

A Risk Analysis of Disk Backup or Repository Maintenance

Mark Burgess[†] and Trond Reitan[‡]

[†]*Oslo University College, P.O.Box 4, St Olavs Plass, 0130 Oslo, Norway*

[‡]*University of Oslo, P.O. Box 1053 Blindern, 0316 Oslo*

Abstract

We discuss a simple model of disk backups and other maintenance processes that include change to computer data. We determine optimal strategies for scheduling such processes. A maximum entropy model of random change provides a simple and intuitive guide to the process of sector based disk change and leads to an easily computable optimum time for backup that is robust to changes in the model. We conclude with some theoretical considerations about strategies for organizing backup information..

Key words: Computer models, dynamical system, system administration, mean field theory

1 Introduction

Change is a characteristic of computer systems as they interact with users by keyboard or by network access. Some changes are desirable, while others are considered to be contrary to policy and must be repaired through maintenance. In this paper, we consider the problem of formulating optimal and desirable policies for maintaining systems undergoing change. We are particularly interested in the common, yet understudied problem of disk backups.

While backup schedules and technologies have been discussed a lot — see refs. [1–20], to our knowledge no one has tried to formulate a theory of risk management using backup as a maintenance process explicitly, though see [21] for related ideas. The closest one comes is perhaps the fault tree analysis in ref. [22].

System policy refers to the decisions and schedules that govern the configuration and running of a human-computer system. Policy is generally im-

plemented in terms of rules for behavior and procedures for maintenance. It applies both to humans and to computers. Clearly, policy should be based on a sound model of system behavior [23] if it is to lead to a successful decision-making. Policy success entails both the efficiency and security of the system.

In this paper, we shall consider only the aspects of policy that are programmable into a computer. This means, we effectively identify policy with rules for correctness of system state. We can then think of change in terms of a simple linear response model.

Suppose we imagine, heuristically, that the set of all data is given by Q , and that the subset of data that are redundantly stored as an up-to-date version, according to policy, is P . We can define the maintenance set δQ in terms of deviation from policy:

$$\delta Q(t) = Q(t) - P(t). \tag{1}$$

Thus, when a system obeys backup policy the need for maintenance (backup) δQ is zero. The important point here is that the relationship between P and Q changes as a function of time, because changes are being made to the data in real-time.

2 Disk backups

Disk backups are an important area for system policy. They involve deciding when to schedule the archiving of data, in order to protect against accidental or catastrophic loss. Most system administrators have a routine for this that is based on minimizing their perception of load on the system rather than on minimizing the risk of data loss.

For example, system administrators often take backups during the night, when they believe the system is least active. However, this practice is motivated more by folklore than by science, nor is it necessarily possible to find such a time in a globalized network age, where systems are active twenty-four hours per day, seven days per week. A more scientific approach is called for.

In this paper we would like to ask a specific question: when should a backup be made in order to minimize the risk of loss? Risk can be defined in many ways. Here we shall take a look at some traditional and pragmatic approaches to defining the meaning of risk. In statistics, risk is defined as expected loss, thus for this application we get the following definition;

Definition 1 (Maintenance Risk) *A numerical value proportional to the expected loss of objects in the maintenance set for a given policy.*

The total risk is usually the maintenance risk plus the expected cost of implementing the policy.

$$R_{total} = R_{maintenance} + C_{policy} \quad (2)$$

In equation (2) we see two terms. One is the maintenance risk and the other is the cost of the policy. The cost of the policy can stem from different factors, but the most important one will be the number of backups done. If we are only interested in finding out when we ought to do backup, but have already established how often to do so, the cost term will be irrelevant. If we want to determine the number of backups (and possibly the form of backup to be used), this term is crucial. We will take a short look at this term in a later section.

Accidents and catastrophes can happen unpredictably at any time. We shall assume that their arrival is a flat, homogeneous process. Changes to data also occur with a predictable pattern and the number of changes increases with time. Thus the longer the time elapsed without updating changes to data, the greater the chance of loss by accident or catastrophe. We can use this simple idea to define the idea of risk as a generalization of the familiar notion of Mean Time To Repair [24,25], namely a numerical value proportional to the expectation value of the Time To Repair, calculated over all objects in the maintenance set. Later on, we will take a look at such a risk function. Note that the form of the risk function in eq. (2) is more general than this and can describe models where maintenance risk is not proportional to the expectation value of the Time To Repair, as will be seen in section 6. Section 7 will however use this idea.

Why can we not simply keep perfectly up-to-date copies? Ideally, one can imagine a situation in which an original and backup were perfectly synchronized, by always copying every change to multiple copies (copy on write). This technology exists but it is not always practical or affordable from a resource point of view. The maintenance risk might be minimal, but the cost of such a policy might be great. Thus the total risk can not be expected to be minimal for such a policy.

We shall not argue about the reasons for use or non-use of mirroring. Rather we consider the effect of system behavior on decisions about a fixed regimen of batch-copying. This is the traditional understanding of a backup procedure in system administration.

3 Backup and random processes

As was described in the previous section, risk is described as expected loss. Since there might be factors that are out of our control, such as user behavior and system failures, we can not describe future events with certainty. Instead, we need to introduce such factors as random processes and the different relevant features of the future state might be seen as stochastic variables. Losses in the future will be a function of the system state, and will thus also be a stochastic variable. Our best guess at what the loss will be is it's expectation value.

An expectation value can be viewed as that value, which the mean of measurements of a stochastic variable approaches, when the number of measurement increases to infinity. When the variable takes only finite values, the expectation is the weighted sum of the different outcomes weighted by the probability of the outcomes, thus we write $EX = \sum_{i=1}^m x_i Pr(X = x_i)$ when the stochastic variable can take m different values, with the set of possible outcomes being $\{x_i\}$. When the outcomes are continuous, the sum becomes an integral and the probability becomes a probability density.

A real-world sample will always be of finite size, and will thus not be exactly equal to the expectation value. However, the mean value will be an unbiased estimate of that expectation. Thus when we do not know the expectation value, a sample mean can be a good guess on what to expect. For instance, if we want to know the expected user activity at Fridays, we might compute the mean activity on Fridays for the sample we have.

The maintenance theorem [23,26] was introduced as a way of describing the level of outstanding maintenance in a system, given a schedule for repair. In this paper we investigate the meaning of the maintenance theorem for risk management specifically in relation to disk backups. We will also take a look at some competing risk functions. One of the problems one faces here is in obtaining appropriate data to measure the changes taking place.

To quantify a model for backup, we should address a number of competing processes in the problem:

- Rates of user activity (leading to changes on disk)
- Rates of change or event arrivals on the disk subsystem (clustered or independent arrivals).
- Rate of change detection (often scheduled intervals)
- Rate of repair (e.g. capacity of the backup communication channel).

Since the first of these is more easy to measure than the latter; one can hope that the first two of these will be correlated.

A maintenance process, such as a backup, can be thought of as a queue [27,28] in which change ‘arrivals’ are processed and dispatched by a renewal process. In traditional queuing theory, arrival events are often assumed to follow the pattern of a Poisson arrival process, i.e. as a stream of random, independent arrivals. This assumption is made because it is simple and has special analytical properties. Poisson arrival models cope well with truly independent events; however, it is known from observation that many computer arrival processes are not Poisson processes, for a variety of reasons. Arrivals of events are often clustered, or come in ‘bursts’, exhibiting power law frequency distributions. Other data are inhomogeneous Poisson processes, i.e. they have time-varying means, and even time-varying variances. The only way to determine the correct type of model is to observe actual systems and collect data, at each new site.

3.1 User activity process

The arrival times of events at a computer system are the direct results of service requests or the execution of programs by users and can be measured. Such data reveal the patterns of system usage that lead to changes. It is natural to expect that the frequency of data changes would be correlated with user activity.

We refer here specifically to measurements made in ref. [29], in which arrival data from a number of international sources were examined.

The most basic observation we can make about user behavior is that it follows a basic daily rhythm [29]. For example, data from the measurements of user activity at Oslo University College, shown in fig. 1, we see a distinct maximum in user processes around midday and a lull at around 5:00 or 6:00 in the morning. In addition there is a weekly rhythm that typically peaks around Tuesday and lulls on Saturday. While the details of this profile will change from site to site, its essence is the same.

Clearly user behavior must be correlated somehow with changes to disk files (see fig. 2), since there is a direct causation involved. The only exception could be scheduled batch jobs, which add a constant signal. Even processes that are driven by remote users, through network services, are mediated by processes running on the local host. A knowledge of the patterns of user activity is thus necessary.

The data in fig. 1 and fig. 2 show that the arrival process is basically inhomogeneous: it exhibits a periodic trend.

Before delving further into the subject, a few definitions might be in order;

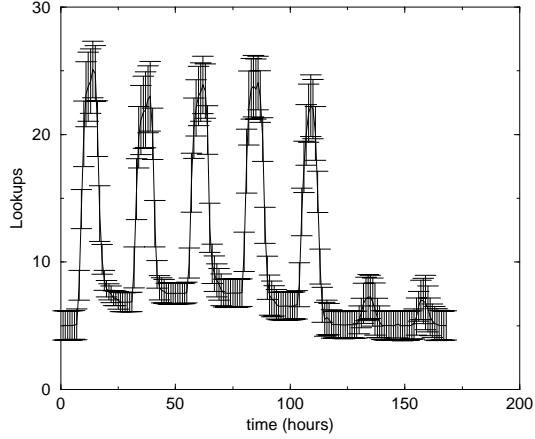


Fig. 1. The weekly average of hostname database lookups from Windows computers. The peaks occur at about midday of each weekday.

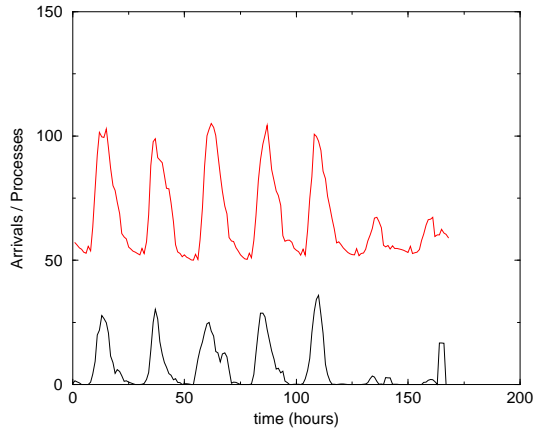


Fig. 2. A heuristic comparison between arrivals at a given time of week and the corresponding numbers of processes on the same host. These graphs exhibit the same general pattern. Correlations can be used to identify the probable dominant cause of the arrival process.

- $Q(t)$ is the random process of arrivals as a function of time. Since future events of this type are unknown, we will treat this as a stochastic variable. Ideally, this process should give the number of files updated at a completely specific time (where one nanosecond is treated differently as another). So, in order for a time integral to yield the number of files updated, this variable needs to be the number of files updated at time t times a delta function centered at t . In order to treat this in a practical manner, we may think of this variable as the number of files updated in a small interval, such as using a time resolution of one second. Thus $Q(t)$ can be thought of as the actual number of files updated on a given second starting at t and ending at $t + 1s$.
- $N(t_0, t_1) = \int_{t_0}^{t_1} Q(t)dt$ is the actual accumulated change in the time interval $[t_0, t_1]$. For the actual data, this will thus be a set of stepwise increments.
- $\eta(t_1, t_0) = EN(t_1, t_0)$ is the expected accumulated change in the same in-

terval.

- $q(t) = EQ(t)$ is the expected number of arrivals, and can be estimated using a statistical model for the random process $q(t)$ or simply by doing a sample mean. In this section we will take a look at possible models for $q(t)$. Note that since we usually do not have a fixed probability for an update at a given specific time, but rather for a given time interval, the expectation of $Q(t)$ will be modeled here as a continuous function.

When the starting time t_0 is given, typically $t_0 = 0$, we will simply write $N(t)$ and $\eta(t)$.

3.2 Inter-arrival times

The distribution of inter-arrival times describes the nature of the maintenance queue.

In a Poisson arrival process, the probability that the number of changes, which have arrived within time t , lies in the class $N(0, t) = n$ is:

$$P(N(0, t) = n) = \frac{(\lambda t)^n}{n!} \exp(-\lambda t). \quad (3)$$

In general, the inhomogeneous arrival rate $\lambda(t)$ is a property of the locally steady-state process, though here we have assumed a fixed rate, λ .

To test whether a Poisson process is suitable to characterize disk changes, we measured the distribution of 48,000 last-change events on disk filesystems belonging to a well-used computer disk at Oslo University College. The inter-arrival time distribution exhibited a highly clustered distribution of data, with an apparent long tail (see fig. 3).

To check whether a parameterized Poisson model in fact could describe this, the 48,000 measured change events were then modeled using a parameterized, inhomogeneous Poisson arrival process, modulated by hourly, daily, weekly and monthly factors. This model was not successful, however. It showed huge deviations that could only indicate a fundamental problem with the Poisson assumption.

The *deviance* of such a model measures the number of parameters required to model the arrival process, compared to a fully saturated model, i.e. one that has as many parameters as there are measurements. We found that there is essentially no reduction in the number of parameters needed, using a Poisson arrival assumption. This is clear evidence that the Poisson model is fundamentally incorrect for disk change in general.

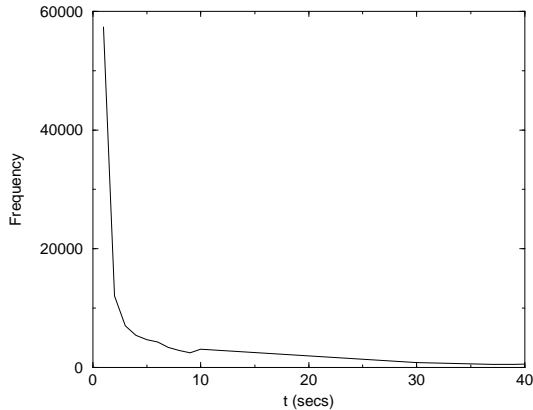


Fig. 3. A sketch of the distribution of inter-arrival times for file changes. A log-log plot has a gradient of about $3/2$, hence the power law in eq. (4)

What about other models? Recently much attention has been given to so-called self-similar processes that exhibit power-law distributions [30,31], or generalized Lévy processes [26,32]. Since most changes occur closely together, within seconds, this short time part of the spectrum dominates by several orders of magnitude over the longer times. However, there is also a non-zero probability of measuring inter-arrival times that are minutes and hours and even days.

The 48,000 measurements showed a power law type frequency-distribution of low order for the arrival process. An approximate fit gives a distribution of the inter-arrival times on the following form:

$$f_{ia}(\Delta t) = A \Delta t^{-\frac{3}{2}}, \quad (4)$$

where A is a constant amplitude. One could add an exponential term to make this normalizable, but we do not require this here. We can note that many phenomena that are driven by social networks are described by power laws (see, for instance, [33] for a discussion of this).

What this distribution tells us is that, if any changes at all take place, they are likely to occur in rapid bursts, i.e. ‘any’ means ‘many’. Thus, in terms of risk, if one loses any data, one stands to lose a lot. Thus it becomes important to minimize the time to backup.

We note, however, that relative time-scales are important here. Arrivals occur in milli-second bursts, while backups are typically carried out over intervals of many hours. Seen in this light, the detailed type of arrival process becomes of marginal relevance to the risk computation. The hourly inhomogeneity (trend) has dominant importance. Within each inhomogeneous period the distribution of events is fairly even, on average and simple assumptions, like maximum entropy models, are sufficient to approximate them.

3.3 Periodic change detection and updating

The maintenance theorem of ref. [23] tells us that a regular maintenance of the system can lead to an average stability if it is judiciously chosen. We therefore need to ascertain what the appropriate maintenance regimen should be.

The magnitude of fluctuations in the arrival process is key to deciding what is possible here. How should we deal with these changes? Several questions occur to us here:

- (1) When should a detector start scanning the filesystem for change, over the periodic (e.g. daily) cycle?
- (2) What is the risk associated with waiting once a change is made, or having a dead time between backup processes?
- (3) What is the cost associated with the backup process itself?

If one does not deal with changes immediately upon arrival, one has the problem of finding the changes again amongst the rest of the data. In order to detect changes, without immediate notification by the filesystem, we need a detection process, or some form of spatial disk scan. Like the temporal arrival process, this requires a model for coordinatization of space (locations on a disk or filesystem) and time. The detection process scans through the system, either using its hierarchical file structure, or using its disk block structure to detect change. There are two strategies one might use here:

- Detect and update changes to disk blocks or sectors. Programs like `dump` or `dd` use this approach.
- Detect and update changes to entire files. A file might consist of many sectors, but for the purposes of backup, it is regarded as a single entity. Programs like `tar` or `cfengine` use this approach.

The strategies for scheduling these traversals in time have two extremes:

- Backups parse file tree as quickly as possible (spans shortest possible time)
- Backups parse file tree slowly (spans a large interval of time, several runs do not overlap), i.e. as a “nice” process.

Finally, one must pick a time to start the scan.

In the first case above, the backup process presents the shortest interruption to system resource availability but with high load (backup can be a disk and CPU intensive process that disrupts system performance for users). In the latter case, one presents a low load to the system over a longer interval of time. The extreme case here would be to have a continuous scan of the file tree, picking up modified files continuously and backup them up as a low priority process.

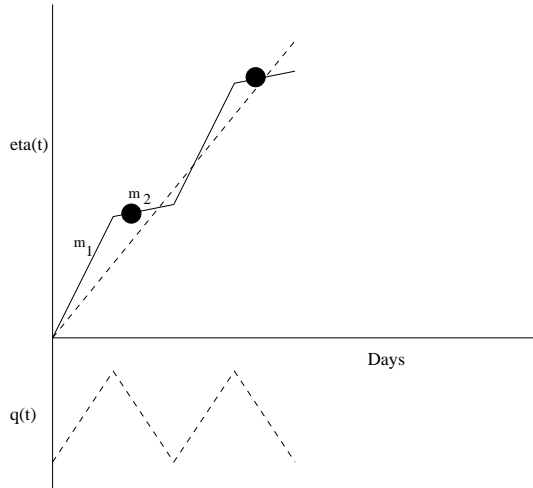


Fig. 4. A simplified schematic plot of a daily data rate $q(t)$ and cumulative change $\eta(t)$ of data on a user disk. The straight dotted line shows the rate of continuous detection and transfer required to backup data change as a continuous process. The dark blobs indicate short burst backups.

To decide between these two strategies one require a quantitative policy.

In the first case, one takes a rapid sharply focused snapshot of the filesystem. In the latter case, we take a blurred snapshot capturing more changes over a longer time. It is not obvious which of these strategies can capture most changes over a shortest ‘risk interval’. Clearly there is a choice to be made here: this choice must either be made as an *ad hoc* policy decision, or by optimization of some criterion, e.g. by minimizing this risk of loss of data in the system. We wish to explore the optimal solution.

4 How policy can reduce risk

Let us examine more closely the principle with which we might evaluate these strategies in terms of data capture rates. If a backup is to succeed, then its performance must be such that its average capture rate is equal to the average rate at which changes are made to the file system. The expected cumulative change to the file system at rate $q(t)$, over an interval Δt is (see fig. 4):

$$\eta(\Delta t) = \int_0^{\Delta t} q(t) dt. \quad (5)$$

Note the regions of differing rates during on and off-peak times, m_1 and m_2 , where $m_1 > m_2$. In region m_1 there is high risk of losing changes: there is m_1/m_2 times the risk in region m_2 . The likelihood of disruption by taking

backup, on the other hand is greater in region m_1 , so we might try to even out this disruption by capturing at a rate not exceeding m_2/m_1 of the total rate of change. The simplest solution, when these are in balance, is thus when $m_1 = m_2$ and we have the constant rate of capture with dotted line, in fig. 4.

Now, if we want to perform a quick non-disruptive backup, then this diagram tells us to perform the backup at the positions marked by the dark blobs, since these lie at the end of the period of maximum change. By placing the backup after the maximum rate period, we assume capture of the largest number of changes since the last backup, while at the same time minimizing the disruption.

It is almost obvious from the diagram that, in this simplified picture, there is no advantage to choosing any special time for a short-burst daily backup except to minimize disruption. We do not capture any greater number of changes in this way, on average. However, if we consider making more than one backup per day, or using a continuous scan technique, there are advantages to picking the time more carefully and a more detailed analysis is required.

If we change any of these simplifying assumptions to make the model more realistic, some of the details will change, but the essence will most likely remain somewhere between continuous transfer and short burst rates. Let us now be more quantitative.

5 The general risk function

The risk function defines what we mean by an 'optimal' strategy. Thus, a certain care will have to be used when we want to specify this function. The task is to translate rather vague and somewhat subjective wants and needs into a mathematical function.

The risk is described by a function that computes the expected loss for each possible strategy. The strategy space might have both discrete and continuous elements. In the context of doing backups, the strategy space over a given time interval (say a month) is described by the number of backups done in this time interval and the time when each backup should be done. There could be confounding factors found in the cost term in the risk function, such as deciding between full backup and backup of the last change. See eqs. (2) and (8). Such complications will not be investigated in later chapters, though such factors can be covered by the formalism in this section.

The risk function must necessarily have at least two components. The first component is the reason why one does backups at all, namely the risk of

losing valuable files. The expected loss of files for a given backup strategy will thus be at the core of this component of the risk. The 'loss of file' part of the risk function does not need to depend linearly on the expected number of files lost during a time interval. In fact any function that increases monotonically with the expected loss of files is a candidate. However, we might take the 'production' and 'loss' of files as the 'economy' of this system. In this case, we can set the 'loss of file' part of the risk function as linearly dependent on the expected number of files lost for a given strategy, with the slope equal to unity.

In order to compare cost and maintenance risk on the same footing, we would need a common economy for the expected loss. Since we in most applications are interested in the number of files lost, this might be the most appropriate economy, and the cost of backup, originally given in money and work, would have to be translated to the file economy using a proportionality constant.

The cost part of the risk function is needed in order to 'punish' backup strategies that run the backups too frequently. The reason we need it is to avoid unreasonable solutions, and the reason why this is deemed so is that there is a cost to doing backups. First of all, there is a starting cost in the hardware and competence needed to do backups. We will assume that we are dealing with a system where the need for doing backups actually justifies this starting cost. Thus we will ignore the starting cost. A constant term in the loss function will not modify the optimum of the risk function.

The cost of doing backups might be expected to increase with the number of backups done. There are several reasons for this. First off, the more often one does backup the more hardware in the form of for instance tapes and tape machines one needs. This cost comes in the form of money. Secondly, if one doubles the number of backups one might also double the effort of the administrator(s) in running the backup system. Thus the administrator(s) will have less time for other work. This results in a loss of productivity for the system administrator(s), both costing money and loss of file production. Thirdly, running backups slows the system, making it harder for the users to produce the files that one wants backup of. This comes in the form of loss of file production.

In introducing the first component of the risk function, we noted that it might be least difficult to use the loss of files (or loss of file production) as the 'economic' penalty. Thus the monetary loss described here needs to be translated into the worth of an average file. If one has an idea of how much each file in the system is worth (on average) one can just invert this relation. If the produced files is worth nothing than obviously there would be no need to do backups.

Note that any given backup can jeopardize files arriving at the time of backup, as described in the previous section. Whether this is seen as a price of the policy or as an independent file loss is a matter of taste, but in the following form, we will take this into account as a separate factor and assume that the cost of a given policy can be specified exactly.

Thus we can write (2) for a given time interval between backup done at time t_0 and backup done at time t_1 .

$$R_{policy}(t_0, t_1) = C_{policy}(t_0, t_1) + E(\text{files lost in } [t_0, t_1]) + E(\text{files lost during backup at } t_1) \quad (6)$$

Note that when t_0 and t_1 are fixed, there might not be much more of the policy to be specified. In later sections, this will be assumed. Thus this equation is mainly interesting as a step toward a risk function for a given policy over time.

If we want to examine periodic backup routines such as daily backups, the cost function is irrelevant and we need only concentrate on the last two terms. When the backup routine does not significantly raise the probability of loss, the two terms can often be joined together, as seen in section 7.

Ignoring the last term, or joining it with the middle term, one can take a closer look at eq. (6) by dividing the interval into single specific times;

$$R_{policy}(t_0, t_1) = C_{policy}(t_0, t_1) + \int_{t_0}^{t_1} E(Q(t)I(\text{arrival at time } t \text{ lost between } t \text{ and } t_1))dt \quad (7)$$

where I is the indicator function, defined as one if the statement is true and zero otherwise.

When one wants a backup policy for a general time interval, where more than one backup might be considered, the general form becomes;

$$R(policy) = C_{policy}(t_0, t_1, t_2, \dots, t_n) + \sum_{i=1}^n \int_{t=t_{i-1}}^{t_i} E(Q(t)I(\text{arrival at time } t \text{ lost between } t \text{ and } t_i))dt \quad (8)$$

where t_i is the time of the i th backup and t_0 is the time when the backup policy was initialized. Note that the number of backups, n , will also form part

of the backup policy. Also note that if file loss during backup is important and can not be incorporated in the second term of this equation, then one would rather use eq. (6) to form an equation similar to eq. (8), only with one extra term.

This rather complicated formula might be rather difficult to optimize in general. The cost function of the policy is on a very general form. In addition the formula demands the expectation of the interplay between file arrivals at a given time and file loss at a time interval following that. Making such an expectation from sample data might be near to impossible.

Thus we will in the next section take a look at minimizing risk for a given set of extra assumptions, such that sample data can likely be used to form the necessary expectations.

6 Model 1: Finding a backup policy for a specific, ideal, risk function

6.1 The risk function for backup strategies

In this section we will take a look at using the risk function in a toy model where we can find an exact expression for the update time, given the expected number of file changes over time. We want to do a full analysis of the problem of finding an optimal strategy, that is one wants both the optimal number of backups and the optimal time of each backup over a given time interval. Given a large time interval one would thus gain both the overall backup frequency and the frequency for specific parts of the interval. A couple of assumptions will be specified, which are more or less realistic. Thus this might best be seen as an example rather than a continuation of the previous section.

The time interval for which one wants to make a backup strategy can be specified by the start time, t_0 , and the end time $t_0 + T$. The first thing we need to do when describing a strategy is to find the optimal number of backups that ought to be done in the time interval, n . This is the discrete part of the strategy specification. For n backups we can choose freely when to do each backup. We will denote the backup times with $t_1, t_2, \dots, t_n = t_0 + T$, where $t_{i-1} < t_i$ for $1 \leq i \leq n$. We will also need the time between one backup and the next; $\Delta t_i = t_i - t_{i-1}$. We have here assumed that the run time of each backup is small in terms of the time between backups. Thus we have ignored the last term in eq. (6) and used (7) to form a risk function of the form found in eq. (8).

When handling the cost component of the risk function, the easiest assumption one can make is that the cost of backup increases linearly with the number of backups done. This assumption can be disputed, since if one does incremental backup, one needs more capacity if it has been a long time since last backup than if another backup has been recently done. However the time spent by an administrator on doing backups need not increase with the backup size and might easily dominate the backup cost. This complication will thus be disregarded in this section. Under such assumptions, the backup policy is fully specified by the number of updates n and the backup times t_1, \dots, t_n .

There is also a fixed price for being able to do backup, such as the infrastructure to do the backup. This one time investment is assumed done already, since we do not want to discuss whether backups should be done at all, here, but rather how often and when. Thus we can say that the cost is proportional to the number of backups done in a given time interval. The cost component of the risk function thus comes in the form of;

$$C(n, t_0, t_1, \dots, t_n) = Cn. \quad (9)$$

Here C is equal to the loss of file production due to an extra backup plus the money needed to do an to an extra backup translated into file loss. Any increasing function of n might do the trick to represent the cost of doing backups, but we hope this simple form of the cost of doing backups proves sufficient for our purposes. The total risk function describing the expected loss over a time interval thus becomes;

$$R(n, t_0, t_1, \dots, t_n) = E(\text{file loss}) + Cn \quad (10)$$

where n is the number of backups in the time interval $(t_0, t_n) = (t_0 + T)$ and t_1, \dots, t_n is the time of each backup.

6.2 *A more detailed look at the maintenance risk function*

The maintenance component of the risk function might be decomposed into the expected loss between each backup, see (8). We will here assume that the actual file updating intensity, $Q(t)$ is independent of the the event of losing a file update later on, given the expected updating intensity, $q(t)$. Thus both the file loss event and $Q(t)$ can be modeled using the expected updating intensity, but all stochastic activity in the two variables are in all other respects independent. When the chance of losing a file is proportional to the time to next backup, such as when using Mean Time To Repair [24,25], this is an

implicit assumption. In this section, we will assume that the probability of time loss in t_0 to t_1 is proportional to $N(t_0, t_1)$.

In such a setting, the conditional independence might be less appropriate: it might be more natural to consider file loss arising from actual file updating activity, rather than expected file update activity. We will not try to handle that problem here, but simply assume that such an approach makes sense in some circumstances, say where file loss is not directly caused by file activity but rather by some process having to do with user activity which can be estimated by the expected file activity.

Note also that a probability can not be proportional to a real number in general, as probability has an upper bound. We will here further assume that the proportionality constant is so small that the probability for file loss in a given backup cycle never is larger than say 10% for any realistic file activity. This approximation should not be any great assumption, as in most realistic cases, the probability of file loss should be less than 10%.

Using conditional independence we get

$$R(n, t_1, \dots, t_n) = Cn + \sum_{i=1}^n \int_{t_{i-1}}^{t_i} q(t) E(I(\text{file loss in } t \text{ to } t_i)) dt \quad (11)$$

Since the expectation of an indicator is a probability and since we can decompose the event of file loss in $[t, t_i]$ to an interval over file loss for an infinitesimal time interval, which we assumed was proportional to $q(t)$ with a proportionality constant we can call γ , we get;

$$\begin{aligned} R(n, t_1, \dots, t_n) &= Cn + \gamma \sum_{i=1}^n \int_{t_{i-1}}^{t_i} q(t) \int_t^{t_i} q(t') dt' dt = \\ &= Cn + \gamma \sum_{i=1}^n \int_{t_{i-1}}^{t_i} q(t) \eta(t, t_i) dt = \\ &= Cn + \gamma \sum_{i=1}^n \eta(t_{i-1}, t_i)^2 - \sum_{i=1}^n \int_{t_{i-1}}^{t_i} q(t) \eta(t_{i-1}, t) dt \end{aligned} \quad (12)$$

Here the proportionality constant γ can be estimated by looking at the historic need for file backups versus the square file activity at the same time. A simple linear regression between file backup need per file and square file activity can yield such an estimate.

If we can assume that the last factor in eq. (12) either is negligible or if $\int_{t_{i-1}}^{t_i} q(t)\eta(t_{i-1}, t)dt$ is proportional to $\eta(t_{i-1}, t_i)^2$, then the expression can be further simplified. The proportionality above is open to some question, but for $q(t)$ constant over a fixed time and zero otherwise, it can be seen to be true with a proportionality constant of $1/2$. For situations close to that, the assumption can be expected to be approximately correct. Thus we get;

$$R(n, t_1, \dots, t_n) = Cn + \gamma \sum_{i=1}^n \eta(t_{i-1}, t_i)^2 \quad (13)$$

where we have absorbed the above mentioned proportionality assumption into the constant γ .

Thus we have four assumptions working here;

- (1) Conditional independence between $Q(T)$ and the event of file loss given $q(t)$.
- (2) $\text{Cost}(\text{policy}) \propto n$.
- (3) $P(\text{file loss at time } t) \propto q(t)$.
- (4) $\int_{t_{i-1}}^{t_i} q(t)\eta(t_{i-1}, t)dt \propto \eta(t_{i-1}, t_i)^2$ or $\int_{t_{i-1}}^{t_i} q(t)\eta(t_{i-1}, t)dt \ll \eta(t_{i-1}, t_i)^2$

In order to work with this expression we also need a sensible cost for doing backup in terms of lost files C . When γ and C have been estimated we can find the optimal backup strategy for this, admitting, rather simple minded model.

6.3 Optimizing strategy

The task of finding the minimal risk can be split in two. For a given number of updates, n , one can find the *optimal times* to do backup; $(t_1, \dots, t_{n-1}, t_n = t_0 + T)$. Given these backup times the risk for a given n is set. The second task is to minimize the risk in regard to the *number of backups*, n . The expression for the risk function given n can be minimized, given that

$$\begin{aligned} \sum_{i=1}^n N_{(t_{i-1}, t_i)} &= N_{(t_0, t_0+T)} \Rightarrow \\ \sum_{i=1}^n \eta(t_{i-1}, t_i) &= \eta(t_0, t_0 + T) \end{aligned} \quad (14)$$

Since $\eta(\cdot, \cdot)$ is assumed known for all input arguments, and since the start and end time is assumed known, then so is $\eta(t_0, t_0 + T) \equiv N_t$.

Before going to the risk function, it might be wise to recall that the time of backups can also be written using the time between backups, $\Delta t_i = t_i - t_{i-1}$, so that a given t_i can be written as $t_i = t_0 + \Delta t_1 + \Delta t_2 + \dots + \Delta t_i$. Using that t_0 is fixed, so that the dependency of t_0 can be assumed known, we can write $\eta(t_0, t_i) = \eta(t_i) = \eta(t_0 + \Delta t_1 + \Delta t_2 + \dots + \Delta t_i)$, where η having two parameters is the expected number of updates between two times and η having one parameter is the expected number of updates from t_0 to the given time. Thus the expected number of files updated between two backups becomes $\eta(t_{i-1}, t_i) = \eta(t_0 + \Delta t_1 + \Delta t_2 + \dots + \Delta t_i) - \eta(t_0 + \Delta t_1 + \Delta t_2 + \dots + \Delta t_{i-1})$. Note that since $\Delta t_1 + \Delta t_2 + \dots + \Delta t_n = T$, we do not need to specify the last time between backups, Δt_n .

Putting this parametrization into eq. (13), one gets;

$$\begin{aligned}
R(n, \Delta t_1, \dots, \Delta t_{n-1}) &= Cn + \\
2\gamma \sum_{i=1}^{n-1} [\eta(\Delta t_1 + \dots + \Delta t_i)^2 - \eta(\Delta t_1 + \dots + \Delta t_i)\eta(\Delta t_1 + \dots + \Delta t_{i-1})] &- \\
2\gamma N_t \eta(\Delta t_1 + \dots + \Delta t_{n-1}) + \gamma N_t^2 &\quad (15)
\end{aligned}$$

One can now find the derivative of the risk function for a given time between updates Δt_i for a fixed $i < n$;

$$\begin{aligned}
\frac{1}{2\gamma} \frac{\partial R(n, \Delta t_1, \dots, \Delta t_{n-1})}{\partial \Delta t_i} &= -\eta(\Delta t_1 + \dots + \Delta t_{i-1})\eta'(\Delta t_1 + \dots + \Delta t_i) + \\
&2\eta(\Delta t_1 + \dots + \Delta t_i)\eta'(\Delta t_1 + \dots + \Delta t_i) - \\
&\eta'(\Delta t_1 + \dots + \Delta t_i)\eta(\Delta t_1 + \dots + \Delta t_{i+1}) - \\
&\eta(\Delta t_1 + \dots + \Delta t_i)\eta'(\Delta t_1 + \dots + \Delta t_{i+1}) + \\
&2\eta(\Delta t_1 + \dots + \Delta t_{i+1})\eta'(\Delta t_1 + \dots + \Delta t_{i+1}) - \\
&\eta'(\Delta t_1 + \dots + \Delta t_{i+1})\eta(\Delta t_1 + \dots + \Delta t_{i+2}) - \\
&\eta(\Delta t_1 + \dots + \Delta t_{i+1})\eta'(\Delta t_1 + \dots + \Delta t_{i+2}) + \dots \\
&N_t \eta'(\Delta t_1 + \dots + \Delta t_{n-1}) = \\
(-\eta(\Delta t_1 + \dots + \Delta t_{i-1}) + 2\eta(\Delta t_1 + \dots + \Delta t_i) - \eta(\Delta t_1 + \dots + \Delta t_{i+1})) & \\
\eta'(\Delta t_1 + \dots + \Delta t_i) + \frac{1}{2\gamma} \frac{\partial R(n, t_1, \dots, t_n)}{\partial \Delta t_{i+1}} &\quad (16)
\end{aligned}$$

Since both partial derivatives should be zero for the policy that gives minimal risk, then so should the difference between two subsequent derivatives. There will remain three terms:

$$-\eta(\Delta t_1 + \dots + \Delta t_{i-1}) + 2\eta(\Delta t_1 + \dots + \Delta t_i) - \eta(\Delta t_1 + \dots + \Delta t_{i+1}) = 0 \quad (17)$$

Setting $a_i \equiv \eta(\Delta t_1 + \dots \Delta t_i)$ one thus gets the difference equation;

$$-a_{i-1} + 2a_i - a_{i+1} = 0 \tag{18}$$

In order to solve this homogeneous difference equation, one can find the characteristic polynomial, see [34]. In this case, this is $f(x) = (1-x)^2$. The solution to such an equation is $a_j = b + cj$. Setting $a_0 = \eta(0) = 0$ and $a_n = N_t$ yields $a_j = jN_T/n$, so that;

$$\eta(t_{i-1}, t_i) = \frac{N_t}{n} \tag{19}$$

The time of backups should thus be chosen so that the expected number of updates is equal in each time interval between backups. The policy rule found here is easy to interpret, and can be communicated to administrators with no mathematical training. We have verified, informally, that this scheme in fact agrees with the practices used by a number of system administrators.

The assumptions here were formulated so that an easy interpretation was likely. The result is valid for these assumptions, so the question becomes whether the assumptions are correct. The assumption of the probability of file loss being proportional to the file activity over a backup cycle is the assumption that might most easily be criticized in practical cases.

The result summed up in eq. (19) does not depend on the parameters C or γ . Thus if the total backup intensity, n , is fixed in a large time period, $[t_0, t_0 + T]$, then no estimation of such parameters is necessary. It is only for the sake of finding the backup intensity that these two parameters are needed.

Putting the solution for optimal backup times into eq. (13), one gets;

$$R(n) = Cn + \gamma \frac{N_t^2}{n} \tag{20}$$

where n now is the only part of the policy left to be determined. This function is convex for $n > 0$. This means that the minimal risk can be found simply by differentiation with respect to n and setting this derivative equal to zero. Thus one gets that

$$n = N_T \sqrt{\frac{\gamma}{C}}. \tag{21}$$

Thus the number of updates needed is proportional to the expected number of file updates in the time interval $(t_0, t_0 + T)$. It increases with increasing probabilities for the need for backup of each file, γ , and decreases with increasing cost of doing each backup, C .

This suggests that a strategy of copy-on-write is the optimal one, or alternatively a coarse-grained approximation to this.

6.4 *Two examples*

It is worth stressing that while the number of assumptions and the mathematical path for deriving the strategy in Model 1 is cumbersome, the strategy itself is easy to use. The following two examples will hopefully illustrate that point.

For our first example, assume that experience shows that on Mondays, Tuesdays, Wednesdays and Fridays, the file activity can be classified as normal. On Thursdays the file activity is twice the normal level, while on weekends it is half the normal level. It has been determined that the update frequency during normal load should be equal to one backup per day. Thus the total backup frequency must have already been established. Then the principle of having equal expected file activity between backups established in eq. (19) says that there should be two backups on Thursdays, none on Saturdays and one on Sundays.

The example here is cyclic with period one week. Thus it does not matter if we want a policy for the next year or for the next week as the policy for the next year will simply be a repeated process of the policy for a week. Note also that the principle of equal expected file activity does not determine when in the cycle one should start doing backups. As long as the expected file activity is equal between backups, the model does not distinguish between two different policies. Thus here, doing backups at midnight, plus midday for Thursday is neither better nor worse than doing the backups at 6 in the morning, plus six in the afternoon for Thursdays. If a model that differentiates between two such policies is wanted, then a policy built on other assumptions is needed, see for instance section 7.

For the second example, the backup intensity is also to be determined for the next ten weeks. Each backup costs \$100 in this case study and the loss of a file is estimated to cost \$0.5, thus $C = 200$. No large scale change of file activity is expected. The largest time interval where one have different expectations for different times of in the interval is one week in this example. Thus expectations are cyclic with a period of one week, such as in the first example. The expected file activity during different parts of the week is also

the same as before.

For example, local data show that the expected number of files updated in a week is $N_t = 10000$. Experience also shows that about 15 file updates are lost each Thursdays, the day with the highest activity, when no backup is done during that day. Thus $\gamma = 15/10000 * 2/7 = 0.00043$. Then the optimal number of updates per week is $n = 14.64$. Having a fixed number number of times to do backup per week makes this part of the administration more easy. It would also be nice to do the backups at the same time of the day for days with normal load. Thus $n = 14$ is better than $n = 15$. The policy then becomes two backups during normal load, four on Thursdays and one each for Saturday and Sunday.

7 Model 2: The windshield-wiper or snow-clearing model

7.1 Introduction

We turn next to a more realistic model, which is also more specialized. It will here be assumed that the backup intensity has already been determined, and that the expected number of updates is periodic. Instead of the chance of file update-loss being proportional to the change activity in the period, we will instead use the Maintenance Theorem and assume that the chance or risk of file update-loss is proportional to the time to the next backup. Note that also here, such a proportionality can only be an approximation for low loss probability, since if not, we could get probabilities above 1. We will not assume that the backup time is so small that loss within the backup operation can be neglected.

The resulting strategy will tell us when to do backup inside each repeated period (typically one day). As mentioned previously, Model 1 could not differentiate between to strategies with equal backup intensity when the file arrivals were periodic. As long as the expected number of updates are the same, the risk is the same for the loss function used there. Here we will find that the time of backup is important, and a few examples will show how the optimal backup time can be found. It should however be noted that the approach used here requires more mathematics in daily use, as well as being more restricted in the assumption of period file arrivals and fixed backup intensity equal to the time interval of the period.

7.2 Motivation

The process of periodic maintenance, counteracting a random arrival process, makes one think of the process of clearing raindrops from a car windshield, or snow from the roads. Random arrivals can occur anywhere on the system (i.e. the windshield) but we must sweep the system in a sequential manner. The rain drops do not stop while we are wiping, so there is always rain to be cleared. What then is the best strategy?

Unlike rain, which can run directly down a drain (like copy on write), snow builds up and requires a special clearing effort, just like a disk that accumulates change. First instinct is to wait for the snow to stop and then clear it, but what if the snow does not stop? Alternatively, suppose that the snow falls only during the night; if we wait until the snow stops, then the amount of time the roads are clear for traffic is reduced because we use up that valuable time on clearing the snow. Perhaps it would be better to start clearing the snow while it is still snowing, so that we are finished just as the snow stops? How can we find out the optimal strategy?

Once again, there are two issues: when and where the snow falls. Sweeping is a space-time process, while the arrival is generally evenly spread. We do not have any data at the level of disk sectors to gauge how evenly spread change is across the sectors of disks during normal usage: this is an empirical study that remains for the future. However, we can still make an estimate.

If we assume a maximum entropy arrival process in space (the location of changes on a disk) so that changes are evenly spread (like snow) and continuous scanning with respect to the system locations, like a windshield-wiper or snow plough, then there are always just as many new arrivals in front of the windshield wiper as there are behind it. There is no advantage to starting at any particular place. Simply scanning from end to end of the system with a clearing process is enough.

However, scheduling in time is important. If we set the clearing process on ‘intermittent wipe’, while the rain or snow falls constantly, then there might be *more* outstanding raindrops in front of the wiper blade than behind it in a single sweep. The discrepancy will be proportional to probability of arrivals during the time elapsed between the backups.

Consider the diagram in fig. 5. We consider a single daily period of the arrival process $q(t)$ and it will be understood that this pattern is repeated over many similar occurrences. The expected number of arrivals as a function of time, $q(t)$, is measured in change quanta (sectors etc), so that all changes are of the same size, and that a backup is centered on some time t_0 during the period. Our aim is to minimize the risk as a function of t_0 , i.e. determine the optimal

time for starting the backup $t_0 - \frac{1}{2}t_b$. Note that the risk of loss is entirely placed at the source of the changes; if we want to imagine that parts of the maintenance process are also capable of introducing risk of loss this must be handled with an expanded model. We shall omit this complication for the present paper [35].

7.3 Risk functions

Let us define the time it takes for a backup to complete to be t_b so that, if changes can be copied at a rate $r(t_0)$ at time t_0 then we have that the expected backup time is;

$$t_b(t_0) = \frac{\int_0^P q(t)dt}{r(t_0)}. \quad (22)$$

where P is the time interval of the period. Note that we are allowing for the fact that the system's ability to copy changes or to repair itself might be altered by a heavy load. In other words, the maintenance process might itself inflict a heavy load on the system.

Because we have periodicity and one backup per period, we do not have to study several backups such as is done in eq. (7) but study the single backup case of eq. (6) instead. Also, we might drop the backup cost term, since the backup intensity has already been established. The assumption here is that the probability of file loss is independent of the actual ($Q(t)$) or expected ($q(t)$) arrival intensity, but is rather a linear function of the time between arrival and the backup of that specific arrival. This can not be the case for arbitrary large times between arrival and backup, but we use it here under the assumption that the change of file loss is low for any specific realistic file arrival. Thus $E(Q(t)I(\text{arrival at time } t \text{ lost between } t \text{ and } t_1)) = q(t)t_{\text{wait}}(t)$ and we get the following risk function:

$$R(t_0, t_b) = R_0 \int_0^P q(t)t_{\text{wait}}(t)dt. \quad (23)$$

We would like to minimize this risk, given that a single scan of the backup will be performed, centered on time t_0 .

To evaluate this risk, we split up the daily period into three regions: before the backup, during the backup and after the backup (see fig. 5). Note that we if we

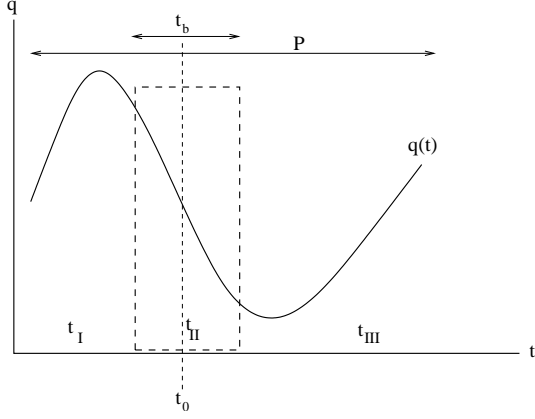


Fig. 5. The windshield wiper model scans across files in a time t_b , such that the average time that a file has to wait for backup is $E_Q t$. There are three regions of currently unknown sizes: region I before backup, region II during backup and region III after backup. The file change process $q(t)$ brings q arrivals at time t and provides a probabilistic weight to the expected risk: if more files are arriving, the risk of loss will be higher.

should be painstakingly precise, the actual time spent on backup will depend on the actual file arrival rate $Q(t)$ and that it thus would be quite difficult to separate the regions. In order to make the model tractable, we assume that the regions can be separated according to the expected time used on backup, $t_b(t_0)$.

For simplicity we shall assume, initially, that the backup rate is constant to avoid unnecessary complication. In a regular backup or maintenance scenario, this need not be far from the truth. The t_{wait} function is now defined piecewise over the three regions:

- The average time to the next backup, in region I, is simply the distance between now and the average backup time t_0 (we do not know exactly when the backup of a specific object will be made within the region II), i.e. $t_{\text{wait-I}} = (t_0 - t)$.
- In region II, half of the files that arrive will be backed up during the backup while another half will have missed the boat and will have to wait a full period P to be picked up. The average time to backup is thus the average of P and the same as region I, i.e. $t_{\text{wait-II}} = \frac{1}{2}P + t_0 - t$.
- In region III, all of the files have to wait until the next backup, and apart from the extra distance, they satisfy the same average expression as in region I, i.e. $t_{\text{wait-III}} = (P + t_0 - t)$.

One can discuss the precise formulation of these expressions, but the result turns out to be not strongly dependent on the details. The crucial factors to risk are the competition between the arrivals $q(t)$ and the linear, periodic distance function. Thus we have:

$$\begin{aligned}
R(t_0, t_b) = & R_0 \int_0^{t_0 - \frac{1}{2}t_b} q(t)(t_0 - t)dt + R_0 \int_{t_0 - \frac{1}{2}t_b}^{t_0 + \frac{1}{2}t_b} q(t)\left(\frac{1}{2}P + t_0 - t\right)dt + \\
& + R_0 \int_{t_0 + \frac{1}{2}t_b}^P q(t)(P + t_0 - t)dt = \\
& R_0 \left[t_0 \int_0^P q(t)dt - \int_0^P tq(t)dt + \frac{1}{2}P \int_{t_0 - \frac{1}{2}t_b}^{t_0 + \frac{1}{2}t_b} q(t)dt + P \int_{t_0 + \frac{1}{2}t_b}^P q(t)dt \right] \quad (24)
\end{aligned}$$

Introducing previously used notation of $\eta(t)$ which is the accumulated arrivals from time 0, we can express this as;

$$R(t_0, t_b) = R_0 \left[t_0 \eta(P) - \frac{1}{2}P \left(\eta\left(t_0 + \frac{1}{2}t_b\right) + \eta\left(t_0 - \frac{1}{2}t_b\right) \right) + K \right] \quad (25)$$

where K is independent of t_0 and thus a constant. We then get that the risk is minimal if;

$$\frac{1}{2} \left(\eta'\left(t_0 + \frac{1}{2}t_b\right) + \eta'\left(t_0 - \frac{1}{2}t_b\right) \right) = \frac{\eta(P)}{P} \quad (26)$$

and $\eta''(t_0) = q'(t_0) < 0$. For a given estimate for the expected arrival intensity, the accumulated intensity can be made through a simple integration or numerical summation. If one has an analytic expression for $\eta(t)$ then in some cases one can solve eq. (26) analytically. If not, the equation can be solved through numerical methods such as Newton's method or by analyzing a plot of eq. (25).

If one can assume $t_b \ll t_0$ one can simplify eq. (26) to

$$q(t_0) = \eta'(t_0) = \frac{\eta(P)}{P} \quad (27)$$

This can be solved graphically by plotting $\eta(t)$ versus a line connecting $(0, 0)$ with $(P, \eta(P))$ and finding the points where $\eta(t)$ is parallel to the line and curves downward, see figure 6.

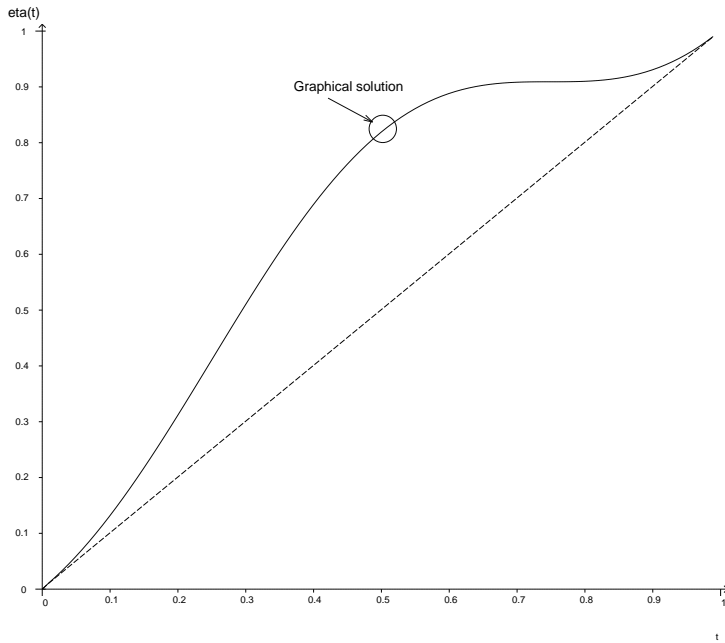


Fig. 6. A plot of an accumulated arrival function (that used in example 1) along with the graphical solution to the optimal strategy for t_0 .

7.4 Example 1: A simple arrival function

We must now specify the expected arrivals distribution over the period. This can be measured, as in fig. 2, or we can approximate it by a simple function that captures the main features of the process. For this, we choose the trial function

$$\begin{aligned}
 q(t) &\rightarrow q_0 \left(1 + \sin \left(\frac{2\pi t}{P} \right) \right) \rightarrow \\
 \eta(t) &= q_0 \left(t + \frac{P}{2\pi} \left(1 - \cos \left(\frac{2\pi t}{P} \right) \right) \right)
 \end{aligned} \tag{28}$$

This trial function is crude but exhibits the main features of the actual measurements over a single cycle. Since we treat only a single cycle, it is assumed that the distribution is itself an expectation value of behavior over many cycles, but again we need not concern ourselves with anything but the qualitative features here, since the conclusions are fairly robust.

The function $R(t_0, t_b)$ is a function of the midpoint of the backup and its length, in this parameterization, but we control only $t_0 - \frac{1}{2}t_b$ in practice, or t_0 implicitly. We can plot the function as a surface (see fig. 8).

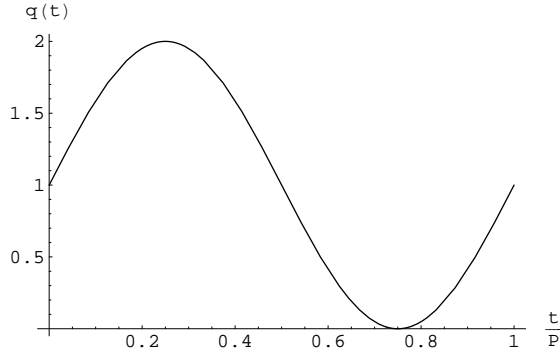


Fig. 7. A plot of the approximate arrivals function. Axis gradations are measured in dimensionless units of P .

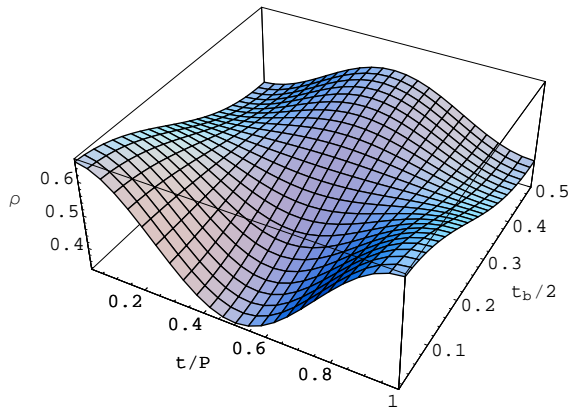


Fig. 8. A surface plot of the risk function as a function of t_0 and $\frac{1}{2}t_b$ axis gradations are measured in dimensionless units of P . For small t_b , it has a clear minimum at $t_0 = P/2$. Note that this is *not* the time of minimum activity.

The result shows a very robust conclusion: namely that, for backups that take a relatively short time compared to the cycle time (which is the usual case – perhaps a few hours out of the day), the time of least risk occurs when one centers the backup around $t_0 = P/2$, i.e. at the time of changeover from maximum to minimum of activity of the system (compare to fig. 6 and 7). This can easily be seen from eq. (27), which yields $\sin(2\pi t_0/P) = 0$ with \sin going from positive to negative (negative curvature for $\eta(t_0)$). Thus $t_0 = \frac{1}{2}P$. Note that if the more general expression $q(t) = A + B\sin(2\pi(x - \phi)/P)$ was used, this would only serve to adjust t_0 to $\frac{1}{2}P + \phi$. Note also that since $q(\frac{1}{2}P - \Delta t) - 1 = -(q(\frac{1}{2}P + \Delta t) - 1)$, the result $t_0 = \frac{1}{2}P$ will hold even when $t_b > 0$, see eq. (26), as long as the solution is not transformed from a minimum of the risk to a maximum, see fig. 8.

It is a fortunate coincidence that this happens also to be the time at which the system generally begins to have most resources to give to the problem, but the conclusion does not depend on this fact. We have not used the backup rate $r(t_0)$ explicitly here. Choosing t_b to be small simply means that there are

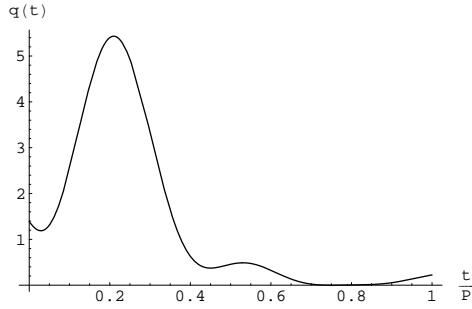


Fig. 9. A plot of a more realistic but still approximate arrivals function. Axis gradations are measured in dimensionless units of P .

sufficient resources to copy the changed data in a short interval of time.

Consider an example of this. Suppose we have a backup that takes two hours to complete, and we know that the maximum activity occurs at 12:00 noon each day. Then the minimal risk occurs if we start the backup at around $t_0 - \frac{1}{2}t_b = 18:00 - \frac{1}{2} \times 2$, i.e. at 17:00 hours.

As we increase the size duration of the backup $t_b \rightarrow P$, the conclusion becomes unstable at about $t_b \sim P/4$. In other words, if a backup takes more than about a quarter of a day to complete (six hours), then there is no particular advantage to starting the backup at any time, assuming that copying resources are constant and available. As we approach a full day $t_b \rightarrow P$ (continuous and very slow windshield-wipe operation, as opposed to a fast intermittent wipe), it even seems advantageous to start closer to the start of the active region, at around 6:00 a.m. in the extreme case. This is because one has as good a chance of picking up changes as they occur as one has by waiting until after they have already happened.

7.5 Example 2: A more realistic $q(t)$

The simple sinusoidal function is not a good representation of a real system. The actual arrivals function in fig. 2 peaks at 12, low point around 6am, with a small dip in the curve. This form can be simulated by the scaled function:

$$q(t) = \left(\frac{3}{2} + \sin \left(5\pi t - \frac{2}{3}\pi \right) \right) (1 + \sin(2\pi t)) e^{-\pi(t-1/4)} \quad (29)$$

The advantage of this form is that it resembles the true function and that it is still simple enough to submit to analysis. On the time scale of the figure the arrivals function approximates the measured values in fig. 2. 0 and 1 correspond to 06:00 hours, 0.25 corresponds to 12:00, 0.5 is 18:00 hours and

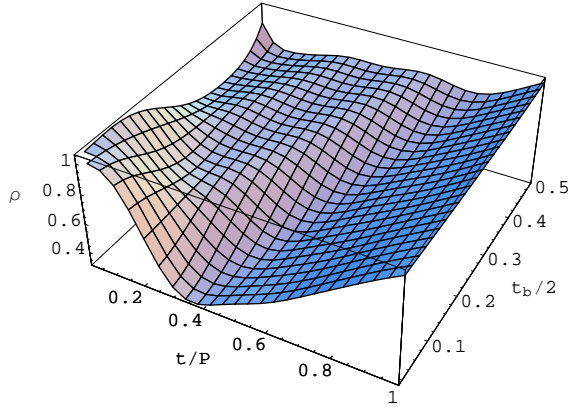


Fig. 10. A plot of the risk surface. Axis gradations are measured in dimensionless units of P .

0.75 is 00:00 or 24:00.

We repeat the same analysis with this new function and obtain the risk surface shown in fig. 10. As one might expect the qualitative features are the same as in the simple test function, but the optimal low-risk point has moved slightly closer to the peak of the arrivals function.

Using computer algebra and numerics, we find a minimum for small t_b to be at about $0.32P$, which is in the afternoon just before 14:00 hours. As the length of t_b increases, the minimum risk point moves almost linearly toward midnight. When $t_b = 0.3$, i.e. sequential backup takes as much as six hours, then the optimal time t_0 moves to about $20 : 00 - 6 : 00/2 = 17 : 00$ hours..

The introduction of this asymmetry reveals the reason for the result of the simple test function in section 7.3. The result of $P/2$ there is a balance between the risk of getting too far away from the peak of activity and being too close to no arrivals.

8 Some comments on software

There are several programs in common use that represent different approaches top file copying. It is interesting to relate these to the analyses presented here.

The well-known programs `rdump`, `ufsdump` and `tar` use a filesystem based approach to scanning files. These programs label files distinguishably but treat them as closed quanta; i.e. if a file is marked as changed the entire file is copied as a virtual unit. These programs are CPU intensive but fast and furious at copying, but contain little in the way of risk countermeasures. Cfengine treats

files in the same way but, due to its security checks and insistence on host autonomy, scans filesystems quite inefficiently and therefore tends to copy with low CPU load over long times. Using `cfengine` to take backup copies therefore results in relatively large t_b .

The program `rsync` [36] is a curiosity in this regard. It scans filesystems and treats files as virtual units, but once a change is discovered, it changes strategy and attempts to optimize the copy into byte differences. Since the underlying disk structure is in sectors, this will result in an unknown number of sector writes. The efficiency of `rsync` in terms of sector operations would make an interesting study in its own right.

`dd` copies raw device blocks (which are whole numbers of fundamental sectors). It therefore satisfies the model of indistinguishable entity copying, somewhat like a copy on write, but without random access. The `dd` limitation is that it copies as a stream so that the ordering of data changes as the device blocks are allocated. The result has no consistent structure, but satisfies our model as long as one adjusts for the changing locations of data in the stream. Another way of saying this is that it does not work convergently in the sense of ref. [37].

Many of the programs mentioned here are biased by the expectation of writing to tape, i.e. serial non-random access medium. This makes it hard to model their actual behavior in a given situation without a knowledge of the environment in which they operate. For that reason, we do not attempt to go beyond this kind of analysis in this paper (which is long enough already). Some of the results here are therefore only fully realized in copy-on-write duplication.

There are clearly advantages to a file-based backup if the purpose of backup is to archive versions. One needs to be able to distinguish sectors in order to retrieve from a backup copy without doing a complete restore.

9 Conclusions

We have examined the process of making a backup copy of data for either archiving or for redundancy. We examine the backup problem as a study of the risk of loss and use the maintenance paradigm to discuss this. We assume that the risk of loss occurs entirely at the data sources, i.e. that the backup process is itself reliable. To take into account unreliability in the backup procedure in non-trivial situations, some graph theoretical modifications to the present arguments are required.

Based on empirical data we have developed methods that determine the op-

imum time for a backup to be taken so as to minimize the risk of loss by a sudden catastrophe. For a user activity profile that peaks around midday, we find that middle to late afternoon is the optimum solution. This result should be recomputed in different environments.

We allow for the possibility that some data might be prioritized over others, or that different levels of consistency can be introduced by decisions about multiplicity of backup copies.

We emphasize that there are other aspects of backup strategy that we have not been able to take into account within the space of this paper. These include the choice of copying speed as well as considering multiple risk models that have potentially conflicting interests. Our model of constant copy rate depends on there being sufficient system headroom to make backup a bearable load on the system. On critically loaded systems, this is invalidated and the problem becomes considerably harder to solve. Hopefully few systems ever get into such difficulty. We hope to return to these matters in future work.

References

- [1] M. Poepping. Backup and restore for unix system. *Proceedings of the Large Installation System Administration Workshop (USENIX Association: Berkeley, CA, 1987)*, page 10, 1987.
- [2] C.B. Hommel. System backup in a distributed responsibility environment. *Proceedings of the Large Installation System Administration Workshop (USENIX Association: Berkeley, CA, 1987)*, page 8, 1987.
- [3] E. Zwicky. Backup at ohio state. *Proceedings of the Workshop on Large Installation Systems Administration (USENIX Association: Berkeley, CA, 1988)*, page 43, 1988.
- [4] P.E. Pareseghian. A simple incremental file backup system. *Proceedings of the Workshop on Large Installation Systems Administration (USENIX Association: Berkeley, CA, 1988)*, page 41, 1988.
- [5] S. Hecht. The andrew backup system. *Proceedings of the Workshop on Large Installation Systems Administration (USENIX Association: Berkeley, CA, 1988)*, page 35, 1988.
- [6] H.E. Harrison. A flexible backup system for large disk farms, or what to do with 20 gigabytes. *Proceedings of the Workshop on Large Installation Systems Administration (USENIX Association: Berkeley, CA, 1988)*, page 33, 1988.
- [7] T. Kovacs C.J. Yashinovitz and J. Kalucki. An optical disk backup/restore system. *Proceedings of the Workshop on Large Installation Systems Administration III (USENIX Association: Berkeley, CA, 1989)*, page 123, 1989.

- [8] K. Montgomery and D. Reynolds. Filesystem backups in a heterogeneous environment. *Proceedings of the Workshop on Large Installation Systems Administration III (USENIX Association: Berkeley, CA, 1989)*, page 95, 1989.
- [9] S.M. Romig. Backup at ohio state, take 2. *Proceedings of the Fourth Large Installation System Administrator's Conference (LISA IV) (USENIX Association: Berkeley, CA, 1990)*, page 137, 1990.
- [10] L.Y. Weissler. Backup without tapes. *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V) (USENIX Association: Berkeley, CA)*, page 191, 1991.
- [11] E.D. Zwicky. Torture testing backup and archive programs: things you ought to know but probably would rather not. *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V) (USENIX Association: Berkeley, CA)*, page 181, 1991.
- [12] J. Engquist. A database for unix backup. *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V) (USENIX Association: Berkeley, CA)*, page 89, 1991.
- [13] S. Shumway. Issues in on-line backup. *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V) (USENIX Association: Berkeley, CA)*, page 81, 1991.
- [14] R. Kolstad. A next step in backup and restore technology. *Proceedings of the Fifth Large Installation Systems Administration Conference (LISA V) (USENIX Association: Berkeley, CA)*, page 73, 1991.
- [15] M. Metz and H. Kaye. DeeJay: The dump jockey: a heterogeneous network backup system. *Proceedings of the Sixth Systems Administration Conference (LISA VI) (USENIX Association: Berkeley, CA)*, page 115, 1992.
- [16] J. Da Silva and Ólafur Guðmundsson. The amanda network backup manager. *Proceedings of the Seventh Systems Administration Conference (LISA VII) (USENIX Association: Berkeley, CA)*, page 171, 1993.
- [17] W. Curtis Preston. Using gigabyte ethernet to backup six terabytes. *Proceedings of the Twelfth Systems Administration Conference (LISA XII) (USENIX Association: Berkeley, CA)*, page 87, 1998.
- [18] E. Melski. Burt: The backup and recovery tool. *Proceedings of the Thirteenth Systems Administration Conference (LISA XIII) (USENIX Association: Berkeley, CA)*, page 207, 1999.
- [19] W.C. Preston. *Unix Backup and Recovery*. O'Reilly, London, 1999.
- [20] E.D. Zwicky. Further torture: more testing of backup and archive programs. In *Proceedings of The 17th Annual Large Installation Systems Administration Conference (LISA 2003)*, San Diego, California, USA, October 2003.
- [21] C. Qian, S. Nakamura, and T. Nakagawa. Replacement and minimal repair policies for a cumulative damage model with maintainance. *Computers and Mathematics with Applications*, 46:1111–1118, 2003.

- [22] R. Apthorpe. A probabilistic approach to estimating computer system reliability. *Proceedings of the Fifteenth Systems Administration Conference (LISA XV) (USENIX Association: Berkeley, CA)*, page 31, 2001.
- [23] M. Burgess. On the theory of system administration. *Science of Computer Programming*, 49:1, 2003.
- [24] A. Høyland and M. Rausand. *System Reliability Theory: Models and Statistical Methods*. J. Wiley & Sons, New York, 1994.
- [25] M. Burgess. *Principles of Network and System Administration*. J. Wiley & Sons, Chichester, 2000.
- [26] M. Burgess. *Analytical Network and System Administration — Managing Human-Computer Systems*. J. Wiley & Sons, Chichester, 2004.
- [27] G.R. Grimmett and D.R. Stirzaker. *Probability and random processes (3rd edition)*. Oxford scientific publications, Oxford, 2001.
- [28] R. Jain. *The art of computer systems performance analysis*. Wiley Interscience, New York, 1991.
- [29] M. Burgess, H. Haugerud, T. Reitan, and S. Straumsnes. Measuring host normality. *ACM Transactions on Computing Systems*, 20:125–160, 2001.
- [30] V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modelling. *IEEE/ACM Transactions on networking*, 3(3):226, 1995.
- [31] W. Willinger, V. Paxson, and M.S. Taqqu. Self-similarity and heavy tails: structural modelling of network traffic. *in A practical guide to heavy tails: statistical techniques and applications*, pages 27–53, 1996.
- [32] K.I. Sato. *Levy Processes and Infinitely Divisible Distributions*. Cambridge studies in advanced mathematics, Cambridge, 1999.
- [33] A.L. Barabási. *Linked*. (Perseus, Cambridge, Massachusetts), 2002.
- [34] P. Henrici. *Elements of Numerical Analysis*, chapter 6, Linear difference equations. John Wiley & Sons, Inc., 1964.
- [35] M. Burgess. Data redundancy in partially reliable and ad hoc networks. *Proceedings of the VIII IFIP/IEEE IM conference on network management*, page (submitted), 2005.
- [36] A. Tridgell and P. Mackerras. The rsync algorithm. *Technical report of the Australian National University*, 1996.
- [37] M. Burgess. Configurable immunity model of evolving configuration management. *Science of Computer Programming*, 51:197, 2004.