Stochastic event based model checking for predicting component failures and service availability

Muffy Calder and Michele Sevegnani

Abstract—When a component failure occurs in a complex, critical system it can be difficult to assess the impact and prioritise and schedule repairs. We propose stochastic event based modelling, based on continuous time Markov chains, and analysis by model checking stochastic temporal logic properties, can inform assessments and operational decisions. By relating the status of components to service availability, we quantify the risk of service failure now, and in given times the future. Decisions about which component to repair, and when, can then be taken according to those risks. The quantified risks can also inform evaluation of designs. We demonstrate the approach through application to an industrial case study and we calibrate the models with rates inferred from historical field data about failures over a one year period. Our analysis allows us to predict the likelihood of no-service within 48 hours, before and after making repairs to specific components, and to estimate expected costs for maintentance, over a specific time period (*e.g.* a month).

Index Terms—Computer network reliability, decision support systems, formal verification, predictive models, stochastic systems.

1 Introduction

Operational decision making about *which* components to fix in a complex, critical service, and *how quickly*, in the event of component failures and reduced redundancy, is difficult. Ideally, when failures are uncovered (*e.g.* through monitoring and sensing), they would be fixed immediately. But this might not be possible, *e.g.* due to limited resources, the nature of the failure, and/or physical distance or access to the component.

The problem we address is: how can we assess if the failure is critical, and how can we prioritise and schedule repairs, to make best use of resources, while ensuring the service is operating within acceptable levels of risk of failure, up until such time as the failure can be repaired or ameliorated?

Key to our approach is predicting future behaviours and the risk of service failure from: i) a state in which faults have occurred, and ii) a state after repairs are made to selected components. We call a system configuration in which there are faulty components *degraded*, so our approach entails quantification of risk of service failure from degraded configurations, both at the current time, and at future times.

We consider systems in which components have state and collections of components provide a service. There may be hierarchies of components in which a lower level provides a service to a higher level, and numerous levels of redundancy. Typical examples of such systems include communications infrastructure and condition monitoring.

We propose a stochastic event-based modelling approach, with analysis by model checking stochastic temporal logic formulae that quantify risks. By relating component failures to service availability, our analysis supports offline and online decision making, as well as evaluation of design aspects such as system architectures, monitoring practices, degrees of autonomy and levels of redundancy and hierarchy. We demonstrate the approach through application to an industrial case study of a critical service in which component failures are sensed and monitored. The service has been deployed for several years and so a novel aspect of this work is we calibrate the model(s) based on inferences over historical field data. This means our reasoning can inform decision making in the actual deployed system.

We advocate temporal properties and model checking because this allows us to quantify *all* possible future behaviours, whereas simulation only ever considers one behaviour (or trace) at a time. Thus we can deliver a comprehensive analysis.

Service availablity is our primary concern and so the key temporal property is: from a degraded configuration, what is the likelihood over the next t hours of the system reaching a no-service state? Knowledge of how that likelihood varies over time can help us quantify the risk to the system posed by that fault and the urgency of repair, and contribute to answering questions such as "do I need to fix the fault right now, in the next 4 hours, or can I wait until tomorrow?". For example, if we find the probability is well below an established safety threshold during the next 4 hours, but thereafter rises exponentially above the threshold, we would conclude that a repair need not be immediate, but must be completed within the next 4 hours. There may be other parameters (*e.g.* resources and costs) to minimise or maximise.

1.1 Overview of approach

Our approach is *hierarchical*, *compositional* and *event-based*, the last reflecting that components transition between *states*.

• *M. Calder and M. Sevegnani are with the School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, UK.*
E-mail: {muffy.calder,michele.sevegnani}@glasgow.ac.uk

Manuscript received July 21, 2015; revised September 17, 2015.

Each component is regarded as a process, and the overall system is the concurrent composition of all components, synchronising on common events. Two key decisions are the level of abstraction for components and how to model the passage of time in the system. For the former, since our concern is availability of higher level services, we employ counter abstraction models for lower levels of the hierarchy (a counter abstraction records the *numbers* of processes in a particular state, rather than details of which process is in which state). We assign *rates* to events, thus our models are continuous time Markov chains (CTMCs), *i.e.* the state space is discrete but time is continuous. By adopting CTMCs as the underlying semantics, we can relate our models to mean time between failure (MTBF) values, if required: if the MTBF is r, then the associated rate for the (failure) event is $1/r$ and the probability the event occurs/has completed by time t is exponential: $1 - e^{r \cdot t}$. Note, we can instantiate rates in a variety of ways: rates derived from safety and business cases, rates inferred from actual historical field data, and hypothetical rates that reflect possible changes to business or technical processes. Throughout, we assume prior knowledge of key events and system *structure*.

1.2 Contributions

Our focus is a case study of a communications link monitoring system with four levels of hierarchy and high degrees of redundancy. The contributions of the paper are:

- an event based, parameterised, CTMC model of the case study system,
- identification of degraded configurations in which the service is available, but there is reduced redundancy,
- steady state temporal logic properties that are used to *validate* the model against expected and historical behaviour,
- transient temporal logic properties that are used to *predict* the likelihood of service availability and maintenance costs over a time period from a given state, and to distinguish envelopes of degraded behaviours,
- how transient property results can inform decision making concerning which components to repair and how quickly,
- how steady state and transient temporal logic property results can inform design evaluations,
- example analyses of the model with event rates inferred from actual field data over a one year period.

The paper is organised as follows. The next section contains an overview of the case study system and in following that we review basic concepts of CTMCs and continuous stochastic logic (**CSL**). Section 4 contains an overview of the case study model and Section 5 defines rates for the model based on inferences over historical data. Section 6 defines the propositions, steady-state and transient **CSL** properties we use for analysis, along with results for an example sector. Section 7 shows how, by way of examples, analysis of transient properties can inform decision making concerning which component to repair, and to what timescale. In Section 8, we define *envelopes of behaviour* for transient properties and degraded configurations, which quantify the effect of the status of lower level components on service availability. In Section 9, we consider two properties concerning behaviours *after* reaching no-service: recoverability and survivability, and in Section 10 we present analysis of (maintenance) costs using PRISM rewards, again illustrating with examples. An overview of a web app we developed to make the analysis process more accessible is presented in Section 11, and in Section 12 we summarise our modelling and analysis framework, reflecting upon what we have learned from the case study. Related work is discussed in Section 13, followed by conclusions.

2 Case study: a communications link monitoring system

The primary components of the system are *channels*, *frequencies*, *sites*, and *sectors*. There are 35 sectors, each of which is allocated a fixed set of frequencies, plus an emergency frequency. There are 17 sites, each with several antennas, or *channels*, that transmit (Tx) and receive (Rx) on different frequencies. There is redundancy by design: every sector is allocated several frequencies, a frequency is covered by more than one site, and in every site there are idle backup channels. We refer to the main channel as the A channel and the backup channel as the B channel. Sites are monitored for power line status, communication link status, and there are sensors for intrusion and flooding.

2.1 Monitoring

The monitoring system senses components in real-time and uses a colour coding to indicate status: green – functioning or serviceable; red – faulty, raise an alarm; blue – under maintenance; amber – reduced redundancy and possibly not fully functioning (for example, when one antenna goes down for a given frequency).

3 Technical background

3.1 Continuous Time Markov Chains

Following [1], given a finite set of atomic propositions AP, a *(labelled) continuous-time Markov chain* (CTMC) is a triple $\mathcal{C} = (S, R, L)$ where S is a finite set of states with a designated initial state, $R : S \times S \to \mathbb{R}_{\geq 0}$ a rate matrix, and $L : S \to 2^{AP}$ a labelling of states. The exit rate $E(s) = \sum_{s' \in S} R(s, s')$ denotes the probability of taking a transition from s within t time units and is equal to $1 - e^{-E(s) \cdot t}$. If $R(s, s') > 0$ for more than one state s', a *race* between outoing transitions from s exits. That is, the probability of moving from s to s' in a single transition is the probability that the delay of going from s to s' finishes before the delays of any other outgoing transition (from s). We use an informal, graphical notation for indicating the states and transitions of a CTMC, for example, in Fig. 1.

3.2 Continuous Stochastic Logic

We use Continuous Stochastic Logic (**CSL**) [2], a stochastic extension of the Computational Tree Logic (**CTL**) that

allows the expression of a probability measure of the satisfaction of a temporal property in either transient or steady-state behaviours. The formulae of **CSL** are state formulae Φ with path formulae Ψ:

$$\begin{array}{lll} \Phi & ::= & true \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\bowtie p}\left[\,\Psi\,\right] \mid \mathcal{S}_{\bowtie p}\left[\,\Psi\,\right] \\ \Psi & ::= & \mathbf{X}\,\Phi \mid \Phi\,\mathbf{U}^{\mathrm{I}}\,\Phi \end{array}$$

where a ranges over a set of atomic propositions AP, $\bowtie \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, and I is an interval of $\mathbb{R}_{\geq 0}$.

Informally, path formula $\mathbf{X}\,\Phi$ is true on a path starting in s if Φ is satisfied in the next state following s in the path, whereas $\Phi_1\,\mathbf{U}^{\mathrm{I}}\,\Phi_2$ is true on a path ω if Φ_2 holds at some time instant in the interval I in a state s' in ω and at all preceding times Φ_1 holds. We additionally use the path operator (syntactic sugar): the *eventually* operator $\mathbf{F}$ (future) where $\mathbf{F}^{\mathrm{I}}\,\Phi \equiv true\,\mathbf{U}^{\mathrm{I}}\,\Phi$.

A transient formula $\mathcal{P}_{\bowtie p}\left[\,\Psi\,\right]$ is true in state s, denoted by $s \models \mathcal{P}_{\bowtie p}\left[\,\Psi\,\right]$, if the probability that Ψ is satisfied by the paths starting from state s meets the bound $\bowtie p$. A steady-state formula $\mathcal{S}_{\bowtie p}\left[\,\Psi\,\right]$ is true in a state s if the steady-state (long-run) probability of being in a state that satisfies Ψ meets the bound $\bowtie p$.

We use the PRISM probabilistic model checker [3], which allows us to leave the bound $\bowtie\ p$ unspecified. The probability is calculated in PRISM thus: $\mathcal{P}_{=?}\left[\,\Psi\,\right]$ and $\mathcal{S}_{=?}\left[\,\Psi\,\right]$. Additionally, PRISM allows for *experimentation*: the verification of an open formula, when the range, and step size of the variable(s) are specified. For example, a typical property is $\mathcal{P}_{=?}\left[\,\mathbf{F}^{\leq t}\phi\,\right]$, which delivers the probability that we can reach a state in which ϕ is true, within t units of time (*e.g.* hours or minutes). We typically consider hours as the unit of time and vary t from 1 to 48 (*i.e.* behaviour over the next 48 hours).

3.3 PRISM rewards

PRISM allows for the augmentation of models with *rewards* (or, equivalently, costs) that are associated with states or transitions. The model checker can analyse properties that relate to the expected values of these rewards by using the $\mathcal{R}$ operator, which works in a similar fashion to the $\mathcal{P}$ and $\mathcal{S}$ operators, except that it depends on the name of a reward structure. Reward structures defined (named) rewards on transitions or state. We employ rewards on transitions and *cumulative* and *steady-state* reward properties. A cumulative (reward) property has the form $\mathcal{R}\{\texttt{reward}\}_{=?}\left[\,\mathbf{C} \leq t\,\right]$, which corresponds to the reward (named `reward`) accumulated along *all* paths until t time units have elapsed. A steady state (reward) property has the form $\mathcal{R}\{\texttt{reward}\}_{=?}\left[\,\mathbf{S}\,\right]$, which corresponds to the (named) reward per unti time, in the long run.

3.4 PRISM language of reactive modules

There are several formalisms for specifying Markov processes based on rate transition matrix descriptions, state-transition graphs, amd guarded command languages such as Reactive Modules [4], which is the basis of the PRISM language. The PRISM language matches our requirements very well, as it supports processes and compositionality. See [3] for more details.

Processes are represented by modules consisting of action-labelled guarded commands (which denote transitions), and modules are composed over all common actions. Each module has the form: local variable declarations followed by a non-deterministic choice between guarded commands.

A guarded command has the form:
$[action]\ guard \rightarrow rate : update$
meaning the process makes a transition to a state described by the *update* at the given *rate* when the *guard* is true. In the update, if x is a variable, then x' denotes the value of x in the next state. The $action$ label is optional (*i.e.* events can be anonymous or named). An update may be a choice between two or more assignments, indicating by the $+$ operator, for example
$[action]\ guard \rightarrow rate_1 : update_1 + rate_2 : update_2$
meaning there is a race condition between the two updates. Transitions are synchronised when they have the same action labels, in which case the the rate of the synchronised transition is the *product* of all the individual rates. For example if one module contains
$[action_1]\ guard_1 \rightarrow rate_1 : x' = 2$
and it synchronises with a module that contains
$[action_1]\ guard_2 \rightarrow rate_2 : y' = 3$
and $guard_1$ and $guard_2$ are *true*, then in a next state, $x = 2$ and $y = 3$, and the rate of the transition to that state is $rate_1 \cdot rate_2$.

4 Overview of the Case Study Model

After experimentation with a number of different abstractions, we found the following as an ideal compromise between detail, efficiency of analysis, and ease of expression of key properties. Recall the the primary components of the system are *channels*, *frequencies*, *sites*, and *sectors*. Components are modelled by PRISM modules, and events that must be synchronised have the same action labels.

4.1 Channels

A **channel** is characterised by three parameters: whether it is receiver (Rx) or transmitter (Tx), the frequency, and the site reference. There are four possible states for a receiver/-transmitter: S for serviceable (green), F for faulty (red), M for under maintenance (blue) and E for external site failure (red). There is no reduced redundancy in a single channel (*i.e.* there is no amber for an individual channel).

We will not reason about individual channels directly: we employ a *counter abstraction* whereby a **pair of A and B channels** is represented by a single module and state labels indicate the *counts* of the constituent channels. For example, state (label) SS means that both A and B channels are serviceable, state SF means that one channel is serviceable and the other is faulty (note, the label notation is not positional). The CTMC for a pair of channels is given on the left-hand side of Fig. 1. States are colour coded to indicate servicability so whereas individual channels may be green/blue/red, channel pairs are green/amber/red[1].

1. A component that is under maintenance is not serviceable, therefore we have abstracted away from the "under maintenance" (blue) class.

4.2 Sites

The **site environment** is characterised by major physical events that cause a failure of the site (*e.g.* instrusion, powerline and backup generator failure, flooding) or minor events that mean the site is more likely to fail, but is still functioning (*e.g.* powerline failure but backup generators functioning). Minor events typically precede major (failure) events and so we have three states for the site environment: E0 for serviceable (green), E1 site minor event (amber), and E2 site major event (red).

A **site** is represented by the concurrent composition of three modules: the transmitters, the receivers, and the site environment. We denote a site by a triple (Tx,Rx,Env) consisting of the two channel pairs and a site environment. This means the notation for a site is positional: for example, site $(\mathsf{SF}, \mathsf{SS}, \mathsf{E0})$ is distinguished from $(\mathsf{SS}, \mathsf{SF}, \mathsf{E0})$. The former denotes a configuration where the transmitter is reduced redundancy and the receiver is serviceable, whereas the latter denotes a configuration where the transmitter is serviceable and the receiver is reduced redundancy.

States are labelled and classified by three colours: W (working) for serviceable (green), R for reduced redundancy (amber), and N for no-service (red). Fig. 1 is the CTMC for a channel pair and site environment with symbolic rates a, b, c, *etc.* A key aspect of the model is the interaction between the site environment and the channels: the transition between E1 and E2 in the site environment synchronises with any channel transition to state E (red arrows in Fig 1); that is, an external site event causes the channel to move to state E. Similarly, the (site environment) transition between E2 and E0 synchronises with the channels (reset) transition to SS (green arrows in Fig. 1). Note, not all configurations are reachable. For example, the configuration $(\mathsf{SS}, \mathsf{E}, \mathsf{E2})$ is not possible because of synchronisation on site failure: when a site failure occurs, both the transmitter and receiver synchronise on this event and move to (channel) state E.

4.3 Sectors

An *n-ary* **sector** is represented by its n constituent sites. Without loss of generality, we assume a sector with three sites. WWW denotes a serviceable sector (green), NNN is a no-service sector (red), and amber is for a reduced redundancy sector, which consists of all remaining states, *i.e.* the language defined by $L \setminus \{\mathsf{WWW}, \mathsf{NNN}\}$, where $L = (\mathsf{W}|\mathsf{N}|\mathsf{R})(\mathsf{W}|\mathsf{N}|\mathsf{R})(\mathsf{W}|\mathsf{N}|\mathsf{R})$. Note, this notation is positional. We assume the rates for events for transmitters and receivers are identical and if either the transmitter *or* receiver is no-service, then the entire site is no-service. The rates for events will usually differ from site to site; for example, for a given sector, the rate for an event e for the first site may be different from the rate for the same event e for the second site (in that sector).

For a given frequency, a sector is represented by the concurrent composition of its constituent sites, which typically varies between 2 and 5 sites. Note that sites within a sector are independent (*i.e.* there is no synchronisation).

Table 1 contains a summary of the (labels of the) states that are represented in a model with a ternary sector, using regular expression notation, *e.g.* '|' for disjunction and '$*$' for wildcard. Strictly speaking, the labels of states in a CTMC are the propositions that are true in that state. Here, we introduce a convenient labelling for states that indicates the properties of that state.

As example, the code snippet in Fig. 2 specifies the PRISM modules for the transmitters (Tx) and the environment for site X, in the context of rate declarations. State labels are represented by (local) integer variables `s0_X` and `env_X`, *e.g.* 2 for FF, *etc.* Note the last two transition choices in the transmitters module, labelled by `alarm_major_X` and `fix_X`, cause the synchronisation with the site environment.

4.4 Rates

The model is governed by seven rates, which we refer to as $a, \ldots, g$; these are indicated in in Fig. 1. Note there are two transitions from the repairing state: a *quick*, local repair that returns to the serviceable state, and a *slower* transition to the under maintenance state. The former reflects an error that can usually be fixed by a remote reboot. The latter reflects the fact it may take some time for an engineer to physically reach a site and/or repair the fault. Interviews with engineers indicated the ratio between these rates is typically about $3:1$.

Rate a indicates the failure rate of a *single* channel. Intuitively, it describes the transition of a channel from state S to state F (downwards arrows in the Fig. 1). Since state SS contains two channels that can individually and independently fail, the rate for transition $\mathsf{SS} \to \mathsf{SF}$ must be $2a$.

Rate b is the rate of a quick repair. It describes the transition of a channel from state F to state S (without passing through an M state). Interviews with engineers revealed that the time to repair a single channel and a pair of channels is the same, we use b (*not* $2b$) as the rate for transition $\mathsf{FF} \to \mathsf{SF}$.

Rate c is the rate for slow repairs and describes the transition of a channel from state F to state M. Events of this kind are always in a race condition with b-rated events. In order to reflect the $1:3$ ratio between quick and slow repairs, c is defined as $b/3$.

Similarly, rate d is the duration of a repair of an under maintenance channel (M), *i.e.* a transition of a channel from state M to state S.

Rates e and g are the rates for external site events and site failures, respectively, and g is the rate of (external) site repair.

4.5 Which rates?

Actual rates depend on how we intend to use the model, *e.g.* to evaluate how the system architecture is *designed* to meets service requirements, or how the system architecture *actually* meets service requirements, when in operation. For the former, we instantiate rates with those derived from typical MTBF values, interviews with engineers, and inspection of the business cases. For the latter, we instantiate rates according to historical, field data. Since this is more novel challenging, for the remainder of this paper we focus on analysis based on rates inferred from historical data.

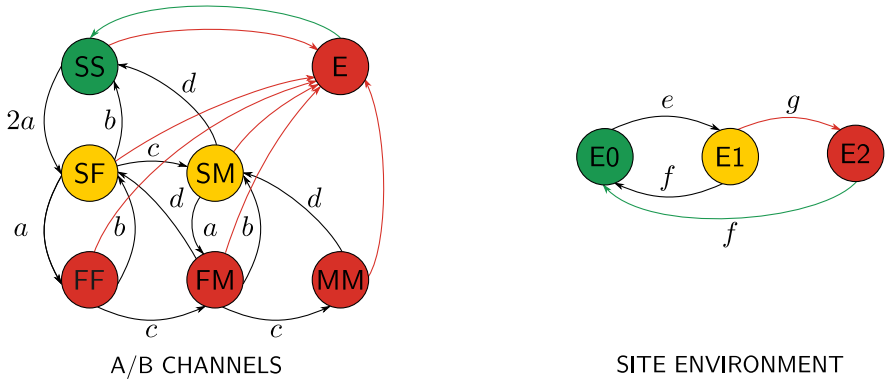

Fig. 1. CTMC for channel pair and site environment. Synchronisation on red and green transitions. Rates: a = channel failure, b = channel quick repair, c = channel slow repair, d = channel under maintenance repair, e = site minor event, f = site repair, g = site failure.

TABLE 1
State labelling and colour coding.

Component	Colour	States	Description
channel	green	S	serviceable channel
	blue	M	under maintenance channel
	red	F	faulty channel
	red	E	site failure
channel pair	green	SS	serviceable AB
(A,B)	amber	SF|SM	reduced redundancy AB
	red	FF|FM|MM|E	no-service AB
site	green	SS, SS, E0	W serviceable site
(Tx,Rx,Env)	amber	SS, SS, E1	R reduced redundancy site
	amber	SF,(SM|SF|SS),(E0|E1)	R reduced redundancy site
	amber	SM,(SM|SF|SS)(E0|E1)	R reduced redundancy site
	amber	(SM|SF|SS), SF(E0|E1)	R reduced redundancy site
	amber	(SM|SF|SS), SM(E0|E1)	R reduced redundancy site
	red	E, E, E2	N no-service site
	red	(FF|FM|MM) ∗ ∗	N no-service site
	red	∗(FF|FM|MM)∗	N no-service site
ternary sector	green	WWW	serviceable sector
(site,site,site)	amber	all other combinations	reduced redundancy sector
	red	NNN	no-service sector

5 Inferred Rates from Field Data

The company gave us access to their SAP incident ticketing system, which they employ for long term storage of logged failures. The data logs record failure occurrences and repair durations, as well as a textual description, which allow us to categorise events. Inference of rates was by manual inspection, sector by sector, for nominated time periods. Note, longer term, we aim to influence the design of readouts and tickets, and subsequently to automate the inference process.

As an example, we give results for one sector, which we call *FIR*, over a one year period: February 2012 to February 2013. The data included 61 alarms, of which 24 were site events. From this data we calculated mean inter-failure times, which we then used to define failure rates (namely rates a, e and g), and we calculated and used repair duration times and mean repair duration times to define repair rates. The results are reported in Table 2.

TABLE 2
Inferred rates from historical event data for sector FIR.

Rate	Inferred value
Mean inter-failure time	452 h
Mean repair time	18 h
Response	57 m
Site event	1107 h
Percentage of quick repairs	15
Site failure	1 every 11.33 years

Examination of the field data confirmed the inferred rates are of the expected orders of magnitude and also our assumption (as told to us during interviews with engineering staff) that the duration of repairs is independent of the number of channels (requiring repair). However, analysis raised some issues that require further consideration. First, some events were impossible to detect by analysing the chosen data set. For example, the textual descriptions for repair events did not specify whether an event was a quick or a

```
// Ratio quick repairs to slow repairs
const double x=3;
// Channels – mean times in hours
const double failure=..
const double repair=..
const double qrepair=..
// Site environment
const double event=..
const double site_failure=..
const double fix_e_event=..
// Rates
const double a=1/failure;
const double b=1/qrepair;
const double c=b/x;
const double d=1/repair;
const double e=1/event;
const double f=1/fix_e_event;
const double g=1/site_failure;

module Site_Tx_X  // A/B channels
  // 0=SS, 1=SF, 2=FF, 3=SM, 4=FM, 5=SM, 6=E
  s0_X : [0..6];

  [] s0_X=0 -> 2*a:(s0_X'=1);
  [] s0_X=1 -> a:(s0_X'=2);
  [quick_0_X] s0_X=1 -> b:(s0_X'=0);
  [] s0_X=1 -> c:(s0_X'=3);
  [quick_0_X] s0_X=2 -> b:(s0_X'=1);
  [] s0_X=2 -> c:(s0_X'=4);
  [alarm_0_X] s0_X=3 -> a:(s0_X'=4);
  [repair_0_X] s0_X=3 -> d:(s0_X'=0);
  [quick_0_X] s0_X=4 -> b:(s0_X'=3);
  [] s0_X=4 -> c:(s0_X'=5);
  [repair_0_X] s0_X=4 -> d:(s0_X'=1);
  [repair_0_X] s0_X=5 -> d:(s0_X'=3);
  [alarm_major_X] true -> 1:(s0_X'=6);
  [fix_X] s0_X=6 -> 1:(s0_X'=0);
endmodule

module Site_env_X  // Site environment
  env_X : [0..2];  // 0=E0, 1=E1, 2=E2

  [] env_X=0 -> e:(env_X'=1);
  [alarm_major_X] env_X=1 -> g:(env_X'=2);
  [fix_X] env_X=1 -> f:(env_X'=0);
  [fix_X] env_X=2 -> f:(env_X'=0);
endmodule
```

Fig. 2. PRISM specification of transmitters and site environment modules, for site X.

slow repair. Therefore, in order to infer the ratio between rates b and c, we assumed that repair events with a duration *greater* than 2 hours were slow repairs. Second, rare events such as site failures did not occur in the time span covered by the data set; we had to inspect data from previous years to find an occurrence. Third, we identified two classes of events that may require a different representation model:

- dependent events such as the contemporaneous failure of both the A and B channels, and
- deterministic events such as scheduled maintenance.

We will return to these issues in Section 12.2.

6 Temporal Logic Properties

We now turn our attention to **CSL** properties for analysis, considering both *steady-state* and *transient* properties. We begin with the underlying propositions.

TABLE 3
Atomic propositions for status of channel pairs, sites and sectors.

$$
\begin{aligned}
\texttt{serviceable\_chan}(c) &= (c = \mathsf{SS}) \\
\texttt{serviceable\_env}(e) &= (e = \mathsf{E0}) \\
\texttt{serviceable\_site}(s) &= \texttt{serviceable\_chan}(\mathrm{Tx}_s) \\
&\quad \wedge \texttt{serviceable\_chan}(\mathrm{Rx}_s) \\
&\quad \wedge \texttt{serviceable\_env}(\mathrm{Env}_s) \\
\texttt{serviceable\_sector}(A) &= \textstyle\bigwedge_{s \text{ site in } A} \texttt{serviceable\_site}(s) \\
\\
\texttt{rr\_chan}(c) &= (c = \mathsf{SF}) \vee (c = \mathsf{SM}) \\
\texttt{rr\_env}(e) &= (e = \mathsf{E1}) \\
\texttt{rr\_site}(s) &= \neg(\texttt{serviceable\_site}(s) \\
&\quad \vee \texttt{noservice\_site}(s)) \\
\texttt{rr\_sector}(A) &= \textstyle\bigvee_{s \text{ site in } A} \texttt{rr\_site}(s) \\
\\
\texttt{noservice\_chan}(c) &= (c = \mathsf{FF}) \vee (c = \mathsf{FM}) \\
&\quad \vee (c = \mathsf{MM}) \vee (c = \mathsf{E}) \\
\texttt{noservice\_env}(e) &= (e = \mathsf{E2}) \\
\texttt{noservice\_site}(s) &= \texttt{noservice\_chan}(\mathrm{Tx}_s) \\
&\quad \vee \texttt{noservice\_chan}(\mathrm{Rx}_s) \\
&\quad \vee \texttt{noservice\_env}(\mathrm{Env}_s) \\
\texttt{noservice\_sector}(A) &= \textstyle\bigwedge_{s \text{ site in } A} \texttt{noservice\_site}(s)
\end{aligned}
$$

6.1 Atomic propositions

The atomic propositions indicate the status (*i.e.* level of service) of channel pairs, sites, *etc.* The levels of service: servicable, reduced redundancy, and no-service, are defined in Table 3.

6.2 Steady state properties

Steady state properties express *long run* behaviour[2] and we use these properties for *validation* of the model. Typically we examine steady state behaviour for a given sector, computing the likelihood to be in a *service* state, a *reduced redundancy* state, or a *no-service* state, in the long run. Namely, we consider three steady state properties:

$$
\begin{aligned}
&\mathcal{S}_{=?}\left[\,\texttt{serviceable\_sector}(A)\,\right] \\
&\mathcal{S}_{=?}\left[\,\texttt{rr\_sector}(A)\,\right] \\
&\mathcal{S}_{=?}\left[\,\texttt{noservice\_sector}(A)\,\right]
\end{aligned}
$$

6.3 Transient properties

Transient properties express the probability of reaching a state that satisfies a proposition *within a period of time*. For our analysis, the crucial question is: what is the the likelihood of reaching *no-service* in a given *sector* within time t. This is expressed, for sector A, by the transient property

$$
\mathcal{P}_{=?}\left[\,\mathbf{F}^{\leq t}(\texttt{noservice\_sector}(A))\,\right] \tag{1}
$$

By *experimentation* in PRISM with property (1) we can consider different instantiations of t, to plot how the likelihood changes over over time. But there is another parameter to consider: the *state* from which we compute the likelihood. (Note, hereafter we use configuration and state interchangeably.) In standard model checking, the given state is, by

2. Note, in a CTMC, long run behaviour is distribution over states.

default, the initial state of the system. In our case, this would be the all-green configuration (serviceable channels, sites, sectors, *etc.*). However, we are considering a deployed system in which failures have occurred and the interesting cases are the *degraded*, amber configurations. Specifically, once we have reduced-redundancy, we require to quantify the criticality of the situation and take informed decisions – for example, do I need to fix a fault now, or can I wait? And if I can wait, for how long should I wait?

6.4 Example results

For example sector *FIR*, steady state analysis results are given in the left hand column in Table 4, indicating that in the long run, the sector is serviceable for the majority of time (over 88%). This accords with the experiences of the engineers we interviewed. For further validation, we analysed the historical data for that sector (over one year), to calculate the percentage time spent in a *service* state, etc. These results are indicated in the right hand column in Table 4. As can be seen, the model results align very well with actual performance in that sector over a one year period. For completeness, we give detailed results for the 52 configurations with probability $> 10^{-3}$ in Fig. 3a, noting the log scale for probabilities and use of shades of green to indicate degree of degradation/reduced-redundancy.

For transient property analysis, recall we require to choose a state (from which to perform the analysis). For sector FIR there are $389,017$ states, of which 1 is fully serviceable (WWW), $166,375$ are no-service (NNN), and $222,641$ are degraded, reduced-redundancy configurations. The degraded configurations we examined for the FIR sector, with three sites that we call A, B, and C, are given in Table 5. We refer to Table 1 for the definitions of W, R and N. Observe that both N and R can be the result of many different site configurations[3], so we randomly selected one for each occurrence of R and N.

Figures 3b, 3c, and 3d give the results for the probability of reaching a no-service configuration, from different degraded configurations, over a time interval of 48 hours.

We are considering a critical service (in a safety-critical domain) and so we expect probabilities to be very low. However, observe the orders of magnitude difference on the Y axis. In Fig. 3b, the scale is 10^{-7}, whereas in Fig. 3c, the scale is 10^{-4}, and in Fig. 3d, the scale is 10^{-3}. Also, observe that in Fig. 3b the steepest trajectory is WWN, which contains one no-service site, and in Fig. 3d, the trajectory with highest probability, RNN, has two no-service sites. However, in the same Figures, WNN also contains two no-service sites, but one serviceable site and the overall probability of service failure is constantly low.

Following similar analysis of different sectors with different topologies, we observed the contribution of additional sites increases service availability, however that increase is inversely proportional to the number of additional sites. The example in Fig. 4a illustrates this: the difference between 3- and 4-ary sites is negligible. Overall, these results show that site redundancy (*i.e.* sector topology) is the most crucial factor affecting the behaviour of the system and we also conclude the system is not sensitive to the number n of sites, when $n > 3$. This implies the plots for the ternary site given in Fig. 3b to 3d are good approximations for sectors with more sites.

Finally, we remark that channel redundancy *within* a site is a contributory factor to overall behaviour. When both channels A and B are serviceable, *i.e.* the site is W, then this redundancy guarantees safe service levels in the time frame $0 - 48$ hours, even in the extreme configuration in which only one site is in configuration W. For examples of this, see Fig. 3c, in which the plot for WRR is effectively flat, and similarly in Fig. 3d, in which the plots for WRN and WNN are also effectively flat.

7 Transient Properties for Decision Making

We now show, with reference to an example, how predictions of no-service can inform operational decision making. Consider the following scenario:

1) the current configuration of the system is RRR,
2) the system safety threshold (*i.e.* probability of no-service) is 4×10^{-3}, and
3) the mean repair time is 20 hours.

We predict the behaviour of the system by checking the transient property of reaching no-service as explained in the previous section: the plot, from the current configuration, is indicated with the solid line in Fig. 6a. We remark that the solid line denotes the expected behaviour *if no assumptions are changed in the system*, *i.e.* if we assume the current failure and repair rates. Now consider the shaded area in Fig. 6a, which indicates the probabilities above the safety threshold. The prediction shows that the system is likely to become unsafe after 20 hours. We reach the conclusion that within 20 hours, we want to be on *another trajectory*, which is below the system safety threshold. We can do this by altering one or more rates so as to, in effect, transition to a more favourable configuration in an alternative CMTC, *i.e.* in one that is structurally the same but has different transition rates. For example, we could ensure that maintenance on one of the no-service sites is prioritised, effectively pushing down the mean repair time to 15 hours. In this case, the expected behaviour of the system over the next 48 hours improves because the system becomes unsafe only after 34 hours instead of 20 hours. This is shown in Fig. 6a with the dashed line. Now consider the behaviour from a configuration with one serviceable site, WRR; this is the configuration of the current system (RRR) after the site repair is successfully completed. The expected behaviour is indicated by the dotted line in Fig. 6a. As can be seen, configuration WRR is much safer because within the time frame, the safety threshold is never reached.

Further, assume we choose to prioritise site maintenance and the one site is repaired after 20 hours (a random value taken by the exponential variable when the mean repair time is 15 hours). The transient property never reaches the system safety threshold, as shown by the solid line in Fig. 6b. The dotted line shows the original trajectory: the probability of no-service if the repair is never performed. The discontinuity indicates exactly when the current state

3. For example, R could be SF, SM, E0 or SF, SS, E1; N could be E, E, E2 or FF, SS, E0 or E, E, E2.

TABLE 4
Comparison of model long run behaviour and manual analysis of historical data for sector FIR.

Status	Proposition	Model result	Result from historical data
serviceable	`serviceable_sector`(FIR)	88.46%	86.54%
reduced-redundancy	`rr_sector`(FIR)	11.53%	13.56%
no-service	`noservice_sector`(FIR)	10^{-8}%	0.00%

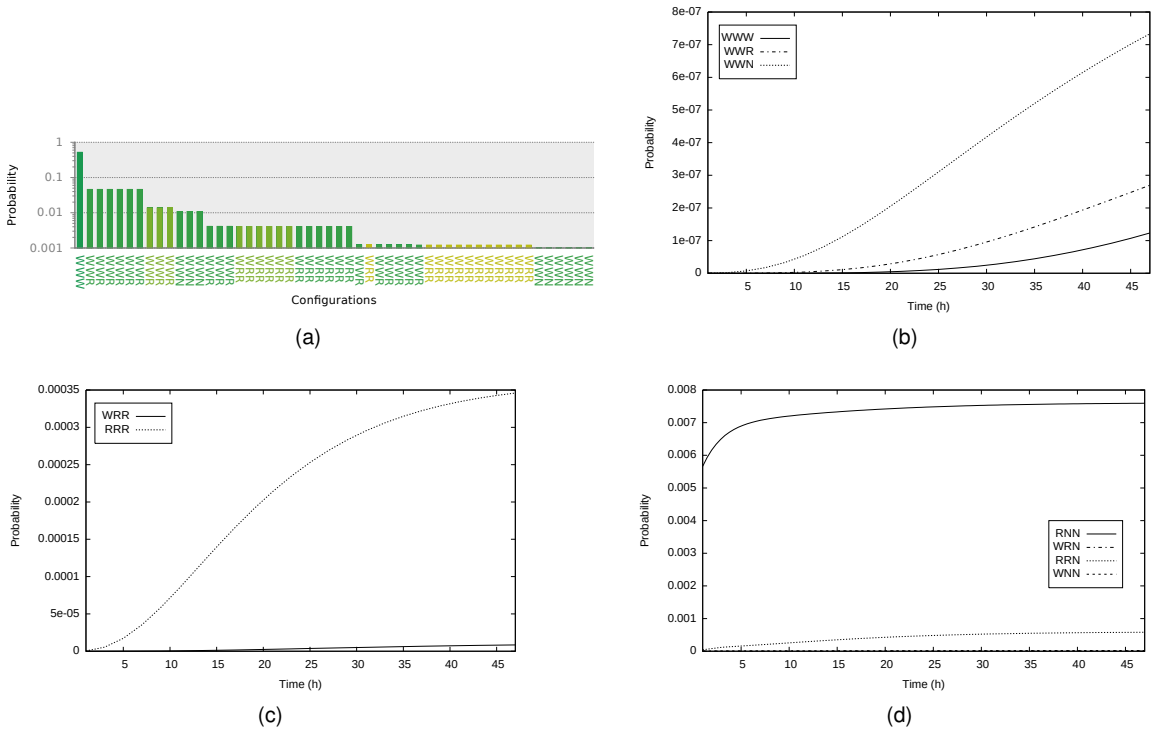

Fig. 3. Analysis results for sector FIR: (a) steady-state distribution of the 52 most probable configurations; (b) prediction of no-service from WWW, WWR and WWN configurations; (c) prediction of no-service from WRR and RRR configurations; (d) prediction of no-service from RNN, WRN, RRN and WNN configurations.

TABLE 5
Selected degraded configurations for sector FIR.

Site A	Site B	Site C
W	W	N
W	N	N
W	W	R
W	R	R
R	R	R
R	R	N
R	N	N

of the system is updated to WRR (at time 20h) because the site has became serviceable.

Fig. 5 is a pictorial representation of decision making; transitions indicate component failures. On the left we have the initial (green) state and on the right the (red) no-service states. The (amber) degraded configurations are the majority of states in between these two extremes: the dashed edge indicates the decision to make a discrete transition from one degraded confituration to another (more favourable) one.

Note, we can employ a similar approach to predict the behaviour of the system *after* specific events occur, such as scheduled maintenance or rare site failures (since they have such a small influence over transient probabilities, within a short time frame). In such cases, we are moving the trajectory *up*, instead of *down* at the the discontinuity, *i.e.* we are increasing likelihood of no-service.

It may be tempting to consider as more favourable states those that are far (in terms of the number of transitions in the *shortest* path) from a no-service configuration, in the belief that configurations closer to a no-service configuration are more degraded than more distant configurations. But this is misguided, because the *length* of the path is possibly irrelevant. For example, from one amber configuration it may require only two events to reach a no-service configuration, yet both those events are very rare. On the other hand, from another amber configuration, it may take more discrete events (*i.e.* failures) before we reach no-service, yet all of them may be quite likely. So, in the former case, the probability of reaching no-service within a fixed time may well be lower than in the latter case, depending on choice of time interval.

We illustrate with an example. The distance to no-service is at most 6, because every configuration is at most 3 steps

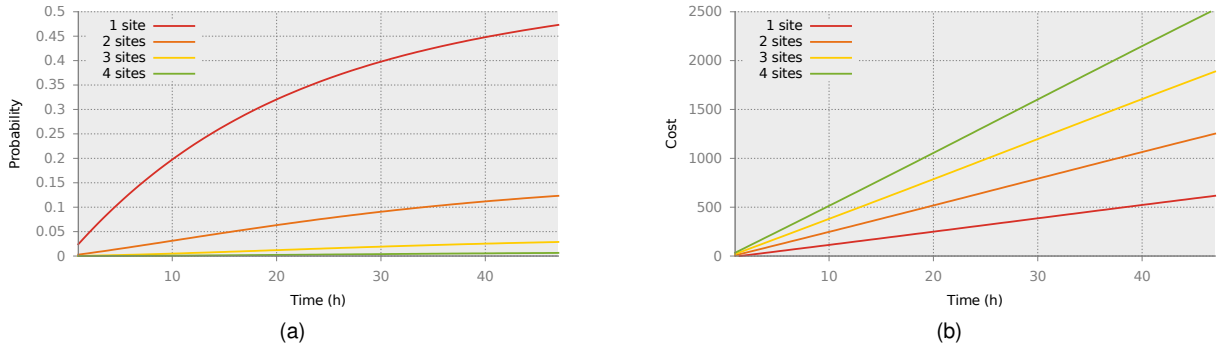

Fig. 4. Comparison of 1- to 4-ary sector topologies over 48 hours: (a) probability of no-service; (b) maintenance cost. Each site configuration is SM, SM, E1 $\in$ R.

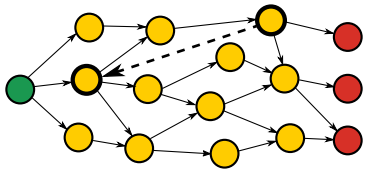

Fig. 5. Changing configuration after analysis of service availability.

away from a site failure. For example, even configuration WWW becomes NNN in 6 steps when, in each site, transitions E0 $\rightarrow$ E1 $\rightarrow$ E2 take place.

Consider the following reduced-redundancy configurations: (SS, FS, E0)(SS, FS, E0)(SS, FF, E0) $\in$ RRN, and (SS, SS, E0)(SS, SS, E0)(E, E, E2) $\in$ WWN. The first one is only two transitions away from no-service, while the second one is four transitions away. This is because two channel failures FS $\rightarrow$ FF and two site failures E0 $\rightarrow$ E1 $\rightarrow$ E2 are needed to take place in order to reach NNN in the first and in the second cases, respectively. But the first is not *more* degraded because the probability of reaching a no-service configuration within 48 hours is 2.251×10^{-5} and 3.304×10^{-4} for the first and the second configurations, respectively. The second configuration is more degraded – despite being more distant from no-service than the first one: an example of how can often be misleading in a probabilistic setting intuition.

8 Envelopes of Behaviour for Transient Properties

When we require to reason about behaviour from a *given* degraded configuration, we know exactly the configuration of all the component sites. And if the we do not know the exact configuration of all the components, we can simply select random representatives, as above. An interesting question is can we *quantify* the effect of the choice of status of the lowest level components on the analysis? To answer this we consider how to identify, for a given transient property, upper and lower probability bounds induced by the different possible combinations of the lower level component states; in our case, this refers specifically to the status of the sites. These bounds will allow us to define *envelopes of behaviour*, for a property. We illustrate through examples of the likelihood of reaching no-service, within 48 hours, when applied to the FIR sector[4].

FIR is a ternary sector that has 25 possible degraded, or reduced-redundancy configurations.[5] This can be reduced to 8 cases, by symmetry. For each configuration, we define the lower bound to be the lower bound for the most degraded site, and conversely we define upper bound to be the upper bound for least degraded site. The lower/upper bounds for the most/least degraded sites are found by analysis of all the possible component sites. We illustrate by example. Assume the first two sites are W, then perform transient property analysis (for the *sector*) for all possible R and N configurations. For the former, there are 17 cases to consider, which reduce to 11 after removing symmetric cases; for the latter, there are 37 to consider, which reduce to 31. Results are shown in Figures 7a (WWR) and 7b (WWN); in each, for comparison, we also give the result for (SS, SS, E0), which of course is not degraded. To ease interpretation, configurations are colour coded according to the level of degradation: recall, green means no degradation, while red means consistent degradation. It is not suprising that clusters occur, depending on the site environment: all the configurations with E1 in clusters C^1_{R} and C^1_{N} are considerably more degraded than the configurations with E0 in cluster C^0_{R} and C^0_{N}, and no-service configuration E, E, E2 $\in C^2_{\mathsf{N}}$ is even further degraded.

We use these bounds to define the lower and upper bounds of the envelopes of behaviour for a property p, using the notation $\downarrow_p$ and $\uparrow_p$ to denote lower and upper, respectively. For instance, the lower bound for configuration WNN, WNN $\downarrow_p$, is obtained by selecting the lower bound for both N sites, namely SS, FF, E0. The lower bound for WWR is defined by taking the lower bound for R, namely SS, FS, E0.

Details of the site configurations for each bound are given in Fig. 6, with the corresponding probabilities of no-service over 48 hours shown in Fig. 8. In the latter, for simplicity, we omit the subscript p. Note that the red shaded area between RNN $\uparrow$ and RNN $\downarrow$ indicates the envelope for any configuration in the form RNN.

4. This is the property defined by (1).

5. Each site can take one of 3 forms ($3 \times 3 \times 3$ configurations), from which we remove WWW and NNN.

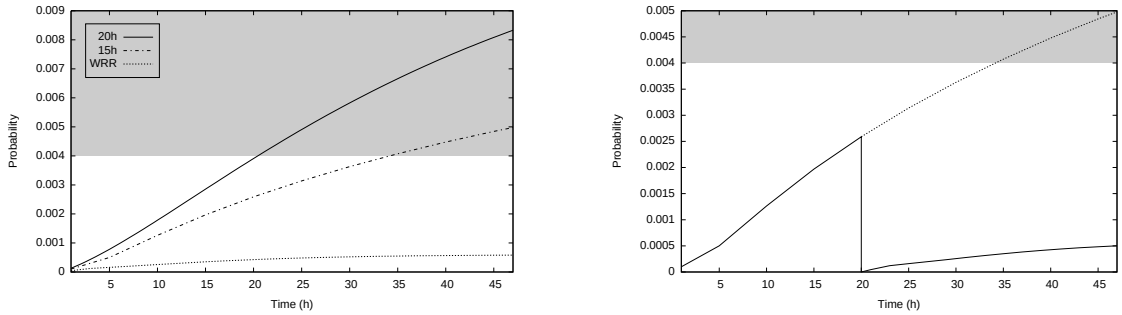

Fig. 6. Transient property for service availability: (a) comparison of different configurations and repair rates and system safety threshold; (b) before and after discrete transition to a new state, in context of 4×10^{-3} system safety threshold.

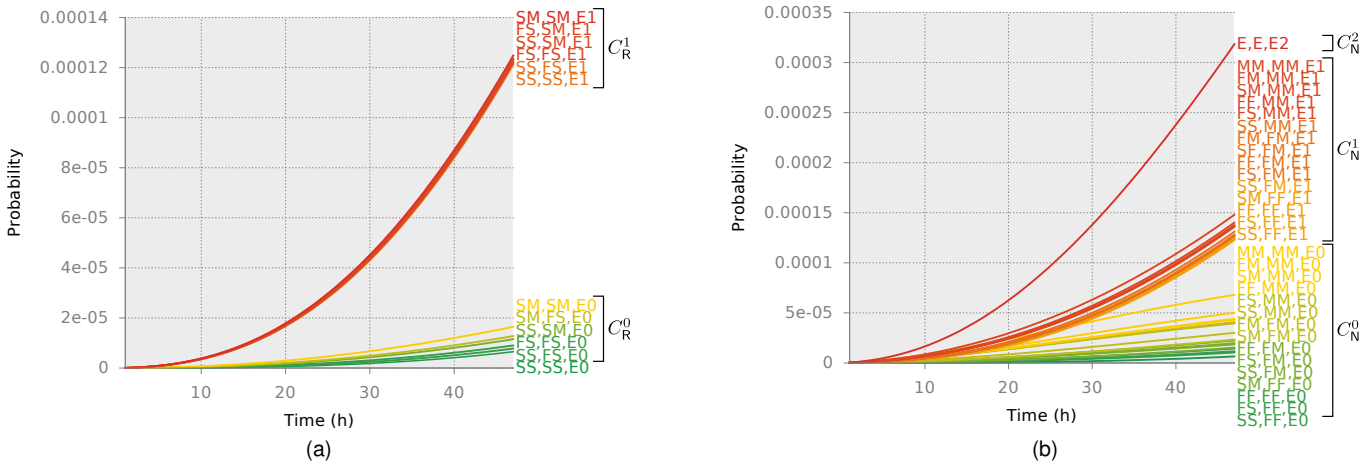

Fig. 7. Likelihood of no-service within 48 hours for all the WWR (a) and the WWN (b) configurations.

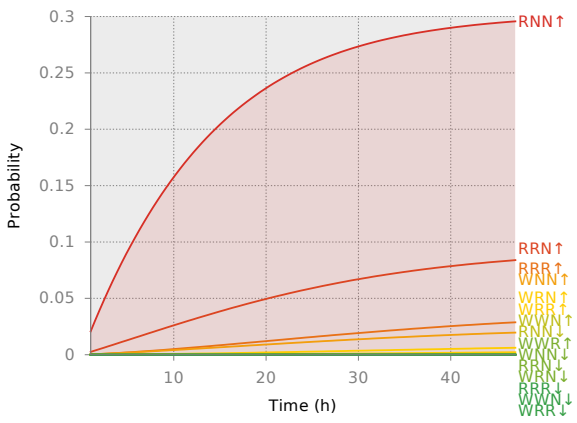

Fig. 8. Envelopes of behaviour for property p, for degraded configurations.

TABLE 6
Site configurations for bounds for property p.

Bound	Site A	Site B	Site C
RNN $\uparrow_p$	SM,SM,E1	E,E,E2	E,E,E2
RRN $\uparrow_p$	SM,SM,E1	SM,SM,E1	E,E,E2
RRR $\uparrow_p$	SM,SM,E1	SM,SM,E1	SM,SM,E1
WNN $\uparrow_p$	SS,SS,E0	E,E,E2	E,E,E2
WRN $\uparrow_p$	SS,SS,E0	SM,SM,E1	E,E,E2
WRR $\uparrow_p$	SS,SS,E0	SM,SM,E1	SM,SM,E1
WWN $\uparrow_p$	SS,SS,E0	SS,SS,E0	E,E,E2
RNN $\downarrow_p$	SS,FS,E0	SS,FF,E0	SS,FF,E0
WWR $\uparrow_p$	SS,SS,E0	SS,SS,E0	SM,SM,E1
WNN $\downarrow_p$	SS,SS,E0	SS,FF,E0	SS,FF,E0
RRN $\downarrow_p$	SS,FS,E0	SS,FS,E0	SS,FF,E0
WRN $\downarrow_p$	SS,SS,E0	SS,FS,E0	SS,FF,E0
RRR $\downarrow_p$	SS,FS,E0	SS,FS,E0	SS,FS,E0
WWN $\downarrow_p$	SS,SS,E0	SS,SS,E0	SS,FF,E0
WRR $\downarrow_p$	SS,SS,E0	SS,FS,E0	SS,FS,E0
WWR $\downarrow_p$	SS,SS,E0	SS,SS,E0	SS,FS,E0

8.1 Limitations

In this case study we have been able to assume symmetries in configurations. However, if for example, the transmitters and receivers have different rates, then when identifying bounds for R and N would have to consider all 17 WWR configurations (instead of 11) and all 37 WWN configurations (instead of 31). Additionally, if rates vary across sites, then analysis has to be repeated independently for each site.

Another possible shortcoming is that, in some cases, the envelope may be too broad. For example, consider configuration RNN, with envelope bounds RNN $\downarrow_p$ and RNN $\uparrow_p$ (as indicated in Figure 8). That means the proba-

bility of no-service at 48 hours may take any value in the range $[2.54 \times 10^{-4}, 2.96 \times 10^{-1}]$. If this is considered too broad, we may specify intermediate bounds for different sub-classes of RNN configurations (*i.e.* other than the worst and best case scenarios). For example, upper and lower bounds for RNN configurations with all site environments set to E1 are (SM,SM,E1)(MM,MM,E1)(MM,MM,E1) and (SS,SS,E1)(SS,FF,E1)(SS,FF,E1) respectively; this can be confirmed by inspecting the plots in Figures 7a and 7b.

9 Recoverability and Survivability

So far, we have considered properties that define the likelihood of reaching a no-service state from degraded (*i.e.* reduced-redundancy) configurations. Now, we turn our attention to properties *after reaching* a no-service state. We consider two stochastic properties: *recoverability* and *survivability*, as proposed by Cloth and Haverkort in [5]. Both properties refer to behaviours *after* a disaster has occurred – in our case the "disaster" is reaching a no-service state. As before, we refer to examples taken from the FIR sector. We also write `noservice` as a shorthand for `noservice_sector`(FIR).

9.1 Recoverability

Recoverability is the probability of recovering service within time bound t. In our model, this is expressed in **CSL** by:

$$\mathcal{P}_{=?}\left[\,\mathbf{F}^{\leq t}(\neg\texttt{noservice})\,\right] \tag{2}$$

We denote this property by r and give results over time interval $t \leq 48$ h in Fig. 9. The state(s) from which we perform analysis are combinations of the bounds of the clusters[6] for N identified in Fig. 7b.

Observe the envelope of behaviour for any no-service configuration is given by:

$$\mathsf{NNN}\uparrow_r = (\mathsf{SS},\mathsf{FF},\mathsf{E1})(\mathsf{SS},\mathsf{FF},\mathsf{E1})(\mathsf{SS},\mathsf{FF},\mathsf{E1}) \text{ and}$$
$$\mathsf{NNN}\downarrow_r = (\mathsf{E},\mathsf{E},\mathsf{E2})(\mathsf{E},\mathsf{E},\mathsf{E2})(\mathsf{E},\mathsf{E},\mathsf{E2})\ .$$

Generally, the patterns are preserved, *i.e.* SS,FF,E0 is less degraded than MM,MM,E0. However, it interesting to observe that configurations with E0 take longer to recover than configurations with E1, which is the opposite behaviour we observed for N sites in reduced redundancy configurations.

Note that the property r (trivially) evaluates to (probability) 1, for any time bound, when the initial state is a reduced-redundancy configuration, this is because in that state $\neg$`noservice` already holds.

9.2 Survivability

Survivability is the ability of a no-service configuration to recover service, in a timely manner and within a given probability bound. This is a subtle elaboration on property (2) and expressed in **CSL** by

$$\texttt{noservice} \Rightarrow \mathcal{P}_{\leq q}\left[\,\mathbf{F}^{\leq t}(\neg\texttt{noservice})\right]$$

There are two free variables, a time bound t and a probability q. If we inspect Figure 9, for a particular (no-service) configuration, we can conclude the configuration is *survivable* for all points (t, q) on and below the curve, whereas the points above the curve indicate time bound/probability pairs for which the system is *not survivable*. For example, the plot for (MM,MM,E0)(MM,MM,E0)(MM,MM,E0) indicates the configuration is not recoverable within 10 hours with probability greater than 0.8, therefore it is not survivable. If we choose pair $(40, 0.9)$ instead, the configuration is recoverable (the point lays below the curve), thus survivable. Note we can conclude the *system* is survivable for $(48, 0.9)$ because all the configurations are survivable for this pair (this follows from the lower bound $\mathsf{NNN}\downarrow_r$).

6. For example, the upper bund of cluster C^1_{N} is MM, MM, E1.

10 Cost analysis with transition rewards

We may analyse *costs* of particular behaviours, and make decisions based on those costs, using the facility in PRISM to specify rewards and perform analysis of reward-based properties. As an example, we associate a cost with each transition in the model representing a maintenance intervention, and then reason about (*i.e.* forecast) the expected maintenance costs over one month period, for a given configuration, using a *cumulative reward* property, as follows.

We augment our model with the reward structure for site X given in Fig. 10. Each transition is assigned a cost – most have no precondition (*i.e.* the condition is simply `true`), but a condition is used to disambiguate the two transitions labelled by `fix_X`. For confidentiality reasons, we refer here to costs that are fictional but reflect actual proportions: `q` $= 10$, `r` $= 100$, `s0` $= 100$, `s1` $= 3000$. The cumulative reward property is

$$\mathcal{R}\{\texttt{cost}\}_{=?}\,[\,\mathbf{C} \leq 730\,]$$

where 730 is the time bound expressed in hours. Results for 19 configurations are given in Table 7. Note the maintenance cost of the more degraded configurations are higher than those for less degraded configurations. This is because when considering short time windows, maintenance interventions are more likely to be scheduled in more degraded configurations.

Cost analysis can be used along with safety analysis for decision making as described in Section 7. We note that while total maintenance costs increase linearly with the number of sites (when changing sector topology), as indicated in Fig. 4b, where each site is in the upper bound $\mathsf{SM},\mathsf{SM},\mathsf{E1} \in \mathsf{R}$ configuration, recall the contribution to overall safety, from each site, decreases exponentially with the number of sites, as indicated in Fig. 4a

Finally, we note that the reward structure defined in Fig. 10 also allows for the analysis of steady state properties *i.e.* the cost in the long-run. The corresponding formula is

$$\mathcal{R}\{\texttt{cost}\}_{=?}\,[\,\mathbf{S}\,]$$

which for the example FIR sector evaluates to 6.93. This represents the expected long-run cost rate per unit of time (one hour in our model), and is independent of the initial configuration.

This concludes our analysis. In the next section we comment briefly on the model implementation and then we reflect on our overall approach.

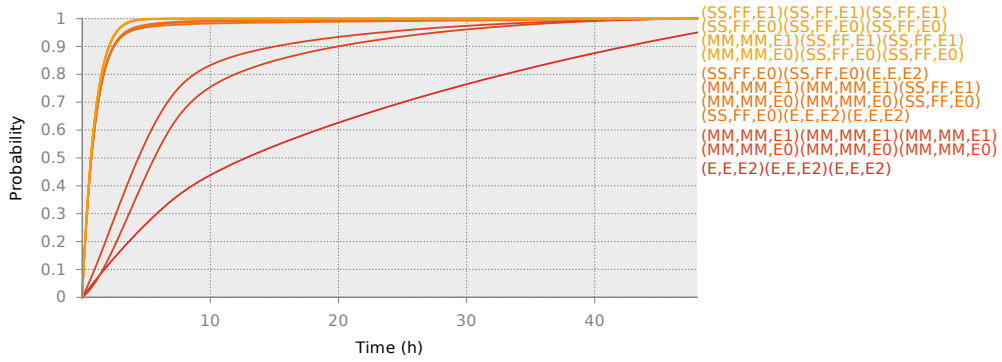

Fig. 9. Recoverability of selected no-service configurations within 48 hours.

```
rewards cost
  // Transmitter
  [quick_0_X] true: q;[repair_0_X] true: r;
  // Receiver
  [quick_1_X] true: q;[repair_1_X] true: r;
  // Site environment
  [fix_X] e_X=1: s0;[fix_X] e_X=2: s1;
endrewards
```

Fig. 10. PRISM specification of `cost` reward structure for site X.

TABLE 7
Expected monthly maintenance cost for 19 configurations

Site A	Site B	Site C	Monthly cost
E,E,E2	E,E,E2	E,E,E2	13448.19
SM,SM,E1	E,E,E2	E,E,E2	11851.93
SS,SS,E0	E,E,E2	E,E,E2	10558.47
SM,SM,E1	SM,SM,E1	E,E,E2	10255.67
SS,SS,E0	SM,SM,E1	E,E,E2	8962.21
SM,SM,E1	SM,SM,E1	SM,SM,E1	8659.41
SS,SS,E0	SS,SS,E0	E,E,E2	7668.75
SS,SS,E0	SM,SM,E1	SM,SM,E1	7365.95
SS,SS,E0	SS,SS,E0	SM,SM,E1	6072.49
SS,FF,E0	SS,FF,E0	SS,FF,E0	4964.70
SS,SF,E0	SS,FF,E0	SS,FF,E0	4933.75
SS,SF,E0	SS,SF,E0	SS,FF,E0	4902.81
SS,SS,E0	SS,FF,E0	SS,FF,E0	4902.81
SS,SF,E0	SS,SF,E0	SS,SF,E0	4871.86
SS,SS,E0	SS,SF,E0	SS,FF,E0	4871.86
SS,SS,E0	SS,SF,E0	SS,SF,E0	4840.92
SS,SS,E0	SS,SS,E0	SS,FF,E0	4840.92
SS,SS,E0	SS,SS,E0	SS,SF,E0	4809.97
SS,SS,E0	SS,SS,E0	SS,SS,E0	4779.02

11 Implementation

To make the models accessible to a variety of users, we developed a multi-platform web app, as illustrated in Fig. 11 running on an Android tablet. The system is a client-server architecture, implemented in Node.js[7] that relies on remote PRISM instances for heavyweight computations. The web app supports simple instantiation, using sliders, of model parameters such as rates channels and sites, sector topology, percentage of quick repairs, and current configuration. Default values are provided. Analysis results, displayed on the device, are in the formats as given in this paper (*e.g.* graphs or bar charts) and PRISM raw textual output.

7. https://nodejs.org/

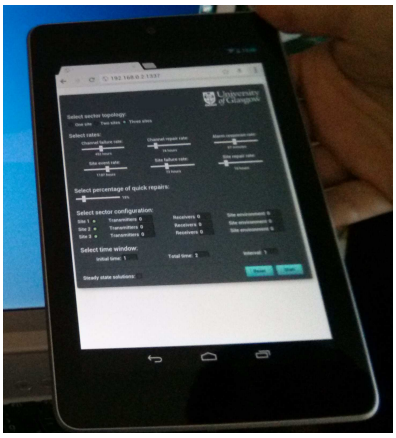

Fig. 11. Web app for setting rates and topologies running on an Android tablet.

12 Discussion

12.1 Summary of modelling and analysis framework

Our overall framework is depicted in Fig. 12 and summarised as follows. Model definition and analysis is indicated by solid lines, feedback from the analysis is indicated by the dashed lines, property outputs are screenshots from the web app. A model is validated by examining the results of steady state temporal logic properties and comparing them with the expected (or required) results from the safety and business cases, and with the observed, operational results inferred from the field data (left-hand side of Fig. 12). An example result of steady state property analysis is given in the bottom left hand side of the Figure. The right hand side of the Figure indicates how quantified predictions of behaviours and cumulative costs, the result of transient temporal logic properties, inform decisions (right hand side of Fig. 12). The framework can be used:

1) at design time, to investigate whether or not a particular architecture meets service requirements,

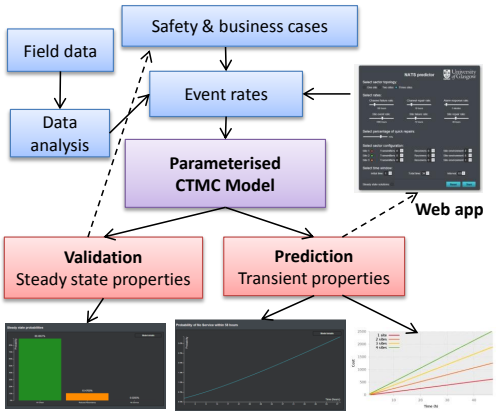

Fig. 12. Modelling and analysis framework, with feedback.

2) after the system has been deployed, and the model is parameterised by operational data, to investigate whether or not a particular architecture actually meets service requirements,
3) in real-time, on-line, to inform operational decision making,
4) as a combination of the second and third in which a "catalogue" of predictions (generated off-line) for a variety of degraded configurations is provided and then consulted as the system in real-time.

12.2 Modelling scheduled events and spatial aspects

One consequence of modelling within a Markovian framework is that we cannot easily model *scheduled*, *i.e.* non-stochastic, events. There are several possibile solutions such as a) remaining within the CTMC paradigm but modelling the new events with hyper-Erlang distributions, which means the state space explodes because of all the interleaving/expansions or b) modelling with probabilistic timed automata, which means the (exponentially distributed) failure rates are discretised by a geometric distribution, or c) modelling with hybrid CTMCs that model the scheduled events as discrete switching between CTMCs. Each of these would result in (possibly unnecessarily) more complicated semantics and analysis techniques. On the other hand, it would be relatively simple to encode any spatial aspects of the system (*e.g.* if transmitters/receivers are mobile) using (stochastic) bigraphs with sharing [6]. Given the data we have seen for scheduled maintenance in this case study is relatively sparse, we have not yet incorporated it into the model.

12.3 Reflections on case study

When we inspected the historical data, we found evidence of dependencies between channel A and B faults. This had not been raised with us previously, and so while on the one hand it was disappointing to uncover possible omissions in the model, this demonstrated the value of a formal approach and analysing historical data. The cause of dependencies is as yet unclear, but in part it may be due to the formats for recording faults and the use of free text. However, there may be further contributory factors such are transmitters and receivers are usually commissioned at the same time (and therefore failures occur a similar times), and more likely, communications network failures typically affect both channels simultaneously. Determining the causes requires further investigation, however, any modifications to the model would be relatively straightforward and would involve the introduction of synchronised Tx and Rx failures.

13 Related Work

Formal modelling and reasoning is well established for safety-critical systems, especially in the context of formal system development [7], and runtime models for managing self adaptation and the complexity of evolving software behaviour while it is executing [8]. But to our knowledge there has been little work using formal modelling and reasoning using (temporal) logics to predict future service availability and inform operational decisions in the presence of component failures. This work is therefore a novel contribution. If we choose in future to model scheduled maintenance by deterministic, timed events, as mentioned in Section 12.2, then we may also consider how these are handled in [9], where a system with *rejuvenation* – a system that is periodically stopped and then restored in a robust state after maintenance – is modelled as a Markov regenerative process and then Markov renewal theory is applied to carry out quantitative analysis. Another approach is considered in [10], which employs a partially observable Markov decision model for a maintenance problem. How these models may provide a suitable semantic underpinning for our framework, especially with regard to reasoning about logical properties, is further work. One issue for quantitative analysis is state space explosion and numerical simulation difficulties in the presence of rare events [11]. We note we have not encountered state space explosion problems nor numerical difficulties, mainly because our modelling approach involves counter abstraction and we do not analyse the system from the standard "initial state", but from degraded configurations that can occur as the system is running (regardless of the probability of reaching them). Moreover, we typically examine transient behaviours over a relatively short, *i.e.* $24 - 48$ hour period, and so state space explosion is not an issue. Concerning run-time modelling and analysis, in [12] we augmented a domestic network management system so that, as the system is running, a simulation trace (consisting of formal, bigraph models) is generated in real time, These are analysed, in real time, according to various state properties, and notification of violations are fed back to the system and to users. Finally, we note an early version of this work was presented in [13]; the main differences here are we provide more details of the model and underlying system, degraded configurations and envelopes, and temporal property analysis including recoverability, survivability, and costs.

14 Conclusions and Future Work

We have proposed an approach for assessing the impact of component failures in complex, critical systems based on stochastic event based modelling and analysis by stochastic

temporal logic model checking. The models are continuous time Markov chains, specified using high level language descriptions. By relating the status of components to service availability, and then reasoning about service availability and costs using continuous stochastic temporal logic, we quantify the risk of service failure and maintenance costs now, from a given degraded configuration, and in the future after elapsed times. Decisions about which component to repair, and when, can then be taken according to those risks and costs. The quantified risks can also inform evaluation of designs, such as levels of redundancy. A novel aspect of our approach is it is based on analysis from (classes) of degraded configurations, not the initial configuration (which is standard for many formal approaches).

This work was motivated by an industrial partner, as part of a larger programme within the company to address system management. However, the approach is applicable to any component based system with discrete, stochastic events and failures. Throughout the paper we have demonstrated our approach through application to a critical communications service in which in which component failures are sensed and monitored. The system has been deployed for several years and we had access to failure data, thus we were able to infer actual failure rates over a one year period and to demonstrate how well the model aligned with actual behaviours. We showed how transient, temporal property analysis allows us to predict the likelihood of no-service within 48 hours, before and after making repairs to specific components, to define envelopes of behaviours, for abstractions over sites, and to estimate expected costs for maintentance, over a specific time period (*e.g.* a month). The modelling and analysis framework is implemented in the PRISM language and model checker, with a bespoke multi-platform web app.

Crucial modelling decisions include the representation of time, nondeterminism, stochasicity, and their inter-relationships. We have chosen CTMCs as our underlying semantics, which gives us concise, realistic and effective models and analysis techniques. But, there are other possibilities if we wish to extend to combining deterministic and stochastic events such as hybrid or probabilistic timed automata, or models such as bigraphs that incorporate spatial aspects. Regardless of the semantics, our overall approach remains the same: to analyse a component based system using temporal logic and to quantify future behaviour from selected degraded configurations. The analysis can be used in a variety of ways: from evaluating whether an architecture meets service requirements, to informing decisions about repairs and costs in an operational system.

ACKNOWLEDGMENTS

We thank our industrial collaborators (Suki Lal, Dave Bellshaw and Terry Wright) for for help and guidance throughout the project. This work was partially funded by the EPSRC grant *Verifying Interoperability Requirements in Pervasive Systems* EP/F033206/1, the University of Glasgow EPSRC funded Impact Acceleration Account and Sevegnani's EPSRC Prize PostDoctoral Fellowship.

REFERENCES

[1] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, 2007.

[2] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen, "Model-Checking Algorithms for Continuous-Time Markov Chains," *IEEE Trans. Software Eng.*, vol. 29, no. 6, pp. 524–541, 2003.

[3] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, ser. LNCS, G. Gopalakrishnan and S. Qadeer, Eds., vol. 6806. Springer, 2011, pp. 585–591.

[4] R. Alur and T. A. Henzinger, "Reactive Modules," *Formal Methods in System Design*, vol. 15, no. 1, pp. 7–48, 1999.

[5] L. Cloth and B. R. Haverkort, "Model checking for survivability!" in *Quantitative Evaluation of Systems, 2005. Second International Conference on the*, Sept 2005, pp. 145–154.

[6] M. Sevegnani and M. Calder, "Bigraphs with sharing," *Theoretical Computer Science*, vol. 577, p. 43 74, 2015.

[7] A. Galloway, F. Iwu, J. A. McDermid, and I. Toyn, "On the formal development of safety-critical software," in *VSTTE*, 2005, pp. 362–373.

[8] U. Aßmann, N. Bencomo, B. H. C. Cheng, and R. B. France, "Models@run.time (Dagstuhl Seminar 11481)," *Dagstuhl Reports*, vol. 1, no. 11, pp. 91–123, 2012. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2012/3379

[9] S. Garg, A. Puliafito, M. Telek, and K. Trivedi, "Analysis of software rejuvenation using markov regenerative stochastic petri net," in *Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on*, 1995, pp. 180–187.

[10] R. Srinivasan and A. Parlikad, "Semi-markov decision process with partial information for maintenance decisions," *Reliability, IEEE Transactions on*, vol. 63, no. 4, pp. 891–898, Dec 2014.

[11] D. Reijsbergen, P.-T. de Boer, W. R. W. Scheinhardt, and B. R. Haverkort, "Rare event simulation for highly dependable systems with fast repairs," *Perform. Eval.*, vol. 69, no. 7-8, pp. 336–355, 2012.

[12] M. Calder, A. Koliousis, M. Sevegnani, and J. Sventek, "Real-time verification of wireless home networks using bigraphs with sharing," *Science of Computer Programming*, vol. 80, Part B, pp. 288–310, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642313001974

[13] M. Calder and M. Sevegnani, "Do I need to fix a failed component now, or can I wait until tomorrow?" *Proceedings Tenth European Dependable Computing Conference (EDCC 2014), IEEE*, 2014. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6820885

Muffy Calder Biography text here.

Michele Sevegnani Biography text here.