# Modelling and Simulation of a Distributed Battery Management System

Darren LIM[1], Student Member, IEEE, and Adnan ANBUKY[2]

[1] Electrical and Computer Engineering Department, University of Canterbury, Christchurch, New Zealand,
email : DTL13@student.canterbury.ac.nz, darren.lim@powerware.com
[2] Powerware (NZ) Limited, 39, Princess Street, P.O. Box 11-188, Christchurch 8030, New Zealand,
email : adnan.anbuky@powerware.com

*Abstract*—**This paper discusses the modelling and simulation of a distributed battery management system with a continuous and discrete-event simulation environment. The simulation model focuses on replicating the generic components within the system into model blocks to provide a structured approach in simulating battery networks. The simulation model also deals with three key network levels, which are the process level, the gateway level and the user or remote management level. A customised library of categorised network components allows for quick, easy and efficient model construction and testing of system or network performance. The simulation model possesses the ability to communicate with external applications such as sophisticated and specialised battery management activities, which allow short-term monitoring and control algorithms for simulating process level interaction, long-term monitoring and supervision at the gateway level and knowledge and data management at the remote management level. These features of the simulation model allow for encapsulation of the primary battery system aspects of battery operation management, network communication management and network data management. A case study of a typical 4-string battery network model is presented to illustrate the use of model building blocks in constructing, designing and configuring battery network simulation models and in determining network performance.**

*Index Terms*—**Battery management system, CAN, simulation model, Extend**

## I. Introduction

Battery networks used for standby or cyclic applications involve a number of strings of battery cells. Monitoring and control of these cells are facilitated through their associated devices or arrays of devices that communicate amongst each other or with a management centre through the local network [1,2]. Existing management systems are more concerned with process-level control methodology [3] rather than network architecture or organisation. Moreover, most existing networks are based on proprietary protocols, while standard open systems interconnection control protocols, like the Controller Area Network (CAN) or Local Operating Network (LON), are suggested, but not widely used [4,5]. Facilitating remote connectivity to the local control network may also be achieved through the use of TCP/IP over the Internet [6]. Furthermore, data and information visibility in these networks could range from real time to a few years of history. Eventhough these discrete technologies are available for providing a solution, appropriate integration to suit the battery management application needs to be examined.

Management of a battery network requires the monitoring of the state of the system in maintaining healthy operational condition and timely operational alarms. Involved management systems also include battery charge manipulation, allowing for testing and charge maintenance to occur. The three main categories of management have been identified as battery operation management, network communication management and battery and network data management. Fig. 1 shows an organisation to the battery management hierarchy.

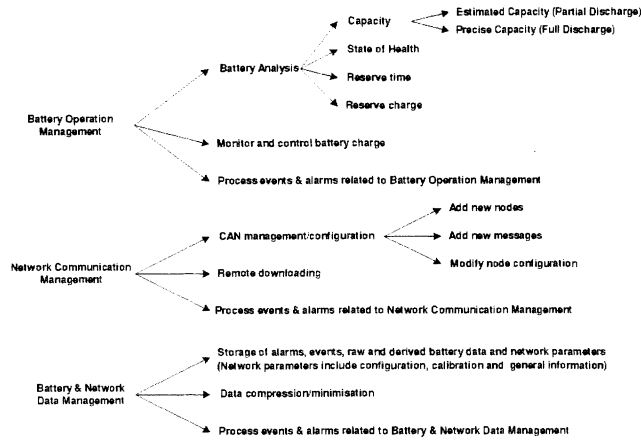



Fig. 1 Battery management system task organisation

The Extend modelling environment by Imagine That, Inc was chosen for the purpose of simulating the distributed battery management network organisation. Extend is a general purpose, open source, graphically oriented, continuous and discrete event simulation application with an integrated authoring environment and development system [7-9]. The discrete event scheduling utilities provided by Extend are suitable for modelling the communication features of the application, while the continuous and random number generation facilities are adequate for modelling the control, sensing, and monitoring processes of the application. Some other useful features of Extend for simulating and developing battery network models include: unlimited layers of hierarchical modelling capabilities for encapsulating node software functional modules, facilities for building custom model components and libraries and creation of user interfaces for facilitating a usable and recognisable battery network model. Extend also

allows for inter-process communication by providing functions for interfacing with other applications. For example, DLL and XCMD utilities provide a standardised interface for linking between Extend and languages other than ModL (Extend's internal programming language), such as Visual C++, C++, C, etc. In the context of battery management, this would allow sophisticated pre-written battery monitoring and control algorithms, such as capacity estimation or reserve time calculation, to be called from within a model block's code. The ODBC and serial port functionality packaged with Extend also allows for importing of battery data from existing databases or acquisition of battery data for simulation comparison. Extend also possesses a rich set of graphical and animation tools that are crucial for debugging network or node performance at the micro-level. Furthermore, the modular approach undertaken in the design of the distributed system organisation is easily translated into a viable network model with the building block model construction approach of Extend. This added advantage allows for rapid model construction as well as testing of network performance.

The development of a battery network simulation model is vital in allowing for the performance evaluation, comparison and testing of battery networks of varying size. Additionally, when network configurations become large, a direct comparison can prove to be difficult or even impossible, thus a valid simulation model would be essential in providing a performance evaluation or feasibility assessment. A network simulation model would also help in the validation and experimentation of proposed network organisations whether in terms of physical battery network topologies or scenarios involving multiple independent or inter-connected battery networks. If network node functionality requires an upgrade through the addition of new functional software modules, this can be easily implemented in the network model to determine if there are any adverse effects on the stability of the network organisation. A battery network simulation model also allows for operational tracking of network behaviour whether in the immediate future or the long-term.

## II. System and Model Organisation

The proposed system organisation for a CAN-based battery management system [10-12] is depicted in Fig. 2. The local network nodes are organised to be self-sufficient controllers that retain short-term battery history and exercise the sensing, monitoring and control activities on the associated network components, like a group of battery cells. Each node possesses the capability of interfacing with hardware modules relevant to sensing of battery parameters or charge manipulation over a sense line. Local and remote access (by remote users or the remote service centre) to the control network is arranged through an embedded low-cost gateway that facilitates Internet connectivity through HTTP over TCP/IP to the local CAN interface. Further network extension could be gained either through bridging more than one control network at the gateway level or using more than one gateway. The latter allows for better system reliability by relieving the control network of the bottleneck of a single point of communication failure. The designed system software has been based on utilising the bare minimum resources to keep system cost down and to avoid expensive and excessive off-the shelf middleware technology that could prove to be a burden to the limited processing power of the system rather than a benefit. By not having such middleware layers, it is possible to reduce communication overhead and have few system requirements. However, it is still possible to seamlessly integrate middleware technology such as CANOpen, CAN Kingdom or DeviceNet.

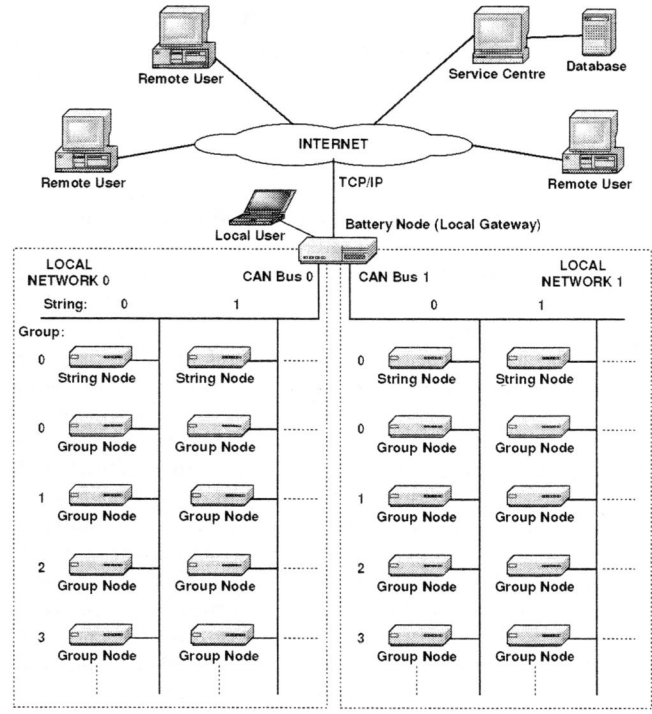



Fig. 2 Real-time distributed intelligence battery network architecture

The design of the simulation model focuses on the generic components within the battery management system and the primary elements of the communication channel from the process level to the user level. The former promotes reusable model blocks, while the latter defines the flow of data and information within the system. A customised model library has been developed as a design tool for model construction and configuration of battery management networks of varying size. The library contains different categories of related elements such as basic network components (i.e. various types of network nodes and their related configurations), battery network components (i.e. groups of battery cells, current shunts, etc) as well as generic network building blocks (i.e. bus connectors of varying size). Although the modelled network communication protocol in this case is CAN, other protocols to suit the application can easily be modelled by replacing the CAN model blocks with that of other communication protocols, such as LON or Ethernet.

### *A. Network Node Modelling*

The local network nodes consist of 3 logical types as shown in Fig. 2. These are the group node, the string node and the battery node. Each group node supervises a group of cells and is able to acquire individual cell voltages and group temperature, monitor cell condition and manipulate cell charge [5]. String nodes are situated at the beginning of

each string and are physically equivalent to group nodes, with the exception that each string node interfaces to a current shunt instead of a group of cells, to allow monitoring and acquisition of string current. Essentially, the battery node is the Internet interface or the local gateway. Battery nodes permit remote user access to the local network nodes in its domain through the use of a dynamic user interface with reliable two-way communication, and allows configuration of network parameters, enabling/disabling of network messages and storage of battery network relevant data.

The software organisation on each node encapsulates the core activities required for performing battery management. For group and string nodes, this includes sensing and storing raw battery data, processing of raw battery data into monitoring information such as state-of-charge or capacity, utilising monitoring data in controlling battery charge and communication of battery data to higher network layers. The battery node software organisation allows for higher-level data processing to facilitate sophisticated battery management functionality and communication and control of the local nodes through the use of command messages. Each node's software organisation is separated or organised into modules containing tasks that perform dedicated functions in achieving overall node functionality. Details of each node's modules and respective tasks have been discussed in [10].

When developing control architectures, simulation techniques are a better support for architecture configuration than analytical methods [13]. However, analytical results obtained from previous work on scheduling analysis in CAN and hard real-time operating system task communications [14-19] were applied to the CAN message set and modular node software design of the system and used as input parameters to the simulation model. Analytical results such as message transmission times and task or module execution times within each node were modelled as delay blocks within the simulation model. The delay blocks populate the communication channel from the process to the user. Additionally, network parameters such as bus transmission speed can be easily entered into the model as configuration parameters.

### *B. Distributed Data Modelling*

The data and information relevant to the network nodes are organised in three layers as shown in Fig. 3 [11]: the local node's data buffers for short-term data storage, the local gateway's database for medium-term data storage and the remote service centre's full-scale database for long-term data storage. These data storage facilities provide buffering and isolation from the user and process time domains. In the simulation model, elements of the distributed data architecture such as the group or string node's data buffers and the battery node's database are modelled as simple resource blocks with a fixed capacity. The capacity of the data storage facilities were determined by the actual physical memory limitations of the associated devices, namely the group and string node's microcontroller and the local gateway's memory chips.

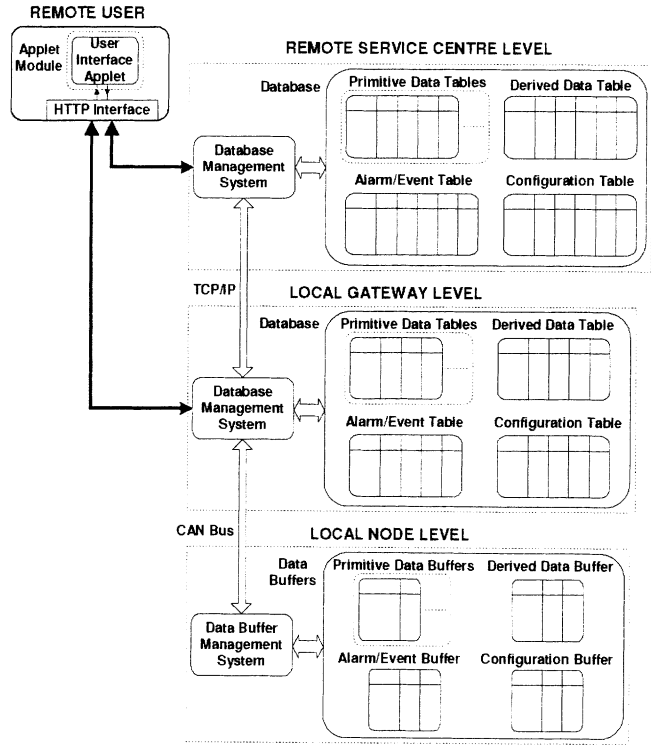



Fig. 3 Distributed data organisation

### *C. Network Communication Modelling*

The majority of communication on the bus or network traffic consists of regular or periodic transmission of data and status messages from string nodes and group nodes to the local gateway. These types of messages are assumed to be periodic and are given higher priority on the bus. The random aspect of CAN bus traffic occurs in the form of alarm messages from the string and group nodes and user initiated command messages from the battery node. Random messages are allocated lower priorities than periodic messages. The use of a global array in the simulation model to store all valid CAN message identifiers allows for easy modification and testing of various scenarios in allocating or assigning different priorities to different message types. In order to facilitate future expansion of process related functionality, gaps have also been left in the assigned message identifiers to allow for grouping of anticipated messages of similar content types. The simulation model allows for the transmission of these anticipated messages to be modelled on the network before proceeding with implementation, which has the desired result of determining the potential for network congestion.

Fig. 4 illustrates the flow of data within the communication channel of the system organisation. Items are entered into the system simulation model through the use of item generators at the process level. Some examples of items that are generated include cell voltage items, string current items and group temperature items. When these items are generated, they are assigned attributes according to a range of accepted physical values (e.g. an acceptable range of cell voltages would be between 1.8V and 2.5V, with a nominal voltage of 2.27V) and passed into the string or group node blocks. This procedure simulates the process of data acquisition. Combinations of these items are then batched into group data or string data items. After being delayed by a specific amount of time as determined by

analysis, the data items are passed to the string or group node's data buffer block. When messages are scheduled for transmission, the data items are formulated into message items and assigned priority and CAN message identifier attributes. Additional attributes associated with the message properties, such as message header length, number of messages, number of data bytes and length of message(s), are also calculated and assigned at this stage. The items then enter a FIFO queue block that releases items according to priority. This is an inherent feature of the implemented CAN communication modules or tasks.

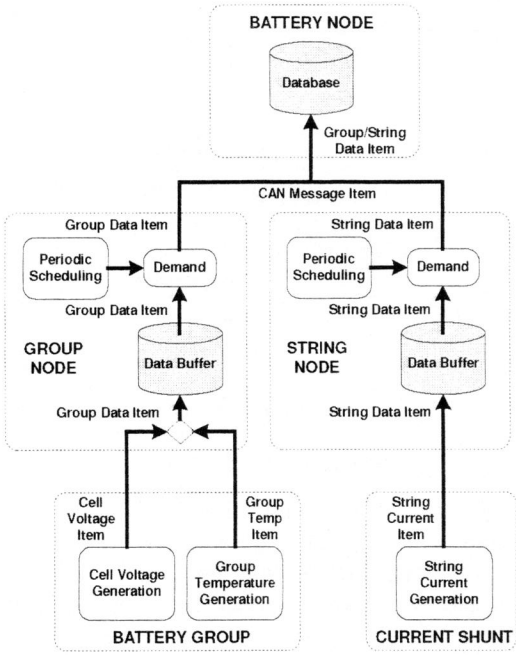



Fig. 4 Communication channel modelling

When the message item successfully gains access to the CAN bus, it exits the node model block and enters the CAN bus block. Here, delay related to message transmission or propagation time on the bus is simulated with a delay block. The amount of delay assigned to each message item is determined based on calculations of the message length attributes and the bus transmission rate parameter. As the message items exit the CAN bus block, they then enter the battery node model block, and into a sequence of blocks which model the acceptance filtering scheme of the CAN protocol. At this stage the message item is reformulated into a data item and attributes such as the CAN message identifier (which determines message content type) and message data length are then extracted to determine the appropriate database table block in which to deposit the item. Note also, that each entity is delayed according to analytical results during the stages of CAN message reception and data storage.

## III. A Simulation Case Study

An example case study of a 48V 4-string battery network simulation model implemented in Extend is presented in Fig. 5. Each string contains 5 nodes per string (1 string node and 4 group nodes) and the CAN bus rate was set at 125 kbits/s. The transmission of data and status messages from the string/group nodes to the battery node is assumed to have a periodicity of 1 second. Random command messages from the battery node were also included in the simulation. As depicted in the diagram, the size, scale and structure of the modelled battery network is easily discernible when initially viewing the model.

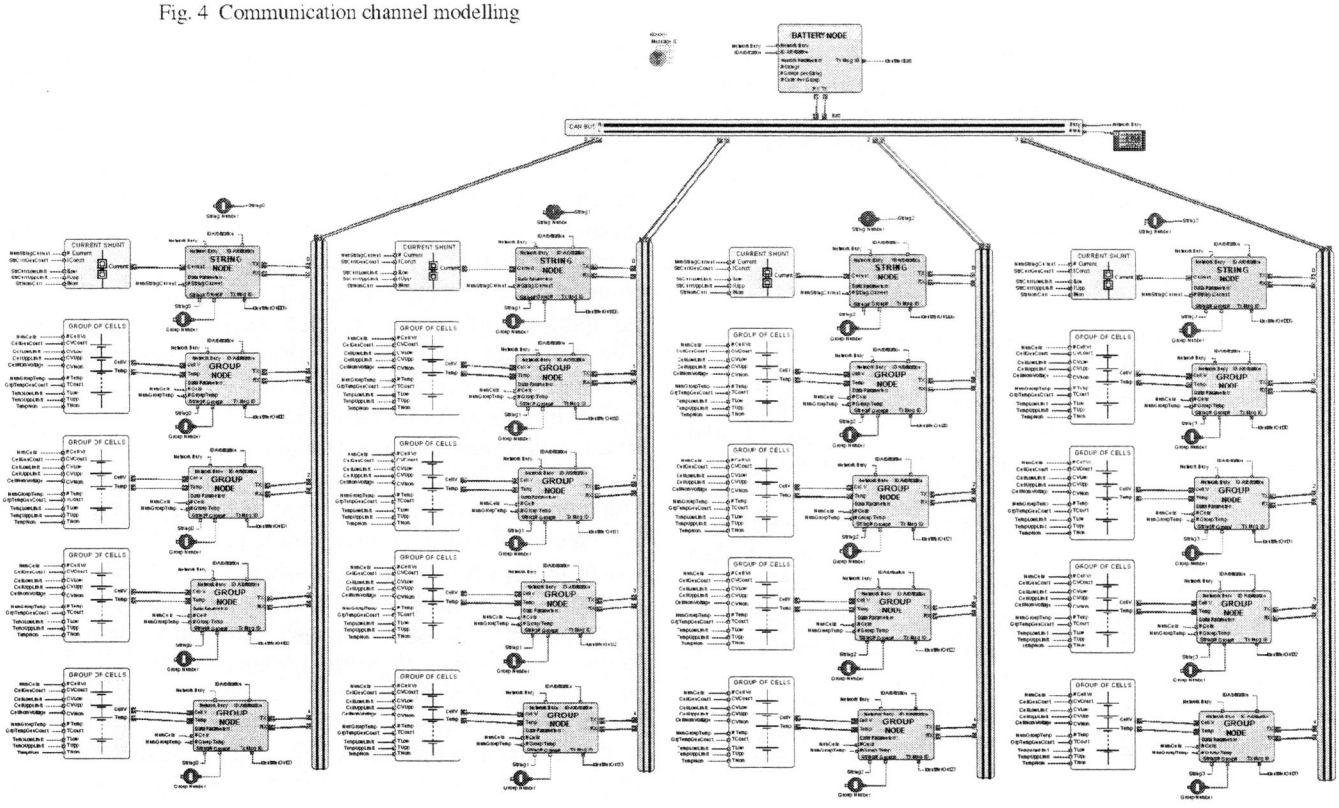
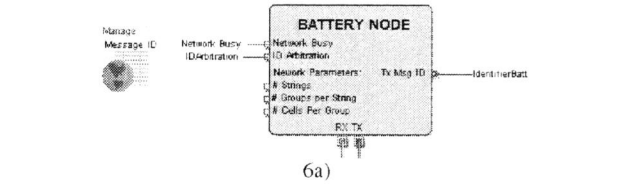
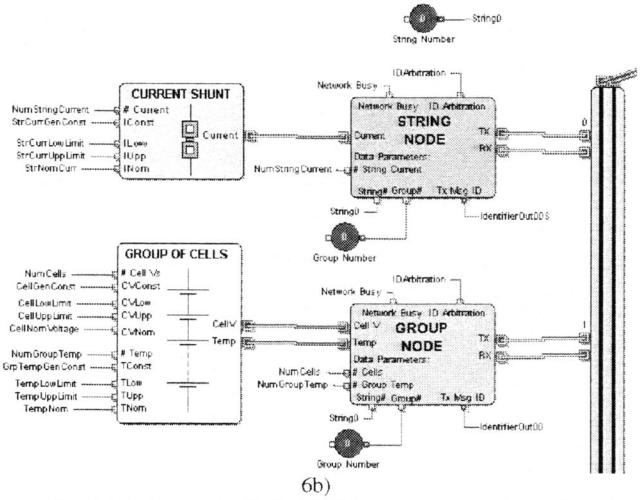


Fig. 5 Extend model of a 48V 4-string (1 string node and 4 group nodes per string) battery network

The diagram also illustrates the use of generic model building blocks such as the network node blocks, bus connector blocks and network component blocks (i.e. the current shunt block and group of cells block). Fig. 6a provides a more detailed view of the battery node block, while Fig. 6b illustrates the detail of the group and string node blocks and their associated group of cells and current shunt network component blocks. The former figure also illustrates the global array block for storing valid CAN message identifiers. These diagrams depict each block's use of connectors for configuration of operational parameters.




6a)




6b)

Fig. 6a) Battery node block detail, b) group node, string node and network component block details

Fig. 7 shows the results of group and string node bus traffic (bits/s) over a simulation period of 30 seconds for the example 4-string battery network. In the simulation, a zero releasing policy was used (i.e. all nodes began acquiring data at time *t=0*). The graph shows that the CAN network is able to absorb the initial response of having all network nodes release messages onto the bus without overloading the communication channel, while gradually achieving a stable bus traffic state well below the bus limit.

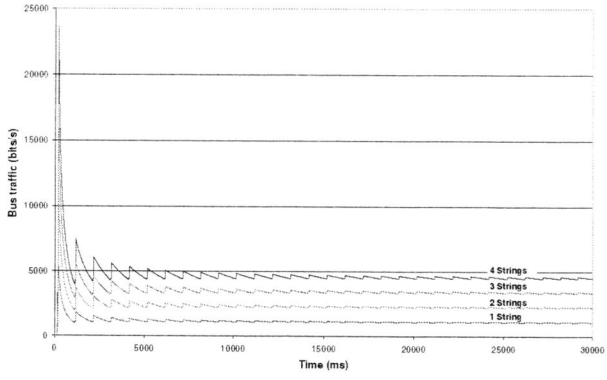



Fig. 7 Bus traffic for battery networks containing up to 4 strings

## IV. Model Limitation

The main issue experienced during the design of the model was the issue of preserving an individual item's attributes when items were batched or unbatched. Original attributes that were previously assigned to the individual items before being batched were found to be missing when the composite item was unbatched. For example, cell voltage items and temperature items were assigned cell voltage or temperature values before being batched into a group data item and transmitted across the CAN bus block. When the group data item was unbatched into its individual cell voltage and temperature items at the battery node block, the individual items no longer retained their assigned voltage or temperature values. Correspondence with technical support revealed an issue with the internal implementation of the batching and unbatching blocks, although a temporary solution was eventually found. In light of this however, a new version of the simulation package with an updated batching library has just recently been released. Further testing will be required to determine whether the problem of preserving the uniqueness of batched items discussed above is still prevalent in this latest version.

## V. Conclusions

In conclusion, a modelling and simulation environment for designing and performance testing of a proposed CAN-based distributed battery management system has been successfully implemented with a continuous and discrete-event simulation package. The simulation model focused on replicating generic components within the system into model building blocks that were stored in a customised library. These building blocks facilitated simple and efficient model construction and testing of network performance. The simulation model possesses the ability to communicate with external applications such as sophisticated battery monitoring and control algorithms to model system activity at the process level. In essence, the simulation model encapsulates the control, communication and data management aspects of the battery management system and is able to reflect the management capacity of the system organisation. A case study of a typical 4-string battery network model was also presented to illustrate the use of model building blocks in constructing and configuring battery network simulation models and the use of graphical tools in determining network performance.

Investigation into the modelling and simulation of HTTP over TCP/IP to allow for remote user and remote service centre access is currently being undertaken. This work would be used to determine the effect of another source of randomness on the system, namely data packet loss across the Internet, as well as determine the capacity of the battery node or local gateway in hosting a maximum number of remote users. Network scalability through the use of bridging or multiple gateways is also an area of the simulation model currently under investigation. These options would facilitate a broad range of battery network sizes and configurations to be modelled and tested before implementation.

## V. REFERENCES


[1] "Telecommunications Management Network (TMN)", *IEC Web ProForum Tutorials*, March, 2002, available: http://www.iec.org

[2] "Element Management Systems (EMSs)", *IEC Web ProForum Tutorials*, March. 2002, available: http://www.iec.org

[3] J. Chatzakis, K. Kalaitzakis, N. C. Voulgaris and S. N. Manias, "Designing a New Generalised Battery Management System", *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, Oct. 2003, pages 990-999.

[4] A. Anbuky, Z. Ma and D. Sanders, "Distributed VRLA Battery Management Organisation with Provision for Embedded Internet Interface", in *Proceedings of the International Telecommunications Energy Conference*, 2000, Phoenix, Arizona, USA, paper 37.2.

[5] A. Anbuky, P. Hunter, T. Johnson and D. Lim, "VRLA Battery Intelligent Node", in *Proceedings of the International Telecommunications Energy Conference*, 2002, Montreal, Quebec, Canada, paper 11.1.

[6] D. Bühler, G. Nusser, G. Gruhler and W. Küchlin, "A Java Client/Server System for Accessing Arbitrary CANopen Fieldbus Devices via the Internet", *South African Computer Journal*, no. 24, Nov., 1999, pages 239-243.

[7] D. Krahl, "Modeling with Extend", in *Proceedings of the Winter Simulation Conference*, 1999, Phoenix, Arizona, USA, pages 188-195.

[8] D. Krahl, "The Extend Simulation Environment", in *Proceedings of the Winter Simulation Conference*, 2002, San Diego, California, USA, pages 205-213.

[9] S. Redman and S. Law, "An Examination of Implementation in Extend, Arena and Silk", in *Proceedings of the Winter Simulation Conference*, 2002, San Diego, California, USA, pages 550-556.

[10] D. Lim, A. Anbuky and H. Sirisena, "CAN Based Network for Standby Power System Management", in *Proceedings of the Australasian Universities Power Engineering Conference*, 2003, Christchurch, New Zealand.

[11] D. Lim, A. Anbuky and H. Sirisena, "A Battery Distributed Management Network", in *Proceedings of the Systemics, Cybernetics and Informatics World Multiconference*, 2003, Orlando, Florida, USA.

[12] D. Lim and A. Anbuky, "A Distributed Industrial Battery Management Network": *IEEE Transactions on Industrial Electronics – Special Section on Distributed Network-Based Control Systems and Applications*, submitted August 2003, accepted for publication.

[13] P. Castelpietra, Y. Q. Song, F. Simonot-Lion and M. Attia, "Analysis and Simulation Methods for Performance Evaluation of a Multiple Networked Embedded Architecture", *IEEE Transactions on Industrial Electronics*, vol. 49, no. 6, Dec., 2002, pages 1251-1264.

[14] K. Tindell and A Burns, "Guaranteeing Message Latencies on Controller Area Network (CAN)", in *Proceedings of the International CAN Conference*, 1994, New Orleans, USA.

[15] K. Tindell, A. Burns and A. Wellings, "Calculating Controller Area Network (CAN) Message Response Times", in *Proceedings of the IFAC Workshop on Distributed Computer Control Systems*, 1994, Toledo, Spain, pages 35-40.

[16] K. Tindell and A. Burns, "Guaranteed Message Latencies for Distributed Safety-Critical Hard Real-Time Control Networks", Department of Computer Science, University of York, England, Tech. Rep., 1994.

[17] M. Ellims, S. Parker and J. Zurlo, "Design and Analysis of a Robust Real-Time Engine Control Network", *IEEE Micro*, vol. 2, issue 4, Jul./Aug., 2002, pages 20-27.

[18] K. Tindell and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems", *Microprocessing and Microprogramming – Euromicro Journal (Special Issue on Parallel Embedded Real-Time Systems)*, vol. 40, 1994, pages 117-143.

[19] D. B. Stewart, "Measuring Execution Time and Real-Time Performance", in *Proceedings of the Embedded Systems Conference*, 2002, Chicago, USA, class 203/213.