Statistical analysis and prioritisation of alarms in mobile networks

Stefan Wallin*

Data Ductus Nord AB,
Torget 6 SE-931 31 Skellefteå, Sweden

Luleå University of Technology,
Department of Computer Science and Electrical Engineering,
SE-931 87 Skellefteå, Sweden
E-mail: stefan.wallin@ltu.se
*Corresponding author

Viktor Leijon

Luleå University of Technology,
Department of Computer Science and Electrical Engineering,
SE-971 87 Luleå, Sweden
E-mail: viktor.leijon@ltu.se

Leif Landén

Data Ductus Nord AB,
Torget 6 SE-931 31 Skellefteå, Sweden
E-mail: leif.landen@dataductus.se

Abstract: Telecom service providers are faced with an overwhelming flow of alarms, which makes good alarm classification and prioritisation very important. This paper first provides statistical analysis of data collected from a real-world alarm flow and then presents a quantitative characterisation of the alarm situation. Using data from the trouble ticketing system as a reference, we examine the relationship between mechanical classification of alarms and the human perception of them. Using this knowledge of alarm flow properties and trouble ticketing information, we suggest a neural network-based approach for alarm classification. Tests using live data show that our prototype assigns the same severity as a human expert in 50% of all cases, compared to 17% for a naïve approach.

Keywords: communication system operations and management; neural network applications; alarm systems.

Reference to this paper should be made as follows: Wallin, S., Leijon, V. and Landén, L. (2009) 'Statistical analysis and prioritisation of alarms in mobile networks', *Int. J. Business Intelligence and Data Mining*, Vol. 4, No. 1, pp.4–21.

Biographical notes: Stefan Wallin received his MSc at Linköping University in Sweden in 1989. Since then, he has worked with network management solutions and standards; first for Ericsson from 1989 until 1995 and there after as a Senior Partner at Data Ductus AB, Sweden. Most of his work addresses integrated management solutions for mobile operators. He is also an appreciated speaker and trainer. He is also a part-time PhD student at Luleå University. His research interests are network and service monitoring for mobile networks.

Viktor Leijon received his Master of Science and Licentiate of Technology degrees from Luleå University of Technology, where he is currently pursuing his PhD in Computer Science. He has spent five years working as a Consultant in the Network Management Industry before returning to academia. His primary research interests are the foundations for concurrent real-time programming.

Leif Landén received his BSc Degree at Mid Sweden University of Sweden and is working as a Software Designer for Mobile Systems at Data Ductus AB. His research interests are neural networks and genetic algorithms.

1 Introduction

A medium-sized telecom network operations centre receives several hundred thousand alarms per day. This volume of alarms creates severe challenges for the operations staff. Fundamental questions that need answers in order to improve the state of affairs are:

- Which alarms can be filtered out?
- How can we group and correlate alarms?
- How can we prioritise the alarms?

While extensive research efforts are focused on alarm correlation (Steinder and Sethi, 2004), the target for the work presented in this paper is *filtering* and *prioritisation* of alarms.

Although all alarm systems support advanced filtering mechanisms, the problem is defining the filtering rules. Being able to filter out a high percentage of alarms would increase efficiency of the network management centre since network administrators would only have to work with relevant problems.

Because there is such a high volume of alarms and tickets this kind of filtering and prioritisation is of vital importance if operators are to determine which alarms are most critical to resolve (Wilkonzon and Lucas, 2002). Today, prioritisation of alarms and trouble tickets is largely performed manually by network administrators who use a combination of their experience and support systems such as inventory and SLA management systems to determine the priority of an alarm. This manual process makes the organisation dependent on a few individual experts (Wallin and Leijon, 2006). Furthermore, the priority information is typically only available in the trouble ticket system and not in the alarm system.

Two hypotheses are studied in this paper: that *statistic analysis techniques can be used to find alarm filtering strategies* and that *a learning neural network could suggest relevant priorities by capturing network administrators' knowledge*.

We start by define the inner workings of a telecom alarm flow (Section 2) and then describe how our data was extracted from the database (Section 3).

This paper takes four steps towards automatic alarm prioritisation:

- we present some statistical properties of a real world alarm flow taken from a mobile service provider (Section 4)
- we show important properties such as that 11% of all alarms belong to easily identifiable classes of alarms which never give rise to actions from operators and that over 82% belong to classes where less than one alarm in a thousand generate an action
- we describe the construction, training and validation of a neural network which successfully assigns priorities to incoming alarms (Section 5)
- using statistical analysis we show that the neural network performs significantly better than a naïve but realistic alternative.

2 Defining the alarm flow

The operational activities at a service provider's Network Management Center are focused on managing a constant flow of alarms (Wallin and Leijon, 2006). A primary goal is to resolve the most important problems as quickly as possible. A simplified process description for lowering error impact is:

- group alarms that are related to the same problem
- associate the alarms with a trouble ticket to manage the problem resolution process
- assign a priority to the trouble ticket
- analyse and fix the problem.

Alarms are refined and distributed from the detection point in individual network elements, such as base stations, via subnetwork managers up to the overall integrated network management systems. Various interface technologies and models for alarms are used across these interfaces. X.733 (ITU, 1992) is the *de facto* standard for *alarm* interfaces and all later standard efforts are based on X.733 to some degree. It contains basic definitions of parameters in alarm notifications.

The 3GPP Alarm IRP (3GPP, 2004) defines alarms using a state focused definition. Furthermore, it models operator actions that can change the state of an alarm. The state where the alarm is acknowledged and cleared is the final state, and the life cycle of the alarm ends.

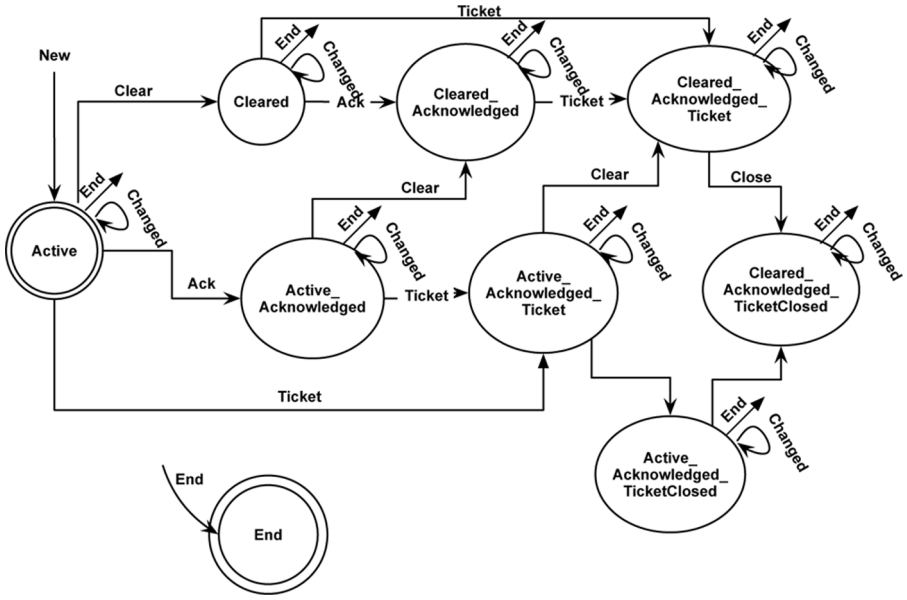
An X.733 compliant alarm has the following main attributes.

- *Managed object*: Identification of the faulty resource. It points not only to the network element but also down to the individual logical or physical component.

- *Event type*: Category of the alarm (communications, quality of service, processing error, equipment alarm, environmental alarm).
- *Probable cause*: Cause of the alarm, where the values are defined in various standards.
- *Specific problem*: Further refinement of probable cause.
- *Perceived severity*: An indication of how it is perceived that the capability of the managed object has been affected. Values are `indeterminate`, `warning`, `minor`, `major`, `critical` and `clear`.
- *Event time*: The time of the last event referring to this alarm.
- *Additional text*: Free form text describing the alarm.

For the purpose of this study, we have defined a Finite State Machine for alarms as shown in Figure 1. It is a simplification and abstraction of major standards and typical telecom management systems.

Figure 1 Finite state machine for management view of alarms



From the resource point of view, the main events are `new` and `clear`, which moves the alarm into the `active` or `cleared` state. Note, however, that the cleared state does not imply that the network administrator considers the problem solved. This is managed by the trouble ticket process. In order to manage the problem, the user acknowledges and associates a ticket with the alarm. We will refer to this as 'handling' the alarm. A trouble ticket contains information such as priority, affected services, and responsible work group. The mobile operator we studied used a priority in the trouble ticket system ranging from 1 to 6. Priority 1 is the most urgent and indicates a problem that needs to be resolved within hours, whereas

Priority 6 has no deadline. When the problem is solved, the administrator closes the trouble ticket. The life cycle of an alarm ends when a user decides that the alarm needs no further attention. This is indicated with `end` in the above state diagram. In many cases, this is automatically performed when the associated trouble ticket is closed. There is a short cut to bypass the trouble ticket process in order to end alarms that do not represent real problems.

Whether an alarm notification should be considered a *new* or *changed* alarm is a topic of its own. According to X.733, a notification with the same managed object, event type, probable cause and specific problem is considered to change an existing alarm. The three last parameters identify the type of alarm, and we will refer to the triple <`Event type`, `Probable cause`, `Specific problems`> as 'alarm type'. We will refer to alarms with the same managed object and alarm type as 'associated alarms', and they will be grouped together under one main alarm, the changed alarm in the state diagram.

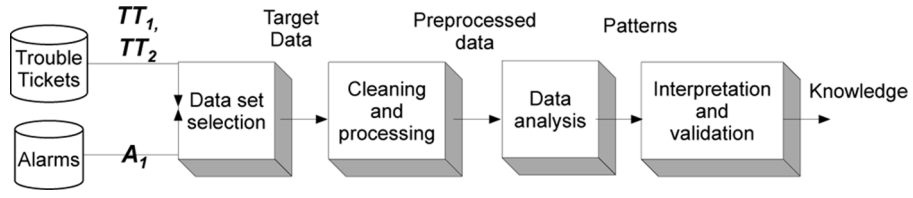
The X.733 definition of alarm type has created various vendor-specific mechanisms since the parameters are static and defined by standards. In real life, a vendor needs to be able to add new alarm types in a deployed system. The approaches to circumvent this differ between vendors, some use their own non-standardised probable cause values, and some use free text specific problem fields. A second problem is how to identify the managed object, different protocols and vendors use different naming schemes. The actual resolution varies from vendor to vendor. These two fundamental problems create major challenges for alarm systems and alarm analysis.

In the described alarm flow, network administrators need to answer two important questions: *Do I need to handle this alarm? What is its priority?* The operator we studied primarily uses the event time, managed object, and alarm type attributes in combination with their own experience and lookups in support systems to judge the alarm relevance.

3 Data mining process

Figure 2 illustrates the overall process in data mining (Bose and Mahapatra, 2001). Each step in this process is described below.

Figure 2 Data mining process



3.1 Data set selection

We used historical databases of alarms and trouble tickets as input. Two separate data sets were extracted at different times:

- $\{TT_1\}$: only trouble tickets (85,814 tickets associated with 2,60,973 alarms)
- $\{A_1, TT_2\}$: alarms and corresponding tickets (35,83,112 main alarms, 16,1,50,788 associated alarms and 90,091 trouble tickets).

The alarms were from approximately 50 different types of equipment from many different vendors.

The records in the trouble ticket database contain the most important fields from the alarms such as managed object, specific problem and additional text. In this way the trouble ticket database alone is sufficient to perform analysis and neural network training on alarms with associated tickets. This is the case for $\{TT_1\}$ which was used for training and testing of the priority network, as described in Section 5.

In order to perform full statistical analysis of the alarms as well as to train the neural network to judge if the alarm should be handled or not, we extracted a second data set $\{A_1, TT_2\}$.

3.2 Cleaning and processing

The cleaning and processing of the alarms meant calculating *time dimensions*, and generating unique *alarm type identifiers*. It is not always easy to judge what identifies the true original time of the alarm detection since some equipment sets an absolute time stamp in the alarm, whereas in other cases it is not time stamped until it reaches the network management systems. Since we are studying the administrative processes around alarm management and not temporal alarm relationships, we simplify the data by using the time the alarm was created in the database as reference. This would not be a valid approach in the alarm correlation case where the time of alarm detection is relevant.

A fundamental problem is to define what identifies a unique alarm type. As described in Section 2 this is not as simple as applying the X.733 rule of event type, probable cause and specific problem. The data in our alarm database had free text alarm types embedded as part of the additional text field. This was implemented as part of every equipment integration into the overall network management system. We extracted this text and made it a column in the database. The data set contained about 3500 unique specific problem values.

3.3 Data analysis

The data analysis was done using both statistical analysis as described in Section 4 and as training/testing data for the neural network, as described in Section 5.

4 Quantitative analysis of alarm flow

4.1 Basic description of the data

This is a quantitative analysis of the alarm set A_1, containing a total of 3,583,112 alarms, 12,567,676 associated alarms (on average 3.51 per main alarm) and 90,091 tickets. Among the alarms which are connected to tickets there is an average of 36 associated alarms, while unconnected alarms had only three associated alarms.

The distribution of severities is shown in Table 1 and shows us that the distribution of severities is very uneven.

Table 1 Distribution of priorities in A_1

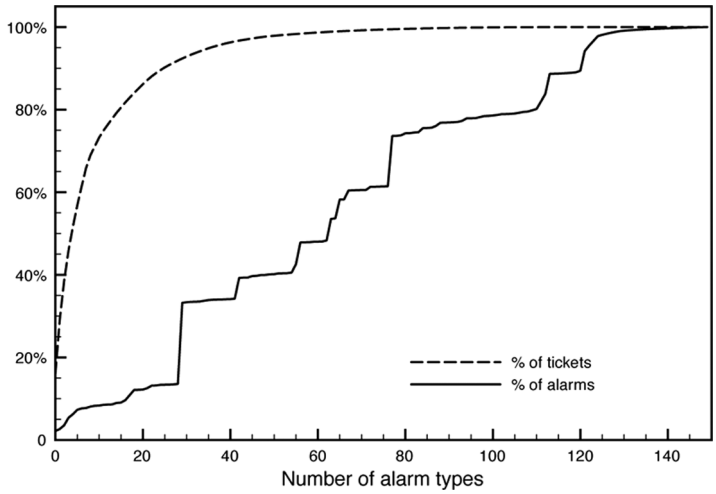
Severity	*Frequency* (%)
Indeterminate	0.1
Critical	17.5
Major	22.8
Minor	5.0
Warning	54.6

Just under half of the alarms completely lack associated alarms, and 98.8% of them have less than ten associated alarms.

4.2 Alarm types

In total there are over 3500 different alarm types defined (see Section 3.2), but most of them are very rare. Given the distribution shown in Figure 3 we see that some form of the Pareto principle seems to be at work here. We get 90% of all our tickets from the 26 most common alarm types. This indicates that there is a great deal to gain from improving automation for these 26 alarm types, or using methods such as improved alarm enrichment and correlation or perhaps helping semi-automate the ticket creation process.

Figure 3 Cumulative distribution of tickets and alarms by alarm type



On the other side of the scale, 10.6% of all alarms belong to alarm types that have never given rise to an actual ticket, and if we look at alarms which belong to an alarm type that results in a ticket less than once every 1000 alarms, we get 82.1% of all alarms. While this requires a detailed study, it does suggest that a great reduction in alarm volume can be achieved by exploring filtering on these alarms.

4.3 Temporal properties

The material contains data from the May–September. A total of 120 days are covered with an average of 29859 alarms a day, or 1244 alarms per hour. In contrast, an average of only 116 tickets were created per day. The differences between the months are small.

One common wisdom in the telecom industry is that many alarms are caused by operator activity, that many the problems come from upgrades, reconfigurations and other events, implying that the operations themselves cause many alarms. To test this we calculate the average number of alarms and tickets for the weekdays (see Table 2). We can see that the busiest day (Wednesday) has 24.5% more alarms than the calmest day (Sunday), while having more than double the number of tickets.

Table 2 Number of alarms and tickets by weekday

Day	*No. alarms*	*Alarms/day*	*No. tickets*	*Tickets/day*
Monday	5,07,308	29,842	9247	544
Tuesday	5,44,093	32,005	12,798	753
Wednesday	5,48,529	32,266	9195	541
Thursday	5,62,218	31,234	9761	542
Friday	5,27,041	31,002	8405	494
Saturday	4,52,910	26,642	6017	354
Sunday	4,40,705	25,924	4386	258

4.4 Severities vs. priorities

This section studies the relationship between priorities and severities, and refers to data from data set TT_2 (see Section 2).

The number of alarms associated with a trouble ticket varies from 1 to 1161, an average of 5.2 with a standard deviation of 17. This is an indication that the relationship between alarms and tickets is complex. In an ideal world, alarms should indicate a problem and not individual symptoms. If this were true, the fan-out between tickets and alarms would have been much lower.

In Figure 4 we can see that the distribution of alarm severities associated with a single trouble ticket is low. If, for example, a trouble-ticket is associated with both a `major (2)` and a `warning (4)` alarm, it has a difference of two steps, as illustrated by Figure 4. The associated alarms have the same severity in more than 50% of the cases.

Discussions with the network administrators led to the hypothesis that if we looked at the maximum alarm severity associated with the trouble ticket, we would get good correlation. Figure 5 shows the analysis of this assumption. It clearly illustrates that we do not have a mapping between alarm severity and corresponding priority. For example, we see that Priority 4 is distributed across all severities, and it is largest in the `warning` and `critical` severity. For further discussion on how badly severities and priorities correlate, see Section 5.4.

After studying the weak correlation between alarm severity and ticket priority, we looked for another correlation: the alarm type vs. priority. We observe a strong

correlation between some alarm types and their priority, but for other alarm types there is no correlation at all. So there is no direct and naïve alarm algorithm associating priority and alarm information.

Figure 4 Differences in associated alarm severity for tickets

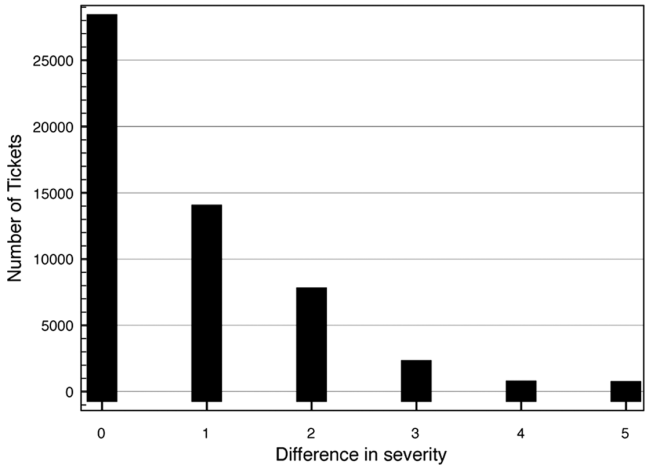
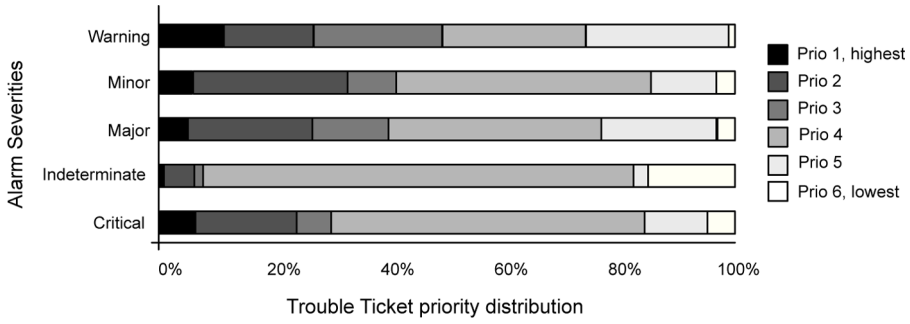


Figure 5 Ticket priorities per maximum severity



5 Using neural networks for prioritising alarms

5.1 The architecture

Current solutions for *automatic* alarm prioritisation are mainly based on service impact tools and expert systems which use information from external systems to modify alarms. This allows a prioritisation algorithm to decide upon priorities since the alarm is bound to topology, service and business impact (Wallin and Leijon, 2007). This type of solution have some intrinsic problems (Sterritt, 2002):

- *Maintenance of service models*: Formal models of network and service topology have to be maintained, which is complex and costly. Also, the change rate of network topology and service structures is challenging to handle.

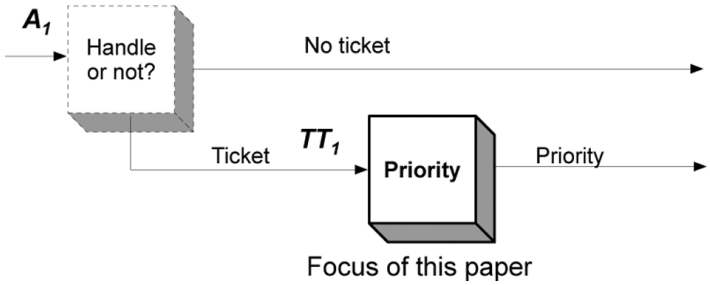
- *Maintenance of impact rules*: Correlation rules are typically expressed using Rete-based expert systems (Forgy, 1991), which require extensive programming.
- *Capturing operators knowledge*: To write the rules for the expert system, the developers need to have input from the network administrators. However, experienced operators are often critical resources in the organisation and cannot be allocated time to formalise rules.

The goal of our work is to set priorities in alarms at the time of reception by using neural networks. We let the neural network learn from the experienced network administrators rather then to building complex correlation rules and service models. As stated by Penido et al.:

> "Experienced decision makers do not rely on formal models of decision making, but rather on their previous experience. They use their expertise to adapt solutions to problems to the current situation." (Penido et al., 1999)

The alarm management process can be viewed as a two-step procedure, see Figure 6. First of all, is the alarm important enough to create a trouble ticket? Secondly, if we create the trouble ticket what is the priority? We focus on the latter in this paper.

Figure 6 The two-step neural network training



Deciding if an alarm deserves to be handled or not seems to be a more complicated question, based on some preliminary tests, information outside of the raw alarms is probably used in the decision process.

We have integrated a neural network architecture into an alarm and trouble ticket system. The neural network uses the manually assigned trouble ticket priorities and associated alarms as learning data. When the alarm system receives an alarm, it interrogates the trained neural network which generates a suggested priority to be put into the alarm information. The priority indicates if the alarm should be handled or not and, if so, the suggested priority.

The A_1 data set was used to train the network to judge if a trouble ticket should be created. This works since the alarm database contains a field for each alarm telling if it has a trouble ticket or not. The TT_1 data set was used to train the network to assign priorities to the alarms. The trouble ticket database contains all relevant attributes of the associated alarms.

We used the following alarm fields to construct the input to the neural network:

- *Managed object*: the resource emitting the alarm
- *Specific problem*: alarm type
- *Additional text*: free text field with alarm information
- *Perceived severity*: original severity as reported from the equipment
- *Associated alarms count*: the number of alarm notifications referring to the same alarm.

The selection of the above alarm attributes was based on discussions with network administrators to find the most significant attributes used for manual correlation and prioritisation. `Additional text` and `Specific problem` are encoded using the soundex algorithm to remove the influence of numbers and convert the strings to equal length.

We used an open source *neural network engine* named `libF2N2` (lif2n2, 2007). The lib2f2n2 library uses linear activation

$$f(x) = x \tag{1}$$

for the input layer and the logistic function

$$f(x) = 1/(1 + \mathrm{e}^{-x}) \tag{2}$$

for all successive layers. The neural network uses back-propagation (Rumelhart et al., 1986) as the learning mechanism. It only supports iterative back-propagation, not batch back-propagation.

Two variables play a special role during learning: the *learning rate* and the *momentum* of the neural network. Learning rate indicates what portion of the error should be considered when updating the weights of the network. Momentum indicates how much of the last update that should be reused in this change. Momentum is an effective way to move a network towards a good generalisation, but can also be a problem if the momentum moves us away from the optimum weights.

5.2 Assigning priorities using neural networks

In Section 4.4 we showed that severities cannot be used as priorities. Our solution will, however, give network administrators a proposed priority based on their experience.

The priority of an alarm partially depends on its grouping with other alarms. The grouping in this case is performed by the network administrators when performing manual alarm correlation and creating the trouble ticket. The neural network, on the other hand, will analyse alarms one by one and assign a priority.

5.3 Test configurations

The prototype was tested with different settings both for learning rate, momentum and neural network structure. The tests are outlined in Table 3. Since the

learning is slow, approximately 3 min per epoch, we stopped the learning when the error rate stabilised. Each test was performed with data *not* used during the training and is randomly chosen from the data-set. About 10% of the data set was used for training.

Table 3 Test results: L = Layers, N = Neurons, O = Output, LR = Learning Rate, M = Momentum, Tr E = Training Error, Te E = Test Error

Test	*Epochs*	*L*	*N*	*O*	*LR*	*M*	*Tr E (%)*	*Te E (%)*
1	1200	3	200	6	0.01	0.3	3.1	18.1
2	680	3	100	6	0.03	0.3	2.4	17.7
3a	100	4	50	6	0.03	0.2	6.1	16.0
3b	1000	4	50	6	0.03	0.2	4.7	16.9
4	300	3	70	1	0.05	0.2	31.8	12.8
5	1000	2	100	6	0.05	0.2	3.1	30.0

`Layers`, `Neurons` and `Output` in Table 3 describe the architecture of the neural network. The number of neurons used in each hidden layer is given by the `Neurons` field of the table. The 6 neuron `Output` (Tests 1–3 and 5) was a mapping where each neuron represented one priority and the one that got the highest result was the proposed priority. The 1 neuron `Output` (Test 4) was a scaled priority where 0.1 represented Priority 1 and 0.25 represented 2 and so on. `Training Error` is the average of the mean square error of the last epoch in the training. `Testing Error` is the average error of each prioritised alarm. A priority error of one, e.g., assigning Priority 3 instead of 4, equaled 20% in the 6 neuron output and 15% in the 1 neuron output.

The tests show us that the number of layers are more important than the number of neurons in each layer. The extremely low error on Test 3 shows that 4 layers is a better alternative than 3. Test 5 shows that with only 2 layers the application does not learn to prioritise. Adding more neurons does not make the neural network prioritise better. Notice how the error drops when decreasing the number of neurons between Tests 1 and 2. Although the `Testing Error` drops considerably when using the 1 neuron `Output` the high `Training Error` makes us reluctant to use it.

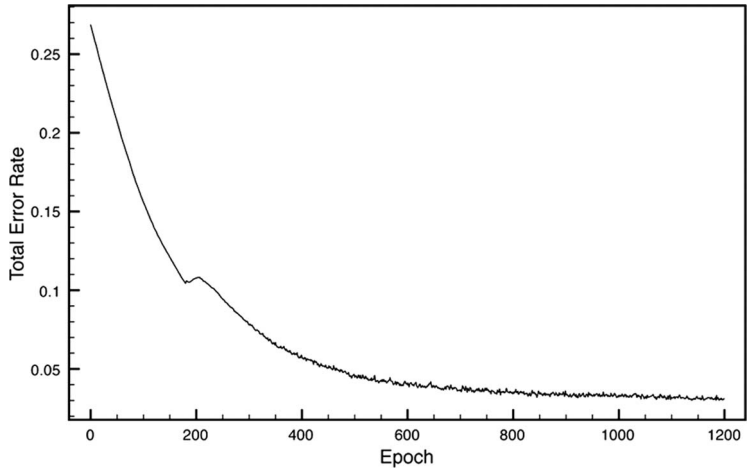
In Test 3 we can see that we do not necessarily get a better result from longer training. More work is needed to find a suitable method of when to stop training. We have no direct correlation between `Training Error` and `Testing Error`. The high `Training Error` on Test 4 is accompanied by a low `Testing Error` and in Test 5 we have the opposite relation.

Figure 7 shows how mean square error descends during training for Test 1. All tests, except Test 4, produced similar graphs.

5.4 Results

Having identified approach 3b, see Table 3, as the most promising one, we decided to statistically evaluate the success of this algorithm. For all the confidence intervals below we have used a standard t-test.

Figure 7 Mean square error decrease during learning



We compared the neural network to four competing approaches:

- The neural network approach as described for 3b.
- The trivial severity approach, simply scaling the severity into the priority by multiplying the severity by 1.25.
- Always selecting the statically optimal priority optimising for 'least average error', computed from the reference. The statically optimal choice turned out to be 3.
- Selecting the priority at random for each alarm, but using the same distribution of priorities as the reference.

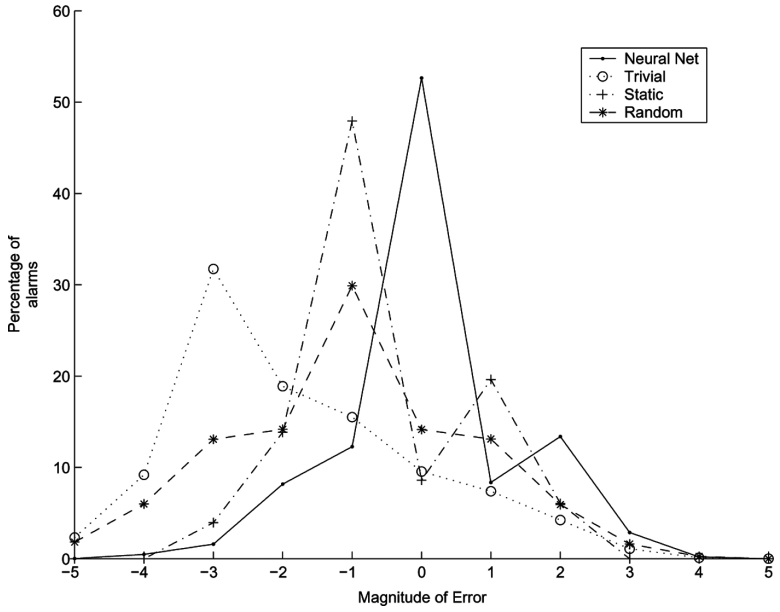
The average errors and variances for the methods, together with the 99% confidence interval on their mean errors are given in Table 4.

Table 4 Average error and variants for the methods

Method	*Average error*	*Error variance*	*Confidence interval*	*Difference from A*
1	0.8001	1.3450	[0.7938 . . . 0.8064]	–
2	1.9054	0.9463	[1.8979 . . . 1.9129]	[1.0973 . . . 1.1132]
3	1.1922	0.4063	[1.1881 . . . 1.1963]	[0.3973 . . . 0.3869]
4	1.6161	1.4241	[1.6080 . . . 1.6234]	[0.8235 . . . 0.8084]

We have plotted the relative frequency of the errors in Figure 8. A negative error errs on the side of caution, judging an alarm to be more serious than it actually is. A positive error on the other hand is an underestimation of the importance of the alarm. For the purpose of this study we consider both types of error to be equally bad. We can see that while the neural network is the only method centred around zero, the others generally overestimate the seriousness of the alarms slightly.

Figure 8 Distribution of errors (%)



Having seen what appears to be a distinctive advantage for the neural networks, we apply t-tests to try to determine how the effect of the other methods compares to A. We did this by comparing the error pairwise for each alarm in the test set and computing the 99% confidence intervals. The results are presented in the final column of Table 4.

It is spectacular how well the 'always pick priority 3'-method does. This is because it guesses in the middle, so it is often wrong, but usually just a single step. This rule is, of course, worthless from a prioritisation point of view.

The comparisons are all to the advantage of the neural network. It has a statistically secure advantage over the other methods. This suggests a distinct advantage of using neural networks for alarm prioritisation. This becomes an even stronger conclusion when one considers the distribution of errors in Figure 8.

6 Related work

Data mining, or knowledge discovery in databases, is being used in different domains such as finance, marketing, fraud detection, manufacturing and network management (Brachman et al., 1996). The major categories of machine learning used are rule induction, neural networks, case-based reasoning, genetic algorithms, and inductive logic programming (Bose and Mahapatra, 2001). The following problem types are typically addressed:

- *Classification*: The training data is used to find classes of the data set. New data can then be analysed and categorised. This is the main theme of our work, where we are looking for ticket priorities based on the alarm data.

- *Prediction*: This focuses on finding possible current and future values such as finding and forecasting faults in a telecommunication network. This is covered well by related research efforts.
- *Association*: Attempts to find associations between items such as dependencies between network elements and services (Ensel, 1999).
- *Detection*: Focuses on finding irregularities in the data and seeks to explain the cause. Within the telecom sector a common application is to detect churn and fraud.

Bose and Mahapatra (2001) performed a study to analyse the usage of data mining techniques across domains and problem types. They found that 7% of the usage was in the telecom sector and almost evenly spread among classification, prediction and detection.

Gardner and Harle (1997) illustrate the classification category. They use a self-organising map, a Kohonen network (Kohonen, 2001), to categorise alarms. In contrast with conventional ANN networks, a self-organising map does not require a correct output as training data. The primary application is analysis and classification of input where unknown data clusters may exist. The network is in a sense self-learning. This is in contrast to our prioritisation scenario where we *have* a complete output definition.

Most research efforts related to alarm handling focus on correlation (Frölich et al., 1997; Jakobson and Weissman, 1993; Liu et al., 1999; Meira, 1997; Penido et al., 1999). This mainly falls into the *prediction* and *detection* categories above. Alarm correlation refers to the process of grouping alarms that have a reciprocal relationship (Gardner and Harle, 1996). The aim is "the determination of the cause" (Sterritt, 2002). Wietgrefe et al. (1997) uses neural networks to perform the correlation. Common for these efforts is that they look at the stream of alarms and tries to find the root cause or the triggering alarm. For example, in the Wietgrefe study, the learning process is fed with alarms as inputs and the triggering alarm as output.

We are not trying to find the root cause of the alarms, neither to group them. In contrast, our problem is in some sense a simpler one, but overlooked; *to prioritise the individual alarms*. The *main* input of our analysis is the manual alarm prioritisation in trouble tickets along with the alarms. Previous efforts have mostly focused on the alarm databases themselves. The use of the trouble ticket database in the learning process makes the solution adapt to expert knowledge. Training a network with root cause alarms is a fundamental challenge since it is hard to find a true output set.

Sasisekharan et al. (1996) combine statistical methods and machine learning techniques to identify "patterns of chronic problems in telecom networks". Network behaviour, diagnostic data, and topology are used as input in the solution. This solution covers the challenging aspect of problem prediction. It is more focused on data-mining techniques and uses topology as input. It shows a strength in combining several approaches.

Levy and Chillarege (2003) also combine data mining and machine learning in an implementation of an alarm system. They come to the same conclusion as we do regarding the Pareto distribution of the alarms: "there is a lot of value in the

ability to get an early warning on the 10% of causes that create 90% of field failures". This further emphasises the foundation for the work presented in this paper where we are focusing on pinpointing the relevant alarms to be handled.

Klemettinen et al. (1999) presents a complementary solution to alarm correlation, using semi-automatic recognition of patterns in alarm databases. The output is a suggestion of rules, users can navigate and understand the rules. This is in contrast with neural networks solutions like ours, where there is no explanation to users why our network suggests alarms to be handled, and the suggested priority.

7 Conclusion and future work

The network operator that was the source for our data indicated that priority estimates that were within one step of the true value would be useful, something we manage for most of the alarms as shown by Figure 8. This is a statistically significant improvement for operators compared with the currently available alarm severity.

We have presented a solution that adjusts automatically to network administrators by learning from the trouble ticket database. This is an efficient use of human expert knowledge compared to traditional approaches using rule-based systems which has the underlying problem of converting human knowledge to rules. It also avoids the complex problem of having a complete set of rules and topology information in an ever-changing environment.

This solution has several benefits for the service providers:

- priorities are available immediately as the alarms arrive
- captures network administrators' knowledge without disturbing their business-critical work
- adapts to changing behaviour and changing network topologies.

Further, we have provided some characterisation of the alarm flow, showing the long-tail behaviour of alarm types, providing an opportunity for further study of the possibility of exploiting this from two sides, both from alarm filtering at the 'non-important' side and for alarm automation/enrichment at the 'important' side.

The work presented in this paper was run using a historic database of alarms and trouble tickets. The next step is to deploy the test configuration in a running system to study how it adapts continuously. We will also apply the tests using data from a different operator.

References

3GPP. 3GPP TS 32.111: (2004) *Alarm Integration Reference Point (IRP)*.

Bose, I. and Mahapatra, R.K. (2001) 'Business data mining, a machine learning perspective', *Information and Management*, Vol. 39, No. 3, pp.211–225.

Brachman, R.J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G. and Simoudis, E. (1996) 'Mining business databases', *Commun. ACM*, Vol. 39, No. 11, pp.42–48, ISSN 0001-0782, doi: http://doi.acm.org/10.1145/240455.240468.

Ensel, C. (1999) 'Automated generation of dependency models for service management', *Workshop of the OpenView University Association*, Bologna, Italy.

Forgy, C.L. (1991) 'Rete: a fast algorithm for the many pattern/many object pattern match problem', *IEEE Computer Society Reprint Collection*, pp.324–341, http://portal.acm.org/citation.cfm?id=115710.115736#

Frölich, P., Nejdl, W., Jobmann, K. and Wietgrefe, H. (1997) 'Model-based alarm correlation in cellular phone networks', *Fifth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Haifa, Israel, pp.197–204.

Gardner, R.D. and Harle, D.A. (1996) 'Methods and systems for alarm correlation', *Global Telecommunications Conference (GLOBECOM'96)*, Vol. 1, pp.136–140.

Gardner, R.D. and Harle, D.A. (1997) 'Alarm correlation and network fault resolution using the Kohonen self organising map', *Global Telecommunications Conference (GLOBECOM'97)*, Vol. 3.

ITU. (1992) *X.733: Information Technology – Open Systems Interconnection – Systems Management: Alarm Reporting Function*, Geneva, Switzerland.

Jakobson, G. and Weissman, M.D. (1993) 'Alarm correlation', *Network, IEEE*, Vol. 7, No. 6, pp.52–59.

Klemettinen, M., Mannila, H. and Toivonen, H. (1999) 'Rule discovery in telecommunication alarm data', *Journal of Network and Systems Management*, Vol. 7, No. 4, pp.395–423.

Kohonen, T. (2001) *Self-Organizing Maps*, Springer, http://www.springer.com/physics/book/978-3-540-67921-9

Levy, D. and Chillarege, R. (2003) 'Early warning of failures through alarm analysis-a case study in telecom voice mail systems', *Proceedings of the 14th International Symposium on Software Reliability Engineering*, IEEE Computer Society Washington, DC, USA, p.271.

lif2n2. (2007) *libF2N2 Feedforward Neural Networks*, Accessed 10th of August, http://libf2n2.sourceforge.net/

Liu, G., Mok, A.K. and Yang, J.E. (1999) 'Composite events for network event correlation', *Proceedings of the Sixth IFIP/IEEE International Symposium Network Management*, Boston, MA, USA, pp.247–260.

Meira, D.M. (1997) *A Model For Alarm Correlation in Telecommunications Networks*, PhD Thesis, Federal University of Minas Gerais, November.

Penido, G., Nogueira, J.M. and da Mata Machado, C. (1999) 'An automatic fault diagnosis and correction system for telecommunications management', *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management*, Boston, MA, USA, pp.777–791.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986) 'Learning internal representations by error propagation', in Rumelhart, D.E. and McClelland, J.L. (Eds.): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, Cambridge, MA, USA, pp.318–362, ISBN 0-262-68053-X.

Sasisekharan, R., Seshadri, V. and Weiss, S.M. (1996) 'Data mining and forecasting in large-scale telecommunication networks', *IEEE Expert: Intelligent Systems and Their Applications*, Vol. 11, No. 1, pp.37–43, ISSN 0885-9000, doi: http://doi.ieeecomputersociety.org/10.1109/64.482956.

Steinder, M. and Sethi, A.S. (2004) 'A survey of fault localization techniques in computer networks', *Science of Computer Programming*, Vol. 53, No. 2, pp.165–194.

Sterritt, R. (2002) 'Towards autonomic computing: effective event management', *Proceedings of the 27th Annual Software Engineering Workshop*, Greenbelt, MD, USA, pp.40–47, NASA Goddard/IEEE.

Wallin, S. and Leijon, V. (2006) 'Rethinking network management solutions', *IT Professional*, Vol. 8, No. 6, pp.19–23.

Wallin, S. and Leijon, V. (2007) 'Multi-purpose models for QoS monitoring', *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops-Volume 01*, Niagara Falls, Canada, pp.900–905.

Wietgrefe, H., Tuchs, K-D., Jobmann, K., Carls, G., Fröhlich, P., Nejdl, W. and Steinfeld, S. (1997) 'Using neural networks for alarm correlation in cellular phone networks', *International Workshop on Applications of Neural Networks to Telecommunications (IWANNT)*, Melbourne, Australia, pp.248–255.

Wilkonzon, J. and Lucas, D. (2002) 'Better alarm handling – a practical application of human factors', *Measurement and Control*, Vol. 35, No. 2, pp.52–55.