Constraint on Repair Resources, Optimal Number of Repairers and Optimal Size of a Serviced System

Marin Todinov
Department of Computing and Communication Technologies
Oxford Brookes University
Wheatley, Oxford

***Abstract*—The focus of this paper is the analysis of the constraint on the repair resources caused by breakdowns of components in large systems. The study has been conducted by creating a very efficient discrete-event simulator, based on a min-heap data structure, for determining the probability of constraint on the repair resources.**

In finding the right balance between the number of repairers and salary costs, an exact optimisation algorithm has been proposed for the first time. The algorithm determines the optimal number of repairers which guarantees that the probability of constraint on the repair resources will not exceed an acceptable tolerable level. In addition, an exact optimisation algorithm has been proposed for the first time, for determining the maximum size of the system that can be serviced by a specified number of repairers so that the probability of constraint on the repair resources remains below a specified tolerable level. Unlike heuristic optimisation algorithms, the proposed algorithms are exact and always guarantee optimal solutions.

The presented results are of significant importance to operators of computer networks, production systems, transportation networks, water distribution systems, electrical distribution networks etc. They are a solid basis for management decisions regarding the optimal number of maintenance personnel needed to service the breakdowns in large systems. Increasing the number of repairers beyond the optimal level leads to high salary costs while reducing the number of repairers below the optimal number leads to a poor quality of service.

Keywords—constraint on the repair resources; discrete-event simulation; optimization; repairs; optimal size of a system

I. INTRODUCTION

Complex systems include many components experiencing failures at random times and with random repair durations.

The constraint on the repair resources caused by overlapping repair intervals and insufficient number of repairers is an important factor which decreases the availability and quality of service of computer systems.

A particular pattern of random breakdowns combined with a random duration of the repair has been shown in Fig.1. The breakdown events have been marked by b1,b2,b3,..., while the return from repair events corresponding to the breakdown events have been denoted by r1,r2,r3,....

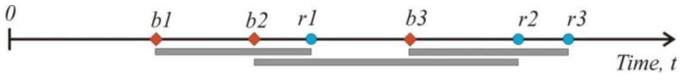

Fig. 1. A specific pattern of random breakdowns and random repair times

Each breakdown engages at least one repairer. In the zones where the repair times overlap, for example zones (*b2*, *r1*) and (*b3*, *r2*), the repairers are engaged in more than one breakdown. This means that several members of the repair team will be simultaneously engaged in fixing breakdowns. The quantity of required repairers depends on the type of breakdown. The return-from-repair events (*r1*,*r2* and *r3* in Fig.1) are associated with restoring the failed devices to a full working state and a release of engaged repairers.

If for a particular breakdown event, the number of remaining repairers is not sufficient to cover the corresponding repair, a constraint on the repair resources occurs, which will result in delays and degraded quality of service of the system.

If the quantity of repairers is more than the optimal quantity, the result will be overstaffing and extra costs. Consequently, there is a delicate balance between the number of repairers and the probability of a delay caused by overlapping repairs. This balance can be found by creating an optimisation model which involves numerous simulations of breakdown-repair histories revealing the probability of a delay caused by insufficient number of available repairers. The maximum tolerable probability of such a delay will be specified in advance, in order to guarantee the required quality of service of the system.

Monte Carlo simulation techniques and discrete-event simulation techniques [1] have become the methods of choice for studying the behavior of complex systems. Monte Carlo simulations related to studying the behavior of the various layers building computer networks already exist. The benefits of simulation techniques applied to analysis of computer systems, modelling and design have been highlighted in [2]. Simulations have also been conducted related to the data traffic carried by computer networks [3,4]. The study of comprehensive review articles on simulation of telecommunication networks [5] shows that, to the best of our knowledge, the constraint on repair resources caused by overlaping repair times *has not yet been studied.* This gap defines the objectives of the present paper:

1) To analyse the constraint on the repair resources caused by component breakdowns in large systems;

2) To determine the optimal number of repairers needed to reduce the probability of a constraint on the repair resources below a specified level;

3) To determine the maximum size of the system that can be serviced by a specified number of repairers, for a specified probability of constraint on the repair resources

Breakdown/failure data related to large systems are vital both in terms of optimal system design and system evaluation. The availability of breakdown data is particularly important to computer networks. According to a number of sources [6,7,10] the negative exponential distribution is an appropriate model for random breakdowns of electronic devices in a given time interval.

If the breakdown density (number of random breakdowns per unit time interval) is denoted by λ, the time to a random breakdown is given by the negative exponential distribution

$$B(t) = 1 - \exp(-\lambda t) \qquad (1)$$

where $B(t)$ is the probability that the time to a breakdown will be smaller than or equal to a specified time *t*. The random delay for repair after a random breakdown was modelled by the negative exponential distribution

$$R(t) = 1 - \exp(-t / MTTR) \qquad (2)$$

where $R(t)$ is the probability that the repair time will be smaller than a given value *t*, and $MTTR$ is the mean time to repair. The negative exponential distribution (2) has been traditionally used for modelling service times [8]. A recent study [9] has indicated that the assumption that repair times follow the negative exponential distribution practically does not affect the calculated availability of various complex systems.

The breakdown/failure density λ in the negative exponential distribution (1) can be estimated from breakdown/failure data related to the components building the computer system. From *n* recorded times to a breakdown t_1, $t_2, ..., t_n$, the mean time to failure/breakdown $MTTF$ can be estimated [6,7] from

$$MTTF \approx \frac{t_1 + t_2 + ... + t_n}{n} \qquad (3)$$

For time to a breakdown following the negative exponential distribution (1), it can be shown that the breakdown/failure frequency λ is related to the mean time to failure (*MTTF*) by the simple relationship [6,7,10]:

$$\lambda = \frac{1}{MTTF} \qquad (4)$$

from which the breakdown/failure frequency λ can be determined if the MTTF is available and *vice versa.*

In developing the model of random times to a breakdown for the separate components, breakdown frequencies determined from analysis of past data published in the literature (e.g. [11]) will be used.

A failed device in the computer system comes back in operation after a delay specified by the time it takes for the component to be fixed [10]. As a result, for any component/device building the computer system, each breakdown event is followed by a random delay for repair (Fig.2) in the interval (0, op_int), during which the computer system is operated.

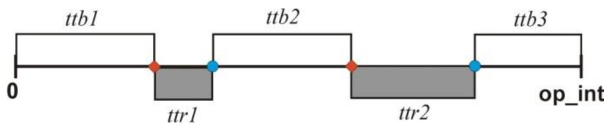

Fig. 2. Random times to breakdown ttb1,ttb2 and ttb3 and random times to repair ttr1,ttr2 for a selected component building the computer system.

Gaining access to failure data is often difficult because failure data are sensitive and not readily disclosed by manufacturers. A number of studies on breakdowns in computer systems already exist ([12, 13,14,15,16,17]. A major drawback in many of these studies ([14], [15], [16], [17]) is that the breakdowns are not attached to specific devices and components. For example, Plank and Elwasif [14] provide time to a breakdown and time to recovery histograms, but these are related to the systems as a whole, not to particular components building the investigated systems. Murphy and Gent [16] discuss system crashes caused by 'software failures', 'hardware failures', 'system management' and 'other reasons', without providing specific data relevant to the separate components building the systems. A similar discussion has been presented by Kalyanakrishnan et al [17], who discuss breakdowns in terms of 'hardware or firmware problems', 'connectivity problems', 'crucial application failures' and problems with a software component'.

The analyses are relevant only to the particular investigated system but not very useful for modelling the behaviour of newly designed systems with similar components, because no sufficient specific failure data were presented about the components building the systems. The research conducted as part of the present study could not identify anonymized failure databases similar to existing databases for the military electronic equipment (for example MIL-HDBK-217F [18]), where electronic components are listed with their breakdown/failure rates. The existence of such databases could help significantly the assessment of new computer systems.

The need for such databases is reinforced by studies [13] indicating that the component failures are not uniformly distributed among the different devices building the computer systems. A small fraction of devices are responsible for the majority of the recorded failures and this circumstance is vital for modelling the expected availability of new systems. Some devices, due to their nature, are subjected to a much higher workload compared to other devices and are more prone to breakdowns (e.g. file servers conducting a large number of I/O operations). Some devices including electro-mechanical parts (disk drives) are much more prone to breakdowns compared to devices which do not include such parts.

Despite these difficulties, failure data related to specific componets in computer networks have been presented by Gill et al [11], Schroeder and Gibson [19,20], Labotitz and Ahuja [21] and Sahoo et al, [13].

In their analysis, Schroeder and Gibson [19] noted that the system size is not a significant factor in the repair time. A set of failure data has been recently collected at a high-performance computing site and was made available online (Failure data [22]). Statistical failure analysis of web server systems has been presented by Fujii and Dohi [23], where failure rate functions characterising web servers can be found.

Valuable failure data sets have been presented by Gill et al [11], collected by computer systems operators employing a ticketing system. The tickets contain important information about when and how the breakdown events have been discovered as well as when they were resolved. In addition, the tickets also contain a description of the cause of the problem, and the specific device at fault. Event logs were collected during one year of operation and two types of breakdowns have been defined: 'link failures' in the case where the connection between two devices was down and 'device failures', when a device was not functioning for routing/forwarding traffic.

II. Determining the Optimal Number of Repairers and the Optimal Size of the Serviced System

A. *Determining the optimal number of repairers which guarantee a probability of constraint on the repair resources below a specified tolerable level*

Determining analytically the probability of constraint on the repair resources for more than a single available repairer is a complicated task. This task can be simplified by constructing a discrete-event simulator for modelling the failure-repair history. Furthermore, there is an optimal balance between the number of repairers and the probability of constraint on the repair resources. This optimal balance can be found by creating an optimisation model that involves a suitable variation of the number of repairers followed by determining the probability of constraint on the repair resources, until the optimal balance is achieved.

An initial amount of repair resources is assigned and the discrete-event simulator is called to calculate the probability of constraint on the repair resources. Clearly, for zero number of repairers, the probability of constraint on the repair resources will be higher than the target probability α and close to unity (Fig.3). This is because any single component breakdown will create a constraint on the repair resources simply because there will be no available repairers to handle the breakdown.

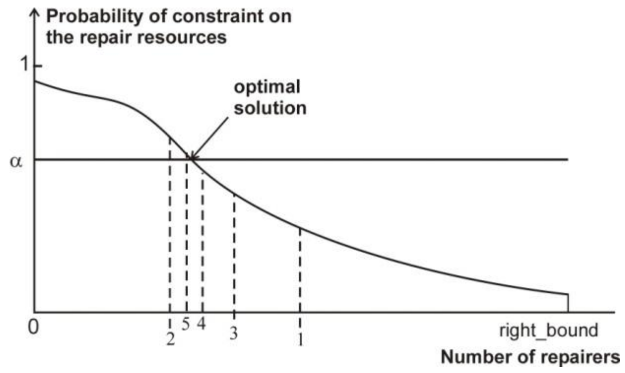

Fig. 3. A procedure for determining the optimal number of repairers for a specified probability α of constraint on the repair resources.

Similarly, for a large number of repairers (number of repairers = right_bound), it is highly likely that there will be an available repairer for every single breakdown and the probaility of runing out of repairers will be smaller than the target probability α . Between these two extremes lies the optimal solution where the number of repairers is just enough to guarantee the target probability α of constraint on the repair resources. This optimal solution can be found by a repeated bisection of the interval (0, right_bound).

The probability of constraint on the repair resources is determined from risk management considerations and is specified as an input paratemer by the manager of the computer system. This probability depends on the criticality of the supplied service. A probability of constraint on the repair resources equal to 20% may be sufficient for non-critical computer systems (e.g. school network; computer LANs) but it is insufficient for critical computer sustems (e.g. server backbone networks, nuclear power plant moniotoring systems, computer networks providing high-speed data transfer for electrical distribution systems; data storage computer systems; secure remote access networks, etc.). For critical computer systems, a large probability of a constraint on the repair resources means a large probability that repairers will not be available when needed, which could be associated with serious consequences. Setting the correct level of the probability of constraint on the repair resources is done by experts, after a careful risk assessement and is not a subject of this study.

B. *Determining the maximum size of the system which guarantees a probability of constraint on the repair resources below a specified tolerable level*

For a given number of repairers, the maximum size of the computer system which guarantees a probability of constraint on the repair resources below a specified tolerable level can be determined by a similar repeated bisection technique (Fig.4).

This time, the parameter which is varied is the number of components in the computer system while the number of repairers is kept constant.

Clearly, for zero number of components, the probability of constraint on the repair resources is zero. For a large number of components experiencing breakdowns, the probability of constraint on the repair resources will be larger than the specified tolerable level α. The optimal solution is located between these two extremes. It corresponds to a point where the number of components in the system is just enough to guarantee the target probability α of constraint on the repair resources (Fig.4). The optimal solution can be found by a repeated bisection. A serviced system size, significantly smaller than the optimal solution, means that the available repairers are not used efficiently. A serviced system size, significantly larger than the optimal solution, means that there exists an increased probability of constraint on the repair resourecs and a low quality of service.

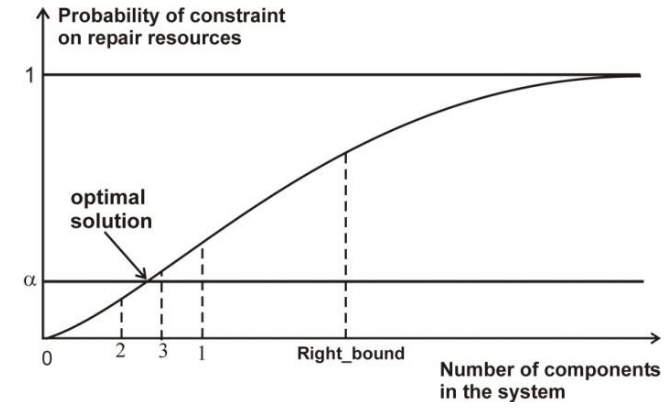

Fig. 4. An optimisation procedure for determining the optimal size of the serviced system

C. An optimization algorithm

The breakdown events mark the engagement of a repairer. The return-from-repair events mark the release of repairers. Each event is represented as a record ‘ev_record’ with four fields:

'time_of_event' - stands for the time of occurence of the event;

'e-type' - stands for the type of event ('breakdown' or 'return from repair');

'req_rep' - stands for the quantity of engaged/released repairers;

'id' - stands for the component index.

The traditional way of implementing discrete-event simulators is by linked lists [1,24]. In this study, the discrete-event simulator has been implemented by using a min-heap (priority queue). The main reason is that the retrieval of the event with the smallest time from a linked list is an operation of time complexity $O(n)$, where *n* is the number of events in the list. In contrast, the retrieval of the event with the smallest time, followed by restoring the min-heap property, is an operation of time complexity $O(\log_2 n)$.

The events are placed in a min-heap which is essentially a binary tree coded in an ordinary array. In the min-heap, the time stamp 'te' of each predecessor node is smaller than each of the time stamps of its successor nodes (te1 < te2; te1< te3; te2<te4; te2<te5; te3<te6; te3<te7; te4<te8; te4<te9).

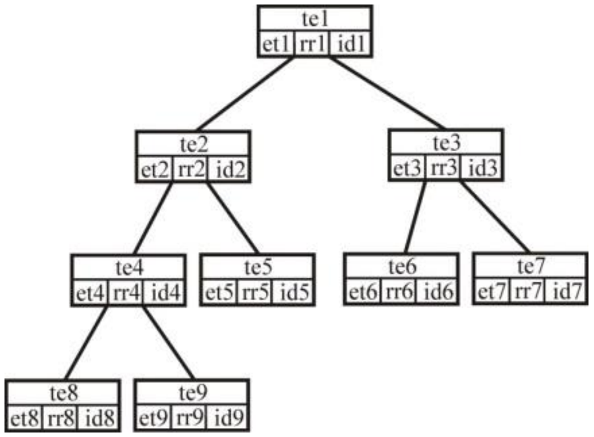

Fig. 5. Min-heap used for implementing the discrete-event simulator.

As a result, inserting an event in a min-heap with *n* events or removing an event, followed by restoring the min-heap property, are both operations of time complexity $O(\log_2 n)$. Standard functions ‘*sift*’ and ‘*buble up*’ can be used to restore the basic property of the min-hip: the descendants of each element of the min-heap must be with a larger time stamp. Detailed discussions regarding the programming implementation of min-heaps can be found in [25].

Algorithm

```
procedure
generate_ttb_and_ret_from_repair(current_time, j);
 {
  // Generates the time to breakdown event and the
     return-from-repair event for component i and places
     these events in the min-heap
 }
procedure remove_event()
   {
     // Removes the event with the smallest time from the top of the
        min-heap
   }
function prob_constraint_rep_res(num_repairers)
{

count=0;    // Counts the number of simulation trials  during which
                a constraint on repairers has occurred

for i=1 to num_simul_trials do
   {
    // initialises the quantity of remaining repair resources
    rem_resources = num_repairers;

       // initialise the current time of the discrete-event simulator
    current_time = 0;

    heapsize=0;    //clears the min-heap
// for all components generates events 'time of a breakdown' and
'time of return from repair' and places  them in the min-heap:
```

```
for j=1 to num_components do
generate_ttb_and_ret_from_repair(current_time,j);
// while there are events in the min-heap and the the current time is
smaller than the operational time do the while loop:
  while (heapsize>0 and current_time < op_int) do
         {
          ev=a[1]; //Take the event on the top of the min-heap
           // Takes the time stamp of the top event in the min-heap
          current_time = ev.time_of_event;
        if (ev.e_type = 'breakdown')  {
            // checks for a constraint on the repair resources
            if (rem_resources < ev.req_rep)
             {
               count=count+1;
               break;
             }
     else {
       // allocates resources for the current breakdown event
       rem_resources = rem_resources - ev.req_rep;

    // removes the first event from the min-heap;
    remove_event();
          }
                                              }
 else {  // the event is a return-from- repair event
        rem_resources = rem_resources + ev.req_rep;
        remove_event();
 // generate time of a breakdown event and time of a recovery event
   and place them in the min-heap;
generate_ttf_and_ret_from_repair(current_time,
                                 ev.id);
      }
         }
    }
prob_of_constr = count/num_simul_trials;
return prob_of_constr;
}
// Code of the repeated bisection part
  left=0; right = right_bound;
 while (left + 1 < right) do
{
 mid = (left+right) / 2;
  cur_probability = prob_constraint_rep_res(mid);
    if (cur_probability <=alfa ) right = mid;
      else left = mid;
}
    optim_num_rep = right;
```

The probability of constraint on the repair resources depends on the number of available repairers which is an important parameter of the function ***prob_constraint_rep_res***. This parameter is passed to the function through the variable ‘`num_repairers`’. At any point of the simulation, the variable ‘`rem_resources`’ shows the current number of remaining repairers. The variable ‘`current_time`’ tracks the time of occurence of the current event. The content of the clock ‘`current_time`’ is compared to the length of the operation interval ‘`op_int`’ and if this is exceeded, the current simulation is terminated. At the start of each simulation trial, an empty min-heap is initialised by making the size of the min-heap zero with the statement '`heapsize=0`'.

The simulation starts with generating all breakdown events and return from repair events for all components and placing them in the min-heap. This is done by the procedure ***generate_ttb_and_ret_from_repair***`(current_time,j)`, which takes as parameters the current time of the discrete-event simulator and the index 'j' of the component. The random time to a breakdown for the component with index 'j' is generated by a function which takes as parameter the breakdown frequency of the component with index 'j' and generates a random time to a breakdown by using the *inverse transformation method* [26] for sampling from the negative exponential distribution (1). This method consists of generating a random number x_i uniformly distributed in the interval (0,1) [27,28] and determining the corresponding random time to a breakdown t_i through the inverse function of the distribution of the time to a breakdown:

$$t_i = -\frac{1}{\lambda}\ln[1-x_i] \tag{5}$$

where λ is the breakdown/failure frequency of the corresponding component.

The actual time of the component breakdown is obtained by adding the current time of the simulator to the random time to a breakdown. Next, the procedure ***generate_ttb_and_ret_from_repair***`(current_time,j)` generates a random time to repair. A random time to repair is generated by taking the mean time to repair MTTR of the failed component and using the inverse transformation method to sample from the negative exponential distribution (2). Again, a uniform random number x_i is generated [27,28] in the interval (0,1) and the random time to repair t_i is obtained from

$$t_i = -MTTR \times \ln[1-x_i] \tag{6}$$

The return time from repair is obtained by adding the current time of the simulator, the random time to a breakdown and the random time to repair.

Checks are also performed whether the breakdown time and the time of return from repair are smaller than the length of the operation interval '`op_int`'. If this is the case, event records are made with the corresponding time stamps and are subsequently inserted in the min-heap. The min-heap data structure is contained in the array a[] whose elements are event records.

For the sake of simplicity, the implementation details related to inserting an event in the min-heap have been omitted here.

In the simulation loop, the nested while-loop goes through the events from the min-heap by always taking the event from the top of the mean-heap (Fig.5) with the statement '`ev=a[1]`'. This is the event with the smallest time stamp, therefore this is the event which will occur next. The current time of the simulator is updated with the earliest time by the statement '`current_time = ev.time_of_event`'.

Next, for the extracted from the min-heap event, a check is performed whether there will be available repairers to cover the demand for repair. If the remaining repairers are smaller than the required number of repairers, a constraint on the repair resources has occurred, the counter 'count' is inclemented and the while-loop is exited immediately by the statement '**break**'. This is done by the fragment:

```
if (rem_resources < ev.req_rep)
  {
    count=count+1;
    break;
   }
```

If the remaining repairers are sufficient to cover the demand for repair from the current breakdown event, the fragment

```
else {
   rem_resources = rem_resources - ev.req_rep;
   remove_event();
     }
```

allocates repairers to the breakdown event and decreases the amount of available repairers by the amount 'ev.req_rep' required by the breakdown event. Next, the breakdown event is removed from the min-heap by calling the procedure '***remove_event***()'. Implementation details have been omitted here, for the sake of simplicity. New events with random breakdown time and random return from repair time are generated only when the current event is of type 'return from repair'. This is done in the fragment

```
else {
  rem_resources = rem_resources + ev.req_rep;
  remove_event();
  generate_ttb_and_ret_from_repair(current_time,
                                  ev.comp_indx);
     }
```

In this fragment, a release of engaged repairers is made first by the statement

```
rem_resources = rem_resources + ev.req_rep;
```

which is followed by removing the event from the min-heap by calling the procedure ***remove_event()***.

This is followed by calling the procedure ***generate_ttb_and_ret_from_repair***(current_time, ev.id), which generates new time-to-a-breakdown event and return-from-repair event for the corresponding component, with index 'ev.id'.

These steps are repeated while there are still events in the min-heap or the current time of the simulator is smaller than the length of the operational interval 'op_int'. The probability of constraint on the repair resources is obtained in the statement

```
prob_of_constr = count/num_simul_trials;
```

by dividing the number of times constraint on the repair resources has been registered (the content of the counter 'count') to the total number of simulation trials num_simul_trials.

Next, the fragment of the repeated bisection follows, from which the discrete-event simulator is called a number of times until the optimum is reached.

Finally, the algorithm for determining the optimal size of the system that can be serviced by a given number of repairers (so that the probability of constraint on the repair resources remains below a specified level) has also been implemented. This algorithm is very similar to the one described earlier and will not be presented here.

III. A Simulation Study and Results Related to a Large Computer System Experiencing Random Component Failures and Random Repair Times

In the simulation study, the random time to a breakdown was modelled by the negative exponential distribution (1) while the random repair time was modelled by equation (2). The data related to the number densities of the breakdowns related to the sparate devices and the mean time to repair have been taken from published studies ([11,13,19,20,21].

By using the described algorithm, the probability of constraint on the repair resources for a large computer system has been studied. The computer system included 11 different types of components: Workstations, WLAN transmitters, Printers, Servers; VoIP Servers, Routers, Switches, Fiber optical cables, Ethernet cables, Console cables and Monitoring computers.

Each of these devices is characterised its own breakdown/failure rate and mean time to repair. A single repairer was assumed for each of the failed devices.

The overall number of devices which experience breakdowns and require a repairer was 632. The simulation was run on a computer with 3 GHz Quad Core CPU. For 3 available repairers, a probability of constraint on the repair resources equal to 0.46 was calculated within 3.73 seconds, based on 10000 simulation trials. This demonstrates the high computational efficiency of the developed simulator based on a min-heap. The convergence was also very good because increasing the number of simulations to 100000 resulted in a very similar value (0.45) for the probability of constraint on the repair resources.

The probability of constraint on the repair resources as a function of the number of available repairers is given in Fig.6. As can be seen from the plot in Fig.6, there is a critical number of repairers for which the probability of constraint on the repair resources decreases sharply. For the considered computer system, this critical number is 4. For 3 repairers, the probability of constraint on the repair resources is 46%. Adding one more repairer however, causes the probability to drop sharply to 4.4%. The subsequent increase of the number of repairers causes only an insignificant decrease of this probability. The conclusion is that keeping more repairers than the critical number increases the salary costs without bringing a substantial decrease in the probability of constraint on the repair resources and an increase in the quality of service. Conversely, a number of repairers smaller than the critical number is associated with a large probability of constraint on the repair resources and a reduced quality of service.

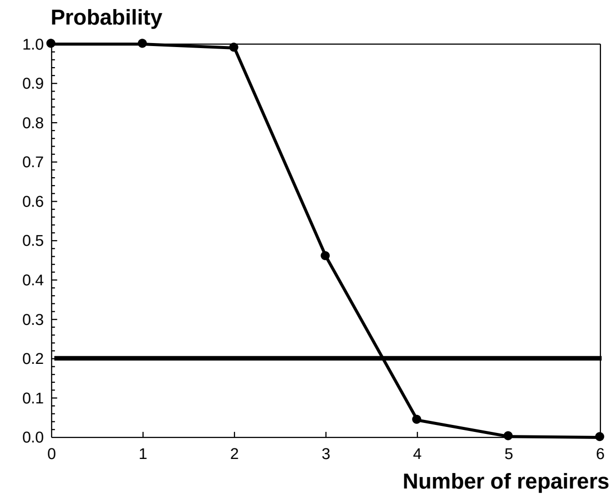

Fig. 6. Variation of the probability of constraint on the repair resources with the number of repairers

Determining the optimal number of repairers was conducted by using the same set of input data. The specified level of the probability of constraint on the repair resources was 20% ($\alpha = 0.2$).

The optimal solution was bounded between the values 0 and 30. Indeed, a number of repaires equal to zero yields 100% for the probability of constraint on the repair resources. A number of repairers equal to 30 yielded 0% probability of constraint on the repair resources. Consequently, the optimal solution must be located within the interval (0,30). Running the repeated bisection algorithm yield an optimal number of repairers equal to 4. The corresponding probability of running out of repair resources for this number of repairers is 0.044 (4.4%). The results from the previous simulation served as a validation test for the optimisation procedure. As it can be verified from Fig.6, four repairers is indeed the optimal number of repairers because 4 is the smallest number of repairers which yields a probability of constraint on the repair resources smaller than 20%. Three repairers yield probability equal to 0.46 - significantly larger than the specified maximum tolerable probability of 20%.

To determine the optimal size of the computer system which can be serviced by a given number of repairers, a weighted averaged breakdown frequency of 0.358 year^{-1} and a weighted average time to repair equal to 0.28 days has been used, taken from published breakdown data related to computer networks. Two levels of the maximum tolerable probability of constraint on the repair resources have been specified: $\alpha = 20\%$ and $\alpha = 2\%$. The optimal size of the network serviced by a different number of repaires, at the specified probability of constraint on the repair resources, is shown in Fig.7.

A serviced system size significantly smaller than the optimal size means that the available repairers are not used efficiently. A serviced system size significantly larger than the optimal size means that there is an increased probability of constraint on the repair resourecs which entails a low quality of service.

The developed discrete-event simulator also permits investigating the constraint on the repair resources related to specific components.

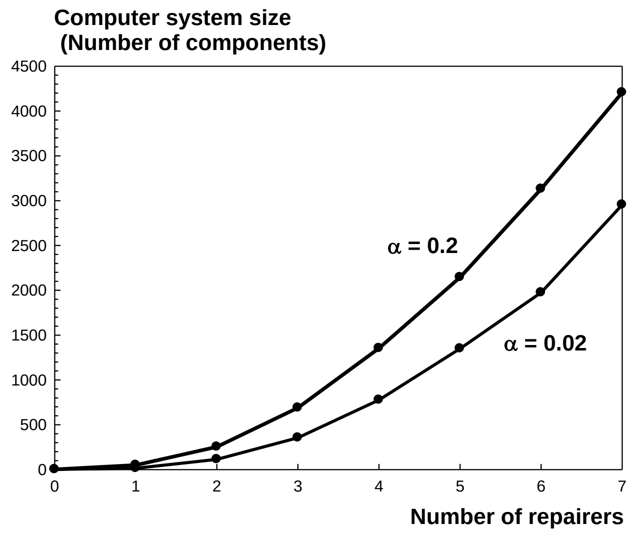

Fig. 7. Optimal serviceable size of the computer system for different number of available repairers

The failures of top of rack switches require the intervention of an operator. The failure frequency of top of rack switches according to [11] is $\lambda = 0.038$ year^{-1}. The cumulative empirical distribution of the time to repair is given in Fig.8.

The simulation based on 300 switches and 10000 simulation histories revealed a probability of constraint on the repair resources equal to 0.11. The time for computing this result on a computer with 3 GHz Quad Core CPU was 1.18 seconds. The empirical time to repair distribution in Fig.8 was sampled by combining the inverse transformation method and linear interpolation (the arrows in Fig.8).

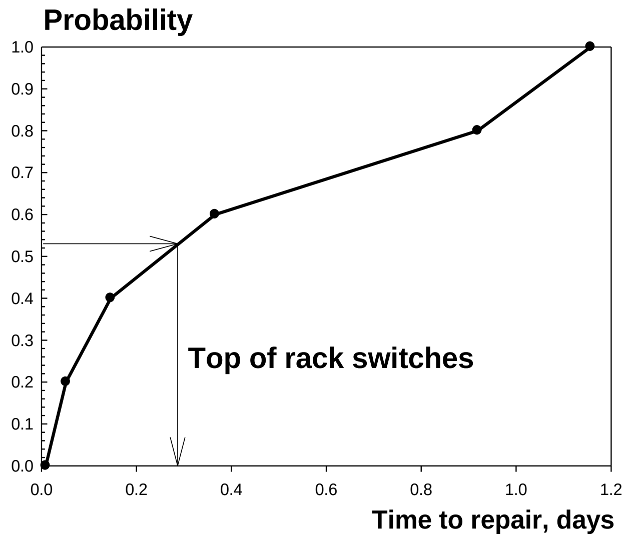

Fig. 8. Time to repair distribution of top of rack switches in computer systems (according to Gill et al.[11])

IV. Investigating the Influence of Critical Parameters on the Probability of Constraint on the Repair Resources

A. *Investigating the influence of the failure frequency and mean repair time on the probability of constraint on the repair resources*

The influence of the breakdown frequency on the probability of constraint on the repair resources has been tracked by keeping all the parameters constant except the breakdown frequency λ of the components, which was varied in the interval (0 -:- 4.0 year^{-1}), Fig.9. Ten components have been used, for each of which the same breakdown frequency λ has been assumed and one repairer was required for each component failure. For the purposes of the parametric study, the repair times have been assumed to follow a Gaussian distribution. A common mean time to repair MTTR=1.5 days has been assumed for each component, with a standard deviation σ equal to 0.15 days.

The operational interval was set to be 1 year. Two distinct cases have been investigated: (i) A single available repairer and (ii) two available repairers. The results have been summarised in Fig.9.

The results in Fig.9 reveal *a unexpectedly large probability of constraint on the repair resources* for a single available repairer. For the system including 10 components, each of which experiences on average 2 breakdowns a year ($\lambda = 2$, year^{-1}), with 1.5 days duration for repair, the probability of constraint on the repair resources during one year of operation is 73%! If the components experience $\lambda = 3$ instead of 2 expected number of failures per year, the probability of constraint on the repair resources is already 94%! These unexpected results show how easy it is to underestimate the probability of constraint on the repair resources. The result from such a underestimation are poor management decisions related to the number of people necessary to maintain a large system.

From the graphs, it can also be seen that increasing the number of repairers decreases significantly the probability of constraint on the repair resources. The significant decrease of the probability of constraint on the repair resources with the inclusion of a second repairer can be explained by the following.

If a single repairer is present, a constraint on the repair resources occurs whenever the repair times of two failed components overlap. The overlap means that a repairer is required at two different places (for two different failed components). The result is a constraint on the repair resources and a delay.

If two repairers are available, even if in the presence of overlapping repair times from two failed components, the repairers will work in parallel and no constraint on repair resources will occur. A constraint on the repair resources with two available repairers occurs only if three repairs simultaneously overlap (have a common point in time), the probability of which is relatively small. Providing an extra repairer brings a dramatic decrease in the probability of constraint on the repair resources.

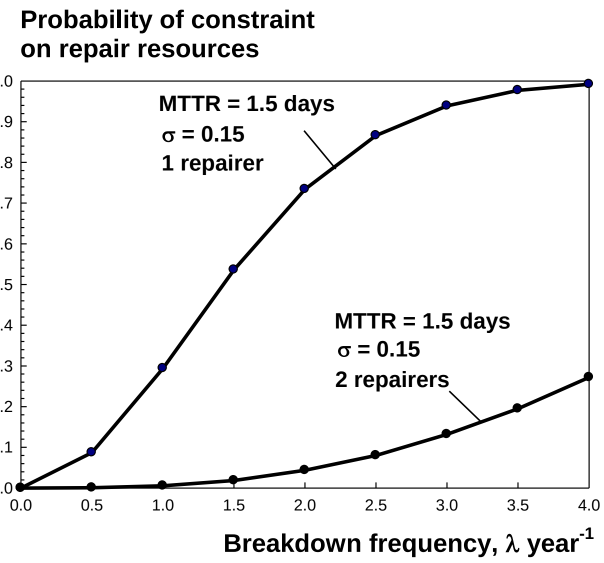

Fig. 9. Dependences of the probability of constraint on repair resources on the breakdown frequency of the components for different number of repairers

Next, the influence of the mean repair time on the probability of constraint on the repair resources was investigated by keeping all the parameters constant except the mean repair time. To separate the influence of the mean repair time from the influence of the standard deviation of the repair time, Gaussian distribution has been assumed for the distribution of the repair times. Note that for a negative exponential distribution of the times to repair, the mean and the standard deviation are both equal to MTTR [29], and their influences on the probability of constraint on the repair resources cannot be separated.

The mean repair time was varied in the interval (1-:-20 days; Fig.10). Ten components have been used, for each of which two different breakdown frequencies were assumed $\lambda = 1.0$ year^{-1} and $\lambda = 0.5$ year^{-1} and one repairer was required for each component failure. The number of available repairers was one; the standard deviation of all repair times was assumed to be constant: $\sigma = 0.15$ days. The results are presented in Fig.10.

With increasing the mean time to repair, the probability of constraint on the repair resources monotonically increases. Large times to repair yield a relatively small increase in the probability of constraint on the repair resources. No matter how large the mean time to repair is, there is always a non-zero probability that, within one year of operation, there will be no component breakdowns in the system. As a result, the probability of constraint on the repair resources tends asymptotically to unity. At a constant mean time to repair, the breakdown frequency has a strong effect on the probability of constraint on the repair resources.

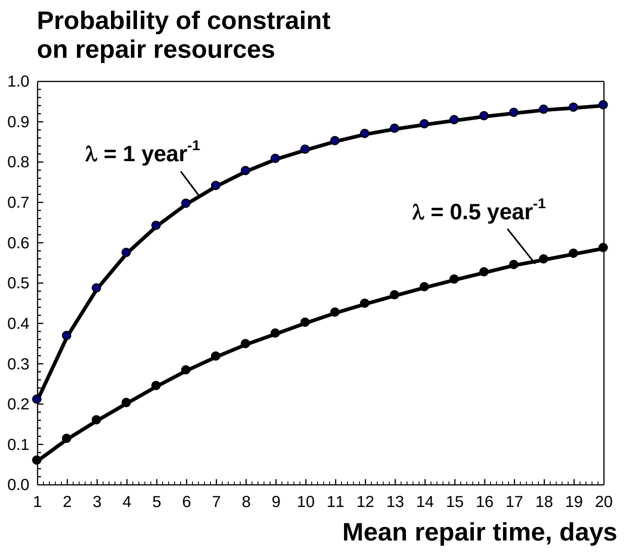

Fig. 10. Dependence of the probability of constraint on repair resources on the mean repair time

By using the discrete-event simulator, the influence of the standard deviation of the time for repair, on the probability of constraint on the repair resources, has been investigated. A system including ten identical devices, each characterised by a breakdown frequency $\lambda = 0.5$ year^{-1}, has also been simulated. Again, to separate the influence of the standard deviation of the repair time from the influence of the mean repair time, the times to repair of the devices have been assumed to follow a Gaussian distribution. The mean repair time was MTTR=60 days and the standard deviation of the repair time was varied from 0 to 10 days. The simulation experiments were repeated for one, two and three available repairers. The results indicated that the probability of constraint on the repair resources is insensitive to the variation of the standard variation of the repair time. These results showed that the probability of constraint on the repair resources cannot be reduced by reducing the variances of the repair times. The only way to reduce the probability of constraint on the repair resources is to reduce the mean repair times.

An important extension of this study is the analytical treatment of the problem related to estimating the probability of constraint on the repair resources. In addition, the developed methods could be applied with success to optimise the maintenance of production systems, transportation networks, water distribution systems and electrical distribution networks.

V. Conclusions

1) An optimisation algorithm has been proposed for the first time, for determining the optimal number of repairers maintaining a computer system of given size. The optimal solution guarantees that the probability of constraint on the repair resources will not exceed a maximum tolerable level.

2) An optimisation algorithm has been proposed for the first time, for determining the optimal size of a computer system which can be serviced by a given number of repairers. The optimal solution guarantees that the probability of constraint on the repair resources does not exceed a maximum tolerable level.

3) Both optimization algorithms are based on a repeated bisection. Unlike heuristic optimisation algorithms, the proposed algorithms are exact and always guarantee optimal solutions.

4) A very efficient discrete-event simulator has been created for the first time, for determining the constraint on the repair resources in large computer systems. The discrete-event simulator handles very large systems including thousands of components and is characterised by a very high computational speed.

5) The implementation of the discrete-event simulator is based on a min-heap data structure which guarantees that each operation involving inserting and deleting an event has a logarithmic running time. This running time associated with the insertion and removal of events is at the heart of the high computational speed of the developed discrete-event simulator.

6) The simulation results indicated the existence of a critical number of repairers at which the probability of constraint on the repair resources decreases sharply.

7) The parametric studies revealed an unexpectedly high probability of constraint on the repair resources for a single available repairer. This unexpected result shows how easy it is to underestimate the probability of constraint on the repair resources, which leads to poor management decisions.

8) The parametric studies indicated that providing more than a single repairer decreases dramatically the probability of a constraint on the repair resources.

9) Breakdown data and maintenance data related to computer systems should be component-specific. This provides the opportunity to use discrete-event simulators for optimizing the maintenance of computer systems.

10)An anonymized breakdown data base, similar to databases already existing for military electronic equipment, is a solid base for assessing and designing new computer systems and making correct management decisions.

11)This study could be extended by estimating the probability of constraint on the repair resources analytically. The developed methods could also be applied for optimising the maintenance of production systems, transportation networks, water distribution systems and electrical distribution networks.

References

[1] J. Banks, J.S. Carson, B.L. Nelson, D.M. Nicol, Discrete-event simulation 4th ed., Prentice Hall, 2005.

[2] H. Al-Bahadili, Simulation in Computer Network Design and Modelling: Use and Analysis, Petra University, Jordan, 2012.

[3] M. Zukeman, D.Neame, R.Addie, Internet Traffic modelling and future technology implications, Proceedings of the 2003 InfoCom Conference, San Francisco, CA, 2003.

[4] P.Barford and M.Crovella, An architecture for a WWW workload generator, *Proceedings of the 1998 SIGMETRICS Conference*, Madison, WI, pp.151-160, 1998.

[5] N.I.Sarkar N.I., S.A. Halim, "A Review of Simulation of Telecommunication Networks: Simulators, Classification, Comparison, Methodologies, and Recommendations", Journal of Selected Areas in Telecommunications (JSAT), March Edition, 2011, pp.10-17.

[6] K.S.Trivedi, Probability and statistics with reliability, queuing and computer science applications, 2nd ed., Wiley, 2002.

[7] L.C. Wolstenholme, Reliability modelling, a statistical approach, Chapman & Hall, 1999.

[8] N.A.Weiss, *A course in probability*, Pearson Education, Inc., 2006.

[9] C.M.Carter, A.W. Malerich, The Exponential Repair Assumption: Practical Impacts, Proceedings of the Reliability and Maintainability Symposium, 2007. RAMS '07. Annual, Orlando, FL (2007).

[10] C.E.Ebeling, *An introduction to Reliability and Maintainability Engineering*, McGraw-Hill, (1997).

[11] P.Gill, N.Jain, N.Nagappan, "Understanding Network Failures in Data Centers: Measurement", Analysis, and Implications, SIGCOMM’11, August 15-19, 2011.

[12] D. Tang, R.K. Iyer, and S.S. Subramani., "Failure analysis and modelling of a VAX cluster system". In *FTCS*, 1990.

[13] R.K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang. "Failure data analysis of a large-scale heterogeneous server environment", In *Proc. of DSN’04*, 2004.

[14] J.S. Plank and W. R. Elwasif. Experimental assessment of workstation failures and their impact on checkpointing systems. In *FTCS’98*, 1998.

[15] D. Nurmi,, J. Brevik, and R. Wolski. Modeling machine availability in enterprise and wide-area distributed computing environments. In *Euro-Par’05*, 2005.

[16] B. Murphy and T. Gent. Measuring system and software reliability using an automated data collection process. *Quality and Reliability Engineering International*, 11(5), 1995.

[17] M. Kalyanakrishnam, Z. Kalbarczyk, and R. Iyer. "Failure data analysis of a LAN of Windows NT based computers", In *SRDS-18*, 1999.

[18] MIL-HDBK-217F, *Reliability prediction of electronic equipment*, US Department of Defence, Washington, DC, (1991).

[19] B. Schroeder., G.A.Gibson, "A large-scale study of failures in high-performance computing systems", Proceedings of the International Conference on Dependable Systems and Networks (DSN 2006), Philadelphia, June 25-28, 2006.

[20] B. Schroeder, G.A.Gibson, "The computer failure data repository (CFDR)", Workshop on Reliability Analysis of System Failure Data (RAF'07) MSR Cambridge, UK, March 2007.

[21] C. Labovitz C. and A. Ahuja. Experimental study of internet stability and wide-area backbone failures. In *The Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, 1999.

[22] Failure data, 2006 (related to computer networks) http://www.pdl.cmu.edu/FailureData/ and http://www.lanl.gov/projects/computerscience/data/, 2006.

[23] T.Fujii and T.Dohi, Statistical analysis of a web server system, 2009 International Conference on Availability, Reliability and Security, pp.554-559, 2009.

[24] L.Leemis, L. M., Park, S. K. Discrete-event simulation: A first course. Upper Saddle River, N.J.: Pearson Prentice Hall (2006).

[25] T.H.Cormen, T.C.E.Leiserson, R.L.Rivest, and C.Stein, *Introduction to Algorithms*, 2nd ed., MIT Press and McGraw-Hill, (2001).

[26] S. Ross, *Simulation 2nd edition*, Harcourt academic press, 1997.

[27] L’Ecuyer, Efficient and portable random number generators, *Communications of the ACM*, vol.31, pp.742-749, 1988.

[28] Park S. and K.Miller, *Random number generators*, Commun. ACM, vol.31 (10), (1988), pp. 1192-1201.

[29] S.Ross, Introduction to probability models, 7th ed., Harcourt Academic press, 2000.