination and future route prediction based on trajectory mining [2], real-time monitoring of water quality using temporal trajectory of live fish [3], analyzing bird migrations trajectory[4], automatic monitoring of vehicles [5], etc. Not only, various fields of applications and researches need to collect, represent and explore knowledge of trajectory's data, e.g. Transportation and logistics, management, discovering animal behavior, marketing and business, criminology, monitoring territory, fleet management, etc. However, with the ever-growing competitive markets, companies also need to optimize costs of services and provide high quality of locations based services to survive and have share.

The main objective of this paper is to present a performance and scalable geographical data collection framework for moving object [6]. Indeed, this scalable testing framework distinguishes itself from others in two ways: firstly, by introducing a physical mobility simulator that generates geographical data without necessity of thousands of physical devices; secondly, by being based on the unified trajectories meta-model for a large use.

The remainder of the paper is organized as follows: section 2 will provide an overview of the Unified Moving Object Trajectories Meta-Model. Section 3 will present the system architecture. Whereas section 4 highlighting the scalable data collection conception and the adopted technology. Section 5 will provide an evaluation of the proposed framework when dealing with a very large amount and simultaneous collected data in real time. Finally, section 6 will provide conclusions of the proposed framework.

2. UNIFIED MOVING OBJECT TRAJECTORIES META-MODEL OVERVIEW

2.1 Basic definition
Several researches have been interested in presenting and modeling moving object's trajectory. The following is a summary of the basic trajectory presentations described in literature.

- Raw trajectory (see Figure 1) is presented as a recording of a sequence of geometric location in 2D spatial system (xi, yi, ti) of an object at specific space-time domain.

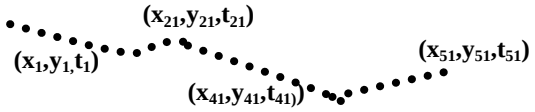

Fig 1: Raw trajectory presentation

- Structured trajectory [4] (see Figure 2) is defined as a raw trajectory structured into segments corresponding to meaningful steps in the trajectory trace. It is presented using four spatio-temporal components: the beginning B, end E, move M and stop S of trajectory (for example: travel).

Fig 2: Structured trajectory presentation

- Semantic trajectory [4] (see Figure 3) expresses the application oriented meaning using the four components (beginning B, end E, move M and stop S). Stop, move, begin and end are no more spatio-temporal positions, but semantic objects linked to general geographic knowledge and applications geographic data.

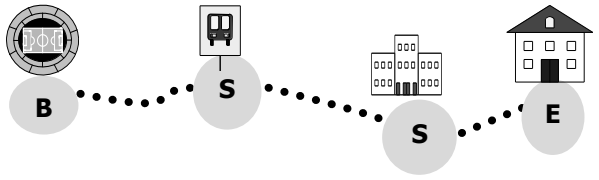

Fig 3: Semantic trajectory presentation

- Trajectory based on Region of interest presents movement patterns in both spatial and temporal contexts [7] (see Figure 4) by defining spatial neighborhood and temporal tolerance.

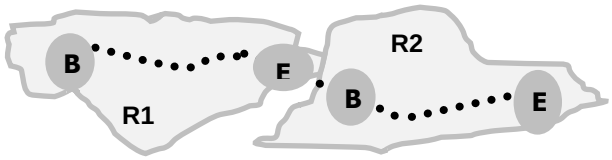

Fig 4: Trajectory with Region of Interest presentation

2.2 Unified Moving Object Moving Object Trajectories' Meta-Model

The Unified Moving Object Trajectories' Meta-model [6] describes a general meta-model that could be used by different application domains; it can also use an object approach and integrates previous trajectories models described in literature; like geometric, structured and semantic trajectories defined in [4] and [8-12]. It includes also the hybrid spatio-semantic model given in [5].

Using the space-time event ontology, the meta-model models Space according to OGC Spatial Data Model [13], Observation domain of trajectory, according to OGC Sensor Meta Model and OGC Feature Type [13], Physical and virtual activities between the beginning and the end of Space Time Path [14-15], sensors used for collecting moving object's traces, and Movement patterns using composite Region of Interest.

Based on ontology and Event approach, the unified trajectories meta-model allows presenting previous models [4, 16-19], added new interesting patterns to enrich and complete the model for a large use by location based services as Composite Region of Interest, Activities, space time path, and Mechanism of detection provided.

By instantiating the unified moving object trajectories meta-model, the formulation of complex spatio-temporal queries is simple for different locations based services. The following queries are formulated easily using our proposed model in PostGIS spatial database as:

- Example 1: The formulation of the query "finding all places (restaurant, supermarket and administrations) visited by a moving object (MO)" using the instantiated raw trajectory data model is shown in Figure 5.

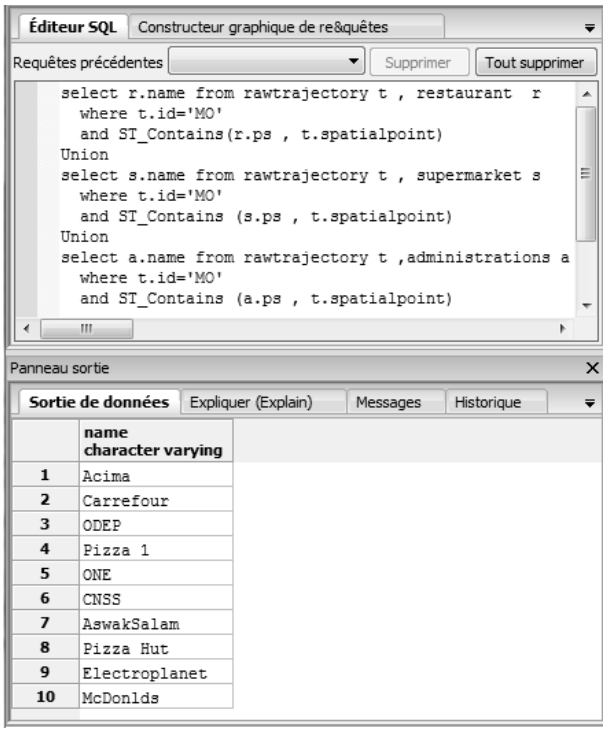

Fig 5: Example of query using the instantiated raw trajectory data model

- Example 2: The formulation of the query "finding all roads with moving object (MO) when it took him/her over 10min" using the instantiated semantic trajectory data model is shown in Figure 6.

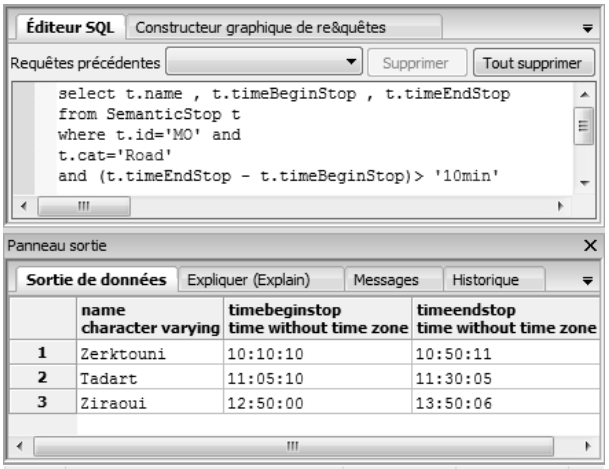

Fig 6: Example of query using the instantiated semantic trajectory data model

- Example 3: The formulation of the query "Figuring out the number of trajectories that visited each commercial region" using the instantiated trajectory with region of interest data model is shown in Figure 7.

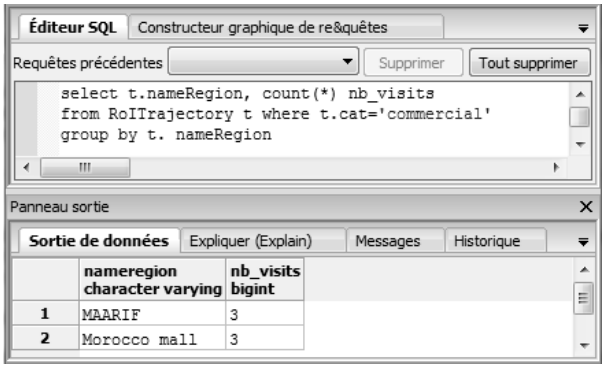

Fig 7: Example of query using trajectory with region of interest data model

- Example 4: The formulation of the query "finding the space time path (activities or process) of a specific person at a specific time" using the instantiated space time path data model is shown in Figure 8.

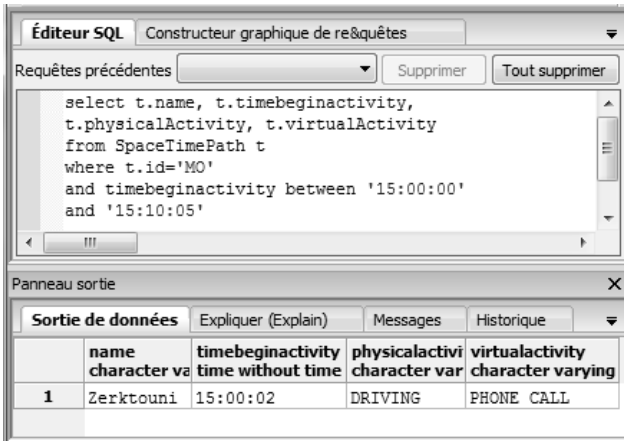

Fig 8: Example of query using space time path data model

- Example 5: The formulation of the query "which mechanism of detection used to capture information when the moving object was at the airport" is shown in Figure 9.

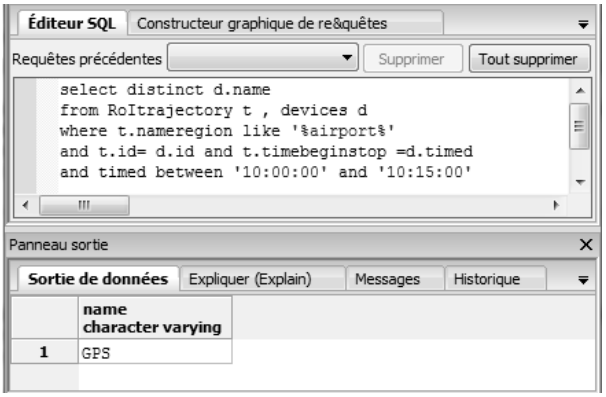

Fig 9: Example of query using mechanism of detection data model

3. SYSTEM ARCHITECTURE

The system architecture proposed for the unified moving object data model [20] is mainly based on Service-Oriented Architecture (SOA). By architecting trajectory's components using Services, and placing these services within a SOA messaging substrate, an efficient performance is provided that is easy to maintain and interoperable framework that increases several location based services for both performance and quality.

Service-Oriented Architecture (SOA) was used that is strongly guided by business. Moreover, it provides a uniform means to offer, discover, interact and use abilities to integrate trajectories services (e.g. tracking, visualizing and querying moving object's trajectory) with other applications and locations based services with measurable preconditions and expectations.

The architecture system, shown in Figure 10, is composed of five main layers:

- Data collection to collect spatio-temporal data from positioning captor.
- Trajectory pre-processing layer for efficient data pre-processing involving outlier detection, reducing duplicated data and measuring errors.
- Trajectory generator layer allows production of different kinds of trajectories needed by locations based services.
- Persistence manager layer that manages geographic, Trajectory-warehouse and trajectories data bases.
- Trajectory applications layer [20].

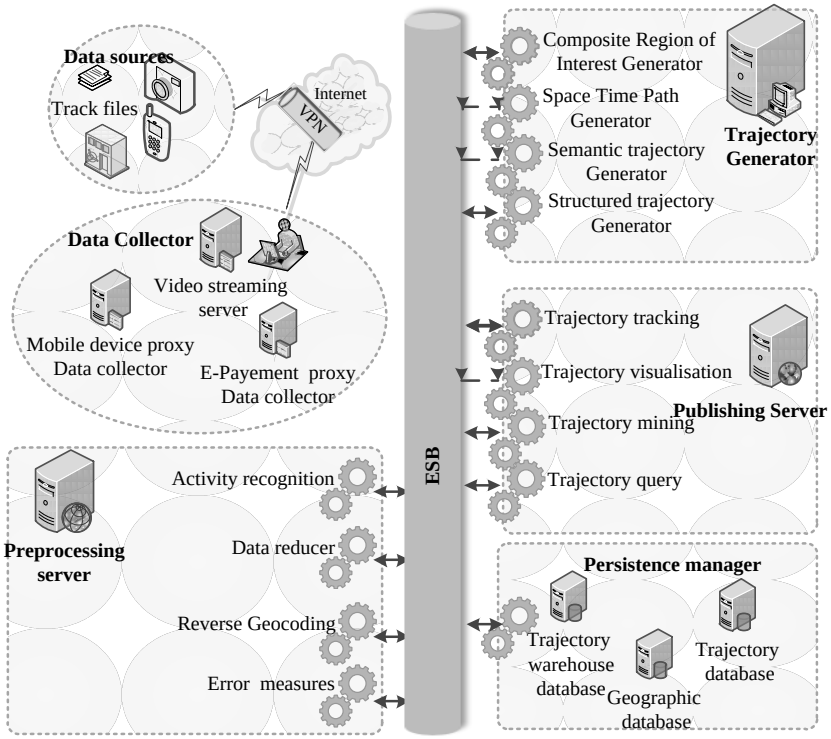

Fig.10 Unified moving objects system architecture

4. SCALABLE DATA COLLECTION

Providing performance, scalable and interoperable trajectories data collection framework improves locations based services qualities. In the following, a description of the collection system proposed for the Unified Moving Object Trajectories Meta-Model.

4.1 Possibilities for data collection

Using Web services to collect data is generally an easy way to adopt in SOA architecture; that is thanks to developed tools and frameworks supporting. However, collecting geographic data, especially in online mode that is based on tremendous short messages, consumes bandwidth due to the fact that Web services involve several SOAP layers of configuration that are incredibly expensive and time consuming (a sizeable differences number of bytes of data message and the number of bytes that are exchanged). Moreover, communication between collection server and sensors generally passes through a wireless network like GPRS or UMTS, which are still very expensive in terms of bandwidth, compared to bytes carried over cable or DSL.

The second solution analyzed to collect real time data is based on sockets. Parameter influencing decision is the response time of data collecting from a mobile phone equipped with GPS to the collection server. Therefore, the testing system of data collection was developed by using both technologies: Web services and Sockets. The result of this test, shown in Figure 11, demonstrates that sockets are faster than web services to gather data.

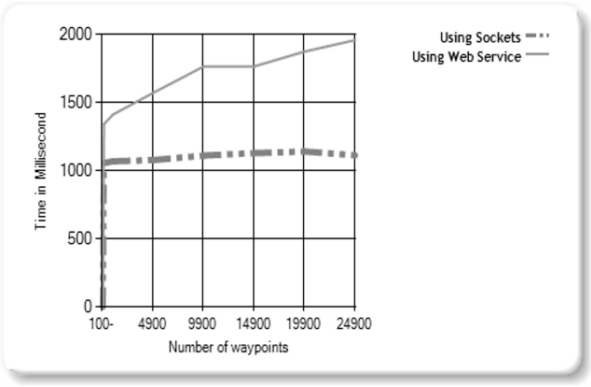

Fig 11: Variation of response time using Sockets and Web Services in a collection server

4.2 Scalable data collection Framework

Using Sockets interfaces aim to make real-time data collection applications possible between data sources (GPS equipped mobile phones, device camera, vehicles with navigational equipment or location based services, etc.) and data collection server. The client runs a program to send geographical traces of moving object, and the collection server runs a program to receive a massive geographical data in real time from many client programs. Hence, moving objects programs can send spatio-temporal data concurrently, and the server collection can handle clients concurrently using TCP as transport layer protocol.

4.3 Data collection server socket

Sockets are a low-level network programming features used in a standard protocol running on the same network. In data collection server, .Net sockets API are used to create an endpoint bidirectional communication between the collection server and client data sources. Sockets are used as a transport

mechanism for creating high-performance communication link between two endpoints (client & server) [21].

Sockets can be classified into synchronous and asynchronous mode depending on the type of operation performed on that socket. On one hand, when blocking (synchronous) sockets, programs are "blocked" and waiting to be dealt with until the operation on the socket is fully completed. On the other hand, for asynchronous sockets, the server application is permitted to respond events (non-blocking) upon the completion and execution of the process.

In the collection system, asynchronous mode was used with a pool of sockets objects and reusing them to collect geographical data and get notification to recognize error locations and successful operations. In reality, a socket pool increases performance on a server in a situation where there are many clients connecting and disconnecting quickly. In order to develop the scalable system collection for the unified trajectories meta-model, .Net sockets classes are likely used. Both classes, *System.Net.Sockets.Socket and System.Net.Sockets.Socket-AsyncEventArgs* are used by specialized high-performance socket applications and specifically designed for network server applications that require high performance [22]. In addition, communication between data sources simulators and server data collection is managed by event programming to collect in a non-blocking mode on different thread. As a result, none of the clients' sockets is suspended while waiting for the network operation to complete. The following diagram shows classes and activities diagrams that make up the data collection system.

4.4 Data collection class diagram

Class diagram, shown in Figure 12, describes potential utility classes of the Socket data collection Server. The SocketAsyncEventArgs class is a context object for the System.Net.Sockets.Socket class, which supports non-blocking network I/O in Net framework.

Using this advanced class, a high performance data collection system was provided. *SocketListener* class manages connection and messaging with multiple mobile clients who use the *acceptAsync* method of the *SocketAsyncEventArgs* class and may have a list of connections objects to store the accepted sessions. The *SocketClient* class uses the *connectAsync* method of the *SocketAsyncEventArgs* for the sake of connection. *SocketListener* class's main role is to listen and response to the messaging transactions of each socket session. Each connection has a *SocketAsyncEventArgs* instance for the messaging operations (receive and send). So, the *SocketAsyncEventArgs* pool manager would provide the instance library of many *SocketAsyncEventArgs* instances.

Buffers in *.NET* are managed by windows system and not by .Net framework. If a socket uses a byte array its buffer can cause memory fragmentation. That's why, we use the buffer block in *SocketAsyncEventArgs* object, and then we copy that data to another byte array. *Data Holder* has this byte array to which we can copy the data whereas the Buffer manager class enables buffers to be easily reused and guards against fragmenting heap memory. When each connection is created by the Socket Listener, the listener assigns the total buffer from buffer manager to the connections.

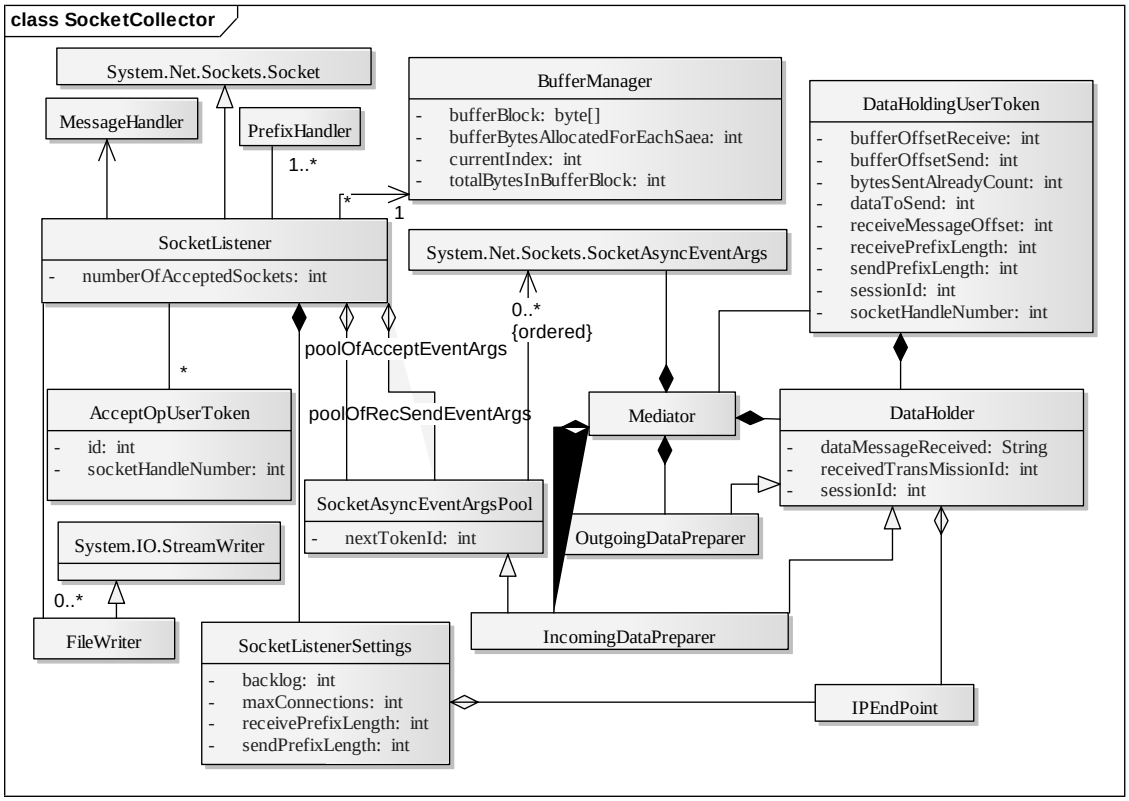

Fig.12 Data collection class diagram

There are two pools in context of *SocketAsyncEventArgsPool* to deal with accepting connection and receiving/sending socket operations. Once the accept operation completes, a reference to the socket transferred to another *SocketAsyncEventArgs* object as fast as required.*SocketListenerSettings* class holds a lot of settings that are passed to the *SocketListener*, Prefix Handler manages how to get data in the TCP buffer, whereas *MessageHandler* class creates the array where the complete message will be

stored and handled with collected messages data whether it requires one or many receives operations.

4.5 Data collection activities diagram

Activities diagram, shown in Figure 13, illustrates the collection socket activities diagram. The initialization of server data collection socket involves creating the socket, binding it to an address, and setting up the listen queue for mobiles clients to connect. Once an application client connects, it is added to the list of active clients.

When a connection is demanded, the server will accept or refuse the incoming connection sent from client's socket program. Later, this connection creates a new socket object asynchronously with a specific ID or gets it from the pool to collect trajectory's raw data. An active working socket object is used until the connection timeout ends or is reset by the client's program using the passive socket. Once trajectories raw data is completed and all bytes are transferred to the endpoint. So that, the socket is disposed and all resources released will be reused again.

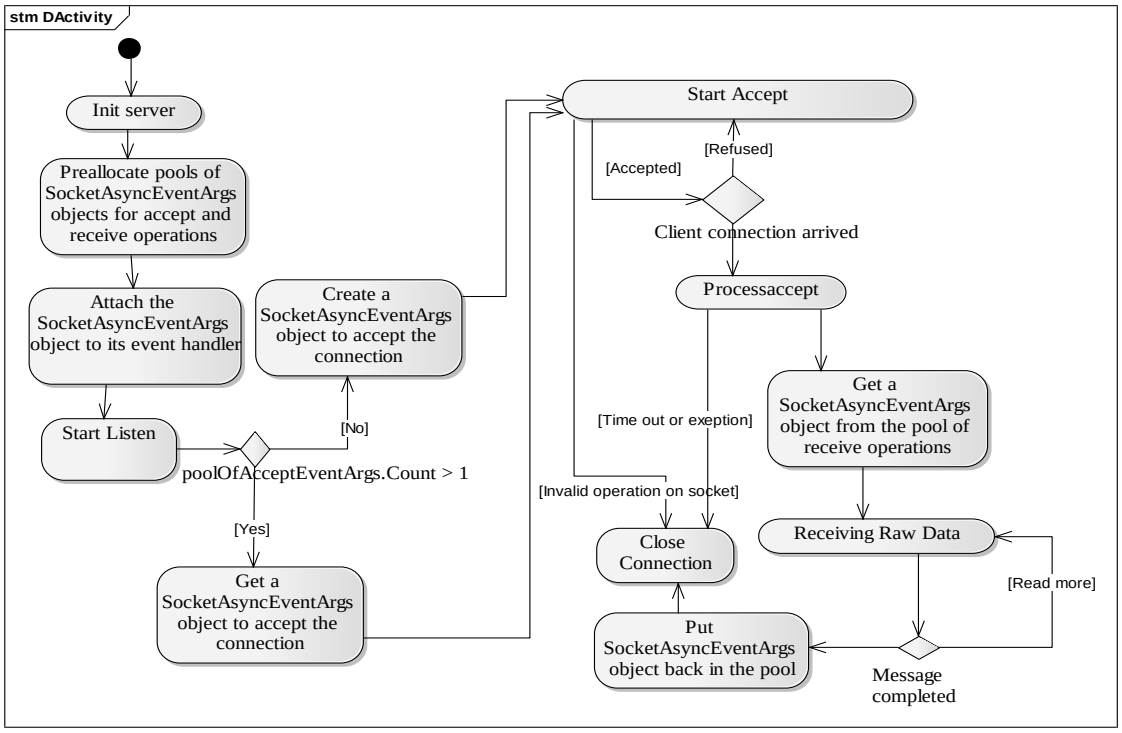

Fig 13: Data collection activities diagram

5. EXPERIMENTATION

In this section, there will be a brief analysis of the framework intended as a scalable moving object's data collection system for the unified moving object meta-model.

Basically, the collection framework offers components to collect spatio-temporal data from GPS enabling devices using .Net sockets, and benefits from executing all code in .NET Common Language Runtime to increase the system performance compared to unmanaged codes that decrease performance because of additional requirement security checks.

In order to test the scalability of the proposed collection system, a vehicle tracking simulator was developed to collect spatio-temporal data and simulate GPS devices. Then, a socket client's application, in mobile devices, sends data to server collection. Finally, the spatio-temporal data collection server gathers different form of geographical data. In the following, these components are described and the worst-case testing is analyzed.

5.1 Vehicle tracking simulator

Simulator, given in Figure 14, was developed to provide a case study for evaluating and testing the collection framework. It allows capturing real spatio-temporal data from vehicles. Users launch the simulator by choosing number of vehicles to track, time delay, and also define geographic points where the trajectory of each vehicle (moving object) starts and ends. This simulator was developed using GoogleMap Javascript API to generate driving directions and display them in the map, C# as programming language with ASP.NET Web Pages.

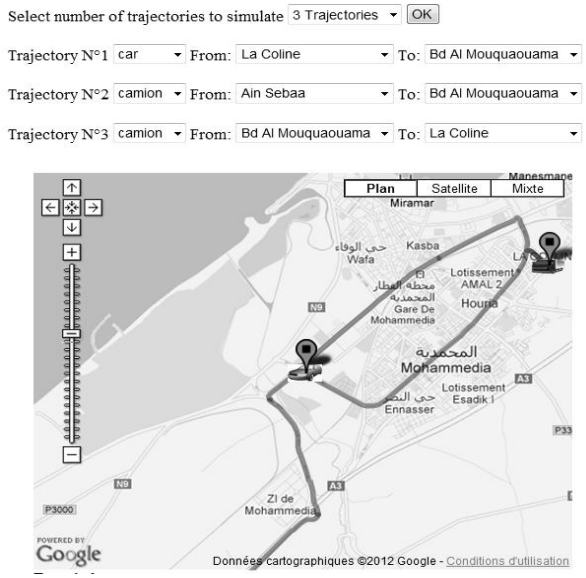

Fig 14: Vehicle tracking simulator

5.2 Worst-case testing

The goal is to test the performance and scalability of our proposed spatio-temporal data collection framework under worst-case conditions. The worst case testing strategy is as follows: First, the collection server socket is listening to connection and the performance monitor is launched to capture the system performance. Then, from the client's side, moving objects to track is chosen. After that, the number of moving object's trajectories is grown to simulate and analyze the performance of the collection server. Figure 15 describes the time consumed to collect and save collected data like when collecting one message from 1 till 10 000 moving objects, when taking 0.5s to 1s as time delay in a Windows 7 Ultimate computer, 64 bit Operating System, Processor core i5 CPU, and 4 GB installed Memory (RAM).The total time consumed for collecting and saving messages varies between 24 ms when collecting 1message from 1 moving object, 41s when collecting and saving messages from 10000 moving objects tracked, and about 4s to collect and save 50 000 messages from one moving object.

The structure of each message sent from GPS enabled devices is as follows: moving_object_id; latitude; longitude; date; time; observation. Whereas structure of messages collected using banking transactions or Electronic payment terminals are as follows: movingObjectId; electronicTerminalId; date; time; amount; creditCardKind; creditCardCryptedNmber; debit_or_credit.(geographic position of moving object was recognized from electronicTerminalId geographic).

As regards our proposed framework scalability, tests demonstrate that variation of system resources like Memory (% Committed Bytes In Use, Available Mbytes), Physical Disk (% Disk Read Time, % Disk Write Time, % Idle Time, Avg. Disk Queue Length, Avg. Disk Read Queue Length, Avg. Disk Transfers/sec, Disk Write Bytes/sec, Disk Writes/sec, Split IO/Sec), Processor (% Interrupt Time, % Processor Time is lightly and controlled. Therefore, our collection framework is able to handle a growing amount of work in a capable manner.

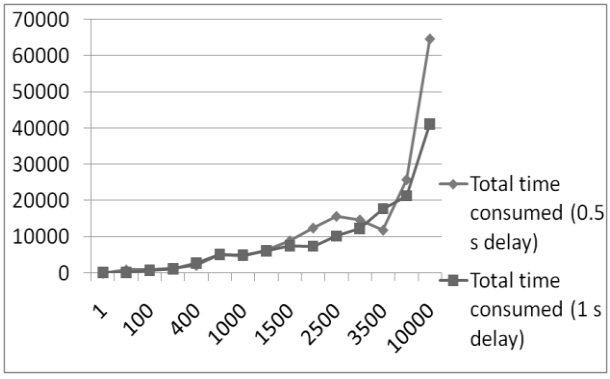

Fig 15: Data collection system total time consumed (collecting and saving data)

As a result, our collection system is scalable and efficient for different types of applications such as: Tourist Guides Community; fleet management, tracking of goods, trucks, and taxis; advertising; urban cleanliness and sanitation; protection of goods, vehicles and antitheft; Behavioral studies of human beings. The following table summarizes the proposed moving object data collection.

Table 1. Data collection characteristics.

Collection server characteristics (used for testing our proposed framework)	Windows 7 Ultimate computer, 64 bit Operating System, Processor core i5 CPU, and 4 GB installed Memory (RAM)
Trajectory data model	Based on the unified moving object trajectories meta-model
Trajectories data simulator	C#, Google Maps API, Asp .Net
Tested trams generated	Messages sent from GPS enabled devices: moving_object_id; latitude; longitude; date; time; observation. Messages collected using banking transactions or Electronic payment terminals: moving_object_id; electronic_terminal_id; date; time; amount; credit_card_kind; credit_card_crypted_number; debit_or_credit.
Interoperability	High
Performance test and scalability	Time consuming : 0.5s <= time delay between transmission <= 1s 24 ms for collecting and saving 1message from 1 moving object 41s for collecting and saving messages from 10000 moving objects tracked ≈ 4s to collect and save 50 000 messages from one moving object. Scalability: Variation of system resources like Memory, Physical Disk and Processor is lightly and controlled.
Complexity encapsulation	Using managed code, the collection system gets the maximum benefit from the features of the .NET framework like building applications in security by using code access security, and using the automatic lifetime control of objects, which includes garbage collection and scalability features.
Technologies	.Net Sockets, C#, ASP NET

6. CONCLUSION

All in all, this article included a brief outline concerning the basics of a scalable data collection framework for the Unified Moving Object Trajectories Meta Model.

Using Sockets interfaces aim to make real-time data collection possible between data sources and data collection server. Moving object device's runs socket program to send geographical traces, and the collection server runs a program to receive massive geographical data in real time from different clients program's and different sensors (GPS enabled devices, Database transactions, IP Cameras).

By using vehicle tracking simulator that generates and gives tremendous spatio-temporal data of different moving objects. Performed test have shown that the proposed collection framework is scalable and well performed when dealing with a very large geographic data in real time. The data collection will be used in the near future for storing trajectories in a spatial data warehouse in the goal of enabling knowledge discovery and patterns mining.

7. REFERENCES

[1] Fitzke J, Greve K. Frei oder umsonst? - Nutzergenerierte Geoinformation zwischen Freiheit und Kostenlosigkeit. In: Angewandte Geoinformatik - 22. GIT-Symposium. 1. ed., Wichmann, Berlin, 2010, pp. 732–741.

[2] Chen L, Mingqi L, Chen G. A system for destination and future route prediction based on trajectory mining. In:Pervasive and Mobile Computing, Elsevier Science Publishers, 2010, pp. 657-676.

[3] Heng M., Tsai TF, Cheng C. Real-time monitoring of water quality using temporal trajectory of live fish. Expert Systems with Applications, 2010, pp. 5158–5171.

[4] Spaccapietra S, Parent C, Damiani MD, Macedo JA, Porto F, Vangenot C. A Conceptual view on trajectories. Data and Knowledge Engineering, 2008, pp. 26–146.

[5] Kaplan ED. Understanding GPS Principles and Applications. Artech House Publishers, 1996.

[6] Boulmakoul A, Karim L, Lbath A. Moving Object Trajectories Meta-Model and Spatial-Temporal Queries. International Journal of Database Management Systems, 2012, volume 4, Number 2, p 35-54.

[7] Giannotti F, Nanni M, Pedreschi D, Pinellin F. Trajectory Pattern Mining. International Conference on Knowledge Discovery and Data Mining, 2007, pp. 330-339.

[8] Gomez L,Vaisman A. Efficient Constraint Evaluation in Categorical Sequential Pattern Mining for Trajectory Databases. In EDBT, 2009, pp. 541-552.

[9] Mouza C, Rigaux P. Mobility Patterns. Journal of Geoinformatica, Kluwer Academic Publishers Hingham, MA, USA, December 2005, Volume 9 Issue 4, pp. 297-319.

[10] Yan Z, Spaccapietra S. Towards Semantic Trajectory Data Analysis: A Conceptual and Computational Approach. 35th Very Large Data Base (VLDB), Lyon, France, 2009.

[11] Chen L, Mingqi L, Chen G. A system for destination and future route prediction based on trajectory mining. Pervasive and Mobile Computing, Volume 6 Issue 6, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, 2010, pp. 657-676.

[12] Tiakas E, PapadopoulosAN, Nanopoulos A, Manolopoulos Y, Stojanovic D, Djordjevic-Kajan S. Searching for similar trajectories in spatial networks.Expert Systems with Applications, 2008, pp. 772-788.

[13] Open Geospatial Consortium. Reference number of this OpenGIS® document: OGC 07-022r1 Version: 1.0 Category. http://www.opengeospatial.org/legal/; 2008.

[14] Shaw S L. A Space-Time GIS for Analyzing Human Activities and Interactions in Physical and Virtual Spaces. Center for Intelligent Systems and Machine Learning, 2011.

[15] Faisal I, Khokhar A, Schonfeld D. Object Trajectory-Based Activity Classification and Recognition Using Hidden Markov Models. Image Processing, IEEE Transactions, 2007, pp. 1912 – 1919.

[16] Meng X, Ding Z. DSTTMOD: A Discrete Spatio-Temporal Trajectory Based Moving Object Databases System. DEXA, LNCS 2736, Springer, 2003, pp. 444-453.

[17] Wolfson O, Xu B, Chamberlain S, Jiang L. Moving objects databases: Issues and solutions. Proceeding of the 10th International Conference on Scientific and Statistical Database Management (SSDBM), USA, IEEE Computer Society, 1998, pp. 111-122.

[18] Yan Z, Parent C, Spaccapietra S, Chakraborty D. Hybrid Model and Computing Platform for Spatio-Semantic Trajectories. 7th Extended Semantic Web Conference, Heraklion, Greece, 2010.

[19] Giannotti F, Nanni M, Pedreschi D, Pinellin F. Trajectory Pattern Mining. International Conference on Knowledge Discovery and Data Mining, 2007, pp. 330-339.

[20] Boulmakoul A, Karim L, Lbath A: System Architecture for Heterogeneous Moving Objects Trajectory Models Using Different Sensors. IEEE SoSE2012, the Seventh International Conference on System of Systems Engineering in Genoa, Italy, 2012, pp. 405-410.

[21] Daryn K. Get closer to the wire with high-performance sockets in .NET. Las Vegas, Nevada; MSDN Library; URL: http://msdn.microsoft.com/en-us/magazine/cc300760.aspx.

[22] MSDN Library; URL: http://msdn.microsoft.com/en-US/enus/library/system.net.sockets.socketasynceventargs.aspx