Analysing Efficiency of New Load Distribution Algorithm with Horizontal Scaling

Loránd Nagy, Attila Hilt, László Jánosi, and Gábor Járó

***Abstract*—As the distributed software and microservice architecture is spreading, the importance of load distribution is increasing. There are multiple indicators to measure the quality of the distribution, in the past the main interest was load balancing, i.e. evenly disrtibuting the load among available units. In this paper another aspect is investigated, how well can the total load be distributed among the least number of units, keeping the others in a gracefully shut down state. For this purpose an unevenly distributing algorithm is presented and compared to the model of traditional load distribution. The algorithm is designed to support horizontal scaling of processing units via attempting to operate most of the active units at a design load level, allowing only one at the time to take less traffic. This paper focuses on the specific case of load distribution inside telecommunication core networks.**

***Index Terms*—Load distribution, load balancing, distributed system, scaling, graceful shutdown, Round-Robin, horizontal scaling, simulation, telecommunication.**

I. Introduction

In traditional telecommunication networks, the design capacity is permanently available regardless of the actual traffic. This means that the network element (NE) constantly requires all the assigned hardware (HW) which consumes energy. Furthermore, in most cases these – either physical or virtualized – components can not be reassigned for tasks other than their designated ones. In telecommunication network elements, Round-Robin algorithm can be used to model some of the load distribution processes, especially that it is – and its modified versions are as well – a rather popular choice for load distribution and load balancing. However, such methods are facing serious challenges in solutions that are moving towards greater flexibility, i.e. scalability of the system. The main issue of these methods is that the distribution of the load is attempted by distributing the requests. As the actual load (used capacity) of a functional unit (FU) is generated by the handled requests this could be a viable solution, but the indeterministic nature of requests (e.g. varying call lengths) may weaken the correlation between them. Therefore the load distributing algorithms, focusing more on distributing requests, are trying to act as load balancers and are rather inflexible. As an example, the Round-Robin algorithm assumes that every FU has the same capacity and performance in the system and it is not capable of decreasing the target performance of selected units. These are the reasons why we have to find a solution that addresses these problems and is able to support the later-described horizontal scaling.

Loránd Nagy (e-mail: lorand.1.nagy@nokia.com), Attila Hilt (e-mail: attila.hilt@nokia.com), László Jánosi (e-mail: laszlo.janosi@nokia.com), and Gábor Járó (e-mail: gabor.jaro@nokia.com) are with Nokia Networks, H-1092 Budapest, Köztelek u. 6. Hungary.

Manuscript received September 12, 2016.
Paper accepted October 2, 2016.

However, it is important to notice that Round-Robin, Weighted Fair Queueing and their variants have been used (for load balancing) for a long time. Naturally many results have already been achieved in dynamical load balancing [1], [2]. Furthermore, e.g. Session Initiation Protocol (SIP) based services are also very well-investigated [3]. Therefore in general it can be stated that scalability of distributed systems and load distribution that supports it are interests of the research almost as far as the first appearrance of Cloud.

Development in 'Telco' network solutions enables almost all core network elements e.g. TAS, MSS, and MME (see Fig. 1.) to run in the Cloud [4]. In virtualized environment, scalability of Virtual Network Functions (VNFs) becomes a very important requirement. There are two basic methods for scaling; horizontal and vertical. The first method adapts the number of functional units to the total load amount in a distributed system, while the latter allocates resources dynamically to a unit – or to a number of units. Throughout this paper the horizontal scaling is used. Practically it allows to add or remove resources (henceforth functional units) anytime to or from the system with minimal effect on the operation. To ensure the quality of handling requests, removal of a functional unit shall be graceful in the sense that it cannot cause performance degradation and ongoing processes must not be interrupted.

For better efficiency, an algorithm is needed which uses only certain (higher order) functional units continuously. The main idea of the new algorithm is that it starts loading additional (lower order) FUs if the load of the higher order units reaches a pre-defined level (upper threshold). The lower order FUs shall receive only a smaller portion of the load to ensure that the higher order units are loaded approximately to their design load level. Another goal of this logic is to decrease the amount of requests handled by lower order units to decrease the expected time to reach gracefully shut down (GRSD) state after deactivation (from receiving requests) as a result of decreasing load. Thereby, in Cloud-based environment, traffic could be concentrated on a few functional units and further FUs are loaded only if it is necessary.

Our goal is to provide a method which loads the available functional units within a single network element (either TAS or MSS) according to the aforementioned logic. This way, the operation of the network element can be more energy- and cost-efficient since unused FUs could be shut down automatically depending on the actual traffic.

It is ought to be highlighted that the algorithm described in this paper is applicable in any distributed telecommunication system, hence it is independent from any protocol (i.e. SIP, Megaco, SS7). Moreover, the unknown length of the running processes, which makes the seamless removal of functional units harder, is not a problem for this algorithm. Typically, the length of calls has an unknown probability distribution, hence reaching the gracefully shut down state can be delayed indefinitely. However in call handling it is crucial that even very long calls must not be terminated by removal of that FU which controls them.[1]

Henceforth, in Section II. we describe the problem in detail and show some related examples from real mobile networks. In Section III. we interprete our solution in both theoretical and practical point of view. In Section IV. we compare Round-Robin and the new algorithm and in Section V. we summarize the results so far.

II. Problems and Real-life Examples

Before we describe the problem in details, we take a closer look at the core network, which is in the scope of our investigation and introduce how it can be moved onto the Cloud.

A. The Core Network

The Core Network (Figure 1) consists of four main domains:

1) Packet Core;
2) Services Domain which includes e.g. MSS (Mobile Switching Center Server), TAS (Telecommunication Application Server) and the IMS (IP Multimedia Subsystem);
3) Registers (HLR or HSS) that are common for the Packet Core and the Services Domain;
4) OSS (Operations Support System) which manages all of the network elements, including the radio network (2G, 3G, 4G/LTE) as well.

The same resources (CPU, memory, storage, IP interfaces and networking) must be provided for (core) network elements in order to be able to operate in Cloud based environment (just as in traditional, e.g. ATCA based HW environment) [5].

On Cloud, storage capacity is provided through SAN (Storage Area Network) and network connections are provided by SDN (Software-Defined Networking). As it can be seen in Fig. 2, telecommunication applications (which are SW applications) are located on the top of a cloud infrastructure. Similarly, OSS and CAM (Cloud Application Management, which controls the horizontal scalability of certain network elements) can run on Cloud.

Based on ETSI NFV architecture [5], MSS and TAS are virtualized network functions (VNFs) [6], [7]. The function of MSS and TAS in mobile and IMS network is defined in detail by the following 3GPP technical standars: TS 23.002 [8], TS23.218 [9] and TS 23.228 [10].

[1]In real networks there are limitations for the maximal call duration (typically 60 or 120 minutes), but that is enforced independently of the management of functional units.

B. Load from a real network

We collected data from several mobile network operators[2]. First of all, it can be observed that the load of the functional units fluctuates – following a recognizable pattern – significantly on both daily and weekly basis. It can be seen in Fig. 3. how rapidly the load changes between night hours and (daytime) busy hours. It is also clearly shown in the same figure that – compared to weekdays – weekend traffic is reduced (red circle). Moreover, sudden big load can occur anytime, even during busy hours (green circle).

Of course, traffic also fluctuates at the level of the entire network (Fig. 4.), which foresees the possibility of switching functional units off. For example – if we take a closer look at daytime and nighttime traffic in the network – fluctuations can be measured up to 60%. This means less functional units would be needed in nighttime compared to the busy hours. Currently, all the resources are reserved continuously, while for example charging data processing is done mostly at night, which requires additional resources.

These analyses show that the utilization of the resources does not often reach even the 35% level, which is far from the recommended 60 to 70% level. Therefore, one of the goals is more efficient resource utilization because it leads to an optimized resource management and helps to decrease the energy consumption of the system.

C. Problems with Round-Robin

As it was mentioned earlier, Round-Robin (scheduling) is used widely for call-handling (or call distribution) in telecommunication networks. Consider the following simple example to this algorithm. If the functional units are represented by urns and balls represent the calls, the algorithm works as follows: we throw the balls one-by-one into the urns consequently in arrival time and if we run out of empty urns, then we start over from the first urn. This process is rather simple and easy to implement, however not appropriate for several reasons. First of all, it cannot be used for priority-based or biased load balancing. It can be seen in Fig. 5. that the average load of the functional units are not really balanced. This can be explained with the fact mentioned in the introduction: Round-Robin does not take into account the actual load of the (internal) functional units and assumes that every FU has the same capacity and performance in the system. The second problem is its deterministic behaivour. It is shown in Fig. 6. that the (control plane) CPU usage of a single call is not constant: there are setup and end phases. Based on the elapsed time between the two phases – i.e. the length of the call: T, where the distribution of T is unknown – we can talk about short and long calls. Since we do not know the real distribution of the call lengths, we cannot predict the end of a call. This causes the problem that there are FUs with higher load levels as a result of handling many short calls, while the others are less loaded, since they get more long calls.

Assume we have twenty urns and blue balls represent the long calls. Moreover assume as well, that every twentieth call

[2]We do not name specific operators for confidentiality reasons, so from now on, we indicate every operator as unknown.

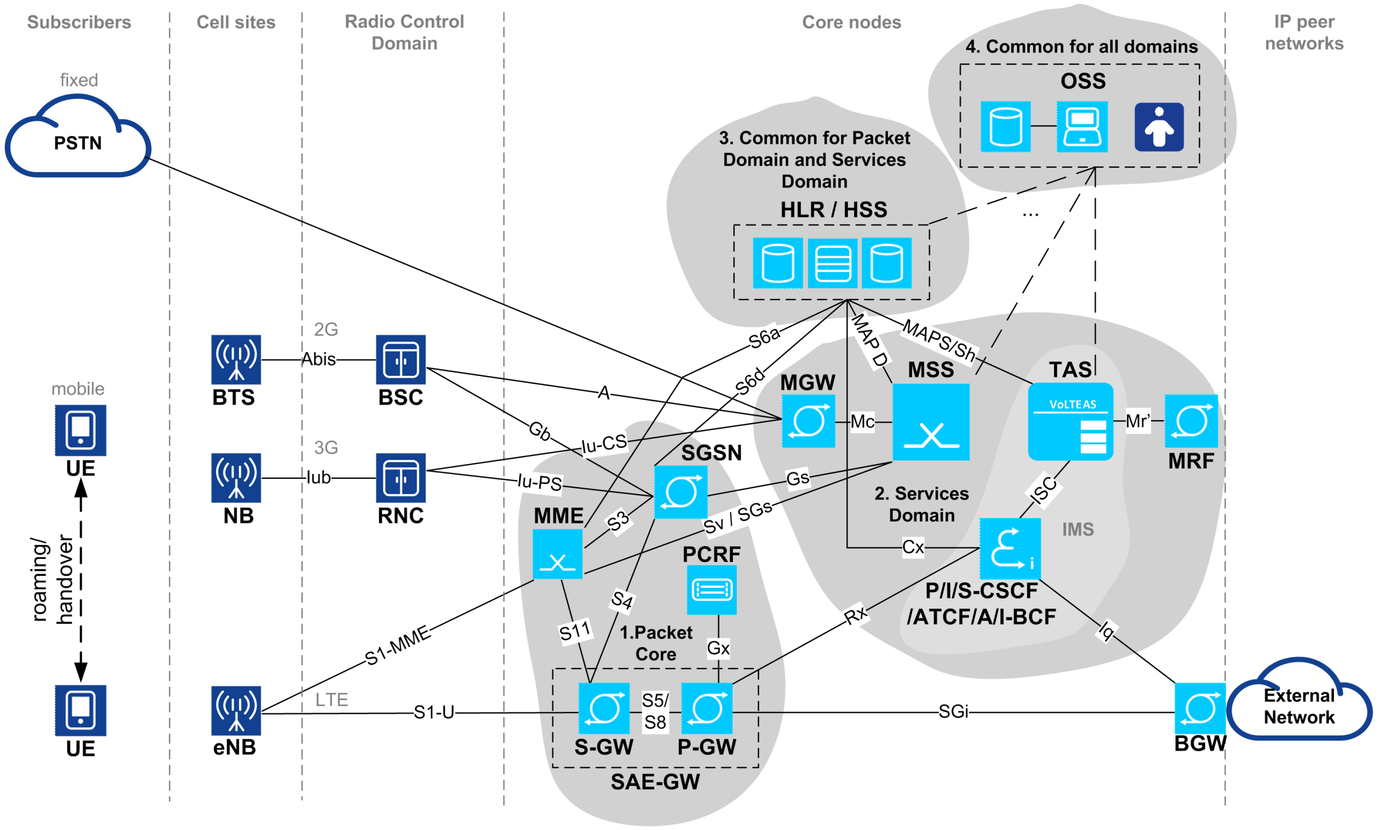


Fig. 1. Topology of the core network

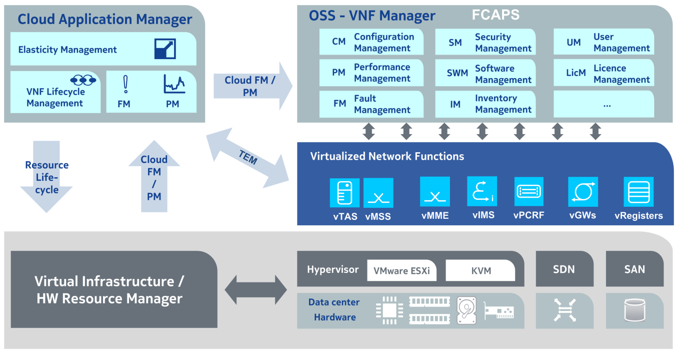


Fig. 2. Network elements of the core network in Cloud based environment [7]

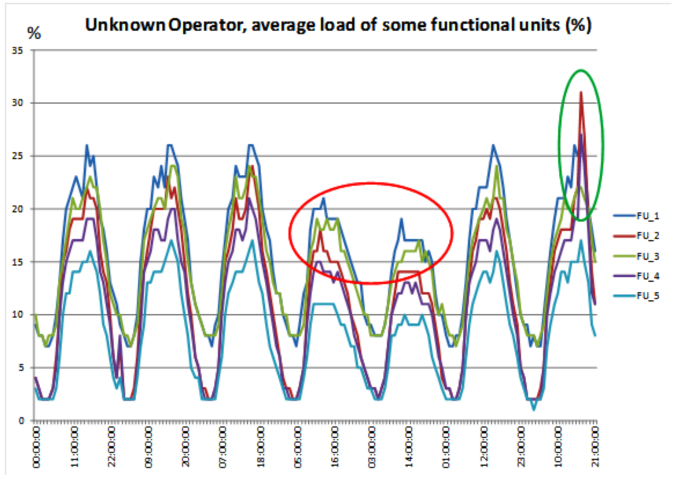


Fig. 3. Average load of FUs of a network element
one week measurement, and one hour resolution

is a long call. In this scenario the twentieth urn receives all the blue balls (i.e. long calls).

Considering these observations, the analyses and the properties of Round-Robin, this kind of call-handling is simple, but far from being optimal. This is the motivation for the search of a better load distribution algorithm. The problem requires a more sophisticated algorithm that is designed to address the particular concerns raised against Round-Robin.

III. New algorithm for Biased Load Balancing with Graceful Shutdown

In our solution, one of the most important targets is to use only a number of functional units from the available ones, proportionally to the actual load. Practically we do not endeavour to use all of the functional units at the same time nor to load them evenly. Another concept is to add or remove FUs to or from the system based on the current amount and the trend of the traffic. As a consequence of this concept, the load distribution among units will not be uniform, this is the reason for which – instead of load balancing – the terms load distribution or biased load balancing are used.

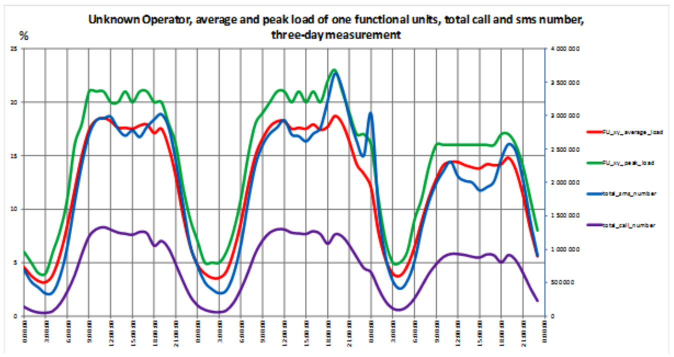


Fig. 4. Average- [red] and peak-load [green] of a single FU, compared to the total number of voice calls [purple] and short messages (SMS) [blue], three days measurement

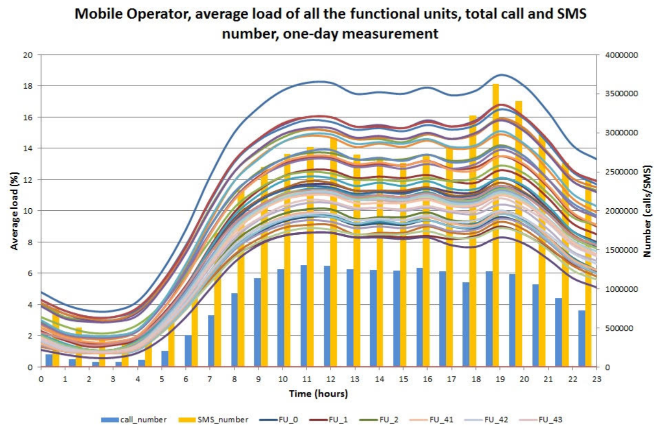


Fig. 5. The effect of calls [blue columns] and messages [yellow columns] to the load of FUs

A. Advantages of Load Distributors

Application of load distributors (LDs) is inevitable in most distributed systems. With their help, incoming requests can be routed to the internal functional units of the network element. Furthermore, the inner architecture, complexity and operation of the system can be hidden by load distributors, i.e. the outside world can communicate with the inner functional units only through these load distributors.

Load distributors can be as simple as a proxy, although in many cases there are more functional requirements. A common example is when the load distributors shall act so that the load distribution among the inner functional units is uniform. In such cases a distributor is called Load Balancer (LB). In telecommunication network elements even the load balancing can be a distributed task, i.e. the external communication towards the inner functional units is routed through several LDs (Fig. 7).

B. Priority-based Horizontal Scaling Algorithm

From now on, we always assume that the system contains at least one LB, moreover later, in the simulations, standalone load distributor (LD-BA) is used. In real network elements – due to redundancy and resiliency reasons – there are minimum 2 LBs. The number of load balancers in fact depends on the internal network planning of the NE (in IP based systems) and the required protocols as well.[3]

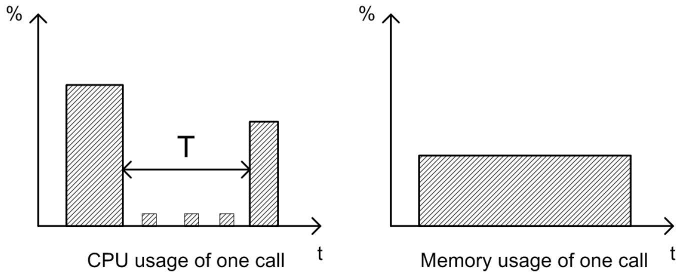


Fig. 6. CPU and memory usage of a call

To resolve the detailed problems our proposed solution is a load distribution algorithm called Horizontal Scaling Algorithm (HSA). HSA is based on the concepts described in the patent [14] submitted by Nokia. This algorithm is designed to be implemented in LBs. The key concepts of HSA are as follows:

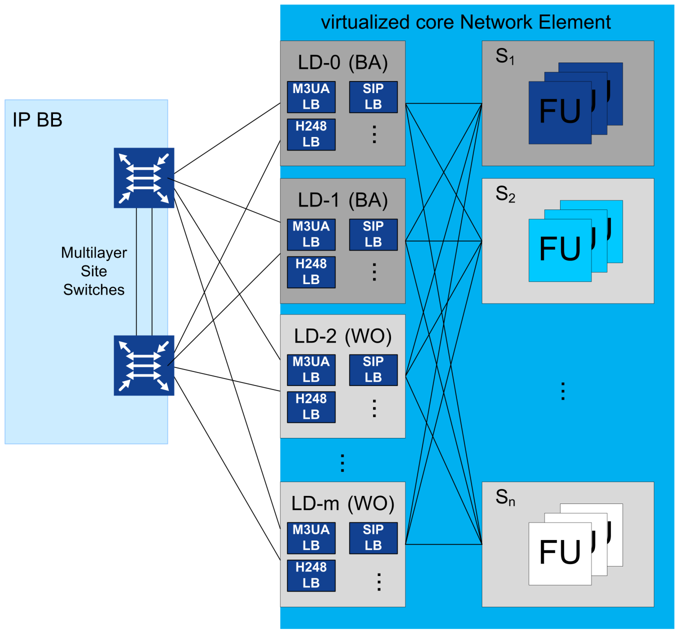


Fig. 7. Planned network element configuration, where FUs are in sets

a) First of all, we organize FUs into groups or sets (S_i, $i = 1, \dots, n$). Such groups can contain any number of FUs (at least one, at most as much as totally available). On one hand, we can minimize the chance – by using groups – that a newly activated FU is overloaded immediately after being switched on. On the other hand, we can manage more CPU resource at once. Also this is a model for a scaling solution used in real networks. This kind of configuration can be seen in Fig. 7.
b) Secondly, we fix an upper load threshold (e.g. 65%) for the average target load of the groups (also referred to as sets). The aim of this is that we route requests to the

[3]Different protocols usually require separate load distributor units, e.g. SIP and Diameter.

active groups as long as the average load of them reaches the given upper threshold. If the target load is exceeded, a new group is placed into operation (Fig. 8.). Hereby we force the system to activate a new set if and only if it is needed based on the load caused by the current traffic. Similarly, we fix a lower threshold (e.g. 10%) for the average load of the FUs within each group. This is used when the average load of the group, that was activated as last, is under the threshold we can deactivate that group.

c) In addition, each group gets a preference number for which the following are true:
   - each FU has the same preference number within its group,
   - if a FU is switched off (or in idle mode), its preference number is zero,
   - if the load of a group reaches the upper threshold, its preference number is maximum (M),
   - if a new group is loaded, its preference number is dynamically changed between 0 and M (depending on traffic).

   We use these to make resource handling and load distribution between groups adaptive in case of traffic changes.

d) Finally, we randomize the call distribution. The process can be seen in Fig. 9. The timeline shows the incoming calls. The arrival of calls can be modeled with a Poisson process [11] (although some results yield that in fact this is not the most accurate model [12]). Such subdivision of an interval $[0, I_{\max}]$ can be seen in the lower part of Fig. 9. in which each subinterval belongs to a different FU. The colour of the FUs indicates the group that they belong to. The lengths of the subintervals are determined in real time by the following formula:

$$L_{\mathrm{FU}_k} = P_{\mathrm{FU}_k} \times F_{\mathrm{FU}_k},$$

   where L_{FU_k} is the length of subinterval that belongs to the FU with index k, P_{FU_k} is the preference number and F_{FU_k} is the free capacity of the of FU_k. The bigger the preference number of a FU is, the longer the corresponding subinterval is. This is only altered by including the free capacity of the FU in the formula. The reason for that is to decrease the chance of assigining load to FUs that are able to handle less requests among units with the same priority number. The call distribution is random, because every call gets a random number from $[0, I_{\max}]$. The call is sent to that FU, which belongs to the subinterval that contains the generated random number. If we recall the urn model and the problems that appeared during the Round-Robin it seems to be a logical choice to randomize call distribution.

C. HSA in practice

As this research was motivated by telecommunication business needs, the first approach on HSA was to present a proof-of-concept simulation. Therefore we created a computer program to simulate calls and units to process them. For the computer simulation we created a simple c++ console

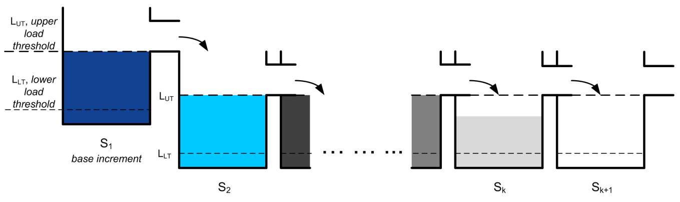
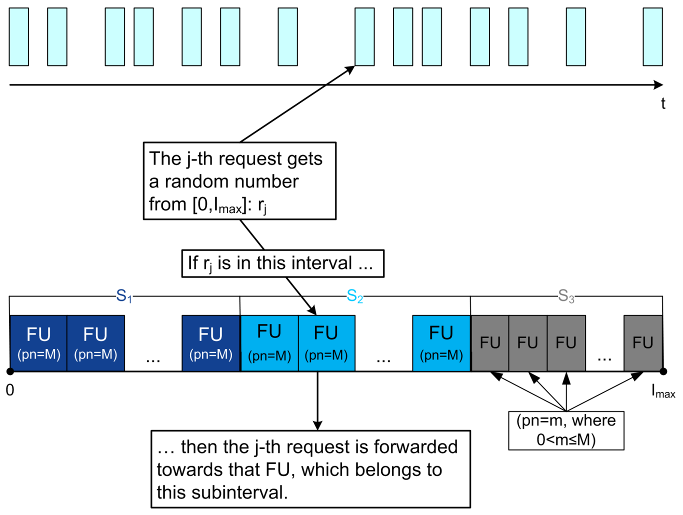


Fig. 8. Bucket chain model



t
The j-th request gets a random number from [0,Imax]: rj
If rj is in this interval ...
S1
S2
S3
FU (pn=M)
FU
0
Imax
(pn=m, where 0<m≤M)
... then the j-th request is forwarded towards that FU, which belongs to this subinterval.

Fig. 9. Illustration of the randomization concept for call distribution

application. The program used a simple representation of every element as a custom class:

call: an integer for ID, and double precision floating point numbers (doubles) for start time, end time, and call length (in seconds)

functional unit: an integer for ID and a vector of calls, with methods to measure load (the program setup defined the maximal number of half-calls[4] for a unit to represent load as percentage) at given time and accept calls (or reject in case of overload)

group of FUs: boolean flags to indicate status of activation, graceful shutdown, an array of FU types and an integer storing the last used FU index (for Round-Robin simulation) with additional doubles to store relevant history data (past load, time of activation), methods to process arriving calls for both Round-Robin and HSA, to measure load and free capacity

With these elements the simulation program used a file input to simulate call data, then used that data to simulate the response of the system, first using the Round-Robin algorithm then the HSA. The Round-Robin was started with all groups activated, however the HSA just used one group at the start. Both methods stopped at every second to evaluate status. While the Round-Robin algorithm just removed unused data and wrote active call number and average load for FUs in each group, the HSA evaluated the actual status and intervened if it was

[4]Half-call: In network elements, the signalling tasks related to a call are done separately for the originating and terminating sides – i.e. the caller and called parties – hence the signalling indicated here is related only to a half call.

required. The logic of the decision is described in [14]. The reason for simulating calls instead of using measurement data is:

- key use-cases in proving the usability of the method can be created more easily,
- exact measurements containing call start and call end times with precision to the second are not available and even if such data existed, the simulation would need to change the specifications of the simulated processing units to create all needed scenarios.

However a goal is to create simulations with real life measurement data to compare the simulated results to the measured values. To do this we needed to create a model for calls that can be defined through total call numbers during predefined time periods.

It is a commonly accepted model for independently arriving requests (like customers arriving into a shop, or calls started through a network etc.) to use Poisson distribution for the number of total requests [11]. The Poisson distribution has a great advantage, namely that we know exactly the distribution of the time between two events (arrivals or call starts) which is the exponential distribution. Therefore if we have the number of calls arriving in a period we can create a model that provides us with a total call number for that period as a random variable that follows Poisson distribution and its expected value is the given number of calls. More generally we can say that the total number of calls follows an inhomogeneous Poisson process and we are using a rate function that is piecewise constant. If we accept the Poisson process we only need to create a function in the simulation that gives us random numbers with exponential distribution with the given parameter. Generating genuine random numbers is a popular topic among programmers and mathematicians as well. There are many algorithms to create numbers that follow certain distributions, but in fact most methods can not generate real randomness, that is usually achieved by seeding the random number generating algorithms with some random measurement data (with known distribution of course). The random numbers our simulation used were generated with functions provided by the <random> extension library of c++. These functions also needed some seeding which was provided as the sum of three numbers. The first was obtained from the system clock, as it is a common practice for such purpose. The second was the value of the time variable of the simulation. Finally, a number was generated by another built-in random number generator that did not require seeding. This allowed us to create pseudo-random numbers with the needed distributions, since the the chance of repeated seed values was neglectable.

Pseudo-randomness means in our case that the probability density functions (PDF) of the generated numbers do not match exactly the PDFs of the desired distributions. For example the uniform random number is not uniform, functions like the one the simulation uses has a small skewness, meaning that it favors one side of the interval. This could indeed cause a problem, but the effect of it can be dramatically reduced by using a small interval, where the skewness is virtually undetectable. For other distributions we can generate lots of numbers and then examine the difference between theoretical and empirical moments. In our simulations all generated distributions had error for the first two moments under 0.05%.

Our modeling of calls now has a method to create call starts at a defined rate (or following a rate function). To complete it we also need a model to terminate these calls. The first idea was to use exponential distribution as it was already implemented. In fact, for the first simulations we used this method. The reason why we accepted a less accurate model, is that our simulation focuses on the load created by the call setups, termination has much smaller effect and we did not consider ongoing calls as relevant load inducing parts (Fig. 6.).

However, the exponential distribution generates short calls with high probability. Our goal was to use a model where even long calls can occur, in order to see if deactivated sets can reach the graceful shutdown state. Hence we used a simple model, where call length was defined as a log-normally distributed random variable. We chose this specific model although there is no intuitive explanation for the results in [12]. During the simulations, we used log-normal distribution with expected value: 90 (sec) and standard deviation: 50 (sec). With these decisions for modeling we could start the implementation of the simulation. This paper in fact describes the second phase of the simulation where phase one was the first implementation of the logic and concentrated all efforts on creating a code that can show the behaviour of the logic in basic use-cases. For this first simulation we used the exponential call length model and data tailored for specific situations and all parameters were set to fulfill the needs of the scenario. The first simulations provided results proving that further investigation is needed. Even though the first results proved that the proposed new algorithm is viable, it did not provide any comparison between the old and the proposed new logic. Therefore the main goal was to create a better model that can simulate calls and provide this data set for the call distribution algorithm which should be able to run both based on the Round-Robin method and the HSA. This settles the basic requirements for the program, these three features were needed.

IV. Comparison of Round-Robin and the new HSA algorithms

During the comparison we used data based on measurements from real operators. Initial testing ran with arbitrary data that we created. The real results came from two methods, where the first was to take data that represents some sort of periodicity and investigate the load results of that with 15 minutes sampling for the rate of the Poisson process. The second approach was to get data from operator measurements and create a new set of data with a better – one minute – resolution.

We used a standard spline[5] interpolation for the original points and evaluated the function at each minute. This method results in data that holds call numbers measured on greater time intervals than its time resolution. To get data consistent

[5]Approximation for the joining curve with qubic polynomials on each interval with continuous first and second derivatives between intervals.

with the original, the result has to be divided by the ratio of the lengths of the original and the finer intervals.

In the following sections three scenarios will be presented for both LD algorithms. In the first two there were five sets, each containing six functional units, in the last scenario only three sets were used but still with six functional units in each. The main difference between these sonfigurations was that in the latter the functional unit capacity was increased and the traffic for a whole day was simulated.

A. *Results for Round Robin based decision*

In the first scenario the simulation used data where the incoming traffic was generated by an inhomogeneous Poisson process, where the rate function was constant over 900 seconds long periods. The results are shown in Fig. 10. The purpose of this setting was to present the behaviour of the algorithms when the load is changing significantly. It can be observed, that the load generated by the calls was equally distributed among groups of FUs (sets). This result shows how this concept is working if there is just one loop of Round-Robin. In real realizations there are concurring loops, causing the load differences on the presented measurements. It is important to note that the average load of the groups indicates that there is constantly significant free capacity for each group.

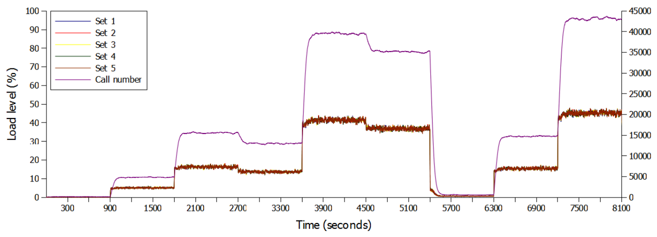


Fig. 10. Simulation result of Round-Robin (basic use-cases)

The second scenario used a more continuous rate function. The aim of this scenario was to present how the load is distributed among FU groups. These results confirm the original assessment of the problem, that the groups of functional units are basically loaded equally. Also, as it can be seen in Fig. 11. the efficiency of this setup is not ideal.

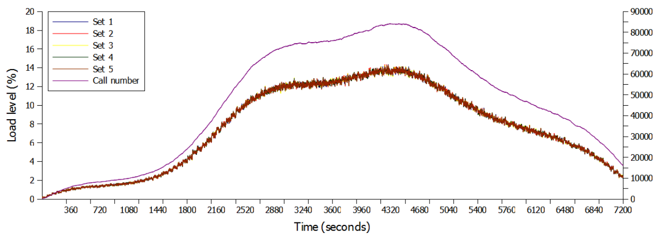


Fig. 11. Simulation result of Round-Robin (simple traffic profile)

B. *Results for HSA*

In this case, we are able to see that the algorithm works as planned (see Fig. 12.). In every situation, there are three states a set can assume:

(i) inactive,
(ii) activated and working with load under the design level,
(iii) activated and working on the design level.

The second – less desired – state is always unique in the sense that at any time only one group of FUs can assume that state, the others are either inactive or working at the design limit.

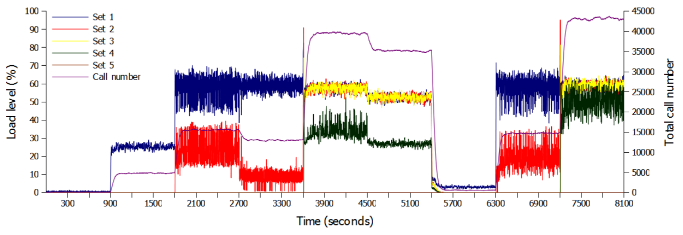


Fig. 12. Simulation result of HSA (basic use-cases)

This provides us with sets operating mostly around the design limit which was set to be 60%[6]. There are some characteristic changes in load, which should be investigated. This first result presents the quick responses the system gives since the data resolution was low thus the changes were sudden.

The first case we should take a closer look at is when the load of groups passes the upper threshold. Take the first occurrence for example when the load suddenly grows high, pushing the first group of FUs over the threshold which triggers the activation of a second set. After this we can see that the load of both sets fluctuates with a higher amplitude. Knowing the algorithm this is caused by the fact that the load of the lower set was around the lower threshold which makes the pair of sets to trade a part of their priority numbers back and forth. In fact this is not a problem though, since the load of the first set is in close range to the design level and no increase in load goes beyond 70%.

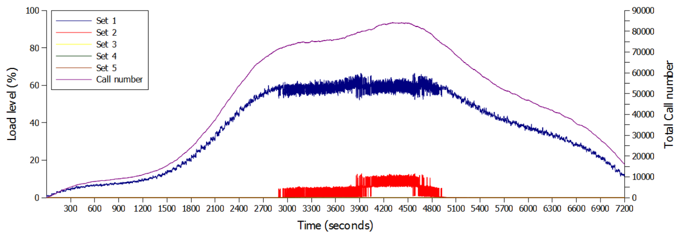


Fig. 13. Simulation result of HSA (simple traffic profile)

In the next interval the load decreases causing the second set to operate under the lower threshold but it remains active and it even decreases the load fluctuations. This is again a consequence of the circumstances, basically the same happens as before with the difference that now the second set gets changed into inactive and back to active repeatedly.

The other interesting case is when the load is growing or decreasing very suddenly (Fig. 12.). In the first case multiple sets are activated but the sudden increase creates a rapid overload in some of those. But the activation of sets and

[6]This is not a setting though it can be achieved by setting the upper threshold to 65%.

increasing the preference number solves the problem. This problem did not appear with the Round-Robin algorithm. The second case where the load drops suddenly the logic turns all but one to inactive without any problem.

In the case of smaller load, only fewer sets are activated (Fig. 13.) which provides a better utilization of resources:

Set 1 (blue line) is loaded until it reaches the upper threshold;
Set 2 (red line) is activated only after Set 1 passes the threshold.

C. Differences between the results

As a last scenario, a simulation for an entire day was created. For this simulation the capacity of the FUs was increased and the number of sets was decreased to only three. Another change was, that the lower threshold was decreased to 10%. The total number of calls during the one day period was almost 19 million, and the traffic had cc. 1.5 million BHCA (Busy Hour Call Attempts). The rate function was generated with the spline method from a measurement on a real network element. Also the rate function was rescaled so that the generated load corresponds to the design load of the configuration.

The results with the Round-Robin algorithm are presented in Fig. 14. The result is similar to the previous cases, i.e. the load is distributed evenly among the sets. The results from the HSA simulation can be seen in Fig. 15. Again, the result shows that only the set activated most lately can remain permanently under the design load level (which is less than the upper threshold).

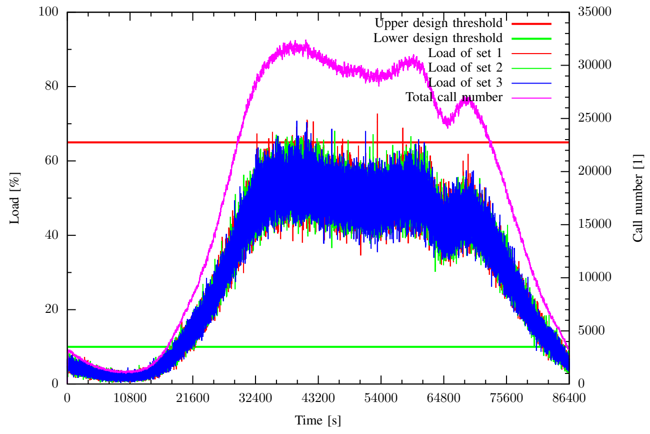


Fig. 14. Simulation result of Round-Robin (one day traffic)

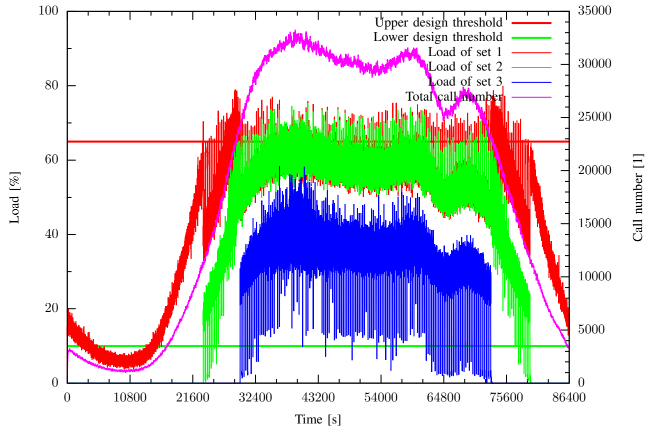


Fig. 15. Simulation result of HSA (one day traffic)

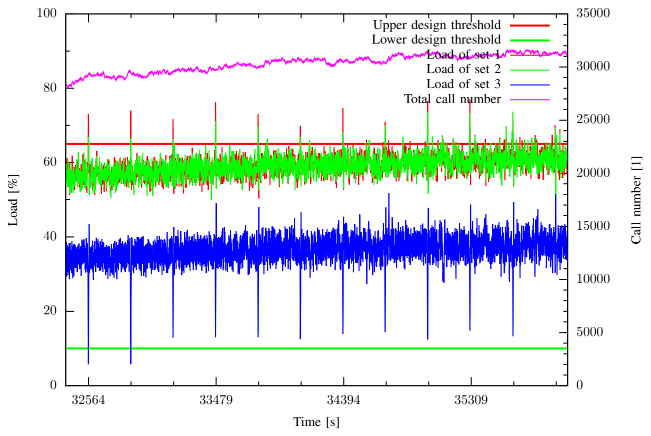


Fig. 16. Section of the 10th hour of Fig. 15

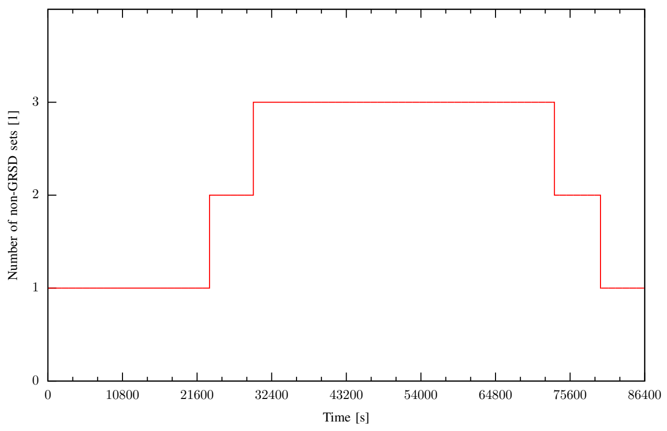


Fig. 17. Number of non-GRSD sets through HSA simulation

An interesting result is that there is an oscillation in the load of sets in Fig. 15. which can be seen better in a section, presented in Fig. 16. As it can be seen the period of these oscillations coincide with the five minutes – i.e. the length of the warming period, while a newly activated set can not be deactivated. The root cause of this phenomenon is that the thresholds and all decisions in the algorithm are strictly enforced. There is no tolerance for passing a threshold, if such event occurs the corresponding intervention is executed. The solution for this problem could be the introduction of soft- and strict thresholds.

Another interesting result is the number of sets that shall be in a powered on state. These are the sets which are not in gracefully shut down (GRSD) state. The number of such sets during the daily simulation is presented in Fig. 17. This result shows that even with the excessive oscillation, the sets are not entering and exiting GRSD state back and forth. In [15] possible decrease of energy consumption – by using HSA instead of Round-Robin – was approximated for different scenarios. It is also visible that – by assuming that energy consumption is related to the set status – the efficiency of the HSA is better compared to Round-Robin, where the number of non GRSD sets is always maximal.

The results presented above show that the new algorithm has a real potential for allowing certain groups of FUs to gracefully shut down. If not needed, they will not even be started. The sets which are activated and later deactivated will not automatically reach the graceful state since they can contain ongoing calls and those can not be terminated. In the simulations calls of extreme length did not occur therefore in most cases inactive sets reached the graceful state.

V. Conclusion

In this paper a short description of load distribution mechanisms in telecommunication network elements was given. The model of the currently used load distribution and load balancing methods was introduced, with emphasis on their weak points. As a proposed solution a new load distribution algorithm was introduced, which supports the horizontal scaling in the distributed system. The algorithm – Horizontal Scaling Algorithm (HSA) – was described in detail and a simulation was proposed.

The presented results of the simulations confirmed the expectations towards the HSA, as it distributed the load among the units more efficiently. Also some issues were identified, which need further investigation. However the overall conclusion is that the new algorithm offers a better distribution method in Cloud based environment compared to the solutions like Round-Robin.

A. Development history

The basic idea of Biased Load Balancing was presented at HTE Infokom 2014 (Kecskemét, Hungary, http://www.hte.hu/web/infokom2014) [13]. A patent was also filed by Nokia in November 2014 [14]. The first proof-of-concept simulation was done in June 2015. The results of the simulations performed in 2015. were presented at a workshop [15] in December 2015. Since then the algorithm was tested in multiple scenarios and the comparison simulation was created.

B. Improvement ideas for the logic

There are a few points where the HSA could be improved. The first one is changing active statuses and preference numbers back and forth for a longer time period. This is a performance issue for the possible application since no action needed is more efficient than any kind of action being executed. We believe that improvements are possible by changing the thresholds and the associated logic. The effect of this issue however is lower than the importance of the next ones.

The second issue is that extremely long calls could prevent groups reaching the GRSD state. The ability to transfer ongoing calls between FUs could improve efficiency by helping more units to reach the graceful state.

The third point is the overload caused by sudden load growth. The idea for this one is rather straightforward, there should be an additional branch in the algorithm. This should decide if the current rate of load growth is bigger than the maximal tolerable increase. Such data can be determined from the load history (stored for a defined window of time) and performance measurement results. If this would be evaluated as true, a set should be activated even if the current load would not require it.

Acknowledgements

The authors would like to thank *István Bakos*, who provided useful comments and some materials for this paper. The authors would also like to express their gratitude to *Gyula Bódog*† for his support in the patent which gave the foundation of this paper.

References

[1] Sagar Dhakal, Majeed M. Hayat, Jorge E. Pezoa, Cundong Yang, David A. Bader: "Dynamic Load Balancing in Distributed Systems in the Presence of Delays: A Regeneration-Theory Approach", *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 18, NO. 4*, Apr. 2007.

[2] Parveen Jain, Daya Gupta: "An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service", *International Journal of Recent Trends in Engineering, Vol 1, No. 1*, May 2009.

[3] Nico Janssens, Xueli An, Koen Daenen, Claudio Forlivesi: "Dynamic Scaling of Call-Stateful SIP Services in the Cloud", Lecture Notes in Computer Science Volume 7289, pp 175-189, 2012.

[4] Gergely Csatári, Tímea László: "NSN Mobile Core Network Elements in Cloud, A proof of concept demo", *Proc. Of IEEE International Conference on Communications, Budapest, Hungary*, 9-13 Jun. 2013.

[5] ETSI GS NFV 002, Network Functions Virtualisation (NFV), Architectural Framework, *V1.1.1*, Oct. 2013. http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf

[6] Nokia Networks: "Open Telecom Application Server Cloud 15.8, Operating Documentation, version 2", *Open Telecom Application Server Product Description, DN09161694, Issue 4-0-0*, © Nokia 2016.

[7] Nokia Networks: "VoLTE and VoWiFi System Documentation, 16.5, Version 1", *Telco Cloud VNF Operations and Maintenance on VMware, System Description, DN09157808, Issue 6-0*, © Nokia 2016.

[8] 3GPP, TS 23.002, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, Network architecture, *Release 12, V12.5.0*, Jun. 2014.

[9] 3GPP, TS 23.218, 3rd Generation Partnership Project, Technical Specification Group Core Network and Terminals, IP Multimedia (IM) session handling, IM call model, Stage 2, *Release 12, V12.3.0*, Sept. 2013.

[10] 3GPP, TS 23.228, 3rd Generation Partnership Project, Technical Specification Group Services and System Aspects, IP Multimedia Subsystem (IMS), Stage 2, *Release 12, V12.5.0*, Jun. 2014.

[11] Athanasios Papoulis, S. Unnikrishna Pillai: "Probability, Random Variables and Stochastic Processes", *4th edition, McGraw Hill*, 2002.

[12] Pedro O.S. Vaz de Melo, Leman Akoglu, Christos Faloutsos, Antonio A.F. Loureiro: "Surprising Patterns for the Call Duration Distribution of Mobile Phone Users", *Machine Learning and Knowledge Discovery in Databases, Lecture Notes in Computer Science Volume 6323, pp 354-369*, 2010.

[13] István Bakos, Gyula Bódog, Attila Hilt, László Jánosi, Gábor Járó: "Resource and call management optimization of TAS/MSS in Cloud environment (in Hungarian), Infokom'2014 Conference, Hungary, Oct. 2014

[14] István Bakos, Gyula Bódog, Attila Hilt, László Jánosi, Gábor Járó: "Optimized resource management in core network element on Cloud based environment", Patent PCT/EP2014/075539, Nov. 2014

[15] Loránd Nagy, Attila Hilt, Lászlo Jánosi, Gábor Járó, and István Bakos: "Simulation of Load Distribution in Telco-Cloud Using Horizontal Scaling Algorithm", Presentation AUF Doctoral Seminar, BUTE (Budapest University of Technology and Economics) Budapest, Hungary, Dec. 2015.