Using OMNeT++ for Energy Optimization Simulations in Mobile Core Networks

Martin Dräxler
University of Paderborn
Warburger Straße 100
33098 Paderborn, Germany
martin.draexler@upb.de

Frederic Beister
University of Paderborn
Warburger Straße 100
33098 Paderborn, Germany
frederic.beister@upb.de

Stephan Kruska
Ericsson GmbH Eurolab R&D
Ericsson Allee 1
52134 Herzogenrath,
Germany
stephan.kruska@ericsson.com

Jörg Aelken
Ericsson GmbH Eurolab R&D
Ericsson Allee 1
52134 Herzogenrath,
Germany
joerg.aelken@ericsson.com

Holger Karl
University of Paderborn
Warburger Straße 100
33098 Paderborn, Germany
holger.karl@upb.de

ABSTRACT

Optimizing energy consumption in mobile core networks is one of the major challenges towards a greener, less energy consuming mobile network infrastructure. We developed a simulation model for mobile core networks which is able to handle large-scale scenarios including millions of users and the corresponding amount of network infrastructure and mobile core service nodes. It can be used to assess the impact of local and global optimization strategies in routing and power management. Since it encompasses multiple system levels, it is also possible to assess synergies of optimization done at different levels.

To achieve reasonable performance, our OMNeT++-based simulator uses a flow-based traffic model instead of simulating individual packets. The simulator also includes a flexible power model that can describe the power consumption of atomic as well as modular devices. It can be populated with device datasheets and measurements from real-world devices. This allows realistic energy measurements on different levels of granularity, ranging from entire sites down to individual line cards.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communications; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*; I.6.6 [**Simulation and Modeling**]: Simulation Output Analysis

Simutools 2012, March 19-23, Desenzano del Garda, Italy

DOI 10.4108/icst.simutools.2012.247676

General Terms

Design, Experimentation, Performance

Keywords

OMNeT++, Power Model, Energy Consumption, Mobile Core Network, Mobile Access Network

1. INTRODUCTION

One major challenge towards a "greener" more energy efficient operation of communication networks is the energy optimization of mobile access and core networks. Energy efficiency can be optimized in the wireless network as well as on the device level in the wired network. Our approach in the project "ComGreen"[1] is to look at the wired access and core network as a whole and to optimize its energy efficiency by adapting the network and its services dynamically to the current or projected load.

In order to analyze our strategies for improving the energy efficiency, we firstly needed a simulator which can deal with a large simulated network in order to see effects only happening when a large mobile core network is available. Secondly we needed an implementation of a suitable power model which is able to cope with fine-grained changes in energy consumption of the whole network down to each individual physical device. Both of these main features as well as the overall architecture of the simulator will be addressed in this paper.

The simulator has been implemented using the OMNeT++ [23] framework, because of its modularity and the built-in features to dynamically set up a simulation. Together with the INET framework [24] and some extensions, like optical links [9] or LTE signaling [19, 1], it seemed easy to simulate a mobile core network. Unfortunately, all INET-based models work on packet level which did not scale well for the size of our scenario. An implementation with an existing LTE/SAE model in NS2 [18] or NS3 [17] was also discarded for the same reason. Also purely commercial simulator tools like OPNet[2]

[1] `http://www.communicate-green.de`
[2] `http://www.opnet.com`

or QualNet[3] did not promise a better implementation than OMNet++, however we did not have the opportunity to use them for testing.

There are also models for power consumption available as OMNeT++ models, like the one used in MiXiM [10], but they all focus on radio power consumption and batteries in mobile devices. We developed a power model to represent the power consumption of modular devices, e.g. a switch chassis with different line cards. The theoretical background and development process of our power model is described in detail in [2].

The remainder of the paper is organized as follows. In Section 2 we introduce our scenario. Section 3 describes the power model we used in the simulator. The simulator itself is described in Section 4 including its overall architecture and the most important components, namely simulation control, traffic generation, flow routing and power consumption calculation. Furthermore, we analyze the performance of our simulator in Section 5 and conclude our work in Section 6.

2. SCENARIO

One goal of the ComGreen project is to reduce energy consumption in mobile core networks. To evaluate our approaches in a suitably large scenario, we decided to model the German federal state of Hesse in the simulator. Hesse has approximately six million inhabitants on an area of approximately 21100 km^2 [8]. Figure 1 depicts the distribution of population density in Hesse. Based on these figures we identified five different scenario regions: *city center*, *urban*, *suburban*, *settlement* and *rural*. Each of the five scenario regions defines its own base station density, available radio technologies (2G, 3G, 4G, WiFi) and used backhaul and access technologies, for example optical links or radio-relay links.

Based on the distribution of the scenario regions, we can generate a complete scenario network, consisting of four main parts: the base stations, the low RAN aggregation network (LRAN, tree topology), the high RAN (HRAN, ring topology) and the core network (redundant mesh topology). The core network also contains mobile core service nodes for the used radio technologies, for example MME, S-GW and P-GW for LTE [20, 11]. These service nodes and the base stations will act as traffic sources and sinks within the simulation.

We use MATLAB to generate the site distribution and network topology. Our network generator creates random mobile networks including base stations, the aggregation network, and primary and secondary sites as well as aggregation sites with the corresponding network and mobile core equipment. The network generator reads network parameters such as overall area, size of cities, scenario type distribution, cell size per technology and cell types per scenario type (e.g. 4G only in cities) and calculates a complete simulation network. This is done by first laying out the base stations and their technologies and then clustering them into the access and core network. This allows us to create arbitrarily large or small demo networks that conform to our specifications regarding the mentioned parameters without manually designing them. There is no need for real network data, which is especially hard to get. Furthermore, we can evaluate if our optimization strategies work equally well for different networks. Our OMNeT++ code can load the resulting network description and simulate it without further user interaction.

[3] `http://www.scalable-networks.com`

Table 1: Scenario Generation Input

Complete Area	21115 km^2
Overall Urban Area (includes City Center)	32 km^2
Overall Suburban Area	216 km^2
Overall Settlement Area	3014 km^2
Overall Rural Area	17853 km^2
Cell size in Urban Area (includes City Center)	0.2163 km^2
Cell size in Suburban Area	1 km^2
Cell size in Settlement Area	1 km^2
Cell size in Rural Area	10 km^2

The above described methodology enables us to generate and simulate any kind of complex network topology without changing the actual simulator implementation. One example for the flexibility of this approach is this scenario as utilized in the context of the "ComGreen" project.

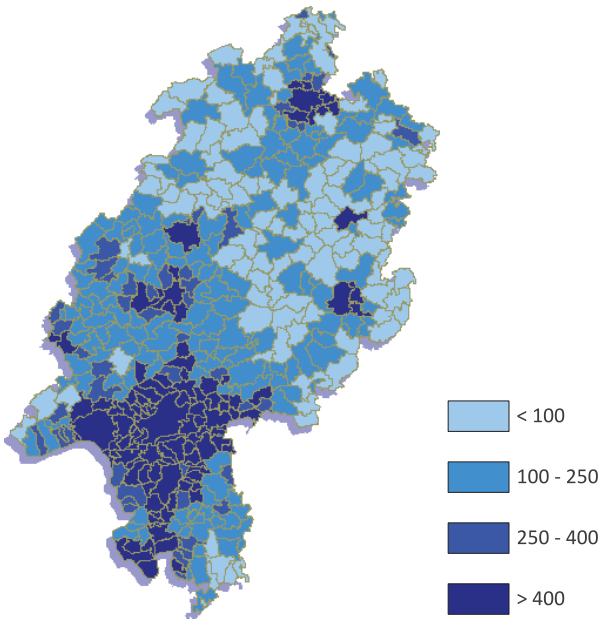


Figure 1: Population Density in Hesse (in inhabitants per km^2) [8]

Figure 2 shows the complete scenario network with three big city areas, co-located with three HRAN rings (blue), and the core network (dark blue). It has been generated by our MATLAB generator by inputting the data of the federal state of Hesse listed in Table 1. This yields a site distribution as listed in Table 2. The scenario regions are positioned according to their geographic coordinates and match the population densities as in Figure 1. We also generated a representative cut-out region including all main parts of the network for faster testing runs which is also listed in Table 2.

3. POWER MODEL

In order to model the power consumption of devices in our simulation we needed a modular power model that can

Table 2: Scenario Figures

	Hesse (complete)	Cut-out region
Cell Sites	~5500	~400
LRAN Access Sites	~1400	~100
HRAN & Core Sites	~75	~5
Total Sites	~6975	~505
Traffic Sources	~8500	~640
Devices	~27000	~2100

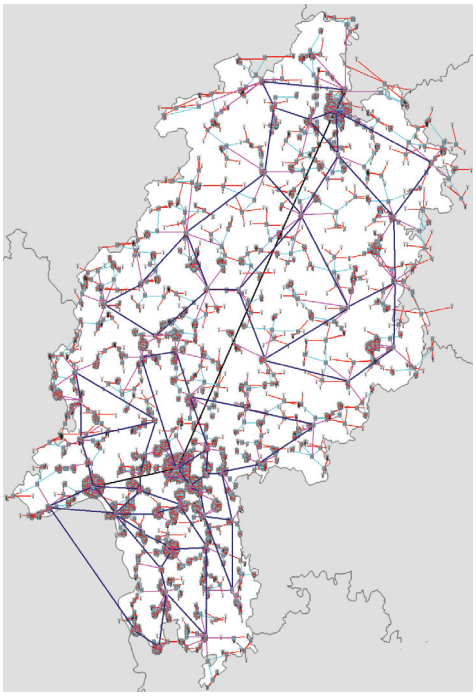


Figure 2: Scenario Network (Hesse)

be integrated into OMNeT++. We first looked at existing power models for network devices but found that most power models were very specific to a certain device or network and were not suitable for our simulator concept. We found many power models that could be characterized as device-level, monolithic and stateless power models that consist of a more or less complex algorithm to compute the power consumption of a device from its utilization/load (e.g. [3]). Among these, we found the model by Mahadevan et al. [12, 13, 14] most promising. It is focused on switches but captures most of the aspects we wanted to model. The actual power model we used is an instance of the generic power model we propose in [2] inspired by the model by Mahadevan et al.

The generic formula to calculate the current power consumption of a device d is

$$P_d = P_{d,stat} + P_{d,dyn}(\text{load})$$

where $P_{d,stat}$ is the static portion of the power consumption (i.e. the power consumption that is present even when a device handles no load and is independent of the load) and $P_{d,dyn}$ is the dynamic portion of the power consumption that depends on the load the device currently has to handle.

Further, we define

$$P_{d,stat} = P_{d,stat,base} + P_{d,stat,conf}(c_d)$$

which splits the static power consumption in actual base power consumption and power consumption that depends on the current device configuration such as e.g. power state or port speed.

The actual values or formulas for $P_{d,dyn}$, $P_{d,stat,base}$ and $P_{d,stat,conf}$ need to be specified for each device type. $P_{d,dyn}$ depends on the load, $P_{d,stat,base}$ is a static power consumption value and $P_{d,stat,conf}$ depends on the device configuration.

An extension to this power model that we implemented in our simulator allows the handling of heat dissipation of devices by the surrounding environment. Each device can be set to emit a certain amount of heat $H_d(\text{load})$ which – just like the power consumption – depends on the current load. Heat is handled on a per-site basis. The power consumption of a site with devices D is thus

$$P_{\text{cooling}}(\sum_{\text{d}\in\text{D}} \text{H}_\text{d}) + \sum_{d\in D} P_d$$

where P_{cooling} describes the heat-dependent power consumption of the given site.

4. SIMULATOR

The simulator is built with OMNeT++ [23], a discrete event simulation environment. Although OMNeT++ originally works message and event-based, we do not use a packet-based simulation like INET, but instead use the OMNeT++ messages to update flow information between the nodes. This mechanism will be explained in more detail in Section 4.1.4 after an overview on the overall architecture of the simulator.

4.1 Overall Architecture

All parts of the simulator can be grouped into one of four categories: simulation control, traffic generation, load and power statistics and devices. A component view of the simulator architecture is depicted in Figure 3. Components of the simulator are drawn as rectangular shapes and storages for simulation input and output data are drawn as cylinder shapes. A connection between two components indicates that these two components will exchange information within the simulation run.

The complete simulation run can be divided into four phases:

1. Initialize Topology & Devices
2. Initialize Traffic Generation
3. Main Simulation Loop
   (a) Generate/Update Traffic Input
   (b) Route Traffic Flows
   (c) Change Device States
   (d) Calculate Runtime Statistics
4. Calculate Simulation Statistics

4.1.1 Simulation Control

The whole simulation run, starting from loading the input data to finally calculating statistics, is controlled by a group of components called *managers*. Their initialization is the

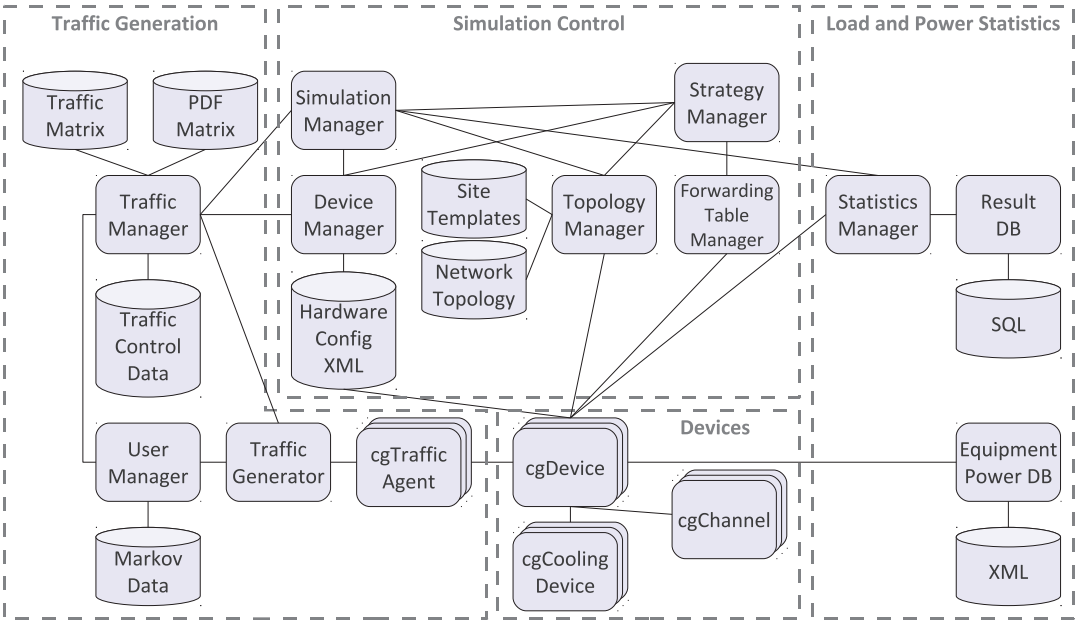


Figure 3: Overall Simulator Architecture

first step in every simulation run. During the simulation the managers serve as central points to retrieve information about parts of the simulation, such as devices, traffic information or gathered statistics.

4.1.2 Topology Manager & Device Manager

First of all the *topology manager* loads the topology information for the desired scenario, which is stored in several CSV files. For each base station there is a 5-tuple `(id, x-position, y-position, scenario region, {technologies})`, for each LRAN aggregation or HRAN site a 6-tuple `(id, x-position, y-position, aggregation level, scenario region, {children})` and for the inter-HRAN links 5-tuples `(id, link type, link technology, source, destination)`. With this information the topology manager is able to create the actual site topology, a tree for the aggregation network and a ring or mesh topology for the HRAN and backbone, and also determine the equipment for each site. For the equipment we use a template system which takes all available information on a site from the CSV files (e.g. scenario region, technologies or level) and maps it to a template id. This template id is then used to load a NED template file, the OMNeT++ built-in language to describe sites and devices. The devices in the template file are then renamed for unique names, parameterized and finally instantiated for each site. The topology manager also instantiates channels between devices which have to be connected. These channels are also parameterized according to the devices they are connected to. Figure 4 shows templates for different sites, for example the template in Figure 4(a) would be instantiated for a base station site with both 2G and 3G as required technologies.

During the topology creation all devices on the sites are also registered with the *device manager* and linked to a hardware configuration. These hardware configurations are stored in XML format and contain the device capabilities (e.g. bandwidth, processing delay or heat dissipation) and the input for the power model calculation. During the simulation the device configuration can be queried at the device manager.

4.1.3 Forwarding Table Manager & Strategy Manager

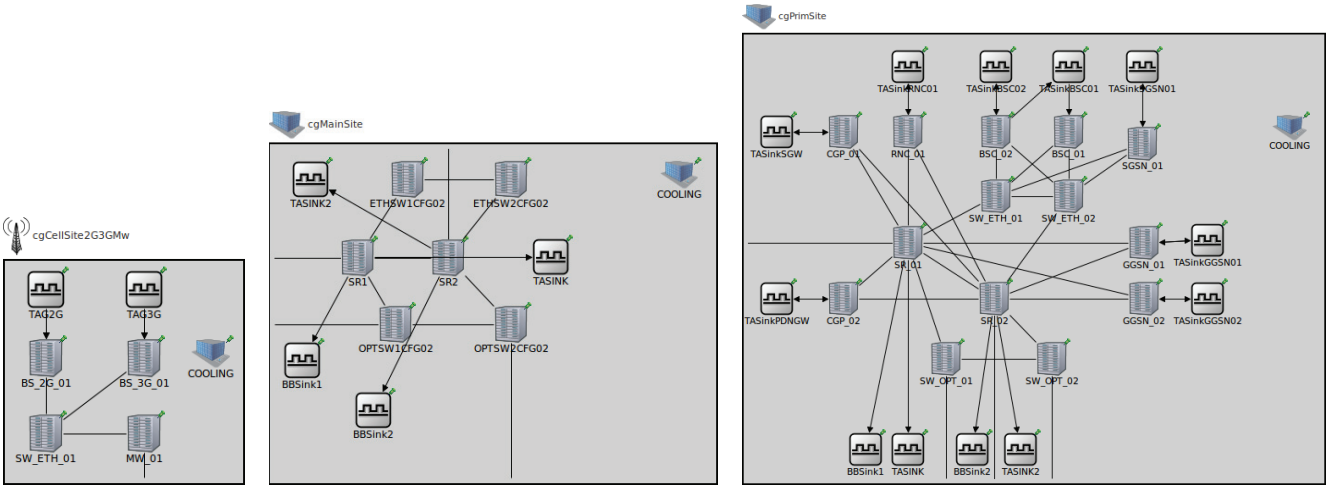
In order to manage routing information more efficiently and achieve better performance we use a centralized entity for routing information, the *forwarding table manager*. This manager can be queried by every device to obtain forwarding information to a given destination device in the network. By default the simulator uses simple shortest path routing using the `cTopology` mechanism built into OMNeT++.

More complex routing policies are implemented by using the *strategy manager*. As we want to use the simulator to test energy efficient routing strategies, the strategy manager is able to obtain power consumption data from the device manager and topology information from the topology manager.

4.1.4 Flow Modeling

As mentioned before, the whole simulation is based on traffic flows, instead of individual packets. One traffic flow is identified as a 4-tuple including the source device, destination device, traffic type and the amount of traffic. Instead of sending multiple messages to simulate packets between the source and the destination, we use a single message to propagate the current state of the traffic flow between the source and the destination. If the parameters of the traffic flow change a new message has to be sent to propagate the updated state of the traffic flow.

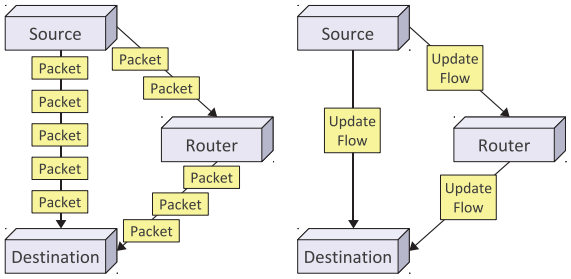
This flow-based approach has two advantages over a traditional OMNeT++ messaging-based approach: First, the amount of messages in the simulation is substantially reduced, leading to less overhead from message handling and improved performance. If you compare both approaches in Figure 5, message handling is invoked and a routing decision is needed



(a) Cell Site (b) Main Site (c) Primary Site

Figure 4: Different Site Types

for every packet in the packet-based approach, while in the flow based approach both is only needed once per flow update. Second, we can easily implement new routing strategies based on the OpenFlow paradigm [15] and easily port them to an OpenFlow testbed for testing on real hardware.



(a) Packet-based (b) Flow-based

Figure 5: Packet- and flow-based message handling

4.1.5 *Devices & Channels*

Devices, or in our implementation `cgDevices`, are the components which represent physical devices in the simulation. Because of the parameterization of devices via the XML hardware configurations, we do not need special device modules for different devices. Instead, we use a generic device module, the `cgDevice`, which can be subclassed for special functionality if necessary nonetheless.

Whenever a device receives a traffic message it queries the *forwarding table manager* for the next hop and determines the gate and channel to route the traffic message there. The device also uses the traffic message to update its own traffic or load state as well as the load state of the channel used for routing the traffic message. More details on traffic messages and their routing will be given in the next section.

Apart from `cgDevices` there are also `cgChannels` to model the connection between devices. The `cgChannels` are also configured according to the devices they are attached to and can add link properties like propagation delay. They can also be used to represent aggregated links (LAG) and are used to facilitate the calculation of the current load and power consumption on the devices. Figure 6 illustrates how the `cgDevices` and `cgChannels` interact.

There is also a module called `cgCoolingDevice` to integrate heat dissipation and cooling into the power model calculation.

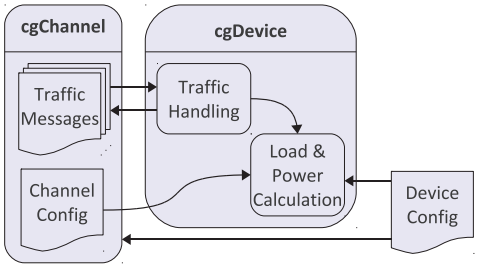


Figure 6: `cgDevice` and `cgChannel`

4.2 Traffic Generation & Routing

The traffic generation during the simulation is controlled by the *traffic manager* which loads its traffic control data at the beginning of the simulation run. The actual traffic data for all traffic flows is then generated by the *traffic generator*. The `cgTrafficAgents` act as the connector between the *traffic generator* and a `cgDevice`. A `cgTrafficAgent` instance can act both as a traffic source by injecting traffic flows into the network as well as a traffic sink by terminating traffic flows. Figure 4(a) displays a cell site with two base stations (2G and 3G) which are both connected to a `cgTrafficAgent` as they inject traffic from the mobile network into the simulation. Figures 4(b) and 4(c) show core network sites where `cgTrafficAgents` are used as traffic sources and sinks, for example as a GGSN or RNC. We can also use the `cgTrafficAgents` to generate background traffic on different levels in the access and core network.

4.2.1 *Traffic Generation Modes*

The simulator offers three different modes to generate traffic which can be selected in the configuration for each simulation run. The main differences between the three

modes are their performance and the included level of statistical correlation between different traffic sources. This means that for every scenario the user has to take into account this trade-off between performance and level of statistical correlation and there is no all-purpose approach.

Trace-based.

Trace-based traffic generation requires a full trace to be loaded for each traffic source in the simulation. The trace contains a vector of traffic values which are loaded sequentially by the traffic source which generates the traffic flows based on each trace entry. We typically use one trace entry per second, which means a trace for one day contains 86.400 entries for each traffic source. For our scenario with 8500 traffic sources this amounts to a total of roughly 3 gigabytes per full trace. Although the trace data has to be read from the hard disk, the trace-based traffic generation does not require any complex calculation, making it comparatively fast. All correlation between traffic sources has to be already included in the trace.

In our setup, the traces are part of the output of an existing wireless simulator, written in MATLAB. The wireless simulator uses simple user behavior and different types of traffic and is not subject of this paper. Other traces can of course be generated from any suitable input or simulator.

Probability density function-based.

The Probability density function-based traffic generation works similar to the trace based traffic generation, but it uses a probability density function (PDF) for each traffic source instead of a trace. The traffic value for each traffic source is generated by the simulator according the the PDF. In order to model larger changes in traffic over time the PDFs can also be changed for different time periods, either with a fixed configuration or with a Markov chain.

This mode obviates large traces but requires a little more calculation for the traffic generation. Although there is no need to store large traces in memory, this mode is unable to model any correlation between different traffic sources, as all PDFs are statistically independent.

User simulation.

Both previous methods for traffic generation interpret traffic sources as aggregation points of traffic from multiple individual users. We typically use one traffic source per base station and the trace or PDF aggregates the traffic from all users connected to this base station. Our simulator is also able to generate traffic on a per-user base and simulate the allocation of users to base stations. In this mode the simulator maintains a list of users, their connection to one or more base stations, and their behavior in terms of traffic values. Each user can be in one state, corresponding to a traffic value. The state transitions are configured as a Markov model. All users also have a position, allowing to use different strategies for base station allocation including connecting to the nearest base station with enough capacity or randomly picking one base station within a certain range.

This mode incorporates correlation between base stations but takes considerably more calculation than the other modes. As a consequence this mode gravely decreases simulation performance and should thus only be used in smaller scenarios or cut-out sections of a large scenario.

4.2.2 *Traffic Routing*

Obviously, traffic cannot just be injected into the network, but it also has to be routed to a specific destination. Accordingly, every traffic flow has a specific destination where it is routed to. In addition to simple destinations, we also integrated a destination label stack. With this stack of destinations, the traffic message can be sent on a path through multiple `cgTrafficAgents` with each `cgTrafficAgent` popping a new destination from the stack until the final destination is reached.

With the help of the destination label stack we can model signaling traffic and user plane traffic. For example, we can pass LTE user plane traffic [20, 11] from an eNB or base station through a S-GW to a P-GW and into a backbone network, as illustrated in Figure 7. The same mechanism also works for background traffic in the core network.

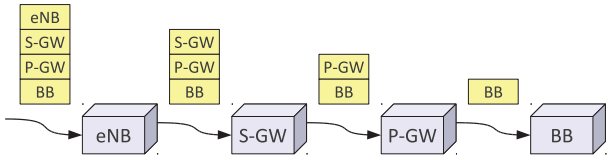


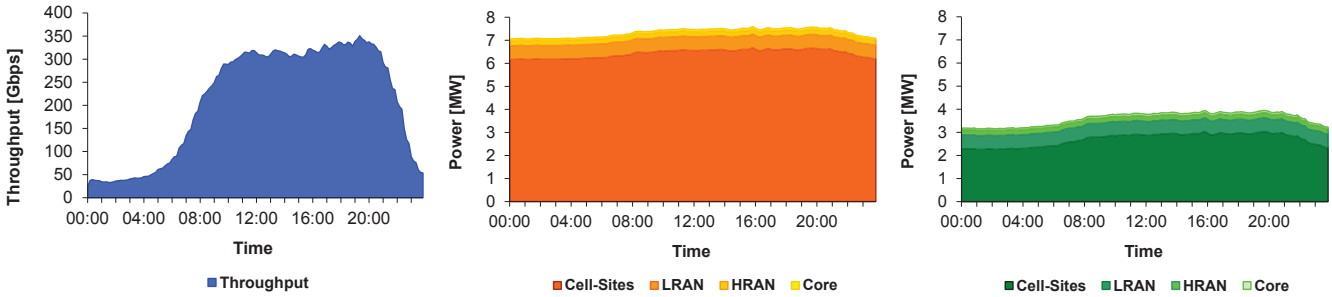
Figure 7: Destination Label Stack

4.3 Load and Power Calculation

In the simulation loop, i.e. while the actual simulation is running and simulation time passes, the simulator calculates runtime statistics for each simulated device and channel. These runtime statistics include but are not limited to current load and current power consumption of each device and current load on the channels. The current load on a device or channel (e.g. in kb/s or number of active sessions in case of core network devices) has been passed through the devices and channels as flow information earlier in the simulation loop, during the routing phase. Furthermore, each device is aware of its incoming and outgoing links. Our power model is used to calculate the power consumption of the device for the current load situation.

Information about each device regarding the parameters of the power model such as modularity, static and dynamic power consumption parameters, as well as possible and current configuration, is taken either directly from the XML datasheets or it is referenced from an external database for power consumption parameters. The data entered comes from own measurements or from the vendor. The datasheets contain all information that is needed to apply the power model calculations to the current load in order to receive the current power consumption value for each device.

If the device is modular, i.e. consists of child devices, the load on the parent device has to be distributed onto the atomic child devices. For aggregated links, a policy has to state which ports of the device are used and how the load is distributed onto the physical links. For (multiple) regular links, a policy has to state which ports are used. The load is then distributed onto the child devices according to the selected policy. As part of possible optimization strategies (e.g. in order to disable as many ports or even line cards as possible) this can be used to assess different methods of selecting ports for linking network elements. It is even possible to simulate other methods to minimize the number



(a) Traffic Throughput (b) Linear Power Characteristics (c) Base-Stations with Power-States

Figure 8: Simulation Results - Throughput and power consumption over 24 hours

of used interface, e.g. dynamically distributing traffic to interfaces.

The concrete steps to calculate the current power consumption from the load on the device are as follows.

1. Read current load on the device on a per-gate basis (e.g. 42 kb/s from gate 1 to gate 5 where gates are the connection points to `cgChannels`)
2. Read device modularity information from the datasheet
3. For modular devices, distribute the load such that the power consumption can be calculated for the atomic child devices (which might even be multiple levels deeper in the modular device hierarchy)
4. For each atomic device with load
   (a) Read device configuration and respective power model parameters from the datasheet
   (b) Calculate the power consumption by combining the power model parameters and the current load
   (c) Calculate the load that is imposed to other devices (e.g. generated heat)
5. For modular devices, sum up the resulting power consumption values to receive one power consumption value for the top-level device

The interesting part here is step 4b. $P_{d,dyn}(\text{load})$ as specified in the power model is used to calculate the power consumption of each device. The simulator allows arbitrary formulas for calculating the power consumption. A typical example is

$$P_{d,dyn}(\text{load}) = \frac{\text{load}}{\text{maxLoad}_\text{d}} \cdot \text{maxPower}_\text{d}$$

which maps a load of zero to a power consumption of zero, a certain maximum load of maxLoad_d to a certain maximum power consumption of maxPower_d and lets the power consumption grow linearly between these two extremes. The constants in this formula have to be included in the datasheet.

It might also be possible that a device can be in different *power states* (e.g. power save, performance and balanced) that influence maxPower_d and maxLoad_d. This means that the simulation has to trigger state changes when a certain device has to handle more load or when a certain optimization strategy demands it.

The cooling of a site is modeled via a special `cgCoolingDevice`. Each site contains one of these special devices. Their base power consumption includes their own cooling and their dynamic power consumption is calculated from the sum of the heat dissipation values of all other devices at the site. The actual algorithm to calculate the power consumption from the heat dissipation is taken from a datasheet.

4.4 Simulation Results

In order to test our simulator we have created and simulated the complete scenario described in Section 2 with a data trace, based on a user and base station distribution developed within the ComGreen project [22]. Two simulation runs with the same data trace were executed, a first one with linear device power characteristics and a second one with base-station devices supporting power states.

To obtain the linear power characteristics, we conducted two measurements for each device, one measurement with zero load (resulting in $P_{d,stat}$) and one measurement with maximum load (maxLoad_d) as taken from the device specifications (resulting in $P_{d,stat} + \text{maxPower}_d$). The power consumption of the device was then modeled as a linear function between these two measurements according to the simple power model mentioned in the previous section. The power characteristics for the base-station devices supporting power states were based on the linear power characteristics with the additional enhancement to switch between four power states with different linear characteristics. The base-station devices were modeled to change their power state dynamically depending on the traffic throughput.

Figure 8 shows the simulated traffic throughput and power consumption of all devices in the network over 24 hours for the different parts of the simulated network. Figure 8(a) depicts the traffic throughput for all network devices for both simulation runs. The change in power consumption during the daytime as seen in Figure 8(b) is visible but not very large. This reflects the current behavior of network devices quite well. Most devices have a very high static power consumption and only a small part of the overall power consumption depends on the load. This is one possible fact to look at for future optimization strategies. Due to the high number of base-station devices in the network, one approach could be to enhance their adaptation to changing traffic load conditions. Figure 8(c) illustrates the lower power consumption of base-station devices supporting power states. Due to the fact that the base-station devices are located on cell-sites, only this fraction shows decreased consumption while the consumptions of the other network regions remain the same.

Table 3: Simulator Performance Statistics

	Hesse (complete)	Cut-out region
events/s	44950 ± 608	65890 ± 484
simsec/s	68.36 ± 0.92	1583 ± 12
peak memory usage	1169 ± 7 MB	237.9 ± 6.6 MB
CPU time	1437 ± 25 s	58.99 ± 0.41 s

4.5 Strategies

With the strategy manager we can integrate different global optimization approaches for the energy consumption. With the help of the manager these strategies can influence the routing of traffic and the dynamic activation and deactivation of devices throughout the network. The approaches include power-aware routing as proposed by Zhang et al. [25] or Heller et al. [7], load balancing among line cards as proposed by Singh and Yiu [21] and energy-aware ISP operation as proposed by Chiaraviglio et al. [4, 5]. Of course the `cgDevice` can also be extended to include local optimization strategies.

5. PERFORMANCE ANALYSIS

To evaluate how well our flow-based approach performs and to determine how much additional calculation our power model needs, we performed a small performance analysis with the simulator. First we ran two configurations, one for the complete Hesse scenario and one for the cut-out region described in Section 2. Traffic was generated by a trace with one value per 10 minute interval for 24 hours, thus injecting 144 different traffic values per base station. No optimization strategies for routing or energy efficiency were enabled. The simulation was run on a Intel Xeon X5650 CPU with 2.67 GHz. Table 3 shows the values and the 99% confidence interval for the most important metrics. With the trace-based traffic mode the simulator is able to run the large scale scenario with good performance, leaving enough room to integrate additional optimization strategies.

The simulation we performed uses only one of the possible simulator configurations. The user can adapt the simulation parameters such as traffic generation mode, size of the network, granularity of traffic values, overall simulated time and others to adjust the running time. For comparison, we ran a user-based simulation with 6 million users and the cut-out region at approximately 0.002 simsec/sec which shows that the user-based simulation is more suitable for smaller networks and shorter timespans.

We also analyzed how much computing time is spent on which parts of the simulator, using *callgrind* from the Valgrind framework [16]. Figure 9 shows the results: Roughly 75% of the time are spent on message and load handling in the `cgDevices` (D) and 25% are spent on the handling of traffic in the `cgTrafficAgents` (TA). The time spent on the other parts is negligible. For both parts most of the time is spent on processing the load messages and updating the load conditions for the power model calculation.

6. CONCLUSION & FUTURE WORK

In order to assess the impact of local and global optimization strategies in routing or power management and the synergies of different optimizations in mobile access and core networks, we needed a suitable simulation model and a reasonably fast simulator. After determining that existing simulation models were unable to handle our large-scale scenarios we decided to create a new one.

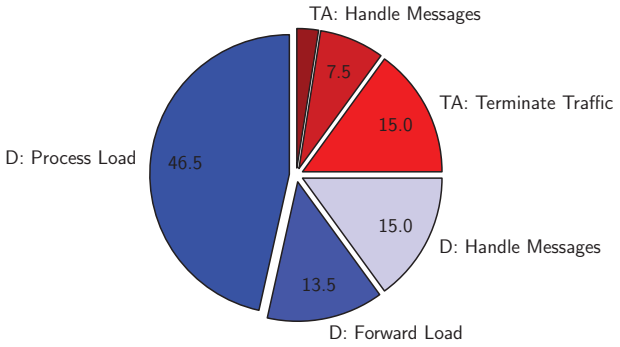


Figure 9: CPU Usage of Simulator Components

We introduced our flow-based simulation model which is able to calculate the energy consumption on a fine-grained basis to determine the energy savings achieved by our optimization approaches.

Currently, the user model does not include user movement, which could be included according to the model from the MOMENTUM project [6]. Also advanced concepts for correlation between users, like social interaction, could be integrated.

Finally, there is also the idea to integrate our simulator with a wireless simulator to form one coherent simulator for energy consumption in mobile networks. Although our simulator architecture is flexible enough for the integration, the overall performance of an integrated simulator remains questionable.

7. ACKNOWLEDGMENTS

The development of the simulator has been funded by the German Federal Ministry of Economics and Technology (BMWi) within the project “ComGreen” as part of the “IT2Green” initiative.

All industry partners participating in the development of the simulator use OMNEST, the commercially licensed version of OMNeT++.

8. REFERENCES

[1] M. Arouri, Z. Atiyyeh, A. Mousa, A. Eleyan, and H. Badr. A Simulation Implementation of the LTE-Uu Interface Datalink Layer in OMNeT++. In *Mobile Networks and Management*, pages 270–284, 2010.

[2] F. Beister, M. Dräxler, J. Aelken, and H. Karl. Power Model Design for ICT Systems – A Generic Approach. Technical Report TR-RI-12-317, University of Paderborn, 2012.

[3] J. C. Cardona Restrepo, C. G. Gruber, and C. Mas Machuca. Energy Profile Aware Routing. In *ICC Workshops 2009. IEEE International Conference on Communications Workshops*, pages 1–5. IEEE, 2009.

[4] L. Chiaraviglio, M. Mellia, and F. Neri. Energy-Aware Backbone Networks: A Case Study. In *IEEE International Conference on Communications Workshops*, pages 1–5. IEEE, 2009.

[5] L. Chiaraviglio, M. Mellia, and F. Neri. Minimizing ISP Network Energy Cost: Formulation and Solutions. *IEEE/ACM Transactions on Networking*, PP(99), 2011.

[6] L. Ferreira et al. MOMENTUM Deliverable D1.4: Final report on traffic estimation and services characterisation. `http://momentum.zib.de`.

[7] B. Heller et al. ElasticTree: Saving energy in data center networks. In *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, NSDI'10, pages 17–17. USENIX Association, 2010.

[8] Hessisches Statistisches Landesamt. Interaktive Regionalkarten Hessen. `http://www.statistik-hessen.de/regionalkarten/applet-de/`, 2011.

[9] K. Kim. Integration of OMNeT++ Hybrid TDM/WDM-PON Models into INET Framework. In *OMNeT++ Workshop*, 2011.

[10] A. Köpke et al. Simulating wireless and mobile networks in OMNeT++ the MiXiM vision. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, Simutools '08, pages 71:1–71:8. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[11] P. Lescuyer and T. Lucidarme. *Evolved Packet System (EPS): The LTE and SAE Evolution of 3G UMTS*. Wiley Publishing, 2008.

[12] P. Mahadevan, S. Banerjee, and P. Sharma. Energy proportionality of an enterprise network. In *Proceedings of the first ACM SIGCOMM workshop on Green networking*, Green Networking '10, pages 53–60. ACM, 2010.

[13] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. A power benchmarking framework for network devices. In *Proceedings of the 8th International IFIP-TC 6 Networking Conference*, NETWORKING '09, pages 795–808. Springer-Verlag, 2009.

[14] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan. Energy Aware Network Operations. In *Proceedings of the 28th IEEE international conference on Computer Communications Workshops*, INFOCOM'09, pages 25–30. IEEE, 2009.

[15] N. McKeown et al. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74, March 2008.

[16] N. Nethercote et al. Valgrind. `http://http://www.valgrind.org/`.

[17] G. Piro, N. Baldo, and M. Miozzo. An LTE module for the ns-3 network simulator. In *WNS3, Workshop on NS-3, SIMUTools*, 2011.

[18] Q. Qiu, J. Chen, Q. Ling-di Ping, and X. Pan. LTE/SAE Model and its Implementation in NS 2. In *5th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 299–303, 2009.

[19] F. Sandu, S. Cserey, and E. Mile-Ciobanu. Simulation of LTE Signaling. *Advances in Electrical and Computer Engineering*, 10(2):108–114, 2010.

[20] S. Sesia, I. Toufik, and M. Baker. *LTE, The UMTS Long Term Evolution: From Theory to Practice*. Wiley Publishing, 2009.

[21] S. Singh and C. Yiu. Putting the cart before the horse: merging traffic for energy conservation. *IEEE Communications Magazine*, 49(6):78–82, 2011.

[22] A. Uzun et al. Communicate Green Deliverable B1.1: Description of the Scenarios and Models. `http://comgreen.snet.tu-berlin.de`, 2011.

[23] A. Varga. OMNeT++ Discrete Event Simulation System. `http://www.omnetpp.org/`.

[24] A. Varga et al. INET Framework. `http://inet.omnetpp.org/`.

[25] M. Zhang, C. Yi, B. Liu, and B. Zhang. GreenTE: Power-aware traffic engineering. In *18th IEEE International Conference on Network Protocols (ICNP)*, pages 21–30. IEEE, 2010.

APPENDIX

A. ACRONYMS

2G second-generation wireless telephone technology (e.g. GSM)

3G third-generation mobile telecommunications standards (e.g. UMTS)

4G fourth generation of cellular wireless standards (e.g. LTE)

eNB eNodeB, name of the base station in LTE

GSM Global System for Mobile Communications, a mobile communications standard

GGSN Gateway GPRS Support Node, a core network node in GPRS

GPRS General Packet Radio Service, the packet-based part of 2G and 3G networks

HRAN High Radio Access Network, the core network part between LRAN and the sites of the core network equipment, often mesh or ring topology

LAG Link Aggregation, combining multiple physical links to form one logical link

LRAN Low Radio Access Network, the often tree-shaped core network part between the base stations and the meshed HRAN

LTE Long Term Evolution, a mobile communications standard

MME Mobility Management Entity, core network node in the SAE

PDF Probability Density Function

P-GW Packet Data Network Gateway, a core network node in SAE

RAN Radio Access Network

RNC Radio Network Controller

SAE Service Architecture Evolution, core network architecture of LTE

SGSN Serving GPRS Support Node, a core network node in GPRS

S-GW Serving Gateway, a core network node in SAE

UMTS Universal Mobile Telecommunications System. a mobile communications standard

WiFi a mobile communications standard, including IEEE 802.11