Abstract— Automated main distribution frames (AMDFs) are employed in telephone exchanges or central offices to provide automated connectivity between subscriber loops and central office equipment. The major drawbacks of current implementations however are cost and reliability. A simulation model for evaluating the behaviour of various configurations of a modular AMDF implementations is presented. The model introduces permanent equipment card redundancies and allows client profiles to be created in order for researchers and designers to explore typical scenarios. Various statistics are recorded throughout the simulation which allows easy comparison of multiple simulation run results. The model can be used to evaluate and determine the optimal modular AMDF design for various design constraints.

Index Terms— Automated Main Distribution Frame; Loop Management System; High Availability; High Reliability OMNeT++; Simulation; Modelling.

I. INTRODUCTION

The most significant cost associated with operating and maintaining a telecommunications service-provider business is labour [1]. In wireline telecommunication companies copper management is especially labour intensive since it typically involves several manual processes. A large portion of the labour entails performing moves, additions and changes to services at last-mile copper plants. These plants in the telecommunication service network provided by Telkom S.A. Ltd. in South Africa are known as Telkom Exchanges [2]. Employed in telecommunication, the Main Distribution Frame (MDF) is a passive device which terminates cables, allowing arbitrary interconnections to be made. Turning up (installing and activating) a service, for instance, requires on-site technicians to manually connect/disconnect jumper cables in a connection matrix in order to establish connectivity to a subscriber. Such a connection matrix board in the MDF may contain millions of cross-connection points [3], [4]. Population growth and an increase in the use of available services result in more frequent cross-connect changes. The result is an increase in call-outs and the associated travel costs. The increasing churn rate provides a compelling incentive for the automation of the jumper connection process.

Early attempts at automating the jumpering process entailed the implementation of a laser-tracking robotic arm to physically establish cross-connects in a switching matrix in [1], [3], [4], [5]. Current Automated MDFs (AMDFs) implementations employ any-to-any non-blocking actuator matrices to establish cross-connects. Possible actuators include solid-state switches, relays, micro-relays or stepper motors [6]. Chip-level advances, such as Microelectromechanical Systems (MEMS), provide an attractive alternative to realising compact switching matrices.

The main components of the access network are the copper cables themselves and their termination equipment. Copper lines have existed since the inception of telecommunication networks; for many years this asset has been viewed as old-fashioned and prone to be replaced by fibre. Recently, it was reported that a multibillion-rand project has been launched to connect over 2.5 million homes in six major cities in South Africa with a high-speed fibre network [7]. However, the network will not be completed until the later stages of this decade. Technological advances are still being achieved in copper today; speeds in excess of 100 Mbit and more are achievable with copper in new xDSL developments [8]. It would therefore still be beneficial to manage last-mile copper loops with the implementation of an AMDF. In order to investigate the effect of various configurations, a simulation model of the AMDF must be created and evaluated.

Complex systems often require the development of a software model to simulate the behaviour of the system under various conditions. Simulation tools have been widely used in network research, especially during the development of new systems to ensure that the optimal solution is chosen. A number of network simulators exist, including NS-2 [9], OPNET [10] and OMNeT++ [11], [12], [13]. OMNeT++ is an appropriate simulation environment for network simulation because of its public-source modular design, clear structure and strong Graphical User Interface (GUI) support. More importantly, OMNeT++ also permits the simulation of circuit-based simulations (black box functionality) along with packet-based simulations. Due to its flexibility, OMNeT++ has...
gained widespread popularity for any type of simulation.

An extensive, reconfigurable model of a modular AMDF was created using OMNeT++. The model allows the user to setup various configurations with any number of clients and AMDF modules. The evaluation of typical scenarios concerned with AMDFs with various design constraints (such as cost and power consumption) can be investigated.

The paper is organised as follows. Section II provides a brief description of MDFs, the problem with current frames and the implementation of AMDFs. Section III describes the OMNeT++ environment and why it was selected as the development tool. Section IV presents the AMDF simulation model. The performance of the model is evaluated in Section V. Concluding remarks are made in Section VI.

II. AUTOMATED MAIN DISTRIBUTION FRAMES

A. Role of the MDF

A MDF acts as an interface where permanent outside telephone lines terminate within a telephone exchange [1]. At this termination point, all the outside lines are interconnected to specific telecommunication equipment to provide individual subscribers with their appropriate telecommunications service, such as DSL or voice switches. In larger Central Offices (COs), apart from MDFs, Intermediate Distribution Frames (IDFs) are used to further distribute lines. IDFs also serve as isolation points for troubleshooting. Possible service equipment includes Plain Old Telephone Service (POTS), Voice-over-Internet Protocol (VoIP), and Digital Subscriber Lines (DSL), etc. Fig. 1 depicts an example of the setup within an exchange.

![Diagram of telephone exchange](image)

Fig. 1. Example of telephone exchange terminating outside plant facilities and interconnecting subscribers to termination equipment. Adapted from [1].

B. The Problem with Existing Manual Frames

Maintaining distribution frames is very labour intensive since jumper cross-connects must be completed manually. To this end, technicians are dispatched to the MDF or IDF to physically perform changes. Meticulous scheduling as well as coordination is required between different service-provider groups [1]. Manual operations are particularly prone to human errors such as improper insertions (punch-down) and incorrect cross-connects. The problems can drastically delay the turn-up of new services and can even result in the loss of existing services. This, in turn, may cause customer dissatisfaction and increase the time-to-revenue for the service or result in unnecessary operational expenditures.

Another problematic aspect is line testing [1], [3], [4]. Line testing in itself can be time-consuming as technicians are required to disconnect subscribers/office equipment cables from the MDF or IDFs and connect measurement devices at the ports. Normally a handheld testset is connected to a newly configured circuit to verify the accuracy and correctness of the installation.

C. Available AMDFs

The principle issues that inhibit the widespread adoption of AMDFs are cost, reliability and scalability [1]. Current manufacturers of AMDFs include Lucent Technologies and Simpler Networks [14], Network Automation [15] and Nexans [16]. All these manufacturers employ MEMS-based technologies for the switching matrices. However, these manufacturers only offer fixed-sized modules that limit cost-effective scalability (especially under very strict cost constraints). Furthermore, no significant provision is made for component-level or CO equipment redundancy; the main focus is placed only on the automation of the jumpering process itself. In addition to these limitations, available AMDFs generally make no provision for automated fault detection and correction.

The creation and implementation of a configurable modular AMDF model allows the evaluation of the complete system for various configurations and client activity profiles. The cost effectiveness of the configurations and scenarios can be assessed and easily compared.

D. Requirements for Automation

As mentioned before, the primary goal of an AMDF is to alleviate the manual processes involved with MDFs as well as enhancing the accuracy of copper management in order to reduce cost and increase revenue. The following requirements must generally be met for copper automation [1], [17]:

1) **Any-to-Any Non-Blocking Connectivity**: The AMDF must have the ability to switch any facility wire pair to any equipment pair regardless of any existing connections or port utilisation in the system.

2) **Scalability**: Telephone exchanges vary greatly in size across the world. An AMDF must cost-effectively scale to meet different size requirements.

3) **Reliability and high availability**: Low failure rates of equipment and high service availability are paramount.

4) **Operations Support Systems (OSS) Integration**: The AMDF must incorporate any other service activation, inventory, and workflow systems in order that cross-connects automatically take place as a software-driven flow.

5) **Automated Connection Verification and Continuous Testing**: Along with OSS integration, testing methods...
currently in place at the telecommunication company have to be efficiently incorporated.

III. THE OMNeT++ ENVIRONMENT

OMNeT++ is a C++-based Discrete Event Simulator (DES) for modelling communication networks, multiprocessors and other distributed or parallel systems introduced in 1997 [11] - [13]. The motivation for developing OMNeT++ was to produce a powerful open-source discrete event simulation tool that can be used by academic, educational and research-oriented commercial institutions for the simulation of computer networks and distributed or parallel systems [11]. OMNeT++ was designed from the onset to support large-scale network simulations by constructing hierarchical models built from reusable components.

OMNeT++ provides a user-friendly GUI to aid in the visualisation of communication networks. The user can automatically or manually define the topology of the network and position of each module with the appropriate interconnects for aesthetically pleasing graphical models. The GUI exhibits a feature to animate simulations, which greatly aids in the debugging phase of the model. The user is able to visually track messages being passed between modules. The display characteristics (size, colour, etc.) of modules can dynamically be modified during the simulation run. Data and statistic collection classes provide a means to collect data from simulations, which greatly aids in the evaluation of a network.

Multiple simulation runs on the same network model can be performed to observe the network’s performance for various configurations or conditions. This technique is very useful for parameter optimisation of instantiated modules. Display string parameters can also help to define network topologies. OMNeT++ is not limited to fixed model topologies (such as star, tree, ring, etc.) typical to other simulation packages.

A. Model Structure and NED Descriptions

An OMNeT++ model consists of modules that communicate with message passing [11], [12]. These modules are termed simple modules, which are written in C++ using the simulation class library. Simple modules can be grouped into single entities, called compound modules, in order to create a structured hierarchy in the simulated network. Compound modules may pass or inherit parameters or expressions of parameters to their submodules. The physical layout of the model is described by Network Description (NED) files, which includes topology, interconnects and animations.

B. Module Functionality

Functionality can be added to the entities created by the NED descriptions [11], [12]. As mentioned before, OMNeT++ is a DES, which means that code segments are executed whenever events occur (handled by the simulation kernel). Events are represented by messages, which are passed between simple/compound modules. Whenever a message arrives at a module, an event is issued. Timeouts are scheduled by sending a self-message to the module.

Functionality is added via one of two programming models: coroutine-based and event-processing functions [11]. When using coroutine-based programming, the module code runs in its own non-preemptively scheduled thread, controlled by the simulation kernel. When using an event-processing function, the simulation kernel simply calls the given function of the module object with the message as argument, and the function returns immediately. Coroutine-based functionality has one drawback in that each simple module requires its own CPU stack and may result in a performance penalty. Scalability is therefore limited and memory violations may occur for tens of thousands of modules. Coroutine-based functionality should only be considered if multiple delay and wait statements are required for single events.

C. Parallel Distributed Simulation

OMNeT++ provides the ability to distribute a network model over multiple CPU cores. As cluster configurations are becoming a norm in research computing, OMNeT++ incorporated the functionality of Message Passing Interface (MPI) to distribute the model over a network. The user is required to manually assign modules in the network model to their own Local Process (LP), each running in its own thread. This functionality may significantly speed up the simulation run for a large number of modules if the correct partitioning is employed. If most of the modules are implemented as coroutines and share the same processor core, all but one of the modules will be suspended at any time during the simulation; it is therefore sensible to evenly divide the workload over multiple cores to improve throughput. The ideal partitioning usually has to be determined through trial and error.

IV. THE MODEL

The AMDF model consists of several million modules (including clients, controllers and individual switches) when tens of thousands of clients are simulated. A hierarchical model is used to employ modularity. This allows the user to reflect the logical structure of the actual system in the model structure. Fig. 2 depicts an example for the logical structure in an abstract diagram.

A. The Network

The top-level entity of the model is the overall AMDF. It represents the environment of clients requesting services establishments or discontinuations. The user specifies (via a single configuration file) the total number of clients, number of clients per AMDF module/segment, line delays, equipment lifetimes, CO equipment redundancies for high availability, technician visitation hours, etc., for a single simulation run. The model automatically builds a modular AMDF based on the provided input parameters. For instance, it would be logical to assign the switching matrices (all AMDF modules) to a single LP and evenly distribute the total number of clients.
to the remaining LPs for parallel simulation since the clients account for most of the processing by continuously generating requests.

Fig. 2. Abstract diagram of an AMDF. The Networks Operations Centre (NOC) issues commands to the main controller based on client requests.

Fig. 3. Topology of the network for 15 clients and three AMDF modules (five clients per AMDF module).

B. Switches and switching matrix

The switches are modelled as simple entities that merely pass messages only if it assumes the CLOSED state. For visualisations, the switches appear red for open switches, and green for closed switches. Line splitters and joiners are implemented purely for simulation purposes since multiple connections to a single gate are not permitted in OMNeT++. The actual matrix is implemented as an any-to-any cross-connect matrix. Any input (client) can be connected to any output (equipment). The switch states are controlled by the module controller, which receives instructions from the main controller. Fig. 4 shows the switching matrix of one AMDF module.

A global parameter adds an entire column for the redundant cards. For instance, 20% redundancy for five clients yields one extra redundant equipment card for each service.

C. Main controller

The main controller receives requests from the call centre once a client request is received. The appropriate command is passed to the local controller of the corresponding AMDF module to which the client’s facility wire pair is physically connected. The main controller does not have the ability to control the switches directly.

D. Module controller

The local module controller is in direct control of the switching matrix. Once all the details of the client and his/her request are obtained, the affected switch number is obtained (from a local record of clients and switch numbers) and the appropriate operation is performed on the corresponding switch. This controller also handles automatic fault correction. If one of the CO equipment cards fails, the controller automatically switches to one of the redundant cards (if available) and severs the connection to the faulty card until a technician attends to the problem. Statistics is collected during the simulation run include the total and average equipment failures per time interval, the total average technician visitations and fixing times, the total and average downtime experienced by clients, and the number of times (and total time) the system exceeded its maximum switching capacity for given redundancies.

E. Clients

The clients are implemented as coroutines. Utilising the built-in statistic distribution functions of OMNeT++, a client randomly issues requests to establish new connections or disconnect existing services throughout the entire run of the simulation. The array of clients takes up most of the processing time. The statistical frequencies of client requests are configurable to allow client profiles to be created in order to explore typical scenarios.

F. Alarms

Equipment cards are assigned random lifetimes with the built-in statistic distribution functions. An event is scheduled when the alarm informs a local controller that one of its equipment cards is faulty. The local controller subsequently performs a switch to one of the redundant equipment cards and a technician visitation is scheduled.

G. Technician

Once a call-out is made, the technician arrives at the AMDF, turns off the alarm and starts to replace the faulty equipment card. An event is scheduled when the replacement
of the card is completed. At this stage, the local controller switches the client back to the replaced card with a longer expected lifetime. Technician call-out response times and average repair times are configurable.

V. EVALUATION

In order to evaluate the feasibility of the model, the features programmed into the model adhering to the requirements in section II had to be assessed. It was thought that the major potential problem with the current model would be scalability and parallel DES (PDES). Since coroutines are used to implement some of the modules, scalability inherently becomes a problem on extremely large models. A criterion was developed specifically for OMNeT++ in order to evaluate the potential parallelism of an OMNeT++ model [18].

As a benchmark, an unpartitioned model was run for 3200 clients, 100 clients per module, 20% redundancy in express mode (non-debug mode used for long simulation runs). The inter-partition link delays were varied and the corresponding execution times documented. The events/sec \( (P) \) and events/simsec \( (E) \) values were obtained with \( P = 437,305 \) and \( E = 320 \). Note that \( P \) and \( E \) are properties of the model as is, and cannot be changed during runtime. To increase the effectiveness of the PDES implementation, the link delays between partitions (and LPs) should be maximised, since the null message protocol in MPI uses link delays as lookahead \( (L) \). \( L \) should be extremely large compared to \( 1/E \ (L >> 1/E) \) and MPI end-to-end latency \( (\tau) \). A cluster typically exhibits higher values for \( \tau \), since the actual speed of the switch limits its performance. However, high-speed interconnects used in most clusters exhibit values ranging from 5 µs to 30 µs, typically 20 µs [18]. The number of partitions is denoted by \( n \). The coupling factor, \( \lambda \), is considered:

\[
\lambda = L \cdot E / (\tau \cdot P \cdot n)
\]

It follows from the definition that if \( \lambda < 1 \), frequent blocking is guaranteed and good performance from the simulation cannot be expected. Ideally, \( \lambda \) should be over 10.0. Plotting \( \lambda \) as a function of \( L \) and \( n \) for the specific \( P \) and \( E \) values yielded the graph in Fig. 4. From the figure it is evident that a large number of cores require much larger link delays to maintain values much greater than one.

The effect of null message passing overhead is apparent for larger number of cores. It should therefore be noted that in some instances a lower number of cores can result in a more favourable speedup for specific \( L \) values. If the value of \( L \) is too high, the frequency of the events to be generated between LPs will drop, resulting in a lower value for \( E \). In addition, the simulation model may not be a true reflection of the actual model if link delays are too high.

The simulation was run with a simulation time limit of 100,000 simulation seconds (27.78 hours) and the elapsed time was found to be around 75.9 seconds on a single core for all values of \( L \).

The hardware environment was a Linux-based cluster of 2.66GHz 2x Quad-Core, 8GB RAM PCs, interconnected via Gigabit Ethernet. The tasks performed during the run included various client request, equipment failures at random times, technician visitations, replacement of faulty equipment, and automatic fault correction. The results are tabulated in Table 1 and the corresponding speedups are shown in Fig. 5. Client request intervals and equipment card Mean Time Between Failures (MTBF) were unrealistically shortened in order to stress the model in terms of discrete events per second. It is therefore expected that longer simulation runs (in excess of tens of millions of simulation seconds) will follow the tendency of the graph in Fig. 5 for lower client request frequencies and MTBFs.

<table>
<thead>
<tr>
<th>Link delay (L)</th>
<th>Number of cores (n)</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>100ms</td>
<td></td>
<td>75.9</td>
<td>error</td>
<td>error</td>
<td>30.2</td>
<td>100.7</td>
<td></td>
</tr>
<tr>
<td>5s</td>
<td></td>
<td>75.9</td>
<td>67.2</td>
<td>21.4</td>
<td>12.2</td>
<td>100.7</td>
<td></td>
</tr>
<tr>
<td>10s</td>
<td></td>
<td>75.5</td>
<td>67.2</td>
<td>20.8</td>
<td>10.2</td>
<td>19.3</td>
<td>88.3</td>
</tr>
<tr>
<td>100s</td>
<td></td>
<td>75.6</td>
<td>67.1</td>
<td>19.9</td>
<td>9.6</td>
<td>16.7</td>
<td>10.0</td>
</tr>
<tr>
<td>1000s</td>
<td></td>
<td>75.8</td>
<td>67.1</td>
<td>19.7</td>
<td>9.4</td>
<td>12.6</td>
<td>5.6</td>
</tr>
</tbody>
</table>

Table 1. Simulation execution times for various \( L \) and \( n \) values.

Some configurations yielded errors since the link delays are not seen as sufficiently large according to the Null Message Algorithm (NMA) class, which is used to synchronise the simulation times of the parallel LPs. Note that no significant speedup is observed between one and two cores since all client coroutines are still executed on a single core. Otherwise, a significant overall speedup is observed for \( n > 2 \). This is due to the fact that the client coroutines are evenly divided between cores. It is clear that the presence of MPI communication overhead heavily affects the performance of the PDES model heavily between eight and 16 cores, especially for smaller link delays.
The performance penalty is due to the extra overhead added by the Ethernet switch connecting the PCs and as a result of more frequent inter-module messages and NMA synchronisation messages. Furthermore, since the load is distributed, $E$ and $P$ are significantly reduced for all LPs. The positive effect of using a multi-node cluster becomes apparent for extremely large link delays. Unfortunately, such large link delays are unacceptable for most models (including the current model) to accurately model the actual system. It is therefore concluded that the best possible speedup can be obtained using MPI within a single node in the cluster for the current model.

VI. CONCLUSION

A fully functional model of a modular AMDF was realised. The model allows the user to investigate the effect of the requirements of a modular AMDF such as client division between modules, automatic cross-connect jumpering, automatic fault detection and correction, equipment redundancy, high-availability and downtimes. The model is highly configurable in order to allow the effect of various configurations to be investigated under different scenarios. Parallel simulation is implemented via MPI using the NMA as the synchronisation scheme. The optimal number of CPU cores and inter-partition link delays to maximise speedup were investigated. The speedup obtained with the parallelised model allows various configurations to be evaluated within significantly reduced timeframes. The presented model is the result of the first phase of a larger project. The model will be used to investigate various candidate designs prior to prototype development.

REFERENCES


Marthinus I. Botha received his undergraduate degree in Electronic Engineering in 2009 from the University of Pretoria and is presently studying towards his Masters of Electronic Engineering degree at the same institution. His research interests include robotics, computer systems, embedded systems and real-time and reactive systems.